

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Principles of Management
Year: 3rd Year

Subject Code-HU601
Semester: Sixth

Module Number	Topics	Number of Lectures
1	Introduction:	7L
	Basic concepts of Management: Definition, essence, Functions, Roles, Level.	2
	Functions of Management Planning : Concept, Nature, Types, Analysis, Management, objectives	2
	Structure : Concept, Structure, Principles, Centralization, Decentralization, Spn of Management, Organizational Effectiveness	3
		9L
2	Management and Society : Concept, external environment, CSR, Corporate Governance, Ethical Standards.	3
	People Management : Overview, Job design, Recruitment and Selection, Stress Management	3
	Managerial competencies : Communication, Motivation, Team Effectiveness, Conflict Management, Creativity, Enterprenuership.	3
		18L
3.	Leadership concept : Nature, Styles, Decision Making, Process, Tools and Techniques.	3
	Economic, Financial and quantitative Analysis : Production Markets, National Income Accounting, Financial Function, and goals, Fianancialsttements, Ratio Analysis.	3
	Quantitative Methods : Statistical Interference, Forecasting, Regression Analysis, Ststistical Quality Control	3
4	Customer Management : Market planning and research, Market Mix, Advertising and Brand Management.	3
	Operations and Technology Management : Production and Operations Management , Logistics, & supply chain Management.	3
	TQM, Kaizen and Six Sigma, MIS.	3

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Faculty In-Charge

HOD, Humanities Dept.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name:-Control Systems-II

Subject Code:-EE601

Year: Third Year

Semester: -SIXTH

Module No.	Topics	Number of Lectures(H)
1.	State variable model of continuous dynamic systems	13H
	1.Introduction: Differential equation Vs State variable	1
	2.Converting higher order linear differential equations into State Variable (SV) form	1
	3.Obtaining SV model from Transfer Function	1
	4.Obtaining characteristic equation and transfer functions from SV model	2
	5.Obtaining SV equation directly for R-L-C and spring-mass dashpot systems	2
	6. Concept and properties associated with state equations. Linear transformations on state variables	2
	7. Canonical forms of SV equations. Companion forms	2
	Solutions of state equations. State transition matrix, properties of state transition matrix	
	8.Controllability and Observability	1
9. Linear state variable feedback controller, the pole allocation problems. Linear system design by state variable feedback	1	
2.	Analysis of discrete time (sampled data) systems using Z-transform	10H
	1.Difference equation. Inverse Z transforms	2
	2. Stability and damping in Z domain. Practical sampled data systems and computer control system. Practical and theoretical samplers	2
	3. Sampling as Impulse modulation. Sampled spectra and aliasing. Anti-aliasing filters. Zero order hold.	2
	4.Approximation of discrete (Z-domain) controllers with ZOH by Tustin transform and other methods	2
	5. State variable analysis of sampled data system. Digital compensator design using frequency response	2
	Introduction to nonlinear systems:	15H

3.	1. Block diagram and state variable representation of nonlinear systems	1
	2. Characteristics of common nonlinearities	1
	3. Phase plane analysis of linear and nonlinear second order systems	2
	4. Methods of obtaining phase plane trajectories by graphical method, isoclines method. Qualitative analysis of simple control systems by phase plane methods	3
	5. Describing function analysis. Limit cycles in nonlinear systems. Prediction of limit cycles using describing function technique	2
	6. Stability concepts for nonlinear systems. BIBO Vs state stability	1
	7. Definitions of Lyapunov functions. Lyapunov analysis of LTI systems, Asymptotic stability, Global asymptotic stability	3
	8. The first and second methods of Lyapunov to analyze nonlinear systems	2
TOTAL HOUR REQUIRED=38		

ASSIGNMENT:

MODULE1:

1. Describe the various models of state variable analysis.
2. What is state transition matrix?
3. Discuss the key properties of state transition matrix.
4. What is controllability and observability?
5. Determine the state variable model of a series R-L-C circuit.

MODULE2:

1. What is the significance of Z transform?
2. What is a sampler?
3. What is sampled spectra and aliasing?
4. Write a short note on anti-aliasing filter?
5. Write a short note on ZOH

MODULE3:

1. Describe the common non-linearities.
2. What is phase plane analysis? Why is it significant?

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name:-Control Systems-II

Subject Code:-EE601

Year: Third Year

Semester: -SIXTH

3. Write a short note on limit cycle.
4. Describe Lyapunov's stability criteria.
5. What is asymptotic stability? What is global asymptotic stability?

Faculty In-Charge

HOD, EE Dept.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject-Power System-II
Year-3rd Year

Subject Code-EE602
Semester-Sixth

Module No.	Topics	Planned Lectures(L)
1	Representation of power system components	02L
	1.Single phase representation of balanced three phase network	1L
	2.The on line diagram and impedance or reactance diagram per unit(PU) system	1L
2	Distribution substation	06L
	Type of Substation	1L
	Substation Equipment and accessories	2L
	Earthing (system & equipments)	1L
	Feeder Distribution	1L
	radial and loop system	1L
3	Load Flow Studies	08L
	Net work Model Formulation	1L
	Formation of Y_{ims}	1L
	Load flow Problem	2L
	Gauss Siegel Method	1L
	Newton rap son Method	1L
	Decoupled load flow studies	1L
	Comparison of load flow method	1L
4	Flow in Electric System	08L
	Transient on Transmission line	1L
	Short Circuit of a Synchronous Machine Under no load and loaded condition	1L
	Symmetrical component transformation	1L
	Sequence impedance and sequence network of power system	1L
	Synchronous Machine. Transmission line and transformer	2L
	Symmetrical component analysis of unsymmetrical faults	1L
	Single line to ground fault ,double line to ground fault	1L
	Line to Line Fault	
5	Power System Stability	04L
	Steady State Stability	1L
	Transient stability	1L
	Equal area criteria, swing equation	1L
	Multi machine stability concept	1L
6	Power System Protection	19L
	Protective zone, Relaying element and quantities	1L
	Protective relay	1L
	Basic requirement and type of protection	1L
	Phase and amplitude comparator	1L
	Grading(time and current)	1L
	Classification of electromagnetic relay	1L
	Directional relay, Distance relay, Differential relay	1L
	Basic aspect of static and digital relay	1L
	Relay protection scheme for transformer,feeder,generator,and motor	3L
	Circuit breaker	2L
	Circuit breaking transients, transient recovery voltage	1L
	Current chopping. Resistance switching	

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

	Circuit breaker rating arc and arc extinction	1L
	Circuit breaker type, oil circuit breaker ,vacuum circuit breaker, air blast circuit breaker, sf_6 circuit breaker and operating mechanism	2L
	Circuit breaker mechanism	1L
	Advantage and disadvantage of different type breaker	1L
	Total contact hour	47L

Text Book:-

1. Electric power system by Subir Roy
2. A text book on power system Engineering by Soni, Gupta ,Bhatnagar

Referance Book:-

1. Protection and Switch gear by B.Bhalja.R.PMaheswari,NGChothani

Hand Book:-

A Hand Book of Electrical Power Distibution by G. Ramamurthy

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

1. Draw the single line diagram for a balanced three phase power incoming in HT and LT site.

ASSIGNMENT FOR MODULE-2

1. Describe the different types of substation

2. Draw the single line diagram of single bus-bar system and single bus bar system with sectionalisation

3. Draw the single line diagram of duplicate single bus bar system substation.

ASSIGNMENT FOR MODULE-3

1. Load flow solution using by Gauss Siedel Law and Newton Raphson method.

ASSIGNMENT FOR MODULE-4

1. What is symmetrical faults on 3-phase system. Deduce formula for symmetrical fault calculation.

2. Two Generators having rated 15000KVA, 20000KVA with 30% and 50% reactance feeds with a common bus having 12000v. Then find the short circuit current that will flow into a complete 3-phase short circuit at a particular point connect with main bus..

3. Why does unsymmetrical fault occur on 3-phase system ?

ASSIGNMENT FOR MODULE-5

1. Discussion about steady state stability and transient stability

ASSIGNMENT FOR MODULE-6

1. Write down the difference types of breakers with its constructional details

2. What is directional relay and why it is frequently used

3. What is protective relay? Explain its function in an electrical system

4. Derive the equation for torque developed in an induction relay.

5. What is difference between an over current relay and current differential relay

6. Why are the differential relay more sensitivity than over current relay

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Power Electronics

Year: 3rd Year

Subject Code-EE603

Semester: Sixth

Module Number	Topics	Number of Lectures
1	Introduction:	4L
	1. Concept of power electronics, application of power electronics, uncontrolled converters, advantages and disadvantages of power electronics converters.	1
	2. Power electronics systems, power diodes.	1
	3. Power transistors, Power MOSFETS.	1
	4. IGBT and GTO.	1
2	PNPN devices:	5L
	1. Thyristors, brief description of members of Thyristor family with symbol, V-I characteristics and applications.	1
	2. Two transistor model of SCR, SCR turn on methods, switching characteristics, gate characteristics, ratings.	1
	3. SCR protection, series and parallel operation.	1
	4. Gate triggering circuits.	1
5. different commutation techniques of SCR.	1	
3.	Phase controlled converters:	6L
	1. Principle of operation of single phase and three phase half wave, half controlled, full controlled converters with R, R-L Load.	1
	2. Principle of operation of single phase and three phase half wave, half controlled, full controlled converters with RLE load.	1
	3. Effects of freewheeling diodes and source inductance on the performance of converters.	
	4. External performance parameters of converters.	1
	5. Techniques of power factor improvement.	1
6. Single phase and three phase dual converters.	1	
4	DC-DC converters:	5L
	1. Principle of operation, control strategies, step up choppers.	1
	2. Types of choppers circuits based on quadrant of operation	1
	3. Performance parameters	1
	4. Multiphase choppers	1
5. Switching mode regulators.	1	
	Inverters:	10L
	1. Definition, classification of inverters based on nature of input source.	1

5 、	2. Wave shape of output voltage	1
	3. Method of commutation & connections	1
	4. Principle of operation of three phase bridge inverter with R and R-L loads	1
	5. Principle of operation of three phase bridge inverter with R and R-L loads	1
	6. Principle of operation of three phase bridge inverter with RLC loads	1
	7. Performance parameters of inverters,	1
	8. Methods of voltage control of inverters	1
	9. Methods of harmonic reduction of inverters	1
	10. Brief idea of Resonant Pulse inverters.	1
	6	AC controllers:
1. Principle of on-off and phase control		1
2. Single phase and three phase controllers with R loads		1
3. Single phase and three phase controllers with RL loads		1
4. Principle of operation of cycloconverters		1
5. Circulating and non circulating mode of operation, single phase to single phase step up and step down cycloconverters		1
6. Three phase to single phase Cycloconverters, three phase to three phase Cycloconverter.		1
7	Applications:	4L
	1. Speed control of AC and DC motors	1
	2. HVDC transmission	1
	3. Static circuit breaker	1
	4. UPS, static VAR controller.	1
Total Number Of Hours = 40		

Faculty In-Charge

HOD, EE Dept.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

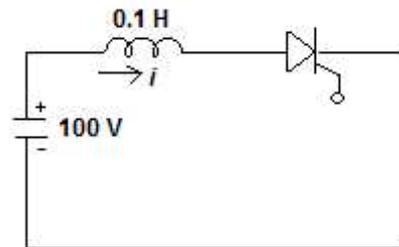
Assignment:

Module-1(Introduction):

1. Write the the definition of power electronics. Write down the advantages and disadvantage of power electronics converters.
2. Explain the basic structure of IGBT.

Module-2(PNP devices):

3. An SCR has a V_g-I_g characteristics given as $V_g=1.5+8 I_g$. In a certain application, the gate voltage consists of rectangular pulses of 12 V and of duration 50 μ s with duty cycle 0.2. Find a) The value of R_g series resistor in gate circuit to limit the peak power dissipation in the gate to 5 watts. b) Calculate average power dissipation in the gate.
4. If the V_g-I_g characteristics of an SCR is assumed to be a straight line passing through the origin with a gradient of 3×10^3 , calculate the required gate source resistance. Given $E_{gs}=10V$ and allowable $P_g=0.021$ ohm.
5. Explain the two transistor model of SCR.
6. If the latching current in the circuit shown in fig bellow is 4 mA, obtain the minimum width of the gating pulse required to properly turn on the SCR.



Module-3 (Phase controlled converters):

1. A highly inductive d.c. load requires 12 A at 150 V from a 230 V single phase a.c. supply. Give design details for requirement using a) M-2 connection b) B-2 connection for $\alpha=30$ deg.
2. A single -phase semiconductor semi converter is operated from 120 V, 50 Hz a.c. supply. The load current with an average value I_{dc} is continuous and ripple free firing angle $\alpha=\pi/6$. Determine
 - a) Displacement factor b) Harmonic factor for input current. C) Input power factor
3. Explain the operation for three phase fully controlled rectifier with RL load.
4. For three phase half-wave converter driving a large inductive load, derive an equation for.
 - a) Form factor and ripple factor. b) Thyristor and supply current ratings. Also calculate thyristor current ratings for $R=10$ ohm and $\alpha=30$ deg.

Module-4(DC-DC converters):

1. Explain the basic operation of Buck converter.

2. A step down dc chopper has a resistive load of $R=15\text{ ohm}$ and input voltage $E_{dc}=200\text{ V}$. When the chopper remains ON, its voltage drop is 2.5 V the chopper frequency is 1kHz if the duty cycle is 50% determine,
 - A) Average output voltage
 - B) RMS output voltage
 - C) Chopper frequency
 - D) Effective input resistance of chopper
3. A step-up chopper is used to deliver load voltage of 500V from a 220 V d.c source. If the blocking period of the thyristor is $80\mu\text{s}$, compute the required pulse width.

Module-5(Inverter):

1. Explain the basic operation of three phase inverter 120 deg mode of conduction.
2. A single phase half bridge inverter H-type has a resistive load of $R=3\text{ ohm}$ and the d.c. input voltage $E_{dc}=50\text{ V}$ calculate,
 - A) RMS output voltage at the fundamental frequency E_1 .
 - B) The output power, P_0
 - C) The average and peak current of each thyristor.
 - D) The peak reverse-blocking voltage of each thyristor.
3. A three phase bridge inverter is supplies from a 600 V source. For a star connected load of 15 ohm/phase , find the RMS load current, the load-power and thyristor rating for i) 120 deg conduction mode and b) 180 deg conduction mode.

Module-6 (Ac voltage controller):

1. Explain the basic operation full wave (bi directional) A.C voltage controller.
2. A single phase a.c voltage regulator with R-L load has following details,

Supply voltage= 230 V , 50 Hz , $R=4\text{ ohm}$, $wl=3\text{ ohm}$.

 - a) The control range of firing angle.
 - b) The maximum value of RMS load current.
 - c) The maximum power and power factor.
 - d) The maximum value of A_v and RMS thyristor current.
 - e) The maximum possible value of di/dt that may occur in the SCR.
 - f) The conduction angle $\alpha=0\text{ deg}$ and $\alpha=120\text{ deg}$, assuming a gate pulse of duration π radian.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Software Engineering
Year: 3rd Year

Subject Code-EE604A
Semester: Sixth

Module Number	Topics	Number of Lectures
1	Module I	8L
	1. Overview of System Analysis & Design, Business System Concept, System Development Life Cycle, Waterfall Model , Spiral Model, Feasibility Analysis, Technical Feasibility, Cost-Benefit Analysis, COCOMO model.	8
2	Module II	7L
	1. System Requirement Specification – DFD, Data Dictionary, ER diagram, Process Organization & Interactions.	3
	2. System Design – Problem Partitioning, Top-Down and Bottom-Up design; Decision tree, decision table and structured English; Functional vs.	4
3	Module III	8L
	1. Coding & Documentation - Structured Programming, OO Programming, Information Hiding, Reuse, System	2
	1. Testing – Levels of Testing, Integration Testing, Test case Specification, Reliability Assessment. , Validation & Verification Metrics, Monitoring & Control.	6
4	Module IV	12L
	1. Software Project Management – Project Scheduling, Staffing, Software Configuration Management, Quality Assurance, Project Monitoring.	7
	2. CASE TOOLS: Concepts, use and application.	5
5	Module V	6L
	1. Characteristic of software maintenance , Types of software maintenance, Software reverse engineering, Basic issues in any reuse of program, CASE TOOLS: Concepts, use and application	6
Total Number Of Hours = 41		

Faculty In-Charge

HOD, CSE Dept.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Assignment:

Module-I:

1. What are the difference between Waterfall Model and Spiral Model? What do you mean by Feasibility Analysis?
2. Explain Cost- Benefit Analysis, COCOMO model

Module-II:

1. What do you mean by System Requirement Specification?
2. Write short note on the followings:
 - DFD
 - Data Dictionary
 - ER diagram
 - Process Organization & Interactions

Module-III:

1. Explain the different levels of Testing. What is Integration Testing?
2. What do you mean by Reliability Assessment?

Module-IV:

1. How Software Project Management is done? What are the steps?
2. How the quality of any product should be assured?

Module-V:

1. What are the basic issues in any reuse of program?
2. Explain the differ types of software maintenance.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Database Management System

Subject Code-EE604B

Year:3rd Year

Semester: Sixth

Module Number	Topics	Number of Lectures
1	Introduction:	2L
	1. Concept & Overview of DBMS, Data Models, Database Languages	1
	2. Database Administrator, Database Users, Three Schema architecture of DBMS	1
2	Entity-Relationship Model:	4L
	1. Basic concepts, Design Issues	1
	2. Entity-RelationshipDiagram	2
	3. Weak Entity Sets, Extended E-R features.	1
	Relational Model:	5L
	1. Structure of relational Databases, Relational Algebra	2
	2. Relational Calculus,Extended Relational Algebra Operations,	2
3. Views, Modifications of the Database.	1	
3	SQL and Integrity Constraints:	7L
	1. Concept of DDL, DML, DCL	1
	2. Basic Structure, Set operations, Aggregate Functions	2
	3. Null Values, Domain Constraints, Referential Integrity Constraints	1
	4. assertions, views, Nested Sub queries,	1
	Relational Database Design:	8L
	1. Functional Dependency, Different anomalies in designing a Database	2
	2. Normalization using functional dependencies	2
	3. Decomposition, Boyce-Codd Normal Form, 3NF	2
	4. Normalization using multi-valued dependencies, 4NF, 5NF	2
4	Transaction:	4L
	1. Transaction concept, transaction model,serializability,transaction isolation level	2
	2. Transaction atomicity and durability, transaction isolation and atomicity	2
	Concurrency control and recovery system:	3L
	1. lock based protocol, dead lock handling, time stamp based and validation based protocol	2
	2. failure classification, storage, recovery algorithm, recovery and atomicity,backup	1

5	Internals of RDBMS:	3L
	1. Physical data structures, Query optimization: join algorithm	2
	2. Statistics and cost based optimization	1
6	File Organization & Index Structures:	5L
	1. File & Record Concept, Placing file records on Disk, Fixed and Variable sized Records	1
	2. Types of Single-Level Index (primary secondary, clustering), Multilevel Indexes	2
	3. Dynamic Multilevel Indexes using B tree and B+ tree	2
Total Number Of Hours = 39L		

Faculty In-Charge

HOD, CSE Dept.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Database Management System
Year: 3rd Year

Subject Code-EE604B
Semester: Sixth

Assignment:

Module-I:

1. Explain the three Schema architecture of DBMS.
2. What do you mean by DBA?

Module-II:

1. What do you mean by ENTITY?
2. Explain Weak Entity Sets and Extended E-R features.

Module-III:

1. Explain DDL, DML, and DCL. Differentiate 3NF and BCNF.
2. What is Domain Constraints and Referential Integrity Constraints?

Module-IV:

1. What do you mean by Transaction atomicity?
2. Explain different transaction model. What is serializability?

Module-V:

1. How do you optimize query?
2. Explain cost based optimization process.

Module-VI:

1. What are the types of Single-Level Index?
2. How dynamic Multilevel Indexes using B tree and B+ tree has done in Database?

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Object Oriented Programming Using Java
Year: 3rd Year

Subject Code-EE504C
Semester: Sixth

Module Number	Topics	Number of Lectures
1	Introduction:	6L
	1. Concepts of object oriented programming language, Major and minor elements, Object, Class.	2
	2. Relationships among objects, aggregation, links.	2
	3. Relationships among classes association, aggregation, using, instantiation, meta-class, grouping constructs.	2
2	Object oriented concepts:	3L
	1. Difference between OOP and other conventional programming – advantages and disadvantages.	1
	2. Class, object, message passing, inheritance, encapsulation, polymorphism.	2
3.	Class & Object proprieties:	11L
	1. Basic concepts of java programming – advantages of java, byte-code & JVM.	2
	2. Data types, access specifiers, operators, control statements & loops, array, creation of class, object, constructor, finalize and garbage collection.	2
	3. Use of method overloading, this keyword, use of objects as parameter & methods returning objects, call by value & call by reference	2
	4. Static variables & methods, garbage collection, nested & inner classes	2
	5. Basic String handling concepts- String (discuss charAt() , compareTo(), equals(), equalsIgnoreCase(), indexOf(), length() , substring(), toCharArray() , toLowerCase(), toString(), toUpperCase() , trim() , valueOf() methods) & StringBuffer classes (discuss append(), capacity(), charAt(), delete(), deleteCharAt(), ensureCapacity(), getChars(), indexOf(), insert(), length(), setCharAt(), setLength(), substring(), toString() methods).	2
	6. Concept of mutable and immutable string, command line arguments, basics of I/O operations – keyboard input using BufferedReader & Scanner classes.	1
4	Reusability properties	4L
	1. Super class & subclasses including multilevel hierarchy, process of	1

	constructor calling in inheritance.	
	2. Use of super and final keywords with super() method, dynamic method dispatch.	1
	3. Use of abstract classes & methods, interfaces.	1
	4. Creation of packages, importing packages, member access for packages.	1
	Exception handling & Multithreading	10L
5	1. Exception handling basics.	1
	2. Different types of exception classes, use of try & catch with throw, throws & finally, creation of user defined exception classes.	3
	3. Basics of multithreading, main thread, thread life cycle, creation of multiple threads, thread priorities.	3
	4. Thread synchronization, inter-thread communication, deadlocks for threads, suspending & resuming threads.	3
	Applet Programming (using swing)	7L
6	1. Basics of applet programming, applet life cycle, difference between application & applet programming	1
	2. Parameter passing in applets, I/O in applets, use of repaint(), getDocumentBase(), getCodeBase() methods.	1
	3. Concept of delegation event model and listener, layout manager (basic concept), creation of buttons (JButton class only) & text fields.	5

Faculty In-Charge

HOD, CSE Dept.

Assignment:

Module-1(Introduction):

1. Explain different properties of object oriented programming language.

Module-2 (Object oriented concepts):

1. Advantages and disadvantages of java over C and C++.
2. Explain with examples: encapsulation, polymorphism.

Module-3(Class & Object proprieties):

1. Explain different steps of java source code compilation and execution.
2. Why java is called platform independent programming language.
3. Explain with examples different access specifiers of java.
4. Explain finalize and garbage collection of java.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

5. Explain the significant of static keyword.
6. String vs StringBuffer class.

Module-4(Reusability properties):

1. Explain different inheritance with examples.
2. Explain uses of this, this(), super, super().
3. Abstract class vs interface.
4. Member access for packages.

Module-5(Exception handling & Multithreading):

1. Different ways of exception handling.
2. Different ways of implementing concept of multithreading.
3. Discuss problems in multithreading and their solutions.

Module-3(Applet Programming (using swing)):

1. Benefits of applet.
2. Different programs with applet.
3. Different components of swing.
4. Different event handling and layouts in swing.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: VLSI & Microelectronics

Subject Code-EE604D

Year: 3rd Year

Semester: Sixth

Module Number	Topics	Number of Lectures
1	Introduction to VLSI Design	6L
	1. VLSI Design Concepts, Moor's Law, Scale of Integration (SSI, MSI, LSI, VLSI, ULSI – basic idea only), Types of VLSI Chips (Analog & Digital VLSI)	1L
	2. Design styles- ASIC, PLA, PAL etc.	1L
	3. FPGA, Gate Array based design etc.	1L
	4. Top down, Bottom up, Semi custom, Full custom etc.	1L
	5. Design principles (Digital VLSI-Concept of Regularity etc.)	1L
	6. Design Domains (Behavioral, Structural, physical-Y chart)	1L
2	MOS Structures	12L
	1. E-MOS & D-MOS	1L
	2. Charge inversion in E-MOS	1L
	3. Threshold voltage, Flat-Band voltage, Potential balance & Charge balance	1L
	4. Inversion, MOS capacitances	1L
	5. three-terminal MOS structure with Body-effect	1L
	6. four-terminal MOS transistor: Drain current	1L
	7. I-V characteristics, Current-voltage equations (simple derivation),	1L
	8. scaling in MOSFET: General scaling, Constant voltage scaling & Constant field scaling	1L
	9. Short channel effects	2L
	10. CMOS inverter, Simple Combinational Gates-NAND gate and NOR gate using CMOS	2L
3	Micro-electronic Processes for VLSI Fabrication	10L
	1. Silicon Semiconductor Technology- An Overview, Wafer processing	1L
	2. Oxidation, Epitaxial deposition, Ion-implantation, Diffusion	1L
	3. Cleaning, Etching	1L
	4. Photo-lithography– Positive & Negative photo-resist	1L
	5. Basic CMOS Technology – Steps in fabricating CMOS	1L

	6. Basic n-well CMOS process, p-well CMOS process, Twin tub process	1L
	7. Silicon on insulator (SoI)	1L
	8. Layout Design Rules	1L
	9. Stick diagram with examples	1L
	10. Continue: Stick diagram with examples	1L
	Hardware Description Language	12L
4	1. Introduction, HDL and software languages, simulation, synthesis, VHDL, capabilities	1L
	2. Basic terminologies, entity, architecture,	1L
	3. Dataflow, structural, behavioural, mixed	1L
	4. Configuration declaration, package declaration, package body	1L
	5. Basic language elements	2L
	6. Details of Behavioural	L
	7. Details of Dataflow	1L
	8. Details of structural	L
Total Number Of Hours = 37L		

Angshuman Khan
Faculty In-Charge

Sandip Das
HOD, ECE Dept.

Assignment:

Module-1(Introduction to VLSI Design):

1. State Moore's law.
2. Describe VLSI design cycle. Why is it called cycle?
3. Write short note: regularity, modularity, locality.
4. What is hierarchical decomposition?
5. Describe 'Y' chart.
6. Discuss: CPLD, ROM, PLA, PAL, top-down & bottom-up approach, semicustom & full custom design.

Module-2 (MOS structure)

1. Describe accumulation, depletion, inversion, and pinch-off conditions of NMOS.
2. Draw C-V characteristics of MOS capacitor.
3. Discuss threshold voltage.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

4. Derive the drain current equation of nMOS.
5. Design XOR gate using CMOS.
6. Prove that size of PMOS is 2.5 times of NMOS.

Module-3 (Micro-electronic Processes for VLSI Fabrication):

1. State Lambda rule and Micron rule?
2. Draw the lay-out of NAND2 and NOR2.
3. Draw the stick diagram of CMOS.
4. What is SOI and twin tub process?
5. Describe n-well fabrication process.
6. Describe the fabrication process of CMOS.

Module-4(HDL):

1. State the difference between hardware and software language.
2. What is the difference between synthesis and simulation?
3. What is the difference between variable and signal?
4. What is '9 value logic'?
5. Design D latch and D flipflop in dataflow style.
6. Write a short note on derived datatypes.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Embedded Systems
Year: 3rd Year

Subject Code-EC605A
Semester: Sixth

Module Number	Topics	Number of Lectures
1	Introduction to Embedded System	5L
	1. Embedded system Vs General computing systems	2L
	2. History of Embedded systems	1L
	3. Purpose of Embedded systems, Microprocessor and Microcontroller	1L
	4. Hardware architecture of the real-time systems.	1L
2	Devices and Communication Buses	10L
	1. I/O types, serial and parallel communication devices I	1L
	2. I/O types, serial and parallel communication devices II	1L
	3. wireless communication devices	1L
	4. timer and counting devices, watchdog timer, real-time clock	1L
	5. Difference between serial and parallel communication devices	1L
	6. serial bus communication protocols	1L
	7. parallel communication network using ISA	1L
	8. parallel communication network using PCI, PCT-X, CAN, HDLC	1L
	9. Internet embedded system network protocols	1L
	10. USB, Bluetooth	1L
3	Program Modelling Concepts	5L
	1. Fundamental issues in Hardware software co-design	1L
	2. Unified Modelling Language (UML)	1L
	3. Hardware Software trade-offs DFG model	1L
	4. state machine programming model	1L
	5. model for multiprocessor system	1L
4	Real Time Operating Systems	8L
	1. Operating system basics, Tasks, Process and Threads	2L
	2. Multiprocessing and multitasking	2L
	3. task communication, task synchronization	2L
	4. qualities of good RTOS	2L
5	Examples of Embedded System	6L
	1. Mobile phones, RFID, WISENET	2L
	2. Robotics, Biomedical Applications, Brain machine interface etc	2L
	3. Popular microcontrollers used in embedded systems, sensors, actuators	2L
6	Programming concepts and embedded programming in C, C++, JAVA	4L
	1. Programming concepts and embedded programming in C	1L
	2. Programming concepts and embedded programming in C++	3L

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

	3. Programming concepts and embedded programming in C, C++,JAVA	
Total Number Of Hours = 38		

Faculty In-Charge

HOD, ECE Dept.

Assignment:

Module-1(Introduction to Embedded System):

1. What is an Embedded System? Give an example.
2. Embedded systems are quoted as single functioned systems. Justification is required in this case.
3. Define Real time.
4. Throw an example for real time and reactive embedded system.
5. What are the major categories of the Embedded Systems? Give an example for each division.
6. Is LCD projector an Embedded System? Please justify.
7. Microcontroller architectures are used in many embedded applications. Describe typical features of a microcontroller and give examples of the peripherals that it might provide.
8. Many embedded systems target real-time applications. What characterises a real-time system? Give three examples of embedded real-time systems.

Module-2 (Devices and Communication Buses):

1. What is digital signal controller ?
2. What are the components of embedded system?
3. What is the significance of watchdog timer in ES?
4. What is the difference between mutexes and semaphores?
5. Describe the serial communication ports.
6. How RS-232 communication works?
7. Describe SI,SCI,SPI communication protocols.

Module-3(Program Modelling Concepts):

1. Why is it required to have an abstraction layer in UML?
2. What is the difference between Responsibility-driven and data-driven approaches?
3. What is the function of attribute facets in UML?
4. What are the different types of implementation terms in UML?
5. How does the rule chaining concept work in UML?
6. List the UML Diagram Types.
7. What are the different types of implementation terms in UML?

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Module-4(Real Time Operating Systems):

1. What is a non re-entrant code?
2. How is RTOS different from other OS?
3. Is unix a multitasking or multiprocessing operating system? whats the difference between the two?
4. What is a core dump?
5. What is stack overflow and heap overflow?
6. Windows also has multiple processes has process priorities switches between multiple process, how RTOS is different from that?
7. How will u create a process in UNIX or our OS OSE?
8. What is a flat memory model and a shared memory model?
9. What is paging, segmentation y do we need it?
10. Write a code to check whether a stack grows upwards or downwards?
11. Why do we require semaphore?
12. Write a small piece of code protecting a shared memory variable with a semaphore?
13. What are the different types of semaphores and where they are used?
14. What are the different inter process communications?
15. What is present in .bss
16. What are the different segments of a program (code, data,stack,heapetc)
17. What is OS scheduling mechanism in our OSE?
18. How and where the context related information is stored PCB i guess
19. What is a make command and what are al its uses?
20. What if no process responding how it is handled (watch dog timeout mechanism)
21. What is an ELF
22. What is priority inversion?

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Communication Engineering

Subject code: EE605B

Year: 3rd Year

Semester: Sixth

Module Number	Topics	Number of Lectures
1.	Elements of Communication system, Analog Modulation & Demodulation, Noise, SNR Analog to Digital Conversion:	8L
	1. Introduction to Base Band transmission & Modulation (basic concept)	1
	2. Elements of Communication systems (mention of transmitter, receiver and channel); origin of noise and its effect, Importance of SNR in system design.	1
	3. Basic principles of Linear Modulation (Amplitude Modulation)	1
	4. Basic principles of Non-linear modulation (Angle Modulation - FM, PM)	1
	5. Sampling theorem, Sampling rate, Impulse sampling, Reconstruction from samples, Aliasing	1
	6. Analog Pulse Modulation - PAM (Natural & flat topped sampling), PWM, PPM	1
	7. Basic concept of Pulse Code Modulation, Block diagram of PCM	1
	8. Multiplexing - TDM, FDM	1
2.	Digital Transmission:	8L
	1. Generation of AM: Concept of Gated, Square law modulators, Switching modulator Concept of Quantization & Quantization error, Uniform Quantize	1
	2. Non-uniform Quantizer, A-law & μ law companding (mention only)	1
	3. Encoding, Coding efficiency, Line coding & properties, NRZ & RZ, AMI, Manchester coding PCM, DPCM	2
	4. Baseband Pulse Transmission, Matched filter (mention of its importance and basic concept only), Error rate due to noise	2
	5. ISI, Raised cosine function, Nyquist criterion for distortion-less base-band binary transmission, Eye pattern, Signal power in binary digital signals	2
	Digital Carrier Modulation & Demodulation Techniques:	8L

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

3.	1. Bit rate, Baud rate; Information capacity, Shanon's limit	2
	2. M-ary encoding, Introduction to the different digital modulation techniques - ASK, FSK, PSK, BPSK, QPSK, mention of 8 BPSK, 16 BPSK	2
	3. Introduction to QAM, mention of 8QAM, 16 QAM without elaboration	1
	4. Delta modulation, Adaptive delta modulation (basic concept and importance only, no details.	1
	5. introduction to the concept of DPCM, Delta Modulation, Adaptive Delta modulation and their relevance	1
	6. Spread Spectrum Modulation - concept only	1
	Information Theory & Coding:	8L
4.	1. Introduction, News value & Information content (IL), Entropy	2
	2. Mutual information; Information rate;Shanon-Fano algorithm for encoding	2
	3. Shannon's Theorem - Source Coding Theorem ; Channel Coding Theorem, Information Capacity Theorem (basic understanding only)	2
	4. Error Control & Coding - basic principle only	2
Total Number Of Hours = 32		

Signature of Faculty

Signature of HOD

Assignment:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Module-1:

- 1. Explain communication system with proper block diagram.**
- 2. Explain the need for modulation in a communication system.**
- 3. Explain how AM is generated?**
- 4. Explain the concept of multiplexing.**
- 5. Explain the working principle of TDM/FDM.**

Module-2:

- 1. What is companding? Why it is needed?**
- 2. Explain uniform and non-uniform quantization?**
- 3. What is Nyquist criterion for zero intersymbol interference?**
- 4. What is the function of raised cosine function?**
- 5. What are limitations of delta modulation?**
- 6. What is quantization noise?**
- 7. Given an input data: 010110, Represent it using the line codes.**

Module-3:

- 1. What is the difference between MSK and QPSK?**
- 2. With Proper equations and signal constellation explain the concept of Quadri phase shift Keying.**
- 3. Explain the concept of OFDM.**
- 4. Draw an eye diagram and mention the significance of its different parts.**

Module-4:

- 1. State and Prove source coding theorem**
- 2. Define information.**
- 3. Suppose the six symbols of the alphabet S and their probabilities are given by $S=\{S_0, S_1, S_2, S_3, S_4, S_5\}$ with probabilities $P_0=1/16, P_1=1/16, P_2=1/8, P_3=1/4, P_4=1/4, P_5=1/4$. Calculate entropy, $H(S)$ and use Huffman coding to code each symbols, average length code and efficiency.**
- 4. Mention the properties of mutual information.**

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Digital Signal Processing

Subject Code-EE605C

Year: 3rd Year

Semester: Sixth

Module Number	Topics	Number of Lectures
1	Discrete time signal and system in time domain: -	8L
	1. Introduction of DSP and its applications.	1
	2. Type of digital signals, sampling theorem, Shannon's theorem and aliasing effect	1
	3. Various operation of digital signals.	1
	4. Discrete time system	2
	5. Concept of various type of convolution and its properties	2
	6. Concept of correlation and its application	1
2	Harmonics Analysis of Signals	8L
	1. Brief introduction of Fourier transform	1
	2. Discrete Time Fourier Transform and its properties	1
	3. Some numerical on DTFT	1
	4. Discrete Fourier Transform and its properties	1
	5. Some numerical on DFT	1
	6. Concept of Overlap save and overlap add method	1
	7. Concept of FFT and its application	1
8. 2 Radix FFT algorithms	1	
3.	Z Transform:	5L
	1. Brief introduction of Z transform and ROC with properties	1
	2. Z transform and its properties	1
	3. Some numerical on Z transform	1
	4. Analysis and designing the system using z transform	2
4	LTI DTS in frequency domain: -	3L
	1. Introduction of Transfer function and frequency response	1
	2. Introduction of linear phase filter, Group d and phase delay.	1
	3. Concept of simple analog and digital filter	1
5	Digital FIR Filter design:-	6L
	1. Introduction of FIR filter and its structure	1
	2. Discrete parallel and cascade FIR filter	1
	3. Introduction of Linear phase filter and its characteristics	1

	4. Basic concept of window signal and window functions	1
	5. FIR filter design using various window function	1
	6. design high pass, low pass and band pass filter designing	1
	Digital IIR Filter design: -	4L
6	1. Introduction of IIR filter and its structure	1
	2. Discrete parallel and cascade IIR filter	1
	3. Basic concept of bilinear transform	1
	4. IIR filter design using bilinear transform	1
	DSP processor (TMS 320C 54X):	4L
7	1. Introduction of DSP processor and its applications	1
	2. Characteristic and features of TMS 320 DSP family	1
	3. Functional block diagram and architecture of TMS 320C 54X series DSP processor	1
	4. Addressing mode and instruction set of TMS 320C 54X	1

Faculty In-Charge

HOD, ECE

Assignment:

Module-1 (Discrete time signal and system in time domain):

- (a) If $x(n) = 0$ for $n < 0$, derive an expression for $x(n)$ in terms of its even part, $x_e(n)$, and, using this expression, find $x(n)$ when $x_e(n) = (0.9)^{|n|}u(n)$.

(b) If $x_1(n)$ is even and $x_2(n)$ is odd, what is $y(n) = x_1(n) \cdot x_2(n)$?
- Determine whether or not the signals below are periodic and, for each signal that is periodic, determine the fundamental period.

(a) $x[n] = \text{Re}\{e^{j\pi n/17}\} + \text{Im}\{e^{j\pi n/17}\}$

(b) $x[n] = e^{j\pi n/17} \cos(n/17)$
- Given the sequence $x(n) = (6 - n)[u(n) - u(n - 6)]$, make a sketch of

(a) $y_1(n) = x(4 - n)$

(b) $y_1(n) = x(n^2 - 2n + 1)$

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

4. Consider the sequence

$$x(n) = \left(\frac{3}{2}\right)^n u(-n)$$

- (a) Find out the Energy of $x(n)$
- (b) Find out the Power of $x^2(n)$

5. Determine which systems are Linear, shift invariant and causal

- (a) $y(n) = x(n)\sin(n/2)$
- (b) $y(n) = \text{Re}\{x(n)\}$

6. (a) Find the convolution of the two finite-length sequences:

$$x(n) = 0.5n[u(n) - u(n-6)] \text{ and } h(n) = 2\sin(n/2)[u(n+3) - u(n-4)]$$

(b) If the response of a linear shift-invariant system to a unit step (i.e., the step response) is

$$s(n) = n(1/2)^n u(n). \text{ Find the unit sample response, } h(n).$$

7. Find the periodic convolution of the two finite-length sequences:

$$x[n] = [1, 4, -3, 6, 4] \text{ and } h[n] = [2, -4, 3] \dots \dots \dots [\text{by using circular matrix method}]$$

8. Two causal finite length sequences are given following.

$$y[n] = [6, 11, -13, 16, 1, 9, 2] \text{ and } h[n] = [2, 5, -1, 4]$$

If $y[n] = h[n]*x[n]$, then find out the sequence $x[n]$.

Module-2 (Harmonics Analysis of Signals):

1. The DFT of a discrete sequence $x[n]$ is $X[k] = [1, 2, 3, 4]$

- (a) Find out the DFT of the sequence $g[n] = x[n-2]$.
- (b) Find out the IDFT of sequence of $X[k] = [3, 4, 1, 2]$.
- (c) Find out the value of 'a' and 'b' from the given DFT of $x[n]$ is

$$X[k] = [0, a, 2+j, -1, b, j].$$

2. If $x[n] = [1, -2, 3, -1, 5, -9]$, find out the value of

- (a) $X[0]$,
- (b) $X[3]$,
- (c) $\sum_{k=0}^5 X[k]$,
- (d) $\sum_{k=0}^5 (-1)^k X[k]$
- (e) $X[9]$

3. (a) What is the difference between DTFT and DFT.

- (b) Explain how the DFT is coming from DTFT.
- (c) What is Twiddle factor?

- Find out the DFT of given sequence by using matrix method
 $x[n] = \delta[n] + 2\delta[n-1] + \delta[n-2] + 2\delta[n-3]$
- Determine the Fourier transform of the signal: $x(n) = 2^n u(n) - 2^{-n} u(-n)$.
- Determine sectional convolution whose impulse response is $h[n] = \{1, 1, 1\}$ and input is $x[n] = \{3, -1, 0, 1, 3, 2, 0, 1, 2, 1\}$ using overlap add method.

Module-3(Z Transform):

- If the z transform of $x[n]$ is $32/(Z^2-16)$, then calculate the z transform of $(1/2^n) x[n]$
- The transfer function of a causal system is given as
 $H(Z) = (Z+0.5)(Z+2)(Z-1)^2/Z^4$,
then calculate the inverse z transform of $H(Z)$.
- The transfer function of a causal system is given as $H[Z] = \frac{5Z^2}{Z^2-Z-6}$,
then calculate the value of $h[n]$.
- Describe correlation and multiplication property of z-transform.
- Find the inverse Z transform of $X(Z) = \frac{Z(Z^2-4Z+5)}{(Z-3)(Z-1)(Z-2)}$
for ROC- i) $2 < |Z| < 3$
ii) $|Z| > 3$
iii) $|Z| < 1$
- Find out the Z transform of following signal
 $x[n] = (-1)^n \cos(\sqrt{3}n) u[n]$

7. Write down the difference equation for Transfer function

$$H(Z) = \frac{1 + \left(\frac{1}{4}\right)Z^{-1} + \left(\frac{1}{2}\right)Z^{-2} + \left(\frac{1}{3}\right)Z^{-3}}{1 + Z^{-1} + \left(\frac{1}{4}\right)Z^{-2} - \left(\frac{1}{3}\right)Z^{-3} + \left(\frac{1}{2}\right)Z^{-4}}$$

Module-4 (LTI DTS in frequency domain):

- (a) Find out the impulse response of given system equation
 $y[n] = x[n] + 2x[n-1] - 4x[n-2]$
(b) Calculate and draw the frequency and phase response of above system.
(c) How do you explain the nature and properties of the above system?
- Prove that the IIR filter is always a nonlinear phase filter.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

3. Find out the solution of difference equation of $y[n] - (1/6)y[n-1] - (1/6)y[n-2] = 4x[n]$ if $x[n] = u[n]$, $n \geq 0$, and $y[-1] = 0$, $y[-2] = 12$.
4. The signal $x[n] = \{1, 0.5\}$ is applied to a digital filter and the resulting output is $y[n] = de[n] - 2de[n-1] - de[n-2]$.
 - (a) What is the frequency response $H(F)$ of this filter?
 - (b) Find the impulse response $h[n]$ of this filter.
 - (c) Find the filter difference equation

Module-5 (Digital FIR Filter design):

1. What type of window(s) may be used to design a low-pass filter with a passband cutoff frequency $\omega_p = 0.35\omega_c$, and transition width $\omega_s - \omega_p = 0.025\omega_c$, and a maximum stopband deviation of $\delta_s = 0.003$?
2. Use the window design method to design a minimum-order low-pass filter with a passband cutoff frequency $\omega_p = 0.45\omega_c$, a stopband cutoff frequency $\omega_s = 0.5\omega_c$, and a maximum stopband deviation $\delta_s = 0.005$.
3. Suppose that we would like to design a low-pass filter of order $N = 128$ with a passband cutoff frequency $\omega_p = 0.48\omega_c$ and a stopband cutoff frequency of $\omega_s = 0.52\omega_c$
 - (a) Find the approximate passband and stopband ripple if we were to use a Kaiser window design.
 - (b) If an equiripple filter were designed so that it had a passband ripple equal to that of the Kaiser window design found in part (a), how small would the stopband ripple be?
4. We would like to design an equiripple low-pass filter of order $N = 30$. For a type I filter of order N , what is the minimum number of alternations that this filter may have. and what is the maximum number?
5. For a low-pass filter with $\delta_s = \delta_p$, what is the difference in the stopband attenuation in decibels between a Kaiser window design and an equiripple filter if both filters have the same transition width?

Module-5 (Digital IIR Filter design):

1. Find the minimum order and the 3-dB cutoff frequency of a continuous-time Butterworth filter that will satisfy the following Frequency response constraints:
$$\begin{aligned} |H(j\omega)| &= 0.95 & \omega &= 16,000 \\ |H_a(j\omega)| &= 0.1 & \omega &> 24,000 \end{aligned}$$

2. Use the bilinear transformation to design a first-order low-pass Butterworth filter that has a 3-dB cutoff frequency $\omega_c = 0.5\omega_s$.
 3. Use the bilinear transformation to design a second-order bandpass Butterworth filter that has 3-dB cutoff frequencies $\omega_l = 0.4\omega_s$ and $\omega_u = 0.6\omega_s$.
 4. If the specifications for an analog low-pass filter are to have a -1-dB cutoff frequency of 1 kHz and a maximum stopband ripple $\delta_s = 0.01$ for $f/f_s > 5$ kHz, determine the required filter order for the following:
 - (a) Butterworth filter
 - (b) Type I Chebyshev filter
 - (c) Type II Chebyshev filter
 - (d) Elliptic filter
 5. If an analog filter has an equiripple passband, will the digital filter designed using the impulse invariance method have an equiripple passband? Will it have an equiripple passband if the bilinear transformation is used?
 6. Can an analog allpass filter be mapped to a digital allpass filter using the bilinear transformation?
-
7. An IIR low-pass digital filter is to be designed to meet the following specifications:

Passband cutoff frequency of $0.221\omega_s$ with a passband ripple less than 0.01

Stopband cutoff frequency of $0.241\omega_s$ with a stopband attenuation greater than 40 dB

 - (a) Determine the filter order required to meet these specifications if a digital Butterworth filter is designed using the bilinear transformation.
 - (b) Repeat for a digital Chebyshev filter.
 - (c) Compare the number of multiplications required to compute each output value using these filters, and compare them to an equiripple linear phase filter.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Control Systems-II Lab

Course Code: EE691

L-T-P scheme: 0-0-3

Course Credit: 2

Objectives:

1. To provide the students with a hands-on experience on the theoretical concepts through simple experiments.
2. To develop the ability to design and validate their knowledge through open ended experiments.

Learning Outcomes:

On successful completion of this lab course, the students would be able to

1. Understanding on the topics of state space
2. Understanding of nonlinear systems.

Course Contents:

List of Experiments:

1. STUDY OF A PRACTICAL POSITION CONTROL SYSTEM USING SIMULINK AND HARDWARE.
2. TUNING OF P, PI, AND PID CONTROLLER FOR FIRST ORDER PLANT WITH DEAD TIME USING Z-N METHOD .(MATLAB AND HARWARE)
3. DESIGN OF LEAD AND LAG COMPENSATION USING CACSAD TOOLS (PSPICE, MATLAB, SciLab may be used).
4. STATE VARIABLE ANALYSIS USING CACSAD COMMAND TOOL. (PSPICE, MATLAB, SciLab may be used).
5. STATE VARIABLE ANALYSIS USING CACSAD BLOCK DIAGRAM TOOL. (PSPICE, MATLAB, SciLab may be used).
6. PERFORMANCE ANALYSIS OF A DISCRETE TIME SYSTEM USING CACSAD TOOL. (PSPICE, MATLAB, SciLab may be used).
7. STUDYING THE EFFECTS OF NONLINEARITY IN A FEEDBACK CONTROLLED SYSTEM USING TIME RESPONSE. (PSPICE, MATLAB, SciLab may be used).
8. STUDYING THE EFFECTS OF NONLINEARITY IN A FEEDBACK CONTROLLED SYSTEM USING PHASE PLANE PLOTS. (PSPICE, MATLAB, SciLab may be used).

Text Book:

- 1) Katsuhiko Ogata, (2002), Modern Control Engineering, Prentice Hall of India Private Ltd., New Delhi.
- 2) Nagrath I.J. and Gopal M., (2006), Control Systems Engineering, New Age International Publisher, New Delhi.

Recommended Systems/Software Requirements:

SCILAB, MATLAB

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

EXPERIMENT NO. 1

TITLE: STUDY OF A PRACTICAL POSITION CONTROL SYSTEM. Obtaining closed step responses for gain setting corresponding to over-damped and under-damped responses. Determination of rise time and peak time using individualized components in SIMULINK. Determination of un-damped natural frequency and damping ratio from the experimental data.

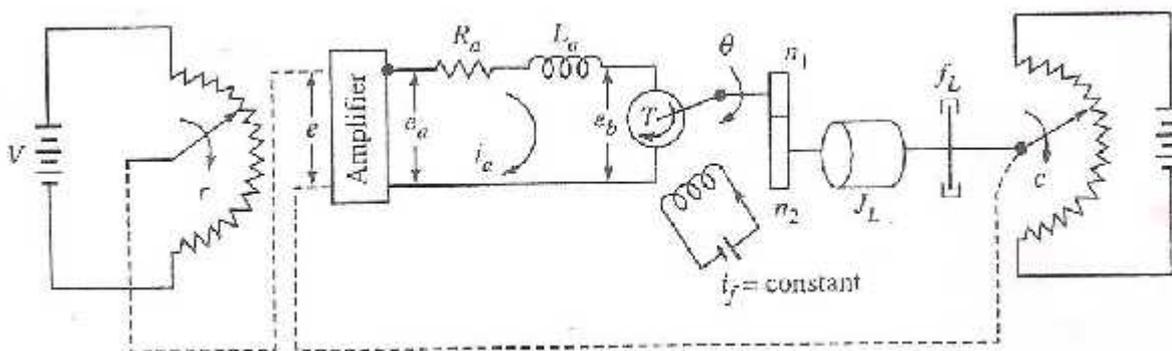
OBJECTIVE:

- a. Modeling a practical position control system using individualized components in SIMULINK.
- b. Obtaining step responses for different gain settings.
- c. Determination of un-damped natural frequency and damping ratio.

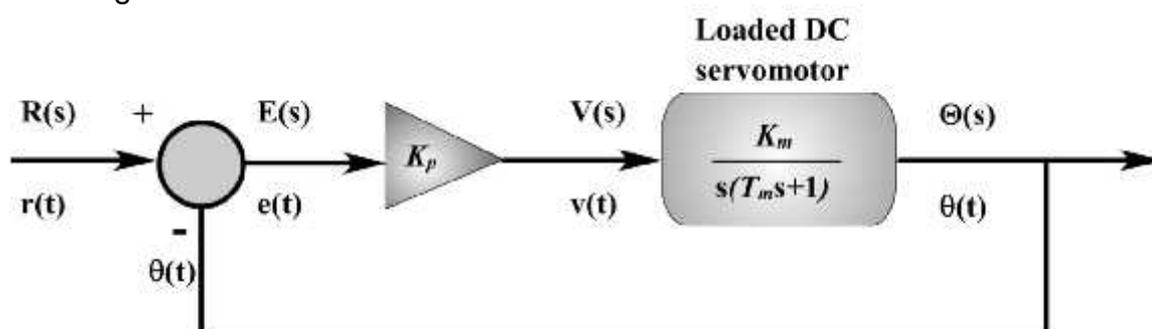
Objectives:

- a. Modeling a practical position control system using individualized components in SIMULINK.
- b. Obtaining step responses for different gain settings.
- c. Determination of un-damped natural frequency and damping ratio.

The practical setup of a position control system is shown in the figure below. The input and output of the system are the input shaft position and the output shaft position respectively.



The detailed block diagram considering individual components of the system is shown in the figure below.



UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Here,

K_p = Amplifier Gain

And considering $K_m = 10$ and $T_m = 0.23$

The closed loop system transfer function is given by:

$$\frac{\Theta(s)}{R(s)} = \frac{K_p K_m}{s^2 T_m + s + K_p K_m}$$

Depending on system parameters and the controller gain that we use, the step response could be oscillatory (under damped), critically damped or over damped.

Critical damping condition

For critical damping, the denominator of the closed loop transfer function should have equal roots. The roots will be equal if and only if

$$1 - 4K_p K_m T_m = 0$$

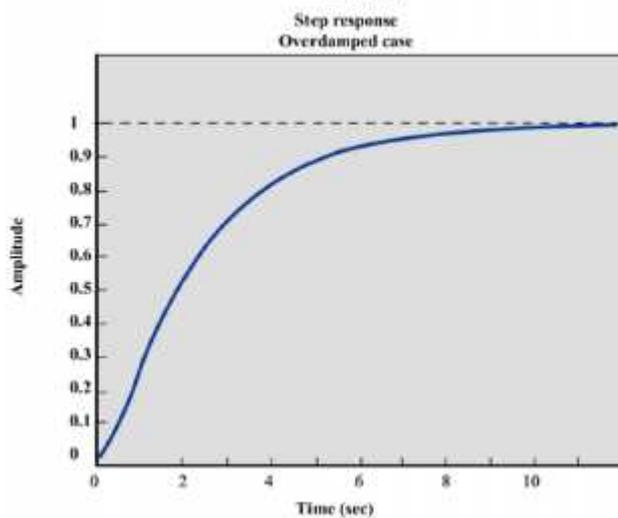
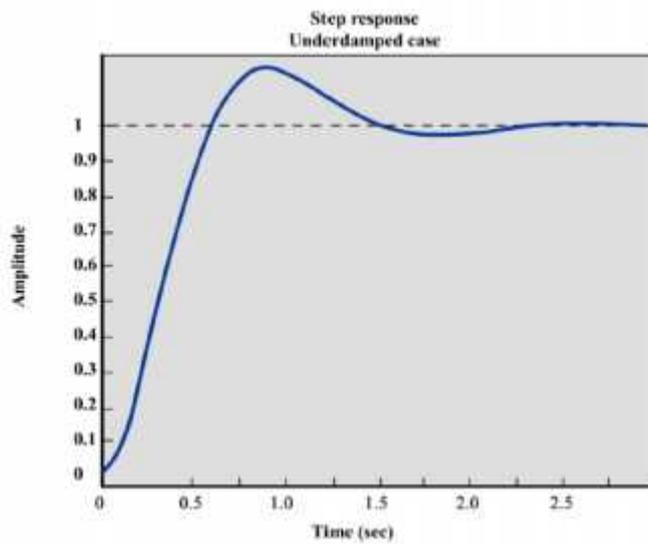
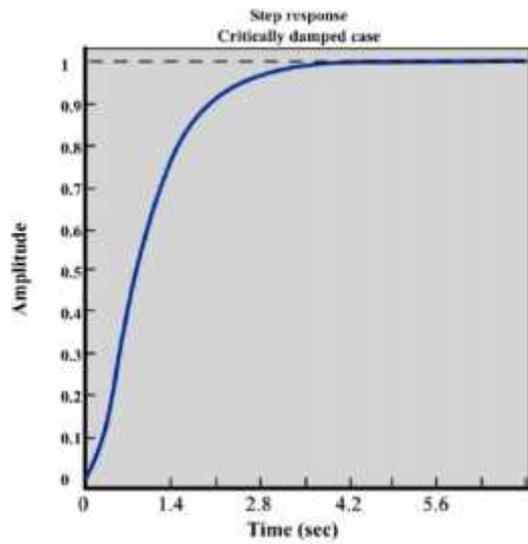
or

$$K_p^{cr} = K_p = \frac{1}{4K_m T_m}$$

Thus the value of amplifier gain at critical damping condition is

$$K_p = \frac{1}{4K_m T_m} = \frac{1}{4(10)(0.23)} = 0.1$$

The following graphs show the step response for different gain values

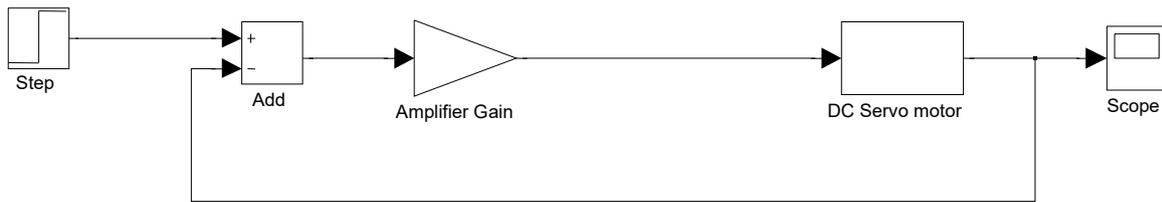


Experiment

1. Using SIMULINK develop the block diagram of the position control system.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual



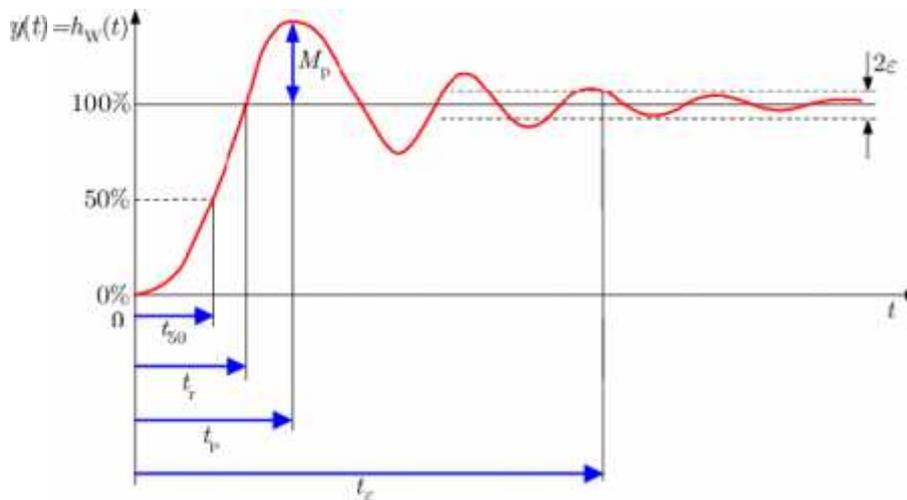
Take $T_m=0.23$, K_m =Last two digits of your Roll no.

2. Determine the value of Amplifier gain to have critical damping condition.
3. Simulate the system for different value of Amplifier gain.
4. Tabulate the simulation result for different values of the amplifier gain from the output response graph.

Amplifier Gain (K_p)	Rise Time (t_r)	Delay Time (t_d)	Peak Time (t_p)	Peak Value (M_p)	Settling Time (t_{ss})

Report

Calculate the **un-damped natural frequency** ($\zeta \omega_n$) and **damping ratio** (ζ) using the following method.



The **Peak Time (t_p)** is given by

$$t_p = \frac{f}{\zeta \omega_n \sqrt{1 - \zeta^2}}$$

The **Rise Time (t_r)** is given by

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

$$t_r = \frac{f - \cos^{-1} \zeta}{\zeta_n \sqrt{(1 - \zeta^2)}}$$

The **Delay Time (t_d)** is given by

$$t_d = \frac{1 + 0.7\zeta}{\zeta_n}$$

Using the above relationship and the result obtained from the experimental data determine the **un-damped natural frequency (ζ_n)** and **damping ratio (ζ)** and tabulate the result.

Amplifier Gain (K_P)	Un-damped natural frequency (ζ_n)	Damping ratio (ζ)

EXPERIMENT NO. 2

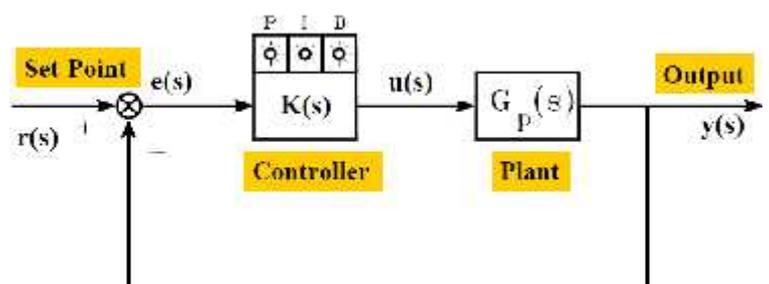
TUNING OF P, PI, AND PID CONTROLLER FOR FIRST ORDER PLANT WITH DEAD TIME USING Z-N METHOD. Process parameters (time constant and delay/lag) will be provided, the students would compute controller gains by using Z-N method. Steady state and transient performance of the closed loop plant with and without steady disturbances will have to be noted. Theoretical phase and gain margins will have to be manually computed for each gain settings.

Objectives:

- d. Introduction to PID controller using Z-N Method
- e. Computation of controller gain using Z-N method.
- f. Closed loop transient response for P, PI, PID controller action.

PID - Controller

The PID – Controller is the most widely used control strategy in industry. It is used for various control problems such as automated systems or plants. A PID-Controller consists of three different elements, which is why it is sometimes called a three term controller. PID stands for:



P Proportional control

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

- I Integral control
- D Derivative control.

PID – control can be implemented to meet various design specifications for the system. These can include the rise and settling time as well as the overshoot and accuracy of the system step response.

To understand the operation of a PID feedback controller, the three terms should be considered separately.

Proportional Control

Proportional control is a pure gain adjustment acting on the error signal to provide the driving input to the process. The P term in the PID – controller is used to adjust the speed of the system.

Integral Control

Integral control is implemented through the introduction of an integrator. Integral control is used to provide the required accuracy for the control system.

Derivative Control

Derivative action is normally introduced to increase the damping in the system. The derivative term also amplifies the existing noise which can cause problems including instability.

PID Transfer Function

If we now look at the general transfer function of a PID-controller, the three terms can be recognised as follows:

$$G_{PID}(s) = \frac{U(s)}{E(s)} = K_p \left(1 + \frac{1}{T_i s} + T_d s \right)$$

Where:

K_p is the proportional gain

T_i is the integral time constant

T_d is the derivative time constant

If we now rearrange that a little we come up with a more conventional transfer function form:

$$U(s) = K_p E(s) + K_i \frac{1}{s} E(s) + K_d s E(s)$$

The effects of increasing each of the controller parameters K_p , K_i and K_d can be summarized as

Response	Rise Time	Overshoot	Settling Time	S-S Error
K_p	Decrease	Increase	NT	Decrease
K_i	Decrease	Increase	Increase	Eliminate
K_d	NT	Decrease	Decrease	NT

NT: No definite trend. Minor change.

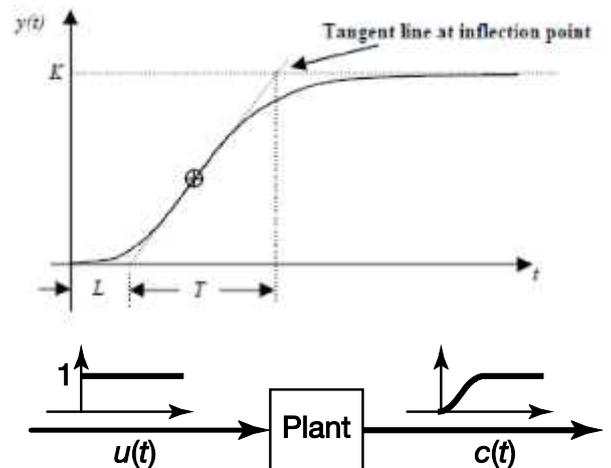
Ziegler-Nichols Tuning

In 1942 Ziegler and Nichols, both employees of Taylor Instruments, described simple mathematical procedures, the first and second methods respectively, for tuning PID controllers. These procedures are now accepted as standard in control systems practice. Both techniques make a priori assumptions on the system model, but do not require that these models be specifically known. Ziegler-Nichols formulae for specifying the controllers are based on plant step responses.

The First Method

The first method is applied to plants with step responses of the form displayed in Figure . This type of response is typical of a first order system with transportation delay, such as that induced by fluid flow from a tank along a pipe line. It is also typical of a plant made up of a series of first order systems. The response is characterised by two parameters, L the delay time and T the time constant. These are found by drawing a tangent to the step response at its point of inflection and noting its intersections with the time axis and the steady state value. The plant model is therefore

$$G(s) = \frac{Ke^{-sL}}{Ts+1}$$



Ziegler and Nichols derived the following control parameters based on this model:

PID Type	K_p	$T_i = K_p/K_i$	$T_d = K_d/K_p$
P	$\frac{T}{L}$	∞	0
PI	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{T}{L}$	$2L$	$0.5L$

The Controller Transfer Function is given by

$$\begin{aligned} G_c(s) &= K_p \left(1 + \frac{1}{T_i s} + T_d s \right) \\ &= 1.2 \frac{T}{L} \left(1 + \frac{1}{2Ls} + 0.5Ls \right) \\ &= 0.6T \frac{\left(s + \frac{1}{L} \right)^2}{s} \end{aligned}$$

Second Method

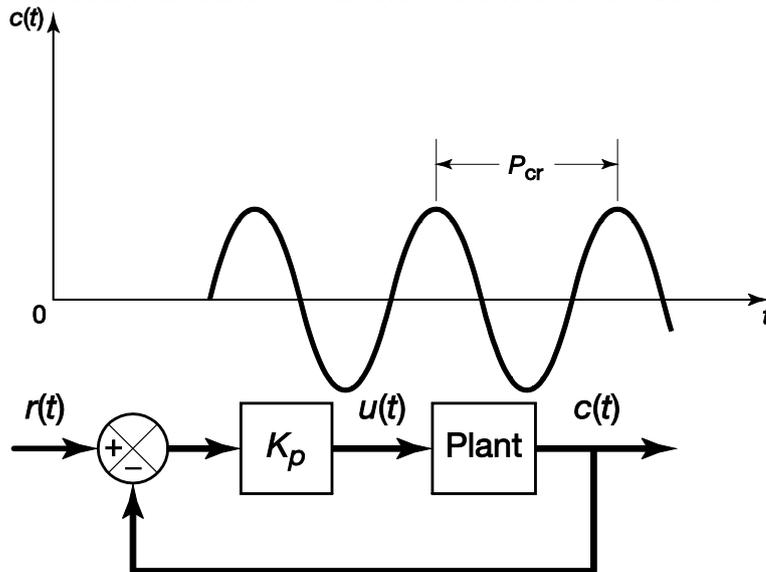
UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

The second method targets plants that can be rendered unstable under proportional control. The technique is designed to result in a closed loop system with 25% overshoot. This is rarely achieved as Ziegler and Nichols determined the adjustments based on a specific plant model. The steps for tuning a PID controller via the 2nd method is as follows:

Using only proportional feedback control:

1. Reduce the integrator and derivative gains to 0.
2. Increase K_p from 0 to some critical value $K_p=K_{cr}$ at which sustained oscillations occur. If it does not occur then another method has to be applied.



3. Note the value K_{cr} and the corresponding period of sustained oscillation, P_{cr}

The controller gains are now specified as follows:

PID Type	K_p	T_i	T_d
P	$0.5 K_{cr}$	∞	0
PI	$0.45 K_{cr}$	$\frac{P_{cr}}{1.2}$	0
PID	$0.6 K_{cr}$	$\frac{P_{cr}}{2}$	$\frac{P_{cr}}{8}$

The Controller Transfer Function is given by

$$\begin{aligned}
 G_c(s) &= K_p \left(1 - \frac{1}{T_i s} + T_d s \right) \\
 &= 0.6 K_{cr} \left(1 + \frac{1}{0.5 P_{cr} s} + 0.125 P_{cr} s \right) \\
 &= 0.075 K_{cr} P_{cr} \frac{\left(s - \frac{4}{P_{cr}} \right)^2}{s}
 \end{aligned}$$

Experiment - A

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

1. Consider the Plant Transfer Function

$$G(s) = \frac{K}{s^2 + 11s + 6}$$

Take K=Last two digits of your roll no.

2. Find step response of the plant and determine delayed time (**L**) and Time Constant (**T**) of the equivalent Transfer function

$$G(s) = \frac{Ke^{-sL}}{Ts + 1}$$

3. Calculate the P, PI, & PID controller gains using Z-N First Method.
4. Tabulate the gains.
5. Compare the closed loop step response for the system with PID controller & without PID controller using SIMULINK.

Experiment - B

1. Consider the system Transfer Function

$$G(s) = \frac{K}{s(s+1)(s+3)}, \text{ Take K=Last two digits of your roll no.}$$

2. Use Routh-Hurwitz Criterion to determine the Proportional gain value to have sustained oscillation.
3. Determine the step response for the calculated proportional gain value using SIMULINK. From the response curve determine the P_{CR}
4. Calculate the P, PI, & PID controller gains using Z-N Second Method.
5. Tabulate the gains.
6. Compare the closed loop step response for the system with PID controller & without PID controller using SIMULINK.

EXPERIMENT NO. 3

DESIGN OF LEAD AND LAG COMPENSATION USING CACSAD TOOLS (Plant transfer function will be provided. Step response is to be obtained. (PSPICE, MATLAB, SciLab may be used).

Objectives:

- g. Introduction to SISOTOOLS.
- h. Design of a Compensator for a given system and its step response.

A lead-lag compensator is a component in a control system that improves an undesirable frequency response in a feedback control system.

We can realize lead and lag compensation by introducing a cascade compensator of the form

$$C(s) = \frac{K_c \cdot (s + z)}{(s + p)}$$

For Lag Compensator

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

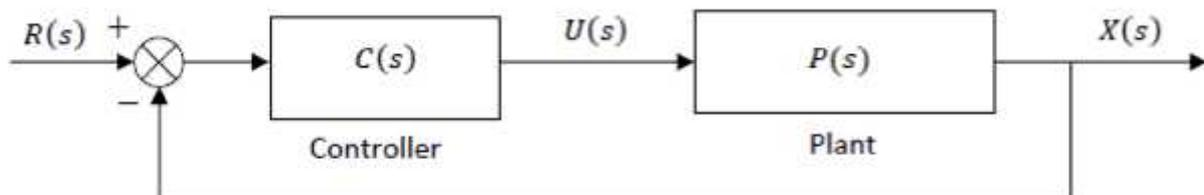
Lab Manual

$$C(s) = \frac{K_c \cdot (s + z)}{(s + p)} \text{ where } |z| > |p|$$

For Lead Compensator

$$C(s) = \frac{k \cdot (s + z)}{(s + p)} \text{ where } |p| > |z|$$

Introduction to Matlab 'sisotool'



Generic Unity Feedback Control System

A. Getting Started

- Enter the transfer function P(s) for the plant, in your workspace (i.e., from the Matlabcommand prompt).
- Type 'sisotool' at the command prompt.
- Click **Close** when the help window comes up.
- Click on **View Open Loop Bode** to turn off the bode plot. (Whatever is checked in this list will be displayed in the window.)

B. Loading the Transfer Function

- 1. Click on **File Import**.
- A window on the left will show you the transfer functions in your workspace, while the window on the right will let you choose the control system configuration.
- We will usually be assigning P(s) to block 'G' (the plant). Double-click the space next to 'G' and type your transfer function name and hit Enter. You must hit enter or nothing will happen.
- Once you hit enter, you should be able to click the OK button at the bottom of the window. Then the window will close.
- After you enter the transfer function, the root locus will be displayed. Double check to make sure that the open-loop poles and zeros of your plant are in the correct locations.

C. Generating the Step Response

- Click on **Analysis Responseto Step Command**.
- You will probably two curves on your step response plot. To fix this click on **Analysis Other Loop Responses...** Make sure only rto y is checked, and then click OK.
- You can now click on the pink boxes on the root locus (the current closed-loop poles for the given gain) and move them along the root locus. Essentially, you are exploring different controller gain values by doing this. Note how the step response changes as you move the closed-loop pole locations.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

- The values of the closed-loop poles will appear at the bottom of the root locus window as you click and hold the mouse on the pink boxes representing them. This only gives you the value of the closed-loop pole you are clicking on. If you need the other closed-loop pole locations, you will have to click on them on each of the other branches.

D. Entering the Compensator (Controller)

- Click **Compensators** **Edit** **C**. Click on **Add Lead** or **Add Lag** to enter Compensator. You will be able to make changes to these values later. After you are done, click **OK** to exit this window.
- Look at the form of $C(s)$ to be sure it is correct. Then look at the root locus window and see how it changed once the compensator was added.
- You can again see how the step response changes with the compensator by clicking on the closed-loop poles (the pink squares) and dragging them along the root locus.
- You can also change the location of the poles/zeros of the compensator by clicking on them and dragging them. Be careful not to inadvertently change the poles and zeros of the plant.

E. Adding Design Constraints

- Click **Edit** **Root Locus** **Design Constraints** then either **New** to add new constraints or **Edit** to edit existing constraints.
- At this point you can choose from settling time, percent overshoot, damping ratio, and natural frequency constraints.

EXPERIMENT NO. 4

STATE VARIABLE ANALYSIS USING CACSAD COMMAND TOOL. Familiarization and use of CACSAD command for state variable analysis. Obtaining transfer function from SV model and vice versa. Obtaining step response for a SISO system given in SV form. (Using MATLAB)

Objectives:

- i. Familiarization with the MATLAB commands for state variable analysis.
- Obtaining transfer function from a given SV model using MATLAB programming.
- j. Obtaining SV model from a given transfer function using MATLAB programming.
- k. Obtaining step response for a SISO system given in SV form using MATLAB programming

ASSIGNMENT - 1

1. Open MATLAB command window. Observe the commands ss2tf, tf2ss, ss2zp, eig, inv, ctrbf, obsv, lyap, canon, size, initial, rank.
2. Find out the Eigen Values & Eigen Vectors of the following matrices.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

$$\text{a) } A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad \text{b) } A = \begin{bmatrix} 4 & -1 & 5 \\ 2 & 1 & 3 \\ 6 & -7 & 9 \end{bmatrix}$$

verify the result with theoretical calculations.

ASSIGNMENT - 2

1. Obtain the state-space representation of the following system using MATLAB

$$\text{a) } \frac{C(s)}{R(s)} = \frac{12s+7}{s^3+5s^2+32s+7}$$
$$\text{b) } \frac{C(s)}{R(s)} = \frac{s^4+3s^3+12s^2+7s+8}{s^5+9s^4+5s^3+7s^2+7}$$

Draw the State Diagram of the above systems.

2. Find the Transfer Function of the following systems using MATLAB program

$$\text{a) } \dot{X}(t) = \begin{bmatrix} 0 & 1 \\ 2 & -3 \end{bmatrix} X(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t), \quad Y(t) = [1 \quad 0]X(t)$$

Check the result with theoretical calculation.

$$\text{b) } \dot{X}(t) = \begin{bmatrix} 0 & 3 & 7 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 5 & -6 & 9 & 5 \end{bmatrix} X(t) + \begin{bmatrix} 0 \\ 5 \\ 7 \\ 2 \end{bmatrix} u(t), \quad Y(t) = [1 \quad 3 \quad 6 \quad 5]X(t)$$

3. Find the unit step response of the following systems using MATLAB and SIMULINK.

$$\text{a) } \dot{X}(t) = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} X(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t), \quad Y(t) = [1 \quad 0]X(t)$$

$$\text{b) } \dot{X}(t) = \begin{bmatrix} -5 & 2 & 0 \\ 0 & -9 & 1 \\ 0 & 0 & -3 \end{bmatrix} X(t) + \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix} u(t), \quad Y(t) = [0 \quad 1 \quad 1]X(t)$$

$$\text{at initial condition } x(0) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

4. Obtain the Unit-Step response, unit-ramp response & unit-impulse response of the following system using MATLAB program

$$\dot{X}(t) = \begin{bmatrix} 1 & -1.5 \\ 2 & 0 \end{bmatrix} X(t) + \begin{bmatrix} 1.5 \\ 0 \end{bmatrix} u(t), \quad Y(t) = [1 \quad 0]X(t)$$

Also obtain the Bode Plot & Nyquist Plot for the above system

5. For the system below using SIMULINK

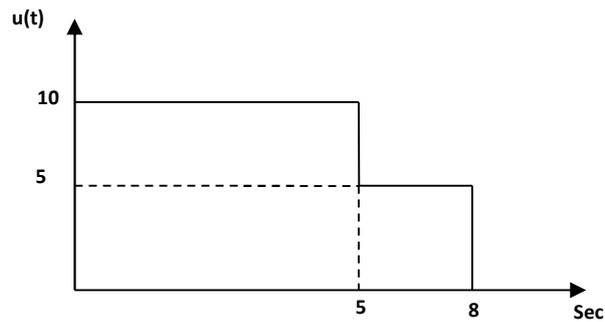
UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

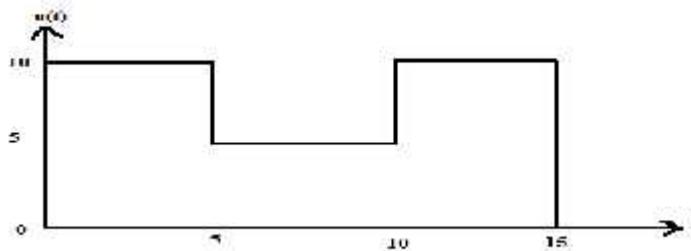
a.
$$\dot{X}(t) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 2 & -3 & -4 \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} u(t), \quad Y(t) = [1 \ 0 \ 0]x(t)$$

b. find the response due to initial condition $x(0) = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$

c. find the response due to an input $u(t)$, shown below.



and



UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Assignment - 1

The feedback control system has

$H(s)=1$ and

$G(s) = \frac{K}{s(s+2)(s+3)}$, K is the last two digits of your roll number

Using sisotool determine

- The range of controller gain for the system to be stable
- Value of controller gain for the complex dominant-poles damping ratio of 0.6. for this value of gain obtain the frequency response GM and PM. Also obtain step response and the time-domain specifications.

Assignment - 2

For the above system design a lead-lagcompensator for the following unit step response specifications.

- Dominant poles damping ratio $\zeta=0.707$
- Settling time less than 8 seconds.
- Peak response less than 1.2

EXPERIMENT NO. 5

STATE VARIABLE ANALYSIS USING CACSAD BLOCK DIAGRAM TOOL. Familiarization and use of CACSAD BLOCK DIAGRAM TOOL for state variable analysis. Obtaining step response and initial condition response for a single input, two output system given in SV form. (PSPICE, MATLAB, SciLab may be used).

Objectives:

- Familiarization with the MATLAB commands for state variable analysis.
- Obtaining step response and initial condition response for a SISO system given in SV form using MATLAB programming
- Obtaining step response and initial condition response for a Two input system given in SV form using MATLABprogramming.

ASSIGNMENT – 1

- Open MATLAB command window. Observe the commands jordan, fliplr, flipud, ctrbf, obsv, initial, kalman, acker, place.
- Write a MATLAB program to determine the STM for the given A matrix as
 - $A = \begin{bmatrix} 0 & -1 \\ 2 & -3 \end{bmatrix}$
Verify the result by theoretical calculation.
 - $A = \begin{bmatrix} 0 & 1 \\ -11 & -5 \end{bmatrix}$

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

ASSIGNMENT – 2

1. Test the Controllability and Observability of the following systems

$$\text{a) } \dot{X}(t) = \begin{bmatrix} 0.5 & 0 \\ 0 & -2 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t), \quad Y(t) = [0 \quad 1]x(t)$$
$$\text{b) } \dot{X}(t) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -3 \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} u(t), \quad Y(t) = [1 \quad 0 \quad 2]x(t)$$

2. Write a MATLAB program to determine the state feedback gain matrix so that closed loop poles are located at -2, -5 & -6

$$\text{c) } \dot{X}(t) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -30 & -1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(t), \quad Y(t) = [1 \quad 0 \quad 0]x(t)$$
$$\text{d) } \dot{X}(t) = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ -8 & 5 & -7 \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} u(t), \quad Y(t) = [0 \quad 1 \quad 1]x(t)$$

Verify obtained result with theoretical calculations

HINTS : Use 'place' or 'acker' command

EXPERIMENT NO. 6

PERFORMANCE ANALYSIS OF A DISCRETE TIME SYSTEM USING CACSAD TOOL. Familiarization and use of CACSAD block diagram tool for Digital Control System. Study of closed response of a continuous system with a digital controller with sample and hold. (PSPICE, MATLAB, SciLab may be used).

Objectives:

- o. Familiarization with the MATLAB commands for Digital control system analysis.
- p. Familiarization and use of SIMULINK block diagram tool for Digital Control System & Study of closed response of a continuous system with a digital controller with sample and hold.

ASSIGNMENT – 1

5. Open MATLAB command window. Observe the commands

ZTRANS, IZTRANS, c2d, D2C, D2D, zplane, residuez, freqz

6. Determine the z-transform of the following sequences using MATLAB

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

- i) $x(n) = \frac{1}{4^n} u(n)$
- ii) $x(n) = \frac{4^n}{3^n} u(n)$

7. Determine the inverse Z Transform of the followings

- i) $F(Z) = \frac{2Z^2 + 1.5Z}{Z^2 - 1.5Z + 0.5}$
- ii) $F(Z) = \frac{6 - 9Z^{-1}}{1 - 2.5Z^{-1} + Z^{-2}}$

8. Use the z-transform to solve the difference equation

$$y(k) - 0.6y(k-1) + 0.05y(k-2) = 32u(k-1) - 3u(k-2)$$

$$u(k) = \begin{cases} k, & k \geq 0 \\ 0, & k < 0 \end{cases}$$

$$y(k) = 0, \quad k < 0.$$

ASSIGNMENT – 2

1. Find out the discrete transfer function of the following continuous time systems.

2. $G(s) = \frac{2s}{s^2 - 1.2s + 0.5}$

3. $G(s) = \frac{s + 10}{s^2 + 2s + 0.5}$

[Hint : use c2d command]

2. Find out the continuous time transfer function of the following discrete time systems.

a. $G(Z) = \frac{2Z^2 + 1.5Z}{Z^2 - 1.5Z + 0.5}$

b. $G(Z) = \frac{10}{Z^2 - 1.5Z + 0.5}$

3. Consider the system described by the following difference equation

$$y(n) + 0.25y(n-1) = 5x(n) - 0.75x(n-1) - 1.75x(n-2)$$

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Determine the step & Impulse response for the system.

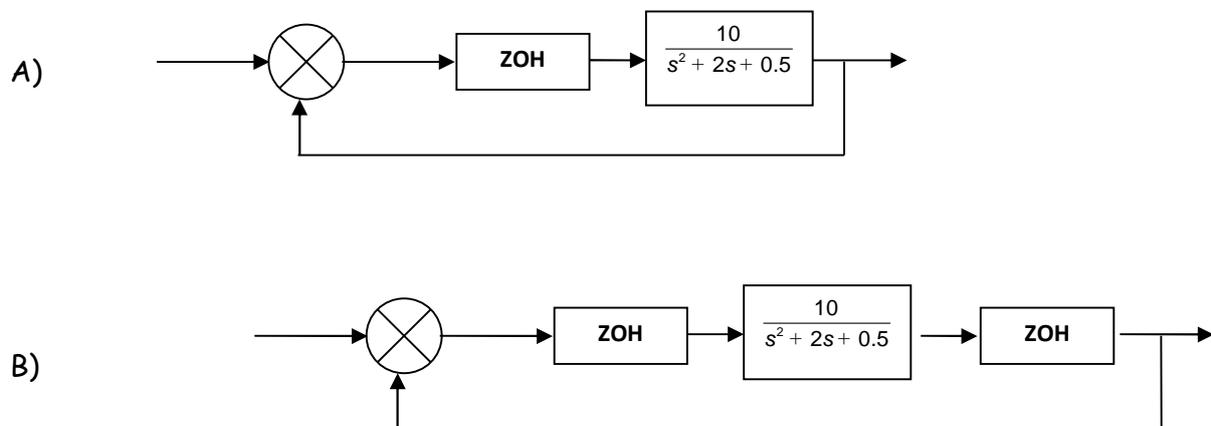
4. Write a MATLAB program to determine the pulse transfer function of the system if sampling time $T=0.1$ sec

$$G(Z) = \frac{1}{s^2 + 10s + 20}$$

Also get the step response for sampling time of $1/20$ sec

ASSIGNMENT – 3

Use simulink to determine the step & ramp response of the following systems



Observe the responses along with all the signals of the system

EXPERIMENT NO. 7

STUDYING THE EFFECTS OF NONLINEARITY IN A FEEDBACK CONTROLLED SYSTEM USING TIME RESPONSE. Determination of step response with a limiter nonlinearity introduced into the forward path of 2nd order unity feedback control systems. The open loop plant will have one pole at the origin and the other pole will be in LHP or RHP. To verify that (i) with open loop stable pole, the response is slowed down for larger amplitude input and (ii) for unstable plant, the closed loop system may become oscillatory with large input amplitude. (PSPICE, MATLAB, SciLab may be used).

Objectives:

- q. Introduction to the limiter type of nonlinearity.
- r. Determination of step response of a 2nd order unity feedback system with limiter nonlinearity in the forward path having open loop stable pole.
- s. Determination of step response of a 2nd order unity feedback system with limiter nonlinearity in the forward path having open loop unstable pole..

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

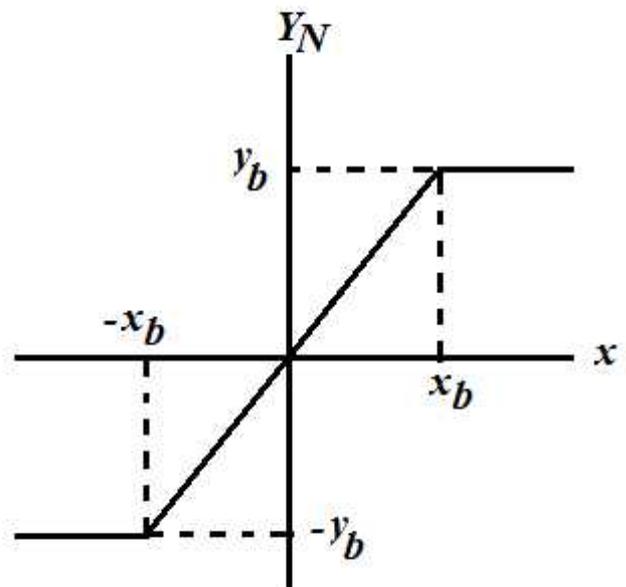
Lab Manual

Limiter type nonlinearity

A *limiter* is a common nonlinearity in control system, it simply limits the output to a certain threshold value. This type of nonlinearity is found in almost all type of amplifiers (electronic, mechanical, electromechanical etc.) where limited output power can not follow a large input signals. Sometime a limiter is very useful for limiting the output magnitude. For example, in a design in which the error signal driving the controller can reach very large values, it might be best to limit the error magnitude at the controller input to prevent excessive controller output magnitude.

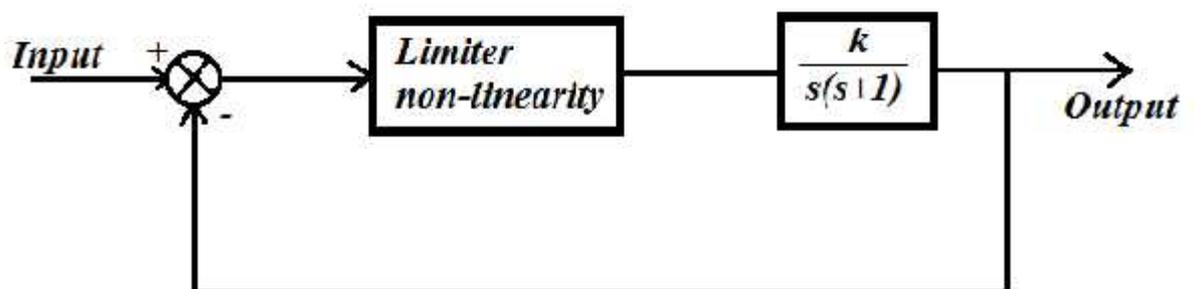
The figure shows the typical characteristic of a limiter and is described by the expression

$$\begin{aligned} Y_N &= kx && \text{for } -x_b \leq x \leq x_b \\ &= -y_b && \text{for } -x_b > x \\ &= y_b && \text{for } x > x_b \end{aligned}$$



Experiment – 1

1. Consider the system represented by the block diagram



- Take k as the last two digits of your roll number
2. Develop the block diagram using SIMULINK.

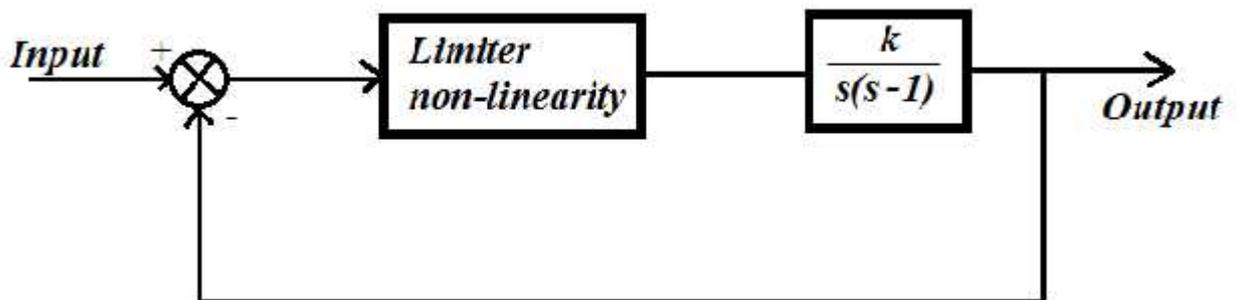
UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

3. Take the input a unit step and the slew rate of the limiter as 0.5
4. Record the response.
5. Increase the step magnitude from unity to 50 and then to 100 and record the response.
6. Repeat the steps 3 to 5 for slew rate 1.0

Experiment – 2

1. Consider the system represented by the block diagram



Take k as the last two digits of your roll number

2. Develop the block diagram using SIMULINK.
3. Take the input a unit step and the slew rate of the limiter as 0.5
4. Record the response.
5. Increase the step magnitude from unity to 50 and then to 100 and record the response.
6. Repeat the steps 3 to 5 for slew rate 1.0

EXPERIMENT NO. 8

STUDYING THE EFFECTS OF NONLINEARITY IN A FEEDBACK CONTROLLED SYSTEM USING PHASE PLANE PLOTS. Determination of phase plane trajectory and possibility of limit cycle of common nonlinearities. CACSAD block diagram tool will be used. (PSPICE, MATLAB, SciLab may be used).

Objectives:

- t. Introduction to Phase Plane Analysis & Phase Plane Plots.
- u. Determination of phase plane trajectory and possibility of limit cycle of common nonlinearities.

Phase Plane Analysis & Phase Plane Plots

Phase plane analysis is a graphical method for studying second-order systems.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

The phase plane method is concerned with the graphical study of second-order autonomous systems described by

$$\dot{x}_1 = f_1(x_1, x_2)$$

$$\dot{x}_2 = f_2(x_1, x_2)$$

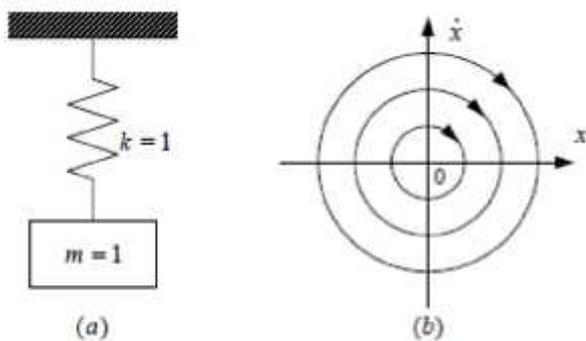
Where,

x_1, x_2 : states of the system

f_1, f_2 : nonlinear functions of the states

Geometrically, the state space of this system is a plane having x_1, x_2 as coordinates. This plane is called phase plane. The solution of the above equations with time varies from zero to infinity can be represented as a curve in the phase plane. Such a curve is called a phase plane trajectory. A family of phase plane trajectories is called a phase portrait of a system.

Phase portrait of a mass-spring system



The governing equation of the mass-spring system in Fig.(a) is the familiar linear second-order differential equation

$$\ddot{x} + x = 0$$

Assume that the mass is initially at rest, at length x_0 . Then the solution of this equation is

$$x(t) = x_0 \cos(t)$$

$$\dot{x}(t) = -x_0 \sin(t)$$

Eliminating time t from the above equations, we obtain the equation of the trajectories

$$x^2 + \dot{x}^2 = x_0^2$$

This represents a circle in the phase plane. Its plot is given in Fig. (b).

The nature of the system response corresponding to various initial conditions is directly displayed on the phase plane. In the above example, we can easily see that the system trajectories neither converge to the origin nor diverge to infinity. They simply circle around the origin, indicating the marginal nature of the system's stability.

A major class of second-order systems can be described by the differential equations of the form

$$\ddot{x} = f(x, \dot{x})$$

In state space form, this dynamics can be represented with $x_1 = x$ & $x_2 = \dot{x}$ & as follows

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = f(x_1, x_2)$$

Phase Plane Analysis of Linear Systems

The general form of a linear second-order system is

$$\dot{x}_1 = ax_1 + bx_2 \quad \text{-----(a)}$$

$$\dot{x}_2 = cx_1 + dx_2 \quad \text{-----(b)}$$

Transform these equations into a scalar second-order differential equation in the form $b\dot{x}_2 = bcx_1 + d(\dot{x}_1 - ax_1)$.

Consequently, differentiation of (a) and then substitution of (b) leads to

$$\ddot{x}_1 = (a + d)\dot{x}_1 + (cb - ad)x_1.$$

Therefore, we will simply consider the second-order linear system described by

$$\ddot{x} + a\dot{x} + bx = 0 \quad \text{-----}(c)$$

To obtain the phase portrait of this linear system, we solve for the time history

$$x(t) = k_1 e^{\lambda_1 t} + k_2 e^{\lambda_2 t} \quad \text{for } \lambda_1 \neq \lambda_2$$

$$x(t) = k_1 e^{\lambda_1 t} + k_2 t e^{\lambda_2 t} \quad \text{for } \lambda_1 = \lambda_2$$

where the constants λ_1, λ_2 are the solutions of the characteristic equation

$$s^2 + as + b = (s - \lambda_1)(s - \lambda_2) = 0$$

The roots λ_1, λ_2 can be explicitly represented as

$$\lambda_1 = \frac{-a + \sqrt{a^2 - 4b}}{2} \quad \text{and} \quad \lambda_2 = \frac{-a - \sqrt{a^2 - 4b}}{2}$$

For linear systems described by (c), there is only one singular point ($b \neq 0$), namely the origin. However, the trajectories in the vicinity of this singularity point can display quite different characteristics, depending on the values of a and b . The following cases can occur

- λ_1, λ_2 are both real and have the same sign (+ or -)
- λ_1, λ_2 are both real and have opposite sign
- λ_1, λ_2 are complex conjugates with non-zero real parts
- λ_1, λ_2 are complex conjugates with real parts equal to 0

We now briefly discuss each of the above four cases

Stable or unstable node (Fig.a -b)

The first case corresponds to a node. A node can be stable or unstable:

$\lambda_1, \lambda_2 < 0$: singularity point is called stable node.

$\lambda_1, \lambda_2 > 0$: singularity point is called unstable node.

There is no oscillation in the trajectories.

Saddle point (Fig.c)

The second case ($\lambda_1 < 0 < \lambda_2$) corresponds to a saddle point.

Because of the unstable pole λ_2 , almost all of the system trajectories diverge to infinity.

Stable or unstable locus (Fig. d-e)

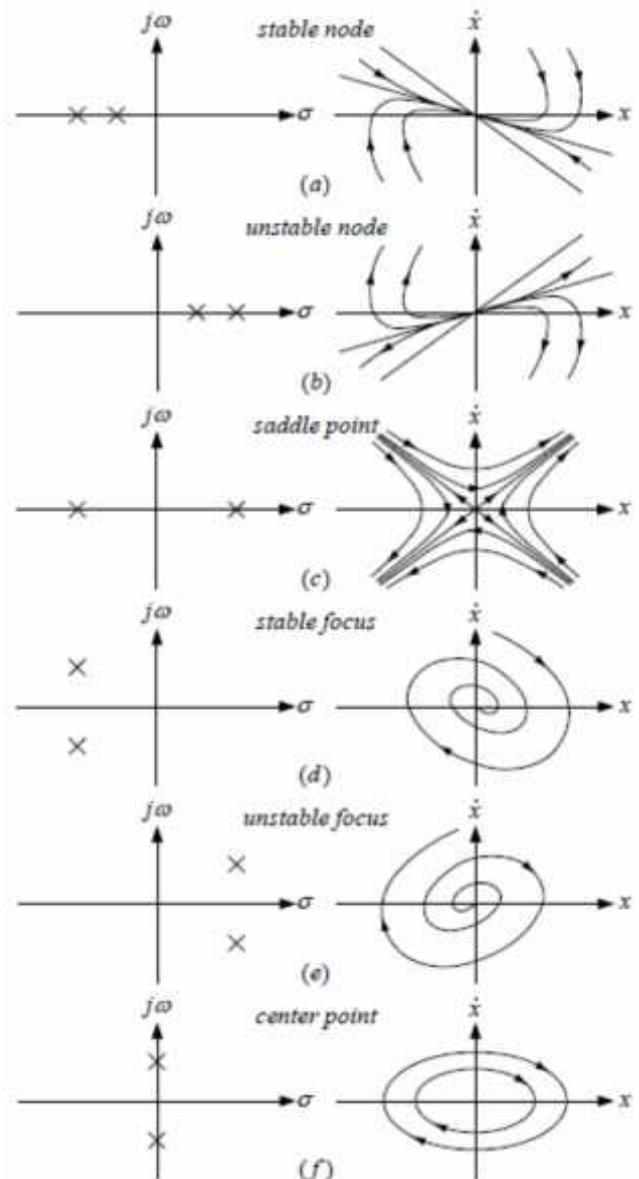
The third case corresponds to a focus.

$\text{Re}(\lambda_1, \lambda_2) < 0$: stable focus

$\text{Re}(\lambda_1, \lambda_2) > 0$: unstable focus

Center point (Fig. f)

The last case corresponds to a certain point. All trajectories are ellipses and the singularity point is the centre of these ellipses.



Phase Plane Analysis of Nonlinear Systems

In discussing the phase plane analysis of nonlinear system, two points should be kept in mind:

- Phase plane analysis of nonlinear systems is related to that of linear systems, because the local behavior of nonlinear systems can be approximated by the behavior of a linear system.
- Nonlinear systems can display much more complicated patterns in the phase plane, such as multiple equilibrium points and limit cycles.

Local behavior of nonlinear systems

If the singular point of interest is not at the origin, by defining the difference between the original state and the singular point as a new set of state variables, we can shift the singular point to the origin. Therefore, without loss of generality, we may simply consider the equation

$$\dot{x}_1 = f_1(x_1, x_2)$$

$$\dot{x}_2 = f_2(x_1, x_2)$$

with a singular point at 0. Using Taylor expansion, Eqs. can be rewritten in the form

$$\dot{x}_1 = a x_1 + b x_2 + g_1(x_1, x_2)$$

$$\dot{x}_2 = c x_1 + d x_2 + g_2(x_1, x_2)$$

where g_1, g_2 contain higher order terms. In the vicinity of the origin, the higher order terms can be neglected, and therefore, the nonlinear system trajectories essentially satisfy the linearized equation

$$\dot{x}_1 = ax_1 + bx_2$$

$$\dot{x}_2 = cx_1 + dx_2$$

As a result, the local behavior of the nonlinear system can be approximated by the patterns shown in the previous figures..

Limit cycle

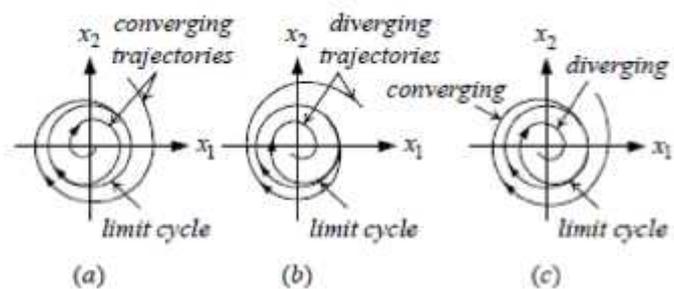
In the phase plane, a limit cycle is defined as an isolated closed curve. The trajectory has to be both closed, indicating the periodic nature of the motion, and isolated, indicating the limiting nature of the cycle (with nearby trajectories converging or diverging from it).

Depending on the motion patterns of the trajectories in the vicinity of the limit cycle, we can distinguish three kinds of limit cycles.

- **Stable Limit Cycles:** all trajectories in the vicinity of the limit cycle converge to it as $t \rightarrow \infty$ (Fig. a).

- **Unstable Limit Cycles:** all trajectories in the vicinity of the limit cycle diverge to it as $t \rightarrow \infty$ (Fig. b)

- **Semi-Stable Limit Cycles:** some of the trajectories in the vicinity of the limit cycle converge to it as $t \rightarrow \infty$ (Fig. c)



UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Assignment # 1

6. Develop the phase plane trajectory using MATLAB for the system described by the dynamic equation

$$\ddot{y} + \dot{y} + 9y = 0$$

7. Consider the system represented by the state space model

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -10 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} k \\ 0 \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Take k = Last two digits of your roll no.

Develop the phase plane trajectory using MATLAB for a unit step & unit impulse input.

Assignment # 2

Consider the system represented by the state space model

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = k(1 - x_1^2) - x_1$$

Develop the phase plane trajectory using MATLAB. For k ranging from 1 to 10.

Assignment # 3

Consider the system represented by the state space model

$$\dot{x}_1 = x_2 - x_1(x_1^2 + x_2^2 - 1)$$

$$\dot{x}_2 = -x_1 - x_2(x_1^2 + x_2^2 - 1)$$

Develop the phase plane trajectory using MATLAB .

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Power System-II Lab

Course Code: EE692

L-T-P scheme: 0-0-3

Course Credit: 2

Objectives:

1. To learn and understand the characteristics of an Three phase over voltage relay.
2. To provide an understanding of the design aspects of single phase over current relay.
3. To provide an efficient understanding of performance of load flow analysis using Newton Raphson method. (by simulation).
4. To provide an efficient understanding of performance of load flow analysis using Gauss-seidel method (by simulation).
5. To learn and understand the performance of a single phase under voltage relay.

Learning Outcomes: The students will have a detailed knowledge of the concepts of three phase over voltage relay, single phase under voltage relay. Aware of a variety of approaches load flow analysis. Student will learn the basics behind single phase overcurrent relay. Upon the completion of Power system course, the student will be able to:

-) **Understand** and implement characteristics of an various three phase relay circuit.
-) **Use** modern tools and circuit to understand the performance of load flow study.
-) **Understand** the concepts of single phase over current relay and single phase under voltage relay

Course Contents:

Exercises that must be done in this course are listed below:

Exercise No. 1: To study the different characteristics of single phase over current relay.

Exercise No. 2: To study the characteristics of single phase under voltage relay.

Exercise No. 3: To study the characteristics of three phase over voltage relay.

Exercise No. 4: To study on AC load flow using Gauss-seidel method (by simulation).

Exercise No. 5: To study on AC load flow using Newton Raphson method. (by simulation).

Text Book:

1. Power System Protection and Switchgear, adri Ram, D. N. Vishwakarma, Tata McGraw-Hill Education.

2. Etap 6.0 power station user guide, by etap.

3. Protection and Switchgear, Bhavesh Bhalja, Maheshwari, Nilesh Chothani OUP India.

4. Handbook of Switchgears By Bhel, Tata McGraw-Hill Education, 2005 - Electric switchgear.

Recommended Systems/Software Requirements:

1. Intel based desktop PC with minimum of 166 MHZ or faster processor with at least 64 MB RAM and 100 MB free disk space.

2. Etap 6.0 power station. EMTP.

Testing kit of relay protection various type of feeder, transformers, alternator and motors protection

UNIVERSITY OF ENGINEERING AND MANAGEMENT,
JAIPUR
Lab Manual

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Experiment No: 1 Study the operating characteristics of IDMT over current relay

Aim: To Study the operating characteristics of IDMT over current relay.

Apparatus Required:

Theory:

The function of a relay is to detect abnormal conditions in the system and to initiate through appropriate circuit breakers the disconnection of faulty circuits so that interference with the general supply is minimized. Relays are of many types. Some depend on the operation of an armature by some form of electromagnet. A very large number of relays operate on the induction principle. When a relay operates it closes contacts in the trip circuit which is normally connected across 110 V D.C. supply from a battery. The passage of current in the coil of the trip circuit actuates the plunger, which causes operation of the circuit breaker, disconnecting the faulty system.

In the laboratory, a 3-phase contactor simulates the operation of the circuit breaker. The closure of the relay contacts short-circuits the 'no-volt' coil of the contactor, which, in turn, disconnects the faulty system.

The protective relaying which responds to a rise in current flowing through the protected element over a pre-determined value is called 'overcurrent protection' and the relays used for this purpose are known as overcurrent relays. Earth fault protection can be provided with normal overcurrent relays, if the minimum earth fault current is sufficient in magnitude. The design of a comprehensive protection scheme in a power system requires the detailed study of time-current characteristics of the various relays used in the scheme. Thus it is necessary to obtain the time-current characteristics of these relays.

The overcurrent relay works on the induction principle. The moving system consists of an aluminium disc fixed on a vertical shaft and rotating on two jewelled bearings between the poles of an electromagnet and a damping magnet. The winding of the electromagnet is provided with seven taps (generally 0, which are brought on the front panel, and the required tap is selected by a push-in-type plug. The pick-up current setting can thus be varied by the use of such plug multiplier setting. The pick-up current values of earth fault relays are normally quite low.

The operating time of all overcurrent relays tends to become asymptotic to a definite minimum value with increase in the value of current. This is an inherent property of the electromagnetic relays due to saturation of the magnetic circuit. By varying the point of saturation, different characteristics can be obtained and these are

1. Definite time
2. Inverse Definite Minimum Time (IDMT)
3. Very Inverse
4. Extremely Inverse.

The torque of these relays is proportional to $\phi_1 \phi_2 \sin \alpha$, where ϕ_1 and ϕ_2 are the two fluxes and α is the angle between them. Where both the fluxes are produced by the same quantity (single quantity relays) as in the case of current or voltage operated, the torque T is proportional to I^2 , or $T = K I^2$, for coil current below saturation. If the core is made to saturate at very early stages such that with increase

UNIVERSITY OF ENGINEERING AND MANAGEMENT,
JAIPUR
Lab Manual

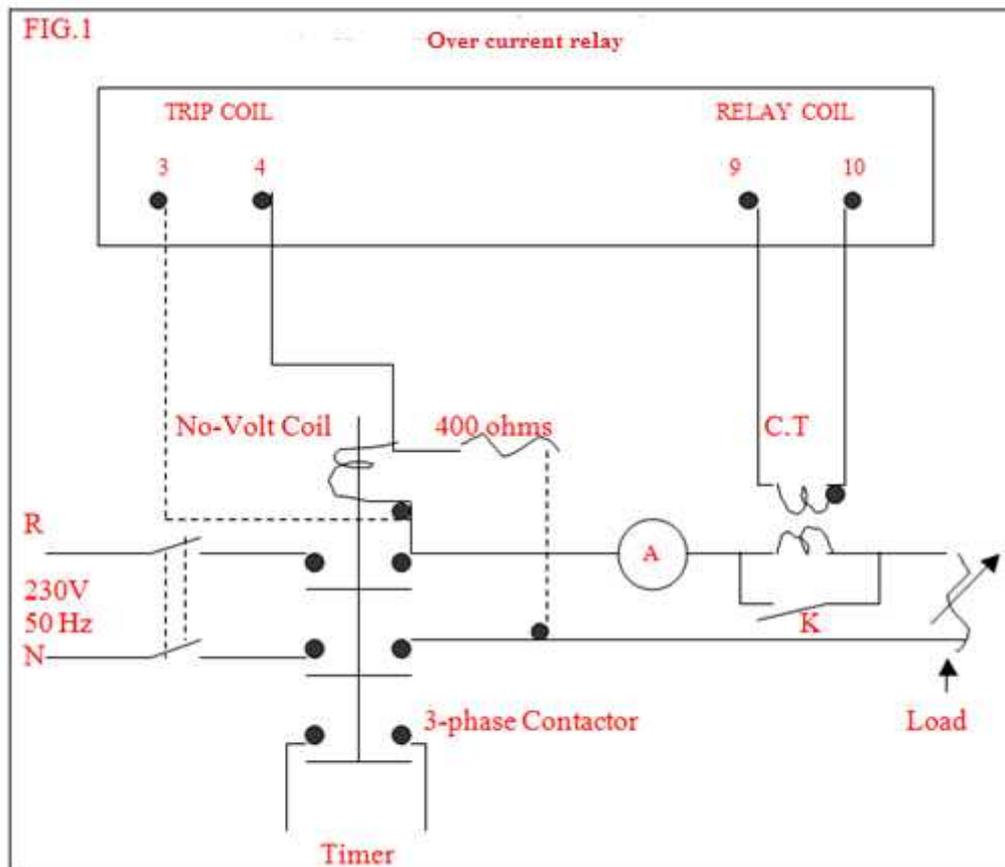
of I, K decreases so that the time of operation remains the same over the working range. The time-current characteristic obtained is known as definite-time characteristic.

If the core is made to saturate at a later stage, the characteristic obtained is known as IDMT. The time-current characteristic is inverse over some range and then after saturation assumes the definite time form. In order to ensure selectivity, it is essential that the time of operation of the relays should be dependent on the severity of the fault in such a way that more severe the fault, the less is the time to operate, this being called the inverse-time characteristic. This will also ensure that a relay shall not operate under normal overload conditions of short duration.

It is essential also that there shall be a definite minimum time of operation, which can be adjusted to suit the requirements of the particular installation. At low values of operating current the shape of the curve is determined by the effect of the restraining force of the control spring, while at high values the effect of saturation predominates. Different time settings can be obtained by moving a knurled clamping screw along a calibrated scale graduated from 0.1 to 1.0 in steps of 0.05. This arrangement is called Time Multiplier Setting and will vary the travel of the disc required to close the contacts. This will shift the time-current characteristic of the relay parallel to itself.

By delaying the saturation to a further point, the Very Inverse and Extremely Very Inverse time current characteristics can be obtained.

Circuit Diagram:



UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Procedure:

1. Study the construction of the relay and identify the various parts.
2. Connect as in FIG.1
3. Set the pick-up value of the current marked 1 A(100 % f. l current) by inserting the plug in the groove.
4. Set the Time Multiplier Setting (TMS) initially at 1.0.
5. Adjust the load current to about 1.3 times the f.l current by shorting the switch K. Open the switch K to permit this adjusted current to flow through the relay and record the time taken for this overload condition.
6. Vary the value of the load current in steps and record the time taken for the operation of the relay in each case with the help of the timer.
7. Repeat steps 5 and 6 for TMS values of 0.2, 0.4, 0.6 and 0.8.
8. Repeat the above experiment with different pick up current values using the plug setting bridge.

Observation table:

Type of Relay:

Pick-up current (plug setting multiplier) = ----- Amps

S.No	Current, A	Current(A) times the plug setting multiplier	Operating time in sec. for TMS of				
			0.2	0.4	0.6	0.8	1.0

Data Processing & analysis:

1. Plot operating time versus the multiples of plug setting values for different TMS on the same graph for overcurrent relay.
2. Plot on a log -log sheet operating time versus the multiples of the plug setting value for different time multiplier settings on the same graph sheet for the over current relay.
3. Plot the curves for earth fault relay as in steps 1 and 2 respectively.
4. Observe the Inverse nature of the characteristics as well as the definite time required for the operation of the relay in either case.

Result:

Post Experiment Questions:

1. When is it necessary to add a direction feature for overcurrent protection?
2. Can the overcurrent and earth fault relays be made to operate instantaneously and how?
3. What are the equations of IDMT, very Inverse and Extremely Inverse overcurrent relays?

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Experiment No: 2 study the performance of a single phase undervoltage relay.

Aim: To Study the operating characteristics of IDMT single phase under voltage relay.

Apparatus Required:

Theory:

The function of a relay is to detect abnormal conditions in the system and to initiate through appropriate circuit breakers the disconnection of faulty circuits so that interference with the general supply is minimized. Relays are of many types. Some depend on the operation of an armature by some form of electromagnet. A very large number of relays operate on the induction principle. When a relay operates it closes contacts in the trip circuit which is normally connected across 110 V D.C. supply from a battery. The passage of current in the coil of the trip circuit actuates the plunger, which causes operation of the circuit breaker, disconnecting the faulty system.

In the laboratory, a 3-phase contactor simulates the operation of the circuit breaker. The closure of the relay contacts short-circuits the 'no-volt' coil of the contactor, which, in turn, disconnects the faulty system.

The protective relaying which responds to a drop in voltage flowing through the protected element over a pre-determined value is called 'under voltage protection' and the relays used for this purpose are known as under voltage relays. The design of a comprehensive protection scheme in a power system requires the detailed study of time-current characteristics of the various relays used in the scheme. Thus it is necessary to obtain the time-current characteristics of these relays.

The under voltage relay works on the induction principle. The moving system consists of an aluminium disc fixed on a vertical shaft and rotating on two jewelled bearings between the poles of an electromagnet and a damping magnet. The winding of the electromagnet is provided with seven taps (generally 0, which are brought on the front panel, and the required tap is selected by a push-in type plug. The pick-up current setting can thus be varied by the use of such plug multiplier setting. The pick-up current values of earth fault relays are normally quite low.

The operating time of all under voltage relays tends to become asymptotic to a definite minimum value with increase in the value of current. This is an inherent property of the electromagnetic relays due to saturation of the magnetic circuit. By varying the point of saturation, different characteristics can be obtained and these are

1. Definite time
2. Inverse Definite Minimum Time (IDMT)
3. Very Inverse
4. Extremely Inverse.

The torque of these relays is proportional to $\phi_1 \phi_2 \sin \alpha$, where ϕ_1 and ϕ_2 are the two fluxes and α is the angle between them. Where both the fluxes are produced by the same quantity (single quantity relays) as in the case of current or voltage operated, the torque T is proportional to I^2 , or $T = K I^2$, for coil current below saturation. If the core is made to saturate at very early stages such that with increase of I , K decreases so that the time of operation remains the same over the working range. The time-current characteristic obtained is known as definite-time characteristic.

If the core is made to saturate at a later stage, the characteristic obtained is known as IDMT. The time-current characteristic is inverse over some range and then after saturation assumes the definite time form. In order to ensure selectivity, it is essential that the time of operation of the relays should be

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

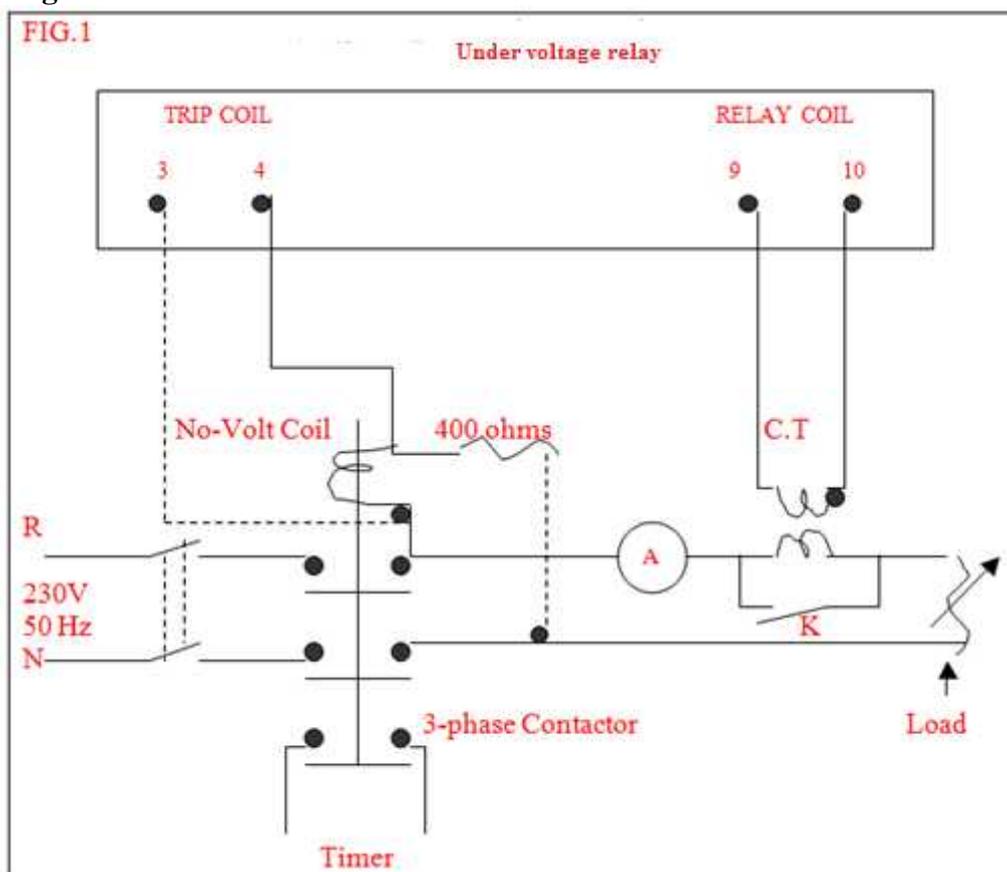
Lab Manual

dependent on the severity of the fault in such a way that more severe the fault, the less is the time to operate, this being called the inverse-time characteristic. This will also ensure that a relay shall not operate under normal overload conditions of short duration.

It is essential also that there shall be a definite minimum time of operation, which can be adjusted to suit the requirements of the particular installation. Different time settings can be obtained by moving a knurled clamping screw along a calibrated scale graduated from 0.1 to 1.0 in steps of 0.05. This arrangement is called Time Multiplier Setting and will vary the travel of the disc required to close the contacts. This will shift the time-current characteristic of the relay parallel to itself.

By delaying the saturation to a further point, the Very Inverse and Extremely Very Inverse time current characteristics can be obtained.

Circuit Diagram:



Procedure:

1. Study the construction of the relay and identify the various parts.
2. Connect as in FIG.1
3. Set the pick-up value of the voltage marked 1 A(100 % f. l current) by inserting the plug in the groove.
4. Set the Time Multiplier Setting (TMS) initially at 1.0.
5. Adjust the load voltage to about 1.3 times the f.l current by shorting the switch K. Open the switch K to permit this adjusted current to flow through the relay and record the time taken for this overload condition.
6. Vary the value of the load voltage in steps and record the time taken for the operation of the relay in each case with the help of the timer.
7. Repeat steps 5 and 6 for TMS values of 0.2, 0.4, 0.6 and 0.8.
8. Repeat the above experiment with different pick up current values using the plug setting bridge

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

Observation table:

Type of Relay:

Pick-up current (plug setting multiplier) = ----- Volts

S.No	Voltage (V)	Voltage (V) times the plug setting multiplier	Operating time in sec. for TMS of				
			0.2	0.4	0.6	0.8	1.0

Data Processing & analysis:

1. Plot operating time versus the multiples of plug setting values for different TMS on the same graph for under voltage relay.
2. Plot on a log -log sheet operating time versus the multiples of the plug setting value for different time multiplier settings on the same graph sheet for the under voltage relay.
3. Plot the curves for earth fault relay as in steps 1 and 2 respectively.
4. Observe the Inverse nature of the characteristics as well as the definite time required for the operation of the relay in either case.

Result:

Post Experiment Questions:

1. When is it necessary to add a direction feature for under protection?
2. Can the under voltage and earth fault relays be made to operate instantaneously and how?
3. What are the equations of IDMT, very Inverse and Extremely Inverse Under voltage relays?

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Experiment No: 3 study the performance of three phase overvoltage relay.

Aim: To Study the operating characteristics of IDMT three phase over voltage relay.

Apparatus Required:

Theory:

Overvoltages occur in a system when the system voltage rises over 110% of the nominal rated voltage. Overvoltage can be caused by a number of reasons, sudden reduction in loads, switching of transient loads, lightning strikes, failure of control equipment such as voltage regulators, neutral displacement,. Overvoltage can cause damage to components connected to the power supply and lead to insulation failure, damage to electronic components, heating, flashovers, etc.

Overvoltage relays can be used to identify overvoltages and isolate equipment. These relays operate when the measured voltage exceeds a predetermined set-point. The voltage is usually measured using a Potential Transformers. The details of the ratio of the potential transformer are also entered into the relay. These relays are usually provided with a time delay. The time delay can be either instantaneous, fixed time or for IDMT (inverse definite minimum time) curves.

Generally, overvoltage relays are provided with sufficient time delay in order to avoid unwanted trippings due to transients (See article on Transients).

These relays can be used to isolate feeders and other equipment connected to the network. In the case of generators, these relay also switch off the excitation system to the generators thereby preventing voltage build-up.

The function of a relay is to detect abnormal conditions in the system and to initiate through appropriate circuit breakers the disconnection of faulty circuits so that interference with the general supply is minimized. Relays are of many types. Some depend on the operation of an armature by some form of electromagnet. A very large number of relays operate on the induction principle. When a relay operates it closes contacts in the trip circuit which is normally connected across 110 V D.C. supply from a battery. The passage of current in the coil of the trip circuit actuates the plunger, which causes operation of the circuit breaker, disconnecting the faulty system.

In the laboratory, a 3-phase contactor simulates the operation of the circuit breaker. The closure of the relay contacts short-circuits the 'no-volt' coil of the contactor, which, in turn, disconnects the faulty system.

The protective relaying which responds to a rise in voltage flowing through the protected element over a pre-determined value is called over voltage protection' and the relays used for this purpose are known as over voltage relays. The design of a comprehensive protection scheme in a power system requires the detailed study of time-current characteristics of the various relays used in the scheme. Thus it is necessary to obtain the time- current characteristics of these relays.

The over voltage relay works on the induction principle. The moving system consists of an aluminium disc fixed on a vertical shaft and rotating on two jewelled bearings between the poles of an electromagnet and a damping magnet. The winding of the electromagnet is provided with seven taps (generally 0, which are brought on the front panel, and the required tap is selected by a push-in -type plug. The pick-up current setting can thus be varied by the use of such plug multiplier setting. The pick-up current values of earth fault relays are normally quite low.

The operating time of all over voltage relays tends to become asymptotic to a definite minimum value with increase in the value of current. This is an inherent property of the electromagnetic relays due to saturation of the magnetic circuit. By varying the point of saturation, different characteristics can be obtained and these are

1. Definite time
2. Inverse Definite Minimum Time (IDMT)
3. Very Inverse
4. Extremely Inverse.

The torque of these relays is proportional to $\phi_1 \phi_2 \sin \alpha$, where ϕ_1 and ϕ_2 are the two fluxes and α is the angle between them. Where both the fluxes are produced by the same quantity (single quantity relays) as in the case of current or voltage

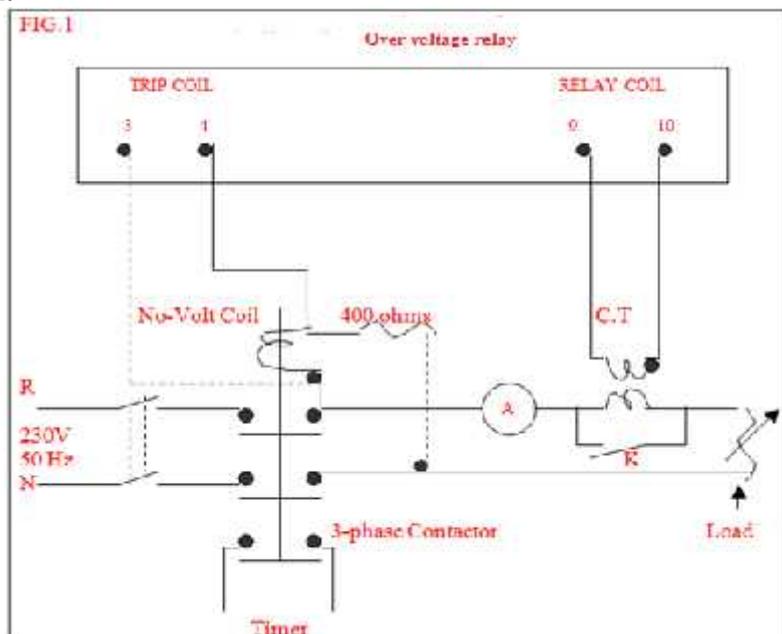
operated, the torque T is proportional to I^2 , or $T = K I^2$, for coil current below saturation. If the core is made to saturate at very early stages such that with increase of I , K decreases so that the time of operation remains the same over the working range. The time - current characteristic obtained is known as definite -time characteristic.

If the core is made to saturate at a later stage, the characteristic obtained is known as IDMT. The time-volt characteristic is inverse over some range and then after saturation assumes the definite time form. In order to ensure selectivity, it is essential that the time of operation of the relays should be dependent on the severity of the fault in such a way that more severe the fault, the less is the time to operate, this being called the inverse-time characteristic. This will also ensure that a relay shall not operate under normal overload conditions of short duration.

It is essential also that there shall be a definite minimum time of operation, which can be adjusted to suit the requirements of the particular installation. At low values of operating current the shape of the curve is determined by the effect of the restraining force of the control spring, while at high values the effect of saturation predominates. Different time settings can be obtained by moving a knurled clamping screw along a calibrated scale graduated from 0.1 to 1.0 in steps of 0.05. This arrangement is called Time Multiplier Setting and will vary the travel of the disc required to close the contacts. This will shift the time-current characteristic of the relay parallel to itself.

By delaying the saturation to a further point, the Very Inverse and Extremely Very Inverse time current characteristics can be obtained.

Circuit Diagram:



UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Procedure:

1. Study the construction of the relay and identify the various parts.
2. Connect as in FIG.1
3. Set the pick-up value of the voltage marked 1 A(100 % f.l current) by inserting the plug in the groove.
4. Set the Time Multiplier Setting (TMS) initially at 1.0.
5. Adjust the load voltage to about 1.3 times the f.l current by shorting the switch K. Open the switch K to permit this adjusted current to flow through the relay and record the time taken for this overload condition.
6. Vary the value of the load voltage in steps and record the time taken for the operation of the relay in each case with the help of the timer.
7. Repeat steps 5 and 6 for TMS values of 0.2, 0.4,0.6 and 0.8.
8. Repeat the above experiment with different pick up current values using the plug setting bridge.

Observation table:

Type of Relay:

Pick-up current (plug setting multiplier) = ----- Volts(R phase)

Pick-up current (plug setting multiplier) = ----- Volts(Y phase)

Pick-up current (plug setting multiplier) = ----- Volts(B phase)

S.No	Voltage (V) Phase R	Voltaage (V) times the plug setting multiplier for R phase	Operating time in sec. for TMS of				
			0.2	0.4	0.6	0.8	1.0
S.No	Voltage (V) Phase Y	Voltaage (V) times the plug setting multiplier for Y phase	Operating time in sec. for TMS of				
			0.2	0.4	0.6	0.8	1.0
S.No	Voltage (V) Phase B	Voltaage (V) times the plug setting multiplier for B phase	Operating time in sec. for TMS of				
			0.2	0.4	0.6	0.8	1.0

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

Data Processing & analysis:

1. Plot operating time versus the multiples of plug setting values for different TMS on the same graph for over voltage relay.
2. Plot on a log-log sheet operating time versus the multiples of the plug setting value for different time multiplier settings on the same graph sheet for the over voltage relay for phase R, Y and B.
4. Observe the Inverse nature of the characteristics as well as the definite time required for the operation of the relay in either case.

Result:

Post Experiment Questions:

1. When is it necessary to add a direction feature for under protection?
2. Can the under voltage and earth fault relays be made to operate instantaneously and how?
3. What are the equations of IDMT, very Inverse and Extremely Inverse over voltage relays?

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Experiment No: 4 study on AC load flow using –seidelmethod (by simulation).

Aim: To study on AC load flow using seidel methodmethod (by simulation).

Apparatus Required:

Theory:

Load flow studies [1,2] are used to ensure that electrical power transfer from generators to consumers through the grid system is stable, reliable and economic. Conventional techniques for solving the load flow problem are iterative, using the Newton-Raphson or the Gauss-Seidel methods. Recently, however, there has been much interest in the application of stochastic search methods, such as Genetic Algorithms [3,4,5], to solving power system problems. The increasing presence of distributed alternative energy sources, often in geographically remote locations, complicates load flow studies and has triggered a resurgence of interest in the topic.

The principles of power system load flow studies are taught within elective modules in the later years of undergraduate electrical engineering courses, or as essential components of specialist masters programmes in electrical power engineering. From the educational viewpoint, therefore, the topic is important, yet a complete coverage presents some significant challenges. Pre-requisites include fundamental concepts from a.c. circuit analysis, such as phasor notation, impedance and admittance, power and reactive power, three-phase and per-unit systems, all of which are regarded as 'difficult' by many students. The load flow solution techniques bring extra mathematical hurdles, including matrix representation (with complex number coefficients), iterative methods and probability functions.

To assist in the teaching of load flow analysis techniques the Excel Workbook illustrates four different methods of solving a simple load flow problem, which nevertheless has all of the features to be found in a larger-scale system. The Workbook allows students to vary both the problem being solved, by adjusting the power or voltage levels and line impedances, and also the parameters of the solution methods, such as numerical acceleration factors. These notes present an overview of the general power system load flow problem and describe its solution using four techniques: Newton-Raphson, Gauss-Seidel, Genetic Algorithm and Simulated Annealing. Also included are illustrative numerical results relating to the particular power system configuration analysed in the Workbook.

2. Formulation of the Load Flow Problem

Load flow studies are based on a nodal voltage analysis of a power system. As an example, consider the very simple system represented by the single-line diagram in Fig. 1. Here two generators (1 and 2) are interconnected by one transmission line and are separately connected to a load (3) by two other lines. If the phasor currents injected into the system are I_1 , I_2 , and I_3 , and the lines are modelled by

simple series admittances, then it is possible to draw the equivalent circuit for one representative phase of the balanced three-phase system, as shown in Fig. 2.

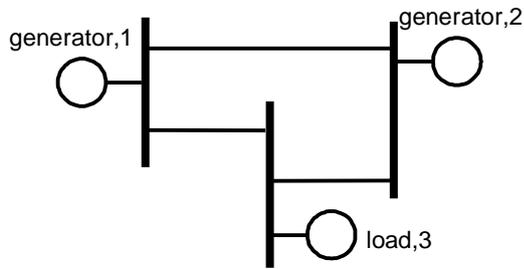


Fig. 1 Single-line diagram of a simple example power system

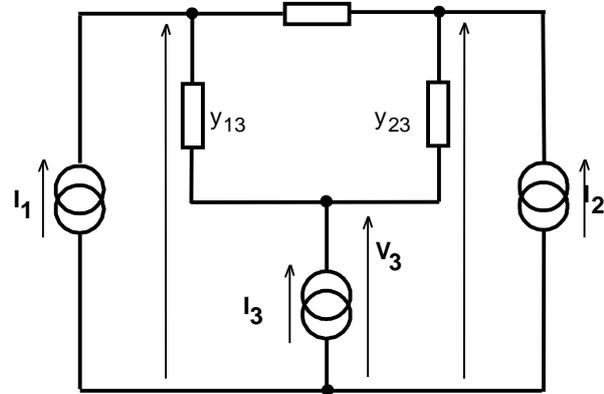


Fig. 2 Equivalent circuit for one phase of the system shown in Fig.1

For the circuit in Fig. 2 the nodal voltage equations can be written directly. For example, at node 1:

$$I_1 = (Y_{12} + Y_{13})V_1 - Y_{12}V_2 - Y_{13}V_3 \quad (1)$$

In general, for a system with r nodes, then at node n :

$$I_n = Y_{n1}V_1 + Y_{n2}V_2 + \dots + Y_{nn}V_n + \dots + Y_{nr}V_r = \sum_{k=1}^r Y_{nk}V_k \quad (2)$$

where: Y_{nn} = sum of all admittances connected to node n

Y_{nk} = - (sum of all admittances connected between nodes n and k) = Y_{kn}

I_n = current injected at node n

For the complete system of r nodes:

$$\begin{bmatrix} I_1 \\ \vdots \\ I_n \\ \vdots \\ I_r \end{bmatrix} = \begin{bmatrix} Y_{11} & \dots & Y_{1n} & \dots & Y_{1r} \\ \vdots & & \vdots & & \vdots \\ Y_{n1} & \dots & Y_{nn} & \dots & Y_{nr} \\ \vdots & & \vdots & & \vdots \\ Y_{r1} & \dots & Y_{rn} & \dots & Y_{rr} \end{bmatrix} \begin{bmatrix} V_1 \\ \vdots \\ V_n \\ \vdots \\ V_r \end{bmatrix} \quad \text{or } [I] = [Y][V] \quad (3)$$

where $[Y]$ is the nodal admittance matrix. Formulation of the load flow problem is most conveniently carried out with the terms in the nodal admittance matrix

expressed in polar notation: $Y_{kn} = |Y_{kn}| \angle \theta_{kn}$

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

= $Y_{kn} \angle \theta_{kn}$. The Excel Workbook (Sheet 2) allows

users to enter series impedance data for the three lines and then automatically calculates the terms in the nodal admittance matrix.

Conventional circuit analysis proceeds directly from equation (3) by inverting the nodal admittance matrix and hence solving for the nodal voltages $[V]$. However, the load flow problem is complicated by the lack of uniformity in the data about electrical conditions at the nodes. There are three distinct types of nodal data, which relate to the physical nature of the power system:

- a) Load nodes, where complex power $S_n = P_n + jQ_n$ taken from or injected into the system is defined. Such nodes may also include links to other systems. At these load nodes, the voltage magnitude $|V_n|$ and phase angle δ_n must be calculated.
- b) Generator nodes, where the injected power, P_n , and the magnitude of the nodal voltage $|V_n|$ are specified. These constraints reflect the generator's operating characteristics, in which power is controlled by the governor and terminal voltage is controlled by the automatic voltage regulator. At the generator nodes the voltage phase angle δ_n must be calculated
- c) At least one node, termed the 'floating bus' or 'slack bus', where the nodal voltage magnitude $|V_n|$ and phase angle δ_n are specified. This node acts as the reference node and is commonly chosen to have a phase angle $\delta = 0^\circ$. The power and reactive power delivered at this node are not specified.

In the system configuration of Fig. 1, which is analysed in the Excel Workbook, each type of node is represented with node 1 being a floating bus, node 2 being a generator node and node 3 being a load node. Consequently values must be specified for the power (P_2) injected at node 2, and the power (P_3) and reactive

power (Q_3) injected at node 3. All three of these power values may be changed by the user, though default values are provided ($P_2 = 1.0$; $P_3 = -1.5$; $Q_3 = -0.2$), with

negative values indicating that power or reactive power is being drawn from the system. The magnitude of the voltage at node 1 can be specified, with the default

value being 1.0 pu, while the phase angle is fixed at 0° ($V = 1.0 \angle 0^\circ$). At the

generator node (node 2), the voltage magnitude can be set by the user with the

default value being 1.1 pu ($V_2 = 1.1 \angle \delta_2$), and the phase angle δ_2 is calculated during

the load flow solution. At the load node (node 3) the voltage magnitude and phase

angle have to be calculated ($V_3 = |V_3| \angle \delta_3$). So the complete load flow problem for

his particular power system configuration involves the calculation of the voltage magnitude V_3 and the phase angles δ_2, δ_3 .

Let us start the procedure with bus-2 of the 5 bus 7 line system given in fig: 4.1. Since this is load bus, both the real and reactive power into this bus is known. We can therefore write

$$V_2^{(1)} = \frac{1}{Y_{22}} \left[\frac{P_{2,3W} - jQ_{2,3W}}{V_2^{*(0)}} - Y_{21}V_1 - Y_{23}V_3^{(0)} - Y_{24}V_4^{(0)} - Y_{25}V_5^{(0)} \right] \quad (4.13)$$

from (4.12)

$$V_2^{(1)} = \frac{1}{Y_{22}} \left[\frac{-0.96 + j0.62}{1} - 1.05Y_{21} - Y_{23} - Y_{24} - 1.02Y_{25} \right]$$

From the data given in Table 4.3 we can write

It is to be noted that since the real and reactive power is drawn from this bus, both these quantities appear in the above equation with a negative sign. With the values of the Y bus elements given in Table 4.2 we get $V_2^{(1)} = 0.9927 \angle -2.5959^\circ$.

$$V_3^{(1)} = \frac{1}{Y_{33}} \left[\frac{P_{3,3W} - jQ_{3,3W}}{V_3^{*(0)}} - Y_{31}V_1 - Y_{32}V_2^{(1)} - Y_{34}V_4^{(0)} - Y_{35}V_5^{(0)} \right] \quad (4.14)$$

The first iteration voltage of bus-3 is given by

Note that in the above equation since the update for the bus-2 voltage is already available, we used the 1st iteration value of this rather than the initial value. Substituting the numerical

$$V_4^{(1)} = \frac{1}{Y_{44}} \left[\frac{P_{4,3W} - jQ_{4,3W}}{V_4^{*(0)}} - Y_{41}V_1 - Y_{42}V_2^{(1)} - Y_{44}V_3^{(1)} - Y_{45}V_5^{(0)} \right] \quad (4.15)$$

data we get $V_3^{(1)} = 0.9883 \angle -2.8258^\circ$. Finally the bus-4 voltage is given by

Solving we get $V_4^{(1)} = 0.9968 \angle -3.4849^\circ$

Procedure:

As can be seen from Table 4.3 that a total number of 4 real and 3 reactive powers are known to us. We must then calculate each of these from (4.6) and (4.7) using the values of the voltage magnitudes and their angle obtained after each iteration. The power mismatches are then calculated from (4.9) and (4.10). The process is assumed to have converged when each of $P_2, P_3, P_4, P_5, Q_2, Q_3$ and Q_4 is below a small pre-specified value. At this

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Sometimes to accelerate computation in the P-Q buses the voltages obtained from (4.12) is

$$V_{i,acc}^{(k)} = (1 - \lambda)V_{i,acc}^{(k-1)} + \lambda V_i^{(k)} = V_{i,acc}^{(k-1)} + \lambda \{V_i^{(k)} - V_{i,acc}^{(k-1)}\} \quad (4.21)$$

multiplied by a constant. The voltage update of bus- i is then given by

where λ is a constant that is known as the **acceleration factor**. The value of λ has to be below 2.0 for the convergence to occur. Table 4.4 lists the values of the bus voltages after the 1st iteration and number of iterations required for the algorithm to converge for different values of λ . It can be seen that the algorithm converges in the least number of iterations when λ is 1.4 and the maximum number of iterations are required when λ is 2. In fact the algorithm will start to diverge if larger values of acceleration factor are chosen. The system data after the convergence of the algorithm will be discussed later.

Table 4.4 Gauss-Seidel method: bus voltages after 1st iteration and number of iterations required for convergence for different values of λ .

}	Bus voltages (per unit) after 1 st iteration				No of iterations for convergence
	V ₂	V ₃	V ₄	V ₅	
1	0.9927 Z2.6	0.9883 Z2.83	0.9968 Z3.48	1.02 Z0.89	28
2	0.9874 Z5.22	0.9766 Z8.04	0.9918 Z14.02	1.02 Z4.39	860
1.8	0.9883 Z4.7	0.9785 Z6.8	0.9903 Z11.12	1.02 Z3.52	54
1.6	0.9893 Z4.17	0.9807 Z5.67	0.9909 Z8.65	1.02 Z2.74	24
1.4	0.9903 Z3.64	0.9831 Z4.62	0.9926 Z6.57	1.02 Z2.05	14
1.2	0.9915 Z3.11	0.9857 Z3.68	0.9947 Z4.87	1.02 Z1.43	19

Exercise :

Consider a 500kVA delta/star transformer 11/0.433 kV supplying a LT load of 300 kVA (consider 50% motor load, 50% lighting load). A utility (swing grid) supplying a 11 kV line direct to that 200 kVA transformer. At utility side 11kV connected with 3MVA 33/11 transformer at utility side and X/R ratio of is 70 deg. Run the load flow analysis using Etap.

Etap Diagrams:

Experiment No: 5 study on AC load flow using Newton-Raphson method (by simulation).

Aim: To study on AC load flow using Newton-Raphson method (by simulation).

Theory:

Apparatus Required:

The Newton-Raphson load flow program is tested on the system of Fig. 4.1 with the system data and initial conditions given in Tables 4.1 to 4.3. From (4.41) we can write

$$L_{23}^{(0)} = -|Y_{23}V_2^{(0)}V_3^{(0)}|\sin(\theta_{23} + \delta_3 - \delta_2) = -|Y_{23}|\sin\theta_{23} = -B_{23} = -4.8077$$

Similarly from (4.39) we have

$$\begin{aligned} Q_2^{(0)} &= -|V_2^{(0)}|^2 B_{22} - \sum_{k=2}^n |Y_{2k}V_2^{(0)}V_k^{(0)}|\sin(\theta_{2k} + \delta_k - \delta_2) \\ &= -B_{22} - 1.05B_{21} - B_{23} - B_{24} - 1.02B_{25} = -0.6327 \end{aligned}$$

Hence from (4.42) we get

$$L_{22}^{(0)} = -Q_2^{(0)} - |V_2^{(0)}|^2 B_{22} = -0.6327 - B_{22} = 18.8269$$

In a similar way the rest of the components of the matrix $J_{11}^{(0)}$ are calculated. This matrix is given by

$$J_{11}^{(0)} = \begin{bmatrix} 18.8269 & -4.8077 & 0 & -3.9231 \\ -4.8077 & 11.1058 & -3.8462 & -2.4519 \\ 0 & -3.8462 & 5.8077 & -1.9615 \\ -3.9231 & -2.4519 & -1.9615 & 12.4558 \end{bmatrix}$$

For forming the off diagonal elements of J_{21} we note from (4.44) that

$$M_{23}^{(0)} = -|Y_{23}V_2^{(0)}V_3^{(0)}|\cos(\theta_{23} + \delta_2 - \delta_3) = -G_{23} = 0.9615$$

Also from (4.38) the real power injected at bus-2 is calculated as

$$\begin{aligned} P_2^{(0)} &= |V_2^{(0)}|^2 G_{22} + \sum_{k=2}^n |Y_{2k}V_2^{(0)}V_k^{(0)}|\cos(\theta_{2k} + \delta_k - \delta_2) \\ &= G_{22} + 1.05G_{21} + G_{23} + G_{24} + 1.02G_{25} = -0.1115 \end{aligned}$$

Hence from (4.45) we have

$$M_{22} = P_2^{(0)} - |V_2^{(0)}|^2 G_{22} = -3.7654$$

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Similarly the rest of the elements of the matrix J_{21} are calculated. This matrix is then given as

$$J_{21}^{(0)} = \begin{bmatrix} -3.7654 & 0.9615 & 0 & 0.7846 \\ 0.9615 & -2.2212 & 0.7692 & 0.4904 \\ 0 & 0.7692 & -1.1615 & 0.3923 \end{bmatrix}$$

For calculating the off diagonal elements of the matrix J_{12} we note from (4.47) that they are negative of the off diagonal elements of J_{21} . However the size of J_{21} is (3 X 4) while the size of J_{12} is (4 X 3). Therefore to avoid this discrepancy we first compute a matrix M that is given by

$$M = \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ M_{41} & M_{42} & M_{43} & M_{44} \end{bmatrix}$$

The elements of the above matrix are computed in accordance with (4.44) and (4.45). We can then define

$$J_{21} = M(1:3, 1:4) \text{ and } J_{12} = -M(1:4, 1:3)$$

Furthermore the diagonal elements of J_{12} are overwritten in accordance with (4.48). This matrix is then given by

$$J_{12}^{(0)} = \begin{bmatrix} 3.5423 & -0.9615 & 0 \\ -0.9615 & 2.2019 & -0.7692 \\ 0 & -0.7692 & 1.1462 \\ 0.7846 & -0.4904 & -0.3923 \end{bmatrix}$$

Finally it can be noticed from (4.50) that $J_{22} = J_{11}(1:3, 1:3)$. However the diagonal elements of J_{22} are then overwritten in accordance with (4.51). This gives the following matrix

$$J_{22}^{(0)} = \begin{bmatrix} 17.5615 & -4.8077 & 0 \\ -4.8077 & 10.8996 & -3.8462 \\ 0 & -3.8462 & 5.5408 \end{bmatrix}$$

From the initial conditions the power and reactive power are computed as

$$P_{calc}^{(0)} = [-0.1115 \quad -0.0096 \quad -0.0077 \quad -0.0098]^T$$

$$Q_{calc}^{(0)} = [-0.6327 \quad -0.1031 \quad -0.1335]^T$$

Consequently the mismatches are found to be

$$\Delta P^{(0)} = [-0.8485 \quad -0.3404 \quad -0.1523 \quad 0.2302]^T$$

$$\Delta Q^{(0)} = [0.0127 \quad -0.0369 \quad 0.0535]^T$$

Then the updates at the end of the first iteration are given as

$$\begin{bmatrix} \delta_2^{(0)} \\ \delta_3^{(0)} \\ \delta_3^{(0)} \\ \delta_4^{(0)} \end{bmatrix} = \begin{bmatrix} -4.91 \\ -6.95 \\ -7.19 \\ -3.09 \end{bmatrix} \text{ deg} \quad \begin{bmatrix} |V_2|^{(0)} \\ |V_3|^{(0)} \\ |V_4|^{(0)} \end{bmatrix} = \begin{bmatrix} 0.9864 \\ 0.9817 \\ 0.9913 \end{bmatrix}$$

The load flow converges in 7 iterations when all the power and reactive power mismatches are below 10^{-6} .

Procedure:

The Newton-Raphson procedure is as follows:

Step-1: Choose the initial values of the voltage magnitudes $|V|^{(0)}$ of all nload buses and $n - 1$ angles $\delta^{(0)}$ of the voltages of all the buses except the slack bus.

Step-2: Use the estimated $|V|^{(0)}$ and $\delta^{(0)}$ to calculate a total $n - 1$ number of injected real power $P_{calc}(0)$ and equal number of real power mismatch $\Delta P^{(0)}$.

Step-3: Use the estimated $|V|^{(0)}$ and $\delta^{(0)}$ to calculate a total np number of injected reactive power $Q_{calc}(0)$ and equal number of reactive power mismatch $\Delta Q^{(0)}$.

Step-3: Use the estimated $|V|^{(0)}$ and $\delta^{(0)}$ to formulate the Jacobian matrix $J^{(0)}$.

Step-4: Solve (4.30) for $\delta^{(0)}$ and $|V|^{(0)} \div |V|^{(0)}$.

Step-5 : Obtain the updates from

$$\delta^{(1)} = \delta^{(0)} + \Delta \delta^{(0)}$$

$$|V|^{(1)} = |V|^{(0)} \left[1 + \frac{\Delta |V|^{(0)}}{|V|^{(0)}} \right]$$

Step-6: Check if all the mismatches are below a small number. Terminate the process if yes.

Otherwise go back to step-1 to start the next iteration with the updates given by (above two equation)

Exercise :

Consider a 200kVA delta/star transformer 11/0.433 kV supplying a LT load of 160 kVA (consider 50% motor load, 50% lighting load). A utility (swing grid) supplying a 11 kV line direct to that 200 kVA transformer. At utility side 11kV bus Having MVA_{sc} rating of 1000MVA_{sc} and X/R ratio of is 70 deg. Run the load flow analysis using Etap.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR
Lab Manual

Etap diagrams:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Power Electronics Lab

Course Code: EE693

L-T-P scheme: 0-0-3

Course Credit: 2

Objectives:

1. To learn and understand the characteristics of an SCR, Triac.
2. To provide an understanding of the design aspects of different triggering circuits of an SCR.
3. To provide an efficient understanding of performance of single phase half controlled symmetrical and asymmetrical bridge converter.
4. To provide an efficient understanding of performance of three phase controlled converter with R & R-L load.
5. To learn and understand the performance of a Cycloconverter, Dual converter.

Learning Outcomes: The students will have a detailed knowledge of the concepts of characteristics of an SCR, Triac devices. Aware of a variety of approaches to triggering circuits of an SCR. Student will learn the basics behind performance of single phase half controlled symmetrical and asymmetrical bridge converter, Dual converter. Upon the completion of Power Electronics practical course, the student will be able to:

-) **Understand** and implement characteristics of an various power electronic switches like SCR, Triac.
-) **Use** modern tools and circuit to understand the performance of Dual converter (Used for crane drive, conveyor belt drive).
-) **Understand** the concepts of single phase half controlled symmetrical and asymmetrical bridge converter
-) **Analyze** and simulate three phase AC controller with R and R-L load (RL load like induction motor load and its speed variations)
-) **Implement** a Cycloconverter and study the performance used for VVVF drive for induction motor.
-) **Simulate** different triggering circuits of an SCR and **Implement** the triggering circuits for SCR, Power Mosfet, IGBT .
-) **Understand** the performance of step down chopper with R and R-L load (chopper used for HVDC transmission, renewable energy like solar and wind energy system) .
-)

Course Contents:

Exercises that must be done in this course are listed below:

Exercise No.1: Study the characteristics of an SCR.

Exercise No.2: Study the characteristics of a Triac

Exercise No. 3: Study of performance of single phase controlled converter with and without source inductance (simulation)

Exercise No. 4: Study of performance of single phase half controlled symmetrical and asymmetrical bridge converter.(simulation)

Exercise No. 5: Study of performance of three phase controlled converter with R & R-L load.(simulation)

Exercise No. 6: Study of performance of single phase AC voltage controller with R and R-L load (simulation)

Exercise No. 7: Study of performance of a Dual converter.

Exercise No. 8: Study of performance of step down chopper with R and R-L load.

Text Book:

1. Fundamental of Power Electronics with MATLAB, Randall Shaffer, Cengage Learning.
2. SPICE for Power electronics and electric power, M.H. Rashid & H.M. Rashid, Taylor & Francis.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

5. Modeling & Simulation using MATLAB-SIMILINK , S. Jain, Wiley India
6. MATLAB & SIMULINK for Engineers, A.K. Tyagi, Oxford University Press.

Recommended Systems/Software Requirements:

1. Intel based desktop PC with minimum of 166 MHZ or faster processor with at least 64 MB RAM and 100 MB free disk space.
2. Matlab 32 bit or 64 bit 2009a/2009b.
3. Testing kit for TRIAC,SCR,DUAL converter, Chopper, Inverter and converter circuit, Microcontroller train kit(PV59RD2V51),uc8051,128kb memory.
4. Chip burning kit.
5. Spare equipments components like MOSFET(IRF540,MCTE2E,IN407,2K,10k,50uf 230V).

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Experiment No: 1 V-I characteristics of an SCR.

Aim: To perform an experiment to study the V-I characteristics of SCR and determine the break over voltage, on state resistance holding current and latching current.

Apparatus required: SCR, Power supplies, Wattage resistors, Ammeter, Voltmeter etc.

Theory:

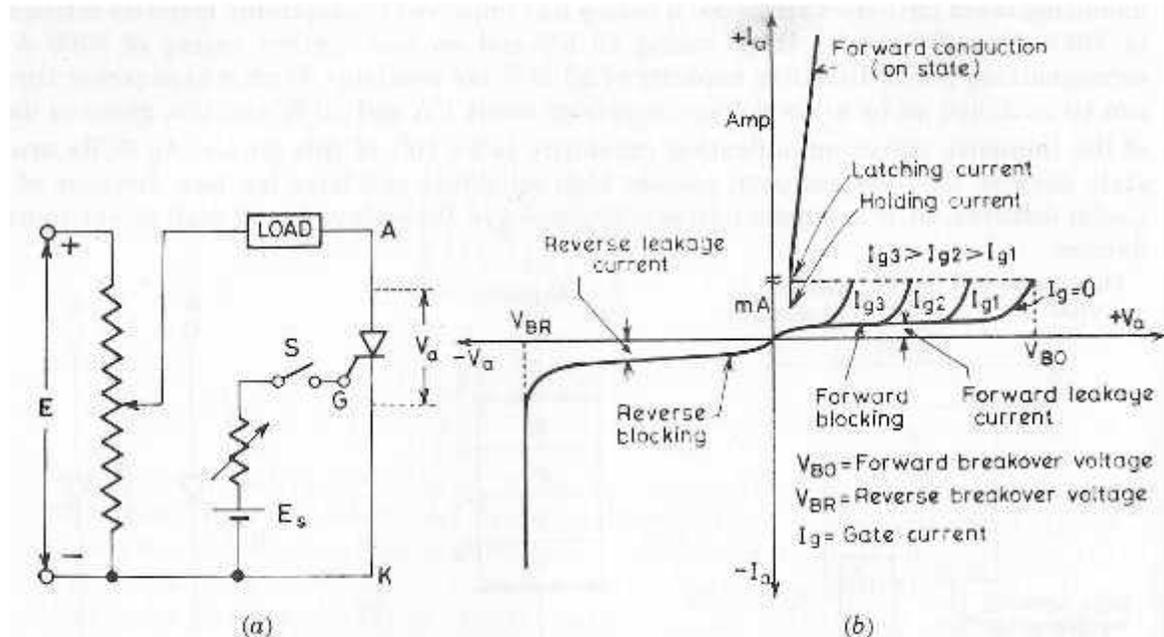


Fig 1 (a) Elementary circuit for obtaining thyristor V-I characteristics
(b) Static V-I characteristics of a thyristor.

An elementary circuit diagram for obtaining static V-I characteristics of a thyristor is shown in Fig. 1 (a). The anode and cathode are connected to main source through the load. The gate and cathode are fed from a source E_s which provides positive gate current from gate to cathode.

Fig. 1 (b) shows static V-I characteristics of a thyristor. Here V_a is the anode voltage across thyristor terminals A, K and I_a is the anode current. Typical SCR V-I characteristic shown in Fig. 1 (b) reveals that a thyristor has three basic modes of operation; namely, reverse blocking mode, forward blocking (off-state) mode and forward conduction (on- state) mode. These three modes of operation are now discussed below:

Reverse Blocking Mode: When cathode is made positive with respect to anode with switch S open, Fig. 1 (a), thyristor is reverse biased as shown in Fig. 1 (a). Junctions J_1 J_3 are seen to be reverse biased whereas junction J_2 is forward biased. The device behaves as if two diodes are connected in series with reverse voltage applied across them. A small leakage current of the order of a few milliamperes (or a few microamperes depending upon the SCR rating) flows. This is reverse blocking mode, called the off-state, of the thyristor. If the reverse voltage is increased, then at a critical breakdown level, called reverse break- down voltage V_{BR} , an avalanche occurs at J_1 and J_3 and the reverse current increases rap- idly. A large current associated with V_{BR} gives

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

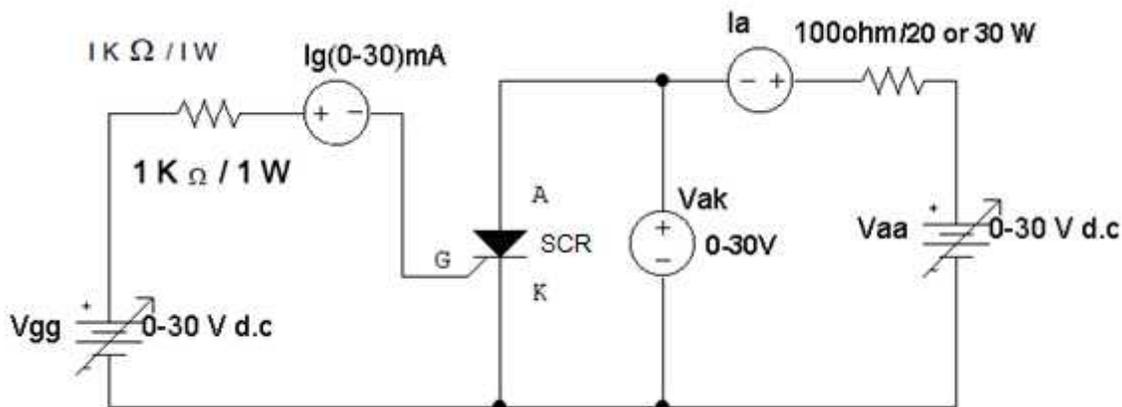
exceed its permissible temperature rise. It should, therefore, be ensured that maximum working reverse voltage across a thyristor does not exceed V_{BR} . When reverse voltage applied across a thyristor is less than V_{BR} , the device offers a high impedance in the reverse direction. The SCR in the reverse blocking mode may therefore be treated as an open switch.

Note that V-I characteristic after avalanche breakdown during reverse blocking mode is applicable only when load resistance is zero, Fig. 1 (b). In case load resistance is present, a large anode current associated with avalanche breakdown at V_{BR} would cause substantial voltage drop across load and as a result, V-I characteristic in third quadrant would bend to the right of vertical line drawn at V_{BR} .

Forward Blocking Mode: When anode is positive with respect to the cathode, with gate circuit open, thyristor is said to be forward biased as shown in Fig. 1 (b). It is seen from this figure that junctions J1, J3 are forward biased but junction J2 is reverse biased. In this mode, a small current, called forward leakage current, flows as shown in Figs. 4.2 (b) and Fig 1 (b). In case the forward voltage is increased, then the reverse biased junction J2 will have an avalanche breakdown at a voltage called forward break over voltage V_{BO} . When forward voltage is less than V_{BO} , SCR offers high impedance. Therefore, a thyristor can be treated as an open switch even in the forward blocking mode.

Forward Conduction Mode: In this mode, thyristor conducts currents from anode to cathode with a very small voltage drop across it. A thyristor is brought from forward blocking mode to forward conduction mode by turning it on by exceeding the forward breakover voltage or by applying a gate pulse between gate and cathode. In this mode, thyristor is in on-state and behaves like a closed switch. Voltage drop across thyristor in the on state is of the order of 1 to 2 V depending on the rating of SCR. It may be seen from Fig. 1 (b) that this voltage drop increases slightly with an increase in anode current. In conduction mode, anode current is limited by load impedance alone as voltage drop across SCR is quite small. This small voltage drop v_T across the device is due to ohmic drop in the four layers.

Circuit Diagram:



UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

Designing Equations:-

Let $I_a = 300\text{mA}$
 $V_{scr} = 1\text{v}$
 $V_{aa} = 30\text{v}$

Wkt,

$$V_{aa} = I_a R_L + V_{scr}$$

$$R_L = \frac{V_{aa} - V_{scr}}{I_a}$$

$$R_L = \frac{30 - 1}{300 \times 10^{-3}} = 96.66\Omega \approx 100\Omega$$

Wattage:-

Power in watts = $I^2 R_L = (300 \times 10^{-3})^2 \times 100 = 9 \text{ watts}$ (select 20 watts)
 \therefore Load resistor = $R_L = 100\Omega$, 20watts

Gate Resistance (Rg):-

Let $I_g = 15\text{mA}$
 $V_{gt} = 1\text{v}$
 $V_{gg} = 15\text{v}$

Wkt,

$$V_{gg} = I_g R_g + V_{gt}$$

$$R_g = \frac{V_{gg} - V_{gt}}{I_g}$$

$$R_g = \frac{15 - 1}{15 \times 10^{-3}} = 933\Omega \approx 1\text{K}\Omega$$

Wattage:-

Power in watts = $I^2 R_g = (15 \times 10^{-3})^2 \times 10^3 = 0.225\text{w}$ (select 1 watt)
 \therefore Gate resistor = $R_g = 1\text{K}\Omega$, 1 watts

Procedure:

1. Connections are made as shown in the circuit diagram.
2. The value of gate current I_G , is set to convenient value by adjusting V_{GG} .
3. By varying the anode cathode voltage V_{AA} gradually in step by step, note down the corresponding values of V_{AK} and I_A . Note down V_{AK} and I_A at the instant of firing of SCR. and after firing (by reducing the voltmeter ranges and ammeter ranges) then increase the supply voltage V_{AA} . Note down corresponding values of V_{AK} and I_A .
4. The point at which SCR fires, gives the value of break over voltage V_{BO} .
5. A graph of V_{AK} V/S I_A is to be plotted.
6. The on state resistance can be calculated from the graph by using a formula.
7. The gate supply voltage V_{GG} is to be switched off.
8. Observe the ammeter reading by reducing the anode cathode supply voltage V_{AA} . The point at which ammeter reading suddenly goes to zero gives the value of holding current I_H
9. Steps no.2, 3, 4, 5, 6, 7, 8 are repeated for another value of gate current I_G .

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR
Lab Manual

Observations:

For forward bias For reverse bias

S.No.	Voltage (V)	Current (mA)	S.No.	Voltage (V)	Current (μ A)

Result and Discussion:

The V-I characteristics of silicon controlled rectifier is plotted on the graph which is true according to theory.

Precaution:

1. Keep your hand away from main supply.
2. Do not switch on the power supply unless you have checked the circuit connections as per the circuit diagram.

Post Experiment Question:

- Q1. What is SCR?
Q2. What are various turn on method for thyristor?

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

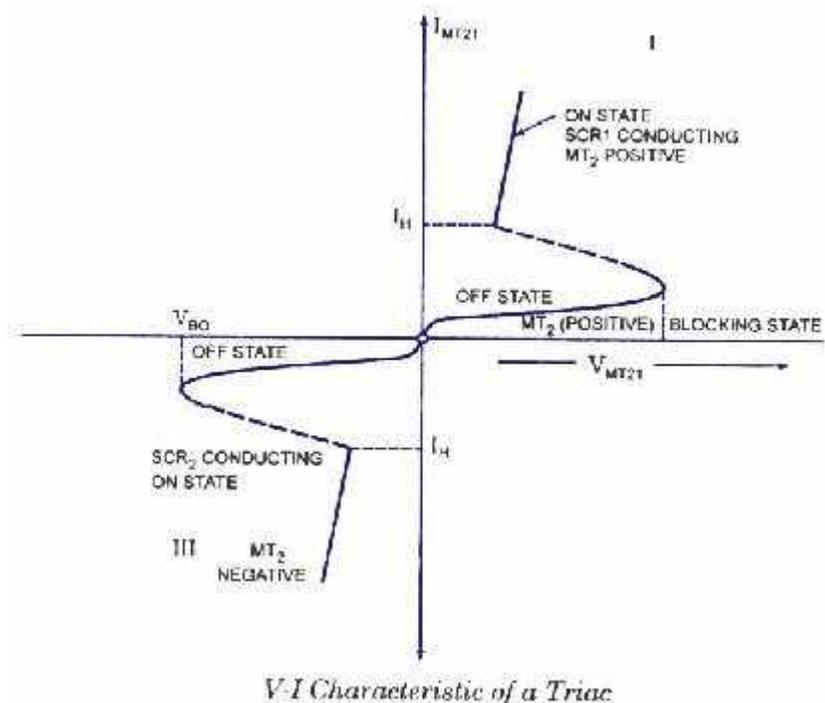
Lab Manual

Experiment No: 2V-I characteristics of a Triac.

Aim: To perform an experiment to study the V-I characteristics of Triac and determine the break over voltage, on state resistance holding current and latching current.

Apparatus required: Triac, Power supplies, Wattage resistors, Ammeter, Voltmeter etc.

Theory:



Typical V-I characteristics of a triac are shown in figure. The triac has on and off state characteristics similar to SCR but now the characteristic is applicable to both positive and negative voltages. This is expected because triac consists of two SCRs connected in parallel but opposite in directions.

MT_2 is positive with respect to MTX in the first quadrant and it is negative in the third quadrant. As already said in previous blog posts, the gate triggering may occur in any of the following four modes.

Quadrant I operation : V_{MT2} , positive; V_{G1} positive

Quadrant II operation : V_{MT21} positive; V_{G1} negative

Quadrant III operation : V_{MT21} negative; V_{G1} negative

Quadrant IV operation : V_{MT21} negative; V_{G1} positive

Where V_{MT21} and V_{G1} are the voltages of terminal MT_2 and gate with respect to terminal

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

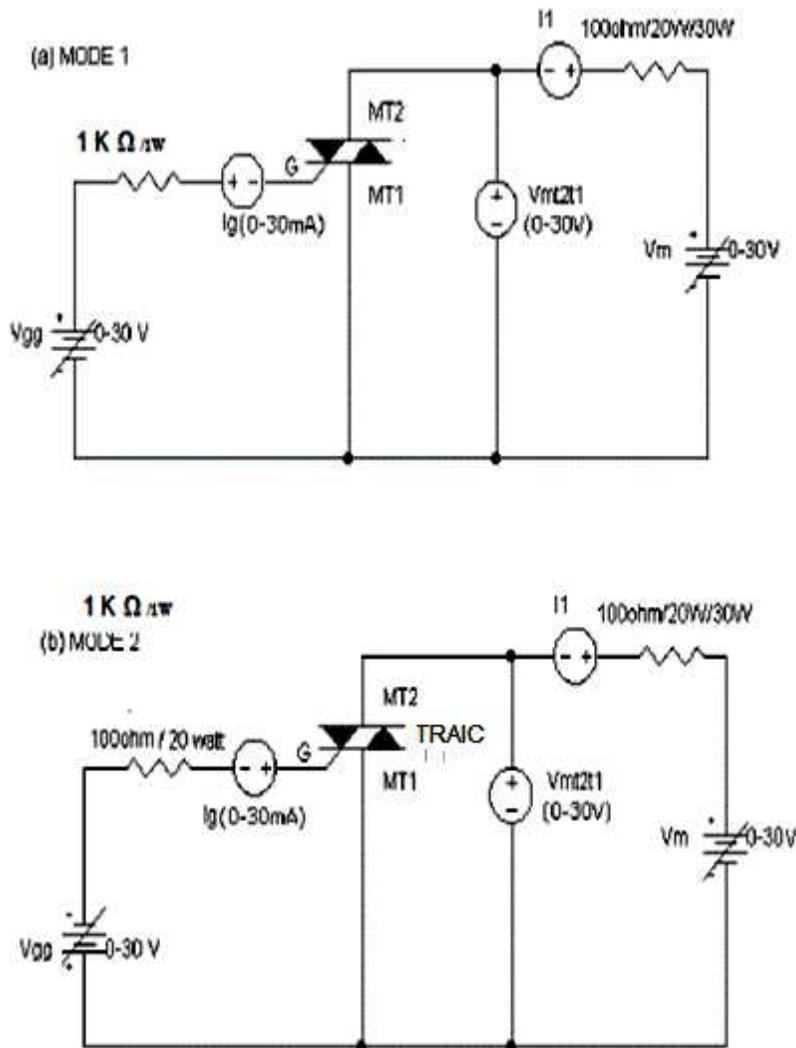
Lab Manual

through it. This large inrush of current must be restricted by employing external resistance, otherwise the device may get damaged. The gate is the control terminal of the device. By applying proper signal to the gate, the firing angle of the device can be controlled. The circuits used in the gate for triggering the device are called the gate-triggering circuits. The gate-triggering circuits for the triac are almost same like those used for SCRs. These triggering circuits usually generate trigger pulses for firing the device. The trigger pulse should be of sufficient magnitude and duration so that firing of the device is assured. Usually, a duration of 35 μ s is sufficient for sustaining the firing of the device.

A typical triac has the following voltage/current values:

- | Instantaneous on-state voltage – 1.5 Volts
- | On-state current – 25 Amperes
- | Holding current, I_H - 75 Milli Amperes
- | Average triggering current, I_G – 5 Milli Amperes

Circuit Diagram:



UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR
Lab Manual

Observations:

For forward bias For reverse bias

S.No.	Voltage (V)	Current (mA)	S.No.	Voltage (V)	Current (μ A)

Result and Discussion:

The V-I characteristics of TRIAC is plotted on the graph which is true according to theory

Precaution:

1. Keep your hand away from main supply.
2. Do not switch on the power supply unless you have checked the circuit connections as per the circuit diagram.

Post Experiment Question:

- Q1. What is TRIAC?
- Q2. What do you mean by gate threshold current?
- Q3. What are the applications of TRIAC?

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Experiment No: 3 Observe the output waveforms single phase half controlled converter using R and RL load.

Aim: To study the operation of single phase half controlled converter using R and RL load and to observe the output waveforms

Apparatus required:

1. Power thyristors ,Rheostat ,CRO ,Transformer (1-phase) 230V/24V , Connection wires.
2. Matlab 2009a or2009b (for simulation).
3. PSIM module (for simulation).
4. 8051 Trainer kit, driver circuit module.

For Hardware experiment purpose:

S. no.	Name of equipment's/components	Specification	Range/Rating	Qty.
1.				
2.				
3.				
4.				
5.				

Theory:

A semi converter uses two diodes and two thyristors and there is a limited control over the level of dc output voltage. A semi converter is one quadrant converter. A one-quadrant converter has same polarity of dc output voltage and current at its output terminals and it is always positive. It is also known as two- pulse converter. Figure shows half controlled rectifier with R load. This circuit consists of two SCRs T1 and T2, two diodes D1 and D2. During the positive half cycle of the ac supply, SCR T1 and diode D2 are forward biased when the SCR T1 is triggered at a firing angle $t = \alpha$, the SCR T1 and diode D2 comes to the on state. Now the load current flows through the path L - T1- R load –D2 - N. During this period, we output voltage and current are positive. At $t = \pi$, the load voltage and load current reaches to zero, then SCR T1 and diode D2 comes to off state since supply voltage has been reversed. During the negative half cycle of the ac supply, SCR T2 and diode D1 are forward biased. When SCR T2 is triggered at a firing angle $t = \pi + \alpha$, the SCR T2 and diode D1 comes to on state. Now the load current flows through the path N - T2- R load – D1 -L. During this period, output voltage and output current will be positive. At $t = 2\pi$, the load voltage and load current reaches to zero then SCR T2 and diode D1 comes to off state since the voltage has been reversed. During the period ($\pi + \alpha$ to 2π) SCR T2 and diode D1 are conducting.

$$V_{out} = (\sqrt{2}V_s)(1 + \cos\alpha) / \pi$$

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Circuit Diagram

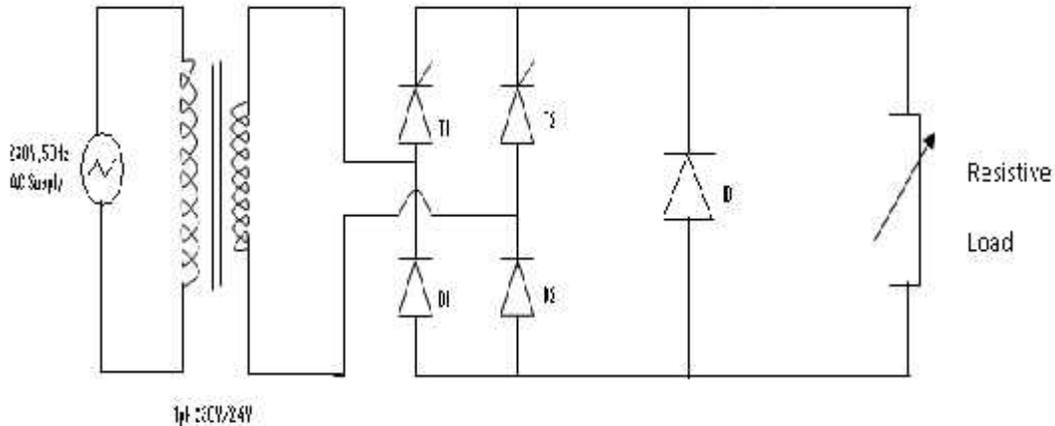
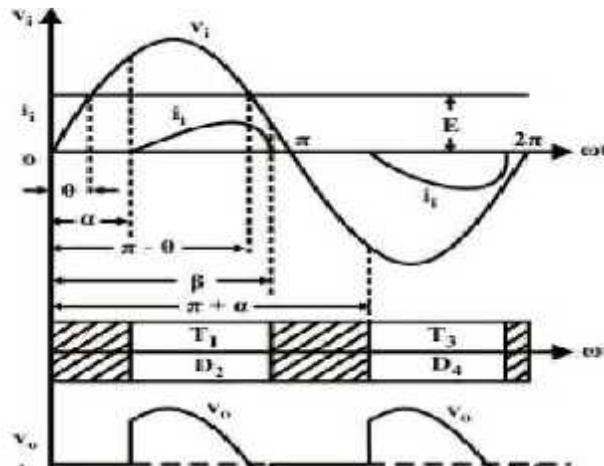


Fig: Single Phase Half Controlled Bridge Rectifier

Model Graph:



Result and Discussions:

Post Experiment Question:

1. What is conduction angle?
2. What are the effects of adding freewheeling diode in this circuit?
3. What are the effects of removing the freewheeling diode in single phase semi converter?
4. Why is the power factor of semi converters better than that of full converters?
5. What is the inversion mode of converters?

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

MatlabOutput waveforms:

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Experiment No: 4 Observe the output waveforms of single phase fully controlled converter using R and RL load.

Aim: To study the operation of single phase fully controlled converter using R and RL load and to observe the output waveforms.

Apparatus required:

1. Power thyristors ,Rheostat ,CRO ,Transformer (1-phase) 230V/24V , Connection wires.
2. Matlab 2009a or 2009b (for simulation).
3. PSIM module (for simulation).
4. 8051 Trainer kit, driver circuit module.

For Hardware experiment purpose:

S. no.	Name of equipment's/components	Specification	Range/Rating	Qty.
1.				
2.				
3.				
4.				
5.				

Theory:

A fully controlled converter or full converter uses thyristors only and there is a wider control over the level of dc output voltage. With pure resistive load, it is a single quadrant converter. Here, both the output voltage and output current are positive. With RL-load it becomes a two-quadrant converter. Here, output voltage is either positive or negative but output current is always positive. Figure shows the quadrant operation of fully controlled bridge rectifier with R-load. Fig shows single phase fully controlled rectifier with resistive load. This type of full wave rectifier circuit consists of four SCRs. During the positive half cycle, SCRs T1 and T2 are forward biased. At $t = \alpha$, SCRs T1 and T3 are triggered, then the current flows through the L-T1-Rload-T3-N. At $t = \pi$, supply voltage falls to zero and the current also goes to zero.

Hence SCRs T1 and T3 are turned off. During negative half cycle (to 2π), SCRs T3 and T4 are forward biased. At $t = \pi + \alpha$, SCRs T2 and T4 are triggered, then the current flows through the path N-T2-Rload-T4-L. At $t = 2\pi$, supply voltage and current go to zero, SCRs T2 and T4 are turned off. The Fig-3, shows the current and voltage waveforms for this circuit. For large power dc loads, 3-

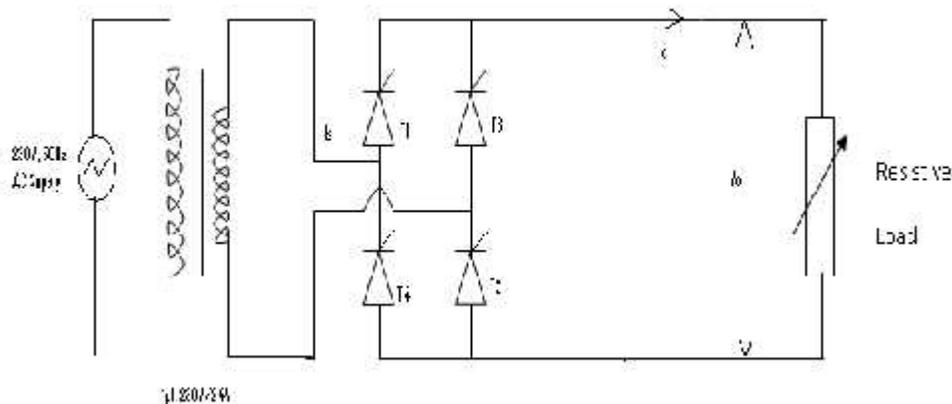
UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

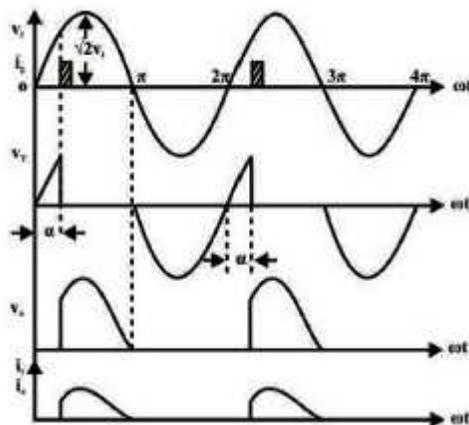
converters are 3-phase half-wave converter, 3-phase semiconverter, 3-phase fullcontrolled and 3-phase dual converter. Three-phase half-wave converter is rarely used in industry because it introduces dc component in the supply current. Semiconverters and full converters are quite common in industrial applications. A dual is used only when reversible dc drives with power ratings of several MW are required. The advantages of three-phase converters over single-phase converters are as under: In 3-phase converters, the ripple frequency of the converter output voltage is higher than in single-phase converter. Consequently, the filtering requirements for smoothing out the load current are less. The load current is mostly continuous in 3-phase converters. The load performance, when 3-phase converters are used, is therefore superior as compared to when single-phase converters are used.

$$V_{out} = (2V_s)(\cos \alpha) / \quad I_{avg} = V_{avg} / R$$

Circuit Diagram:



Model Graph:



Observation Table:

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR
Lab Manual

S.No.	FiringAngle(α)	Output Voltage(Volts)	Time period(ms)
1			
2			
3			

Result and Discussions:

Post Experiment Question:

1. If firing angle is greater than 90 degrees, the inverter circuit formed is called as?
2. What is displacement factor?
3. What is Dc output voltage of single phase full wave controller?
4. What are the effects of source inductance on the output voltage of a rectifier?
5. What is commutation angle of a rectifier?
6. What are the advantages of three phase rectifier over a single phase rectifier?

Matlab Output Waveforms:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Experiment No: 5 Observe the output of three Phase Fully Controlled Bridge Converter.

Aim: To study the operation of three Phase Fully Controlled Bridge Converter using R and RL load and to observe the output waveforms.

Apparatus required:

5. Power thyristors ,Rheostat ,CRO ,Transformer (1-phase) 230V/24V , Connection wires.
6. Matlab 2009a or2009b (for simulation).
7. PSIM module (for simulation).
8. 8051 Trainer kit, driver circuit module.

For Hardware experiment purpose:

S. no.	Name of equipment's/components	Specification	Range/Rating	Qty.
1.				
2.				
3.				
4.				
5.				

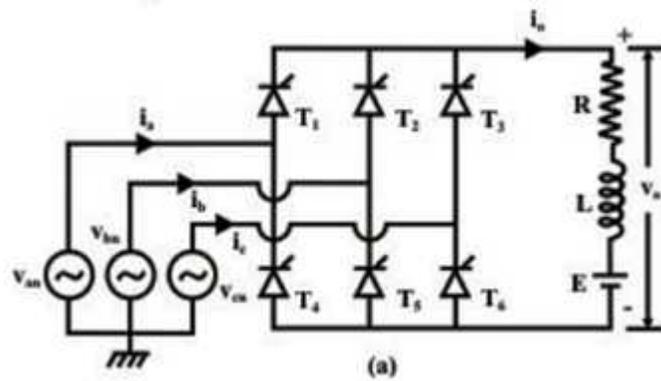
Theory & Formulae used:

The figure shows the power circuit of three phase fully controlled converter. The load is fed via a three phase half wave connection to one of the three supply lines, no neutral being required. Hence transformer connection is optional. However, for isolation of output from supply source, or for higher output requirement, the transformer is to be connected. If transformer is used, then one winding is connected in delta because the delta connection gives the circulating path for third harmonic current. This circuit consists of two groups of SCRs, positive group and negative group. Here, Scr T1, T2, T3 forms a positive group. Whereas SCR T4, T5, T6 forms a negative group. The positive group SCRs are turned on when the supply voltages are positive and negative group SCRs are turned on when the supply voltage are negative.

Continuous Conduction Mode:

The firing angle of each phasor is varied through a range of 60° to 180° . The minimum firing angle is 60° and is taken as $\alpha=0$, at 60° . When the phasor R-Y is allowed to conduct at α between zero, it continues to conduct by 60° when the phasor R-B is fired. The condition is shifted from SCR T5 to SCR T6. T5 is commutated off by reverse voltage of phase Y and Y across it. The Phasor R-B conducts after another 60° after which it is replaced by phasor (Y-B) when phase Y voltage assumes greater value than B or R. Hence load current is continuous for between 0 to The waveforms are shown in figure.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR
Lab Manual



Device / Mode	V_{T1}	V_{T2}	V_{T3}	V_{T4}	V_{T5}	V_{T6}	V_o
$T_1 T_2$	0	0	v_{ba}	v_{ca}	v_{ca}	v_{cb}	v_{ac}
$T_2 T_3$	v_{ab}	0	0	v_{ca}	v_{cb}	v_{cb}	v_{bc}
$T_3 T_4$	v_{ab}	v_{ac}	0	0	v_{cb}	v_{ab}	v_{ba}
$T_4 T_5$	v_{ac}	v_{ac}	v_{bc}	0	0	v_{ab}	v_{ca}
$T_5 T_6$	v_{ac}	v_{bc}	v_{bc}	v_{ba}	0	0	v_{cb}
$T_6 T_1$	0	v_{bc}	v_{ba}	v_{ba}	v_{ca}	0	v_{ab}
NONE	-	-	-	-	-	-	E

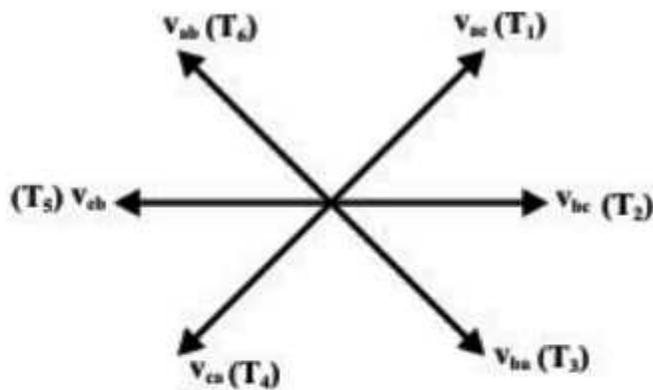
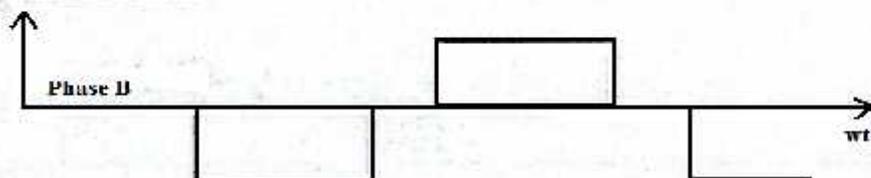
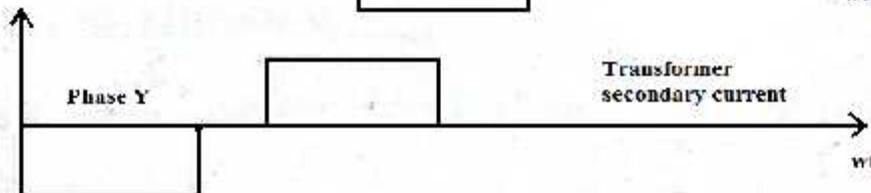
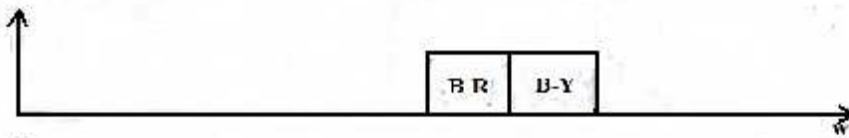
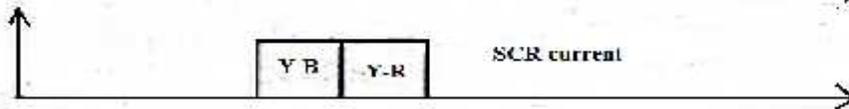
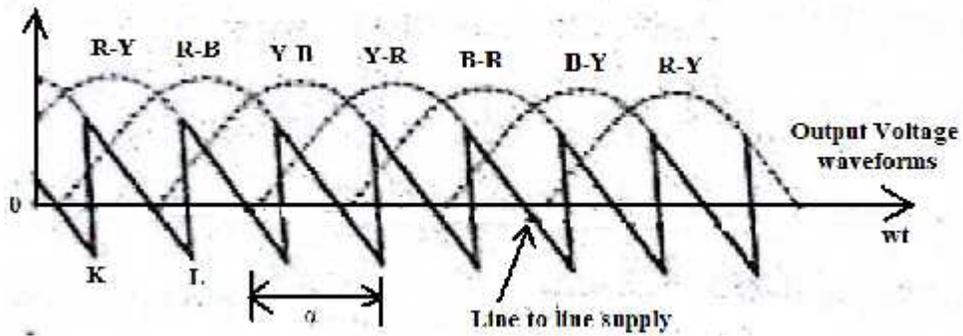
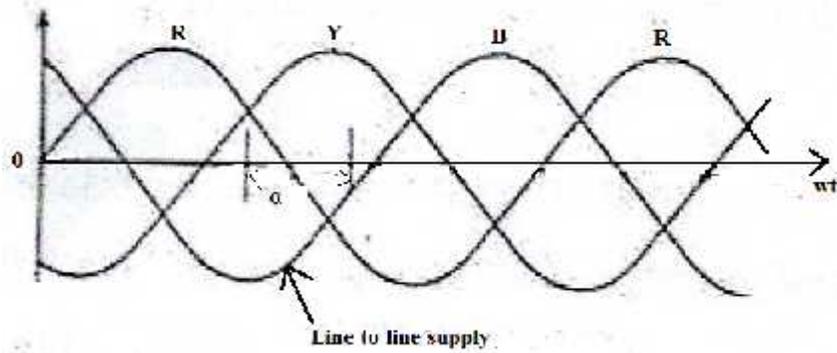


Fig. : operation of a three phase full controlled bridge converter
 (a) circuit diagram,
 (b) conduction table,
 (c) phaser diagram of line voltages.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual



UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR
Lab Manual

Observation Table:

S.No.	FiringAngle(α)	Output Voltage(Volts)	Time period(ms)
1			
2			
3			

Result and Discussions:

Post Experiment Question:

1. If firing angle is greater than 90 degrees, the inverter circuit formed is called as?
2. What is displacement factor?
3. What is Dc output voltage of three phase full wave controller?
4. What are the effects of source inductance on the output voltage of a rectifier?
5. What is commutation angle of a rectifier?

Matlab Output waveforms:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Experiment No: 6 Observe the output 1-phase ac voltage control using triac.

Aim: To study the operation of single Phase ac voltage control using triac to observe the output waveforms.

Apparatus required:

1. Traic ,Rheostat ,CRO ,Transformer (1-phase) 230V/24V , Connection wires.
2. Matlab 2009a or2009b (for simulation).
3. PSIM module (for simulation).
4. 8051 Trainer kit, driver circuit module.
5. Lamp – 60W, Resistor - 100W / 1W, Potentio meter – 100KW, Capacitor – 0.1mF / 400V
Resistor – 1KW, DIAC – DB3 , TRIAC BT 136,Unearthed oscilloscope.

For Hardware experiment purpose:

S. no.	Name of equipment's/components	Specification	Range/Rating	Qty.
1.				
2.				
3.				
4.				
5.				

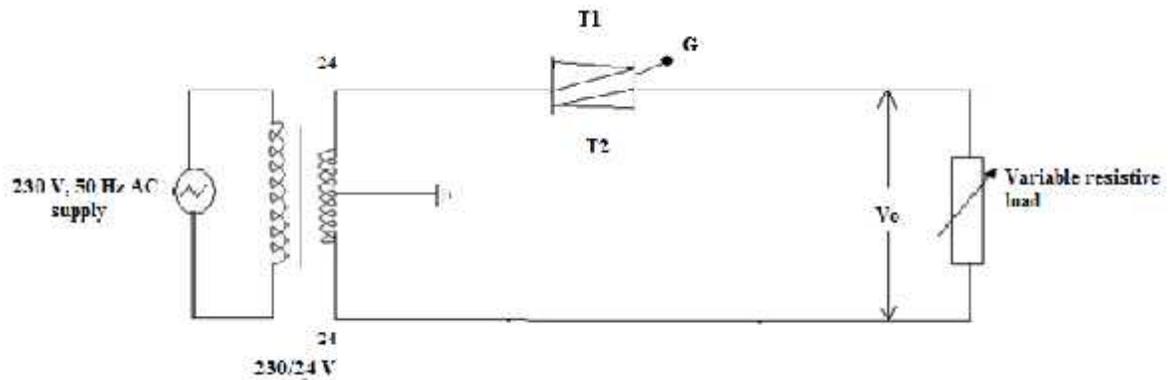
Theory:

Triac is a bidirectional thyristor with three terminals. Triac is the word derived by combining the capital letters from the words TRIode and AC. In operation triac is equivalent to two SCRs connected in anti- parallel. It is used extensively for the control of power in ac circuit as it can conduct in both the direction. Its three terminals are MT1 (main terminal 1), MT2 (main terminal 2) and G (gate).

Circuit Diagram:

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual



Circuit Operation:

1. When potentiometer is in minimum position drop across potentiometer is zero and hence maximum voltage is available across capacitor. This V_c shorts the diac ($V_c > V_{bo}$) and triggers the triac turning triac to ON – state there lamp glows with maximum intensity.
2. When the potentiometer is in maximum position voltage drop across potentiometer is maximum. Hence minimum voltage is available across capacitor ($V_c \ll V_{bo}$) hence triac to is not triggered hence lamp doesnot glow.
3. When potentiometer is in medium position a small voltage is available across capacitor hence lamp glows with minimum intensity.

Model waveforms:

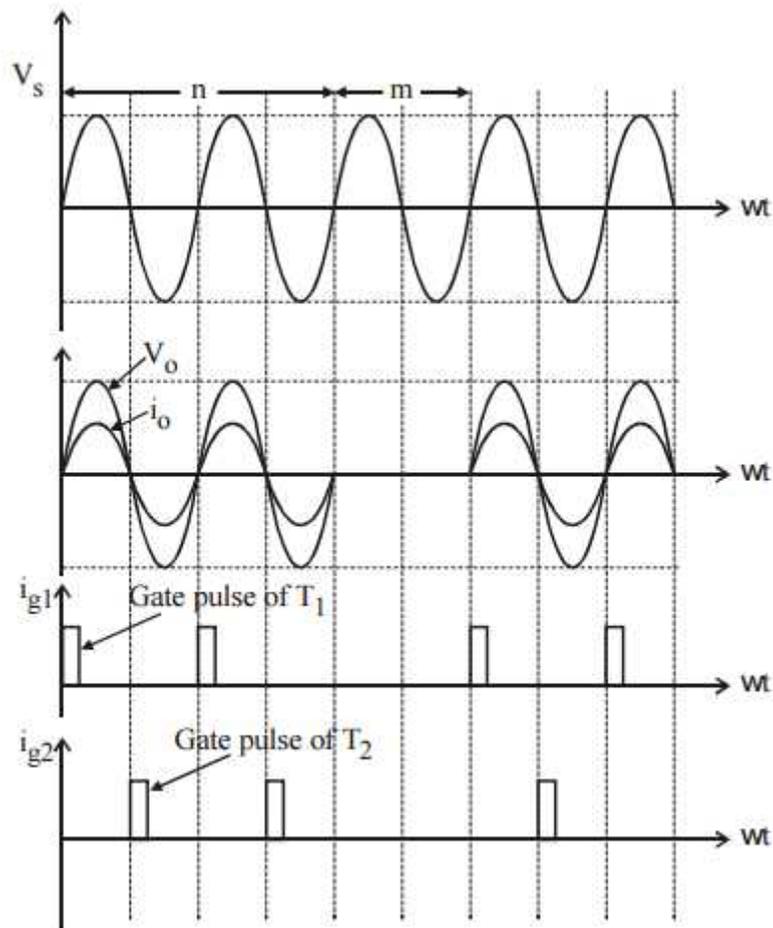


Fig.: Waveforms

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

S.No.	Firing Angle(α)	Output Voltage(Volts)	Time period(ms)
1			
2			
3			

Result and Discussions:

Post Experiment Question:

1. What are the performance parameters of ac voltage controllers?
2. What type of commutation is used in this circuit?
3. What are the effects of load inductance on the performance of AC voltage controllers?
4. What is extinction angle?
5. What are the disadvantages of unidirectional controllers?
6. What are the advantages of ON-OFF control?

Matlab waveforms:

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Experiment No: 7 Observe the output 1-phase ac voltage control using triac.

Aim: To study the operation of single Phase ac voltage control using triac to observe the output waveforms.

Apparatus required:

1. SCR, Rheostat, CRO, Transformer (1-phase) 230V/24V, Connection wires.
2. Matlab 2009a or 2009b (for simulation).
3. PSIM module (for simulation).
4. 8051 Trainer kit, driver circuit module.

For Hardware experiment purpose:

S. no.	Name of equipment's/components	Specification	Range/Rating	Qty.
1.				
2.				
3.				
4.				
5.				

Theory:

Dual Converter is an Electronic Device or Circuit made by the combination of two bridges. One of them works as Rectifier (Converts A.C. to D.C.) and other bridge works as Inverter (converts D.C. into A.C.). Thus an electronic circuit or device in which two processes take place at same time, is known as Dual Converter. A dual converter may be single phase or 3 phase device. The simple diagram given above is of single phase dual converter. The difference between single phase and three phase dual converter is just that in Three phase we use three phase rectifier at first stage, while in single phase dual converter we make use of single phase rectifier circuit at first bridge.

Single Phase Dual Converter:

As explained above that in single phase dual converter we use single phase rectifier circuit for converting single phase A.C. into steady D.C. Bridge No. 1 consists of Rectifier. Then the rectified D.C. is fed to a filter which removes pulses from rectified D.C. and converts it to a pure D.C. by filtering. After that, this pure D.C. is fed to load and from load it is given to inverter circuit which converts this D.C. to A.C. and finally this A.C. of inverter is taken as output.

Three Phase Dual Converter:

In three phase dual converter, we make use of three phase rectifier which converts 3 phase A.C. supply to D.C. The rest of the process is same and same elements are used. The output of three phase rectifier is fed to filter and after filtering the pure D.C. is fed to load. At last the supply from load is given to last bridge that is inverter. It does the invert process of rectifier and converts D.C. into 3 phase A.C. which appears at output.

Circuit Operation:

The fully controlled converter can produce a reversible direct output voltage with output current in one direction, and in terms of a conventional voltage/ current diagram (Fig.1:) is said to be capable of operation in two quadrants, the first and fourth. Such a range of operation is useful for certain purposes, examples being the control of a dc torque motor, i.e. a motor used to provide unidirectional torque with reversible rotation (Fig.2), and a DC transmission link between two AC systems in which power can be transmitted in either direction according to the polarity of the voltage with current flows always in one direction. Equally a converter may be used under steady-state conditions in the first quadrant only but transiently in the second quadrant in order to extract

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

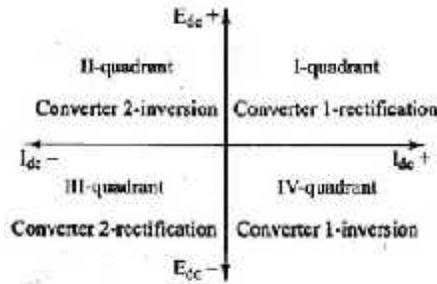


Fig.1: Voltage-Current diagram

If four-quadrant operation of a DC motor is required, i.e. reversible rotation and reversible torque, a single converter needs the addition of either a change-over contractor to reverse the armature connections or a means of reversing the field current in order to change the relationship between the converter voltage and the direction of rotation of the motor. Both of these are practicable in suitable phase and three-phase dual-converter using bridge type converter circuits. The output terminals of each converter having the same potential are connected together through a reactor. The four possible quadrants of converter operation thus resulting can be translated into four (steady-state) combinations of motor-torque and rotation as shown in Fig. 3

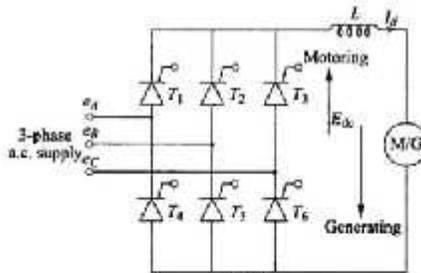


Fig.2

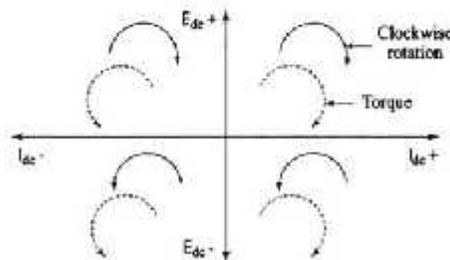


Fig.3: Torque-Rotation Diagram

Principle of Dual Converter

The basic principle of operation of dual converter can be explained with reference to the simplified equivalent diagram of the DC circuit shown in Fig.4. In this simplified representation, assumption is made that the dual converters are ideal and they produce pure DC output terminals. As shown in fig.4, each two-quadrant converter is assumed to be a controllable direct voltage source, connected in series with a diode. Diode D1 and D2 represent the unidirectional current flow characteristics of the converters. The current in load circuit can, however, flow in either direction.

Lab Manual

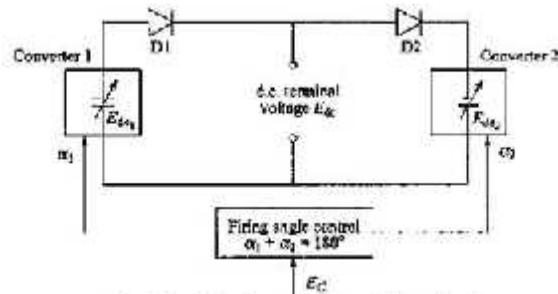
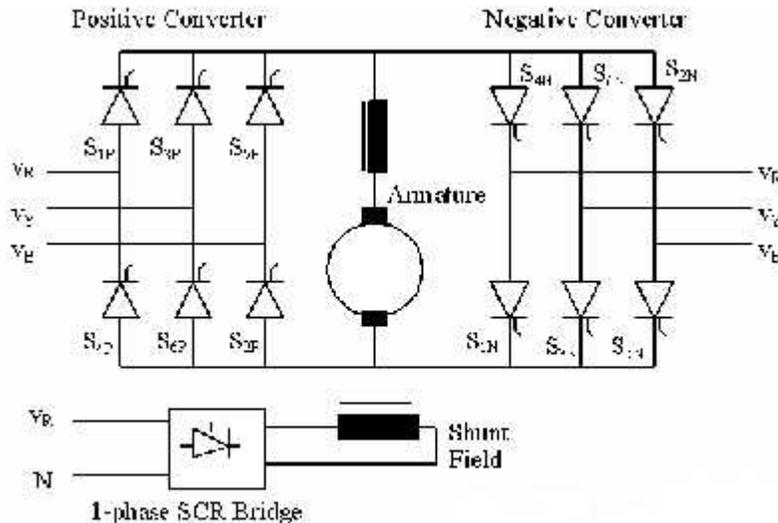


Fig.4: Ideal dual converter equivalent circuit

Dual Converter without circulating current mode

In a dual converter without circulating current operating mode, the flow of circulating current is completely inhibited through automatic control of the firing pulses, so that only that converter which carries the load current is in conduction and the other converter is temporarily blocked. Since only one converter operates at a time and the other is in blocking state, no reactor is required between the converters. At a particular instant, suppose converter 1 (fig.5 :) is operating as a rectifier and is supplying the load current while pulses to second converter are blocked. For the inversion operation, converter 1 is first blocked by removing its firing pulses and load current is reduced to zero. Converter 2 is made to conduct by applying the firing pulses to it. The current in converter 2 would now build up through the load in the reverse direction. So long as converter 2 is in operation, converter 1 is in the blocking state since the firing pulses are withdrawn from it. The pulses to converter 2 are applied after a delay time (current-free safety interval) of 10 to 20ms. This delay time ensures reliable commutation of thyristors in converter 1. If the converter 2 is triggered before the converter 1 has been completely turned-off, a large circulating current would flow between the two converters. Irregular jumps in the level of the DC terminal voltage at the point of current reversal must be avoided in order to achieve a smooth changeover of current from one converter to the other. Thus the firing pulse control should, ideally, be such that the mean DC terminal voltage of the converter 2, at the instant of current reversal, is the same as that of the converter 1. From the above discussion, it becomes clear that such a mode of operation requires sophisticated control system which automatically blocks and unblocks the individual converters in accordance with the direction of load current suitably with a safety intervals.

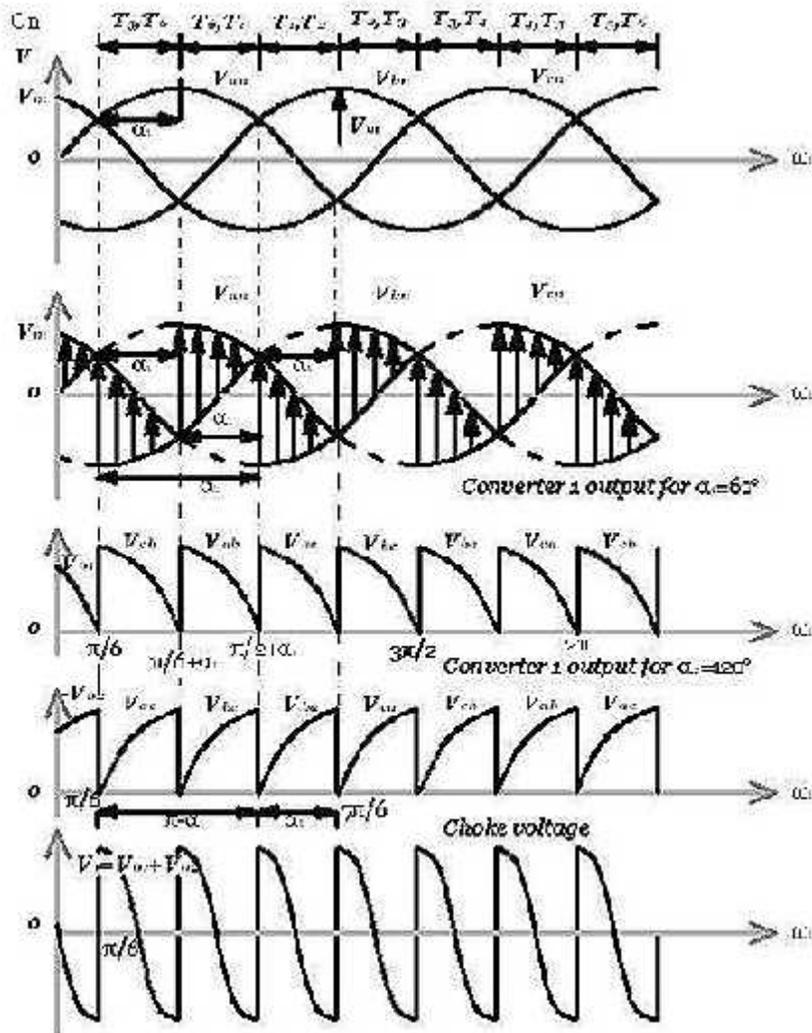
Circuit Diagram:



UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Model output waveform:



Observation Table:

S.No.	Firing Angle(α)	Output Voltage(Volts)	Time period(ms)
1			
2			
3			

Result and Discussions:

Post Experiment Question:

1. What are the Applications of Dual Converter.
2. Dual converter is a combination of a _____

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Experiment No: 8 Observe the performance of step down chopper with R and R-L load.

Aim: To study the performance of step down chopper with R and R-L load.

Apparatus required:

1. SCR, Rheostat, CRO, DC voltage source (Regulated power supply), Connection wires.
2. PSIM module (for simulation).
3. 8051 Trainer kit, driver circuit module.

For Hardware experiment purpose:

S. no.	Name of equipment's/components	Specification	Range/Rating	Qty.
1.				
2.				
3.				
4.				
5.				

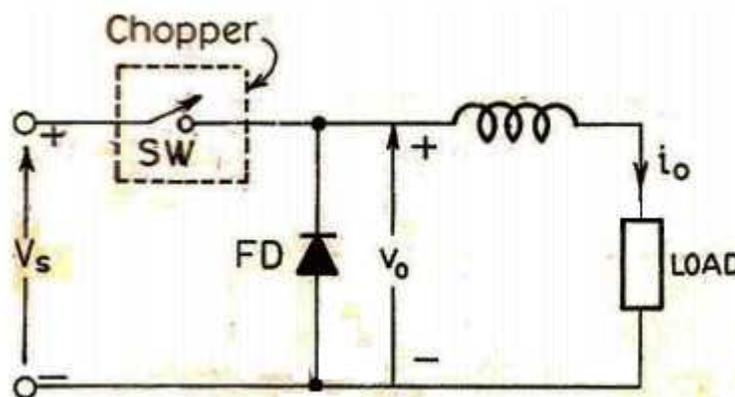
Theory:

A chopper is a high speed ON/OFF switch. It connects source to load and disconnect the load from the source at very fast speed. Hence a chopped output voltage is obtained from a constant DC supply. During the period T_{on} , chopper is ON and load voltage is equal to source voltage V_s . During interval T_{off} , copper is OFF, load current flows through freewheeling diode. As a result load terminal are short circuited by FD and load voltage is therefore zero during T_{off} . In this manner a chopped dc voltage is produced at the load terminals. During T_{on} , load current rises, whereas during T_{off} , load current decays. The average load voltage of the chopper can be given by:

$$V_0 = \frac{T_{ON}}{T_{ON} + T_{OFF}} V_s = \alpha V_s$$

Where α is the duty cycle .

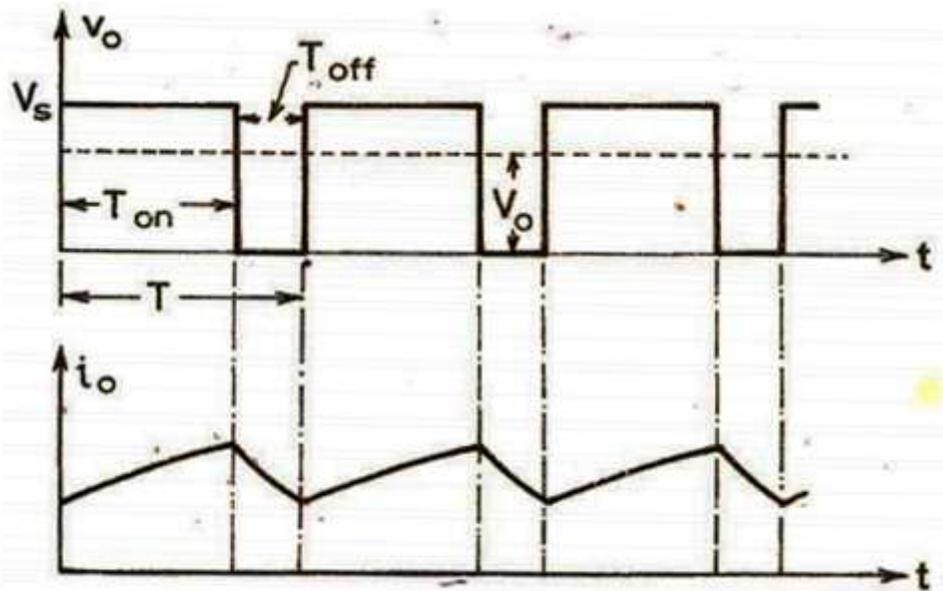
Circuit Diagram:



UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Model Waveforms:



Procedure:

Step 1 Probe across load resistance (V_o).

Step 2 Keep the multiplication factor of the CRO's probe at the maximum position (10X or 100X - whichever is available).

Step 3 Switch on the experimental kit and firing circuit kit.

Step 4 Set the duty cycle (duty ratio) to 0.1 (10%) and capture output waveforms on oscilloscope.

Step 5 Measure the RMS value of the output and take screenshot of output waveform.

Step 6 Now change the duty cycle to 0.2 (20%).

Step 7 Measure the RMS value of the output and take screenshot of output waveform.

Step 8 Continue Step # 6 for different values of duty cycle like 30%, 40%... till 90%.

Observation Table:

S.No.	FiringAngle(α)	Output Voltage(Volts)	Time period(ms)
1			
2			
3			

Result and Discussions:

Post Experiment Question:

1. What is the relationship of RMS value of output with the duty cycle?
2. What happens to RMS value of the output when duty cycle is increased from 10% to 90%?

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR
Lab Manual

Matlab waveforms:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Software Engineering Lab

Course Code: EE694A

L-T-P scheme: 0-0-3

Course Credit: 2

Objectives:

1. To learn and understand system calls related to files, processes, signals, semaphores and implement system programs based on that.
2. To provide an understanding of the design aspects of operating system.
3. To provide an efficient understanding of the language translation peculiarities by designing a complete translator for a mini language.

Learning Outcomes: The students will have a detailed knowledge of the concepts of process and shared memory, aware of a variety of approaches to process management and main-memory management, including interference, deadlock, scheduling, fragmentation, thrashing, learn the basics behind file systems and input output systems and understand the fundamentals of network and distributed operating systems. Upon the completion of Operating Systems practical course, the student will be able to:

-) **Understand** and implement basic services and functionalities of the operating system using system calls.
-) **Use** modern operating system calls and synchronization libraries in software/ hardware interfaces.
-) **Understand** the benefits of thread over process and implement synchronized programs using multithreading concepts.
-) **Analyze** and simulate CPU Scheduling Algorithms like FCFS, Round Robin, SJF, and Priority.
-) **Implement** memory management schemes and page replacement schemes.
-) **Simulate** file allocation and organization techniques.
-) **Understand** the concepts of deadlock in operating systems and implement them in multiprogramming system.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Database Management System Lab

Course Code: CS694B

L-T-P Scheme: 0-0-3

Course Credits: 2

Objective:

At the end of the semester, the students should have clearly understood and implemented the following:

1. Stating a database design problem.
2. Preparing ER diagram
3. Finding the data fields to be used in the database.
4. Selecting fields for keys.
5. Normalizing the database including analysis of functional dependencies.
6. Installing and configuring the database server and the front end tools.
7. Designing database and writing applications for manipulation of data for a stand alone and shared database including concepts like concurrency control, transaction roll back, logging, report generation etc.
8. Get acquainted with SQL. In order to achieve the above objectives, it is expected that each students will chose one problem. The implementation shall being with the statement of the objectives to be achieved, preparing ER diagram, designing of database, normalization and finally manipulation of the database including generation of reports, views etc. The problem may first be implemented for a standalone system to be used by a single user. All the above steps may then be followed for development of a database application to be used by multiple users in a client server environment with access control. The application shall NOT use web techniques. One exercise may be assigned on creation of table, manipulation of data and report generation using SQL.

Learning Outcomes:

- Ability to build normalized databases.
- Knowledge of Entity Relationship Modelling.
- Familiarity with SQL, embedded SQL and PLSQL.
- Familiarity with query processing and query optimization techniques.
- Understanding of transaction processing.
- Ability to handle recovery and concurrency issues.
- Familiarity with ODBC, JDBC.

Course Contents:

Exercises that must be done in this course are listed below:

Exercise No.1: ER Model: An entity-relationship model (ERM) is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system

Exercise No. 2: EER Model: In computer science, the enhanced entity-relationship (EER) model is a high-level or conceptual data model incorporating extensions to the original entity-relationship (ER) model, used in the design of databases. It was developed by a need to reflect more precisely properties and constraints that are found in more complex databases.

Exercise No. 3: Relational Model: The relational model for database management is a database model based on first-order predicate logic, first formulated and proposed in 1969 by E.F. Codd. The model uses the concept of a mathematical relation, which looks somewhat like a table of values - as its basic building block, and has its theoretical basis in set theory and first-order predicate logic.

Exercise No. 4: 1 NF: First normal form (1NF or Minimal Form) is a normal form used in database normalization. A relational database table that adheres to 1NF is one that meets a certain minimum set of criteria. These criteria are basically concerned with ensuring that the table is a faithful representation of a relation and that it is free of repeating groups.

Exercise No. 5: 2 NF: Second normal form (2NF) is a normal form used in database normalization. 2NF was originally defined by E.F. Codd in 1971. A table that is in first normal form (1NF) must

Exercise No. 6: 3 NF: The Third normal form (3NF) is an important form of database normalization. 3NF is said to hold if and only if both of the following conditions hold:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

- The relation R (table) is in second normal form (2NF)
- Every non-prime attribute of R is non-transitively dependent (i.e. directly dependent) on every candidate key of R.

Exercise No. 7: BCNF: A relation R is in Boyce-Codd normal form (BCNF) if and only if every determinant is a candidate key. 4The definition of BCNF addresses certain (rather unlikely) situations which 3NF does not handle.

Exercise No. 8: SQL-1: In this lab., we discuss basic SQL operations like creating a table, deleting a table, changing the schema of the table, primary key and foreign key constraints on a table and creating indexes on tables.

Exercise No. 9: SQL-2: Its scope includes efficient data insert, query, update and delete, schema creation and modification, and data access control. In this lab., we discuss SQL operations for populating the tables like inserting into a table, deleting values from a table, and updating the content of the tables.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

1 Introduction of Data Base

What is a database?

A database is a collection of information that is organized so that it can easily be accessed, managed, and updated. In one view, databases can be classified according to types of content: bibliographic, full-text, numeric, and images.

In computing, databases are sometimes classified according to their organizational approach. The most prevalent approach is the relational database, a tabular database in which data is defined so that it can be reorganized and accessed in a number of different ways. A distributed database is one that can be dispersed or replicated among different points in a network. An object-oriented programming database is one that is congruent with the data defined in object classes and subclasses.

Computer databases typically contain aggregations of data records or files, such as sales transactions, product catalogs and inventories, and customer profiles. Typically, a database manager provides users the capabilities of controlling read/write access, specifying report generation, and analyzing usage. Databases and database managers are prevalent in large mainframe systems, but are also present in smaller distributed workstation and mid-range systems such as the AS/400 and on personal computers. SQL (Structured Query Language) is a standard language for making interactive queries from and updating a database such as IBM's DB2, Microsoft's Access, and database products from Oracle, Sybase, and Computer Associates.

A **Database Management System (DBMS)** is a set of computer programs that controls the creation, maintenance, and the use of a database. It allows organizations to place control of database development in the hands of database administrators (DBAs) and other specialists. A DBMS is a system software package that helps the use of integrated collection of data records and files known as databases. It allows different user application programs to easily access the same database. DBMSs may use any of a variety of database models, such as the network model or relational model. In large systems, a DBMS allows users and other software to store and retrieve data in a structured way. Instead of having to write computer programs to extract information, user can ask simple questions in a query language. Thus, many DBMS packages provide Fourth-generation programming language (4GLs) and other application development features. It helps to specify the logical organization for a database and access and use the information within a database. It provides facilities for controlling data access, enforcing data integrity, managing concurrency, and restoring the database from backups. A DBMS also provides the ability to logically present database information to users

A DBMS is a set of software programs that controls the organization, storage, management, and retrieval of data in a database. DBMSs are categorized according to their data structures or types. The DBMS accepts requests for data from an application program and instructs the operating system to transfer the appropriate data. The queries and responses must be submitted and received according to a format that conforms to one or more applicable protocols. When a DBMS is used, information systems can be changed much more easily as the organization's information requirements change. New categories of data can be added to the database without disruption to the existing system.

Database servers are computers that hold the actual databases and run only the DBMS and related software. Database servers are usually multiprocessor computers, with generous memory and RAID disk arrays used for stable storage. Hardware database accelerators, connected to one or more servers via a high-speed channel, are also used in large volume transaction processing environments. DBMSs are found at the heart of most database applications. DBMSs may be built around a custom multitasking kernel with built-in networking support, but modern DBMSs typically rely on a standard operating system to provide these functions.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

A database management system (DBMS), sometimes just called a *database manager*, is a program that lets one or more computer users create and access data in a database. The DBMS manages user requests (and requests from other programs) so that users and other programs are free from having to understand where the data is physically located on storage media and, in a multi-user system, who else may also be accessing the data. In handling user requests, the DBMS ensures the *integrity* of the data (that is, making sure it continues to be accessible and is consistently organized as intended) and *security* (making sure only those with access privileges can access the data). The most typical DBMS is a relational database management system (RDBMS). A standard user and program interface is the Structured Query Language (SQL). A newer kind of DBMS is the object-oriented database management system (ODBMS).

A DBMS can be thought of as a *file manager* that manages data in databases rather than files in file systems. In IBM's mainframe operating systems, the nonrelational data managers were (and are, because these legacy application systems are still used) known as *access methods*.

A DBMS is usually an inherent part of a database product. On PCs, Microsoft Access is a popular example of a single- or small-group user DBMS. Microsoft's SQL Server is an example of a DBMS that serves database requests from multiple (client) users. Other popular DBMSs (these are all RDBMSs, by the way) are IBM's DB2, Oracle's line of database management products, and Sybase's products.

IBM's Information Management System (IMS) was one of the first DBMSs. A DBMS may be used by or combined with transaction managers, such as IBM's Customer Information Control System (CICS).

Real-Life Database Examples

To say that the databases are everywhere would be an understatement. They virtually permeate our lives: Online stores, health care providers, clubs, libraries, video stores, beauty salons, travel agencies, phone companies, government agencies like FBI, INS, IRS, and NASA — they all use databases. These databases can be very different in their nature and usually have to be specifically designed to cater to some special customer needs. Here are some examples.

Note All relational databases can be divided into two main categories according to their primary function — *online transaction processing (OLTP)* and *data warehouse* systems. OLTP typically has many users simultaneously creating and updating individual records; in other words it's volatile and computation-intensive. Data warehouse is a database designed for information processing and analysis, with focus on planning for the future rather than on day-to-day operations. The information in these is not going to change very often, which ensures the information consistency (repeatable result) for the users. In the real world most systems are hybrids of these two, unless specifically designed as data warehouse.

Order management system database

A typical database for a company that sells building materials might be arranged as follows: The company must have at least one customer. Each customer in the database is assigned one or more addresses, one or more contact phones, and a default salesperson who is the liaison between the customer and the company. The company sells a variety of products. Each product has a price, a description, and some other characteristics. Orders can be placed for one or more product at a time. Each product logically forms an order line. When an order is complete it can be shipped and then invoiced. Invoice number and shipment number are populated automatically in the database and can not be changed by users. Each order has a status assigned to it: COMPLETE, SHIPPED, INVOICED, and so on. The database also contains specific shipment information (bill of lading number, number of boxes shipped, dates, and so on). Usually one shipment contains one order, but the database is designed in such a way that one order can be distributed between more than one shipment, as well as one shipment can contain more than one order. Some constraints also exist in the database. For example, some fields cannot be empty, and some other fields can contain only certain types of information.

You already know that a database is a multi user environment by definition. It's a common practice to group users according to the functions they perform and security levels they are entitled to. The order management system described here could have three different user groups: Sales department clerks' function is to enter or modify order and customer information; shipping department employees create and update shipment data; warehouse supervisors handle products. In addition, all three user groups view diverse database information under different angles, using reports and ad-hoc queries.

We'll use this database, which we'll call ACME, throughout this book for examples and exercises. ACME database is a simplified version of a real production database. It has only 13 tables, and the real one would easily have over a hundred.

Health care provider database

A health provider company has multiple offices in many different states. Many doctors work for the company, and each doctor takes care of multiple patients. Some doctors just work in one office, and others work in different offices on different days. The database keeps information about each doctor, such as name, address, contact phones, area of specialization, and so on. Each patient can be assigned to one or more doctors. Specific patient information is also kept in the database (name, address, phones, health record number, date of birth, history of appointments, prescriptions, blood tests, diagnoses, etc.). Customers can schedule and cancel appointments and order prescription drugs either over the phone or using the company Web site. Some restrictions apply — for example, to see a specialist, the patient needs an approval from his/her primary physician; to order a prescription the patient should have at least one valid refill left, and so on.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Now, what are the main database user groups? Patients should be able to access the database using a Web browser to order prescriptions and make appointments. This is all that patients may do in the database. Doctors and nurses can browse information about their patients, write and renew prescriptions, schedule blood tests and X-Rays, and so on. Administrative staff (receptionists, pharmacy assistants) can schedule appointments for patients, fill prescriptions, and run specific reports.

Again, in real life this database would be far more complicated and would have many more business rules, but our main goal now is just to give a general idea what kind of information a database could contain.

The health provider and order management system databases are both examples of a typical *hybrid* database (though the former is probably closer to an OLTP).

Scientific database

A database for genome research and related research areas in molecular and cellular biology can be a good example of a scientific database. It contains gene catalogs for completely sequenced genomes and some partial genomes, genome maps and organism information, and data about sequence similarities among all known genes in all organisms in the database. It also contains information on molecular interaction networks in the cell and chemical compounds and reactions.

This database has just one user group — all researchers have the same access to all the information. This is an example of a data warehouse.

Nonprofit organization database

A database of an antique automobile club can be pretty simple. Also, such an organization would not typically have too many members, so the database is not going to be very large. You need to store members' personal information such as address, phone number, area of interest, and so on. The database might also contain the information about the autos (brand, year, color, condition, etc.). Autos are tied to their owners (members of the club). Each member can have one or more vehicles, and a vehicle can be owned by just one member.

The database would only have a few users — possibly, the chairman of the club, an assistant, and a secretary.

The last two examples are not business-critical databases and don't have to be implemented on expensive enterprise software. The data still have to be kept safely and should not be lost, but in case of, let's say, hardware failure it probably can wait a day or two before the database is restored from a backup. So, the use of a free database, like MySQL, PostgreSQL, or even nonrelational Posgres is appropriate. Another good choice might be MS Access, which is a part of Microsoft Office Tools; if you bought MS Office just because you want to use Word and Excel, you should be aware that you've got a free relational database as well. (MS Access works well with up to 15 users.)

3. Overview of SQL DDL, DML and DCL Commands With Examples.

DDL is Data Definition Language statements. Some examples:

CREATE - to create objects in the database

ALTER - alters the structure of the database

DROP - delete objects from the database

TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed

COMMENT - add comments to the data dictionary

GRANT - gives user's access privileges to database

REVOKE - withdraw access privileges given with the GRANT command

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

DML is Data Manipulation Language statements. Some examples:

SELECT - retrieve data from the a database

INSERT - insert data into a table

UPDATE - updates existing data within a table

DELETE - deletes all records from a table, the space for the records remain

CALL - call a PL/SQL or Java subprogram

EXPLAIN PLAN - explain access path to data

LOCK TABLE - control concurrency

DCL is Data Control Language statements. Some examples:

COMMIT - save work done

SAVEPOINT - identify a point in a transaction to which you can later roll back

ROLLBACK - restore database to original since the last COMMIT

SET TRANSACTION - Change transaction options like what rollback segment to use

Basic SQL DDL Commands.

To practice basic SQL DDL Commands such as CREATE, DROP, etc.

1. SQL - CREATE TABLE

Syntax: CREATE TABLE tablename (column_name data_type constraints, ...)

Example:

INPUT:

```
SQL> CREATE TABLE Emp ( EmpNo short CONSTRAINT PKey PRIMARY KEY,
EName VarChar(15), Job Char(10) CONSTRAINT Unik1 UNIQUE,
Mgr short CONSTRAINT FKey1 REFERENCES EMP (EmpNo),
Hiredate Date, DeptNo short CONSTRAINT FKey2 REFERENCES DEPT(DeptNo));
```

RESULT: Table created.

```
SQL> Create table prog20 (pname varchar2(20) not null, doj date not null, dob date not null,
sex varchar(1) not null, prof1 varchar(20), prof2 varchar(20), salary number(7,2) not null);
```

RESULT:

Table created.

```
SQL> desc prog20;
```

Name Null? Type

```
PNAME NOT NULL VARCHAR2(20)
```

```
DOJ NOT NULL DATE
```

```
DOB NOT NULL DATE
```

```
SEX NOT NULL VARCHAR2(1)
```

```
PROF1 VARCHAR2(20)
```

```
PROF2 VARCHAR2(20)
```

```
SALARY NOT NULL NUMBER(7,2)
```

2. SQL - ALTER TABLE

INPUT:

```
SQL> ALTER TABLE EMP ADD CONSTRAINT Pkey1 PRIMARY KEY (EmpNo);
```

RESULT: Table Altered.

Similarly, ALTER TABLE EMP DROP CONSTRAINT Pkey1;

3. SQL - DROP TABLE

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

– Deletes table structure – Cannot be recovered – Use with caution

INPUT:

SQL> DROP TABLE EMP; Here EMP is table name

RESULT: Table Dropped.

4. **TRUNCATE** TRUNCATE TABLE <TABLE NAME>;

Basic SQL DML Commands.

To practice basic SQL DML Commands such as INSERT, DELETE, etc.

1. SQL - INSERT INTO

Syntax: INSERT INTO tablename VALUES (value list)

Single-row insert

INSERT INTO S VALUES('S3','SUP3','BLORE',10)

Inserting one row, many columns at a time

INSERT INTO S (SNO, SNAME) VALUES ('S1', 'Smith');S1' Smith'

Inserting many rows, all/some columns at a time.

INSERT INTO NEW_SUPPLIER (SNO, SNAME)

SELECT SNO, SNAME FROM S

WHERE CITY IN ('BLORE','MADRAS')

Other Examples:

INPUT:

SQL>Insert into prog values ('kkk','05-may-56');

RESULT: 1 row created.

INPUT:

SQL>Insert into prog20 values('Hema','25-sept-01'28-jan-85','f','c','c++','25000');

RESULT: 1 row created.

INPUT:

SQL>Insert into prog values('&pname','&doj');

SQL> Insert into prog values('&pname','&doj');

Enter value for pname: ravi

Enter value for doj: 15-june-81

RESULT:

old 1: Insert into prog values('&pname','&doj')

new 1: Insert into prog values('ravi','15-june-81')

1 row created.

2. SQL - UPDATE

Syntax: UPDATE tablename SET column_name =value [WHERE condition]

Examples:

UPDATE SET CITY = 'KANPUR' WHERE SNO='S1'

UPDATE EMP SET SAL = 1.10 * SAL

SQL> update emp set sal=20000 where empno=7369;

1 row updated.

3. SQL - DELETE FROM

Syntax: DELETE FROM tablename WHERE condition

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Examples:

DELETE FROM SP WHERE PNO= 'P1'

DELETE FROM SP

INPUT:

SQL>Delete from emp where empno=7369;

RESULT: 1 row deleted.

Basic SQL DCL Commands.

To practice basic SQL DCL Commands such as COMMIT, ROLLBACK etc.

1. COMMIT

Save changes (transactional).

Syntax:

COMMIT [WORK] [COMMENT '*comment_text*']

COMMIT [WORK] [FORCE '*force_text*' [,*int*]]

FORCE - will manually commit an in-doubt *distributed* transaction

force_text - transaction identifier (see the DBA_2PC_PENDING view)

int - sets a specific SCN.

If a network or machine failure prevents a distributed transaction from committing properly, Oracle will store any commit comment in the data dictionary along with the transaction ID.

INPUT:

SQL>commit;

RESULT: Commit complete.

2. ROLLBACK

Undo work done (transactional).

Syntax:

ROLLBACK [WORK] [TO [SAVEPOINT] '*savepoint_text_identifier*'];

ROLLBACK [WORK] [FORCE '*force_text*'];

FORCE - will manually rollback an in-doubt *distributed* transaction

INPUT:

SQL>rollback;

RESULT:Rollback complete.

3. SAVEPOINT

Save changes to a point (transactional).

Syntax:

SAVEPOINT *text_identifier*

Example:

UPDATE employees

SET salary = 95000

WHERE last_name = 'Smith';

SAVEPOINT justsmith;

UPDATE employees

SET salary = 1000000;

SAVEPOINT everyone;

SELECT SUM(salary) FROM employees;

ROLLBACK TO SAVEPOINT justsmith;

COMMIT;

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Writing and Practice of Simple Queries.

To write simple queries and practice them.

1. Get the description of EMP table.

SQL>desc emp;

RESULT:

Name Null? Type

EMPNO NOT NULL NUMBER(4)
ENAME VARCHAR2(10)
JOB VARCHAR2(9)
MGR NUMBER(4)
HIREDATE DATE
SAL NUMBER(7,2)
COMM NUMBER(7,2)
DEPTNO NUMBER(3)
AGE NUMBER(3)
ESAL NUMBER(10)

2. Get the description DEPT table.

SQL>desc dept;

RESULT:

Name Null? Type

DEPTNO NOT NULL NUMBER(2)
DNAME VARCHAR2(14)
LOC VARCHAR2(13)

3. List all employee details.

SQL>select * from emp;

RESULT:

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO AGE ESAL

7369 SMITH CLERK 7902 17-DEC-80 800 0 20 25 0
7499 ALLEN SALESMAN 7698 20-FEB-81 1600 300 30 25 0
7521 WARD SALESMAN 7698 22-FEB-81 1250 500 30 25 0
7566 JONES MANAGER 7839 02-APR-81 2975 500 20 25 0
7698 BLAKE MANAGER 7839 01-MAY-81 2850 1400 30 25 0

4. List all employee names and their salaries, whose salary lies between 1500/- and 3500/- both inclusive.

INPUT

SQL>select ename from emp where sal between 1500 and 3500;

RESULT

ENAME

ALLEN
JONES
BLAKE
CLARK
SCOTT
TURNER

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

FORD
russel
greg
9 rows selected.

5. List all employee names and their and their manager whose manager is 7902 or 7566 or 7789.

INPUT SQL>select ename from emp where mgr in(7602,7566,7789);

RESULT

ENAME

SCOTT

FORD

6. List all employees which starts with either J or T.

INPUT SQL>select ename from emp where ename like 'J%' or ename like 'T%';

RESULT:

ENAME

JONES

TURNER

JAMES

7. List all employee names and jobs, whose job title includes M or P.

INPUT SQL>select ename,job from emp where job like 'M%' or job like 'P%';

RESULT:

ENAME JOB

JONES MANAGER

BLAKE MANAGER

CLARK MANAGER

KING PRESIDENT

8. List all jobs available in employee table.

INPUT SQL>select distinct job from emp;

RESULT:

JOB

ANALYST

CLERK

MANAGER

PRESIDENT

SALESMAN

assistant

clerk

7 rows selected.

9. List all employees who belongs to the department 10 or 20.

INPUT SQL>select ename from emp where deptno in (10,20);

RESULT:

ENAME

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

SMITH
JONES
CLARK
SCOTT
KING
ADAMS
FORD
MILLER
8 rows selected.

10. List all employee names , salary and 15% rise in salary.

INPUT SQL>select ename , sal , sal+0.15* sal from emp;

RESULT:

ENAME SAL SAL+0.15*SAL

SMITH 800 920
ALLEN 1600 1840
WARD 1250 1437.5
JONES 2975 3421.25
MARTIN 1250 1437.5
BLAKE 2850 3277.5
CLARK 2450 2817.5

7 rows selected.

11. List minimum , maximum , average salaries of employee.

INPUT SQL>select min(sal),max(sal),avg(sal) from emp;

RESULT:

MIN(SAL) MAX(SAL) AVG(SAL)

3 5000 1936.94118

12. Find how many job titles are available in employee table.

INPUT SQL>select count (distinct job) from emp;

RESULT:

COUNT(DISTINCTJOB)

7

13. What is the difference between maximum and minimum salaries of employees in the organization?

INPUT SQL>select max(sal)-min(sal) from emp;

RESULT:

MAX(SAL)-MIN(SAL)

4997

14. Display all employee names and salary whose salary is greater than minimum salary of the company and job title starts with 'M'.

INPUT SQL>select ename,sal from emp where job like 'M%' and sal > (select min (sal) from emp);

RESULT

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

ENAME SAL

JONES 2975
BLAKE 2850
CLARK 2450

15. Find how much amount the company is spending towards salaries.

INPUT SQL>select sum (sal) from emp;

RESULT

SUM(SAL)

32928

16. Display name of the dept. with deptno 20.

INPUT SQL>select ename from emp where deptno = 20;

RESULT

ENAME

SMITH
JONES
SCOTT
ADAMS

17. List ename whose commission is NULL.

INPUT SQL>select ename from emp where comm is null;

ENAME

RESULT -----

CLARK
SCOTT
KING
ADAMS
JAMES
FORD
6 rows selected.

18. Find no.of dept in employee table.

INPUT SQL>select count (distinct ename) from emp;

RESULT

COUNT(DISTINCTENAME)

17

19. List ename whose manager is not NULL.

INPUT SQL>select ename from emp where mgr is not null;

RESULT

ENAME

SMITH
ALLEN
WARD
JONES
MARTIN

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

5 rows selected.

Writing Queries using GROUP BY and other clauses.

To write queries using clauses such as GROUP BY, ORDER BY, etc. and retrieving information by joining tables.

Source tables: emp, dept, programmer, software, study.

Order by : The order by clause is used to display the results in sorted order.

Group by : The attribute or attributes given in the clauses are used to form groups. Tuples with the same value on all attributes in the group by clause are placed in one group.

Having: SQL applies predicates (conditions) in the having clause after groups have been formed, so aggregate function be used.

1. Display total salary spent for each job category.

INPUT SQL>select job,sum (sal) from emp group by job;

RESULT

JOB SUM(SAL)

ANALYST 6000
CLERK 23050
MANAGER 8275
PRESIDENT 5000
SALESMAN 5600
assistant 2200
clerk 2003

7 rows selected.

2. Display lowest paid employee details under each manager.

INPUT SQL>select ename, sal from emp where sal in (select min(sal) from emp group by mgr);

RESULT

ENAME SAL

chai 3
JAMES 950
MILLER 1000
ADAMS 1100
russel 2200

5 rows selected.

3. Display number of employees working in each department and their department name.

INPUT SQL> select dname, count (ename) from emp, dept where emp.deptno=dept.deptno group by dname;

RESULT

DNAME COUNT(ENAME)

ACCOUNTING 3

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

RESEARCH 5

SALES 9

4. Display the sales cost of package developed by each programmer.

INPUT SQL>select pname, sum(scost) from software group by pname;

RESULT

PNAME SUM(SCOST)

john 12000

kamala 12000

raju 12333

3 rows selected.

5. Display the number of packages sold by each programmer.

INPUT SQL>select pname, count(title) from software group by pname;

RESULT

PNAME COUNT(TITLE)

john 1

kamala 1

raju 1

ramana 1

rani 1

5 rows selected.

6. Display the number of packages in each language for which the development cost is less than thousand.

INPUT SQL>select devin, count(title) from software where dcost < 1000 group by devin;

RESULT

DEVIN COUNT(TITLE)

cobol 1

7. Display each institute name with number of students.

INPUT SQL>select splace, count(pname) from study group by splace;

RESULT

SPLACE COUNT(PNAME)

BDPS 2

BITS 1

BNRILLIANI 1

COIT 1

HYD 1

5 rows selected.

8. How many copies of package have the least difference between development and selling cost, were sold?

INPUT SQL>select sold from software where scost – dcost=(select min(scost – dcost) from software);

RESULT

SOLD

11

9. Which is the costliest package developed in Pascal.

INPUT SQL>select title from software where devin = 'PASCAL' and dcost = (select max(dcost)from software where devin = 'PASCAL');

RESULT

no rows selected

10. Which language was used to develop most no .of packages.

INPUT SQL>select devin, count (*) from software group by devin having count(*) = (select max(count(*)) from software group by devin);

RESULT

DEVIN COUNT(*)

jsp 2

11. Who are the male programmers earning below the average salary of female programmers?

INPUT SQL>select pname from programmer where sal < (select avg(sal) from programmer where sex = 'F') and sex = 'M';

RESULT

PNAME

vijay

12. Display the details of software developed by the male programmers earning more than 3000/-.

INPUT SQL>select programmer.pname, title, devin from programmer, software where sal > 3000 and sex = 'M' and programmer.pname = software.pname;

RESULT

no rows selected

13. Display the details of software developed in c language by female programmers of pragathi.

INPUT SQL>select software.pname, title, devin, scost, dcost, sold from programmer, software, study where devin = 'c' and sex ='F' and splace = 'pragathi' and programmer.pname = software.pname and software.pname = study.pname;

14. Which language has been stated by the most of the programmers as proficiency one?

INPUT SQL>select prof1, count(*) from programmer group by prof1 having count (*) = (select max (count (*)) from programmer group by prof1);

Writing Nested Queries.

To write queries using Set operations and to write nested queries.

Set Operations:

UNION - OR

INTERSECT - AND

EXCEPT - - NOT

NESTED QUERY:- A nested query makes use of another sub-query to compute or retrieve the information.

1. Find the name of the institute in which the person studied and

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

developed the costliest package.

INPUT SQL>select splace, pname from study where pname = (select pname from software where scost = (select max (scost) from software));

RESULT

SPLACE PNAME

SAHBHARI MARY

2. Find the salary and institute of a person who developed the highest selling package.

INPUT SQL> select study.pname, sal, splace from study, programmer where study.pname = programmer.pname and study.pname = (select pname from software where scost = (select max (scost) from software));

RESULT

PNAME SAL SPLACE

MARY 4500 SABHARI

3. How many packages were developed by the person who developed the cheapest package.

INPUT SQL>select pname, count (title) from software where dcost = (select min(dcost) from software) group by pname;

RESULT

PNAME COUNT(TITLE)

VIJAY 1

4. Calculate the amount to be recovered for those packages whose development cost has not yet recovered.

INPUT SQL>select title , (dcost-scost) from software where dcost > scost;

5. Display the title, scost, dcost, difference of scost and dcost in the descending order of difference.

INPUT SQL> select title, scost, dcost, (scost - dcost) from software descending order by (scost-dcost);

6. Display the details of those who draw the same salary.

INPUT SQL> select p.pname, p.sal from programmer p, programmer t where p.pname <> t.pname and p.sal = t.sal;(or)

INPUT SQL>select pname,sal from programmer t where pname<>t.pname and sal= t.sal;

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Writing Queries using functions.

AIM: To write queries using single row functions and group functions.

1. Display the names and dob of all programmers who were born in january.

INPUT SQL>select pname , dob from programmer where to_char (dob,'MON')='JAN';

2. Calculate the experience in years of each programmer and display along with programmer name in descending order.

INPUT SQL> select pname, round (months_between(sysdate, doj)/12, 2) "EXPERIENCE" from programmer order by months_between (sysdate, doj) desc;

3. List out the programmer names who will celebrate their birthdays during current month.

INPUT SQL>select pname from programmer where to_char(dob,'MON') like to_char (sysdate, 'MON');

4. Display the least experienced programmer's details.

INPUT SQL>select * from programmer where doj = (select max (doj) from programmer);

5. Who is the most experienced programmer knowing pascal.

INPUT SQL>select pname from programmer where doj = (select min (doj) from programmer);

6. Who is the youngest programmer born in 1965.

INPUT SQL> select pname , dob from programmer where dob = (select max (dob) from programmer where to_char (dob,'yy') = 65);

7. In which year, most of the programmers are born.

INPUT SQL>select to_char (dob , 'YY') from programmer group by to_char (dob, 'YY') having count(*) = (select max (count(*)) from programmer group by to_char(dob, 'YY');

8. In which month most number of programmers are joined.

INPUT SQL>select to_char (doj,'YY') from programmer group by to_char (doj,'YY') having count (*) = (select max (count(*)) from programmer group by to_char (doj,'YY');

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

9. What is the length of the shortest name in programmer table ?

INPUT SQL>select length (pname) from programmer where length (pname) = select min (length (pname) from programmer);

10. Display the names of the programmers whose name contains up to 5 characters.

INPUT SQL>select pname from programmer where length (pname) <=5;

11. Display all packages names in small letters and corresponding programmer names in uppercase letters.

INPUT SQL>select lower (title), upper (pname) from software;

Writing Queries on views.

AIM: To write queries on views.

1. Create a view from single table containing all columns from the base table.

SQL>create view view1 as (select * from programmer);

2. Create a view from single table with selected columns.

SQL>create a view view2 as (select pname,dob,doj,sex,sal from programmer);

3. Create a view from two tables with all columns.

SQL>create view xyz as select * from programmer full natural join software;

4. Create a view from two tables with selected columns.

SQL> create view lmn as (select programmer, pname, title, devin from programmer, software where sal < 3000 and programmer.pname = software.pname);

5. Check all DML commands with above 4 views.

INPUT SQL> insert into view1 values ('ramu','12-sep-03','28-jan-85','f','dbase','oracle',74000);

RESULT

1 row created;

INPUT SQL>update view1 set salary =50000 where pname like 'raju';

RESULT 1 row updated.

Note: update command does not works for all queries on views.

INPUT SQL>delete from view1 where pname like 'raju';

RESULT 1 row deleted.

6. Drop views which you generated.

INPUT SQL>drop view view1;

RESULT View dropped;

INPUT SQL>drop view view2;

RESULT View dropped;

INPUT SQL>drop view xyz;

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

The CREATE TABLE Command

Example:

Create all tables shown in the previous chapter along with their structure.

```
CREATE TABLE "DBA_BANKSYS"."BRANCH_MSTR"(  
"BRANCH_NO" VARCHAR2(10), "NAME" VARCHAR2(25));
```

```
CREATE TABLE "DBA_BANKSYS"."EMP_MSTR"(  
"EMP_NO" VARCHAR2(10), "BRANCH_NO" VARCHAR2(10),  
"FNAME" VARCHAR2(25), "MNAME" VARCHAR2(25),  
"LNAME" VARCHAR2(25), "DEPT" VARCHAR2(30),  
"DESIG" VARCHAR2(30));
```

```
CREATE TABLE "DBA_BANKSYS"."CUST_MSTR"(  
"CUST_NO" VARCHAR2(10), "FNAME" VARCHAR2(25),  
"MNAME" VARCHAR2(25), "LNAME" VARCHAR2(25),  
"DOB_INC" DATE, "OCCUP" VARCHAR2(25),  
"PHOTOGRAPH" VARCHAR2(25), "SIGNATURE" VARCHAR2(25),  
"PANCOPY" VARCHAR2(1), "FORM60" VARCHAR2(1));
```

```
CREATE TABLE "DBA_BANKSYS"."SPRT_DOC"(  
"ACCT_CODE" VARCHAR2(4), "TYPE" VARCHAR2(40),  
"DOCS" VARCHAR2(75));
```

```
CREATE TABLE "DBA_BANKSYS"."ACCT_MSTR"(  
"ACCT_NO" VARCHAR2(10), "SF_NO" VARCHAR2(10),  
"LF_NO" VARCHAR2(10), "BRANCH_NO" VARCHAR2(10),  
"INTRO_CUST_NO" VARCHAR2(10), "INTRO_ACCT_NO" VARCHAR2(10),  
"INTRO_SIGN" VARCHAR2(1), "TYPE" VARCHAR2(2),  
"OPR_MODE" VARCHAR2(2), "CUR_ACCT_TYPE" VARCHAR2(4),  
"TITLE" VARCHAR2(30), "CORP_CUST_NO" VARCHAR2(10),  
"APLNDT" DATE, "OPNDT" DATE,  
"VERI_EMP_NO" VARCHAR2(10), "VERI_SIGN" VARCHAR2(1),  
"MANAGER_SIGN" VARCHAR2(1), "CURBAL" NUMBER(8, 2),  
"STATUS" VARCHAR2(1));
```

```
CREATE TABLE "DBA_BANKSYS"."FD_MSTR"(  
"FD_SER_NO" VARCHAR2(10), "SF_NO" VARCHAR2(10),  
"BRANCH_NO" VARCHAR2(10), "INTRO_CUST_NO" VARCHAR2(10),  
"INTRO_ACCT_NO" VARCHAR2(10), "INTRO_SIGN" VARCHAR2(1),  
"ACCT_NO" VARCHAR2(10), "TITLE" VARCHAR2(30),  
"CORP_CUST_NO" VARCHAR2(10), "CORP_CNST_TYPE" VARCHAR2(4),  
"VERI_EMP_NO" VARCHAR2(10), "VERI_SIGN" VARCHAR2(1),  
"MANAGER_SIGN" VARCHAR2(1));
```

```
CREATE TABLE "DBA_BANKSYS"."FDSLAB_MSTR"(  
"FDSLAB_NO" NUMBER(2), "MINPERIOD" NUMBER(5),  
"MAXPERIOD" NUMBER(5), "INTRATE" NUMBER(5,2));
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
CREATE TABLE "DBA_BANKSYS"."FD_DTLS"(
"FD_SER_NO" VARCHAR2(10), "FD_NO" VARCHAR2(10),
"TYPE" VARCHAR2(1), "PAYTO_ACCTNO" VARCHAR2(10),
"PERIOD" NUMBER(5), "OPNDT" DATE,
"DUEDT" DATE, "AMT" NUMBER(8,2),
"DUEAMT" NUMBER(8,2), "INTRATE" NUMBER(3),
"STATUS" VARCHAR2(1), "AUTO_RENEWAL" VARCHAR2(1));
```

```
CREATE TABLE "DBA_BANKSYS"."ACCT_FD_CUST_DTLS"(
"ACCT_FD_NO" VARCHAR2(10), "CUST_NO" VARCHAR2(10));
```

```
CREATE TABLE "DBA_BANKSYS"."NOMINEE_MSTR"(
"NOMINEE_NO" VARCHAR2(10), "ACCT_FD_NO" VARCHAR2(10),
"NAME" VARCHAR2(75), "DOB" DATE,
"RELATIONSHIP" VARCHAR2(25));
```

```
CREATE TABLE "DBA_BANKSYS"."ADDR_DTLS"(
"ADDR_NO" NUMBER(6), "CODE_NO" VARCHAR2(10),
"ADDR_TYPE" VARCHAR2(1), "ADDR1" VARCHAR2(50),
"ADDR2" VARCHAR2(50), "CITY" VARCHAR2(25),
"STATE" VARCHAR2(25), "PINCODE" VARCHAR2(6));
```

```
CREATE TABLE "DBA_BANKSYS"."CNTC_DTLS"(
"ADDR_NO" NUMBER(6), "CODE_NO" VARCHAR2(10),
"CNTC_TYPE" VARCHAR2(1), "CNTC_DATA" VARCHAR2(75));
```

```
CREATE TABLE "DBA_BANKSYS"."TRANS_MSTR"(
"TRANS_NO" VARCHAR2(10), "ACCT_NO" VARCHAR2(10),
"DT" DATE, "TYPE" VARCHAR2(1),
"PARTICULAR" VARCHAR2(30), "DR_CR" VARCHAR2(1),
"AMT" NUMBER(8,2), "BALANCE" NUMBER(8,2));
```

```
CREATE TABLE "DBA_BANKSYS"."TRANS_DTLS"(
"TRANS_NO" VARCHAR2(10), "INST_NO" NUMBER(6),
"INST_DT" DATE, "PAYTO" VARCHAR2(30),
"INST_CLR_DT" DATE, "BANK_NAME" VARCHAR2(35),
"BRANCH_NAME" VARCHAR2(25), "PAIDFROM" VARCHAR2(10));
```

Output for each of the above CREATE TABLE statements:

Table created.

Inserting Data Into Tables

Example:

Insert the values into the BRANCH_MSTR table (For values refer to 6th chapter under Test Records)

```
INSERT INTO BRANCH_MSTR (BRANCH_NO, NAME) VALUES('B1', 'Vile Parle (HO)');
INSERT INTO BRANCH_MSTR (BRANCH_NO, NAME) VALUES('B2', 'Andheri');
INSERT INTO BRANCH_MSTR (BRANCH_NO, NAME) VALUES('B3', 'Churchgate');
INSERT INTO BRANCH_MSTR (BRANCH_NO, NAME) VALUES('B4', 'Sion');
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
INSERT INTO BRANCH_MSTR (BRANCH_NO, NAME) VALUES('B5', 'Borivali');
INSERT INTO BRANCH_MSTR (BRANCH_NO, NAME) VALUES('B6', 'Matunga');
```

Output for each of the above INSERT INTO statements:

1 row created.

Insert the values into the EMP_MSTR table (For values refer to 6th chapter under Test Records)

```
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG)
VALUES('E1', 'B1', 'Ivan', 'Nelson', 'Bayross', 'Administration', 'Managing Director');
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG)
VALUES('E2', 'B1', 'Amit', null, 'Desai', 'Loans And Financing', 'Head Of Dept. ');
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG)
VALUES('E3', 'B1', 'Maya', 'Mahima', 'Joshi', 'Accounts', 'Head Of Dept. ');
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG)
VALUES('E4', 'B1', 'Peter', 'Iyer', 'Joseph', 'Client Servicing', 'Clerk');
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG)
VALUES('E5', 'B1', 'Mandhar', 'Dilip', 'Dalvi', 'Marketing', 'Head Of Dept. ');
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG)
VALUES('E6', 'B1', 'Sonal', 'Abdul', 'Khan', 'Administration', 'Admin. Executive');
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG)
VALUES('E7', 'B1', 'Anil', 'Ashutosh', 'Kambli', 'Administration', 'Office Asst. ');
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG)
VALUES('E8', 'B1', 'Seema', 'P.', 'Apte', 'Client Servicing', 'Clerk');
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG)
VALUES('E9', 'B1', 'Vikram', 'Vilas', 'Randive', 'Accounts', 'Office Asst. ');
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG)
VALUES('E10', 'B1', 'Anjali', 'Sameer', 'Pathak', 'Marketing', 'Marketing Manager');
```

Output for each of the above INSERT INTO statements:

1 row created.

Insert the values into the CUST_MSTR table (For values refer to 6th chapter under Test Records)

```
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP,
PHOTOGRAPH, SIGNATURE, PANCOPY, FORM60)
VALUES('C1', 'Ivan', 'Nelson', 'Bayross', '25-JUN-1952', 'Self Employed',
'D:/ClntPht/C1.gif', 'D:/ClntSgnt/C1.gif', 'Y', 'Y');
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP,
PHOTOGRAPH, SIGNATURE, PANCOPY, FORM60)
VALUES('C2', 'Chriselle', 'Ivan', 'Bayross', '29-OCT-1982', 'Service',
'D:/ClntPht/C2.gif', 'D:/ClntSgnt/C2.gif', 'N', 'Y');
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP,
PHOTOGRAPH, SIGNATURE, PANCOPY, FORM60)
VALUES('C3', 'Mamta', 'Arvind', 'Muzumdar', '28-AUG-1975', 'Service',
'D:/ClntPht/C3.gif', 'D:/ClntSgnt/C3.gif', 'Y', 'Y');
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP,
PHOTOGRAPH, SIGNATURE, PANCOPY, FORM60)
VALUES('C4', 'Chhaya', 'Sudhakar', 'Bankar', '06-OCT-1976', 'Service',
'D:/ClntPht/C4.gif', 'D:/ClntSgnt/C4.gif', 'Y', 'Y');
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH, SIGNATURE, PANCOPY, FORM60)

VALUES('C5', 'Ashwini', 'Dilip', 'Joshi', '20-NOV-1978', 'Business', 'D:/ClntPht/C5.gif', 'D:/ClntSgnt/C5.gif', 'Y', 'Y');

INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH, SIGNATURE, PANCOPY, FORM60)

VALUES('C6', 'Hansel', 'I.', 'Colaco', '01-JAN-1982', 'Service', 'D:/ClntPht/C6.gif', 'D:/ClntSgnt/C6.gif', 'N', 'Y');

INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH, SIGNATURE, PANCOPY, FORM60)

VALUES('C7', 'Anil', 'Arun', 'Dhone', '12-OCT-1983', 'Self Employed', 'D:/ClntPht/C7.gif', 'D:/ClntSgnt/C7.gif', 'N', 'Y');

INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH, SIGNATURE, PANCOPY, FORM60)

VALUES('C8', 'Alex', 'Austin', 'Fernandes', '30-SEP-1962', 'Executive', 'D:/ClntPht/C8.gif', 'D:/ClntSgnt/C8.gif', 'Y', 'Y');

INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH, SIGNATURE, PANCOPY, FORM60)

VALUES('C9', 'Ashwini', 'Shankar', 'Apte', '19-APR-1979', 'Service', 'D:/ClntPht/C9.gif', 'D:/ClntSgnt/C9.gif', 'Y', 'Y');

INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH, SIGNATURE, PANCOPY, FORM60)

VALUES('C10', 'Namita', 'S.', 'Kanade', '10-JUN-1978', 'Self Employed', 'D:/ClntPht/C10.gif', 'D:/ClntSgnt/C10.gif', 'Y', 'Y');

INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH, SIGNATURE, PANCOPY, FORM60)

VALUES('O11', null, null, null, '14-NOV-1997', 'Retail Business', null, null, 'Y', 'N');

INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH, SIGNATURE, PANCOPY, FORM60)

VALUES('O12', null, null, null, '23-OCT-1992', 'Information Technology', null, null, 'Y', 'N');

INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH, SIGNATURE, PANCOPY, FORM60)

VALUES('O13', null, null, null, '05-FEB-1989', 'Community Welfare', null, null, 'Y', 'N');

INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH, SIGNATURE, PANCOPY, FORM60)

VALUES('O14', null, null, null, '24-MAY-1980', 'Retail Business', null, null, 'N', 'Y');

Output for each of the above INSERT INTO statements:

1 row created.

Insert the values into the SPRT_DOC table (For values refer to 6th chapter under Test Records)

INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)

VALUES('0S', 'Individuals / Savings Bank Account', 'Driving Licence / Ration Card / Passport');

INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)

VALUES('0S', 'Individuals / Savings Bank Account', 'Birth Certificate / School Leaving Certificate');

INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)

VALUES('1C', 'Propriety / Sole Trading Concerns', 'Letter From The Propriety');

INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)

VALUES('2C', 'Partnership Concerns', 'Letter From The Partners');

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
VALUES('2C', 'Partnership Concerns', 'Partnership Deed / Registration Certificate');
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
VALUES('3C', 'Hindu Undivided Family Businesses', 'Letter From The Karta');
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
VALUES('3C', 'Hindu Undivided Family Businesses', 'List Of Members');
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
VALUES('4C', 'Limited Companies', 'Copy Of Board Of Directors" Resolution For Opening The Account');
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
VALUES('4C', 'Limited Companies', 'Memorandum and Articles Of Association');
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
VALUES('4C', 'Limited Companies', 'Certificate Of Incorporation');
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
VALUES('4C', 'Limited Companies', 'Certificate Of Commencement Of Business / Registration Certificate');
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
VALUES('5C', 'Trust Accounts', 'Trust Deed');
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
VALUES('5C', 'Trust Accounts', 'Resolution Of Trustees');
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
VALUES('5C', 'Trust Accounts', 'List Of Trusties');
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
VALUES('6C', 'Clubs / Societies', 'Resolution');
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
VALUES('6C', 'Clubs / Societies', 'Constitution And Bye-laws');

INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
VALUES('6C', 'Clubs / Societies', 'Certificate Of Registration');
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
VALUES('7C', 'Legislative Bodies', 'Letter From The Authority');
```

Output for each of the above INSERT INTO statements:

1 row created.

Insert the values into the ACCT_MSTR table (For values refer to 6th chapter under Test Records)

```
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO,
INTRO_ACCT_NO, INTRO_SIGN, TYPE, OPR_MODE, CUR_ACCT_TYPE, TITLE, CORP_CUST_NO,
APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN, MANAGER_SIGN, CURBAL, STATUS)
VALUES('SB1', 'SF-0001', 'JAN03-05', 'B1', 'C1', 'SB1', 'Y', 'SB', 'SI', '0S', null, null, '05-JAN-2003', '05-JAN-
2003', 'E1', 'Y', 'Y', 500, 'A');
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO,
INTRO_ACCT_NO, INTRO_SIGN, TYPE, OPR_MODE, CUR_ACCT_TYPE, TITLE, CORP_CUST_NO,
APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN, MANAGER_SIGN, CURBAL, STATUS)
VALUES('CA2', 'SF-0002', 'JAN03-10', 'B1', 'C1', 'SB1', 'Y', 'CA', 'JO', '1C', 'Uttam Stores', 'O11', '07-JAN-
2003', '10-JAN-2003', 'E1', 'Y', 'Y', 2000, 'A');
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO,
INTRO_ACCT_NO, INTRO_SIGN, TYPE, OPR_MODE, CUR_ACCT_TYPE, TITLE, CORP_CUST_NO,
APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN, MANAGER_SIGN, CURBAL, STATUS)
VALUES('SB3', 'SF-0003', 'JAN03-22', 'B1', 'C4', 'SB3', 'Y', 'SB', 'SI', '0S', null, null, '20-JAN-2003', '22-JAN-
2003', 'E4', 'Y', 'Y', 500, 'A');
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO, INTRO_SIGN, TYPE, OPR_MODE, CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN, MANAGER_SIGN, CURBAL, STATUS)

VALUES('CA4', 'SF-0004', 'FEB03-05', 'B1', 'C4', 'SB3', 'Y', 'CA', 'AS', '4C', 'Sun"s Pvt. Ltd.', 'O12', '02-FEB-2003', '05-FEB-2003', 'E4', 'Y', 'Y', 2000, 'A');

INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO, INTRO_SIGN, TYPE, OPR_MODE, CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN, MANAGER_SIGN, CURBAL, STATUS)

VALUES('SB5', 'SF-0005', 'FEB03-15', 'B1', 'C1', 'SB1', 'Y', 'SB', 'JO', '0S', null, null, '14-FEB-2003', '15-FEB-2003', 'E1', 'Y', 'Y', 500, 'A');

INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO, INTRO_SIGN, TYPE, OPR_MODE, CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN, MANAGER_SIGN, CURBAL, STATUS)

VALUES('SB6', 'SF-0006', 'FEB03-27', 'B1', 'C5', 'SB6', 'Y', 'SB', 'ES', '0S', null, null, '27-FEB-2003', '27-FEB-2003', 'E4', 'Y', 'Y', 500, 'A');

INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO, INTRO_SIGN, TYPE, OPR_MODE, CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN, MANAGER_SIGN, CURBAL, STATUS)

VALUES('CA7', 'SF-0007', 'MAR03-14', 'B1', 'C8', 'CA7', 'Y', 'CA', 'AS', '6C', 'Puru Hsg. Soc', 'O13', '14-MAR-2003', '14-MAR-2003', 'E4', 'Y', 'Y', 2000, 'A');

INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO, INTRO_SIGN, TYPE, OPR_MODE, CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN, MANAGER_SIGN, CURBAL, STATUS)

VALUES('SB8', 'SF-0008', 'MAR03-29', 'B1', 'C9', 'SB8', 'Y', 'SB', 'SI', '0S', null, null, '27-MAR-2003', '29-MAR-2003', 'E1', 'Y', 'Y', 500, 'A');

INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO, INTRO_SIGN, TYPE, OPR_MODE, CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN, MANAGER_SIGN, CURBAL, STATUS)

VALUES('SB9', 'SF-0009', 'APR03-05', 'B1', 'C10', 'SB9', 'Y', 'SB', 'JO', '0S', null, null, '05-APR-2003', '05-APR-2003', 'E4', 'Y', 'Y', 500, 'A');

INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO, INTRO_SIGN, TYPE, OPR_MODE, CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN, MANAGER_SIGN, CURBAL, STATUS)

VALUES('CA10', 'SF-0010', 'APR03-19', 'B1', 'C10', 'SB9', 'Y', 'CA', 'AS', '3C', 'Ghar Karobar', 'O14', '19-APR-2003', '19-APR-2003', 'E4', 'Y', 'Y', 2000, 'A');

Output for each of the above INSERT INTO statements:

1 row created.

Insert the values into the FD_MSTR table (For values refer to 6th chapter under Test Records)

INSERT INTO FD_MSTR (FD_SER_NO, SF_NO, BRANCH_NO, ACCT_NO, TITLE, CORP_CUST_NO, CORP_CNST_TYPE, INTRO_CUST_NO, INTRO_ACCT_NO, INTRO_SIGN, VERI_EMP_NO, VERI_SIGN, MANAGER_SIGN)

VALUES ('FS1', 'SF-0011', 'B1', 'CA2', 'Uttam Stores', 'O11', '1C', null, null, 'N', 'E1', 'Y', 'Y');

INSERT INTO FD_MSTR (FD_SER_NO, SF_NO, BRANCH_NO, ACCT_NO, TITLE, CORP_CUST_NO, CORP_CNST_TYPE, INTRO_CUST_NO, INTRO_ACCT_NO, INTRO_SIGN, VERI_EMP_NO, VERI_SIGN, MANAGER_SIGN)

VALUES ('FS2', 'SF-0012', 'B1', 'CA4', 'Sun"s Pvt. Ltd.', 'C12', '4C', null, null, 'N', 'E1', 'Y', 'Y');

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

INSERT INTO FD_MSTR (FD_SER_NO, SF_NO, BRANCH_NO, ACCT_NO, TITLE, CORP_CUST_NO, CORP_CNST_TYPE, INTRO_CUST_NO, INTRO_ACCT_NO, INTRO_SIGN, VERI_EMP_NO, VERI_SIGN, MANAGER_SIGN)

VALUES ('FS3', 'SF-0013', 'B1', 'CA7', 'Puru Hsg. Soc', 'O13', '6C', null, null, 'N', 'E4', 'Y', 'Y');

INSERT INTO FD_MSTR (FD_SER_NO, SF_NO, BRANCH_NO, ACCT_NO, TITLE, CORP_CUST_NO, CORP_CNST_TYPE, INTRO_CUST_NO, INTRO_ACCT_NO, INTRO_SIGN, VERI_EMP_NO, VERI_SIGN, MANAGER_SIGN)

VALUES ('FS4', 'SF-0014', 'B1', 'CA10', 'Ghar Karobar', 'O14', '3C', null, null, 'N', 'E4', 'Y', 'Y');

INSERT INTO FD_MSTR (FD_SER_NO, SF_NO, BRANCH_NO, ACCT_NO, TITLE, CORP_CUST_NO, CORP_CNST_TYPE, INTRO_CUST_NO, INTRO_ACCT_NO, INTRO_SIGN, VERI_EMP_NO, VERI_SIGN, MANAGER_SIGN)

VALUES ('FS5', 'SF-0015', 'B1', null, null, null, '0S', 'C7', 'SB6', 'Y', 'E4', 'Y', 'Y');

Output for each of the above INSERT INTO statements:

1 row created.

Insert the values into the FDSLAB_MSTR table (For values refer to 6th chapter under Test Records)

INSERT INTO FDSLAB_MSTR (FDSLAB_NO, MINPERIOD, MAXPERIOD, INTRATE)

VALUES(1, 1, 30, 5);

INSERT INTO FDSLAB_MSTR (FDSLAB_NO, MINPERIOD, MAXPERIOD, INTRATE)

VALUES(2, 31, 92, 5.5);

INSERT INTO FDSLAB_MSTR (FDSLAB_NO, MINPERIOD, MAXPERIOD, INTRATE)

VALUES(3, 93, 183, 6);

INSERT INTO FDSLAB_MSTR (FDSLAB_NO, MINPERIOD, MAXPERIOD, INTRATE)

VALUES(4, 184, 365, 6.5);

INSERT INTO FDSLAB_MSTR (FDSLAB_NO, MINPERIOD, MAXPERIOD, INTRATE)

VALUES(5, 366, 731, 7.5);

INSERT INTO FDSLAB_MSTR (FDSLAB_NO, MINPERIOD, MAXPERIOD, INTRATE)

VALUES(6, 732, 1097, 8.5);

INSERT INTO FDSLAB_MSTR (FDSLAB_NO, MINPERIOD, MAXPERIOD, INTRATE)

VALUES(7, 1098, 1829, 10);

Output for each of the above INSERT INTO statements:

1 row created.

Insert the values into the FD_DTLS table (For values refer to 6th chapter under Test Records)

INSERT INTO FD_DTLS (FD_SER_NO, FD_NO, TYPE, PAYTO_ACCTNO, PERIOD, OPNDT, DUEDT, AMT, DUEAMT, INTRATE, STATUS, AUTO_RENEWAL)

VALUES('FS1', 'F1', 'S', 'CA2', 365, '02-APR-2003', '01-APR-2004', 5000, 5350.00, 6.5, 'A', 'Y');

INSERT INTO FD_DTLS (FD_SER_NO, FD_NO, TYPE, PAYTO_ACCTNO, PERIOD, OPNDT, DUEDT, AMT, DUEAMT, INTRATE, STATUS, AUTO_RENEWAL)

VALUES('FS1', 'F2', 'S', 'CA2', 365, '02-APR-2003', '01-APR-2004', 5000, 5350.00, 6.5, 'A', 'N');

INSERT INTO FD_DTLS (FD_SER_NO, FD_NO, TYPE, PAYTO_ACCTNO, PERIOD, OPNDT, DUEDT, AMT, DUEAMT, INTRATE, STATUS, AUTO_RENEWAL)

VALUES('FS2', 'F3', 'S', 'CA4', 366, '25-MAY-2003', '25-MAY-2004', 10000, 10802.19, 7.5, 'A', 'Y');

INSERT INTO FD_DTLS (FD_SER_NO, FD_NO, TYPE, PAYTO_ACCTNO, PERIOD, OPNDT, DUEDT, AMT, DUEAMT, INTRATE, STATUS, AUTO_RENEWAL)

VALUES('FS2', 'F4', 'S', 'CA4', 366, '15-JUN-2003', '15-JUN-2004', 10000, 10802.19, 7.5, 'A', 'Y');

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
INSERT INTO FD_DTLS (FD_SER_NO, FD_NO, TYPE, PAYTO_ACCTNO, PERIOD, OPNDT, DUEDT, AMT, DUEAMT, INTRATE, STATUS, AUTO_RENEWAL)
VALUES('FS3', 'F5', 'S', 'CA7', 183, '24-JUN-2003', '24-DEC-2003', 2000, 2060.16, 6, 'A', 'Y');
```

```
INSERT INTO FD_DTLS (FD_SER_NO, FD_NO, TYPE, PAYTO_ACCTNO, PERIOD, OPNDT, DUEDT, AMT, DUEAMT, INTRATE, STATUS, AUTO_RENEWAL)
VALUES('FS4', 'F6', 'S', 'CA10', 732, '19-JUL-2003', '20-JUL-2005', 5000, 5902.47, 8.5, 'A', 'Y');
INSERT INTO FD_DTLS (FD_SER_NO, FD_NO, TYPE, PAYTO_ACCTNO, PERIOD, OPNDT, DUEDT, AMT, DUEAMT, INTRATE, STATUS, AUTO_RENEWAL)
VALUES('FS5', 'F7', 'S', 'SB6', 366, '27-JUL-2003', '27-JUL-2004', 5000, 5401.10, 7.5, 'A', 'N');
```

Output for each of the above INSERT INTO statements:

1 row created.

Insert the values into the ACCT_FD_CUST_DTLS table (For values refer to 6th chapter under Test Records)

```
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB1', 'C1');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA2', 'C2');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA2', 'C3');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB3', 'C4');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA4', 'C4');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA4', 'C5');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB5', 'C1');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB5', 'C4');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB6', 'C5');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB6', 'C7');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA7', 'C6');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA7', 'C8');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB8', 'C9');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB9', 'C3');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB9', 'C10');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA10', 'C10');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA10', 'C9');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS1', 'C2');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS1', 'C3');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS2', 'C4');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS2', 'C5');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS2', 'C5');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS3', 'C6');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS3', 'C8');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS4', 'C10');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS4', 'C9');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS5', 'C5');
```

Output for each of the above INSERT INTO statements:

1 row created.

Insert the values into the NOMINEE_MSTR table (For values refer to 6th chapter under Test Records)

```
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
VALUES('N1', 'CA2', 'Joseph Martin Dias', '17-SEP-1984', 'Colleague');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N2', 'CA2', 'Nilesh Sawant', '25-AUG-1987', 'Colleague');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N3', 'SB1', 'Chriselle Ivan Bayross', '25-JUN-1952', 'Daughter');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N4', 'SB3', 'Mamta Arvind Muzumdar', '28-AUG-1975', 'Friend');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N5', 'SB6', 'Preeti Suresh Shah', '12-FEB-1978', 'Friend');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N6', 'SB8', 'Rohit Rajan Sahakarkar', '30-MAY-1985', 'Relative');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N7', 'CA10', 'Namita S. Kanade', '10-JUN-1978', 'Niece');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N8', 'FS1', 'Rohit Rajan Sahakarkar', '30-MAY-1985', 'Relative');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N9', 'FS2', 'Joseph Martin Dias', '17-SEP-1984', 'Colleague');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N10', 'FS2', 'Nilesh Sawant', '25-AUG-1987', 'Colleague');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N11', 'FS3', 'Chriselle Ivan Bayross', '25-JUN-1952', 'Colleague');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N12', 'FS3', 'Mamta Arvind Muzumdar', '28-AUG-1975', 'Friend');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N13', 'FS4', 'Namita S. Kanade', '10-JUN-1978', 'Relative');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N14', 'FS5', 'Pramila P. Pius', '10-OCT-1985', 'Niece');
```

Output for each of the above INSERT INTO statements:

1 row created.

Insert the values into the ADDR_DTLS table (For values refer to 6th chapter under Test Records)

```
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE,
PINCODE)
```

```
VALUES(1, 'B1', 'H', 'A/5, Jay Chambers,', 'Service Road, Vile Parle (East)', 'Mumbai', 'Maharashtra', '400057');
```

```
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE,
PINCODE)
```

```
VALUES(2, 'B2', 'B', 'BSES Chambers, 10th floor,', 'Near Rly. Station, Andheri (West)', 'Mumbai', 'Maharashtra',
'400058');
```

```
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE,
PINCODE)
```

```
VALUES(3, 'B3', 'B', 'Prabhat Complex, No. 5 / 6,', 'Opp. Air India Bldg., Churchgate,', 'Mumbai', 'Maharashtra',
'400004');
```

```
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE,
PINCODE)
```

```
VALUES(4, 'B4', 'B', '23/A, Swarna Bldg., Smt. Rai Marg,', 'Eastern Express Highway, Kurla (East)', 'Mumbai',
'Maharashtra', '400045');
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(5, 'B5', 'B', 'Vikas Centre, Shop 37, Near National Park,', 'Western Express Highway, Borivali (East)', 'Mumbai', 'Maharashtra', '400078');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(6, 'B6', 'B', '24/A, Mahim Plaza, First Floor,', 'Mahim (West)', 'Mumbai', 'Maharashtra', '400016');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(7, 'E1', 'N', 'F-12, Diamond Palace, West Avenue,', 'North Avenue, Santacruz (West)', 'Mumbai', 'Maharashtra', '400056');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(8, 'E2', 'C', 'Desai House, Plot No. 25, P.G. Marg,', 'Near Malad Rly. Stat., Malad (West)', 'Mumbai', 'Maharashtra', '400078');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(9, 'E3', 'N', 'Room No. 56, 3rd Floor, Swamibhavan,', 'J. P. Road Junction, Andheri (East)', 'Mumbai', 'Maharashtra', '400059');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(10, 'E4', 'C', '301, Thomas Palace, Opp. Indu Child Care,', 'Yadnik Nagar, Andheri (West)', 'Mumbai', 'Maharashtra', '400058');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(11, 'E5', 'C', '456/A, Bldg. No. 4, Vahatuk Nagar,', 'Amboli, Andheri (West)', 'Mumbai', 'Maharashtra', '400058');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(12, 'E6', 'N', '201, Meena Towers, Nr. Sun Gas Agency,', 'S. V. Rd., Goregoan (West)', 'Mumbai', 'Maharashtra', '400076');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(13, 'E7', 'N', 'Patel Chawl, Rm. No. 15, B. P. Lal Marg,', 'Mahim (West)', 'Mumbai', 'Maharashtra', '400016');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(14, 'E8', 'C', 'A - 10, Neelam, L. J. Road,', 'Mahim (East)', 'Mumbai', 'Maharashtra', '400016');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(15, 'E9', 'N', '1/12 Bal Govindas Society, M. B. Raut Rd.', 'Dadar (East)', 'Mumbai', 'Maharashtra', '400028');

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(16, 'E10', 'C', 'Pathak Nagar, Cadal Road,', 'Mahim (West)', 'Mumbai', 'Maharashtra', '400016');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(17, 'C1', 'C', 'F-12, Diamond Palace, West Avenue,', 'North Avenue, Santacruz (West)', 'Mumbai', 'Maharashtra', '400056');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(18, 'C2', 'C', 'F-12, Silver Stream,', 'Santacruz (East)', 'Mumbai', 'Maharashtra', '400056');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(19, 'C3', 'C', 'Magesh Prasad,', 'Saraswati Baug, Jogeshwari(E)', 'Mumbai', 'Maharashtra', '400060');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(20, 'C4', 'C', '4, Sampada,', 'Kataria Road, Mahim,', 'Mumbai', 'Maharashtra', '400016');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(21, 'C5', 'C', '104, Vikram Apts. Bhagat Lane,', 'Shivaji Park, Mahim,', 'Mumbai', 'Maharashtra', '400016');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(22, 'C6', 'C', '12, Radha Kunj, N.C Kelkar Road,', 'Dadar,', 'Mumbai', 'Maharashtra', '400028');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(23, 'C7', 'C', 'A/14, Shanti Society, Mogal Lane,', 'Mahim,', 'Mumbai', 'Maharashtra', '400016');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(24, 'C8', 'C', '5, Vagdevi, Senapati Bapat Rd.', 'Dadar,', 'Mumbai', 'Maharashtra', '400016');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(25, 'C9', 'C', 'A-10 Nutan Vaishali,', 'Shivaji Park, Mahim,', 'Mumbai', 'Maharashtra', '400016');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(26, 'C10', 'C', 'B-10, Makarand Society,', 'Cadal Road, Mahim,', 'Mumbai', 'Maharashtra', '400016');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(27, 'N1', 'C', '307/E, Meena Mansion,', 'R. S. Road, Andheri (West)', 'Mumbai', 'Maharashtra', '400058');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(28, 'N2', 'C', 'Smt. Veenu Chawl, Sawant Colony Rd.', 'Opp. Veer Road, Matunga (West)', 'Mumbai', 'Maharashtra', '400016');

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(29, 'N3', 'C', 'F-12, Silver Stream,', 'Santacruz (East)', 'Mumbai', 'Maharashtra', '400056');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(30, 'N4', 'C', 'Magesh Prasad,', 'Saraswati Baug, Jogeshwari(E)', 'Mumbai', 'Maharashtra', '400060');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(31, 'N5', 'C', 'Rita Apartment, Room No. 46, 2nd Floor,', 'J. P. Road, Andheri (East)', 'Mumbai', 'Maharashtra', '400067');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(32, 'N6', 'N', '106/A, Sunrise Apmt., Opp. Vahatuk Nagar,', 'Kevni-Pada, Jogeshwari (West)', 'Mumbai', 'Maharashtra', '400102');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(33, 'N7', 'C', 'Pathak Nagar, Cadal Road,', 'Mahim (West)', 'Mumbai', 'Maharashtra', '400016');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(34, 'O11', 'H', 'Shop No. 4, Simon Streams,', 'V. P. Road, Andheri (West)', 'Mumbai', 'Maharashtra', '400058');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(35, 'O12', 'H', '230-E, Patel Chambers,', 'Service Road, Vile Parle (East)', 'Mumbai', 'Maharashtra', '400057');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(36, 'O13', 'H', 'G-2, Puru Hsg. Society,', 'Senapati Bapat Rd., Dadar,', 'Mumbai', 'Maharashtra', '400016');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(37, 'O14', 'H', 'B-10, Makarand Society,', 'Cadal Road, Mahim,', 'Mumbai', 'Maharashtra', '400016');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(38, 'N8', 'N', '106/A, Sunrise Apmt., Opp. Vahatuk Nagar,', 'Kevni-Pada, Jogeshwari (West)', 'Mumbai', 'Maharashtra', '400102');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(39, 'N9', 'C', '307/E, Meena Mansion,', 'R. S. Road, Andheri (West)', 'Mumbai', 'Maharashtra', '400058');

INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)

VALUES(40, 'N10', 'C', 'Smt. Veenu Chawl, Sawant Colony Rd.', 'Opp. Veer Road, Matunga (West)', 'Mumbai', 'Maharashtra', '400016');

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
```

```
VALUES(41, 'N11', 'C', 'F-12, Silver Stream,', 'Santacruz (East)', 'Mumbai', 'Maharashtra', '400056');
```

```
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
```

```
VALUES(42, 'N12', 'C', 'Magesh Prasad', 'Saraswati Baug, Jogeshwari(E)', 'Mumbai', 'Maharashtra', '400060');
```

```
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
```

```
VALUES(43, 'N13', 'C', 'Pathak Nagar, Cadal Road,', 'Mahim (West)', 'Mumbai', 'Maharashtra', '400016');
```

```
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
```

```
VALUES(44, 'N14', 'C', '405, Vahatuk Nagar, Kevni-Pada,', 'Jogeshwari (West)', 'Mumbai', 'Maharashtra', '400102');
```

Output for each of the above INSERT INTO statements:

1 row created.

Insert the values into the CNTC_DTLS table

```
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
```

```
VALUES(1, 'B1', 'O', '26124571');
```

```
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
```

```
VALUES(1, 'B1', 'F', '26124533');
```

```
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
```

```
VALUES(1, 'B1', 'E', 'admin_vileparle@bom2.vsnl.in');
```

```
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
```

```
VALUES(2, 'B2', 'O', '26790014');
```

```
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
```

```
VALUES(2, 'B2', 'E', 'admin_andheri@bom2.vsnl.in');
```

```
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
```

```
VALUES(3, 'B3', 'O', '23457855');
```

```
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
```

```
VALUES(3, 'B3', 'E', 'admin_churchgate@bom2.vsnl.in');
```

```
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
```

```
VALUES(4, 'B4', 'O', '25545455');
```

```
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
```

```
VALUES(4, 'B4', 'E', 'admin_sion@bom2.vsnl.in');
```

```
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

VALUES(5, 'B5', 'O', '28175454');

INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(5, 'B5', 'E', 'admin_borivali@bom2.vsnl.in');

INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(6, 'B6', 'O', '24304545');

INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(6, 'B6', 'E', 'admin_matunga@bom2.vsnl.in');

INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(7, 'E1', 'R', '26045953');

INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(8, 'E2', 'R', '28883779');

INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(9, 'E3', 'R', '28377634');

INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(10, 'E4', 'R', '26323560');

INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(11, 'E5', 'R', '26793231');

INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(12, 'E6', 'R', '28085654');

INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(13, 'E7', 'R', '24442342');

INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(14, 'E8', 'R', '24365672');

INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(15, 'E9', 'R', '24327349');

INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(16, 'E10', 'R', '24302579');

INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(17, 'C1', 'R', '26405853');

INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(17, 'C1', 'O', '26134553');

INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(17, 'C1', 'O', '26134571');

INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(17, 'C1', 'M', '9820178955');

INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(18, 'C2', 'R', '26045754');

INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(18, 'C2', 'O', '26134571');

INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(19, 'C3', 'R', '28324567');

INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
VALUES(19, 'C3', 'O', '26197654');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(20, 'C4', 'R', '24449852');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(20, 'C4', 'O', '28741370');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(21, 'C5', 'R', '24302934');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(21, 'C5', 'O', '22819964');

INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(22, 'C6', 'R', '24217592');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(23, 'C7', 'R', '24372247');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(24, 'C8', 'O', '26480903');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(25, 'C9', 'R', '24313408');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(25, 'C9', 'M', '9821176651');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(26, 'C10', 'R', '24362680');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(26, 'C10', 'O', '28973355');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(26, 'C10', 'M', '9820484648');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(27, 'N1', 'R', '26762154');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(28, 'N2', 'R', '24307887');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(29, 'N3', 'R', '260455754');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(30, 'N4', 'R', '28645489');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(31, 'N5', 'R', '30903564');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(32, 'N6', 'R', '26793771');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(33, 'N7', 'R', '24304455');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(34, 'O11', 'O', '26790055');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(34, 'O11', 'F', '26784409');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(35, 'O12', 'O', '26120455');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(35, 'O12', 'O', '26120456');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(35, 'O12', 'F', '26121450');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(35, 'O12', 'E', 'admin@sunpvtltd.com');
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(35, 'O12', 'W', 'www.sunpvtltd.com');
```

```
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(36, 'O13', 'O', '24301090');
```

```
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(36, 'O13', 'O', '24301196');
```

```
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(37, 'O14', 'O', '24321122');
```

```
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(38, 'N8', 'R', '26793771');
```

```
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(39, 'N9', 'R', '26762154');
```

```
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(40, 'N10', 'R', '24307887');
```

```
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(41, 'N11', 'R', '26045754');
```

```
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(42, 'N12', 'R', '28645489');
```

```
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(43, 'N13', 'R', '24304455');
```

```
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(44, 'N14', 'R', '26790180');
```

```
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA)
VALUES(44, 'N14', 'R', '26771275');
```

Output for each of the above INSERT INTO statements:

1 row created.

Insert the values into the TRANS_MSTR table (For values refer to 6th chapter under Test Records)

```
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T1', 'SB1', '05-JAN-2003', 'C', 'Initial Payment', 'D', 500, 500);
```

```
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T2', 'CA2', '10-JAN-2003', 'C', 'Initial Payment', 'D', 2000, 2000);
```

```
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T3', 'SB3', '22-JAN-2003', 'C', 'Initial Payment', 'D', 500, 500);
```

```
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T4', 'CA4', '05-FEB-2003', 'B', 'Initial Payment', 'D', 2000, 2000);
```

```
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T5', 'SB5', '15-FEB-2003', 'B', 'Initial Payment', 'D', 500, 500);
```

```
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T6', 'SB6', '27-FEB-2003', 'C', 'Initial Payment', 'D', 500, 500);
```

```
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T7', 'CA7', '14-MAR-2003', 'B', 'Initial Payment', 'D', 2000, 2000);
```

```
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T8', 'SB8', '29-MAR-2003', 'C', 'Initial Payment', 'D', 500, 500);
```

```
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T9', 'SB9', '05-APR-2003', 'C', 'Initial Payment', 'D', 500, 500);
```

```
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T10', 'SB9', '15-APR-2003', 'B', 'CLR-204907', 'D', 3000, 3500);
```

```
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
VALUES('T11', 'SB9', '17-APR-2003', 'C', 'Self', 'W', 2500, 1000);
```

```
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)  
VALUES('T12', 'CA10', '19-APR-2003', 'B', 'Initial Payment', 'D', 2000, 2000);
```

```
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)  
VALUES('T13', 'SB9', '05-JUN-2003', 'B', 'CLR-204908', 'D', 3000, 4000);
```

```
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)  
VALUES('T14', 'SB9', '27-JUN-2003', 'C', 'Self', 'W', 2500, 1500);
```

Output for each of the above INSERT INTO statements:

1 row created.

Insert the values into the TRANS_DTLS table (For values refer to 6th chapter under Test Records)

```
INSERT INTO TRANS_DTLS (TRANS_NO, INST_NO, INST_DT, PAYTO, INST_CLR_DT, BANK_NAME,  
BRANCH_NAME, PAIDFROM)
```

```
VALUES('T4', 098324, '02-FEB-2003', 'Self', '05-FEB-2003', 'HDFC', 'Vile Parle (East)', '2982');
```

```
INSERT INTO TRANS_DTLS (TRANS_NO, INST_NO, INST_DT, PAYTO, INST_CLR_DT, BANK_NAME,  
BRANCH_NAME, PAIDFROM)
```

```
VALUES('T5', 232324, '14-FEB-2003', 'Self', '15-FEB-2003', 'India Bank', 'Andheri (West)', '30434');
```

```
INSERT INTO TRANS_DTLS (TRANS_NO, INST_NO, INST_DT, PAYTO, INST_CLR_DT, BANK_NAME,  
BRANCH_NAME, PAIDFROM)
```

```
VALUES('T7', 434560, '14-MAR-2003', 'Self', '14-MAR-2003', 'ICICI Bank', 'Bandra (West)', '4882');
```

```
INSERT INTO TRANS_DTLS (TRANS_NO, INST_NO, INST_DT, PAYTO, INST_CLR_DT, BANK_NAME,  
BRANCH_NAME, PAIDFROM)
```

```
VALUES('T10', 204907, '14-APR-2003', 'Self', '15-APR-2003', 'Memon Co-operative Bank', 'Jogeshwari (West)',  
'1767');
```

```
INSERT INTO TRANS_DTLS (TRANS_NO, INST_NO, INST_DT, PAYTO, INST_CLR_DT, BANK_NAME,  
BRANCH_NAME, PAIDFROM)
```

```
VALUES('T12', 100907, '19-APR-2003', 'Self', '19-APR-2003', 'Memon Co-operative Bank', 'Jogeshwari (West)',  
'2001');
```

```
INSERT INTO TRANS_DTLS (TRANS_NO, INST_NO, INST_DT, PAYTO, INST_CLR_DT, BANK_NAME,  
BRANCH_NAME, PAIDFROM)
```

```
VALUES('T13', 204908, '01-JUN-2003', 'Self', '05-JUN-2003', 'Memon Co-operative Bank', 'Jogeshwari (West)',  
'1767');
```

Output for each of the above INSERT INTO statements:

1 row created.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

4. Relational Databases

The frustration with the inadequate capabilities of network and hierarchical databases resulted in the invention of the *relational data model*. The relational data model took the idea of the network database some several steps further. Relational models — just like hierarchical and network models — are based upon tables and use parent/child relationships. (Though this relationship was implemented through column values as opposed to a low-level physical pointer defining the relationship; more on that later in the chapter.)

Tables

A table is a basic building unit of the relational database. It is a fairly intuitive way of organizing data and has been around for centuries. A table consists of rows and columns (called *records* and *fields* in database jargon). Each table has a *unique* name in the database (i.e., unique *fully qualified name*, the one that includes schema or database name as a prefix).

Note The Dot (.) notation in a fully qualified name is commonly used in the programming world to describe hierarchy of the objects and their properties. This could refer not only to the database objects but also to the structures, user-defined types, and such. For example, a table field in an MS SQL Server database could be referred to as ACME.DBO.CUSTOMER.CUST_ID_N where ACME is a database name, DBO is the table owner (Microsoft standard), CUSTOMER is the name of the table, and CUST_ID_N is the column name in the CUSTOMER table.

Each field has a unique name within the table, and any table must have at least one field. The number of fields per table is usually limited, the actual limitation being dependent on a particular implementation. Unlike legacy database structure, records in a table are not stored or retrieved in any particular order, the task of sorting the record in relational databases systems (RDBMS) is relegated to SQL.

A record thus is composed of a number of cells, where each cell has a unique name and might contain some data. A table that has no records is called an empty table.

Data within the field must be of the same type, for example, the field AMOUNT contains only numbers, and field DESCRIPTION, only words. The set of the data within one field is said to be column's *domain*.

Note Early databases — relational or otherwise — were designed to contain only text data; modern databases store anything that could be converted into binary format: pictures, movies, audio records, and so on.

The good relational design would make sure that such a record describes an *entity* — another relational database term to be discussed later in the book but worth mentioning here. To put it in other words, the record should not contain irrelevant information: CUSTOMER table deals with the customer information only, its records should not contain information about, say, products that this customer ordered.

There is no theoretical limit on the number of rows a table could have, though some implementations impose restrictions; also there are (or at least ought to be) practical considerations to the limits: data retrieval speed, amount of storage, and so on.

Relationships

Tables in RDBMS might or might not be related. As it was mentioned before, RDBMS is built upon parent/child relationship notion (hence the name — *relational*), but unlike "in legacy databases (hierarchical, network) these relations are based solely on the values in the table columns; these relationships are meaningful in logical terms, not in low-level computer specific pointers. Let's take the example of our fictitious order entry database (the one that we will design, build, and use throughout the book). The ORDER_HEADER table is related to CUSTOMER table since both of these tables have a *common set of values*: The field ORDHDR_CUSTID_FN (customer ID) in ORDER_HEADER (and its values) corresponds to CUST_ID_N in CUSTOMER. The field CUST_ID_N is said to be a *primary key* for the CUSTOMER table and a *foreign key* for the ORDER_HEADER table (under different name).

Primary key

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

The *primary key* holds more than one job in RDBMS. We've said already that it is used to define a relationship; but its primary role is to uniquely identify each record in a table.

In the days of legacy databases, the records were always stored in some predefined order; if such an order had to be broken (because somebody had inserted records in a wrong order or business rule was changed), then the whole table (and, most likely, the whole database) had to be rebuilt. The RDBMS abolishes fixed order for the records, but it still needs some mechanism of identifying the records uniquely, and the primary key, based on the idea of a field (or fields) that contains set unique values, serves exactly this purpose.

By its very nature, the primary key cannot be empty; this means that in a table with defined primary key, the primary key fields must contain data for each record.

Note Though it is not a requirement to have a primary key on each and every table, it is considered to be a good practice to have one; in fact, many RDBMS implementations would warn you if you create a table without defining a primary key. Some purists go even further, specifying that the primary key should be *meaningless* in the sense that they would use some generated unique value (like EMPLOYEE_ID) instead of, say, Social Security numbers (despite that these are unique as well).

A primary key could consist of one or more columns, i.e., though some fields may contain duplicate values, their combination (set) is unique through the entire table. A key that consists of several columns is called a *composite key*.

Note In the world of RDBMS, only tables that have primary keys can be related. Though the primary key is a cornerstone for defining relation in RDBMS, the actual implementations (especially early ones) have not always provided a built-in support for this logical concept. In practice, the task of enforcing uniqueness of a chosen primary key was the responsibility of programmers (requiring them to check for existing values before inserting new records, for example). Today all major relational database products have built-in support for primary keys; on a very basic level this means that the database does its own checking for unique constraint violations and will raise an error whenever an attempt to insert a duplicate record is made.

Foreign key

Let's go back to our CUSTOMER and ORDER_HEADER tables. By now you understand why the CUST_ID_N was designated as a primary key — it has unique value, no customer can possibly have more than one ID, and no ID could be assigned to more than one customer. To track what customers placed which orders, you need something that will provide a link between customers and their orders.

Table ORDER_HEADER has its own primary key — ORDHDR_ID_N which uniquely identifies orders; in addition to that it will have a foreign key ORDHDR_CUSTID_FN field. The values in that field correspond to the values in the CUST_ID_N primary key field for the CUSTOMER table. Note that, unlike the primary key, the foreign key is not required to be unique — one customer could place several orders.

Now, by looking into ORDER_HEADER table you can find which customers placed particular orders. The table ORDER_HEADER became related to table CUSTOMER. It became easy to find a customer based on orders, or find orders for a customer. You no longer need to know database layout, order of the records in the table, or master some low-level proprietary programming language to query data; it's now possible to run ad-hoc queries formulated in standard English-like language — the Structured Query Language.

Invasion of RDBMS

In spite of the clear advantages of the relational database model, it took some time for it to become workable. One of the main reasons was the hardware. The logically clear and clean model proved to be quite a task to implement, and even then it required much more in terms of memory and processing power than legacy databases.

The development of relational databases was driven by the need of the medium to big businesses to gather, preserve, and analyze data. In 1965, Gordon Moore, the cofounder of Intel, made his famous observation that the number of transistors per square inch on the integrated circuits (IC) doubles every year ever since the IC was invented. Surprisingly, this rule still holds true. More powerful machines made it feasible to implement and sell RDBMS; cheap memory and powerful processors made them fast; perpetually growing appetites for information made RDBMS products a commodity, drastically

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

cutting their price down. Today, according to some estimates, less than 10 percent of the market is being held by the database legacy "dinosaurs" — mostly because of significant investment made by their owners more than 20 years ago. For better or for worse, relational database systems have come to rule on planet Earth.

5. Stating a database design problem.

Before I start with the list, let me be honest for a minute. I used to have a preacher who made sure to tell us before some sermons that he was preaching to himself as much as he was to the congregation. When I speak, or when I write an article, I have to listen to that tiny little voice in my head that helps filter out my own bad habits, to make sure that I am teaching only the best practices. Hopefully, after reading this article, the little voice in your head will talk to you when you start to stray from what is right in terms of database design practices.

So, the list:

- 1. Poor design/planning**
- 2. Ignoring normalization**
- 3. Poor naming standards**
- 4. Lack of documentation**
- 5. One table to hold all domain values**
- 6. Using identity/guid columns as your only key**
- 7. Not using SQL facilities to protect data integrity**
- 8. Not using stored procedures to access data**
- 9. Trying to build generic objects**
- 10. Lack of testing**

Poor design/planning

"If you don't know where you are going, any road will take you there" – George Harrison

Prophetic words for all parts of life and a description of the type of issues that plague many projects these days.

Let me ask you: would you hire a contractor to build a house and then demand that they start pouring a foundation the very next day? Even worse, would you demand that it be done without blueprints or house plans? Hopefully, you answered "no" to both of these. A design is needed make sure that the house you *want* gets built, and that the land you are building it on will not sink into some underground cavern. If you answered yes, I am not sure if anything I can say will help you.

Like a house, a good database is built with forethought, and with proper care and attention given to the needs of the data that will inhabit it; it cannot be tossed together in some sort of reverse implosion.

Since the database is the cornerstone of pretty much every business project, if you don't take the time to map out the needs of the project and how the database is going to meet them, then the chances are that the whole project will veer off course and lose direction. Furthermore, if you don't take the time at the start to get the database design right, then you'll find that any substantial changes in the database structures that you need to make further down the line could have a huge impact on the whole project, and greatly increase the likelihood of the project timeline slipping.

Far too often, a proper planning phase is ignored in favor of just "getting it done". The project heads off in a certain direction and when problems inevitably arise – due to the lack of proper designing and planning – there is "no time" to go back and fix them properly, using proper techniques. That's when the "hacking" starts, with the veiled promise to go back and fix things later, something that happens very rarely indeed.

Admittedly it is impossible to predict every need that your design will have to fulfill and every issue that is likely to arise, but it is important to mitigate against potential problems as much as possible, by careful planning.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Ignoring Normalization

Normalization defines a set of methods to break down tables to their constituent parts until each table represents one and only one "thing", and its columns serve to fully describe only the one "thing" that the table represents.

The concept of normalization has been around for 30 years and is the basis on which SQL and relational databases are implemented. In other words, SQL was created to work with normalized data structures. Normalization is **not** just some plot by database programmers to annoy application programmers (that is merely a satisfying side effect!)

SQL is very additive in nature in that, if you have bits and pieces of data, it is easy to build up a set of values or results. In the **FROM** clause, you take a set of data (a table) and add (JOIN) it to another table. You can add as many sets of data together as you like, to produce the final set you need.

This additive nature is extremely important, not only for ease of development, but also for performance. Indexes are most effective when they can work with the entire key value. Whenever you have to use **SUBSTRING**, **CHARINDEX**, **LIKE**, and so on, to parse out a value that is combined with other values in a single column (for example, to split the last name of a person out of a full name column) the SQL paradigm starts to break down and data becomes become less and less searchable.

So normalizing your data is essential to good performance, and ease of development, but the question always comes up: "How normalized is normalized *enough*?" If you have read any books about normalization, then you will have heard many times that 3rd Normal Form is essential, but 4th and 5th Normal Forms are really useful and, once you get a handle on them, quite easy to follow and well worth the time required to implement them.

In reality, however, it is quite common that not even the first Normal Form is implemented correctly.

Whenever I see a table with repeating column names appended with numbers, I cringe in horror. And I cringe in horror quite often. Consider the following example **Customer** table:

Customer

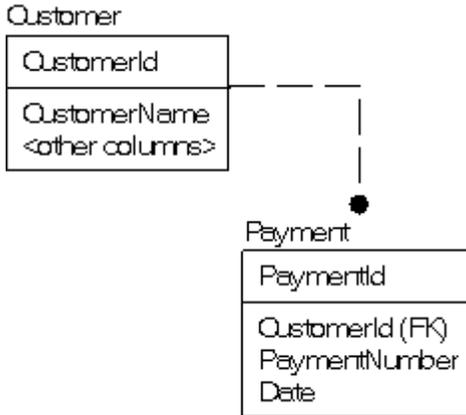
CustomerId
CustomerName
<other columns>
Payment1
Payment2
Payment3
Payment4
Payment5
Payment6
Payment7
Payment8
Payment9
Payment10
Payment11
Payment12

Are there always 12 payments? Is the order of payments significant? Does a NULL value for a payment mean UNKNOWN (not filled in yet), or a missed payment? And when was the payment made?!?

A payment does not describe a **Customer** and should not be stored in the **Customer** table. Details of payments should be stored in a **Payment** table, in which you could also record extra information about the payment, like when the payment was made, and what the payment was for:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual



In this second design, each column stores a single unit of information about a single "thing" (a payment), and each row represents a specific instance of a payment.

This second design is going to require a bit more code early in the process but, it is far more likely that you will be able to figure out what is going on in the system without having to hunt down the original programmer and kick their butt...sorry... figure out what they were thinking

Poor naming standards

"That which we call a rose, by any other name would smell as sweet"

This quote from Romeo and Juliet by William Shakespeare sounds nice, and it is true from one angle. If everyone agreed that, from now on, a rose was going to be called dung, then we could get over it and it would smell just as sweet. The problem is that if, when building a database for a florist, the designer calls it dung and the client calls it a rose, then you are going to have some meetings that sound far more like an Abbott and Costello routine than a serious conversation about storing information about horticulture products.

Names, while a personal choice, are the first and most important line of documentation for your application. I will not get into all of the details of how best to name things here– it is a large and messy topic. What I want to stress in this article is the need for **consistency**. The names you choose are not just to enable you to identify the purpose of an object, but to allow all future programmers, users, and so on to quickly and easily understand how a component part of your database was intended to be used, and what data it stores. No future user of your design should need to wade through a 500 page document to determine the meaning of some wacky name.

Consider, for example, a column named, **X304_DSCR**. What the heck does that mean? You might decide, after some head scratching, that it means "X304 description". Possibly it does, but maybe **DSCR** means discriminator, or discretizator?

Unless you have established **DSCR** as a corporate standard abbreviation for description, then **X304_DESCRIPTION** is a much better name, and one leaves nothing to the imagination.

That just leaves you to figure out what the **X304** part of the name means. On first inspection, to me, X304 sounds like more like it should be data in a column rather than a column name. If I subsequently found that, in the organization, there was also an X305 and X306 then I would flag that as an issue with the database design. For maximum flexibility, data is stored in columns, not in column names.

Along these same lines, resist the temptation to include "metadata" in an object's name. A name such as **tblCustomer** or **colVarcharAddress** might seem useful from a development perspective, but to the end user it is just confusing. As a developer,

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

you should rely on being able to determine that a table name is a table name by context in the code or tool, and present to the users clear, simple, descriptive names, such as **Customer** and **Address**.

A practice I strongly advise against is the use of spaces and quoted identifiers in object names. You should avoid column names such as "Part Number" or, in Microsoft style, [Part Number], therefore requiring you users to include these spaces and identifiers in their code. It is annoying and simply unnecessary.

Acceptable alternatives would be **part_number**, **partNumber** or **PartNumber**. Again, consistency is key. If you choose **PartNumber** then that's fine – as long as the column containing invoice numbers is called **InvoiceNumber**, and not one of the other possible variations.

Lack of documentation

I hinted in the intro that, in some cases, I am writing for myself as much as you. This is the topic where that is most true. By carefully naming your objects, columns, and so on, you can make it clear to anyone what it is that your database is modeling. However, this is only step one in the documentation battle. The unfortunate reality is, though, that "step one" is all too often the *only* step.

Not only will a well-designed data model adhere to a solid naming standard, it will also contain definitions on its tables, columns, relationships, and even default and check constraints, so that it is clear to everyone how they are intended to be used. In many cases, you may want to include sample values, where the need arose for the object, and anything else that you may want to know in a year or two when "future you" has to go back and make changes to the code.

NOTE:

Where this documentation is stored is largely a matter of corporate standards and/or convenience to the developer and end users. It could be stored in the database itself, using extended properties. Alternatively, it might be maintained in the data modeling tools. It could even be in a separate data store, such as Excel or another relational database. My company maintains a metadata repository database, which we developed in order to present this data to end users in a searchable, linkable format. Format and usability is important, but the primary battle is to have the information available and up to date.

Your goal should be to provide enough information that when you turn the database over to a support programmer, they can figure out your minor bugs and fix them (yes, we all make bugs in our code!). I know there is an old joke that poorly documented code is a synonym for "job security." While there is a hint of truth to this, it is also a way to be hated by your coworkers and never get a raise. And no good programmer I know of wants to go back and rework their own code years later. It is best if the bugs in the code can be managed by a junior support programmer while you create the next new thing. Job security along with raises is achieved by being the go-to person for new challenges.

One table to hold all domain values

"One Ring to rule them all and in the darkness bind them"

This is all well and good for fantasy lore, but it's not so good when applied to database design, in the form of a "ruling" domain table. Relational databases are based on the fundamental idea that every object represents one and only one thing. There should never be any doubt as to what a piece of data refers to. By tracing through the relationships, from column name, to table name, to primary key, it should be easy to examine the relationships and know exactly what a piece of data means.

The big myth perpetrated by architects who don't really understand relational database architecture (me included early in my career) is that the more tables there are, the more complex the design will be. So, conversely, shouldn't condensing multiple tables into a single "catch-all" table simplify the design? It does sound like a good idea, but at one time giving Pauly Shore the lead in a movie sounded like a good idea too.

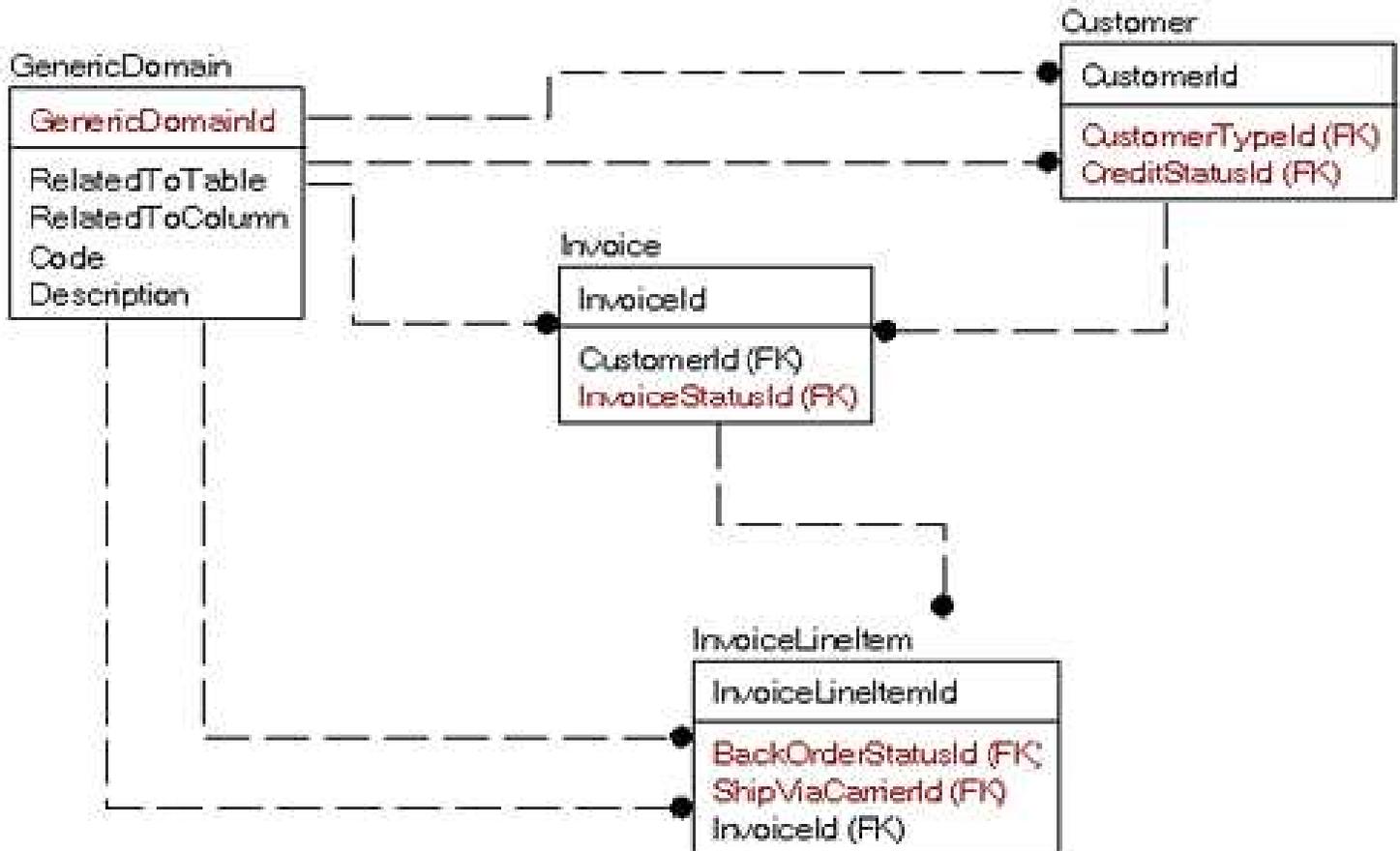
UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

For example, consider the following model snippet where I needed domain values for:

-) Customer CreditStatus
-) Customer Type
-) Invoice Status
-) Invoice Line Item BackOrder Status
-) Invoice Line Item Ship Via Carrier

On the face of it that would be five domain tables...but why not just use one generic domain table, like this?



This may seem a very clean and natural way to design a table for all but the problem is that it is just not very natural to work with in SQL. Say we just want the domain values for the **Customer** table:

```
SELECT *
FROM Customer
JOIN GenericDomain as CustomerType
  ON Customer.CustomerTypeId = CustomerType.GenericDomainId
  and CustomerType.RelatedToTable = 'Customer'
  and CustomerType.RelatedToColumn = 'CustomerTypeId'
JOIN GenericDomain as CreditStatus
  ON Customer.CreditStatusId = CreditStatus.GenericDomainId
```

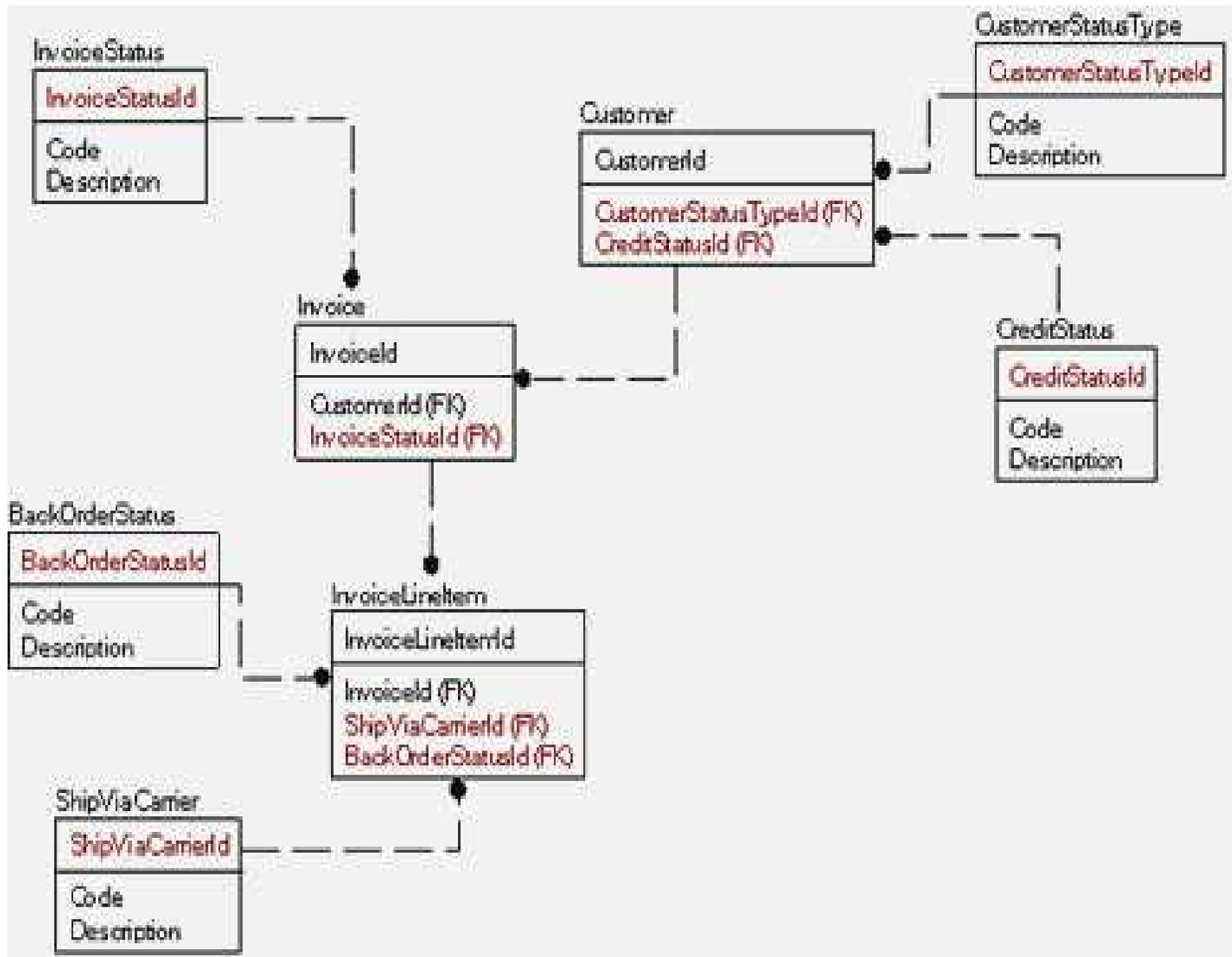
UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

and CreditStatus.RelatedToTable = 'Customer'
and CreditStatus.RelatedToColumn = ' CreditStatusId'

As you can see, this is far from being a natural join. It comes down to the problem of mixing apples with oranges. At first glance, domain tables are just an abstract concept of a container that holds text. And from an implementation centric standpoint, this is quite true, but it is not the correct way to build a database. In a database, the process of normalization, as a means of breaking down and isolating data, takes every table to the point where one row represents one thing. And each domain of values is a distinctly different thing from all of the other domains (unless it is not, in which case the one table will suffice.). So what you do, in essence, is normalize the data on each usage, spreading the work out over time, rather than doing the task once and getting it over with.

So instead of the single table for all domains, you might model it as:



Looks harder to do, right? Well, it is initially. Frankly it took me longer to flesh out the example tables. But, there are quite a few tremendous gains to be had:

-) Using the data in a query is much easier:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
SELECT *
FROM Customer
  JOIN CustomerType
    ON Customer.CustomerTypeId = CustomerType.CustomerTypeId
  JOIN CreditStatus
    ON Customer.CreditStatusId = CreditStatus.CreditStatusId
```

-) Data can be validated using foreign key constraints very naturally, something not feasible for the other solution unless you implement ranges of keys for every table – a terrible mess to maintain.
-) If it turns out that you need to keep more information about a **ShipViaCarrier** than just the code, 'UPS', and description, 'United Parcel Service', then it is as simple as adding a column or two. You could even expand the table to be a full blown representation of the businesses that are carriers for the item.
-) All of the smaller domain tables will fit on a single page of disk. This ensures a single read (and likely a single page in cache). If the other case, you might have your domain table spread across many pages, unless you cluster on the referring table name, which then could cause it to be more costly to use a non-clustered index if you have many values.
-) You can still have one editor for all rows, as most domain tables will likely have the same base structure/usage. And while you would lose the ability to query all domain values in one query easily, why would you want to? (A union query could easily be created of the tables easily if needed, but this would seem an unlikely need.)

I should probably rebut the thought that might be in your mind. "What if I need to add a new column to all domain tables?" For example, you forgot that the customer wants to be able to do custom sorting on domain values and didn't put anything in the tables to allow this. This is a fair question, especially if you have 1000 of these tables in a very large database. First, this rarely happens, and when it does it is going to be a major change to your database in either way.

Second, even if this became a task that was required, SQL has a complete set of commands that you can use to add columns to tables, and using the system tables it is a pretty straightforward task to build a script to add the same column to hundreds of tables all at once. That will not be as easy of a change, but it will not be so much more difficult to outweigh the large benefits.

The point of this tip is simply that it is better to do the work upfront, making structures solid and maintainable, rather than trying to attempt to do the least amount of work to start out a project. By keeping tables down to representing one "thing" it means that most changes will only affect one table, after which it follows that there will be less rework for you down the road.

Using identity/guid columns as your only key

First Normal Form dictates that all rows in a table must be uniquely identifiable. Hence, every table should have a primary key. SQL Server allows you to define a numeric column as an **IDENTITY** column, and then automatically generates a unique value for each row. Alternatively, you can use **NEWID()** (or **NEWSEQUENTIALID()**) to generate a random, 16 byte unique value for each row. These types of values, when used as keys, are what are known as **surrogate keys**. The word surrogate means "something that substitutes for" and in this case, a surrogate key should be the stand-in for a natural key.

The problem is that too many designers use a surrogate key column as the *only* key column on a given table. The surrogate key values have no actual meaning in the real world; they are just there to uniquely identify each row.

Now, consider the following **Part** table, whereby **PartID** is an **IDENTITY** column and is the primary key for the table:

PartID	PartNumber	Description
1	XXXXXXXXX	The X part
2	XXXXXXXXX	The X part
3	YYYYYYYYY	The Y part

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

How many rows are there in this table? Well, there seem to be three, but are rows with **PartIDs** 1 and 2 actually the same row, duplicated? Or are they two different rows that should be unique but were keyed in incorrectly?

The rule of thumb I use is simple. If a human being could not pick which row they want from a table without knowledge of the surrogate key, then you need to reconsider your design. This is why there should be a key of some sort on the table to guarantee uniqueness, in this case likely on **PartNumber**.

In summary: as a rule, each of your tables should have a natural key that means something to the user, and can uniquely identify each row in your table. In the very rare event that you cannot find a natural key (perhaps, for example, a table that provides a log of events), then use an artificial/surrogate key.

Not using SQL facilities to protect data integrity

All fundamental, non-changing business rules should be implemented by the relational engine. The **base rules** of nullability, string length, assignment of foreign keys, and so on, should all be defined **in the database**.

There are many different ways to import data into SQL Server. If your base rules are defined in the database itself can you guarantee that they will never be bypassed and you can write your queries without ever having to worry whether the data you're viewing adheres to the base business rules.

Rules that are optional, on the other hand, are wonderful candidates to go into a business layer of the application. For example, consider a rule such as this: "For the first part of the month, no part can be sold at more than a 20% discount, without a manager's approval".

Taken as a whole, this rule smacks of being rather messy, not very well controlled, and subject to frequent change. For example, what happens when next week the maximum discount is 30%? Or when the definition of "first part of the month" changes from 15 days to 20 days? Most likely you won't want go through the difficulty of implementing these complex temporal business rules in SQL Server code – the business layer is a great place to implement rules like this.

However, consider the rule a little more closely. There are elements of it that will probably never change. E.g.

-) The maximum discount it is ever possible to offer
-) The fact that the approver must be a manager

These aspects of the business rule very much ought to get enforced by the database and design. Even if the substance of the rule is implemented in the business layer, you are still going to have a table in the database that records the size of the discount, the date it was offered, the ID of the person who approved it, and so on. On the **Discount** column, you should have a **CHECK** constraint that restricts the values allowed in this column to between 0.00 and 0.90 (or whatever the maximum is). Not only will this implement your "maximum discount" rule, but will also guard against a user entering a 200% or a negative discount by mistake. On the **ManagerID** column, you should place a foreign key constraint, which reference the Managers table and ensures that the ID entered is that of a real manager (or, alternatively, a trigger that selects only **EmployeeIDs** corresponding to managers).

Now, at the very least we can be sure that the data meets the very basic rules that the data must follow, so we never have to code something like this in order to check that the data is good:

```
SELECT CASE WHEN discount < 0 then 0 else WHEN discount > 1 then 1...
```

We can feel safe that data meets the basic criteria, every time.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Not using stored procedures to access data

Stored procedures are your friend. Use them whenever possible as a method to insulate the database layer from the users of the data. Do they take a bit more effort? Sure, initially, but what good thing doesn't take a bit more time? Stored procedures make database development much cleaner, and encourage collaborative development between your database and functional programmers. A few of the other interesting reasons that stored procedures are important include the following.

Maintainability

Stored procedures provide a known interface to the data, and to me, this is probably the largest draw. When code that accesses the database is compiled into a different layer, performance tweaks cannot be made without a functional programmer's involvement. Stored procedures give the database professional the power to change characteristics of the database code without additional resource involvement, making small changes, or large upgrades (for example changes to SQL syntax) easier to do.

Encapsulation

Stored procedures allow you to "encapsulate" any structural changes that you need to make to the database so that the knock on effect on user interfaces is minimized. For example, say you originally modeled one phone number, but now want an unlimited number of phone numbers. You could leave the single phone number in the procedure call, but store it in a different table as a stopgap measure, or even permanently if you have a "primary" number of some sort that you always want to display. Then a stored proc could be built to handle the other phone numbers. In this manner the impact to the user interfaces could be quite small, while the code of stored procedures might change greatly.

Security

Stored procedures can provide specific and granular access to the system. For example, you may have 10 stored procedures that all update table X in some way. If a user needs to be able to update a particular column in a table and you want to make sure they never update any others, then you can simply grant to that user the permission to execute just the one procedure out of the ten that allows them perform the required update.

Performance

There are a couple of reasons that I believe stored procedures enhance performance. First, if a newbie writes ratty code (like using a cursor to go row by row through an entire ten million row table to find one value, instead of using a WHERE clause), the procedure can be rewritten without impact to the system (other than giving back valuable resources.) The second reason is plan reuse. Unless you are using dynamic SQL calls in your procedure, SQL Server can store a plan and not need to compile it every time it is executed. It's true that in every version of SQL Server since 7.0 this has become less and less significant, as SQL Server gets better at storing plans ad hoc SQL calls (see note below). However, stored procedures still make it easier for plan reuse and performance tweaks. In the case where ad hoc SQL would actually be faster, this can be coded into the stored procedure seamlessly.

In 2005, there is a database setting (**PARAMETERIZATION FORCED**) that, when enabled, will cause all queries to have their plans saved. This does not cover more complicated situations that procedures would cover, but can be a big help. There is also a feature known as **plan guides**, which allow you to override the plan for a known query type. Both of these features are there to help out when stored procedures are not used, but stored procedures do the job with no tricks.

And this list could go on and on. There are drawbacks too, because nothing is ever perfect. It can take longer to code stored procedures than it does to just use ad hoc calls. However, the amount of time to design your interface and implement it is well worth it, when all is said and done.

Trying to code generic T-SQL objects

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

I touched on this subject earlier in the discussion of generic domain tables, but the problem is more prevalent than that. Every new T-SQL programmer, when they first start coding stored procedures, starts to think "I wish I could just pass a table name as a parameter to a procedure." It does sound quite attractive: one generic stored procedure that can perform its operations on any table you choose. However, this should be avoided as it can be very detrimental to performance and will actually make life more difficult in the long run.

T-SQL objects do not do "generic" easily, largely because lots of design considerations in SQL Server have clearly been made to facilitate reuse of plans, not code. SQL Server works best when you minimize the unknowns so it can produce the best plan possible. The more it has to generalize the plan, the less it can optimize that plan.

Note that I am not specifically talking about dynamic SQL procedures. Dynamic SQL is a great tool to use when you have procedures that are not optimizable / manageable otherwise. A good example is a search procedure with many different choices. A precompiled solution with multiple OR conditions might have to take a worst case scenario approach to the plan and yield weak results, especially if parameter usage is sporadic.

However, the main point of this tip is that you should avoid coding very generic objects, such as ones that take a table name and twenty column names/value pairs as a parameter and lets you update the values in the table. For example, you could write a procedure that started out:

```
CREATE PROCEDURE updateAnyTable
@tableName sysname,
@columnName1 sysname,
@columnName1Value varchar(max)
@columnName2 sysname,
@columnName2Value varchar(max)
...
```

The idea would be to dynamically specify the name of a column and the value to pass to a SQL statement. This solution is no better than simply using ad hoc calls with an UPDATE statement. Instead, when building stored procedures, you should build specific, dedicated stored procedures for each task performed on a table (or multiple tables.) This gives you several benefits:

-) Properly compiled stored procedures can have a single compiled plan attached to it and reused.
-) Properly compiled stored procedures are more secure than ad-hoc SQL or even dynamic SQL procedures, reducing the surface area for an injection attack greatly because the only parameters to queries are search arguments or output values.
-) Testing and maintenance of compiled stored procedures is far easier to do since you generally have only to search arguments, not that tables/columns/etc exist and handling the case where they do not

A nice technique is to build a code generation tool in your favorite programming language (even T-SQL) using SQL metadata to build very specific stored procedures for every table in your system. Generate all of the boring, straightforward objects, including all of the tedious code to perform error handling that is so essential, but painful to write more than once or twice.

In my Apress book, Pro SQL Server 2005 Database Design and Optimization, I provide several such "templates" (manly for triggers, abut also stored procedures) that have all of the error handling built in, I would suggest you consider building your own (possibly based on mine) to use when you need to manually build a trigger/procedure or whatever.

Lack of testing

When the dial in your car says that your engine is overheating, what is the first thing you blame? The engine. Why don't you immediately assume that the dial is broken? Or something else minor? Two reasons:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

-) The engine is the most important component of the car and it is common to blame the most important part of the system first.
-) It is all too often true.

As database professionals know, the first thing to get blamed when a business system is running slow is the database. Why? First because it is the central piece of most any business system, and second because it also is all too often true.

We can play our part in dispelling this notion, by gaining deep knowledge of the system we have created and understanding its limits through **testing**.

But let's face it; testing is the first thing to go in a project plan when time slips a bit. And what suffers the most from the lack of testing? Functionality? Maybe a little, but users will notice and complain if the "Save" button doesn't actually work and they cannot save changes to a row they spent 10 minutes editing. What really gets the shaft in this whole process is deep system testing to make sure that the design you (presumably) worked so hard on at the beginning of the project is actually implemented correctly.

But, you say, the users accepted the system as working, so isn't that good enough? The problem with this statement is that what user acceptance "testing" usually amounts to is the users poking around, trying out the functionality that they understand and giving you the thumbs up if their little bit of the system works. Is this reasonable testing? Not in any other industry would this be vaguely acceptable. Do you want your automobile tested like this? "Well, we drove it slowly around the block once, one sunny afternoon with no problems; it is good!" When that car subsequently "failed" on the first drive along a freeway, or during the first drive through rain or snow, then the driver would have every right to be very upset.

Too many database systems get tested like that car, with just a bit of poking around to see if individual queries and modules work. The first real test is in production, when users attempt to do real work. This is especially true when it is implemented for a single client (even worse when it is a corporate project, with management pushing for completion more than quality).

Initially, major bugs come in thick and fast, especially performance related ones. If the first time you have tried a full production set of users, background process, workflow processes, system maintenance routines, ETL, etc, is on your system launch day, you are extremely likely to discover that you have not anticipated all of the locking issues that might be caused by users creating data while others are reading it, or hardware issues caused by poorly set up hardware. It can take weeks to live down the cries of "SQL Server can't handle it" even after you have done the proper tuning.

Once the major bugs are squashed, the fringe cases (which are pretty rare cases, like a user entering a negative amount for hours worked) start to raise their ugly heads. What you end up with at this point is software that irregularly fails in what seem like weird places (since large quantities of fringe bugs will show up in ways that aren't very obvious and are really hard to find.)

Now, it is far harder to diagnose and correct because now you have to deal with the fact that users are working with live data and trying to get work done. Plus you probably have a manager or two sitting on your back saying things like "when will it be done?" every 30 seconds, even though it can take days and weeks to discover the kinds of bugs that result in minor (yet important) data aberrations. Had proper testing been done, it would never have taken weeks of testing to find these bugs, because a proper test plan takes into consideration all possible types of failures, codes them into an automated test, and tries them over and over. Good testing won't find all of the bugs, but it will get you to the point where most of the issues that correspond to the original design are ironed out.

If everyone insisted on a strict testing plan as an integral and immutable part of the database development process, then maybe someday the database won't be the first thing to be fingered when there is a system slowdown.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Summary

Database design and implementation is the cornerstone of any data centric project (read 99.9% of business applications) and should be treated as such when you are developing. This article, while probably a bit preachy, is as much a reminder to me as it is to anyone else who reads it. Some of the tips, like planning properly, using proper normalization, using a strong naming standards and documenting your work– these are things that even the best DBAs and data architects have to fight to make happen. In the heat of battle, when your manager's manager's manager is being berated for things taking too long to get started, it is not easy to push back and remind them that they pay you now, or they pay you later. These tasks pay dividends that are very difficult to quantify, because to quantify success you must fail first. And even when you succeed in one area, all too often other minor failures crop up in other parts of the project so that some of your successes don't even get noticed.

The tips covered here are ones that I have picked up over the years that have turned me from being mediocre to a good data architect/database programmer. None of them take extraordinary amounts of time (except perhaps design and planning) but they all take more time upfront than doing it the "easy way". Let's face it, if the easy way were that easy in the long run, I for one would abandon the harder way in a second. It is not until you see the end result that you realize that success comes from starting off right as much as finishing right.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

6. Preparing ER diagram & DFD

An Entity-Relationship diagram is a visual representation of the structure of a database. An E-R diagram visually specifies all entities, primary keys, foreign keys, artificial keys and secondary keys. In addition, a dashed line defines relationships and a dot at the end of a dashed line indicates the "many" part of a one-to-many relationship.

In software engineering, an entity-relationship model (ERM) is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database, and its requirements in a top-down fashion. Diagrams created by this process are called entity-relationship diagrams, ER diagrams, or ERDs.

There are many ER diagramming tools. Some free software ER diagramming tools that can interpret and generate ER models, SQL and do database analysis are MySQL Workbench and DBDesigner (open-source). A freeware ER tool that can generate database and application layer code (webservices) is the RISE Editor.

Some of the proprietary ER diagramming tools are ARIS, Avolution, dbForge Studio for MySQL, DeZign for Databases, ER/Studio, Devgems Data Modeler, ERwin, MEGA International, OmniGraffle, Oracle Designer, PowerDesigner, Rational Rose, Sparx Enterprise Architect, SQLyog, System Architect, Toad Data Modeler, SQL Maestro, Microsoft Visio, Visible Analyst, and Visual Paradigm.

Some free software diagram tools just draw the shapes without having any knowledge of what they mean, nor do they generate SQL. These include Kivio and Dia. DIA diagrams, however, can be translated with tedia2sql.

Open Paint. Click "All Programs," click "Accessories," and then click "Paint." Paint is the graphic software application that is included with the Vista Operating System.

Open a New file. Click "File," then click "New."

Select a Rectangle symbol. Go to the Paint toolbox and click the "Rectangle symbol," then go to the canvas and with your mouse draw the "Rectangle symbol."

Identify the Customer Entity. Inside the rectangle, type "CustomerNum," as the primary key of the Customer entity, then draw a horizontal line. Inside the rectangle, type "LastName(SK)," skip a line then type "FirstName(SK);" the (SK) represents the secondary keys.

Identify the Sales Entity. Inside the rectangle, type "SalesID," as the primary key

of the Sales, then draw a horizontal line. Inside the rectangle, type "SalesDate," skip a line then type "JewelryPiece," skip a line then type "CustomerNum(FK);" the (FK) represents the foreign key.

Identify the relationship between the Customer and Sales Entities. Go to the Paint toolbox and click the "Line symbol," then go to the canvas and draw dashes between the Customer Entity and the Sales Entity

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

DFD

A DFD contains four kinds of symbol:

1. **Processes** -- The only *active* elements. Processes cause something to happen. They have embedded descriptions, often in verb-object form. (Sometimes informally called "bubbles" because of their shape in an early version of SA.)
2. **Terminators** -- Represent users or other systems, i.e. entities outside the boundary of the system being described.
3. **Dataflows** -- Composite data items (or **objects**) that pass either
 - o from any element to a process (input dataflow) or
 - o from a process to any element (output dataflow)
4. **Data stores** -- Holding places for dataflows; often implemented by databases.

Each symbol is *labelled* with a description in English or another natural language.

Some common-sense rules

Systems analysts apply this checklist to look for errors in their DFDs:

1. Every process must have at least one input dataflow (Violators are called "magic" processes, since they claim to do something based on no input, not even a trigger.)
2. Every process must have at least one output dataflow (Violators are called "black hole" processes, since their inputs are swallowed up for no reason.)
3. Every dataflow must connect two elements. One of them must be a process; the other can be a terminator, a data store or another process.
4. Each dataflow diagram should contain no more than six or seven processes and no more than six or seven data stores, and all the processes should be conceptually at the same level of detail. If a part of the system is too big or too complicated to describe in an easily grasped diagram, break it down into two or three lower-level diagrams. (We sometimes see hanging on an office wall a huge *tour de force* DFD that tries to describe an entire large system at a low level of detail with several dozen processes and convoluted intersecting dataflow arrows. That's not something to be proud of. It doesn't communicate to any audience.)
5. For every process, one of the following must eventually be true:
 - a. The description label is so simple and unambiguous that every reader will understand it in exactly the same way.
 - b. It is expanded or decomposed into a separate lower-level dataflow diagram that preserves exactly the same net inputs and outputs, but shows internal detail, such as data stores and internal processes.
 - c. It is rigorously described by a separate process specification (business rule, decision rule, function definition, algorithm, etc.).

The starting point: Context (level-0) diagram

The systems analyst begins by preparing the top-level DFD. This "context diagram" shows the entire system as a single process. Interactions with users and other external entities are shown as dataflows.

The context diagram, although often almost trivially simple, serves two essential purposes:

- J It clarifies to the user audience the analyst's understanding of the *scope* of the proposed system, the kinds of users the system will have, and the data coming out from and going into the system. A surprising number of misunderstandings are exposed at this early stage.
- J It motivates and establishes a framework for the more complicated next level (below).

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

The system diagram (level-1 DFD)

After everyone agrees that the context diagram is correct and complete, the systems analyst examines the first-level breakdown of major functions. Most systems can be decomposed into between two and seven major areas.

The result is called the "system diagram". It gives a clear overview of the system and serves as a base for further decomposition.

The end

The dataflow diagrams are complete when:

-) Every process on every DFD complies with rule number 5 above.
-) Every dataflow shown on every DFD is defined in the data dictionary.

There's more to come, but the remaining components of the system specification (or detailed user requirements documentation) have little or no effect on the *functionality* of the proposed system. Note that the information contained in these documents is essential not only as a foundation for building a custom application but also as a basis for evaluating and choosing a packaged application software product.

DATA FLOW DIAGRAMS

Systems Analysis

- Focus is the *logical* view of the system, not the physical
- "What" the system is to accomplish, not how

- Tools:

- data flow diagrams
- data dictionary
- process specification
- entity-relationship diagrams

Data Flow Diagram:

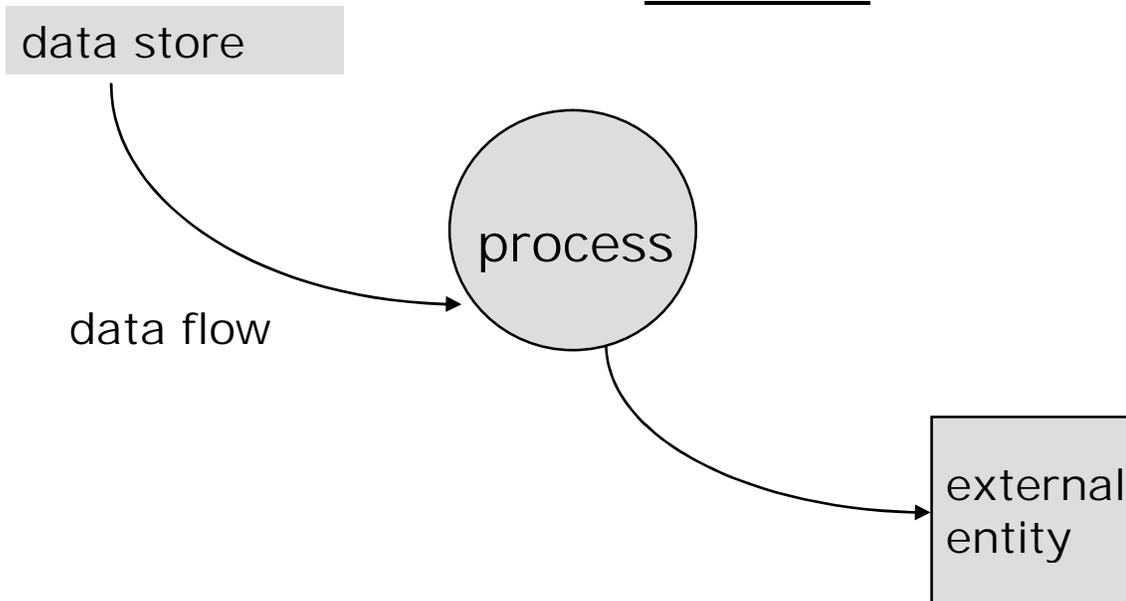
"a network representation of a system. The system may be automated, manual, or mixed. The DFD portrays the system in terms of its component pieces, with all interfaces among the components indicated."

- Tom DeMarco

hence DFDs:

focus on the *movement* of data between external entities and processes, and between processes and data stores

Example Data Flow Diagram



Data Flow Diagrams are:

- Used to perform structured analysis to determine logical requirements
- A graphical tool, useful for communicating with users, managers, and other IS personnel
- Useful for analyzing existing as well as proposed systems
- A relatively simple technique to learn and use

Why Conduct Process Modeling?

- Understand components of current logical or physical system for purpose of rebuilding in a different physical form/technology, possibly with some changed functionality
- Find inefficiencies in current system
- Re-engineer current system

Sources/Sinks

(external entities)

- Any class of people, an organization, or another system which exists outside the system you are studying.
- Form the boundaries of the system.
- The system and external entities exchange data in the form of data flows.
- Must be named, titles preferred to names of individuals - use a noun

Data Flows

- data in motion
- marks movement of data through the system - a pipeline to carry data
- connects the processes, external entities and data stores
- Unidirectional
- originate OR end at a process (or both)
- name as specifically as possible - reflect the composition of the data - a noun
- do not show control flow! Control flow is easy to identify- a signal with only one byte - (on/off).

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

- HINT: if you can't name it: either it's control flow, doesn't exist or you need to get more information!

Processes

- transform incoming data flows into outgoing data flows
- represent with a bubble or rounded square
- name with a strong VERB/OBJECT combination; examples:
 - create_exception_report
 - validate_input_characters
 - calculate_discount

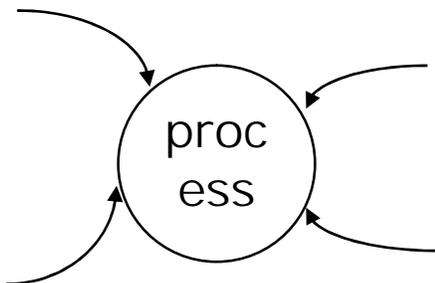
Data Stores

- data at rest
- represents holding areas for collection of data, processes add or retrieve data from these stores
- name using a noun (do not use 'file')
- only processes are connected to data stores
- show net flow of data between data store and process. For instance, when access a DBMS, show only the result flow, not the request

Data Flow Diagram Don'ts

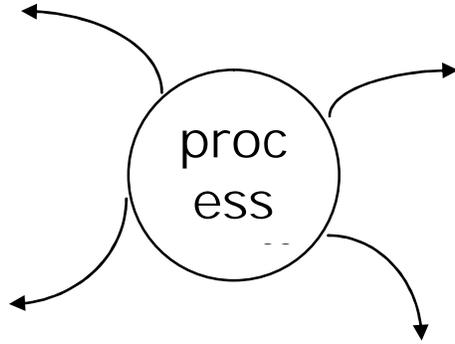
1. BLACK HOLES
2. MIRACLES
3. Let it get too COMPLEX: 7 ± 2 processes
4. Leave things UNLABELED
(corollary: labels should have meaning)
5. Data stores that are "SOURCES" or "SINKS"
6. Data flows that are UNASSOCIATED with a PROCESS
7. *Expect your diagram to be "perfect" the first time!*

Data Flow Diagram Don'ts

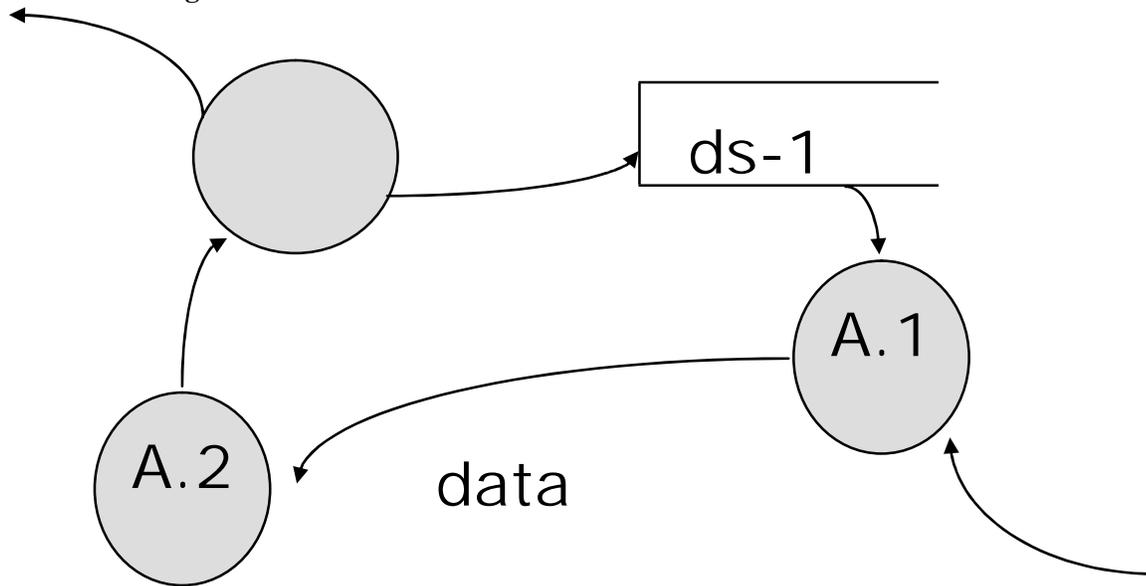


1. 'Black Hole'

2. 'It's a Miracle'



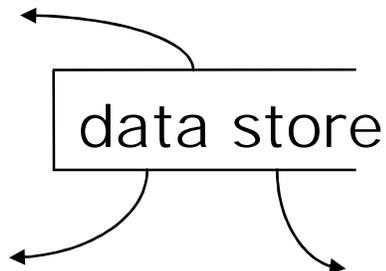
Data Flow Diagram Don'ts



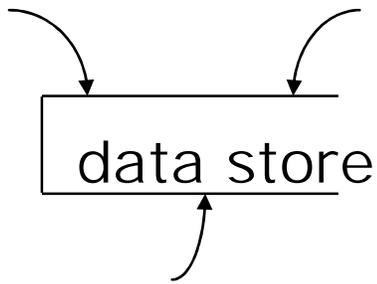
4. Leave Things Unlabeled

Corollary: Labels Should Have Meaning

Data Flow Diagram Don'ts



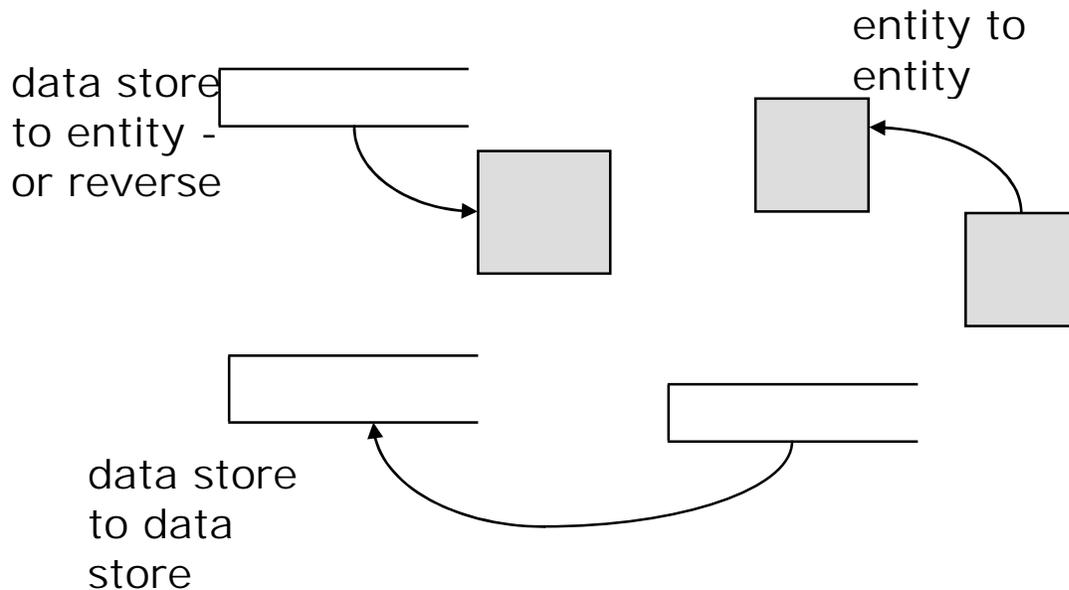
5. Miracle data source



5. Black hole data source

Data Flow Diagram Don'ts

6. Data Flows Unassociated With a Process



Diagramming A System

- multiple DFDs are required to represent a system
- DFDs are created at increasing levels of detail

Different Types of DFDs

- Context diagram
- Level-0 diagram (system diagram)
- Level-*n* diagram
- Primitive diagram

Context Diagram

- defines the scope of the system by identifying the system boundary
- contains:

—one process (which represents the entire system)

—all sources/sinks (external entities)

—data flows linking the process to the sources and sinks (external entities)

Example Context Diagram

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Constructing a Context Diagram

- identify and list sources/sinks (external entities)
- identify and list inputs to and outputs from sources/sinks (external entities)
- create context diagram

Level-0 Diagram

- describes the overall processing of the system
- show one process for each major processing step or functional requirement
- data flows from the context appear on system diagram also (level balancing)
- can show a single data store to represent all data in aggregate at this level
- can draw duplicate sources, sinks and data stores to increase legibility

Drawing a Level-0 Diagram

- list the major data stores
- list major business steps
- draw a segment for each business step
- assemble into single DFD
- re-organize until satisfied
- number processes

Functional Decomposition

- similar to a series of more detailed maps
- *iterative* process of breaking the description of a system into finer and finer detail to create a set of charts in which *one process* on a given chart is explained in greater detail on another chart
- referred to as exploding, partitioning, or leveling
- must use your judgment to decide what goes on each level
- show error and exception handling on lower levels (if at all)

Lower Level Diagrams

- explode the processes shown on the level-0 diagram
- each process is represented by its own DFD
- balance data

—data flows on upper level appear on lower level, or

—data flows on upper level are broken into component pieces with components shown on lower level

- each lower level shows greater and greater detail
- follow numbering convention

Balancing DFDs

- conserve data from level to level - inputs and outputs on the higher level must appear somewhere on the lower level

Advanced Rules

- Composite data flow on one level can be split into its component data flows on the next level - but new data cannot be added and all data in the composite must be included in the sub-flows
- The inputs to a process must be sufficient to produce the outputs.
- Lowest level DFDs may add new data flows to represent exception handling, i.e., error messages

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

- May repeat data stores or sources/sink to avoid crossing lines

Additional Guidelines

- the inputs to a process are *different* from the outputs of that process
- objects in a set of DFDs have unique names
- do not change data flow names on lower levels unless you are decomposing a data flow into component pieces.
- never explode a single process into another single process. If you cannot partition the process, then the lower level DFD is not needed.
- expect to iterate, put down the DFD and go back to it a few times to create something satisfactory.

Other Questions about Lower level diagrams

1. How deep? (how many levels?)

- if the process has only one input or one output, probably cannot partition further;
- can you describe the process in English in about 1/2 page?

2. How broad? (how many processes on a level?)

- $7 \pm$ two is a reasonable heuristic
- may temporarily place much of the system on a single diagram then re-draw into separate levels

Quality Guidelines

- Completeness

–all components included & in project dictionary

- Consistency

–between levels: balancing, leveling

- Timing considerations

–assume system never starts and never stops

- Iterative nature

–revisions are common

- Drawing primitives (lowest level)

–when to stop?

7 Finding the data fields to be used in the database

1. Plan
2. Executing the Plan
3. First way to enable input of new records via query based form
4. Second way to enable input of new records via query based form
5. If it won't work...
6. A Big Warning
7. Moving on....
8. Postscript
9. Lastly
10. Editorial Philosophy

The Plan _____

Part of the art of creating good databases lies in choosing what fields will appear in what tables. The tables are the bedrock of any database. There are serious rules about what should and should not be in them... try to find discussions of **data normalization** to help you start learning about those rules.

The rules support a couple of ideas:

-) Don't enter anything twice in the database.
-) Keep things simple.
-) Avoid entering things you don't need to enter.

In connection with that last point, consider the following database application, and try to imagine the many, many similar situations.

Imagine that you are keeping track of some stock market investments. (Yes, I know that what follows would not serve an investor's *real* needs. It is just to illustrate a point.)

In the database we are going to have a record for each holding. If Fred had...

-) 100 shares of IBM
-) 50 shares of Google, and
-) 10 shares of Exxom

....then he'd have 3 records in his database. Each record would have the following fields: Company name, Number of shares owned, Value of a single share. (That last field would be a pain... if Fred wanted an up-to-date idea of his shares' total value, he'd have to go into the database and change the values in each of the "PerShare" fields.)

You might think it would be nice to have a "Value of the holding" **field**. I.e., if IBM were worth \$80 per share, Fred's 100 shares would be worth \$8000, and that "8000" could be stored **in the table**. This would be a **Bad Idea**: In a sense you would be breaking the "Don't enter something twice" rule (the "Value of Holding" field would, essentially, duplicate a combination of the "Shares" and "PerShare" fields), **and** it would break the "Don't enter things you don't need to" rule.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

But that doesn't mean that Fred can't make the computer work out for him that the IBM holdings are worth \$8000! *How he gets that information, the right way, is the subject of this tutorial!*

Before we proceed: A detail. Investors will want to know the value of their shares on specific dates. The per share value of a share varies from day to day. The database, as described here, will either need all the prices re-entered for whatever day is of interest, or a more complex database ("easily" created) will be needed. To keep things simple, which is all we need for the skills under consideration, just think of this database of a way to know the cost of different investments, and imagine that the PerShare figure records what was paid for the shares on the day they were purchased. (And yes, I do realize that if this database were to be used in the real world, the date the shares were purchased would also be in the table.)

And a bit of bad news before we proceed: As described in the main part of this tutorial, you can have a form to view information from a database, complete with some calculated fields (i.e. the "Value of holding" information). However, you will have, for now, to use a *separate* form if you want to make changes to any table or tables underlying the form with calculated fields. And, more bad news, you will probably have to click buttons to make the "info display" form update itself after any changes on the "data entry" form. Sorry! I'm learning all the time, and may "crack" this one sometime... but for now, here's "a way", be it ever so crude...

Executing the plan_____

Some earlier tutorials give you more support than this one does. If you find it difficult to complete the tasks specified below, you might want to work through some of the earlier tutorials first.

Create a table called "Main". Do it however you wish, with the wizard, or in design mode. Be sure to set the first field as the primary key, with the "Autovalue" property set to "Yes".

Define the fields with the following field types, names, lengths, etc, as shown:

ID (Primary key, Integer, AutoValue)

Shares (Integer)

PerShare (Number, length 16, decimal places: 2)(This for price)

Company (Text [varchar], length:5)

Put some data in the table. Two records will suffice.

Close the table.

Now that you have the table, there are two ways you can provide yourself with a way to edit what is in it. For simple things, you may need nothing more than a simple form. But if you want a calculated field displayed, then this morning (July 2009, and again March 2010) I can only do that with a form based on a query. (I could have *sworn* I had a form working with a calculated field, without a query in 2009... but I can't do it now.) The calculated field will show the value of the shares.

Create a query. I used the wizard.

Include ALL of the fields. (You may not need to, but for now: include them, just in case.)

Do not set a sort order. (You probably could, but I'm trying to keep this simple.)

Do not set any search conditions.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Step 4: Accept the default "Detailed" type query. This will allow you to skip a few steps.

Next step: In at least some pre-version 3.1 variants of DataBase, at the next step, the default aliases include "Main." in front of "ID", "Shares", etc. If you are using an old version, and seeing this, edit the "Main." off of the aliases. (E.g., make the alias for "Main.Shares" just "Shares", etc.). If you are using DataBase version 3.1, you won't need to do this, and the step where you will be able to see the aliases will be step 7. I suspect it was step 7 for a long time... but there was what I think was a typo in this tutorial, and the tutorial said the "Aliases" step was step 5. (If you can confirm that it was once step 5, I'd be interested. Or if you are still using DataBase 2.4, please check what its "Aliases" step is for me? (And then upgrade yourself!))

Call the query "QueryWithCalc" ... even though it doesn't have a calculated field yet!

Before you click "Finish" to leave step 8, select "Modify query" for what happens next.

The main design window for the query should open. Across the bottom is a table, with four populated columns. Click in the "Field" cell, the top one, of the fifth column, the first empty one. Enter....

Shares*PerShare

... being careful to use the same capitalization as you used in naming those fields... it does matter. Do not be alarmed if DataBase puts some quotation marks around the names. If you have a field name with spaces in the name, enclose the name in quote marks, e.g.

"Per Share"

Next, fill in **Value of Shares** for the alias of the fourth column. (It doesn't have to be bold, I just wanted to avoid using quote marks to delimit the alias, as you shouldn't use them with your entry.)

Put a tick in the "visible" box, if it isn't ticked already.

Finally, save the query definition and run it. You should see sensible results, and you can alter the contents of the database.... if all is well. (Don't fool with the "ID" field's contents.) (The time I tried making a query without the ID field, I could see sensible results... but not change anything.)

I hope you wouldn't expect to be able to directly change the "Value of Holding" entry? Think about it. You can try.. no harm will come of doing so.

N.B. **If** you find that you can change the number of shares in a record, or change the cost per share field, then when you do it, the "Holding value" will not change immediately. It **will** change when you...

-) Go to a different *record* (changing *fields* is not enough), or...
-) Click on the "Save Current Record" icon, or...
-) Close the query.

So! We have a query that fetches the number of shares, and the value per share from the table, and then works out for us the value of the holding. In a sense, we have achieved our goal... but it would be better to press on, and complete additional steps....

Now use the wizard to create a new form.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

In step 1, select our newly created query "QueryWithCalc", and ask for all of its fields to be included in the form.

Step 2: Don't set up a sub-form, just click "Next".

Step 5: I used a "Data Sheet" arrangement; I don't doubt others would work, too.

In step 6, set data entry to "display all", without "change" restrictions.

Step 7: Any style will do.

Step 8: Use "FormWithCalc" for the form's name, or "FormBasedOnQueryWithCalc", and tell the wizard that you want to work with the form after it is created.

Click "Finish", to complete the wizard, and you should find you have a good interface to the table, with the calculated field telling you the value of each holding.

Ta da? Yes... if you are only trying to inspect data in the table, and learn the values of different holdings, i.e. the answer to "number of share multiplied by value per share."

At this point, this morning (9Mar10, DataBase 3.1.0, WinXP) I *am able* to change data in the underlying table via the query based form! As long as, for now, I only change data in *existing records*.... but see below.

I have wrestled with this sort of thing again and again, getting different answers every time! I have not (yet!) tested altering data in tables from forms displaying the *results of a query pulling data from more than one table*. That may be A Different Story.)

If at this point you try to add a new *record*, i.e. a new row of data, you will probably get "Error writing data to database... Input required... 'Value of Shares'...". But! We can fix this! There are two ways. In either case, start by closing the form.

First way to enable input of new records via query based form

This is a little more "cumbersome" than the second, but it appeals to my belief that the more checking you have the better. The second solution turns off more than you need to.

Open the form for editing. Right click on the column heading "Value of Shares" (the calculated field).

Click on the "Column..." option. A dialog headed "Properties: Formatted Field" should open. Select the "Data" tab. Set "Input required" to "No".

That should do it! Next I'll give you an alternative solution in case for some reason you don't like that. (Or in case it won't work for you. If it won't, and you are using DataBase 3.1.0 or later, please contact me, citing "fdb1calcf1", and I will look at this *again*.)

Second way to enable input of new records via query based form

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

The form should be closed. In the DataBase main project manager window, select "Forms" in the left hand panel, a then right click on "FormWCalc". Click on the "Database" entry at the bottom of the pop up menu. Click on "Advanced Settings". Remove the tick in front of "Form data input checks for required fields".

Save the edited form. Open it for normal use... you should now be able to enter data in the table through the form which is based on the query. If you can't, and you are using DataBase 3.1.0 or later, please contact me, citing "fdb1calcf1", and I will look at this *again*.)

If you worry (as I think you should) about the ramifications of this turning off of checks, I can at least reassure you that *some* checks remain in place. For instance, I tried to enter "xx" into a field that was of type "integer". The "xx" was converted to a zero.

If it won't work...

There is a "crude" answer you can use when you can't get data to pass back to your tables from a form: use multiple forms.... one (query based) with whatever you need to see, and another, simpler, form (or forms) for changing data in the table(s).

A Big Warning

Don't fall to the temptation of working directly with a query in hopes of changing what is in a table. When you change entries in the result of a query, it will appear to work, but you are only changing what you see on the screen, not what is in the underlying table.

Moving on....

I hope you have a sensible result, after working through what is above. Apologies if the answer is still flawed at this time... I've spent hours on this over the years. But do "complain" if need be!

Postscript _____

Ha! Progress? Preliminary experiments suggest that the above techniques won't work if your query draws data from more than one underlying table. Sigh. But a least a candidate for why the thing "works" sometimes, and not others. So how DO we do it in multi-table situations?

An enquiry at the excellent oooFurum.org gave rise to this information from Villeroy, who has supplied many excellent answers for the community over the years. Note that he supports my idea that you need to include the primary key in the form or query, even if you don't *display* it. But note the additional point he makes: You DO need to display the primary key if its "autovalue" property isn't set to "yes". Also, alas, note that he says you can't use the techniques above on a query drawing information from more than one table.... unless, as they say, you know otherwise??

Your form is writable if it is bound to some record set from a single table and if the table's primary key is included.

This means that the form is bound to the entire table and the table is editable OR the form is bound to some row set like

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
SELECT FROM "Single_Table" WHERE ... ORDER BY...
```

This row set is editable just like the table if and only if the field list includes the primary key of the single table.

If the PK is an auto-value you don't have to display the PK by means of a form control. Visible or not, It is part of the form's row set.

Subforms follow the same rules. A pair of subform and parent form reflects a one-to-many relation and makes this relation editable. The editable parent selects editable records in the subform.

As I said above, I've wrestled with this for some time. The following may be useful of interesting if you find that what's above doesn't Just Work.

In a discussion at the Open Office forum, two comments were made, which may be of interest:

-) Forms don't calculate anything, but they can be used to display results calculated in queries.
-) You could add a subform to the form, add textbox to subform and as datasource use query which calculates the value.

The first suggests that what as of 9Mar10 I think works never has. But I've been through periods of not being able to make it work!

The second suggests up an interesting alternate path to a solution. I believe that answer is even suited to forms that display more than one record at a time. I wonder if it would require a full blown query, or whether the subform could be asked more directly just to display the result of "Shares"*"PerShare". (I didn't have much luck when I tried to do this without a query... but I live in hope that it CAN be done, at least for a form displaying just one record. (There are ways if you want to get into using macros.)(Adding a sub-form IS covered in another of my pages, but that page talks about a great deal else! (Adding the subform isn't hard... when you know how... but what you need to do in order to add a subform may not seem "obvious"... it wasn't to me, anyway!))

Another forum.org discussion which may be useful is a form that can BOTH be used to enter new data, AND which can display a calculated result from fields. Also uses a macro.

Hmmm... "stranger, stranger and harder...."

In exploring a question from a reader, 3 March 10, I've stumbled into something that may or may not be true, may or may not be useful....

My current best guess... needs work... is....

If you just want to look at some data, e.g. the investment data used as the example at the top of the page, AND have a calculated field, then The Way To Go is to set up a query to "harvest" the data, and calculate what will be in the calculated field... as set out in the main part of what starts at the top of this page.

**** B U T ****: If you want to ADD OR EDIT RECORDS via the same form, then things get a little tiresome. However, it MAY (**P.S. I now (9 March 2010) think that the following is unnecessarily complicated, that the simple answer in the main part of this page WORKS.**) be possible to proceed as follows. What follows is

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

NOT extensively tested, and the text needs work, but may get you where you want to go, if you want to be a pioneer! Set up a form. Put a datagrid on it, based directly on the underlying table. Continuing with the example I started with, on that you would DISPLAY the three "obvious" fields: Shares, PerShare, Company. I think you'd want the ID field available... it may be anyway.... but not showing. Set up a query to return ID and "Shares * PerShare"

Add a subform to the main form, have it display the "Shares*PerShare" part of the query.

I THINK you can get that subform lined up on the screen with the main form, creating something that almost looks like one datagrid with 4 columns. The human won't know (or care) that the first 3 come from one control, and the last from a separate control... but that MAY get you around the problem of creating a form which can display calculated fields AND be used to edit the underlying table. (If any reader KNOWS this idea is a non-starter, or better yet, tries it and gets it to WORK!!, I'd be delighted to hear from said reader!)

Lastly _____

It makes little sense to show the record ID on the form displaying the table's data and the calculated values of the share holdings. It, in this example, is just an "internal" thing, of interest only to DataBase. I think that the ID field must remain in the query if you have a query based on more than one table (which may or may not be able to pass data back to the underlying tables, but you are allowed to do the following. (At the time of writing, I haven't tested these techniques with queries pulling data from multiple tables. Apologies if since writing this, I've gone on to the more advanced case, found it working, and forgotten to remove this warning.)

Open the form (the one based on the query) for editing.

Right-click on the "ID" column heading. Click on "Hide columns". Save your form.

The form should still work, but not *display* the ID column, which is a necessary column, for the computer, but one of little interest to a user of the database.

Editorial Philosophy

I dislike 'fancy' websites with more concern for a flashy appearance than for good content. For a pretty picture, I can go to an art gallery. Of course, an attractive site WITH content deserves praise... as long as that pretty face doesn't cost download time. In any case....

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

8. Selecting fields for keys

A primary key is a field in the table that Access can use to identify each field uniquely. For instance, you may have several John Smiths in your database, so how can you tell them apart? Access suggests that you create a field, perhaps a code number, for each record, with no two records having the same value in this field. If you mark this field as the primary key, then Access will make sure no two entries are ever the same.

A good example of a primary key field is one set up as an AutoNumber type. This will automatically fill in a code number for each record, starting at 1 for the first.

Primary key - Primary key means main key

def:- A primary key is one which uniquely identifies a row

of a table. this key does not allow null values and also

does not allow duplicate values. for ex,

empno	empname	salary
-------	---------	--------

1	firoz	35000
---	-------	-------

2	basha	34000
---	-------	-------

3	chintoo	40000
---	---------	-------

it will not the values as follows:

1	firoz	35000
---	-------	-------

1	basha	34000
---	-------	-------

	chintoo	35000
--	---------	-------

Unique key - single and main key

A unique is one which uniquely identifies a row of a table, but there is a Difference like it will not allow duplicate values and it will allow number of null values(In oracle).

it allows only a single null value(In sql server 2000)

Both will function in a similar way but a slight difference

will be there. So, decalaring it as a primary key is the

best one.

foreign key - a foreign key is one which will refer to a

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

primary key of another table

for ex,

emp_table dept_table

empno empname salary deptno deptno deptname

In the above relation, deptno is there in emp_table which

is a primary key of dept_table. that means, deptno is

referring the dept_table.

primary key Definition: The primary key of a relational table uniquely identifies each record in the table. It can either be a normal attribute that is guaranteed to be unique (such as Social Security Number in a table with no more than one record per person) or it can be generated by the DBMS (such as a globally unique identifier, or GUID, in Microsoft SQL Server). Primary keys may consist of a single attribute or multiple attributes in combination.

For more information on keys, read the article Database Keys. For more on selecting appropriate primary keys for a table, read Choosing a Primary Key.

Candidate key Definition: A candidate key is a combination of attributes that can be uniquely used to identify a database record without any extraneous data. Each table may have one or more candidate keys. One of these candidate keys is selected as the table primary key.

Super key Definition: A superkey is a combination of attributes that can be uniquely used to identify a database record. A table might have many superkeys. Candidate keys are a special subset of superkeys that do not have any extraneous information in them.

In relational database design, a unique key can uniquely identify each row in a table, and is closely related to the Superkey concept. A unique key comprises a single column or a set of columns. No two distinct rows in a table can have the same value (or combination of values) in those columns if NULL values are not used. Depending on its design, a table may have arbitrarily many unique keys but at most one primary key.

Unique keys do not enforce the NOT NULL constraint in practice. Because NULL is not an actual value (it represents the lack of a value), when two rows are compared, and both rows have NULL in a column, the column values are not considered to be equal. Thus, in order for a unique key to uniquely identify each row in a table, NULL values must not be used, however a column defined as unique key column allows only one null value, which in turn can uniquely identify that row/tuple.

A unique key must uniquely identify all *possible* rows that exist in a table and not only the currently existing rows. Examples of unique keys are Social Security numbers (associated with a specific person^{[1][2]}) or ISBNs (associated with a specific book). Telephone books and dictionaries cannot use names, words, or Dewey Decimal system numbers as candidate keys because they do not uniquely identify telephone numbers or words.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

A primary key is a special case of unique keys. The major difference is that for unique keys the implicit NOT NULL constraint is not automatically enforced, while for primary keys it is enforced. Thus, the values in unique key columns may or may not be NULL. Another difference is that primary keys must be defined using another syntax. Thus Primary Key column allows no row having NULL while Unique Key column allows only one row having null value.

The relational model, as expressed through relational calculus and relational algebra, does not distinguish between primary keys and other kinds of keys. Primary keys were added to the SQL standard mainly as a convenience to the application programmer.

Unique keys as well as primary keys can be referenced by foreign keys.

Examples:

Imagine we have a STUDENTS table that contains a record for each student at a university. The student's unique student ID number would be a good choice for a primary key in the STUDENTS table. The student's first and last name would not be a good choice, as there is always the chance that more than one student might have the same name.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

9. Normalizing the database including analysis of functional dependencies

Illogically or inconsistently stored data can cause a number of problems. In a relational database, a logical and efficient design is just as critical. A poorly designed database may provide erroneous information, may be difficult to use, or may even fail to work properly.

Most of these problems are the result of two bad design features called: redundant data and anomalies. Redundant data is unnecessary reoccurring data (repeating groups of data). Anomalies are any occurrence that weakens the integrity of your data due to irregular or inconsistent storage (delete, insert and update irregularity, that generates the inconsistent data).

Basically, normalisation is the process of efficiently organising data in a database. There are two main objectives of the normalization process: eliminate redundant data (storing the same data in more than one table) and ensure data dependencies make sense (only storing related data in a table). Both of these are valuable goals as they reduce the amount of space a database consumes and ensure that data is logically stored.

The process of designing a relational database includes making sure that a table contains only data directly related to the primary key, that each data field contains only one item of data, and that redundant (duplicated and unnecessary) data is eliminated. The task of a database designer is to structure the data in a way that eliminates unnecessary duplication(s) and provides a rapid search path to all necessary information. This process of specifying and defining tables, keys, columns, and relationships in order to create an efficient database is called normalization.

Normalisation is part of successful database design. Without normalisation, database systems can be inaccurate, slow, and inefficient and they might not produce the data you expect.

When normalising a database you should achieve four goals:

1. Arranging data into logical groups such that each group describes a small part of the whole
2. Minimizing the amount of duplicated data stored in a database
3. Building a database in which you can access and manipulate the data quickly and efficiently without compromising the integrity of the data storage
4. Organising the data such that, when you modify it, you make the changes in only one place

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

10.Front End & Back End Concept

Front end and **back end** are generalized terms that refer to the initial and the end stages of a process. The front end is responsible for collecting input in various forms from the user and processing it to conform to a specification the back end can use. The front end is an interface between the user and the back end.

Front end may refer to:

-) The front of a vehicle body
-) Front-end load, a charge in investing
-) Front End Loader, a band
-) Front-end loader, construction equipment
-) Front-end loading, in project management
-) RF front end, in electronics

The support components of a computer system. It typically refers to the database management system (DBMS), which is the storehouse for the data.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

11. Overview of Script Language, Java Script, Java, VB Script, VB.Net,

A high-level programming language that is interpreted by another program at runtime rather than compiled by the computer's processor as other programming languages (such as C and C++) are. Scripting languages, which can be embedded within HTML, commonly are used to add functionality to a Web page, such as different menu styles or graphic displays or to serve dynamic advertisements. These types of languages are client-side scripting languages, affecting the data that the end user sees in a browser window. Other scripting languages are server-side scripting languages that manipulate the data, usually in a database, on the server.

Scripting languages came about largely because of the development of the Internet as a communications tool. JavaScript, ASP, JSP, PHP, Perl, Tcl and Python are examples of scripting languages.

JavaScript is an implementation of the ECMAScript language standard and is typically used to enable programmatic access to computational objects within a host environment. It can be characterized as a prototype-based object-oriented scripting language that is dynamic, weakly typed and has first-class functions. It is also considered a functional programming language^[6] like Scheme and OCaml because it has closures and supports higher-order functions.

JavaScript is primarily used in the form of client-side JavaScript, implemented as part of a web browser in order to provide enhanced user interfaces and dynamic websites. However, its use in applications outside web pages is also significant.

JavaScript and the Java programming language both use syntaxes influenced by that of C syntax, and JavaScript copies many Java names and naming conventions; but the two languages are otherwise unrelated and have very different semantics. The key design principles within JavaScript are taken from the Self and Scheme programming languages.

The primary use of JavaScript is to write functions that are embedded in or included from HTML pages and that interact with the Document Object Model (DOM) of the page

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

JavaScript	
Filename extension	.js
Internet media type	application/javascript, text/javascript
Uniform Type Identifier	com.netscape.javascript-source
Type of format	Scripting language

JavaScript and Java

Common misconception is that JavaScript is similar or closely related to Java. It is true that both have a C-like syntax, the C language being their most immediate common ancestor language. They are both object-oriented, typically sandboxed (when used inside a browser), and are widely used in client-side Web applications. In addition, JavaScript was designed with Java's syntax and standard library in mind. In particular, all Java keywords are reserved in JavaScript, JavaScript's standard library follows Java's naming conventions, and JavaScript's Math and Date objects are based on classes from Java 1.0.

But the similarities end there. Java has static typing; JavaScript's typing is dynamic (meaning a variable can hold an object of any type and cannot be restricted). Java is loaded from compiled bytecode; JavaScript is loaded as human-readable source code. Java's objects are class-based; JavaScript's are prototype-based. JavaScript also has many functional features based on the Self language.

Java

Java is a programming language and more. It originated from Sun Microsystem's Oak project and Sun still develop, maintain and supply it. Although it's not open source, nor delivered under the GPL license, much is available as a no-charge download from Sun's various web sites. At its first release, Java primarily was used as a language for writing applications to be embedded in browsers (known as applets), but it has grown into many other areas. These days, applets are very much a minority use of Java, although still an important one. Other uses include web server-side programming (using "servlets" or "Java Server Pages") and large-scale, enterprise-wide applications using resource servers, "Enterprise Beans" and more.

2.1 The fundamental elements of Java

Java is an object oriented language, and all code you write is organised into classes. If you structure the way a class is defined and called according to certain rules, then that class may be usable as a program, or as an applet, or as a servlet. The code you

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

actually write ("source code") is English-like text, and you save it into a regular text file, just as you do with other programming languages.

Source Code

Let's see an example of the source code of a Java program:

```
// Tiniest of programs to check compile / interpret tools
public class Hello {
public static void main(String[] args) {
System.out.println("A program to exercise the Java tools");
}
}
```

We've saved this example into a file called Hello.java. Note that the file name should be the same as the class name declared in the file followed by ".java". This rule can be broken with some environments and compilers, but it's a good rule to follow. Java is case sensitive. Note that we have started our class name with a capital, followed by lower case letters – another suggested convention.

Class files

Some languages are interpreted and run directly from the source code, but Java isn't one of those. It's a language that's designed to run quickly, and interpreting the whole thing "public", etc., every time it's called is inefficient, so we compile the Java into a binary format. Let's use the "javac" program, supplied by Sun as part of their free downloads, to do the conversion:

```
bash-2.04$ ls
Hello.java
bash-2.04$ javac Hello.java
bash-2.04$ ls
Hello.class Hello.java
```

```
bash-2.04$
```

If things are OK, javac doesn't produce any message. But if your source code isn't in the correct syntax or refers to something that doesn't exist, you'll probably get an error message at compile time, such as:

```
bash-2.04$ javac Oops.java
Oops.java:5: cannot resolve symbol
symbol : class string
location: class Oops
public static void main(string[] args) {
^
```

```
1 error
```

```
bash-2.04$
```

Edit the source file, correct your error, save the file and compile again. You may have to go round this cycle a number of times until you get rid of all your errors. By the way, the mistake here was that we used a lower case "s" not a capital "S" for the word "String". Once you compile successfully, the class file is in a published binary format, but it's very rare for most programmers to have to get involved at that level. Suffice it to say at this stage that the class file is compact and is independent of host computer architecture. Unlike most compilers, javac does not produce a snippet of executable code tuned for the particular processor architecture on which it is run. You might like to see what the format looks like. Here's a binary dump:

```
0000000 ? ? ? \0 \0 \0 . \0 035 \n \0 006 \0 017 \t
0000020 \0 020 \0 021 \b \0 022 \n \0 023 \0 024 \a \0 025 \a
0000040 \0 026 001 \0 006 < i n i t > 001 \0 003 ( )
0000060 V 001 \0 004 C o d e 001 \0 017 L i n e N
0000100 u m b e r T a b l e 001 \0 004 m a i
0000120 n 001 \0 026 ( [ L j a v a / l a n g
0000140 / S t r i n g ; ) V 001 \0 \n S o u
0000160 r c e F i l e 001 \0 \n H e l l o .
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
0000200 j a v a \f \0 \a \0 \b \a \0 027 \f \0 030 \0
0000220 031 001 \0 $ A p r o g r a m t o
0000240 e x e r c i s e t h e J a
0000260 v a t o o l s \a \0 032 \f \0 033 \0 034
0000300 001 \0 005 H e l l o 001 \0 020 j a v a /
0000320 l a n g / O b j e c t 001 \0 020 j a
0000340 v a / l a n g / S y s t e m 001 \0
0000360 003 o u t 001 \0 025 L j a v a / i o /
0000400 P r i n t S t r e a m ; 001 \0 023 j
0000420 a v a / i o / P r i n t S t r e
0000440 a m 001 \0 \a p r i n t l n 001 \0 025 (
0000460 L j a v a / l a n g / S t r i n
0000500 g ; ) V \0 ! \0 005 \0 006 \0 \0 \0 \0 002
0000520 \0 001 \0 \a \0 \b \0 001 \0 \t \0 \0 \0 035 \0 001
0000540 \0 001 \0 \0 \0 005 * Â· \0 001 Â± \0 \0 \0 001 \0
0000560 \n \0 \0 \0 006 \0 001 \0 \0 \0 003 \0 \t \0 \v \0
0000600 \f \0 001 \0 \t \0 \0 \0 % \0 002 \0 001 \0 \0 \0
0000620 \t ? \0 002 022 003 Â¶ \0 004 Â± \0 \0 \0 001 \0 \n
0000640 \0 \0 \0 \n \0 002 \0 \0 \0 006 \0 \b \0 \a \0 001
0000660 \0 \r \0 \0 \0 002 \0 016
```

0000670

The Java Runtime Environment

How do we use a binary class that won't run on your computer? We run it through another program known as a Java Virtual Machine (JVM) which interprets each of the elements of the class file and performs the actions stipulated. There are going to be many things that you want to do in your Java that others want to do as well, things as simple as outputting text. So along with the JVM, Sun provide a large library of extra classes which give you the facilities you need without having to write them yourself. There are so many extra classes that they're arranged into "packages" to make them more manageable, and the combination of the Java Virtual Machine and these standard packages is known as the Java Runtime Environment or JRE.

Let's run the example program that we compiled earlier in the "java" JRE, which lets us run an appropriate class as a stand-alone program:

```
bash-2.04$ java Hello
```

A program to exercise the Java tools

```
bash-2.04$
```

The Java World

Well House Consultants' courses specialise in the teaching of programming languages and their application, so you'll be going on in subsequent sections to cover the details of what to put into the source files and (unless this is a compressed course) how to think through the design of your application and source code so that it's well structured, easy to use, easy to maintain, and easy to update in the future as your requirement(s) develop. At first, a lot of this may seem very theoretic, so here we'll give you a brief glance further into the Java world that you're headed for.

Java development environments and tools

As your Java source code grows in size, you'll find that it's quite a task to keep track of all the various classes and other components involved. Development environments such as JDeveloper, JBuilder and Forte provide you with facilities to make this management much easier, and with shortcuts that let you enter and edit code far more efficiently than you could with a standard editor. We find that it's best to teach you the fundamentals of the Java language before we expose you to these tools. Other wise, you're likely to find yourself spending a great deal of time trying to understand some of the suggested code and options that the tool offers but which we haven't yet covered. To give you a flavour of the look and feel of a Java Development

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Environment, here are some screen shots from Oracle's Jdev. It's easier to learn than some of the environments we've seen, but you still need to understand the questions being asked and the code before you can make good use of it

```
package wellho;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.PrintWriter;
import java.io.IOException;
public class demolet extends HttpServlet implements SingleThreadModel
{
private static final String CONTENT_TYPE = "text/html; charset=windows-1252";
public void init(ServletConfig config) throws ServletException
{
super.init(config);
}
/**
* Process the HTTP doGet request.
*/
public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException
{
String var0show = "";
try
{
var0show = request.getParameter("showthis");
}
catch(Exception e)
{
e.printStackTrace();
}
response.setContentType(CONTENT_TYPE);
PrintWriter out = response.getWriter();
out.println("<html>");
out.println("<head><title>demolet</title></head>");
out.println("<body>");
out.println("<p>The servlet has received a GET. This is the reply.</p>");
out.println("</body></html>");
out.close();
}
/**
* Process the HTTP doPost request.
*/
public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
{
String var0show = "";
try
{
var0show = request.getParameter("showthis");
}
catch(Exception e)
{
e.printStackTrace();
}
}
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
response.setContentType(CONTENT_TYPE);
PrintWriter out = response.getWriter();
out.println("<html>");
out.println("<head><title>demolet</title></head>");
out.println("<body>");
out.println("<p>The servlet has received a POST. This is the reply.</p>");
out.println("</body></html>");
out.close();
}

}
```

Sun's distribution includes a number of tools in addition to **java** and **javac**, including:

jar a tool for creating and manipulating java archive files (jars) **javadoc** generates API documentation from source files

javap generates a human-readable description of the API of a class file **jdb** a text-based debugger Java is ideally suited for larger projects. Such projects may involve considerable management of development and runtime files, such as sources, classes, libraries etc., where many of the files are derived from others and need to be updated when any of the files on which they depend are changed. Apache Ant is a Java-based build tool. In theory, it is somewhat like make, but without make's wrinkles. It's open source and becoming very popular. See <http://ant.apache.org/> for further details.

Java Runtime Environments

Stand-alone programs probably aren't what you'll be writing in the longer term, even though they're excellent for learning the fundamentals of Java. Later on you'll be slotting your classes into other Java runtime environments, some of which are open source and others are commercial products. Server side, there are a number of environment interfaces you may use. These include:

Servlets Executable programs with a web interface

JSPs Embedding server executable content within a web page

RMI and **EJBs** Providing object servers

JREs that support these interfaces include Apache Tomcat, BEA WebLogic JRockit and IBM's WebSphere application server. Client side, most users want to access information via their browser these days and plugins that support Java are available for most common browsers. Java is built in to certain browsers too, but do beware that it's often in old (or even ancient) versions. AppletViewer, a part of Sun's distribution, is a JRE with the same interface that a browser provides but without the caching or overhead of a browser, and is useful for development and testing. And it turns out that the JRE that you use for stand-alone programs can do much more. You can have what looks like a stand-alone program at "the top" but have it act as a client or server (or both!). You can have it provide a graphic user interface (GUI) in much the same way that an applet would, and much more.

Java distributions

For one programmer, Java might be the tool he uses to program a toaster. Another might be using it to provide the core financial accounting services for a major bank. Is it possible for both of these requirements to be met by the same downloaded language? Yes...and no.

Java is described as a "simple" language and indeed at it's centre it is. The syntax is not overcomplex, the facilities of the language itself are relatively few, which make it into something of a lean and mean core facility. The real power comes in the standard packages that are available, and that's where the requirements of our domestic appliance programmer will vary from the requirements of our banker.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

The Java 2 Standard Edition (J2SE) provides the essential compiler, tools, runtimes, and APIs for writing, deploying, and running applets and applications in the Java programming language. All developers will need to download a copy of J2SE, or its equivalent.

The Java 2 Enterprise Edition (J2EE) technology and its component-based model

simplifies enterprise development and deployment. The J2EE platform manages the infrastructure and supports the Web services to enable development of secure, robust and inter-operable business applications. The download has essential extra classes and environments to use in addition to the J2SE, which you will also need. The Java 2 Micro Edition (J2ME) specifically addresses the consumer space, which covers the range of extremely tiny commodities, such as smart cards or a pager, up to the set-top box. This download provides a highly optimised runtime environment. Again, to develop code, you'll also need J2SE. All of the above can be downloaded from <http://java.sun.com> Also available is a Java Runtime environment (J2RE) which will be required by users rather than developers. It includes the JVM and the classes that make up the JRE, but not tools such as the compiler. Different licensing rules will apply.

Java standard packages

Much of the power of a Java application is vested not in the language itself, but in all the standard classes provided and optional classes available. There are so many classes that they've been organised into bundles (called "packages") for easier management. The first release of Java included eight packages. These days it depends on just what edition you're talking about, but you'll probably find you have somewhere more than 100 packages in your JRE. Standard package names start "java." or "javax.". For example, there's *java.lang* to provide basic language facilities, *java.net* to provide network access, *javax.swing* to provide the Swing GUI and *javax.servlet* to provide Servlet support. You'll come across many more standard packages as you learn Java; however, as a rule of thumb, if the package name starts "java" or "javax", it's standard, but if it starts with something else like "com", it isn't.

Java versions

Java started off as Java release 1.0, and progressed through to Java 1.5. Sun were very conservative in moving the release numbers forward, so much so that the "1." Just became a part of the name. In summer 2004, Java 1.5 was re-branded Java 5. Java 1.2 was a major step, with a tripling of the number of packages provided. At that point Sun rebranded the new version the "Java 2 Platform". Although Java is designed to be processor- and operating-system independent, you need to be careful as you develop code that you do so for a runtime environment that will be available to your user. See how the number of standard packages and classes has increased:

Classes packages

Java	1.0	212	8
Java	1.1	504	23
Java	2 1.2	1520	59
Java	2 1.3	1842	76
Java	2 1.4	2991	135
Java	2 5.0	more	more

As well as the extra packages, there have been some language changes, such as an **assert** statement added to the language at release 1.4. Note that there is a very large installed base of browsers running Java 1.1, so you may still want to use it for writing simple applets. If that's the case, you'll need to obtain a copy of the JDK (Java Development Kit) for that release. The JDK became the SDK (Software Development Kit), now Java 2 SDK Standard edition, Version 1.5 or 5.0

VB Script

VBScript (Visual Basic Scripting Edition) is a scripting language developed by Microsoft for Windows operating systems.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

A VBScript code must be executed within a host environment. It allows you to interact with the host environment to perform some programming tasks.

A host environment will usually:

-) Provide you a specific way to enter your VBScript source code.
-) Provide you some basic objects defined in the VBScript core specification.
-) Provide you some specific objects to let your code to interact with the host application.
-) Provide you additional objects to let your code to access certain operating system resources.

Examples of VBScript host environments

-) Internet Explorer (IE) - Allows you to include VBScript code in HTML documents to be executed while IE is rendering HTML documents on the screen. This is also called client side scripting.
-) Internet Information Services (IIS) - Allows you to include VBScript code in HTML documents to be executed while IIS is fetching HTML documents on the Web server to deliver to client machines. This is also called server side scripting.
-) Windows Script Host (WSH) - Allows you to include VBScript code in script files to be executed directly on the Windows operating system.

VBScript version history:

1996 VBScript 1.0
1997 VBScript 2.0 - Renamed to 5.0 later
2002 VBScript 5.6
2007 VBScript 5.7

VBScript is actually is a limited variation of Microsoft's Visual Basic programming language. Therefore VBScript shares the same language syntax as Visual Basic.

Visual Basic can be used to develop stand alone Windows applications. Visual Basic can also be used to write macro codes for other Windows applications like Microsoft Access.

This section provides tutorial example on how to embed a VBScript code in a HTML document to be executed by Internet Explorer on the client machine.

Internet Explorer (IE) is a Microsoft application that can be used a Web browser to view Web pages. IE also supports a VBScript host environment that allows you to embed VBScript codes into source codes of Web pages - HTML documents.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

VBScript codes embeded in HTML documents will be executed while IE is rendering HTML documents on the browser window. This is also called client side scripting, because script codes are executed on the client machine instead of the server machine.

To add VBScript codes into your HTML documents, you need to use the "script" tag with the "language=vbscript" attribute. Inside the "script" tag, you can place any number of VB statements. Here is the syntax of adding VBScript codes in HTML documents:

```
... (HTML tags)
<script language=vbscript>
... (VB statements)
</script>
... (HTML tags)
```

Now let's try to write our first VBScript code in a HTML document.

1. Open the Notepad to enter the following HTML document:

```
<html>
<body>
<script language="vbscript">
document.write("Hello world! - VBScript in IE")
</script>
</body>
</html>
```

2. Save the HTML document as hello_vb.html.

3. View the HTML document with IE. You should see the following message in the IE window:

Hello world! - VBScript in IE

Congratulations. You have successfully written a VBScript code for the host environment supported in IE!

What happened here was:

-) We have added a "script" tag in our HTML document, hello_vb.html.
-) We included a simple VBScript code inside the "script" tag.
-) The VBScript code calls the "document.write" function, which is a function provided by the IE host environment to insert a text string into the HTML document.
-) We ran IE to view hello_vb.html and got exactly what we expected.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Internet Information Services (IIS) is a Microsoft product that offers and manages the Internet services, like the Web (HTTP) server, and the email (SMTP) server.

IIS also supports a VBScript host environment that allows you to embed VBScript codes into source codes of Web pages - HTML documents. VBScript codes embedded in HTML documents will be executed while IIS is fetching HTML documents on the Web server to deliver to the client machine. This is also called server side scripting, because script codes are executed on the server machine instead of the client machine.

One way to add VBScript codes into your HTML documents for IIS to execute is to use the ASP (Active Server Pages) technology. If you have IIS installed on your Windows system, you can use the following steps to run a simple VBScript code in IIS.

1. Go to Control Panel, then Administrative Tools, then Internet Services Manager, and right mouse click on Default Web Site, then select properties command.
2. Click on Home Directory tab on the properties dialog box, then click the Configuration button.
3. Click on App Mappings tab on the configuration dialog box, then check to see the following line in the mapping area to make sure that ASP is supported by IIS:

Extension	Executable Path	Verbs
.asp	c:\winnt\system32\inetsrv\asp.dll	GET,HEAD,POST,TRACE

4. Create the following hello.asp file:

```
<% @ language="vbscript"%>
<html><body>
<%
response.write("Hello world! - VBScript in IIS")
%>
</body></html>
```

5. Copy hello.asp to \inetpub\wwwroot, which is the directory where IIS takes HTML documents.

6. Run Internet Explorer (IE) with this url: <http://localhost/hello.asp>.

7. You should see "Hello world! - VBScript in IIS" on the IE window.

Congratulations. You have successfully written a VBScript code for the host environment supported in IIS!

What happened here was:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

) We checked the IIS setting to ensure that ASP is supported.

) We created a simple ASP page - a HTML document with a simple VBScript code.

) The VBScript code calls the "response.write" function, which is a function provided by the IIS host environment to insert a text string into the HTML document.

) We ran IE to view the resulting HTML document generated by IIS and got exactly what we expected.

Windows Script Host (WSH) is a Windows administration tool that provides host environments for several scripting languages including VBScript.

VBScript codes included in script files will be executed by WSH directly on the Windows operating system.

If you are running a Windows XP system, you can try these steps to run a simple VBScript code with Windows Script Host:

1. Create a script file called hello.vbs:

```
WScript.StdOut.WriteLine "Hello World! - VBScript in WSH"
```

2. Run hello.vbs with the "cscript" command in a command window:

```
C:\herong>cscript hello.vbs
Microsoft (R) Windows Script Host Version 5.6
Copyright (C) Microsoft Corporation 1996-2001. All rights reserved.
```

```
Hello World! - VBScript in WSH
```

Congratulations. You have successfully written a VBScript code for the host environment provided by WSH!

What happened here was:

) We created a simple VBScript code file.

) The VBScript code calls the "WScript.StdOut.WriteLine" function, which is a function provided by the WSH host environment to print a text string to the standard output channel - the command window in this case.

Using Visual Basic with Microsoft Access

Microsoft Access is a Microsoft application that can be used to store and manage data in database tables. Microsoft Access also supports a macro module that allows you to write macro code with Visual Basic (VB) language.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

If you have Microsoft Access installed on your Windows system, you can follow the steps below to create a simple application in Visual Basic language within Microsoft Access.

1. Run Microsoft Access, and create a blank Access Database called vb_tutorial.mdb.
2. Click Insert > Module from the menu. The Microsoft Visual Basic window shows up.
3. Enter the following code into the empty code module:

```
Sub Main()  
    MsgBox ("Hello world! - Visual Basic in Access")  
End Sub
```

4. Click File > Save from the menu. Enter "Hello" as the module name and save it.
5. Click Run > Run Sub/UserForm from the menu. The macro selection dialog box shows up.
6. Select "Main" macro, and click "Run". A dialog box shows up with the following message:

```
Hello world! - Visual Basic in Access
```

Congratulations. You have successfully written a Visual Basic macro in Microsoft Access!

What happened here was:

-) We have added a VB macro called "Hello" to our Access database, vb_tutorial.mdb.
-) We have added a VB procedure called "Main" in the VB macro. Access calls this procedure as a macro.
-) The "Main" procedure calls the "MsgBox" function, which is a VB built-in function that displays Windows dialog box with the specified text message.
-) We ran the "Main" procedure and got exactly what we expected.

Visual Basic .NET (VB 7)

The original Visual Basic .NET was released alongside Visual C# and ASP.NET in 2002. Significant changes broke backward compatibility with older versions and caused a rift within the developer community

Visual Basic .NET 2003 (VB 7.1)

Visual Basic .NET 2003 was released with *version 1.1* of the .NET Framework. New features included support for the .NET Compact Framework and a better VB upgrade wizard. Improvements were also made to the performance and reliability of the .NET IDE (particularly the background compiler) and runtime. In addition, Visual Basic .NET 2003 was available in the *Visual Studio .NET 2003 Academic Edition* (VS03AE). VS03AE is distributed to a certain number of scholars from each country without cost.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Visual Basic 2005 (VB 8.0)

Visual Basic 2005 is the name used to refer to the update to Visual Basic .NET, Microsoft having decided to drop the .NET portion of the title.

For this release, Microsoft added many features, including:

-) *Edit and Continue*
-) Design-time expression evaluation.
-) The *My* pseudo-namespace (overview, details), which provides:
 - easy access to certain areas of the .NET Framework that otherwise require significant code to access
 - dynamically-generated classes (notably *My.Forms*)
-) Improvements to the VB-to-VB.NET converter
-) The *Using* keyword, simplifying the use of objects that require the Dispose pattern to free resources
-) *Just My Code*, which when debugging hides (steps over) boilerplate code written by the Visual Studio .NET IDE and system library code
-) Data Source binding, easing database client/server development

The above functions (particularly *My*) are intended to reinforce Visual Basic .NET's focus as a rapid application development platform and further differentiate it from C#.

Visual Basic 2005 introduced features meant to fill in the gaps between itself and other "more powerful" .NET languages, adding:

-) .NET 2.0 languages features such as:
 - generics
 - Partial classes, a method of defining some parts of a class in one file and then adding more definitions later; particularly useful for integrating user code with auto-generated code
 - Nullable Types
-) XML comments that can be processed by tools like NDoc to produce "automatic" documentation
-) Operator overloading
-) Support for unsigned integer data types commonly used in other languages

'IsNot' operator patented

One other feature of Visual Basic 2005 is the IsNot operator that makes 'If X IsNot Y' equivalent to 'If Not X Is Y', which gained notoriety when it was found to be the subject of a Microsoft patent application.

Visual Basic 2005 Express

Part of the Visual Studio product range, Microsoft created a set of free development environments for hobbyists and novices, the Visual Studio 2005 Express series. One edition in the series is Visual Basic 2005 Express Edition, which was succeeded by Visual Basic 2008 Express Edition in the 2008 edition of Visual Studio Express.

The Express Editions are targeted specifically for people learning a language. They have a streamlined version of the user interface, and lack more advanced features of the standard versions. On the other hand, Visual Basic 2005 Express Edition *does* contain the Visual Basic 6.0 converter, so it is a way to evaluate feasibility of conversion from older versions of Visual Basic.

Visual Basic 2008 (VB 9.0)

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Visual Basic 9.0 was released together with the Microsoft .NET Framework 3.5 on November 19, 2007.

For this release, Microsoft added many features, including:

-) A true conditional operator, "If(boolean, value, value)", to replace the "Iif" function.
-) Anonymous types
-) Support for LINQ
-) Lambda expressions
-) XML Literals
-) Type Inference
-) Extension methods

Visual Basic 2010 (VB 10.0)

In April 2010, Microsoft released Visual Basic 2010. Microsoft had planned to use the Dynamic Language Runtime (DLR) for that release but shifted to a co-evolution strategy between Visual Basic and sister language C# to bring both languages into closer parity with one another. Visual Basic's innate ability to interact dynamically with CLR and COM objects has been enhanced to work with dynamic languages built on the DLR such as IronPython and IronRuby. The Visual Basic compiler was improved to infer line continuation in a set of common contexts, in many cases removing the " _" line continuation character. Also, existing support of inline Functions was complemented with support for inline Subs as well as multi-line versions of both Sub and Function lambdas.

Relation to older versions of Visual Basic (VB6 and previous)

Whether Visual Basic .NET should be considered as just another version of Visual Basic or a completely different language is a topic of debate. This is not obvious, as once the methods that have been moved around and that can be automatically converted are accounted for, the basic syntax of the language has not seen many "breaking" changes, just additions to support new features like structured exception handling and short-circuited expressions. Two important data type changes occurred with the move to VB.NET. Compared to VB6, the Integer data type has been doubled in length from 16 bits to 32 bits, and the Long data type has been doubled in length from 32 bits to 64 bits. This is true for all versions of VB.NET. A 16-bit integer in all versions of VB.NET is now known as a Short. Similarly, the Windows Forms GUI editor is very similar in style and function to the Visual Basic form editor.

The version numbers used for the new Visual Basic (7, 7.1, 8, 9, ...) clearly imply that it is viewed by Microsoft as still essentially the same product as the old Visual Basic.

The things that *have* changed significantly are the semantics—from those of an object-based programming language running on a deterministic, reference-counted engine based on COM to a fully object-oriented language backed by the .NET Framework, which consists of a combination of the Common Language Runtime (a virtual machine using generational garbage collection and a just-in-time compilation engine) and a far larger class library. The increased breadth of the latter is also a problem that VB developers have to deal with when coming to the language, although this is somewhat addressed by the *My* feature in Visual Studio 2005.

The changes have altered many underlying assumptions about the "right" thing to do with respect to performance and maintainability. Some functions and libraries no longer exist; others are available, but not as efficient as the "native" .NET alternatives. Even if they compile, most converted VB6 applications will require some level of refactoring to take full advantage of the new language. Documentation is available to cover changes in the syntax, debugging applications, deployment and terminology.

Comparative samples

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

The following simple example demonstrates similarity in syntax between VB and VB.NET. Both examples pop up a message box saying "Hello, World" with an OK button.

```
Private Sub Command1_Click()  
    MsgBox "Hello, World"  
End Sub
```

A VB.NET example, MsgBox or the MessageBox class can be used:

```
Public Class Form1  
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e  
        As System.EventArgs) Handles Button1.Click  
        MsgBox("Hello, World")  
    End Sub  
End Class
```

```
Public Class Form1  
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e  
        As System.EventArgs) Handles Button1.Click  
        MessageBox.Show("Hello, World")  
    End Sub  
End Class
```

-) Both Visual Basic 6 and Visual Basic .NET will automatically generate the Sub and End Sub statements when the corresponding button is clicked in design view. Visual Basic .NET will also generate the necessary Class and End Class statements. The developer need only add the statement to display the "Hello, World" message box.
-) Note that all procedure calls must be made with parentheses in VB.NET, whereas in VB6 there were different conventions for functions (parentheses required) and subs (no parentheses allowed, unless called using the keyword Call).
-) Also note that the names Command1 and Button1 are not obligatory. However, these are default names for a command button in VB6 and VB.NET respectively.
-) In VB.NET, the Handles keyword is used to make the sub Button1_Click a handler for the Click event of the object Button1. In VB6, event handler subs must have a specific name consisting of the object's name ("Command1"), an underscore ("_"), and the event's name ("Click", hence "Command1_Click").
-) There is a function called MsgBox in the Microsoft.VisualBasic namespace which can be used similarly to the corresponding function in VB6. There is a controversy about which function to use as a best practice (not only restricted to showing message boxes but also regarding other features of the Microsoft.VisualBasic namespace). Some programmers prefer to do things "the .NET way", since the Framework classes have more features and are less language-specific. Others argue that using language-specific features makes code more readable (for example, using int (C#) or Integer (VB.NET) instead of System.Int32).
-) In VB 2008, the inclusion of ByVal sender as Object, ByVal e as EventArgs has become optional.

The following example demonstrates a difference between VB6 and VB.NET. Both examples close the active window.

Classic VB Example:

```
Sub cmdClose_Click()  
    Unload Me  
End Sub
```

A VB.NET example:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
Sub btnClose_Click(ByVal sender As Object, ByVal e As EventArgs) Handles btnClose.Click
    Me.Close()
End Sub
```

Note the 'cmd' prefix being replaced with the 'btn' prefix, conforming to the new convention previously mentioned.

Visual Basic 6 did not provide common operator shortcuts. The following are equivalent:

VB6 Example:

```
Sub Timer1_Timer()
    Me.Height = Me.Height - 1
End Sub
```

VB.NET example:

```
Sub Timer1_Tick(ByVal sender As Object, ByVal e As EventArgs) Handles Timer1.Tick
    Me.Height -= 1
End Sub
Criticism
```

Long-time Visual Basic users have complained about Visual Basic .NET because initial versions dropped a large number of language constructs and user interface features that were available in VB6 (which is no longer sold by Microsoft), and changed the semantics of those that remained; for example, in VB.NET parameters are (by default) passed by value, not by reference. Detractors refer pejoratively to VB.NET as *Visual Fred* or *DOTNOT*. On March 8, 2005, a petition was set up in response to Microsoft's refusal to extend its mainstream support for VB6.

VB.NET's supporters state that the new language is in most respects more powerful than the original, incorporating modern object oriented programming paradigms in a more natural, coherent and complete manner than was possible with earlier versions. Opponents tend to respond that although VB6 has flaws in its object model, the cost in terms of redevelopment effort is too high for any benefits that might be gained by converting to VB.NET.[*citation needed*]

It is simpler to decompile languages that target Common Intermediate Language (CIL), including VB.NET, compared to languages that compile to machine code. Tools such as .NET Reflector can provide a close approximation to the original code due to the large amount of metadata provided in CIL.[*citation needed*]

Microsoft supplies an automated VB6-to-VB.NET converter with Visual Studio .NET, which has improved over time, but it cannot convert all code, and almost all non-trivial programs will need some manual effort to compile. Most will need a significant level of code refactoring to work optimally. Visual Basic programs that are mainly algorithmic in nature can be migrated with few difficulties; those that rely heavily on such features as database support, graphics, unmanaged operations or on implementation details are more troublesome.[*citation needed*]

In addition, the required runtime libraries for VB6 programs are provided with Windows 98 SE and above, while VB.NET programs require the installation of the significantly larger .NET Framework. The framework is included with Windows 7, Windows Vista, Windows XP Media Center Edition, Windows XP Tablet PC Edition, Windows Server 2008 and Windows Server 2003. For other supported operating systems such as Windows 2000 or Windows XP (Home or Professional Editions), it must be separately installed.

Microsoft's response to developer dissatisfaction has focused around making it easier to move new development and shift existing codebases from VB6 to VB.NET. Their latest offering is the VBRun website, which offers code samples and articles for:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

-) Using VB.NET to complete tasks that were common in VB6, like creating a print preview
-) Integrating VB6 and VB.NET solutions (dubbed *VB Fusion*)

Cross-platform and open-source development

The creation of open-source tools for VB.NET development have been slow compared to C#, although the Mono development platform provides an implementation of VB.NET-specific libraries and a VB.NET 8.0 compatible compiler written in VB.NET, as well as standard framework libraries such as Windows Forms GUI library.

SharpDevelop and MonoDevelop are open-source alternative IDEs.

Examples

The following is a very simple VB.NET program, a version of the classic "Hello world" example created as a console application:

```
Module Module1
```

```
    Sub Main()  
        Console.WriteLine("Hello, world!")  
    End Sub
```

```
End Module
```

The effect is to write the text *Hello, world!* to the command line. Each line serves a specific purpose, as follows:

```
Module Module1
```

This is a module definition, a division of code similar to a class, although modules can contain classes. Modules serve as containers of code that can be referenced from other parts of a program

It is common practice for a module and the code file, which contains it, to have the same name; however, this is not required, as a single code file may contain more than one module and/or class definition.

```
Sub Main()
```

This is the entry point where the program begins execution. *Sub* is an abbreviation of "subroutine."

```
Console.WriteLine("Hello, world!")
```

This line performs the actual task of writing the output. *Console* is a system object, representing a command-line interface and granting programmatic access to the operating system's standard streams. The program calls the *Console* method *WriteLine*, which causes the string passed to it to be displayed on the console. Another common method is using *MsgBox* (a *Message Box*).

Message Boxes are used to display information, error or questions. A Message Box contains a title, text and a button. Multiple types of Messages Boxes are available, which differ in the sound it is making and the picture. The buttons also change according to the type of Message Box used.

```
MessageBox.Show("Hello, World", "Title", MessageBoxButtons.YesNo, MessageBoxIcon.Question)
```

```
MsgBox("Hello World!", MsgBoxStyle.Question, "Title")
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

There are several different options available for the `MessageBox.Show` function. The Buttons and Icons available are:

Buttons
AbortRetryIgnore
OK
OKCancel
RetryCancel
YesNo
YesNoCancel

Icon Name	Icon Image
Asterisk	
Error	
Exclamation	
Hand	
Information	
Question	
Stop	
Warning	

The buttons have a different value. The Message Box can be saved into an Integer and then an if-statement can be used to do certain things depending on which button the user clicked. Example:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim result As Integer = MessageBox.Show("Do you want to ...?", "Title", MessageBoxButtons.YesNo,
    MessageBoxIcon.Asterisk)
    If result = 6 Then
        'yes was clicked
    Else
        'no was clicked
    End If
End Sub
```

The Integer is created by `Dim result As Integer`. `result` is the name of the Integer and can be freely chosen. The Message Box contains two buttons **MessageBoxButtons.YesNo** (Yes and No) the value of Yes is 6 and of No is 7. The following If-statement checks the value of the Integer `result` if it is equal to 6 then the first function will be called otherwise the function else is used.

The following table shows the value of each button.

Button	Value
OK	1

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Cancel	2
Abort	3
Retry	4
Ignore	5
Yes	6
No	7

Another way to use the Message Boxes with multiple answer choices shows the following example:

```
If MsgBox("The process will be started", MsgBoxStyle.OkCancel, "Title") = DialogResult.OK Then
    'ok was clicked
Else
    'cancel was clicked
End If
```

Using a Message Box without a variable is also possible by creating the Message Box in the If-statement. The MessageBox button OK corresponds to the *DialogResult.OK* code. Thus the program runs a function depending on what Button the User clicked.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

12.MYSQL, Visual Foxpro

A **Database Management System (DBMS)** is a set of computer programs that controls the creation, maintenance, and the use of a database. It allows organizations to place control of database development in the hands of database administrators (DBAs) and other specialists. A DBMS is a system software package that helps the use of integrated collection of data records and files known as databases. It allows different user application programs to easily access the same database. DBMSs may use any of a variety of database models, such as the network model or relational model. In large systems, a DBMS allows users and other software to store and retrieve data in a structured way. Instead of having to write computer programs to extract information, user can ask simple questions in a query language. Thus, many DBMS packages provide Fourth-generation programming language (4GLs) and other application development features. It helps to specify the logical organization for a database and access and use the information within a database. It provides facilities for controlling data access, enforcing data integrity, managing concurrency, and restoring the database from backups. A DBMS also provides the ability to logically present database information to users.

Visual FoxPro is a data-centric object-oriented and procedural programming language produced by Microsoft. It is derived from FoxPro (originally known as **FoxBASE**) which was developed by Fox Software beginning in 1984. Fox Technologies merged with Microsoft in 1992, after which the software acquired further features and the prefix "Visual". The last version of FoxPro (2.6) worked under Mac OS, DOS, Windows, and Unix: Visual FoxPro 3.0, the first "Visual" version, dropped the platform support to only Mac and Windows, and later versions were Windows-only. The current version of Visual FoxPro is COM-based and Microsoft has stated that they do not intend to create a Microsoft .NET version.

FoxPro originated as a member of the class of languages commonly referred to as "xBASE" languages, which have syntax based on the dBase programming language. Other members of the xBase language family include Clipper and Recital. (A history of the early years of xBase can be found in the dBase entry.)

Visual FoxPro, commonly abbreviated as VFP, is tightly integrated with its own relational database engine, which extends FoxPro's xBase capabilities to support SQL query and data manipulation. Unlike most database management systems, Visual FoxPro is a full-featured, dynamic programming language that does not require the use of an additional general-purpose programming environment. It can be used to write not just traditional "fat client" applications, but also middleware and web applications.

What Is SQL? SQL (Structured Query Language) is a computer language that allows users to interact with Relational Database Management Systems (RDBMS) to define data type and structure, and to insert, update or delete data instances.

Usually, an RDBMS system will offer a user interface (UI) to allow you to issue SQL statements to directly operate on the RDBMS system:

User <--> UI <--SQL--> RDBMS <--> Data storage

A typical RDBMS system does also offer Application Programming Interface (API) to allow application programs to use SQL statements to interact with the RDBMS system:

User <--> Application <--SQL--> RDBMS <--> Data storage

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

SQL is a non-procedural language. Each SQL statement is an individual execution unit, independent of other statements. There is no conditional statements, jumping statements or looping statements to group multiple statements together into a complex execution unit. There is no way to define a procedure of statements, and no procedure call statements.

In the previous section, we learned how to start and shutdown MySQL server. Now let's see how we can use MySQL client interface to create a table and run queries. First start the MySQL server in one command window, then run the start the MySQL client interface in another command window, and run the following MySQL client commands:

```
\mysql\bin\mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 4.0.18-max-debug
```

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

```
mysql> show databases;
```

```
+-----+
| Database |
+-----+
| mysql   |
| test    |
+-----+
```

2 rows in set (0.06 sec)

```
mysql> use test;
```

Database changed

```
mysql> show tables;
```

Empty set (0.00 sec)

```
mysql> create table hello (message varchar(80));
```

Query OK, 0 rows affected (0.58 sec)

```
mysql> insert into hello (message) values ('Hello world!');
```

Query OK, 1 row affected (0.38 sec)

```
mysql> select * from hello;
```

```
+-----+
| message  |
+-----+
| Hello world! |
+-----+
```

1 row in set (0.04 sec)

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
mysql> drop table hello;  
Query OK, 0 rows affected (0.34 sec)  
  
mysql> quit;  
Bye
```

A stored procedure can have local variables.

To define a local variable, you can use the DECLARE statement:

```
DECLARE variable data_type [DEFAULT value];
```

To assign a new value to a variable, you can use the SET statement:

```
SET variable = expression;
```

The SELECT statement can also be used to assign values to variables:

```
SELECT expression, expression, ... INTO variable, variable, ...  
[FROM clause];
```

Once a variable is defined, it can be used in any expressions in any statements

To selectively executing a group of statements, you can use the IF statement:

```
IF condition THEN  
    statement_list  
ELSE IF condition THEN  
    statement_list  
ELSE IF condition THEN  
    statement_list  
.....  
ELSE  
    statement_list  
END IF
```

To repeatedly executing a group of statements without any conditions, you can use the LOOP statement:

```
LOOP  
    statement_list  
END LOOP
```

To repeatedly executing a group of statements with an ending condition, you can use the REPEAT statement:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
REPEAT
  statement_list
UNTIL condition
END REPEAT
```

To repeatedly executing a group of statements with an ending condition, you can use the WHILE statement:

```
WHILE condition
  statement_list
END WHILE
```

Below is a sample code that uses a while loop to insert multiple rows into a table:

```
-- ProcedureLoop.sql
-- Copyright (c) 2004 by Dr. Herong Yang
--
DROP DATABASE IF EXISTS HyTest;
CREATE DATABASE HyTest;
USE HyTest;
--
DROP PROCEDURE IF EXISTS InitTable;
DELIMITER '/';
CREATE PROCEDURE InitTable(IN N INTEGER)
BEGIN
  DECLARE I INTEGER;
  SET I = 0;
  WHILE I < N DO
    INSERT INTO MyTable VALUES (I, RAND()*N);
    SET I = I + 1;
  END WHILE;
END/
DELIMITER '/';
--
DROP TABLE IF EXISTS MyTable;
CREATE TABLE MyTable (ID INTEGER, Value INTEGER);
CALL InitTable(100);
--
SELECT 'My table:' AS '---';
SELECT * FROM MyTable WHERE ID < 10;
```

Output:

```
My table:
ID      Value
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

0	3
1	63
2	8
3	52
4	35
5	18
6	86
7	75
8	18
9	63

Using DDL to Create Tables and Indexes

-) DLL (Data Definition Language) contains CREATE, ALTER and DROP statements.
-) CREATE TABLE statements are used to create tables.
-) CREATE INDEX statements are used to create indexes.
-) ALTER TABLE statements are used to alter table structures.

The create table statement allows you to create a new table in the database. It has a number of syntax formats:

1. To create a permanent table:

```
CREATE TABLE tbl_name (column_list);
```

where "column_list" defines the columns of the table with the following syntax:

```
col_name col_type col_options,  
col_name col_type col_options,  
...  
col_name col_type col_options
```

2. To create a temporary table:

```
CREATE TEMPORARY TABLE tbl_name (column_list);
```

3. To create a permanent table if it doesn't exist:

```
CREATE TABLE tbl_name IF NOT EXISTS (column_list);
```

4. To create a permanent table with data types and data generated from a select statement:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
CREATE TABLE tbl_name select_statement;
```

To show the columns of an existing table, you can use the show columns statement:

```
SHOW COLUMNS FROM tbl_name;
```

To delete an existing table, you can use the drop table statement:

```
DROP TABLE tbl_name;
```

Here is an example SQL code, CreateTable.sql, showing you how to create a table by selecting data from existing tables:

CreateTable.sql

```
CREATE TABLE IF NOT EXISTS User (Login VARCHAR(8), Password CHAR(8));
INSERT INTO User VALUES ('herong','8IS3K0XW');
INSERT INTO User VALUES ('mike','PZ0JG');
SELECT 'User table: ';
SHOW COLUMNS FROM User;
SELECT * FROM User;
--
CREATE TABLE IF NOT EXISTS UserCopy SELECT * FROM User;
SELECT 'UserCopy table: ';
SHOW COLUMNS FROM UserCopy;
SELECT * FROM UserCopy;
--
CREATE TABLE IF NOT EXISTS UserDump
  SELECT CONCAT(Login,':') AS Login, CHAR_LENGTH>Password) AS Count
  FROM User;
SELECT 'UserCopy table: ';
SHOW COLUMNS FROM UserDump;
SELECT * FROM UserDump;
--
DROP TABLE User;
DROP TABLE UserCopy;
DROP TABLE UserDump
```

If you run the code, you will get:

User table:

User table:

Field	Type	Null	Key	Default	Extra
-------	------	------	-----	---------	-------

Login	varchar(8)	YES	NULL		
-------	------------	-----	------	--	--

Password	varchar(8)	YES	NULL		
----------	------------	-----	------	--	--

Login	Password				
-------	----------	--	--	--	--

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
herong 8IS3K0XW
mike PZ0JG
UserCopy table:
UserCopy table:
Field Type Null Key Default Extra
Login varchar(8) YES
Password varchar(8) YES
Login Password
herong 8IS3K0XW
mike PZ0JG
UserDump table:
UserDump table:
Field Type Null Key Default Extra
Login char(9) YES
Count int(10) YES
Login Count
herong: 8
mike: 5
```

A couple of interesting notes on the output:

-) The show columns statement reports that MySQL sets all columns to be variable length, if one column is variable length.
-) The output of the UserCopy table shows that the create table statement with select sub-statement works perfectly.
-) The output of the UserDump table shows that the columns types are based the data types of the output of the select sub-statement, if it used in the create table statement

An insert statement allows you to insert new rows of data into an existing table. It has a number of syntax formats:

1. To insert a single row of all columns with values resulting from the specified expressions:

```
INSERT INTO tbl_name VALUES (expression, expression, ...);
```

When executed, all expressions will be evaluated, and the resulting values will form the new row, which will be inserted into the specified table. Of course, the number of expressions must be equal to the number of columns in table.

2. To insert a single row with some columns having specified values, and others having default values:

```
INSERT INTO tbl_name (column, column, ...) VALUES (expression,
expression, ...);
```

Notes:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

-) Obviously, duplicated columns are not allowed in the column list.
-) The number of expressions must be equal to the number of specified columns.
-) Default values will be provided for those columns that are not specified in the column list.

3. To insert one or more rows of all columns with a select sub-statement:

```
INSERT INTO tbl_name select_statement;
```

When executed, the output rows of the select sub-statement will be inserted into the specified table. Of course, the number of select expressions in the select statement must be equal to the number of columns of the specified table.

4. To insert one or more rows with some column having values from the specified select sub-statement, and other columns having default values:

```
INSERT INTO tbl_name (column, column, ...) select_statement;
```

Notes:

-) Obviously, duplicated columns are not allowed in the column list.
-) The number of select expressions must be equal to the number of specified columns.
-) Default values will be provided for those columns that are not specified in the column list.

Here is an example SQL code, InsertRows.sql, showing you how to insert rows into an existing table:

```
-- InsertRows.sql
-- Copyright (c) 1999 by Dr. Herong Yang
--
DROP TABLE IF EXISTS User;
CREATE TABLE User (Login VARCHAR(8), Password CHAR(8));
INSERT INTO User VALUES ('herong','8IS3K0XW');
INSERT INTO User (Login) VALUES ('mike');
INSERT INTO User SELECT Login, Password FROM User;
INSERT INTO User (Password) SELECT CONCAT('__',Login) FROM User;
SELECT 'User table:' AS '---';
SELECT * FROM User;
```

If you run the code, you will get:

```
---
User table:
Login Password
herong 8IS3K0XW
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
mike NULL
herong 8IS3K0XW
mike NULL
NULL __herong
NULL __mike
NULL __herong
NULL __mike
```

A delete statement allows you to delete existing rows from an existing table. The syntax of a delete statement is:

```
DELETE FROM tbl_name [WHERE clause]
```

If executed, all rows that satisfy the condition in the where clause will be deleted. If no "where clause" specified, all rows will be deleted.

Here is an example SQL code, DeleteRows.sql, showing you how to delete rows from an existing table:

```
-- DeleteRows.sql
-- Copyright (c) 1999 by Dr. Herong Yang
--
DROP TABLE IF EXISTS User;
CREATE TABLE User (Login VARCHAR(8), Password CHAR(8));
INSERT INTO User VALUES ('herong','8IS3K0X');
INSERT INTO User (Login) VALUES ('mike');
DELETE FROM User WHERE Login = 'herong';
SELECT 'User table:' AS '---';
SELECT * FROM User;
```

If you run the code, you will get:

```
---
User table:
Login Password
mike NULL
```

-) SELECT statements returns data in rows and fields from base tables.
-) The FROM clause defines the base table, filter, aggregation and sorting order.
-) The WHERE clause defines filter conditions.
-) The GROUP BY clause defines aggregation rules.
-) The ORDER BY clause defines sorting orders.
-) The JOIN operation defines how two tables can be combined into a single base table.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

A select statement is also called a query statement. It is normally used to retrieve rows of data selected from specified tables. The generic syntax of a select statement is:

```
SELECT select_expression_list [FROM clause]
```

where "expression_list" defines a list of select expressions, and "FROM clause" defines a select table with rows and columns of values. Column names of the select table can be used as variables in select expressions to represent column values in each row.

When a select statement is executed, a nested loop logic will be performed to generate rows of output data:

Loop on each row of the select table, do:

Loop on each select expression, do:

Evaluate this expression with possible column values in current row of the select table.

End of loop

Return the results of all select expressions as a row of output data

End of loop

Note:

-) The number of columns of an output row equals to the number of select expressions.
-) The number of rows of the output data equals to the number of rows of the select table.
-) If the select table has no rows, no data will be returned.
-) The select table is optional. If it is not specified, only one row of data will be returned.

Sample select statements without FROM clause:

```
SELECT 'Hello world!';  
SELECT 'Apple', 'Orange';  
SELECT CHAR_LENGTH('Hello world!');  
SELECT 1, 4, 9, 16, 25;  
SELECT PI();  
SELECT CURRENT_DATE(), CURRENT_TIME();
```

A join table is the output table of a join operation on two tables. There are several types of join operations:

1. Cross Join - Takes each row in the left table, and joins onto all rows in the right table. The number of columns of the output table is the number of columns of the left table plus the number of columns of the right table. The number of rows of the output table is the number of rows of the left table times the number of rows of the right table. A cross join operation is also called Cartesian product operation. There are two ways to write a cross join table:

```
table_1, table_r  
table_1 CROSS JOIN table_r
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

The cross join operation logic can be expressed as:

```
Loop on each row in the left table (L)
  Loop on each row in the right table (R)
    Generate an output row with all columns of the current row of L
      and all columns of the current row of R
  End of loop
End of loop
```

2. Inner Join - Takes the output table of the cross join operation, and filter out rows that do not satisfy the specified join condition, which should be an equality comparison of one column in the left table and one column in the right table. The syntax form of an inner join table is:

```
table_l INNER JOIN table_r ON table_l.column_l=table_r.column_r
```

The inner join operation logic can be expressed as:

```
Loop on each row in the left table (L)
  Loop on each row in the right table (R)
    If the value of column_l equals to the value of column_r
      Generate an output row with all columns of the current row
        of L and all columns of the current row of R
      Break the loop on R
    End if
  End of loop
End of loop
```

3. Left Outer Join - Takes the output table of the inner join operation, and adds one row for each row in the left table that has no matching rows in the right table. This new output row will contain the row from the left table and null values to occupy the output columns corresponding to the right table. The syntax form of a left outer join table is:

```
table_l LEFT OUTER JOIN table_r ON table_l.column_l=table_r.column_r
```

The left outer join operation logic can be expressed as:

```
Loop on each row in the left table (L)
  Set match_found = FALSE
  Loop on each row in the right table (R)
    If the value of column_l equals to the value of column_r
      Generate an output row with all columns of the current row
        of L and all columns of the current row of R
      Set match_found = TRUE
    End if
  End of loop
  If match_found is FALSE
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
Generate an output row with all columns of the current row of L
and null values for columns corresponding to columns of R
```

```
End if
```

```
End of loop
```

4. Right Outer Join - Takes the output table of the inner join operation, and adds one row for each row in the right table that has no matching rows in the left table. This new output row will contain the row from the right table and null values to occupy the output columns corresponding to the left table. The syntax form of a right outer join table is:

```
table_l RIGHT OUTER JOIN table_r ON table_l.column_l=table_r.column_r
```

The right outer join operation logic can be expressed as:

```
Loop on each row in the right table (R)
```

```
Set match_found = FALSE
```

```
Loop on each row in the left table (L)
```

```
If the value of column_l equals to the value of column_r
```

```
Generate an output row with all columns of the current row
of L and all columns of the current row of R
```

```
Set match_found = TRUE
```

```
End if
```

```
End of loop
```

```
If match_found is FALSE
```

```
Generate an output row with all columns of the current row of R
and null values for columns corresponding to columns of L
```

```
End if
```

```
End of loop
```

To validate the join table logics mentioned in the previous section, I wrote the following SQL code, JointTable.sql:

```
-- JointTable.sql
-- Copyright (c) 1999 by Dr. Herong Yang
--
-- Creating user table
DROP TABLE IF EXISTS User;
CREATE TABLE User (ID INT, Login CHAR(8), Dept_ID INT);
INSERT INTO User VALUES (1,'herong',1);
INSERT INTO User VALUES (2,'mike',2);
INSERT INTO User VALUES (3,'bill',3);
INSERT INTO User VALUES (4,'mary',3);
INSERT INTO User VALUES (5,'lisa',5);
-- Creating dept table
DROP TABLE IF EXISTS Dept;
CREATE TABLE IF NOT EXISTS Dept (ID INT, Name CHAR(16));
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
INSERT INTO Dept VALUES (1,'Math');
INSERT INTO Dept VALUES (3,'Chem');
INSERT INTO Dept VALUES (4,'Law');
INSERT INTO Dept VALUES (5,'English');
INSERT INTO Dept VALUES (5,'Latin');
-- Generating join tables
SELECT 'Running cross join' AS '---';
SELECT * FROM User CROSS JOIN Dept;
SELECT 'Running inner join' AS '---';
SELECT * FROM User INNER JOIN Dept ON User.Dept_ID=Dept.ID;
SELECT 'Running left outer join' AS '---';
SELECT * FROM User LEFT OUTER JOIN Dept ON User.Dept_ID=Dept.ID;
SELECT 'Running right outer join' AS '---';
SELECT * FROM User RIGHT OUTER JOIN Dept ON User.Dept_ID=Dept.ID;
```

Note that:

-) "*" can be used as selection expression. It will be evaluated to a list of values from all columns in the select table.
-) A select expression will have system default column name in the output table. You can change the default column name by using "AS new_name" option.
-) When referring to a column in a table, you can use the fully quantified column name: table_name.column_name.

Output of JoinTabl.sql:

```
---
Running cross join
ID  Login  Dept_ID ID  Name
1   herong 1    1   Math
2   mike   2    1   Math
3   bill   3    1   Math
4   mary   3    1   Math
5   lisa   5    1   Math
1   herong 1    3   Chem
2   mike   2    3   Chem
3   bill   3    3   Chem
4   mary   3    3   Chem
5   lisa   5    3   Chem
1   herong 1    4   Law
2   mike   2    4   Law
3   bill   3    4   Law
4   mary   3    4   Law
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
5 lisa 5 4 Law
1 herong 1 5 English
2 mike 2 5 English
3 bill 3 5 English
4 mary 3 5 English
5 lisa 5 5 English
1 herong 1 5 Latin
2 mike 2 5 Latin
3 bill 3 5 Latin
4 mary 3 5 Latin
5 lisa 5 5 Latin
```

Running inner join

```
ID Login Dept_ID ID Name
1 herong 1 1 Math
3 bill 3 3 Chem
4 mary 3 3 Chem
5 lisa 5 5 English
5 lisa 5 5 Latin
```

Running left outer join

```
ID Login Dept_ID ID Name
1 herong 1 1 Math
2 mike 2 NULL NULL
3 bill 3 3 Chem
4 mary 3 3 Chem
5 lisa 5 5 English
5 lisa 5 5 Latin
```

Running right outer join

```
ID Login Dept_ID ID Name
1 herong 1 1 Math
3 bill 3 3 Chem
4 mary 3 3 Chem
NULL NULL NULL 4 Law
5 lisa 5 5 English
5 lisa 5 5 Latin
```

Notes on the output:

) Surprisingly, the cross join was performed with the outer loop on the right table columns. This is different than my expectation.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

) Inner join, left outer join, and right outer join were performed as expected.

As I mentioned earlier, the WHERE clause modifies the base table by filtering out rows of data that do not satisfy the specified conditions. Here is its syntax:

```
WHERE where_condition
```

where "where_condition" is a predicate operation that will result a true or false condition.

WHERE clause samples:

```
WHERE Salary <= 45000.00
```

```
WHERE Salary > 45000.00 AND Salary <= 65000.00
```

```
WHERE Dept IN ('Math','Chem')
```

```
WHERE (Dept = 'Math' OR Dept = 'Chem') AND Salary <= 45000.00
```

```
WHERE User.Dept_ID = Dept.ID
```

"GROUP BY clause" modifies the base table by grouping original rows into group rows based on identical combined values of the specified group columns. In other words, each resulting row represents a group of original rows that has a unique combination of the values in the specified group columns. Original columns are reduced to the specified group columns only. Group rows can also be filtered out by a specified condition. "GROUP BY clause" syntax is:

```
GROUP BY group_columns [HAVING having_condition]
```

where "group_columns" is a list of columns in the original base table, and "having_condition" is a predicate operation that will result a true or false condition.

Rule 1: Two types of data can be used in select expressions: 1. group columns; 2. a group function of any original columns. Group functions are:

) COUNT(column): Number of original records in the group represented by this resulting record. Actually, the COUNT() will produce the same number regardless of the specified field.

) SUM(column): The sum of all values of the specified column in the group represented by this resulting row.

) MIN(column): The minimum value of the specified column in the group represented by this resulting row.

) MAX(column): The maximum value of the specified column in the group represented by this resulting row.

) AVG(column): The average value of the specified column in the group represented by this resulting row.

For examples, the following is nice salary statistics report per department:

```
SELECT Department, COUNT(Name) AS NumberOfEmployees,  
MIN(Salary) AS MinimumSalary, MAX(Salary) AS MaximumSalary,  
AVG(Salary) as AverageSalary  
FROM Employee WHERE Status='Active' GROUP BY Department
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Rule 2: If multiple group columns are used, rows are grouped into a single rows based the identical combined values of the group columns, not individual identical values. For example, the following statement reports age statistics per department and per sex:

```
SELECT Department, Sex, COUNT(Name) AS NumberOfEmployees,  
MIN(Salary) AS MinimumSalary, MAX(Salary) AS MaximumSalary,  
AVG(Salary) as AverageSalary  
FROM Employee WHERE Status='Active' GROUP BY Department, Sex
```

If there are 10 individual departments, you will get 20 records, assuming that every department has both sexes.

Rule 3: If a having condition is specified, it will be used to filter out the resulting group rows that do not satisfy this condition. Since the having condition is applied on the grouped rows, it can only use group columns and group functions. For example, the following statement report salary statistics only for those departments that have more than 10 active employees:

```
SELECT Department, COUNT(Name) AS NumberOfEmployees,  
MIN(Salary) AS MinimumSalary, MAX(Salary) AS MaximumSalary,  
AVG(Salary) as AverageSalary  
FROM Employee WHERE Status='Active'  
GROUP BY Department HAVING COUNT(Name)>10
```

The following is bad example, "Sex='Male'" can only be used in the WHERE clause, not in the HAVING clause:

```
SELECT Department, COUNT(Name) AS NumberOfEmployees,  
MIN(Salary) AS MinimumSalary, MAX(Salary) AS MaximumSalary,  
AVG(Salary) as AverageSalary  
FROM Employee WHERE Status='Active'  
GROUP BY Department HAVING sex='Male'
```

"ORDER BY clause" modifies the base table by sorting rows according the specified order. "ORDER BY clause" syntax is:

```
ORDER BY order_exp, order_exp, ...
```

where "order_exp" specify a single order expression. If multiple order expressions are specified, the order expression on the left has higher precedence than the one on the right. This means the order expression on the right will only be used to sort rows that are having the same for the order expression on the left.

If ORDER BY clause is used with GROUP BY clause, it must contain only group columns or group functions. For example, the following statement shows us which department has the oldest average age:

```
SELECT department, COUNT(name) AS numberOfEmployees,  
min(age) AS minimumAge, max(age) AS maximumAge,  
AVG(age) as averageAge  
FROM employee WHERE status='Active' GROUP BY department  
ORDER BY AVG(age) DESC
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Using DML to Insert, Update and Delete Records

-) DML (Data Manipulation Language) contains INSERT, UPDATE and DELETE statements.
-) INSERT INTO statements are used to insert records into tables.
-) UPDATE statements are used to update records stored in tables.
-) DELETE FROM statements are used to delete records from tables.

13. PROJECTS

13.1 Student information system for your college.

PROJECT NO 1

Name of the project

1. Design a project of student information system for the college

Objective: for designing the student information system follow these steps:

- a) identify the problem
- b) requirement analysis
- c) preparing E.R diagram(entity relationship diagram)
- d) design DFD(data flow diagram)
- e) create database(table structure)
- f) normalized the database including analysis of functional dependencies
- g) define the relationship between tables

Requirement analysis:

Collect the information from the various sources and the rearrange the information according to the demand of project.

ER DIAGRAM:

An **entity-relationship diagram (ERD)** is an abstract and conceptual representation of data.

DATA FLOW DIAGRAM:

A **data-flow diagram (DFD)** is a graphical representation of the "flow" of data through an information system. DFDs can also be used for the visualization of data processing (structured design).

On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process.

DATABASE:

It is a collection of tables. Tables are a combination of rows and columns. Tables stored the data which is useful and meaningful.

DATABASE MANAGEMENT SYSTEM:

A system which collect the data into the database and performed these operations

- a) store the data into the database
- b) retrieve the data from the database

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

- c) modify the data into the database
- d) delete the unnecessary data

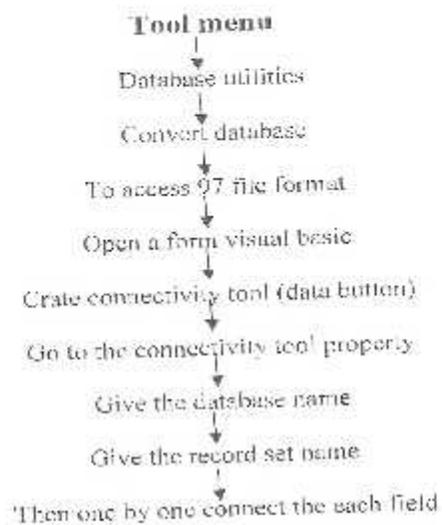
NORMALIZATION:

Normalization is a process which converts the complex database into normal form. It converts the large tables into small tables.

Front end: the application or software which is used to design the graphical part of software (project) is called front end, i.e. Visual basic 6.0

Backend: the application or software which is used to maintain a database of project or where we stored the data (meaningful information) of the project called backend i.e. MS-Access or SQL.

Connectivity steps:



Steps for creating a project:

1. First design forms.
2. Create MDI FORM and link all form in the MDI Form.
3. Create tables in MS-Access.
4. Connect the tables and forms using the connectivity tools.
5. Write the appropriate code.
6. Test the code and verify with user requirement.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

MDI form: the multiple document interfaces designed to simplify the exchange of information among document all under the same roof. With the main application maintain the multiple copies of the application data exchange is easier.

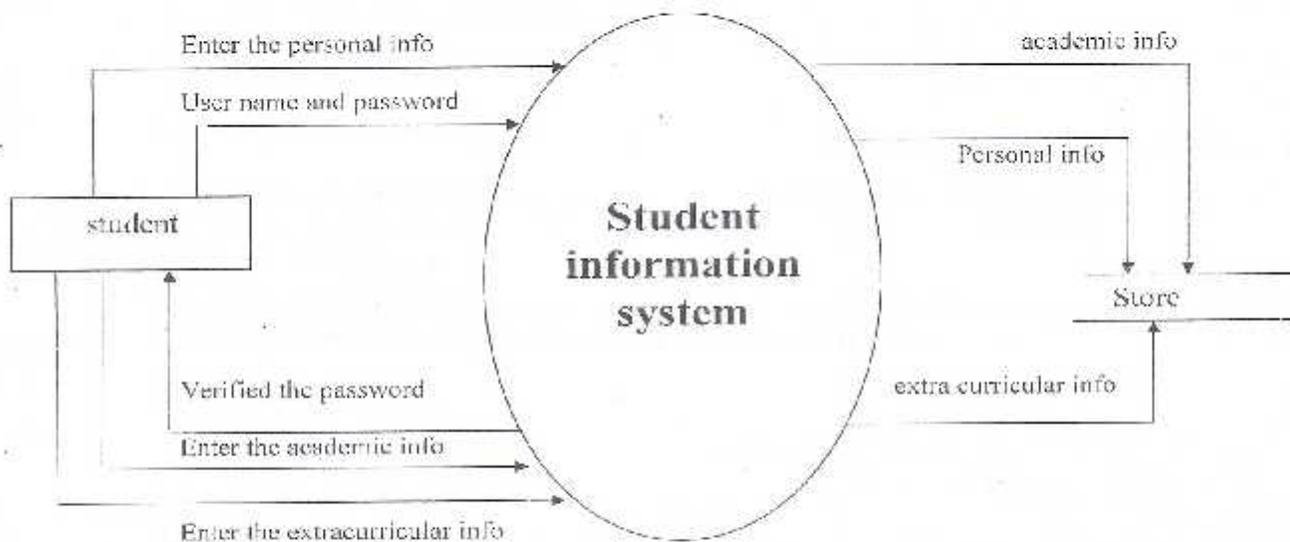
Description of the of the project

The student information system is collecting the all information of the students. In formations are divided into 3 parts:

- a) personal information
- b) academic information
- c) Extra-curricular activity (achievements, prizes etc.)

If we need any information of a particular student then we write the name of the student and related query then all the information are retrieve from the database also generate the reports.

DFD OF STUDENT INFORMATION SYSTEM



O level of DFD OF STUDENT INFORMATION SYSTEM

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Design the Form for Entry Level

STUDENT PERSONAL DETAIL		ACADEMIC DETAIL						
REG NO	07E3KIT047	X%	51	<I%	52	PCM%		
STUDENT NAME	SOML KUMAR	OTHER QUALIFICATION (IF ANY)						
STUDENTS FATHER NAME	Jambabuprasad							
HOME PHONE NO	9334411281							
PERMANENT ADDRESS	Post Pared P.S. Dilta							
DIST	Patna							
STATE	Bihar							
PN NO	001103							
DATE OF BIRTH	15-6-87							
HOSTLE	<input type="checkbox"/>							
STUDENT MOBILE NO	9334411201							
		TECHNICAL EDUCATION DETAIL						
		SESSION	2007-2010	BRANCH	IT			
		SEMESTER WISE PERCENTAGE						
		I	II	III	IV			
		63	52	64	56			
		V	VI	VII	VIII			
		[Red Box]	[Red Box]	[Red Box]	[Red Box]			
		NO OF JACK	TRAINING	PROJECT				
		[Red Box]	[Red Box]	[Red Box]				
MOVE FIRST	MOVE LAST	NEXT RECORD	PREVIOUS RECORD	NEW ENTRY	FIND ENTRY	DELETE ENTRY	PRINT	EXIT

Design form for reports & filtration

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Language Team

Institute of Engineering Subject Studies & Projects

SESSION

2007-2008

BRANCH

Computer Science

Run

Print

Exit

REG NO	STUDENT NAME	STUDENT FATHER NAME	HOME PHONE NO	PERMANENT ADDRESS	CITY	STATE
07EGKCS001	AKASH PAL	CHITEDI PAL		C-30/211-4, TULSIPUR NAYA TANASI		J.P
07EGKCS002	AKHILESH KLMAR BH					
07EGKCS003	AKILESH KUN DUDCY	OMPTAKASH DUDCY	011-27267047	A-119 DUDHIPUR P.O A DEL II		DELHI
07EGKCS005	AMAN RAJ	SWAY PRASHAD YADAV		E&BU GANJ GAISHA T G	Alamgung	palna
07EGKCS006	AMIT AGARWAL					
07EGKCS007	AMIT WISHWASH					
07EGKCS008	ANKIT SACHAN					
07EGKCS009	ATUL SINGH SULKI	ATUL SINGH SULKI	RAJENDRA SINGH SU		529-A.H.K. PURAV KJ	KOTA
07EGKCS010	BHARAT MITAL	RAJENDRA PRASHAD M	0143-2310809	141, GYAN VIHAR COLO	AJMER	RAJASTHAN
07EGKCS011	BIKASH KUMAR					
07EGKCS012	DEBARPAK SHOSH					
07EGKCS013	DEEPAK CHAJCHARY	RAJ CHAND CHOUHARY	9799251536	VILL. BHUJAL PURA, THUNK		JAIPUR
07EGKCS014	DEEPIKA SAKENA	ANILKUMAR SHAFMA		2-K'S TEACHER COLON	KOTA	RAJASTHAN
07EGKCS015	DEVANSHU KLMAR					
07EGKCS016	CHARMEENDEEJ CUC					
07EGKCS017	DEGVIJAY SINGH	SANJEEV SINGH		VILLAGE KANDHOLI PO	BAHFATPUR	RAJASTHAN
07EGKCS018	ELSHAD AHMAD					
07EGKCS019	GOVIND KUMAR GAFO					
07EGKCS020	IRFESH CHAJCHAN	JAIJI CHAJCHAN	0744-3200808	SHR. RAMNABAR OTI	KOTA	RAJASTHAN
07EGKCS021	HIVANSHU SINGH	M.D. NATHUNI MANSIJI		VILLAGE MAFFA SIRPA	SITAMPJHI	3 HAR
07EGKCS022	MAHFF SINGH					
07EGKCS023	M.D. SAHID MANSURI					
07EGKCS024	MOHIT GAHTAV					

SELECT * FROM table1 where session='2007-2008' and branch='CS'

Write down the appropriate code for Student information system like below ex.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

The screenshot displays the Microsoft Visual Basic IDE in Design view. The main workspace is divided into two code windows: 'Project1 - Form2 (Code)' and 'Project1 - Form6 (Code)'. The 'Project1 - Form2 (Code)' window contains the following code:

```
Private Sub Command1_Click()  
Data1.Recordset.MoveNext  
T1x =  
MsgBox ("End of Record")  
End If  
  
End Sub  
  
Private Sub Command2_Click()  
If Data1.Recordset.EOF = False Then  
Data1.Recordset.MovePrevious  
T1x =  
MsgBox ("End of Record")  
End If  
  
End Sub  
  
Private Sub Command3_Click()  
Data1.Recordset.AddNew  
End Sub  
  
Private Sub Command4_Click()  
Unload Me  
Form6.Show  
End Sub  
  
Private Sub Command5_Click()  
Unload Me  
Form1.Show  
End Sub  
  
Private Sub Command6_Click()  
Unload Me  
Form6.Show  
End Sub
```

The 'Project1 - Form6 (Code)' window contains the following code:

```
Private Sub Command1_Click()  
If c1.Text = "2011- 2012" And c2.Text = "Computer Science" Then  
s1.RecordSource = "SELECT * FROM table1 where session='11- 2011' and branch='"  
s1.Refresh  
s1.Caption = s1.RecordSource  
End If  
  
If c1.Text = "2012- 2013" And c2.Text = "Information Technology" Then  
s1.RecordSource = "SELECT * FROM table1 where session='2012- 2013' and branch='"  
s1.Refresh  
s1.Caption = s1.RecordSource  
End If  
  
If c1.Text = "2013- 2014" And c2.Text = " " Then  
s1.RecordSource = "SELECT * FROM table1"  
s1.Refresh  
s1.Caption = s1.RecordSource  
End If  
  
If c1.Text = "2014- 2015" And c2.Text = " " Then  
s1.RecordSource = "SELECT * FROM table1"  
s1.Refresh  
s1.Caption = s1.RecordSource  
End If  
  
If c1.Text = "2015- 2016" And c2.Text = " " Then  
s1.RecordSource = "SELECT * FROM table1"  
s1.Refresh  
s1.Caption = s1.RecordSource  
End If  
  
If c1.Text = "2016- 2017" And c2.Text = " " Then  
s1.RecordSource = "SELECT * FROM table1"  
s1.Refresh  
End If  
  
If c1.Text = "2017- 2018" And c2.Text = " " Then  
s1.RecordSource = "SELECT * FROM table1"  
s1.Refresh  
s1.Caption = s1.RecordSource  
End If  
  
If c1.Text = "2018- 2019" And c2.Text = " " Then  
s1.RecordSource = "SELECT * FROM table1"  
s1.Refresh  
End If
```

The Project Explorer on the right shows the project structure for 'Project1 (Project1.vbp)'. It includes a 'Forms' folder with 'Form1 (Form1.frm)' through 'Form7 (Form7.frm)', and a 'Reports' folder with 'DataReport1 (DataReport1.Dsr)' through 'DataReport8 (DataReport8.Dsr)'. The 'Form6 (Form6.frm)' file is currently selected.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

13.3 A video/ library management system for a shop.

INTRODUCTION

This report will provide a detailed account of the processes our group used to design and implement a database that can be used to manage a library system. Each subsection of the report corresponds to an important feature of database design.

REQUIREMENT ANALYSIS

A library database needs to store information pertaining to its users (or customers), its workers, the physical locations of its branches, and the media stored in those locations. We have decided to limit the media to two types: books and videos. The library must keep track of the status of each media item: its location, status, descriptive attributes, and cost for losses and late returns. Books will be identified by their ISBN, and movies by their title and year. In order to allow multiple copies of the same book or video, each media item will have a unique ID number. Customers will provide their name, address, phone number, and date of birth when signing up for a library card. They will then be assigned a unique user name and ID number, plus a temporary password that will have to be changed. Checkout operations will require a library card, as will requests to put media on hold. Each library card will have its own fines, but active fines on any of a customer's cards will prevent the customer from using the library's services. The library will have branches in various physical locations. Branches will be identified by name, and each branch will have an address and a phone number associated with it. Additionally, a library branch will store media and have employees. Employees will work at a specific branch of the library. They receive a paycheck, but they can also have library cards; therefore, the same information that is collected about customers should be collected about employees.

Functions for customers:

- Log in

- Search for media based on one or more of the following criteria:

 - type (book, video, or both)

 - title

 - author or director

 - year

- Access their own account information:

 - Card number(s)

 - Fines

 - Media currently checked out

 - Media on hold

- Put media on hold

- Pay fines for lost or late items

- Update personal information:

 - Phone numbers

 - Addresses

 - Passwords

Functions for librarians are the same as the functions for customers plus the following:

- Add customers

- Add library cards and assign them to customers

- Check out media

- Manage and transfer media that is currently on hold

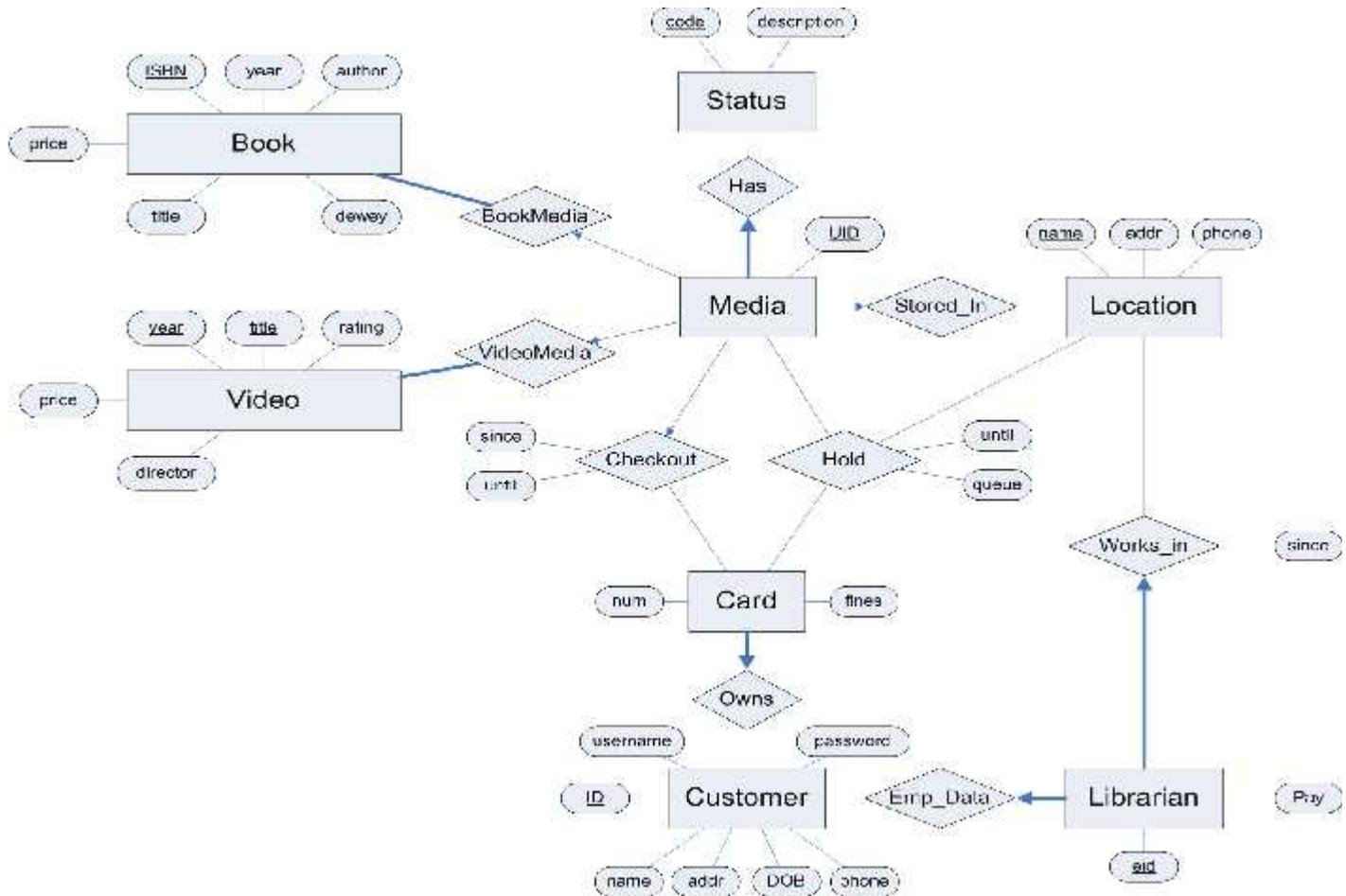
UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

- Handle returns
- Modify customers' fines
- Add media to the database
- Remove media from the database
- Receive payments from customers and update the customers' fines
- View all customer information except passwords

ER DESIGN

It is clear that the physical objects from the previous section – the customers, employees, cards, media, and library branches – correspond to entities in the Entity-Relationship model, and the operations to be done on those entities – holds, checkouts, and so on – correspond to relationships. However, a good design will minimize redundancy and attempt to store all the required information in as small a space as possible. After some consideration, we have decided on the following design:



Notice that the information about books and videos has been separated from the Media entity. This allows the database to store multiple copies of the same item without redundancy. The Status entity has also been separated from Media in order to save space. The Hold relationship stores the entry's place in line (denoted by "queue"); it can be used to create a waiting list of interested customers. The Librarian entity is functionally an extension to Customer, so each Librarian also has a customer associated with it. The librarians will have access to the same

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

features as customers, but they will also perform administrative functions, such as checking media in and out and updating customers' fines.

RELATIONAL DATABASE DESIGN

After coming up with an Entity-Relationship model to describe the library system, we took advantage of the special relationships found in our design, and were able to condense the information to 13 tables. This new design is a database that combines some entities and relationships into common tables.

Table 1: Relational Database Schema

Status	code	description					
Media	media_id	code					
Book	ISBN	title	author	year	dewey	price	
BookMedia	media_id	ISBN					
Customer	ID	name	addr	DOB	phone	username	password
Card	num	fines	ID				
Checkout	media_id	num	since	until			
Location	name	addr	phone				
Hold	media_id	num	name	until	queue		
Stored_In	media_id	name					
Librarian	eid	ID	Pay	name	since		
Video	title	year	director	rating	price		
VideoMedia	media_id	title	year				

Note: the arrows in the diagram represent foreign key constraints.

NORMALIZATION

As stated earlier, the tables in this database are in the Third Normal Form (3 NF.) In order to decompose the relationships into this form, we had to split Status table from the Media table. Each Media object has a status code, and each status code has an associated description.

It would be redundant to store both codes and descriptions in the Media object, so we created a dedicated Status table with the code as the primary key. The other tables were designed with optimization in mind. The Card entity, for instance, was separated from the Customer entity to avoid a functional dependency (since the "num" attribute of the Card entity determines the "fines" attribute.)

PHYSICAL DATABASE DESIGN

The next step was to create the physical database and input some sample data. In order to turn the relational design into a database, we ran the following script in UNCC's Oracle database:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
CREATE TABLE Status ( code INTEGER, description CHAR(30), PRIMARY KEY (code) );
CREATE TABLE Media( media_id INTEGER, code INTEGER, PRIMARY KEY (media_id),
FOREIGN KEY (code) REFERENCES Status );
CREATE TABLE Book(ISBNCHAR(14), title CHAR(128), author CHAR(64),
year INTEGER, dewey INTEGER, price REAL, PRIMARY KEY (ISBN) );
CREATE TABLE BookMedia( media_id INTEGER, ISBN CHAR(14), PRIMARY KEY (media_id),
FOREIGN KEY (media_id) REFERENCES Media,
FOREIGN KEY (ISBN) REFERENCES Book);
CREATE TABLE Customer( ID INTEGER, name CHAR(64), addr CHAR(256), DOB CHAR(10),
phone CHAR(30), username CHAR(16), password CHAR(32), PRIMARY KEY (ID),
UNIQUE (username) );
CREATE TABLE Card( num INTEGER, fines REAL, ID INTEGER, PRIMARY KEY (num),
FOREIGN KEY (ID) REFERENCES Customer );
CREATE TABLE Checkout( media_id INTEGER, num INTEGER, since CHAR(10),
until CHAR(10), PRIMARY KEY (media_id),
FOREIGN KEY (media_id) REFERENCES Media,
FOREIGN KEY (num) REFERENCES Card );
CREATE TABLE Location( name CHAR(64), addr CHAR(256), phone CHAR(30),
PRIMARY KEY (name) );
CREATE TABLE Hold( media_id INTEGER, num INTEGER, name CHAR(64), until CHAR(10),
queue INTEGER, PRIMARY KEY (media_id, num),
FOREIGN KEY (name) REFERENCES Location,
FOREIGN KEY (num) REFERENCES Card,
FOREIGN KEY (media_id) REFERENCES Media );
CREATE TABLE Stored_In( media_id INTEGER, name char(64), PRIMARY KEY (media_id),
FOREIGN KEY (media_id) REFERENCES Media ON DELETE CASCADE,
FOREIGN KEY (name) REFERENCES Location );
CREATE TABLE Librarian( eid INTEGER, ID INTEGER NOT NULL, Pay REAL,
Loc_name CHAR(64) NOT NULL, PRIMARY KEY (eid),
FOREIGN KEY (ID) REFERENCES Customer ON DELETE CASCADE,
FOREIGN KEY (Loc_name) REFERENCES Location(name) );
CREATE TABLE Video( title CHAR(128), year INTEGER, director CHAR(64),
rating REAL, price REAL, PRIMARY KEY (title, year) );
CREATE TABLE VideoMedia( media_id INTEGER, title CHAR(128), year INTEGER,
PRIMARY KEY (media_id), FOREIGN KEY (media_id) REFERENCES Media,
FOREIGN KEY (title, year) REFERENCES Video );
```

The next script populated the database with sample data:

```
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES
(60201, 'Jason L. Gray', '2087 Timberbrook Lane, Gypsum, CO 81637',
'09/09/1958', '970-273-9237', 'jlgrey', 'password1');
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES
(89682, 'Mary L. Prieto', '1465 Marion Drive, Tampa, FL 33602',
'11/20/1961', '813-487-4873', 'mlprieto', 'password2');
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES
(64937, 'Roger Hurst', '974 Bingamon Branch Rd, Bensenville, IL 60106',
'08/22/1973', '847-221-4986', 'rhurst', 'password3');
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES
(31430, 'Warren V. Woodson', '3022 Lords Way, Parsons, TN 38363',
'03/07/1945', '731-845-0077', 'vwwoodson', 'password4');
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES
(79916, 'Steven Jensen', '93 Sunny Glen Ln, Garfield Heights, OH 44125',
'12/14/1968', '216-789-6442', 'sjensen', 'password5');
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES
(93265, 'David Bain', '4356 Pooh Bear Lane, Travelers Rest, SC 29690',
'08/10/1947', '864-610-9558', 'dbain', 'password6');
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES
(58359, 'Ruth P. Alber', '3842 Willow Oaks Lane, Lafayette, LA 70507',
'02/18/1976', '337-316-3161', 'rpalber', 'password7');
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES
(88564, 'Sally J. Schilling', '1894 Wines Lane, Houston, TX 77002',
'07/02/1954', '832-366-9035', 'sjschilling', 'password8');
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES
(57054, 'John M. Byler', '279 Raver Croft Drive, La Follette, TN 37766',
'11/27/1954', '423-592-8630', 'jmbyler', 'password9');
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES
(49312, 'Kevin Spruell', '1124 Broadcast Drive, Beltsville, VA 20705',
'03/04/1984', '703-953-1216', 'kspruell', 'password10');
INSERT INTO Card(num, fines, ID) VALUES ( 5767052, 0.0, 60201);
INSERT INTO Card(num, fines, ID) VALUES ( 5532681, 0.0, 60201);
INSERT INTO Card(num, fines, ID) VALUES ( 2197620, 10.0, 89682);
INSERT INTO Card(num, fines, ID) VALUES ( 9780749, 0.0, 64937);
INSERT INTO Card(num, fines, ID) VALUES ( 1521412, 0.0, 31430);
INSERT INTO Card(num, fines, ID) VALUES ( 3920486, 0.0, 79916);
INSERT INTO Card(num, fines, ID) VALUES ( 2323953, 0.0, 93265);
INSERT INTO Card(num, fines, ID) VALUES ( 4387969, 0.0, 58359);
INSERT INTO Card(num, fines, ID) VALUES ( 4444172, 0.0, 88564);
INSERT INTO Card(num, fines, ID) VALUES ( 2645634, 0.0, 57054);
INSERT INTO Card(num, fines, ID) VALUES ( 3688632, 0.0, 49312);
INSERT INTO Location(name, addr, phone) VALUES ('Texas Branch',
'4832 Deercove Drive, Dallas, TX 75208', '214-948-7102');
INSERT INTO Location(name, addr, phone) VALUES ('Illinois Branch',
'2888 Oak Avenue, Des Plaines, IL 60016', '847-953-8130');
INSERT INTO Location(name, addr, phone) VALUES ('Louisiana Branch',
'2063 Washburn Street, Baton Rouge, LA 70802', '225-346-0068');
INSERT INTO Status(code, description) VALUES (1, 'Available');
INSERT INTO Status(code, description) VALUES (2, 'In Transit');
INSERT INTO Status(code, description) VALUES (3, 'Checked Out');
INSERT INTO Status(code, description) VALUES (4, 'On Hold');
INSERT INTO Media( media_id, code) VALUES (8733, 1);
INSERT INTO Media( media_id, code) VALUES (9982, 1);
INSERT INTO Media( media_id, code) VALUES (3725, 1);
INSERT INTO Media( media_id, code) VALUES (2150, 1);
INSERT INTO Media( media_id, code) VALUES (4188, 1);
INSERT INTO Media( media_id, code) VALUES (5271, 2);
INSERT INTO Media( media_id, code) VALUES (2220, 3);
INSERT INTO Media( media_id, code) VALUES (7757, 1);
INSERT INTO Media( media_id, code) VALUES (4589, 1);
INSERT INTO Media( media_id, code) VALUES (5748, 1);
INSERT INTO Media( media_id, code) VALUES (1734, 1);
INSERT INTO Media( media_id, code) VALUES (5725, 1);
INSERT INTO Media( media_id, code) VALUES (1716, 4);
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
INSERT INTO Media( media_id, code) VALUES (8388, 1);
INSERT INTO Media( media_id, code) VALUES (8714, 1);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
('978-0743289412', 'Lisey"s Story', 'Stephen King',
2006, 813, 10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
('978-1596912366', 'Restless: A Novel', 'William Boyd',
2006, 813, 10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
('978-0312351588', 'Beachglass', 'Wendy Blackburn',
2006, 813, 10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
('978-0156031561', 'The Places In Between', 'Rory Stewart',
2006, 910, 10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
('978-0060583002', 'The Last Season', 'Eric Blehm',
2006, 902, 10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
('978-0316740401', 'Case Histories: A Novel', 'Kate Atkinson',
2006, 813, 10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
('978-0316013949', 'Step on a Crack', 'James Patterson, et al.',
2007, 813, 10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
('978-0374105235', 'Long Way Gone: Memoirs of a Boy Soldier',
'Ishmael Beah', 2007, 916, 10.0);
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
('978-0385340229', 'Sisters', 'Danielle Steel', 2006, 813, 10.0);
INSERT INTO BookMedia(media_id, ISBN) VALUES (8733, '978-0743289412');
INSERT INTO BookMedia(media_id, ISBN) VALUES (9982, '978-1596912366');
INSERT INTO BookMedia(media_id, ISBN) VALUES (3725, '978-1596912366');
INSERT INTO BookMedia(media_id, ISBN) VALUES (2150, '978-0312351588');
INSERT INTO BookMedia(media_id, ISBN) VALUES (4188, '978-0156031561');
INSERT INTO BookMedia(media_id, ISBN) VALUES (5271, '978-0060583002');
INSERT INTO BookMedia(media_id, ISBN) VALUES (2220, '978-0316740401');
INSERT INTO BookMedia(media_id, ISBN) VALUES (7757, '978-0316013949');
INSERT INTO BookMedia(media_id, ISBN) VALUES (4589, '978-0374105235');
INSERT INTO BookMedia(media_id, ISBN) VALUES (5748, '978-0385340229');
INSERT INTO Checkout(media_id, num, since, until) VALUES
(2220, 9780749, '02/15/2007', '03/15/2007');
INSERT INTO Video(title, year, director, rating, price) VALUES
('Terminator 2: Judgment Day', 1991, 'James Cameron', 8.3, 20.0);
INSERT INTO Video(title, year, director, rating, price) VALUES
('Raiders of the Lost Ark', 1981, 'Steven Spielberg', 8.7, 20.0);
INSERT INTO Video(title, year, director, rating, price) VALUES
('Aliens', 1986, 'James Cameron', 8.3, 20.0);
INSERT INTO Video(title, year, director, rating, price) VALUES
('Die Hard', 1988, 'John McTiernan', 8.0, 20.0);
INSERT INTO VideoMedia(media_id, title, year) VALUES
( 1734, 'Terminator 2: Judgment Day', 1991);
INSERT INTO VideoMedia(media_id, title, year) VALUES
( 5725, 'Raiders of the Lost Ark', 1981);
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
INSERT INTO VideoMedia(media_id, title, year) VALUES
( 1716, 'Aliens', 1986);
INSERT INTO VideoMedia(media_id, title, year) VALUES
( 8388, 'Aliens', 1986);
INSERT INTO VideoMedia(media_id, title, year) VALUES
( 8714, 'Die Hard', 1988);
INSERT INTO Hold(media_id, num, name, until, queue) VALUES
(1716, 4444172, 'Texas Branch', '02/20/2008', 1);
INSERT INTO Librarian(eid, ID, pay, Loc_name) Values
(2591051, 88564, 30000.00, 'Texas Branch');
INSERT INTO Librarian(eid, ID, pay, Loc_name) Values
(6190164, 64937, 30000.00, 'Illinois Branch');
INSERT INTO Librarian(eid, ID, pay, Loc_name) Values
(1810386, 58359, 30000.00, 'Louisiana Branch');
INSERT INTO Stored_In(media_id, name) VALUES(8733, 'Texas Branch');
INSERT INTO Stored_In(media_id, name) VALUES(9982, 'Texas Branch');
INSERT INTO Stored_In(media_id, name) VALUES(1716, 'Texas Branch');
INSERT INTO Stored_In(media_id, name) VALUES(1734, 'Texas Branch');
INSERT INTO Stored_In(media_id, name) VALUES(4589, 'Texas Branch');
INSERT INTO Stored_In(media_id, name) VALUES(4188, 'Illinois Branch');
INSERT INTO Stored_In(media_id, name) VALUES(5271, 'Illinois Branch');
INSERT INTO Stored_In(media_id, name) VALUES(3725, 'Illinois Branch');
INSERT INTO Stored_In(media_id, name) VALUES(8388, 'Illinois Branch');
INSERT INTO Stored_In(media_id, name) VALUES(5748, 'Illinois Branch');
INSERT INTO Stored_In(media_id, name) VALUES(2150, 'Louisiana Branch');
INSERT INTO Stored_In(media_id, name) VALUES(8714, 'Louisiana Branch');
INSERT INTO Stored_In(media_id, name) VALUES(7757, 'Louisiana Branch');
INSERT INTO Stored_In(media_id, name) VALUES(5725, 'Louisiana Branch');
```

The database was created and filled with 10 Customers, 11 Cards, 3 Locations and 3 Employees, and 15 media items. A Hold relationship and a Checkout relationship were also created.

GUI DESIGN

The first step in designing the GUI was to choose a means of accessing the database. After evaluating various options, we settled on using the JDBC API. The availability of JavaServer Pages on UNCC's servers was an important factor, as it allowed us to develop our application using a three-tier architecture. By using JDBC we could separate the application logic from the DBMS as well as from clients. In addition to simplifying operations on the database, this architecture makes extending the functionality of our system easier. When adding a new feature or improving an existing one, we will not need to change the database; it will only be necessary to modify the Java portion of the code. Before beginning Java development, however, we needed to define a set of queries that our application would use to communicate with the Oracle database. The queries are presented below. Note that the terms labeled <user input> are to be filled in by the application after it receives input from the user and validates it. Note also that complex procedures that require several steps and modify more than one table – such as operations to check out media or put media on hold – will combine several queries into a single transaction, eliminating the possibility of corrupting the database. Finally, some searches (i.e. searches for Book or Video entries) may have a variable number of search parameters, determined at run-time. For example, users will have the option to search for a book by title only, by author only, by year only, by all three fields, or by any combination of two fields. For simplicity's sake, the search queries listed below contain all possible search parameters, but not their possible combinations.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
/* *\
Functions available to customers
\* */
/* User login and authentication */
SELECT C.ID, C.name, C.addr, C.DOB, C.phone, C.username,
nvl((SELECT 'Librarian'
FROM Librarian L
WHERE L.ID = C.ID), 'Customer') AS role
FROM Customer C
WHERE C.username = <user input> AND C.password = <user input>;
/* Book search for customers */
SELECT B.ISBN, B.title, B.author, B.year,
(SELECT COUNT(*)
FROM BookMedia BM
WHERE BM.ISBN = B.ISBN AND BM.code = 1) AS num_available
FROM Book B
WHERE B.title LIKE '%<user input>%' AND B.author LIKE '%<user input>%' AND
B.year <= <user input> AND B.year >= <user input>;
/* Find all copies of a book (used for placing holds or viewing detailed
information). */
SELECT BM.media_id, S.description,
nvl((SELECT SI.name
FROM Stored_In SI
WHERE SI.media_id = BM.media_id), 'none') AS name
FROM BookMedia BM, Media M, Status S
WHERE BM.ISBN = <user input> AND M.media_id = BM.media_id AND S.code = M.code;
/* Video search for customers */
SELECT V.title, V.year, V.director, V.rating
(SELECT COUNT(*)
FROM VideoMedia VM
WHERE VM.ID = V.ID AND VM.code = 1) AS num_available
FROM Video V
WHERE V.title LIKE '%<user input>%' AND V.year <= <user input> AND V.year <= <user input>
AND V.director LIKE '%<user input>%' AND V.rating >= <user input>;
/* Find all copies of a video (used for placing holds or viewing detailed
information). */
SELECT VM.media_id, S.description,
nvl((SELECT SI.name
FROM Stored_In SI
WHERE SI.media_id = VM.media_id), 'none') AS name
FROM VideoMedia VM, Media M, Status S
WHERE VM.title = <user input> AND VM.year = <user input> AND
M.media_id = VM.media_id AND S.code = M.code;
/* Find the status of a given media item */
SELECT S.description
FROM Status S, Media M
WHERE S.code = M.code AND M.media_id = <user input>;
/* Create a new Hold */
INSERT INTO Hold(media_id, num, name, until, queue) VALUES
(<user input>, <user input>, <user input>, <user input>,
nvl((SELECT MAX(H.queue)
FROM Hold H
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
WHERE H.media_id = <user input>), 0) + 1 );
/* Cancel Hold, Step 1: Remove the entry from hold */
DELETE FROM Hold
WHERE media_id = <user input> AND num = <user input>
/* Cancel Hold, Step 2: Update queue for this item */
UPDATE Hold
SET queue = queue-1
WHERE media_id = <user input> AND queue > <user input>;
/* Functions needed to view information about a customer */
/* View the customer's card(s) */
SELECT CR.num, CR.fines
FROM Card CR
WHERE CR.ID = <user input>;
/* View media checked out on a given card */
SELECT B.title, B.author, B.year, BM.media_id, CO.since, CO.until
FROM Checkout CO, BookMedia BM, Book B
WHERE CO.num = <user input> AND CO.media_id = BM.media_id AND B.ISBN = BM.ISBN
UNION
SELECT V.title, V.director, V.year, VM.media_id, CO.since, CO.until
FROM Checkout CO, VideoMedia VM, Book B
WHERE CO.num = <user input> AND CO.media_id = VM.media_id AND
VM.title = V.title AND VM.year = V.year;
/* View media currently on hold for a given card */
SELECT B.title, B.author, B.year, BM.media_id, H.until, H.queue, SI.name
FROM Hold H, BookMedia BM, Book B, Stored_In SI
WHERE H.num = <user input> AND H.media_id = BM.media_id AND B.ISBN = BM.ISBN
AND SI.media_id = H.media_id
UNION
SELECT V.title, V.director, V.year, VM.media_id, H.until, H.queue, SI.name
FROM Hold H, VideoMedia VM, Book B, Stored_In SI
WHERE H.num = <user input> AND H.media_id = VM.media_id AND
VM.title = V.title AND VM.year = V.year AND SI.media_id = H.media_id;
/* View the total amount of fines the customer has to pay */
SELECT SUM(CR.fines)
FROM Card CR
WHERE CR.ID = <user input>;
/* */
Functions reserved for librarians
\* */
/* Add new customer */
INSERT INTO Customer(ID, name, addr, DOB, phone, username, password) VALUES
(<user input>, <user input>, <user input>, <user input>, <user input>,
<user input>, <user input>);
/* Find a customer */
SELECT C.ID, C.name, C.addr, C.DOB, C.phone, C.username,
nvl((SELECT 'Librarian'
FROM Librarian L
WHERE L.ID = C.ID), 'Customer') AS role
FROM Customer C
WHERE C.username = <user input> AND C.name LIKE '%<user input>%';
/* Add new card and assign it to a customer */
INSERT INTO Card(num, fines, ID) VALUES (<user input>, 0, <user input>);
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
/* Create an entry in Checkout */
INSERT INTO Checkout(media_id, num, since, until) VALUES
(<user input>, <user input>, <user input>, <user input>);
/* Remove the entry for Stored_In */
DELETE FROM Stored_In
WHERE media_id = <user input>;
/* Change the status code of the media */
UPDATE Media
SET code = <user input>
WHERE media_id = <user input>;
/* Remove the entry from Checkout */
DELETE FROM Checkout
WHERE media_id = <user input>;
/* Create the entry in Stored_In */
INSERT INTO Stored_In(media_id, name) VALUES (<user input>, <user input>);
/* Find the next Hold entry for a given media */
SELECT H.num, H.name, H.until
FROM Hold H
WHERE H.queue = 1 AND H.media_id = <user input>;
/* Change the Stored_In entry to the target library branch */
UPDATE Stored_In
SET name = <user input>
WHERE media_id = <user input>;
/* Find the customer that should be notified about book arrival */
SELECT C.name, C.phone, CR.num
FROM Customer C, Card CR, Hold H
WHERE H.queue = 1 AND H.name = <user input> AND H.media_id = <user input> AND
CR.num = H.num AND C.ID = CR.ID;
/* Add a new entry into the Book table */
INSERT INTO Book(ISBN, title, author, year, dewey, price) VALUES
(<user input>, <user input>, <user input>, <user input>, <user input>,
<user input>);
/* Add a new entry into the Video table */
INSERT INTO Video(title, year, director, rating, price) VALUES
(<user input>, <user input>, <user input>, <user input>, <user input>);
/* Add a new Media object */
INSERT INTO Media( media_id, code) VALUES (<user input>, 1);
/* Add a new BookMedia object */
INSERT INTO BookMedia(media_id, ISBN) VALUES (<user input>, <user input>);
/* Add a new VideoMedia object */
INSERT INTO VideoMedia(media_id, title, year) VALUES
(<user input>, <user input>, <user input>);
/* Remove an entry from the BookMedia table */
DELETE FROM BookMedia
WHERE media_id = <user input>;
/* Remove an entry from the VideoMedia table */
DELETE FROM VideoMedia
WHERE media_id = <user input>;
/* Remove an entry from the Media table */
DELETE FROM Media
WHERE media_id = <user input>;
/* Remove an entry from the Book table */
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
DELETE FROM Book
WHERE ISBN = <user input>;
/* Remove an entry from the Video table */
DELETE FROM Video
WHERE title = <user input> AND year = <user input>;
/* Update the customer's fines */
UPDATE Card
SET fines = <user input>
WHERE num = <user input>
```

After learning about the optimizers used by commercial database management systems, we reviewed the above queries for efficiency. They turned out to be simple and efficient enough not to require further optimization. With the query design and optimization finished, we turned our attention to the GUI itself. Our design is laid out in a fairly traditional manner – a navigation bar on the top, a navigation box on the left side of the screen, and a content box on the right. Upon first entering the website, the user is presented with a log-in prompt. When the user attempts to log in, the system compares entered credentials with those stored in the database and presents the user with a menu. The menus change based on the user's role: customers have basic functions to search for media, view their account options, fines, and so on, while librarians get an extended menu with administration-related links. It should be noted that there is no special log in for librarians; instead, the system accesses the database to find out whether the user is a librarian and builds the menu dynamically.

13.4 Inventory management system for your college.

Why use Visual Basic

Visual programming is programming for the user it aims at providing the user with an interface that is intuitive and easy to use in developing such as interface the programmer employee user friendly features such as window menu, buttons and list boxes.

A visual programming environment provides all features that are required to develop graphical user interface as ready to use components. The programmer does not have to write code to create and display commonly required user-friendly features each time around.

When the programmer needs a specific user interface feature such as a button, he selects the appropriate, ready to use component provided by the visual programming environment these component can be moved, resized and renamed as required.

ADVANTAGE OF VISUAL PROGRAMMING

Visual programming enables visual development of graphical user interfaces, easy to use and easy to learn.

One of the principal advantages is that the programmer need not write code to display the required component.

The visual programming environment display a list of available component, the programmer pick up the required component from the list.

The component can be moved resized and even deleted if so required.

There is no restriction in the number of controls that can be placed

Moreover since the programmer is creating the user interface usually he can align, move or size the components as required without having resort to writing code.

PROCESSING

There are many items in a departmental store, which are sold to customer and purchased from supplier. An order is placed by the customer-required details, which are listed below:

Item name

Quantity

Delivery time

The order processing executes, look up the stock of each item is available or not then order fulfilled by the management of departmental store. The system periodically checks the stock of each item if it is found below the reorder level then purchase order placed to the supplier for that item, if the supplier is not able to supply whole order then rest of quantity supplied by the another supplier.

After fulfilled the formalities, bill generated by the system and sent to the customer. Item details maintained by the management this whole process is being done manually. My work area is to automate the above process or to generate a more efficient system

ADVANTAGE OF DATABASE

There is several advantage of storing data in database.

1. All data stored at one location when a database is used, all tables are stored in a single file thus, and we need not deal with separate first button use the single database file. Though all the data is stored in a single file, distinction one main faired because of the use of the tables. Each tables is stored as separate entity in the file.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

2. It is possible to define relationship between tables as will be seen once defined these relationship between tables are also stored in the database.
3. It is possible to define validation at the field as well table level this ensure accuracy of data being stored.
4. We also used query, report, sorting etc.

DISADVANTAGES OF OLD SYSTEM

As we know the manual processing is quite tedious, time consuming, less accurate in comparison to computerized processing. Obviously the present system is not is exception consultant encountering all the above problems.

1. Time consuming.
2. It is very tedious.
3. All information is not placed separately.
4. Lot of paper work.
5. Slow data processing.
6. Not user-friendly environment.
7. It is difficult to found records due file management system.

ADVANTAGES OF NEW SYSTEM

In new computerized system I tried to give these facilities.

1. Manually system changes into computerized system.
2. Friendly user interface.
3. Time saving.
4. Save paper work.
5. Connecting to database so we use different type of queries, data report.
6. Give facility of different type of inquiry.
7. Formatted data.
8. Data's are easily approachable.

REQUIREMENTS OF PROJECT REPORT

Hardware Requirement:

Processor: - Intel Pentium III 833MHz
RAM: - 128 SD-RAM.
Hard Disk: -20 GB or above.
Monitor: - 14" VGA.
Mouse.
Printer: - For print report or Bill.
Floppy Disk Drive: - 1.44MB.

Software Requirement:

Operating system: - Windows 98/2002/NT.
Front End: - Visual Basic 6.0.
(Professional Edition.)
Back end: - MS. Access.
(Some additional feature of VB like, Dtagrind, Data- Report)

TABLES

There are three tables.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

ITEM DETAIL TABLE

It contains information about item like item name, minimum quantity in stock, maximum quantity, and reorder status etc.

A. Item code: - It represents the code to identify an item. It helps to search the item in the stock according to requirement.

B. Item name: - This field shows the name of item.

C. Minimum quantity in stock: - This field helps to know the min-qty in stock.

D. Max quantity: - This field shows max quantity in stock.

E. Reorder status: - This field shows reorder status when quantity goes below to minimum quantity in stock.

Purchase order table

This table contains the information about the purchase order like vender code, order code, supplier name, supplier address, order date, item code, item name, quantity, deliver time etc.

Vender code: - This field determine the code of vender.

Order code :-It determines the code of the order that has been ordered by the customer.

Supplier address: - This field helps to know the address of the supplier.

Order date: - This field shows the date of the order.

Item code: - It determines the code of the item.

Item name: - It contains the name of the item.

Quantity: - It specifies the quantity of the order.

Delivery time: - It shows the time of the deliver.

Selling bill table

This table contains information about order that are given by the customer, customer name, customer's address, unit price, amount and total amount etc.

Customer name: - This field determines the name of the customer.

Customer address: - It determines the address of the customer.

Unit price: - It shows the price per item.

Amount: - it determines the amount per item.

Total Amount: - This field shows the total amount of the item that has been purchase by

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

the customer.



UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

item

ITEM-DETAIL

REGNC. 97/2002 DATE 18/08/2002

ITEM_CODE	ITEM_NAME	QTY_N_STOCK	MAX_QTY	REORDER_QTY
102	ery	25	50	30

Button

main				
<<	>	<	>>	
find	save	delete	update	additem

Start | Project1 - Mems | PROJECT ERP | item | 11:59

form2

SELE-BILL

Regn. 97/2002 date

Customer_name

Customer_add. Bill_no

item_code	item_name	quantity	unit_price	amount
Grand total				<input type="text"/>

Start | Project1 - Microsoft Visual | PROJECT ERP | form2 | 11:59

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

order

PURCHASE-ORDER

REGNC. 97\2002 VENDOR_CODE

SUPPLIER_NAME ORDER_CODE

SUPPLIER_ADD. ORDER_DATE

S.NO	ITEM_CODE	ITEM_NAME	QUANTITY/UNIT	DELIVERY TIME

button

>	<	ADD	CANCEL
DELETE	MAIN	FIND	UPDATE

Start | PROJECT REPORT - Min | Project1 - Microsoft Visual | order | 11:35

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

The screenshot displays the Microsoft Access interface. At the top, the title bar reads "Microsoft Access - [item : Table]". Below it is a menu bar with "File", "Edit", "View", "Insert", "Tools", "Window", and "Help". A toolbar contains various icons for database operations. The main area is divided into two panes:

- Table Design View:** A table with the following fields:

Field Name	Data Type	Description
item_code	Text	
item_name	Text	
qty_in_stock	Text	
max_qty	Text	
reorder_qty	Text	
date	Date/Time	
- Field Properties:** A pane titled "Field Properties" with a "General" tab. It lists various properties for the selected field:

Field Size	50
Format	
Input Mask	
Caption	
Default Value	
Validation Rule	
Validation Text	
Required	No
Allow Zero Length	No
Indexed	Yes (No Duplicates)
Unicode Compression	Yes

At the bottom of the Field Properties pane, a blue text box contains the message: "A field name can be up to 64 characters long, including spaces. Press F1 for help on field names." The status bar at the bottom shows "Design view F6 = Switch panes, F1 = Help" and taskbar icons for "Start", "invent : Database", "item : Table", "order Table", "PROJECT ERP", and a clock showing "13:02".

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

The screenshot displays the Microsoft Access interface. The main window shows a table named 'order : Table' with the following fields:

Field Name	Data Type	Description
vendor_code	Number	
order_code	Number	
date	Date/Time	
supplier_name	Text	
supplier_address	Text	
item_code	Number	
item_name	Text	
order_qty	Number	

The 'Field Properties' window is open, showing the 'General' tab for the selected field. The properties are as follows:

Property	Value
Field Size	Long Integer
Format	
Decimal Places	Auto
Input Mask	
Caption	
Default Value	0
Validation Rule	
Validation Text	
Required	No
Indexed	Yes (Duplicates OK)

A help message is displayed on the right side of the Field Properties window: "A field name can be up to 64 characters long, including spaces. Press F1 for help on field names."

The taskbar at the bottom shows the Start button, several open windows (Inventory Data, Item Table, order : Table, PROJECT R), and the system clock showing 13:34.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

The screenshot shows the Microsoft Access interface with the 'Field Properties' window open for the 'bill' table. The window is divided into two tabs: 'General' and 'Lookup'. The 'General' tab is active, showing various field properties. The 'Description' property is highlighted, with a tooltip explaining its purpose: 'The field description is optional. It helps you describe the field and is also displayed in the status bar when you select this field on a form. Press F1 for help on descriptions.'

Field Name	Data Type	Description
bill_no	Number	
date	Date/Time	
customer_name	Text	
customer_address	Text	
item_code	Number	
item_name	Text	
quantity	Number	
unit_price	Number	

Field Properties

General | Lookup

Field Size: Long Integer
Format:
Decimal Places: Auto
Input Mask:
Caption:
Default Value: 0
Validation Rule:
Validation Text:
Required: No
Indexed: No

The field description is optional. It helps you describe the field and is also displayed in the status bar when you select this field on a form. Press F1 for help on descriptions.

Design View F6 = Switch panes, F1 = Help. NUM

The screenshot shows the Microsoft Access interface with the 'bill' table in Datasheet View. The table contains four records. The status bar at the bottom indicates 'Records: 1 of 3' and 'Datasheet View'. The taskbar shows the current window is 'bill : Table'.

bill_no	date	customer_name	customer_add	item_code	item_name	quan
1	02/03/02	vishru	khovarani	101	breeze	
2	02/03/02	kuldeep	marsover	102	pears	
3	05/05/02	amarddeep	kanta	103	coconut oil	
4				0		

Records: 1 of 3

Datasheet View CAPS NUM

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Microsoft Access - [item : Table]

File Edit View Insert Format Records Tools Window Help

item_code	item_name	qty_in_stock	max_qty	reorder_qty	date
102	ary	25	50	30	02/03/02
103	naamam	55	23	22	02/03/02
105	oj	93	34	0	02/05/02
107	khj	55	79	45	00/04/02
108	niwe	73	89	0	05/02/02
109	we	23	43	56	06/07/02
112	pure	45	32	36	02/03/02
113	e	37	67	0	03/05/02
114	sijk	70	09	00	07/02/02
115	kdbu	87	578	0	03/09/02
117	we	83	00	56	01/02/02
119	COCONUT OIL	50	00		05/05/02
120	DIP	35	59	0	
125ghdsjk	46	50	24	12	12/05/02
177	k.d.	35	57	24	
501	namarr	67	50	0	
*					

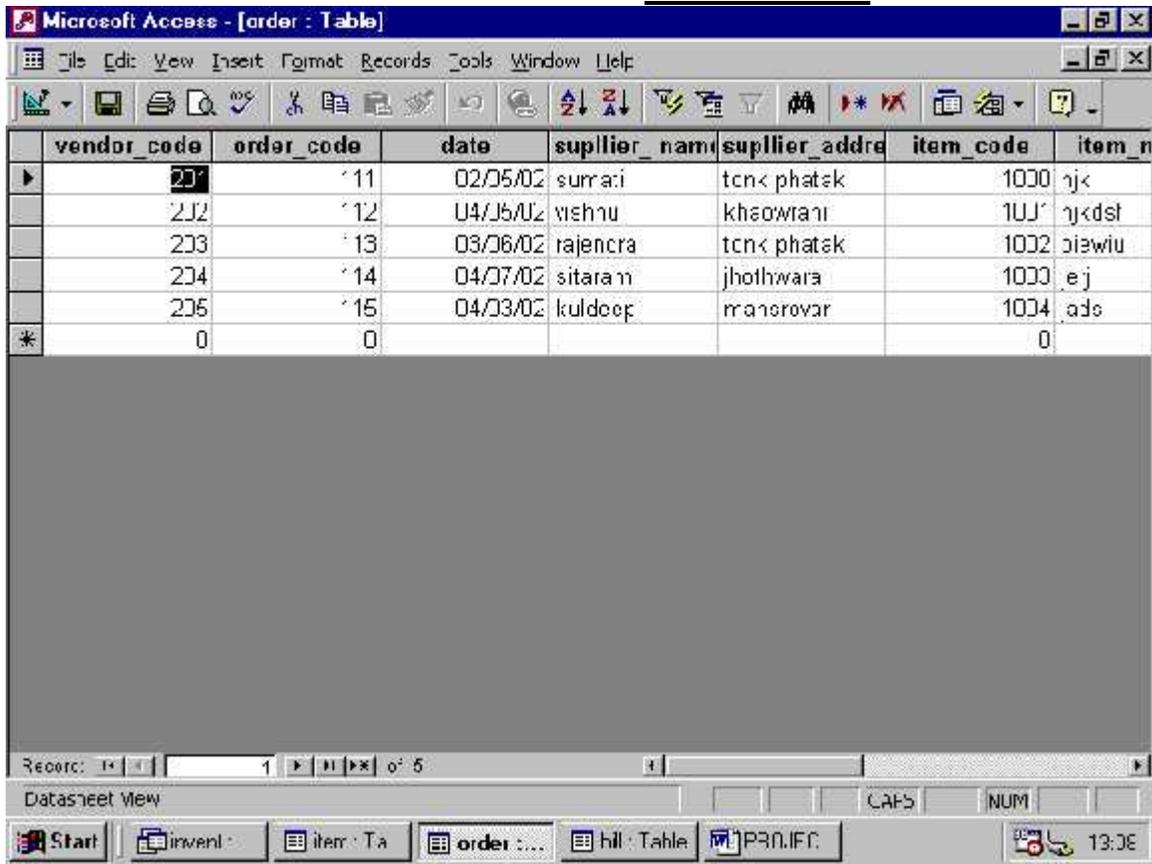
Records: 1 of 16

Datasheet View

Start | Invent | item | order T | hll : Table | PRN.FC | 13:07

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual



vendor_code	order_code	date	supplier_name	supplier_address	item_code	item_name
201	11	02/05/02	sumari	tcn< phatak	1000	gjk
202	12	04/05/02	vehru	khsowrani	1001	gjkdst
203	13	03/06/02	rajendra	tcn< phatak	1002	giewiu
204	14	04/07/02	sitaran	jhothwara	1003	ej
205	15	04/03/02	kuldeep	mansrovar	1004	ads
0	0				0	

14. LAB MANUALS

Lab manual concept is to define how Student design the program on a particular Objective

Here we design a basic concept

Collage Name

Subject : DBMS LAB

Year / Sem :

**Lab Manual
2010-2011**

Branch : CS/IT

Marks : 100

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Taken By

Mr.....

Basic Requirement

1. Hardware Requirement

Hard disk 80 GB

RAM 512 MB

Processor P4 and above

2. Software Requirement

Data base

MS-Access OR MYSQL (Backend)

Language

Visual Basic 6.0 (Front end)

Operating System

Windows XP/ 2003 Server/ Vista/2000

Project

Project Name :

Objective :

Requirement Analysis :

ER Diagram :

Data Flow Diagram :

Database :

Database Management System :

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Normalization :

Front End :

Back End :

Connectivity Steps :

Steps for Creating a Project :

MDI :

Description of the Project :

DFD of Project :

19 Question – Answers

1. What is database?

A database is a logically coherent collection of data with some inherent meaning, representing some aspect of real world and which is designed, built and populated with data for a specific purpose.

2. What is DBMS?

It is a collection of programs that enables user to create and maintain a database. In other words it is general-purpose software that provides the users with the processes of defining, constructing and manipulating the database for various applications.

3. What is a Database system?

The database and DBMS software together is called as Database system.

4. Advantages of DBMS?

-) Redundancy is controlled.
-) Unauthorised access is restricted.
-) Providing multiple user interfaces.
-) Enforcing integrity constraints.
-) Providing backup and recovery.

5. Disadvantage in File Processing System?

-) Data redundancy & inconsistency.
-) Difficult in accessing data.
-) Data isolation.
-) Data integrity.
-) Concurrent access is not possible.
-) Security Problems.

6. Describe the three levels of data abstraction?

The are three levels of abstraction:

-) Physical level: The lowest level of abstraction describes how data are stored.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

-) Logical level: The next higher level of abstraction, describes what data are stored in database and what relationship among those data.
-) View level: The highest level of abstraction describes only part of entire database.

7. Define the "integrity rules"

There are two Integrity rules.

1. Entity Integrity: States that "Primary key cannot have NULL value"
2. Referential Integrity: States that "Foreign Key can be either a NULL value or should be Primary Key value of other relation.

8. What is extension and intension?

Extension - It is the number of tuples present in a table at any instance. This is time dependent.

Intension - It is a constant value that gives the name, structure of table and the constraints laid on it.

9. What is System R? What are its two major subsystems?

System R was designed and developed over a period of 1974-79 at IBM San Jose Research Center. It is a prototype and its purpose was to demonstrate that it is possible to build a Relational System that can be used in a real life environment to solve real life problems, with performance at least comparable to that of existing system.

Its two subsystems are

- Research Storage
- System Relational Data System.

10. How is the data structure of System R different from the relational structure?

Unlike Relational systems in System R

-) Domains are not supported
-) Enforcement of candidate key uniqueness is optional
-) Enforcement of entity integrity is optional
-) Referential integrity is not enforced

11. What is Data Independence?

Data independence means that "the application is independent of the storage structure and access strategy of data". In other words, The ability to modify the schema definition in one level should not affect the schema definition in the next higher level.

Two types of Data Independence:

Physical Data Independence: Modification in physical level should not affect the logical level.

Logical Data Independence: Modification in logical level should affect the view level.

NOTE: Logical Data Independence is more difficult to achieve

12. What is a view? How it is related to data independence?

A view may be thought of as a virtual table, that is, a table that does not really exist in its own right but is instead derived from one or more underlying base table. In other words, there is no stored file that direct represents the view instead a definition of view is stored in data dictionary.

Growth and restructuring of base tables is not reflected in views. Thus the view can insulate users from the effects of restructuring and growth in the database. Hence accounts for logical data independence.

13. What is Data Model?

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

A collection of conceptual tools for describing data, data relationships data semantics and constraints.

- 14. What is E-R model?**
This data model is based on real world that consists of basic objects called entities and of relationship among these objects. Entities are described in a database by a set of attributes.
- 15. What is Object Oriented model?**
This model is based on collection of objects. An object contains values stored in instance variables with in the object. An object also contains bodies of code that operate on the object. These bodies of code are called methods. Objects that contain same types of values and the same methods are grouped together into classes.
- 16. What is an Entity?**
It is a 'thing' in the real world with an independent existence.
- 17. What is an Entity type?**
It is a collection (set) of entities that have same attributes.
- 18. What is an Entity set?**
It is a collection of all entities of particular entity type in the database.
- 19. What is an Extension of entity type?**
The collections of entities of a particular entity type are grouped together into an entity set.
- 20. What is Weak Entity set?**
An entity set may not have sufficient attributes to form a primary key, and its primary key compromises of its partial key and primary key of its parent entity, then it is said to be Weak Entity set.
- 21. What is an attribute?**
It is a particular property, which describes the entity.
- 22. What is a Relation Schema and a Relation?**
A relation Schema denoted by $R(A_1, A_2, \dots, A_n)$ is made up of the relation name R and the list of attributes A_i that it contains. A relation is defined as a set of tuples. Let r be the relation which contains set tuples $(t_1, t_2, t_3, \dots, t_n)$. Each tuple is an ordered list of n -values $t=(v_1, v_2, \dots, v_n)$.
- 23. What is degree of a Relation?**
It is the number of attribute of its relation schema.
- 24. What is Relationship?**
It is an association among two or more entities.
- 25. What is Relationship set?**
The collection (or set) of similar relationships.
- 26. What is Relationship type?**
Relationship type defines a set of associations or a relationship set among a given set of entity types.
- 27. What is degree of Relationship type?**

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

It is the number of entity type participating.

28. **What is DDL (Data Definition Language)?**
A data base schema is specifies by a set of definitions expressed by a special language called DDL.
29. **What is VDL (View Definition Language)?**
It specifies user views and their mappings to the conceptual schema.
30. **What is SDL (Storage Definition Language)?**
This language is to specify the internal schema. This language may specify the mapping between two schemas.
31. **What is Data Storage - Definition Language?**
The storage structures and access methods used by database system are specified by a set of definition in a special type of DDL called data storage-definition language.
32. **What is DML (Data Manipulation Language)?**
This language that enable user to access or manipulate data as organised by appropriate data model.
 -) Procedural DML or Low level: DML requires a user to specify what data are needed and how to get those data.
 -) Non-Procedural DML or High level: DML requires a user to specify what data are needed without specifying how to get those data.
33. **What is DML Compiler?**
It translates DML statements in a query language into low-level instruction that the query evaluation engine can understand.
34. **What is Query evaluation engine?**
It executes low-level instruction generated by compiler.
35. **What is DDL Interpreter?**
It interprets DDL statements and record them in tables containing metadata.
36. **What is Record-at-a-time?**
The Low level or Procedural DML can specify and retrieve each record from a set of records. This retrieve of a record is said to be Record-at-a-time.
37. **What is Set-at-a-time or Set-oriented?**
The High level or Non-procedural DML can specify and retrieve many records in a single DML statement. This retrieve of a record is said to be Set-at-a-time or Set-oriented.
38. **What is Relational Algebra?**
It is procedural query language. It consists of a set of operations that take one or two relations as input and produce a new relation.
39. **What is Relational Calculus?**
It is an applied predicate calculus specifically tailored for relational databases proposed by E.F. Codd. E.g. of languages based on it are DSL ALPHA, QUEL.
40. **How does Tuple-oriented relational calculus differ from domain-oriented relational calculus**
The tuple-oriented calculus uses a tuple variables i.e., variable whose only permitted values

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

are tuples of that relation. E.g. QUEL

The domain-oriented calculus has domain variables i.e., variables that range over the underlying domains instead of over relation. E.g. ILL, DEDUCE.

41. What is normalization?

It is a process of analysing the given relation schemas based on their Functional Dependencies (FDs) and primary key to achieve the properties

-) Minimizing redundancy
-) Minimizing insertion, deletion and update anomalies.

42. What is Functional Dependency?

A Functional dependency is denoted by $X \rightarrow Y$ between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuple that can form a relation state r of R . The constraint is for any two tuples t_1 and t_2 in r if $t_1[X] = t_2[X]$ then they have $t_1[Y] = t_2[Y]$. This means the value of X component of a tuple uniquely determines the value of component Y .

43. When is a functional dependency F said to be minimal?

-) Every dependency in F has a single attribute for its right hand side.
-) We cannot replace any dependency $X \rightarrow A$ in F with a dependency $Y \rightarrow A$ where Y is a proper subset of X and still have a set of dependency that is equivalent to F .
-) We cannot remove any dependency from F and still have set of dependency that is equivalent to F .

44. What is Multivalued dependency?

Multivalued dependency denoted by $X \twoheadrightarrow Y$ specified on relation schema R , where X and Y are both subsets of R , specifies the following constraint on any relation r of R : if two tuples t_1 and t_2 exist in r such that $t_1[X] = t_2[X]$ then t_3 and t_4 should also exist in r with the following properties

$t_3[X] = t_4[X] = t_1[X] = t_2[X]$
 $t_3[Y] = t_1[Y]$ and $t_4[Y] = t_2[Y]$
 $t_3[Z] = t_2[Z]$ and $t_4[Z] = t_1[Z]$
where $Z = (R - (X \cup Y))$

45. What is Lossless join property?

It guarantees that the spurious tuple generation does not occur with respect to relation schemas after decomposition.

46. What is 1 NF (Normal Form)?

The domain of attribute must include only atomic (simple, indivisible) values.

47. What is Fully Functional dependency?

It is based on concept of full functional dependency. A functional dependency $X \rightarrow Y$ is full functional dependency if removal of any attribute A from X means that the dependency does not hold any more.

48. What is 2NF?

A relation schema R is in 2NF if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on primary key.

49. What is 3NF?

A relation schema R is in 3NF if it is in 2NF and for every FD $X \rightarrow A$ either of the following is true

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

-) X is a Super-key of R.
-) A is a prime attribute of R.

In other words, if every non prime attribute is non-transitively dependent on primary key.

50. What is BCNF (Boyce-Codd Normal Form)?

A relation schema R is in BCNF if it is in 3NF and satisfies an additional constraint that for every FD $X \rightarrow A$, X must be a candidate key.

51. What is 4NF?

A relation schema R is said to be in 4NF if for every Multivalued dependency X Y that holds over R, one of following is true

-) X is subset or equal to (or) $XY = R$.
-) X is a super key.

52. What is 5NF?

A Relation schema R is said to be 5NF if for every join dependency $\{R_1, R_2, \dots, R_n\}$ that holds R, one the following is true

-) $R_i = R$ for some i.
-) The join dependency is implied by the set of FD, over R in which the left side is key of R.

53. What is Domain-Key Normal Form?

A relation is said to be in DKNF if all constraints and dependencies that should hold on the the constraint can be enforced by simply enforcing the domain constraint and key constraint on the relation.

54. What are partial, alternate,, artificial, compound and natural key?

Partial Key:

It is a set of attributes that can uniquely identify weak entities and that are related to same owner entity. It is sometime called as Discriminator.

Alternate Key:

All Candidate Keys excluding the Primary Key are known as Alternate Keys.

Artificial Key:

If no obvious key, either stand alone or compound is available, then the last resort is to simply create a key, by assigning a unique number to each record or occurrence. Then this is known as developing an artificial key.

Compound Key:

If no single data element uniquely identifies occurrences within a construct, then combining multiple elements to create a unique identifier for the construct is known as creating a compound key.

Natural Key:

When one of the data elements stored within a construct is utilized as the primary key, then it is called the natural key.

55. What is indexing and what are the different kinds of indexing?

Indexing is a technique for determining how quickly specific data can be found.

Types:

-) Binary search style indexing
-) B-Tree indexing
-) Inverted list indexing
-) Memory resident table
-) Table indexing

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

- 56. What is system catalog or catalog relation? How is better known as?**
A RDBMS maintains a description of all the data that it contains, information about every relation and index that it contains. This information is stored in a collection of relations maintained by the system called metadata. It is also called data dictionary.
- 57. What is meant by query optimization?**
The phase that identifies an efficient execution plan for evaluating a query that has the least estimated cost is referred to as query optimization.
- 58. What is join dependency and inclusion dependency?**
Join Dependency:
A Join dependency is generalization of Multivalued dependency. A JD $\{R_1, R_2, \dots, R_n\}$ is said to hold over a relation R if $R_1, R_2, R_3, \dots, R_n$ is a lossless-join decomposition of R .
There is no set of sound and complete inference rules for JD.
Inclusion Dependency:
An Inclusion Dependency is a statement of the form that some columns of a relation are contained in other columns. A foreign key constraint is an example of inclusion dependency.
- 59. What is durability in DBMS?**
Once the **DBMS** informs the user that a transaction has successfully completed, its effects should persist even if the system crashes before all its changes are reflected on disk. This property is called durability.
- 60. What do you mean by atomicity and aggregation?**
Atomicity:
Either all actions are carried out or none are. Users should not have to worry about the effect of incomplete transactions. **DBMS** ensures this by undoing the actions of incomplete transactions.
Aggregation:
A concept which is used to model a relationship between a collection of entities and relationships. It is used when we need to express a relationship among relationships.
- 61. What is a Phantom Deadlock?**
In distributed deadlock detection, the delay in propagating local information might cause the deadlock detection algorithms to identify deadlocks that do not really exist. Such situations are called phantom deadlocks and they lead to unnecessary aborts.
- 62. What is a checkpoint and When does it occur?**
A Checkpoint is like a snapshot of the **DBMS** state. By taking checkpoints, the **DBMS** can reduce the amount of work to be done during restart in the event of subsequent crashes.
- 63. What are the different phases of transaction?**
Different phases are
) Analysis phase
) Redo Phase
) Undo phase
- 64. What do you mean by flat file database?**
It is a database in which there are no programs or user access languages. It has no cross-file capabilities but is user-friendly and provides user-interface management.
- 65. What is "transparent DBMS"?**

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

It is one, which keeps its Physical Structure hidden from user.

66. Brief theory of Network, Hierarchical schemas and their properties

Network schema uses a graph data structure to organize records example for such a database management system is CTCG while a hierarchical schema uses a tree data structure example for such a system is IMS.

67. What is a query?

A query with respect to **DBMS** relates to user commands that are used to interact with a data base. The query language can be classified into data definition language and data manipulation language.

68. What do you mean by Correlated subquery?

Subqueries, or nested queries, are used to bring back a set of rows to be used by the parent query. Depending on how the subquery is written, it can be executed once for the parent query or it can be executed once for each row returned by the parent query. If the subquery is executed for each row of the parent, this is called a correlated subquery.

A correlated subquery can be easily identified if it contains any references to the parent subquery columns in its WHERE clause. Columns from the subquery cannot be referenced anywhere else in the parent query. The following example demonstrates a non-correlated subquery.

E.g. Select * From CUST Where '10/03/1990' IN (Select ODATE From ORDER Where CUST.CNUM = ORDER.CNUM)

69. What are the primitive operations common to all record management systems?

Addition, deletion and modification.

70. Name the buffer in which all the commands that are typed in are stored

'Edit' Buffer

71. What are the unary operations in Relational Algebra?

PROJECTION and SELECTION.

72. Are the resulting relations of PRODUCT and JOIN operation the same?

No.

PRODUCT: Concatenation of every row in one relation with every row in another.

JOIN: Concatenation of rows from one relation and related rows from another.

73. What is RDBMS KERNEL?

Two important pieces of RDBMS architecture are the kernel, which is the software, and the data dictionary, which consists of the system-level data structures used by the kernel to manage the database

You might think of an RDBMS as an operating system (or set of subsystems), designed specifically for controlling data access; its primary functions are storing, retrieving, and securing data. An RDBMS maintains its own list of authorized users and their associated privileges; manages memory caches and paging; controls locking for concurrent resource usage; dispatches and schedules user requests; and manages space usage within its tablespace structures.

74. Name the sub-systems of a RDBMS

I/O, Security, Language Processing, Process Control, Storage Management, Logging and Recovery, Distribution Control, Transaction Control, Memory Management, Lock Management

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

- 75. Which part of the RDBMS takes care of the data dictionary? How**
Data dictionary is a set of tables and database objects that is stored in a special area of the database and maintained exclusively by the kernel.
- 76. What is the job of the information stored in data-dictionary?**
The information in the data dictionary validates the existence of the objects, provides access to them, and maps the actual physical storage location.
- 77. Not only RDBMS takes care of locating data it also**
determines an optimal access path to store or retrieve the data
76. How do you communicate with an RDBMS?
You communicate with an RDBMS using Structured Query Language (SQL)
- 78. Define SQL and state the differences between SQL and other conventional programming Languages**
SQL is a nonprocedural language that is designed specifically for data access operations on normalized relational database structures. The primary difference between SQL and other conventional programming languages is that SQL statements specify what data operations should be performed rather than how to perform them.
- 79. Name the three major set of files on disk that compose a database in Oracle**
There are three major sets of files on disk that compose a database. All the files are binary. These are
) Database files
) Control files
) Redo logs
The most important of these are the database files where the actual data resides. The control files and the redo logs support the functioning of the architecture itself.
All three sets of files must be present, open, and available to Oracle for any data on the database to be useable. Without these files, you cannot access the database, and the database administrator might have to recover some or all of the database using a backup, if there is one.
- 80. What is an Oracle Instance?**
The Oracle system processes, also known as Oracle background processes, provide functions for the user processes—functions that would otherwise be done by the user processes themselves
Oracle database-wide system memory is known as the SGA, the system global area or shared global area. The data and control structures in the SGA are shareable, and all the Oracle background processes and user processes can use them.
The combination of the SGA and the Oracle background processes is known as an Oracle Instance
- 81. What are the four Oracle system processes that must always be up and running for the database to be useable**
The four Oracle system processes that must always be up and running for the database to be useable include DBWR (Database Writer), LGWR (Log Writer), SMON (System Monitor), and PMON (Process Monitor).
- 82. What are database files, control files and log files. How many of these files should a database have at least? Why?**
Database Files
The database files hold the actual data and are typically the largest in size. Depending on

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

their sizes, the tables (and other objects) for all the user accounts can go in one database file—but that's not an ideal situation because it does not make the database structure very flexible for controlling access to storage for different users, putting the database on different disk drives, or backing up and restoring just part of the database.

You must have at least one database file but usually, more than one files are used. In terms of accessing and using the data in the tables and other objects, the number (or location) of the files is immaterial.

The database files are fixed in size and never grow bigger than the size at which they were created Control Files

The control files and redo logs support the rest of the architecture. Any database must have at least one control file, although you typically have more than one to guard against loss. The control file records the name of the database, the date and time it was created, the location of the database and redo logs, and the synchronization information to ensure that all three sets of files are always in step. Every time you add a new database or redo log file to the database, the information is recorded in the control files.

Redo Logs

Any database must have at least two redo logs. These are the journals for the database; the redo logs record all changes to the user objects or system objects. If any type of failure occurs, the changes recorded in the redo logs can be used to bring the database to a consistent state without losing any committed transactions. In the case of non-data loss failure, Oracle can apply the information in the redo logs automatically without intervention from the DBA. The redo log files are fixed in size and never grow dynamically from the size at which they were created.

83. What is ROWID?

The ROWID is a unique database-wide physical address for every row on every table. Once assigned (when the row is first inserted into the database), it never changes until the row is deleted or the table is dropped.

The ROWID consists of the following three components, the combination of which uniquely identifies the physical storage location of the row.

-) Oracle database file number, which contains the block with the rows
-) Oracle block address, which contains the row
-) The row within the block (because each block can hold many rows)

The ROWID is used internally in indexes as a quick means of retrieving rows with a particular key value. Application developers also use it in SQL statements as a quick way to access a row once they know the ROWID

84. What is Oracle Block? Can two Oracle Blocks have the same address?

Oracle "formats" the database files into a number of Oracle blocks when they are first created—making it easier for the RDBMS software to manage the files and easier to read data into the memory areas.

The block size should be a multiple of the operating system block size. Regardless of the block size, the entire block is not available for holding data; Oracle takes up some space to manage the contents of the block. This block header has a minimum size, but it can grow. These Oracle blocks are the smallest unit of storage. Increasing the Oracle block size can improve performance, but it should be done only when the database is first created.

Each Oracle block is numbered sequentially for each database file starting at 1. Two blocks can have the same block address if they are in different database files.

85. What is database Trigger?

A database trigger is a PL/SQL block that can defined to automatically execute for insert, update, and delete statements against a table. The trigger can e defined to execute once for

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

the entire statement or once for every row that is inserted, updated, or deleted. For any one table, there are twelve events for which you can define database triggers. A database trigger can call database procedures that are also written in PL/SQL.

86. Name two utilities that Oracle provides, which are use for backup and recovery.

Along with the RDBMS software, Oracle provides two utilities that you can use to back up and restore the database. These utilities are Export and Import.

The Export utility dumps the definitions and data for the specified part of the database to an operating system binary file. The Import utility reads the file produced by an export, recreates the definitions of objects, and inserts the data

If Export and Import are used as a means of backing up and recovering the database, all the changes made to the database cannot be recovered since the export was performed. The best you can do is recover the database to the time when the export was last performed.

87. What are stored-procedures? And what are the advantages of using them.

Stored procedures are database objects that perform a user defined operation. A stored procedure can have a set of compound SQL statements. A stored procedure executes the SQL commands and returns the result to the client. Stored procedures are used to reduce network traffic.

88. How are exceptions handled in PL/SQL? Give some of the internal exceptions' name

PL/SQL exception handling is a mechanism for dealing with run-time errors encountered during procedure execution. Use of this mechanism enables execution to continue if the error is not severe enough to cause procedure termination.

The exception handler must be defined within a subprogram specification. Errors cause the program to raise an exception with a transfer of control to the exception-handler block. After the exception handler executes, control returns to the block in which the handler was defined. If there are no more executable statements in the block, control returns to the caller.

User-Defined Exceptions

PL/SQL enables the user to define exception handlers in the declarations area of subprogram specifications. User accomplishes this by naming an exception as in the following example:
ot_failure EXCEPTION;

In this case, the exception name is ot_failure. Code associated with this handler is written in the EXCEPTION specification area as follows:

```
EXCEPTION
```

```
when OT_FAILURE then
```

```
out_status_code := g_out_status_code;
```

```
out_msg := g_out_msg;
```

The following is an example of a subprogram exception:

```
EXCEPTION
```

```
when NO_DATA_FOUND then
```

```
g_out_status_code := 'FAIL';
```

```
RAISE ot_failure;
```

Within this exception is the RAISE statement that transfers control back to the ot_failure exception handler. This technique of raising the exception is used to invoke all user-defined exceptions.

System-Defined Exceptions

Exceptions internal to PL/SQL are raised automatically upon error. NO_DATA_FOUND is a system-defined exception. Table below gives a complete list of internal exceptions.

PL/SQL internal exceptions.

PL/SQL internal exceptions.

Exception Name Oracle Error

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

CURSOR_ALREADY_OPEN ORA-06511
DUP_VAL_ON_INDEX ORA-00001
INVALID_CURSOR ORA-01001
INVALID_NUMBER ORA-01722
LOGIN_DENIED ORA-01017
NO_DATA_FOUND ORA-01403
NOT_LOGGED_ON ORA-01012
PROGRAM_ERROR ORA-06501
STORAGE_ERROR ORA-06500
TIMEOUT_ON_RESOURCE ORA-00051
TOO_MANY_ROWS ORA-01422
TRANSACTION_BACKED_OUT ORA-00061
VALUE_ERROR ORA-06502
ZERO_DIVIDE ORA-01476

In addition to this list of exceptions, there is a catch-all exception named OTHERS that traps all errors for which specific error handling has not been established.

89. Does PL/SQL support "overloading"? Explain

The concept of overloading in PL/SQL relates to the idea that you can define procedures and functions with the same name. PL/SQL does not look only at the referenced name, however, to resolve a procedure or function call. The count and data types of formal parameters are also considered.

PL/SQL also attempts to resolve any procedure or function calls in locally defined packages before looking at globally defined packages or internal functions. To further ensure calling the proper procedure, you can use the dot notation. Prefacing a procedure or function name with the package name fully qualifies any procedure or function reference.

90. Tables derived from the ERD

- a) Are totally unnormalised
 - b) Are always in 1NF
 - c) Can be further denormalised
 - d) May have multi-valued attributes
- (b) Are always in 1NF

91. Spurious tuples may occur due to

- i. Bad normalization
 - ii. Theta joins
 - iii. Updating tables from join
- a) i & ii b) ii & iii
c) i & iii d) ii & iii
(a) i & iii because theta joins are joins made on keys that are not primary keys.

92. A B C is a set of attributes. The functional dependency is as follows

AB -> B
AC -> C
C -> B

- a) is in 1NF
 - b) is in 2NF
 - c) is in 3NF
 - d) is in BCNF
- (a) is in 1NF since $(AC)^+ = \{A, B, C\}$ hence AC is the primary key. Since C B is a FD given, where neither C is a Key nor B is a prime attribute, this it is not in 3NF. Further B is

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

not functionally dependent on key AC thus it is not in 2NF. Thus the given FDs is in 1NF.

93. In mapping of ERD to DFD

- a) entities in ERD should correspond to an existing entity/store in DFD
- b) entity in DFD is converted to attributes of an entity in ERD
- c) relations in ERD has 1 to 1 correspondence to processes in DFD
- d) relationships in ERD has 1 to 1 correspondence to flows in DFD
- (a) entities in ERD should correspond to an existing entity/store in DFD

94. A dominant entity is the entity

- a) on the N side in a 1 : N relationship
- b) on the 1 side in a 1 : N relationship
- c) on either side in a 1 : 1 relationship
- d) nothing to do with 1 : 1 or 1 : N relationship
- (b) on the 1 side in a 1 : N relationship

95. Select 'NORTH', CUSTOMER From CUST_DTLS Where REGION = 'N' Order By CUSTOMER Union Select 'EAST', CUSTOMER From CUST_DTLS Where REGION = 'E' Order By CUSTOMER

The above is

- a) Not an error
- b) Error - the string in single quotes 'NORTH' and 'SOUTH'
- c) Error - the string should be in double quotes
- d) Error - ORDER BY clause
- (d) Error - the ORDER BY clause. Since ORDER BY clause cannot be used in UNIONS

96. What is Storage Manager?

It is a program module that provides the interface between the low-level data stored in database, application programs and queries submitted to the system.

97. What is Buffer Manager?

It is a program module, which is responsible for fetching data from disk storage into main memory and deciding what data to be cache in memory.

98. What is Transaction Manager?

It is a program module, which ensures that database, remains in a consistent state despite system failures and concurrent transaction execution proceeds without conflicting.

99. What is File Manager?

It is a program module, which manages the allocation of space on disk storage and data structure used to represent information stored on a disk.

100. What is Authorization and Integrity manager?

It is the program module, which tests for the satisfaction of integrity constraint and checks the authority of user to access data.

101. What are stand-alone procedures?

Procedures that are not part of a package are known as stand-alone because they independently defined. A good example of a stand-alone procedure is one written in a SQL*Forms application. These types of procedures are not available for reference from other Oracle tools. Another limitation of stand-alone procedures is that they are compiled at run time, which slows execution.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

102. What are cursors give different types of cursors.

PL/SQL uses cursors for all database information accesses statements. The language supports the use two types of cursors

-) Implicit
-) Explicit

103. What is cold backup and hot backup (in case of Oracle)?

) Cold Backup:

It is copying the three sets of files (database files, redo logs, and control file) when the instance is shut down. This is a straight file copy, usually from the disk directly to tape. You must shut down the instance to guarantee a consistent copy.

If a cold backup is performed, the only option available in the event of data file loss is restoring all the files from the latest backup. All work performed on the database since the last backup is lost.

) Hot Backup:

Some sites (such as worldwide airline reservations systems) cannot shut down the database while making a backup copy of the files. The cold backup is not an available option.

So different means of backing up database must be used — the hot backup. Issue a SQL command to indicate to Oracle, on a table space – by – table space basis, that the files of the table space are to be backed up. The users can continue to make full use of the files, including making changes to the data. Once the user has indicated that he/she wants to back up the table space files, he/she can use the operating system to copy those files to the desired backup destination.

The database must be running in ARCHIVELOG mode for the hot backup option.

If a data loss failure does occur, the lost database files can be restored using the hot backup and the online and offline redo logs created since the backup was done. The database is restored to the most consistent state without any loss of committed transactions.

104. What are Armstrong rules? How do we say that they are complete and/or sound

The well-known inference rules for FDs

-) Reflexive rule :
If Y is subset or equal to X then X Y.
-) Augmentation rule:
If X Y then XZ YZ.
-) Transitive rule:
If {X Y, Y Z} then X Z.
-) Decomposition rule :
If X YZ then X Y.
-) Union or Additive rule:
If {X Y, X Z} then X YZ.
-) Pseudo Transitive rule :
If {X Y, WY Z} then WX Z.

Of these the first three are known as Armstrong Rules. They are sound because it is enough if a set of FDs satisfy these three. They are called complete because using these three rules we can generate the rest all inference rules.

105. How can you find the minimal key of relational schema?

Minimal key is one which can identify each tuple of the given relation schema uniquely. For finding the minimal key it is required to find the closure that is the set of all attributes that are dependent on any given set of attributes under the given set of functional dependency.

Algo. I Determining X^+ , closure for X, given set of FDs F

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

1. Set $X^+ = X$
2. Set Old $X^+ = X^+$
3. For each FD $Y Z$ in F and if Y belongs to X^+ then add Z to X^+
4. Repeat steps 2 and 3 until Old $X^+ = X^+$

Algo.II Determining minimal K for relation schema R , given set of FDs F

1. Set K to R that is make K a set of all attributes in R
2. For each attribute A in K
 - a. Compute $(K - A)^+$ with respect to F
 - b. If $(K - A)^+ = R$ then set $K = (K - A)^+$

106. What do you understand by dependency preservation?

Given a relation R and a set of FDs F , dependency preservation states that the closure of the union of the projection of F on each decomposed relation R_i is equal to the closure of F . i.e., $((PR_1(F)) \cup \dots \cup (PR_n(F)))^+ = F^+$
if decomposition is not dependency preserving, then some dependency is lost in the decomposition.

107. What is meant by Proactive, Retroactive and Simultaneous Update.

Proactive Update:

The updates that are applied to database before it becomes effective in real world .

Retroactive Update:

The updates that are applied to database after it becomes effective in real world .

Simultaneous Update:

The updates that are applied to database at the same time when it becomes effective in real world .

108. What are the different types of JOIN operations?

Equi Join: This is the most common type of join which involves only equality comparisons.
The disadvantage in this type of join is that there

15. Examples

```
CREATE TABLESPACE SCT_Admin DATAFILE 'sct_admin.dat' SIZE 10M ONLINE;
```

```
INITIAL EXTENT SIZE 10k
```

```
NEXT EXTENT SIZE 50k MINEXTENTS 1 MAXEXTENTS 999 PCTINCREASE 10
```

```
CREATE TABLESPACE SCT_DATA  
DATAFILE 'SCT_Data.dat'  
SIZE 20M DEFAULT STORAGE(  
INITIAL 10K NEXT 50K  
MINEXTENTS 1  
MAXEXTENTS 999  
PCTINCREASE 10  
)  
ONLINE;
```

```
CREATE TABLESPACE "SCT_DATA"  
LOGGING  
DATAFILE 'D:\ORACLE\ORADATA\ SCT\SCT_DATA.ora'
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
SIZE 20M
AUTOEXTEND ON NEXT 50K
MAXSIZE UNLIMITED
DEFAULT STORAGE(
    INITIAL 10K
    NEXT 50K MINEXTENTS 1
    MAXEXTENTS 999
    PCTINCREASE 10)
```

```
CREATE USER "DBA_SCT"
PROFILE "DEFAULT"
IDENTIFIED BY "<password>"
DEFAULT TABLESPACE "SYSTEM"
TEMPORARY TABLESPACE "TEMP"
ACCOUNT UNLOCK;
```

```
GRANT "CONNECT" TO "DBA_SCT"
WITH ADMIN OPTION;
GRANT "DBA" TO "DBA_SCT"
WITH ADMIN OPTION;
```

*****CREATING TABLESPACE CAR_RENTAL*****

```
CREATE TABLESPACE BANK_SYS
DATAFILE 'Bank_Sys.dat' SIZE 50M
DEFAULT STORAGE(
    INITIAL 10K Next 50K
    MINEXTENTS 1 MAXEXTENTS 999
    PCTINCREASE 10
)
ONLINE;
```

*****CREATING USER DBA_SCT*****

```
CREATE USER "DBA_BANKSYS"
PROFILE "DEFAULT"
IDENTIFIED BY "sct2306"
DEFAULT TABLESPACE "SYSTEM"
TEMPORARY TABLESPACE "TEMP"
ACCOUNT UNLOCK;
```

*****GRANTING PERMISSIONS TO DBA_SCT*****

```
GRANT "DBA" TO "DBA_BANKSYS" WITH ADMIN OPTION;
```

*****CREATING USER SHARANAM*****

```
CREATE USER "SHARANAM"
PROFILE "DEFAULT"
IDENTIFIED BY "SHARANAM"
DEFAULT TABLESPACE "BANK_SYS"
TEMPORARY TABLESPACE "TEMP"
ACCOUNT UNLOCK;
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

*****GRANTING PERMISSIONS TO SHARANAM*****

```
GRANT CREATE TABLE TO "SHARANAM";
GRANT CREATE VIEW TO "SHARANAM";
GRANT INSERT ANY TABLE TO "SHARANAM";
GRANT SELECT ANY TABLE TO "SHARANAM";
GRANT UPDATE ANY TABLE TO "SHARANAM";
GRANT "CONNECT" TO "SHARANAM" WITH ADMIN OPTION;
```

*****CREATING USER HANSEL*****

```
CREATE USER "HANSEL"
  PROFILE "DEFAULT"
  IDENTIFIED BY "HANSEL"
  DEFAULT TABLESPACE "BANK_SYS"
  TEMPORARY TABLESPACE "TEMP"
  ACCOUNT UNLOCK;
```

*****GRANTING PERMISSIONS TO HANSEL*****

```
GRANT CREATE TABLE TO "HANSEL";
GRANT CREATE VIEW TO "HANSEL";
GRANT INSERT ANY TABLE TO "HANSEL";
GRANT SELECT ANY TABLE TO "HANSEL";
GRANT UPDATE ANY TABLE TO "HANSEL";
GRANT "CONNECT" TO "HANSEL" WITH ADMIN OPTION;
```

*****CREATING USER IVAN*****

```
CREATE USER "IVAN"
  PROFILE "DEFAULT"
  IDENTIFIED BY "IVAN"
  DEFAULT TABLESPACE "BANK_SYS"
  TEMPORARY TABLESPACE "TEMP"
  ACCOUNT UNLOCK;
```

*****GRANTING PERMISSIONS TO IVAN*****

```
GRANT CREATE TABLE TO "IVAN";
GRANT CREATE VIEW TO "IVAN";
GRANT INSERT ANY TABLE TO "IVAN";
GRANT SELECT ANY TABLE TO "IVAN";
GRANT UPDATE ANY TABLE TO "IVAN";
GRANT "CONNECT" TO "IVAN" WITH ADMIN OPTION;
```

Creating Tables:

```
CREATE TABLE "DBA_BANKSYS"."BRANCH_MSTR"(
  "BRANCH_NO" VARCHAR2(10),
  "NAME" VARCHAR2(25));
```

Inserting records

```
1) BRANCH_MSTR
INSERT INTO BRANCH_MSTR (BRANCH_NO, NAME) VALUES('B1', 'Vile Parle (HO)');
INSERT INTO BRANCH_MSTR (BRANCH_NO, NAME) VALUES('B2', 'Andheri');
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
INSERT INTO BRANCH_MSTR (BRANCH_NO, NAME) VALUES('B3', 'Churchgate');
INSERT INTO BRANCH_MSTR (BRANCH_NO, NAME) VALUES('B4', 'Sion');
INSERT INTO BRANCH_MSTR (BRANCH_NO, NAME) VALUES('B5', 'Borivali');
INSERT INTO BRANCH_MSTR (BRANCH_NO, NAME) VALUES('B6', 'Matunga');
```

Viewing Record

- 1) Show all employee numbers, first name, middle name and last name who work in the bank.

```
SELECT EMP_NO, FNAME, MNAME, LNAME FROM EMP_MSTR;
```
- 2) Show all the details related to the Fixed Deposit Slab

```
SELECT * FROM FDSLAB_MSTR;
```

Filtering Table Data

- 1) Show the first name along with the last name of the employees of the bank

```
SELECT FNAME, LNAME FROM EMP_MSTR;
```
- 2) Show the records of the branch whose name is Vile Parle (HO)

```
SELECT * FROM BRANCH_MSTR WHERE NAME = 'Vile Parle (HO)';
```
- 3) Show the details of account number and the type of the account whose type is savings bank account

```
SELECT ACCT_NO, TYPE FROM ACCT_MSTR WHERE TYPE = 'SB';
```
- 4) Show different types of occupations of the customer of the bank by eliminating the repeated occupations

```
SELECT DISTINCT OCCUP FROM CUST_MSTR;
```
- 5) Show only the distinct values of the branch details

```
SELECT DISTINCT * FROM BRANCH_MSTR;
```
- 6) Show all the details of the branch according to its name

```
SELECT * FROM BRANCH_MSTR ORDER BY NAME;
```
- 7)

```
SELECT * FROM BRANCH_MSTR ORDER BY NAME DESC;
```

Creating a Table from a Table

- 1) Make a target table named BRANCHES from the source table named BRANCH_MSTR and change the name of the branch to BRANCH_NAME

```
CREATE TABLE BRANCHES
(BRANCH_NO, BRANCH_NAME)
AS SELECT BRANCH_NO, NAME FROM BRANCH_MSTR;
```

Inserting data into a table from another table

- 1) Insert data in the table BRANCHES from the table BRANCH_MSTR

```
INSERT INTO BRANCHES SELECT BRANCH_NO, NAME FROM BRANCH_MSTR;
```
- 2) Insert only those records where the branch name is that of head office

```
INSERT INTO BRANCHES SELECT BRANCH_NO, NAME FROM BRANCH_MSTR
WHERE NAME = 'Vile Parle (HO)';
```

Delete Operation

- 1) Make the BRANCHES table blank

```
DELETE FROM BRANCHES;
```
- 2) Remove only those records whose branch name is Matunga

```
DELETE FROM BRANCHES WHERE BRANCH_NAME = 'Matunga';
```

Updating the contents of a table

- 1) Update the address details by changing its city name to Bombay

```
UPDATE ADDR_DTLS SET City = 'Bombay';
```
- 2) Update the branch details by changing the Vile Parle (HO) to head office

```
UPDATE BRANCHES SET BRANCH_NAME = 'Head Office'
WHERE BRANCH_NAME = 'Vile Parle (HO)';
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Modifying the structure of table

- 1) Enter a new field called City in the table BRANCHES
ALTER TABLE BRANCHES ADD (CITY VARCHAR2(25));
- 2) Drop a column of city in the branches table
ALTER TABLE BRANCHES DROP COLUMN CITY;
- 3) Alter the branches table by modifying its city to hold maximum of 30 characters
ALTER TABLE BRANCHES MODIFY (CITY varchar2(30));

Renaming tables

- 1) Change the name of branches table to branch table
RENAME BRANCHES TO BRANCH;

Truncate Tables

- 1) Truncate the table branch
TRUNCATE TABLE BRANCH;

Destroy table

- 1) Remove the table branch along with its records
DROP TABLE BRANCH;

Examining objects created by a user

- 1) SELECT * FROM TAB;
- 2) Show the details of a table structure of table BRANCH_MSTR
DESCRIBE BRANCH_MSTR;

```
DROP TABLE TMP_FD_AMT;
DROP TABLE TRANS_DTLS;
DROP TABLE TRANS_MSTR;
DROP TABLE CNTC_DTLS;
DROP TABLE ADDR_DTLS;
DROP TABLE ACCT_FD_CUST_DTLS;
DROP TABLE NOMINEE_MSTR;
DROP TABLE FD_DTLS;
DROP TABLE FD_MSTR;
DROP TABLE FDSLAB_MSTR;
DROP TABLE ACCT_MSTR;
DROP TABLE SPRT_DOC;
DROP TABLE CUST_MSTR;
DROP TABLE EMP_MSTR;
DROP TABLE BRANCH_MSTR;
```

```
-- BRANCH_MSTR
CREATE TABLE "DBA_BANKSYS"."BRANCH_MSTR"(
  "BRANCH_NO" VARCHAR2(10),
  "NAME" VARCHAR2(25));
```

```
-- EMP_MSTR
CREATE TABLE "DBA_BANKSYS"."EMP_MSTR"(
  "EMP_NO" VARCHAR2(10),
  "BRANCH_NO" VARCHAR2(10),
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
"FNAME" VARCHAR2(25),  
"MNAME" VARCHAR2(25),  
"LNAME" VARCHAR2(25),  
"DEPT" VARCHAR2(30),  
"DESIG" VARCHAR2(30),  
"MNGR_NO" VARCHAR2(10));
```

-- CUST_MSTR

```
CREATE TABLE "DBA_BANKSYS"."CUST_MSTR"(  
  "CUST_NO" VARCHAR2(10),  
  "FNAME" VARCHAR2(25),  
  "MNAME" VARCHAR2(25),  
  "LNAME" VARCHAR2(25),  
  "DOB_INC" DATE NOT NULL,  
  "OCCUP" VARCHAR2(25),  
  "PHOTOGRAPH" VARCHAR2(25),  
  "SIGNATURE" VARCHAR2(25),  
  "PANCOPY" VARCHAR2(1),  
  "FORM60" VARCHAR2(1));
```

-- SPRT_DOC

```
CREATE TABLE "DBA_BANKSYS"."SPRT_DOC"(  
  "ACCT_CODE" VARCHAR2(4),  
  "TYPE" VARCHAR2(40),  
  "DOCS" VARCHAR2(75));
```

-- ACCT_MSTR

```
CREATE TABLE "DBA_BANKSYS"."ACCT_MSTR"(  
  "ACCT_NO" VARCHAR2(10),  
  "SF_NO" VARCHAR2(10),  
  "LF_NO" VARCHAR2(10),  
  "BRANCH_NO" VARCHAR2(10),  
  "INTRO_CUST_NO" VARCHAR2(10),  
  "INTRO_ACCT_NO" VARCHAR2(10),  
  "INTRO_SIGN" VARCHAR2(1),  
  "TYPE" VARCHAR2(2),  
  "OPR_MODE" VARCHAR2(2),  
  "CUR_ACCT_TYPE" VARCHAR2(4),  
  "TITLE" VARCHAR2(30),  
  "CORP_CUST_NO" VARCHAR2(10),  
  "APLNDT" DATE,  
  "OPNDT" DATE,  
  "VERI_EMP_NO" VARCHAR2(10),  
  "VERI_SIGN" VARCHAR2(1),  
  "MANAGER_SIGN" VARCHAR2(1),  
  "CURBAL" NUMBER(8, 2) DEFAULT 0,  
  "STATUS" VARCHAR2(1) DEFAULT 'A');
```

-- FD_MSTR

```
CREATE TABLE "DBA_BANKSYS"."FD_MSTR"(  
  "FD_SER_NO" VARCHAR2(10),  
  "SF_NO" VARCHAR2(10),
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
"BRANCH_NO" VARCHAR2(10),  
"INTRO_CUST_NO" VARCHAR2(10),  
"INTRO_ACCT_NO" VARCHAR2(10),  
"INTRO_SIGN" VARCHAR2(1),  
"ACCT_NO" VARCHAR2(10),  
"TITLE" VARCHAR2(30),  
"CORP_CUST_NO" VARCHAR2(10),  
"CORP_CNST_TYPE" VARCHAR(4),  
"VERI_EMP_NO" VARCHAR2(10),  
"VERI_SIGN" VARCHAR2(1),  
"MANAGER_SIGN" VARCHAR2(1));
```

-- FDSLAB_MSTR

```
CREATE TABLE "DBA_BANKSYS"."FDSLAB_MSTR"(  
  "FDSLAB_NO" NUMBER(2),  
  "MINPERIOD" NUMBER(5),  
  "MAXPERIOD" NUMBER(5),  
  "INTRATE" NUMBER(5,2));
```

-- FD_DTLS

```
CREATE TABLE "DBA_BANKSYS"."FD_DTLS"(  
  "FD_SER_NO" VARCHAR2(10),  
  "FD_NO" VARCHAR2(10),  
  "TYPE" VARCHAR2(1),  
  "PAYTO_ACCTNO" VARCHAR2(10),  
  "PERIOD" NUMBER(5),  
  "OPNDT" DATE,  
  "DUEDT" DATE,  
  "AMT" NUMBER(8,2),  
  "DUEAMT" NUMBER(8,2),  
  "INTRATE" NUMBER(3),  
  "STATUS" VARCHAR2(1) DEFAULT 'A',  
  "AUTO_RENEWAL" VARCHAR2(1));
```

-- ACCT_FD_CUST_DTLS

```
CREATE TABLE "DBA_BANKSYS"."ACCT_FD_CUST_DTLS"(  
  "ACCT_FD_NO" VARCHAR2(10),  
  "CUST_NO" VARCHAR2(10));
```

-- NOMINEE_MSTR

```
CREATE TABLE "DBA_BANKSYS"."NOMINEE_MSTR"(  
  "NOMINEE_NO" VARCHAR2(10),  
  "ACCT_FD_NO" VARCHAR2(10),  
  "NAME" VARCHAR2(75),  
  "DOB" DATE,  
  "RELATIONSHIP" VARCHAR2(25));
```

-- ADDR_DTLS

```
CREATE TABLE "DBA_BANKSYS"."ADDR_DTLS"(  
  "ADDR_NO" NUMBER(6),  
  "CODE_NO" VARCHAR2(10),  
  "ADDR_TYPE" VARCHAR2(1),
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
"ADDR1" VARCHAR2(50),  
"ADDR2" VARCHAR2(50),  
"CITY" VARCHAR2(25),  
"STATE" VARCHAR2(25),  
"PINCODE" VARCHAR2(6);
```

-- CNTC_DTLS

```
CREATE TABLE "DBA_BANKSYS"."CNTC_DTLS"(  
  "ADDR_NO" NUMBER(6),  
  "CODE_NO" VARCHAR2(10),  
  "CNTC_TYPE" VARCHAR2(1),  
  "CNTC_DATA" VARCHAR2(75));
```

-- TRANS_MSTR

```
CREATE TABLE "DBA_BANKSYS"."TRANS_MSTR"(  
  "TRANS_NO" VARCHAR2(10),  
  "ACCT_NO" VARCHAR2(10),  
  "DT" DATE,  
  "TYPE" VARCHAR2(1),  
  "PARTICULAR" VARCHAR2(30),  
  "DR_CR" VARCHAR2(1),  
  "AMT" NUMBER(8,2),  
  "BALANCE" NUMBER(8,2));
```

-- TRANS_DTLS

```
CREATE TABLE "DBA_BANKSYS"."TRANS_DTLS"(  
  "TRANS_NO" VARCHAR2(10),  
  "INST_NO" NUMBER(6),  
  "INST_DT" DATE,  
  "PAYTO" VARCHAR2(30),  
  "INST_CLR_DT" DATE,  
  "BANK_NAME" VARCHAR2(35),  
  "BRANCH_NAME" VARCHAR2(25),  
  "PAIDFROM" VARCHAR2(10));
```

-- TMP_FD_AMT

```
CREATE TABLE "DBA_BANKSYS"."TMP_FD_AMT"(  
  "FD_AMT" NUMBER(6));
```

-- Records for BRANCH_MSTR

```
INSERT INTO TMP_FD_AMT (FD_AMT) VALUES(5000);  
INSERT INTO TMP_FD_AMT (FD_AMT) VALUES(10000);  
INSERT INTO TMP_FD_AMT (FD_AMT) VALUES(15000);  
INSERT INTO TMP_FD_AMT (FD_AMT) VALUES(20000);  
INSERT INTO TMP_FD_AMT (FD_AMT) VALUES(25000);  
INSERT INTO TMP_FD_AMT (FD_AMT) VALUES(30000);  
INSERT INTO TMP_FD_AMT (FD_AMT) VALUES(4000);  
INSERT INTO TMP_FD_AMT (FD_AMT) VALUES(50000);
```

-- Records for BRANCH_MSTR

```
INSERT INTO BRANCH_MSTR (BRANCH_NO, NAME) VALUES('B1', 'Vile Parle (HO)');  
INSERT INTO BRANCH_MSTR (BRANCH_NO, NAME) VALUES('B2', 'Andheri');
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
INSERT INTO BRANCH_MSTR (BRANCH_NO, NAME) VALUES('B3', 'Churchgate');
INSERT INTO BRANCH_MSTR (BRANCH_NO, NAME) VALUES('B4', 'Mahim');
INSERT INTO BRANCH_MSTR (BRANCH_NO, NAME) VALUES('B5', 'Borivali');
INSERT INTO BRANCH_MSTR (BRANCH_NO, NAME) VALUES('B6', 'Darya Ganj');
```

-- Records for EMP_MSTR

```
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG, MNGR_NO)
VALUES('E1', 'B1', 'Ivan', 'Nelson', 'Bayross', 'Administration', 'Managing Director', NULL);
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG, MNGR_NO)
VALUES('E2', 'B2', 'Amit', null, 'Desai', 'Loans And Financing', 'Finance Manager', NULL);
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG, MNGR_NO)
VALUES('E3', 'B3', 'Maya', 'Mahima', 'Joshi', 'Client Servicing', 'Sales Manager', NULL);
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG, MNGR_NO)
VALUES('E4', 'B1', 'Peter', 'Iyer', 'Joseph', 'Loans And Financing', 'Clerk', 'E2');
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG, MNGR_NO)
VALUES('E5', 'B4', 'Mandhar', 'Dilip', 'Dalvi', 'Marketing', 'Marketing Manager', NULL);
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG, MNGR_NO)
VALUES('E6', 'B6', 'Sonal', 'Abdul', 'Khan', 'Administration', 'Admin. Executive', 'E1');
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG, MNGR_NO)
VALUES('E7', 'B4', 'Anil', 'Ashutosh', 'Kambli', 'Marketing', 'Sales Asst.', 'E5');
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG, MNGR_NO)
VALUES('E8', 'B3', 'Seema', 'P.', 'Apte', 'Client Servicing', 'Clerk', 'E3');
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG, MNGR_NO)
VALUES('E9', 'B2', 'Vikram', 'Vilas', 'Randive', 'Marketing', 'Sales Asst.', 'E5');
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG, MNGR_NO)
VALUES('E10', 'B6', 'Anjali', 'Sameer', 'Pathak', 'Administration', 'HR Manager', 'E1');
```

-- Records for CUST_MSTR

```
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH,
SIGNATURE, PANCOPYY, FORM60)
VALUES('C1', 'Ivan', 'Nelson', 'Bayross', '25-JUN-1952', 'Self Employed',
'D:/ClntPht/C1.gif', 'D:/ClntSgnt/C1.gif', 'Y', 'Y');
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH,
SIGNATURE, PANCOPYY, FORM60)
VALUES('C2', 'Chriselle', 'Ivan', 'Bayross', '29-OCT-1982', 'Service',
'D:/ClntPht/C2.gif', 'D:/ClntSgnt/C2.gif', 'N', 'Y');
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH,
SIGNATURE, PANCOPYY, FORM60)
VALUES('C3', 'Mamta', 'Arvind', 'Muzumdar', '28-AUG-1975', 'Service',
'D:/ClntPht/C3.gif', 'D:/ClntSgnt/C3.gif', 'Y', 'Y');
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH,
SIGNATURE, PANCOPYY, FORM60)
VALUES('C4', 'Chhaya', 'Sudhakar', 'Bankar', '06-OCT-1976', 'Service',
'D:/ClntPht/C4.gif', 'D:/ClntSgnt/C4.gif', 'Y', 'Y');
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH,
SIGNATURE, PANCOPYY, FORM60)
VALUES('C5', 'Ashwini', 'Dilip', 'Joshi', '20-NOV-1978', 'Business',
'D:/ClntPht/C5.gif', 'D:/ClntSgnt/C5.gif', 'Y', 'Y');
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH,
SIGNATURE, PANCOPYY, FORM60)
VALUES('C6', 'Hansel', 'I.', 'Colaco', '01-JAN-1982', 'Service',
'D:/ClntPht/C6.gif', 'D:/ClntSgnt/C6.gif', 'N', 'Y');
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH, SIGNATURE, PANCOPYY, FORM60)
```

```
VALUES('C7', 'Anil', 'Arun', 'Dhone', '12-OCT-1983', 'Self Employed', 'D:/ClntPht/C7.gif', 'D:/ClntSgnt/C7.gif', 'N', 'Y');
```

```
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH, SIGNATURE, PANCOPYY, FORM60)
```

```
VALUES('C8', 'Alex', 'Austin', 'Fernandes', '30-SEP-1962', 'Executive', 'D:/ClntPht/C8.gif', 'D:/ClntSgnt/C8.gif', 'Y', 'Y');
```

```
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH, SIGNATURE, PANCOPYY, FORM60)
```

```
VALUES('C9', 'Ashwini', 'Shankar', 'Apte', '19-APR-1979', 'Service', 'D:/ClntPht/C9.gif', 'D:/ClntSgnt/C9.gif', 'Y', 'Y');
```

```
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH, SIGNATURE, PANCOPYY, FORM60)
```

```
VALUES('C10', 'Namita', 'S.', 'Kanade', '10-JUN-1978', 'Self Employed', 'D:/ClntPht/C10.gif', 'D:/ClntSgnt/C10.gif', 'Y', 'Y');
```

```
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH, SIGNATURE, PANCOPYY, FORM60)
```

```
VALUES('O11', null, null, null, '14-NOV-1997', 'Retail Business', null, null, 'Y', 'N');
```

```
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH, SIGNATURE, PANCOPYY, FORM60)
```

```
VALUES('O12', null, null, null, '23-OCT-1992', 'Information Technology', null, null, 'Y', 'N');
```

```
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH, SIGNATURE, PANCOPYY, FORM60)
```

```
VALUES('O13', null, null, null, '05-FEB-1989', 'Community Welfare', null, null, 'Y', 'N');
```

```
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH, SIGNATURE, PANCOPYY, FORM60)
```

```
VALUES('O14', null, null, null, '24-MAY-1980', 'Retail Business', null, null, 'N', 'Y');
```

```
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH, SIGNATURE, PANCOPYY, FORM60)
```

```
VALUES('O15', null, null, null, '02-APR-2000', 'Retail Business', null, null, 'Y', 'N');
```

```
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH, SIGNATURE, PANCOPYY, FORM60)
```

```
VALUES('O16', null, null, null, '13-JAN-2002', 'Marketing', null, null, 'Y', 'N');
```

-- Records for SPRT_DOC

```
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
```

```
VALUES('0S', 'Individuals / Savings Bank Account', 'Driving Licence / Ration Card / Passport');
```

```
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
```

```
VALUES('0S', 'Individuals / Savings Bank Account', 'Birth Certificate / School Leaving Certificate');
```

```
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
```

```
VALUES('1C', 'Propriety / Sole Trading Concerns', 'Letter From The Propriety');
```

```
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
```

```
VALUES('2C', 'Partnership Concerns', 'Letter From The Partners');
```

```
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
```

```
VALUES('2C', 'Partnership Concerns', 'Partnership Deed / Registration Certificate');
```

```
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
```

```
VALUES('3C', 'Hindu Undivided Family Businesses', 'Letter From The Karta');
```

```
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
```

```
VALUES('3C', 'Hindu Undivided Family Businesses', 'List Of Members');
```

```
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
```

```
VALUES('4C', 'Limited Companies', 'Copy Of Board Of Directors" Resolution For Opening The Account');
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
  VALUES('4C', 'Limited Companies', 'Memorandum and Articles Of Association');
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
  VALUES('4C', 'Limited Companies', 'Certificate Of Incorporation');
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
  VALUES('4C', 'Limited Companies', 'Certificate Of Commencement Of Business / Registration Certificate');
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
  VALUES('5C', 'Trust Accounts', 'Trust Deed');
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
  VALUES('5C', 'Trust Accounts', 'Resolution Of Trustees');
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
  VALUES('5C', 'Trust Accounts', 'List Of Trustees');
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
  VALUES('6C', 'Clubs / Societies', 'Resolution');
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
  VALUES('6C', 'Clubs / Societies', 'Constitution And Bye-laws');
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
  VALUES('6C', 'Clubs / Societies', 'Certificate Of Registration');
INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)
  VALUES('7C', 'Legislative Bodies', 'Letter From The Authority');

-- Records for ACCT_MSTR
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
  CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
  VALUES('SB1', 'SF-0001', 'NOV03-05', 'B1', 'C1', 'SB1', 'Y', 'SB', 'SI', '0S', null, null,
  '05-NOV-2003', '05-NOV-2003', 'E1', 'Y', 'Y', 500, 'A');
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
  CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
  VALUES('CA2', 'SF-0002', 'NOV03-10', 'B2', 'C1', 'SB1', 'Y', 'CA', 'JO', '1C', 'Uttam Stores', 'O11',
  '07-NOV-2003', '10-NOV-2003', 'E1', 'Y', 'Y', 3000, 'A');
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
  CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
  VALUES('SB3', 'SF-0003', 'NOV03-22', 'B3', 'C4', 'SB3', 'Y', 'SB', 'SI', '0S', null, null,
  '20-NOV-2003', '22-NOV-2003', 'E4', 'Y', 'Y', 500, 'A');
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
  CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
  VALUES('CA4', 'SF-0004', 'DEC03-05', 'B5', 'C4', 'SB3', 'Y', 'CA', 'AS', '4C', 'Sun"s Pvt. Ltd.', 'O12',
  '02-DEC-2003', '05-DEC-2003', 'E4', 'Y', 'Y', 12000, 'A');
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
  CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
  VALUES('SB5', 'SF-0005', 'DEC03-15', 'B6', 'C1', 'SB1', 'Y', 'SB', 'JO', '0S', null, null,
  '14-DEC-2003', '15-DEC-2003', 'E1', 'Y', 'Y', 500, 'A');
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
VALUES('SB6', 'SF-0006', 'DEC03-27', 'B4', 'C5', 'SB6', 'Y', 'SB', 'ES', '0S', null, null,
'27-DEC-2003', '27-DEC-2003', 'E4', 'Y', 'Y', 500, 'A');
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
VALUES('CA7', 'SF-0007', 'JAN04-14', 'B1', 'C8', 'CA7', 'Y', 'CA', 'AS', '6C', 'Puru Hsg. Soc', 'O13',
'14-JAN-2004', '14-JAN-2004', 'E4', 'Y', 'Y', 22000, 'A');
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
VALUES('SB8', 'SF-0008', 'JAN04-29', 'B2', 'C9', 'SB8', 'Y', 'SB', 'SI', '0S', null, null,
'27-JAN-2004', '29-JAN-2004', 'E1', 'Y', 'Y', 500, 'A');
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
VALUES('SB9', 'SF-0009', 'FEB04-05', 'B4', 'C10', 'SB9', 'Y', 'SB', 'JO', '0S', null, null,
'05-FEB-2004', '05-FEB-2004', 'E4', 'Y', 'Y', 500, 'A');
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
VALUES('CA10', 'SF-0010', 'FEB04-19', 'B6', 'C10', 'SB9', 'Y', 'CA', 'AS', '3C', 'Ghar Karobar', 'O14',
'19-FEB-2004', '19-FEB-2004', 'E4', 'Y', 'Y', 32000, 'A');

INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
VALUES('SB11', 'SF-0011', 'MAR04-10', 'B1', 'C1', 'SB1', 'Y', 'SB', 'SI', '0S', null, null,
'05-MAR-2004', '10-MAR-2004', 'E1', 'Y', 'Y', 500, 'A');
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
VALUES('CA12', 'SF-0012', 'MAR04-10', 'B2', 'C1', 'SB5', 'Y', 'CA', 'JO', '1C', 'Suresh Stores', 'O15',
'07-MAR-2004', '10-MAR-2004', 'E1', 'Y', 'Y', 5000, 'A');
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
VALUES('SB13', 'SF-0013', 'MAR04-22', 'B3', 'C4', 'SB3', 'Y', 'SB', 'SI', '0S', null, null,
'20-MAR-2004', '22-MAR-2004', 'E4', 'Y', 'Y', 500, 'A');
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
VALUES('CA14', 'SF-0014', 'APR04-05', 'B5', 'C4', 'SB3', 'Y', 'CA', 'AS', '4C', 'Moon's Pvt. Ltd.', 'O16',
'02-APR-2004', '05-APR-2004', 'E4', 'Y', 'Y', 10000, 'A');
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
VALUES('SB15', 'SF-0015', 'APR04-15', 'B6', 'C1', 'SB1', 'Y', 'SB', 'JO', '0S', null, null,
'14-APR-2004', '15-APR-2004', 'E1', 'Y', 'Y', 500, 'A');
```

-- Records for FD_MSTR

```
INSERT INTO FD_MSTR (FD_SER_NO, SF_NO, BRANCH_NO, ACCT_NO, TITLE, CORP_CUST_NO,
CORP_CNST_TYPE, INTRO_CUST_NO,
INTRO_ACCT_NO, INTRO_SIGN, VERI_EMP_NO, VERI_SIGN, MANAGER_SIGN)
VALUES ('FS1', 'SF-1001', 'B2', 'CA2', 'Uttam Stores', 'O11', '1C', null, null, 'N', 'E1', 'Y', 'Y');
INSERT INTO FD_MSTR (FD_SER_NO, SF_NO, BRANCH_NO, ACCT_NO, TITLE, CORP_CUST_NO,
CORP_CNST_TYPE, INTRO_CUST_NO,
INTRO_ACCT_NO, INTRO_SIGN, VERI_EMP_NO, VERI_SIGN, MANAGER_SIGN)
VALUES ('FS2', 'SF-1002', 'B5', 'CA4', 'Sun's Pvt. Ltd.', 'C12', '4C', null, null, 'N', 'E1', 'Y', 'Y');
INSERT INTO FD_MSTR (FD_SER_NO, SF_NO, BRANCH_NO, ACCT_NO, TITLE, CORP_CUST_NO,
CORP_CNST_TYPE, INTRO_CUST_NO,
INTRO_ACCT_NO, INTRO_SIGN, VERI_EMP_NO, VERI_SIGN, MANAGER_SIGN)
VALUES ('FS3', 'SF-1003', 'B1', 'CA7', 'Puru Hsg. Soc', 'O13', '6C', null, null, 'N', 'E4', 'Y', 'Y');
INSERT INTO FD_MSTR (FD_SER_NO, SF_NO, BRANCH_NO, ACCT_NO, TITLE, CORP_CUST_NO,
CORP_CNST_TYPE, INTRO_CUST_NO,
INTRO_ACCT_NO, INTRO_SIGN, VERI_EMP_NO, VERI_SIGN, MANAGER_SIGN)
VALUES ('FS4', 'SF-1004', 'B6', 'CA10', 'Ghar Karobar', 'O14', '3C', null, null, 'N', 'E4', 'Y', 'Y');
INSERT INTO FD_MSTR (FD_SER_NO, SF_NO, BRANCH_NO, ACCT_NO, TITLE, CORP_CUST_NO,
CORP_CNST_TYPE, INTRO_CUST_NO,
INTRO_ACCT_NO, INTRO_SIGN, VERI_EMP_NO, VERI_SIGN, MANAGER_SIGN)
VALUES ('FS5', 'SF-1005', 'B4', null, null, null, '0S', 'C7', 'SB6', 'Y', 'E4', 'Y', 'Y');
```

-- Record for FDSLAB_MSTR

```
INSERT INTO FDSLAB_MSTR (FDSLAB_NO, MINPERIOD, MAXPERIOD, INTRATE) VALUES(1, 1, 30, 5);
INSERT INTO FDSLAB_MSTR (FDSLAB_NO, MINPERIOD, MAXPERIOD, INTRATE) VALUES(2, 31, 92, 5.5);
INSERT INTO FDSLAB_MSTR (FDSLAB_NO, MINPERIOD, MAXPERIOD, INTRATE) VALUES(3, 93, 183, 6);
INSERT INTO FDSLAB_MSTR (FDSLAB_NO, MINPERIOD, MAXPERIOD, INTRATE) VALUES(4, 184, 365, 6.5);
INSERT INTO FDSLAB_MSTR (FDSLAB_NO, MINPERIOD, MAXPERIOD, INTRATE) VALUES(5, 366, 731, 7.5);
INSERT INTO FDSLAB_MSTR (FDSLAB_NO, MINPERIOD, MAXPERIOD, INTRATE) VALUES(6, 732, 1097, 8.5);
INSERT INTO FDSLAB_MSTR (FDSLAB_NO, MINPERIOD, MAXPERIOD, INTRATE) VALUES(7, 1098, 1829, 10);
```

-- Record for FD_DTLS

```
INSERT INTO FD_DTLS (FD_SER_NO, FD_NO, TYPE, PAYTO_ACCTNO, PERIOD, OPNDT, DUEDT, AMT,
DUEAMT,
INTRATE, STATUS, AUTO_RENEWAL)
VALUES('FS1', 'F1', 'S', 'CA2', 365, '02-JAN-2004', '01-JAN-2005', 15000, 16050.00, 6.5, 'A', 'Y');
INSERT INTO FD_DTLS (FD_SER_NO, FD_NO, TYPE, PAYTO_ACCTNO, PERIOD, OPNDT, DUEDT, AMT,
DUEAMT,
INTRATE, STATUS, AUTO_RENEWAL)
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
VALUES('FS1', 'F2', 'S', 'CA2', 365, '02-JAN-2004', '01-JAN-2005', 5000, 5350.00, 6.5, 'A', 'N');
INSERT INTO FD_DTLS (FD_SER_NO, FD_NO, TYPE, PAYTO_ACCTNO, PERIOD, OPNDT, DUEDT, AMT,
DUEAMT,
INTRATE, STATUS, AUTO_RENEWAL)
VALUES('FS2', 'F3', 'S', 'CA4', 366, '25-MAR-2004', '25-MAR-2005', 10000, 10802.19, 7.5, 'A', 'Y');
INSERT INTO FD_DTLS (FD_SER_NO, FD_NO, TYPE, PAYTO_ACCTNO, PERIOD, OPNDT, DUEDT, AMT,
DUEAMT,
INTRATE, STATUS, AUTO_RENEWAL)
VALUES('FS2', 'F4', 'S', 'CA4', 366, '15-APR-2004', '15-APR-2005', 10000, 10802.19, 7.5, 'A', 'Y');
INSERT INTO FD_DTLS (FD_SER_NO, FD_NO, TYPE, PAYTO_ACCTNO, PERIOD, OPNDT, DUEDT, AMT,
DUEAMT,
INTRATE, STATUS, AUTO_RENEWAL)
VALUES('FS3', 'F5', 'S', 'CA7', 183, '24-APR-2004', '24-OCT-2006', 2000, 2060.16, 6, 'A', 'Y');
INSERT INTO FD_DTLS (FD_SER_NO, FD_NO, TYPE, PAYTO_ACCTNO, PERIOD, OPNDT, DUEDT, AMT,
DUEAMT,
INTRATE, STATUS, AUTO_RENEWAL)
VALUES('FS4', 'F6', 'S', 'CA10', 732, '19-MAY-2004', '20-MAY-2006', 5000, 5902.47, 8.5, 'A', 'Y');
INSERT INTO FD_DTLS (FD_SER_NO, FD_NO, TYPE, PAYTO_ACCTNO, PERIOD, OPNDT, DUEDT, AMT,
DUEAMT,
INTRATE, STATUS, AUTO_RENEWAL)
VALUES('FS5', 'F7', 'S', 'SB6', 366, '27-MAY-2004', '27-MAY-2005', 15000, 16203.30, 7.5, 'A', 'N');
```

-- Record for ACCT_FD_CUST_DTLS

```
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB1', 'C1');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA2', 'C2');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA2', 'C3');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB3', 'C4');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA4', 'C4');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA4', 'C5');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB5', 'C1');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB5', 'C4');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB6', 'C5');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB6', 'C7');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA7', 'C6');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA7', 'C8');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB8', 'C9');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB9', 'C3');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB9', 'C10');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA10', 'C10');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA10', 'C9');

INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB11', 'C1');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA12', 'C2');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA12', 'C3');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB13', 'C4');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA14', 'C4');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA14', 'C5');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB15', 'C1');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB15', 'C4');

INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS1', 'C2');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS1', 'C3');
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS2', 'C4');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS2', 'C5');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS2', 'C5');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS3', 'C6');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS3', 'C8');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS4', 'C10');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS4', 'C9');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS5', 'C5');
```

-- Record for NOMINEE_MSTR

```
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N1', 'CA2', 'Joseph Martin Dias', '17-SEP-1984', 'Colleague');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N2', 'CA2', 'Nilesh Sawant', '25-AUG-1987', 'Colleague');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N3', 'SB1', 'Chriselle Ivan Bayross', '25-JUN-1952', 'Daughter');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N4', 'SB3', 'Mamta Arvind Muzumdar', '28-AUG-1975', 'Friend');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N5', 'SB6', 'Preeti Suresh Shah', '12-FEB-1978', 'Friend');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N6', 'SB8', 'Rohit Rajan Sahakarkar', '30-MAY-1985', 'Relative');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N7', 'CA10', 'Namita S. Kanade', '10-JUN-1978', 'Niece');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N8', 'FS1', 'Rohit Rajan Sahakarkar', '30-MAY-1985', 'Relative');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N9', 'FS2', 'Joseph Martin Dias', '17-SEP-1984', 'Colleague');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N10', 'FS2', 'Nilesh Sawant', '25-AUG-1987', 'Colleague');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N11', 'FS3', 'Chriselle Ivan Bayross', '25-JUN-1952', 'Colleague');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N12', 'FS3', 'Mamta Arvind Muzumdar', '28-AUG-1975', 'Friend');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N13', 'FS4', 'Namita S. Kanade', '10-JUN-1978', 'Relative');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N14', 'FS5', 'Pramila P. Pius', '10-OCT-1985', 'Niece');
```

-- Record for ADDR_DTLS

```
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(1, 'B1', 'H', 'A/5, Jay Chambers,', 'Service Road, Vile Parle (East)',,
'Mumbai', 'Maharashtra', '400057');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(2, 'B2', 'B', 'BSES Chambers, 10th floor,',
'Near Rly. Station, Andheri (West)',, 'Mumbai', 'Maharashtra', '400058');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(3, 'B3', 'B', 'Prabhat Complex, No. 5 / 6,', 'Opp. Air India Bldg., Churchgate,',
'Mumbai', 'Maharashtra', '400004');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(4, 'B4', 'B', '23/A, Swarna Bldg., Smt. Rai Marg,',
'Eastern Express Highway, Kurla (East)',, 'Mumbai', 'Maharashtra', '400045');
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(5, 'B5', 'B', 'Vikas Centre, Shop 37, Near National Park,',
      'Western Express Highway, Borivali (East)', 'Mumbai', 'Maharashtra', '400078');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(6, 'B6', 'B', '24/A, Mahima Plaza, First Floor,', 'Darya Ganj,',
      'New Delhi', 'Delhi', '110004');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(7, 'E1', 'N', 'F-12, Diamond Palace, West Avenue,',
      'North Avenue, Santacruz (West)', 'Mumbai', 'Maharashtra', '400056');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(8, 'E2', 'C', 'Desai House, Plot No. 25, P.G. Marg,',
      'Near Malad Rly. Stat., Malad (West)', 'Mumbai', 'Maharashtra', '400078');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(9, 'E3', 'N', 'Room No. 56, 3rd Floor, Swamibhavan,',
      'J. P. Road Junction, Andheri (East)', 'Mumbai', 'Maharashtra', '400059');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(10, 'E4', 'C', '301, Thomas Palace, Opp. Indu Child Care,',
      'Yadnik Nagar, Andheri (West)', 'Mumbai', 'Maharashtra', '400058');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(11, 'E5', 'C', '456/A, Bldg. No. 4, Vahatuk Nagar,',
      'Amboli, Andheri (West)', 'Mumbai', 'Maharashtra', '400058');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(12, 'E6', 'N', '201, Meena Towers, Nr. Sun Gas Agency,',
      'S. V. Rd., Goregoan (West)', 'Mumbai', 'Maharashtra', '400076');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(13, 'E7', 'N', 'Patel Chawl, Rm. No. 15, B. P. Lal Marg,',
      'Mahim (West)', 'Mumbai', 'Maharashtra', '400016');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(14, 'E8', 'C', 'A - 10, Neelam, L. J. Road,', 'Mahim (East)',
      'Mumbai', 'Maharashtra', '400016');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(15, 'E9', 'N', '1/12 Bal Govindas Society, M. B. Raut Rd.',
      'Dadar (East)', 'Mumbai', 'Maharashtra', '400028');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(16, 'E10', 'C', 'Pathak Nagar, Cadal Road,', 'Mahim (West)', 'New Delhi',
      'Delhi', '110016');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(17, 'C1', 'C', 'F-12, Diamond Palace, West Avenue,',
      'North Avenue, Santacruz (West)', 'Mumbai', 'Maharashtra', '400056');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(18, 'C2', 'C', 'F-12, Silver Stream,', 'Santacruz (East)', 'Mumbai',
      'Maharashtra', '400056');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(19, 'C3', 'C', 'Magesh Prasad,', 'Saraswati Baug, Jogeshwari(E)',
      'Mumbai', 'Maharashtra', '400060');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(20, 'C4', 'C', '4, Sampada,', 'Kataria Road, Mahim,', 'Mumbai',
      'Maharashtra', '400016');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(21, 'C5', 'C', '104, Vikram Apts. Bhagat Lane,', 'Shivaji Park, Mahim,',
      'Mumbai', 'Maharashtra', '400016');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
VALUES(22, 'C6', 'C', '12, Radha Kunj, N.C Kelkar Road,', 'Dadar,', 'Mumbai',
'Maharashtra', '400028');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(23, 'C7', 'C', 'A/14, Shanti Society, Mogal Lane,', 'Mahim,', 'Mumbai',
'Maharashtra', '400016');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(24, 'C8', 'C', '5, Vagdevi, Senapati Bapat Rd.', 'Dadar,', 'Mumbai',
'Maharashtra', '400016');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(25, 'C9', 'C', 'A-10 Nutan Vaishali,', 'Shivaji Park, Mahim,', 'Mumbai',
'Maharashtra', '400016');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(26, 'C10', 'C', 'B-10, Makarand Society,', 'Cadad Road, Mahim,', 'Mumbai',
'Maharashtra', '400016');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(27, 'N1', 'C', '307/E, Meena Mansion,', 'R. S. Road, Andheri (West)',
'Mumbai', 'Maharashtra', '400058');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(28, 'N2', 'C', 'Smt. Veenu Chawl, Sawant Colony Rd.',
'Opp. Veer Road, Matunga (West)', 'Mumbai', 'Maharashtra', '400016');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(29, 'N3', 'C', 'F-12, Silver Stream,', 'Santacruz (East)', 'Mumbai',
'Maharashtra', '400056');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(30, 'N4', 'C', 'Magesh Prasad,', 'Saraswati Baug, Jogeshwari(E)',
'Mumbai', 'Maharashtra', '400060');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(31, 'N5', 'C', 'Rita Apartment, Room No. 46, 2nd Floor,',
'J. P. Road, Andheri (East)', 'Mumbai', 'Maharashtra', '400067');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(32, 'N6', 'N', '106/A, Sunrise Apmt., Opp. Vahatuk Nagar,',
'Kevni-Pada, Jogeshwari (West)', 'Mumbai', 'Maharashtra', '400102');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(33, 'N7', 'C', 'Pathak Nagar, Cadad Road,', 'Mahim (West)', 'Mumbai',
'Maharashtra', '400016');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(34, 'O11', 'H', 'Shop No. 4, Simon Streams,',
'V. P. Road, Andheri (West)', 'Mumbai', 'Maharashtra', '400058');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(35, 'O12', 'H', '230-E, Patel Chambers,', 'Service Road, Vile Parle (East)',
'Mumbai', 'Maharashtra', '400057');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(36, 'O13', 'H', 'G-2, Puru Hsg. Society,', 'Senapati Bapat Rd., Dadar,',
'Mumbai', 'Maharashtra', '400016');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(37, 'O14', 'H', 'B-10, Makarand Society,', 'Cadad Road, Mahim,',
'Mumbai', 'Maharashtra', '400016');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(38, 'N8', 'N', '106/A, Sunrise Apmt., Opp. Vahatuk Nagar,',
'Kevni-Pada, Jogeshwari (West)', 'Mumbai', 'Maharashtra', '400102');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(39, 'N9', 'C', '307/E, Meena Mansion,', 'R. S. Road, Andheri (West)',
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
'Mumbai', 'Maharashtra', '400058');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(40, 'N10', 'C', 'Smt. Veenu Chawl, Sawant Colony Rd.,',
      'Opp. Veer Road, Matunga (West)', 'Mumbai', 'Maharashtra', '400016');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(41, 'N11', 'C', 'F-12, Silver Stream,', 'Santacruz (East)', 'Mumbai',
      'Maharashtra', '400056');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(42, 'N12', 'C', 'Magesh Prasad', 'Saraswati Baug, Jogeshwari(E)',
      'Mumbai', 'Maharashtra', '400060');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(43, 'N13', 'C', 'Pathak Nagar, Cadal Road,', 'Mahim (West)', 'Mumbai',
      'Maharashtra', '400016');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(44, 'N14', 'C', '405, Vahatuk Nagar, Kevni-Pada,', 'Jogeshwari (West)',
      'Mumbai', 'Maharashtra', '400102');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(45, 'C6', 'N', '203/A, Prachi Apmt.', 'Andheri (East)', 'Mumbai',
      'Maharashtra', '400058');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(46, 'O15', 'H', 'Shop No. 4, Sai Compound,',
      'Service Road, Vile Parle (East)', 'Mumbai', 'Maharashtra', '400057');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(47, 'O15', 'H', 'G-4, Sagar Chambers,', 'G. P. Road, Andheri (West)',
      'Mumbai', 'Maharashtra', '400058');
```

-- Record for CNTC_DTLS

```
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(1, 'B1', 'O', '26124571');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(1, 'B1', 'F', '26124533');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(1, 'B1', 'E',
'admin_vileparle@bom2.vsnl.in');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(2, 'B2', 'O', '26790014');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(2, 'B2', 'E',
'admin_andheri@bom2.vsnl.in');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(3, 'B3', 'O', '23457855');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(3, 'B3', 'E',
'admin_churchgate@bom2.vsnl.in');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(4, 'B4', 'O', '25545455');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(4, 'B4', 'E',
'admin_sion@bom2.vsnl.in');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(5, 'B5', 'O', '28175454');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(5, 'B5', 'E',
'admin_borivali@bom2.vsnl.in');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(6, 'B6', 'O', '24304545');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(6, 'B6', 'E',
'admin_matunga@bom2.vsnl.in');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(8, 'E2', 'R', '28883779');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(9, 'E3', 'R', '28377634');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(10, 'E4', 'R', '26323560');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(11, 'E5', 'R', '26793231');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(12, 'E6', 'R', '28085654');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(13, 'E7', 'R', '24442342');
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(14, 'E8', 'R', '24365672');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(15, 'E9', 'R', '24327349');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(16, 'E10', 'R', '24302579');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(17, 'C1', 'R', '26405853');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(17, 'C1', 'O', '26134553');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(17, 'C1', 'O', '26134571');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(17, 'C1', 'M', '9820178955');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(18, 'C2', 'R', '26045754');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(18, 'C2', 'O', '26134571');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(19, 'C3', 'R', '28324567');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(19, 'C3', 'O', '26197654');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(20, 'C4', 'R', '24449852');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(20, 'C4', 'O', '28741370');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(21, 'C5', 'R', '24302934');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(21, 'C5', 'O', '22819964');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(22, 'C6', 'R', '24217592');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(23, 'C7', 'R', '24372247');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(24, 'C8', 'O', '26480903');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(25, 'C9', 'R', '24313408');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(25, 'C9', 'M', '9821176651');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(26, 'C10', 'R', '24362680');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(26, 'C10', 'O', '28973355');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(26, 'C10', 'M',
'9820484648');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(27, 'N1', 'R', '26762154');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(28, 'N2', 'R', '24307887');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(29, 'N3', 'R', '260455754');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(30, 'N4', 'R', '28645489');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(31, 'N5', 'R', '30903564');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(32, 'N6', 'R', '26793771');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(33, 'N7', 'R', '24304455');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(34, 'O11', 'O', '26790055');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(34, 'O11', 'F', '26784409');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(35, 'O12', 'O', '26120455');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(35, 'O12', 'O', '26120456');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(35, 'O12', 'F', '26121450');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(35, 'O12', 'E',
'admin@sunpvtltd.com');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(35, 'O12', 'W',
'www.sunpvtltd.com');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(36, 'O13', 'O', '24301090');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(36, 'O13', 'O', '24301196');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(37, 'O14', 'O', '24321122');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(38, 'N8', 'R', '26793771');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(39, 'N9', 'R', '26762154');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(40, 'N10', 'R', '24307887');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(41, 'N11', 'R', '26045754');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(42, 'N12', 'R', '28645489');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(43, 'N13', 'R', '24304455');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(44, 'N14', 'R', '26790180');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(44, 'N14', 'R', '26771275');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(45, 'C6', 'R', '28274784');
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(46, 'O15', 'O', '26170055');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(46, 'O15', 'F', '26174409');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(47, 'O16', 'O', '26790455');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(47, 'O16', 'F', '26781450');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(47, 'O16', 'E',
'admin@moonmltg.com');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(47, 'O16', 'W',
'www.moonmltg.com');
```

-- Record for TRANS_MSTR

```
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T1', 'SB1', '05-NOV-2003', 'C', 'Initial Payment', 'D', 500, 500);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T2', 'CA2', '10-NOV-2003', 'C', 'Initial Payment', 'D', 2000, 2000);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T3', 'CA2', '13-NOV-2003', 'C', 'Self', 'D', 3000, 5000);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T4', 'SB3', '22-NOV-2003', 'C', 'Initial Payment', 'D', 500, 500);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T5', 'CA2', '10-DEC-2003', 'C', 'Self', 'W', 2000, 3000);
```

```
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T6', 'CA4', '05-DEC-2003', 'B', 'Initial Payment', 'D', 2000, 2000);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T7', 'SB5', '15-DEC-2003', 'B', 'Initial Payment', 'D', 500, 500);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T8', 'SB6', '27-DEC-2003', 'C', 'Initial Payment', 'D', 500, 500);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T9', 'CA7', '14-JAN-2004', 'B', 'Initial Payment', 'D', 2000, 2000);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T10', 'SB8', '29-JAN-2004', 'C', 'Initial Payment', 'D', 500, 500);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T11', 'SB9', '05-FEB-2004', 'C', 'Initial Payment', 'D', 500, 500);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T12', 'SB9', '15-FEB-2004', 'B', 'CLR-204907', 'D', 3000, 3500);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T13', 'SB9', '17-FEB-2004', 'C', 'Self', 'W', 2500, 1000);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T14', 'CA10', '19-FEB-2004', 'B', 'Initial Payment', 'D', 2000, 2000);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T15', 'SB9', '05-APR-2004', 'B', 'CLR-204908', 'D', 3000, 4000);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T16', 'SB9', '27-APR-2004', 'C', 'Self', 'W', 2500, 1500);
```

```
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T17', 'SB1', '05-NOV-2003', 'C', 'Initial Payment', 'D', 500, 500);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T18', 'CA2', '10-NOV-2003', 'C', 'Initial Payment', 'D', 2000, 2000);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T19', 'SB3', '22-NOV-2003', 'C', 'Initial Payment', 'D', 500, 500);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T20', 'CA4', '05-DEC-2003', 'B', 'Initial Payment', 'D', 2000, 2000);
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T21', 'SB5', '15-DEC-2003', 'B', 'Initial Payment', 'D', 500, 500);
```

-- Record for TRANS_DTLS

```
INSERT INTO TRANS_DTLS (TRANS_NO, INST_NO, INST_DT, PAYTO, INST_CLR_DT, BANK_NAME,
BRANCH_NAME, PAIDFROM)
```

```
VALUES('T6', 098324, '02-DEC-2003', 'Self', '05-DEC-2003', 'HDFC', 'Vile Parle (East)', '2982');
```

```
INSERT INTO TRANS_DTLS (TRANS_NO, INST_NO, INST_DT, PAYTO, INST_CLR_DT, BANK_NAME,
BRANCH_NAME, PAIDFROM)
```

```
VALUES('T7', 232324, '14-DEC-2003', 'Self', '15-DEC-2003', 'India Bank', 'Andheri (West)', '30434');
```

```
INSERT INTO TRANS_DTLS (TRANS_NO, INST_NO, INST_DT, PAYTO, INST_CLR_DT, BANK_NAME,
BRANCH_NAME, PAIDFROM)
```

```
VALUES('T9', 434560, '14-JAN-2004', 'Self', '14-JAN-2004', 'ICICI Bank', 'Bandra (West)', '4882');
```

```
INSERT INTO TRANS_DTLS (TRANS_NO, INST_NO, INST_DT, PAYTO, INST_CLR_DT, BANK_NAME,
BRANCH_NAME, PAIDFROM)
```

```
VALUES('T12', 204907, '14-FEB-2004', 'Self', '15-FEB-2004', 'Memon Co-operative Bank', 'Jogeshwari (West)',
'1767');
```

```
INSERT INTO TRANS_DTLS (TRANS_NO, INST_NO, INST_DT, PAYTO, INST_CLR_DT, BANK_NAME,
BRANCH_NAME, PAIDFROM)
```

```
VALUES('T14', 100907, '19-FEB-2004', 'Self', '19-FEB-2004', 'Memon Co-operative Bank', 'Jogeshwari (West)',
'2001');
```

```
INSERT INTO TRANS_DTLS (TRANS_NO, INST_NO, INST_DT, PAYTO, INST_CLR_DT, BANK_NAME,
BRANCH_NAME, PAIDFROM)
```

```
VALUES('T15', 204908, '01-APR-2004', 'Self', '05-APR-2004', 'Memon Co-operative Bank', 'Jogeshwari (West)',
'1767');
```

```
INSERT INTO TRANS_DTLS (TRANS_NO, INST_NO, INST_DT, PAYTO, INST_CLR_DT, BANK_NAME,
BRANCH_NAME, PAIDFROM)
```

```
VALUES('T20', 098324, '02-DEC-2003', 'Self', '05-DEC-2003', 'HDFC', 'Vile Parle (East)', '2982');
```

```
INSERT INTO TRANS_DTLS (TRANS_NO, INST_NO, INST_DT, PAYTO, INST_CLR_DT, BANK_NAME,
BRANCH_NAME, PAIDFROM)
```

```
VALUES('T21', 232324, '14-DEC-2003', 'Self', '15-DEC-2003', 'India Bank', 'Andheri (West)', '30434');
```

```
COMMIT;
```

Primary Key

Sample 1

Drop the CUST_MSTR table, if it already exists. Create a table CUST_MSTR such that the contents of the column CUST_NO is a primary key i.e. it is not null.

```
DROP TABLE CUST_MSTR;
```

```
CREATE TABLE CUST_MSTR (
```

```
"CUST_NO" VARCHAR2(10) PRIMARY KEY,
```

```
"FNAME" VARCHAR2(25),
```

```
"MNAME" VARCHAR2(25),
```

```
"LNAME" VARCHAR2(25),
```

```
"DOB_INC" DATE,
```

```
"OCCUP" VARCHAR2(25),
```

```
"PHOTOGRAPH" VARCHAR2(25),
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
"SIGNATURE" VARCHAR2(25),
"PANCOPY" VARCHAR2(1),
"FORM60" VARCHAR2(1));
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH,
SIGNATURE, PANCOPY, FORM60)
VALUES('C1', 'Ivan', 'Nelson', 'Bayross', '25-JUN-1952', 'Self Employed',
'D:/ClntPht/C1.gif', 'D:/ClntSgnt/C1.gif', 'Y', 'Y');
```

Sample 2

Drop the FD_MSTR table, if it already exists. Create a table FD_MSTR where there is a composite primary key mapped to the columns FD_SER_NO and CORP_CUST_NO.

Since this constraint spans across columns, it must be described at table level.

```
DROP TABLE FD_MSTR;
```

```
CREATE TABLE "DBA_BANKSYS"."FD_MSTR"(  
  "FD_SER_NO" VARCHAR2(10),  
  "SF_NO" VARCHAR2(10),  
  "BRANCH_NO" VARCHAR2(10),  
  "INTRO_CUST_NO" VARCHAR2(10),  
  "INTRO_ACCT_NO" VARCHAR2(10),  
  "INTRO_SIGN" VARCHAR2(1),  
  "ACCT_NO" VARCHAR2(10),  
  "TITLE" VARCHAR2(30),  
  "CORP_CUST_NO" VARCHAR2(10),  
  "CORP_CNST_TYPE" VARCHAR(4),  
  "VERI_EMP_NO" VARCHAR2(10),  
  "VERI_SIGN" VARCHAR2(1),  
  "MANAGER_SIGN" VARCHAR2(1),  
  PRIMARY KEY(FD_SER_NO, CORP_CUST_NO));
```

```
INSERT INTO FD_MSTR (FD_SER_NO, SF_NO, BRANCH_NO, ACCT_NO, TITLE, CORP_CUST_NO,  
CORP_CNST_TYPE, INTRO_CUST_NO, INTRO_ACCT_NO, INTRO_SIGN, VERI_EMP_NO, VERI_SIGN,  
MANAGER_SIGN)
```

```
VALUES ('FS1', 'SF-0011', 'B1', 'CA2', 'Uttam Stores', 'O11', '1C', null, null, 'N', 'E1', 'Y', 'Y');
```

```
INSERT INTO FD_MSTR (FD_SER_NO, SF_NO, BRANCH_NO, ACCT_NO, TITLE, CORP_CUST_NO,  
CORP_CNST_TYPE, INTRO_CUST_NO, INTRO_ACCT_NO, INTRO_SIGN, VERI_EMP_NO, VERI_SIGN,  
MANAGER_SIGN)
```

```
VALUES ('FS2', 'SF-0012', 'B1', 'CA4', 'Sun"s Pvt. Ltd.', 'C12', '4C', null, null, 'N', 'E1', 'Y', 'Y');
```

Foreign Key

Sample 1

Drop the table EMP_MSTR, if it already exists. Create a table EMP_MSTR with its primary as EMP_NO. the foreign key is BRANCH_NO in the BRANCH_MSTR table

```
DROP TABLE EMP_MSTR;
```

```
CREATE TABLE "DBA_BANKSYS"."EMP_MSTR"(  
  "EMP_NO" VARCHAR2(10) PRIMARY KEY,  
  "BRANCH_NO" VARCHAR2(10) REFERENCES BRANCH_MSTR,  
  "FNAME" VARCHAR2(25),  
  "MNAME" VARCHAR2(25),  
  "LNAME" VARCHAR2(25),  
  "DEPT" VARCHAR2(30),  
  "DESIG" VARCHAR2(30));
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Sample 2

Drop the table ACCT_FD_CUST_DTLS, if it already exists. Create a table ACCT_FD_CSUT_DTLS with CUST_NO as foreign key referencing column CUST_NO in the CUST_MSTR table

```
DROP TABLE ACCT_FD_CUST_DTLS;  
CREATE TABLE "DBA_BANKSYS"."ACCT_FD_CUST_DTLS"(  
    "ACCT_FD_NO" VARCHAR2(10),  
    "CUST_NO" VARCHAR2(10),  
    FOREIGN KEY (CUST_NO) REFERENCES CUST_MSTR(CUST_NO));
```

Assigning User Defined Names To Constraints

Sample 1

Drop the CUST_MSTR table, if it already exists. Create a table CUST_MSTR with a primary key constraint on the column CUST_NO and also define its constraint name.

```
DROP TABLE CUST_MSTR;  
CREATE TABLE CUST_MSTR (  
    "CUST_NO" VARCHAR2(10) CONSTRAINT p_CUSTKey PRIMARY KEY,  
    "FNAME" VARCHAR2(25),  
    "MNAME" VARCHAR2(25),  
    "LNAME" VARCHAR2(25),  
    "DOB_INC" DATE,  
    "OCCUP" VARCHAR2(25),  
    "PHOTOGRAPH" VARCHAR2(25),  
    "SIGNATURE" VARCHAR2(25),  
    "PANCOPY" VARCHAR2(1),  
    "FORM60" VARCHAR2(1));
```

Sample 2

Drop the table EMP_MSTR, if it already exists. Create a table EMP_MSTR with its foreign key as BRANCH_NO. The foreign key is BRANCH_NO in the BRANCH_MSTR table and also define the name of the foreign key

```
DROP TABLE EMP_MSTR;  
CREATE TABLE "DBA_BANKSYS"."EMP_MSTR"(  
    "EMP_NO" VARCHAR2(10),  
    "BRANCH_NO" VARCHAR2(10),  
    "FNAME" VARCHAR2(25),  
    "MNAME" VARCHAR2(25),  
    "LNAME" VARCHAR2(25),  
    "DEPT" VARCHAR2(30),  
    "DESIG" VARCHAR2(30),  
    CONSTRAINT f_BranchKey  
    FOREIGN KEY (BRANCH_NO) REFERENCES BRANCH_MSTR);
```

Candidate Key

Sample 1

Drop the FD_MSTR table, if it already exists. Create a table FD_MSTR where there is a candidate primary key mapped to the columns FD_SER_NO and CORP_CUST_NO.

Since this constraint spans across columns, it must be described at table level.

```
DROP TABLE FD_MSTR;  
CREATE TABLE "DBA_BANKSYS"."FD_MSTR"(  
    "FD_SER_NO" VARCHAR2(10),  
    "SF_NO" VARCHAR2(10),
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
"BRANCH_NO" VARCHAR2(10),
"INTRO_CUST_NO" VARCHAR2(10),
"INTRO_ACCT_NO" VARCHAR2(10),
"INTRO_SIGN" VARCHAR2(1),
"ACCT_NO" VARCHAR2(10),
"TITLE" VARCHAR2(30),
"CORP_CUST_NO" VARCHAR2(10),
"CORP_CNST_TYPE" VARCHAR(4),
"VERI_EMP_NO" VARCHAR2(10),
"VERI_SIGN" VARCHAR2(1),
"MANAGER_SIGN" VARCHAR2(1),
PRIMARY KEY(FD_SER_NO, CORP_CUST_NO));
INSERT INTO FD_MSTR (FD_SER_NO, SF_NO, BRANCH_NO, ACCT_NO, TITLE, CORP_CUST_NO,
CORP_CNST_TYPE, INTRO_CUST_NO, INTRO_ACCT_NO, INTRO_SIGN, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN)
VALUES ('FS1', 'SF-0011', 'B1', 'CA2', 'Uttam Stores', 'O11', '1C', null, null, 'N', 'E1', 'Y', 'Y');
INSERT INTO FD_MSTR (FD_SER_NO, SF_NO, BRANCH_NO, ACCT_NO, TITLE, CORP_CUST_NO,
CORP_CNST_TYPE, INTRO_CUST_NO, INTRO_ACCT_NO, INTRO_SIGN, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN)
VALUES ('FS2', 'SF-0012', 'B1', 'CA4', 'Sun's Pvt. Ltd.', 'C12', '4C', null, null, 'N', 'E1', 'Y', 'Y');
```

Unique Key

Sample 1

Drop the CUST_MSTR table, if it already exists. Create a table CUST_MSTR such that the contents of the column CUST_NO are unique across the entire column.

```
DROP TABLE CUST_MSTR;
```

```
CREATE TABLE CUST_MSTR (
"CURT_NO" VARCHAR2(10) UNIQUE,
"FNAME" VARCHAR2(25),
"MNAME" VARCHAR2(25),
"LNAME" VARCHAR2(25),
"DOB_INC" DATE,
"OCCUP" VARCHAR2(25),
"PHOTOGRAPH" VARCHAR2(25),
"SIGNATURE" VARCHAR2(25),
"PANCOPY" VARCHAR2(1),
"FORM60" VARCHAR2(1));
```

```
INSERT INTO CUST_MSTR (CURT_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH,
SIGNATURE, PANCOPY, FORM60)
```

```
VALUES('C1', 'Ivan', 'Nelson', 'Bayross', '25-JUN-1952', 'Self Employed',
'D:/ClntPht/C1.gif', 'D:/ClntSgnt/C1.gif', 'Y', 'Y');
```

```
INSERT INTO CUST_MSTR (CURT_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH,
SIGNATURE, PANCOPY, FORM60)
```

```
VALUES('C1', 'Chriselle', 'Ivan', 'Bayross', '29-OCT-1982', 'Service',
'D:/ClntPht/C2.gif', 'D:/ClntSgnt/C2.gif', 'N', 'Y');
```

```
INSERT INTO CUST_MSTR (CURT_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH,
SIGNATURE, PANCOPY, FORM60)
```

```
VALUES('C2', 'Mamta', 'Arvind', 'Muzumdar', '28-AUG-1975', 'Service',
'D:/ClntPht/C3.gif', 'D:/ClntSgnt/C3.gif', 'Y', 'Y');
```

Sample 2

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Drop the CUST_MSTR table, if it already exists. Create a table CUST_MSTR such that the contents of the column CUST_NO are unique across the entire column.

```
DROP TABLE CUST_MSTR;
```

```
CREATE TABLE CUST_MSTR (  
    "CUST_NO" VARCHAR2(10),  
    "FNAME" VARCHAR2(25),  
    "MNAME" VARCHAR2(25),  
    "LNAME" VARCHAR2(25),  
    "DOB_INC" DATE,  
    "OCCUP" VARCHAR2(25),  
    "PHOTOGRAPH" VARCHAR2(25),  
    "SIGNATURE" VARCHAR2(25),  
    "PANCOPY" VARCHAR2(1),  
    "FORM60" VARCHAR2(1),
```

```
    UNIQUE(CUST_NO));
```

```
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH,  
SIGNATURE, PANCOPY, FORM60)
```

```
    VALUES('C1', 'Ivan', 'Nelson', 'Bayross', '25-JUN-1952', 'Self Employed',  
    'D:/ClntPht/C1.gif', 'D:/ClntSgnt/C1.gif', 'Y', 'Y');
```

```
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH,  
SIGNATURE, PANCOPY, FORM60)
```

```
    VALUES('C1', 'Chriselle', 'Ivan', 'Bayross', '29-OCT-1982', 'Service',  
    'D:/ClntPht/C2.gif', 'D:/ClntSgnt/C2.gif', 'N', 'Y');
```

```
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH,  
SIGNATURE, PANCOPY, FORM60)
```

```
    VALUES('C2', 'Mamta', 'Arvind', 'Muzumdar', '28-AUG-1975', 'Service',  
    'D:/ClntPht/C3.gif', 'D:/ClntSgnt/C3.gif', 'Y', 'Y');
```

NULL Value Concepts

Sample 1

First drop the table CUST_MSTR if it exist and then create it again but make Date of Birth field not null. Refer to the details of table in chapter 6

```
CREATE TABLE "DBA_BANKSYS"."CUST_MSTR"(  
    "CUST_NO" VARCHAR2(10),  
    "FNAME" VARCHAR2(25),  
    "MNAME" VARCHAR2(25),  
    "LNAME" VARCHAR2(25),  
    "DOB_INC" DATE NOT NULL,  
    "OCCUP" VARCHAR2(25),  
    "PHOTOGRAPH" VARCHAR2(25),  
    "SIGNATURE" VARCHAR2(25),  
    "PANCOPY" VARCHAR2(1),  
    "FORM60" VARCHAR2(1));
```

```
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH,  
SIGNATURE, PANCOPY, FORM60)
```

```
    VALUES('O14', null, null, null, null, 'Retail Business', null, null, 'N', 'Y');
```

Check Constraint

Sample 1

Create a table CUST_MSTR with the following check constraints:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

1. Data values being inserted into the column CUST_NO must start with the capital letter C
2. Data values being inserted into the column FNAME, MNAME and LNAME should be in upper case only

```
CREATE TABLE CUST_MSTR(  
    "CUST_NO" VARCHAR2(10) CHECK(CUST_NO LIKE 'C%'),  
    "FNAME" VARCHAR2(25) CHECK (FNAME = upper(Fname)),  
    "MNAME" VARCHAR2(25) CHECK (MNAME = upper(Mname)),  
    "LNAME" VARCHAR2(25) CHECK (LNAME = upper(Lname)),  
    "DOB_INC" DATE,  
    "OCCUP" VARCHAR2(25),  
    "PHOTOGRAPH" VARCHAR2(25),  
    "SIGNATURE" VARCHAR2(25),  
    "PANCOPY" VARCHAR2(1),  
    "FORM60" VARCHAR2(1));
```

Sample 2

Create a table CUST_MSTR with the following check constraints:

1. Data values being inserted into the column CUST_NO must start with the capital letter C
2. Data values being inserted into the column FNAME, MNAME and LNAME should be in upper case only

```
CREATE TABLE CUST_MSTR(  
    "CUST_NO" VARCHAR2(10),  
    "FNAME" VARCHAR2(25),  
    "MNAME" VARCHAR2(25),  
    "LNAME" VARCHAR2(25),  
    "DOB_INC" DATE,  
    "OCCUP" VARCHAR2(25),  
    "PHOTOGRAPH" VARCHAR2(25),  
    "SIGNATURE" VARCHAR2(25),  
    "PANCOPY" VARCHAR2(1),  
    "FORM60" VARCHAR2(1),  
CHECK (CUST_NO LIKE 'C%'),  
CHECK (FNAME = upper(Fname)),  
CHECK (MNAME = upper(Mname)),  
CHECK (LNAME = upper(Lname)));
```

DEFINING DIFFERENT CONSTRAINTS ON A TABLE

Sample 1

Create FD_MSTR table where

1. The BRANCH_NO is the foreign key from the table BRANCH_MSTR
2. The CORP_CUST_NO is the primary key of this table and a foreign key from CUST_MSTR
3. The FD_SER_NO is a primary key
4. The VERI_EMP_NO is a foreign key from the table EMP_MSTR
5. The CORP_CNST_TYPE will have values ØS, 1C, 2C, 3C, 4C, 5C, 6C, 7C for different types of companies

```
CREATE TABLE "DBA_BANKSYS"."FD_MSTR"(  
    "FD_SER_NO" VARCHAR2(10),  
    "SF_NO" VARCHAR2(10),  
    "BRANCH_NO" VARCHAR2(10),  
    "INTRO_CUST_NO" VARCHAR2(10),  
    "INTRO_ACCT_NO" VARCHAR2(10),  
    "INTRO_SIGN" VARCHAR2(1),  
    "ACCT_NO" VARCHAR2(10),  
    "TITLE" VARCHAR2(30),  
    "CORP_CUST_NO" VARCHAR2(10),
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
"CORP_CNST_TYPE" VARCHAR(4),
"VERI_EMP_NO" VARCHAR2(10),
"VERI_SIGN" VARCHAR2(1) ,
"MANAGER_SIGN" VARCHAR2(1),
CONSTRAINT PK PRIMARY KEY (FD_SER_NO, CORP_CUST_NO),
CONSTRAINT FK_BR FOREIGN KEY (BRANCH_NO)
    REFERENCES BRANCH_MSTR,
CONSTRAINT FK_CU FOREIGN KEY (CORP_CUST_NO)
    REFERENCES CUST_MSTR,
CONSTRAINT FK_EM FOREIGN KEY (VERI_EMP_NO)
    REFERENCES EMP_MSTR,
CONSTRAINT CHK CHECK (CORP_CNST_TYPE IN ('ØS', '1C', '2C', '3C', '4C', '5C', '6C', '7C')));
```

User Constraints Table

Sample 1

View the contents of the table CUST_MSTR

```
SELECT Owner, Constraint_Name, Constraint_type
FROM USER_CONSTRAINTS
WHERE Table_Name = 'CUST_MSTR' ;
```

DEFINING INTEGRITY CONSTRAINTS VIA THE ALTER TABLE COMMAND

Sample 1

Alter the table EMP_MSTR by adding a primary key on the column EMP_NO

```
ALTER TABLE EMP_MSTR
ADD PRIMARY KEY (EMP_NO);
```

Sample 2

Add FOREIGN KEY constraint on the column VERI_EMP_NO belonging to the table FD_MSTR, which references the table EMP_MSTR. Modify column MANAGER_SIGN to include the NOT NULL constraint

```
ALTER TABLE FD_MSTR
ADD CONSTRAINT F_EmpKey FOREIGN KEY(VERI_EMP_NO)
REFERENCES EMP_MSTR
MODIFY (MANAGER_SIGN NOT NULL);
```

DROPPING INTEGRITY CONSTRAINTS VIA THE ALTER TABLE COMMAND

Sample 1

Drop the PRIMARY KEY constraint from EMP_MSTR.

```
ALTER TABLE EMP_MSTR DROP PRIMARY KEY;
```

Sample 2

Drop FOREIGN KEY constraint on column VERI_EMP_NO in table FD_MSTR

```
ALTER TABLE FD_MSTR DROP CONSTRAINT F_EmpKey;
```

DEFAULT VALUE CONCEPTS

Sample 1

Create ACCT_MSTR table where the column CURBAL is the number and by default it should be zero. The other column STATUS is a varchar2 and by default it should have character A (Refer to table in the chapter 6)

```
CREATE TABLE "DBA_BANKSYS"."ACCT_MSTR"(  

```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
"ACCT_NO" VARCHAR2(10),
"SF_NO" VARCHAR2(10),
"LF_NO" VARCHAR2(10),
"BRANCH_NO" VARCHAR2(10),
"INTRO_CUST_NO" VARCHAR2(10),
"INTRO_ACCT_NO" VARCHAR2(10),
"INTRO_SIGN" VARCHAR2(1),
"TYPE" VARCHAR2(2),
"OPR_MODE" VARCHAR2(2),
"CUR_ACCT_TYPE" VARCHAR2(4),
"TITLE" VARCHAR2(30),
"CORP_CUST_NO" VARCHAR2(10),
"APLNDT" DATE,
"OPNDT" DATE,
"VERI_EMP_NO" VARCHAR2(10),
"VERI_SIGN" VARCHAR2(1),
"MANAGER_SIGN" VARCHAR2(1),
"CURBAL" NUMBER(8, 2) DEFAULT 0,
"STATUS" VARCHAR2(1) DEFAULT 'A');
```

```
DROP TABLE TMP_FD_AMT;
DROP TABLE TRANS_DTLS;
DROP TABLE TRANS_MSTR;
DROP TABLE CNTC_DTLS;
DROP TABLE ADDR_DTLS;
DROP TABLE ACCT_FD_CUST_DTLS;
DROP TABLE NOMINEE_MSTR;
DROP TABLE FD_DTLS;
DROP TABLE FDSLAB_MSTR;
DROP TABLE FD_MSTR;
DROP TABLE ACCT_MSTR;
DROP TABLE SPRT_DOC;
DROP TABLE CUST_MSTR;
DROP TABLE EMP_MSTR;
DROP TABLE BRANCH_MSTR;
```

-- BRANCH_MSTR

```
CREATE TABLE "DBA_BANKSYS"."BRANCH_MSTR"(
  "BRANCH_NO" VARCHAR2(10),
  "NAME" VARCHAR2(25),
  CONSTRAINT PK_BRANCHMSTR_BRANCHNO PRIMARY KEY(BRANCH_NO),
  CONSTRAINT CHK_BRANCHMSTR_BRANCHNO CHECK(BRANCH_NO LIKE 'B%'));
```

-- EMP_MSTR

```
CREATE TABLE "DBA_BANKSYS"."EMP_MSTR"(
  "EMP_NO" VARCHAR2(10),
  "BRANCH_NO" VARCHAR2(10),
  "FNAME" VARCHAR2(25),
  "MNAME" VARCHAR2(25),
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
"LNAME" VARCHAR2(25),
"DEPT" VARCHAR2(30),
"DESIG" VARCHAR2(30),
"MNGR_NO" VARCHAR2(10),
CONSTRAINT PK_EMPMSTR_EMPNO PRIMARY KEY(EMP_NO),
CONSTRAINT CHK_EMPMSTR_EMPNO CHECK(EMP_NO LIKE 'E%'),
CONSTRAINT FK_EMPMSTR_BRANCHNO FOREIGN KEY(BRANCH_NO)
    REFERENCES BRANCH_MSTR(BRANCH_NO),
CONSTRAINT FK_EMPMSTR_MNGRNO FOREIGN KEY(MNGR_NO)
    REFERENCES EMP_MSTR(EMP_NO));
```

-- CUST_MSTR

```
CREATE TABLE "DBA_BANKSYS"."CUST_MSTR"(
    "CUST_NO" VARCHAR2(10),
    "FNAME" VARCHAR2(25),
    "MNAME" VARCHAR2(25),
    "LNAME" VARCHAR2(25),
    "DOB_INC" DATE NOT NULL,
    "OCCUP" VARCHAR2(25),
    "PHOTOGRAPH" VARCHAR2(25),
    "SIGNATURE" VARCHAR2(25),
    "PANCOPY" VARCHAR2(1),
    "FORM60" VARCHAR2(1),
    CONSTRAINT PK_CUSTMSTR_CUSTNO PRIMARY KEY(CUST_NO),
    CONSTRAINT CHK_CUSTMSTR_CUSTNO
        CHECK(CUST_NO LIKE 'C%' OR CUST_NO LIKE 'O%'),
    CONSTRAINT CHK_CUSTMSTR_PANCOPY CHECK(PANCOPY IN('Y', 'N')),
    CONSTRAINT CHK_CUSTMSTR_FORM60 CHECK(FORM60 IN('Y', 'N')));
```

-- SPRT_DOC

```
CREATE TABLE "DBA_BANKSYS"."SPRT_DOC"(
    "ACCT_CODE" VARCHAR2(4),
    "TYPE" VARCHAR2(40),
    "DOCS" VARCHAR2(75));
```

-- ACCT_MSTR

```
CREATE TABLE "DBA_BANKSYS"."ACCT_MSTR"(
    "ACCT_NO" VARCHAR2(10),
    "SF_NO" VARCHAR2(10),
    "LF_NO" VARCHAR2(10),
    "BRANCH_NO" VARCHAR2(10),
    "INTRO_CUST_NO" VARCHAR2(10),
    "INTRO_ACCT_NO" VARCHAR2(10),
    "INTRO_SIGN" VARCHAR2(1),
    "TYPE" VARCHAR2(2),
    "OPR_MODE" VARCHAR2(2),
    "CUR_ACCT_TYPE" VARCHAR2(4),
    "TITLE" VARCHAR2(30),
    "CORP_CUST_NO" VARCHAR2(10),
    "APLNDT" DATE,
    "OPNDT" DATE,
    "VERI_EMP_NO" VARCHAR2(10),
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
"VERI_SIGN" VARCHAR2(1),
"MANAGER_SIGN" VARCHAR2(1),
"CURBAL" NUMBER(8, 2) DEFAULT 0,
"STATUS" VARCHAR2(1) DEFAULT 'A',
CONSTRAINT PK_ACCTMSTR_ACCTNO PRIMARY KEY(ACCT_NO),
CONSTRAINT FK_ACCTMSTR_BRANCHNO FOREIGN KEY(BRANCH_NO)
    REFERENCES BRANCH_MSTR(BRANCH_NO),
CONSTRAINT FK_ACCTMSTR_INTRO_CUSTNO FOREIGN KEY(INTRO_CUST_NO)
    REFERENCES CUST_MSTR(CUST_NO),
CONSTRAINT FK_ACCTMSTR_INTROACCTNO FOREIGN KEY(INTRO_ACCT_NO)
    REFERENCES ACCT_MSTR(ACCT_NO),
CONSTRAINT CHK_ACCTMSTR_INTROSIGN CHECK(INTRO_SIGN IN('Y', 'N')),
CONSTRAINT CHK_ACCTMSTR_TYPE CHECK(TYPE IN('SB', 'CA')),
CONSTRAINT CHK_ACCTMSTR_OPRMODE CHECK(OPR_MODE IN('SI', 'ES', 'JO', 'AS')),
CONSTRAINT CHK_ACCTMSTR_CURACCTTYPE
    CHECK(CUR_ACCT_TYPE IN('0S', '1C', '2C', '3C', '4C', '5C', '6C', '7C')),
CONSTRAINT CHK_ACCTMSTR_CORPCUSTNO CHECK(CORP_CUST_NO LIKE 'O%'),
CONSTRAINT FK_ACCTMSTR_VERIEMPNO FOREIGN KEY(VERI_EMP_NO)
    REFERENCES EMP_MSTR(EMP_NO),
CONSTRAINT CHK_ACCTMSTR_VERISIGN CHECK(VERI_SIGN IN('Y', 'N')),
CONSTRAINT CHK_ACCTMSTR_MANAGERSIGN CHECK(MANAGER_SIGN IN('Y', 'N')),
CONSTRAINT CHK_ACCTMSTR_STATUS CHECK(STATUS IN('A', 'S', 'T'));
```

-- FD_MSTR

```
CREATE TABLE "DBA_BANKSYS"."FD_MSTR"(
    "FD_SER_NO" VARCHAR2(10),
    "SF_NO" VARCHAR2(10),
    "BRANCH_NO" VARCHAR2(10),
    "INTRO_CUST_NO" VARCHAR2(10),
    "INTRO_ACCT_NO" VARCHAR2(10),
    "INTRO_SIGN" VARCHAR2(1),
    "ACCT_NO" VARCHAR2(10),
    "TITLE" VARCHAR2(30),
    "CORP_CUST_NO" VARCHAR2(10),
    "CORP_CNST_TYPE" VARCHAR(4),
    "VERI_EMP_NO" VARCHAR2(10),
    "VERI_SIGN" VARCHAR2(1),
    "MANAGER_SIGN" VARCHAR2(1),
    CONSTRAINT PK_FDMSTR_FDSENO PRIMARY KEY(FD_SER_NO),
    CONSTRAINT FK_FDMSTR_BRANCHNO FOREIGN KEY(BRANCH_NO)
        REFERENCES BRANCH_MSTR(BRANCH_NO),
    CONSTRAINT FK_FDMSTR_INTRO_CUSTNO FOREIGN KEY(INTRO_CUST_NO)
        REFERENCES CUST_MSTR(CUST_NO),
    CONSTRAINT FK_FDMSTR_INTROACCTNO FOREIGN KEY(INTRO_ACCT_NO)
        REFERENCES ACCT_MSTR(ACCT_NO),
    CONSTRAINT CHK_FDMSTR_INTROSIGN CHECK(INTRO_SIGN IN('Y', 'N')),
    CONSTRAINT CHK_FDMSTR_ACCTNO CHECK(ACCT_NO LIKE 'CA%' OR ACCT_NO LIKE 'SB%'),
    CONSTRAINT CHK_FDMSTR_CORPCUSTNO CHECK(CORP_CUST_NO LIKE 'O%'),
    CONSTRAINT CHK_FDMSTR_CORPCNSTTYPE
        CHECK(CORP_CNST_TYPE IN('0S', '1C', '2C', '3C', '4C', '5C', '6C', '7C')),
    CONSTRAINT FK_FDMSTR_VERIEMPNO FOREIGN KEY(VERI_EMP_NO)
        REFERENCES EMP_MSTR(EMP_NO),
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
CONSTRAINT CHK_FDMSTR_VERISIGN CHECK(VERI_SIGN IN('Y', 'N')),  
CONSTRAINT CHK_FDMSTR_MANAGERSIGN CHECK(MANAGER_SIGN IN('Y', 'N'));
```

-- FDSLAB_MSTR

```
CREATE TABLE "DBA_BANKSYS"."FDSLAB_MSTR"(  
    "FDSLAB_NO" NUMBER(2),  
    "MINPERIOD" NUMBER(5),  
    "MAXPERIOD" NUMBER(5),  
    "INTRATE" NUMBER(5,2),  
    CONSTRAINT PK_FDSLABMSTR_FDSLABNO PRIMARY KEY(FDSLAB_NO));
```

-- FD_DTLS

```
CREATE TABLE "DBA_BANKSYS"."FD_DTLS"(  
    "FD_SER_NO" VARCHAR2(10),  
    "FD_NO" VARCHAR2(10),  
    "TYPE" VARCHAR2(1),  
    "PAYTO_ACCTNO" VARCHAR2(10),  
    "PERIOD" NUMBER(5),  
    "OPNDT" DATE,  
    "DUEDT" DATE,  
    "AMT" NUMBER(8,2),  
    "DUEAMT" NUMBER(8,2),  
    "INTRATE" NUMBER(3),  
    "STATUS" VARCHAR2(1) DEFAULT 'A',  
    "AUTO_RENEWAL" VARCHAR2(1),  
    CONSTRAINT PK_FDDTLS_FDNO PRIMARY KEY(FD_NO),  
    CONSTRAINT FK_FDDTLS_FDSENO FOREIGN KEY(FD_SER_NO)  
        REFERENCES FD_MSTR(FD_SER_NO),  
    CONSTRAINT CHK_FDDTLS_TYPE CHECK(TYPE IN('S', 'R')),  
    CONSTRAINT CHK_FDDTKS_PAYTOACCTNO CHECK(PAYTO_ACCTNO LIKE 'CA%' OR PAYTO_ACCTNO  
    LIKE 'SB%'),  
    CONSTRAINT CHK_FDDTLS_STATUS CHECK(STATUS IN('A', 'C', 'M')),  
    CONSTRAINT CHK_FDDTLS_AUTORENEWAL CHECK(AUTO_RENEWAL IN('Y', 'N')));
```

-- ACCT_FD_CUST_DTLS

```
CREATE TABLE "DBA_BANKSYS"."ACCT_FD_CUST_DTLS"(  
    "ACCT_FD_NO" VARCHAR2(10),  
    "CUST_NO" VARCHAR2(10),  
    CONSTRAINT CHK_ACCTFDCUSTDTLS_ACCTFDNO  
        CHECK(ACCT_FD_NO LIKE 'CA%' OR ACCT_FD_NO LIKE 'FS%'  
        OR ACCT_FD_NO LIKE 'SB%'),  
    CONSTRAINT FK_ACCTFDCUSTDTLS_CUSTNO FOREIGN KEY(CUST_NO)  
        REFERENCES CUST_MSTR(CUST_NO));
```

-- NOMINEE_MSTR

```
CREATE TABLE "DBA_BANKSYS"."NOMINEE_MSTR"(  
    "NOMINEE_NO" VARCHAR2(10),  
    "ACCT_FD_NO" VARCHAR2(10),  
    "NAME" VARCHAR2(75),  
    "DOB" DATE,  
    "RELATIONSHIP" VARCHAR2(25),
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
CONSTRAINT PK_NOMINEEMSTR_NOMINEENO PRIMARY KEY(NOMINEE_NO),
CONSTRAINT CHK_NOMINEEMSTR_ACCTFDNO
CHECK(ACCT_FD_NO LIKE 'CA%' OR ACCT_FD_NO LIKE 'FS%'
OR ACCT_FD_NO LIKE 'SB%');
```

-- ADDR_DTLS

```
CREATE TABLE "DBA_BANKSYS"."ADDR_DTLS"(
"ADDR_NO" NUMBER(6),
"CODE_NO" VARCHAR2(10),
"ADDR_TYPE" VARCHAR2(1),
"ADDR1" VARCHAR2(50),
"ADDR2" VARCHAR2(50),
"CITY" VARCHAR2(25),
"STATE" VARCHAR2(25),
"PINCODE" VARCHAR2(6),
CONSTRAINT PK_ADDRDTLS_ADDRNO PRIMARY KEY(ADDR_NO),
CONSTRAINT CHK_ADDRDTLS_CODENO
CHECK(CODE_NO LIKE 'B%' OR CODE_NO LIKE 'C%' OR CODE_NO LIKE 'E%'
OR CODE_NO LIKE 'N%' OR CODE_NO LIKE 'O%'),
CONSTRAINT CHK_ADDRDTLS_ADDRRTYPE
CHECK(ADDR_TYPE IN('C', 'N', 'H', 'B')));
```

-- CNTC_DTLS

```
CREATE TABLE "DBA_BANKSYS"."CNTC_DTLS"(
"ADDR_NO" NUMBER(6),
"CODE_NO" VARCHAR2(10),
"CNTC_TYPE" VARCHAR2(1),
"CNTC_DATA" VARCHAR2(75),
CONSTRAINT FK_CNTCDTLS_ADDRNO FOREIGN KEY(ADDR_NO)
REFERENCES ADDR_DTLS(ADDR_NO),
CONSTRAINT CHK_CNTCDTLS_CODENO
CHECK(CODE_NO LIKE 'B%' OR CODE_NO LIKE 'C%' OR CODE_NO LIKE 'E%'
OR CODE_NO LIKE 'N%' OR CODE_NO LIKE 'O%'),
CONSTRAINT CHK_CNTCDTLS_CNTCTYPE
CHECK(CNTC_TYPE IN('R', 'O', 'M', 'P', 'E', 'F', 'W')));
```

-- TRANS_MSTR

```
CREATE TABLE "DBA_BANKSYS"."TRANS_MSTR"(
"TRANS_NO" VARCHAR2(10),
"ACCT_NO" VARCHAR2(10),
"DT" DATE,
"TYPE" VARCHAR2(1),
"PARTICULAR" VARCHAR2(30),
"DR_CR" VARCHAR2(1),
"AMT" NUMBER(8,2),
"BALANCE" NUMBER(8,2),
CONSTRAINT PK_TRANSMSTR_TRANSNO PRIMARY KEY(TRANS_NO),
CONSTRAINT CHK_TRANSMSTR_ACCTNO
CHECK(ACCT_NO LIKE 'CA%' OR ACCT_NO LIKE 'SB%'),
CONSTRAINT CHK_TRANSMSTR_TYPE CHECK(TYPE IN('B', 'C', 'D')),
CONSTRAINT CHK_TRANSMSTR_DRCR CHECK(DR_CR IN('D', 'W')));
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

-- TRANS_DTLS

```
CREATE TABLE "DBA_BANKSYS"."TRANS_DTLS"(  
    "TRANS_NO" VARCHAR2(10),  
    "INST_NO" NUMBER(6),  
    "INST_DT" DATE,  
    "PAYTO" VARCHAR2(30),  
    "INST_CLR_DT" DATE,  
    "BANK_NAME" VARCHAR2(35),  
    "BRANCH_NAME" VARCHAR2(25),  
    "PAIDFROM" VARCHAR2(10),  
    CONSTRAINT FK_TRANSDTLS_TRANSNO FOREIGN KEY(TRANS_NO)  
        REFERENCES TRANS_MSTR(TRANS_NO));
```

-- TMP_FD_AMT

```
CREATE TABLE "DBA_BANKSYS"."TMP_FD_AMT"(  
    "FD_AMT" NUMBER(6));
```

-- Records for BRANCH_MSTR

```
INSERT INTO TMP_FD_AMT (FD_AMT) VALUES(5000);  
INSERT INTO TMP_FD_AMT (FD_AMT) VALUES(10000);  
INSERT INTO TMP_FD_AMT (FD_AMT) VALUES(15000);  
INSERT INTO TMP_FD_AMT (FD_AMT) VALUES(20000);  
INSERT INTO TMP_FD_AMT (FD_AMT) VALUES(25000);  
INSERT INTO TMP_FD_AMT (FD_AMT) VALUES(30000);  
INSERT INTO TMP_FD_AMT (FD_AMT) VALUES(4000);  
INSERT INTO TMP_FD_AMT (FD_AMT) VALUES(50000);
```

-- Records for BRANCH_MSTR

```
INSERT INTO BRANCH_MSTR (BRANCH_NO, NAME) VALUES('B1', 'Vile Parle (HO));  
INSERT INTO BRANCH_MSTR (BRANCH_NO, NAME) VALUES('B2', 'Andheri');  
INSERT INTO BRANCH_MSTR (BRANCH_NO, NAME) VALUES('B3', 'Churchgate');  
INSERT INTO BRANCH_MSTR (BRANCH_NO, NAME) VALUES('B4', 'Mahim');  
INSERT INTO BRANCH_MSTR (BRANCH_NO, NAME) VALUES('B5', 'Borivali');  
INSERT INTO BRANCH_MSTR (BRANCH_NO, NAME) VALUES('B6', 'Darya Ganj');
```

-- Records for EMP_MSTR

```
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG, MNGR_NO)  
    VALUES('E1', 'B1', 'Ivan', 'Nelson', 'Bayross', 'Administration', 'Managing Director', NULL);  
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG, MNGR_NO)  
    VALUES('E2', 'B2', 'Amit', null, 'Desai', 'Loans And Financing', 'Finance Manager', NULL);  
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG, MNGR_NO)  
    VALUES('E3', 'B3', 'Maya', 'Mahima', 'Joshi', 'Client Servicing', 'Sales Manager', NULL);  
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG, MNGR_NO)  
    VALUES('E4', 'B1', 'Peter', 'Iyer', 'Joseph', 'Loans And Financing', 'Clerk', 'E2');  
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG, MNGR_NO)  
    VALUES('E5', 'B4', 'Mandhar', 'Dilip', 'Dalvi', 'Marketing', 'Marketing Manager', NULL);  
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG, MNGR_NO)  
    VALUES('E6', 'B6', 'Sonal', 'Abdul', 'Khan', 'Administration', 'Admin. Executive', 'E1');  
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG, MNGR_NO)  
    VALUES('E7', 'B4', 'Anil', 'Ashutosh', 'Kambli', 'Marketing', 'Sales Asst.', 'E5');  
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG, MNGR_NO)  
    VALUES('E8', 'B3', 'Seema', 'P.', 'Apte', 'Client Servicing', 'Clerk', 'E3');
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG, MNGR_NO)
VALUES('E9', 'B2', 'Vikram', 'Vilas', 'Randive', 'Marketing', 'Sales Asst.', 'E7');
```

```
INSERT INTO EMP_MSTR (EMP_NO, BRANCH_NO, FNAME, MNAME, LNAME, DEPT, DESIG, MNGR_NO)
VALUES('E10', 'B6', 'Anjali', 'Sameer', 'Pathak', 'Administration', 'HR Manager', 'E1');
```

-- Records for CUST_MSTR

```
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH,
SIGNATURE, PANCOPY, FORM60)
```

```
VALUES('C1', 'Ivan', 'Nelson', 'Bayross', '25-JUN-1952', 'Self Employed',
'D:/ClntPht/C1.gif', 'D:/ClntSgnt/C1.gif', 'Y', 'Y');
```

```
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH,
SIGNATURE, PANCOPY, FORM60)
```

```
VALUES('C2', 'Chriselle', 'Ivan', 'Bayross', '29-OCT-1982', 'Service',
'D:/ClntPht/C2.gif', 'D:/ClntSgnt/C2.gif', 'N', 'Y');
```

```
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH,
SIGNATURE, PANCOPY, FORM60)
```

```
VALUES('C3', 'Mamta', 'Arvind', 'Muzumdar', '28-AUG-1975', 'Service',
'D:/ClntPht/C3.gif', 'D:/ClntSgnt/C3.gif', 'Y', 'Y');
```

```
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH,
SIGNATURE, PANCOPY, FORM60)
```

```
VALUES('C4', 'Chhaya', 'Sudhakar', 'Bankar', '06-OCT-1976', 'Service',
'D:/ClntPht/C4.gif', 'D:/ClntSgnt/C4.gif', 'Y', 'Y');
```

```
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH,
SIGNATURE, PANCOPY, FORM60)
```

```
VALUES('C5', 'Ashwini', 'Dilip', 'Joshi', '20-NOV-1978', 'Business',
'D:/ClntPht/C5.gif', 'D:/ClntSgnt/C5.gif', 'Y', 'Y');
```

```
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH,
SIGNATURE, PANCOPY, FORM60)
```

```
VALUES('C6', 'Hansel', 'I.', 'Colaco', '01-JAN-1982', 'Service',
'D:/ClntPht/C6.gif', 'D:/ClntSgnt/C6.gif', 'N', 'Y');
```

```
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH,
SIGNATURE, PANCOPY, FORM60)
```

```
VALUES('C7', 'Anil', 'Arun', 'Dhone', '12-OCT-1983', 'Self Employed',
'D:/ClntPht/C7.gif', 'D:/ClntSgnt/C7.gif', 'N', 'Y');
```

```
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH,
SIGNATURE, PANCOPY, FORM60)
```

```
VALUES('C8', 'Alex', 'Austin', 'Fernandes', '30-SEP-1962', 'Executive',
'D:/ClntPht/C8.gif', 'D:/ClntSgnt/C8.gif', 'Y', 'Y');
```

```
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH,
SIGNATURE, PANCOPY, FORM60)
```

```
VALUES('C9', 'Ashwini', 'Shankar', 'Apte', '19-APR-1979', 'Service',
'D:/ClntPht/C9.gif', 'D:/ClntSgnt/C9.gif', 'Y', 'Y');
```

```
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH,
SIGNATURE, PANCOPY, FORM60)
```

```
VALUES('C10', 'Namita', 'S.', 'Kanade', '10-JUN-1978', 'Self Employed',
'D:/ClntPht/C10.gif', 'D:/ClntSgnt/C10.gif', 'Y', 'Y');
```

```
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH,
SIGNATURE, PANCOPY, FORM60)
```

```
VALUES('O11', null, null, null, '14-NOV-1997', 'Retail Business', null, null, 'Y', 'N');
```

```
INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH,
SIGNATURE, PANCOPY, FORM60)
```

```
VALUES('O12', null, null, null, '23-OCT-1992', 'Information Technology', null, null, 'Y', 'N');
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH, SIGNATURE, PANCOPY, FORM60)

VALUES('O13', null, null, null, '05-FEB-1989', 'Community Welfare', null, null, 'Y', 'N');

INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH, SIGNATURE, PANCOPY, FORM60)

VALUES('O14', null, null, null, '24-MAY-1980', 'Retail Business', null, null, 'N', 'Y');

INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH, SIGNATURE, PANCOPY, FORM60)

VALUES('O15', null, null, null, '02-APR-2000', 'Retail Business', null, null, 'Y', 'N');

INSERT INTO CUST_MSTR (CUST_NO, FNAME, MNAME, LNAME, DOB_INC, OCCUP, PHOTOGRAPH, SIGNATURE, PANCOPY, FORM60)

VALUES('O16', null, null, null, '13-JAN-2002', 'Marketing', null, null, 'Y', 'N');

-- Records for SPRT_DOC

INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)

VALUES('0S', 'Individuals / Savings Bank Account', 'Driving Licence / Ration Card / Passport');

INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)

VALUES('0S', 'Individuals / Savings Bank Account', 'Birth Certificate / School Leaving Certificate');

INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)

VALUES('1C', 'Propriety / Sole Trading Concerns', 'Letter From The Propriety');

INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)

VALUES('2C', 'Partnership Concerns', 'Letter From The Partners');

INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)

VALUES('2C', 'Partnership Concerns', 'Partnership Deed / Registration Certificate');

INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)

VALUES('3C', 'Hindu Undivided Family Businesses', 'Letter From The Karta');

INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)

VALUES('3C', 'Hindu Undivided Family Businesses', 'List Of Members');

INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)

VALUES('4C', 'Limited Companies', 'Copy Of Board Of Directors" Resolution For Opening The Account');

INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)

VALUES('4C', 'Limited Companies', 'Memorandum and Articles Of Association');

INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)

VALUES('4C', 'Limited Companies', 'Certificate Of Incorporation');

INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)

VALUES('4C', 'Limited Companies', 'Certificate Of Commencement Of Business / Registration Certificate');

INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)

VALUES('5C', 'Trust Accounts', 'Trust Deed');

INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)

VALUES('5C', 'Trust Accounts', 'Resolution Of Trustees');

INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)

VALUES('5C', 'Trust Accounts', 'List Of Trustees');

INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)

VALUES('6C', 'Clubs / Societies', 'Resolution');

INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)

VALUES('6C', 'Clubs / Societies', 'Constitution And Bye-laws');

INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)

VALUES('6C', 'Clubs / Societies', 'Certificate Of Registration');

INSERT INTO SPRT_DOC (ACCT_CODE, TYPE, DOCS)

VALUES('7C', 'Legislative Bodies', 'Letter From The Authority');

-- Records for ACCT_MSTR

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
VALUES('SB1', 'SF-0001', 'NOV03-05', 'B1', 'C1', 'SB1', 'Y', 'SB', 'SI', '0S', null, null,
'05-NOV-2003', '05-NOV-2003', 'E1', 'Y', 'Y', 500, 'A');
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
VALUES('CA2', 'SF-0002', 'NOV03-10', 'B2', 'C1', 'SB1', 'Y', 'CA', 'JO', '1C', 'Uttam Stores', 'O11',
'07-NOV-2003', '10-NOV-2003', 'E1', 'Y', 'Y', 3000, 'A');
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
VALUES('SB3', 'SF-0003', 'NOV03-22', 'B3', 'C4', 'SB3', 'Y', 'SB', 'SI', '0S', null, null,
'20-NOV-2003', '22-NOV-2003', 'E4', 'Y', 'Y', 500, 'A');
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
VALUES('CA4', 'SF-0004', 'DEC03-05', 'B5', 'C4', 'SB3', 'Y', 'CA', 'AS', '4C', 'Sun's Pvt. Ltd.', 'O12',
'02-DEC-2003', '05-DEC-2003', 'E4', 'Y', 'Y', 12000, 'A');
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
VALUES('SB5', 'SF-0005', 'DEC03-15', 'B6', 'C1', 'SB1', 'Y', 'SB', 'JO', '0S', null, null,
'14-DEC-2003', '15-DEC-2003', 'E1', 'Y', 'Y', 500, 'A');
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
VALUES('SB6', 'SF-0006', 'DEC03-27', 'B4', 'C5', 'SB6', 'Y', 'SB', 'ES', '0S', null, null,
'27-DEC-2003', '27-DEC-2003', 'E4', 'Y', 'Y', 500, 'A');
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
VALUES('CA7', 'SF-0007', 'JAN04-14', 'B1', 'C8', 'CA7', 'Y', 'CA', 'AS', '6C', 'Puru Hsg. Soc', 'O13',
'14-JAN-2004', '14-JAN-2004', 'E4', 'Y', 'Y', 22000, 'A');
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
VALUES('SB8', 'SF-0008', 'JAN04-29', 'B2', 'C9', 'SB8', 'Y', 'SB', 'SI', '0S', null, null,
'27-JAN-2004', '29-JAN-2004', 'E1', 'Y', 'Y', 500, 'A');
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
VALUES('SB9', 'SF-0009', 'FEB04-05', 'B4', 'C10', 'SB9', 'Y', 'SB', 'JO', '0S', null, null,
      '05-FEB-2004', '05-FEB-2004', 'E4', 'Y', 'Y', 500, 'A');
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
      CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
VALUES('CA10', 'SF-0010', 'FEB04-19', 'B6', 'C10', 'SB9', 'Y', 'CA', 'AS', '3C', 'Ghar Karobar', 'O14',
      '19-FEB-2004', '19-FEB-2004', 'E4', 'Y', 'Y', 32000, 'A');

INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
      CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
VALUES('SB11', 'SF-0011', 'MAR04-10', 'B1', 'C1', 'SB1', 'Y', 'SB', 'SI', '0S', null, null,
      '05-MAR-2004', '10-MAR-2004', 'E1', 'Y', 'Y', 500, 'A');
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
      CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
VALUES('CA12', 'SF-0012', 'MAR04-10', 'B2', 'C1', 'SB5', 'Y', 'CA', 'JO', '1C', 'Suresh Stores', 'O15',
      '07-MAR-2004', '10-MAR-2004', 'E1', 'Y', 'Y', 5000, 'A');
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
      CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
VALUES('SB13', 'SF-0013', 'MAR04-22', 'B3', 'C4', 'SB3', 'Y', 'SB', 'SI', '0S', null, null,
      '20-MAR-2004', '22-MAR-2004', 'E4', 'Y', 'Y', 500, 'A');
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
      CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
VALUES('CA14', 'SF-0014', 'APR04-05', 'B5', 'C4', 'SB3', 'Y', 'CA', 'AS', '4C', 'Moon"s Pvt. Ltd.', 'O16',
      '02-APR-2004', '05-APR-2004', 'E4', 'Y', 'Y', 10000, 'A');
INSERT INTO ACCT_MSTR (ACCT_NO, SF_NO, LF_NO, BRANCH_NO, INTRO_CUST_NO, INTRO_ACCT_NO,
INTRO_SIGN, TYPE, OPR_MODE,
      CUR_ACCT_TYPE, TITLE, CORP_CUST_NO, APLNDT, OPNDT, VERI_EMP_NO, VERI_SIGN,
MANAGER_SIGN, CURBAL, STATUS)
VALUES('SB15', 'SF-0015', 'APR04-15', 'B6', 'C1', 'SB1', 'Y', 'SB', 'JO', '0S', null, null,
      '14-APR-2004', '15-APR-2004', 'E1', 'Y', 'Y', 500, 'A');

-- Records for FD_MSTR
INSERT INTO FD_MSTR (FD_SER_NO, SF_NO, BRANCH_NO, ACCT_NO, TITLE, CORP_CUST_NO,
CORP_CNST_TYPE, INTRO_CUST_NO,
      INTRO_ACCT_NO, INTRO_SIGN, VERI_EMP_NO, VERI_SIGN, MANAGER_SIGN)
VALUES ('FS1', 'SF-1001', 'B2', 'CA2', 'Uttam Stores', 'O11', '1C', null, null, 'N', 'E1', 'Y', 'Y');
INSERT INTO FD_MSTR (FD_SER_NO, SF_NO, BRANCH_NO, ACCT_NO, TITLE, CORP_CUST_NO,
CORP_CNST_TYPE, INTRO_CUST_NO,
      INTRO_ACCT_NO, INTRO_SIGN, VERI_EMP_NO, VERI_SIGN, MANAGER_SIGN)
VALUES ('FS2', 'SF-1002', 'B5', 'CA4', 'Sun"s Pvt. Ltd.', 'O12', '4C', null, null, 'N', 'E1', 'Y', 'Y');
INSERT INTO FD_MSTR (FD_SER_NO, SF_NO, BRANCH_NO, ACCT_NO, TITLE, CORP_CUST_NO,
CORP_CNST_TYPE, INTRO_CUST_NO,
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
INTRO_ACCT_NO, INTRO_SIGN, VERI_EMP_NO, VERI_SIGN, MANAGER_SIGN)
VALUES ('FS3', 'SF-1003', 'B1', 'CA7', 'Puru Hsg. Soc', 'O13', '6C', null, null, 'N', 'E4', 'Y', 'Y');
INSERT INTO FD_MSTR (FD_SER_NO, SF_NO, BRANCH_NO, ACCT_NO, TITLE, CORP_CUST_NO,
CORP_CNST_TYPE, INTRO_CUST_NO,
INTRO_ACCT_NO, INTRO_SIGN, VERI_EMP_NO, VERI_SIGN, MANAGER_SIGN)
VALUES ('FS4', 'SF-1004', 'B6', 'CA10', 'Ghar Karobar', 'O14', '3C', null, null, 'N', 'E4', 'Y', 'Y');
INSERT INTO FD_MSTR (FD_SER_NO, SF_NO, BRANCH_NO, ACCT_NO, TITLE, CORP_CUST_NO,
CORP_CNST_TYPE, INTRO_CUST_NO,
INTRO_ACCT_NO, INTRO_SIGN, VERI_EMP_NO, VERI_SIGN, MANAGER_SIGN)
VALUES ('FS5', 'SF-1005', 'B4', null, null, null, '0S', 'C7', 'SB6', 'Y', 'E4', 'Y', 'Y');
```

-- Record for FDSLAB_MSTR

```
INSERT INTO FDSLAB_MSTR (FDSLAB_NO, MINPERIOD, MAXPERIOD, INTRATE) VALUES(1, 1, 30, 5);
INSERT INTO FDSLAB_MSTR (FDSLAB_NO, MINPERIOD, MAXPERIOD, INTRATE) VALUES(2, 31, 92, 5.5);
INSERT INTO FDSLAB_MSTR (FDSLAB_NO, MINPERIOD, MAXPERIOD, INTRATE) VALUES(3, 93, 183, 6);
INSERT INTO FDSLAB_MSTR (FDSLAB_NO, MINPERIOD, MAXPERIOD, INTRATE) VALUES(4, 184, 365, 6.5);
INSERT INTO FDSLAB_MSTR (FDSLAB_NO, MINPERIOD, MAXPERIOD, INTRATE) VALUES(5, 366, 731, 7.5);
INSERT INTO FDSLAB_MSTR (FDSLAB_NO, MINPERIOD, MAXPERIOD, INTRATE) VALUES(6, 732, 1097, 8.5);
INSERT INTO FDSLAB_MSTR (FDSLAB_NO, MINPERIOD, MAXPERIOD, INTRATE) VALUES(7, 1098, 1829, 10);
```

-- Record for FD_DTLS

```
INSERT INTO FD_DTLS (FD_SER_NO, FD_NO, TYPE, PAYTO_ACCTNO, PERIOD, OPNDT, DUEDT, AMT,
DUEAMT,
INTRATE, STATUS, AUTO_RENEWAL)
VALUES('FS1', 'F1', 'S', 'CA2', 365, '02-JAN-2004', '01-JAN-2005', 15000, 16050.00, 6.5, 'A', 'Y');
INSERT INTO FD_DTLS (FD_SER_NO, FD_NO, TYPE, PAYTO_ACCTNO, PERIOD, OPNDT, DUEDT, AMT,
DUEAMT,
INTRATE, STATUS, AUTO_RENEWAL)
VALUES('FS1', 'F2', 'S', 'CA2', 365, '02-JAN-2004', '01-JAN-2005', 5000, 5350.00, 6.5, 'A', 'N');
INSERT INTO FD_DTLS (FD_SER_NO, FD_NO, TYPE, PAYTO_ACCTNO, PERIOD, OPNDT, DUEDT, AMT,
DUEAMT,
INTRATE, STATUS, AUTO_RENEWAL)
VALUES('FS2', 'F3', 'S', 'CA4', 366, '25-MAR-2004', '25-MAR-2005', 10000, 10802.19, 7.5, 'A', 'Y');
INSERT INTO FD_DTLS (FD_SER_NO, FD_NO, TYPE, PAYTO_ACCTNO, PERIOD, OPNDT, DUEDT, AMT,
DUEAMT,
INTRATE, STATUS, AUTO_RENEWAL)
VALUES('FS2', 'F4', 'S', 'CA4', 366, '15-APR-2004', '15-APR-2005', 10000, 10802.19, 7.5, 'A', 'Y');
INSERT INTO FD_DTLS (FD_SER_NO, FD_NO, TYPE, PAYTO_ACCTNO, PERIOD, OPNDT, DUEDT, AMT,
DUEAMT,
INTRATE, STATUS, AUTO_RENEWAL)
VALUES('FS3', 'F5', 'S', 'CA7', 183, '24-APR-2004', '24-OCT-2006', 2000, 2060.16, 6, 'A', 'Y');
INSERT INTO FD_DTLS (FD_SER_NO, FD_NO, TYPE, PAYTO_ACCTNO, PERIOD, OPNDT, DUEDT, AMT,
DUEAMT,
INTRATE, STATUS, AUTO_RENEWAL)
VALUES('FS4', 'F6', 'S', 'CA10', 732, '19-MAY-2004', '20-MAY-2006', 5000, 5902.47, 8.5, 'A', 'Y');
INSERT INTO FD_DTLS (FD_SER_NO, FD_NO, TYPE, PAYTO_ACCTNO, PERIOD, OPNDT, DUEDT, AMT,
DUEAMT,
INTRATE, STATUS, AUTO_RENEWAL)
VALUES('FS5', 'F7', 'S', 'SB6', 366, '27-MAY-2004', '27-MAY-2005', 15000, 16203.30, 7.5, 'A', 'N');
```

-- Record for ACCT_FD_CUST_DTLS

```
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB1', 'C1');
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA2', 'C2');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA2', 'C3');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB3', 'C4');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA4', 'C4');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA4', 'C5');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB5', 'C1');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB5', 'C4');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB6', 'C5');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB6', 'C7');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA7', 'C6');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA7', 'C8');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB8', 'C9');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB9', 'C3');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB9', 'C10');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA10', 'C10');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA10', 'C9');
```

```
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB11', 'C1');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA12', 'C2');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA12', 'C3');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB13', 'C4');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA14', 'C4');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('CA14', 'C5');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB15', 'C1');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('SB15', 'C4');
```

```
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS1', 'C2');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS1', 'C3');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS2', 'C4');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS2', 'C5');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS2', 'C5');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS3', 'C6');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS3', 'C8');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS4', 'C10');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS4', 'C9');
INSERT INTO ACCT_FD_CUST_DTLS (ACCT_FD_NO, CUST_NO) VALUES('FS5', 'C5');
```

-- Record for NOMINEE_MSTR

```
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N1', 'CA2', 'Joseph Martin Dias', '17-SEP-1984', 'Colleague');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N2', 'CA2', 'Nilesh Sawant', '25-AUG-1987', 'Colleague');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N3', 'SB1', 'Chriselle Ivan Bayross', '25-JUN-1952', 'Daughter');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N4', 'SB3', 'Mamta Arvind Muzumdar', '28-AUG-1975', 'Friend');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N5', 'SB6', 'Preeti Suresh Shah', '12-FEB-1978', 'Friend');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N6', 'SB8', 'Rohit Rajan Sahakarkar', '30-MAY-1985', 'Relative');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N7', 'CA10', 'Namita S. Kanade', '10-JUN-1978', 'Niece');
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N8', 'FS1', 'Rohit Rajan Sahakarkar', '30-MAY-1985', 'Relative');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N9', 'FS2', 'Joseph Martin Dias', '17-SEP-1984', 'Colleague');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N10', 'FS2', 'Nilesh Sawant', '25-AUG-1987', 'Colleague');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N11', 'FS3', 'Chriselle Ivan Bayross', '25-JUN-1952', 'Colleague');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N12', 'FS3', 'Mamta Arvind Muzumdar', '28-AUG-1975', 'Friend');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N13', 'FS4', 'Namita S. Kanade', '10-JUN-1978', 'Relative');
INSERT INTO NOMINEE_MSTR (NOMINEE_NO, ACCT_FD_NO, NAME, DOB, RELATIONSHIP)
VALUES('N14', 'FS5', 'Pramila P. Pius', '10-OCT-1985', 'Niece');

-- Record for ADDR_DTLS
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(1, 'B1', 'H', 'A/5, Jay Chambers,', 'Service Road, Vile Parle (East)',
'Mumbai', 'Maharashtra', '400057');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(2, 'B2', 'B', 'BSES Chambers, 10th floor,',
'Near Rly. Station, Andheri (West)', 'Mumbai', 'Maharashtra', '400058');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(3, 'B3', 'B', 'Prabhat Complex, No. 5 / 6,', 'Opp. Air India Bldg., Churchgate,',
'Mumbai', 'Maharashtra', '400004');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(4, 'B4', 'B', '23/A, Swarna Bldg., Smt. Rai Marg,',
'Eastern Express Highway, Kurla (East)', 'Mumbai', 'Maharashtra', '400045');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(5, 'B5', 'B', 'Vikas Centre, Shop 37, Near National Park,',
'Western Express Highway, Borivali (East)', 'Mumbai', 'Maharashtra', '400078');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(6, 'B6', 'B', '24/A, Mahima Plaza, First Floor,', 'Darya Ganj,',
'New Delhi', 'Delhi', '110004');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(7, 'E1', 'N', 'F-12, Diamond Palace, West Avenue,',
'North Avenue, Santacruz (West)', 'Mumbai', 'Maharashtra', '400056');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(8, 'E2', 'C', 'Desai House, Plot No. 25, P.G. Marg,',
'Near Malad Rly. Stat., Malad (West)', 'Mumbai', 'Maharashtra', '400078');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(9, 'E3', 'N', 'Room No. 56, 3rd Floor, Swamibhavan,',
'J. P. Road Junction, Andheri (East)', 'Mumbai', 'Maharashtra', '400059');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(10, 'E4', 'C', '301, Thomas Palace, Opp. Indu Child Care,',
'Yadnik Nagar, Andheri (West)', 'Mumbai', 'Maharashtra', '400058');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(11, 'E5', 'C', '456/A, Bldg. No. 4, Vahatuk Nagar,',
'Amboli, Andheri (West)', 'Mumbai', 'Maharashtra', '400058');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(12, 'E6', 'N', '201, Meena Towers, Nr. Sun Gas Agency,',
'S. V. Rd., Goregoan (West)', 'Mumbai', 'Maharashtra', '400076');
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(13, 'E7', 'N', 'Patel Chawl, Rm. No. 15, B. P. Lal Marg,',
'Mahim (West)', 'Mumbai', 'Maharashtra', '400016');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(14, 'E8', 'C', 'A - 10, Neelam, L. J. Road,', 'Mahim (East)',
'Mumbai', 'Maharashtra', '400016');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(15, 'E9', 'N', '1/12 Bal Govindas Society, M. B. Raut Rd.',
'Dadar (East)', 'Mumbai', 'Maharashtra', '400028');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(16, 'E10', 'C', 'Pathak Nagar, Cadal Road,', 'Mahim (West)', 'New Delhi',
'Delhi', '110016');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(17, 'C1', 'C', 'F-12, Diamond Palace, West Avenue,',
'North Avenue, Santacruz (West)', 'Mumbai', 'Maharashtra', '400056');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(18, 'C2', 'C', 'F-12, Silver Stream,', 'Santacruz (East)', 'Mumbai',
'Maharashtra', '400056');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(19, 'C3', 'C', 'Magesh Prasad,', 'Saraswati Baug, Jogeshwari(E)',
'Mumbai', 'Maharashtra', '400060');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(20, 'C4', 'C', '4, Sampada,', 'Kataria Road, Mahim,', 'Mumbai',
'Maharashtra', '400016');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(21, 'C5', 'C', '104, Vikram Apts. Bhagat Lane,', 'Shivaji Park, Mahim,',
'Mumbai', 'Maharashtra', '400016');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(22, 'C6', 'C', '12, Radha Kunj, N.C Kelkar Road,', 'Dadar,', 'Mumbai',
'Maharashtra', '400028');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(23, 'C7', 'C', 'A/14, Shanti Society, Mogal Lane,', 'Mahim,', 'Mumbai',
'Maharashtra', '400016');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(24, 'C8', 'C', '5, Vagdevi, Senapati Bapat Rd.', 'Dadar', 'Mumbai',
'Maharashtra', '400016');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(25, 'C9', 'C', 'A-10 Nutan Vaishali,', 'Shivaji Park, Mahim,', 'Mumbai',
'Maharashtra', '400016');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(26, 'C10', 'C', 'B-10, Makarand Society,', 'Cadal Road, Mahim,', 'Mumbai',
'Maharashtra', '400016');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(27, 'N1', 'C', '307/E, Meena Mansion,', 'R. S. Road, Andheri (West)',
'Mumbai', 'Maharashtra', '400058');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(28, 'N2', 'C', 'Smt. Veenu Chawl, Sawant Colony Rd.',
'Opp. Veer Road, Matunga (West)', 'Mumbai', 'Maharashtra', '400016');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
VALUES(29, 'N3', 'C', 'F-12, Silver Stream,', 'Santacruz (East)', 'Mumbai',
'Maharashtra', '400056');
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
VALUES(30, 'N4', 'C', 'Magesh Prasad', 'Saraswati Baug, Jogeshwari(E)',  
      'Mumbai', 'Maharashtra', '400060');  
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)  
VALUES(31, 'N5', 'C', 'Rita Apartment, Room No. 46, 2nd Floor',  
      'J. P. Road, Andheri (East)', 'Mumbai', 'Maharashtra', '400067');  
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)  
VALUES(32, 'N6', 'N', '106/A, Sunrise Apmt., Opp. Vahatuk Nagar',  
      'Kevni-Pada, Jogeshwari (West)', 'Mumbai', 'Maharashtra', '400102');  
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)  
VALUES(33, 'N7', 'C', 'Pathak Nagar, Cadal Road', 'Mahim (West)', 'Mumbai',  
      'Maharashtra', '400016');  
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)  
VALUES(34, 'O11', 'H', 'Shop No. 4, Simon Streams',  
      'V. P. Road, Andheri (West)', 'Mumbai', 'Maharashtra', '400058');  
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)  
VALUES(35, 'O12', 'H', '230-E, Patel Chambers', 'Service Road, Vile Parle (East)',  
      'Mumbai', 'Maharashtra', '400057');  
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)  
VALUES(36, 'O13', 'H', 'G-2, Puru Hsg. Society', 'Senapati Bapat Rd., Dadar',  
      'Mumbai', 'Maharashtra', '400016');  
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)  
VALUES(37, 'O14', 'H', 'B-10, Makarand Society', 'Cadal Road, Mahim',  
      'Mumbai', 'Maharashtra', '400016');  
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)  
VALUES(38, 'N8', 'N', '106/A, Sunrise Apmt., Opp. Vahatuk Nagar',  
      'Kevni-Pada, Jogeshwari (West)', 'Mumbai', 'Maharashtra', '400102');  
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)  
VALUES(39, 'N9', 'C', '307/E, Meena Mansion', 'R. S. Road, Andheri (West)',  
      'Mumbai', 'Maharashtra', '400058');  
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)  
VALUES(40, 'N10', 'C', 'Smt. Veenu Chawl, Sawant Colony Rd.',  
      'Opp. Veer Road, Matunga (West)', 'Mumbai', 'Maharashtra', '400016');  
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)  
VALUES(41, 'N11', 'C', 'F-12, Silver Stream', 'Santacruz (East)', 'Mumbai',  
      'Maharashtra', '400056');  
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)  
VALUES(42, 'N12', 'C', 'Magesh Prasad', 'Saraswati Baug, Jogeshwari(E)',  
      'Mumbai', 'Maharashtra', '400060');  
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)  
VALUES(43, 'N13', 'C', 'Pathak Nagar, Cadal Road', 'Mahim (West)', 'Mumbai',  
      'Maharashtra', '400016');  
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)  
VALUES(44, 'N14', 'C', '405, Vahatuk Nagar, Kevni-Pada', 'Jogeshwari (West)',  
      'Mumbai', 'Maharashtra', '400102');  
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)  
VALUES(45, 'C6', 'N', '203/A, Prachi Apmt.', 'Andheri (East)', 'Mumbai',  
      'Maharashtra', '400058');  
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)  
VALUES(46, 'O15', 'H', 'Shop No. 4, Sai Compound',  
      'Service Road, Vile Parle (East)', 'Mumbai', 'Maharashtra', '400057');  
INSERT INTO ADDR_DTLS (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)  
VALUES(47, 'O15', 'H', 'G-4, Sagar Chambers', 'G. P. Road, Andheri (West)',
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

'Mumbai', 'Maharashtra', '400058');

-- Record for CNTC_DTLS

```
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(1, 'B1', 'O', '26124571');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(1, 'B1', 'F', '26124533');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(1, 'B1', 'E',
'admin_vileparle@bom2.vsnl.in');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(2, 'B2', 'O', '26790014');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(2, 'B2', 'E',
'admin_andheri@bom2.vsnl.in');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(3, 'B3', 'O', '23457855');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(3, 'B3', 'E',
'admin_churchgate@bom2.vsnl.in');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(4, 'B4', 'O', '25545455');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(4, 'B4', 'E',
'admin_sion@bom2.vsnl.in');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(5, 'B5', 'O', '28175454');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(5, 'B5', 'E',
'admin_borivali@bom2.vsnl.in');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(6, 'B6', 'O', '24304545');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(6, 'B6', 'E',
'admin_matunga@bom2.vsnl.in');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(8, 'E2', 'R', '28883779');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(9, 'E3', 'R', '28377634');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(10, 'E4', 'R', '26323560');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(11, 'E5', 'R', '26793231');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(12, 'E6', 'R', '28085654');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(13, 'E7', 'R', '24442342');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(14, 'E8', 'R', '24365672');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(15, 'E9', 'R', '24327349');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(16, 'E10', 'R', '24302579');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(17, 'C1', 'R', '26405853');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(17, 'C1', 'O', '26134553');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(17, 'C1', 'O', '26134571');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(17, 'C1', 'M', '9820178955');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(18, 'C2', 'R', '26045754');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(18, 'C2', 'O', '26134571');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(19, 'C3', 'R', '28324567');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(19, 'C3', 'O', '26197654');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(20, 'C4', 'R', '24449852');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(20, 'C4', 'O', '28741370');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(21, 'C5', 'R', '24302934');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(21, 'C5', 'O', '22819964');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(22, 'C6', 'R', '24217592');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(23, 'C7', 'R', '24372247');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(24, 'C8', 'O', '26480903');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(25, 'C9', 'R', '24313408');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(25, 'C9', 'M', '9821176651');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(26, 'C10', 'R', '24362680');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(26, 'C10', 'O', '28973355');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(26, 'C10', 'M',
'9820484648');
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(27, 'N1', 'R', '26762154');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(28, 'N2', 'R', '24307887');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(29, 'N3', 'R', '260455754');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(30, 'N4', 'R', '28645489');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(31, 'N5', 'R', '30903564');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(32, 'N6', 'R', '26793771');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(33, 'N7', 'R', '24304455');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(34, 'O11', 'O', '26790055');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(34, 'O11', 'F', '26784409');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(35, 'O12', 'O', '26120455');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(35, 'O12', 'O', '26120456');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(35, 'O12', 'F', '26121450');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(35, 'O12', 'E',
'admin@sunpvtltd.com');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(35, 'O12', 'W',
'www.sunpvtltd.com');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(36, 'O13', 'O', '24301090');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(36, 'O13', 'O', '24301196');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(37, 'O14', 'O', '24321122');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(38, 'N8', 'R', '26793771');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(39, 'N9', 'R', '26762154');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(40, 'N10', 'R', '24307887');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(41, 'N11', 'R', '26045754');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(42, 'N12', 'R', '28645489');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(43, 'N13', 'R', '24304455');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(44, 'N14', 'R', '26790180');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(44, 'N14', 'R', '26771275');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(45, 'C6', 'R', '28274784');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(46, 'O15', 'O', '26170055');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(46, 'O15', 'F', '26174409');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(47, 'O16', 'O', '26790455');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(47, 'O16', 'F', '26781450');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(47, 'O16', 'E',
'admin@moonmltg.com');
INSERT INTO CNTC_DTLS (ADDR_NO, CODE_NO, CNTC_TYPE, CNTC_DATA) VALUES(47, 'O16', 'W',
'www.moonmltg.com');
```

-- Record for TRANS_MSTR

```
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T1', 'SB1', '05-NOV-2003', 'C', 'Initial Payment', 'D', 500, 500);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T2', 'CA2', '10-NOV-2003', 'C', 'Initial Payment', 'D', 2000, 2000);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T3', 'CA2', '13-NOV-2003', 'C', 'Self', 'D', 3000, 5000);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T4', 'SB3', '22-NOV-2003', 'C', 'Initial Payment', 'D', 500, 500);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T5', 'CA2', '10-DEC-2003', 'C', 'Self', 'W', 2000, 3000);

INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T6', 'CA4', '05-DEC-2003', 'B', 'Initial Payment', 'D', 2000, 2000);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
VALUES('T7', 'SB5', '15-DEC-2003', 'B', 'Initial Payment', 'D', 500, 500);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T8', 'SB6', '27-DEC-2003', 'C', 'Initial Payment', 'D', 500, 500);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T9', 'CA7', '14-JAN-2004', 'B', 'Initial Payment', 'D', 2000, 2000);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T10', 'SB8', '29-JAN-2004', 'C', 'Initial Payment', 'D', 500, 500);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T11', 'SB9', '05-FEB-2004', 'C', 'Initial Payment', 'D', 500, 500);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T12', 'SB9', '15-FEB-2004', 'B', 'CLR-204907', 'D', 3000, 3500);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T13', 'SB9', '17-FEB-2004', 'C', 'Self', 'W', 2500, 1000);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T14', 'CA10', '19-FEB-2004', 'B', 'Initial Payment', 'D', 2000, 2000);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T15', 'SB9', '05-APR-2004', 'B', 'CLR-204908', 'D', 3000, 4000);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T16', 'SB9', '27-APR-2004', 'C', 'Self', 'W', 2500, 1500);
```

```
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T17', 'SB1', '05-NOV-2003', 'C', 'Initial Payment', 'D', 500, 500);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T18', 'CA2', '10-NOV-2003', 'C', 'Initial Payment', 'D', 2000, 2000);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T19', 'SB3', '22-NOV-2003', 'C', 'Initial Payment', 'D', 500, 500);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T20', 'CA4', '05-DEC-2003', 'B', 'Initial Payment', 'D', 2000, 2000);
INSERT INTO TRANS_MSTR (TRANS_NO, ACCT_NO, DT, TYPE, PARTICULAR, DR_CR, AMT, BALANCE)
VALUES('T21', 'SB5', '15-DEC-2003', 'B', 'Initial Payment', 'D', 500, 500);
```

-- Record for TRANS_DTLS

```
INSERT INTO TRANS_DTLS (TRANS_NO, INST_NO, INST_DT, PAYTO, INST_CLR_DT, BANK_NAME,
BRANCH_NAME, PAIDFROM)
VALUES('T6', 098324, '02-DEC-2003', 'Self', '05-DEC-2003', 'HDFC', 'Vile Parle (East)', '2982');
INSERT INTO TRANS_DTLS (TRANS_NO, INST_NO, INST_DT, PAYTO, INST_CLR_DT, BANK_NAME,
BRANCH_NAME, PAIDFROM)
VALUES('T7', 232324, '14-DEC-2003', 'Self', '15-DEC-2003', 'India Bank', 'Andheri (West)', '30434');
INSERT INTO TRANS_DTLS (TRANS_NO, INST_NO, INST_DT, PAYTO, INST_CLR_DT, BANK_NAME,
BRANCH_NAME, PAIDFROM)
VALUES('T9', 434560, '14-JAN-2004', 'Self', '14-JAN-2004', 'ICICI Bank', 'Bandra (West)', '4882');
INSERT INTO TRANS_DTLS (TRANS_NO, INST_NO, INST_DT, PAYTO, INST_CLR_DT, BANK_NAME,
BRANCH_NAME, PAIDFROM)
VALUES('T12', 204907, '14-FEB-2004', 'Self', '15-FEB-2004', 'Memon Co-operative Bank', 'Jogeshwari (West)',
'1767');
INSERT INTO TRANS_DTLS (TRANS_NO, INST_NO, INST_DT, PAYTO, INST_CLR_DT, BANK_NAME,
BRANCH_NAME, PAIDFROM)
VALUES('T14', 100907, '19-FEB-2004', 'Self', '19-FEB-2004', 'Memon Co-operative Bank', 'Jogeshwari (West)',
'2001');
INSERT INTO TRANS_DTLS (TRANS_NO, INST_NO, INST_DT, PAYTO, INST_CLR_DT, BANK_NAME,
BRANCH_NAME, PAIDFROM)
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
VALUES('T15', 204908, '01-APR-2004', 'Self', '05-APR-2004', 'Memon Co-operative Bank', 'Jogeshwari (West)', '1767');
```

```
INSERT INTO TRANS_DTLS (TRANS_NO, INST_NO, INST_DT, PAYTO, INST_CLR_DT, BANK_NAME, BRANCH_NAME, PAIDFROM)
```

```
VALUES('T20', 098324, '02-DEC-2003', 'Self', '05-DEC-2003', 'HDFC', 'Vile Parle (East)', '2982');
```

```
INSERT INTO TRANS_DTLS (TRANS_NO, INST_NO, INST_DT, PAYTO, INST_CLR_DT, BANK_NAME, BRANCH_NAME, PAIDFROM)
```

```
VALUES('T21', 232324, '14-DEC-2003', 'Self', '15-DEC-2003', 'India Bank', 'Andheri (West)', '30434');
```

```
COMMIT;
```

Example 1:

```
SELECT FD_NO, TYPE, PERIOD, OPNDT, DUEDT, AMT, INTRATE, DUEAMT,  
       ROUND(AMT + (AMT * ROUND(SYSDATE - OPNDT)/365 * (INTRATE/100)), 2)  
FROM FD_DTLS WHERE DUEDT > SYSDATE;
```

Example 2:

```
SELECT Fd_No, Type, Period, OpnDt, DueDt, Amt, IntRate, DueAmt,  
       ROUND(Amt + (Amt * ROUND(SysDate - OpnDt)/365 * (IntRate/100)), 2) "Pre Maturity Amount"  
FROM Fd_Dtls WHERE DueDt > SysDate;
```

Example 3:

```
SELECT * FROM Trans_Mstr WHERE Amt >= 500 AND Amt <= 5000 AND Amt <= 5000  
AND TO_CHAR(Dt) = TO_CHAR(SysDate);
```

Example 4:

```
SELECT Cust_no, FName || ' ' || MName || ' ' || LName "Customers"  
FROM Cust_Mstr, Addr_Dtls  
WHERE Cust_Mstr.Cust_No = Addr_Dtls.Code_No  
AND (Occup = 'Information Technology' OR Occup = 'Self Employed')  
AND Cust_No LIKE 'C%';
```

Example 5:

```
SELECT Cust_No, FName || ' ' || MName || ' ' || LName "Customers",  
       ROUND((SYSDATE - DOB_Inc)/365) "Age"  
FROM Cust_Mstr  
WHERE (ROUND((SYSDATE - DOB_Inc)/365) < 25 AND LName='Bayross')  
OR (ROUND((SYSDATE - DOB_Inc)/365) > 25  
AND ROUND((SYSDATE - DOB_Inc)/365) < 55) AND Cust_No LIKE 'C%';
```

Example 6:

```
SELECT Acct_No, Type, Opr_Mode, OpnDt, CurBal, Status  
FROM Acct_Mstr WHERE NOT (Opr_Mode = 'SI' OR Opr_Mode = 'JO');
```

Example 7:

```
SELECT * FROM Trans_Mstr WHERE TO_CHAR(DT, 'MM') BETWEEN 01 AND 03;  
SELECT * FROM Trans_Mstr  
WHERE TO_CHAR(DT, 'MM') >= 01 AND TO_CHAR(DT, 'MM') <= 03;
```

Example 8:

```
SELECT DISTINCT Acct_No FROM Trans_Mstr
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

WHERE TO_CHAR(DT, 'MM') NOT BETWEEN 01 AND 04;

Example 9:

```
SELECT Fname, Lname, DOB_INC "Birthdate", Occup FROM Cust_Mstr
WHERE Fname LIKE 'Ch%';
```

Example 10:

```
SELECT Fname, Lname, DOB_INC "Birthdate", Occup FROM Cust_Mstr
WHERE Fname LIKE '_a%' OR Fname LIKE '_s%';
```

Example 11:

```
SELECT Fname, Lname, DOB_INC "Birthdate", Occup FROM Cust_Mstr
WHERE Fname LIKE 'Iv__';
```

Example 12:

```
SELECT Fname, Lname, DOB_INC "Birthdate", Occup FROM Cust_Mstr
WHERE Fname IN('Hansel', 'Mamta', 'Namita', 'Aruna');
```

Example 13:

```
SELECT Fname, Lname, DOB_INC "Birthdate", Occup FROM Cust_Mstr
WHERE Fname NOT IN('Hansel', 'Mamta', 'Namita', 'Aruna');
```

Example 14:

```
DESC DUAL
SELECT * FROM DUAL;
SELECT 2*2 FROM DUAL;
```

Example 15:

```
SELECT SYSDATE FROM DUAL;
```

Example 16:

```
SELECT AVG(CurBal) "Average Balance" FROM Acct_Mstr;
```

Example 17:

```
SELECT MIN(CurBal) "Minimum Balance" FROM Acct_Mstr;
```

Example 18:

```
SELECT COUNT(Acct_No) "No. of Accounts" FROM Acct_Mstr;
```

Example 19:

```
SELECT MAX(CurBal) "Maximum Balance" FROM Acct_Mstr;
```

Example 20:

```
SELECT SUM(CurBal) "Total Balance" FROM Acct_Mstr;
```

Example 21:

```
SELECT ABS(-15) "Absolute" FROM DUAL;
```

Example 22:

```
SELECT POWER(3,2) "Raised" FROM DUAL;
```

Example 23:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
SELECT ROUND(15.19,1) "Round" FROM DUAL;
```

Example 24:

```
SELECT SQRT(25) "Square Root" FROM DUAL;
```

```
SELECT EXP(5) "Exponent" FROM DUAL;
```

```
SELECT LOG(25, 4) "Log" FROM DUAL;
```

```
SELECT LN(4) "Log" FROM DUAL;
```

```
SELECT VARIANCE(Char_Length) "Variance" FROM ALL_TAB_COLUMNS;
```

```
SELECT EXTRACT(YEAR FROM DATE '2004-07-02') "Year",  
       EXTRACT(MONTH FROM SYSDATE) "Month" FROM DUAL;
```

```
SELECT GREATEST(4, 5, 17) "Num", GREATEST('4', '5', '17') "Text" FROM DUAL;
```

```
SELECT LEAST(4, 5, 17) "Num", LEAST('4', '5', '17') "Text" FROM DUAL;
```

```
SELECT MOD(15, 7) "Mod1", MOD(15.7, 7) "Mod2" FROM DUAL;
```

```
SELECT TRUNC(125.815, 1) "Trunc1", TRUNC(125.815, -2) "Trunc2" FROM DUAL;
```

```
SELECT FLOOR(24.8) "Flr1", FLOOR(13.15) "Flr2" FROM DUAL;
```

```
SELECT CEIL(24.8) "CeilFlr1", CEIL(13.15) "Ceil2" FROM DUAL;
```

Example 25:

```
SELECT LOWER('IVAN BAYROSS') "Lower" FROM DUAL;
```

Example 26:

```
SELECT INITCAP('IVAN BAYROSS') "Title Case" FROM DUAL;
```

Example 27:

```
SELECT UPPER('Ms. Carol') "Capitalised" FROM DUAL;
```

```
SELECT ASCII('a') "ASCII1", ASCII('Aa') "ASCII2" FROM DUAL;
```

```
SELECT COMPOSE('a' || UNISTR('\0301')) "Composed" FROM DUAL;
```

```
SELECT DECOMPOSE(COMPOSE('a' || UNISTR('\0301'))) "Decomposed" FROM DUAL;
```

```
SELECT DUMP('SCT') "Dump1", DUMP('SCT', 1017) "Dump2" FROM DUAL;
```

```
SELECT INSTR('SCT on the net', 't') "Instr1", INSTR('SCT on the net', 't', 1, 2) "Instr2"  
FROM DUAL;
```

```
SELECT SOUNDEX('SCT on the net') "Sound" FROM DUAL;
```

```
SELECT TRANSLATE('1sct523', '123', '7a9') "Change" FROM DUAL;
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
SELECT SUBSTR('This is a test', 6, 2) "Extracted" FROM DUAL;

SELECT LENGTH('SHARANAM') "Length" FROM DUAL;

SELECT LTRIM('NISHA','N') "Left" FROM DUAL;

SELECT RTRIM('SUNILA','A') "RTRIM" FROM DUAL;

SELECT TRIM(' Hansel ') "Trim both sides" FROM DUAL;
SELECT TRIM(LEADING 'x' FROM 'xxxHanselxxx') "Remove prefixes" FROM DUAL;
SELECT TRIM(BOTH 'x' FROM 'xxxHanselxxx') "Remove prefixes N suffixes" FROM DUAL;
SELECT TRIM(BOTH '1' FROM '123Hansel12111') "Remove string" FROM DUAL;

SELECT LPAD('Page 1',10,*) "Lpad" FROM DUAL;

SELECT RPAD(Fname,10,'x') "RPAD Example" FROM Cust_Mstr
      WHERE Fname = 'Ivan';

SELECT VSIZE('SCT on the net') "Size" FROM DUAL;

UPDATE Acct_Mstr SET CurBal = CurBal + TO_NUMBER(SUBSTR('$100',2,3));

SELECT TO_CHAR(17145, '$099,999') "Char" FROM DUAL;

SELECT TO_CHAR(DT, 'Month DD, YYYY') "New Date Format" FROM Trans_Mstr
      WHERE Trans_No = 'T1';

INSERT INTO CUST_MSTR(CUST_NO, FNAME, MNAME, LNAME, DOB_INC)
      VALUES('C1', 'Ivan', 'Nelson', 'Bayross',
      TO_DATE('25-JUN-1952 10:55 A.M.', 'DD-MON-YY HH:MI A.M.));

SELECT ADD_MONTHS(SYSDATE, 4) FROM DUAL;

SELECT SYSDATE, LAST_DAY(SYSDATE) "LastDay" FROM DUAL;

SELECT MONTHS_BETWEEN('02-FEB-92', '02-JAN-92') "Months" FROM DUAL;

SELECT NEXT_DAY('06-JULY-02', 'Saturday') "NEXT DAY" FROM DUAL;

SELECT TRUNC(To_Date('01-JUL-04'), 'YEAR') "Year" FROM DUAL;

SELECT ROUND(TO_DATE('01-JUL-04'), 'YEAR') "Year" FROM DUAL;

SELECT NEW_TIME(TO_DATE('2004/07/01 01:45', 'yyyy/mm/dd HH24:MI'), 'AST', 'MST') "MST"
      FROM DUAL;

SELECT TO_CHAR(SYSDATE, 'DD-MM-YY') FROM DUAL;

SELECT TO_DATE ('06/07/02', 'DD/MM/YY') FROM DUAL;

SELECT Trans_No, Acct_No, TO_CHAR(DT, 'DD/MM/YY') "Transaction Date",
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
Particular, DR_CR, Amt, Balance  
FROM Trans_Mstr WHERE Acct_No = 'SB9' ORDER BY TO_CHAR(DT, 'MM');
```

```
INSERT INTO Cust_Mstr (Cust_No, Fname, Lname, Dob_Inc)  
VALUES('C100', 'Sharanam', 'Shah',  
TO_DATE('03/Jan/1981 12:23:00', 'DD/MON/YY hh:mi:ss'));
```

```
SELECT Cust_No, Fname, Lname, Dob_Inc FROM Cust_Mstr WHERE Cust_No LIKE 'C_';
```

```
SELECT Cust_No, Fname, Lname, TO_CHAR(DOB_Inc, 'DDTH-MON-YY') "DOB_DDTH"  
FROM Cust_Mstr WHERE Cust_No LIKE 'C_';
```

```
SELECT Cust_No, Fname, Lname, TO_CHAR(Dob_Inc, 'DDSP') "DOB_DDSP"  
FROM Cust_Mstr WHERE Cust_No LIKE 'C_';
```

```
SELECT Cust_No, Fname, Lname, TO_CHAR(Dob_Inc, 'DDSPTH') "DOB_DDSPTH"  
FROM Cust_Mstr WHERE Cust_No LIKE 'C_';
```

```
SELECT UID FROM DUAL;
```

```
SELECT USER FROM DUAL;
```

```
SELECT SYS_CONTEXT('USERENV', 'NLS_DATE_FORMAT') "SysContext" FROM DUAL;
```

```
SELECT USERENV('LANGUAGE') FROM DUAL;
```

```
SELECT COALESCE(ADDR1, ADDR2, CITY) Addr FROM ADDR_DTLS;
```

Example 1:

```
SELECT BRANCH_NO "Branch No.", COUNT(EMP_NO) "No. Of Employees"  
FROM EMP_MSTR GROUP BY BRANCH_NO;
```

Example 2:

```
SELECT VERI_EMP_NO "Emp. No.", COUNT(ACCT_NO) "No. Of A/Cs Verified"  
FROM ACCT_MSTR GROUP BY VERI_EMP_NO;
```

Example 3:

```
SELECT BRANCH_NO "Branch No.", TYPE "A/C Type", COUNT(ACCT_NO) "No. Of A/Cs"  
FROM ACCT_MSTR GROUP BY BRANCH_NO, TYPE;
```

Example 4:

```
SELECT CUST_NO, COUNT(ACCT_FD_NO) "No. Of A/Cs Held" FROM ACCT_FD_CUST_DTLS  
WHERE ACCT_FD_NO LIKE 'CA%' OR ACCT_FD_NO LIKE 'SB%'  
GROUP BY CUST_NO HAVING COUNT(ACCT_FD_NO)>1;
```

Example 5:

```
SELECT BRANCH_NO, COUNT(ACCT_NO) "No. Of A/Cs Activated"  
FROM ACCT_MSTR WHERE TO_CHAR(OPNDT, 'DD-MM-YYYY') > '03-01-2003'  
GROUP BY BRANCH_NO HAVING COUNT(ACCT_NO) > 1;
```

Example 6:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
SELECT CUST_NO, COUNT(ACCT_FD_NO) "No. Of A/Cs Or FDs Held"  
FROM ACCT_FD_CUST_DTLS GROUP BY CUST_NO HAVING COUNT(ACCT_FD_NO) = 1;
```

Example 7:

```
SELECT CUST_NO, COUNT(ACCT_FD_NO) "No. Of A/Cs or FDs Held"  
FROM ACCT_FD_CUST_DTLS GROUP BY CUST_NO HAVING COUNT(ACCT_FD_NO) > 1;
```

Example 8:

```
SELECT FD_SER_NO, FD_NO, SUM(AMT), SUM(DUEAMT)  
FROM FD_DTLS  
GROUP BY ROLLUP (FD_SER_NO, FD_NO);
```

Example 9:

```
SELECT BRANCH_NO, ACCT_NO, SUM(CURBAL) FROM ACCT_MSTR  
GROUP BY CUBE (BRANCH_NO, ACCT_NO);
```

Example 10:

```
SELECT CODE_NO "Cust. No.", ADDR1 || ' ' || ADDR2 || ' ' || CITY || ', ' || STATE || ', ' || PINCODE "Address"  
FROM ADDR_DTLS WHERE CODE_NO IN(SELECT CUST_NO FROM CUST_MSTR  
WHERE FNAME = 'Ivan' AND LNAME = 'Bayross');  
SELECT CUST_NO FROM CUST_MSTR WHERE FNAME = 'IVAN' AND LNAME = 'BAYROSS';  
SELECT CODE_NO "Cust. No.", ADDR1 || ' ' || ADDR2 || ' ' || CITY || ', ' || STATE || ', ' || PINCODE "Address"  
FROM ADDR_DTLS WHERE CODE_NO IN('C1');
```

Example 11:

```
SELECT (FNAME || ' ' || LNAME) "Customer" FROM CUST_MSTR  
WHERE CUST_NO IN(SELECT CODE_NO FROM ADDR_DTLS  
WHERE CODE_NO LIKE 'C%' AND PINCODE NOT IN(SELECT PINCODE  
FROM ADDR_DTLS WHERE CODE_NO LIKE 'B%'));  
SELECT PINCODE FROM ADDR_DTLS WHERE CODE_NO LIKE 'B%';  
SELECT (FNAME || ' ' || LNAME) "Customer" FROM CUST_MSTR  
WHERE CUST_NO IN(SELECT CODE_NO FROM ADDR_DTLS  
WHERE CODE_NO LIKE 'C%'  
AND PINCODE NOT IN('400057', '400058', '400004', '400045', '400078', '110004'));  
SELECT CODE_NO FROM ADDR_DTLS WHERE CODE_NO LIKE 'C%'  
AND PINCODE NOT IN('400057', '400058', '400004', '400045', '400078', '110004');  
SELECT (FName || ' ' || LName) "Customer" FROM Cust_Mstr  
WHERE Cust_No IN('C1', 'C2', 'C3', 'C4', 'C5', 'C6', 'C7', 'C8', 'C9', 'C10');
```

Example 12:

```
SELECT (FNAME || ' ' || LNAME) "Customer" FROM CUST_MSTR  
WHERE CUST_NO IN(SELECT CUST_NO FROM ACCT_FD_CUST_DTLS  
WHERE ACCT_FD_NO IN(SELECT FD_SER_NO FROM FD_DTLS WHERE AMT > 5000));  
SELECT FD_SER_NO FROM FD_DTLS WHERE AMT > 5000;  
SELECT (FNAME || ' ' || LNAME) "Customer" FROM CUST_MSTR  
WHERE CUST_NO IN(SELECT CUST_NO FROM ACCT_FD_CUST_DTLS  
WHERE ACCT_FD_NO IN('FS1', 'FS2', 'FS2', 'FS5'));  
SELECT CUST_NO FROM ACCT_FD_CUST_DTLS  
WHERE ACCT_FD_NO IN('FS1', 'FS2', 'FS2', 'FS5');  
SELECT (FNAME || ' ' || LNAME) "Customer" FROM CUST_MSTR  
WHERE CUST_NO IN('C2', 'C3', 'C4', 'C5', 'C5', 'C5');
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Example 13:

Example 14:

```
SELECT LENGTH(City), COUNT(Addr_Dtls.Code_No) "No. Of Customers" FROM Addr_Dtls
    WHERE Code_No LIKE 'C%' GROUP BY Addr_Dtls.City;
SELECT Addr_Dtls.City "Len", COUNT(Addr_Dtls.Code_No) "No. Of Customers"
    FROM Addr_Dtls WHERE Code_No LIKE 'C%' GROUP BY Len;
SELECT Addr_Dtls.City, COUNT(Addr_Dtls.Code_No) "No. Of Customers"
    FROM Addr_Dtls WHERE Code_No LIKE 'C%' GROUP BY 1;
```

Example :

```
SELECT Branch_No, Acct_No, SUM(CurBal) FROM Acct_Mstr
    WHERE Type = 'CA' AND Corp_Cust_No IS NOT NULL
    GROUP BY ROLLUP (Branch_No, Acct_No);
```

Example :

```
SELECT Branch_No, Acct_No, SUM(CurBal) FROM Acct_Mstr
    WHERE Type = 'CA' AND Corp_Cust_No IS NOT NULL
    GROUP BY CUBE (Branch_No, Acct_No);
```

Example :

```
SELECT Code_No "Cust. No.", Addr1 || ' ' || Addr2 || ' ' || City || ', ' || State
    || ', ' || Pincode "Address"
    FROM Addr_Dtls WHERE Code_No IN(SELECT Cust_No FROM Cust_Mstr
    WHERE FName = 'Ivan' AND LName = 'Bayross');
```

Example :

```
SELECT A.Acct_No, A.CurBal, A.Branch_No, B.AvgBal
    FROM Acct_Mstr A, (SELECT Branch_No, AVG(CurBal) AvgBal FROM Acct_Mstr
    GROUP BY Branch_No) B
    WHERE A.Branch_No = B.Branch_No AND A.CurBal > B.AvgBal;
```

Example :

```
SELECT Acct_No, CurBal, Branch_No FROM Acct_Mstr A
    WHERE CurBal > (SELECT AVG(CurBal) FROM Acct_Mstr
    WHERE Branch_No = A.Branch_No);
```

Example :

```
SELECT FName, LName FROM Cust_Mstr
    WHERE (FName, LName) IN(SELECT FName, LName FROM EMP_MSTR);
```

Example :

```
SELECT Emp_No, (FName || ' ' || LName) "Name", Dept FROM Emp_Mstr E
    ORDER BY (SELECT Name FROM Branch_Mstr B WHERE E.Branch_No = B.Branch_no);
```

Example :

```
SELECT Emp_No, FName, LName FROM Emp_Mstr E
    WHERE EXISTS(SELECT 'SCT' FROM Acct_Mstr WHERE Veri_Emp_No = E.Emp_No);
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Example :

```
SELECT Branch_No, Name FROM Branch_Mstr B
WHERE NOT EXISTS(SELECT 'SCT' FROM Emp_Mstr
WHERE Branch_No = B.Branch_No);
```

Example :

```
SELECT E.Emp_No, (E.FName || ' ' || E.MName || ' ' || E.LName) "Name", B.Name "Branch", E.Dept, E.Desig
FROM Emp_Mstr E INNER JOIN Branch_Mstr B ON B.Branch_No = E.Branch_No;
SELECT E.Emp_No, (E.FName || ' ' || E.MName || ' ' || E.LName) "Name", B.Name "Branch", E.Dept, E.Desig
FROM Emp_Mstr E, Branch_Mstr B WHERE B.Branch_No = E.Branch_No;
```

Example :

```
SELECT (E.FName || ' ' || E.LName) "Name", E.Dept, C.Cntc_Type, C.Cntc_Data
FROM Emp_Mstr E LEFT JOIN Cntc_Dtls C ON E.Emp_No = C.Code_No;
SELECT (E.FName || ' ' || E.LName) "Name", E.Dept, C.Cntc_Type, C.Cntc_Data
FROM Emp_Mstr E, Cntc_Dtls C WHERE E.Emp_No = C.Code_No(+);
```

Example :

```
SELECT Emp.Fname "Employee", Mngr.Fname "Manager"
FROM Emp_Mstr Emp, Emp_Mstr Mngr
WHERE Emp.Mngr_No = Mngr.Emp_No;
```

Example :

```
SELECT First.Intro_Cust_No "Cust. No.",
(SELECT Fname || ' ' || Lname FROM Cust_Mstr
WHERE Cust_No = First.Intro_Cust_No) "Customer", First.Acct_No
FROM Acct_Mstr First, Acct_Mstr Second
WHERE First.Intro_Cust_No = Second.Intro_Cust_No
AND First.Acct_no <> Second.Acct_no;
```

Example :

```
SELECT 'Account No. ' || Acct_No || ' was introduced by Customer No. '
|| Intro_Cust_No || ' At Branch No. ' || Branch_No FROM Acct_Mstr;
SELECT 'Account No. ' || Acct_No || ' was introduced by Customer No. '
|| Intro_Cust_No || ' At Branch No. ' || Branch_No "Accounts Opened"
FROM Acct_Mstr;
```

Example :

```
SELECT Cust_No "ID", Fname || ' ' || Lname "Customer / Employees"
FROM Cust_Mstr, Addr_Dtls
WHERE Cust_Mstr.Cust_No = Addr_Dtls.Code_No
AND Addr_Dtls.City = 'Mumbai' AND Addr_Dtls.Code_No LIKE 'C%'

UNION

SELECT Emp_No "ID", Fname || ' ' || Lname "Customer / Employees"
FROM Emp_Mstr, Addr_Dtls
WHERE Emp_Mstr.Emp_No = Addr_Dtls.Code_No
AND Addr_Dtls.City = 'Mumbai' AND Addr_Dtls.Code_No LIKE 'E%';
SELECT Cust_No "ID", Fname || ' ' || Lname "Customer / Employees"
FROM Cust_Mstr, Addr_Dtls
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
WHERE Cust_Mstr.Cust_No = Addr_Dtls.Code_No
      AND Addr_Dtls.City = 'Mumbai' AND Addr_Dtls.Code_No LIKE 'C%';
SELECT Emp_No "ID", Fname || ' ' || Lname " Customer / Employees"
FROM Emp_Mstr, Addr_Dtls
      WHERE Emp_Mstr.Emp_No = Addr_Dtls.Code_No
      AND Addr_Dtls.City = 'Mumbai' AND Addr_Dtls.Code_No LIKE 'E%';
```

Example :

```
SELECT DISTINCT Cust_No FROM Acct_FD_Cust_Dtls
      WHERE Acct_FD_No LIKE 'CA%' OR Acct_FD_No LIKE 'SB%'
INTERSECT
SELECT DISTINCT Cust_No FROM Acct_FD_Cust_Dtls
      WHERE Acct_FD_No LIKE 'FS%';
SELECT DISTINCT Cust_No FROM Acct_FD_Cust_Dtls
      WHERE Acct_FD_No LIKE 'CA%' OR Acct_FD_No LIKE 'SB%';
SELECT DISTINCT Cust_No FROM Acct_FD_Cust_Dtls
      WHERE Acct_FD_No LIKE 'FS%';
```

Example :

```
SELECT DISTINCT Cust_No FROM Acct_FD_Cust_Dtls
      WHERE Acct_FD_No LIKE 'CA%' OR Acct_FD_No LIKE 'SB%'
MINUS
SELECT DISTINCT Cust_No FROM Acct_FD_Cust_Dtls
      WHERE Acct_FD_No LIKE 'FS%';
SELECT DISTINCT Cust_No FROM Acct_FD_Cust_Dtls
      WHERE Acct_FD_No LIKE 'CA%' OR Acct_FD_No LIKE 'SB%';
SELECT DISTINCT Cust_No FROM Acct_FD_Cust_Dtls
      WHERE Acct_FD_No LIKE 'FS%';
```

Address Field In The Index

```
SELECT ROWID, ACCT_NO FROM ACCT_MSTR;
```

Example 1:

```
SELECT ACCT_NO, OPNDT, VERI_EMP_NO FROM ACCT_MSTR WHERE VERI_EMP_NO = 'E1';
```

Example 2:

```
SELECT ACCT_NO, OPNDT, VERI_EMP_NO FROM ACCT_MSTR WHERE VERI_EMP_NO = 'E1';
```

Example 3:

```
CREATE INDEX idxVeriEmpNo ON ACCT_MSTR (VERI_EMP_NO);
```

Example 4:

```
CREATE INDEX idxTransAcctNo ON TRANS_MSTR (TRANS_NO, ACCT_NO);
```

Example 5:

```
CREATE UNIQUE INDEX idx_CustNo ON CUST_MSTR (CUST_NO);
```

Example 6:

```
CREATE INDEX idx_CustNo ON CUST_MSTR (CUST_NO) REVERSE;
```

Example 7:

```
ALTER INDEX idx_CustNo REBUILD NOREVERSE;
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Example 8:

```
CREATE BITMAP INDEX bitidx_TransNo ON TRANS_DTLS (TRANS_NO);
```

Example 9:

```
CREATE INDEX idx_Name ON CUST_MSTR (UPPER(FNAME));
```

Example 10:

```
DROP INDEX idx_CustNo;
```

Example 11:

```
DELETE FROM EMP_MSTR WHERE ROWID NOT IN(SELECT MIN(ROWID)
    FROM EMP_MSTR GROUP BY EMP_NO, FNAME, DEPT);
```

```
DELETE FROM EMP_MSTR WHERE ROWID NOT IN('AAAHebAABAAAMVqAAA',
    'AAAHebAABAAAMVqAAB', 'AAAHebAABAAAMVqAAC', 'AAAHebAABAAAMVqAAD',
    'AAAHebAABAAAMVqAAE', 'AAAHebAABAAAMVqAAF', 'AAAHebAABAAAMVqAAG',
    'AAAHebAABAAAMVqAAH', 'AAAHebAABAAAMVqAAI', 'AAAHebAABAAAMVqAAJ');
```

```
SELECT EMP_NO, FNAME, DEPT FROM EMP_MSTR;
```

Example 12:

```
SELECT ROWNUM, BRANCH_NO, NAME FROM BRANCH_MSTR WHERE ROWNUM < 4;
```

Example 13:

```
CREATE VIEW vw_Customers AS SELECT * FROM CUST_MSTR;
```

Example 14:

```
CREATE VIEW vw_Employees AS SELECT FNAME, MNAME, LNAME, DEPT
    FROM EMP_MSTR;
```

Example 15:

```
CREATE VIEW vw_Transactions AS
    SELECT ACCT_NO "Account No.", DT "Date", Type, DR_CR "Mode", AMT "Amount"
    FROM TRANS_MSTR;
```

Example 15:

```
SELECT FNAME, LNAME, DEPT FROM vw_Employees
    WHERE DEPT IN('Marketing', 'Loans And Financing');
```

Example 16:

```
CREATE VIEW vw_Nominees AS
    SELECT NOMINEE_NO, ACCT_FD_NO, NAME FROM NOMINEE_MSTR;
```

When an INSERT operation is performed using the view:

```
INSERT INTO vw_Nominees VALUES('N100', 'SB432', 'Sharanam');
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Example 17:

```
CREATE VIEW vw_Branch AS
    SELECT BRANCH_NO, NAME, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE
    FROM BRANCH_MSTR, ADDR_DTLS
    WHERE ADDR_DTLS.CODE_NO = BRANCH_MSTR.BRANCH_NO;
```

When an INSERT operation is performed using the view

```
INSERT INTO vw_Branch VALUES('B7', 'Dahisar', 'B', 'Vertex Plaza, Shop 4,', 'Western Express Highway, Dahisar (East)',
'Mumbai', 'Maharashtra', '400078');
```

Example 18:

```
DROP VIEW vw_Branch;
```

Example 19:

```
CREATE CLUSTER "DBA_BANKSYS"."BRANCH_INFO"("BRANCH_NO" VARCHAR2(10));
```

```
CREATE TABLE "DBA_BANKSYS"."BRANCH_MSTR"(
    "BRANCH_NO" VARCHAR2(10) PRIMARY KEY, "NAME" VARCHAR2(25))
    CLUSTER BRANCH_INFO(BRANCH_NO);
```

```
CREATE TABLE "DBA_BANKSYS"."ADDR_DTLS"(
    "ADDR_NO" NUMBER(6) PRIMARY KEY, "CODE_NO" VARCHAR2(10),
    "ADDR_TYPE" VARCHAR2(1), "ADDR1" VARCHAR2(50),
    "ADDR2" VARCHAR2(50), "CITY" VARCHAR2(25),
    "STATE" VARCHAR2(25), "PINCODE" VARCHAR2(6));
    CLUSTER BRANCH_INFO(BRANCH_NO);
```

Example 20:

```
CREATE SEQUENCE ADDR_SEQ INCREMENT BY 1 START WITH 1
    MINVALUE 1 MAXVALUE 999 CYCLE;
```

Example 21:

```
INSERT INTO ADDR_DTLS
    (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
    VALUES(ADDR_SEQ.NextVal, 'B5', 'B', 'Vertex Plaza, Shop 4,', 'Western Express Highway, Dahisar (East)',
'Mumbai', 'Maharashtra', '400078');
```

Example 22:

```
INSERT INTO ADDR_DTLS
    (ADDR_NO, CODE_NO, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE)
    VALUES(TO_CHAR(SYSDATE, 'MMYY') || TO_CHAR(ADDR_SEQ.NextVal), 'B5', 'B', 'Vertex Plaza, Shop 4,',
'Western Express Highway, Dahisar (East)', 'Mumbai', 'Maharashtra', '400078');
```

Example 23:

```
ALTER SEQUENCE ADDR_SEQ INCREMENT BY 2 CACHE 30;
```

Example 24:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

DROP SEQUENCE ADDR_SEQ;

Example 25:

```
CREATE SNAPSHOT NEW_EMP
  PCTFREE 10 PCTUSED 70
  TABLESPACE System
  STORAGE (INITIAL 50K NEXT 50K PCTINCREASE 0)
  REFRESH
    START WITH ROUND(SYSDATE + 7) + 2/24
    NEXT NEXT_DATE(TRUNC(SYSDATE, 'MONDAY') + 2/24
  AS SELECT * FROM EMP_MSTR;
```

Example 26:

```
DROP SNAPSHOT New_Client
```

12. SECURITY MANAGEMENT USING SQL GRANTING AND REVOKING PERMISSIONS

Example 1:

```
GRANT ALL ON EMP_MSTR TO Sharanam;
```

Example 2:

```
GRANT SELECT, UPDATE ON CUST_MSTR TO Hansel;
```

Example 3:

```
GRANT ALL ON ACCT_MSTR TO Ivan WITH GRANT OPTION;
```

Example 4:

```
SELECT * FROM Sharanam.FD_MSTR;
```

Example 5:

```
GRANT SELECT ON Vaishali.TRANS_MSTR TO Chhaya;
```

Example 6:

```
REVOKE DELETE ON NOMINEE_MSTR FROM Anil;
```

Example 7:

```
REVOKE ALL ON NOMINEE_MSTR FROM Anil;
```

Example 8:

```
REVOKE SELECT ON Alex.FDSLAB_MSTR FROM Rocky;
```

13. OOPS IN ORACLE ORACLE 9i DATABASE FLAVOURS

Example 1:

```
CREATE TYPE ADDRESS_TY AS OBJECT(
  STREET VARCHAR2(50), CITY VARCHAR2(25), STATE VARCHAR2(25), ZIP NUMBER);
```

Example 2:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
CREATE TYPE PERSON_TY AS OBJECT(  
    NAME VARCHAR2(25), ADDRESS ADDRESS_TY);
```

Example 3:

```
CREATE TYPE ADDRESS_TY AS OBJECT(  
    STREET VARCHAR2 (50), CITY VARCHAR2 (25), STATE VARCHAR2 (25), ZIP NUMBER);
```

Example 4:

```
CREATE TYPE PERSON_TY AS OBJECT(  
    NAME VARCHAR2 (25), ADDRESS ADDRESS_TY);
```

Example 5:

```
CREATE TABLE CUSTOMER(  
    CUSTOMER_ID NUMBER, PERSON PERSON_TY);
```

Example 6:

```
DESC CUSTOMER;
```

Example 7:

```
DESC PERSON_TY;
```

Example 8:

```
DESC ADDRESS_TY;
```

Example 9:

```
SELECT COLUMN_NAME, DATA_TYPE FROM USER_TAB_COLUMNS  
    WHERE TABLE_NAME = 'CUSTOMER';
```

Example 10:

```
SELECT ATTR_NAME, LENGTH, ATTR_TYPE_NAME FROM USER_TYPE_ATTRS  
    WHERE TYPE_NAME = 'PERSON_TY';
```

Example 11:

Query USER_TYPE_ATTRS again to see the attributes of the ADDRESS_TY datatype:

```
SELECT ATTR_NAME, LENGTH, ATTR_TYPE_NAME FROM USER_TYPE_ATTRS  
    WHERE TYPE_NAME = 'ADDRESS_TY';
```

Example 12:

```
INSERT INTO CUSTOMER VALUES(1,      PERSON_TY('Sharanam',  
    ADDRESS_TY('Dadar', 'Mumbai', 'Maharashtra', 400016)));
```

```
INSERT INTO CUSTOMER VALUES(2,      PERSON_TY ('Vaishali',  
    ADDRESS_TY ('Balgovinddas Rd', 'Mumbai', 'Maharashtra', 400016)));
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Example 13:

```
SELECT CUSTOMER_ID FROM CUSTOMER;
```

Example 14:

```
SELECT * FROM CUSTOMER;
```

Example 15:

```
SELECT CUSTOMER_ID, CLIENT.PERSON.NAME FROM CUSTOMER CLIENT;
```

Example 16:

```
SELECT CLIENT.PERSON.ADDRESS.STREET FROM CUSTOMER CLIENT;
```

Example 17:

```
SELECT CLIENT.PERSON.NAME, CLIENT.PERSON.ADDRESS.CITY FROM CUSTOMER CLIENT  
WHERE CLIENT.PERSON.ADDRESS.CITY LIKE 'M%';
```

Example 18:

```
UPDATE CUSTOMER CLIENT SET CLIENT.PERSON.ADDRESS.CITY = 'Bombay'  
WHERE CLIENT.PERSON.ADDRESS.CITY = 'Mumbai';
```

Example 19:

```
DELETE FROM CUSTOMER CLIENT WHERE CLIENT.PERSON.ADDRESS.STREET = 'Dadar';
```

Example 20:

```
CREATE OR REPLACE TYPE ADDRESS_TY AS OBJECT(  
    STREET VARCHAR2 (50), CITY VARCHAR2 (25), STATE VARCHAR2 (25), ZIP NUMBER);
```

Next, create PERSON_TY that uses ADDRESS_TY:

```
CREATE OR REPLACE TYPE PERSON_TY AS OBJECT(  
    NAME VARCHAR2 (25), ADDRESS ADDRESS_TY);
```

Next, create CUSTOMER_TY that uses PERSON_TY:

```
CREATE OR REPLACE TYPE CUSTOMER_TY AS OBJECT(  
    CUSTOMER_ID NUMBER, PERSON PERSON_TY);
```

```
CREATE OR REPLACE VIEW CUSTOMER_OV (CUSTOMER_ID, PERSON) AS  
    SELECT CUSTOMER_ID, PERSON_TY (NAME, ADDRESS_TY (STREET, CITY, STATE, ZIP))  
    FROM CUSTOMER;
```

```
INSERT INTO CUSTOMER VALUES(1, 'Sharanam', 'Dadar', 'Mumbai', 'Maharashtra', 400016);
```

```
INSERT INTO CUSTOMER
```

```
VALUES(2, 'Vaishali', 'Balgovinddas Rd', 'Mumbai', 'Maharashtra', 400016);
```

```
INSERT INTO CUSTOMER VALUES(3, 'Hansel', 'Darya Rd', 'Ahemdabad', 'Gujarat', 300042);
```

Example 21:

```
CREATE OR REPLACE VIEW CUSTOMER_OV (CUSTOMER_ID, PERSON) AS  
    SELECT CUSTOMER_ID, PERSON_TY (NAME, ADDRESS_TY (STREET, CITY, STATE, ZIP))  
    FROM CUSTOMER WHERE STATE = 'Maharashtra';
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Example 22:

```
INSERT INTO CUSTOMER VALUES(4, 'Silicon Chip Technologies', 'A/5 Jay Chambers', 'Vile Parle (E)', 'Maharashtra', 400057);
```

Example 23:

```
INSERT INTO CUSTOMER_OV VALUES(5, PERSON_TY ('Jasper International', ADDRESS_TY ('A/7 Jay Chambers', 'Vile Parle (E)', 'Maharashtra', 400057)));
```

Example 24:

1. For creating TYPE ADDRESS_TY:

```
CREATE OR REPLACE TYPE ADDRESS_TY AS OBJECT(  
    STREET VARCHAR2(50), CITY VARCHAR2(25), STATE VARCHAR2(25), ZIP NUMBER);
```

2. For creating TYPE NAME_TY:

```
CREATE OR REPLACE TYPE NAME_TY AS OBJECT(  
    NAME VARCHAR2(25), ADDRESS ADDRESS_TY);
```

3. For creating TYPE DEPENDENT_TY:

```
CREATE OR REPLACE TYPE DEPENDENT_TY AS OBJECT(  
    RELATION VARCHAR2(15), NAME NAME_TY, AGE NUMBER);
```

4. For creating a NESTED TABLE:

```
CREATE OR REPLACE TYPE DEPENDENT_LIST AS TABLE OF DEPENDENT_TY;
```

5. For creating TYPE EMPLOYEE_INFO_TY:

```
CREATE OR REPLACE TYPE EMPLOYEE_INFO_TY AS OBJECT(  
    EMPLOYEE_ID NUMBER(5), NAME NAME_TY, SALARY NUMBER(10,2),  
    DEPT_ID NUMBER(5), DEPENDENTS DEPENDENT_LIST);
```

6. For creating the TABLE EMPLOYEE_INFO of the TYPE EMPLOYEE_INFO_TY:

```
CREATE TABLE EMPLOYEE_INFO OF EMPLOYEE_INFO_TY  
    OIDINDEX OID_EMPLOYEE_INFO  
    NESTED TABLE DEPENDENTS STORE AS DEPENDENTS_TY;
```

1. Inserting values in the nested table:

```
INSERT INTO EMPLOYEE_INFO EMP VALUES(1, NAME_TY('Sharanam',  
    ADDRESS_TY('JAY Chambers', 'VILE PARLE', 'MUMBAI', 400057)),8000,10,  
    DEPENDENT_LIST(  
        DEPENDENT_TY('Sister', NAME_TY('Stuti',  
            ADDRESS_TY('Balgovinddas RD', 'Dadar', 'Mumbai', 400016)), 19),  
        DEPENDENT_TY('Mother', NAME_TY('Gopi',  
            ADDRESS_TY('Balgovinddas RD', 'Dadar', 'Mumbai', 400016)), 40),  
        DEPENDENT_TY('Father', NAME_TY('Chaitanya',
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

ADDRESS_TY('Balgovinddas RD', 'Dadar', 'Mumbai', 400016)), 42));

2. Inserting only detail table values in the nested table:

```
INSERT INTO THE (SELECT DEPENDENTS FROM EMPLOYEE_INFO) DEPENDS
VALUES(DEPENDENT_TY('Friend', NAME_TY('Vaishali',
ADDRESS_TY('Balgovinddas Rd', 'Dadar', 'Mumbai', 400016)),23));
INSERT INTO THE (SELECT DEPENDENTS FROM employee_info) DEPENDS
VALUES(DEPENDENT_TY('Colleague', NAME_TY('Hansel',
ADDRESS_TY('Subhash Rd', 'Parle', 'Mumbai', 400057)), 22));
```

3. Updating values of a child record in the nested table:

```
UPDATE THE (SELECT DEPENDENTS FROM EMPLOYEE_INFO) DEPENDS
SET DEPENDS.RELATION = 'Wife' WHERE DEPENDS.RELATION = 'Friend';
```

4. Deleting values of a child record in the nested table:

```
DELETE THE (SELECT DEPENDENTS FROM EMPLOYEE_INFO) DEPENDS
WHERE DEPENDS.RELATION = 'Colleague';
```

Example 25:

```
CREATE TABLE COMPANY_INFO(NAME VARCHAR2(50), ADDRESS VARCHAR2(1000));
```

Example 26:

```
CREATE TYPE COMPANY_ADDRESS_TY AS VARRAY(3) OF VARCHAR2(1000);
```

Example 27:

```
CREATE TABLE COMPANY_INFO(
COMPANY_NAME VARCHAR2(50), ADDRESS COMPANY_ADDRESS_TY);
```

Example 28:

```
DESC COMPANY_INFO;
```

Example 29:

```
SELECT COLUMN_NAME, DATA_TYPE FROM USER_TAB_COLUMNS
WHERE TABLE_NAME = 'COMPANY_INFO';
```

Example 30:

```
SELECT TYPECODE, ATTRIBUTES FROM USER_TYPES
WHERE TYPE_NAME = 'COMPANY_ADDRESS_TY';
```

Example 31:

```
SELECT TYPE_NAME, COLL_TYPE, UPPER_BOUND FROM USER_COLL_TYPES
WHERE TYPE_NAME = 'COMPANY_ADDRESS_TY';
```

Example 32:

```
INSERT INTO COMPANY_INFO VALUES('Silicon Chip Technologies',
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
COMPANY_ADDRESS_TY('A/5 Jay Chambers, Service Road, Vile Parle (E), Mumbai 57', NULL, NULL));
INSERT INTO COMPANY_INFO VALUES('Jasper International',
    COMPANY_ADDRESS_TY('S.D.F II, Seepz, Andheri(E), Mumbai', 'ABBA House, MIDC, Andheri (E), Mumbai',
    'Emmar Commercial Complex, A/5-407, S.V. Road, Borivli(W)');
```

Examples For The Use Of REF

1. For creating a TYPE object:

```
CREATE TYPE DEPT_TY AS OBJECT(
    DNAME VARCHAR2(100), ADDRESS VARCHAR2(200));
```

2. For creating a TABLE object using the above TYPE object:

```
CREATE TABLE DEPT OF DEPT_TY;
```

3. For creating a TABLE object that references to the TYPE object and also specifies the SCOPE:

```
CREATE TABLE EMP(
    ENAME VARCHAR2(100), ENUMBER NUMBER, EDEPT REF DEPT_TY SCOPE IS DEPT);
```

4. For inserting values in the DEPT table:

```
INSERT INTO DEPT VALUES(DEPT_TY('Sales', '501 Baliga Street'));
INSERT INTO DEPT VALUES(DEPT_TY('Accounts', '84 Darya Ganj'));
```

5. For viewing the DEPT table:

```
SELECT * FROM DEPT;
```

6. For viewing the REF from the DEPT table:

```
SELECT REF(D) FROM DEPT D;
```

7. For inserting a row into the EMP table for an employee in Sales department:

```
INSERT INTO EMP SELECT 'Nirmal Pandey', 1, REF(d) FROM DEPT D
    WHERE D.DNAME = 'Sales';
```

8. For viewing records from the EMP table:

```
SELECT * FROM EMP;
```

9. For viewing the ENAME, ENUMBER and the details of EDEPT column of the EMP table using the DEREf routine:

```
SELECT ENAME, ENUMBER, DEREf (EDEPT) FROM EMP;
```

14. ADVANCE FEATURES IN SQL * PLUS

CODE A TREE-STRUCTURED QUERY

Example 1:

```
SELECT LPAD(' ', LEVEL * 4) || FNAME || ' ' || LNAME "Employee Hierarchy"
    FROM EMP_MSTR
    CONNECT BY PRIOR EMP_NO = MNGR_NO START WITH MNGR_NO IS NULL;
```

Example 2:

```
SELECT * FROM (SELECT B.NAME "BRANCH",
    DECODE(E.BRANCH_NO, 'B1', (SELECT COUNT(EMP_NO) FROM EMP_MSTR
    WHERE BRANCH_NO = 'B1')) "B1",
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
DECODE(E.BRANCH_NO, 'B2', (SELECT COUNT(EMP_NO) FROM EMP_MSTR
WHERE BRANCH_NO = 'B2')) "B2",
DECODE(E.BRANCH_NO, 'B3', (SELECT COUNT(EMP_NO) FROM EMP_MSTR
WHERE BRANCH_NO = 'B3')) "B3",
DECODE(E.BRANCH_NO, 'B4', (SELECT COUNT(EMP_NO) FROM EMP_MSTR
WHERE BRANCH_NO = 'B4')) "B4",
DECODE(E.BRANCH_NO, 'B5', (SELECT COUNT(EMP_NO) FROM EMP_MSTR
WHERE BRANCH_NO = 'B5')) "B5",
DECODE(E.BRANCH_NO, 'B6', (SELECT COUNT(EMP_NO) FROM EMP_MSTR
WHERE BRANCH_NO = 'B6')) "B6"
FROM EMP_MSTR E, BRANCH_MSTR B
WHERE B.BRANCH_NO = E.BRANCH_NO
GROUP BY B.NAME, E.BRANCH_NO) ORDER BY 3;
```

Example 3:

```
SELECT DUMP(ACCT_NO) FROM ACCT_MSTR;
```

Example 4:

1

```
UPDATE EMP_MSTR SET MNAME = NULL;
```

```
RENAME EMP_MSTR TO EMP_MSTR_BASE;
```

```
CREATE VIEW EMP_MSTR
```

```
AS SELECT EMP_NO, BRANCH_NO, FNAME, LNAME, DEPT, DESIG FROM EMP_MSTR_BASE;
```

2

```
CREATE TABLE EMP_MSTR_NEW
```

```
AS SELECT EMP_NO, BRANCH_NO, FNAME, LNAME, DEPT, DESIG FROM EMP_MSTR;
```

```
DROP TABLE EMP_MSTR CASCADE CONSTRAINTS;
```

```
RENAME EMP_MSTR_NEW TO EMP_MSTR;
```

3

```
ALTER TABLE EMP_MSTR DROP COLUMN MNAME;
```

4

```
ALTER TABLE EMP_MSTR SET UNUSED COLUMN MNAME;
```

```
SELECT * FROM SYS.DBA_UNUSED_COL_TABS;
```

```
ALTER TABLE EMP_MSTR DROP UNUSED COLUMNS;
```

Example 5:

Solution 1

```
RENAME BRANCH_MSTR TO BRANCH_MSTR_BASE;
```

```
CREATE VIEW BRANCH_MSTR(BRANCH_NO, BRANCH_NAME)
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
AS SELECT * FROM BRANCH_MSTR_BASE;
```

Solution 2:

```
CREATE TABLE BRANCH_MSTR_NEW(BRANCH_NO, BRANCH_NAME)
AS SELECT * FROM BRANCH_MSTR;
```

```
DROP TABLE BRANCH_MSTR CASCADE CONSTRAINTS;
```

```
RENAME BRANCH_MSTR_NEW TO BRANCH_MSTR;
```

Solution 3:

```
ALTER TABLE BRANCH_MSTR ADD (BRANCH_NAME VARCHAR2(25));
```

```
UPDATE BRANCH_MSTR SET BRANCH_NAME = NAME;
```

```
ALTER TABLE BRANCH_MSTR DROP COLUMN NAME;
```

Example 6:

Solution 1: Using Sub Queries

```
SELECT ROWNUM RN, EMP_NO, FNAME FROM EMP_MSTR WHERE (ROWID, 0)
IN (SELECT ROWID, MOD(ROWNUM,2) FROM EMP_MSTR);
```

Solution 2: Using dynamic views

```
SELECT * FROM (SELECT ROWNUM RN, EMP_NO, FNAME FROM EMP_MSTR) E
WHERE MOD(E.RN,2) = 0;
```

Solution 3: Using GROUP BY and HAVING

```
SELECT ROWNUM, EMP_NO, FNAME FROM EMP_MSTR
GROUP BY ROWNUM, EMP_NO, FNAME
HAVING MOD(ROWNUM,2) = 0 OR ROWNUM = 2-0;
```

Example 7:

```
CREATE TABLE CUSTOMERS (CUST_NO NUMBER, NAME VARCHAR2(25));
```

```
INSERT INTO CUSTOMERS VALUES(0, 'Sharanam');
```

```
INSERT INTO CUSTOMERS VALUES(0, 'Vaishali');
```

```
INSERT INTO CUSTOMERS VALUES(0, 'Hansel');
```

```
INSERT INTO CUSTOMERS VALUES(0, 'Chhaya');
```

```
INSERT INTO CUSTOMERS VALUES(0, 'Ivan');
```

```
SELECT * FROM CUSTOMERS;
```

```
SELECT * FROM CUSTOMERS;
```

```
CREATE SEQUENCE SEQ_CUSTNO START WITH 1 INCREMENT BY 1;
```

```
UPDATE CUSTOMERS SET CUST_NO = SEQ_CUSTNO.NEXTVAL;
```

```
SELECT * FROM CUSTOMERS;
```

```
CREATE UNIQUE INDEX idxCUST_NO ON CUSTOMERS(CUST_NO);
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

INSERT INTO CUSTOMERS VALUES(1, 'Sharanam');

Example 8:

```
SELECT TO_CHAR(SYSDATE, 'DD-MON-YYYY HH:MI:SS') "Date",
       TO_CHAR(SYSDATE+1, 'DD-MON-YYYY HH:MI:SS') "By 1 Day",
       TO_CHAR(SYSDATE+1/24, 'DD-MON-YYYY HH:MI:SS') "By 1 Hour",
       TO_CHAR(SYSDATE+1/1440, 'DD-MON-YYYY HH:MI:SS') "By 1 Minute",
       TO_CHAR(SYSDATE+ 1/86400 , 'DD-MON-YYYY HH:MI:SS') "By 1 Second" FROM DUAL;
```

Example 9:

```
SELECT ACCT_NO, COUNT(*) "TRANSACTIONS PERFORMED"
       FROM TRANS_MSTR GROUP BY ACCT_NO;
```

Example 10:

```
SELECT CUST_NO,
       SUM(DECODE(SUBSTR(ACCT_FD_NO, 1, 2), 'CA', 1, 0)) "CURRENT ACCOUNTS",
       SUM(DECODE(SUBSTR(ACCT_FD_NO, 1, 2), 'SB', 1, 0)) "SAVINGS ACCOUNTS",
       SUM(DECODE(SUBSTR(ACCT_FD_NO, 1, 2), 'FS', 1, 0)) "FIXED DEPOSITS",
       COUNT(ACCT_FD_NO) "TOTAL"
       FROM ACCT_FD_CUST_DTLS GROUP BY CUST_NO;
```

Example 11:

Solution 1:

```
SELECT * FROM (SELECT ROWNUM RN, FNAME FROM EMP_MSTR
              WHERE ROWNUM < 8) WHERE RN BETWEEN 4 and 7;
```

Solution 2:

```
SELECT ROWNUM RN, FNAME FROM EMP_MSTR
       GROUP BY ROWNUM, FNAME HAVING ROWNUM BETWEEN 4 AND 7;
```

Solution 3:

```
SELECT ROWNUM RN, FNAME FROM EMP_MSTR WHERE ROWID IN(
       SELECT ROWID FROM EMP_MSTR WHERE ROWNUM <= 7
       MINUS
       SELECT ROWID FROM EMP_MSTR WHERE ROWNUM < 4);
```

Example 12:

Solution 1:

```
ALTER USER hansel IDENTIFIED BY hansel123;
```

Solution 2:

```
Password hansel;
```

Solution 3:

```
Password;
```

Example 13:

Solution:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
SELECT 'CUSTOMER NAME: ' || FNAME || ' ' || MNAME || ' ' || LNAME || CHR(10) ||  
      'BIRTHDATE: ' || DOB_INC || CHR(10) || 'OCCUPATION: ' || OCCUP "Customer Details"  
FROM CUST_MSTR WHERE CUST_NO LIKE 'C%';
```

Example 14:

Solution 1: FOR UPPER-CASE LETTERS

```
SELECT TO_CHAR(TO_DATE(34654,'J'),'JSP') FROM DUAL;
```

Solution 2: FOR TITLE-CASE LETTERS

```
SELECT TO_CHAR(TO_DATE(34654,'J'),'Jsp') FROM DUAL;
```

Solution 3: FOR LOWER-CASE LETTERS

```
SELECT TO_CHAR(TO_DATE(34654,'J'),'jSP') FROM DUAL;
```

```
SELECT 'Rupees ' || DECODE(TRUNC(34654.23), 0, 'ZERO',  
      TO_CHAR(TO_DATE(TRUNC(34654.23),'J'),'JSP')) || ' AND ' ||  
      DECODE(TRUNC(MOD(34654.23,1)*100), 0, 'ZERO',  
      TO_CHAR(TO_DATE(TRUNC(MOD(34654.23,1)*100),'J'),'JSP')) || ' Paise'  
FROM DUAL;
```

Example 15:

Solution:

```
/* Suppress page headers, titles and all formatting */
```

```
SET PAGESIZE 0
```

```
/* Switch off the SQL text before/after any variable substitution */
```

```
SET VERIFY OFF
```

```
/* Set line size, make this as big as desired */
```

```
SET LINES 700
```

```
/* Delete any blank spaces at the end of each spooled line */
```

```
SET TRIMSPOOL ON
```

```
/* Switch off the lines number display returned by the query */
```

```
SET FEEDBACK OFF
```

```
/* Switch off SELECT output to the screen */
```

```
SET TERMOUT OFF
```

```
/* Separate each column by a comma character (CSV output) */
```

```
SET COLSEP ','
```

```
/* Put the SELECT output into a file*/
```

```
SPOOL MY_EMP_REPORT.TXT
```

```
SELECT EMP_NO, FNAME, LNAME, B.NAME, DEPT, DESIG
```

```
      FROM EMP_MSTR E, BRANCH_MSTR B WHERE E.BRANCH_NO = B.BRANCH_NO;
```

```
SPOOL OFF
```

Example16:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
SELECT NVL(FNAME, 'A'), NVL(MNAME, 'Corporate'), NVL(LNAME, 'Customer'), DOB_INC, OCCUP, PANCOPY,  
FORM60 FROM CUST_MSTR;
```

Example 17:

```
CREATE TABLE MyFriends (NAME VARCHAR2(15));
```

```
INSERT INTO MyFriends VALUES ('Neeta');
```

```
INSERT INTO MyFriends VALUES ('Mita');
```

```
INSERT INTO MyFriends VALUES ('Dipu');
```

```
INSERT INTO MyFriends VALUES ('Deepu');
```

```
INSERT INTO MyFriends VALUES ('Dipa');
```

```
INSERT INTO MyFriends VALUES ('Anil');
```

```
INSERT INTO MyFriends VALUES ('Sunil');
```

```
COMMIT;
```

```
SELECT * FROM MyFriends;
```

```
SELECT * FROM MyFriends WHERE SOUNDEX(NAME) = SOUNDEX('Nita');
```

```
SELECT * FROM MyFriends WHERE SOUNDEX(NAME) = SOUNDEX('Deep');
```

```
SELECT SOUNDEX(NAME), NAME, SOUNDEX('DEEP') FROM MYFRIENDS;
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Object Oriented Programming Using Java Lab

Course Code: CS694C

L-T-P scheme: 0-0-3

Course Credit: 2

Objectives:

1. To strengthen their problem solving ability by applying the characteristics of an object oriented approach.
2. To introduce object oriented concepts in Java.

Learning Outcomes:

1. Explain what constitutes an object-oriented approach to programming and identify the potential benefits of object-oriented programming over other approaches.
2. Apply an object-oriented approach to developing applications of varying complexities.

Course Contents:

Exercises that must be done in this course are listed below:

Exercise No. 1: Class creation with main method and steps of source code compilation and execution.

Exercise No. 2: Design a stack and a queue.

Exercise No. 3: Design different types of linked lists for different operations.

Exercise No. 4: Methods of String.

Exercise No. 5: Implement different types of polymorphism: overloading and overriding.

Exercise No. 6: Implement different types of inheritance.

Exercise No. 7: Use of package with access specifier.

Exercise No. 8: Write a program using static keyword.

Exercise No. 9: Write a program to use this, this(), super, super().

Exercise No. 10: Exception handling.

Exercise No. 11: Threading.

Exercise No. 12: Applet programming and Action Event.

Exercise No. 13: Swing programming and Layout.

Text Book:

1. E. Balagurusamy – " Programming With Java: A Primer" – 3rd Ed. – , Tata Mc Graw Hill.
2. Herbert Schildt, Java: The Complete Reference (Tata Mcgraw Hill Education Private , 7th Ed).

Recommended Systems/Software Requirements:

Java Development Kit and Java Runtime Environment, preferable latest version.

Experiment No: 1: Class creation with main method and steps of source code compilation and execution.

Description:

Create a file with .java extension. Write a class with main method. Then compile that source code using javac command(java compiler) then execute generated bytecode using java command.

```
/* First java program First.java */
public class First {
    public static void main(String args[]){
        System.out.println("Welcome");
    }
}
```

Steps of execution:

javac First.java //For compilation

OUTPUT:

Welcome

Experiment No: 2A: Design a stack.

Aim: Write a program in java to design stack.

Description:

A stack is a container of objects that are inserted and removed according to the last-in first-out (LIFO) principle. In the pushdown stacks only two operations are allowed: push the item into the stack, and pop the item out of the stack. A stack is a limited access data structure - elements can be added and removed from the stack only at the top. push adds an item to the top of the stack, pop removes the item from the top. A helpful analogy is to think of a stack of books; you can remove only the top book, also you can add a new book on the top.

/* Java program Stack.java */

```
class Stack {

    //stack holds 10 values

    int stack[] = new int[10];
    //top of the stack
    int tos;

    Stack() {
        //initially top of stack -1, denotes empty
        tos = -1;
    }

    //push or add an element top of the stack
    void push(int element) {
        //if stack is full
        if (tos == 9) {
            System.out.println("Stack is full!");
        } else {
            stack[++tos] = element;
        }
    }

    //pop or remove element from top of stack
    int pop() {
        //if no element in stack
        if (tos < 0) {
            System.out.println("Stack is empty");
            //0 indicates operation fails
            return 0;
        } else {
            return stack[tos--];
        }
    }
}
```

INPUT:

If push(1) is invoked, 1 will be added onto the stack. Again if push(2) is invoked then 2 will be added onto the stack and now top of the stack is 2.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

OUTPUT:

If return value of pop() is printed then 2 will be printed as top of the element is 2.

Experiment No: 2B: Design a queue.

Aim: Write a program in java to design queue.

Description:

A queue is a container of objects (a linear collection) that are inserted and removed according to the first-in first-out (FIFO) principle. An excellent example of a queue is a line of students in the food court of the UC. New additions to a line made to the back of the queue, while removal (or serving) happens in the front. In the queue only two operations are allowed enqueue and dequeue. Enqueue means to insert an item into the back of the queue, dequeue means removing the front item. The picture demonstrates the FIFO access.

/* Java program Queue.java */

```
public class Queue {

    private static final int capacity = 3;
    int arr[] = new int[capacity];
    int size = 0;
    int top = -1;
    int rear = 0;

    public void push(int pushedElement) {
        if (top < capacity - 1) {
            top++;
            arr[top] = pushedElement;
        } else {
            System.out.println("Overflow !");
        }
    }

    public int pop() {
        int e = arr[rear];
        if (top >= rear) {
            rear++;
        } else {
            System.out.println("Underflow !");
        }
        return e;
    }

    public void display() {
        if (top >= rear) {
            System.out.println("Elements in Queue : ");
            for (int i = rear; i <= top; i++) {
                System.out.println(arr[i]);
            }
        }
    }
}
```

INPUT:

The push() method is for inserting element in a queue. pop() to get element from the rear

OUTPUT:

If display() is invoked, all the elements will be printed from the rear position to top position.

Experiment No: 3A: Design Singly Link List.

Aim: Write a program in java to design singly link list.

Description:

Singly Linked Lists are a type of data structure. It is a type of list. In a singly linked list each node in the list stores the contents of the node and a pointer or reference to the next node in the list. It does not store any pointer or reference to the previous node. It is called a singly linked list because each node only has a single link to another node. To store a single linked list, you only need to store a reference or pointer to the first node in that list. The last node has a pointer to nothingness to indicate that it is the last node.

```
/* Java program SinglyLinkedList.java */
```

```
public class SinglyLinkedList<T> {

    private Node<T> head;
    private Node<T> tail;

    public void add(T element) {

        Node<T> nd = new Node<T>();
        nd.setValue(element);
        System.out.println("Adding: " + element);
        /**
         * check if the list is empty
         */
        if (head == null) {
            //since there is only one element, both head and
            //tail points to the same object.
            head = nd;
            tail = nd;
        } else {
            //set current tail next link to new node
            tail.setNextRef(nd);
            //set tail as newly created node
            tail = nd;
        }
    }

    public void addAfter(T element, T after) {

        Node<T> tmp = head;
        Node<T> refNode = null;
        System.out.println("Traversing to all nodes..");
        /**
         * Traverse till given element
         */
        while (true) {
            if (tmp == null) {
                break;
            }
            if (tmp.compareTo(after) == 0) {
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
        break;
    }
    tmp = tmp.getNextRef();
}
if (refNode != null) {
    //add element after the target node
    Node<T> nd = new Node<T>();
    nd.setValue(element);
    nd.setNextRef(tmp.getNextRef());
    if (tmp == tail) {
        tail = nd;
    }
    tmp.setNextRef(nd);
} else {
    System.out.println("Unable to find the given element...");
}
}
```

```
public void deleteFront() {

    if (head == null) {
        System.out.println("Underflow...");
    }
    Node<T> tmp = head;
    head = tmp.getNextRef();
    if (head == null) {
        tail = null;
    }
    System.out.println("Deleted: " + tmp.getValue());
}
```

```
public void deleteAfter(T after) {

    Node<T> tmp = head;
    Node<T> refNode = null;
    System.out.println("Traversing to all nodes..");
    /**
     * Traverse till given element
     */
    while (true) {
        if (tmp == null) {
            break;
        }
        if (tmp.compareTo(after) == 0) {
            //found the target node, add after this node
            refNode = tmp;
            break;
        }
        tmp = tmp.getNextRef();
    }
    if (refNode != null) {
        tmp = refNode.getNextRef();
        refNode.setNextRef(tmp.getNextRef());
        if (refNode.getNextRef() == null) {
            tail = refNode;
        }
    }
}
```

```

        System.out.println("Deleted: " + tmp.getValue());
    } else {
        System.out.println("Unable to find the given element...");
    }
}

public void traverse() {

    Node<T> tmp = head;
    while (true) {
        if (tmp == null) {
            break;
        }
        System.out.println(tmp.getValue());
        tmp = tmp.getNextRef();
    }
}

public static void main(String a[]) {
    SinglyLinkedList<Integer> sl = new SinglyLinkedList<Integer>();
    sl.add(3);
    sl.add(32);
    sl.add(54);
    sl.add(89);
    sl.addAfter(76, 54);
    sl.deleteFront();
    sl.deleteAfter(76);
    sl.traverse();
}
}

class Node<T> implements Comparable<T> {

    private T value;
    private Node<T> nextRef;

    public T getValue() {
        return value;
    }

    public void setValue(T value) {
        this.value = value;
    }

    public Node<T> getNextRef() {
        return nextRef;
    }

    public void setNextRef(Node<T> ref) {
        this.nextRef = ref;
    }

    @Override
    public int compareTo(T arg) {
        if (arg == this.value) {
            return 0;

```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
}  
}  
}
```

INPUT:

Input is provided in main method.

OUTPUT:

```
Adding: 3  
Adding: 32  
Adding: 54  
Adding: 89  
Traversing to all nodes..  
Deleted: 3  
Traversing to all nodes..  
Deleted: 89 32 54 76
```

Experiment No: 3B: Design Doubly Linked List.

Aim: Write a program in java to design doubly link list.

Description:

A doubly-linked list is a linked data structure that consists of a set of sequentially linked records called nodes. Each node contains two fields, called links that are references to the previous and to the next node in the sequence of nodes. The beginning and ending nodes previous and next links, respectively, point to some kind of terminator, typically a sentinel node or null, to facilitate traversal of the list. If there is only one sentinel node, then the list is circularly linked via the sentinel node. It can be conceptualized as two singly linked lists formed from the same data items, but in opposite sequential orders.

```
/* Java program DoublyLinkedList.java */
```

```
import java.util.NoSuchElementException;
```

```
public class DoublyLinkedList<E> {
```

```
    private Node head;  
    private Node tail;  
    private int size;
```

```
    public DoublyLinkedListImpl() {  
        size = 0;  
    }
```

```
    /**  
     * this class keeps track of each element information  
     *  
     * @author java2novice  
     *  
     */
```

```
    private class Node {
```

```
        E element;  
        Node next;  
        Node prev;
```

```

        this.element = element;
        this.next = next;
        this.prev = prev;
    }
}

/**
 * returns the size of the linked list
 *
 * @return
 */
public int size() {
    return size;
}

/**
 * return whether the list is empty or not
 *
 * @return
 */
public boolean isEmpty() {
    return size == 0;
}

/**
 * adds element at the starting of the linked list
 *
 * @param element
 */
public void addFirst(E element) {
    Node tmp = new Node(element, head, null);
    if (head != null) {
        head.prev = tmp;
    }
    head = tmp;
    if (tail == null) {
        tail = tmp;
    }
    size++;
    System.out.println("adding: " + element);
}

/**
 * adds element at the end of the linked list
 *
 * @param element
 */
public void addLast(E element) {

    Node tmp = new Node(element, null, tail);
    if (tail != null) {
        tail.next = tmp;
    }
    tail = tmp;
    if (head == null) {
        head = tmp;
    }
}

```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
}

/**
 * this method walks forward through the linked list
 */
public void iterateForward() {

    System.out.println("iterating forward..");
    Node tmp = head;
    while (tmp != null) {
        System.out.println(tmp.element);
        tmp = tmp.next;
    }
}

/**
 * this method walks backward through the linked list
 */
public void iterateBackward() {

    System.out.println("iterating backward..");
    Node tmp = tail;
    while (tmp != null) {
        System.out.println(tmp.element);
        tmp = tmp.prev;
    }
}

/**
 * this method removes element from the start of the linked list
 *
 * @return
 */
public E removeFirst() {
    if (size == 0) {
        throw new NoSuchElementException();
    }
    Node tmp = head;
    head = head.next;
    head.prev = null;
    size--;
    System.out.println("deleted: " + tmp.element);
    return tmp.element;
}

/**
 * this method removes element from the end of the linked list
 *
 * @return
 */
public E removeLast() {
    if (size == 0) {
        throw new NoSuchElementException();
    }
    Node tmp = tail;
    tail = tail.prev;
```

```

        size--;
        System.out.println("deleted: " + tmp.element);
        return tmp.element;
    }

    public static void main(String a[]) {

        DoublyLinkedListImpl<Integer> dll = new DoublyLinkedListImpl<Integer>();
        dll.addFirst(10);
        dll.addFirst(34);
        dll.addLast(56);
        dll.addLast(364);
        dll.iterateForward();
        dll.removeFirst();
        dll.removeLast();
        dll.iterateBackward();
    }
}

```

INPUT:

Input is provided in main method.

OUTPUT:

```

adding: 10
adding: 34
adding: 56
adding: 364
iterating forward.. 34 10 56 364
deleted: 34
deleted: 364
iterating backward.. 56 10

```

Experiment No: 4A: Methods of String: indexOf()

Aim: Write a program in java to implement indexOf().

Description:

The String class is immutable (constant), i.e. Strings in java, once created and initialized, cannot be changed. The String is a final class, no other class can extend it, and you cannot change the state of the string. String values cannot be compare with '==', for string value comparison, use equals() method. String class supports various methods, including comparing strings, extracting substrings, searching characters & substrings, converting into either lower case or upper case, etc.

Below method shows how to get index of a specified character or string from the given string. By using indexOf() method you get get the position of the specified string or char from the given string. You can also get the index strting from a specified position of the string.

/* Java program MyStringIndexof.java */

```

public class MyStringIndexof {

    public static void main(String[] a) {

        String str = "Use this string for testing this";
        System.out.println("Basic indexOf() example");
        System.out.println("Char 's' at first occurrence: " + str.indexOf('s'));
        System.out.println("String \"this\" at first occurrence: " + str.indexOf("this"));
        /**

```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
System.out.println("First occurrence of char 's' from 4th index onwards : "
+ str.indexOf('s', 4));
System.out.println("First occurrence of String \"this\" from 6th index onwards: "
+ str.indexOf("this", 6));

}
}
```

INPUT: Input is provided in main method.

OUTPUT:

```
Basic indexOf() example
Char 's' at first occurrence: 1
String "this" at first occurrence: 4
First occurrence of char 's' from 4th index onwards : 7
First occurrence of String "this" from 6th index onwards: 28
```

Experiment No: 4B: Methods of String: lastIndexOf()

Aim: Write a program in java to implement lastIndexOf().

Description:

Below example shows how to get index of a given character or string from a string in the reverse order, means last occurring index. By using lastIndexOf() method you can get last occurrence of the reference string or character.

/* Java program MyStrLastIndexOf.java */

```
public class MyStrLastIndexOf {

    public static void main(String a[]) {

        String str = "Use this string for testing this";
        System.out.println("Basic lastIndexOf() example");
        System.out.println("Char 's' at last occurrence: " + str.lastIndexOf('s'));
        System.out.println("String \"this\" at last occurrence: " + str.lastIndexOf("this"));
        /**
         * Returns the last occurrence from specified start index, searching
         * backward starting at the specified index.
         */
        System.out.println("first occurrence of char 's' from 24th index backwards: "
+ str.lastIndexOf('s', 24));
        System.out.println("First occurrence of String \"this\" from 26th index backwards: "
+ str.lastIndexOf("this", 26));
    }
}
```

INPUT: Input is provided in main method.

OUTPUT:

```
Basic lastIndexOf() example
Char 's' at last occurrence: 31
String "this" at last occurrence: 28
first occurrence of char 's' from 24th index backwards: 22
First occurrence of String "this" from 26th index backwards: 4
```

Aim: Write a program in java to implement startWith().

Description:

Below example shows how to find whether a string value start with another string value. By using startsWith() method, you can get whether the string starts with the given string or not. Also this method tells that the string occurrence at a specific position.

```
/* Java program MyStrStartsWith.java */
```

```
public class MyStrStartsWith {  
  
    public static void main(String a[]) {  
  
        String str = "This is an example string.";  
        System.out.println("Is this string starts with \"This\"? "  
            + str.startsWith("This"));  
        System.out.println("Is this string starts with \"is\"? "  
            + str.startsWith("is"));  
        System.out.println("Is this string starts with \"is\" after index 5? "  
            + str.startsWith("is", 5));  
    }  
}
```

INPUT: Input is provided in main method.

OUTPUT:

```
Is this string starts with "This"? true  
Is this string starts with "is"? false  
Is this string starts with "is" after index 5? true
```

Experiment No: 4D: Methods of String: endsWith()

Aim: Write a program in java to implement endsWith().

Description:

Below example shows how to find whether a string value ends with another string value. By using endsWith() method, you can get whether the string ends with the given string or not. Also this method tells that the string occurrence at a specific position.

```
/* Java program MyStringEnd.java */
```

```
public class MyStringEnd {  
  
    public static void main(String a[]) {  
  
        String str = "This is a java string example";  
        if (str.endsWith("example")) {  
            System.out.println("This String ends with example");  
        } else {  
            System.out.println("This String is not ending with example");  
        }  
        if (str.endsWith("java")) {  
            System.out.println("This String ends with java");  
        } else {  
            System.out.println("This String is not ending with java");  
        }  
    }  
}
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

INPUT: Input is provided in main method.

OUTPUT:

This String ends with example
This String is not ending with java

Experiment No: 4E: Methods of String: split()

Aim: Write a program in java to implement split().

Description:

Below example shows how to split or brake a string. The split() method splits the string based on the given regular expression or delimiter, and returns the tokens in the form of array. Below example shows splitting string with space, and second split is based on any kind of spaces, that includes tab, enter, line breaks, etc.

/* Java program MyStrSplit.java */

```
public class MyStrSplit {  
  
    public static void main(String a[]) {  
  
        String str = "This program splits a string based on space";  
        String[] tokens = str.split(" ");  
        for (String s : tokens) {  
            System.out.println(s);  
        }  
        str = "This program splits a string based on space";  
        tokens = str.split("\\s+");  
    }  
}
```

INPUT: Input is provided in main method.

OUTPUT:

This
program
splits
a
string
based
on
space

Experiment No: 4F: Methods of String: getChars()

Aim: Write a program in java to implement getChars().

Description:

Below example shows how to copy range of characters from the given string to another character array. By suing getChars() method, you can copy range of characters from the given string.

/* Java program MyCharArrayCopy.java */

```
public class MyCharArrayCopy {
```

```

String str = "Copy chars from this string";
char[] ch = new char[5];
/**
 * The getChars() method accepts 4 parameters first one is the start
 * index from string second one is the end index from string third one
 * is the destination char array fourth one is the start index to append
 * in the char array.
 */
str.getChars(5, 10, ch, 0);
System.out.println(ch);
}
}

```

INPUT: Input is provided in main method.

OUTPUT:

chars

Experiment No: 4G: Methods of String : replace()

Aim: Write a program in java to implement replace().

Description:

Below example shows how to get replace character or a string into a string with the given string. String provides replace() method to replace a specific character or a string which occurs first. replaceAll() method replaces a specific character or a string at each occurrence.

/* Java program MyStringReplace.java */

```

public class MyStringReplace {

    public static void main(String a[]) {

        String str = "This is an example string";
        System.out.println("Replace char 's' with 'o':" + str.replace('s', 'o'));

        System.out.println("Replace first occurrence of string 'is' with 'ui':"
            + str.replaceFirst("is", "ui"));

        System.out.println("Replacing 'is' everywhere with 'no':"
            + str.replaceAll("is", "no"));
    }
}

```

INPUT: Input is provided in main method.

OUTPUT:

Replace char 's' with 'o': This is an example string

Replace first occurrence of string "is" with "ui": This is an example string Replacing "is" everywhere with "no": This no an example string

Experiment No: 4H: Methods of String : equals()

Aim: Write a program in java to implement equals().

Description:

The below example shows how to compare two string objects in java. You can not use "=="

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Also you can ignore case during string compare by calling equalsIgnoreCase() method. '==' operator compares the object reference but not the string value.

/* Java program MyStringEquals.java */

```
public class MyStringEquals {

    public static void main(String a[]) {
        String x = "JUNK";
        String y = "junk";
        /**
         * We cannot use '==' operator to compare two strings. We have to use
         * equals() method.
         */
        if (x.equals(y)) {
            System.out.println("Both strings are equal.");
        } else {
            System.out.println("Both strings are not equal.");
        }
        /**
         * We can ignore case with equalsIgnoreCase() method
         */
        if (x.equalsIgnoreCase(y)) {
            System.out.println("Both strings are equal.");
        } else {
            System.out.println("Both strings are not equal.");
        }
    }
}
```

INPUT: Input is provided in main method.

OUTPUT:

Both strings are not equal.
Both strings are equal.

Experiment No: 4I: Methods of String: concat()

Aim: Write a program in java to implement concat().

Description:

Below example shows different ways of append or concat two string objects. You can append two strings by just using "+" sign. Also you can concatenate two string objects by calling concat() method.

/* Java program MyStringConcat.java */

```
public class MyStringConcat {

    public static void main(String a[]) {
        String b = "jump ";
        String c = "No jump";
        /**
         * We can do string concatenation by two ways. One is by using '+'
         * operator, shown below.
         */
    }
}
```

```

    System.out.println(d);
    /**
     * Another way is by using concat() method, which appends the specified
     * string at the end.
     */
    d = b.concat(c);
    System.out.println(d);
}
}

```

INPUT: Input is provided in main method.

OUTPUT:

```

    jump No jump
    jump No jump

```

Experiment No: 4J: Methods of String: copyValueOf()

Aim: Write a program in java to implement copyValueOf().

Description:

Below example shows how to convert character array to a string object. By using String.copyValueOf() method you can convert char array to string object. Also you can copy range of character array to string.

/* Java program MyArrayCopy.java */

```

public class MyArrayCopy {

    public static void main(String a[]) {
        char ch[] = {'M', 'y', ' ', 'J', 'a', 'v', 'a', ' ', 'e', 'x', 'a', 'm', 'p', 'l', 'e'};
        /**
         * We can copy a char array to a string by using copyValueOf() method.
         */
        String chStr = String.copyValueOf(ch);
        System.out.println(chStr);
        /**
         * We can also copy only range of characters in a char array by
         * copyValueOf() method.
         */
        String subStr = String.copyValueOf(ch, 3, 4);
        System.out.println(subStr);
    }
}

```

INPUT: Input is provided in main method.

OUTPUT:

```

    My Java example
    Java

```

Experiment No: 4K: Methods of String: getBytes()

Aim: Write a program in java to implement getBytes().

Description:

Sometimes we have to convert string object into byte array. You can use getBytes() method

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
/* Java program SinglyLinkedList.java */

public class MyStringBytes {

    public static void main(String a[]) {

        String str = "core java api";
        byte[] b = str.getBytes();
        System.out.println("String length: " + str.length());
        System.out.println("Byte array length: " + b.length);
    }
}
```

INPUT: Input is provided in main method.

OUTPUT:

String length: 13
Byte array length: 13

Experiment No: 5A: Implement polymorphism: overloading.

Aim: Write a program in java to implement method overloading.

Description:

Polymorphism is the capability of a method to do different things based on the object that it is acting upon. In other words, polymorphism allows you define one interface and have multiple implementations. I know it sounds confusing. Don't worry we will discuss this in detail. It is a feature that allows one interface to be used for a general class of actions. An operation may exhibit different behavior in different instances. The behavior depends on the types of data used in the operation. It plays an important role in allowing objects having different internal structures to share the same external interface. Polymorphism is extensively used in implementing inheritance.

Following concepts demonstrate different types of polymorphism in java.

- A) Method Overloading
- B) Method Overriding

In Java, it is possible to define two or more methods of same name in a class, provided that there argument list or parameters are different. This concept is known as Method Overloading.

```
/* Java program Overload.java */

class OverloadDemo {

    void test() {
        System.out.println("No Parameters");
    }

    //overload test with one integer parameter
    void test(int a) {
        System.out.println("a : " + a);
    }

    //overload test with two integer parameter
    void test(int a, int b) {
        System.out.println("a & b : " + a + " " + b);
    }
}
```

```

double test(double a) {
    return a * a;
}
}

class Overload {

    public static void main(String args[]) {
        //create an instance for class OverloadDemo
        OverloadDemo od = new OverloadDemo();

        //call all test methods
        od.test();
        od.test(5);
        od.test(2, 3);
        System.out.println("Result : " + od.test(11.1));
    }
}

```

INPUT: Input is provided in main method

OUTPUT:

```

No Parameters
a : 5
a & B : 2 3
Result : 123.21

```

Experiment No: 5B: Implement different types of polymorphism: overriding

Aim: Write a program in java to implement method overriding.

Description:

Sub class has the same method as of super class. In such cases sub class overrides the super class method without even touching the source code of the super class. This feature is known as method overriding.

/* Java program OverrideTest.java */

```

class Surface {

    int length = 2;
    int width = 3;

    Surface(int l, int w) {
        length = l;
        width = w;
    }

    void showArea() {
        System.out.println("Area of the Surface is : " + (length * width));
    }
}

class Box extends Surface {

    int height = 4;

```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
        System.out.println("Area of the Surface within Box is : " + (length * width));
    }

    void showVolume() {
        System.out.println("Volume of the Cube is : " + (length * width * height));
    }
}

class OverrideTest {

    public static void main(String args[]) {

        //create an instance of Superclass
        Surface s = new Surface();

        //access member of superclass
        s.showArea();

        //create an instance of subclass
        Box b = new Box();

        //access member of subclass & superclass
        b.showArea();
        b.showVolume();
    }
}
```

INPUT: Input is provided in main method.

OUTPUT:

```
Area of the Surface is : 6
Area of the Surface within Box is : 6
Volume of the Cube is : 24
```

Experiment No: Implement inheritance.

Aim: Write a program in java to implement inheritance.

Description:

Inheritance is one of the features of Object-Oriented Programming (OOPs). Inheritance allows a class to use the properties and methods of another class. In other words, the derived class inherits the states and behaviors from the base class. The derived class is also called subclass and the base class is also known as super-class. The derived class can add its own additional variables and methods. These additional variable and methods differentiates the derived class from the base class.

```
/* Java program Inheritance.java */
// A class to display the attributes of the vehicle
```

```
class Vehicle {

    String color;
    int speed;
    int size;

    void attributes() {
        System.out.println("Color : " + color);
    }
}
```

```

        System.out.println("Size : " + size);
    }
}

// A subclass which extends for vehicle
class Car extends Vehicle {

    int CC;
    int gears;

    void attributescar() {
        // The subclass refers to the members of the superclass
        System.out.println("Color of Car : " + color);
        System.out.println("Speed of Car : " + speed);
        System.out.println("Size of Car : " + size);
        System.out.println("CC of Car : " + CC);
        System.out.println("No of gears of Car : " + gears);
    }
}

public class Test {

    public static void main(String args[]) {
        Car b1 = new Car();
        b1.color = "Blue";
        b1.speed = 200;
        b1.size = 22;
        b1.CC = 1000;
        b1.gears = 5;
        b1.attributescar();
    }
}

```

INPUT: Input is provided in main method.

OUTPUT:

```

Color of Car : Blue
Speed of Car : 200
Size of Car : 22
CC of Car : 1000
No of gears of Car : 5

```

Experiment No: 7: Use of package with access specifier.

Aim: Write a program in java of package and access specifiere.

Description:

Access Modifiers is the way of specifying the accessibility of a class and its members with respective to other classes and members.

Packages in Java are a mechanism to encapsulate a group of classes, interfaces and sub packages. Many implementations of Java use a hierarchical file system to manage source and class files. It is easy to organize class files into packages. All we need to do is put related class files in the same directory, give the directory a name that relates to the purpose of the classes, and add a line to the top of each class file that declares the package name, which is the same as the directory name where they reside. In java there are already many predefined packages that we use while programming. For example: java.lang, java.io, java.util etc. However one of the most useful feature of java is that we can define our own packages

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
// top-level interface declaration with public modifier
public interface IPub { ... }
```

```
package pack1;
// top-level class declaration with default modifier
class CDef { ... }
// top-level interface declaration with default modifier
interface CDef { ... }
```

```
package pack1;
// another class in same package Pack1
class A1 {
    // public Class is accessible within the package
    CPub cPubObj;

    // default Class is accessible within the package
    CDef cDefObj;
}
```

```
package pack1;
// default Interface is accessible within the package
class B1 implements IDef { ... }
```

```
package pack1;
// public Interface is accessible within the package
class C1 implements IPub { ... }
```

```
package Pack2;
// Class in other package Pack1
class A2 {

    // public Class is accessible in other package
    CPub cPubObj;

    // default Class is NOT accessible in other package
    CDef cDefObj;
}
```

```
package pack2;
// default Interface is NOT accessible outside the package
class B2 implements IDef { ... }
```

```
package pack2;
// public Interface is accessible outside the package
class C2 implements IPub { ... }
```

INPUT: Not Applicable for this program.

OUTPUT:

public Members: If members are declared as public inside a class then such members are accessible to the classes which are inside and outside of the package where this class is visible. This is the least restrictive of all the accessibility modifiers.

protected Members: If members are declared as protected then these are accessible to all classes in the package and to all subclasses of its class in any package where this class is visible.

Default Members: When no accessibility modifier is specified for the member then implicitly it is declared as Default. These are accessible only to the other classes in the class's package.

private Members: This is the most restrictive of all accessibility modifiers. These members are accessible only within the same class. These are not accessible from any other class within a class's package also.

Experiment No: 8: Write a program using static keyword.

Aim: Write a program in java to implement use of static keyword.

Description:

The static keyword in java is used for memory management mainly. We can apply java static keyword with variables, methods, blocks and nested class. The static keyword belongs to the class than instance of the class.

The static can be:

1. variable (also known as class variable)
2. method (also known as class method)
3. block
4. nested class

```
/* Java program UseStatic.java */
class UseStatic {
    static int a = 3;
    static int b;

    static void dis(int x) {
        System.out.println("x = "+x);
        System.out.println("a = "+a);
        System.out.println("b = "+b);
    }

    static {
        System.out.println("Static block initialized.");
        b = a * 4;
    }

    public static void main(String args[]) {
        dis(9);
    }
}
```

INPUT: Input is provided in main method.

OUTPUT:

```
Static block initialized.
x = 9
a = 3
b = 12
```

Experiment No: 9A: Write a program to use this.

Aim: Write a program in java to show the use of this.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

There can be a lot of usage of java this keyword. In java, this is a reference variable that refers to the current object.

Uses of this keyword:

1. this keyword can be used to refer current class instance variable.
2. this keyword can be used to invoke current class method (implicitly)
3. this can be passed as an argument in the method call.
4. this can be passed as argument in the constructor call.
5. this keyword can also be used to return the current class instance
- 6.

```
/* Java program Student.java */  
//example of this keyword
```

```
class Student {  
  
    int id;  
    String name;  
  
    Student(int id, String name) {  
        this.id = id;  
        this.name = name;  
    }  
  
    void display() {  
        System.out.println(id + " " + name);  
    }  
  
    public static void main(String args[]) {  
        Student s1 = new Student(111, "Ram");  
        Student s2 = new Student(222, "Shan");  
        s1.display();  
        s2.display();  
    }  
}
```

INPUT: Input is provided in main method.

OUTPUT:

```
111 Ram  
222 Shan
```

Experiment No: 9B: Write a program to use this().

Aim: Write a program in java to show the use of this().

Description:

The this() constructor call can be used to invoke the current class constructor (constructor chaining). This approach is better if you have many constructors in the class and want to reuse that constructor. this() can be used to invoke current class constructor.

```
/* Java program Student.java */
```

```
class Student {  
  
    int id;  
    String name;  
    String city;
```

```

Student(int id, String name) {
    this.id = id;
    this.name = name;
}

Student(int id, String name, String city) {
    this(id, name);//now no need to initialize id and name
    this.city = city;
}

void display() {
    System.out.println(id + " " + name + " " + city);
}

public static void main(String args[]) {
    Student e1 = new Student(111, "Ram");
    Student e2 = new Student(222, "Shan", "Kolkata");
    e1.display();
    e2.display();
}
}

```

INPUT: is provided in main method.

OUTPUT:

```

111 Ram null
222 Shan Kolkata

```

Experiment No: 9C: Write a program to use super.

Aim: Write a program in java to show the use of super.

Description:

The super keyword in java is a reference variable that is used to refer immediate parent class object. Whenever you create the instance of subclass, an instance of parent class is created implicitly i.e. referred by super reference variable. super is used to refer immediate parent class instance variable and method.

/* Java program Bike.java */

//example of super keyword

```

class Vehicle {

    int speed = 50;
}

class Bike extends Vehicle {

    int speed = 100;

    void display() {
        System.out.println(super.speed);//will print speed of Vehicle now
    }

    public static void main(String args[]) {
        Bike b = new Bike();
        b.display();
    }
}

```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

INPUT: is provided in main method.

OUTPUT: 50

Experiment No: 9D: Write a program to use super().

Aim: Write a program in java to show the use of super().

Description: super() is used to invoke parent class constructor.

/* Java program Bike.java */

```
class Vehicle {  
  
    Vehicle() {  
        System.out.println("Vehicle is created");  
    }  
}  
  
class Bike extends Vehicle {  
  
    Bike() {  
        super();//will invoke parent class constructor  
        System.out.println("Bike is created");  
    }  
  
    public static void main(String args[]) {  
        Bike b = new Bike();  
    }  
}
```

INPUT: Input is provided in main method.

OUTPUT:

Vehicle is created
Bike is created

Experiment No: 10: Exception handling.

Aim: Write a program in java to show the use of exception handling.

Description:

An exception is an event, which occurs during the execution of a program, that interrupts the normal flow of the program. It is an error thrown by a class or method reporting an error in code. The 'Throwable' class is the superclass of all errors and exceptions in the Java language. Exceptions are broadly classified as 'checked exceptions' and 'unchecked exceptions'. All RuntimeExceptions and Errors are unchecked exceptions. Rest of the exceptions are called checked exceptions. Checked exceptions should be handled in the code to avoid compile time errors. Exceptions can be handled by using 'try-catch' block. Try block contains the code which is under observation for exceptions. The catch block contains the remedy for the exception. If any exception occurs in the try block then the control jumps to catch block. If a method doesn't handle the exception, then it is mandatory to specify the exception type in the method signature using 'throws' clause. We can explicitly throw an exception using 'throw' clause.

/* Java program MyClass.java */

```

public class MyClass{

    public void show(String[] str) throws ArithmeticException,
    ArrayIndexOutOfBoundsException{
        if(str.length<2){
            throw new ArrayIndexOutOfBoundsException("Array length is "
+str.length+" It must be 2");
        } else if(n2==0){
            throw new ArithmeticException("Divided by 0, not possible");
        }else{
            int n1=Integer.parseInt(str[0]);
            int n2=Integer.parseInt(str[1]);
            int result=0;

            result=n1/n2;
            System.out.print("Result : "+result);
        }
    }
    public static void main(String args[]) {
    try {
        new MyClass().show(args);
    } catch(ArithmeticException aex){
        System.out.println(aex);
    } catch(ArrayIndexOutOfBoundsException aibex){
        System.out.println(aibex);
    }
    }
}

```

INPUT-1: If java MyClass 10 2 (command is executed)

OUTPUT-1: Result : 5

INPUT-2: If java MyClass 10 0 (command is executed)

OUTPUT-2: Divided by 0, not possible

INPUT-3: If java MyClass 10 (command is executed)

OUTPUT-4: Array length is 1 It must be 2

Experiment No: 11: Threading

Aim: Write a program in java to implement threading.

Description:

Threading is a facility to allow multiple tasks to run concurrently within a single process. Threads are independent, concurrent execution through a program, and each thread has its own stack.

In Java threads can be implemented in two ways. One is by 'Extending Thread Class' and the other way is by 'Implementing Runnable Interface'

Extending Thread Class is required to 'override run()' method. The run method contains the actual logic to be executed by thread.

Creation of thread object never starts execution, we need to call 'start()' method to run a thread. Examples gives you more details. Other methods supported by Threads are given below.

join(): It makes to wait for this thread to die. You can wait for a thread to finish by calling its join() method.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

yield(): It makes current executing thread object to pause temporarily and gives control to other thread to execute.

notify(): This method is inherited from Object class. This method wakes up a single thread that is waiting on this object's monitor to acquire lock.

notifyAll(): This method is inherited from Object class. This method wakes up all threads that are waiting on this object's monitor to acquire lock.

wait(): This method is inherited from Object class. This method makes current thread to wait until another thread invokes the notify() or the notifyAll() for this object.

/* Java program ThreadTest.java */

```
public class ThreadTest {

    public static void main(String args[]) {

        System.out.println("Create ...");

        MyThread th1 = new MyThread("TH : 1");
        MyThread th2 = new MyThread("TH : 22");
        MyThread th3 = new MyThread("TH : 333");

        MyThread2 th21 = new MyThread2("TH2 : 1");
        MyThread2 th22 = new MyThread2("TH2 : 22");
        MyThread2 th23 = new MyThread2("TH2 : 333");

        System.out.println("Start ...");

        th1.start();
        th2.start();
        th3.start();
        new Thread(th21).start();
        new Thread(th22).start();
        new Thread(th23).start();
    }
}

class MyThread extends Thread {

    public MyThread(String str) {
        super(str);
    }

    public synchronized void access() {

        for (int i = 0; i < 5; i++) {
            System.out.println("Loop " + i + ": " + getName());

            try {
                if (getName().equals("TH : 1")) {
                    sleep(4000);
                }

            } catch (InterruptedException e) {
            }
        }
    }
}
```

```

public void run() {
    access();
    System.out.println("-----");
    System.out.println("End: " + getName());
    System.out.println("-----");
}
}

class MyThread2 implements Runnable {

    String str;

    public MyThread2(String str) {
        this.str = str;
    }

    public synchronized void access() {

        for (int i = 0; i < 5; i++) {
            System.out.println("Loop " + i + ": " + str);

            try {
                if (str.equals("TH : 1")) {
                    Thread.sleep(4000);
                }

            } catch (InterruptedException e) {
            }
        }
    }

    public void run() {
        access();
        System.out.println("-----");
        System.out.println("End: " + str);
        System.out.println("-----");
    }
}

```

INPUT: Input is provided in main method.

OUTPUT: Output sequences may changes time to time.

```

Create ...
Start ...
Loop 0: TH : 22
Loop 0: TH2 : 22
Loop 1: TH : 22
Loop 1: TH2 : 22
Loop 0: TH : 1
Loop 2: TH2 : 22
Loop 3: TH2 : 22
Loop 4: TH2 : 22
-----
End: TH2 : 22
-----
Loop 2: TH : 22
Loop 3: TH : 22

```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

End: TH : 22

Loop 0: TH2 : 1
Loop 1: TH2 : 1
Loop 2: TH2 : 1
Loop 3: TH2 : 1
Loop 4: TH2 : 1

End: TH2 : 1

Loop 0: TH : 333
Loop 1: TH : 333
Loop 2: TH : 333
Loop 3: TH : 333
Loop 4: TH : 333

End: TH : 333

Loop 0: TH2 : 333
Loop 1: TH2 : 333
Loop 2: TH2 : 333
Loop 3: TH2 : 333
Loop 4: TH2 : 333

End: TH2 : 333

Loop 1: TH : 1
Loop 2: TH : 1
Loop 3: TH : 1
Loop 4: TH : 1

End: TH : 1

Experiment No: 12: Applet programming

Aim: Write a program in java to implement action event with applet.

Description:

A Java applet is a small application which is written in Java or another programming language that compiles to Java bytecode and delivered to users in the form of that bytecode. The user launches the Java applet from a web page, and the applet is then executed within a Java Virtual Machine (JVM) in a process separate from the web browser itself. A Java applet can appear in a frame of the web page, a new application window, Sun's AppletViewer, or a stand-alone tool for testing applets. Java applets were introduced in the first version of the Java language, which was released in 1995.

Changing the state of an object is known as an event. For example, click on button, dragging mouse etc. The java.awt.event package provides many event classes and Listener interfaces for event handling. In computing, an event is an action or occurrence recognized by software that may be handled by the software. Computer events can be generated or triggered by the system, by the user or in other ways. Typically, events are handled synchronously with the program flow, that is, the software may have one or more dedicated places where events are handled, frequently an event loop. A source of events includes the user, who may interact with the software by way of, for example, keystrokes on the keyboard. Another source is a hardware device such as a timer. Software can also trigger its own set of events into the event loop, e.g. to communicate the completion of a task. Software that changes its behavior in response to events is said to be event-driven. often with the

with the source. Once event is received by the listener, they process the event and then return. Events are supported by a number of Java packages, like java.util, java.awt and java.awt.event. Any action that user performs on a GUI component must be listened and necessary action should be taken. For example, if a user clicks on a Exit button, then we need to write code to exit the program. So for this, we need to know that the user has clicked the button. This process of knowing is called as listening and the action done by the user is called an event. Writing the corresponding code for a user action is called as Event handling. An event listener in Java is an interface that contains methods called handlers in which corresponding action code is to be written. An event class contains the information about an event.

Event source is the GUI component or model on which an event is generated or in other words an action is done.

An adapter class is an abstract class implementing a listener interface. This is essential when we don't want to write all the handlers. For example, MouseListener interface contains a lot of methods such as mousePressed(), mouseReleased().. and we want to write only one of them, we use adapter class. This class implements all the methods of an interface giving them an empty body while itself being abstract. For every action user performs, a corresponding event object is generated. This generated event object should be sent to the corresponding listener so that we can handle that event and write the code accordingly. The process of sending of event object to its corresponding listener is called as event dispatching. Events cannot be dispatched if they aren't generated and an event, except MouseEvent cannot be generated on a disabled component.

/* Applet Program & Action Event */

```
import javax.swing.*;
import java.applet.*;
import java.awt.*;
import java.awt.event.*;

public class MyApplet extends Applet implements ActionListener {
    JTextField txtEmail;
    JLabel lblMsg;
    JButton btnSubmit;

    public void init(){
        txtEmail=new JTextField(20);
        btnSubmit = new JButton("Submit");
        lblMsg = new JLabel("*****");
        setBackground(Color.YELLOW);

        add(lblMsg);
        add(txtEmail);
        add(btnSubmit);

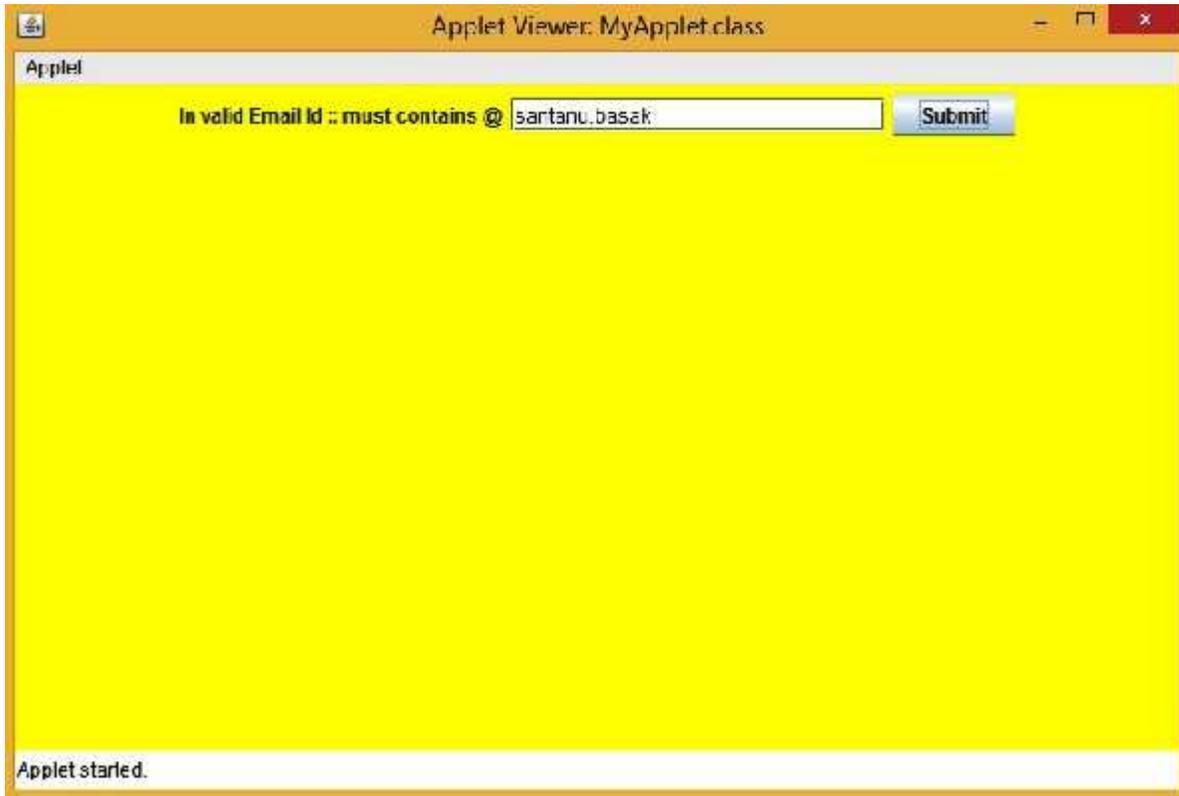
        btnSubmit.addActionListener(this);
    }
    public void actionPerformed(ActionEvent ae){
        submit();
    }
    private void submit() {
        if(!txtEmail.getText().contains("@")){
            lblMsg.setText("In valid Email Id :: must contains @");
        }else if(!txtEmail.getText().contains(".com")){
            lblMsg.setText("In valid Email Id :: must contains .com");
        }else{
            lblMsg.setText("Email Id valid");
        }
    }
}
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

INPUT: Typing santanu.basak in the textfield, press Submit button.

OUTPUT:



Experiment No: 13: Swing programming and Layout

Aim: Write a program in java to implement layout with swing.

Description:

Layout means the arrangement of components within the container. In other way we can say that placing the components at a particular position within the container. The task of layouting the controls is done automatically by the Layout Manager. The LayoutManagers are used to arrange components in a particular manner. LayoutManager is an interface that is implemented by all the classes of layout managers. There are following classes that represent the layout managers:

```
java.awt.BorderLayout
java.awt.FlowLayout
java.awt.GridLayout
java.awt.CardLayout
java.awt.GridBagLayout
javax.swing.BoxLayout
javax.swing.GroupLayout
javax.swing.ScrollPaneLayout
javax.swing.SpringLayout
```

The BorderLayout is used to arrange the components in five regions: north, south, east, west and center. Each region (area) may contain one component only. It is the default layout of frame or window. The BorderLayout provides five constants for each region:

```
public static final int NORTH
public static final int SOUTH
public static final int EAST
```

```

/* Java program BorderLayoutExample.java */

import javax.swing.*;

public class BorderLayoutExample {

    JFrame frm;
    JButton btn1;
    JButton btn2;
    JButton btn3;
    JButton btn4;
    JButton btn5;

    public BorderLayoutExample() {
        init();
    }

    private void init() {
        frm = new JFrame("Border Layout");

        btn1 = new JButton("Button 1");
        btn2 = new JButton("Button 2");
        btn3 = new JButton("Button 3");
        btn4 = new JButton("Button 4");
        btn5 = new JButton("Button 5");

        prepare();
    }

    private void prepare() {
        frm.add(btn1, "South");
        frm.add(btn2, "North");
        frm.add(btn3, "East");
        frm.add(btn4, "West");
        frm.add(btn5, "Center");

        frm.setSize(400, 400);

        frm.setVisible(true);
    }

    private void show() {
        frm.setVisible(true);
    }

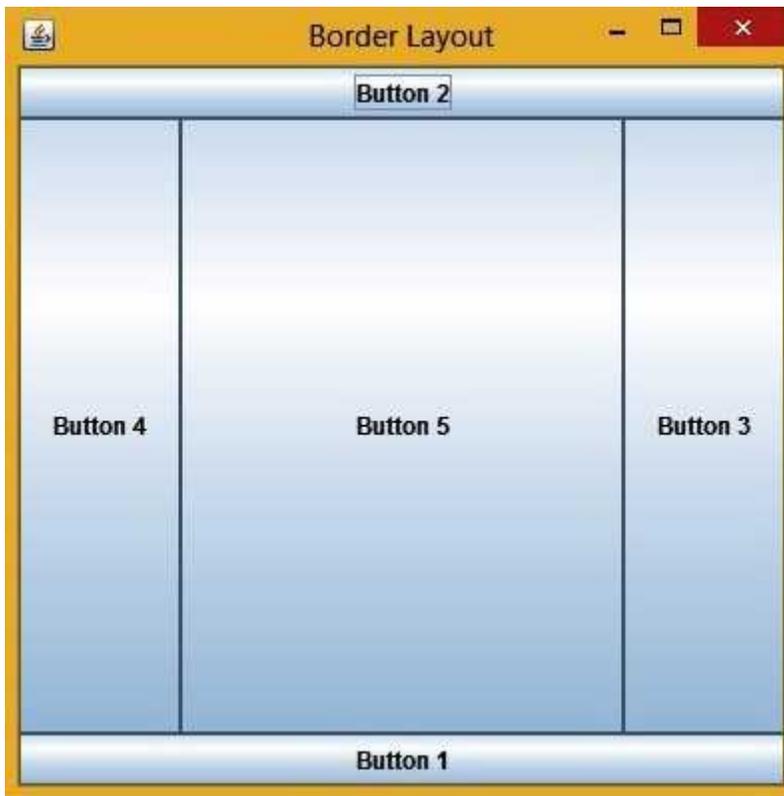
    public static void main(String args[]) {
        BorderLayoutExample reg = new BorderLayoutExample();
        reg.show();
    }
}

```

INPUT: Input is not required.

OUTPUT:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual



UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: VLSI& Microelectronics Lab

Course Code: EE694D

L-T-P scheme: 0-0-3

Course Credit: 2

Objectives: The overall course objective is to teach electrical engineering students fundamental concepts of hardware description languages and advanced techniques in digital system design. Specific objectives include the following:

1. Learn VHDL (Very high speed integrated circuit Hardware Description Language).
2. Utilize VHDL to design and analyze digital systems including arithmetic units and state machines.
3. Learn field programmable gate array (FPGA) technologies and utilize associated computer aided design (CAD) tools to synthesize and analyze digital systems.
4. Learn testing strategies and construct test-benches.
5. Conduct laboratory experiments using an FPGA based development board to prototype digital systems and to confirm the analysis done in class.
6. Prepare informative and organized lab reports that describe the methodologies employed, the results obtained, and the conclusions made in a laboratory experiment.

Learning Outcomes: The students will have a detailed knowledge of the concepts of IEEE and ANSI standard HDL. Upon the completion of Operating Systems practical course, the student will be able to:

-) **Understand** and implement basic digital logic circuits of VLSI.
-) **Model** complex digital systems at several levels of abstractions; behavioral and structural, synthesis and rapid system prototyping.
-) **Develop and Simulate** register-level models of hierarchical digital systems.
-) **Design and model** complex digital system independently or in a team
-) Carry out **implementations** of registers and counters.
-) **Simulate and synthesize** all type of digital logic circuits used in VLSI.
-) Finally **design** a CPU.

Course Contents:

Exercises that must be done in this course are listed below:

Exercise No.1: Design of basic Gates: AND, OR, NOT.

Exercise No. 2: Design of universal gates

Exercise No. 3: Design of XOR and XNOR gate.

Exercise No. 4: Design of 2:1 MUX .

Exercise No. 5: Design of 2 to 4 Decoder.

Exercise No. 6: Design of Half-Adder and Full Adder.

Exercise No. 7: Design of 8:3 Priority Encoder.

Exercise No. 8: Design of 4 Bit Binary to Grey code Converter.

Exercise No. 9: Design of all Flip-Flops.

Exercise No. 10: Design of Shift register.

Exercise No. 11: Design of ALU.

Text Book:

1. J. Bhaskar, A VHDL Primer, 3rd edition, Prentice Hall.

Recommended Systems/Software Requirements:

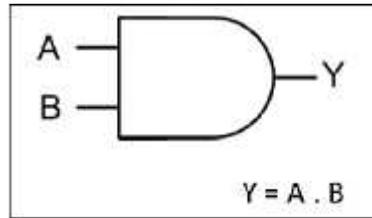
1. Intel based desktop PC with minimum of 1GHZ or faster processor with at least 1GB RAM and 8 GB free disk space.
2. Xilinx ISE14.2 software in Windows XP or Linux Operating System.

Experiment No: 1 Design of basic Gates: AND, OR, NOT.

Aim: Write VHDL code for designing AND, OR, NOT

AND Gate

2 Input AND gate		
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1



VHDL codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity AND1 is
port ( a : in STD_LOGIC; b : in STD_LOGIC; c : out STD_LOGIC ) ;
end AND1;
architecture behavioral of AND1 is
begin
process (a, b)
begin
if (a= "1" and b="1")
then c<="1"; else c<="0";
end if;
end process;
end behavioral;
```

OR Gate:

2 Input OR gate		
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1



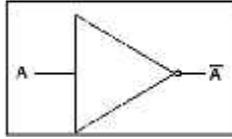
VHDL codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity OR1 is
port ( a : in STD_LOGIC; b : in STD_LOGIC; c : out STD_LOGIC ) ;
end OR1;
architecture behavioral of OR1 is
begin
process (a, b)
begin
if (a="0" and b="0") then c<= "0"; else c<="1";
end if;
end process;
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

NOT gate	
A	\bar{A}
0	1
1	0



VHDL Codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity NOT1 is
port ( a : in STD_LOGIC; c : out STD_LOGIC) ;
end NOT1;
architecture behavioral of NOT1 is
begin
process (a)
begin
if (a="0") then c<="1";
else c<="0";
end if;
end process;
end behavioral;
```

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.

Output: Student will check the output.

Discussion: Student will conclude here.

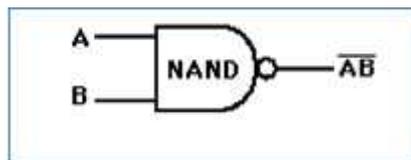
Experiment No.-2 : Design Universal gates

Aim: Write VHDL code for universal gates: NAND and NOR gate.

Apparatus: Xilinx ISE14.2 software

NAND gate:

2 Input NAND gate		
A	B	\overline{AB}
0	0	1
0	1	1
1	0	1
1	1	0



VHDL codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity NAND1 is
port ( a : in STD_LOGIC; b : in STD_LOGIC; c : out STD_LOGIC) ;
end NAND1;
architecture
behavioral of NAND1 is
```

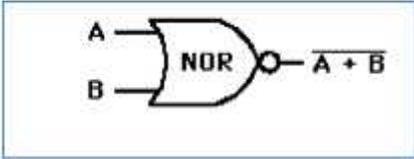
```

process (a, b)
begin
if (a= "1"and b="1")then c<= "0";
else c<="1";
end if;
end process;
end behavioral;

```

NOR gate:

2 Input NOR gate		
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0



VHDL codes:

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity NOR1 is
port (a : in STD_LOGIC; b : in STD_LOGIC; c : out STD_LOGIC) ;
end NOR1;
architecture behavioral of NOR1 is
begin
process (a, b)
begin
if (a="0"and b= "0") then c<="0";
else c<="0";
end if;
end process;
end behavioral;

```

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.

Output: Student will check the output.

Discussion: Student will conclude here.

Experiment No.-3 : Design XOR and XNOR gate

Aim: Write VHDL code for XOR and XNOR gate.

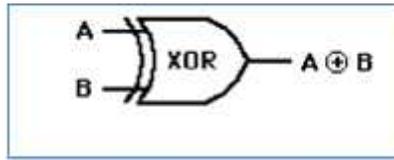
Apparatus: Xilinx ISE 14.2 software

XOR gate:

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

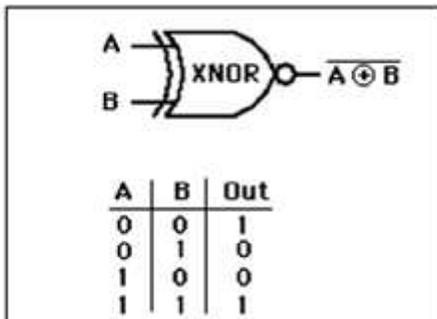
A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0



VHDL codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity XOR1 is
port ( a : in STD_LOGIC; b : in STD_LOGIC; c : out STD_LOGIC ) ;
end XOR1;
architecture behavioral of XOR1 is
begin
process (a, b)
variable (s1, s2, s3, s4:STD_LOGIC)
begin
s1:=NOT a;
s2:=NOT b;
s3:=s1 AND b;
s4:=s2 AND a;
c<=s3 OR s4;
end process;
end behavioral;
```

XNOR gate:



VHDL codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity XNOR1 is
port ( a : in STD_LOGIC; b : in STD_LOGIC; c : out STD_LOGIC ) ;
end XNOR1;
architecture behavioral of XNOR1 is
begin
```

```

variable (s1, s2, s3, s4:STD_LOGIC)
begin
s1:=NOT a;
s2:=NOT b;
s3:=a AND b;
s4:=s1 AND s2;
c<=s3 OR s4;
end process;
end behavioral;

```

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.

Output: Student will check the output.

Discussion: Student will conclude here.

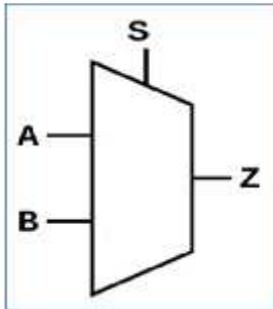
Experiment No.-4: Design 2:1 MUX

Aim: Write VHDL code for 2:1 mux using other basic gates.

Apparatus: Xilinx ISE 14.2 software

2:1 MUX:

A digital multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line.



S	Z
0	A
1	B

$$Z = A\bar{S} + B$$

VHDL Codes:

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity mux_2 to 1 is
port (a : in STD_LOGIC; b : in STD_LOGIC; s : in STD_LOGIC z : out STD_LOGIC) ;
end mux_2 to 1;
architecture behavioral of mux_2 to 1 is
begin
process (a, b, s)
begin
if (s="0")then
z<=a;
else z<=b;
end if;
end process;
end behavioral;

```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

2:1 mux using BASIC gates:

VHDL Codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity mux_2 to 1 is
port (a : in STD_LOGIC; b : in STD_LOGIC; s : in STD_LOGIC z : out STD_LOGIC) ;
end mux_2 to 1 ;
architecture behavioral of mux_2 to 1 is
begin
process (a, b, s)
variable (s1, s2, s3:STD_LOGIC)
begin s1:=NOT s; s2:=s1 AND a; s3:=s AND b; z<=s2 OR s3;
end process;
end behavioral;
```

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.

Output: Student will check the output.

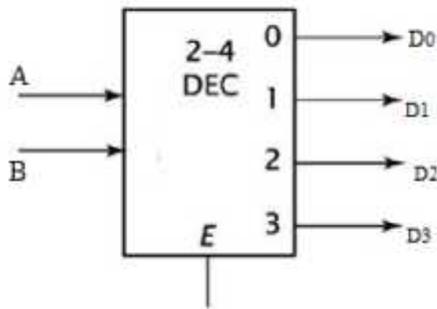
Discussion: Student will conclude here.

Experiment No.-5 :Design 2:4 Decoder

Aim: Write VHDL code for 2:4 decoder.

Apparatus: Xilinx ISE 14.2 software

2:4 decoder: A decoder is a combinational circuit that converts binary information from n inputs line to a maximum of 2ⁿ unique output lines.



E	A	B	D0	D1	D2	D3
0	X	X	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

VHDL Codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity decoder_2_to_4 is
port (a : in STD_LOGIC_VECTOR; E : in STD_LOGIC; d : out STD_LOGIC_VECTOR (3 downto 0) );
end decoder_2_to_4 ;
architecture behavioral of decoder_2_to_4 is
begin process (a)
begin
case a is when "00"=> d<="0001";
when "01"=> d<="0010";
```

```

when others=> d<="1000";
end case;
end process;
end behavioral;

```

Using DATA FLOW approach

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity decoder_2_to_4 is
port (a : in STD_LOGIC; b : in STD_LOGIC; E : in STD_LOGIC d : out STD_LOGIC_VECTOR (3 downto 0)) ;
end decoder_2_to_4;
architecture dataflow of decoder_2_to_4 is
signal(abar, bbar: STD_LOGIC)
begin
abar<=NOT a;
bbar<=NOT b;
d(0)<=abar AND bbar AND E;
d(1)<=abar AND b AND E;
d(2)<=a AND bbar AND E;
d(3)<=a AND b AND E;
end dataflow;

```

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.

Output: Student will check the output.

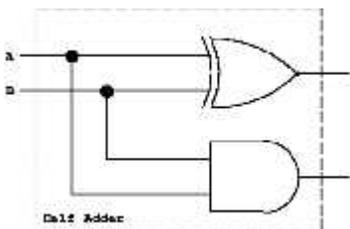
Discussion: Student will conclude here.

Experiment No.-6: Design Half adder and Full adder

Aim: Write VHDL code for Half-adder, full-adder.

Apparatus: Xilinx ISE 14.2 software

Half-adder:



A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

VHDL codes:

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
Entity half_adder is
port (a : in STD_LOGIC; b : in STD_LOGIC; s : out STD_LOGIC; c : out STD_LOGIC);
end half_adder;

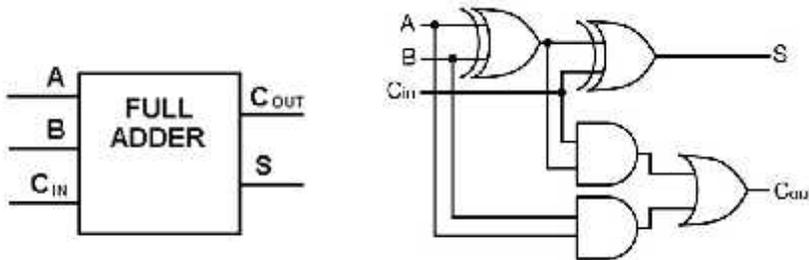
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
begin
  process (a,b)
  begin
    if (a="0" and b="0") then s<="0"; c<="0";
    elsif (a="1" and b="1") s<="0"; c<="1";
    else s<="1"; c<="0";
    end if;
  end process;
end behavioral;
```

Full-Adder:



A	B	C	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

VHDL codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
Entity full_adder is
port ( a : in STD_LOGIC_VECTOR (0 to 2); s : out STD_LOGIC_VECTOR (0 to 1));
end full_adder;
architecture behavioral of full_adder is
```

```
begin
  process (a)
  begin
    case a is
      when "000" => s<="00";
      when "001" => s<="10";
      when "010" => s<="10";
      when "011" => s<="01";
      when "100" => s<="10";
      when "101" => s<="01";
      when "110" => s<="01";
```

end case;
end process;
end behavioral;

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.

Output: Student will check the output.

Discussion: Student will conclude here.

Experiment No-7 : Design 3:8 Decoder

Aim: Write VHDL code for 3:8decoder.

Apparatus: Xilinx ISE 14.2 software

3:8 decoder

Inputs			outputs							
A	B	C	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

VHDL Codes:

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL; use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity decoder_3_to_8 is
port (a : in STD_LOGIC_VECTOR (2 downto 0);
d : out STD_LOGIC_VECTOR (7 downto 0);
end decoder_3_to_8;
architecture Behavioural of decoder_3_to_8 is
begin
proess(a)
begin
case a is
when "000"=> d<="00000001";
when "001"=> d<="00000010";

```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
when "011"=> d<="00001000";
when "100"=> d<="00010000";
when "101"=> d<="00100000";
when "110"=> d<="01000000";
when others=> d<="10000000";
end case;
end process;
end Behavioural;
```

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.

Output: Student will check the output.

Discussion: Student will conclude here.

Experiment No.-8 : Design 8:3 priority encoder

Aim: Write VHDL code for 8:3priority encoder.

Apparatus: Xilinx ISE 8.1 software

8:3 priority encoder:

An encoder is a digital circuit that performs inverse operation of decoder. An encoder has 2^n input lines and n output lines. The output lines generate the binary code corresponding to the input value.

Truth-table for 8:3priority encoder

Inputs								outputs		
A7	A6	A5	A4	A3	A2	A1	A0	D2	D1	D0
0	0	0	0	0	0	0	0	X	X	X
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	X	0	0	1
0	0	0	0	0	1	X	X	0	1	0
0	0	0	0	1	X	X	X	0	1	1
0	0	0	1	X	X	X	X	1	0	0
0	0	1	X	X	X	X	X	1	0	1
0	1	X	X	X	X	X	X	1	1	0
1	X	X	X	X	X	X	X	1	1	1

VHDL Codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
```

```
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

```

entity p_encoder_8_to_3 is
port (a : in STD_LOGIC_VECTOR (7downto 0);
d : out STD_LOGIC_VECTOR (2downto 0));
endp_encoder_8_to_3;

architecture behavioral of p_encoder_8_to_3 is
begin
process (a)
begin
case a is
when "00000001"=> d<="000";
when "0000001X"=> d<="001";
when "000001XX"=> d<="010";
when "00001XXX"=> d<="011";
when "0001XXXX"=> d<="100";
when "001XXXXX"=>d<="101";
when "01XXXXXX"=> d<="110";
when "1XXXXXXX"=> d<= "111";
when others=> d<="XXX";
end case;
end process;
end behavioral;

```

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.

Output: Student will check the output.

Discussion: Student will conclude here.

Experiment No.-8: Design Binary to Gray converter

Aim: Design of 4 Bit Binary to Grey code Converter.

Apparatus: Xilinx ISE 14.2 software

Binary top gray converter:

The binary to grey converter is a combinational circuit that takes binary number as input and converts it into grey code. Grey code differs from the preceding and succeeding number by a single bit.

VHDL Codes:

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL; use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

```

```

entity b2g is
port (b : in std_logic_vector (3 downto0);
g : out std_logic_vector (3 downto 0));
end b2g;

architecture behavioral of b2g is
begin

```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
begin
case b is
when "0000" => g<= "0000";
  when "0001" => g<= "0001";
when "0010" => g<= "0011";
when "0011" => g<= "0010";
  when "0100" => g<= "0110";
when "0101" => g<= "0111";
when "0110" => g<= "0101";
  when "0111" => g<= "0100";
when "1000" => g<= "1100";
when "1001" => g<= "1101";
  when "1010" => g<= "1111";
when "1011" => g<= "1110";
  when "1100" => g<= "1010";
when "1101" => g<= "1011";
  when "1110" => g<= "1001";
when others => g<= "1000";
end case;
end process;
end behavioral;
```

Data flow model for binary to grey code converter:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL; use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity bin2grey_conv is
  port (b : in std_logic_vector (3 downto 0);
        g : out std_logic_vector (3 downto 0));
end bin2grey_conv;
architecture dataflow of bin2grey_conv is
begin
g(3)<=b(3);
g(2)<=(b(3)) xor (b(2));
g(1)<=b(2) xor b(1);
g(0)<=b(1) xor b(0);
end dataflow;
```

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.

Output: Student will check the output.

Discussion: Student will conclude here.

Experiment No.-9: Design flip-flops

Aim: Study all Flip-flops using VHDL

Apparatus: Xilinx ISE 14.2 software

(1) S-R flip-flop:

S	R	Q_{n+1}
0	0	Q _n
0	1	0
1	0	1
1	1	

VHDL Codes:

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL; use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL; entity flipflop_SR is
port (s, r, clk, rst : in std_logic; q : out std_logic);
end flipflop_SR;
architecture behavioral of flipflop_SR is
begin
process (s, r, clk, rst)
begin
if (clk= "1" and clk'event) then if (rst= "1") then
q<= "0";
elsif (rst= "0") then
q<= "1";
elsif (s= "0" and r= "0" and rst= "0") then
q<=q;
elsif (s= "0" and r= "1" and rst= "0") then
q<= "0";
elsif (s= "1" and r= "0" and rst= "0") then
q<= "1";
elsif (s= "1" and r= "1" and rst= "0") then
q<= "U";
end if;
end if;
end process;
end behavioral;

```

(2) J-K flip-flop:

J	K	Q_{n+1}

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

0	1	0
1	0	1
1	1	Not Q_n

VHDL Codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL; use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL; entity flipflop_JK is
port (j, k, clk, rst : in std_logic; q : inout std_logic);
end flipflop_JK;
architecture behavioral of flipflop_JK is
begin
process (j, k, clk, rst)
begin
if (clk= "1" and clk'event) then if (rst= "1") then
q<= "0";
elsif (rst= "0") then
q<= "1";
elsif (j= "0" and k= "0" and rst= "0") then
q<=q;
elsif (j= "0" and k= "1" and rst= "0") then
q<= "0";
elsif (j= "1" and k= "0" and rst= "0") then
q<= "1";
elsif (j= "1" and k= "1" and rst= "0") then
q<= NOT q;
end if;
end if;
end process;
end behavioral;
```

(3) D flip-flop:



D	Q_{n+1}
0	0

VHDL Codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL; use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL; entity flipflop_Dis
port (d,clk, rst : in std_logic; q : inoutstd_logic);
end flipflop_D;
architecture behavioral of flipflop_D is
begin
process (d,clk, rst)
begin
if (clk= "1" and clk'event) then if (rst= "1") then
q<= "0";
else
q<=q;
end if;
end if;
end process;
end behavioral;
```

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.

Output: Student will check the output.

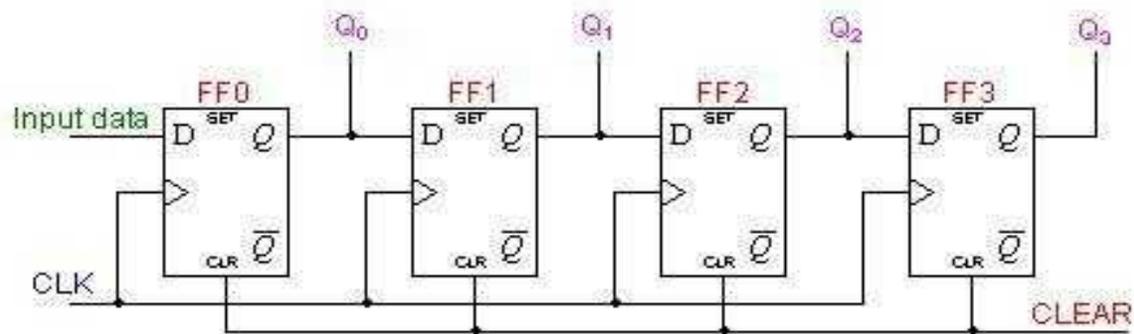
Discussion: Student will conclude here.

Experiment No.-10 : Design Shift register

Aim: Design of 8-bit shift register using VHDL.

Apparatus: Xilinx ISE 14.2 software

Shift register



VHDL Codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity eftshift is
port (a : inoutbit_vector (0 to 7);
r, l, rst, load, clk : in bit;
q : out bit_vector (0 to 7));
end leftshift;
architecture behavioral of leftshift is
begin
process (load, rst, a, clk)
begin
if (rst = '1') then
q <= (others => '0');
else if (load = '1') then
q <= a;
else if (clk = '1' and clk'event) then
q <= q << 1;
end if;
end if;
end process;
```

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

```
elsif(load= "0") then if (rst= "1") then q<= "00000000";
else
if (l= "1") then
q<=a slll;
end if;
if (r= "1") then
q<= a srl;
end if;
end if;
end if;
end if;
end process;
end behavioral;
```

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.

Output: Student will check the output.

Discussion: Student will conclude here.

Experiment No. 11: Design ALU

Aim: Write VHDL program to perform Arithmetic Logic Unit (ALU) operation.

Apparatus: Xilinx ISE 14.2 software

ALU:

An ALU performs arithmetic and logical operation. It receives data from register file and perform operations on it given by control signals generated through control unit.

Sel	Unit	Operation
000	Arithmetic Unit	$z \leq x$
001		$z \leq x+1$
010		$z \leq y$
011		$z \leq x+y$
100	Logic Unit	$z \leq \text{not } x$
101		$z \leq x \text{ and } y$
110		$z \leq x \text{ or } y$
111		$z \leq x \text{ xor } y$

VHDL codes:

Entity ALU is

```
Port(x,y : in std_logic_vector(0 to 7);
```

```
sel : in std_logic_vector (0 to 2);
```

```
z : out std_logic_vector (0 to 7));
```

```
end ALU;
```

architecture dataflow of ALU is

```
signal arith, logic : std_logic_vector (0 to 7);
```

```
begin
```

```
with sel (0 to 1) select
```

```
arith <= x when "00"; x+1 when "01"; y when "10"; x+y when others;
```

```
with sel (0 to 1) select
```

```
logic <= not x when "00"; x and y when "01"; x or y when "10";
```

```
x xor y when others;
```

```
with sel (2) select
```

```
z <= arith when "0"; logic when others;
```

```
end dataflow;
```

Output: Student will check the output.

Discussion: Student will conclude here.