1. Entity Relationship Model

Q1) To draw ER Model and Relational Model for a given database

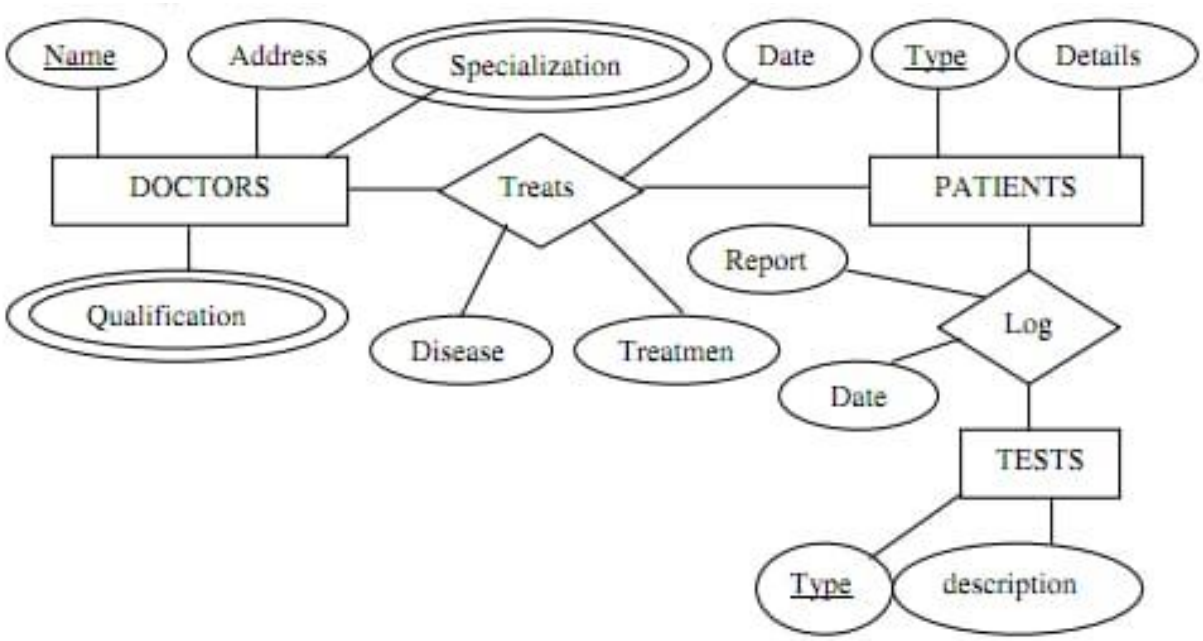
a) Case study 1: List the data requirements for the database of the company which keeps track of the company employee, department and projects. The database designers provide the following description

1. The company is organized into departments. Each department has unique name, unique number, and particular employee to manage the department. We keep track of the start date and the employee begins managing the department. The department has several locations.

2. The department controls a number of projects each of which has a unique name, unique number and a single location.

3. We store each employee names social security number, address ,salary, sex and dob. An employee is assigned one department but may work on several projects which are not necessarily controlled by the same department. We keep track of the department of each employee works on each project and for insurance purpose. We keep each dependents first name, sex, dob and relation.

b) Case study 2: Construct an E-R diagram for a hospital with a set of patients and a set of medical doctors. Associate with each patient a log of the various tests and examinations conducted. Also Construct appropriate tables for the ER Diagram



2 .Entity Relationship

Q2) Create one-to-many Relationship between Manager and Employee Relations Create following Relations with the given fields

a) EMPLOYEE

Emp Id (PK),

Emp Name (Should be in the upper case),

Department (Should be Finance, Purchase or Sales)

Salary

Mgrid

```
mysql> desc employee;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| EMP_ID     | int(11)     | NO   | PRI | NULL    |       |
| EMP_NAME   | varchar(20) | YES  |     | NULL    |       |
| DEPARTMENT | varchar(20) | YES  |     | NULL    |       |
| SALARY     | varchar(20) | YES  |     | NULL    |       |
| MGR_ID     | int(11)     | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
5 rows in set (0.13 sec)
```

b) MANAGER

Mgr id

Mgr Name

No. of Employees controlled

=ANS

```
mysql> DESC MANAGER;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| MGR_ID          | int(11)     | YES  | MUL | NULL    |       |
| MGR_NAME        | varchar(20) | YES  |     | NULL    |       |
| NO_OF_EMP_CONTR | varchar(30) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Using above table solve the following queries

a) Display details of all those employee whose salary is higher than Rs.50.

=ANS

```
mysql> select * from employee
    -> where salary > 50;
+--------+----------+------------+--------+--------+
| EMP_ID | EMP_NAME | DEPARTMENT | SALARY | MGR_ID |
+--------+----------+------------+--------+--------+
|    102 | EMP_2    | Purchase   | 60     |      1 |
|    105 | EMP_5    | Sales      | 65     |      2 |
|    106 | EMP_6    | Sales      | 55     |      2 |
+--------+----------+------------+--------+--------+
3 rows in set (0.00 sec)
```

b) Display the details of employees who ae working in Purchase department.

=ANS

```
mysql> select * from employee
    -> where department='Purchase';
+--------+----------+------------+--------+--------+
| EMP_ID | EMP_NAME | DEPARTMENT | SALARY | MGR_ID |
+--------+----------+------------+--------+--------+
|    101 | EMP_1    | Purchase   | 30     |      1 |
|    102 | EMP_2    | Purchase   | 60     |      1 |
|    103 | EMP_3    | Purchase   | 40     |      1 |
+--------+----------+------------+--------+--------+
3 rows in set (0.00 sec)
```

3. Normalization

Q1) Determine the functional dependencies. Remove partial dependency and transitive dependencies in given table. (i.e. convert it into 3NF).

Student = (RollNo, Name, Course_Code, Course_Name, Fees)

4. DDL Command

Q1) Creation of Database and table-DDL COMMAND Create a table called EMP with the following structure.

Name	Type
EMPNO	NUMBER(6)
ENAME	VARCHAR2(20)
JOB	VARCHAR2(20)
DEPTNO	NUMBER(3)
SAL	NUMBER(7,2)

Allow NULL for all columns except ENAME and JOB.

a) -Add a column experience to the emp table. experience numeric null allowed.

=ANS

```
mysql> ALTER TABLE EMP
    -> ADD EXPERIENCE INT(5);
Query OK, 0 rows affected (1.19 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DESC EMP;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| EMPNO      | int(6)      | YES  |     | NULL    |       |
| ENAME      | varchar(20) | YES  |     | NULL    |       |
| JOB        | varchar(10) | YES  |     | NULL    |       |
| DEPTNO     | int(3)      | YES  |     | NULL    |       |
| SAL        | int(7)      | YES  |     | NULL    |       |
| EXPERIENCE | int(5)      | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
6 rows in set (0.03 sec)
```

b) Modify the column width of the job field of emp table.

=**ANS**

```
mysql> ALTER TABLE EMP
    -> MODIFY JOB VARCHAR(5);
Query OK, 0 rows affected (1.06 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DESC EMP;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| EMPNO      | int(6)      | YES  |     | NULL    |       |
| ENAME      | varchar(20) | YES  |     | NULL    |       |
| JOB        | varchar(5)  | YES  |     | NULL    |       |
| DEPTNO     | int(3)      | YES  |     | NULL    |       |
| SAL        | int(7)      | YES  |     | NULL    |       |
| EXPERIENCE | int(5)      | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
6 rows in set (0.13 sec)
```

Q2) Create dept table with the following structure.

Name	Type
DEPTNO	NUMBER(2)
DNAME	VARCHAR2(10)
LOC	VARCHAR(10)

DEPTNO as the primary key

```
mysql> CREATE TABLE DEPT
    -> (DEPTNO INT(2) PRIMARY KEY,
    -> DNAME VARCHAR(10),
    -> LOC VARCHAR(10));
Query OK, 0 rows affected (0.29 sec)

mysql> DESC DEPT;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| DEPTNO | int(2)      | NO   | PRI | NULL    |       |
| DNAME  | varchar(10) | YES  |     | NULL    |       |
| LOC    | varchar(10) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
3 rows in set (0.08 sec)
```

a) Create the emp1 table with ename and empno, add constraints to check the empnovalue while entering (i.e) empno> 100.

```
mysql> CREATE TABLE EMP1
    -> (ENAME VARCHAR(255),
    -> EMPNO INT CHECK (EMPNO>100));
Query OK, 0 rows affected (0.31 sec)

mysql> DESC EMP1;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| ENAME | varchar(255) | YES  |     | NULL    |       |
| EMPNO | int(11)     | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.06 sec)
```

b) Drop a column experience to the emp table.

```
mysql> ALTER TABLE EMP
    -> DROP EXPERIENCE;
Query OK, 0 rows affected (0.33 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DESC EMP;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| EMPNO  | int(6)      | YES  |     | NULL    |       |
| ENAME  | varchar(20) | YES  |     | NULL    |       |
| JOB    | varchar(5)  | YES  |     | NULL    |       |
| DEPTNO | int(3)      | YES  |     | NULL    |       |
| SAL    | int(7)      | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

5 DML Command

Q1) Simple SQL Query-1-DML COMMAND

A) Write syntax of all DML Command

Ans=:- DML is the Data Manipulation Language .

1) INSERT

:- INSERT INTO Table_Name VALUES( );

2) UPDATE

:- UPDATE Table_Name SET Column_Name = “ ” WHERE Column_Name=“ ”;

3) DELETE

:- DELETE FROM Table_Name WHERE Column_Name=“ ”;

B) Create database where create following tables.

a) Emp(EMPNO INT, ENAME VARCHAR(20), JOB VARCHAR(10), DEPTNO INT, SAL INT(7)) Allow NULL for all columns except ename and job.

=EXAMPLE

```
mysql> SELECT * FROM EMP;
+-------+-------+----------+--------+------+
| EMPNO | ENAME | JOB      | DEPTNO | SAL  |
+-------+-------+----------+--------+------+
|  NULL | EMP 1 | Purchase |   NULL | NULL |
|  NULL | EMP 2 | Sales    |   NULL | NULL |
|  NULL | EMP 3 | Accounts |   NULL | NULL |
|  NULL | EMP 4 | Purchase |   NULL | NULL |
|  NULL | EMP 5 | Sales    |   NULL | NULL |
|  NULL | EMP 6 | Purchase |   NULL | NULL |
|  NULL | EMP 7 | Accounts |   NULL | NULL |
|  NULL | EMP 8 | Sales    |   NULL | NULL |
+-------+-------+----------+--------+------+
8 rows in set (0.01 sec)
```

b) Dept (DEPTNO INT, DNAME VARCHAR(10),LOC VARCHAR(10)) Deptno as the Primary key

Insert at least 8 records into tables and solve the following given queries using Emp and Dept key.

```
mysql> CREATE TABLE DEPT
    -> (DEPTNO INT PRIMARY KEY,
    -> DNAME VARCHAR(10),
    -> LOC VARCHAR(10));
Query OK, 0 rows affected (0.17 sec)

mysql> DESC DEPT;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| DEPTNO | int(11)     | NO   | PRI | NULL    |       |
| DNAME  | varchar(10) | YES  |     | NULL    |       |
| LOC    | varchar(10) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM DEPT;
+--------+--------+------+
| DEPTNO | DNAME  | LOC  |
+--------+--------+------+
|    101 | DEPT 1 | NULL |
|    102 | DEPT 2 | NULL |
|    103 | DEPT 3 | NULL |
|    104 | DEPT 4 | NULL |
|    105 | DEPT 5 | NULL |
|    106 | DEPT 6 | NULL |
|    107 | DEPT 7 | NULL |
|    108 | DEPT 8 | NULL |
+--------+--------+------+
8 rows in set (0.00 sec)
```

a) Create the Emp1 table with ename and empno, add constraint to check the empno value while entering (i.e) empno>100.

=EXAMPLE

```
mysql> CREATE TABLE EMP1
    -> (ENAME VARCHAR(255),
    -> EMPNO INT CHECK (EMPNO>100));
Query OK, 0 rows affected (0.31 sec)

mysql> DESC EMP1;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| ENAME | varchar(255) | YES  |     | NULL    |       |
| EMPNO | int(11)     | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.06 sec)
```

b) Update the EMP table to set the Salary of all employees to Rs.15000/- who are working as ASP.

=EXAMPLE

```
mysql> UPDATE EMP
    -> SET SAL=15000;
Query OK, 8 rows affected (0.06 sec)
Rows matched: 8  Changed: 8  Warnings: 0

mysql> SELECT * FROM EMP;
+-------+-------+----------+--------+-------+
| EMPNO | ENAME | JOB      | DEPTNO | SAL   |
+-------+-------+----------+--------+-------+
|  NULL | EMP 1 | Purchase |   NULL | 15000 |
|  NULL | EMP 2 | Sales    |   NULL | 15000 |
|  NULL | EMP 3 | Accounts |   NULL | 15000 |
|  NULL | EMP 4 | Purchase |   NULL | 15000 |
|  NULL | EMP 5 | Sales    |   NULL | 15000 |
|  NULL | EMP 6 | Purchase |   NULL | 15000 |
|  NULL | EMP 7 | Accounts |   NULL | 15000 |
|  NULL | EMP 8 | Sales    |   NULL | 15000 |
+-------+-------+----------+--------+-------+
8 rows in set (0.00 sec)
```

c) Delete only those who are working as lecturer.

d) List the records in the EMP table orderby Salary in ascending order.

=EXAMPLE

```
mysql> SELECT * FROM EMP
    -> ORDER BY SAL ASC;
+-------+-------+----------+--------+-------+
| EMPNO | ENAME | JOB      | DEPTNO | SAL   |
+-------+-------+----------+--------+-------+
|  NULL | EMP 1 | Purchase |   NULL | 15000 |
|  NULL | EMP 2 | Sales    |   NULL | 15000 |
|  NULL | EMP 3 | Accounts |   NULL | 15000 |
|  NULL | EMP 4 | Purchase |   NULL | 15000 |
|  NULL | EMP 5 | Sales    |   NULL | 15000 |
|  NULL | EMP 6 | Purchase |   NULL | 15000 |
|  NULL | EMP 7 | Accounts |   NULL | 15000 |
|  NULL | EMP 8 | Sales    |   NULL | 15000 |
+-------+-------+----------+--------+-------+
8 rows in set (0.00 sec)
```

e) Display total salary spent for each job category.

=EXAMPLE

```
mysql> SELECT * FROM SALARY_SPENT;
+--------------+--------------+
| JOB_CATEGORY | TOTAL_SALARY |
+--------------+--------------+
| Purchase     |        10000 |
| Sales        |         8000 |
| Accounts     |        18000 |
+--------------+--------------+
3 rows in set (0.00 sec)

mysql> SELECT SUM(TOTAL_SALARY) FROM SALARY_SPENT;
+--------------------+
| SUM(TOTAL_SALARY) |
+--------------------+
|              36000 |
+--------------------+
1 row in set (0.00 sec)
```

f) Add Constraints to the EMP table empno as the primary key and deptno as the foreign key.

g) Add Columns DOB to the emp table.

=EXAMPLE

```
mysql> DESC EMP;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| EMPNO  | int(11)     | NO   | PRI | NULL    |       |
| ENAME  | varchar(20) | YES  |     | NULL    |       |
| JOB    | varchar(10) | YES  |     | NULL    |       |
| DEPTNO | int(11)     | YES  |     | NULL    |       |
| SAL    | int(11)     | YES  |     | NULL    |       |
| DOB    | varchar(20) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

6. SQL Functions

Q1) Simple SQL Query2: SQL Functions

a) List all the aggregates Functions with Example?

1. COUNT:- Counts the Number of rows in a specified column or all rows in a table.
EXAMPLE:- SELECT COUNT(*) FROM Table_Name ;

2. SUM:- Calculate the sum of values in a Columns.
EXAMPLE:- SELECT SUM(Column_Name) FROM Table_Name ;

3. AVG:- Computes the Average of Values in a Columns.
EXAMPLE:- SELECT AVG(Column_Name) FROM Table_Name ;

4. MIN:- Retrieves the Minimum values in a Column.
EXAMPLE:- SELECT MIN(Column_Name) FROM Table_Name ;

5. MAX:- Retrieves the Maximum values in a Column.
EXAMPLE:- SELECT MAX(Column_Name) FROM Table_Name ;

6. GROUP_CONCAT:- Concatenates strings from multiple rows into a single string.
EXAMPLE:-SELECT GROUP_CONCAT(column_name SEPARATOR ' ')
FROM table_name;`

7. STDDEV:- Calculates the standard deviation of values in a column.
EXAMPLE:-SELECT STDDEV(column_name) FROM table_name;`

8. VARIANCE:-Computes the statistical variance of values in a column.
EXAMPLE:-SELECT VARIANCE(column_name) FROM table_name;`

9. FIRST:- Returns the first value in an ordered set.
EXAMPLE:-SELECT FIRST(column_name) FROM table_name ORDER BY column_name;`

10. LAST:-Returns the last value in an ordered set.
EXAMPLE:-SELECT LAST(column_name) FROM table_name ORDER BY column_name;`

b) List all the string function with EXAMPLE.

1. LENGTH( ):- Return the length of a string.

EXAMPLE:- SELECT LENGTH(‘Hello World’) ;

OUTPUT:- 11

2. UPPER( ):- Converts a string to uppercase.

EXAMPLE:- SELECT UPPER(‘hello, world’);

OUTPUT:- HELLO, WORLD

3. LOWER( ):- Converts a string to lowercase.

EXAMPLE:- SELECT LOWER('Hello, World');

OUTPUT:- hello, world

4. CONCAT( ):- Concatenates two or more strings.

EXAMPLE:- SELECT CONCAT('Hello', ' ', 'world!');

OUTPUT:- Hello world!

5. SUBSTRING( ):- Extracts a portion of a string based on a starting position and length.

EXAMPLE:- SELECT SUBSTRING('Hello, world!', 1, 5);

OUTPUT:- Hello

6. TRIM( ):- Removes specified characters or spaces from the beginning or end of string.

EXAMPLE:- SELECT TRIM(' Hello, world! ');

OUTPUT:- Hello, world!

7. REPLACE( ):- Replaces occurrences of a substring within a string with another substring.

EXAMPLE:- SELECT REPLACE('Hello, world!', 'Hello', 'Hi');

OUTPUT:- Hi, world!

8. CHARINDEX( ):- Returns the starting position of a substring within a string.

EXAMPLE:- SELECT CHARINDEX ('world', 'Hello, world!');

OUTPUT:- 7

9. LEFT( )*: Returns a specified number of characters from the left of a string.

EXAMPLE:- SELECT LEFT('Hello, world!', 5);

OUTPUT:- Hello

10. RIGHT( ):- Returns a specified number of characters from the right of a string.

EXAMPLE:- SELECT RIGHT('Hello, world!', 6);
OUTPUT:- world!

c) Using above EMP table solve the following queries.

1. Display all the details of the records whose employee name start With ‘A’.

```
mysql> select * from emp order by ename;
+-------+---------+----------+--------+-------+------+
| EMPNO | ENAME   | JOB      | DEPTNO | SAL   | DOB  |
+-------+---------+----------+--------+-------+------+
|   105 | Aman    | Purchase |   NULL | 15000 | NULL |
|   107 | Bob     | Sales    |   NULL | 15000 | NULL |
|   102 | Chandr  | Accounts |   NULL | 15000 | NULL |
|   106 | Nikhil  | Accounts |   NULL | 15000 | NULL |
|   101 | shakil  | Sales    |   NULL | 15000 | NULL |
|   104 | Suraj   | Sales    |   NULL | 15000 | NULL |
|   103 | Yougesh | Purchase |   NULL | 15000 | NULL |
+-------+---------+----------+--------+-------+------+
7 rows in set (0.08 sec)
```

2. Display all the details of the records whose employee name does not start With ‘A’

```
mysql> SELECT * FROM EMP ORDER BY ENAME DESC;
+-------+---------+----------+--------+-------+------+
| EMPNO | ENAME   | JOB      | DEPTNO | SAL   | DOB  |
+-------+---------+----------+--------+-------+------+
|   103 | Yougesh | Purchase |   NULL | 15000 | NULL |
|   104 | Suraj   | Sales    |   NULL | 15000 | NULL |
|   101 | shakil  | Sales    |   NULL | 15000 | NULL |
|   106 | Nikhil  | Accounts |   NULL | 15000 | NULL |
|   102 | Chandr  | Accounts |   NULL | 15000 | NULL |
|   107 | Bob     | Sales    |   NULL | 15000 | NULL |
|   105 | Aman    | Purchase |   NULL | 15000 | NULL |
+-------+---------+----------+--------+-------+------+
7 rows in set (0.05 sec)
```

3. Calculate the total and Average Salary amount of the EMP table.

```
mysql> SELECT SUM(SAL) AS TOTAL_SALARY, AVG(SAL) AS AVERAGE_SALARY FROM EMP;
+--------------+----------------+
| TOTAL_SALARY | AVERAGE_SALARY |
+--------------+----------------+
|       105000 |     15000.0000 |
+--------------+----------------+
1 row in set (0.08 sec)
```

4. Determine the max and min salary and rename the column as max_salary and min_salary

```
mysql> DESC EMP;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| EMPNO      | int(11)     | NO   | PRI | NULL    |       |
| ENAME      | varchar(20) | YES  |     | NULL    |       |
| JOB        | varchar(10) | YES  |     | NULL    |       |
| DEPTNO     | int(11)     | YES  |     | NULL    |       |
| MIN_SALARY | varchar(20) | YES  |     | NULL    |       |
| MAX_SALARY | varchar(20) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
6 rows in set (0.13 sec)
```

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

```
mysql> SELECT * FROM EMP;
+-------+---------+----------+--------+------------+------------+
| EMPNO | ENAME   | JOB      | DEPTNO | MIN_SALARY | MAX_SALARY |
+-------+---------+----------+--------+------------+------------+
|   101 | shakil  | Sales    |   NULL | 15000      | NULL       |
|   102 | Chandr  | Accounts |   NULL | 15000      | NULL       |
|   103 | Yougesh | Purchase |   NULL | 15000      | NULL       |
|   104 | Suraj   | Sales    |   NULL | 15000      | NULL       |
|   105 | Aman    | Purchase |   NULL | 15000      | NULL       |
|   106 | Nikhil  | Accounts |   NULL | 15000      | NULL       |
|   107 | Bob     | Sales    |   NULL | 15000      | NULL       |
+-------+---------+----------+--------+------------+------------+
7 rows in set (0.00 sec)

mysql> SELECT COUNT(DISTINCT JOB) AS JOB_TITLES FROM EMP;
+------------+
| JOB_TITLES |
+------------+
|          3 |
+------------+
1 row in set (0.00 sec)
```

6. Count the total records in the emp table.

```
mysql> SELECT * FROM EMP;
+-------+---------+----------+--------+------------+------------+
| EMPNO | ENAME   | JOB      | DEPTNO | MIN_SALARY | MAX_SALARY |
+-------+---------+----------+--------+------------+------------+
|   101 | shakil  | Sales    |   NULL | 15000      | NULL       |
|   102 | Chandr  | Accounts |   NULL | 15000      | NULL       |
|   103 | Yougesh | Purchase |   NULL | 15000      | NULL       |
|   104 | Suraj   | Sales    |   NULL | 15000      | NULL       |
|   105 | Aman    | Purchase |   NULL | 15000      | NULL       |
|   106 | Nikhil  | Accounts |   NULL | 15000      | NULL       |
|   107 | Bob     | Sales    |   NULL | 15000      | NULL       |
+-------+---------+----------+--------+------------+------------+
7 rows in set (0.01 sec)

mysql> SELECT COUNT(*) AS TOTAL_RECORDS FROM EMP;
+---------------+
| TOTAL_RECORDS |
+---------------+
|             7 |
+---------------+
1 row in set (0.00 sec)
```

7. Set Operations

Q1) Advance SQL Queries using Set Operations.

a). List all the set operators?

1. UNION:- Combines the result of two or more select statements, removing duplicate rows.

2. UNION ALL:- Similar to union but includes all rows, and includes duplicates.

3. INTERSECT:- Retrieves rows that are common between two select statements.

4. MINUS OR EXCEPT:- Return distinct rows from the first Select statement that are not present in the result of the second Select statement.

b) Using above emp table solve the following queries.

1. Display all the DEPT numbers available with the DEPT and EMP tables avoiding duplicates.

```
mysql> SELECT DEPTNO FROM DEPT
    -> UNION
    -> SELECT EMPNO FROM EMP;
+---------+
| DEPTNO  |
+---------+
|     101 |
|     102 |
|     103 |
|     104 |
|     105 |
|     106 |
|     107 |
|     108 |
+---------+
8 rows in set (0.02 sec)
```

2. Display all the DEPT numbers available with the DEPT and EMP tables.

3. Display all the DEPT numbers available in EMP and not in dept tables and vice versa.

8. Sub Query

Q1) Advance SQL queries using sub query.

a) Using above EMP table solve the following queries.

1. Display all employee names and salary whose salary is greater than minimum salary of the company and job titles starts with ‘M’.

```
mysql> select emp_name, salary from employee
    -> where salary > (select min(salary) from employee)
    -> and department like 'm%';
+----------+--------+
| emp_name | salary |
+----------+--------+
| Manish   | 40     |
+----------+--------+
1 row in set (0.02 sec)
```

2. Issue a query to find all the employee who work in the same job as Arjun.

```
+--------+----------+------------+--------+--------+
| EMP_ID | EMP_NAME | DEPARTMENT | SALARY | MGR_ID |
+--------+----------+------------+--------+--------+
|    101 | shakil   | maneger    | 30     |      1 |
|    102 | Manoj    | Purchase   | 60     |      1 |
|    103 | Mangesh  | Purchase   | 40     |      1 |
|    104 | Manish   | maneger    | 40     |      2 |
|    105 | Chandra  | Sales      | 65     |      2 |
|    106 | Arjun    | Sales      | 55     |      2 |
+--------+----------+------------+--------+--------+
6 rows in set (0.00 sec)

mysql> select * from employee
    -> where department = (select department from employee where emp_name = 'arjun' );
+--------+----------+------------+--------+--------+
| EMP_ID | EMP_NAME | DEPARTMENT | SALARY | MGR_ID |
+--------+----------+------------+--------+--------+
|    105 | Chandra  | Sales      | 65     |      2 |
|    106 | Arjun    | Sales      | 55     |      2 |
+--------+----------+------------+--------+--------+
```

3. Issue a query to display information about employees who earn more then any employee in dept.

9. Joins

Q1). Advanced SQL queries using Sub query

1. Display the employee details, departments that the departments are same in both EMP and DEPT.

```
mysql> select * from emp
    -> inner join dept
    -> on emp.empno = dept.deptno;
+-------+---------+------------+--------+---------+------------+
| EMPNO | ENAME   | Department | DEPTNO | DNAME   | Department |
+-------+---------+------------+--------+---------+------------+
|   101 | shakil  | Sales      |    101 | shakil  | sales      |
|   102 | Chandr  | Accounts   |    102 | chandra | Accounts   |
|   103 | Yougesh | Purchase   |    103 | Yougesh | Purchase   |
|   104 | Suraj   | Sales      |    104 | Suraj   | sales      |
|   105 | Aman    | Purchase   |    105 | Aman    | Purchase   |
+-------+---------+------------+--------+---------+------------+
5 rows in set (0.00 sec)
```

2. Display all the employee details, departments implementing a left outer join.

```
mysql> select * from emp
    -> left join dept
    -> on emp.empno = dept.deptno;
+-------+---------+------------+--------+---------+------------+
| EMPNO | ENAME   | Department | DEPTNO | DNAME   | Department |
+-------+---------+------------+--------+---------+------------+
|   101 | shakil  | Sales      |    101 | shakil  | sales      |
|   102 | Chandr  | Accounts   |    102 | chandra | Accounts   |
|   103 | Yougesh | Purchase   |    103 | Yougesh | Purchase   |
|   104 | Suraj   | Sales      |    104 | Suraj   | sales      |
|   105 | Aman    | Purchase   |    105 | Aman    | Purchase   |
|   106 | Nikhil  | Accounts   |   NULL | NULL    | NULL       |
|   107 | manoj   | Maneger    |   NULL | NULL    | NULL       |
+-------+---------+------------+--------+---------+------------+
7 rows in set (0.00 sec)
```

3. Display the employee name and department name in which they are working implementing a right outer join.

```
mysql> select e.ename, e.department, d.dname, d.department from emp as e
    -> right join dept as d
    -> on e.ename = d.dname;
+---------+------------+---------+------------+
| ename   | department | dname   | department |
+---------+------------+---------+------------+
| shakil  | Sales      | shakil  | sales      |
| Yougesh | Purchase   | Yougesh | Purchase   |
| Suraj   | Sales      | Suraj   | sales      |
| Aman    | Purchase   | Aman    | Purchase   |
| NULL    | NULL       | chandra | Accounts   |
+---------+------------+---------+------------+
```

4. Display the employee name and department name in which they are working implementing a full outer join

```
mysql> select emp.ename, dept.dname from emp
    -> left join dept
    -> on emp.empno = dept.deptno
    -> union
    -> select emp.ename, dept.dname from emp
    -> right join dept
    -> on emp.empno = dept.deptno;
+---------+---------+
| ename   | dname   |
+---------+---------+
| shakil  | shakil  |
| Chandr  | chandra |
| Yougesh | Yougesh |
| Suraj   | Suraj   |
| Aman    | Aman    |
| Nikhil  | NULL    |
| manoj   | NULL    |
+---------+---------+
```

10. PL-SQL

Q1) Advance SQL queries using PL-SQL

a) Write a PL/SQL program to swap two numbers.

```
mysql> set @num := 10; set @num2 := 20;
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> set @swap := @num1; set @num1 := @num2; set @num2 := @swap;
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> select concat("Swap Numbers: num1= ", @num1, ", num2= ", @num2);
+-----------------------------------------------------+
| concat("Swap Numbers: num1= ", @num1, ", num2= ", @num2) |
+-----------------------------------------------------+
| Swap Numbers: num1= 20, num2= 10                    |
+-----------------------------------------------------+
```

b) Write a PL/SQL program to find the largest of two numbers.

```
mysql> set @num1 = 10; set @num2 = 40; set @num3 = 0;
Query OK, 0 rows affected (0.01 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> call find(@num1, @num2, @num3);
Query OK, 0 rows affected (0.17 sec)

mysql> select @num3 as largest_number;
+----------------+
| largest_number |
+----------------+
|             40 |
+----------------+
1 row in set (0.00 sec)
```

11. Procedure And Function

Q1) Advanced SQL queries using PROCEDURE AND FUNCTION

a) Write syntax of procedure and function.

1) Procedure

DELIMITER //
CREATE PROCEDURE Procedure_name (Parameter_list)
BEGIN

- Procedure body (SQL Statements)

END //
DELIMITER ;

2).Function

CREATE FUNCTION function_name (parameter_list) RETURN return_type

BEGIN

- Function body (SQL Statements)
- RETURN Value_to _return;

END ;

b) Create a procedure to print the odd numbers from 1 to 10.

```
DELIMITER //

CREATE PROCEDURE PrintOddNumbers()
BEGIN
    DECLARE counter INT DEFAULT 1;

    WHILE counter <= 10 DO
        IF counter % 2 != 0 THEN
            SELECT counter AS OddNumber;
        END IF;
        SET counter = counter + 1;
    END WHILE;

END //
DELIMITER ;
```

```
CALL PrintOddNumbers();
```

```
+-----------+
| OddNumber |
+-----------+
|         1 |
|         3 |
|         5 |
|         7 |
|         9 |
+-----------+
```

12. Practice Question 1

Q1) Write a PL/SQL Program to find the total and average of 6 subject and display the grade.

```
SET SERVEROUTPUT ON;

DECLARE
    subject1 NUMBER := 85;
    subject2 NUMBER := 70;
    subject3 NUMBER := 92;
    subject4 NUMBER := 78;
    subject5 NUMBER := 64;
    subject6 NUMBER := 90;

    total NUMBER;
    average NUMBER;
    grade CHAR(1);
BEGIN
    total := subject1 + subject2 + subject3 + subject4 + subject5 + subject6;
    average := total / 6;

    IF average >= 90 THEN
        grade := 'A';
    ELSIF average >= 80 THEN
        grade := 'B';
    ELSIF average >= 70 THEN
        grade := 'C';
    ELSIF average >= 60 THEN
        grade := 'D';
    ELSE
        grade := 'F';
    END IF;

    DBMS_OUTPUT.PUT_LINE('Total marks: ' || total);
    DBMS_OUTPUT.PUT_LINE('Average marks: ' || average);
    DBMS_OUTPUT.PUT_LINE('Grade: ' || grade);
END;
/
```