INSTITUTE VISION AND MISSION

Vision

To emerge as a destination for higher education by transforming learners into achievers by creating, encouraging and thus building a supportive academic environment.

Mission

To impart Quality Technical Education and to undertake Research and Development with a focus on application and innovation which offers an appropriate solution to the emerging societal needs by making the students globally competitive, morally valuable and socially responsible citizens.

DEPARTMENT VISION AND MISSION

Vision

To emerge as a center of excellence with global reputation with adaption of rapid advancements in the field of computer specialization.

Mission

1. To provide a strong theoretical and practical background in area of computer science with an emphasize on software development.

2. To inculcate Professional behavior, strong ethical values, leadership qualities, research capabilities and lifelong learning.

3. To educate students to become effective problem solvers, apply knowledge with social sensitivity for the betterment of the society and humanity as a whole.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

Programme educational objectives are broad statements that describe the career and professional accomplishments that the programme is preparing graduates to achieve within 3 to 5 years after graduation.

The **Programme Educational Objectives** of the B. Tech CSE programme are:

- **PEO1:** To apply the knowledge of mathematics, basic science and engineering solving the real world computing problems to succeed higher education and professional careers.
- **PEO2:** To develop the skills required to comprehend, analyze, design and create innovative computing products and solutions for real life problems.
- **PEO3:** To inculcate professional and ethical attitude, communication and teamwork skills, multi-disciplinary approach and an ability to relate computer engineering issues with social awareness.

LABORATORY OUTCOMES:

CO [1] To understand the basic concepts of database systems.
CO[2] Students will be able to construct an Entity-Relationship (E-R) model from given specifications.
CO[3] Students able to develop SQL queries
Using relational algebra
CO[4] To distinguish between normalization principles and data redundancy on database
CO[5] Students will be able to build and maintain a simple database using transaction management and recovery techniques .
CO[6] Students will be able to relate different data storage structures and access techniques

Do’s

1. Come with completed observation and record

2. Wear apron and ID card before entering into the lab.

3. Know the location of the fire extinguisher and the first aid box and how to use them in case of an emergency.

4. Read and understand how to carry out an activity thoroughly before coming to the laboratory.

5. Report any broken plugs or exposed electrical wires to your lecturer/laboratory technician immediately.

6. Write in time, out time and system details in the login register.

Don’ts

1. Do not eat or drink in the laboratory.

2. Do not operate mobile phones in the lab. Keep mobile phones either in silent or switched off mode.

3. Do not change system settings.

4. Do not disturb your neighbouring students. They may be busy in completing tasks.

5. Do not remove anything from the computer laboratory without permission.

6. Do not use pen drives.

7. Do not misbehave.

INTRODUCTION

Database:

—A Collection of data files (or) Collection of interrelated data (or) an individual enterprise (or) A Collection of related files (Containing records) into different groups is known as Database‖

Database System:

—The database and DBMS software together is called as Database system‖.

Database Management System:

—It is a Collection of programs that are used to store the data, Manipulate the data & retrieve the information‖y

Advantages of DBMS:

Reduction of Redundancies
Data independence & efficient Access
Data integrity
Data Security
Reduced Application Development
Conflict Resolution
Data Administration
Concurrent Access
Crash Recovery

Disadvantages of DBMS:

Conversion costs
Installation & Management Cost
New Specialized Personnel
Need for Explicit Backup & Recovery
Security Breaches

DBMS Lab

Objective:

This lab enables the students to practice the concepts learnt in the subject DBMS by developing a database for an exam**ple company named —Roadway Travels‖ whose description** is as follows: The student is expected to practice the designing, developing and querying a **database in the context of example database —Roadway Travel‖. Students are expected to use —Mysql‖ database.**

Roadways Travels:

—Roadway travels‖ is in business since 1997 with several buses connecting different places in India .Its main office is located in Hyderabad.

The company wants to computerize its operations in the following areas.
1) Reservation
2) Ticketing
3) Cancellation

Reservations:

Reservations are directly handled by booking office. Reservation can be made 60 days in advance in either cash or credit .In Case the ticket is not available, a wait listed ticket is issued to the customer. This ticket is confirmed against the cancellation.

Cancellation and Modifications:

Cancellations are also directly handed at the booking office. Cancellation charges will be charged. Wait listed tickets that do not get confirmed are fully refunded.

WEEK- 1

E-R MODEL:

Analyze the problem carefully and come up with the entities in it. Identify what data has to be persisted in the database. This contains the entities, attributes etc. Identify the primary keys for all the entities. Identify the other keys like candidate keys, partial keys, if any.

Example: **Entities**:

1. Bus
2. Ticket
3. Passenger

Primary key Attributes:

1. Ticket Id (Ticket Entity)
2. Passport Id (Passenger Entity)

A part from the above mentioned entities you can identify more. The above mentioned are few.

E-R Model:

Definition: An entity-relationship (ER) diagram is a specialized graphic that illustrates the interrelationships between entities in a database. (or)

The E-R model is a top-down approach to database design that is based on uniquely identifiable object. It begins by identifying that are uniquely distinguishable called entities and relationship among these entities.

Entity: It is a 'thing' in the real world with an independent existence. (or)
A real word object that can be distinguished from other objects is called an entity.

There are two types of entities:

1) Strong Entity
2) Weak Entity

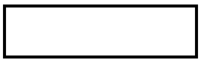
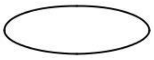
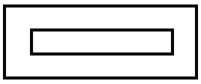
Entity type: It is a collection (set) of entities that have same attributes.

Entity set: It is a collection of all entities of particular entity type in the database.

Extension of entity type: The collections of entities of a particular entity type are grouped together into an entity set.

Attribute: It is a particular property, which describes the entity.(or) Each entity & relationship has a property called Attributes.

The difference symbols which are used to draw E-R models:

1. Rectangle (Strong entity set) 
2. Ellipse (Attributes) 
3. Rhombus (Relationship set)
4. Double Rectangle (Weak Entity set) 
5. Undirected Line (Flow of Relationship)
6. Directed Line (Flow of Relationship)

Super Key:

A Super Key of an entity set is a set of one (or) more attributes whose values uniquely determine each entity.

Candidate Key:

A Candidate Key of an entity set is a minimal super key.
For **example**: 1) Customer_Id is Candidate Key of Customer
2)Account_ Number is Candidate Key of Account

Primary Key:

Although several Candidate keys may exist, one of the Candidate keys is selected to be the Primary 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.

Example: **Entities**:

1. Bus
2. Ticket
3. Passenger

Primary key Attributes:

1. Ticket Id (Ticket Entity)
2. Passport Id (Passenger Entity)

Bus Attributes:

BusNo,

Dept_time,

Source,

Destination,

WeekDay

Ticket Attributes:

Ticket No,
Journey Date,
Source,
Destination,
Age,
Arrival Time,
Departure Time,
SEX

Passenger Attributes:

Name, PNO_NO,
Ticket No,
PPNO,
SEX,
Age

Reservation Attributes:

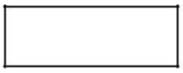
No_of_Seats,
PNR_NO,
Status,
Journey Date,
Address,
Contact No

Cancellation Attributes:

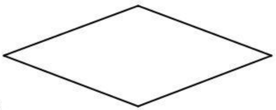
No_of_Seats,
PNR_NO,
Waiting List,
Status,
Journey Date,
Address,
Contact No

ER *diagrams often use symbols to represent three different types of information.*

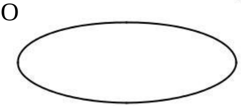
1) Boxes are commonly used to represent entities.



2) Diamonds are normally used to represent relationships



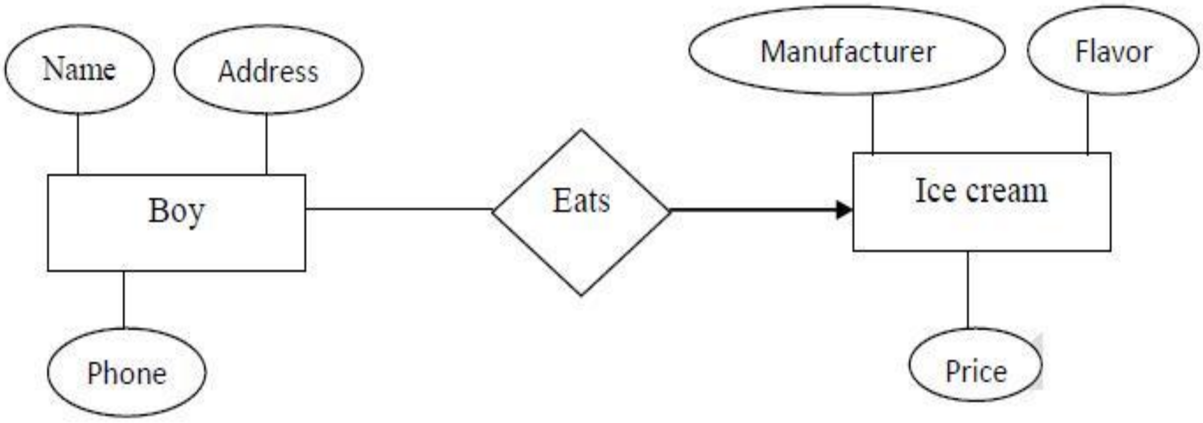
3) Ovals are used to represent attributes.



Week -2

Concept design with ER model:

Relate the entities appropriately. Apply cardinalities for each relationship. Identify Strong entities and Weak entities (if any). Indicate the type of relationships (total/Partial). Try to incorporate generalization, Aggregation, Specialization etc. wherever required.



Relationship:-

It is defined as an association among several entities. A relationship can be one-to-one, one-to-many (or) many-to-many.
A Collection of similar relationships is called a relationship set and is denoted by a Rhombus.

Strong Entity: It is the one that does not depend on other entities.

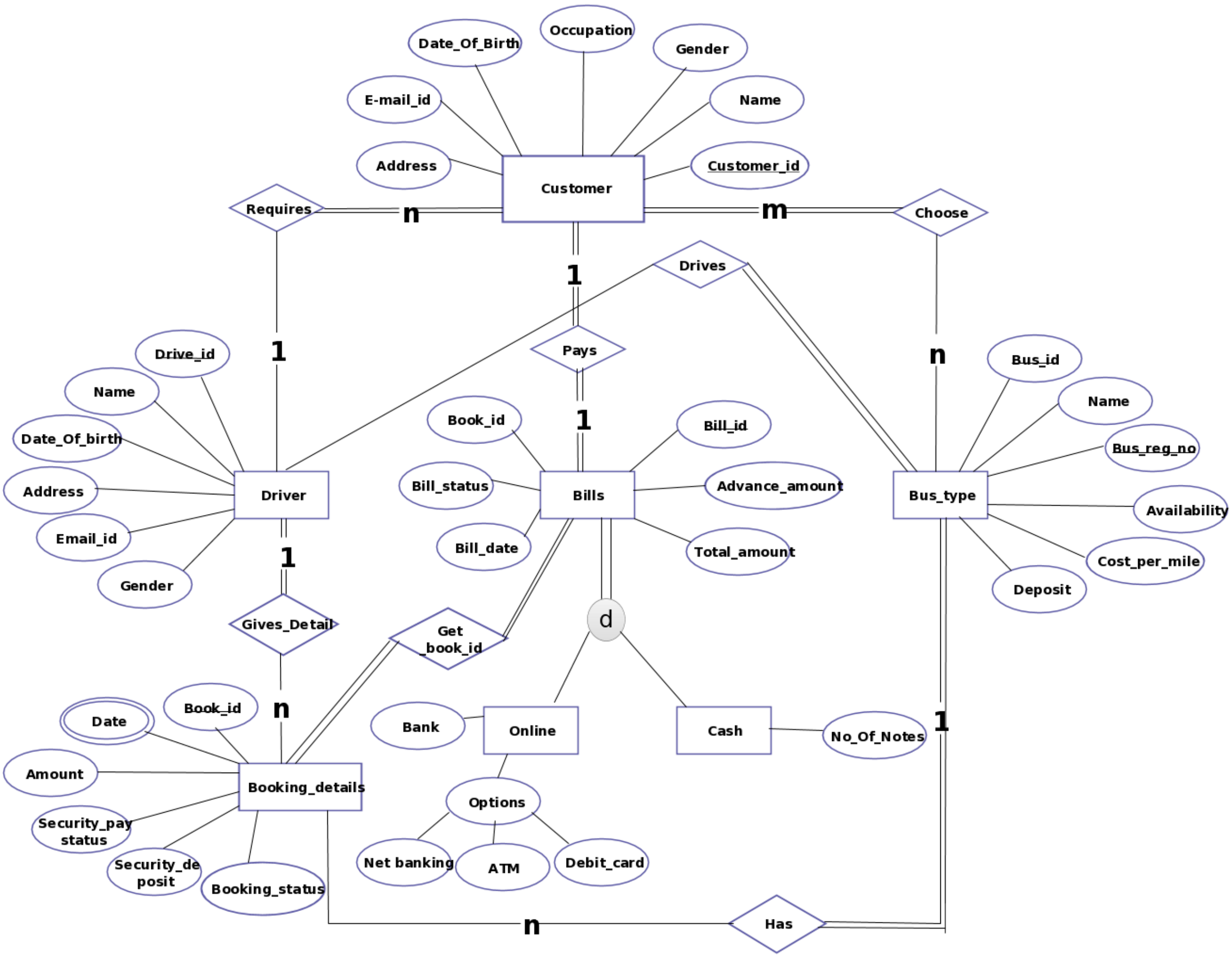
Weak Entity: It is the one that depends on other entities for existence.

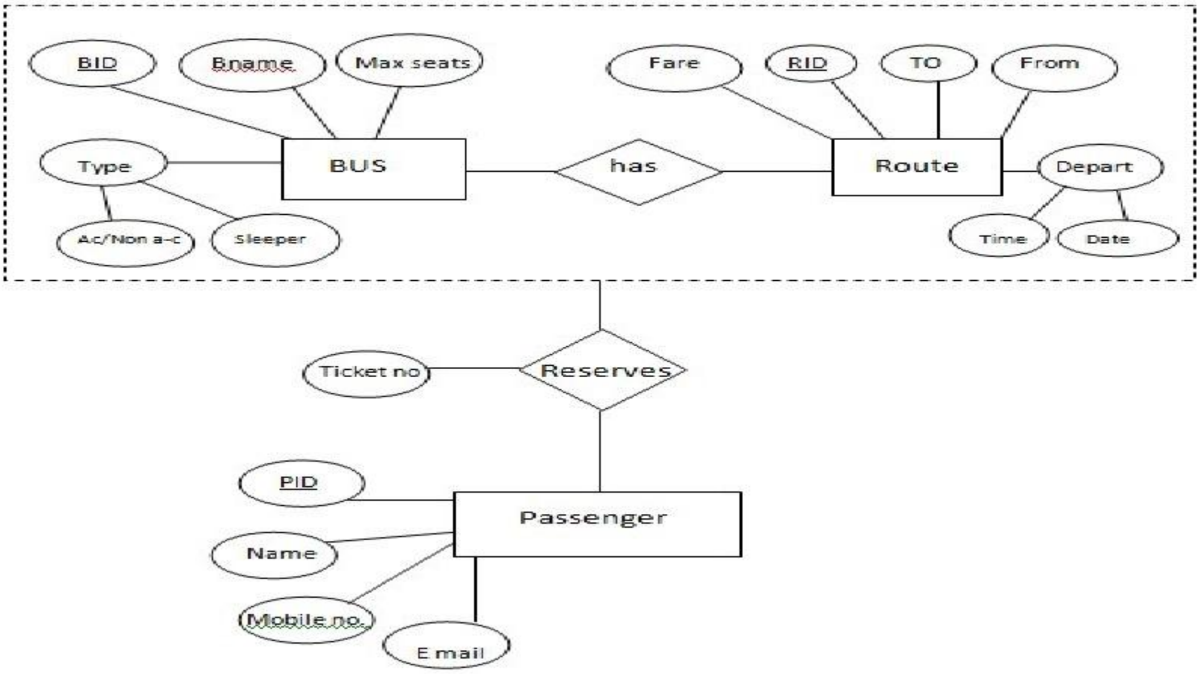
Specialization: All the entities within an entity set do not share all the attributes.

Generalization: Generalization is a special case of specialization.
The high level entity is called Super Class & low-level entity is called a Sub Class.

Aggregation: The drawback of ER diagrams is that can not show the relationship among relationship.

Bus Reservation System




BID
Bname
Max seats
Fare
RID
TO
From
Type
BUS
has
Route
Depart
Ac/Non a-c
Sleeper
Time
Date
Ticket no
Reserves
PID
Passenger
Name
Mobile no.
E mail

WEEK -3

RELATIONAL MODEL:

Relationship:-

It is defined as an association among several entities. A relationship can be one-to-one, one-to-many (or) many-to-many.

A Collection of similar relationships is called a relationship set and is denoted by a Rhombus. **Unary Relationship**---Where the association is within a single entity

Binary Relationship---A relationship that associates two entities
Ternary Relationship---A relationship that association three entities
Quaternary Relationship—A relationship that associates with four entities

Attribute: It is a particular property, which describes the entity. (or) Each entity & relationship has a property called Attributes.

1) **Simple Attribute** -- These attributes are also called as atomic attribute. These cannot be subdivided further.
2) **Composite Attributes** -- An attribute which can be further divided into smaller components are called Composite Attribute.
3) **Single-Valued Attributes** -- Certain Attribute take only a single value in all instances.
4) **Multi-Valued Attributes --** Attributes that can have more than one value at a time for an instance.
5) **Stored & Derived Attributes --** Some attributes need not be stored but can be derived from available other attributes.

BUS:

```
SQL> CREATE TABLE BUS (BusNo varchar (10) primary key, Source varchar
(15), Destination varchar (15), Weekday varchar (10));

Table Created.

SQL> insert into Bus values('AP01','Hyderabad','Karimnagar',‘Sunday‘);
1 row created.
SQL> insert into Bus values('AP02','Delhi','Mumbai',‘Monday‘);
1 row created.
SQL> insert into Bus values('AP03','Chennai','Srinagar',‘Wednesday‘);
1 row created.
SQL> insert into Bus values('AP04','Bangalore','Hyderabad',‘Sunday‘);
1 row created.
SQL> insert into Bus
values('AP05','Karimnagar','Warangal',‘Friday‘); 1 row created.
```

```
SQL> Select * from Bus;

 BusNo            Source              Destination              Weekday
-----------     -------------     ------------------         ----------------
 AP01             Hyderabad           Karimnagar                Sunday
 AP02               Delhi               Mumbai                  Monday
 AP03              Chennai             Srinagar                Wednesday
 AP04             Bangalore            Hyderabad                Sunday
 AP05             Karimnagar           Warangal                 Friday
```

PASSENGER:

```
SQL> CREATE TABLE Passenger(PNR_No NUMERIC (9) primary key, Ticket_No
Numeric(9),Name varchar (15),Age number(4),Sex Char(10),PPNO varchar (15),Category
varchar (8));

Table created.

SQL> insert into Passenger
values(1000,01,'Anitha',35,'F',10,‘A/C‘); 1 row created.
SQL> insert into Passenger values(2000,02,'Haritha',30,'F',20,‘NONA/C‘)
; 1 row created.
SQL> insert into Passenger values(3000,03,'Srilatha',22,'F',30, ‘A/C‘)
; 1 row created.
SQL> insert into Passenger values(4000,04,'Chaitanya',25,'M',40, ‘A/C‘)
; 1 row created.
SQL> insert into Passenger values(5000,05,'Saketh',28,'M',50, ‘NONA/C‘)
; 1 row created.
SQL> insert into Passenger values(6000,06,'Anirudh',27,'M',60,
‘NONA/C‘); 1 row created.

SQl> Select * from Passenger;

 PNR_No  Ticket_No       Name          Age     Sex       PPNO      Category
------------   --------------  ----------------  --------  ---------    -------------  --------------
 1000            1          Anitha         35       F         10         A/C
 2000            2          Haritha        30       F         20        NONA/C
 3000            3          Srilatha        22      F         30         A/C
 4000            4          Chaitanya      25       M          40        A/C
 5000            5           Saketh        28       M         50        NONA/C
 6000            6          Anirudh        27       M          60       NONA/C
```

TICKET:

```
SQL> CREATE TABLE Ticket(Ticket_No NUMERIC(9) Primary Key,Journey_date
Date,Age NUMERIC(4),Sex varchar(10),Source Varchar(10),Arrival_time
varchar(6),Destination Varchar(10),Dep_time varchar(6));
```

Table created.

```
SQL> insert into Ticket values(1,‘01-jan-
2010‘,35,'F','Hyderabad',9,'Karimnagar',23); 1 row created.
SQL> insert into Ticket values(2,‘02-jan-
2010‘,30,'F','Delhi',5,'Mumbai',18); 1 row created.
SQL> insert into Ticket values(3,‘03-jan-
2010‘,22,'F','Chennai',6,'Srinagar',16); 1 row created.
SQL> insert into Ticket values(4,‘04-jan-2010‘,25,'M','Bangalore',8,'Hyderabad',14);
1 row created.
SQL> insert into Ticket values(5,‘05-jan-2010‘,28,'M','Karimnagar',5,'Warangal',13);
1 row created.
```

```
SQL> Select * from Ticket;
Ticket_No  Journey_date   Age  Sex    Source   Arrival_time  Destination  Dep_Time
---------  ------------   ---- ----   -------- ------------  -----------  --------
   1       01-jan-2010    35   F     Hyderabad      9         Karimnagar     23
   2       02-jan-2010    30   F       Delhi        5           Mumbai       18
   3       03-jan-2010    22   F     Chennai        6          Srinagar      16
   4       04-jan-2010    25   M     Bangalore      8         Hyderabad      14
   5       05-jan-2010    28   M    Karimnagar      5          Warangal      13
```

RESERVATION:

```
SQL> CREATE TABLE Reserve(PNR_No NUMERIC(9), foreign key(PNR_NO) references
passenger(PNR_NO),Journey_date date,No_of_seats number(8),Address
Varchar(40),Contact_No NUMERIC(10),Status Char(2));
```

Table Created.

```
SQL> insert into reserve values(1000,'1-feb-2010‘,5,'Ramanthpur',0123456789,'Y');
1 row created.
SQL> insert into reserve values(2000,‘2-feb-
2010‘,2,'KPHB',1234567890,'Y'); 1 row created.
SQL> insert into reserve values(3000,‘3-feb-2010‘,3,'Dilsukhnagar',1234567809,'Y');
1 row created.
SQL> insert into reserve values(4000,‘4-feb-2010‘,4,'Tarnaka',1234123412,'Y');
1 row created.
SQL> insert into reserve values(5000,‘5-feb-
2010‘,5,'DDCOLONY',1234512345,'Y'); 1 row created.
```

```
SQL> Select * from Reserve;

 PNR_No   Journey_date          No_of_seats      Address        Contact_No      Status
------------ --------------------    ------------------    -------------    ------------------ ----------
    1000        1-feb-2010                1           Ramanthpur      0123456789        Y
    2000        2-feb-2010                2              KPHB        1234567890        Y
    3000        3-feb-2010                3          Dilsukhnagar   1234567809        Y
    4000        4-feb-2010                4            Tarnaka       1234123412         Y
    5000        5-feb-2010                5          DDCOLONY    1234512345        Y
```

CANCELLATION:

```
SQL> CREATE TABLE Cancellation(PNR_No NUMERIC(9), foreign
key(PNR_NO) references passenger(PNR_NO),Journey_date date,No_of_seats
NUMERIC(8), Address Varchar(40),Contact_No NUMERIC(10),Status char(2));

Table Created.

SQL> insert into cancellation values(1000,'1-Jan-
2010',5,'Ramanthpur',0123456789,'N'); 1 row created.
SQL> insert into cancellation values(2000,'2-feb-
2010',5,'KPHB',1234567890,'N'); 1 row created.
SQL> insert into cancellation values(3000,'3-feb-
2010',4,'Dilsukhnagar',1234567809,'N'); 1 row created.
SQL> insert into cancellation values(4000,'4-Apr-
2010',2,'tarnaka',1234123412,'N'); 1 row created.
SQL> insert into cancellation values(5000,'5-May-
2010',6,'DDCOLONY',1234512345,'N'); 1 row created.

SQL> Select * from Cancellation;

 PNR_No   Journey_date          No_of_seats      Address        Contact_No      Status
------------ --------------------     ------------------    -------------     ------------------ ----------
    1000        1-jan-2010                  5           Ramanthpur      0123456789       Y
    2000        2-feb-2010              5              KPHB          1234567890       Y
    3000        3-feb-2010              4          Dilsukhnagar     1234567809       Y
    4000        4-Apr-2010              2            Tarnaka        1234123412        Y
    5000        5-May-2010              6          DDCOLONY      1234512345       Y
```

WEEK -4:

Normalization:

Database normalization is a technique for designing relational database tables to minimize duplication of information and, in so doing, to safeguard the database against certain types of logical (or) structural problems, namely data anomalies. For example, when multiple instances of a given piece of information occur in a table, the possibility exists that these instances will not be kept consistent when the data within the table is updated, leading to a loss of data integrity. A table that is sufficiently normalized is less vulnerable to problems of this kind, because its structure reflects the basic assumptions for when multiple instances of the same information should be represented by a single instance only.

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.

Functional Dependencies: 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.

Normal forms: The process of normalization is based on the concept of normal forms, Each & every normal form has its own set of properties & Constraints. A table / relation is said to be in normal form if it satisfies all the properties of that normal form.

1 NF : The domain of attribute must include only atomic (simple, indivisible) values.

2 NF: 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.

3 NF: 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.

4 NF: 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.

5 NF: 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.

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.

Week 5 :

Installation of MySql and practicing DDL Commands:

Installation of MySql. In this week you will learn Creating Databases, How to create tables, altering the database, dropping tables & databases if not required. You will also try truncate, rename commands etc.

Example for creation of a table.

Create table passenger (passport id Integer primary key, name char(50) null, age integer, Sex char);

Different types of commands in SQL:

A). **DDL commands: -** To create a database objects
B). **DML commands: -** To manipulate data of a database objects
C). **DQL command: -** To retrieve the data from a database.
D). **DCL/DTL commands: - To control the data of a database…**

DDL commands:

1. The Create Table Command: - it defines each column of the table uniquely. Each column has minimum of three attributes, a name, data type and size.

Syntax: Create table <table name> (<col1> <datatype>(<size>),<col2> <datatype><size>)); **Ex:** Create table emp(empno number(4) primary key, ename char(10));

2. Modifying the structure of tables: Alter command is used to add the column to the already created table.
a) add new columns

Syntax: Alter table <tablename> add(<new col><datatype(size),<new col>datatype(size)); **Ex:** alter table emp add(sal number(7,2));

3. Dropping a column from a table:
Syntax: Alter table <tablename> drop column <col>;
Ex: alter table emp drop column sal;

4. Modifying existing columns: By using modify, we can change the datatype of a particular field

Syntax: Alter table <tablename> modify(<col><newdatatype>(<newsize>));
Ex: alter table emp modify(ename varchar2(15));

5. Renaming the tables: It is used to change the table name which was already created.

Syntax: Rename <oldtable> to <new table>;
Ex: rename emp to emp1;

6. Truncating the tables: This DDL will support to delete all the rows of a table for permanent

Syntax: Truncate table <tablename>;
Ex: trunc table emp1;

7. Destroying tables. Drop command is used to destroy the existing table (or) a view.

Syntax: Drop table <tablename>;
Ex: drop table emp;

BUS:

```
SQL> CREATE TABLE BUS (BusNo varchar (10) primary key, Source varchar (15),
Destination varchar (15));
Table Created.

SQL> desc Bus

     Name                                            Null?        Type
-----------------------------------------------      -------------------  -------------------------
      BusNo                                                         varchar2(10)
      Source                                                        varchar2(15)
      Destination                                                   varchar2(15)

SQL> Alter table Bus add (Weekday varchar (10));
Table Altered.

SQL> desc Bus

     Name                                            Null?        Type
-----------------------------------------------      -------------------  -------------------------
      BusNo                                                         varchar2(10)
      Source                                                        varchar2(15)
      Destination                                                   varchar2(15)
      Weekday                                                       varchar2(10)

SQL> Alter table Bus modify (BusNo varchar(10));
Table Altered.

SQL> Alter table Bus drop column BusNo;
Table Altered.

SQL> desc Bus

     Name                                            Null?        Type
-----------------------------------------------      -------------------  -------------------------
      Source                                                        varchar2(15)
      Destination                                                   varchar2(15)
      Weekday                                                       varchar2(10)

SQL> Drop table bus;

Table Dropped.
```

```
L> Rename Bus to Bus1;

Table Renamed.

SQL> desc Bus1
     Name                                              Null?       Type
 -----------------------------------------------   ---------   ----------------
      Source                                                     varchar2(15)
      Destination                                                varchar2(15)
      Weekday                                                    varchar2(10)

SQL> Truncate table bus;

Table Truncated.
```

PASSENGER:

```
SQL> CREATE TABLE Passenger(PNR_No NUMERIC (9) primary key,
Ticket_No Numeric(9),Name varchar (15),Age number(4),Sex Char(10),PPNO
varchar (15)); Table created.

SQL> desc Passenger

     Name                                              Null?       Type
 -----------------------------------------------   ---------   ----------------
      PNR_No                                         Not  Null    Number(9)
      Ticket_No                                                   Number(9)
      Name                                                        varchar2(15)
      Age                                                         Number(4)
      Sex                                                         Char(10)

      PPNO                                                        varchar2(15)

SQL> Alter table Passenger add (Category varchar (8));
Table Altered.

SQL> desc Passenger

     Name                                              Null?       Type
 -----------------------------------------------   ---------   ----------------
      PNR_No                                         Not  Null    Number(9)
      Ticket_No                                                   Number(9)
      Name                                                        varchar2(15)
      Age                                                         Number(4)
      Sex                                                         Char(10)
      PPNO                                                        varchar2(15)
```

```
Table Altered.


SQL> Alter table Passenger drop column PNR_NO;

Table Altered.

SQL> desc Passenger
Name                        Null?                        Type
------------------------------      -------------------      -------------------      -----------------------
--
Ticket_No                                   Number(9)
Name                                        varchar2(15)
Age                                         Number(4)
Sex                                         Char(10)
PPNO                                        varchar2(15)
Category                                    varchar2(8)

SQL> Drop table Passenger;
Table Dropped.
SQL> Rename Passenger to Passenger1;
Table Renamed.
SQL> desc Passenger1
Name            Null?                  Type
------------------        -------------------------------      -------------------      -----------------------
--
Ticket_No                Number(9)
Name                     varchar2(15)
Age                      Number(4)
Sex                      Char(10)
PPNO                     varchar2(15)
Category                 varchar2(8)

SQL> Truncate table passenger1;

Table Truncated.

TICKET:

SQL> CREATE TABLE Ticket(Ticket_No NUMERIC(9) Primary Key,Journey_date
Date,Age NUMERIC(4),Sex varchar(10),Source Varchar(10),Arrival_time
varchar(6),Destination Varchar(10));

Table created.

SQL> desc Ticket
Name   Null?   Type
-------------------------------------------------      -------------------      ------------------------
```

```
      TICKET_No                           Not  Null      Number(9)
      Journey_Date                                       Date
      Age                                                 Number(4)
      Sex                                                 varchar2(10)
      Source                                               varchar2(10)
      Arrival_Time                                        varchar2(6)
      Destination                                          varchar2(10)
```

```
SQL> Alter table Ticket add (Dept_Time varchar(6));
Table Altered.
```

```
SQL> desc Ticket

       Name                                      Null?      Type
----------------------------------------------   ---------  -------------------------
      TICKET_No                            Not  Null      Number(9)
      Journey_Date                                         Date
      Age                                                  Number(4)
      Sex                                                   varchar2(10)
      Source                                                varchar2(10)
      Arrival_Time                                          varchar2(6)
      Destination                                            varchar2(10)
      Dept_Time                                              varchar2(6)
```

```
SQL> Alter table Ticket modify (Ticket_No number(9));
Table Altered.
```

```
SQL> Alter table Ticket drop column Ticket_No;
Table Altered.
```

```
SQL> desc Ticket
         Name                                                   Null?      Type
------------------------------------------------      ------------------- -------------------------
     Journey_Date                                                            Date
     Age                                                                     Number(4)
     Sex                                                                     varchar2(10)
     Source                                                                  varchar2(10)
     Arrival_Time                                                            varchar2(6)
     Destination                                                             varchar2(10)
     Dept_Time                                                               varchar2(6)
```

```
SQL> Drop table Ticket;

Table Dropped.

SQL> Rename Ticket to Ticket1;

Table Renamed.
```

```
SQL> desc Ticket1

          Name                                                  Null?       Type
-----------------------------------------------           ------------------  ------------------------
     Journey_Date                                                            Date
     Age                                                                      Number(4)
     Sex                                                                      varchar2(10)
     Source                                                                   varchar2(10)
     Arrival_Time                                                             varchar2(6)
     Destination                                                              varchar2(10)
     Dept_Time                                                                varchar2(6)
```

```
 SQL> Truncate table Ticket1;

Table Truncated.
```

RESERVATION:

```
SQL> CREATE TABLE Reserve(PNR_No NUMERIC(9), foreign key(PNR_NO) references
passenger(PNR_NO),Journey_date date,No_of_seats number(8),Address
Varchar(40),Contact_No NUMERIC(10));

Table Created.

SQL> desc Reserve

    Name                                                Null?        Type
-----------------------------------------------         ------------------  ------------------------
     PNR_No                                                                 Number(9)
     Journey_Date                                                           Date
     No_Of_Seats                                                            Number(8)
     Address                                                                varchar2(40)
     Contact_No                                                             Number(10)

SQL> Alter table Reserve add (status char(2));
Table Altered.

SQL> desc Reserve
      Name                                                  Null?       Type
-----------------------------------------------         ------------------  ------------------------
     PNR_No                                                                 Number(9)
     Journey_Date                                                           Date
     No_Of_Seats                                                            Number(8)
     Address                                                                varchar2(40)
     Contact_No                                                             Number(10)
     Status                                                                 Char(2)
```

```
SQL> Alter table Reserve modify (PNR_No number(9));
```

```
Table Altered.
SQL> Alter table Reserve drop column PNR_No;
Table Altered.

SQL> desc Reserve
        Name                                             Null?        Type
-----------------------------------------------       -------------------  -------------------------
      Journey_Date                                                    Date
      No_Of_Seats                                                     Number(8)
      Address                                                         varchar2(40)
      Contact_No                                                      Number(10)
      Status                                                           Char(2)

SQL> Drop table Reserve;

Table Dropped.

SQL> Rename Reserve to Reserve1;

Table Renamed.

SQL> desc Reserve1
        Name                                             Null?        Type
-----------------------------------------------       -------------------  -------------------------
      Journey_Date                                                    Date
      No_Of_Seats                                                      Number(8)
      Address                                                         varchar2(40)
      Contact_No                                                      Number(10)
      Status                                                           Char(2)

 SQL> Truncate table Reserve1;

Table Truncated.
```

CANCELLATION:

```
SQL> CREATE TABLE Cancellation(PNR_No NUMERIC(9), foreign
key(PNR_NO) references passenger(PNR_NO),Journey_date date,No_of_seats
NUMERIC(8), Address Varchar(40),Contact_No NUMERIC(10));
 Table Created.
```

```
SQL> desc Cancellation

   Name                                           Null?      Type
------------------------------------------------  --------------------  -------------------------
     PNR_No                                                             Number(9)
     Journey_Date                                                       Date
     No_Of_Seats                                                        Number(8)
     Address                                                            varchar2(40)
```

```
     Contact_No                                                         Number(10)

SQL> Alter table Cancellation add (status char(2));
Table Altered.

SQL> desc Cancellation
   Name                                           Null?      Type
------------------------------------------------  --------------------  -------------------------
     PNR_No                                                             Number(9)
     Journey_Date                                                       Date
     No_Of_Seats                                                        Number(8)
     Address                                                            varchar2(40)
     Contact_No                                                         Number(10)
     Status                                                             Char(2)

SQL> Alter table Cancellation modify (PNR_No number(9));
Table Altered.
SQL> Alter table Cancellation drop column PNR_No;
Table Altered.

SQL> desc Cancellation
   Name                                           Null?      Type
------------------------------------------------  --------------------  -------------------------
     Journey_Date                                                       Date
     No_Of_Seats                                                        Number(8)
     Address                                                            varchar2(40)
     Contact_No                                                         Number(10)
     Status                                                             Char(2)

SQL> Drop table Cancellation;

Table Dropped.

SQL> Rename Cancellation to Cancellation1;

Table Renamed.
```

```
SQL> desc Cancellation1
      Name                                         Null?       Type
----------------------------------------     ---------------- ----------------------
    Journey_Date                                               Date
    No_Of_Seats                                                Number(8)
    Address                                                    varchar2(40)
    Contact_No                                                 Number(10)
    Status                                                      Char(2)
```

```
 SQL> Truncate table Cancellation1;

Table Truncated.
```

Week 6 :

DML commands:

DML Commands:-to manipulate data of a database objects
1) insert
2) update
3) delete
4) truncate

1. **Inserting Data into Tables: -** once a table is created the most natural thing to do is load this table with data to be manipulated later.

Syntax:
insert into <tablename> (<col1>,<col2>) values(<exp>,<exp>);

2. **Delete operations.**

a) remove all rows
Syntax:
delete from <tablename>;

b) removal of a specified row/s
Syntax:
delete from <tablename> where <condition>;

3. **Updating the contents of a table.**

a) updating all rows
Syntax:
Update <tablename> set <col>=<exp>,<col>=<exp>;

b) updating seleted records.
Syntax:
Update <tablename> set <col>=<exp>,<col>=<exp> where <condition>;

BUS:

```
SQL> CREATE TABLE BUS (BusNo varchar (10) primary key, Source varchar (15),
Destination varchar (15), Weekday varchar (10));

Table Created.

SQL> insert into Bus values('AP01','Hyderabad','Karimnagar',‘Sunday‘);
1 row created.
SQL> insert into Bus values('AP02','Delhi','Mumbai',‘Monday‘);
1 row created.
```

```
SQL> insert into Bus values('AP03','Chennai','Srinagar',‘Wednesday‘);
1 row created.
SQL> insert into Bus values('AP04','Bangalore','Hyderabad',‘Sunday‘);
1 row created.
SQL> insert into Bus values('AP05','Karimnagar','Warangal',‘Friday‘);
1 row created.

SQL> Select * from Bus;

 BusNo          Source          Destination          Weekday
-----------   -------------   ------------------   ----------------
 AP01          Hyderabad       Karimnagar            Sunday
 AP02            Delhi          Mumbai               Monday
 AP03           Chennai          Srinagar            Wednesday
 AP04          Bangalore        Hyderabad             Sunday
 AP05          Karimnagar        Warangal             Friday

SQL> update bus set Destination=‘Nizamabad‘ where busno=‘AP01‘;

1 row updated.

SQL> Select * from Bus;

 BusNo          Source          Destination          Weekday
-----------   -------------   ------------------   ----------------
 AP01          Hyderabad        Nizambad              Sunday
 AP02            Delhi          Mumbai               Monday
 AP03           Chennai          Srinagar            Wednesday
 AP04          Bangalore        Hyderabad             Sunday
 AP05          Karimnagar        Warangal             Friday

SQL> update bus set weekday=null;

5 rows updated.

SQL> Select * from Bus;

 BusNo          Source          Destination            Weekday
-----------   -------------   ------------------    ----------------
 AP01          Hyderabad        Nizambad
 AP02            Delhi          Mumbai
 AP03           Chennai          Srinagar
 AP04          Bangalore         Hyderabad
 AP05          Karimnagar        Warangal


SQL> delete from bus where Source='Chennai';
```

```
1 row deleted.

SQL> Select * from Bus;



 BusNo            Source            Destination                  Weekday
-----------     -------------     ------------------          ----------------
 AP01           Hyderabad          Nizambad
 AP02              Delhi            Mumbai
 AP04           Bangalore          Hyderabad
 AP05          Karimnagar          Warangal
```

PASSENGER:

```
SQL> CREATE TABLE Passenger(PNR_No NUMERIC (9) primary key, Ticket_No
Numeric(9),Name varchar (15),Age number(4),Sex Char(10),PPNO varchar (15),Category
varchar (8));

Table created.

SQL> insert into Passenger values(1000,01,'Anitha',35,'F',10,‘A/C‘);
1 row created.
SQL> insert into Passenger
values(2000,02,'Haritha',30,'F',20,‘NONA/C‘) ; 1 row created.
SQL> insert into Passenger values(3000,03,'Srilatha',22,'F',30, ‘A/C‘) ;
1 row created.
SQL> insert into Passenger values(4000,04,'Chaitanya',25,'M',40, ‘A/C‘) ;
1 row created.
SQL> insert into Passenger values(5000,05,'Saketh',28,'M',50, ‘NONA/C‘) ;
1 row created.
SQL> insert into Passenger values(6000,06,'Anirudh',27,'M',60, ‘NONA/C‘);
1 row created.

SQl> Select * from Passenger;

 PNR_No  Ticket_No      Name            Age      Sex      PPNO        Category
------------   --------------  ----------------  --------   ---------  -------------  --------------
 1000            1          Anitha          35       F        10          A/C
 2000            2          Haritha         30       F        20        NONA/C
 3000            3          Srilatha        22       F        30           A/C
 4000            4          Chaitanya       25       M        40           A/C
 5000            5           Saketh         28       M        50        NONA/C
 6000            6          Anirudh         27       M         60       NONA/C

SQL> update Passenger set Name=‘Rajitha‘ where pnr_no=‘2000‘;
```

```
1 row updated.

SQl> Select * from Passenger;
```

```
 PNR_No  Ticket_No      Name          Age      Sex      PPNO        Category
------------   --------------   ----------------  --------   ---------   -------------   --------------
 1000            1          Anitha         35       F        10           A/C
 2000            2          Rajitha         30      F         20         NONA/C
 3000            3          Srilatha        22      F         30           A/C
 4000            4          Chaitanya      25      M         40           A/C
 5000            5           Saketh        28      M         50         NONA/C
 6000            6          Anirudh        27      M          60        NONA/C

SQL> update Passenger set sex=null;

6 rows updated.

SQl> Select * from Passenger;

 PNR_No  Ticket_No      Name          Age      Sex      PPNO        Category
------------   --------------   ----------------  --------   ---------   -------------   --------------
 1000            1          Anitha         35                  10            A/C
 2000            2          Rajitha         30                 20          NONA/C
 3000            3          Srilatha        22                  30           A/C
 4000            4          Chaitanya      25                 40            A/C
 5000            5           Saketh        28                 50         NONA/C
 6000            6          Anirudh        27                 60         NONA/C

SQL> delete from Passenger where name='Anirudh';

1 row deleted.

SQl> Select * from Passenger;

 PNR_No  Ticket_No      Name          Age      Sex      PPNO        Category
------------   --------------   ----------------  --------   ---------   -------------   --------------
 1000            1          Anitha        35                   10            A/C
 2000            2          Rajitha         30                 20          NONA/C
 3000            3          Srilatha        22                  30           A/C
 4000            4          Chaitanya      25                 40            A/C
 5000            5           Saketh       28                  50         NONA/C
```

TICKET:

```
SQL> CREATE TABLE Ticket(Ticket_No NUMERIC(9) Primary Key,Journey_date
Date,Age NUMERIC(4),Sex varchar(10),Source Varchar(10),Arrival_time
varchar(6),Destination Varchar(10),Dep_time varchar(6));
Table created.

SQL> insert into Ticket values(1,‘01-jan-
2010‘,35,'F','Hyderabad',9,'Karimnagar',23); 1 row created.
SQL> insert into Ticket values(2,‘02-jan-2010‘,30,'F','Delhi',5,'Mumbai',18);
1 row created.
SQL> insert into Ticket values(3,‘03-jan-2010‘,22,'F','Chennai',6,'Srinagar',16);

1 row created.
SQL> insert into Ticket values(4,‘04-jan-2010‘,25,'M','Bangalore',8,'Hyderabad',14);
1 row created.
SQL> insert into Ticket values(5,‘05-jan-2010‘,28,'M','Karimnagar',5,'Warangal',13);
SQL> Select * from Ticket;

Ticket_No  Journey_date   Age  Sex     Source   Arrival_time  Destination  Dep_Time
---------  ------------  ---- ----  ----------  ------------  -----------  --------
    1       01-jan-2010    35   F    Hyderabad        9        Karimnagar      23
    2       02-jan-2010    30   F      Delhi          5          Mumbai        18
    3       03-jan-2010    22   F     Chennai         6         Srinagar       16
    4       04-jan-2010    25   M    Bangalore        8        Hyderabad       14
    5       05-jan-2010    28   M   Karimnagar        5         Warangal       13


SQL> update Ticket set source=‘Nizamabad‘ where
ticket_no=‘1‘; 1 row updated.

SQL> Select * from Ticket;

Ticket_No  Journey_date   Age  Sex     Source   Arrival_time  Destination  Dep_Time
---------  ------------  ---- ----  ----------  ------------  -----------  --------
    1       01-jan-2010    35   F    Nizamabad        9        Karimnagar      23
    2       02-jan-2010    30   F      Delhi          5          Mumbai        18
    3       03-jan-2010    22   F     Chennai         6         Srinagar       16
    4       04-jan-2010    25   M    Bangalore        8        Hyderabad       14
    5       05-jan-2010    28   M   Karimnagar        5         Warangal       13

SQL> update Ticket set sex=null;

5 rows updated.

SQL> Select * from Ticket;

Ticket_No  Journey_date  Age   Sex     Source     Arrival_time Destination  Dep_Time
---------  ------------  ---- ---- ----------  ------------  -----------  --------
    1       01-jan-2010   35        Nizamabad        9        Karimnagar      23
    2       02-jan-2010   30          Delhi          5          Mumbai        18
```

```
      3          03-jan-2010     22             Chennai            6         Srinagar           16
      4          04-jan-2010     25            Bangalore           8        Hyderabad           14
      5          05-jan-2010     28           Karimnagar            5        Warangal             13

SQL> delete from Ticket where source='Chennai';

1 row deleted.

SQL> Select * from Ticket;

Ticket_No  Journey_date  Age   SexSource          Arrival_time  Destination  Dep_Time
------------- ----------------  ------  ------  -----------  -----------------  --------------  --------------
```

```
   1        01-jan-2010      35        Nizamabad       9      Karimnagar       23
   2        02-jan-2010      30          Delhi         5        Mumbai         18
   4        04-jan-2010      25       Bangalore        8      Hyderabad        14
   5        05-jan-2010      28       Karimnagar       5       Warangal         13
```

RESERVATION:

```
SQL> CREATE TABLE Reserve(PNR_No NUMERIC(9), foreign key(PNR_NO) references
passenger(PNR_NO),Journey_date date,No_of_seats number(8),Address
Varchar(40),Contact_No NUMERIC(10),Status Char(2));

Table Created.

SQL> insert into reserve values(1000,'1-feb-2010‘,5,'Ramanthpur',0123456789,'Y');
1 row created.
SQL> insert into reserve values(2000,‘2-feb-
2010‘,2,'KPHB',1234567890,'Y'); 1 row created.
SQL> insert into reserve values(3000,‘3-feb-2010‘,3,'Dilsukhnagar',1234567809,'Y');
1 row created.
SQL> insert into reserve values(4000,‘4-feb-2010‘,4,'Tarnaka',1234123412,'Y');
1 row created.
SQL> insert into reserve values(5000,‘5-feb-
2010‘,5,'DDCOLONY',1234512345,'Y'); 1 row created.

SQL> Select * from Reserve;

PNR_No   Journey_date        No_of_seats     Address       Contact_No     Status
------------  --------------------    ------------------    -------------    - -----------------  ----------
   1000       1-feb-2010            1         Ramanthpur    0123456789       Y
   2000       2-feb-2010            2            KPHB        1234567890      Y
   3000       3-feb-2010            3         Dilsukhnagar   1234567809      Y
   4000       4-feb-2010            4           Tarnaka      1234123412      Y
   5000       5-feb-2010            5         DDCOLONY       1234512345      Y
SQL> update Reserve set Address=‘Hyd‘ where pnr_no=‘2000‘;
1 row updated.

SQL> Select * from Reserve;

PNR_No   Journey_date        No_of_seats     Address       Contact_No     Status
 ------------  --------------------    ------------------    -------------    - -----------------  ----------
   1000       1-feb-2010            1         Ramanthpur    0123456789       Y
   2000       2-feb-2010            2             Hyd        1234567890      Y
   3000       3-feb-2010            3         Dilsukhnagar   1234567809      Y
   4000       4-feb-2010            4           Tarnaka      1234123412      Y
   5000       5-feb-2010            5         DDCOLONY       1234512345      Y
```

```
SQL> update Reserve set status=null;

5 rows updated.
```

```
SQL> Select * from Reserve;


PNR_No   Journey_date         No_of_seats    Address        Contact_No      Status
------------ --------------------   ------------------   -------------   - -----------------   ----------
    1000       1-feb-2010              1          Ramanthpur    0123456789
    2000       2-feb-2010              2             Hyd           1234567890
    3000       3-feb-2010              3          Dilsukhnagar   1234567809
    4000       4-feb-2010              4            Tarnaka        1234123412
    5000       5-feb-2010              5          DDCOLONY    1234512345

SQL> delete from Reserve where journey_date='2-feb-2010';
1 row deleted.

SQL> Select * from Reserve;

PNR_No   Journey_date         No_of_seats    Address        Contact_No      Status
 ------------ --------------------   ------------------   -------------   - ----------------- ----------
   1000       1-feb-2010              1          Ramanthpur    0123456789
   3000       3-feb-2010              3          Dilsukhnagar   1234567809
   4000       4-feb-2010              4            Tarnaka        1234123412
   5000       5-feb-2010              5          DDCOLONY    1234512345
```

<u>CANCELLATION:</u>

```
SQL> CREATE TABLE Cancellation(PNR_No NUMERIC(9), foreign
key(PNR_NO) references passenger(PNR_NO),Journey_date date,No_of_seats
NUMERIC(8), Address Varchar(40),Contact_No NUMERIC(10),Status char(2));
Table Created.

SQL> insert into cancellation values(1000,'1-Jan-2010',5,'Ramanthpur',0123456789,'N');
1 row created.
SQL> insert into cancellation values(2000,'2-feb-2010',5,'KPHB',1234567890,'N');
1 row created.
SQL> insert into cancellation values(3000,'3-feb-
2010',4,'Dilsukhnagar',1234567809,'N'); 1 row created.
SQL> insert into cancellation values(4000,'4-Apr-
2010',2,'tarnaka',1234123412,'N'); 1 row created.
SQL> insert into cancellation values(5000,'5-May-2010',6,'DDCOLONY',1234512345,'N');
1 row created.
```

```
SQL> Select * from Cancellation;

 PNR_No   Journey_date        No_of_seats      Address      Contact_No        Status
 ------------ --------------------  -------------------  -------------   ------------------      ----------
    1000       1-jan-2010            5        Ramanthpur   0123456789        Y
    2000       2-feb-2010            5           KPHB      1234567890        Y
    3000       3-feb-2010            4       Dilsukhnagar  1234567809        Y
    4000       4-Apr-2010            2          Tarnaka    1234123412        Y
```

```
     5000        5-May-2010               6         DDCOLONY   1234512345        Y
```

```
SQL> update Cancellation set Address='Karimnagar' where
pnr_no='2000'; 1 row updated.

SQL> Select * from Cancellation;

 PNR_No   Journey_date        No_of_seats      Address      Contact_No      Status
 ------------ --------------------  -------------------  -------------   ------------------ ----------
    1000       1-jan-2010            5       Ramanthpur    0123456789        Y
    2000       2-feb-2010            5       Karimnagar    1234567890       Y
    3000       3-feb-2010            4       Dilsukhnagar  1234567809        Y
    4000       4-Apr-2010            2         Tarnaka     1234123412        Y
    5000       5-May-2010            6        DDCOLONY     1234512345        Y

SQL> update Cancellation set status=null;

5 rows updated.

SQL> Select * from Cancellation;

 PNR_No   Journey_date        No_of_seats      Address      Contact_No      Status
 ------------ --------------------  -------------------  -------------   ------------------ ----------
    1000       1-jan-2010            5       Ramanthpur    0123456789
    2000       2-feb-2010            5       Karimnagar    1234567890
    3000       3-feb-2010            4       Dilsukhnagar  1234567809
    4000       4-Apr-2010            2         Tarnaka     1234123412
    5000        5-May-2010                 6            DDCOLONY    1234512345

SQL> delete from cancellation where No_of_seats=2';
1 row deleted.

SQL> Select * from Cancellation;

 PNR_No   Journey_date          No_of_seats      Address       Contact_No       Status
 ------------  --------------------    ------------------    -------------      ------------------  ----------
    1000        1-jan-2010                 5            Ramanthpur     0123456789
    2000        2-feb-2010                 5            Karimnagar     1234567890
    3000        3-feb-2010                 4            Dilsukhnagar   1234567809
    5000        5-May-2010                 6            DDCOLONY    1234512345
```

Week 7 :

Querying:

In this week you are going to practice queries (along with sub queries) using ANY, All, In, Exists, Not Exists, Union, Intersect, Constraints etc

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.

Sub Query: Sub queries, or nested queries, are used to bring back a set of rows to be used by the parent query. Depending on how the sub query 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 sub query is executed for each row of the parent, this is called a correlated sub query.

Correlated sub query: A correlated sub query can be easily identified if it contains any references to the parent sub query columns in its WHERE clause. Columns from the sub query cannot be referenced anywhere else in the parent query. The following example demonstrates a non-correlated sub query.

E.g. Select * From CUST Where '10/03/1990' IN (Select ODATE from ORDER
Where CUST.CNUM = ORDER.CNUM)

Any:

This operator will make inner query to return multiple rows, where from those rows, least value is extracted & then sends that value for comparison at the condition of outer query.

Syntax: select select-list1 from <TN><alias name1>where(condition1 < comparison operator) Any (select select-list2 from <TN><alias name2> where <condition>

All:

This operator also makes inner query to return multiple rows of which it selects the biggest value & sends that value for comparison at the condition of outer query.

Syntax: select <select list> from <TN><alias name1>where<condition (op) All > (select<select list>) from <TN><alias name2> where <conditionlist>

IN:

Syntax: select <select-list> from <TN><alias name>where<condition>IN (select<select-list1>from <TN><alias name2> where <condition>

Exists:

This operator is used to co-related queries that returns those rows of outer query for which the inner query condition get satisfied.

Syntax: select <select-list> from <TN><alias name>where Exists (select * from<TN><alias name> where <condition>

Not Exists:

This operator will return true for that condition **which doesn‘t get** satisfy at inner query. If the condition becomes true it becomes false at the Boolean value & those rows will be avoided at retrieval.

Syntax: select <select-list> from <TN><alias name> where Not Exists(select * from<TN><alias name> where <condition>

Union:

It joins the rows returned by 2 (or) more queries, by avoiding duplicates & arranging the rows in ascending order.

SQL> select source from bus union select source from ticket;

Intersect:

This operator will extract the common values which are found in the rows returned by 2 (or) more queries.

SQL> select source from bus intersect select source from ticket;

Types of data constrains.

a) not null constraint at column level.
Syntax:
<col><datatype>(size)not null

b) unique constraint
Syntax:
Unique constraint at column level.
<col><datatype>(size)unique;

c) unique constraint at table level:
Syntax:
Create table tablename(col=format,col=format,unique(<col1>,<col2>);

d) primary key constraint at column level
Syntax:<col><datatype>(size)primary key;

e) primary key constraint at table level.
Syntax:
Create table tablename(col=format,col=format
primary key(col1>,<col2>);

f) foreign key constraint at column level.
Syntax:
<col><datatype>(size>) references <tablename>[<col>];

g) foreign key constraint at table level
Syntax:
foreign key(<col>[,<col>])references <tablename>[(<col>,<col>)

h) check constraint
check constraint constraint at column level.
Syntax: <col><datatype>(size) check(<logical expression>)

i) check constraint constraint at table level.
Syntax: check(<logical expression>)

<u>Practice the following queries</u>

1) **Display unique PNR_no of all passengers**

SQL>select distinct(Pnr_no) from passenger;

```
PNR_NO
------
  1000
  2000
  3000
  4000
  5000
```

2) **Display all the names of male passengers**

SQL> select name from passenger where sex='M';

```
NAME
---------------
chaitanya
saketh
```

3) **Display the ticket numbers and names of all the numbers**
SQL> select ticket_no,name from passenger;

```
 TICKET_NO   NAME
----------        ---------------
        1          Anitha
        2           haritha
        3          Srilatha
        4          chaitanya
        5          saketh
```

4) **Display the source and destination having journey time more than 10 hours.**

5) Find the ticket numbers of the passengers whose name start with 'A' and ends with 'H'.

```
NAME
---------------
Anith
Anirudh
```

```
SQL>     select ticket_no from passenger where name like 'A%h';

 TICKET_NO
----------
        6
```

6) Find the age of passengers whose age is between 30 and 45.

```
SQL> select name from passenger where age between 30 AND 45;
NAME
---------------
Anitha
Haritha
```

7) Display all the passengers names beginning with 'A'

```
SQL> select name from passenger where name like 'A%';
NAME
---------------
Anitha
Anirudh
```

8) Display the sorted list of the passenger names.

```
SQL> select name from passenger order by name;
NAME
---------------
Anirudh
Anitha
Srilatha
Chaitanya
Haritha
Saketh
```

9) Display the Bus numbers that travel on Sunday and Wednesday.

```
mysql> select BusNo from bus where Weekday='Sunday' or WeekDay='Wednesday';
```

```
+-------+
| BusNo |
+-------+
| AP01 |
| AP03 |
| AP04 |
+-------+
```

10) **Display the details of Passengers who are travelling either in AC or NONAC.**

```
mysql> select * from passenger where category IN('A/C','NONA/C');
--------+----------- +----------- +------ +------ +----- +--------- +
PNR_No | Ticket_No | Name      | Age  | Sex  | PPNO | Category |
--------+----------- +----------- +------ +------ +----- +--------- +
    1000 |       1 | Anitha     | 35 | F     | 10  | A/C       |
    2000 |       2 | haritha    | 30 | F     | 20  | NONA/C   |
    3000 |       3 | Srilatha   | 22 | F     | 30  | A/C       |
    4000 |       4 | chaitanya |   25 | M     | 40  | A/C       |
    5000 |       5 | saketh     | 28 | M    | 50     | NONA/C   |
    6000 |       6 | Anirudh   |   27 | M     | 60  | NONA/C     |
--------+----------- +----------- +------ +------ +----- +--------- +
```

WEEK-8 & WEEK-9:

Querying

You are going to practice queries using Aggregate functions(Count, Sum, Avg, Max & Min) Group By, Having & Creation & dropping of views.

Aggregate Operator:

SQL supports 5 aggregate operators, which can be applied on any column of a relation.

1. Count
2. Sum
3. Avg
4. Max
5. Min

1) Count: It is used to count the number of unique columns (or) values in a given column. **Syntax:** select count (*) from <TN>(alias name);

2) Sum: It is used to retrieve the sum of unique values of a given column.
Syntax: select sum(column name)from <TN> <alias name>where<where condition>

3)Average(Avg): It retrieves the average of all (unique) values in the given column. **Syntax :**
select Avg (column name) from <TN> <alias name>where <where condition>

4)Max: It is used to find the maximum value of a given number
Syntax: select Max (Column name) from <TN> (alias name);

5) Min: It is used to find the minimum value of a given number.
Syntax: select Min (column name) from<TN> (alias number)

Group by & having Clause:

Group By and having clause is used to group the related data into one & having is used to group qualification.

Syntax: select select-list from from-list where condition Groupby Group list Having group-qualification;

View:

The view is a table whose rows are not explicity stored in the database, but are computed as needed from the view definition.

Creating & dropping of views:

―view (or) virtual tables‖

In the SQL Languages a view is a representation of one (or) more tables.

To reduce reductant (duplicate) data to the main possible create allows the creation of an object called a view.
If a view is used to only look at table data and nothing done the view is called ‖read only view‖.

Syntax: create view <view name>as select<col name> from<Table name>
where col name= expression-list group by<grouping creation> having condition

1. Write a Query to display the information present in the passenger and cancellation tables. Hint: use Union Operator

```
 SQL> Select  pnr_no from passenger union select pnr_no from cancellation;

Pnr_no
----------
  1000
  2000
  3000
  4000
  5000
  6000
```

2. **Write a query to display different traveling options available in British Airways.**

3. **Display the number of days in a week on which the 9w01 bus is available**

4. **Find number of tickets booked for each pnr_no using group by clause. Hint: Use Group By on pnr_no**

5. **Find the distinct PNR numbers that are present.**

```
sql> select distinct(PNR_NO) from passenger reserve;

Pnr_no
```

```
----------
  1000
  2000
  3000
  4000
  5000
  6000
```

6. **Find the number of tickets booked in each class where the number of seats is greater than 1. Hint: Use Group by, Where & Having Clauses**

7. **Find the Total number of cancelled seats.**

8. **Write a query to count the number of tickets for the buses, which traveled after the date '14/3/2009' Hint: Use Having Clauses**

Week 10:
Triggers

In this week you are going to work on triggers. Creation of insert trigger, delete trigger, update trigger. Practice triggers using the above database

Eg: CREATE TRIGGER updcheck BEFORE UPDATE ON passenger
FOR EACH ROW
BEGIN
IF NEW.Ticket no> 60 then
Set new.Ticket no= Ticket no;
ELSE
SET New.Ticket no=0;
END IF;
END;

Triggers:

What are triggers?

A trigger is a PL/SQL block or a PL/SQL procedure associated with a table, view, schema, of the database. It executes implicitly whenever a particular event takes place. It can either be:

1. **Application trigger**: Fires whenever an event occurs with a particular application.
2. **Database Trigger**: Fires whenever a data event (such as DML) occurs on a schema or database.

When should we use triggers?

Triggers are to be used when we want to perform related actions and when we want to centralize global operations. Create stored procedures and invoke them in a trigger, if the PL/SQL code is very lengthy. The excessive use of triggers can result in complex interdependencies, which may be difficult to maintain in large applications. Triggers should not be used where functionality needed is already built into the Oracle server or when it is duplicating what is done by other triggers.

Example:

A trigger called check_sal which runs every time a new employee is getting inserted to enforce some rules on what should be the minimum salary.

Elements in a Trigger:

Trigger timing
- o For table: BEFORE, AFTER
- o For view: INSTEAD OF

Trigger event: INSERT, UPDATE, OR DELETE

Table name: On table, view

Trigger Type: Row or statement

When clause: Restricting condition

Trigger body: PL/SQL block

Before triggers:

―Before triggers‖ execute the trigger body before the triggering DML event on a table. These are frequently used to determine whether that triggering statement should be allowed to complete. This situation enables you to eliminate unnecessary processing of the triggering statement and it eventual rollback in cases where an exception is raised in the triggering action.

After triggers

―After triggers‖ are used when the triggering statement is to be completed before the triggering action and to perform a different action on the same triggering statement if a BEFORE trigger is already present.

Instead of Triggers

―Instead of Triggers‖ are used to provide a transparent way of modifying views that cannot be modified directly through SQL DML statements because the view is not inherently modifiable. You can write INSERT, UPDATE, and DELETE statements against the view.

The INSTEAD OF trigger works invisibly in the background performing the action coded in the trigger body directly on the underlying tables.

Triggering user events:

- INSERT
- UPDATE
- DELETE

Trigger Components:

- **Statement**: The trigger body executes once for the triggering event. This is the default. A statement trigger fires once, even if no rows are affected at all.
- **Row:** The trigger body executes once for each row affected by the triggering event. A row trigger is not executed if the triggering event affects no rows.

Trigger Body:

The trigger body is a PL/SQL block or a call to a procedure.

Syntax:

```
CREATE [OR REPLACE] TRIGGER trigger_name
      Timing
           Event1 [OR event2 OR event3]
                ON table_name
Trigger_body
```

Week 11

Procedures

In this session you are going to learn creation of stored procedure, execution of procedure & modification of procedure. Practice procedures using the above database.

Eg: CREATE PROCEDURE myProc()
BEGIN
SELECT COUNT(Tickets) FROM Ticket WHERE age>=40;
End;

Stored Procedure:

A **stored procedure** or in simple a **proc** is a named PL/SQL block which performs one or more specific task. This is similar to a procedure in other programming languages. A procedure has a header and a body. The header consists of the name of the procedure and the parameters or variables passed to the procedure. The body consists or declaration section, execution section and exception section similar to a general PL/SQL Block. A procedure is similar to an anonymous PL/SQL Block but it is named for repeated usage.

We can pass parameters to procedures in three ways.
1) IN-parameters
2) OUT-parameters
3) IN OUT-parameters

A procedure may or may not return any value.

General Syntax to create a procedure is:

```
CREATE [OR REPLACE] PROCEDURE proc_name [list of parameters]
IS
    Declaration section
BEGIN
    Execution section
EXCEPTION Exception
  section
END;
```

IS - marks the beginning of the body of the procedure and is similar to DECLARE in anonymous PL/SQL Blocks. The code between IS and BEGIN forms the Declaration section.

The syntax within the brackets [ ] indicate they are optional. By using CREATE OR REPLACE together the procedure is created if no other procedure with the same name exists or the existing procedure is replaced with the current code.

The below example creates a procedure _employer_details‘ which gives the details of the employee.

```
CREATE OR REPLACE PROCEDURE employer_details IS
    CURSOR emp_cur IS
SELECT first_name, last_name, salary FROM emp_tbl; emp_rec
emp_cur%rowtype;
BEGIN
FOR emp_rec in sales_cur LOOP
dbms_output.put_line(emp_cur.first_name || ' ' ||emp_cur.last_name || ' ' ||emp_cur.salary);
END LOOP;
END;
/
```

How to execute a Stored Procedure?

There are two ways to execute a procedure.

1) From the SQL prompt.

```
  EXECUTE [or EXEC] procedure_name;
```

2) Within another procedure – simply use the procedure name.

```
    procedure_name;
```

NOTE: In the examples given above, we are using backward slash _/‘ at the end of the program. This indicates the oracle engine that the PL/SQL program has ended and it can begin processing the statements.

Week 12 :

Cursors

In this week you need to do the following: Declare a cursor that defines a result set. Open the cursor to establish the result set. Fetch the data into local variables as needed from the cursor, one row at a time. Close the cursor when done.

```
CREATE PROCEDURE myProc (in_customer_id INT)
BEGIN
DECLARE v_id INT;
DECLARE v_name
varchar(30);
DECLARE c1 CURSOR FOR SELECT stdid, stdfirstname FROM Students
WHERE stdId=in_customer_id;
OPEN c1;
FETCH c1 into v_id,
v_name; Close c1;
END;
```

Cursors:

Every SQL statement executed by the Oracle server has an individual cursor associated with it and are called implicit cursors.
There are two types of cursors.

Implicit cursors: Declared for all DML and PL/SQL SELECT statements. **Explicit cursors**: Declared and names by the programmer.

Explicit Cursors:

- Individually process each row returned by a multiple row select statement.
- A PL/SQL program opens a cursor, processes rows returned by a query, and then closes the cursor. The cursor marks the current position in the active set.
  - Can process beyond the first row returned by the query, row by row.
  - Keep track of which row is currently being processed.
  - Allow the programmer to manually control explicit cursors in the PL/QL block.

Controlling Explicit Cursors:

- Declare the cursor by naming it and defining the structure of the query to be performed. Within it.
- Open the cursor: The OPEN statement executes the query and binds the variables that are referenced. Rows identified by the query are called the active set and are now available for fetching.

- Fetch data from the cursor: After each fetch, you test the cursor for any existing row. If there are no more rows to process, then you must close the cursor.
- Close the cursor: The CLOSE statement releases the active set of rows. It is now possible to reopen the cursor to establish a fresh active set

Syntax:

Declaring a cursor:

```
CURSOR cursor_name IS
    Select_statement;
```

Opening a cursor:

```
OPEN cursor_name;
```

Fetch data from a cursor:

FETCH cursor_name INTO [variable1, variable2,….]| record_name];

Closing a cursor:

```
Close cursor_name;
```

Example:

```
Set SERVEROUTPUT ON
DECLARE
V_empno employees.employee_id%TYPE;
V_ename employees.last_name%TYPE;
   CURSOR emp_cursor IS
        SELECT employee_id,
        last_name FROM employees;
BEGIN
   OPEN emp_cursor;
   FOR I IN 1..10 LOOP
     FETCH emp_cursor INTO v_empno, b_ename;
     DBMS_OUTPUT. PUT_LINE (TO_CHAR(v_empno) || ‗ ‗ || v_ename);
   END LOOP
   CLOSE emp_cursor;
```

Attributes of an Explicit Cursor:

- %ISOPEN [is cursor open]
- %NOTFOUND [is row not found]
- %FOUND [is row found]
- %ROWCOUNT [rows returned so far]

Cursors can be passed parameters. Cursors also have FOR UPDATE option which allows more fine grained control of locking at a table level. WHERE CURRENT OF can be used to apply the update or delete operation to current row in the cursor.

Logical Database design

Faculty Table:

```
create table faculty(
fid varchar2(12) check(fid like('cse%')
                              or
                              fid
                              like('ece%') or
                              fid
                              like('eee%') or
                              fid like('it%')),
fname char(10),
designation char(10) check(designation
                    in('asst','assoc','prof')),
branch char(5),
primary key(fid),
unique(fname));
```

Department Table:

```
create table dept( did
number(3), dname
char(10),
loc char(10), phno
number(10),
strength number(2),
nofac number(2),
hod char(10) references faculty(fname));
```

Subject allocation Table:

```
create table suballot(
year number check(year<5),
branch char(5) refernces
dept(dname), subname char(10),
Fname char(10) refernces faculty(fname));
```

Student personal information:

```
create table student(
sname char(10),
sid number(3),
branch char(5) references dept(dname),
year number(2) check(year<5),
gender char(2) check(gender
in('M','F')), dob date,
doj date,
phnum number(10),
emailid varchar2(15)
        check(emailid like('%@%'))
```

Student Personal Information:

```
create table sacademic(
sname char(10) references student(sname),
sid number(3) references student(sid),
branch char(5) references dept(dname),
year number(2) check(year<5),
percentage number(3) check((year=1
and percentage=null)
                or
        (YEAR=2 OR YEAR=3 OR
YEAR=4), SSC NUMBER(3),
INTER NUMBER(3),
eamcet NUMBER(5),
ATTEN NUMBER(3),
M1SUB1  NUMBER(2)  CHECK(M1SUB1<26),
M1SUB2  NUMBER(2)  CHECK(M1SUB2<26),
M1SUB3  NUMBER(2)  CHECK(M1SUB3<26),
M1SUB4  NUMBER(2)  CHECK(M1SUB4<26),
M1SUB5  NUMBER(2)  CHECK(M1SUB5<26),
M1SUB6  NUMBER(2)  CHECK(M1SUB6<26),
M2SUB1  NUMBER(2)  CHECK(M2SUB1<26),
M2SUB2  NUMBER(2)  CHECK(M2SUB2<26),
M2SUB3  NUMBER(2)  CHECK(M2SUB3<26),
M2SUB4  NUMBER(2)CHECK(M2SUB4<26)  ,
M2SUB5    NUMBER(2)CHECK(M2SUB5<26),
M2SUB6 NUMBER(2) CHECK(M2SUB6<26) );
```

Placement Information:

```
create table placement(
sid number(3) references student(sid),
sname char(10) references
student(sname), branch char(5) references
dept(dname), company char(15),
doselection date,
doj date);
```

Querying Relational Database

Some sample queries

1. Get the details of 1st year students.

   Select * from student where year=1;

2. Get the details of students whose SSC percentage is greater than

   70. Select * from student where ssc>70;

3. Get the details of students whose SSC and Inter percentage is greater than

   65. Select * from student where ssc>70 and inter>65;

4. Get the 1st mid details of all the students

   Select M1sub1,m1sub2,m1sub3,m1sub4,m1sub5,m1sub6 from sacademic;

5. Get the students whose EAMCET rank is available.

   Select * from sacademic where eamcet is not null;

6. Get the total number of students from four

   years. select count(*) from sacademic;

7. **Get the mail id‘s of 4th** year students

   Select emailid from student where year=4;

8. Display students branch wise.

   select sid,sname from sacademic where branch=any(select branch from sacademic group by branch)

9. Get the number of students from each branch year

   wise. select count(*) from sacademic group by branch;

10. Get the faculty name who is going to the subject DBMS.

    Select facname from suballoc where subname=‘dbms‘;

11. Get the total number of faculties from each department

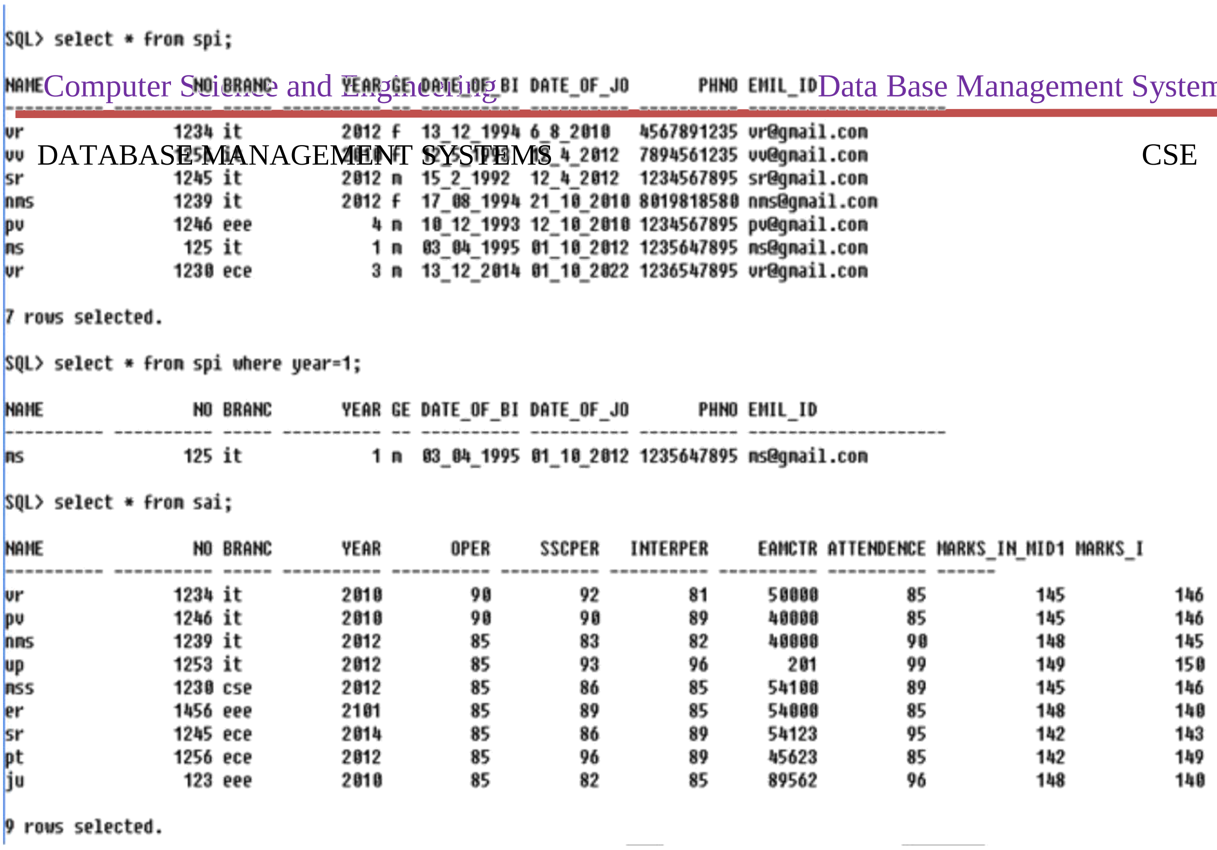
    Select count(*) from faculty group by branch;

12. Display Head of the department details.

    Select * from faculty where fname=any(select hod from dept group by branch);

13. Display faculty information branch wise.

    Select * from faculty group by branch;



```
SQL> select * fron spi;

NAME               NO BRANC       YEAR GE DATE_OF_BI DATE_OF_JO       PHNO EMIL_ID
---------- ---------- ----- ---------- -- ---------- ---------- ---------- --------------------
vr               1234 it          2012 f  13_12_1994 6_8_2010   4567891235 vr@gnail.con
vv         [illegible]_4_2012  7894561235 vv@gnail.con
sr               1245 it          2012 n  15_2_1992  12_4_2012  1234567895 sr@gnail.con
nms              1239 it          2012 f  17_08_1994 21_10_2010 8019818580 nms@gnail.con
pv               1246 eee            4 n  10_12_1993 12_10_2010 1234567895 pv@gnail.con
ns                125 it             1 n  03_04_1995 01_10_2012 1235647895 ns@gnail.con
vr               1230 ece            3 n  13_12_2014 01_10_2022 1236547895 vr@gnail.con

7 rows selected.

SQL> select * fron spi where year=1;

NAME               NO BRANC       YEAR GE DATE_OF_BI DATE_OF_JO       PHNO EMIL_ID
---------- ---------- ----- ---------- -- ---------- ---------- ---------- --------------------
ns                125 it             1 n  03_04_1995 01_10_2012 1235647895 ns@gnail.con

SQL> select * fron sai;

NAME               NO BRANC       YEAR       OPER     SSCPER   INTERPER     EAMCTR ATTENDENCE MARKS_IN_MID1 MARKS_I
---------- ---------- ----- ---------- ---------- ---------- ---------- ---------- ---------- ------
vr               1234 it          2010         90         92         81      50000         85           145          146
pv               1246 it          2010         90         90         89      40000         85           145          146
nms              1239 it          2012         85         83         82      40000         90           148          145
up               1253 it          2012         85         93         96        201         99           149          150
nss              1230 cse         2012         85         86         85      54100         89           145          146
er               1456 eee         2101         85         89         85      54000         85           148          140
sr               1245 ece         2014         85         86         89      54123         95           142          143
pt               1256 ece         2012         85         96         89      45623         85           142          149
ju                123 eee         2010         85         82         85      89562         96           148          140

9 rows selected.
```

14. Display Professors information from all branches.

    Select * from faculty where designation=‘prof‘;

15. Display Students information Placed branch wise.

    Select * from student where sname=any (select sname from placement group by branch);

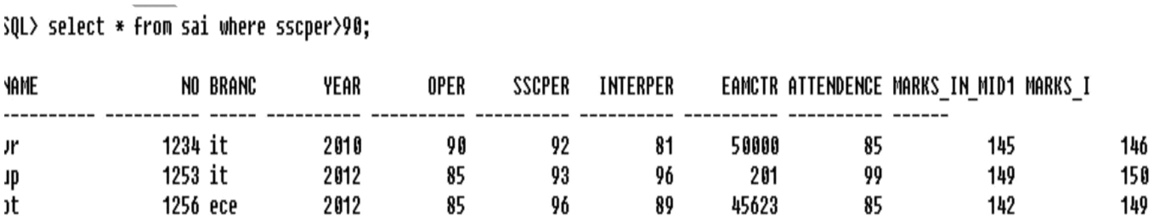
16. Display placements Information Company wise.

    Select company, count(*) from placement group by company;

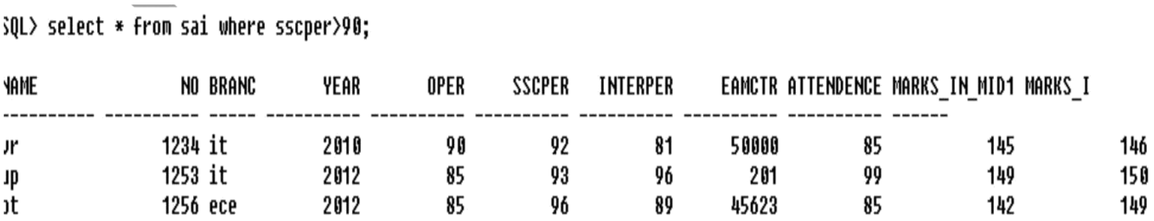
17. Find the student name that placed recently.

    Select sname from placement where doj=dos;

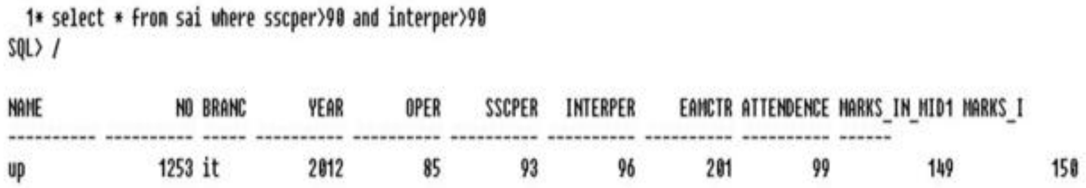
Sample Output Screens



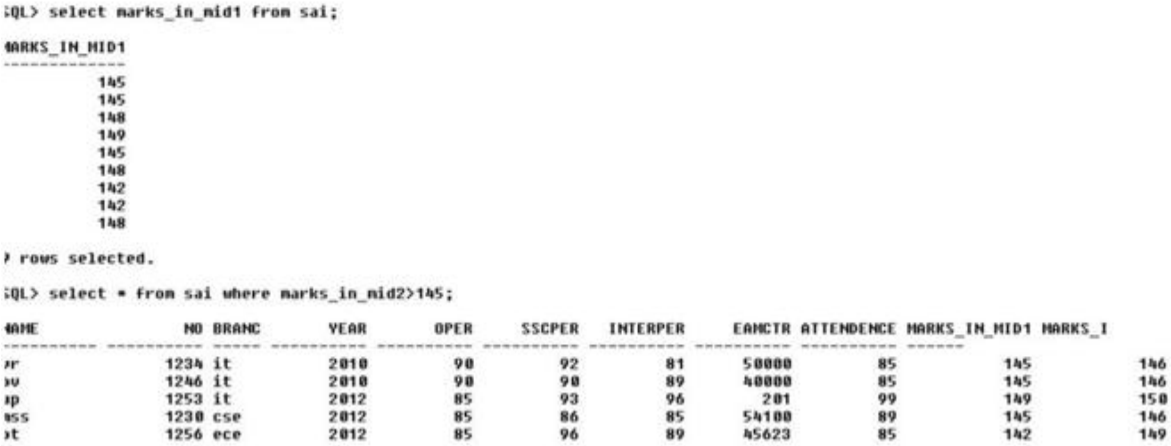
```
SQL> select * from sai where sscper>90;

NAME               NO BRANC       YEAR       OPER     SSCPER   INTERPER     EAMCTR ATTENDENCE MARKS_IN_MID1 MARKS_I
---------- ---------- ----- ---------- ---------- ---------- ---------- ---------- ---------- ------
ur               1234 it          2010         90         92         81      50000         85           145          146
up               1253 it          2012         85         93         96        201         99           149          150
ot               1256 ece         2012         85         96         89      45623         85           142          149
```



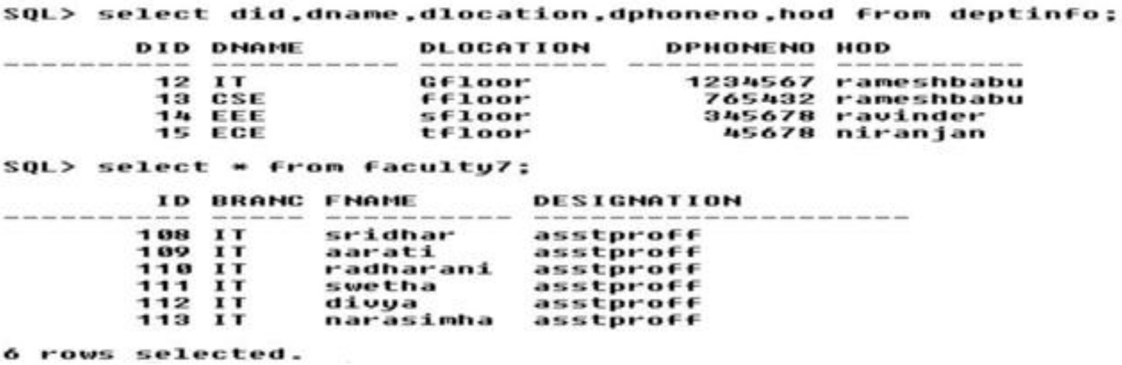
```
SQL> select * from sai where sscper>90;

NAME               NO BRANC       YEAR       OPER     SSCPER   INTERPER     EAMCTR ATTENDENCE MARKS_IN_MID1 MARKS_I
---------- ---------- ----- ---------- ---------- ---------- ---------- ---------- ---------- ------
ur               1234 it          2010         90         92         81      50000         85           145          146
up               1253 it          2012         85         93         96        201         99           149          150
ot               1256 ece         2012         85         96         89      45623         85           142          149
```



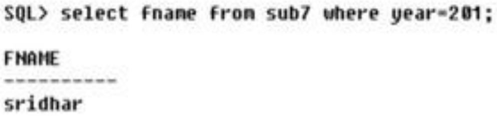
```
  1* select * from sai where sscper>90 and interper>90
SQL> /

NAME               NO BRANC       YEAR       OPER     SSCPER   INTERPER     EAMCTR ATTENDENCE MARKS_IN_MID1 MARKS_I
---------- ---------- ----- ---------- ---------- ---------- ---------- ---------- ---------- ------
up               1253 it          2012         85         93         96        201         99           149          150
```



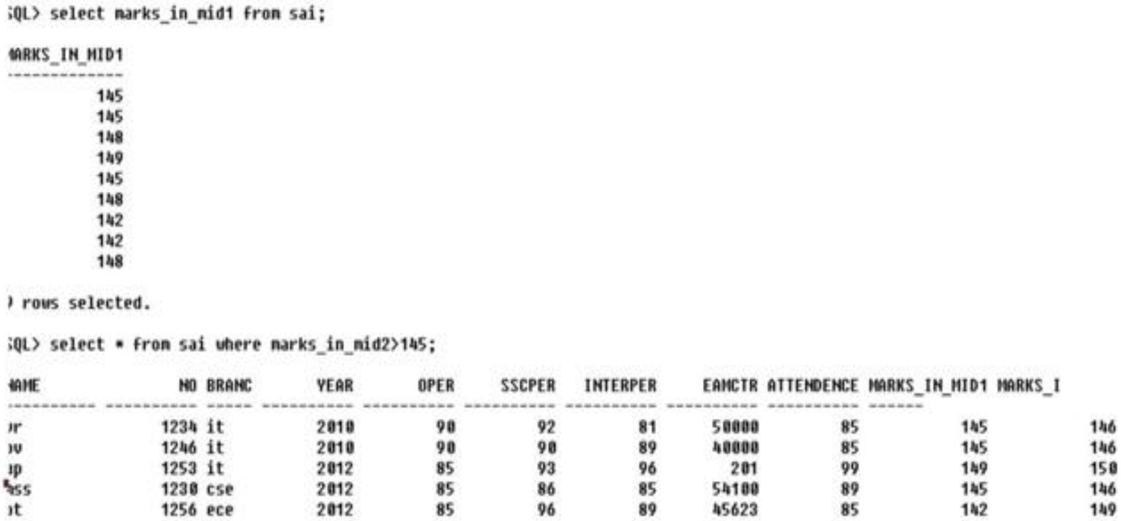
```
;QL> select marks_in_mid1 from sai;

1ARKS_IN_MID1
-------------
          145
          145
          148
          149
          145
          148
          142
          142
          148

 rows selected.

;QL> select * from sai where marks_in_mid2>145;

1AME               NO BRANC       YEAR       OPER     SSCPER   INTERPER     EAMCTR ATTENDENCE MARKS_IN_MID1 MARKS_I
---------- ---------- ----- ---------- ---------- ---------- ---------- ---------- ---------- ------
/r               1234 it          2010         90         92         81      50000         85           145          146
ɔv               1246 it          2010         90         90         89      40000         85           145          146
ıp               1253 it          2012         85         93         96        201         99           149          150
ıss              1230 cse         2012         85         86         85      54100         89           145          146
ɔt               1256 ece         2012         85         96         89      45623         85           142          149
```



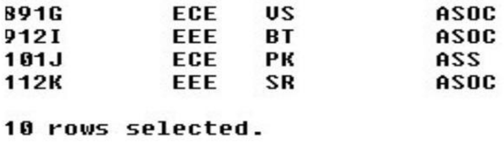
```
SQL> select did,dname,dlocation,dphoneno,hod from deptinfo;

       DID DNAME           DLOCATION      DPHONENO HOD
---------- ---------- ---------- ---------- ----------
        12 IT              GFloor          1234567 rameshbabu
        13 CSE             FFloor           765432 rameshbabu
        14 EEE             sFloor           345678 ravinder
        15 ECE             tFloor            45678 niranjan

SQL> select * from faculty7;

        ID BRANC FNAME      DESIGNATION
---------- ----- ---------- --------------------
       108 IT    sridhar    asstproff
       109 IT    aarati     asstproff
       110 IT    radharani  asstproff
       111 IT    swetha     asstproff
       112 IT    divya      asstproff
       113 IT    narasimha  asstproff

6 rows selected.
```



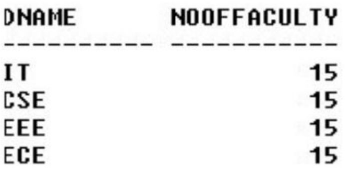
```
SQL> select fname from sub7 where year=201;

FNAME
----------
sridhar
```



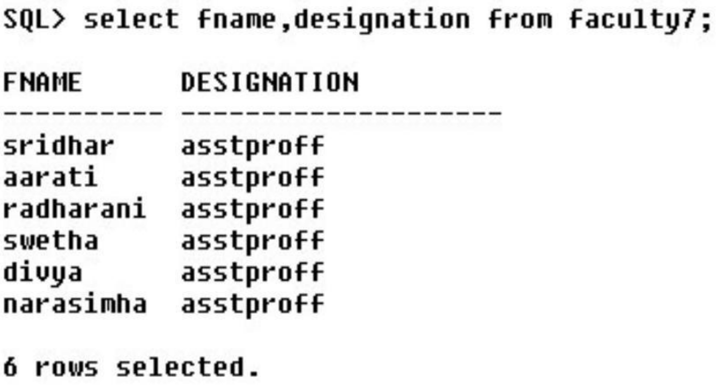
```
;QL> select marks_in_mid1 from sai;

1ARKS_IN_MID1
-------------
          145
          145
          148
          149
          145
          148
          142
          142
          148

 rows selected.

;QL> select * from sai where marks_in_mid2>145;

1AME               NO BRANC       YEAR       OPER     SSCPER   INTERPER     EAMCTR ATTENDENCE MARKS_IN_MID1 MARKS_I
---------- ---------- ----- ---------- ---------- ---------- ---------- ---------- ---------- ------
/r               1234 it          2010         90         92         81      50000         85           145          146
ɔv               1246 it          2010         90         90         89      40000         85           145          146
ıp               1253 it          2012         85         93         96        201         99           149          150
ıss              1230 cse         2012         85         86         85      54100         89           145          146
ɔt               1256 ece         2012         85         96         89      45623         85           142          149
```



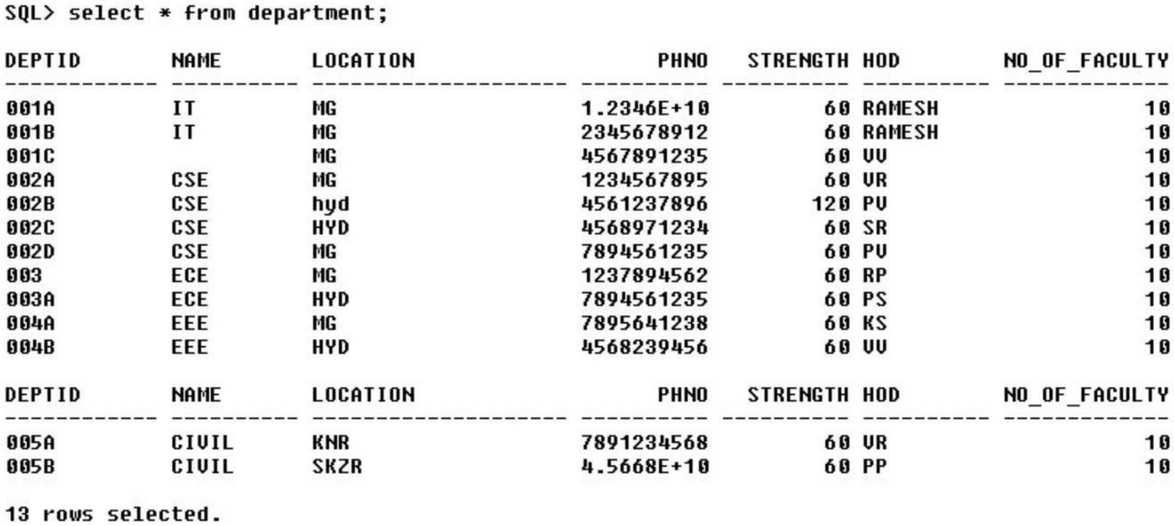
```
891G       ECE    VS           ASOC
912I       EEE    BT           ASOC
101J       ECE    PK           ASS
112K       EEE    SR           ASOC

10 rows selected.
```



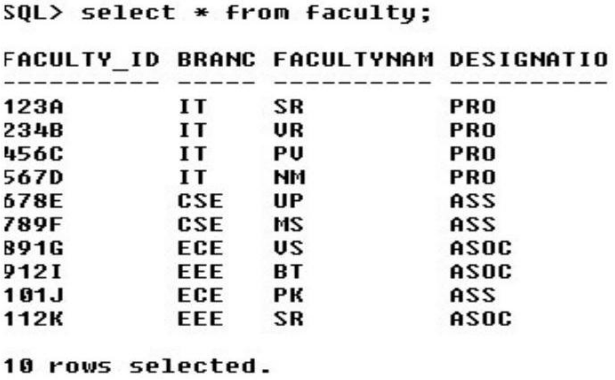
```
DNAME      NOOFFACULTY
---------- -----------
IT                  15
CSE                 15
EEE                 15
ECE                 15
```



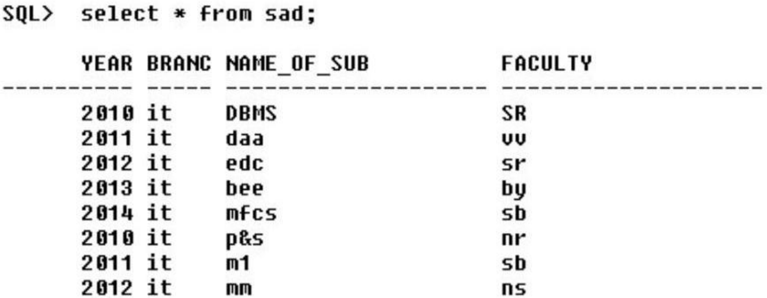
```
SQL> select fname,designation from faculty7;

FNAME      DESIGNATION
---------- --------------------
sridhar    asstproff
aarati     asstproff
radharani  asstproff
swetha     asstproff
divya      asstproff
narasimha  asstproff

6 rows selected.
```



```
SQL> select * from department;

DEPTID       NAME       LOCATION                   PHNO   STRENGTH HOD        NO_OF_FACULTY
------------ ---------- -------------------- ---------- ---------- ---------- -------------
001A         IT         MG                   1.2346E+10         60 RAMESH                10
001B         IT         MG                   2345678912         60 RAMESH                10
001C                    MG                   4567891235         60 VV                    10
002A         CSE        MG                   1234567895         60 VR                    10
002B         CSE        hyd                  4561237896        120 PV                    10
002C         CSE        HYD                  4568971234         60 SR                    10
002D         CSE        MG                   7894561235         60 PV                    10
003          ECE        MG                   1237894562         60 RP                    10
003A         ECE        HYD                  7894561235         60 PS                    10
004A         EEE        MG                   7895641238         60 KS                    10
004B         EEE        HYD                  4568239456         60 VV                    10

DEPTID       NAME       LOCATION                   PHNO   STRENGTH HOD        NO_OF_FACULTY
------------ ---------- -------------------- ---------- ---------- ---------- -------------
005A         CIVIL      KNR                  7891234568         60 VR                    10
005B         CIVIL      SKZR                 4.5668E+10         60 PP                    10

13 rows selected.
```



```
SQL> select * from faculty;

FACULTY_ID BRANC FACULTYNAM DESIGNATIO
---------- ----- ---------- ----------
123A       IT    SR         PRO
234B       IT    VR         PRO
456C       IT    PV         PRO
567D       IT    NM         PRO
678E       CSE   UP         ASS
789F       CSE   MS         ASS
891G       ECE   VS         ASOC
912I       EEE   BT         ASOC
101J       ECE   PK         ASS
112K       EEE   SR         ASOC

10 rows selected.
```



```
SQL>  select * from sad;

      YEAR BRANC NAME_OF_SUB          FACULTY
---------- ----- -------------------- --------------------
      2010 it    DBMS                 SR
      2011 it    daa                  vv
      2012 it    edc                  sr
      2013 it    bee                  by
      2014 it    mfcs                 sb
      2010 it    p&s                  nr
      2011 it    m1                   sb
      2012 it    mm                   ns
```

Additional programs

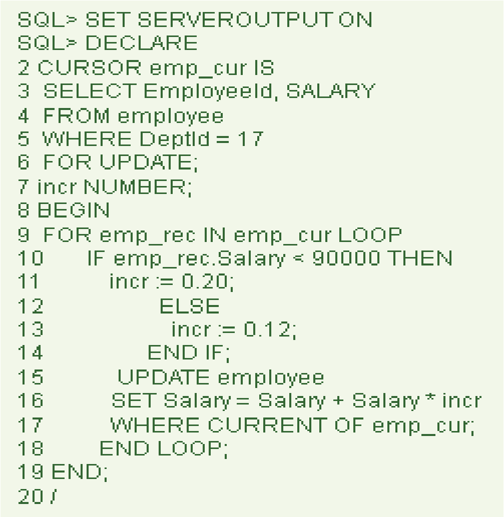
cursors

1)Retrieving the records from the emp table and displaying them one by one using cursors:

```
CREATE OR REPLACE PROCEDURE my_proc IS
var_empno emp.empno%type;
var_ename emp.ename%type;
var_sal emp.sal%type;

//declaring a cursor//
CURSOR EMP_CURSOR IS
select empno, ename, sal from emp;
BEGIN

//opening a cursor//
open EMP_CURSOR;
LOOP

//fetching records from a cursor//
fetch EMP_CURSOR into var_empno, var_ename, var_sal;

//testing exit conditions//
EXIT when EMP_CURSOR%NOTFOUND;
IF (var_sal > 1000) then
DBMS_OUTPUT.put_line(var_empno || ' ' || var_ename || ' ' || var_sal);
ELSE
DBMS_OUTPUT.put_line(var_ename || ' sal is less then 1000');
END IF;
END LOOP;

//closing the cursor//
close EMP_CURSOR;
DBMS_OUTPUT.put_line('DONE');

END;
```

2)
Create a PL/SQL block to increase salary of employees in department 17.

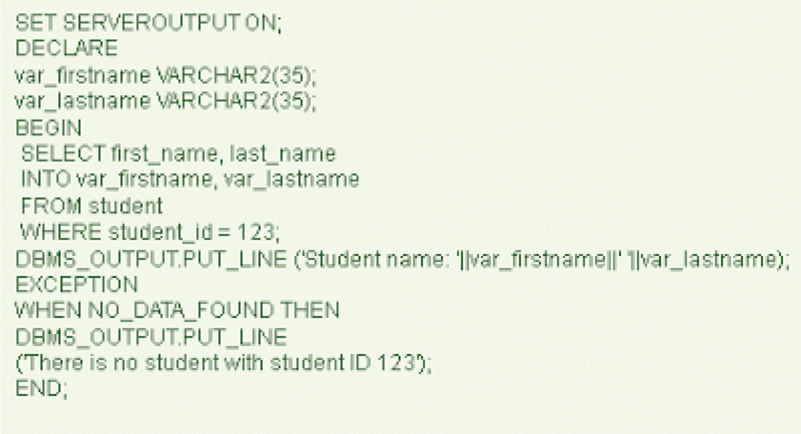
a) The salary increase is 20% for the employees making less than $90,000 and 12% for the employees making $90,000 or more.
b) Use a cursor with a FOR UPDATE clause.
c) Update the salary with a WHERE CURRENT OF clause in a cursor FOR loop (cursor FOR loop problem).



```
SQL> SET SERVEROUTPUT ON
SQL> DECLARE
2 CURSOR emp_cur IS
3  SELECT EmployeeId, SALARY
4  FROM employee
5  WHERE DeptId = 17
6  FOR UPDATE;
7 incr NUMBER;
8 BEGIN
9  FOR emp_rec IN emp_cur LOOP
10      IF emp_rec.Salary < 90000 THEN
11         incr := 0.20;
12                ELSE
13                  incr := 0.12;
14              END IF;
15         UPDATE employee
16         SET Salary = Salary + Salary * incr
17         WHERE CURRENT OF emp_cur;
18       END LOOP;
19 END;
20 /
```

3.

An example of an Implicit cursor



```
SET SERVEROUTPUT ON;
DECLARE
var_firstname VARCHAR2(35);
var_lastname VARCHAR2(35);
BEGIN
 SELECT first_name, last_name
 INTO var_firstname, var_lastname
 FROM student
 WHERE student_id = 123;
DBMS_OUTPUT.PUT_LINE ('Student name: '||var_firstname||' '||var_lastname);
EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE
('There is no student with student ID 123');
END;
```

4) **IMPLICIT CURSOR**

```
SQL> select * from bank;
CNAME ACCNO AMT
------------- ---------- ----------
ram 1001 3000r
anjan 1002 3500
gayathri 2002 3900
```

MAIN PROGRAM

```
:Declare
M bank.amt%type
;E number(2):=0;
Name bank.cname%type;
Cursor c is select accno from bank;
Begin
For
I in c
Loop
Update bank set amt=amt-100 where accno=i.accno;
Exit when sql%notfound;
Select cname,amt into name,m from bank where accno=i.accno;
Dbms_output.put_line('name is:'||name);
Dbms_output.put_line('changed amount is:'||name);
E:=e+to_char(sql%rowcount);
End loop;
Dbms_output.put_line('the total rows selected is:'||e);
End;
```

5.Additional programs in ER Diagrams:

Problem 1: Analyze the problem carefully and come up with the entities in it. Identify what data has to be persisted in the database. This contains the entities, attributes etc. Identify the primary keys for all the entities. Identify the other keys like candidate keys, partial keys, if any for **student database**

Solution:

Identifying Entities:

- Student(student personal information like first name,lastname et.c)
- Department(dept id,name)
- Student course(semester,subject code,subj name)
- Student score

Identifying key Attributes:

- 1.Student-id
- 2.dept-id
- 3.course-id

A part from the above mentioned entities you can identify more. The above mentioned are few.

Other Attributes

Student entity attributes:

Studentidno, FirstName,lastName, DOB,age,stadr,city,state,phno,emailid

Department entity attributes:

Deptid,department name

Student course entity attributes:

Deptid,Subjcode,subjname,semester,year,credits

Student score entity attributes:

Studentid, semester,year,subjcode,marksscored,credits

Problem 2: Analyze the problem carefully and come up with the entities in it. Identify what data has to be persisted in the database. This contains the entities, attributes etc. Identify the primary keys for all the entities. Identify the other keys like candidate keys, partial keys, if any for **Library management system**

Solution:

A library consists of a section, a member, a book, a granter, a publisher.
Section has section id, name and phone number
Member has member id, address, telephone, occupation, member name.
Book has call number, title, author, price.
Publisher has publisher id, name, address, phone number.
Granter has national identify card number, name, address, phone.
Member name can be divided into first name, middle name, last name.
The section, member, book, granter, publisher are uniquely identified by section id, member id, call number, publisher id, national id card number respectively.
One section has many books but one book should keep in one section.
One member can borrow many books.

Many books may publish by one publisher otherwise one publisher may be published many books

IDENTIFYING THE ENTITIES

The entities are
SECTION
MEMBER
BOOK
PUBLISHER
GRANTER

IDENTIFYING KEY ATTRIBUTES

SECTION SECTION ID
MEMBER MEMBER ID
BOOK CALL NUMBER
PUBLISHER PUBLISHER ID
GRANTER NATIONAL IDENTIFY CARD NUMBER

OTHER ATTRIBUTES

SECTION NAME, PH NO
MEMBER ADDRESS, TELEPHONE, OCCUPATION, MEMBER NAME
PUBLISHER NAME, ADDRESS, PHONE NUMBER
BOOK TITLE, AUTHOR, PRICE
GRANTER NAME, ADDRESS, PHONE NUMBER, POST

Problem 3: Analyze the problem carefully and come up with the entities in it. Identify what data has to be persisted in the database. This contains the entities, attributes etc. Identify the primary keys for all the entities. Identify the other keys like candidate keys, partial keys, if any for **ONLINE BOOKSTORE**

Problem : Consider the database of an online bookstore.

• Every book has a title, isbn, year and price. The store also keeps the author and publisher for any book.

• For authors, the database keeps the name, address and the url of their homepage.
For publishers, the database keeps the name, address, phone number and the url of their website.

• The store has several warehouses, each of which has a code, address and phone number.

• The warehouse stocks several books. A book may be stocked at multiple warehouses. (In previous sentence, we are not referring to a particular copy of the book. Consider for example "the complete book" for our course. This book may be stocked at multiple warehouses.)

• The database records the number of copies of a book stocked at various warehouses.

• The bookstore keeps the name, address, email-id, and phone number of its customers.

• A customer owns several shopping basket. A shopping basket is identified by a basketID and contains several books.• Some shopping baskets may contain more than one copy of same book. The database records the number of copies of each book in any shopping basket.

Solution:

Identify all the entities

- -AUTHOR
- -PUBLISHER
- -BOOK
- -CUSTOMER
- -SHOPPING_BASKET

- -WAREHOUSE

Identify the key attribute

- *AUTHOR- name
- *PUBLISHER- name
- *BOOK- ISBN
- *CUSTOMER- email
- *SHOPPING_BASKET- basket_ID
- *WAREHOUSE- code

Identify other relevant attributes

- *AUTHOR- name,address,URL
- *PUBLISHER- name,address,URL,phone
- *BOOK- ISBN,year,title,price
- *CUSTOMER- email, name,address,phone
- *SHOPPING_BASKET- basket_ID
- *WAREHOUSE- code, address,phone

Problem 4: Analyze the problem carefully and come up with the entities in it. Identify what data has to be persisted in the database. This contains the entities, attributes etc. Identify the primary keys for all the entities. Identify the other keys like candidate keys, partial keys, if any for **HOSPITALMANAGEMENT SYSTEM**
SOLUTION:

Identify all the entities

- **Hospital**
- **Patient**
- **Medical Record**
- **Doctor**

Identify the key attribute

- Hosp-id
- Pat-id
- Record-id
- Doc-id

Identify other relevant attributes

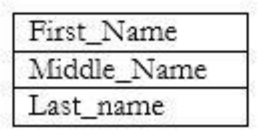
- HCity, HAddress, Hos-Name, Pat-id(Foreign key references to Pat-id of Patient table), Doc-id(Foreign key references to Doc-id of Doctor table)
- PName, PAddress, PDiagnosis, Record-id(Foreign key references to Record-id of Medical Record table), Hosp-id(Foreign key references to Hosp-id of Hospital table)
- Problem, Date_of_examination, Pat-id(Foreign key references to Pat-id of Patient table)
- DName, Qualification, Salary, Hosp-id(Foreign key references to Hosp-id of Hospital table)

Composite Attributes

Composite Attributes which can be divided into subparts.

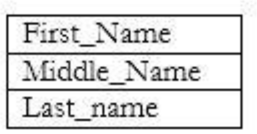
Example: Patient Name, Doctor Name

Patient



First_Name
Middle_Name
Last_name

Doctor



First_Name
Middle_Name
Last_name

VIVA QUESTIONS

1. Explain entity, relation and attributes?
2. What is the difference between weak entity and strong entity?
3. What are the different types of attributes?
4. Explain difference between multi valued and single valued attributes?
5. What is the difference between entity and entity set?

6.PL/SQL PROGRAMS

1. To Write a PL/SQI program to print empno of an employee as well as the corresponding Manager No using various loops

Solution:

```
declare
        e1 emp.empno%type;
        e2 emp.mgr%type;
    begin
        e1:=&empno;
        select mgr into e2 from emp where empno=e1;
        dbms_output.put_line('empno:'||e1 ||'mgr:'||e2);
    exception
        when no_data_found then
        dbms_output.put_line('wrong input');
   end;
```

2. Write a PL/SQL program to update salary of an employee for whom increments are sanctioned. Also record such updations in a log table with entries {empno, ename, old sal, new sal, date of updations}. Using cursors

Solution:

```
declare
                        cursor c1 is
                              selct * from loge where empno1=e1;
                              e1 emp.empno%type;
                              ena emp.ename%type;
                              old_sal emp.sal%type;
                              new_sal emp.sal%type;
                              inc  number(10);
               begin
                       e1:=&no;
                       inc:=&inc;
                select ename,sal into ena,old_sal from emp where empno=e1;
                new_sal:=old_sal+inc;
                insert into loge values(e1,ena,old_sal,new_sal);
                open c1;
                        loop
                         fetch c1 into e2,ena1,old,new;
                         exit when c1%notfound;
                         dbms_output.put_line(e1||' '||ena1||' '||old||' '||new);
                 close c1;
                end;
```

3. Create a Procedure to accept an Empno, and a salary increase amount, if Empno is not found or current salary is NULL then raise exceptions otherwise display total salary. Using procedures

Solution:

```
create or replace procedure empis(e1 emp.empno%type,in1 emp.sal%type) is
                     s1 emp.sal%type;
                    nusal exception;
                    nsal emp.sal%type;
            begin
                    select sal into s1 from emp where empno=e1;
            if s1 is null then
                     raise nusal;
          else
                     nsal:=s1+in1;
                    dbms_output.put_line(nsal);
          end if;
         exception
                     when nusal then
                              dbms_output.put_line('given emp. sal is null');
                    when no_data_found then
                              dbms_output.put_line('wrong empno');
         end empis;
```