Assignment No. 1

Title: Design any database with at least 3 entities and relationships between them. Apply DCL and DDL commands. Draw suitable ER/EER diagram for the system.

Aim: Design any database with at least 3 entities and relationships between them. Apply DCL and DDL commands. Draw suitable ER/EER diagram for the system.

Objective: To understand DCL, DDL commands, and ER/EER diagram for the system.

Theory:

1 Introduction to SQL:

The Structured Query Language (SQL) comprises one of the fundamental building blocks of modern database architecture. SQL defines the methods used to create and manipulate relational databases on all major platforms.

SQL comes in many flavors. Oracle databases utilize their proprietary PL/SQL. Microsoft SQL Server makes use of Transact-SQL. However, all of these variations are based upon the industry standard ANSI SQL.

SQL commands can be divided into two main sublanguages.

1. Data Definition Language
2. Data Manipulation Language

1.2 DATA DEFINITION LANGUAGE (DDL)

It contains the commands used to create and destroy databases and database objects. These commands will primarily be used by database administrators during the setup and removal phases of a database project.

DDL Commands:

a) Create table command :

Syntax

```
CREATE TABLE table_name
(
column_name1 data_type(size),
column_name2 data_type(size),
.....
)
```

Example 1

This example demonstrates how you can create a table named "Person", with four columns. The column names will be "LastName", "FirstName", "Address", and "Age":

```
CREATE TABLE Person
( LastName varchar,
  FirstName varchar,
  Address varchar,
  Age int  )
```

This example demonstrates how you can specify a maximum length for some columns:

Example 2

```
CREATE TABLE Person
(
LastName varchar(30),
FirstName varchar,
Address varchar,
Age int(3)
)
```

Creating table from another (existing table) table:

Syntax

```
CREATE TABLE tablename
[(columnname,columnaname)]]
AS SELECT columnname,columnaname
FROM tablename;
```

b. Alter table command:

Once table is created within a database, we may wish to modify the definition of that table.The ALTER command allows to make changes to the structure of a table without deleting and recreating it.

Let's begin with creation of a table called **testalter_tbl.**

```
mysql> create table testalter_tbl
    -> (
    -> i INT,
    -> c CHAR(1)
    -> );
Query OK, 0 rows affected (0.05 sec)
mysql> SHOW COLUMNS FROM testalter_tbl;
```

```
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| i     | int(11) | YES  |     | NULL    |       |
| c     | char(1) | YES  |     | NULL    |       |
+-------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Dropping, Adding or Repositioning a Column:

Suppose you want to drop an existing column **i** from above MySQL table then you will use **DROP** clause along with **ALTER** command as follows:

```
mysql> ALTER TABLE testalter_tbl DROP i;
```

A **DROP** will not work if the column is the only one left in the table.

To add a column, use ADD and specify the column definition. The following statement restores the **i**column to testalter_tbl:

```
mysql> ALTER TABLE testalter_tbl ADD i INT;
```

After issuing this statement, testalter will contain the same two columns that it had when you first created the table, but will not have quite the same structure. That's because new columns are added to the end of the table by default. So even though **i** originally was the first column in mytbl, now it is the last one.

```
mysql> SHOW COLUMNS FROM testalter_tbl;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| c     | char(1) | YES  |     | NULL    |       |
| i     | int(11) | YES  |     | NULL    |       |
+-------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

To indicate that you want a column at a specific position within the table, either use FIRST to make it the first column or AFTER col_name to indicate that the new column should be placed after col_name. Try the following ALTER TABLE statements, using SHOW COLUMNS after each one to see what effect each one has:

```
ALTER TABLE testalter_tbl DROP i;
ALTER TABLE testalter_tbl ADD i INT FIRST;
ALTER TABLE testalter_tbl DROP i;
ALTER TABLE testalter_tbl ADD i INT AFTER c;
```

The FIRST and AFTER specifiers work only with the ADD clause. This means that if you want to reposition an existing column within a table, you first must DROP it and then ADD it at the new position.

Changing a Column Definition or Name:

To change a column's definition, use **MODIFY** or **CHANGE** clause along with ALTER command. For example, to change column **c** from CHAR(1) to CHAR(10), do this:

```
mysql> ALTER TABLE testalter_tbl MODIFY c CHAR(10);
```

With CHANGE, the syntax is a bit different. After the CHANGE keyword, you name the column you want to change, then specify the new definition, which includes the new name. Try out the following example:

```
mysql> ALTER TABLE testalter_tbl CHANGE i j BIGINT;
```

If you now use CHANGE to convert j from BIGINT back to INT without changing the column name, the statement will be as expected:

```
mysql> ALTER TABLE testalter_tbl CHANGE j j INT;
```

Changing a Column's Default Value:

You can change a default value for any column using ALTER command. Try out the following example.

```
mysql> ALTER TABLE testalter_tbl ALTER i SET DEFAULT 1000;
mysql> SHOW COLUMNS FROM testalter_tbl;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| c     | char(1) | YES  |     | NULL    |       |
| i     | int(11) | YES  |     | 1000    |       |
+-------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Renaming a Table:

To rename a table, use the **RENAME** option of the ALTER TABLE statement. Try out the following example to rename testalter_tbl to alter_tbl.

```
mysql> ALTER TABLE testalter_tbl RENAME TO alter_tbl;
```

c. Drop table command:

DROP command allows us to remove entire database objects from our DBMS. For example, if we want to permanently remove the personal_info table that we created, we'd use the following command:

Syntax

DROP TABLE table_name;

Example

```
DROP TABLE personal_info;
```

DATA INTEGRITY:

Enforcing data integrity ensures the quality of the data in the database. For example, if an employee is entered with an **employee_id** value of "123", the database should not allow another employee to have an ID with the same value.

Two important steps in planning tables are to identify valid values for a column and to decide how to enforce the integrity of the data in the column. Data integrity falls into four categories:

- Entity integrity
- Domain integrity
- Referential integrity
- User-defined integrity

There are several ways of enforcing each type of integrity.

Integrity type	Recommended options
Entity	PRIMARY KEY constraint UNIQUE constraint
Domain	FOREIGN KEY constraint CHECK constraint NOT NULL
Referential	FOREIGN KEY constraint CHECK constraint
User-defined	All column- and table-level constraints in CREATE TABLE StoredProcedures Triggers

ENTITY INTEGRITY:

Entity integrity defines a row as a unique entity for a particular table. Entity integrity enforces the integrity of the identifier column(s) or the primary key of a table (through indexes, UNIQUE constraints, PRIMARY KEY constraints, or IDENTITY properties).

DOMAIN INTEGRITY:

Domain integrity is the validity of entries for a given column. You can enforce domain integrity by restricting the type (through data types), the format (through CHECK constraints and rules), or the range of possible values (through FOREIGN KEY constraints, CHECK constraints, DEFAULT definitions, NOT NULL definitions, and rules).

REFERENTIAL INTEGRITY:

Referential integrity preserves the defined relationships between tables when records are entered or deleted. In Microsoft® SQL Server™, referential integrity is based on relationships between foreign keys and primary keys or between foreign keys and unique keys. Referential integrity ensures that key values are consistent across tables. Such consistency requires that there be no references to nonexistent values and that if a key value changes, all references to it change consistently throughout the database.

a. PRIMARY KEY CONSTRAINT:

Definition:- The primary key of a relational table uniquely identifies each record in the table.

A primary key constraint ensures no duplicate values are entered in particular columns and that NULL values are not entered in those columns.

b. NOT NULL CONSTRAINT:

This constraint ensures that NULL values are not entered in those columns.

c. UNIQUE CONSTRAINT:

This constraint ensures that no duplicate values are entered in those columns.

d. CHECK CONSTRAINT:

The CHECK constraint enforces column value restrictions. Such constraints can restrict a column, for example, to a set of values, only positive numbers, or reasonable dates.

e. FOREIGN KEY CONSTRAINT:

Foreign keys constrain data based on columns in other tables. They are called *foreign keys* because the constraints are foreign--that is, outside the table. For example, suppose a table contains customer addresses, and part of each address is a United States two-character state code. If a table held all valid state codes, a foreign key constraint could be created to prevent a user from entering invalid state codes.

To create a table with different types of constraints:

Syntax

```
CREATE TABLE table_name
(
column_name1 data_type [constraint],
column_name2 data_type [constraint],
.....
)
```

Example

```
Create table customer
( customer-name char(20) not null,
customer-street char(30),
customer-city char(30),
primary key ( customer-name));
```

```
create table branch
( branch-name char(15) not null,
branch-city char(30),
assets number,
primary key ( branch-name));
```

```
create table account
( branch-name char(15),
account-number char(10) not null,
balance number,
primary key ( account-number),
foreign key ( branch-name) references branch,
check (balance>500));
```

```
create table depositor
( customer-name char(20) not null,
account-number char(10) not null,
primary key ( customer-name,account-number),
foreign key ( account-number) references account,
foreign key ( customer-name) references customer);
```

MySQL CREATE INDEX

MySQL, index can be created on a table when the table is created with CREATE TABLE command. Otherwise, CREATE INDEX enables to add indexes to existing tables. A multiple-column index can be created using multiple columns.

The indexes are formed by concatenating the values of the given columns.

CREATE INDEX cannot be used to create a PRIMARY KEY.

Syntax:-

Syntax CREATE INDEX [index name] ON [table name] ([column name]);

Arguments

Name	Description
index name	Name of the index.
table name	Name of the Table
column name	Name of the column.

Example

CREATE INDEX autid ON newauthor(aut_id);

The above MySQL statement will create an INDEX on 'aut_id' column for 'newauthor' table.

MySQL Create UNIQUE INDEX

Using CREATE UNIQUE INDEX, you can create an unique index in MySQL.

CREATE UNIQUE INDEX newautid ON newauthor(aut_id);

The above MySQL statement will create an UNIQUE INDEX on 'aut_id' column for 'newauthor' table.

My SQL Sequence

A sequence is a set of integers 1, 2, 3, that are generated in order on demand. Sequences are frequently used in databases because many applications require each row in a table to contain a unique value and sequences provide an easy way to generate them.

Using AUTO_INCREMENT column:

The simplest way in MySQL to use Sequences is to define a column as AUTO_INCREMENT and leave rest of the things to MySQL to take care.

Example:

Try out the following example. This will create table and after that it will insert few rows in this table where it is not required to give record ID because it's auto incremented by MySQL.

```
mysql> CREATE TABLE insect
    -> (
    -> id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    -> PRIMARY KEY (id),
    -> name VARCHAR(30) NOT NULL, # type of insect
    -> date DATE NOT NULL, # date collected
    -> origin VARCHAR(30) NOT NULL # where collected
);
Query OK, 0 rows affected (0.02 sec)
mysql> INSERT INTO insect (id,name,date,origin) VALUES
    -> (NULL,'housefly','2001-09-10','kitchen'),
    -> (NULL,'millipede','2001-09-10','driveway'),
    -> (NULL,'grasshopper','2001-09-10','front yard');
Query OK, 3 rows affected (0.02 sec)
Records: 3  Duplicates: 0  Warnings: 0
mysql> SELECT * FROM insect ORDER BY id;
+----+-------------+------------+------------+
| id | name        | date       | origin     |
+----+-------------+------------+------------+
|  1 | housefly    | 2001-09-10 | kitchen    |
|  2 | millipede   | 2001-09-10 | driveway   |
|  3 | grasshopper | 2001-09-10 | front yard |
+----+-------------+------------+------------+
```

My SQL Synonym

A *synonym* is merely another name for a table or a view. Synonyms are usually created so that a user can avoid having to qualify another user's table or view to access the table or view. Synonyms can be created as PUBLIC or PRIVATE. A PUBLIC synonym can be used by any user of the database; a PRIVATE synonym can be used only by the owner and any users that have been granted privileges.

Creating Synonyms

The general syntax to create a synonym is as follows:

CREATE [PUBLIC|PRIVATE] SYNONYM *SYNONYM_NAME* FOR TABLE|VIEW

You create a synonym called CUST, short for CUSTOMER_TBL, in the following example. This frees you from having to spell out the full table name.

CREATE SYNONYM CUST FOR CUSTOMER_TBL;

SELECT CUST_NAME FROM CUST;

CUST_NAME

LESLIE GLEASON

NANCY BUNKER

ANGELA DOBKO

WENDY WOLF

MARYS GIFT SHOP

Dropping Synonyms

Dropping synonyms is like dropping most any other database object. The general syntax to drop a synonym is as follows:

DROP [PUBLIC|PRIVATE] SYNONYM *SYNONYM_NAME*

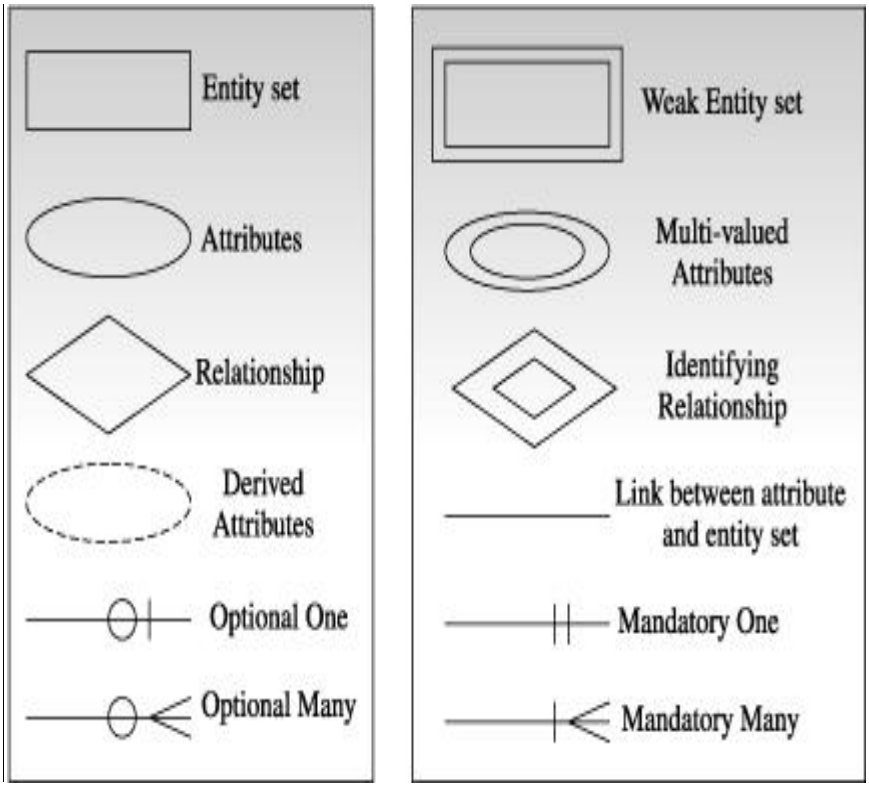
DROP SYNONYM CUST;

Entity Relationship (ER) and Extended Entity Relationship (EER) Diagram

Entity Relationship (ER) Diagram

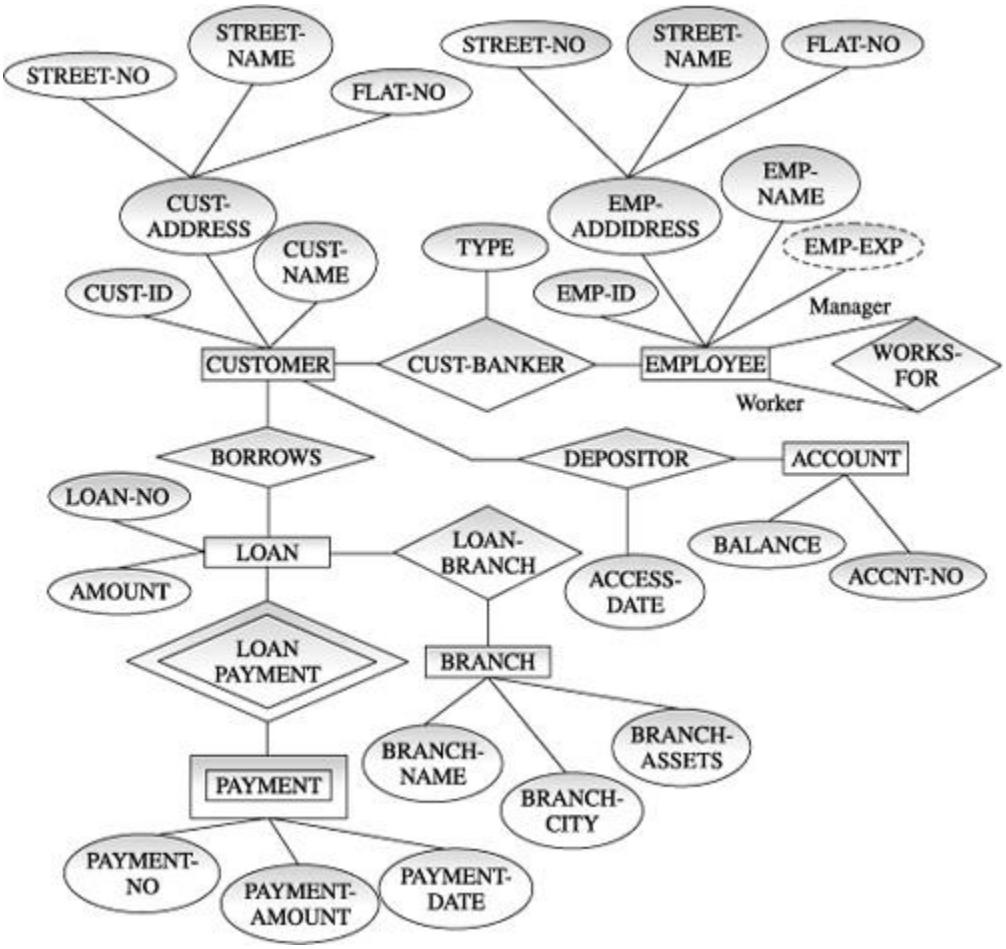
An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is a component of data. In other words, ER diagrams illustrate the logical structure of databases. ER-Diagram is a visual representation of data that describes how data is related to each other.

Basic Building Blocks of ER Diagram



Sample ER Diagram

Complete E-R diagram of banking organization database

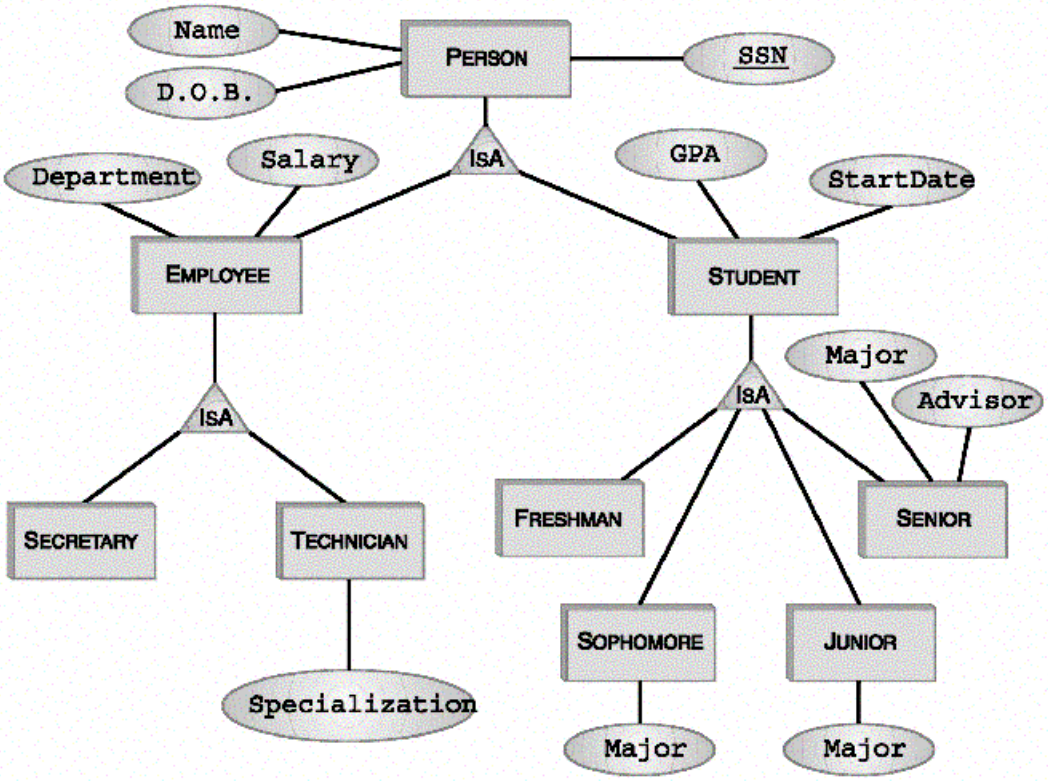


Extended Entity Relationship

The **enhanced entity–relationship** (**EER**) model (or **extended entity–relationship** model) in computer science is a high-level or conceptual data model incorporating extensions to the original entity–relationship (ER) model, used in the design of databases.

The Extended Entity-Relationship Model is a more complex and high-level model that extends an E-R diagram to include more types of abstraction, and to more clearly express constraints. All of the concepts contained within an E-R diagram are included in the EE-R model, along with additional concepts that cover more semantic information. These additional concepts include generalization/specialization, union, inheritance, and subclass/super class.

Sample EER Diagram



Conclusion: Understand DCL, DDL commands, and ER/EER diagram for the system

FAQs:-

1. What is SQL Objects?
2. What is Table and how to create it?
3. What is sequence and how to create it on table?
4. What is index?
5. What ER and EER Diagram
6. Difference between ER and EER Diagram

Assignment No. 2

Title: Design and implement a database and apply at least 10 different DML queries for the following task. For a given input string display only those records which match the given pattern or a phrase in the search string. Make use of wild characters and LIKE operator for the same. Make use of Boolean and arithmetic operators wherever necessary

Aim: Design and implement a database and apply at least 10 different DML queries for the following task. For a given input string display only those records which match the given pattern or a phrase in the search string. Make use of wild characters and LIKE operator for the same. Make use of Boolean and arithmetic operators wherever necessary.
.

Objective: To understand the concept of DML statement like Insert, Select, Update, and LIKE operator.

Theory:

DATA MANIPULATION LANGUAGE (DML):

After the database structure is defined with DDL, database administrators and users can utilize the Data Manipulation Language to insert, retrieve and modify the data contained within it.

INSERT COMMAND:

The INSERT command in MYSQL is used to add records to an existing table.

Format 1:-Inserting a single row of data into a table

Syntax

```
INSERT INTO table_name
[(columnname,columnname)]
VALUES (expression,expression);
```

To add a new employee to the personal_info table

Example

```
INSERT INTO personal_info
values('bart','simpson',12345,$45000)
```

Format 2: Inserting data into a table from another table

Syntax

```
INSERT INTO tablename
SELECT columnname,columnname
FROM tablename
```

SELECT COMMAND:

Syntax

SELECT * FROM tablename.

OR

SELECT columnname,columnname,…..
FROM tablename ;

UPDATE COMMAND:

The UPDATE command can be used to modify information contained within a table.

Syntax UPDATE tablename

SET columnname=expression,columnname=expression,…..
WHERE columnname=expression;

Each year,company gives all employees a 3% cost-of-living increase in their salary. The following SQL command could be used to quickly apply this to all of the employees stored in the database:

Example

```
UPDATE personal_info
SET salary=salary*1.03
```

DELETE COMMAND:

The DELETE command can be used to delete information contained within a table.

Syntax

DELETE FROM tablename

WHERE search condition

The DELETE command with a WHERE clause can be used to remove his record from the personal_info table:

Example

```
DELETE FROM personal_info
WHERE employee_id=12345
```

```
The following command deletes all the rows from the table
```

Example DELETE FROM personal_info;

LIKE Operator

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards used in conjunction with the LIKE operator:

- % - The percent sign represents zero, one, or multiple characters
- _ - The underscore represents a single character

The percent sign and the underscore can also be used in combinations.

LIKE Syntax

```
SELECT column1, column2,
FROM table_name
WHERE columnN LIKE pattern;
```

The basic syntax of % and _ is as follows:

```
SELECT FROM table_name
WHERE column LIKE 'XXXX%'
or
SELECT FROM table_name
WHERE column LIKE '%XXXX%'
or
SELECT FROM table_name
WHERE column LIKE 'XXXX_'
or
SELECT FROM table_name
WHERE column LIKE '_XXXX'
or
SELECT FROM table_name
WHERE column LIKE '_XXXX_'
```

Here are some examples showing different LIKE operators with '%' and '_' wildcards:

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that starts with "a"
WHERE CustomerName LIKE '%a'	Finds any values that ends with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%_%'	Finds any values that starts with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that starts with "a" and ends with "o"

WHERE LIKE Examples

Problem: List all products with names that start with 'Ca'

```
SELECT Id, ProductName, UnitPrice, Package

FROM Product

WHERE ProductName LIKE 'Ca%'
```

Results:

Id	ProductName	UnitPrice	Package
18	Carnarvon Tigers	62.50	16 kg pkg.
60	Camembert Pierrot	34.00	15-300 g rounds

Conclusion: Implemented all SQL DML Commands like Insert, Select, Update, Delete with LIKE

.

FAQs:

1) What is DML (Data Manipulation Language)?

2) What is DML Compiler?

5) Name the sub-systems of a RDBMS.

6) What are primary keys and foreign keys?

7) What is data retrieval?

8) What is difference between delete table and delete from table?

9) What is difference between delete table and drop table?

10) What are the difference character used with LIKE Operator and use of that?

Department of Infromation Technology

Software Laborotory-I

Assignment No:-2 (PART-B)

Problem Statement

Create emp table in company database and insert following data in emp table.

Data for EMP

7369	Smith	Clerk	7902	17/12/80	800		20
7499	Allen	Salesman	7698	20/2/81	1600	300	30
7521	Ward	Salesman	7698	22/2/81	1250	500	30
7566	Jones	Manager	7839	2/4/81	2975		20
7654	Martin	Salesman	7698	28/9/81	1250	1400	30
7698	Blake	Manager	7839	1/5/81	2850		30
7782	Clark	Manager	7839	9/6/81	2450		10
7788	Scott	Analyst	7566	9/12/82	3000		20
7839	King	President		17/11/81	5000		10
7844	Turner	Salesman	7698	8/9/81	1500	0	30
7876	Adams	Clerk	7788	12/1/83	1100		20
7900	James	Clerk	7698	3/12/81	950		30
7902	Ford	Analyst	7566	4/12/81	3000		20
7934	Miller	Clerk	7782	23/1/82	1300		10

Execute following queries on emp table.

a) List the names of analysts and salesmen.

b) List details of employees who have joined before 30 Sep 11.

c) List names of employees who are not managers.

d) List the names of employees whose employee numbers are 7369, 7521, 7839, 7934,7788.

e) List employees not belonging to department 30, 40, or 10.

f) List employee names for those who have joined between 30 June and 31 Dec. ‘81.

g) List the different designations in the company.

h) List the names of employees who are not eligible for commission.

i) List the name and designation of the employee who does not report to anybody.

j) List the employees not assigned to any department.

k) List the employees who are eligible for commission.

l) List employees whose names either start or end with “S”.

m) List names of employees whose names have “i” as the second character.

```
mysql> create database COMPANY;
Query OK, 1 row affected (0.00 sec)
mysql> use COMPANY
Database changed
mysql> create table EMP(EMPNO int,ENAME varchar(15),JOB char(15),MGR int,HIREDATE date,SAL int,COMM int,DEPTNO int);
Query OK, 0 rows affected (0.33 sec)

mysql> insert into EMP values (7369,'Smith','Clerk',7902,'2010-12-17',800,null,20);
Query OK, 1 row affected (0.04 sec)

mysql> insert into EMP values (7499,'Allen','Salesman',7698,'2011-02-20',1600,300,30);
Query OK, 1 row affected (0.06 sec)

mysql> insert into EMP values (7521,'Ward','Salesman',7698,'2011-02-22',1250,500,30);
Query OK, 1 row affected (0.05 sec)

mysql> insert into EMP values (7566,'Jones','Manager',7839,'2011-04-02',2975,null,20);
Query OK, 1 row affected (0.06 sec)

mysql> insert into EMP values (7654,'Martin','Salesman',7698,'2011-09-28',1250,1400,30);
Query OK, 1 row affected (0.06 sec)

mysql> insert into EMP values (7698,'Blake','Manager',7838,'2011-05-01',2850,null,30);
Query OK, 1 row affected (0.05 sec)

mysql> insert into EMP values (7782,'Clark','Manager',7838,'2011-06-09',2450,null,10);
Query OK, 1 row affected (0.04 sec)

mysql> insert into EMP values (7788,'Scott','Analyst',7566,'2012-12-09',3000,null,20);
Query OK, 1 row affected (0.04 sec)

mysql> insert into EMP values (7839,'King','President',null,'2011-11-17',5000,null,10);
```

Query OK, 1 row affected (0.05 sec)

mysql> insert into EMP values (7844,'Turner','Salesman',7698,'2011-09-08',1500,0,30);
Query OK, 1 row affected (0.06 sec)

mysql> insert into EMP values (7876,'Adams','Clerk',7698,'2013-01-12',1100,NULL,20);
Query OK, 1 row affected (0.05 sec)

mysql> insert into EMP values (7902,'Ford','Analyst',7566,'2011-12-04',3000,NULL,20);
Query OK, 1 row affected (0.05 sec)

mysql> select * from EMP;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	Smith	Clerk	7902	2010-12-17	800	NULL	20
7499	Allen	Salesman	7698	2011-02-20	1600	300	30
7521	Ward	Salesman	7698	2011-02-22	1250	500	30
7566	Jones	Manager	7839	2011-04-02	2975	NULL	20
7654	Martin	Salesman	7698	2011-09-28	1250	1400	30
7698	Blake	Manager	7838	2011-05-01	2850	NULL	30
7782	Clark	Manager	7838	2011-06-09	2450	NULL	10
7788	Scott	Analyst	7566	2012-12-09	3000	NULL	20
7839	King	President	NULL	2011-11-17	5000	NULL	10
7844	Turner	Salesman	7698	2011-09-08	1500	0	30
7876	Adams	Clerk	7698	2013-01-12	1100	NULL	20
7902	Ford	Analyst	7566	2011-12-04	3000	NULL	20

12 rows in set (0.00 sec)

a) List the names of analysts and salesmen.

```
mysql> select ENAME from EMP where JOB='Analyst' or JOB='Salesman';
+--------+
| ENAME  |
+--------+
| Allen  |
| Ward   |
| Martin |
| Scott  |
| Turner |
| Ford   |
+--------+
6 rows in set (0.00 sec)
```

b) List details of employees who have joined before 30 Sep 11.

```
mysql> select * from EMP;
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk     | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7839 | King   | President | NULL | 2011-11-17 | 5000 | NULL |     10 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
+-------+--------+-----------+------+------------+------+------+--------+
12 rows in set (0.00 sec)
mysql> select * from EMP where HIREDATE<'2011-09-30';
```

```
+-------+--------+----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB      | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk    | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7499 | Allen  | Salesman | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7521 | Ward   | Salesman | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7566 | Jones  | Manager  | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7654 | Martin | Salesman | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7698 | Blake  | Manager  | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager  | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7844 | Turner | Salesman | 7698 | 2011-09-08 | 1500 |    0 |     30 |
+-------+--------+----------+------+------------+------+------+--------+
8 rows in set (0.00 sec)
```

c) List names of employees who are not managers.

```
mysql> select * from EMP;
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk     | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7839 | King   | President | NULL | 2011-11-17 | 5000 | NULL |     10 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
+-------+--------+-----------+------+------------+------+------+--------+
12 rows in set (0.00 sec)
```

```
mysql> select ENAME from EMP where not(JOB = 'Manager');
+--------+
| ENAME  |
+--------+
| Smith  |
| Allen  |
| Ward   |
| Martin |
| Scott  |
| King   |
| Turner |
| Adams  |
| Ford   |
+--------+
09 rows in set (0.00 sec)
```

d) List the names of employees whose employee numbers are 7369, 7521, 7839, 7934,7788.

```
mysql> select * from EMP;
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk     | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7839 | King   | President | NULL | 2011-11-17 | 5000 | NULL |     10 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
+-------+--------+-----------+------+------------+------+------+--------+
```

12 rows in set (0.00 sec)

mysql> select ENAME from EMP where EMPNO in(7369,7521,7839,7934,7788);

```
+-------+
| ENAME |
+-------+
| Smith |
| Ward  |
| Scott |
| King  |
+-------+
```

4 rows in set (0.03 sec)

e) List employees not belonging to department 30, 40, or 10.

mysql> select * from EMP;

```
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk     | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7839 | King   | President | NULL | 2011-11-17 | 5000 | NULL |     10 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
+-------+--------+-----------+------+------------+------+------+--------+
```

12 rows in set (0.00 sec)

mysql> select * from EMP where not DEPTNO in(30,40,10));

```
+-------+-------+---------+------+------------+------+------+--------+
| EMPNO | ENAME | JOB     | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+-------+---------+------+------------+------+------+--------+
|  7369 | Smith | Clerk   | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7566 | Jones | Manager | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7788 | Scott | Analyst | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7876 | Adams | Clerk   | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7902 | Ford  | Analyst | 7566 | 2011-12-04 | 3000 | NULL |     20 |
+-------+-------+---------+------+------------+------+------+--------+
5 rows in set (0.00 sec)
```

f) List employee names for those who have joined between 30 June and 31 Dec. ‘81.

```
mysql> select * from EMP;
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk     | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7839 | King   | President | NULL | 2011-11-17 | 5000 | NULL |     10 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
+-------+--------+-----------+------+------------+------+------+--------+
12 rows in set (0.00 sec)
```

mysql> select ENAME from EMP where HIREDATE between '2011-06-30' and '2011-12-31';

```
+--------+
| ENAME  |
+--------+
| Martin |
| King   |
| Turner |
| Ford   |
+--------+
```

4 rows in set (0.00 sec)

g) List the different designations in the company.

mysql> select * from EMP;

```
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk     | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7839 | King   | President | NULL | 2011-11-17 | 5000 | NULL |     10 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
+-------+--------+-----------+------+------------+------+------+--------+
```

12 rows in set (0.00 sec)

```
mysql> select distinct(JOB) from EMP;
+-----------+
| JOB       |
+-----------+
| Clerk     |
| Salesman  |
| Manager   |
| Analyst   |
| President |
+-----------+
5 rows in set (0.00 sec)
```

h) List the names of employees who are not eligible for commission.

```
mysql> select * from EMP;
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk     | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7839 | King   | President | NULL | 2011-11-17 | 5000 | NULL |     10 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
+-------+--------+-----------+------+------------+------+------+--------+
12 rows in set (0.00 sec)
```

```
mysql> select ENAME from EMP where COMM is null;
+-------+
| ENAME |
+-------+
| Smith |
| Jones |
| Blake |
| Clark |
| Scott |
| King  |
| Adams |
| Ford  |
+-------+
8 rows in set (0.00 sec)
```

i) List the name and designation of the employee who does not report to anybody.

```
mysql> select * from EMP;
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk     | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7839 | King   | President | NULL | 2011-11-17 | 5000 | NULL |     10 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
+-------+--------+-----------+------+------------+------+------+--------+
12 rows in set (0.00 sec)
```

```
mysql> select ENAME from EMP where MGR is null;
+-------+
| ENAME |
+-------+
| King  |
+-------+
1 rows in set (0.00 sec)
```

j) List the employees not assigned to any department.

```
mysql> select * from EMP;
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk     | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7839 | King   | President | NULL | 2011-11-17 | 5000 | NULL |     10 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
+-------+--------+-----------+------+------------+------+------+--------+
12 rows in set (0.00 sec)

mysql> select * from EMP where DEPTNO is null;
Empty set (0.00 sec)
```

k) List the employees who are eligible for commission.

```
mysql> select * from EMP;
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk     | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7839 | King   | President | NULL | 2011-11-17 | 5000 | NULL |     10 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
+-------+--------+-----------+------+------------+------+------+--------+
12 rows in set (0.00 sec)

mysql> select ENAME from EMP where not(COMM is null);
+--------+
| ENAME  |
+--------+
| Allen  |
| Ward   |
| Martin |
| Turner |
+--------+
4 rows in set (0.00 sec)
```

l) List employees whose names either start or end with “S”.

```
mysql> select * from EMP;
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk     | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7839 | King   | President | NULL | 2011-11-17 | 5000 | NULL |     10 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
+-------+--------+-----------+------+------------+------+------+--------+
12 rows in set (0.00 sec)

mysql> select ENAME from EMP where ENAME like 'S%' or ENAME like '%s';
+-------+
| ENAME |
+-------+
| Smith |
| Jones |
| Scott |
| Adams |
+-------+
4 rows in set (0.00 sec)
```

m) List names of employees whose names have “i” as the second character.

```
mysql> select * from EMP;
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk     | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7839 | King   | President | NULL | 2011-11-17 | 5000 | NULL |     10 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
+-------+--------+-----------+------+------------+------+------+--------+
12 rows in set (0.00 sec)

mysql> select ENAME from EMP where ENAME like '_i%';
+-------+
| ENAME |
+-------+
| King  |
+-------+
1 rows in set (0.00 sec)
```

Assignment No. 3

Title: Execute the aggregate functions like count, sum, avg etc. and date functions like now (), date (), day (), time () etc. on the suitable database.

Aim: Execute the aggregate functions like count, sum, avg etc. and date functions like now (), date (), day (), time () etc. on the suitable database.

Objective: Understand the aggregate functions like count, sum, avg etc. and date functions like now (), date (), day (), time () etc

Theory:

Aggregate Functions

Aggregate functions return a single result row based on groups of rows, rather than on single rows. Aggregate functions can appear in select lists and in ORDER BY and HAVING clauses. They are commonly used with the GROUP BY clause in a SELECT statement. In a query containing a GROUP BY clause, the elements of the select list can be aggregate functions, GROUP BY expressions, constants, or expressions involving one of these.

Aggregate functions are used to compute against a "returned column of numeric data" from your SELECT statement. They basically summarize the results of a particular column of selected data.

SQL has many built-in functions for performing calculations on data.

MIN	returns the smallest value in a given column
MAX	returns the largest value in a given column
SUM	returns the sum of the numeric values in a given column
AVG	returns the average value of a given column
COUNT	returns the total number of values in a given column
COUNT(*)	returns the number of rows in a table
ROUND()	Rounds a numeric field to the number of decimals specified

The AVG ( ) Function

The AVG () function returns the average value of a numeric column.

SQL AVG () Syntax

SELECT AVG (column_name) FROM table_name

SQL AVG() Example

The following SQL statement gets the average value of the "Price" column from the "Products" table:

SELECT AVG(Price) AS PriceAverage FROM Products;

The COUNT ( ) Function

The COUNT() function returns the number of rows that matches a specified criteria.

SQL COUNT(column_name) Syntax

The COUNT(column_name) function returns the number of values (NULL values will not be counted) of the specified column:

```
SELECT COUNT(column_name) FROM table_name;
```

SQL COUNT(*) Syntax

The COUNT(*) function returns the number of records in a table:

```
SELECT COUNT(*) FROM table_name;
```

SQL COUNT(*) Example

The following SQL statement counts the total number of orders in the "Orders" table:

```
SELECT COUNT(*) AS NumberOfOrders FROM Orders;
```

The MAX ( ) Function

The MAX() function returns the largest value of the selected column.

SQL MAX() Syntax

```
SELECT MAX(column_name) FROM table_name;
```

SQL MAX() Example

The following SQL statement gets the largest value of the "Price" column from the "Products" table:

```
SELECT MAX(Price) AS HighestPrice FROM Products;
```

The MIN ( ) Function

The MIN() function returns the smallest value of the selected column.

SQL MIN() Syntax

```
SELECT MIN(column_name) FROM table_name;
```

SQL MIN() Example

The following SQL statement gets the smallest value of the "Price" column from the "Products" table:

```
SELECT MIN(Price) AS SmallestOrderPrice FROM Products;
```

The ROUND ( ) Function

The ROUND () function is used to round a numeric field to the number of decimals specified.

SQL ROUND () Syntax

```
SELECT ROUND(column_name,decimals) FROM table_name;
```

Parameter	Description
column_name	Required. The field to round.
decimals	Required. Specifies the number of decimals to be returned

SQL ROUND () Example

The following SQL statement selects the product name and rounds the price in the "Products" table:

```
SELECT ProductName, ROUND(Price,0) AS RoundedPrice
FROM Products;
```

The SUM ( ) Function

The SUM() function returns the total sum of a numeric column.

SQL SUM() Syntax

```
SELECT SUM(column_name) FROM table_name;
```

SQL SUM() Example

The following SQL statement finds the sum of all the "Quantity" fields for the "OrderDetails" table:

SELECT SUM(Quantity) AS TotalItemsOrdered FROM OrderDetails;

Date Functions

The following table lists the most important built-in date functions.

Function	Description
NOW()	Returns the current date and time
CURDATE()	Returns the current date
CURTIME()	Returns the current time
DATE()	Extracts the date part of a date or date/time expression
EXTRACT()	Returns a single part of a date/time
DATE_ADD()	Adds a specified time interval to a date
DATE_SUB()	Subtracts a specified time interval from a date
DATEDIFF()	Returns the number of days between two dates
DATE_FORMAT()	Displays date/time data in different formats

Date Data Types

- DATE - format YYYY-MM-DD
- DATETIME - format: YYYY-MM-DD HH:MI:SS
- TIMESTAMP - format: YYYY-MM-DD HH:MI:SS
- YEAR - format YYYY or YY

NOW ( ) Function

NOW () returns the current date and time.

- **Syntax**
- NOW()
- **Example**
- The following SELECT statement:
- SELECT NOW(),CURDATE(),CURTIME()

- will result in something like this:

NOW()	CURDATE()	CURTIME()
2014-11-22 12:45:34	2014-11-22	12:45:34

DATE () Function

The DATE () function extracts the date part of a date or date/time expression.

Syntax

DATE (date)

Example

Assume we have the following "Orders" table:

OrderId	ProductName	OrderDate
1	Jarlsberg Cheese	2014-11-22 13:23:44.657

The following SELECT statement:

```
SELECT ProductName, DATE(OrderDate) AS OrderDate
FROM Orders
WHERE OrderId=1
```

Will result in this:

ProductName	OrderDate
Jarlsberg Cheese	2014-11-22

EXTRACT () Function

The EXTRACT () function is used to return a single part of a date/time, such as year, month, day, hour, minute, etc.

Syntax

EXTRACT(unit FROM date)

Example

Assume we have the following "Orders" table:

OrderId	ProductName	OrderDate
1	Jarlsberg Cheese	2014-11-22 13:23:44.657

The following SELECT statement:

```
SELECT EXTRACT(YEAR FROM OrderDate) AS OrderYear,
EXTRACT(MONTH FROM OrderDate) AS OrderMonth,
EXTRACT(DAY FROM OrderDate) AS OrderDay
FROM Orders
WHERE OrderId=1
```

Will result in this:

OrderYear	OrderMonth	OrderDate
2014	11	22

DATE_ADD () Function

The DATE_ADD () function adds a specified time interval to a date.

Syntax

DATE_ADD(date,INTERVAL expr type)

Where date is a valid date expression and expr is the number of interval you want to add.

Example

Assume we have the following "Orders" table:

OrderId	ProductName	OrderDate
1	Jarlsberg Cheese	2014-11-22 13:23:44.657

Now we want to add 30 days to the "OrderDate", to find the payment date.

We use the following SELECT statement:

```
SELECT OrderId,DATE_ADD(OrderDate,INTERVAL 30 DAY) AS OrderPayDate
FROM Orders
```

Result:

OrderId	OrderPayDate
1	2014-12-22 13:23:44.657

DATE_SUB () Function

The DATE_SUB () function subtracts a specified time interval from a date.

Syntax

DATE_SUB(date,INTERVAL expr type)

Where date is a valid date expression and expr is the number of interval you want to subtract.

Example

Assume we have the following "Orders" table:

OrderId	ProductName	OrderDate
1	Jarlsberg Cheese	2014-11-22 13:23:44.657

Now we want to subtract 5 days from the "OrderDate" date.

We use the following SELECT statement:

```
SELECT OrderId, DATE_SUB (OrderDate, INTERVAL 5 DAY) AS SubtractDate
FROM Orders
```

Result:

OrderId	SubtractDate
1	2014-11-17 13:23:44.657

DATEDIFF () Function

The DATEDIFF () function returns the time between two dates.

Syntax

DATEDIFF (date1,date2)

Where date1 and date2 are valid date or date/time expressions.

Example

The following SELECT statement:

```
SELECT DATEDIFF ('2014-11-30','2014-11-29') AS DiffDate
```

Will result in this:

DiffDate
1

Example

The following SELECT statement:

```
SELECT DATEDIFF('2014-11-29','2014-11-30') AS DiffDate
```

Will result in this:

DiffDate
1

DATE_FORMAT ( ) Function

The DATE_FORMAT () function is used to display date/time data in different formats.

Syntax

DATE_FORMAT (date,format)

Where date is a valid date and format specifies the output format for the date/time.

Example

The following script uses the DATE_FORMAT () function to display different formats. We will use the NOW () function to get the current date/time:

```
DATE_FORMAT(NOW(),'%b %d %Y %h:%i %p')
DATE_FORMAT(NOW(),'%m-%d-%Y')
DATE_FORMAT(NOW(),'%d %b %y')
DATE_FORMAT(NOW(),'%d %b %Y %T:%f')
```

The result would look something like this:

```
Nov 04 2014 11:45 PM
11-04-2014
04 Nov 14
04 Nov 2014 11:45:34:243
```

Conclusion: Implemented all Date and Time Functions in SQL.

Department of Infromation Technology

Subject: Software Laborotory-I

Assignment No:-3 (PART-B)

Problem Statement

Create emp table in company database and insert follwing data in emp table.

Data for EMP

7369	Smith	Clerk	7902	17/12/80	800		20
7499	Allen	Salesman	7698	20/2/81	1600	300	30
7521	Ward	Salesman	7698	22/2/81	1250	500	30
7566	Jones	Manager	7839	2/4/81	2975		20
7654	Martin	Salesman	7698	28/9/81	1250	1400	30
7698	Blake	Manager	7839	1/5/81	2850		30
7782	Clark	Manager	7839	9/6/81	2450		10
7788	Scott	Analyst	7566	9/12/82	3000		20
7839	King	President		17/11/81	5000		10
7844	Turner	Salesman	7698	8/9/81	1500	0	30
7876	Adams	Clerk	7788	12/1/83	1100		20
7900	James	Clerk	7698	3/12/81	950		30
7902	Ford	Analyst	7566	4/12/81	3000		20
7934	Miller	Clerk	7782	23/1/82	1300		10

Execute following queries on employee collection.

a) List the number of employees and average salary for employees in department 20.

b) List name, salary and PF amount of all employees. (PF is calculated as 10% of basic salary)

c) List names of employees who are more than 14 years old in the company in 1995.

d) List the employee details in the ascending order of their basic salary.

e) List the employee name and hire date in the descending order of the hire date.

f) List employee name, salary, PF, HRA, DA and gross; order the results in the ascending order of gross. HRA is 50% of the salary and DA is 30% of the salary.

g) List the department numbers and number of employees in each department.

h) List the department number and total salary payable in each department.

i) List the jobs and number of employees in each job. The result should be in the descending order of the number of employees.

j) List the total salary, maximum and minimum salary and average salary of the employees jobwise.

k) List the total salary, maximum and minimum salary and average salary of the employees, for department 20.

l) List the average salary of the employees job wise, for department 20 and display only those rows having an average salary > 1000

m) List the maximum salary paid to a salesman.

n) List the number of employees working with the company.

o) List the number of designations available in the EMP table.

p) List the total salaries paid to the employees.

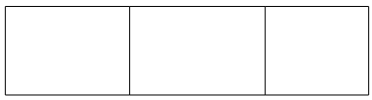
q) List the maximum, minimum and average salary in the company.

a) List the number of employees and average salary for employees in department 20.

```
mysql> use company
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from emp;
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk     | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7839 | King   | President | NULL | 2011-11-17 | 5000 | NULL |     10 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
+-------+--------+-----------+------+------------+------+------+--------+
12 rows in set (0.00 sec)

mysql>  select count(*),avg(sal),deptno from emp where deptno=20;
+----------+-----------+--------+
| count(*) | avg(sal)  | deptno |
+----------+-----------+--------+
|        5 | 2175.0000 |     20 |
```



b) List name, salary and PF amount of all employees. (PF is calculated as 10% of basic salary)

```
mysql> select * from emp;
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk     | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7839 | King   | President | NULL | 2011-11-17 | 5000 | NULL |     10 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
+-------+--------+-----------+------+------------+------+------+--------+
12 rows in set (0.00 sec)
mysql> select ename,sal,((sal*10)/100) as PF from emp;
+--------+------+----------+
| ename  | sal  | PF       |
+--------+------+----------+
| Smith  |  800 |  80.0000 |
| Allen  | 1600 | 160.0000 |
| Ward   | 1250 | 125.0000 |
| Jones  | 2975 | 297.5000 |
| Martin | 1250 | 125.0000 |
| Blake  | 2850 | 285.0000 |
| Clark  | 2450 | 245.0000 |
| Scott  | 3000 | 300.0000 |
| King   | 5000 | 500.0000 |
| Turner | 1500 | 150.0000 |
| Adams  | 1100 | 110.0000 |
| Ford   | 3000 | 300.0000 |
```

+--------+------+----------+

12 rows in set (0.00 sec)

c) List names of employees who are more than 14 years old in the company in 1995.

mysql> select * from emp;

+-------+--------+-----------+------+------------+------+------+--------+

| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO |

+-------+--------+-----------+------+------------+------+------+--------+

| 7369 | Smith | Clerk | 7902 | 2010-12-17 | 800 | NULL | 20 |

| 7499 | Allen | Salesman | 7698 | 2011-02-20 | 1600 | 300 | 30 |

| 7521 | Ward | Salesman | 7698 | 2011-02-22 | 1250 | 500 | 30 |

| 7566 | Jones | Manager | 7839 | 2011-04-02 | 2975 | NULL | 20 |

| 7654 | Martin | Salesman | 7698 | 2011-09-28 | 1250 | 1400 | 30 |

| 7698 | Blake | Manager | 7838 | 2011-05-01 | 2850 | NULL | 30 |

| 7782 | Clark | Manager | 7838 | 2011-06-09 | 2450 | NULL | 10 |

| 7788 | Scott | Analyst | 7566 | 2012-12-09 | 3000 | NULL | 20 |

| 7839 | King | President | NULL | 2011-11-17 | 5000 | NULL | 10 |

| 7844 | Turner | Salesman | 7698 | 2011-09-08 | 1500 | 0 | 30 |

| 7876 | Adams | Clerk | 7698 | 2013-01-12 | 1100 | NULL | 20 |

| 7902 | Ford | Analyst | 7566 | 2011-12-04 | 3000 | NULL | 20 |

+-------+--------+-----------+------+------------+------+------+--------+

12 rows in set (0.00 sec)

mysql> select ename from emp where (1995-year(hiredate) >= 14) ;

Empty set (0.03 sec)

d) List the employee details in the ascending order of their basic salary.

```
mysql> select * from emp;
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk     | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7839 | King   | President | NULL | 2011-11-17 | 5000 | NULL |     10 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
+-------+--------+-----------+------+------------+------+------+--------+
12 rows in set (0.00 sec)

mysql> select * from emp order by sal;
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk     | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
```

| 7839 | King | President | NULL | 2011-11-17 | 5000 | NULL | 10 |

+-------+--------+-----------+------+------------+------+------+--------+

12 rows in set (0.03 sec)

e) List the employee name and hire date in the descending order of the hire date.

```
mysql> select * from emp;
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk     | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7839 | King   | President | NULL | 2011-11-17 | 5000 | NULL |     10 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
+-------+--------+-----------+------+------------+------+------+--------+
12 rows in set (0.00 sec)

mysql> select * from emp order by sal desc;
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7839 | King   | President | NULL | 2011-11-17 | 5000 | NULL |     10 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
```

| 7369 | Smith | Clerk | 7902 | 2010-12-17 | 800 | NULL | 20 |

+-------+--------+-----------+------+------------+------+------+--------+

12 rows in set (0.00 sec)

f) List employee name, salary, PF, HRA, DA and gross; order the results in the ascending order of gross. HRA is 50% of the salary and DA is 30% of the salary.

mysql> select * from emp;

+-------+--------+-----------+------+------------+------+------+--------+

| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO |

+-------+--------+-----------+------+------------+------+------+--------+

| 7369 | Smith | Clerk | 7902 | 2010-12-17 | 800 | NULL | 20 |

| 7499 | Allen | Salesman | 7698 | 2011-02-20 | 1600 | 300 | 30 |

| 7521 | Ward | Salesman | 7698 | 2011-02-22 | 1250 | 500 | 30 |

| 7566 | Jones | Manager | 7839 | 2011-04-02 | 2975 | NULL | 20 |

| 7654 | Martin | Salesman | 7698 | 2011-09-28 | 1250 | 1400 | 30 |

| 7698 | Blake | Manager | 7838 | 2011-05-01 | 2850 | NULL | 30 |

| 7782 | Clark | Manager | 7838 | 2011-06-09 | 2450 | NULL | 10 |

| 7788 | Scott | Analyst | 7566 | 2012-12-09 | 3000 | NULL | 20 |

| 7839 | King | President | NULL | 2011-11-17 | 5000 | NULL | 10 |

| 7844 | Turner | Salesman | 7698 | 2011-09-08 | 1500 | 0 | 30 |

| 7876 | Adams | Clerk | 7698 | 2013-01-12 | 1100 | NULL | 20 |

| 7902 | Ford | Analyst | 7566 | 2011-12-04 | 3000 | NULL | 20 |

+-------+--------+-----------+------+------------+------+------+--------+

12 rows in set (0.00 sec)

mysql> select ename,sal,((sal*10)/100) as PF,((sal*50)/100) as HRA,((sal*30)/100) as DA,(sal+((sal*10)/100)+((sal*50)/100)+((sal*30)/100)) as Gross from emp order by Gross;

ename	sal	PF	HRA	DA	Gross
Smith	800	80.0000	400.0000	240.0000	1520.0000
Adams	1100	110.0000	550.0000	330.0000	2090.0000
Ward	1250	125.0000	625.0000	375.0000	2375.0000
Martin	1250	125.0000	625.0000	375.0000	2375.0000
Turner	1500	150.0000	750.0000	450.0000	2850.0000
Allen	1600	160.0000	800.0000	480.0000	3040.0000
Clark	2450	245.0000	1225.0000	735.0000	4655.0000
Blake	2850	285.0000	1425.0000	855.0000	5415.0000
Jones	2975	297.5000	1487.5000	892.5000	5652.5000
Scott	3000	300.0000	1500.0000	900.0000	5700.0000
Ford	3000	300.0000	1500.0000	900.0000	5700.0000
King	5000	500.0000	2500.0000	1500.0000	9500.0000

12 rows in set (0.00 sec)

g) List the department numbers and number of employees in each department.

```
mysql> select * from emp;                                                    +-------+--------
+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk     | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7839 | King   | President | NULL | 2011-11-17 | 5000 | NULL |     10 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
+-------+--------+-----------+------+------------+------+------+--------+
12 rows in set (0.00 sec)

mysql> select count(*),deptno from emp group by deptno;
+----------+--------+
| count(*) | deptno |
+----------+--------+
|        2 |     10 |
|        5 |     20 |
|        5 |     30 |
+----------+--------+
3 rows in set (0.01 sec)
```

h) List the department number and total salary payable in each department.

```
mysql> select * from emp;
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk     | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7839 | King   | President | NULL | 2011-11-17 | 5000 | NULL |     10 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
+-------+--------+-----------+------+------------+------+------+--------+
12 rows in set (0.00 sec)

mysql> select deptno,sum(sal) as Total_Sal_Payable from emp group by deptno;
+--------+-------------------+
| deptno | Total_Sal_Payable |
+--------+-------------------+
|     10 |              7450 |
|     20 |             10875 |
|     30 |              8450 |
+--------+-------------------+
3 rows in set (0.04 sec)
```

i) List the jobs and number of employees in each job. The result should be in the descending order of the number of employees.

```
mysql> select * from emp;
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk     | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7839 | King   | President | NULL | 2011-11-17 | 5000 | NULL |     10 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
+-------+--------+-----------+------+------------+------+------+--------+
12 rows in set (0.00 sec)

mysql> select job as Job,count(empno) as Num_Of_Emp from emp group by job order by count(empno) desc;
+-----------+------------+
| Job       | Num_Of_Emp |
+-----------+------------+
| Salesman  |          4 |
| Manager   |          3 |
| Analyst   |          2 |
| Clerk     |          2 |
| President |          1 |
+-----------+------------+
5 rows in set (0.00 sec)
```

j) List the total salary, maximum and minimum salary and average salary of the employees jobwise.

mysql> select * from emp;

```
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk     | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7839 | King   | President | NULL | 2011-11-17 | 5000 | NULL |     10 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
+-------+--------+-----------+------+------------+------+------+--------+
12 rows in set (0.00 sec)
```

mysql> select job as Job,sum(sal) as Total,max(sal) as Max,min(sal) as Min,avg(sal) as Average from emp group by job;

```
+-----------+-------+------+------+-----------+
| Job       | Total | Max  | Min  | Average   |
+-----------+-------+------+------+-----------+
| Analyst   |  6000 | 3000 | 3000 | 3000.0000 |
| Clerk     |  1900 | 1100 |  800 |  950.0000 |
| Manager   |  8275 | 2975 | 2450 | 2758.3333 |
| President |  5000 | 5000 | 5000 | 5000.0000 |
| Salesman  |  5600 | 1600 | 1250 | 1400.0000 |
+-----------+-------+------+------+-----------+
5 rows in set (0.00 sec)
```

k) List the total salary, maximum and minimum salary and average salary of the employees, for department 20.

```
mysql> select * from emp;                                                    +-------+--------
+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk     | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7839 | King   | President | NULL | 2011-11-17 | 5000 | NULL |     10 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
+-------+--------+-----------+------+------------+------+------+--------+
12 rows in set (0.00 sec)

mysql> select deptno,sum(sal),max(sal),min(sal),avg(sal) from emp where deptno = 20;
+--------+----------+----------+----------+-----------+
| deptno | sum(sal) | max(sal) | min(sal) | avg(sal)  |
+--------+----------+----------+----------+-----------+
|     20 |    10875 |     3000 |      800 | 2175.0000 |
+--------+----------+----------+----------+-----------+
1 row in set (0.00 sec)
```

l) List the average salary of the employees job wise, for department 20 and display only those rows having an average salary > 1000

mysql> select * from emp;

```
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk     | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7839 | King   | President | NULL | 2011-11-17 | 5000 | NULL |     10 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
+-------+--------+-----------+------+------------+------+------+--------+
12 rows in set (0.00 sec)
```

mysql> select job as Job,sum(sal) as Total,max(sal) as Max,min(sal) as Min,avg(sal) as Average,deptno as Dept from emp where deptno=20 group by job having avg(sal)>1000;

```
+---------+-------+------+------+-----------+------+
| Job     | Total | Max  | Min  | Average   | Dept |
+---------+-------+------+------+-----------+------+
| Analyst |  6000 | 3000 | 3000 | 3000.0000 |   20 |
| Manager |  2975 | 2975 | 2975 | 2975.0000 |   20 |
+---------+-------+------+------+-----------+------+
2 rows in set (0.00 sec)
```

m) List the maximum salary paid to a salesman.

```
mysql> select * from emp;                                                    +-------+--------
+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk     | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7839 | King   | President | NULL | 2011-11-17 | 5000 | NULL |     10 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
+-------+--------+-----------+------+------------+------+------+--------+
12 rows in set (0.00 sec)

mysql> select max(sal) from emp where job='Salesman';
+----------+
| max(sal) |
+----------+
|     1600 |
+----------+
1 row in set (0.00 sec)
```

n) List the number of employees working with the company.

```
mysql> select * from emp;                                                    +-------+--------
+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk     | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7839 | King   | President | NULL | 2011-11-17 | 5000 | NULL |     10 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
+-------+--------+-----------+------+------------+------+------+--------+
12 rows in set (0.00 sec)

mysql> select count(*) from emp;
+----------+
| count(*) |
+----------+
|       12 |
+----------+
1 row in set (0.00 sec)
```

o) List the number of designations available in the EMP table.

```
mysql> select * from emp;
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk     | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7839 | King   | President | NULL | 2011-11-17 | 5000 | NULL |     10 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
+-------+--------+-----------+------+------------+------+------+--------+
12 rows in set (0.00 sec)

mysql> select count(distinct(job)) from emp;
+----------------------+
| count(distinct(job)) |
+----------------------+
|                    5 |
+----------------------+
1 row in set (0.00 sec)
```

p) List the total salaries paid to the employees.

```
mysql> select * from emp;
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk     | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7839 | King   | President | NULL | 2011-11-17 | 5000 | NULL |     10 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
+-------+--------+-----------+------+------------+------+------+--------+
12 rows in set (0.00 sec)

mysql>  select sum(sal) from emp;
+----------+
| sum(sal) |
+----------+
|    26775 |
+----------+
1 row in set (0.00 sec)
```

q) List the maximum, minimum and average salary in the company.

```
mysql> select * from emp;
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7369 | Smith  | Clerk     | 7902 | 2010-12-17 |  800 | NULL |     20 |
|  7499 | Allen  | Salesman  | 7698 | 2011-02-20 | 1600 |  300 |     30 |
|  7521 | Ward   | Salesman  | 7698 | 2011-02-22 | 1250 |  500 |     30 |
|  7566 | Jones  | Manager   | 7839 | 2011-04-02 | 2975 | NULL |     20 |
|  7654 | Martin | Salesman  | 7698 | 2011-09-28 | 1250 | 1400 |     30 |
|  7698 | Blake  | Manager   | 7838 | 2011-05-01 | 2850 | NULL |     30 |
|  7782 | Clark  | Manager   | 7838 | 2011-06-09 | 2450 | NULL |     10 |
|  7788 | Scott  | Analyst   | 7566 | 2012-12-09 | 3000 | NULL |     20 |
|  7839 | King   | President | NULL | 2011-11-17 | 5000 | NULL |     10 |
|  7844 | Turner | Salesman  | 7698 | 2011-09-08 | 1500 |    0 |     30 |
|  7876 | Adams  | Clerk     | 7698 | 2013-01-12 | 1100 | NULL |     20 |
|  7902 | Ford   | Analyst   | 7566 | 2011-12-04 | 3000 | NULL |     20 |
+-------+--------+-----------+------+------------+------+------+--------+
12 rows in set (0.00 sec)

mysql>  select max(sal),min(sal),avg(sal) from emp;
+----------+----------+-----------+
| max(sal) | min(sal) | avg(sal)  |
+----------+----------+-----------+
|     5000 |      800 | 2231.2500 |
+----------+----------+-----------+
1 row in set (0.00 sec)
```

Assignment No.4

Title: Implement nested sub queries. Perform a test for set membership (in, not in), set comparison (<some, >=some, <all etc.) and set cardinality (unique, not unique).

Aim: Implement nested sub queries. Perform a test for set membership (in, not in), set comparison (<some, >=some, <all etc.) and set cardinality (unique, not unique).

Objective:

1. To learn nested sub queries using set comparison and set cardinality operators.

Theory:

Nested Sub Query

Definition of Nested Sub Query:-

A Sub query or Inner query or Nested query is a query within another SQL query and embedded within the WHERE clause. A sub query is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Sub queries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN etc.

A sub query can be nested inside other sub queries. SQL has an ability to nest queries within one another. A sub query is a SELECT statement that is nested within another SELECT statement and which return intermediate results. SQL executes innermost sub query first, then next level.

Sub queries with the SELECT Statement:

Sub queries are most frequently used with the SELECT statement.

The basic syntax is as follows:

```
SELECT column_name [, column_name ]
FROM   table1 [, table2]
WHERE  column_name OPERATOR
    (SELECT column_name [, column_name ]
    FROM table1 [, table2]
    [WHERE])
```

Example of IN Operator

Consider the CUSTOMERS table having the following records:

```
+----+----------+-----+-----------+---------------------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+---------------------+
|  1 | Ramesh |  35 | Ahmedabad         |  2000.00 |
|  2 | Khilan   |  25 | Delhi             |  1500.00 |
|  3 | kaushik  |  23 | Kota              |  2000.00 |
|  4 | Chaitali |  25 | Mumbai            |  6500.00 |
|  5 | Hardik   |  27 | Bhopal            |  8500.00 |
|  6 | Komal    |  22 | MP                |  4500.00 |
|  7 | Muffy    |  24 | Indore            | 10000.00 |
+----+----------+-----+-------------------+----------+
```

Now, let us check following sub query with SELECT statement:

```
SELECT *
    FROM CUSTOMERS
    WHERE ID IN (SELECT ID
                FROM CUSTOMERS
                WHERE SALARY > 4500);
```

This would produce the following result

```
----+----------+-----+---------+----------------------+
| ID | NAME     | AGE | ADDRESS | SALARY   |
+----+----------+-----+---------+----------------------+
|  4 | Chaitali |  25 | Mumbai              |  6500.00 |
|  5 | Hardik   |  27 | Bhopal              |  8500.00 |
|  7 | Muffy    |  24 | Indore              | 10000.00 |
+----+----------+-----+---------+----------------------+
```

Example of NOT IN Operator

Now, let us check following sub query with SELECT statement:

```
SELECT *
    FROM CUSTOMERS
    WHERE ID NOT IN (SELECT ID
                FROM CUSTOMERS
                WHERE SALARY > 4500);
```

This would produce the following result

```
+----+----------+-----+-----------+---------------------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+---------------------+
|  1 | Ramesh |  35 | Ahmedabad         |  2000.00   |
|  2 | Khilan   |  25 | Delhi             |  1500.00   |
|  3 | kaushik  |  23 | Kota              |  2000.00   |
|  6 | Komal    |  22 | MP                |  4500.00   |
|+----+----------+-----+-------------------+----------+
```

Sub queries with the INSERT Statement:

Sub queries also can be used with INSERT statements. The INSERT statement uses the data returned from the sub query to insert into another table. The selected data in the sub query can be modified with any of the character, date or number functions.

Example:

Consider a table CUSTOMERS_BKP with similar structure as CUSTOMERS table. Now to copy complete CUSTOMERS table into CUSTOMERS_BKP, following is the syntax:

*INSERT INTO CUSTOMERS_BKP SELECT * FROM CUSTOMERS WHERE ID IN (SELECT ID FROM CUSTOMERS);*

Sub queries with the UPDATE Statement:

The sub query can be used in conjunction with the UPDATE statement. Either single or multiple columns in a table can be updated when using a sub query with the UPDATE statement.

Example:

Assuming, we have CUSTOMERS_BKP table available which is backup of CUSTOMERS table.

Following example updates SALARY by 0.25 times in CUSTOMERS table for all the customers whose AGE is greater than or equal to 27:

```
UPDATE CUSTOMERS
   SET SALARY = SALARY * 0.25
   WHERE AGE IN (SELECT AGE FROM CUSTOMERS_BKP
            WHERE AGE >= 27);
```

This would impact two rows and finally CUSTOMERS table would have the following records:

```
----+----------+-----+-----------+----------------------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------------------+
|  1 | Ramesh |  35 | Ahmedabad           |   125.00 |
|  2 | Khilan   |  25 | Delhi               |  1500.00 |
|  3 | kaushik  |  23 | Kota                |  2000.00 |
|  4 | Chaitali |  25 | Mumbai              |  6500.00 |
|  5 | Hardik   |  27 | Bhopal              |  2125.00 |
|  6 | Komal    |  22 | MP                  |  4500.00 |
|  7 | Muffy    |  24 | Indore              | 10000.00 |
+----+----------+-----+-----------+----------------------+
```

Sub queries with the DELETE Statement:

The sub query can be used in conjunction with the DELETE statement like with any other statements mentioned above.

Example:

Assuming, we have CUSTOMERS_BKP table available which is backup of CUSTOMERS table.

Following example deletes records from CUSTOMERS table for all the customers whose AGE is greater than or equal to 27:

```
DELETE FROM CUSTOMERS
   WHERE AGE IN (SELECT AGE FROM CUSTOMERS_BKP
            WHERE AGE >= 27);
```

This would impact two rows and finally CUSTOMERS table would have the following records:

```
+----+----------+-----+---------+----------------------+
| ID | NAME     | AGE | ADDRESS | SALARY   |
+----+----------+-----+---------+----------------------+
|  2 | Khilan   |  25 | Delhi               |  1500.00 |
|  3 | kaushik  |  23 | Kota                |  2000.00 |
|  4 | Chaitali |  25 | Mumbai              |  6500.00 |
|  6 | Komal    |  22 | MP                  |  4500.00 |
|  7 | Muffy    |  24 | Indore              | 10000.00 |
+----+----------+-----+---------+----------------------+
```

Conclusion: Implemented all Nested sub query.