UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

Lecture-wise Plan

Subject Name: Principles of Management **Subject Code-HU601**
Year: 3rd Year **Semester: Sixth**

Module Number	Topics	Number of Lectures
1	**Introduction:**	**7L**
	Basic conceptsof 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, Fianancialststements, 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, Humantities Dept.

UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

Lecture-wise Plan

Subject Name: Database Management System — Subject Code-CS601

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 & MANAGEMENT,JAIPUR

Lecture-wise Plan

Subject Name: Database Management System Subject Code-CS601
Year:3rd Year 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: Computer Network　　　　　　Subject Code-CS602

Year: 3rd Year　　　　　　Semester: Sixth

Module Number	Topics	Number of Lectures
1	**Data Communication Fundamentals:**	**10L**
	1. Layered Network Architecture; Data and Signal; Guided Transmission Media; Unguided Transmission Media; Transmission Impairments and Channel Capacity; Transmission of Digital Signal; Analog Data to Analog Signal; Digital Data to Analog Signal; Multiplexing of Signals: The telephone system and DSL technology; Cable MODEM and SONET	10
2	**Data Link control:**	**6L**
	1. Interfacing to the media and synchronization; Error Detection and Correction; Flow and Error control; Data Link Control.	10
3	**Switching Communication Networks:**	**8L**
	1. Circuit switching; Packet switching; Routing in packet switched networks; Congestion control in packet switched networks; X.25; Frame Relay; Asynchronous Transfer Mode Switching (ATM).	8
4	**Broadcast communication networks:**	**10L**
	1. Network Topology; Medium Access Control Techniques; IEEE CSMA/CD based LANs; IEEE Ring LANs; High Speed LANs – Token Ring Based; High Speed LANs – CSMA/CD based; Wireless LANs; Bluetooth; Cellular Telephone Networks; Satellite Networks.	10
5	**Internetworking:**	**6L**
	1. Internetworking Devices; Internet Protocols; TCP/IP; Transport and Application layer protocols. Network Security: Cryptography; Secured Communication; Firewalls.	6
Total Number Of Hours = 40		

Assignments:

Module-1:

1. Write down the functions of OSI Layers
2. What will be SNR value in case of noiseless channel?
3. Define Bandwidth? Create the relationship between Bit Rate and Baud Rate?
4. Write down the names of network impairments?
5. Write down the features and basic components of a computer network
6. What kind of topology is well suited for university or college environment?
7. Why we need layered architecture?
8. What will be the channel capacity of a noisy channel having SNR value= 20dB and Bandwidth=3 KHz?

Module-2:

1. What is the significance of sequence number in Stop & Wait ARQ protocol?
2. Discuss Stop & Wait ARQ with 010101 bit sequence?
3. In Selective-Repeat ARQ, sender window size > 2m–1." Is it correct? Justify.
4. Suppose a sender is using sliding window protocol of window size 15. What will be the window status for the following occurrence? Sender has sent packets 0 to 11 and has received NAK 6.
5. Define ALOHA? Differentiate between Pure and Slotted ALOHA.

Module-3:

1. Differentiate between circuit switching and packet switching.
2. Write short notes on the following topic:
   A. Frame Relay
   B. X.25
   C. ATM
3. Why packet switching is connection less?

Module-4:

1. Discuss CSMA/CA with the help of a flowchart.
2. Why CSMA/CD is not implemented in WLAN?
3. Describe 802.3 header formats. Why padding is required?
4. Describe Bluetooth Architecture.
5. Differentiate between Token Ring and Token Bus.

Module-5:

1. What is distance vector routing protocol? What is the difference between RIP and EGP?
2. Distinguish between gateway and bridge. What is transparent bridge?
3. A network has subnet mask 255.255.255.224 Determine the maximum or number of Host in this network. Also determine the broadcast address of this network.
4. Compare IPv4 and IPv6
5. What is the purpose of subnetting? Find the netid and the host id of the following IP address
   A. 192.167.78.1
   B. 10.10.10.10
   C. 189.32.1.34
6. What is CIDR? Define NAT with proper example?

7. Write the differences between ARP and RARP?
8. Write the differences between TCP and UDP?
9. Differentiate Leaky Bucket Algorithm from Token Bucket Algorithm.
10. Why IP address is 32 Bit? How we need so many addresses? Compare IP address, Mac Address, Port address and Socket address?
11. Define Count to infinity problem? Which routing algorithm faces this problem?
12. Write the short notes on the following
    A. DNS
    B. FTP
    C. EMAIL
    D. MIME
    E. POP3
    F. SMTP
13. How are ‘iterative query resolution and ‘recursive query resolution different from each other in the context of DNS?
14. What do you understand by data privacy? How can authentication, integrity and non-repudiation be implemented by Digital Signature?
15. Define Firewall? Discuss all types of Firewall.

UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

Lecture-wise Plan

Subject Name: Operating System **Subject Code-CS603**
Year: 3rd Year **Semester: Sixth**

Module Number	Topics	Number of Lectures
1	**Introduction:**	**4L**
	1. Introduction toOS.Operatingsystemfunctions,evaluationof O.S., .	1
	2. Evaluation of O.S., Different types of O.S.	1
	3. batch, multi-programmed, time-sharing	1
	4. DifferenttypesofO.S.:real-time,distributed,parallel	1
2	**System Structure:**	**3L**
	1. Computer system operation, I/O structure, storage structure	1
	2. storage hierarchy, different types of protections, operating system structure (simple, layered, virtual machine)	1
	3. O/S services, system calls.	1
	Process Management:	
3.	**Processes:**	**3L**
	1. Concept of processes, process scheduling, operations on processes	2
	2. co-operating processes, inter-process communication.	1
4	**Threads:**	**2L**
	1. overview, benefits of threads	1
	2. user and kernel threads.	1
5	**CPU scheduling:**	**3L**
	1. scheduling criteria, preemptive & non-preemptive scheduling	1
	2. scheduling algorithms (FCFS, SJF, RR, priority),	1
	3. algorithm evaluation, multi-processor scheduling.	1
6	**Process Synchronization:**	**4L**
	1. background, critical section problem, critical region	2
	2. synchronization hardware, classical problems of synchronization, semaphores.	2
7	**Deadlocks:**	**4L**
	1. System model, deadlock characterization	1
	2. Methods for handling deadlocks, deadlock prevention	1
	3. Deadlock avoidance, deadlock detection	1
	4. Recovery from deadlock	1

8	**Storage Management:**	
	Memory Management:	**4L**
	1. Background, logical vs. physical address space	1
	2. Swapping, contiguous memory allocation	1
	3. Paging, segmentation, segmentation with paging	2
9	**Virtual Memory:**	**3L**
	1. Background, demand paging, performance	1
	2. Page replacement, page replacement algorithms (FCFS, LRU)	1
	3. Allocation of frames, thrashing.	1
10	**File Systems:**	**4L**
	1. File concept, access methods, directory structure	1
	2. File system structure, allocation methods (contiguous, linked, indexed)	1
	3. Free-space management (bit vector, linked list, grouping)	1
	4. Directory implementation (linear list, hash table), efficiency & performance.	1
11	**I/O Management:**	**4L**
	1. I/O hardware, polling, interrupts	1
	2. DMA, application I/O interface (block and character devices, network devices clocks and timers, blocking and nonblocking I/O)	2
	3. kernel I/O subsystem (scheduling, buffering, caching, spooling and device reservation, error handling), performance	1
12	**Disk Management:**	**3L**
	1. disk structure, disk scheduling (FCFS, SSTF, SCAN,C-SCAN) ,disk reliability,	2
	2. disk formatting, boot block, bad blocks.	1
13	**Protection & Security:**	
	1. Goals of protection, domain of protection, Security problem, authentication	1
	2. One time password, program threats, System threats, threat monitoring, encryption	1
Total Number Of Hours = 41		

Faculty In-Charge HOD, CSE Dept.

UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

Lecture-wise Plan

Assignment:

Module-1(Introduction):

1. Explain real-time anddistributed Operating system.
2. Why time-sharing is required in OS?

Module-2 (System Structure):

1. What do you mean by storage hierarchy? Explain with diagram.
2. Explain the different types of protections.

Module-3(Process Management):

1. Consider a set of five processes whose arrival time, CPU time needed and priority are giving below.

Process Name	Arrival Time(in ms)	CPU Time (in ms)	Priority
P1	0	10	5
P2	0	5	2
P3	2	3	1
P4	5	20	4
P5	10	2	3

   a. If the scheduling policy is FCFS, find the average waiting time.
   b. If the scheduling policy is SJF, find the average waiting time without pre-emption.
   c. If the scheduling policy is SJF, find the average waiting time with pre-emption.
   d. If the scheduling policy is priority scheduling with pre-emption, find the average waiting time.
   e. If the scheduling policy is Round Robin scheduling and the quantum time is 2 ms, find the average waiting time.

2. Explain inter-process communication system

Module-4(Thread):

1. What are the benefits of thread in OS?
2. Explain user thread and kernel thread.

Module-5 (CPU scheduling):

1. What are the difference between preemptive& non-preemptive scheduling?
2. Explain FCFS scheduling algorithm.

Module-6 (Process Synchronization):

1. Explain critical section problem.
2. What do you mean by semaphore?

Module-7 (Deadlocks):

1. Explain the methods for handling deadlocks. What do you mean by deadlock prevention?
2. How deadlocks are detected? Explain the Deadlock avoidance technique.

Module-8 (Memory Management):

1. Explain contiguous memory allocation in OS.
2. Define Paging, segmentation and segmentation with paging.

Module-9 (Virtual Memory):

1. Define demand paging.
2. Explain how Page replacement is occurs? Describe any one page replacement algorithm.

Module-10 (File Systems):

1. How free spaces are managed in file system?
2. How Directory is implemented in file system?

Module-11 (I/O Management):

1. What is DMA?
2. Explain kernel I/O subsystem.

Module-12 (Disk Management):

1. Explain boot block, bad blocks.
2. Explain disk formatting with proper diagram.

Module-13 (Protection & Security):

1. What do you mean by program threats and System threats?
2. Explain domain of protection.

UNIVERSITY OF ENGINEERING & MANAGEMENT,JAIPUR

Lecture-wise Plan

Subject Name: Information Theory & Coding **Subject Code-CS-604A**
Year: 3rd Year, **Semester: Sixth**

Module Number	Topics	Number of Lectures
1.	**Source Coding**	**7L**
	1. uncertainty and information, average mutual information and entropy	3
	2. information measures for continuous random variables	2
	3. source coding theorem, Huffman codes	2
2.	**Channel Capacity And Coding**	**7L**
	1. Channel models, channel capacity	3
	2. channel coding, information capacity theorem	2
	3. The Shannon limit.	2
3.	**Linear And Block Codes For Error Correction**	**6L**
	1. Matrix description of linear block codes, equivalent codes	2
	2. parity check matrix, decoding of a linear block code	2
	3. perfect codes, Hamming codes	2
4.	**Cyclic Codes**	**7L**
	1. Polynomials, division algorithm for polynomials	2
	2. a method for generating cyclic codes	2
	3. matrix description of cyclic codes, Golay codes.	3
5.	**BCH Codes**	**6L**
	1. Primitive elements, minimal polynomials	2
	2. generator polynomials in terms of minimal polynomials	2
	3. examples of BCH codes	2
6.	**Convolutional Codes**	**7L**
	1. Tree codes, trellis codes, polynomial description of convolutional codes, distance notions for convolutional codes	3
	2. the generating function, matrix representation of convolutional codes, decoding of convolutional codes	2
	3. distance and performance bounds for convolutional codes, examples of convolutional codes, Turbo codes, Turbo decoding	2
Total Number Of Hours = 40		

Faculty In-Charge HOD, CSE Dept.

Assignment :

Module -1 (Source Coding)

Module -2 (Channel Capacity And Coding)

Module -3 (Linear And Block Codes For Error Correction)

Module -4 (Cyclic Codes)

Module -5 (BCH Codes)

Module -6 (Convolutional Codes)

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Computer Graphics Subject Code-CS604B
Year: 3rd Year Semester: Sixth

Module Number	Course Details	Number of Lectures
UNIT 1 1	**Introduction to computer graphics & graphics systems:** • Overview of computer graphics, representing pictures, preparing, presenting & interacting with pictures for presentations • Visualization & image processing; RGB color model, direct coding, lookup table • storage tube graphics display, Raster scan display, 3D viewing devices, Plotters, printers, digitizers, Light pens etc. • Active & Passive graphics devices	**4LH**
2	**Scan conversion:** • Points & lines, Line drawing algorithms • DDA algorithm, Bresenham's line algorithm, Circle generation algorithm • Ellipse generating algorithm; scan line polygon, fill algorithm • boundary fill algorithm, flood fill algorithm.	**6LH**
UNIT 2 3	**2D transformation & viewing:** • Basic transformations: translation, rotation, scaling; • Matrix representations & homogeneous coordinates, transformations between coordinate systems; • Reflection shear; Transformation of points, lines, parallel lines, intersecting lines. • Cohen and Sutherland line clipping, Sutherland-Hodgeman Polygon clipping, Cyrus-beck clipping method	**7LH**
4	**3D transformation & viewing:** • 3D transformations: translation, rotation, scaling & other transformations • Rotation about an arbitrary axis in space • Clipping, view port clipping, 3D viewing.	**7LH**
UNIT 3	**Curves and Hidden surfaces:**	

5	• Curve representation, surfaces, designs, Bezier curves • B-spline curves, end conditions for periodic B-spline curves, rational B-spline curves. • Depth comparison, Z-buffer algorithm, Back face detection • BSP tree method, the Painter's algorithm, scan-line algorithm • Hidden line elimination, wire frame methods , fractal - geometry.	**7LH**
6	**Introduction to Ray-tracing:** • Human vision and color • Lighting, Reflection and transmission models.	**8LH**
	Total Number Of Hours = 44	

Faculty In-Charge HOD, CSE Dept.

Assignment:

Module-1(Introduction to computer graphics & graphics systems):

1. A monochromatic graphic display system has 525 scan lines with an aspect ratio 9:16. If each pixel is displaceable in 512 shades
   (i) How many pixels are displayed on the screen?
   (ii) What is the picture storage memory size?
2. What do you mean by window and viewport? Describe the relationship for window to viewport mapping.

Module-3 (2D transformation & viewing):

1. Prove that successive scaling is multiplicative
2. Write down mid-point ellipse drawing algorithm
3. a rectangular 2D clipping window has its lower left corner at (100,10) and upper right corner at (160,40). Find visible portion of lines A(50,0), B(120,30) and C(120,20), D(140,80) using mid point sub division algorithm.

Module-4 (Curves and Hidden surfaces)

1. Write down the procedure for drawing B-spline curves and also write down its property
2. Derive the condition to be satisfied when joining two Bezier curves with second order continuity at the joints.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Multimedia Technology **Subject Code-CS604C**
Year: 3rd Year **Semester: Sixth**

Module Number	Topics	Number of Lectures
1	**Introduction:**	**2L**
	Multimedia today, Impact of Multimedia, .	1
	Multimedia Systems, Components and Its Applications	1
2	**Text and Audio:**	**6L**
	Text: Types of Text, Ways to Present Text, Aspects of Text Design, Character, Character Set, Codes, Unicode, Encryption;	2
	Audio: Basic Sound Concepts, Types of Sound	1
	Digitizing Sound, Computer Representation of Sound (Sampling Rate, Sampling Size, Quantization)	1
	Audio Formats, Audio tools, MIDI.	2
3.	**Image and Video:**	**8L**
	Image: Formats, Image Color Scheme, Image Enhancement.	2
	Analogue and Digital Video, Recording Formats and Standards (JPEG, MPEG, H.261)	3
	Transmission of Video Signals, Video Capture, and Computer based Animation.	3
4	**Synchronization:**	**4L**
	Temporal relationships	1
	synchronization accuracy specification factors	2
	Quality of service	1
5	**Storage models and Access Techniques:**	**4L**
	Magnetic media, optical media, file systems (traditional, multimedia)	2
	Multimedia devices– Output devices, CD-ROM, DVD, Scanner, CCD	2
6	**Image and Video Database:**	**8L**
	Image representation, segmentation	2
	Similarity based retrieval, image retrieval by color, shape and texture.	2
	indexing- k-d trees, R-trees, quad trees	2
	Case studies- QBIC, Virage.Video Content, querying, video segmentation, indexing	2

7	**Document Architecture and Content Management:**	**9L**
	Content Design and Development, General Design Principles	2
	Hypertext: Concept, Open Document Architecture (ODA),Multimedia and Hypermedia Coding Expert Group (MHEG),	2
	Standard Generalized Markup Language (SGML), Document Type Definition (DTD), Hypertext Markup Language(HTML) in Web Publishing	3
	Case study of Applications	2
8	**Multimedia Applications:**	**4L**
	Interactive television, Video-on-demand, Video Conferencing	1
	Educational Applications, Industrial Applications,	1
	Multimedia archives and digital libraries	1
	Media editors.	1
Total Number Of Hours = 45		

Faculty In-Charge HOD, CSE Dept.

Assignment:

Module-1:

1. Define what is multimedia.
2. Discuss the effects of multimedia in your daily life.
3. Identify five multimedia components.
4. Explain why multimedia is so powerful to increase human-computer interaction.
5. Examine multimedia applications in several areas.

Module-2:

1. Describe how to use text-related element in multimedia design correctly.
2. Compare and contrast between bitmap and vector graphic.
3. Examine how to find graphics and about editing software.

Module-3:

1. Discuss about audio digitization, audio file format and audio software.
2. Explain about digital video standards, formats and technology.
3. Examine basic principles behind animation and techniques
4. Describe the development in multimedia communication.
5. List out five basic type of communication networks.

6. Examine technological advancement and challenges in communication.

7. Discuss why do we need compression.
8. Differentiate between lossy and lossless compression.
9. Explain several image and video compression techniques.

Module-4:
1. Discuss various Quality of Service parameters.
2. What do you mean by Temporal relationships

Module-5:
1. Discuss about various types of multimedia devices.

Module-6:
1. Write the differences between k-d trees, R-trees, quad trees
2. What do you mean video segmentation?

Module-7:
1. Describe about the characteristics of web-based system
2. Examine the examples of web-based multimedia applications.
3. Write Short notes on the following topics
   A. SGML
   B. ODA
   C. MHEG
   D. DTD
   E. HTML

Module-8:
1. Write Short notes on the following topics
   A. Interactive TV
   B. Video on Demand and Video conferencing.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Operations Research **Subject Code-CS605A**
Year: III YEAR **Semester: Sixth**

MODUL E NO.	TOPICS	NUMBER OF LECTURES (35)
	Introduction	**1L**
1	**Linear Programming (LP):**	**7L**
	1. Formulation of LP Problems,Graphical solutions of two decision variable problems	2
	2. General form of LP model, Simplex method	3
	3. Big-M Method	2
	Transportation & Assignment Problems:	**6L**
	1. Nature of a transportation or distribution problem, North-West Corner initial solution	2
	2. Concept of dummy source or destination, Vogel's approximation method	2
	3. Nature of an Assignment problem:-Tabular representation Hungarian method	2
2	**Network Analysis:**	**6L**
	1. Shortest Path; Floyd Algorithm, Maximal Flow Problem (Ford-Fulkerson)	4
	2. PERT-CPM (Cost Analysis, Crashing, Resource Allocation excluded)	2
3	**Inventory Control**	**3L**
	Introduction to EOQ Models of Deterministic and Probabilistic, Safety Stock; Buffer Stock.	3
4	**Waiting line problems:**	**7L**
	1. Structure of a waiting line System distribution service times	1
	2. Queuediscipline,steadystageoperation;SinglechannelmodelwithPoissonarrivalsandexponentialservicetime	2
	3. MultiplechannelmodelwithPoissonarrivalandexponentialservicetimes;	2

	4. Singlechannelmodelwith Poissonarrivalsandarbitraryservicetime(M/G/1); Economic analysis of waiting line	2
5	**Game Theory**	**5L**
	1. Introduction;2-PersonZero-sumGame;SaddlePoint;Mini-MaxandMaxi – Min Theorems (statementonly) and	3
	problems; Games without Saddle Point; Graphical Method; Principle of Dominance	2

Faculty In-Charge HOD, Maths Dept.

Assignments:

Unit-1(Linear Programming Problem):

1. The manager of the milk dairy decides that each cow should get at least 15 units, 20 units and 24 units of nutrients A, B and C daily respectively. Two varieties of feed are available. In feed of variety 1 and variant 2 the contents of nutrient A, B and C are respectively 1 and 3; 2 and 2; 3 and 2 units per kg. The costs of verities 1 and 2 are respectively Rs. 2 and Rs. 3 per kg. How much of feed of each variety should be purchased to feed a cow daily so that the expenditure is minimized. Formulate as an LPP.
2. Solve the LPP using graphical method

$$\text{Max } 2x_1 - 4x_2$$
$$\text{Subject to } \begin{aligned} &3x_1 + 5x_2 \geq 15, \\ &4x_1 + 9x_2 \leq 36, \\ &x_1, x_2 \geq 0 \end{aligned}$$

3. Solve the LPP using graphical method

$$\text{Min } 4x_1 + 6x_2$$
$$\text{Subject to } \begin{aligned} &6x_1 + 5x_2 \geq 20, \\ &5x_1 + x_2 \geq 10, \\ &x_1 + 4x_2 \geq 8, \\ &x_1, x_2 \geq 0 \end{aligned}$$

4. Solve the LPP using simplex method

$$\text{Min } -12x_1 - 15x_2$$
$$\text{Subject to } \begin{aligned} &4x_1 + 3x_2 \leq 12, \\ &2x_1 + 5x_2 \leq 10, \\ &x_1, x_2 \geq 0 \end{aligned}$$

5. Solve the LPP using Artificial variables technique

$$\text{Min } 3x_1 + 2x_2$$

$$\text{Subject to} \quad \begin{aligned} & x_1 + x_2 \geq 2, \\ & x_1 + 3x_2 \leq 3, \\ & x_1 - x_2 = 1, \\ & x_1, x_2 \geq 0 \end{aligned}$$

Transportation Problems:-

1. Find the starting solution in the following transportation problems by
   (i) North-West Corner Method (ii) Least-Cost Method
   (i) Vogel's Approximation Method.

(A)

	D_1	D_2	D_3	D_4	Supply
S_1	19	30	50	10	7
S_2	70	30	40	60	9
S_3	40	8	70	20	18
Demand	5	8	7	14	

	D_1	D_2	D_3	D_4	Supply
S_1	3	7	6	4	5
S_2	2	4	3	2	2
S_3	4	3	8	5	3
Demand	3	3	2	2	

(B)

	D	E	F	G	Available
A	11	13	17	14	250
B	16	18	14	10	300
C	21	24	13	10	400
Demand	200	225	275	250	

		Destination				
source		D_1	D_2	D_3	D_4	Supply
	S_1	20	25	28	31	200
	S_2	32	28	32	41	180
	S_3	18	35	24	32	110
	Demand	150	40	180	170	

Assignment Problems

1. A departmental head has four subordinates, and four tasks to be performed. The subordinates differ in efficiency, and the tasks differ in their intrinsic difficulty. His estimate, of the time each man would take to perform each task, is given in the table below:

Tasks	Men			
	E	F	G	H
A	18	26	17	11
B	13	28	14	26
C	38	19	18	15
D	19	26	24	10

How should the tasks be allocated, one to a man, so as to minimize the total man-hours by using Hungarian assignment method?

2. A pharmaceutical company is producing a single product and is selling it through five agencies located in different cities. All of a sudden, there is a demand for the product in another five cities not having agency of the company. The company is faced with the problem of deciding on how to assign the existing agencies to dispatch the product to needy cities in such a way that the travelling distance is minimized. The distance between the surplus and deficit cities (in Km) is given in the following table:

		Deficit cities				
Surplus cities		**a**	**b**	**c**	**d**	**e**
	A	**85**	**75**	**65**	**125**	**75**
	B	**90**	**78**	**66**	**132**	**78**
	C	**75**	**66**	**57**	**114**	**69**
	D	**80**	**72**	**60**	**120**	**72**
	E	**76**	**64**	**56**	**112**	**68**

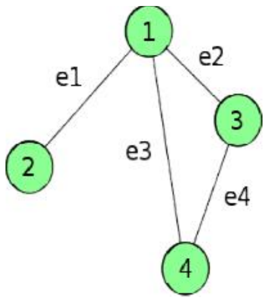
Determine the optimum assignment schedule.

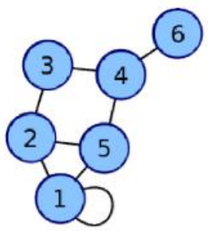
Unit 2: Network Analysis:

1. Draw a graph with 9 vertices having degree 1,5,2,7,10,8.
2. Determine the number of edges with 6 nodes, two of degree four and 4 of degree 2.
3. Find the maximum number of vertices in a connected graph having 17 edges.
4. Construct the graph corresponding to the following adjacency matrix:

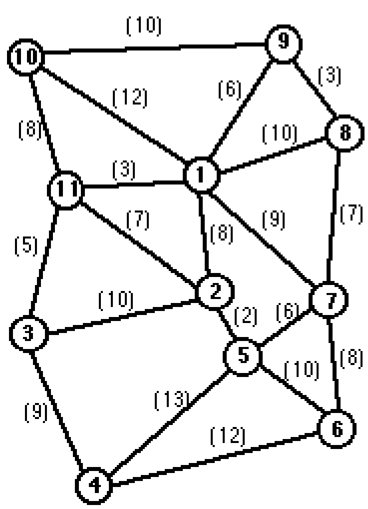
$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

5. Write down the adjacency matrix of the following graphs:





6. Find by kruskal's Algorithm and prim's algorithm a minimal spanning tree of the following graph:



7. Show that the number of vertices of a binary tree can't be even.
8. Show that every connected graph has a spanning tree.
9. Show that every edge of a connected graph is a branch of some spanning tree of G.
10. Prove that the number of cut vertices in a binary tree is always even.

11. Construct a graph having edge connectivity 4, vertex connectivity 3 and degree of each vertex $\geq$ 5.

Unit 4: Queuing theory:-

1. The arrival rate of cars at GRAND jewelers is 20 cars per hour. The service rate at jeweler shop is 8 cars per hour. The only parking area at GRAND jewelers is restricted to 6 cars only. The arrival rate and service rate follows Poisson distribution. Identify the type of queue and then determine the performance measures of this queue. Analyze the result if service rate goes to 20 cars per hour
2. A supermarket has a single cashier. During the peak hours, customers arrive at a rate of 20 customers per hour. The average number of customers that can be processed by the cashier is 24 hour. Calculate:
   (i) Probability that the cashier is idle (iii) Average number of customer in the system
   (ii) Average time a customer spends in the system (iv) Average time a customer spends in the queue
3. Patients arrive at a clinic according to Poisson distribution at the rate of 30 patients per hour. The waiting room does not accommodate more than 14 patients. Examination time per patient is exponential with mean rate 20 per hour.
   (ii) Find the effective arrival rate at the clinic.
   (iii) What is the probability that an arriving patient will not wait?
   (iv) What is the expected waiting time until a patient is discharged from the clinic?
4. In a railway marshalling yard, goods trains arrive at a rate of 30 trains per day. Assuming that the inter-arrival time follows an exponential distribution and the service time distribution is also exponential with an average of 36 minute. Calculate
   (i) Average number of customers in the system
   (ii) (ii) Probability that the queue size exceeds 10
   (iii) Average time a customer spends in the queue
5. If for a period of 2 hours in the day (8 to 10) trains arrive at the yard every 20 minutes but the service time continues to remain 36 minutes, then calculate for this period
   (i) The trains that the yard is empty
   (ii) (ii) What is the probability that an arriving patient will wait?
   (iii) The average number of trains in the system

   On the assumption that the line capacity of the yard is limited to 4 trains only.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Human Resource Management **Subject Code-CS605B**
Year: 3rd Year **Semester: Sixth**

Module Number	Topics	Number of Lectures
1	**Introduction :**	**3L**
	HR Role and Functions, Concept and Significance of HR, Changing role of HR managers - HR functions and Global Environment, role of a HR Manager.	3
2	**Human Resources Planning:**	**5L**
	HR Planning and Recruitment: Planning Process - planning at different levels - Job Analysis - Recruitment and selection processes - Restructuring strategies - Recruitment-Sources of Recruitment-Selection Process-Placement and Induction-Retention of Employees.	5
3	**Training and Development :**	**6L**
	need for skill upgradation - Assessment of training needs - Retraining and Redeployment methods and techniques of training employees and executives - performance appraisal systems.	6
4	**Performance Management System :**	**5L**
	Definition, Concepts and Ethics-Different methods of Performance Appraisal- Rating Errors- Competency management.	5
5	**Industrial Relations :**	**6L**
	Factors influencing industrial relations - State Interventions and Legal Framework - Role of Trade unions - Collective Bargaining - Workers' participation in management. **Case study.**	6
Total Number Of Hours = 25		

Assignment:

Module-1
1. Describe the role of HR.

Module-2
2. Write down all the restructuring strategies.

Module-3
3. What do you mean by performance appraisal system.

Module-4
4. Discuss competency management.

Module-5
5. Write down the factors influencing industrial management.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: ERP **Subject Code-CS605C**
Year: 3rd Year **Semester: Sixth**

Module Number	Topics	Number of Lectures
1	**Overview of ERP:**	**4L**
	1. **The evolution of ERP systems: A historical perspective:** Evolution through Payroll system, Inventory Control system, Materials Requirement Planning (MRPI) system, Manufacturing Resource Planning(MRPII) system, Their advantages and disadvantages. Definition and Concept of ERP, Business reasons for rise and popularity of ERP system- Benefits of an ERP system.	2
	2. **Business processes supported by ERP systems:** Various business functions in an Organization–Purchasing, Materials Management, Manufacturing, Sales & Distribution, Plant Maintenance, Quality Management, Finance & Accounting including Costing, Human Resources etc. ERP market place– SAP, Oracle, PeopleSoft, JDEdwards, Baan, Microsoft's suit of products etc. Business modules in these ERP packages–a brief comparative description of business function modules and sub-modules. Overview of key end-to-end business processes supported in two major ERP systems(preferably SAP and Oracle)–Order to Cash, Procure to Pay, Plan to Produce and Dispatch.	1
2	**Information Technology and ERP systems:**	**9L**
	1. **The evolution of Information Technology(IT): A historical perspective** Evolution of computer generations (hardware and software)–Operating systems, File systems to Database Management systems, Communication Networks. Enabling of ERP systems by IT evolution.	3
	2. **The evolution of ERP systems architecture** Client-Serverbasedarchitecture,Multi-Tierarchitecture–Presentationlayer, Application layer, and Database layer(On- line Transaction Processing–OLTP).Brief discussion on Extended ERP systems- Web-enabled ERP architecture, Service-Oriented Architecture and Cloud Computing. Open Source ERP.	3
	3. **Related technology concepts** ERP and Supply Chain Management(SCM),and Customer Relationship Management(CRM),ERP and Business Intelligence(some of the popular tools like Cognos, Business Objects should be mentioned),ERP and Data warehousing (Data Mart, Data Mining and On-line Analytical Processing- OLAP),ERP and E-business	3
3.	**Implementation of ERP system:**	**11L**
	1. **ERP implementation approach** Single vendor versus Best-of Breed ERP implementation, Big Bang versus Phased(by module/site) implementation, Using ERP of Application Service Provider(ASP).	2
	2. **ERP implementation life cycle** Planning different aspects(Economic viability, Senior Management commitment, Resource requirements, Change managementetc.),UnderstandingrequirementsandProcesspreparation –Gap analysis and Business Process Engineering, User Acceptancecriteria,Design,Configuration,Customization(difference betweenConfigurationandCustomization, advantages and disadvantages), Extensions, Data migration, End-user training, User Acceptance, Going live, Roll-out. Differences between ERP implementation life cycle and Custom Software development phases. Draw backs of ERP system.	4

	3. Organizing implementation Interaction with Vendors, Consultants, and Users. Contracts with Vendors, Consultants, and Employees. Project Management and Monitoring. ERP Project Organization – Formation of Steering Committee and different User Groups. Top Management Commitment and Steering Committee meetings. Change Management, Risks and Challenges in ERP implementation	**2**
	4. Post-implementation Support, Review, Maintenance and Security of ERP systems A typical Support Cycle(Planning, Stabilization, Ongoing and Upgrade phases).Post-implementation Review of ERP systems– measures of review(Efficiency, Effectiveness, and Competitive Advantage),and approaches for review(User attitude survey, Cost/benefit analysis, Compliance audit, Budget performance review, Service level monitoring, Technical review, Product review, Integration review etc.). System maintenance and ERP system maintenance. Software upgrade (patch, release, version).Security and Access control of ERP systems	**3**
4	**Emerging Trends and Future of ERP systems :**	**7L**
	1. **Emerging Technologies and ERP** *Service-orientedArchitecture(SOA)*:EnterpriseSOAlayers–Businessprocesses, Business services, Components and Integration services, Advantages and Drawbacks of SOA , When to use SOA, Difference between multi-layered Client-server architecture and SOA, basic awareness of Net Weaver from SAP, Web sphere from Oracle and .Net from Microsoft. *Enterprise Application Integration(EAI)*: Basic understanding of the concept, Types of EAI(levels)–User Interface, Method(logic), Application Interface, Data. EAI architecture–Typical framework(Business Processes, Components &Services ,Messaging service and Transport service. Mention of some of the leading EAI vendors–IBM, Microsoft, Oracle, SAP,TIBCO. *Radio Frequency Identification(RFID) and ERP*: awareness of RFID technology, Benefits of RFID integrated with ERPs. *M-Commerce*: basic concept and applications, difference with E-Commerce, benefits of integration with ERPs.	5
	2. **Future of ERP** Technology transformation to SOA, more E-Commerce features, Growing mobile applications ,Economical and Easy models of ERP deployment etc	2
Total Number Of Hours = 41		

Faculty In-Charge HOD, CSE Dept.

Assignment:

1. Write down the ERP architecture & system**.**
2. How to business system supported by ERP application?
3. Short Notes: CRP,SCM.
4. Differentiate between OLTP & OLAP.
5. Write down Emerging Technologies & Future of ERP.

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Title of Course: Database Management System Lab
Course Code: CS691
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 4predicate 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:

- 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.

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).

2 Uses of Database in Real Life

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.

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
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
– 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
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;

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
FORD
russel
greg
9 rows selected.

5. List all employee names and their and their manager whose manager is 7902 or 7566 0r 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

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
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
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
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 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;

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’);

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;

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" VARCHAR(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)**);**

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'**);**
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'**);**
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'**);**
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');
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'**);**
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'**);**
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**)**
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'**);**

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'**);**

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'**);**

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'**);**

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)
 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**)**
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'**);**

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**)**
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.

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

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 it is 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 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.

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.

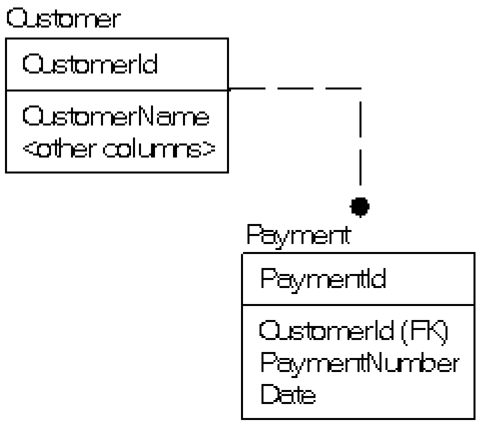
Whenever I see a table with repeating column names appended with numbers, I cringe in horror. And I cringe in horror quite often. Consider the following example **Customer** table:

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 **Paymen**t table, in which you could also record extra information about the payment, like when the payment was made, and what the payment was for:



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, 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 in 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.

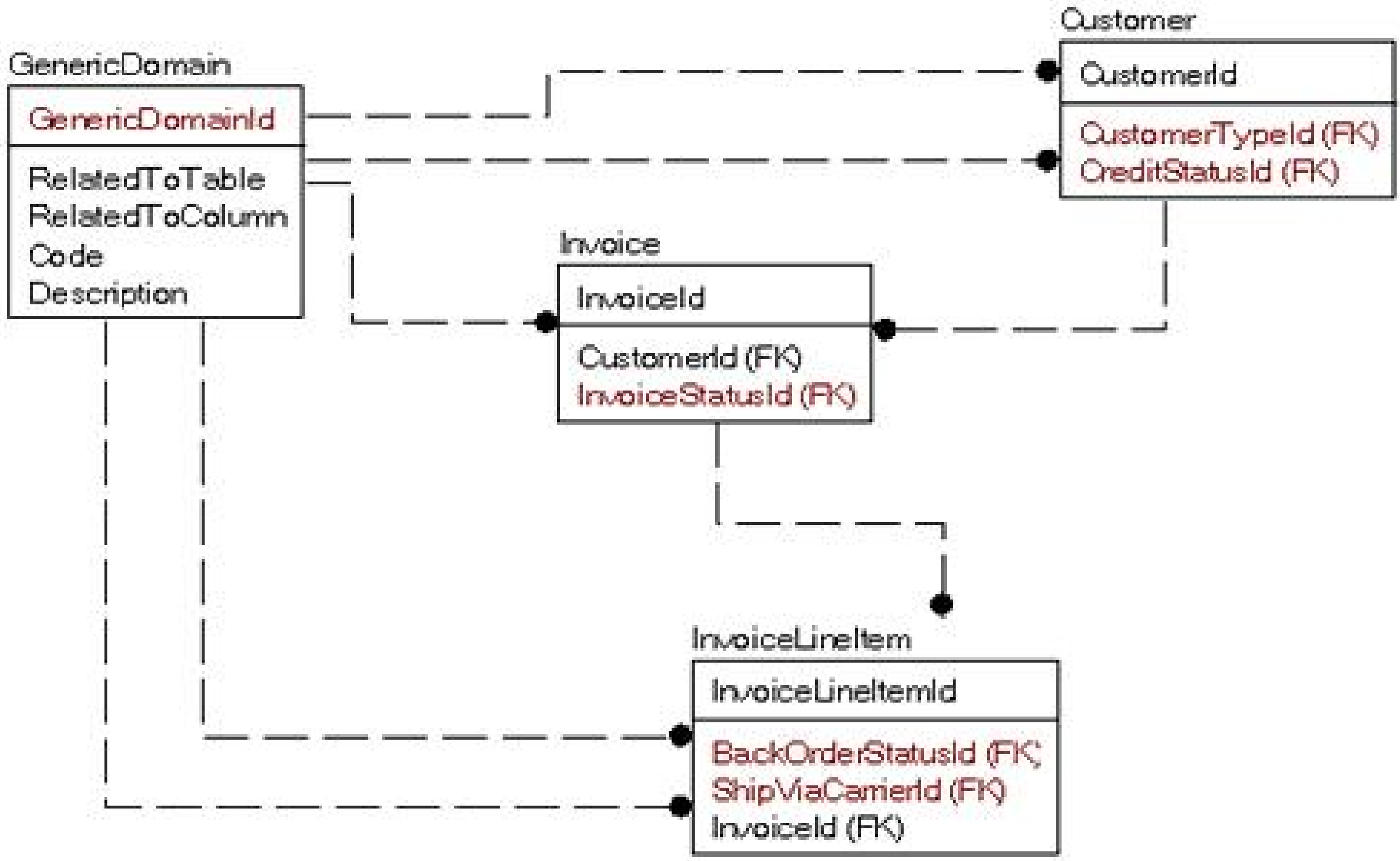
The big myth perpetrated by architects who don't really understand relational database architecture (me included early in my career) is that the more tables there are, the more complex the design will be. So, conversely, shouldn't condensing multiple tables into a single "catch-all" table simplify the 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.

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?

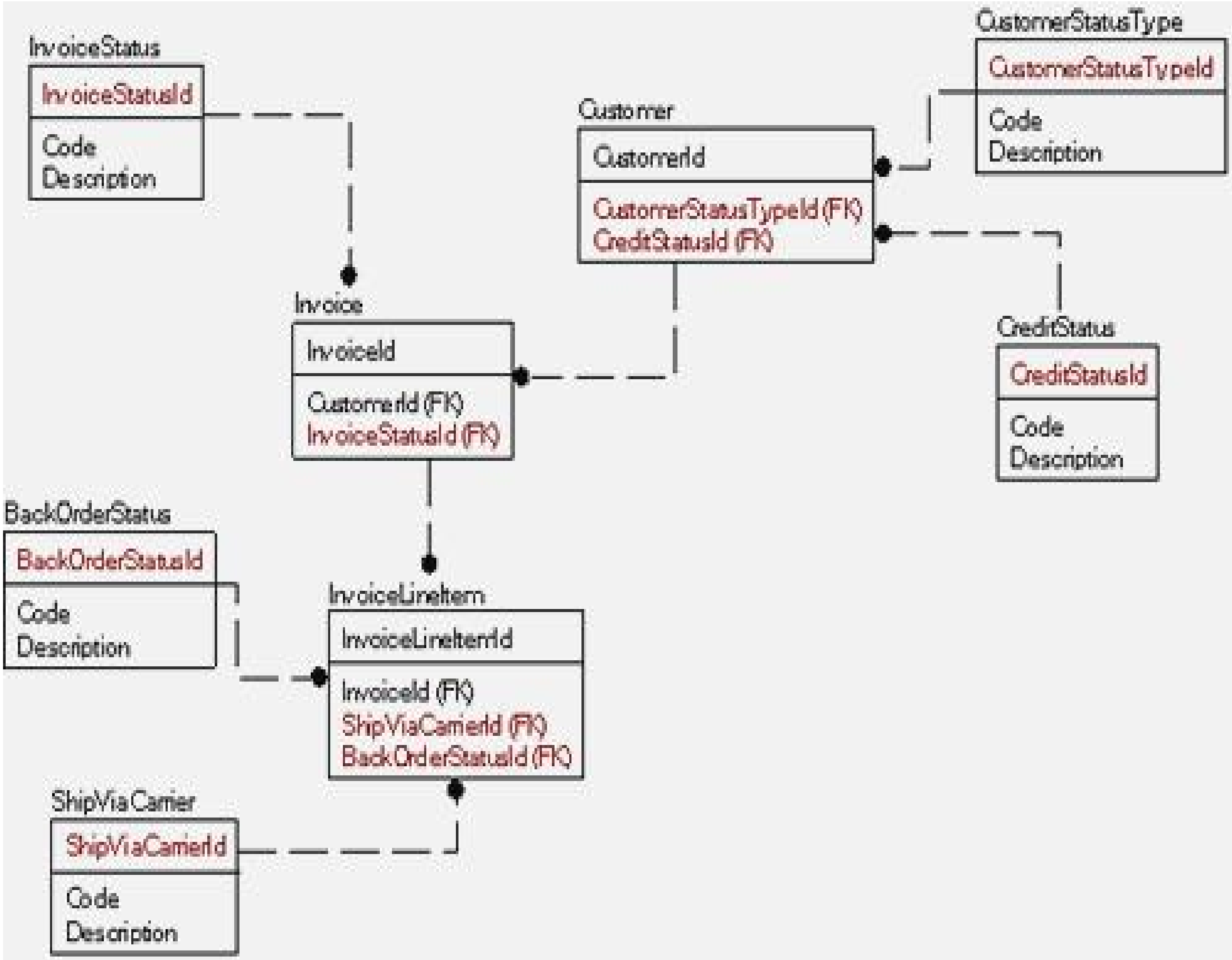


This may seem a very clean and natural way to design a table for all but the problem is that it is just not very natural to work with in SQL. Say we just want the domain values for the **Customer** table:

```
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
      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:

```
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	XXXXXXXX	The X part
2	XXXXXXXX	The X part
3	YYYYYYYY	The Y part

How many rows are there in this table? Well, there seem to be three, but are rows with **PartID**s 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 **EmployeeId**s 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.

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

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:

- 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 cause 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.

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.

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

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
   - from any element to a process (input dataflow) or
   - 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:

- 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.
- It motivates and establishes a framework for the more complicated next level (below).

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:

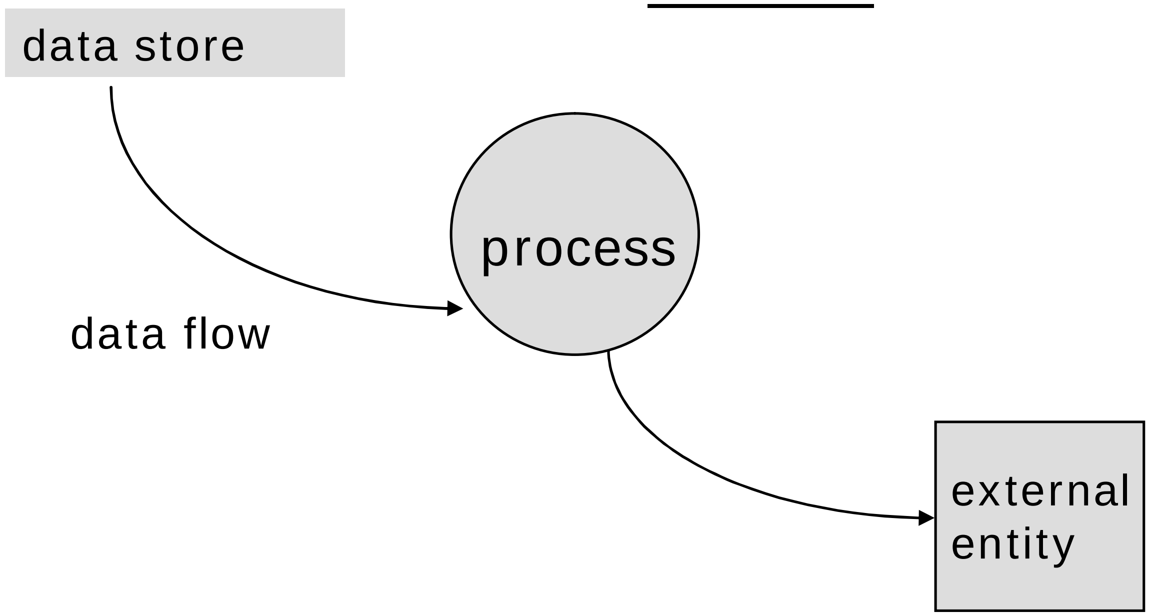
"a network representation of a system. The system may be automated, manual, or mixed. The DFD portrays the system in terms of its component pieces, with all interfaces among the components indicated."

- 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).
- 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

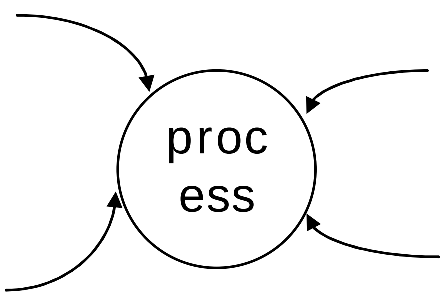
Data Stores

- 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

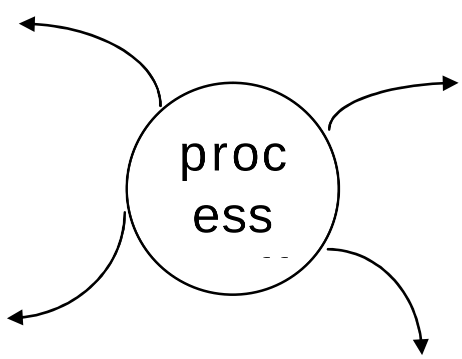
Data Flow Diagram Don'ts

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

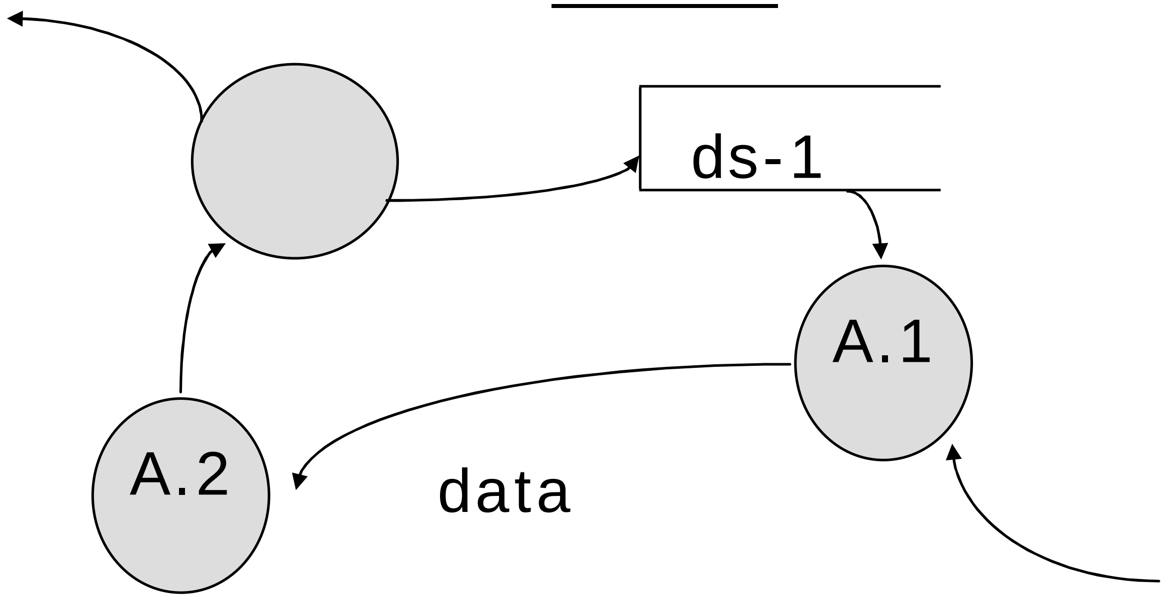


1. 'Black Hole'



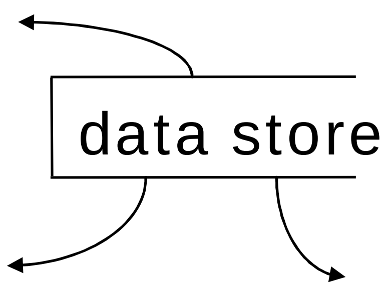
2. 'It's a Miracle'

Data Flow Diagram Don'ts

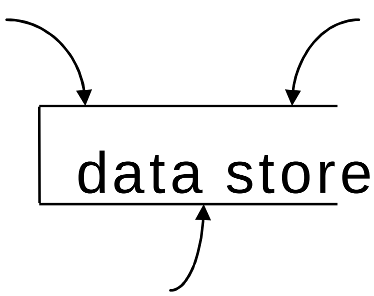


4. Leave Things Unlabeled
Corollary: Labels Should Have Meaning

Data Flow Diagram Don'ts



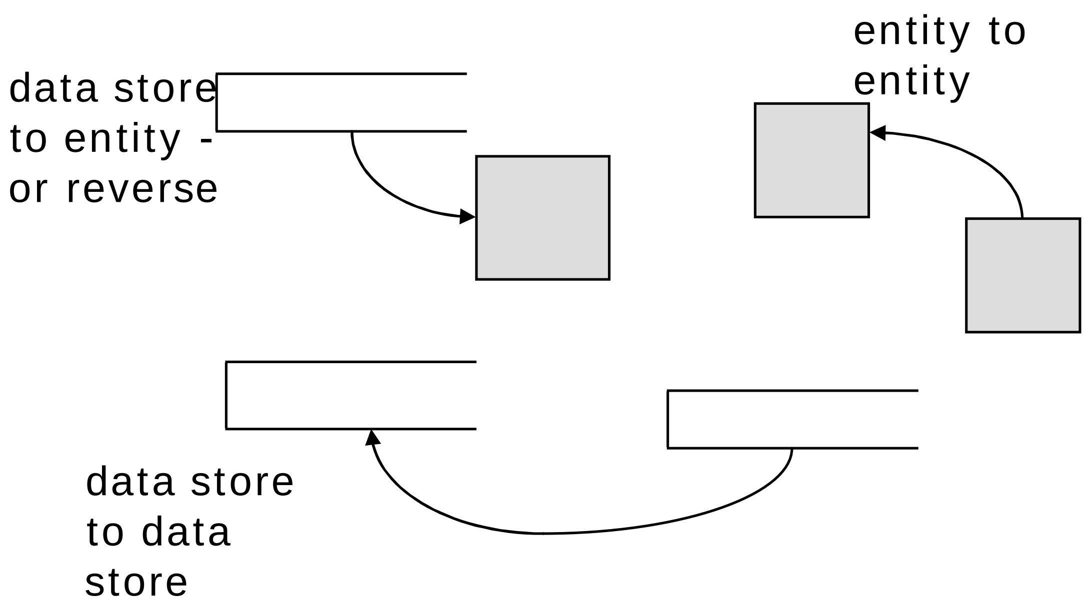
5. Miracle data source



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

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 appears 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
- 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 ± 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 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.

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 though 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.

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 it's "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.

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" that 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

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

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 sub-form 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 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 table**s**, 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....

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 any number of allow 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

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

refering 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.

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.

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

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.

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

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 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 languages1 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 .
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 \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 though 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 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 askedand 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();
}
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.

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 1.5or 5.0

VB Script

VBScript (Visual Basic Scripting Edition) is a scripting language developed by Microsoft for Windows operating systems.

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.

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, becauses 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.

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 embeded 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, becauses 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:

- 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.

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.

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)

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 need for 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

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:

```
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:

- 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")
```

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
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 correspondes to the *DialogResult.OK* code. Thus the program runs a function depending on what Button the User clicked.

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 comptuer 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
```

SQL is a non-precedural 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 complext 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)

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:

```
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
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:

```
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','8IS3KOXW');
```

```
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
herong  8IS3KOXW
mike    PZ0JG
UserCopy table:
UserCopy table:
Field    Type       Null    Key     Default Extra
Login    varchar(8) YES
Password varchar(8) YES
Login   Password
herong  8IS3KOXW
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:

- 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','8IS3KOXW');
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  8IS3KOXW
mike    NULL
herong  8IS3KOXW
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','8IS3KOX');
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.

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_l, table_r
table_l CROSS JOIN table_r
```

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
    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:

```
-- JoinTable.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));
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
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.
- 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
```

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
```

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 column. Tables stored the data which is useful and meaningful.

DATABSE 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

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:

Tool menu
↓
Database utilities
↓
Convert database
↓
To access 97 file format
↓
Open a form visual basic
↓
Crate connectivity tool (data button)
↓
Go to the connectivity tool property
↓
Give the database name
↓
Give the record set name
↓
Then one by one connect the each field

Steps for creating a project:
1. First design forms.
2. Create MDI FORM and link all from 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.

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.

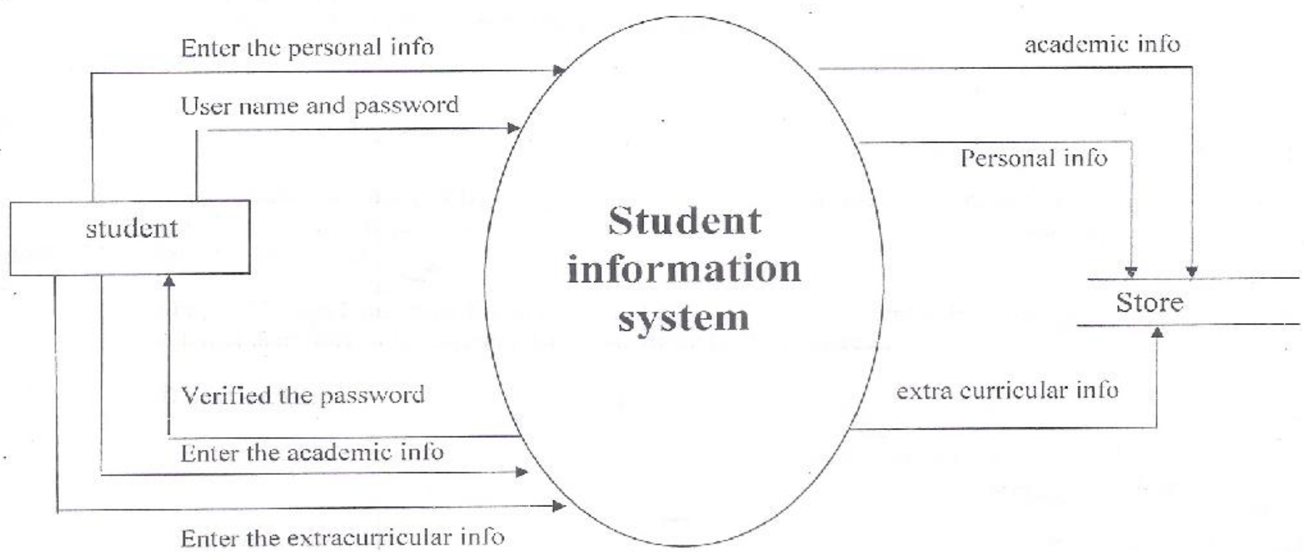
Description of the of the project

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 circular activity (achievements, prizes etc.)

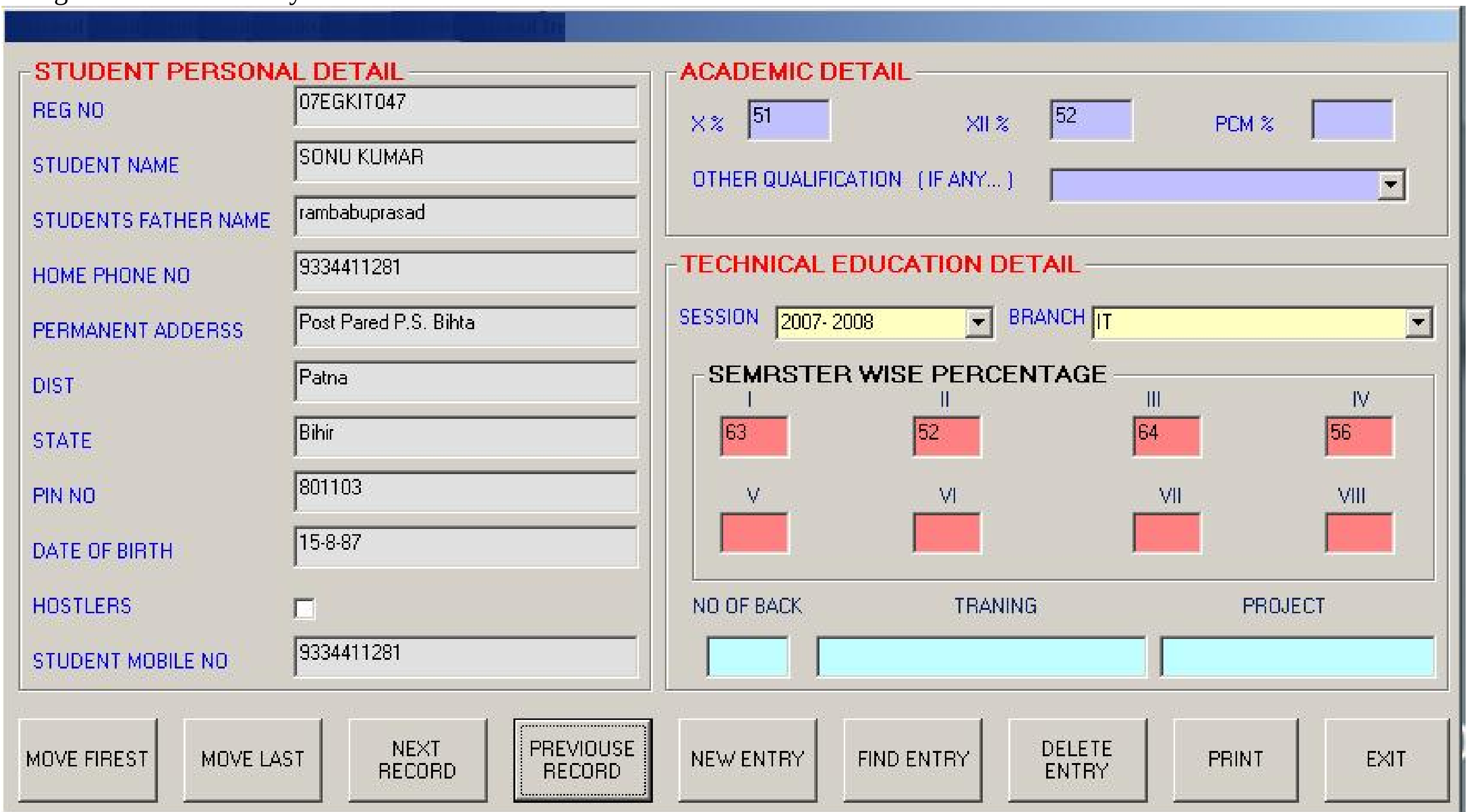
If we need any information of a particular student then we write the name of the student and related query then all the information are retrieve from the database also generate the reports.

DFD OF STUDENT INFORMATION SYSTEM

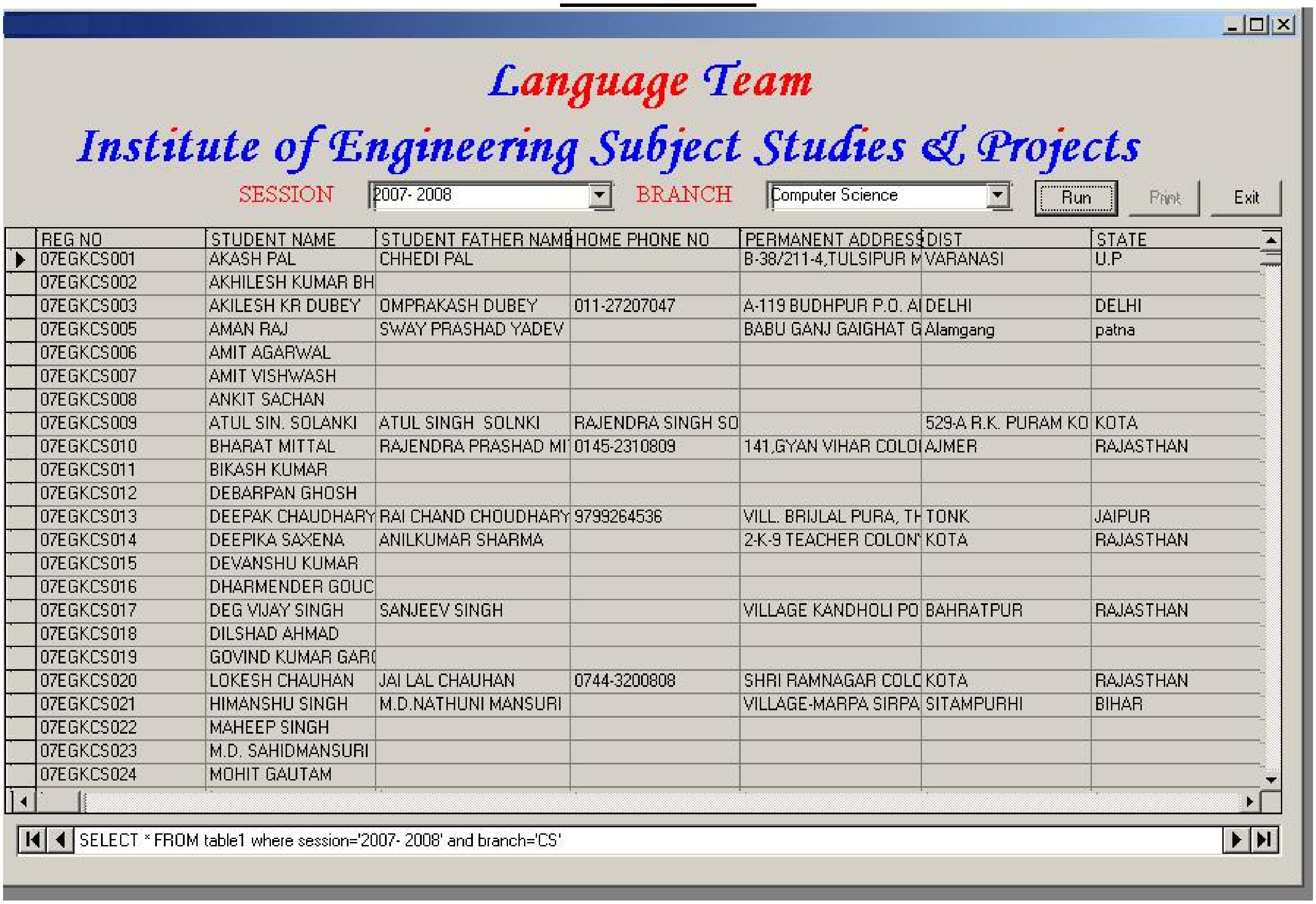


O level of DFD OF STUDENT INFORMATION SYSTEM

Design the Form for Entry Level

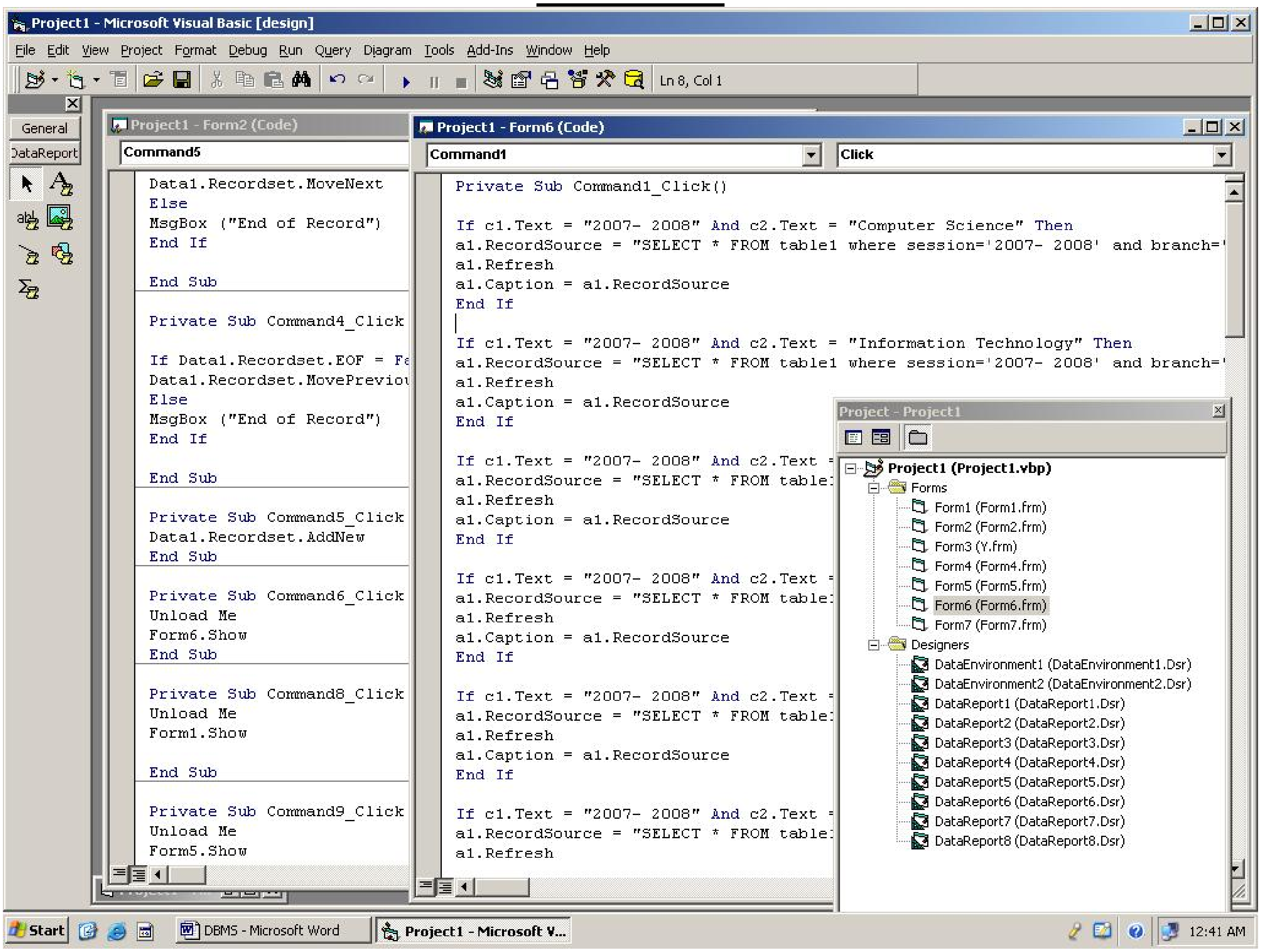


Design form for reports & filtrtation



REG NO	STUDENT NAME	STUDENT FATHER NAME	HOME PHONE NO	PERMANENT ADDRESS	DIST	STATE
07EGKCS001	AKASH PAL	CHHEDI PAL		B-38/211-4,TULSIPUR M	VARANASI	U.P
07EGKCS002	AKHILESH KUMAR BH					
07EGKCS003	AKILESH KR DUBEY	OMPRAKASH DUBEY	011-27207047	A-119 BUDHPUR P.O. AI	DELHI	DELHI
07EGKCS005	AMAN RAJ	SWAY PRASHAD YADEV		BABU GANJ GAIGHAT G	Alamgang	patna
07EGKCS006	AMIT AGARWAL					
07EGKCS007	AMIT VISHWASH					
07EGKCS008	ANKIT SACHAN					
07EGKCS009	ATUL SIN. SOLANKI	ATUL SINGH SOLNKI	RAJENDRA SINGH SO		529-A R.K. PURAM KO	KOTA
07EGKCS010	BHARAT MITTAL	RAJENDRA PRASHAD MI	0145-2310809	141,GYAN VIHAR COLOI	AJMER	RAJASTHAN
07EGKCS011	BIKASH KUMAR					
07EGKCS012	DEBARPAN GHOSH					
07EGKCS013	DEEPAK CHAUDHARY	RAI CHAND CHOUDHARY	9799264536	VILL. BRIJLAL PURA, TH	TONK	JAIPUR
07EGKCS014	DEEPIKA SAXENA	ANILKUMAR SHARMA		2-K-9 TEACHER COLON'	KOTA	RAJASTHAN
07EGKCS015	DEVANSHU KUMAR					
07EGKCS016	DHARMENDER GOUC					
07EGKCS017	DEG VIJAY SINGH	SANJEEV SINGH		VILLAGE KANDHOLI PO	BAHRATPUR	RAJASTHAN
07EGKCS018	DILSHAD AHMAD					
07EGKCS019	GOVIND KUMAR GAR(					
07EGKCS020	LOKESH CHAUHAN	JAI LAL CHAUHAN	0744-3200808	SHRI RAMNAGAR COLC	KOTA	RAJASTHAN
07EGKCS021	HIMANSHU SINGH	M.D.NATHUNI MANSURI		VILLAGE-MARPA SIRPA	SITAMPURHI	BIHAR
07EGKCS022	MAHEEP SINGH					
07EGKCS023	M.D. SAHIDMANSURI					
07EGKCS024	MOHIT GAUTAM					

Write down the appropriate code for Student information system like below ex.



Project1 - Microsoft Visual Basic [design]
File Edit View Project Format Debug Run Query Diagram Tools Add-Ins Window Help
Ln 8, Col 1
General
DataReport
Project1 - Form2 (Code)
Command5
Data1.Recordset.MoveNext
Else
MsgBox ("End of Record")
End If
End Sub
Private Sub Command4_Click
If Data1.Recordset.EOF = Fa
Data1.Recordset.MovePreviou
Else
MsgBox ("End of Record")
End If
End Sub
Private Sub Command5_Click
Data1.Recordset.AddNew
End Sub
Private Sub Command6_Click
Unload Me
Form6.Show
End Sub
Private Sub Command8_Click
Unload Me
Form1.Show
End Sub
Private Sub Command9_Click
Unload Me
Form5.Show
Project1 - Form6 (Code)
Command1
Click
Private Sub Command1_Click()
If c1.Text = "2007- 2008" And c2.Text = "Computer Science" Then
a1.RecordSource = "SELECT * FROM table1 where session='2007- 2008' and branch='
a1.Refresh
a1.Caption = a1.RecordSource
End If
If c1.Text = "2007- 2008" And c2.Text = "Information Technology" Then
a1.RecordSource = "SELECT * FROM table1 where session='2007- 2008' and branch='
a1.Refresh
a1.Caption = a1.RecordSource
End If
If c1.Text = "2007- 2008" And c2.Text
a1.RecordSource = "SELECT * FROM table1
a1.Refresh
a1.Caption = a1.RecordSource
End If
If c1.Text = "2007- 2008" And c2.Text
a1.RecordSource = "SELECT * FROM table1
a1.Refresh
a1.Caption = a1.RecordSource
End If
If c1.Text = "2007- 2008" And c2.Text
a1.RecordSource = "SELECT * FROM table1
a1.Refresh
a1.Caption = a1.RecordSource
End If
If c1.Text = "2007- 2008" And c2.Text
a1.RecordSource = "SELECT * FROM table1
a1.Refresh
Project - Project1
Project1 (Project1.vbp)
Forms
Form1 (Form1.frm)
Form2 (Form2.frm)
Form3 (Y.frm)
Form4 (Form4.frm)
Form5 (Form5.frm)
Form6 (Form6.frm)
Form7 (Form7.frm)
Designers
DataEnvironment1 (DataEnvironment1.Dsr)
DataEnvironment2 (DataEnvironment2.Dsr)
DataReport1 (DataReport1.Dsr)
DataReport2 (DataReport2.Dsr)
DataReport3 (DataReport3.Dsr)
DataReport4 (DataReport4.Dsr)
DataReport5 (DataReport5.Dsr)
DataReport6 (DataReport6.Dsr)
DataReport7 (DataReport7.Dsr)
DataReport8 (DataReport8.Dsr)
Start
DBMS - Microsoft Word
Project1 - Microsoft V...
12:41 AM

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

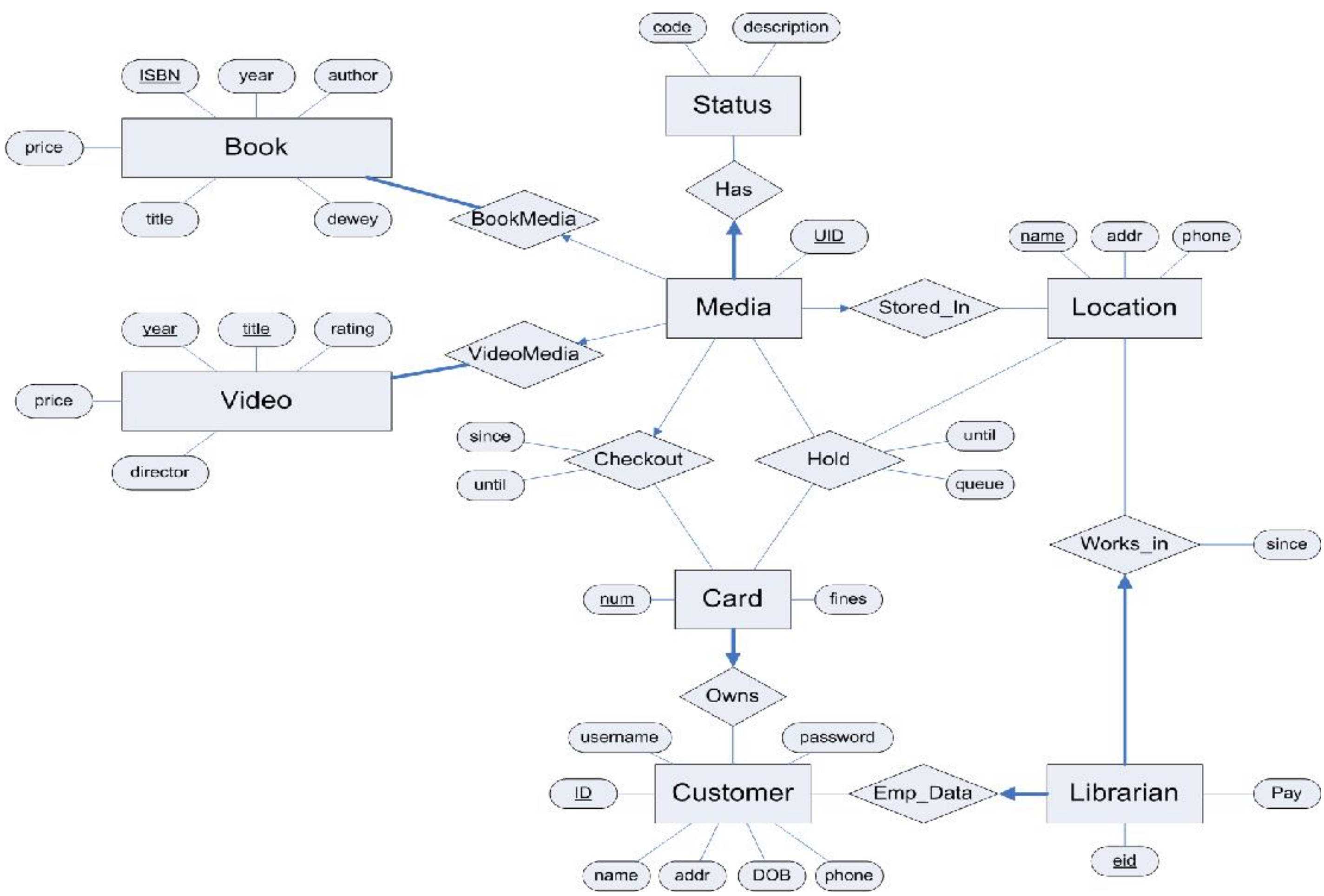
Functions for librarians are the same as the functions for customers plus the following:

- Add customers
- Add library cards and assign them to customers
- Check out media
- Manage and transfer media that is currently on hold
- 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 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	desctiption					
Media	media_id	code					
Book	ISBN	title	author	year	dewey	price	
BookMedia	media_id	ISBN					
Customer	ID	name	addr	DOB	phone	userna me	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:

```
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', 'jlgray', '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', 'wvwoodson', 'password4');
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);
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);
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.

```
/* *\
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
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>);
/* 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 */
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.
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.

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.

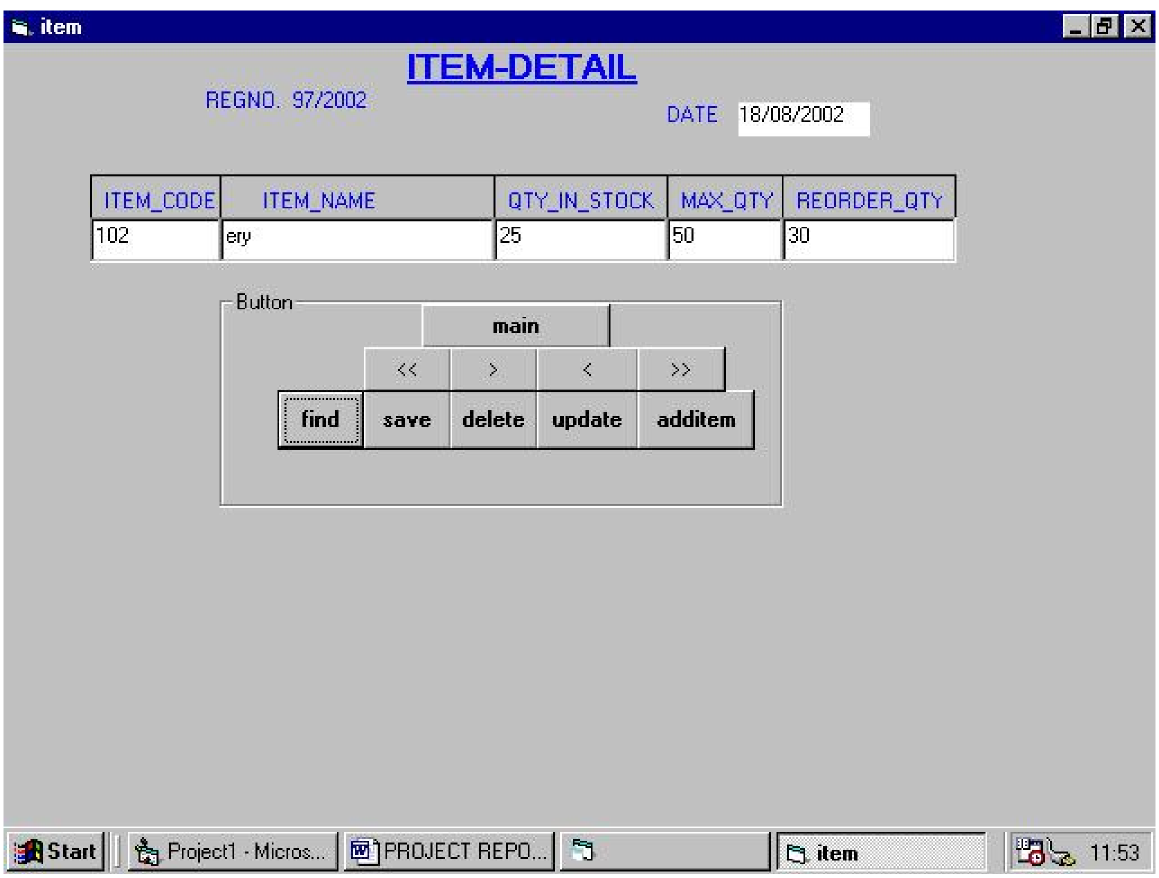
Customer address: - It determines the address of the customer.

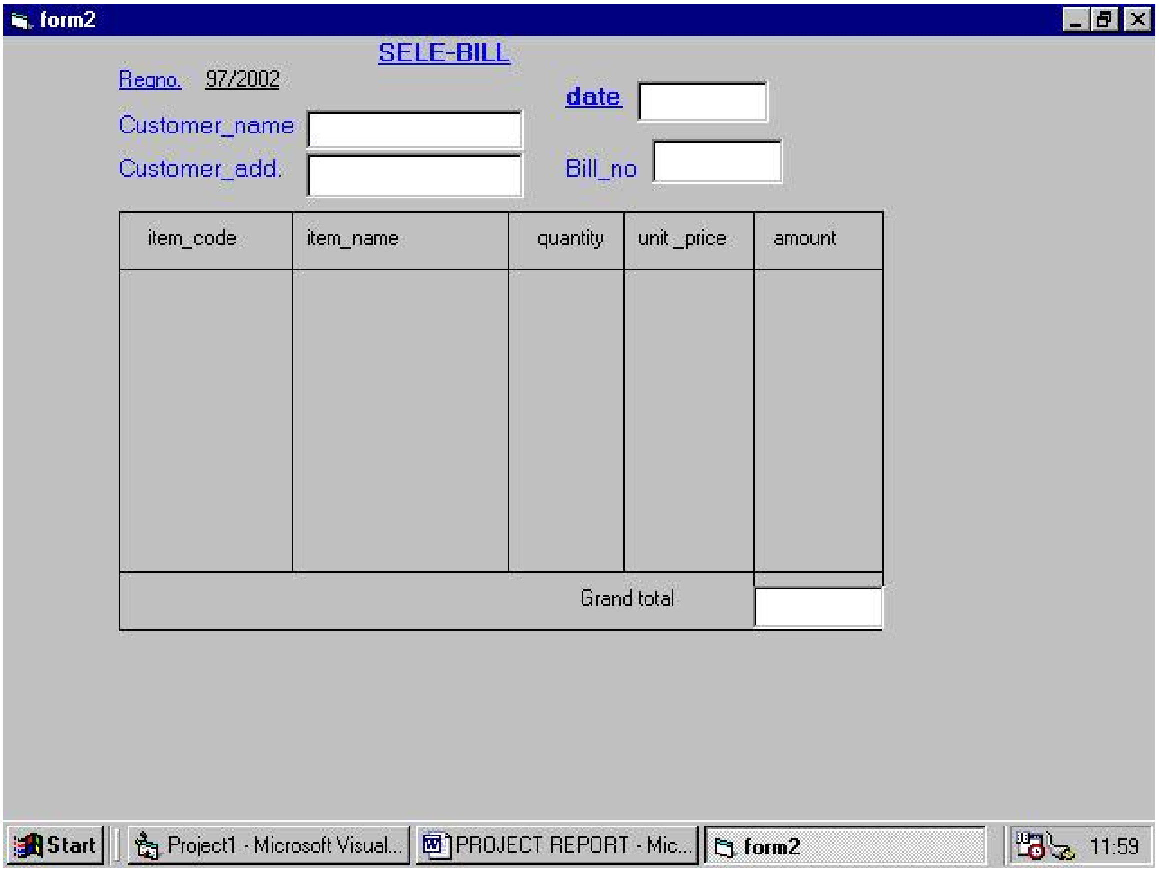
Unit price: - It shows the price per item.

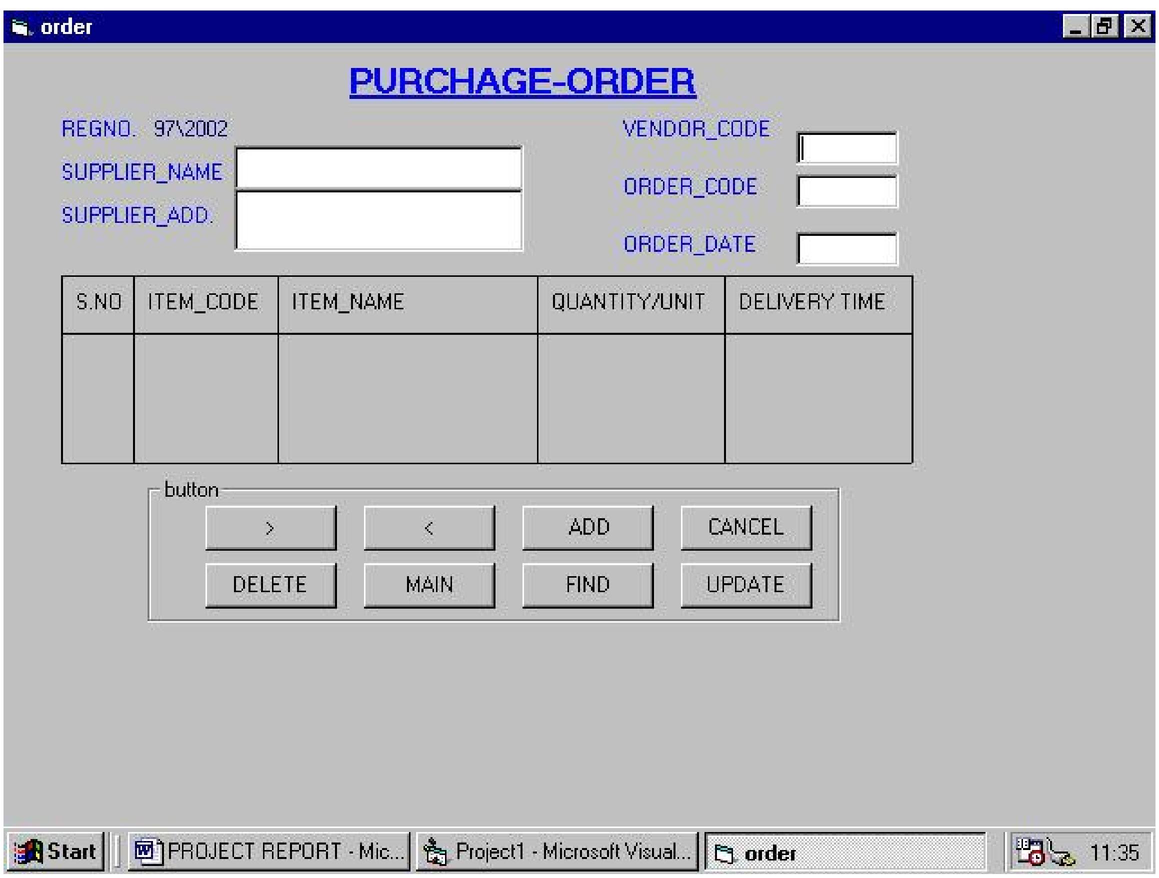
Amount: - it determines the amount per item.

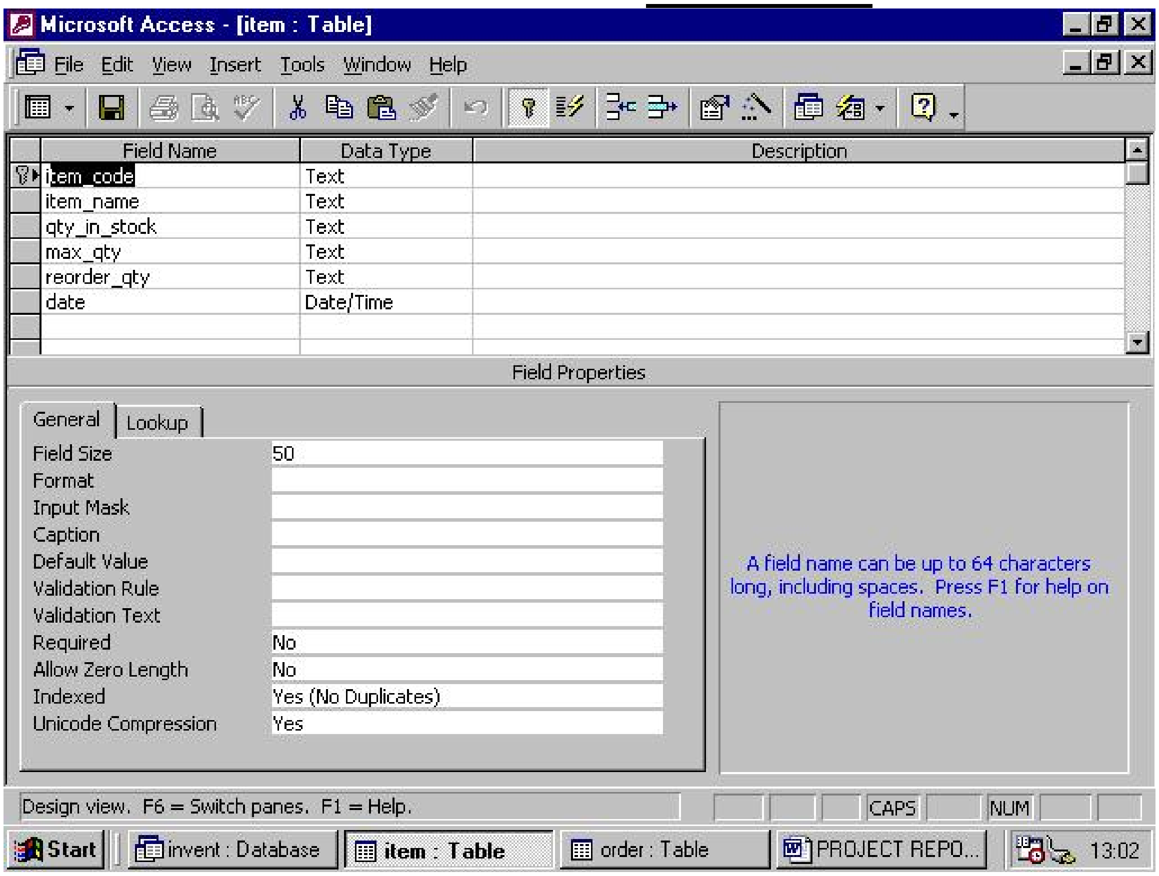
Total Amount: - This field shows the total amount of the item that has been purchase by the customer.

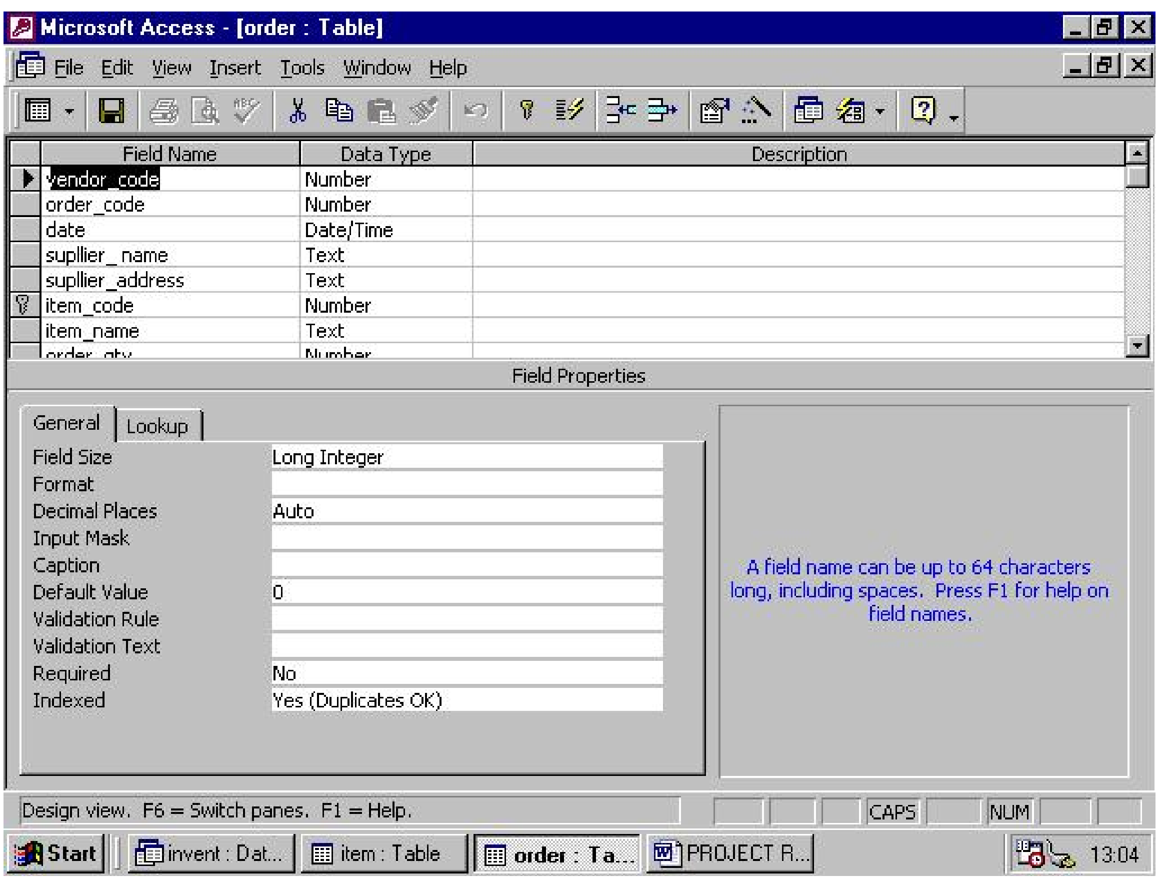

file
item
order
bill
exit
INVENTORY MANAGEMENT SYSTEM
WELCOME TO YOU
LITTLE SHOPPER'S
IN
A NEW ERA
DEPARTMENTL STORE
Start
Project1 - Microsoft Visual...
PROJECT REPORT - Mic...
11:47

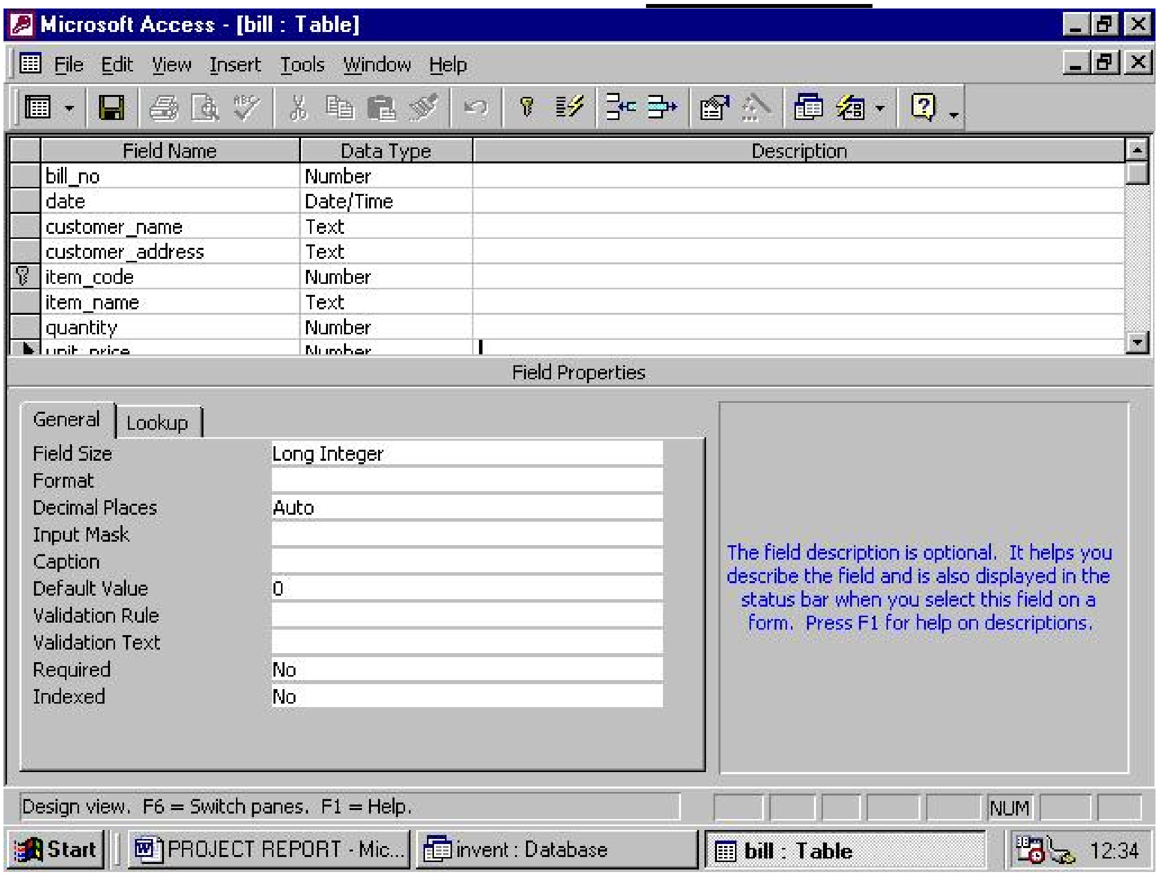

item
ITEM-DETAIL
REGNO. 97/2002
DATE 18/08/2002
ITEM_CODE
ITEM_NAME
QTY_IN_STOCK
MAX_QTY
REORDER_QTY
102
ery
25
50
30
Button
main
<<
>
<
>>
find
save
delete
update
additem
Start
Project1 - Micros...
PROJECT REPO...
item
11:53


form2
SELE-BILL
Regno. 97/2002
date
Customer_name
Customer_add.
Bill_no
item_code
item_name
quantity
unit _price
amount
Grand total
Start
Project1 - Microsoft Visual...
PROJECT REPORT - Mic...
form2
11:59

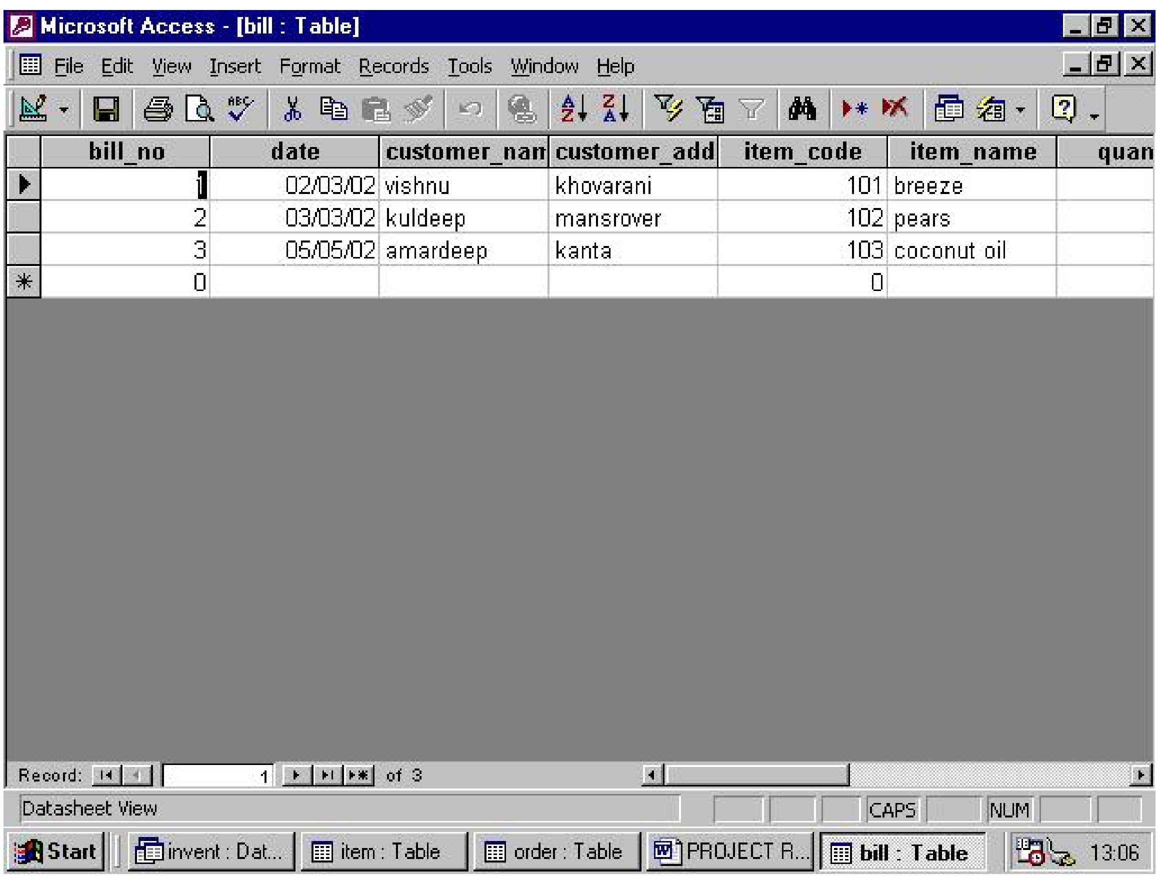

order
PURCHAGE-ORDER
REGNO. 97\2002
VENDOR_CODE
SUPPLIER_NAME
ORDER_CODE
SUPPLIER_ADD.
ORDER_DATE
S.NO
ITEM_CODE
ITEM_NAME
QUANTITY/UNIT
DELIVERY TIME
button
>
<
ADD
CANCEL
DELETE
MAIN
FIND
UPDATE
Start
PROJECT REPORT - Mic...
Project1 - Microsoft Visual...
order
11:35


Microsoft Access - [item : Table]
File Edit View Insert Tools Window Help
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
General
Lookup
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
A field name can be up to 64 characters long, including spaces. Press F1 for help on field names.
Design view. F6 = Switch panes. F1 = Help.
CAPS
NUM
Start
invent : Database
item : Table
order : Table
PROJECT REPO...
13:02


Microsoft Access - [order : Table]
File Edit View Insert Tools Window Help
Field Name
Data Type
Description
vendor_code Number
order_code Number
date Date/Time
supllier_ name Text
supllier_address Text
item_code Number
item_name Text
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 Yes (Duplicates OK)
A field name can be up to 64 characters long, including spaces. Press F1 for help on field names.
Design view. F6 = Switch panes. F1 = Help.
CAPS
NUM
Start
invent : Dat...
item : Table
order : Ta...
PROJECT R...
13:04



Microsoft Access - [bill : Table]
File Edit View Insert Tools Window Help
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
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
Start
PROJECT REPORT - Mic...
invent : Database
bill : Table
12:34



Microsoft Access - [bill : Table]
File Edit View Insert Format Records Tools Window Help
bill_no date customer_nam customer_add item_code item_name quan
1 02/03/02 vishnu khovarani 101 breeze
2 03/03/02 kuldeep mansrover 102 pears
3 05/05/02 amardeep kanta 103 coconut oil
0 0
Record: 1 of 3
Datasheet View
CAPS
NUM
Start
invent : Dat...
item : Table
order : Table
PROJECT R...
bill : Table
13:06

Microsoft Access - [item : Table]

item_code	item_name	qty_in_stock	max_qty	reorder_qty	date
102	ery	25	50	30	02/03/02
103	haamam	56	123	22	02/03/02
105	cj	98	134	0	02/05/02
107	khj	56	79	45	03/04/02
108	hiwe	78	89	0	05/02/02
109	we	23	43	56	06/07/02
112	oure	46	32	36	02/03/02
113	ie	37	67	0	03/05/02
114	kjljk	78	89	88	07/02/02
115	jkldsu	87	578	0	03/09/02
117	we	89	100	56	01/02/02
119	COCONUT OIL	50	100		05/05/02
120	DIP	35	59	0	
125iohdsjk	46	58	24	12	12/05/02
177	jkld	36	57	24	
501	hamam	67	150	0	

Record: 1 of 16

Microsoft Access - [order : Table]

vendor_code	order_code	date	supllier_nam	supllier_addre	item_code	item_n
201	111	02/05/02	sumati	tonk phatak	1000	hjk
202	112	04/05/02	vishnu	khaowrani	1001	hjkdsf
203	113	03/06/02	rajendra	tonk phatak	1002	oiewiu
204	114	04/07/02	sitaram	jhothwara	1003	jerj
205	115	04/03/02	kuldeep	mansrovar	1004	jads
0	0				0	

Record: 1 of 5

14. LAB MANUALS

Lab manual concept is to define how Student design the program on a particular Objective

Hear we design a basic concept

Collage Name

Lab Manual
2010-2011

Subject : DBMS LAB **Branch : CS/IT**

Year / Sem : **Marks : 100**

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)

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Lab Manual

Operating System

Windows XP/ 2003 Server/ Vista/2000

Project

Project Name :

Objective :

Requirement Analysis :

ER Diagram :

Data Flow Diagram :

Database :

Database Management System :

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.
   - 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?
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(A1, A2, …, An) is made up of the relation name R and the list of attributes Ai that it contains. A relation is defined as a set of tuples. Let r be the relation which contains set tuples (t1, t2, t3, ..., tn). Each tuple is an ordered list of n-values t=(v1,v2, ..., vn).

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?
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 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 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 t1 and t2 in r if t1[X] = t2[X] then they have t1[Y] = t2[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 A in F with a dependency Y 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 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 t1 and t2 exist in r such that t1[X] = t2[X] then t3 and t4 should also exist in r with the following properties
t3[x] = t4[X] = t1[X] = t2[X]
t3[Y] = t1[Y] and t4[Y] = t2[Y]
t3[Z] = t2[Z] and t4[Z] = t1[Z]
where [Z = (R-(X U 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 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 A either of the following is true

- 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 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 {R1, R2, ..., Rn} that holds R, one the following is true

- Ri = 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

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 {R1, R2, ..., Rn} is said to hold over a relation R if R1, R2, R3, ..., Rn 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"?**
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

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 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 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
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 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.

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 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 Amstrong 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
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 Ri is equal to the closure of F. i.e.,
((PR1(F)) U … U (PRn(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 .
Simulatneous 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 comparisions. 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'
        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;

************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');
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)';
```

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),
	"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),
	"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),
	"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');
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, 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');
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 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');

-- 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');
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,
```

```
	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)
	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');
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');
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)
        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)
    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),',
            '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');
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)
	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)
```

```
	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),
	"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));
```

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),
    "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 (
    "CUST_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 (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');
```

Sample 2

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:

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),
    "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"(
	"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),
	"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),
    "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_INTROCUSTNO 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_FDSERNO PRIMARY KEY(FD_SER_NO),
    CONSTRAINT FK_FDMSTR_BRANCHNO FOREIGN KEY(BRANCH_NO)
        REFERENCES BRANCH_MSTR(BRANCH_NO),
    CONSTRAINT FK_FDMSTR_INTROCUSTNO 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),
    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_FDSERNO 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),
    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_ADDRTYPE
        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')));

-- 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');
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');
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 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');

-- 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');
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,
    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,
    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');
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');
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');
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)
    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),',
		'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');
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)
        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)
	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
    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:
```
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;

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",
                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:

```
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');
```

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);
```

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
                        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;
```

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');
```

Example 17:

```
CREATE VIEW vw_Branch AS
	SELECT BRANCH_NO, NAME, ADDR_TYPE, ADDR1, ADDR2, CITY, STATE, PINCODE
		FROM BRANCH_MSTR, ADDR_DTS
		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:

```
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:
```
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)));
```

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';
```

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',
            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',
    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",
        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)
	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);

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:

```
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:
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;

<u>UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR</u>
<u>Lab Manual</u>

Title of Course: Computer Network Lab
Course Code: CS692
L-T-P scheme: 0-0-3 **Course Credit: 2**

Objectives:
This practical course provides students with hands on training regarding the design, troubleshooting, modeling and evaluation of computer networks. In this course, students are going to experiment in a real and simulation based test-bed networking environment, and learn about network design and troubleshooting topics and tools such as: network addressing, Address Resolution Protocol, basic troubleshooting tools (like ping, ICMP), IP routing (e.g. RIP), TCP and UDP,DHCP,ACL and many others. Student will have the opportunity to build some simple networking models using the tool and perform simulations that will help them evaluate their design approaches and expected network performance.
Learning Outcomes: The students will have a detailed knowledge network topology, Local area network, IP addressing, familiarization with network simulator, idea about networking devices, network cable and connectors, different types routing protocols, concept of remote access and different types of application layer protocol. Upon the completion of Computer network practical course, the student will be able to:

- **Learn** various network commands.
- **Understand** and implement basic of Network and Network Topology.
- **To get** idea about IP addressing schemes.
- **Understand** the benefits of network.
- **Configure** and simulate various protocols.
- **Access** remote desktop.
- **Connect** to different computer using LAN.
- **Understand** the concepts of access control.

Course Contents:
Exercises that must be done in this course are listed below:
Exercise No.1: Study of different types of Network cables and practically implements the cross-wired cable and straight through cable using clamping tool.
Exercise No. 2: Familiarization with some network devices.
Exercise No. 3: Study of Network IP.
Exercise No. 4: Connect the computers in LAN.
Exercise No. 5: Introduction to Packet Tracer.
Exercise No. 6: Configure network topology using packet tracer.
Exercise No. 7: Configure network topology using packet tracer to find the routing path by IPRoute Command.
Exercise No. 8: Network Configuration using distance vector routing protocol.
Exercise No. 9: Configuration of DHCP Protocol
Exercise No. 10: Telnet Configuration.
Exercise No. 11: Configuration of Access Control List.

Text Book:
1. B. A. Forouzan – "Data Communications and Networking (3rd Ed.) " – TMH
Reference Book:
1. Authorized Self-Study Guide "Interconnecting Cisco Network Devices, Part 1(ICND1), 2nd Edition, January, 2008.

Recommended Systems/Software Requirements:
1. CAT-5/CAT-6 Cables, RJ 45, Cutter, Clamping Tool, Router , Switch and Hub.
2. Intel based desktop PC with minimum of 166 MHZ or faster processor with at least 64 MB RAM and 100 MB free disk space.
3. Turbo C or TC3 compiler in Windows XP or Linux Operating System

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR
Lab Manual

Experiment No: 1 STUDY OF DIFFERENT TYPES OF NETWORK CABLES
Aim: Study of different types of Network cables and practically implements the cross-wired cable and straight through cable using clamping tool.
Apparatus: RJ-45 connector, Clamping Tool, Twisted pair Cable
Description:
RJ-45: Registered Jack 45 (RJ45) is a standard type of physical connector for network cables. RJ45 connectors are most commonly seen with Ethernet cables and networks.

Two standard RJ45 pinouts define the arrangement of the individual eight wires needed when attaching connectors to a cable - the T568A and T568B standards. Both follow a convention of coating individual wires in one of five colors - brown, green, orange, blue and white - with certain stripe and solid combinations.

Following these conventions is essential when building cables to ensure electrical compatibility with other equipment. For historical reasons, T568B has become the more popular standard. The table below summarizes this color coding.

Table-1

T568B / T568A Pinouts		
Pin	**T568B**	**T568A**
1	white with orange stripe	white with green stripe
2	orange	Green
3	with with green stripe	white with orange stripe
4	blue	Blue
5	white with blue stripe	white with blue stripe
6	green	Orange
7	white with brown stripe	white with brown stripe
8	Brown	Brown

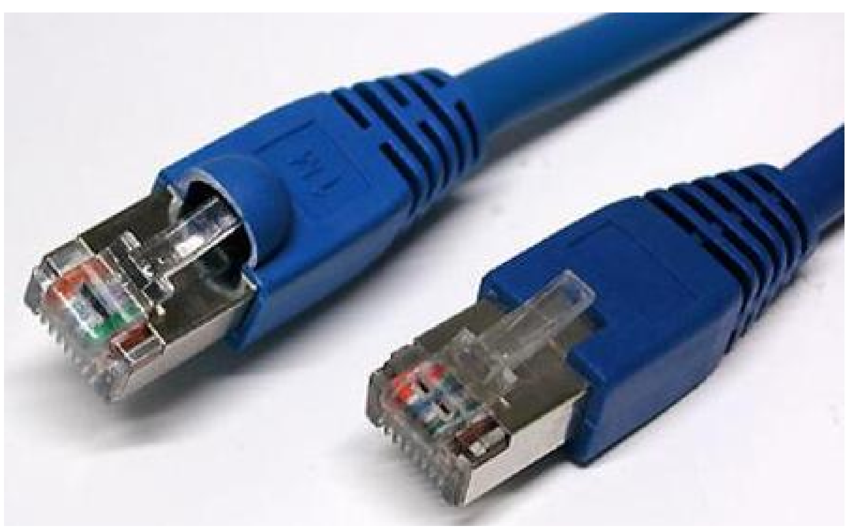


Figure-1: RJ-45

Clamping Tool: A clamping tool is a fastening device used to hold or secure objects tightly together to prevent movement or separation through the application of inward pressure. It is used to fasten RJ-45 with cable tightly.

Figure-2: Clamping Tool

Twisted Pair Cable: Twisted pair is the ordinary copper wire that connects home and many business computers to the telephone company. To reduce crosstalk or electromagnetic induction between pairs of wires, two insulated copper wires are twisted around each other. Each connection on twisted pair requires both wires. Since some telephone sets or desktop locations require multiple connections, twisted pair is sometimes installed in two or more pairs, all within a single cable. For some business locations, twisted pair is enclosed in a shield that functions as a ground. This is known as shielded twisted pair (STP). Ordinary wire to the home is unshielded twisted pair (UTP).

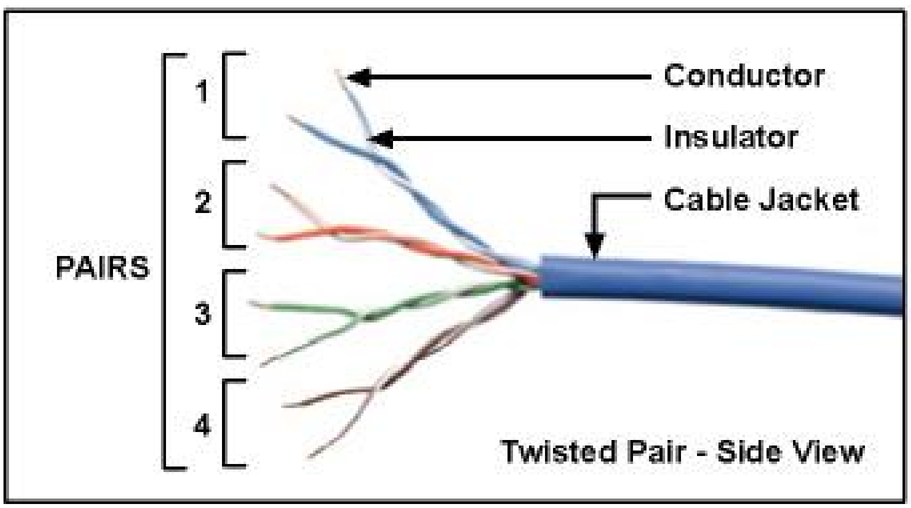


Figure-3: Twisted Pair Cable

Procedure: To do these practical following steps should be done:
1. Start by stripping about of the plastic jacket off the end of the cable. Be very careful at this point, as to not nick or cut into the wires, which are inside. Doing so could alter the characteristics of your cable, or even worse render is useless. Check the wires, one more time for nicks or cuts. If there are any, just whack the whole end off, and start over.
2. Spread the wires apart, but be sure to hold onto the base of the jacket with your other hand. You do not want the wires to become untwisted down inside the jacket. Category 5/6 cable must only have ½ of an inch of 'untwisted' wire at the end; otherwise it will be 'out of spec'. At this point, you obviously have A LOT more than ½ of an inch of un-twisted wire.
3. You have 2 end jacks, which must be installed on your cable. If you are using a pre-made cable, with one of the ends whacked off, you only have one end to install-the crossed over end. Below are two diagrams, which show how you need to arrange the cables for each type of cable end. Decide at this point which end you are making and examine the associated picture below. Figure-4 shows you how to prepare cross wired connection. Figure-5 shows you how to prepare straight through wired connection.

RJ45 Pin # (END 1)	Wire Color	Diagram End #1	RJ45 Pin # (END 2)	Wire Color	Diagram End #2
1	White/Orange		1	White/Green	
2	Orange		2	Green	
3	White/Green		3	White/Orange	
4	Blue		4	White/Brown	
5	White/Blue		5	Brown	
6	Green		6	Orange	
7	White/Brown		7	Blue	
8	Brown		8	White/Blue	

Figure-4: Color Combination (T568B) Table

RJ45 Pin # (END 1)	Wire Color	Diagram End #1	RJ45 Pin # (END 2)	Wire Color	Diagram End #2
1	White/Orange		1	White/Green	
2	Orange		2	Green	
3	White/Green		3	White/Orange	
4	Blue		4	White/Brown	
5	White/Blue		5	Brown	
6	Green		6	Orange	
7	White/Brown		7	Blue	
8	Brown		8	White/Blue	

Figure-5: Color Combination Table

Experiment No: 2 FAMILIARIZATION WITH SOME NETWORK DEVICES
Aim: Study of following network devices

- Switch
- Router
- Hub
- Repeater

Apparatus: Hardware needed
Procedure: Following should be done to understand this practical
1. Switch: A network switch or switching hub is a computer networking device that connects network segments. The term commonly refers to a network bridge that processes and routes data

at the data link layer of the OSI model. Switches that additionally process data at the network layer are often referred to as Layer 3 switches or multilayer switches.

2. Router: A router is an electronic device that interconnects two or more computer networks, and selectively interchanges packets of data between them. Each data packet contains address information that a router can use to determine if the source and destination are on the same network, or if the data packet must be transferred from one network to another. Where multiple routers are used in a range collection of interconnected networks, the routers exchange information about target system addresses, so that each router can build up a table showing the preferred paths between any two systems on the interconnected networks.

3. Repeater: Functioning at Physical Layer a repeater is an electronic device that receives a signal and retransmits it at a higher level and /or higher power, or onto the other side of an obstruction, so that the signal can cover longer distances. Repeater have two ports, so cannot be use to connect for more than two devices.

4. Hub: An Ethernet hub, active hub, network hub, repeater hub, hub or concentrator is a device for connecting multiple twisted pair or fiber optic Ethernet devices together and making them act as single network segment. Hubs work at physical layer of the OSI model. The device is a form of multiport repeater. Repeater hubs also participate in collision detection, forwarding a jam signal to all ports if it detects a collision.

Experiment No: 3 STUDY OF NETWORK IP

Aim: Study of IP address of network.

- Classification of IP address
- Sub netting
- Super netting

Description: In the most widely installed level of the Internet Protocol (IP) today, an IP address is a 32-bit number that identifies each sender or receiver of information that is sent in packets across the Internet. When you request an HTML page or send e-mail, the Internet Protocol part of TCP/IP includes your IP address in the message (actually, in each of the packets if more than one is required) and sends it to the IP address that is obtained by looking up the domain name in the Uniform Resource Locator you requested or in the e-mail address you're sending a note to. At the other end, the recipient can see the IP address of the Web page requestor or the e-mail sender and can respond by sending another message using the IP address it received.

An IP address has two parts: the identifier of a particular network on the Internet and an identifier of the particular device (which can be a server or a workstation) within that network. On the Internet itself - that is, between the router that move packets from one point to another along the route - only the network part of the address is looked at.

The Network Part of the IP Address:

The Internet is really the interconnection of many individual networks (it's sometimes referred to as an internetwork). So the Internet Protocol (IP) is basically the set of rules for one network communicating with any other (or occasionally, for broadcast messages, all other networks). Each network must know its own address on the Internet and that of any other networks with which it communicates. To be part of the Internet, an organization needs an Internet network number, which it can request from the Network Information Center (NIC). This unique network number is included in any packet sent out of the network onto the Internet.

The Local or Host Part of the IP Address:
In addition to the network address or number, information is needed about which specific machine or host in a network is sending or receiving a message. So the IP address needs both the unique network number and a host number (which is unique within the network). (The host number is sometimes called a local or machine address.)

Part of the local address can identify a subnetwork or subnet address, which makes it easier for a network that is divided into several physical subnetworks (for examples, several different local area networks or) to handle many devices.

Apparatus: N/A
Procedure: Following is required to be study under this practical.
Classification of IP address: As show in figure-6 we teach how the IP addresses are classified and when they are used.

Class	Address Range	Supports
Class A	1.0.0.1 to 126.255.255.254	Supports 16 million hosts on each of 127 networks.
Class B	128.1.0.1 to 191.255.255.254	Supports 65,000 hosts on each of 16,000 networks.
Class C	192.0.1.1 to 223.255.254.254	Supports 254 hosts on each of 2 million networks.
Class D	224.0.0.0 to 239.255.255.255	Reserved for multicast groups.
Class E	240.0.0.0 to 254.255.255.254	Reserved.

Figure-6: Classification of IP address

Sub netting: Why we develop sub netting and how to calculate subnet mask to identify subnet address.
Super netting: Why we develop super netting and how to calculate super net mask and how to identify super net address.
Command: Open your computer →go to start button →write run then click on run →write 'CMD' then a black screen will appear then type ipconfig on the screen then you can see the IP address as well as subnet mask.

Input:
Command: C:>ipconfig

Output:

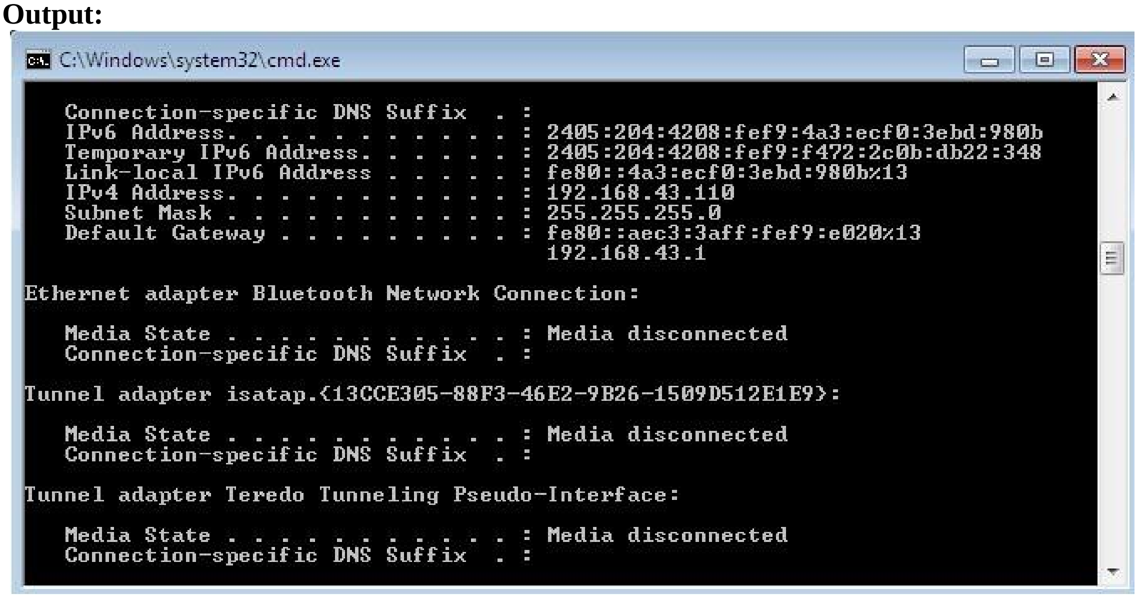


Figure-7: Output Screen

Experiment No-4 CONNECT THE COMPUTERS IN LOCAL AREA NETWORK

Description: A local-area network (LAN) is a computer network that spans a relatively small area. Most often, a LAN is confined to a single room, building or group of buildings, however, one LAN can be connected to other LANs over any distance via telephone lines and radio waves. A system of LANs connected in this way is called a wide-area network (WAN).

Nodes on a Local Area Network:

Most LANs connect workstations and personal computers. Each node (individual computer) in a LAN has its own CPU with which it executes programs, but it also is able to access data and devices anywhere on the LAN. This means that many users can share expensive devices, such as laser printers, as well as data. Users can also use the LAN to communicate with each other, by sending email or engaging in chat sessions.

LANs are capable of transmitting data at very fast rates, much faster than data can be transmitted over a telephone line; but the distances are limited and there is also a limit on the number of computers that can be attached to a single LAN.

Types of Local-Area Networks:

There are many different types of LANs, with **Ethernets** being the most common for PCs. Most Apple Macintosh networks are based on **Apple's AppleTalk** network system, which is built into Macintosh computers.

Aim: Connect two different computers in Local Area Network

Procedure:

On the host computer: On the host computer, follow these steps to share the internet connection:

1. Log on to the host computer as **Administrator** or as **Owner**.
2. Click **Start,** and then click **Control Panel**.
3. Click **Network and Internet Connections**.
4. Click **Network Connections**.
5. Right-click the connection that you use to connect to the internet. For example, if you connect to the Internet by using a modem, right-click the connection that you want under Dial-up/other network available.
6. Click **Properties**.
7. Click the **Advanced** Tab.
8. Under **Internet Connection Sharing**, select the **Allow other network users to connect through this computer's Internet connection** check box.
9. If you are sharing a dial-up Internet connection, select the **Establish a dial-up connection whenever a computer on my network attempts to access the Internet** check box if you want to permit your computer to automatically connect to the Internet.
10. Click **OK.** You receive the following message:

When Internet Connection Sharing is enabled, your LAN adapter will be set to use IP address 192.168.10.1. Your computer may lose connectivity with other computers on your network. If these other computers have static IP address, it is a good idea to set them to obtain their IP addresses automatically. Are you sure you want to enable Internet Connection sharing?

11. Click **YES.**

The connection to the Internet is shared to other computers on the local area network (LAN)
The network adapter that is connected to the LAN is configured with a static IP address of 192.168.0.1 and a subnet mask of 255.255.255.0.

On the client computer: To connect to the Internet by using the shared connection, you must confirm the LAN adapter IP configuration, and then configure the client computer. To confirm the LAN adapter IP configuration, follow these steps:

1. Log on to the client computer as **Administrator** or **Owner**.
2. Click **Start**, and then click **Control Panel**.
3. Click **Network and Internet Connections**.

4. Click **Network Connections.**
5. Right-click **Local Area Connection** and then click **properties.**
6. Click on **General** tab, click **Internet Protocol (TCP/IP)** in the connection uses the following items list, and then click **Properties**.
7. In the **Internet Protocol (TCP/IP) Properties** dialog box, click **Obtain an IP address automatically** (if it is not already selected), and then click **OK.**

Note: You can also assign unique static IP address in the range of 192.168.0.2 to 192.168.0.254. For example, you can assign the following static IP address, subnet mask, and default gateway:
IP Address 192.168.31.202
Subnet mask: 255.255.255.0
Default gateway 192.168.31.1

8. In the **Local Area Connection Properties** dialog box, click **OK.**
9. Quit **Control Panel**.

Experiment No-5 INTRODUCTION TO PACKET TRACER

Aim: Study of basic network command and Network configuration commands.

Description: Packet Tracer is a cross-platform visual simulation program designed by Cisco Systems that allows users to create network topologies and imitate modern computer networks. The software allows users to simulate the configuration of Cisco routers and switches using a simulated command line interface. Packet Tracer makes use of a drag and drop user interface, allowing users to add and remove simulated network devices as they see fit. The software is mainly focused towards Certified Cisco Network Associate Academy students as an educational tool for helping them learn fundamental CCNA concepts. Students enrolled in a CCNA Academy program can freely download and use the tool free of charge for educational use.

In addition to simulating certain aspects of computer networks, Packet Tracer can also be used for collaboration. As of Packet Tracer 5.0, Packet Tracer supports a multi-user system that enables multiple users to connect multiple topologies together over a computer network. Packet Tracer also allows instructors to create activities that students have to complete. Packet Tracer is often used in educational settings as a learning aid. Cisco Systems claims that Packet Tracer is useful for network experimentation

Apparatus (Software): Command Prompt and Packet Tracer.

Procedure: To do this experiment – follow these steps:
In this experiment students have to understand basic networking commands e.g ping, tracert etc.
All commands related to network configuration which includes how to switch to privilege mode and normal mode and how to configure router interface and how to save this configuration to flash memory or permanent memory.

This commands includes

- Configuring the Router commands
- General Commands to configure network
- Privileged Mode commands of a router
- Router Processes & Statistics
- IP Commands
- Other IP Commands e.g. show ip route etc.

Ping: ping sends an ICMP ECHO_REQUEST packet to the specified host. If the host responds, you get an ICMP packet back. Sound strange? Well, you can “ping” an IP address to see if a machine is alive. If there is no response, you know something is wrong.

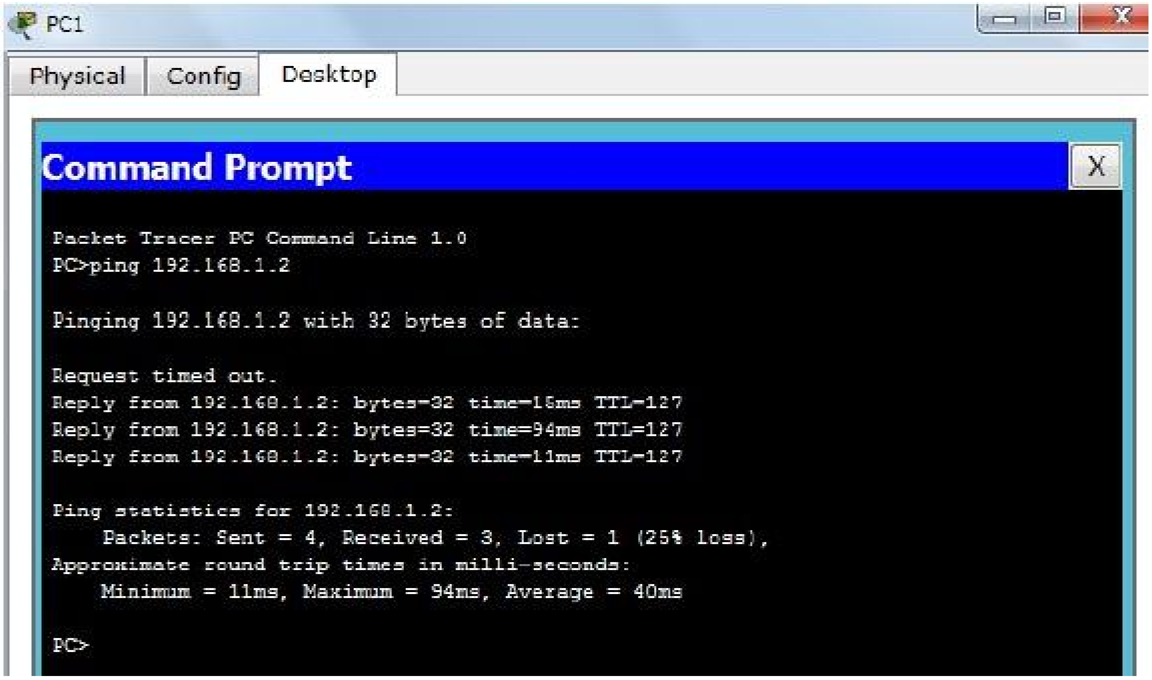


Figure-8: Command Prompt of PC1

Traceroute: Tracert is a command which can show you the path a packet of information takes from your computer to one you specify. It will list all the routers it passes through until it reaches its destination, or fails to and is discarded. In addition to this, it will tell you how long each 'hop' from router to router takes.

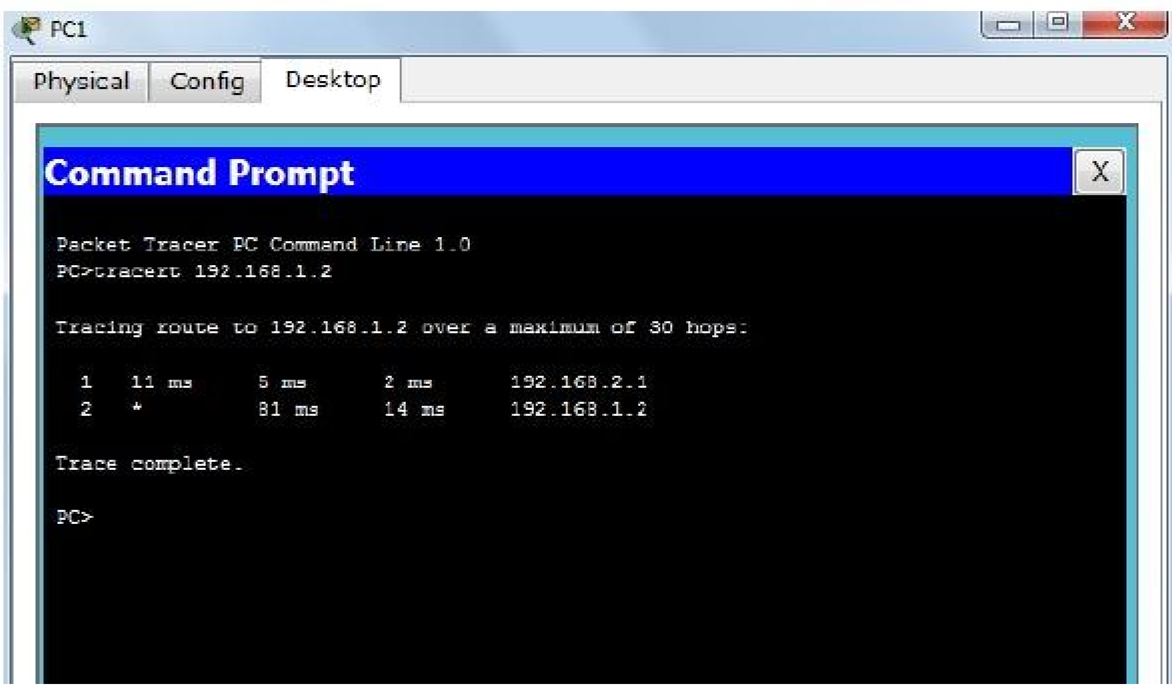


Figure-9: Command Prompt of PC1

Nslookup: Displays information from Domain Name System (DNS) name servers.
NOTE: If you write the command as above it shows as default your pc's server name firstly.
Pathping: A better version of tracert that gives you statistics about packet lost and latency.

Getting Help:
In any command mode, you can get a list of available commands by entering a question mark '?'.
Router>?
To obtain a list of commands that begin with a particular character sequence, type in those characters followed immediately by the question mark (?).
Router# co?
Configure connect copy
To list keywords or arguments, enter a question mark in place of a keyword or argument. Include a space before the question mark.
Router# configure?
Memory Configure from NV memory
Network configure from a TFTP network host
Terminal configure from the terminal
You can also abbreviate commands and keywords by entering characters to make the command unique from other commands. For example, you can abbreviate the **show** command to **sh.**

Configuration Files:
Any time you make changes to the router configuration, you must save the changes to memory because if you do not they will be lost if there is a system reload or power outage. There are two types of configuration files: the running (currently operating) configuration and the startup configuration.
Use the following privileged mode commands to work with configuration files.
Configuration terminal-modify the running configuration manually from the terminal.

- Show running-config- display the running configuration.
- Show startup-config- display the startup configuration
- Copy running-config startup-config- copy the running configuration to the startup configuration.
- Copy startup-config running- config- copy the start up configuration to the running configuration.
- Erase startup-config- erase the startup-configuration in NVRAM.

IP Address Configuration:
Take the following steps to configure the IP address of an interface.
Step-1: Enter privileged EXEC mode:
Router> **enable** password
Step-2: Enter the **configure terminal** command to enter global configuration mode.
Router# **config terminal**
Step-3: Enter the **interface** type slot/port or **interface** type port to enter the interface configuration mode.
Example:
Router(config)#**interface Ethernet 0/1**
Step-4: Enter the IP address and subnet mask of the interface using the **ip address** ipaddress subnetmask command.

Example:
Router(config-if)#**ip address 192.168.10.1 255.255.255.0**
Step-5: Exit the configuration mode by pressing Ctrl-Z
Router(config-if)#Ctrl-Z

Experiment No-6 CONFIGURE NETWORK TOPOLOGY USING PACKET TRACER
Aim: To configure network topology using packet tracer software
Description: Network Topology refers to layout of a network and how different nodes in a network are connected to each other and how they communicate. Topologies are either physical (the physical layout of devices on a network) or logical (the way that the signals act on the network media, or the way that the data passes through the network from one device to the next).
Some of the topologies are:

- Bus
- Mesh
- Star
- Ring
- Hybrid

Apparatus (Software): Packet Tracer 5.5.
Procedure: To implement this practical following network topology is required to be configured using the commands discussed above.
After configuring the given network a packet should be ping from any one machine to another.

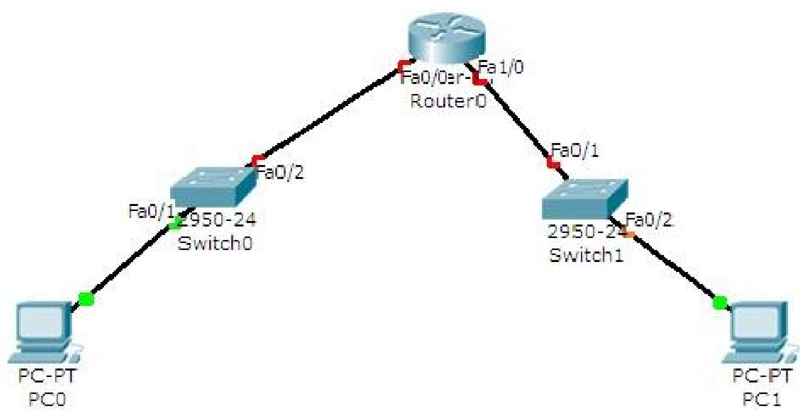


Figure-10: Network Topology

Router-0 Configuration Command:
Continue with configuration dialog? [yes/ no]: no

Press RETURN to get started!
Router>
Router>Enable
Router#config t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# hostname router0
router0(config) # interface fastethernet 0/0 **# Configuration of 0/0 port#**
router0(config-if) # ip address 192.168.1.1 255.255.255.0
router0(config-if) # no shutdown
router0(config-if)# exit
router0(config)# interface fastethernet 1/0 **# Configuration of 1/0 port#**
router0(config-if)# ip address 192.168.2.1 255.255.255.0
router0(config-if)# no shutdown
router0(config-if)# exit

router0(config)# exit
router0# show running-config
Building configuration….
Current configuration: 356 bytes
:
:
:
Version 12.4
Etc etc…….
Ip address 192.168.1.1 255.255.255.0
:
: interface VLAN
No ip address
:
:
:
End
router0#
router0#
router0# copy running-config startup-config
Destination filename[startup-config]?
Building configuration…
[OK]
router0#

IP Configuration of PC-0:

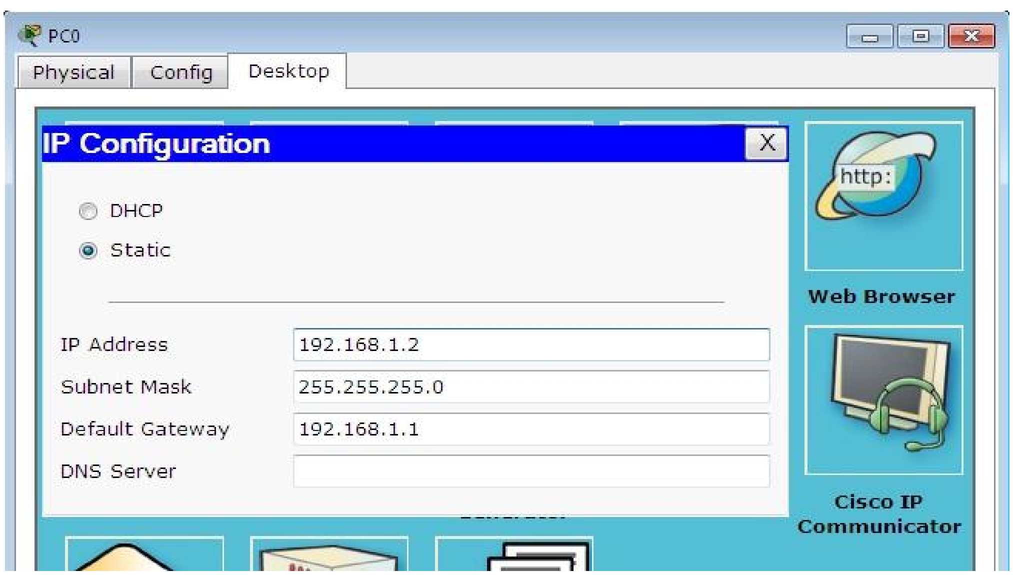


Figure-11: IP Configuration Window of PC0

Here the **IP address of PC-0 is:** 192.168.1.2
Default Gateway: 192.168.1.1 which is nothing but the one side (IP address of 0/0 port of router-0, towards PC-0) IP address of router-0

NOTE: Here PC-0 and 0/0 port of router-0 are in a same network that is why their network id is same(192.168.1)

IP Configuration of PC-1:

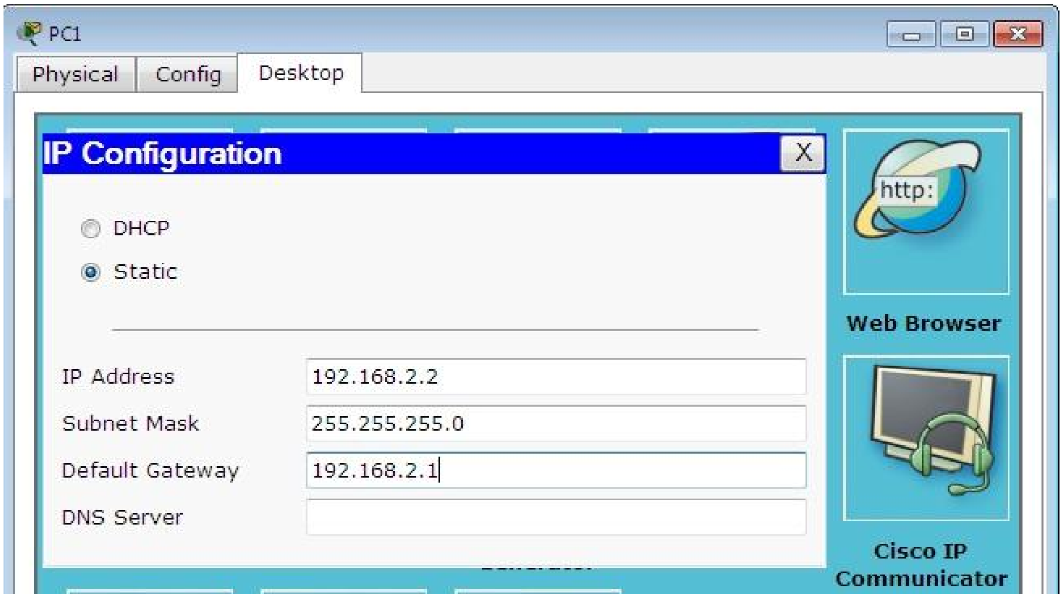


Figure-12: IP Configuration Window of PC1

Here the **IP address of PC-1 is:** 192.168.2.2
Default Gateway: 192.168.2.1 which is nothing but the one side (IP address of 1/0 port of router-0, towards PC-1) IP address of router-0

NOTE: Here PC-1 and 1/0 port of router-0 are in a same network that is why their network id is same (192.168.2)

Output:
After configuration the network looks like

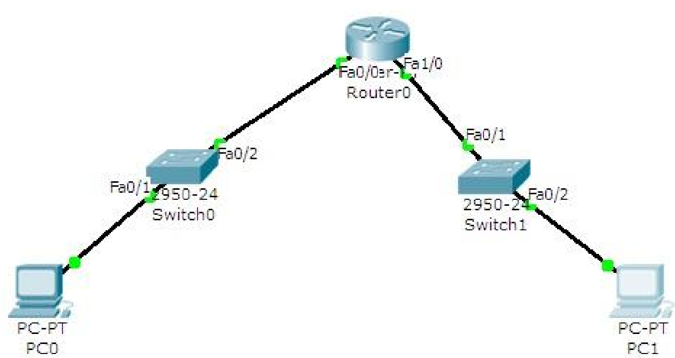


Figure-13: Network Topology

And we can ping PC-1 from PC-0 and If the connection is ok then then PC-0 will get reply from PC-1

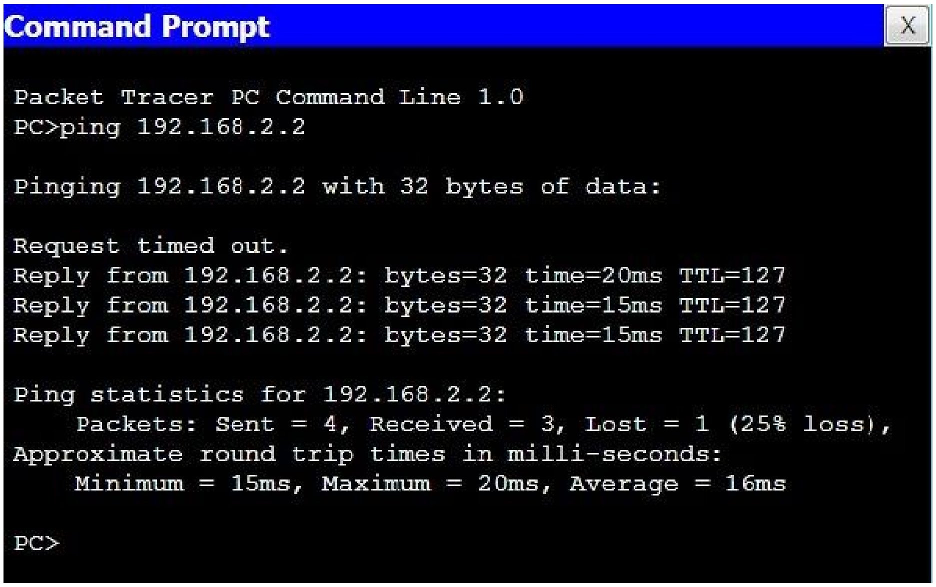


Figure-14: Output Screen

Experiment No-7 CONFIGURE NETWORK TOPOLOGY USING PACKET TRACER TO FIND THE ROUTING PATH BY IPROUTE COMMAND

Aim: To configure network topology to find the routing path.

Description: Routing is the process of selecting a path for traffic in a network, or between or across multiple networks. Routing is performed for many types of networks, including circuit-switched networks, such as the public switched telephone network (PSTN), computer networks, such as the Internet, as well as in networks used in public and private transportation, such as the system of streets, roads, and highways in national infrastructure.

Apparatus (Software): Packet Tracer 5.5.

Procedure: To implement this practical following network topology is required to be configured using the commands learned in previous experiment. After configuring the given network a packet should be ping from any one machine to another.

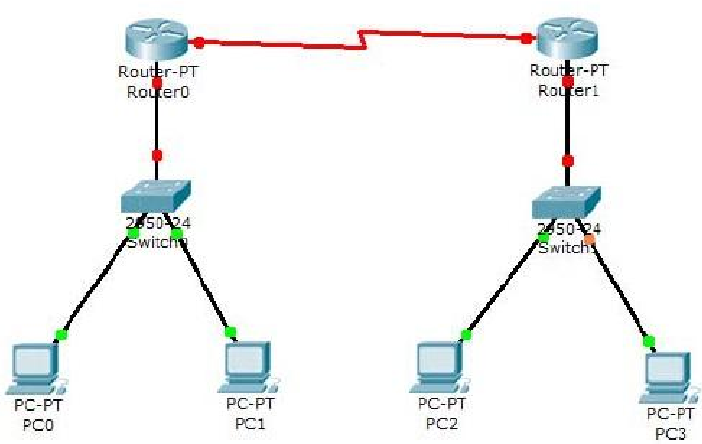


Figure-15: Network Topology.

Router-0 Configuration:

Router>
Router>Enable
Router>config t

Router(config)# hostname router0
router0(config)# interface fastethernet 0/0
router0(config-if)# ip address 192.168.0.254 255.255.255.0
router0(config-if)# no shutdown
router0(config-if)# exit
router0(config)# interface Serial2/0
router0(config-if)# ip address 192.168.1.1 255.255.255.0
router0(config-if)# no shutdown
router0(config-if)# exit
router0(config)# exit
router0#wr
Building configuration….
[OK]
router0# show running-config
< Student should write all comments/message to their laboratory note book after the execution of above command>

Router-1 Configuration:
Router> enable
Router#
Router# configure terminal
Router(config)# interface Serial2/0
Router(config-if)# ip address 192.168.1.2.255.255.255.0
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# exit
Router# config t
Router(config-if)# ip address 192.168.2.254 255.255.255.0
Router(config-if)# no shutdown
Router(config-if)#exit
Router(config)#exit
Router#wr
Building configuration…
[OK]
Router#
Router# show running-config
< Student should write all comments/message to their laboratory note book after the execution of above command>

IP ROUTE Command:

Router> enable
Router> show ip route
192.168.0.0/24[1/0] via 192.168.1.1
192.168.0.0/24 is directly connected, FastEthernet0/0
192.168.1.1/24 is directly connected, Serial2/0

Experiment No-8 NETWORK CONFIGURATION USING DISTANCE VECTOR ROUTING PROTOCOL

Aim: To configure a network using Distance Vector Routing Protocol. And the well known protocol is Routing Information Protocol (RIP)

Description: One of the most important examples of distance vector routing protocol is routing information protocol. The Routing Information Protocol (RIP) defines a way for routers, which connect networks using the Internet Protocol (IP), to share information about how to route traffic among networks. RIP is classified by the Internet Engineering Task Force (IETF) as an Interior Gateway Protocol (IGP), one of several protocols for routers moving traffic around within a larger autonomous system network -- e.g., a single enterprise's network that may be comprised of many separate local area networks (LANs) linked through routers.

Each RIP router maintains a routing table, which is a list of all the destinations (networks) it knows how to reach, along with the distance to that destination. RIP uses a distance vector algorithm to decide which path to put a packet on to get to its destination. It stores in its routing table the distance for each network it knows how to reach, along with the address of the "next hop" router -- another router that is on one of the same networks -- through which a packet has to travel to get to that destination. If it receives an update on a route, and the new path is shorter, it will update its table entry with the length and next-hop address of the shorter path; if the new path is longer, it will wait through a "hold-down" period to see if later updates reflect the higher value as well, and only update the table entry if the new, longer path is stable.

Apparatus (Software): Packet Tracer 5.5.

Procedure:

1. Develop a Topology shown in figure-16 given below
2. Configure all routers
3. Implement RIP protocol in Router in order to configure the network.

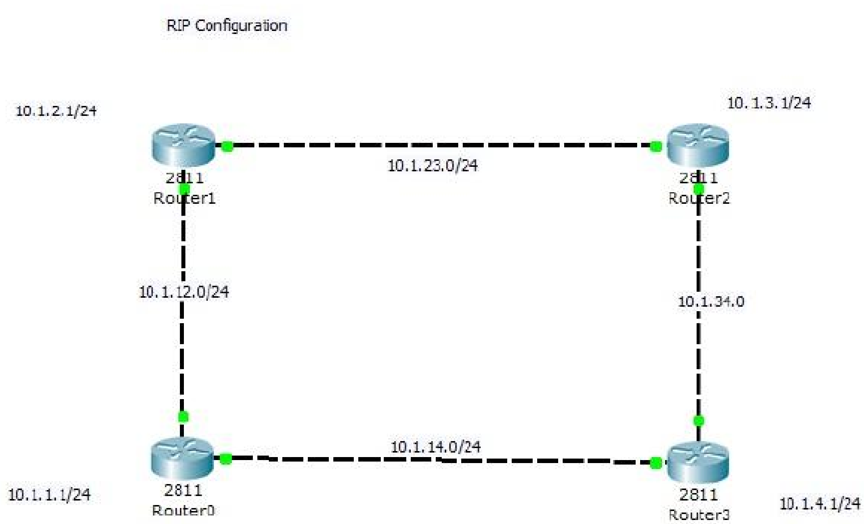


Figure-16: Network Topology.

Router-0 Configuration:

Router> en
Router> config t

```
router0(config)# int lo0
router0(config-if)#ip address 10.1.1.1 255.255.255.0
router0(config-if)#no shut
router0(config-if)#int f0/1
router0(config-if)#ip address 10.1.14.1 255.255.255.0
router0(config-if)#no shut
router0(config-if)#end
router0#wr
Building configuration…
[OK]
router0#
router0 con0 is now available
Press RETURN to get started.
router0>en
router0# config t
router0(config)# router rip
router0(config-router)#net 10.0.0.0
router0(config-router)#
router0(config-router)# end
router0#
```

Router-1 Configuration:

```
Router> en
Router> config t
Router(config)# hostname router1
router1 (config)# int lo0
router1 (config-if)#ip address 10.1.2.1 255.255.255.0
router1 (config-if)#no shut
router1 (config-if)#int f0/1
router1 (config-if)#ip address 10.1.23.1 255.255.255.0
router1 (config-if)#no shut
router1 (config-if)#int f0/0
router1 (config-if)# ip address 10.1.12.2 255.255.255.0
router1 (config-if)# no shut
router1 (config-if)#end
router1#wr
Building configuration…
[OK]
router1#
router1 con0 is now available
Press RETURN to get started.
router1>en
router1# config t
router1 (config)# router rip
router1 (config-router)#net 10.0.0.0
router1 (config-router)#
router1 (config-router)# end
router1#
```

Router-2 Configuration:

```
Router> en
Router> config t
Router(config)# hostname router2
```

router2 (config)# int lo0
router2 (config-if)#ip address 10.1.3.1   255.255.255.0
router2 (config-if)#no shut
router2 (config-if)#int f0/1
router2 (config-if)#ip address 10.1.34.1   255.255.255.0
router2 (config-if)#no shut
router2 (config-if)#int f0/0
router2 (config-if)# ip address 10.1.23.2   255.255.255.0
router2 (config-if)# no shut
router2 (config-if)#end
router2#wr
Building configuration…
[OK]
router2#
router2 con0 is now available
Press RETURN to get started.
router2>en
router2# config t
router2 (config)# router rip
router2 (config-router)#net 10.0.0.0
router2 (config-router)#
router2 (config-router)# end
router2#

Router-3 Configuration:

Router> en
Router> config t
Router(config)# hostname router3
router3 (config)# int lo0
router3 (config-if)#int f0/1
router3 (config-if)#ip address 10.1.14.2   255.255.255.0
router3 (config-if)#no shut
router3 (config-if)#int f0/0
router3 (config-if)# ip address 10.1.34.2   255.255.255.0
router3 (config-if)# no shut
router3 (config-if)#end
router3#wr
Building configuration…
[OK]
router3#
router3 con0 is now available
Press RETURN to get started.
router3>en
router3# config t
router3 (config)# router rip
router3 (config-router)#net 10.0.0.0
router3 (config-router)#
router3 (config-router)# end
router3#

Experiment No-9 CONFIGURATION OF DHCP PROTOCOL
Aim: To configure Dynamic Host Configuration Protocol in order to assign IP address to a PC or router dynamically.
Description: Dynamic Host Configuration Protocol (DHCP) is a protocol for assigning dynamic IP addresses to devices on a network. With dynamic addressing, a device can have a different IP address every time it connects to the network. In some systems, the device's IP address can even change while it is still connected. DHCP also supports a mix of static and dynamic IP addresses.
Apparatus (Software): Packet Tracer 5.5.
Procedure:
1. Develop a Topology shown in figure-17 given below
2. Configure DHCP in router0

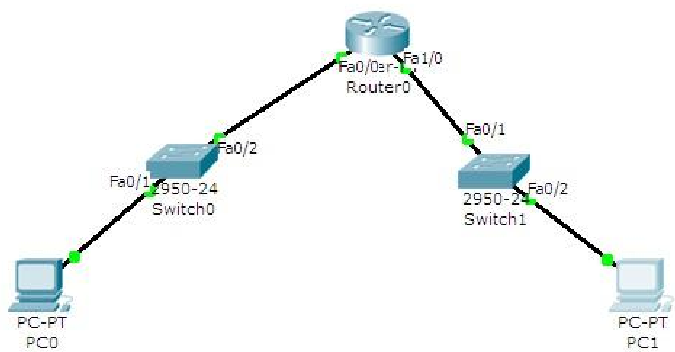


Figure-17: Network Topology.

Configuration of DHCP in Router-0

Router>en
Router# conf t
Router(config)# host R1
R1(config)# int fa1/0
R1(config-if)# ip add 192.168.10.1.2 255.255.255.0
R1(config-if)# no shutdown
R1(config-if)#exit
R1(config)# ip dhcp pool IP10 #pool of 10 Ip addresses #
R1(dhcp-config)#net 192.168.10.0 255.255.255.0
R1(dhcp-config)# default 192.168.10.1   #Router address#
R1(dhcp-config)#exit
R1(config)# ip dhcp exc 192.168.10.1 192.168.10.10
R1(config)# exit
R1#

Output from PC-1:
After the above configuration IP address of PC-1 will be assigned automatically after the request of IP address from the pool of IP address.

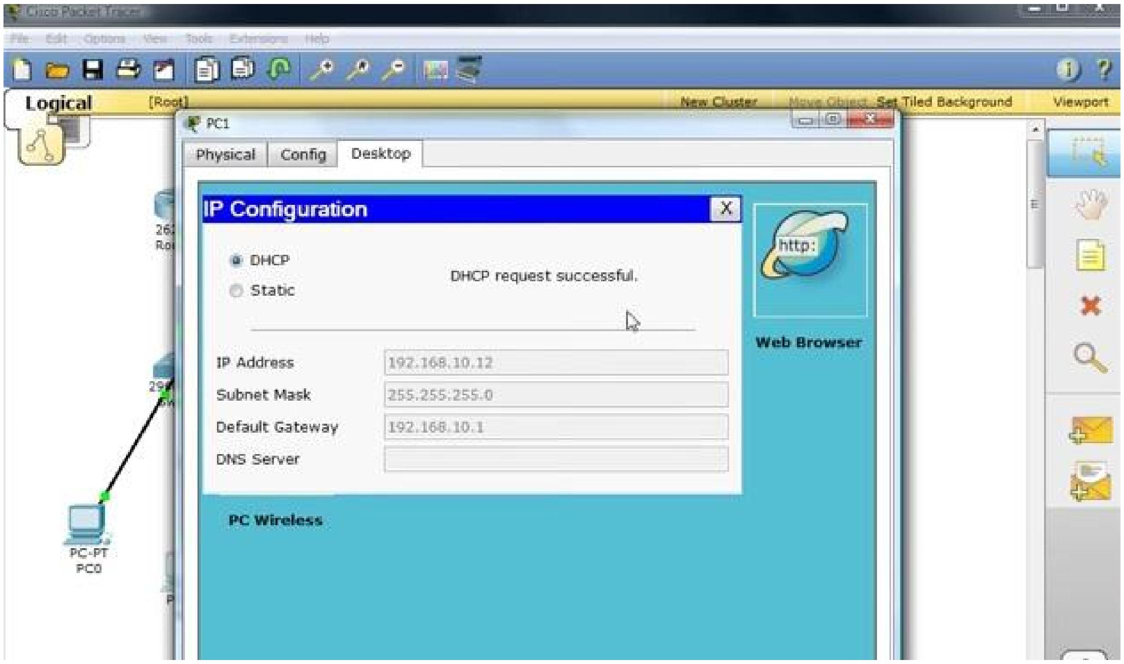


Figure-18: IP configuration box of PC1

Experiment No-10 TELNET CONFIGURATION
Aim: To configure TELNET Protocol in switch to get access of remote device
Description: Telnet is a terminal emulation program for TCP/IP networks such as the Internet. The Telnet program runs on your computer and connects your PC to a server on the network. You can then enter commands through the Telnet program and they will be executed as if you were entering them directly on the server console. This enables you to control the server and communicate with other servers on the network. To start a Telnet session, you must log in to a server by entering a valid username and password. Telnet is a common way to remotely control Web servers. The Telnet protocol is designed to provide a bi-directional, eight-bit byte oriented communications facility to allow for a standard method of interfacing terminal devices and processes. During Telnet configuration in switch we have to implement the concept of VLAN and its port.
About VLAN: A virtual LAN (VLAN) abstracts the idea of the LAN; A VLAN might comprise a subset of the ports on a single switch or subsets of ports on multiple switches. By default, systems on one VLAN don't see the traffic associated with systems on other VLANs on the same network. VLANs allow network administrators to partition their networks to match the functional and security requirements of their systems without having to run new cables or make major changes in their current network infrastructure. IEEE 802.1Q is the standard defining VLANs; the VLAN identifier or tag consists of 12 bits in the Ethernet frame, creating an inherent limit of 4,096 VLANs on a LAN. Ports on switches can be assigned to one or more VLANs, allowing systems to be divided into logical groups -- e.g., based on which department they are associated with -- and rules to be established about how systems in the separate groups are allowed to communicate with each other. These can range from the simple and practical (computers in one VLAN can see the printer on that VLAN, but computers outside that VLAN cannot), to the complex and legal (e.g., computers in the trading departments cannot interact with computers in the retail banking departments).

Apparatus (Software): Packet Tracer 5.5.
Procedure:
1. Develop a network (local area) shown in figure-19 given below
2. Configure TELNET in switch

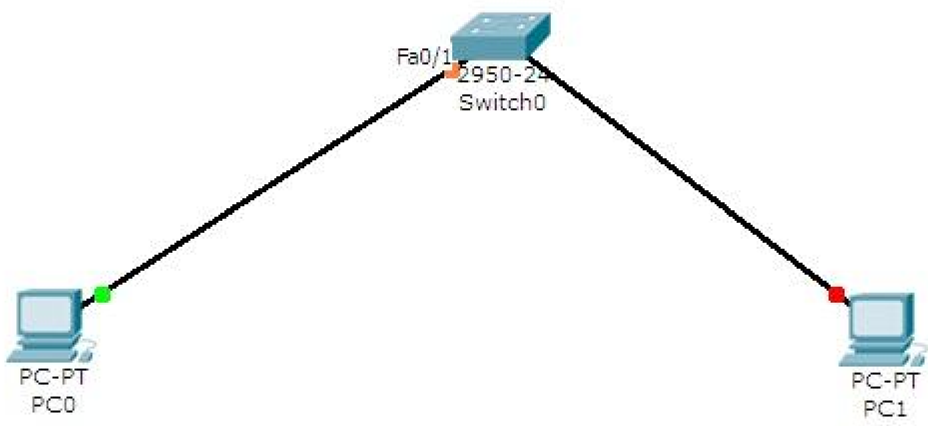


Figure-19: Network Topology

Switch Configuration:
Switch>
Switch>en
Switch# conf t
Switch(config)# line console 0
Switch(config-line)# pass cisco
Switch(config-line)# login
Switch(config-line)# exit
Switch(config)# line vty 0 4
Switch(config-line)# pass cisco
Switch(config-line)# login
Switch(config-line)# exit
Switch(config)# enable pass cisco
Switch(config)# int vlan 1
Switch(config-if)# ip add 172.17.1.1 255.255.0.0
Switch(config-if)# no shutdown
Switch(config-if)#exit
Switch(config)#exit
Switch# wr
Building Configuration...

Input & Output:
From PC-0 we can easily access remote switch by using TELNET.
Input: PC-0>ping 172.17.1.1

Figure-20: Output Screen of TELNET Configuration Test

Experiment No-11 CONFIGURATION OF ACCESS COMTROL LIST

Aim: To configure Access Control List (ACL) in order to give permission to the other IP addresses.

Description: An access control list (ACL), with respect to a computer file system, is a list of permissions attached to an object. An ACL specifies which users or system processes are granted access to objects, as well as what operations are allowed on given objects or you can say for each rule we have two conditions and that is Permit or Deny.

Types of Access Lists

There are two categories of access lists: numbered and named.

Numbered Access Lists:-

Numbered access lists are broken down into several ranges, each dedicated to a specific protocol:

- 1–99 IP standard access list
- 100-199 IP extended access list
- 200-299 Protocol type-code access list
- 300-399 DECnet access list
- 400-499 XNS standard access list
- 500-599 XNS extended access list
- 600-699 Appletalk access list
- 700-799 48-bit MAC address access list
- 800-899 IPX standard access list
- 900-999 IPX extended access list
- 1000-1099 IPX SAP access list
- 1100-1199 Extended 48-bit MAC address access list
- 1200-1299 IPX summary address access list
- 1300-1999 IP standard access list (expanded range)
- 2000-2699 IP extended access list

Named Access Lists:-

Named access lists provide a bit more flexibility. Descriptive names can be used to identify your access-lists. Additionally, individual lines can be removed from a named access-list. However, like numbered lists, all new entries are still added to the bottom of the access list.

There are two common types of named access lists:

• IP standard named access lists

• IP extended named access lists

Apparatus (Software): Packet Tracer 5.5.

Procedure:

Standard IP Access List:

Command Syntax: access-list [1-99] [permit | deny] [source address] [wildcard mask] [log]

Standard IP access-lists are based upon the source host or network IP address, and should be placed closest to the destination network.

Router(config)# access-list 10 deny 172.18.0.0 0.0.255.255 **(Just for an Example)**

Router(config)# access-list 10 permit any

To apply Access Lists we have to configure the Access-Group on the Interface. Likewise we are taking the interface serial 0 as a reference.

Router(config)# int s0

Router(config-if)# ip access-group 10 in

To view all IP access lists configured on the router:

Router# show ip access-list

To view what interface an access-list is configured on:

Router# show ip interface

Router# show running-config

Extended IP Access List:

Command Syntax: access-list [100-199] [permit | deny] [protocol] [source address] [wildcard mask] [destination address] [wildcard mask] [operator [port]] [log]

Router(config)# access-list 101 permit tcp 172.18.0.0 0.0.255.255 host 172.16.10.10 eq 80

Router(config)# access-list 101 deny ip 172.18.0.0 0.0.255.255 172.16.0.0 0.0.255.255

Router(config)# access-list 101 permit ip any any

NOTE: *The above IP address is just taken for the example and don't have real environment existence.

The first line allows the 172.18.x.x network access only to port 80 on the web server. The second line blocks 172.18.x.x from accessing anything else on the 172.16.x.x network. The third line allows 172.18.x.x access to anything else.

To apply this access list, we would configure the following

Router(config)# int e0

Router(config-if)# ip access-group 101 in

Extended IP Access List Port Operators

Router(config)# access-list 101 permit tcp 172.18.0.0 0.0.255.255 host 172.16.10.10 eq 80
We accomplished this using an operator of eq, which is short for equals. Thus, we are identifying host 172.16.10.10 with a port that equals 80. We can use several other operators for port numbers:

- **eq :-** Matches a specific port
- **gt :-** Matches all ports greater than the port specified
- **lt :-** Matches all ports less than the port specified
- **neq :-** Matches all ports except for the port specified
- **range:-** Match a specific inclusive range of ports

ICMP Access List

The specific ICMP port that a "ping" uses is echo. To block specific ICMP parameters, use anextended IP access list. On Router B, we would configure:

Router(config)# access-list 102 deny icmp 172.18.0.0 0.0.255.255 172.16.0.0 0.0.255.255 echo
Router(config)# access-list 102 permit icmp 172.18.0.0 0.0.255.255 172.16.0.0 0.0.255.255
Router(config)# access-list 102 permit ip any any

The first line blocks only ICMP echo requests (pings). The second line allows all other ICMP traffic. The third line allows all other IP traffic.
To apply the access lists on other router, you need to configure the following as:-
Router(config)# int e0
Router(config-if)# ip access-group 102 in

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Lab Manual

Title of Course: Operating System Lab
Course Code: CS693
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.

Course Contents:
Exercises that must be done in this course are listed below:
Exercise No.1: CPU scheduling
Exercise No. 2: File allocation Strategy
Exercise No. 3: Simulate MVT, MFT(Multiprogramming Fixed and Variable)
Exercise No. 4: Simulate all File Organization Techniques
Exercise No. 5: Simulate Banker’s Algorithm for Dead Lock Avoidance
Exercise No. 6: Simulate Banker’s Algorithm for Dead Lock Prevention
Exercise No. 7: Simulate all page replacement Strategies
Exercise No. 8: Simulate Paging Technique of Memory Management
Exercise No. 9: Shell programming
Exercise No. 10: Process
Text Book:
1. Maurice J. Bach, Design of the UNIX Operating System, PHI.

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. Turbo C or TC3 complier in Windows XP or Linux Operating System.

Experiment No: 1(a) ROUND ROBIN SCHEDULING

Aim: Write a C program to implement the various process scheduling mechanisms such as **Round Robin Scheduling.**

Description:

Scheduling is a fundamental operating system function. CPU scheduling is the basis of multi programming operating system. CPU scheduling algorithm determines how the CPU will be allocated to the process. These are of two types.

1. Primitive scheduling algorithms
2. Non-Primitive scheduling algorithms

1) **Primitive Scheduling algorithms**: In this, the CPU can release the process even in the middle of execution. For example: the cpu executes the process p1, in the middle of execution the cpu received a request signal from process p2, then the OS compares the priorities of p1&p2. If the priority p1 is higher than the p2 then the cpu continue the execution of process p1.Otherwise the cpu preempt the process p1 and assigned to process p2.

2) **Non-Primitive Scheduling algorithm**: In this, once the cpu assigned to a process the processor do not release until the completion of that process. The cpu will assign to some other job only after the previous job has finished.

Scheduling methodology:

Though put: It means how many jobs are completed by the CPU with in a time period.

Turnaround time: The time interval between the submission of the process and the time of the completion is the turnaround time.

Turnaround time=Finished time – arrival time

Waiting time: it is the sum of the periods spent waiting by a process in the ready queue

Waiting time=Starting time- arrival time

Response time: it is the time duration between the submission and first response

Response time=First response-arrival time

CPU Utilization: This is the percentage of time that the processor is busy. CPU utilization may range from 0 to 100%.

Round Robin: It is a primitive scheduling algorithm it is designed especially for time sharing systems. In this, the CPU switches between the processes. When the time quantum expired, the CPU switches to another job. A small unit of time called a quantum or time slice. A time quantum is generally is a circular queue new processes are added to the tail of the ready queue. If the process may have a CPU burst of less than one time slice then the process release the CPU voluntarily. The scheduler will then process to next process ready queue otherwise; the process will be put at the tail of the ready queue**.**

Algorithm for Round Robin:

Step 1: Start the process
Step 2: Accept the number of processes in the ready Queue and time quantum (or) time slice
Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time
Step 4: Calculate the no. of time slices for each process where No. of time slice for process(n) = burst time process(n)/time slice
Step 5: If the burst time is less than the time slice then the no. of time slices =1.
Step 6: Consider the ready queue is a circular Q, calculate
(a) Waiting time for process(n) = waiting time of process(n-1)+ burst time of process(n-1) + the time difference in getting the CPU from process(n-1)
(b) Turnaround time for process(n) = waiting time of process(n) + burst time of process(n)+ the time difference in getting CPU from process(n).
Step 7: Calculate
(a) Average waiting time = Total waiting Time / Number of process
(b) Average Turnaround time = Total Turnaround Time / Number of process
Step 8: Stop the process

/* Program to Simulate Round Robin CPU Scheduling Algorithm */

```
#include<stdio.h>
```

```
struct process
{
char pn[10];
int bt,ct,time;
}p[10];
void main()
{
int i,full,n,tq,wt[10],tat[10], time1=0;
float avgwt=0.0;
clrscr();
printf("Enter number of processes:");
scanf("%d",&n);
printf("Enter process name and burst time of %d process\n", n);
for(i=0;i<n;i++)
{
scanf("%s%d",&p[i].pn,&p[i].bt);
p[i].time=p[i].bt;
}
printf("Enter quantum:");
scanf("%d",&tq);
full=n;
while(full)
{
for(i=0;i<n;i++)
{
if(p[i].bt>=tq)
{
p[i].bt-=tq;
time1=time1+tq;
}
else if(p[i].bt!=0)
{
time1+=p[i].bt;
p[i].bt=0;
}
else
continue;
if(p[i].bt==0)
{
full=full-1;
tat[i]=time1;
}
}
}
for(i=0;i<n;i++)
{
p[i].ct=tat[i];
wt[i]=tat[i]-p[i].time;
}
printf("----------------------------------\n");
printf("PN\tBt\tCt\tTat\tWt\n");
printf("----------------------------------\n");
for(i=0;i<n;i++)
{
printf("%2s\t%2d\t%2d\t%2d\t%2d\n",p[i].pn,p[i].time,p[i].ct,tat[i],wt[i]);
avgwt+=wt[i];
}
```

```
avgwt=avgwt/n;
printf(" Average waiting time = %.2f\n",avgwt);
printf("------------------------------------\n");
getch();
}
```

INPUT 1:

Enter number of processes: 5
Enter process name and burst time of 5 processes
1 10
2 5
3 15
4 3
5 20
Enter quantum: 5

OUTPUT 1:

PN	Bt	Rt	Tat	Wt
1	10	28	28	18
2	5	10	10	5
3	15	43	43	28
4	3	18	18	15
5	20	53	53	33

Average waiting time = 19.79

INPUT 2:

Enter number of processes:3
Enter process name and burst time of 3 processes
1 24
2 3
3 3
Enter quantum: 4

OUTPUT 2:

PN	Bt	Ct	Tat	Wt
1	24	30	30	6
2	3	7	7	4
3	3	10	10	7

Average waiting time = 5.67

Experiment No: 1(b) SJF SCHEDULING
Aim: Write a C program to implement the various process scheduling mechanisms such as SJF Scheduling .

Description:
Shortest Job First: The criteria of this algorithm are which process having the smallest CPU burst, CPU is assigned to that next process. If two process having the same CPU burst time FCFS is used to break the tie.
Algorithm for SJF:

Step 2: Accept the number of processes in the ready Queue
Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time
Step 4: Start the Ready Q according the shortest Burst time by sorting according to lowest to highest burst time.
Step 5: Set the waiting time of the first process as ‘0’ and its turnaround time as its burst time.
Step 6: For each process in the ready queue, calculate
(a) Waiting time for process (n)= waiting time of process (n-1) + Burst time of process(n-1)
(b) Turnaround time for Process(n)= waiting time of Process(n)+ Burst time for process(n)
Step 6: Calculate
(c) Average waiting time = Total waiting Time / Number of process
(d) Average Turnaround time = Total Turnaround Time / Number of process
Step 7: Stop the process

/* Program to Simulate Shortest Job First CPU Scheduling Algorithm */

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
int i,j,n,bt[10],compt[10], wt[10],tat[10],temp;
float sumwt=0.0,sumtat=0.0,avgwt,avgtat;
clrscr();
printf("Enter number of processes: ");
scanf("%d",&n);
printf("Enter the burst time of %d process\n", n);
for(i=0;i<n;i++)
{
scanf("%d",&bt[i]);
for(i=0;i<n;i++)
for(j=i+1;j<n;j++)
if(bt[i]>bt[j])
{
temp=bt[i];
bt[i]=bt[j];
bt[j]=temp;
}
compt[0]=bt[0];
for(i=1;i<n;i++)
compt[i]=bt[i]+compt[i-1];
for(i=0;i<n;i++)
{
tat[i]=compt[i];
wt[i]=tat[i]-bt[i];
sumtat+=tat[i];
sumwt+=wt[i];
}
avgwt=sumwt/n;
avgtat=sumtat/n;
printf("------------------------------\n");
printf("Bt\tCt\tTat\tWt\n");
printf("------------------------------\n");
for(i=0;i<n;i++)
{
printf("%2d\t%2d\t%2d\t%2d\n",i,bt[i],compt[i],tat[i],wt[i]);
}
printf("------------------------------\n");
printf(" Avgwt = %.2f\tAvgtat = %.2f\n",avgwt,avgtat);
```

```
getch();
}
```

OUTPUT 1:

```
Enter number of processes: 4
Enter the burst time of 4 processes
6 8 7 3
------------------------------------
Bt   Ct   Tat   Wt
------------------------------------
3     3    3    0
6     9    9    3
7    16   16    9
8    24   24   16
--------------------------------------
Avgwt = 7.00 Avgtat = 13.00
--------------------------------------
```

OUTPUT 2:

```
Enter number of processes: 4
Enter the burst time of 4 processes
8 4 9 5
------------------------------------
Bt    Ct   Tat   Wt
------------------------------------
4     4    4    0
5     9    9    4
8    17   17    9
9    26   26   17
------------------------------------
Avgwt = 7.50 Avgtat = 14.00
------------------------------------
```

Experiment No: 1(c) FCFS SCHEDULING
Aim: Write a C program to implement the various process scheduling mechanisms such

Description:
First-come, first-serve scheduling (FCFS): In this, which process enter the ready queue first is served first. The OS maintains DS that is ready queue. It is the simplest CPU scheduling algorithm. If a process request the CPU then it is loaded into the ready queue, which process is the head of the ready queue, connect the CPU to that process.

Algorithm for FCFS scheduling:
Step 1: Start the process
Step 2: Accept the number of processes in the ready Queue
Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time
Step 4: Set the waiting of the first process as ‘0’ and its burst time as its turnaround time
Step 5: for each process in the Ready Q calculate
(c) Waiting time for process (n) = waiting time of process (n-1) + Burst time of process (n-1)
(d) Turnaround time for Process(n)= waiting time of Process(n)+ Burst time for process(n)
Step 6: Calculate
(e) Average waiting time = Total waiting Time / Number of process
(f) Average Turnaround time = Total Turnaround Time / Number of process
Step 7: Stop the process

/* Program to Simulate First Come First Serve CPU Scheduling Algorithm */

```
#include<stdio.h>
```

```
#include<string.h>
void main()
{
int i,j,n,bt[10],compt[10],at[10], wt[10],tat[10];
float sumwt=0.0,sumtat=0.0,avgwt,avgtat;
clrscr();
printf("Enter number of processes: ");
scanf("%d",&n);
printf("Enter the burst time of %d process\n", n);
for(i=0;i<n;i++)
{
scanf("%d",&bt[i]);
}
printf("Enter the arrival time of %d process\n", n);
for(i=0;i<n;i++)
scanf("%d",&at[i]);
}
compt[0]=bt[0]-at[0];
for(i=1;i<n;i++)
compt[i]=bt[i]+compt[i-1];
for(i=0;i<n;i++)
{
tat[i]=compt[i]-at[i];
wt[i]=tat[i]-bt[i];
sumtat+=tat[i];
sumwt+=wt[i];
}
avgwt=sumwt/n;
avgtat=sumtat/n;
printf("----------------------------------\n");
printf("PN\tBt\tCt\tTat\tWt\n");
printf("----------------------------------\n");
for(i=0;i<n;i++)
{
printf("%d\t%2d\t%2d\t%2d\t%2d\n",i,bt[i],compt[i],tat[i],wt[i]);
}
printf("----------------------------------\n");
printf(" Avgwt = %.2f\tAvgtat = %.2f\n",avgwt,avgtat);
printf("-----------------------------------\n");
getch();
}
```

Output 1:

```
Enter number of processes: 5
Enter the burst time of 5 processes
3 6 4 5 2
Enter the arrival time of 5 processes
0 2 4 6 8
----------------------------------------
PN    Bt    Ct    Tat    Wt
----------------------------------------
0     3      3     3     0
1     6      9     7     1
2     4     13     9     5
3     5     18    12     7
4     2     20    12    10
---------------------------------------
```

```
------------------------------------
```

Output 2:

```
Enter number of processes: 5
Enter the burst time of 5 processes
5 24 16 10 3
Enter the arrival time of 5 processes
0 0 0 0 0
--------------------------------------
PN     Bt   Ct    Tat    Wt
--------------------------------------
0       5     5      5      0
1      24    29    29     5
2      16    45    45    29
3      10    55    55    45
4       3    58    58    55
--------------------------------------
Avgwt = 26.80 Avgtat = 38.40
--------------------------------------
```

Experiment No: 1(d) PRIORITY SCHEDULING

Aim: Write a C program to implement the various process scheduling mechanisms such as Priority Scheduling.

Description:

Priority Scheduling: These are of two types.

One is internal priority, second is external priority. The cpu is allocated to the process with the highest priority. Equal priority processes are scheduled in the FCFS order. Priorities are generally some fixed range of numbers such as 0 to 409. The low numbers represent high priority.

Algorithm for Priority Scheduling:

Step 1: Start the process
Step 2: Accept the number of processes in the ready Queue
Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time
Step 4: Sort the ready queue according to the priority number.
Step 5: Set the waiting of the first process as ‘0’ and its burst time as its turnaround time
Step 6: For each process in the Ready Q calculate
(e) Waiting time for process (n)= waiting time of process (n-1) + Burst time of process(n-1)
(f) Turnaround time for Process (n) = waiting time of Process(n)+ Burst time for process(n)
Step 7: Calculate
(g) Average waiting time = Total waiting Time / Number of process
(h) Average Turnaround time = Total Turnaround Time / Number of process
Step 8: Stop the process

/* Program to Simulate Priority CPU Scheduling Algorithm */

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,j,n,bt[10],p[10],compt[10], wt[10],tat[10],temp1,temp2;
float sumwt=0.0,sumtat=0.0,avgwt,avgtat;
clrscr();
printf("Enter number of processes: ");
scanf("%d",&n);
printf("Enter the burst time of %d process\n", n);
for(i=0;i<n;i++)
scanf("%d",&bt[i]);
```

```
for(i=0;i<n;i++)
scanf("%d",&p[i]):
for(i=0;i<n;i++)
for(j=i+1;j<n;j++)
if(p[i]>p[j])
{
temp1=bt[i];
bt[i]=bt[j];
bt[j]=temp1;
temp2=p[i];
p[i]=p[j];
p[j]=temp2;
}
compt[0]=bt[0]; wt[0]=0;
for(i=1;i<n;i++)
compt[i]=bt[i]+compt[i-1];
for(i=0;i<n;i++)
{
tat[i]=compt[i];
wt[i]=tat[i]-bt[i];
sumtat+=tat[i];
sumwt+=wt[i];
}
avgwt=sumwt/n; avgtat=sumtat/n;
printf("------------------------------\n");
printf("Bt\tCt\tTat\tWt\n");
printf("------------------------------\n");
for(i=0;i<n;i++)
{
printf("%2d\t%2d\t%2d\t%2d\n",bt[i],compt[i],tat[i],wt[i]);
}
printf("------------------------------\n");
printf(" Avgwt = %.2f\tAvgtat = %.2f\n",avgwt,avgtat);
printf("-------------------------------\n");
getch();
}
```

OUTPUT 1:

```
Enter number of processes: 4
Enter the burst time of 4 processes
6 5 3 5
Enter the priority of 4 processes
4 2 6 3
------------------------------------
Bt    Ct   Tat   Wt
-----------------------------------
5      5     5    0
5     10    10    5
6     16    16   10
3     19    19   16
------------------------------------
Avgwt = 7.75 Avgtat = 12.50
-----------------------------------
```

OUTPUT 2:

```
Enter number of processes: 5
Enter the burst time of 5 processes
```

```
Enter the priority of 5 processes
3 1 4 5 2
------------------------------------
Bt    Ct    Tat   Wt
------------------------------------
1     1     1     0
5     6     6     1
10    16    16    6
2     18    18    16
1     19    19    18
------------------------------------
Avgwt = 8.20 Avgtat = 12.0
------------------------------------
```

Experiment No:2(a)SEQUENTIAL FILE ALLOCATION
AIM: Write a C Program to implement Sequential File Allocation method.

Description:
Files are normally stored on the disks. So the main problem is how to allocate space to those files. So that disk space is utilized effectively and files can be accessed quickly. Three major strategies of allocating disc space are in wide use. Sequential, indexed and linked.
Sequential allocation:
In this allocation strategy, each file occupies a set of contiguously blocks on the disk. This strategy is best suited. For sequential files, the file allocation table consists of a single entry for each file. It shows the filenames, staring block of the file and size of the file. The main problem of this strategy is, it is difficult to find the contiguous free blocks in the disk and some free blocks could happen between two files.
Algorithm for Sequential File Allocation:
Step 1: Start the program.
Step 2: Get the number of memory partition and their sizes.
Step 3: Get the number of processes and values of block size for each process.
Step 4: First fit algorithm searches all the entire memory block until a hole which is big enough is encountered. It allocates that memory block for the requesting process.
Step 5: Best-fit algorithm searches the memory blocks for the smallest hole which can be allocated to requesting process and allocates if.
Step 6: Worst fit algorithm searches the memory blocks for the largest hole and allocates it to the process.
Step 7: Analyses all the three memory management techniques and display the best algorithm which utilizes the memory resources effectively and efficiently.
Step 8: Stop the program.

/* Program to simulate sequential file allocation strategy */
```
#include < stdio.h>
#include<conio.h>
void main()
{
int f[50], i, st, len, j, c, k, count = 0;
clrscr();
for(i=0;i<50;i++)
f[i]=0;
printf("Files Allocated are : \n");
x: count=0;
printf("Enter starting block and length of files: ");
scanf("%d%d", &st,&len);
for(k=st;k<(st+len);k++)
if(f[k]==0)
count++;
```

```
{
for(j=st;j<(st+len);j++)
if(f[j]==0)
{
f[j]=1;
printf("%d\t%d\n",j,f[j]);
}
if(j!=(st+len-1))
printf(" The file is allocated to disk\n");
}
else
printf(" The file is not allocated \n");
printf("Do you want to enter more file(Yes - 1/No - 0)");
scanf("%d", &c);
if(c==1)
goto x;
else
exit();
getch();
}
```

OUTPUT 1:

Files Allocated are:
Enter starting block and length of files: 14 3
14 1
15 1
16 1
The file is allocated to disk
Do you want to enter more file (Yes - 1/No - 0)1
Enter starting block and length of files: 14 1
The file is not allocated
Do you want to enter more file (Yes - 1/No - 0)1
Enter starting block and length of files: 14 4
The file is not allocated
Do you want to enter more file (Yes - 1/No - 0)0

OUTPUT 2:

Files Allocated are:
Enter starting block and length of files: 17 4
17 1
18 1
19 1
20 1
The file is allocated to disk
Do you want to enter more file(Yes - 1/No - 0)1
Enter starting block and length of files: 21 3
21 1
22 1
23 1
The file is allocated to disk
Do you want to enter more file (Yes - 1/No - 0)1
Enter starting block and length of files: 19 4
The file is not allocated
Do you want to enter more file (Yes - 1/No - 0)1
Enter starting block and length of files: 25 5
25 1
26 1

28 1
29 1
The file is allocated to disk
Do you want to enter more file(Yes - 1/No – 0)0

Experiment No: 2(b) INDEXED FILE ALLOCATION
AIM: Write a C Program to implement Indexed File Allocation method.

Description:
Indexed allocation:
Indexed allocation supports both sequential and direct access files. The file indexes are not physically stored as a part of the file allocation table. Whenever the file size increases, we can easily add some more blocks to the index. In this strategy, the file allocation table contains a single entry for each file. The entry consisting of one index block, the index blocks having the pointers to the other blocks. No external fragmentation.
Algorithm for Indexed File Allocation:
Step 1: Start.
Step 2: Let n be the size of the buffer
Step 3: check if there are any producer
Step 4: if yes check whether the buffer is full
Step 5: If no the producer item is stored in the buffer
Step 6: If the buffer is full the producer has to wait
Step 7: Check there is any consumer. If yes check whether the buffer is empty
Step 8: If no the consumer consumes them from the buffer
Step 9: If the buffer is empty, the consumer has to wait.
Step 10: Repeat checking for the producer and consumer till required
Step 11: Terminate the process.

```
/* Program to simulate indexed file allocation strategy */
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void main()
{
int f[50], index[50],i, n, st, len, j, c, k, ind,count=0;
clrscr();
for(i=0;i<50;i++)
f[i]=0;
x:printf("Enter the index block: ");
scanf("%d",&ind);
if(f[ind]!=1)
{
printf("Enter no of blocks needed and no of files for the index %d on the disk : \n", ind);
scanf("%d",&n);
}
Else{
printf("%d index is already allocated \n",ind);
goto x;
}
y: count=0;
for(i=0;i<n;i++)
{
scanf("%d", &index[i]);
if(f[index[i]]==0)
count++;
}
if(count==n)
```

```
for(j=0;j<n;j++)
f[index[j]]=1;
printf("Allocated\n");
printf("File Indexed\n");
for(k=0;k<n;k++)
printf("%d-------->%d : %d\n",ind,index[k],f[index[k]]);
}
else
{
printf("File in the index is already allocated \n");
printf("Enter another file indexed");
goto y;
}
printf("Do you want to enter more file(Yes - 1/No - 0)");
scanf("%d", &c);
if(c==1)
goto x;
else
exit(0);
getch();
}
```

OUTPUT 1:

```
Enter the index block: 5
Enter no of blocks needed and no of files for the index 5 on the disk:
4
1 2 3 4
Allocated
File Indexed
5-------->1: 1
5-------->2: 1
5-------->3: 1
5-------->4: 1
Do you want to enter more file (Yes - 1/No - 0)1
Enter the index block: 4
4 indexes is already allocated
Enter the index block: 6
Enter no of blocks needed and no of files for the index 6 on the disk :
2
7 8
A5llocated
File Indexed
6-------->7: 1
6-------->8: 1
Do you want to enter more file (Yes - 1/No - 0)0
```

OUTPUT 2:

```
Enter the index block: 4
Enter no of blocks needed and no of files for the index 4 on the disk:
3
1 2 3
Allocated
File Indexed
4-------->1: 1
4-------->2: 1
4-------->3: 1
Do you want to enter more file (Yes - 1/No – 0)0
```

Experiment No:2(c) LINKED FILE ALLOCATION

AIM: Write a C Program to implement Linked File Allocation method.

Description:

Linked allocation:

It is easy to allocate the files, because allocation is on an individual block basis. Each block contains a pointer to the next free block in the chain. Here also the file allocation table consisting of a single entry for each file. Using this strategy any free block can be added to a chain very easily. There is a link between one block to another block, that's why it is said to be linked allocation. We can avoid the external fragmentation.

Algorithm for Linked File Allocation:

Step 1: Create a queue to hold all pages in memory
Step 2: When the page is required replace the page at the head of the queue
Step 3: Now the new page is inserted at the tail of the queue
Step 4: Create a stack
Step 5: When the page fault occurs replace page present at the bottom of the stack
Step 6: Stop the allocation.

/* Program to simulate linked file allocation strategy */

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void main()
{
int f[50], p,i, st, len, j, c, k, a;
clrscr();
for(i=0;i<50;i++)
f[i]=0;
printf("Enter how many blocks already allocated: ");
scanf("%d",&p);
printf("Enter blocks already allocated: ");
for(i=0;i<p;i++)
{
scanf("%d",&a);
f[a]=1;
}
x: printf("Enter index starting block and length: ");
scanf("%d%d", &st,&len);
k=len;
if(f[st]==0)
{
for(j=st;j<(st+k);j++)
{
if(f[j]==0)
{
f[j]=1;
printf("%d-------->%d\n",j,f[j]);
}
else
{
printf("%d Block is already allocated \n",j);
k++;
}
}
}
else
```

```
printf("Do you want to enter more file(Yes - 1/No - 0)");
scanf("%d", &c);
if(c==1)
goto x;
else
exit(0);
getch();
}
```

OUTPUT 1:

Enter how many blocks already allocated: 3
Enter blocks already allocated: 1 3 5
Enter index starting block and length: 2 2
2-------->1
3 Block is already allocated
4-------->1
Do you want to enter more file(Yes - 1/No - 0)0

OUTPUT 2:

Enter blocks already allocated: 2 4 6 8 10 12
Enter index starting block and length: 3 10
3-------->1
4 Block is already allocated
5-------->1
6 Block is already allocated
7-------->1
8 Block is already allocated
9-------->1
10 Block is already allocated
11-------->1
12 Block is already allocated
13-------->1
14-------->1
15-------->1
16-------->1
17-------->1
Do you want to enter more file (Yes - 1/No - 0)1
Enter index starting block and length: 5 1
5 starting blocks is already allocated
Do you want to enter more file (Yes - 1/No - 0)1
Enter index starting block and length: 18 2
18-------->1
19-------->1
Do you want to enter more file (Yes - 1/No - 0)0

Experiment No: 3(a) MULTIPROGRAM VARIABLE TASK

AIM: Write a program to implement Dynamic allocation of memories in MVT.

Description:

MVT:

MVT stands for multiprogramming with variable number of tasks. Multiprogramming is a technique to execute number of programs simultaneously by a single processor. This is one of the memory management techniques. To eliminate the same of the problems with fixed partitions, an approach known as dynamic partitioning developed. In this technique, partitions are created dynamically, so that each process is loaded into partition of exactly the same size at that process. This scheme is suffering from external fragmentation.

Step1: start the process.
Step2: Declare variables.
Step3: Enter total memory size.
Step4: Allocate memory for os.
Step5: allocate total memory to the pages.
Step6: Display the wastage of memory.
Step7: Stop the process.

/* Program to simulate the MVT */

```
#include<stdio.h>
#include<conio.h>
void main()
{
int m,i,p[15];
char ch;
clrscr();
printf("Enter memory to be allocated: ");
scanf("%d",&m);
printf("Enter process size : ");
scanf("%d", &p[0]);
i=0;
do
{
m=m-p[i];
printf("\nRemaining memory is %d\n",m);
abc:printf("Do you want to continue: ");
fflush(stdin);
scanf("%c",&ch);
i++;
if(ch=='y')
{
printf("Enter the process size: ");
scanf("%d",&p[i]);
}
Else{ printf("\nExternal fragmentation is %d",m);
break;
}
if(m<p[i])
{
printf("\nRequired memory is not available\n");
goto abc;
}
}while((ch=='y')&&(m>=p[i]));
getch();
}
```

OUTPUT 1:

```
Enter memory to be allocated: 1000
Enter process size: 500
Remaining memory is 500
Do you want to continue: y
Enter the process size: 300
Remaining memory is 200
Do you want to continue: y
Enter the process size: 100
Remaining memory is 100
Do you want to continue: n
```

OUTPUT 2:
Enter memory to be allocated: 800
Enter process size: 300
Remaining memory is 500
Do you want to continue: y?
Enter the process size: 200
Remaining memory is 300
Do you want to continue: y
Enter the process size: 200
Remaining memory is 100
Do you want to continue: y
Enter the process size: 100
Remaining memory is 0
Do you want to continue: n
External fragmentation is 0

Experiment No: 3(b) MULTIPROGRAM FIXED TASK
AIM: Write a program to implement Dynamic allocation of memories in MVT.

Description:
MFT:
MFT stands for multiprogramming with fixed no of tasks. MFT is the one of the memory management technique. In this technique, main memory is divided into no of static partitions at the system generated time. A process may be loaded into a partition of equal or greater size. The partition sizes are depending on o.s. in this memory management scheme the o.s occupies the low memory, and the rest of the main memory is available for user space. This scheme suffers from internal as well as external fragmentation.

Algorithm for Multiprogram Fixed Task:
Step1: start the process.
Step2: Declare variables.
Step3: Enter total memory size.
Step4: Allocate memory for operating system.
Step5: allocate total memory to the pages.
Step6: Display the wastage of memory.
Step7: Stop the process.

/* Program to simulate MFT */
```
#include < stdio.h>
#include <conio.h>
void main()
{
int i, p,a[10],c[15],temp=0,total=0;
float b,t;
clrscr();
printf("Enter the total memory ");
scanf("%f",&t);
printf("Enter the processes :");
scanf("%d", &p);
b=t/p;
for(i=0;i<p;i++)
a[i]=b;
for(i=0;i<p;i++)
{
lable: printf("Enter memory for %d process: ",i);
scanf("%d",&temp);
```

```
{
c[i]=a[i]-temp;
printf("Internal fragmentation for this block");
printf("%d",c[i]);
}
Else{ printf("Required memory is not available");
goto lable;
}
total=total+c[i];
}
printf("Total internal fragmentation %d\n",total);
getch();
}
```

OUTPUT 1:

Enter the total memory: 800
Enter the processes: 4
Enter memory for 0 processes: 150
Internal fragmentation for this block: 50
Enter memory for 1 process: 100
Internal fragmentation for this block: 100
Enter memory for 2 processes: 200
Internal fragmentation for this block: 0
Enter memory for 3 processes: 200
Internal fragmentation for this block: 0
Total internal fragmentation 150

OUTPUT 2:

Enter the total memory: 1200
Enter the processes: 4
Enter memory for 0 processes: 300
Internal fragmentation for this block: 0
Enter memory for 1 process: 300
Internal fragmentation for this block: 0
Enter memory for 2 processes: 300
Internal fragmentation for this block: 0
Enter memory for 3 processes: 300
Internal fragmentation for this block: 0
Total internal fragmentation 0

Experiment No:4(a) SINGLE LEVEL DIRECTORY

AIM: Write a program to simulate Single Level Directory.

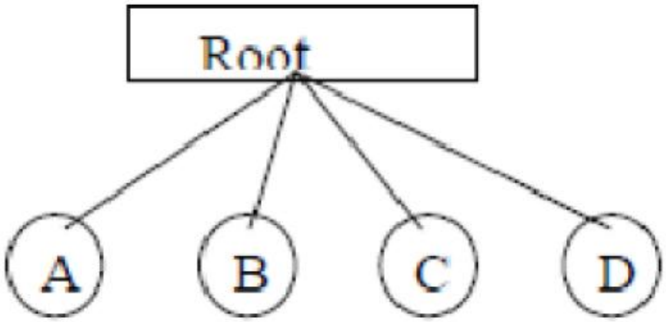
Description:

File Organization Techniques:

a)Single Level Directory
b)Two Level
c)Hierarchical
d)General Graph Directory

The directory contains information about the files, including attributes, location and ownership. Sometimes the directories consisting of subdirectories also. The directory is itself a file, owned by the operating system and accessible by various file management routines.

Single Level Directories: It is the simplest of all directory structures, in this the directory system having only one directory, it consisting of the all files. Sometimes it is said to be root directory. The following dig. Shows single level directory that contains four files (A, B, C, D).



It has the simplicity and ability to locate files quickly. It is not used in the multi-user system, it is used on small embedded system.

Algorithm for Single Level Directory Structure:

Step 1: Start
Step 2: Initialize values gd=DETECT,gm,count,i,j,mid,cir_x;
Initialize character array fname[10][20];
Step 3: Initialize graph function as
Initgraph(& gd, &gm," c:/tc/bgi");
Clear device();
Step 4:set back ground color with setbkcolor();
Step 5:read number of files in variable count.
Step 6:if check i<count
Step 7: for i=0 & i<count
i increment;
Cleardevice();
setbkcolor(GREEN);
read file name;

setfillstyle(1,MAGENTA);
Step 8: mid=640/count;
cir_x=mid/3;
bar3d(270,100,370,150,0,0);
settextstyle(2,0,4);
settextstyle(1,1);
outtextxy(320,125,"rootdirectory");
setcolor(BLUE);
i++;
Step 9:for j=0&&j<=i&&cir_x+=mid
j increment;
line(320,150,cir_x,250);
fillellipse(cir_x,250,30,30);
outtextxy(cir_x,250,fname[i]);
Step 10: End

/* Program to simulate single level directory */

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
int nf=0,i=0,j=0,ch;
char mdname[10],fname[10][10],name[10];
clrscr();
printf("Enter the directory name:");
scanf("%s",mdname);
```

```
scanf("%d",&nf);
do
{
printf("Enter file name to be created:");
scanf("%s",name);
for(i=0;i<nf;i++)
{
if(!strcmp(name,fname[i]))
break;
}
if(i==nf)
{
strcpy(fname[j++],name);
nf++;
}
else
printf("There is already %s\n",name);
printf("Do you want to enter another file(yes - 1 or no - 0):");
scanf("%d",&ch);
}while(ch==1);
printf("Directory name is:%s\n",mdname);
printf("Files names are:");
for(i=0;i<j;i++)
printf("\n%s",fname[i]);
getch();
}
```

OUTPUT 1:

Enter the directory name: sss
Enter the number of files: 3
Enter file name to be created: aaa
Do you want to enter another file (yes - 1 or no - 0):1
Enter file name to be created: bbb
Do you want to enter another file (yes - 1 or no - 0):1
Enter file name to be created: ccc
Do you want to enter another file (yes - 1 or no - 0):0
Directory name is: sss
Files names are:
aaa
bbb
ccc

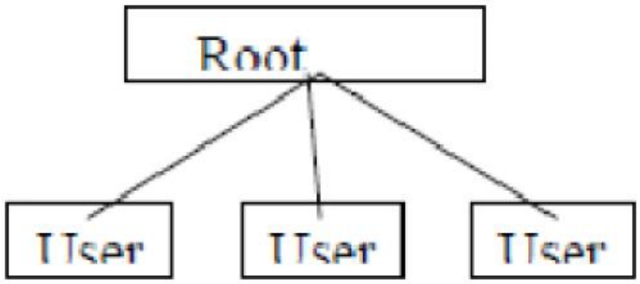
OUTPUT 2:

Enter the directory name: abc
Enter the number of files: 3
Enter file name to be created: xyz
Do you want to enter another file (yes - 1 or no - 0):1
Enter file name to be created: klm
Do you want to enter another file (yes - 1 or no - 0):0
Directory name is: abc
Files name are;
xyz
klm

Experiment No:4(b) TWO LEVEL DIRECTORY

AIM: Write a program to simulate Two Level Directory.

Two Level Directory: The problem in single level directory is different users may be accidentally using the same names for their files. To avoid this problem, each user need a private directory. In this way names chosen by one user don't interface with names chosen by a different user. The following dig 2-level directory



Here root directory is the first level directory it consisting of entries of user directory. User1, User2, User3 are the user levels of directories. A, B, C are the files.

Algorithm for Two Level Directory Structure:

Step 1: Start
Step 2: Initialize structure elements
struct tree_ element char name[20];
Initialize integer variables x, y, ftype, lx, rx, nc, level; struct tree_element *link[5];}typedef structure tree_element node;
Step 3: start main function
Step 4: Step variables gd=DETECT,gm;
node *root;
root=NULL;
Step 5: create structure using
create(&root,0,"null",0,639,320);
Step 6: initgraph(&gd, &gm,"c:\tc\bgi");
display(root);
closegraph();
Step 7: end main function
Step 8: Initialize variables i,gap;
Step 9: if check *root==NULL
(*root)=(node*)malloc(sizeof(node));
enter name of ir file name in dname;
fflush(stdin);
gets((*root)->name);
Step 10: if check lev==0||lev==1
(*root)->ftype=1;
else(*root)->ftype=2;
(*root)->level=lev;
(*root)->y=50+lev*5;
(*root)->x=x;
(*root)->lx=lx;
(*root)->rx=rx;
Step 11:for i=0&&i<5 increment i
(*root)->link[i]=NULL;
if check (*root)->ftype==1
Step 12: if check (lev==0||lev==1)
if check(*root)->level==0
print "how many users"
else print"how many files"
print (*root)->name
read (*root)->nc
Step 13:Then (*root)->nc=0;

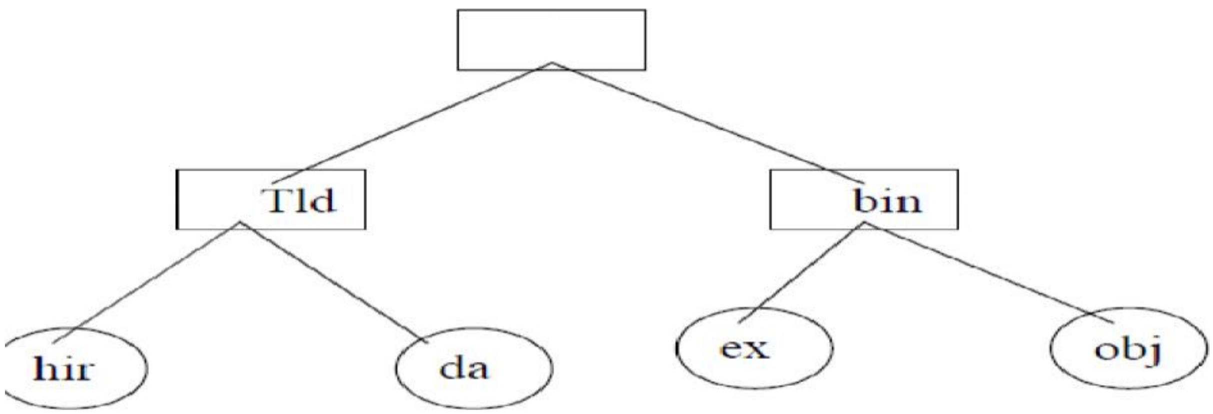
gap=rx-lx;
else gap=(rx-lx)/(*root)->nc;
Step 14:for i=0&&i<(*root)->nc
increment i;
create(&((*root)->link[i]),lev+1,(*root)->name, lx+gap*i,lx+gap*i+gap,lx+gap*i+gap/2);
then
(*root)->nc=0;
Step 15: Initialize e display function
Initialize i
set textstyle(2,0,4);
set textjustify(1,1);
set fillstyle(1,BLUE);
setcolor(14); step 13:if check root!=NULL
Step 16:for i=0&&i<root->nc
increment i
line(root->x,root->y,root->link[i]->x,root->link[i]->y);
Step 17: if check root->ftype==1
bar3d(root->x-20,root->y-10,root->x+20,root->y+10,0,0);
else fill ellipse(root->x,root->y,20,20);
out textxy(root->x,root->y,root->name);
Step 18:for i=0&&i<root->nc
increment i
display(root->link[i]);
Step 19:End

/* Program to simulate two level directory */

```
#include<stdio.h>
#include<graphics.h>
struct tree_element
{
char name[20];
int x,y,ftype,lx,rx,nc,level;
struct tree_element *link[5];
};
typedef struct tree_element node;
void main()
{
int gd=DETECT,gm;
node *root;
root=NULL;
clrscr();
create(&root,0,"null",0,639,320);
clrscr();
initgraph(&gd,&gm,"c:\\tc\\bgi");
display(root);
getch();
closegraph();
}
create(node **root,int lev,char *dname,int lx,int rx,int x)
{
int i,gap;
if(*root==NULL)
{
(*root)=(node*)malloc(sizeof(node));
printf("enter name of dir/file(under %s):",dname);
fflush(stdin);
gets((*root)->name);
```

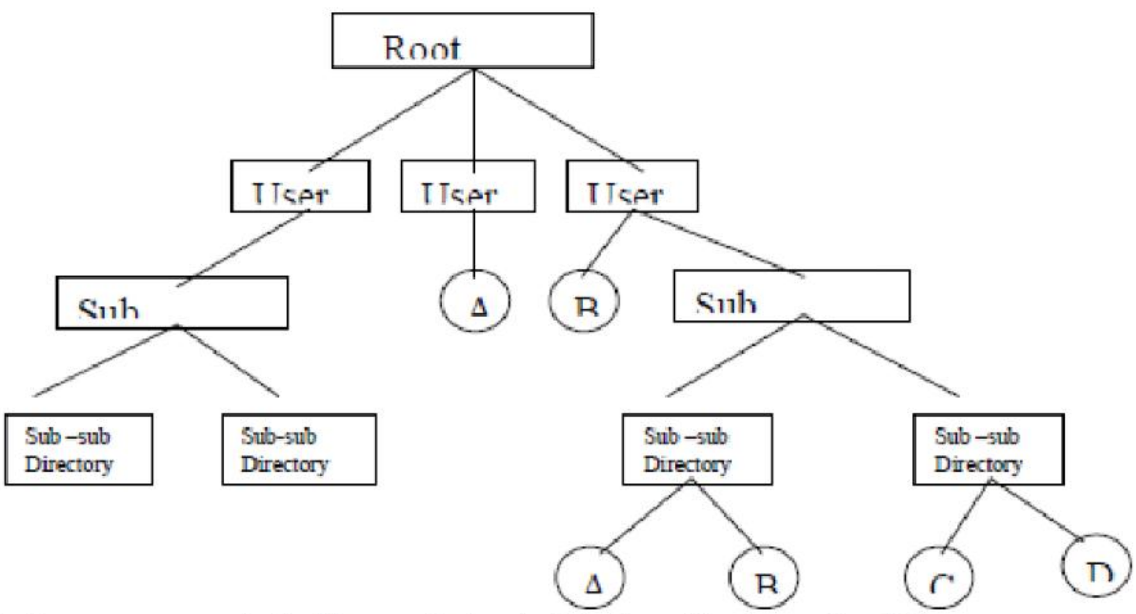
```
(*root)->ftype=1;
else
(*root)->ftype=2;
(*root)->level=lev;
(*root)->y=50+lev*50;
(*root)->x=x;
(*root)->lx=lx;
(*root)->rx=rx;
for(i=0;i<5;i++)
(*root)->link[i]=NULL;
if((*root)->ftype==1)
{
if(lev==0||lev==1)
{
if((*root)->level==0)
printf("How many users");
else
printf("hoe many files");
printf("(for%s):",(*root)->name);
scanf("%d",&(*root)->nc);
}
Else{ (*root)->nc=0;
if((*root)->nc==0)
gap=rx-lx;
else
gap=(rx-lx)/(*root)->nc;
for(i=0;i<(*root)->nc;i++)
create(&((*root)->link[i]),lev+1,(*root)-
>name,lx+gap*i,lx+gap*i+gap,lx+gap*i+gap/2);
}
else
(*root)->nc=0;
}
}
display(node *root)
{
int i;
settextstyle(2,0,4);
settextjustify(1,1);
setfillstyle(1,BLUE);
setcolor(14);
if(root!=NULL)
{
for(i=0;i<root->nc;i++)
{
line(root->x,root->y,root->link[i]->x,root->link[i]->y);
}
if(root->ftype==1)
bar3d(root->x-20,root->y-10,root->x+20,root->y+10,0,0);
else
fillellipse(root->x,root->y,20,20);
outtextxy(root->x,root->y,root->name);
for(i=0;i<root->nc;i++)
{
display(root->link[i]);
}
}
```

OUTPUT:
Enter name of dir/file(under null):sld
How many users (forsld):2
Enter name of dir/file(under sld):tld
How many files (fortld):2
Enter name of dir/file(under tld):hir
enter name of dir/file(under tld):dag
Enter name of dir/file(under sld):bin
How many files (forbin):2
Enter name of dir/file(under bin):exe
Enter name of dir/file(under bin):obj



Experiment No:4(c) HIERARCHICAL DIRECTORY
AIM: Write a program to simulate Hierarchical Directory.

Description:
Hierarchical Directory: The two level directories eliminate name conflicts among users but it is not satisfactory for users but it is not satisfactory for users with a large no of files. To avoid this, create the subdirectory and load the same type of the files into the subdirectory. So, in this method each can have as many directories are needed.



This directory structure looks like tree, that's why it is also said to be tree-level directory structure.

Algorithm for Hierarchical Directory Structure:
Step 1:Start
Step 2: define structure and declare structure variables

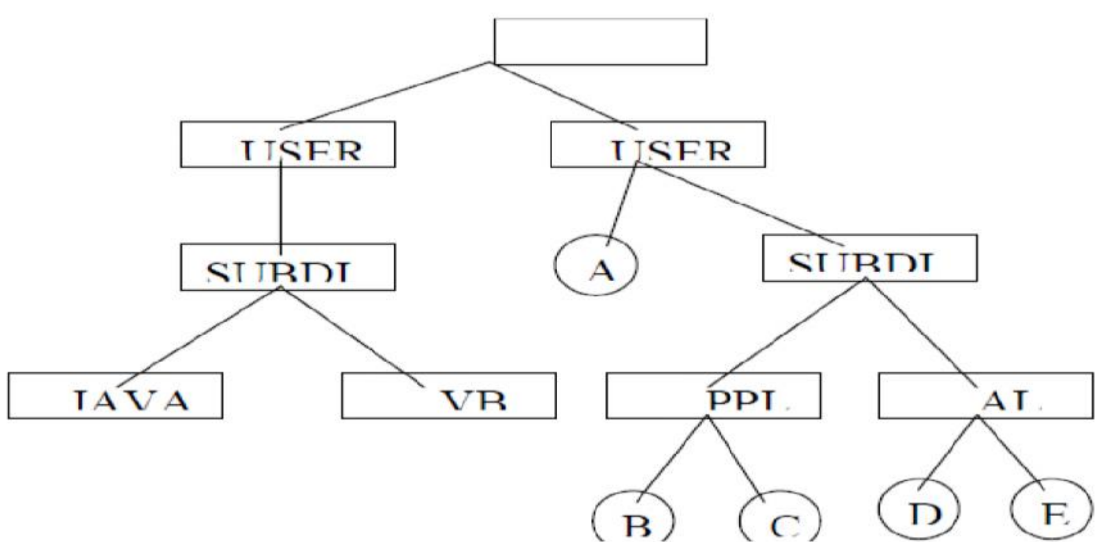
Node *root
Root = NULL
Step 4: create root null
Initgraph &gd,&gm
Display root
Step 5:create a directory tree structure
If check *root==NULL
Display dir/file mane
Step 6: gets *root->name
*root-> level=lev
*root->y=50+lev*50
*root->x=x
*root->lx=lx
*root->rx = rx
Step7: for i=0 to i<5
Root->link[i]=NULL
Display sub dir/ files
Step8: if check *root->nc==0
Gap=rx-lx
Then
Gap =rx-lx/*root->nc
Step9: for i=0 to i<*root->nc
Then
*rot->nc=0
Step10: display the directory tree in graphical mood
Display nood *root
If check rooy !=NULL
Step 11: foe i=0 to i<root->nc
Line of root->x, root->y, root->link[i]->x, root->link[i]-y
Step12: if check root->ftype==1
Bar3d of root->x-20, root->y-10,root->x+20,root->y+10,0
Then
Display root->link[i]
Step 13: Stop.

/* Program to simulate hierarchical directory */

```
#include<stdio.h>
#include<graphics.h>
struct tree_element
{
char name[20];
int x,y,ftype,lx,rx,nc,level;
struct tree_element *link[5];
};
typedef struct tree_element node;
void main()
{
int gd=DETECT,gm;
node *root;
root=NULL;
clrscr();
create(&root,0,"root",0,639,320);
clrscr();
initgraph(&gd,&gm,"c:\\tc\\BGI");
display(root);
getch();
closegraph();
```

```
create(node **root,int lev,char *dname,int lx,int rx,int x)
{
int i,gap;
if(*root==NULL)
{
(*root)=(node *)malloc(sizeof(node));
printf("Enter name of dir/file(under %s) : ",dname);
fflush(stdin);
gets((*root)->name);
printf("enter 1 for Dir/2 for file :");
scanf("%d",&(*root)->ftype);
(*root)->level=lev;
(*root)->y=50+lev*50;
(*root)->x=x;
(*root)->lx=lx;
(*root)->rx=rx;
for(i=0;i<5;i++)
(*root)->link[i]=NULL;
if((*root)->ftype==1)
{
printf("No of sub directories/files(for %s):",(*root)->name);
scanf("%d",&(*root)->nc);
if((*root)->nc==0)
gap=rx-lx;
else
gap=(rx-lx)/(*root)->nc;
for(i=0;i<(*root)->nc;i++)
create(&((*root)->link[i]),lev+1,(*root)-
>name,lx+gap*i,lx+gap*i+gap,lx+gap*i+gap/2);
}
else
(*root)->nc=0;
}
}
display(node *root)
{
int i;
settextstyle(2,0,4);
settextjustify(1,1);
setfillstyle(1,BLUE);
setcolor(14);
if(root !=NULL)
{
for(i=0;i<root->nc;i++)
{
line(root->x,root->y,root->link[i]->x,root->link[i]->y);
}
if(root->ftype==1)
bar3d(root->x-20,root->y-10,root->x+20,root->y+10,0,0);
else
fillellipse(root->x,root->y,20,20);
outtextxy(root->x,root->y,root->name);
for(i=0;i<root->nc;i++)
{
display(root->link[i]);
}
}
```

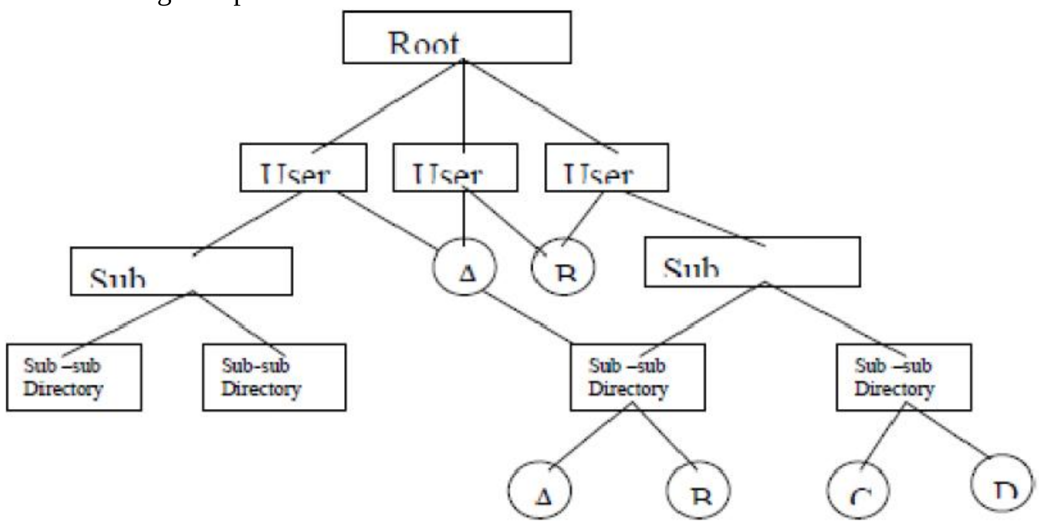
OUTPUT:
Enter Name of dir/file (under root): ROOT
Enter 1 for Dir / 2 For File : 1
No of subdirectories / files (for ROOT) :2
Enter Name of dir/file (under ROOT): USER 1
Enter 1 for Dir /2 for file:1
No of subdirectories /files (for USER 1): 1
Enter Name of dir/file (under USER 1):SUBDIR
Enter 1 for Dir /2 for file:1
No of subdirectories /files (for SUBDIR): 2
Enter Name of dir/file (under USER 1):JAVA
Enter 1 for Dir /2 for file:1
No of subdirectories /files (for JAVA): 0
Enter Name of dir/file (under SUBDIR):VB
Enter 1 for Dir /2 for file:1
No of subdirectories /files (for VB): 0
Enter Name of dir/file (under ROOT):USER2
Enter 1 for Dir /2 for file:1
No of subdirectories /files (for USER2): 2
Enter Name of dir/file (under ROOT):A
Enter 1 for Dir /2 for file:2
Enter Name of dir/file (under USER2):SUBDIR 2
Enter 1 for Dir /2 for file:1
No of subdirectories /files (for SUBDIR 2): 2
Enter Name of dir/file (under SUBDIR2):PPL
Enter 1 for Dir /2 for file:1
No of subdirectories /files (for PPL): 2
Enter Name of dir/file (under PPL):B
Enter 1 for Dir /2 for file:2
Enter Name of dir/file (under PPL):C
Enter 1 for Dir /2 for file:2
Enter Name of dir/file (under SUBDIR):AI
Enter 1 for Dir /2 for file:1
No of subdirectories /files (for AI): 2
Enter Name of dir/file (under AI):D
Enter 1 for Dir /2 for file:2
Enter Name of dir/file (under AI):E
Enter 1 for Dir /2 for file:2



AIM: Write a program to simulate General Graph Directory.

Description:

General graph Directory: When we add links to an existing tree structured directory, the tree structure is destroyed, resulting in a simple graph structure. This structure is used to traversing is easy and file sharing also possible.




```
/*Program to General Graph Directory */
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<string.h>
struct tree_element
{
char name[20];
int x,y,ftype,lx,rx,nc,level;
struct tree_element *link[5];
};
typedef struct tree_element node;
typedef struct
{
char from[20];
char to[20];
}link;
link L[10];
int nofl;
node * root;
void main()
{
int gd=DETECT,gm;
root=NULL;
clrscr();
create(&root,0,"root",0,639,320);
read_links();
clrscr();
initgraph(&gd,&gm,"c:\\tc\\BGI");
draw_link_lines();
display(root);
getch();
closegraph();
}
read_links()
```

```
int i;
printf("how many links");
scanf("%d",&nofl);
for(i=0;i<nofl;i++)
{
printf("File/dir:");
fflush(stdin);
gets(L[i].from);
printf("user name:");
fflush(stdin);
gets(L[i].to);
}
}
draw_link_lines()
{
int i,x1,y1,x2,y2;
for(i=0;i<nofl;i++)
{
search(root,L[i].from,&x1,&y1);
search(root,L[i].to,&x2,&y2);
setcolor(LIGHTGREEN);
setlinestyle(3,0,1);
line(x1,y1,x2,y2);
setcolor(YELLOW);
setlinestyle(0,0,1);
}
}
search(node *root,char *s,int *x,int *y)
{
int i;
if(root!=NULL)
{
if(strcmpi(root->name,s)==0)
{
*x=root->x;
*y=root->y;
return;
}
else
{
for(i=0;i<root->nc;i++)
search(root->link[i],s,x,y);
}
}
}
create(node **root,int lev,char *dname,int lx,int rx,int x)
{
int i,gap;
if(*root==NULL)
{
(*root)=(node *)malloc(sizeof(node));
printf("enter name of dir/file(under %s):",dname);
fflush(stdin);
gets((*root)->name);
printf("enter 1 for dir/ 2 for file:");
scanf("%d",&(*root)->ftype);
(*root)->level=lev;
```

```
(*root)->x=x;
(*root)->lx=lx;
(*root)->rx=rx;
for(i=0;i<5;i++)
(*root)->link[i]=NULL;
if((*root)->ftype==1)
{
printf("no of sub directories /files (for %s):",(*root)->name);
scanf("%d",&(*root)->nc);
if((*root)->nc==0)
gap=rx-lx;
else
gap=(rx-lx)/(*root)->nc;
for(i=0;i<(*root)->nc;i++)
create( & ( (*root)->link[i] ) , lev+1 , (*root)-
>name,lx+gap*i,lx+gap*i+gap,lx+gap*i+gap/2);
}
else
(*root)->nc=0;
}
}
/* displays the constructed tree in graphics mode */
display(node *root)
{
int i;
settextstyle(2,0,4);
settextjustify(1,1);
setfillstyle(1,BLUE);
setcolor(14);
if(root !=NULL)
{
for(i=0;i<root->nc;i++)
{
line(root->x,root->y,root->link[i]->x,root->link[i]->y);
}
if(root->ftype==1)
bar3d(root->x-20,root->y-10,root->x+20,root->y+10,0,0);
else
fillellipse(root->x,root->y,20,20);
outtextxy(root->x,root->y,root->name);
for(i=0;i<root->nc;i++)
{
display(root->link[i]);
}}}
```

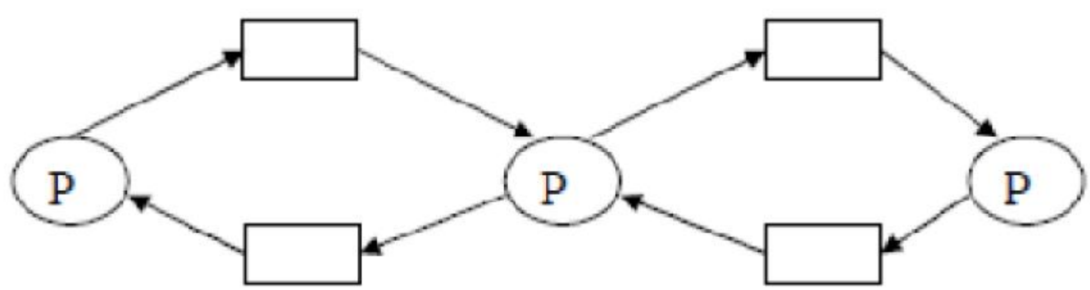
OUTPUT:

Enter Name of dir/file (under root): ROOT
Enter 1 for Dir / 2 For File : 1
No of subdirectories / files (for ROOT) :2
Enter Name of dir/file (under ROOT): USER 1
Enter 1 for Dir /2 for file:1
No of subdirectories /files (for USER 1): 2
Enter Name of dir/file (under USER1): VB
Enter 1 for Dir /2 for file:1
No of subdirectories /files (for VB): 2
Enter Name of dir/file (under VB): A
Enter 1 for Dir /2 for file:2

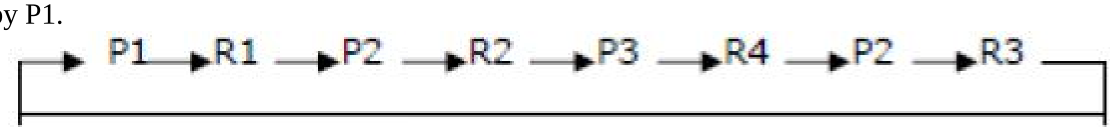
Enter 1 for Dir /2 for file:2
Enter Name of dir/file (under USER1): C
Enter 1 for Dir /2 for file:2
Enter Name of dir/file (under ROOT): USER2
Enter 1 for Dir /2 for file:1
No of subdirectories /files (for USER2): 1
Enter Name of dir/file (under USER2):JAVA
Enter 1 for Dir /2 for file:1
No of subdirectories /files (for JAVA):2
Enter Name of dir/file (under JAVA):D
Enter 1 for Dir /2 for file:2
Enter Name of dir/file (under JAVA):HTML
Enter 1 for Dir /2 for file:1
No of subdirectories /files (for HTML):0
How many links:2
File/Dir: B
User Name: USER 2
File/Dir: HTML
User Name: USER 1

Experiment No: 5 BANKER'S DEADLOCK AVOIDANCE
Aim: To Simulate Bankers Algorithm for Deadlock Avoidance.
Description:

Deadlock: A process request the resources, the resources are not available at that time, so the process enter into the waiting state. The requesting resources are held by another waiting process, both are in waiting state, this situation is said to be Deadlock. A deadlocked system must satisfied the following 4 conditions. These are:
(i) Mutual Exclusion: Mutual Exclusion means resources are in non-sharable mode only, it means only one process at a time can use a process.
(ii) Hold and Wait: Each and every process is the deadlock state, must holding at least one resource and is waiting for additional resources, that are currently being held by another process.



(iii) No Preemption: No Preemption means resources are not released in the middle of the work, they released only after the process has completed its task.
(iv) Circular Wait: If process P1 is waiting for a resource R1, it is held by P2, process P2 is waiting for R2, R2 held by P3, P3 is waiting for R4, R4 is held by P2, P2 waiting for resource R3, it is held by P1.



Deadlock Avoidance: It is one of the method of dynamically escaping from the deadlocks. In this scheme, if a process request for resources, the avoidance algorithm checks before the allocation of resources about the state of system. If the state is safe, the system allocate the resources to the requesting process otherwise (unsafe) do not allocate the resources. So taking care before the allocation said to be deadlock avoidance.
Banker's Algorithm: It is the deadlock avoidance algorithm, the name was chosen because the

Available: A vector of length 'm' indicates the number of available resources of each type. If available[j]=k, there are 'k' instances of resource types Rj available.
Allocation: An nxm matrix defines the number of resources of each type currently allocated to each process. If allocation [i,j]=k, then process Pi is currently allocated 'k' instances of resources type Rj.
Max: An nxm matrix defines the maximum demand of each process. If max [i,j]=k, then Pi may request at most 'k' instances of resource type Rj.
Need: An nxm matrix indicates the remaining resources need of each process. If need [I,j]=k, then Pi may need 'k' more instances of resource type Rj to complete this task. There fore, Need[i,j]=Max[i,j]-Allocation[I,j]

Safety Algorithm:
1. Work and Finish be the vector of length m and n respectively, Work=Available and Finish[i] =False.
2. Find an i such that both
   Finish[i] =False
   Need<=Work

   If no such I exist go to step 4.
3. Work=work Allocation, Finish[i] =True;
4. if Finish[1]=True for all I, then the system is in safe state.

Resource request algorithm
Let Request i be request vector for the process Pi, If request i=[j]=k, then process Pi wants k instances of resource type Rj.
1. if Request<=Need I go to step 2. Otherwise raise an error condition.
2. if Request<=Available go to step 3. Otherwise Pi must since the resources are available.
3. Have the system pretend to have allocated the requested resources to process Pi by modifying the state as follows;

Available=Available-Request I;
Allocation I =Allocation +Request I;
Need i=Need i-Request I;
If the resulting resource allocation state is safe, the transaction is completed and process Pi is allocated its resources. However, if the state is unsafe, the Pi must wait for Request i and the old resource-allocation state is restored.

Algorithm for Banker's Deadlock Avoidance:
1. Start the program.
2. Get the values of resources and processes.
3. Get the avail value.
4. After allocation find the need value.
5. Check whether its possible to allocate.
6. If it is possible then the system is in safe state.
7. Else system is not in safety state.
8. If the new request comes then check that the system is in safety.
9. or not if we allow the request.
10. stop the program.

/* Program to Simulate Bankers Algorithm for Dead Lock Avoidance */
```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
int alloc[10][10],max[10][10];
int avail[10],work[10],total[10];
int i,j,k,n,need[10][10];
int m;
```

```
char finish[10];
clrscr();
printf("Enter the no. of processes and resources:");
scanf("%d%d",&n,&m);
for(i=0;i<=n;i++)
finish[i]='n';
printf("Enter the claim matrix:\n");
for(i=0;i<n;i++)
for(j=0;j<m;j++)
scanf("%d",&max[i][j]);
printf("Enter the allocation matrix:\n");
for(i=0;i<n;i++)
for(j=0;j<m;j++)
scanf("%d",&alloc[i][j]);
printf("Resource vector:");
for(i=0;i<m;i++)
scanf("%d",&total[i]);
for(i=0;i<m;i++)
avail[i]=0;
for(i=0;i<n;i++)
for(j=0;j<m;j++)
avail[j]+=alloc[i][j];
for(i=0;i<m;i++)
work[i]=avail[i];
for(j=0;j<m;j++)
work[j]=total[j]-work[j];
for(i=0;i<n;i++)
for(j=0;j<m;j++)
need[i][j]=max[i][j]-alloc[i][j];
A:for(i=0;i<n;i++)
{
c=0;
for(j=0;j<m;j++)
if((need[i][j]<=work[j])&&(finish[i]=='n'))
c++;
if(c==m)
{
printf("All the resources can be allocated to Process %d",
i+1);
printf("\n\nAvailable resources are:");
for(k=0;k<m;k++)
{
work[k]+=alloc[i][k];
printf("%4d",work[k]);
}
printf("\n");
finish[i]='y';
printf("\nProcess %d executed?:%c \n",i+1,finish[i]);
count++;
}
}
if(count!=n)
goto A;
else
printf("\n System is in safe mode");
printf("\n The given state is safe state");
getch();
```

O
U
T
P

T 1:

Enter the no. of processes and resources: 4 3
Enter the claim matrix:
3 2 2
6 1 3
3 1 4
4 2 2
Enter the allocation matrix:
1 0 0
6 1 2
2 1 1
0 0 2
Resource vector:9 3 6
All the resources can be allocated to Process 2
Available resources are: 6 2 3
Process 2 executed?:y
All the resources can be allocated to Process 3
Avaiable resources are: 8 3 4
Prcess 3 executed?:y
All the resources can be allocated to Process 4
Avilable resources are: 8 3 6
Prcess 4 executed?:y
All the resources can be allocated to Process 1
Aailable resources are: 9 3 6
Prcess 1 executed?:y
System is in safe mode
The given state is safe state

Experiment No:6 BANKER'S DEADLOCK PREVENTION

Aim: To Simulate Bankers Algorithm for Deadlock Prevention.

/* Program to Simulate Bankers Algorithm for Dead Lock Prevention */

```
#include<stdio.h>
#include<conio.h>
void main()
{
char job[10][10];
int time[10],avail,tem[10],temp[10];
int safe[10];
int ind=1,i,j,q,n,t;
clrscr();
printf("Enter no of jobs: ");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter name and time: ");
scanf("%s%d",&job[i],&time[i]);
}
printf("Enter the available resources:");
scanf("%d",&avail);
for(i=0;i<n;i++)
{
temp[i]=time[i];
tem[i]=i;
}
for(i=0;i<n;i++)
for(j=i+1;j<n;j++)
```

```
if(temp[i]>temp[j])
{
t=temp[i];
temp[i]=temp[j];
temp[j]=t;
t=tem[i];
tem[i]=tem[j];
tem[j]=t;
}
}
for(i=0;i<n;i++)
{
q=tem[i];
if(time[q]<=avail)
{
safe[ind]=tem[i];
avail=avail-tem[q];
//printf("%s",job[safe[ind]]);
ind++;
}
else
{
printf("No safe sequence\n");
}
}
printf("Safe sequence is:");
for(i=1;i<ind; i++)
printf(" %s %d\n",job[safe[i]],time[safe[i]]);
getch();
}
```

OUTPUT 1:
Enter no of jobs:4
Enter name and time: A 1
Enter name and time: B 4
Enter name and time: C 2
Enter name and time: D 3
Enter the available resources: 20
Safe sequence is: A 1, C 2, D 3, B 4

Experiment No: 7(a) FIFO PAGE REPLACEMENT ALGORITHM
AIM: To implement FIFO (First In First Out) page replacement algorithm.

Description:
FIFO (First in First Out) algorithm: FIFO is the simplest page replacement algorithm, the idea behind this is, "Replace a page that page is oldest page of main memory" or "Replace the page that has been in memory longest". FIFO focuses on the length of time a page has been in the memory rather than how much the page is being used.

Algorithm for FIFO Page Replacement:
Step 1: Create a queue to hold all pages in memory
Step 2: When the page is required replace the page at the head of the queue
Step 3: Now the new page is inserted at the tail of the queue

/* Program to simulate FIFO page replacement */

```
#include<stdio.h>
```

```
int fr[3],m;
void display();
void main()
{
int i,j,page[20];
int flag1=0,flag2=0,pf=0;
int n, top=0;
float pr;
clrscr();
printf("Enter length of the reference string: ");
scanf("%d",&n);
printf("Enter the reference string: ");
for(i=0;i<n;i++)
scanf("%d",&page[i]);
printf("Enter no of frames: ");
scanf("%d",&m);
for(i=0;i<m;i++)
fr[i]=-1;
for(j=0;j<n;j++)
{
flag1=0;
flag2=0;
for(i=0;i<m;i++)
{
if(fr[i]==page[j])
{
flag1=1;
flag2=1;
break;
}
}
if(flag1==0)
{
for(i=0;i<m;i++)
{
if(fr[i]==-1)
{
fr[i]=page[i];
flag2=1;
break;
}
}
}
if(flag2==0)
{
fr[top]=page[j];
top++;
pf++;
if(top>=m)
top=0;
}
display();
}
pf+=m;
printf("Number of page faults : %d\n", pf);
pr=(float)pf/n*100;
printf("Page fault rate = %f \n", pr);
```

```
}
void display()
{
int i;
for(i=0;i<m;i++)
printf("%d\t", fr[i]);
printf("\n");
}
```

OUTPUT 1:

```
Enter length of the reference string: 12
Enter the reference string: 1 2 3 4 1 2 5 1 2 3 4 5
Enter no of frames: 3
1 -1 -1
1 2 -1
1 2 3
4 2 3
4 1 3
4 1 2
5 1 2
5 1 2
5 1 2
5 3 2
5 3 4
5 3 4
Number of page faults: 9
Page fault rate = 75.000000
```

OUTPUT 2:

```
Enter length of the reference string: 12
Enter the reference string: 1 2 3 4 1 2 5 1 2 3 4 5
Enter no of frames: 4
1 -1 -1 -1
1 2   -1-1
1 2   3 -1
1 2   3   4
1 2   3   4
1 2   3   4
5 2   3   4
5 1   3   4
5 1   2   4
5 1   2   3
4 1   2   3
4 5   2   3
Number of page faults: 10
Page fault rate = 83.333336
```

OUTPUT 3:

```
Enter length of the reference string: 20
Enter the reference string: 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
Enter no of frames: 3
7 -1 -1
7  0  -1
7  0   1
2  0   1
2  0   1
2  3   1
```

4 3 0
4 2 0
4 2 3
0 2 3
0 2 3
0 2 3
0 1 3
0 1 2
0 1 2
0 1 2
7 1 2
7 0 2
7 0 1
Number of page faults: 15
Page fault rate = 75.000000

OUTPUT 4:

Enter length of the reference string: 20
Enter the reference string: 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
Enter no of frames: 4
7 -1 -1 -1
7 0 -1 -1
7 0 1 -1
7 0 1 2
7 0 1 2
3 0 1 2
3 0 1 2
3 4 1 2
3 4 1 2
3 4 1 2
3 4 0 2
3 4 0 2
3 4 0 2
3 4 0 1
2 4 0 1
2 4 0 1
2 4 0 1
2 7 0 1
2 7 0 1
2 7 0 1

Number of page faults: 10
Page fault rate = 50.000000

Experiment No:7(b) LRU PAGE REPLACEMENT ALGORITHM

AIM: To implement page replacement algorithm LRU (Least Recently Used)

Description:

LRU (Least Recently Used): the criteria of this algorithm is "Replace a page that has been used for the longest period of time". This strategy is the page replacement algorithm looking backward in time, rather than forward.

Algorithm for LRU Page Replacement:

Step 1: Create a queue to hold all pages in memory
Step 2: When the page is required replace the page at the head of the queue
Step 3: Now the new page is inserted at the tail of the queue
Step 4: Create a stack
Step 5: When the page fault occurs replace page present at the bottom of the stack

/* Program to simulate LRU page replacement */

```
#include<stdio.h>
#include<conio.h>
void display();
void main()
{
int i,j,page[20],fs[10],n,m;
int index,k,l,flag1=0,flag2=0,pf=0;
int fr[10];
float pr;
clrscr();
printf("Enter length of the reference string: ");
scanf("%d",&n);
printf("Enter the reference string: ");
for(i=0;i<n;i++)
scanf("%d",&page[i]);
printf("Enter no of frames: ");
scanf("%d",&m);
for(i=0;i<m;i++)
fr[i]=-1;
for(j=0;j<n;j++)
{
flag1=0;
flag2=0;
for(i=0;i<m;i++)
{
if(fr[i]==page[j])
{
flag1=1;
flag2=1;
break;
}
}
if(flag1==0)
{
for(i=0;i<m;i++)
{
if(fr[i]==-1)
{
fr[i]=page[j];
flag2=1;
break;
}
}
}
if(flag2==0)
{
for(i=0;i<m;i++)
fs[i]=0;
for(k=j-1,l=1;l<=m;l++,k--)
{
for(i=0;i<m;i++)
{
if(fr[i]==page[k])
fs[i]=1;
}
```

```
for(i=0;i<m;i++)
{
if(fs[i]==0)
index=i;
}
fr[index]=page[j];
pf++;
}
for(i=0;i<m;i++)
printf("%d\t", fr[i]);
printf("\n");
}
pf+=m;
printf("Number of page faults : %d\n", pf);
pr=(float)pf/n*100;
printf("Page fault rate = %f \n", pr);
getch();
}
```

OUTPUT 1:

```
Enter length of the reference string: 20
Enter the reference string: 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
Enter no of frames: 4
7 -1 -1 -1
7 0 -1 -1
7 0 1 -1
7 0 1 2
7 0 1 2
3 0 1 2
3 0 1 2
3 0 4 2
3 0 4 2
3 0 4 2
3 0 4 2
3 0 4 2
3 0 4 2
3 0 1 2
3 0 1 2
3 0 1 2
3 0 1 2
7 0 1 2
7 0 1 2
7 0 1 2
Number of page faults: 8
Page fault rate = 40.000000
```

OUTPUT 2:

```
Enter length of the reference string: 20
Enter the reference string: 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
Enter no of frames: 3
7 -1 -1
7 0 -1
7 0 1
7 0 2
7 0 2
3 0 2
3 0 2
```

3 0 2
3 0 2
3 0 2
3 0 2
3 0 2
3 0 1
3 2 1
0 2 1
0 2 1
7 2 1
7 0 1
7 0 1
Number of page faults: 12
Page fault rate = 60.000000

OUTPUT 3:
Enter length of the reference string: 12
Enter the reference string: 1 2 3 4 1 2 5 1 2 3 4 5
Enter no of frames: 3
1 -1 -1
1 2 -1
1 2 3
4 2 3
1 2 3
1 2 3
1 2 5
1 2 5
1 2 5
1 2 3
1 2 4
5 2 4
Number of page faults: 9
Page fault rate = 75.000000

OUTPUT 4:
Enter length of the reference string: 12
Enter the reference string: 1 2 3 4 1 2 5 1 2 3 4 5
Enter no of frames: 4
1 -1 -1 -1
1 2 -1 -1
1 2 3 -1
1 2 3 4
1 2 3 4
1 2 3 4
1 2 3 4
1 2 3 4
1 2 3 4
1 2 3 4
1 2 3 4
1 2 3 4
Number of page faults: 6
Page fault rate = 50.000000

Experiment No: 7(c) OPTIMAL(LFU) PAGE REPLACEMENT ALGORITHM

AIM: To implement page replacement algorithms

Description:

LFU (Least Frequently Used): The least frequently used algorithm “select a page for replacement, if the page has not been used for the often in the past” or “Replace page that page has smallest count” for this algorithm each page maintains as counter which counter value shows the least count, replace that page. The frequency counter is reset each time is page is loaded.

Algorithm for Optimal(LFU) Page Replacement:

Here we select the page that will not be used for the longest period of time.

Step 1: Create an array.

Step 2: When the page fault occurs replace page that will not be used for the longest period of time.

```
/* Program to simulate optimal page replacement */
#include<stdio.h>
#include<conio.h>
int fr[3], n, m;
void display();
void main()
{
int i,j,page[20],fs[10];
int max,found=0,lg[3],index,k,l,flag1=0,flag2=0,pf=0;
float pr;
clrscr();
printf("Enter length of the reference string: ");
scanf("%d",&n);
printf("Enter the reference string: ");
for(i=0;i<n;i++)
scanf("%d",&page[i]);
printf("Enter no of frames: ");
scanf("%d",&m);
for(i=0;i<m;i++)
fr[i]=-1;
pf=m;
for(j=0;j<n;j++)
{
flag1=0;
flag2=0;
for(i=0;i<m;i++)
{
if(fr[i]==page[j])
{
flag1=1;
flag2=1;
break;
}
}
if(flag1==0)
{
for(i=0;i<m;i++)
{
if(fr[i]==-1)
{
fr[i]=page[j];
flag2=1;
break;
}
}
```

```
if(flag2==0)
{
for(i=0;i<m;i++)
lg[i]=0;
for(i=0;i<m;i++)
{
for(k=j+1;k<=n;k++)
{
if(fr[i]==page[k])
{
lg[i]=k-j;
break;
}
}
}
found=0;
for(i=0;i<m;i++)
{
if(lg[i]==0)
{
index=i;
found = 1;
break;
}
}
if(found==0)
{
max=lg[0];
index=0;
for(i=0;i<m;i++)
{
if(max<lg[i])
{
max=lg[i];
index=i;
}
}
}
fr[index]=page[j];
pf++;
}
display();
}
printf("Number of page faults : %d\n", pf);
pr=(float)pf/n*100;
printf("Page fault rate = %f \n", pr);
getch();
}
void display()
{
int i;
for(i=0;i<m;i++)
printf("%d\t",fr[i]);
printf("\n");
}
```

Enter length of the reference string: 12
Enter the reference string: 1 2 3 4 1 2 5 1 2 3 4 5
Enter no of frames: 3
1 -1 -1
1 2 -1
1 2 3
1 2 4
1 2 4
1 2 4
1 2 5
1 2 5
1 2 5
3 2 5
4 2 5
4 2 5
Number of page faults : 7
Page fault rate = 58.333332

OUTPUT 2:
Enter length of the reference string: 12
Enter the reference string: 1 2 3 4 1 2 5 1 2 3 4 5
Enter no of frames: 3
1 -1 -1
1 2 -1
1 2 3
1 2 4
1 2 4
1 2 5
1 2 5
1 2 5
3 2 5
4 2 5
4 2 5
Number of page faults : 7
Page fault rate = 58.333332

OUTPUT 3:
Enter length of the reference string: 20
Enter the reference string: 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
Enter no of frames: 3
7 -1 -1
7 0 -1
7 0 1
2 0 1
2 0 1
2 0 3
2 0 3
2 4 3
2 4 3
2 4 3
2 0 3
2 0 3
2 0 3
2 0 1
2 0 1
2 0 1

7 0 1
7 0 1
7 0 1
Number of page faults: 9
Page fault rate = 45.000000

OUTPUT 4:
Enter length of the reference string: 20
Enter the reference string: 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
Enter no of frames: 4
7 -1 -1 -1
7 0 -1 -1
7 0 1 -1
7 0 1 2
7 0 1 2
3 0 1 2
3 0 1 2
3 0 4 2
3 0 4 2
3 0 4 2
3 0 4 2
3 0 4 2
3 0 4 2
1 0 4 2
1 0 4 2
1 0 4 2
1 0 4 2
1 0 7 2
1 0 7 2
1 0 7 2
Number of page faults : 8
Page fault rate = 40.000000

Experiment No: 8PAGING TECHNIQUE OF MEMORY MANAGEMENT
AIM: To implement the Memory management policy- Paging.

Description: Paging is an efficient memory management scheme because it is noncontiguous memory allocation method. The basic idea of paging is the physical memory (main memory) is divided into fixed sized blocks called frames, the logical address space is divided into fixed sized blocks, called pages, but page size and frame size should be equal. The size of the frame or a page is depending on operating system. In this scheme the operating system maintains a data structure that is page table; it is used for mapping purpose. The page table specifies the some useful information; it tells which frames are there and so on. The page table consisting of two fields, one is the page number and other one is frame number. Every address generated by the CPU divided into two parts; one is page number and second is page offset or displacement. The pages are loaded into available free frames in the physical memory.
Algorithm for Paging Technique:
Step 1: Read all the necessary input from the keyboard.
Step 2: Pages - Logical memory is broken into fixed - sized blocks.
Step 3: Frames – Physical memory is broken into fixed – sized blocks.
Step 4: Calculate the physical address using the following
Physical address = (Frame number * Frame size) + offset
Step 5: Display the physical address.
Step 6: Stop the process.

```
#include<stdio.h>
void main()
{
int p, ps, i;
int *sa;
clrscr();
printf("Enter how many pages: ");
scanf("%d",&np);
printf("Enter page size: ");
scanf("%d",&ps);
for(i=0;i< np;i++)
{
sa[i]=(int)malloc(ps);
printf("Page %d address is %d\n", i, sa[i]);
}
getch();
}
```

OUTPUT 1:

Enter how many pages: 5
Enter page size: 4
Page 0 address is 3080
Page 1 address is 3088
Page 2 address is 3096
Page 3 address is 3104
Page 4 address is 3112

OUTPUT 2:

Enter how many pages: 7
Enter page size: 8
Page 0 address is 3080
Page 1 address is 3096
Page 2 address is 3112
Page 3 address is 3128
Page 4 address is 3144
Page 5 address is 3160
Page 6 address is 3176

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR
Course Description

Title of Course: Seminar **Course Code: CS681**
L-T –P Scheme: 0-0-3 **Course Credits: 2**

Course Description & Objectives:

1. Understand the diverse social and economic, racial and gender contexts within which Henrietta Lacks lived and died. Understand the themes of this seminar. Appreciate the legacy and implications of these medical, ethical and social understandings on today's society.

2. **Identify**, understand and discuss current, real-world issues.

3. **Distinguish** and **integrate** differing forms of knowledge and academic disciplinary approaches (e.g., humanities and sciences) with that of the student's own academic discipline (e.g., in agriculture, architecture, art, business, economics, education, engineering, natural resources, etc.). And apply a **multidisciplinary strategy** to address current, real-world **issues**.

4. Improve oral and written **communication** skills.

5. Explore an appreciation of the **self** in relation to its larger diverse social and academic contexts.

6. Apply principles of **ethics** and **respect** in interaction with others.

Course Outcomes:

After the completion of this course, the student should be able to:

1. **Learn and integrate.** ***Through independent learning and collaborative study, attain, use, and develop knowledge*** *in the arts, humanities, sciences, and social sciences, with disciplinary specialization and the ability to integrate information across disciplines.*

2. *Use multiple thinking strategies to* ***examine real-world issues****,* ***explore creative avenues of expression****,* ***solve problems, and make consequential decisions***

3. **Learn and integrate.** Communicate**.** *Acquire, articulate, create and convey intended meaning using* ***verbal*** *and* ***non-verbal method of communication*** *that demonstrates respect and understanding in a complex society.*

4. *Use multiple thinking strategies to examine* ***real-world issues****,* ***explore creative avenues of expression****,* ***solve problems, and make consequential decisions.***

5. **Clarify purpose and perspective.** *Explore* ***one's life purpose and meaning*** *through transformational experiences that foster an understanding of* ***self****,* ***relationships****, and* ***diverse global perspectives****.*

6. **ractice citizenship.** *Apply principles of **ethical** leadership, **collaborative engagement**, **socially responsible behavior**, **respect for diversity** in an interdependent world, and a service-oriented commitment to advance and sustain local and **global communities**.*