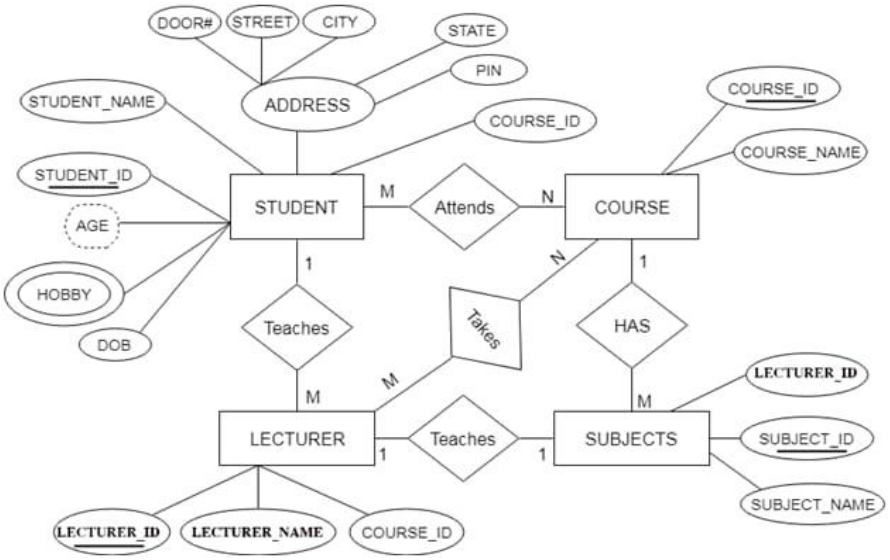
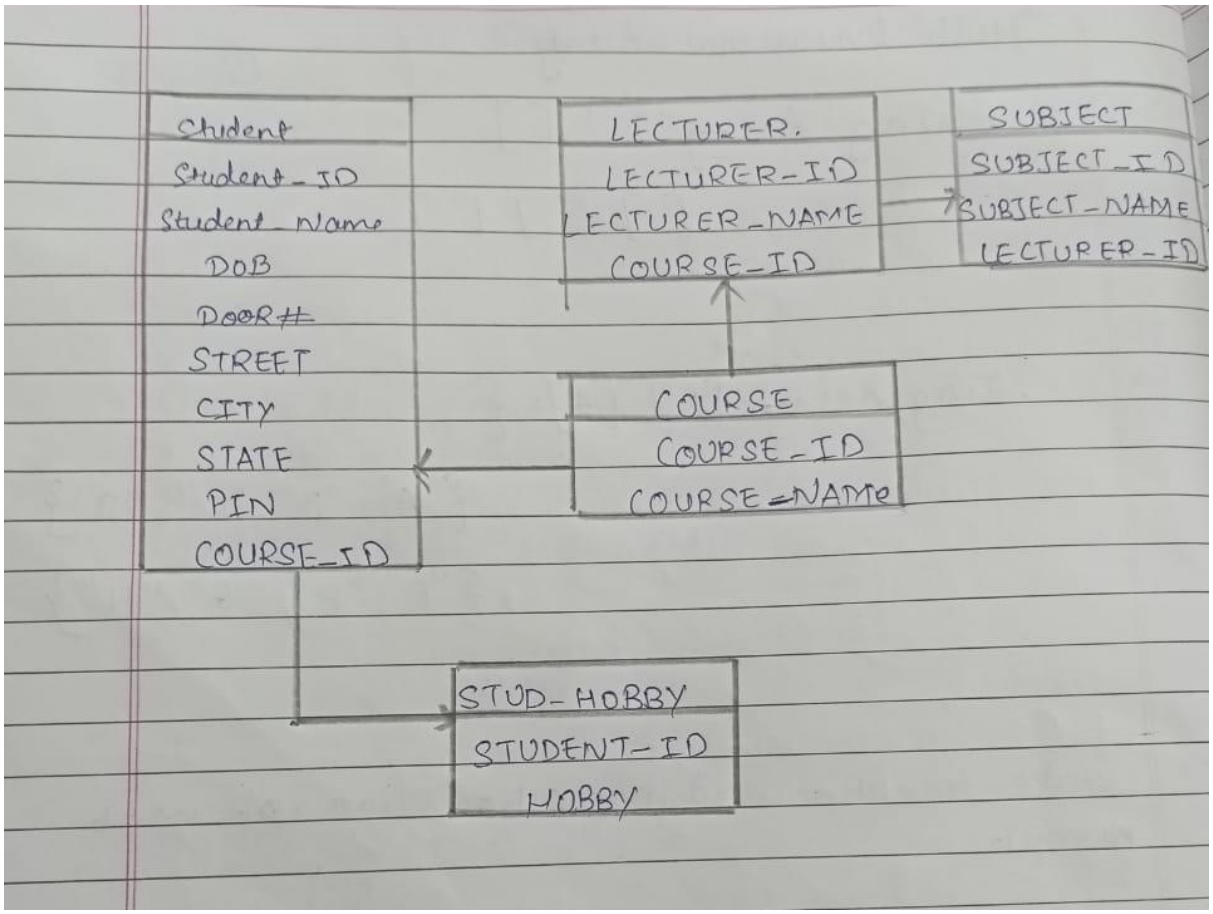
Name: Kaustubh Anant Rane
Roll No: CS23032

PRACTICAL NO 1

AIM: REDUCE THE ER TABLE DIAGRAM



OUTPUT:



PRACTICAL NO: 2

;

 # Viewing all databases

SHOW DATABASES;

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| arpitdb.1          |
| fycs23045          |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
6 rows in set (0.01 sec)
```

 # Creating a database

CREATE DATABASE FYCS23065

```
mysql> create database FYCS23045;
Query OK, 1 row affected (0.03 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| arpitdb.1          |
| fycs23045          |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
6 rows in set (0.01 sec)
```

 # To use the database

USE FYCS23065

```
mysql> use FYCS23045;
Database changed
```

 # Viewing all tables in a database

SHOW TABLES;

```
mysql> show tables;
Empty set (0.02 sec)
```

 # To see the table definition

DESC table1;

```
mysql> DESC table1;
ERROR 1146 (42S02): Table 'fycs23045.table1' doesn't exist
```

 # Creating Table Without Constraints

```
CREATE TABLE Persons (
 ID int ,
 LastName varchar(255) ,
 FirstName  varchar(255),
 Age int);
```

```
mysql> create table persons(
    -> id int,
    -> lastname varchar(255),
    -> firstname varchar(255),
    -> age int);
Query OK, 0 rows affected (0.07 sec)
```

 # Creating Table With Constraints
NOT NULL Constraint
SQL Statement - "ID", "LastName", and "FirstName" columns will NOT accept NULL values when the "Persons" table is created

```
CREATE TABLE Persons2 (
 ID int NOT NULL,
 LastName varchar(255) NOT NULL,
 FirstName varchar(255) NOT NULL,
```

Age
int);

```
mysql> CREATE TABLE PERSON1(
    -> ID INT NOT NULL,
    -> LastName varchar(255) NOT NULL,
    -> FirstName varchar(255) NOT NULL,
    -> Age int
    -> );
Query OK, 0 rows affected (0.12 sec)
```

 # UNIQUE constraint

SQL to creates a UNIQUE constraint on the "ID" column when the "Persons1" table is created:

When on single column-

```
CREATE TABLE Persons3 (
ID int NOT NULL,
 LastName varchar(255) NOT NULL,
 FirstName varchar(255),
 Age int,

 UNIQUE
(ID) );
```

```
mysql> CREATE TABLE PERSON2(
    -> ID int NOT NULL,
    -> LastName varchar(255) NOT NULL,
    -> FirstName varchar(255),
    -> Age int,
    -> UNIQUE(ID)
    -> );
Query OK, 0 rows affected (0.05 sec)
```

When on multiple columns-

```
CREATE TABLE Persons4 (
ID int NOT NULL,
 LastName varchar(255) NOT NULL,
```

```
 FirstName varchar(255),
 Age int,
 CONSTRAINT UC_Person UNIQUE
(ID,LastName) );
```

```
mysql> CREATE TABLE Person3(
    -> ID int NOT NULL,
    -> LastName varchar(255) NOT NULL,
    -> FirstName varchar(255),
    -> Age int,
    -> CONSTRAINT UC_Person UNIQUE(ID,LastName)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

Default value constraint

SQL to set a DEFAULT value for the "City" column when the "Persons" table is created:

```
CREATE TABLE Persons5 (
 ID int NOT NULL,
 LastName varchar(255) NOT NULL,
 FirstName varchar(255),
 Age int,

 City varchar(255) DEFAULT
'Sandnes' );
```

```
mysql> create table person6(
    -> ID int NOT NULL,
    -> LastName varchar(255) NOT NULL,
    -> FirstName varchar(255),
    -> Age int,
    -> City varchar(255) DEFAULT 'Sandnes'
    -> );
Query OK, 0 rows affected (0.10 sec)
```

CHECK Constraint -

SQL to creates a CHECK constraint on the "Age" column when the "Persons" table is created. The CHECK constraint ensures that thage of a person must be 18, or older:

```
CREATE TABLE Persons3 (

 ID int NOT NULL,
 LastName varchar(255) NOT NULL,
 FirstName varchar(255),
 Age int,

 CHECK
(Age>=18) );
```

```
mysql> CREATE TABLE Persons5(
    -> ID int NOT NULL,
    -> LastName varchar(255) NOT NULL,
    -> FirstName varchar(255),
    -> Age int,
    -> CHECK (Age>=18)
    -> );
Query OK, 0 rows affected (0.31 sec)
```

For defining a CHECK constraint on multiple columns, use thefollowing SQL syntax:

```
CREATE TABLE Persons4 (
 ID int NOT NULL,
 LastName varchar(255) NOT NULL,
 FirstName varchar(255),
 Age int,

 City varchar(255),

 CONSTRAINT CHK_Person CHECK (Age>=18 AND
City='Sandnes') );
```

```
mysql> CREATE TABLE Persons6(
    -> ID int NOT NULL,
    -> LastName varchar(255) NOT NULL,
    -> FirstName varchar(255),
    -> Age int,
    -> City varchar(255),
    -> CONSTRAINT CHK_Person CHECK (Age>=18 AND City='Sandnes'
    -> )
    -> );
Query OK, 0 rows affected (0.34 sec)
```

PRIMARY KEY Constraint -

SQL creates a PRIMARY KEY on the "ID" column when the "Persons" table is created:

```
CREATE TABLE Persons2 (
 ID int NOT NULL,
 LastName varchar(255) NOT NULL,
 FirstName varchar(255),
 Age int,

 PRIMARY KEY
(ID) );
```

```
mysql> CREATE TABLE Persons7(
    -> ID int NOT NULL,
    -> LastName varchar(255) NOT NULL,
    -> FirstName varchar(255),
    -> Age int,
    -> PRIMARY KEY(ID)
    -> );
Query OK, 0 rows affected (0.34 sec)
```

For defining a PRIMARY KEY constraint on multiple columns, use the following SQL syntax:

```
CREATE TABLE PersonID int NOT NULL,

 LastName varchar(255) NOT NULL,
 FirstName varchar(255),
 Age int,

 CONSTRAINT PK_Person PRIMARY KEY (ID,LastName) )
```

```
mysql> CREATE TABLE Persons8(
    -> ID int NOT NULL,
    -> LastName varchar(255) NOT NULL,
    -> FirstName varchar(255),
    -> Age int,
    -> CONSTRAINT PK_Person PRIMARY KEY (ID,LastName)
    -> );
Query OK, 0 rows affected (0.36 sec)
```

FOREIGN KEY Constraint

SQL creates a FOREIGN KEY on the "PersonID" column when the "Orders" table is created:

```
CREATE TABLE Orders (

 OrderID int NOT NULL,


 OrderNumber int NOT NULL,
 PersonID int,
 PRIMARY KEY (OrderID),

 FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)

)
```

```
mysql> CREATE TABLE Orders1 (
    ->  OrderID int NOT NULL,
    ->  OrderNumber int NOT NULL,
    ->  PersonID int,
    ->  PRIMARY KEY (OrderID),
    ->  CONSTRAINT FK_PersonOrder FOREIGN KEY (PersonID)
    ->  REFERENCES Person8(PersonID)
    -> );
Query OK, 0 rows affected (0.07 sec)
```

PRACTICAL NO: 3

1) Altering a Table –

● MySQL ALTER statement is used when you want to change the name of your table or any table field.

● It is also used to add or delete an existing column in a table.

● It is also used to delete an existing constraint on the table.

● The ALTER statement is always used with "ADD", "DROP" and "MODIFY" commands according to the situation.

Syntax:
● ALTER TABLE table_name
● ADD new_column_name column_definition
● [FIRST | AFTER column_name];
To add column
ALTER TABLE Student;

ADD StudDIV varchar(40);

```
mysql> ALTER TABLE Student
mysql> ALTER TABLE Student
    -> MODIFY COLUMN Age VARCHAR(40);
Query OK, 0 rows affected (0.10 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DESC Student;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| StuID     | int         | YES  |     | NULL    |       |
| FirstName | varchar(20) | YES  |     | NULL    |       |
| LastName  | varchar(20) | YES  |     | NULL    |       |
| Age       | varchar(40) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

To add multiple column

ALTER TABLE STUDENT
ADD Studaddress varchar(100) NOT NULL

ADD StudMarks int(100) NOT NULL
ABLE Student

```
mysql> DESC Student;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| StuID       | int         | YES  |     | NULL    |       |
| FirstName   | varchar(20) | YES  |     | NULL    |       |
| LastName    | varchar(20) | YES  |     | NULL    |       |
| Age         | varchar(40) | YES  |     | NULL    |       |
| StudAddress | varchar(50) | NO   |     | NULL    |       |
| StudMarks   | int         | NO   |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
6 rows in set (0.01 sec)
```

To modify an existing column

ALTER TABLE Student

MODIFY COLUMN Age VARCHAR(40);

```
mysql> ALTER TABLE Student
    -> MODIFY COLUMN Age VARCHAR(40);
Query OK, 0 rows affected (0.10 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DESC Student;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| StuID     | int         | YES  |     | NULL    |       |
| FirstName | varchar(20) | YES  |     | NULL    |       |
| LastName  | varchar(20) | YES  |     | NULL    |       |
| Age       | varchar(40) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

To drop an existing column
ALTER TABLE Student
DROP COLUMN StudDiv;

```
mysql> ALTER TABLE Student
    -> DROP COLUMN StudDIV;
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DESC Student;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| StuID     | int         | YES  |     | NULL    |       |
| FirstName | varchar(20) | YES  |     | NULL    |       |
| LastName  | varchar(20) | YES  |     | NULL    |       |
| Age       | int         | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

To rename a n existing column

ALTER TABLE Student

CHANGE COLUMN Age Division Varchar(20) NOT NULL;

```
mysql> ALTER TABLE Student
    -> MODIFY COLUMN Age VARCHAR(40);
Query OK, 0 rows affected (0.10 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DESC Student;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| StuID     | int         | YES  |     | NULL    |       |
| FirstName | varchar(20) | YES  |     | NULL    |       |
| LastName  | varchar(20) | YES  |     | NULL    |       |
| Age       | varchar(40) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

2) Truncating Tables

● The TRUNCATE statement in MySQL removes the complete data without removing its structure.
● It is a part of DDL or data definition language command.

● Generally, we use this command when we want to delete an entire data from a table without removing the table structure.

Syntax

The following syntax explains the TRUNCATE command to remove data from the table:

TRUNCATE [TABLE] table_name;
Example#TRUNCATE TABLE customer;

```
mysql> INSERT INTO students(Age, FirstName, LastName, marks) VALUES(17, 'ROHAN', 'SHINDE', 85);
Query OK, 1 row affected (0.02 sec)

mysql> Select * from students;
+-----+-----------+----------+-------+
| Age | FirstName | LastName | marks |
+-----+-----------+----------+-------+
|  17 | ROHAN     | SHINDE   |    85 |
+-----+-----------+----------+-------+
1 row in set (0.00 sec)

mysql> truncate students;
Query OK, 0 rows affected (0.06 sec)

mysql> Select * from students;
Empty set (0.01 sec)
```

3) Dropping Tables

● MYSQL uses a Drop Table statement to delete the existing table.

● This statement removes the complete data of a table along with the whole structure or definition permanently from the database.

● we cannot recover the lost data after deleting it

Syntax

The following are the syntax to remove the table in MySQL:
mysql> DROP TABLE table_name;

OR,
mysql> DROP TABLE schema_name.table_name;
Example #DROP TABLE orders;

```
mysql> select * from students;
+-----+-----------+------------+-------+
| Age | FirstName | LastName   | marks |
+-----+-----------+------------+-------+
|  18 | Rohan     | TAWDE      |    79 |
|  19 | vivek     | yadav      |    89 |
|  18 | rinki     | chaurasiya |    91 |
+-----+-----------+------------+-------+
3 rows in set (0.00 sec)

mysql> DROP TABLE students;
Query OK, 0 rows affected (0.03 sec)

mysql> select * from students;
ERROR 1146 (42S02): Table 'arpitdb.1.students' doesn't exist
mysql>
```

4) Renaming Tables
ALTER TABLE Student
RENAME TO Student

5) Backing up / Restoring a Database

```
mysql> ALTER TABLE student
    -> RENAME to student1;
Query OK, 0 rows affected (0.04 sec)

mysql> show tables;
+----------------------+
| Tables_in_arpitdb.1  |
+----------------------+
| employee table       |
| student1             |
+----------------------+
2 rows in set (0.00 sec)
```

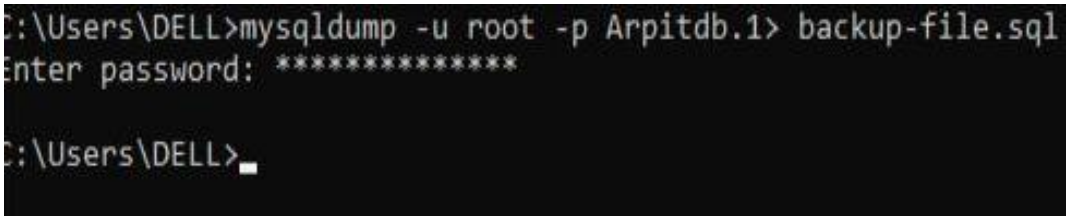
- Take backup using mysqldump

- mysqldump command can be executed from mysql prompt.

● For all the code for mysqldump commands bellow, the testDatabase is the name of the database.

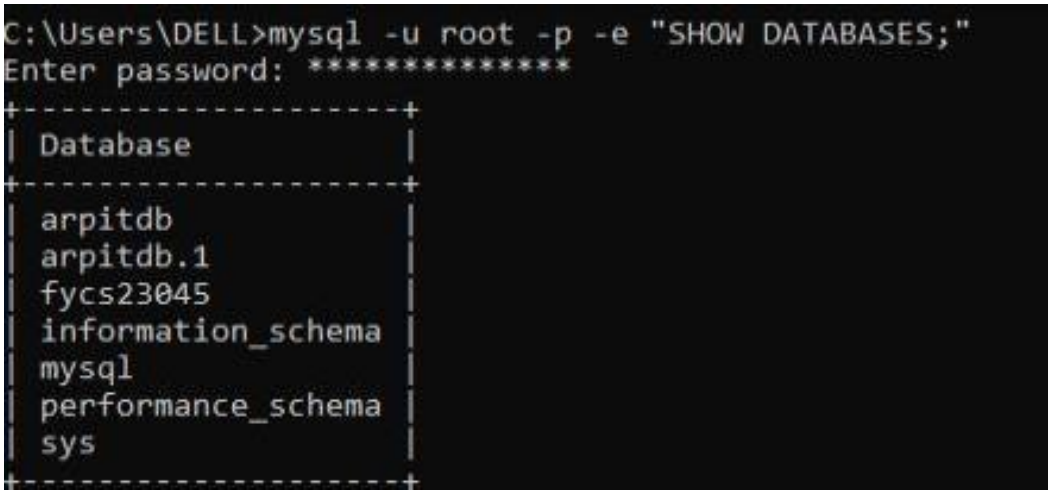
1) Take backup of a database

mysqldump testDatabase > back-file.sql;



2) Restore a database

mysql testDatabase < back-up.sql;



3) Copying data from one server to another

mysqldump –opt database | mysql –host=remote_host –C database

4) Dump several databases with single command

mysqldump --databases database1 [database2 ...] > backup_of_databases.sql

5) Dump all databases using --all-databases option

mysqldump –all-databases > backup_of_all_databases.sql

PRACTICAL NO: 4

Creating Tables (With Constraints)-

Q.) Create an employee table having columns as employee number , employee name , date of joining and salary with the column values not excepting null values and employee number as primary key.

A.)

Mysql>create table Emp2 (EmpNo integer(5) NOT NULL,
EmpName varchar(20) NOT NULL,
JoinDate date NOT NULL,
EmpSal integer(8) NOT NULL,
PRIMARY KEY(EmpNo));
Mysql>show tables;
Mysql> desc Emp2;

```
mysql> CREATE TABLE Emp2(
    -> EmpNo int(5) NOT NULL,
    -> EmpName Varchar(20) NOT NULL,
    -> JoinDate DATE NOT NULL,
    -> EmpSal int(9) NOT NULL,
    -> PRIMARY KEY(EmpNo)
    -> );
Query OK, 0 rows affected, 2 warnings (0.07 sec)

mysql> SHOW TABLES;
+--------------------+
| Tables_in_arpitdb |
+--------------------+
| emp2               |
+--------------------+
1 row in set (0.01 sec)

mysql> DESC emp2;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| EmpNo     | int          | NO   | PRI | NULL    |       |
| EmpName   | varchar(20)  | NO   |     | NULL    |       |
| JoinDate  | date         | NO   |     | NULL    |       |
| EmpSal    | int          | NO   |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

Inserting Records in a Table

Insert at least 10 records to emp1 table, with at least 2 employees having salary less than 10000 and one employee having salary as 10000

```
mysql> INSERT INTO emp3 (emp_id, emp_name, emp_salary) VALUES
    -> (1, 'Employee1', 9500),
    -> (2, 'Employee2', 9800),
    -> (3, 'Employee3', 10000),
    -> (4, 'Employee4', 10500),
    -> (5, 'Employee5', 11000),
    -> (6, 'Employee6', 11500),
    -> (7, 'Employee7', 12000),
    -> (8, 'Employee8', 12500),
    -> (9, 'Employee9', 13000),
    -> (10, 'Employee10', 13500);
Query OK, 10 rows affected (0.02 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM emp3;
+--------+------------+------------+
| emp_id | emp_name   | emp_salary |
+--------+------------+------------+
|      1 | Employee1  |    9500.00 |
|      2 | Employee2  |    9800.00 |
|      3 | Employee3  |   10000.00 |
|      4 | Employee4  |   10500.00 |
|      5 | Employee5  |   11000.00 |
|      6 | Employee6  |   11500.00 |
|      7 | Employee7  |   12000.00 |
|      8 | Employee8  |   12500.00 |
|      9 | Employee9  |   13000.00 |
|     10 | Employee10 |   13500.00 |
+--------+------------+------------+
10 rows in set (0.01 sec)
```

 # Insert One record at a time –

Mysql> insert into Emp1 (EmpNo , EmpName , JoinDate ,EmpSal)
Values (101,’Rajesh’,’2021-10-23’,10000);
Mysql>select * from emp1;

```
mysql> SELECT * FROM Emp1;
+-------+---------+------------+----------+
| EmpNo | EmpName | JoinDate   | EmpSal   |
+-------+---------+------------+----------+
|   101 | Rajesh  | 2021-10-23 | 10000.00 |
+-------+---------+------------+----------+
1 row in set (0.00 sec)
```

 # Insert multiple records at a time –
Mysql> insert into Emp1 values
(101,'Raju','2021-10-23',10000),
(102,'Kamlesh','2020-01-01',19000),

(103,'Ramesh','1998-03-07',20000);

Mysql>select * from emp1;

```
mysql> SELECT * FROM Emp1;
+-------+---------+------------+----------+
| EmpNo | EmpName | JoinDate   | EmpSal   |
+-------+---------+------------+----------+
|   101 | Rajesh  | 2021-10-23 | 10000.00 |
|   102 | Raju    | 2021-10-23 | 10000.00 |
|   103 | Kamlesh | 2020-01-01 | 19000.00 |
|   104 | Ramesh  | 1998-03-07 | 20000.00 |
+-------+---------+------------+----------+
4 rows in set (0.00 sec)
```

 # Updating Records in a Table

A: Change the salary for employee Rajesh as 50000
Mysql> update emp1 set empsal = 50000 where
empname='Rajesh';
Mysql>select * from emp1;

```
mysql> UPDATE Emp1
    -> SET EmpSal = 50000
    -> WHERE EmpName = 'Rajesh';
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM Emp1 WHERE EmpName = 'Rajesh';
+-------+---------+------------+----------+
| EmpNo | EmpName | JoinDate   | EmpSal   |
+-------+---------+------------+----------+
|   101 | Rajesh  | 2021-10-23 | 50000.00 |
+-------+---------+------------+----------+
1 row in set (0.00 sec)
```

B: Change the date of joining to 2019-04-15 for employee 107
Mysql> update emp1 set joindate='2019-04-15' where
empno=103;
Mysql>select * from emp1;

```
mysql> UPDATE Emp1
    -> SET JoinDate = '2019-04-15'
    -> WHERE EmpNo = 103;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM Emp1 WHERE EmpNo = 103;
+-------+---------+------------+----------+
| EmpNo | EmpName | JoinDate   | EmpSal   |
+-------+---------+------------+----------+
|   103 | Kamlesh | 2019-04-15 | 19000.00 |
+-------+---------+------------+----------+
1 row in set (0.00 sec)
```

C**: Increase the salaries to 15000 for all the employees whose salary is less than or equal to 10000**

Mysql> update emp1 set EmpSal = 15000
where empsal <= 10000;
Mysql>select * from emp1;

```
mysql> UPDATE Emp1
    -> SET EmpSal = 15000
    -> WHERE EmpSal <= 10000;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM Emp1 WHERE EmpSal = 15000;
+-------+---------+------------+----------+
| EmpNo | EmpName | JoinDate   | EmpSal   |
+-------+---------+------------+----------+
|   102 | Raju    | 2021-10-23 | 15000.00 |
+-------+---------+------------+----------+
1 row in set (0.00 sec)
```

Deleting Records in a Table

Q.) write a query to remove the employee record for employee no 104 from employee table

A.)
Mysql>select * from emp1;

Mysql> delete from emp1 where empno=104;
Mysql>select * from emp1;

```
mysql> DELETE FROM emp1 WHERE empNo = 104;
Query OK, 1 row affected (0.02 sec)

mysql> SELECT * FROM emp1;
+-------+---------+------------+----------+
| EmpNo | EmpName | JoinDate   | EmpSal   |
+-------+---------+------------+----------+
|   101 | Rajesh  | 2021-10-23 | 50000.00 |
|   102 | Raju    | 2021-10-23 | 15000.00 |
|   103 | Kamlesh | 2019-04-15 | 19000.00 |
+-------+---------+------------+----------+
3 rows in set (0.00 sec)
```

 # Create a customers table with the following columns and data.

+----+----------+-----+-----------+----------+
| **ID** | **NAME** | **AGE** | **ADDRESS** | **SALARY** |
+----+----------+-----+-----------+----------+
1	**Ramesh**	**32**	**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**

```
mysql> CREATE TABLE Customers(
    -> ID INT PRIMARY KEY,
    -> Name Varchar(255),
    -> Age INT,
    -> Address Varchar(255),
    -> Salary DECIMAL(10,2)
    -> );
Query OK, 0 rows affected (0.13 sec)

mysql> INSERT INTO customers VALUES
    -> (1, 'Ramesh', 32, '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);
Query OK, 7 rows affected (0.03 sec)
Records: 7  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM Customers;
+----+----------+------+-----------+----------+
| ID | Name     | Age  | Address   | Salary   |
+----+----------+------+-----------+----------+
|  1 | Ramesh   |   32 | 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 |
+----+----------+------+-----------+----------+
7 rows in set (0.01 sec)
```

Perform Simple Queries with Where Operators

1) fetch the ID, Name and Salary fields from the CUSTOMERS table, where the salary is greater than 2000

SQL> SELECT ID, NAME, SALARY
FROM CUSTOMERS
WHERE SALARY > 2000

```
mysql> SELECT ID, NAME, SALARY
    -> FROM CUSTOMERS
    -> WHERE SALARY > 2000;
+----+----------+----------+
| ID | NAME     | SALARY   |
+----+----------+----------+
|  4 | Chaitali |  6500.00 |
|  5 | Hardik   |  8500.00 |
|  6 | Komal    |  4500.00 |
|  7 | Muffy    | 10000.00 |
+----+----------+----------+
4 rows in set (0.01 sec)
```

2) fetch the ID, Name and Salary fields from the CUSTOMERS table for a customer with the name Hardik.

SQL> SELECT ID, NAME, SALARY
FROM CUSTOMERS
WHERE NAME = 'Hardik';

```
mysql> SELECT ID, NAME, SALARY
    -> FROM CUSTOMERS
    -> WHERE NAME='Hardik';
+----+--------+---------+
| ID | NAME   | SALARY  |
+----+--------+---------+
|  5 | Hardik | 8500.00 |
+----+--------+---------+
1 row in set (0.01 sec)
```

3) fetch id,name and salary form customers whose's name starts with k

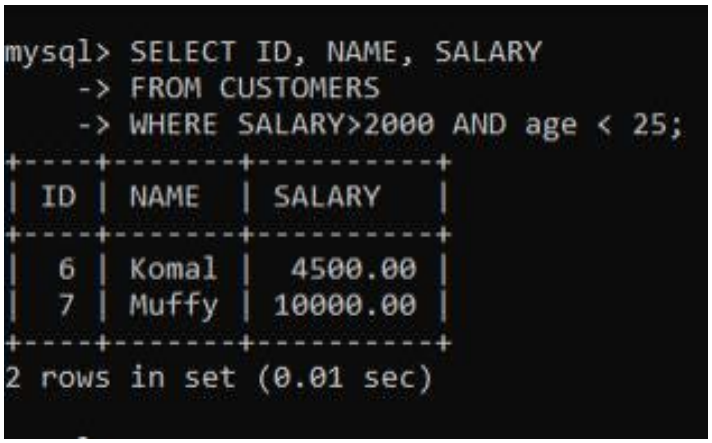
SQL> SELECT ID, NAME, SALARY
FROM CUSTOMERS
WHERE NAME LIKE 'K%';

```
mysql> SELECT ID, NAME, SALARY
    -> FROM CUSTOMERS
    -> WHERE NAME LIKE'K%';
+----+---------+---------+
| ID | NAME    | SALARY  |
+----+---------+---------+
|  2 | Khilan  | 1500.00 |
|  3 | kaushik | 2000.00 |
|  6 | Komal   | 4500.00 |
+----+---------+---------+
3 rows in set (0.01 sec)
```

Perform Where with Keywords and Logical Operators

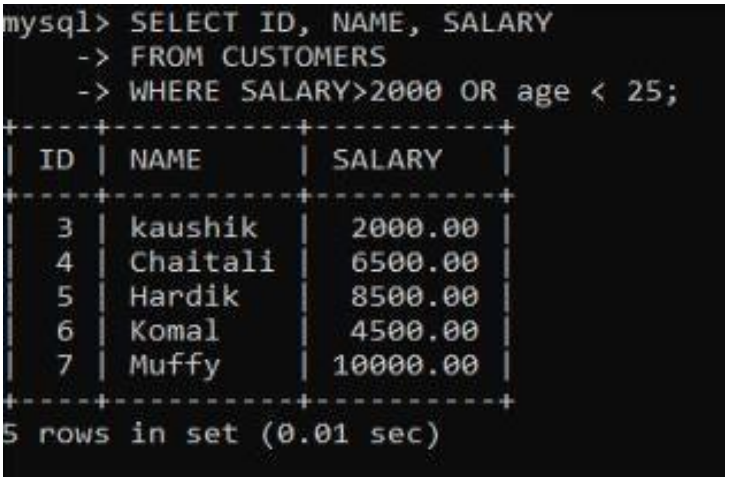
1) (AND) fetch the ID, Name and Salary fields from the CUSTOMERS table, where the salary is greater than 2000 and the age is less than 25 years –

SQL> SELECT ID, NAME, SALARY
FROM CUSTOMERS
WHERE SALARY > 2000 AND age < 25;



2) (OR) fetch the ID, Name and Salary fields from the CUSTOMERS table, where the salary is greater than 2000 OR the age is less than 25 years.

SQL> SELECT ID, NAME, SALARY
FROM CUSTOMERS
WHERE SALARY > 2000 OR age < 25;



3) (IN) Fetch customer name and address from CUSTOMERS table , where address IN Bhopal or MUMBAI

SQL> SELECT name, address
FROM customers
WHERE Address IN ('BHOPAL', 'MUMBAI');

```
mysql> SELECT Name, Address
    -> FROM CUSTOMERS
    -> WHERE Address IN('BHOPAL','MUMBAI');
+----------+---------+
| Name     | Address |
+----------+---------+
| Chaitali | Mumbai  |
| Hardik   | Bhopal  |
+----------+---------+
2 rows in set (0.00 sec)
```

4) (NOT) Fetch customer name and address from CUSTOMERS table , where address is not indore or MUMBAI

SQL> SELECT name, address
FROM customers
WHERE Address NOT IN ('INDORE', 'MUMBAI');

```
mysql> SELECT Name, Address
    -> FROM CUSTOMERS
    -> WHERE Address NOT IN('INDORE','MUMBAI');
+---------+-----------+
| Name    | Address   |
+---------+-----------+
| Ramesh  | Ahmedabad |
| Khilan  | Delhi     |
| kaushik | Kota      |
| Hardik  | Bhopal    |
| Komal   | MP        |
+---------+-----------+
5 rows in set (0.01 sec)
```

5) (BETWEEN) fetch the customer details where customer id between 4 and 7.

SQL> SELECT *
FROM customers
WHERE id BETWEEN 4 AND 7;

```
mysql> SELECT Name, Address
    -> ^C
mysql> SELECT *
    -> FROM CUSTOMERS
    -> WHERE id BETWEEN 4 AND 7;
+----+----------+------+---------+----------+
| ID | Name     | Age  | Address | Salary   |
+----+----------+------+---------+----------+
|  4 | Chaitali |   25 | Mumbai  |  6500.00 |
|  5 | Hardik   |   27 | Bhopal  |  8500.00 |
|  6 | Komal    |   22 | MP      |  4500.00 |
|  7 | Muffy    |   24 | Indore  | 10000.00 |
+----+----------+------+---------+----------+
4 rows in set (0.01 sec)
```

Simple Queries with Aggregate functions

Execute the below statement to create an Works table:

CREATE TABLE Works(

name varchar(45) NOT NULL,
gender varchar(6) NOT NULL,
occupation varchar(35) NOT NULL,
working_date date,
working_hours varchar(10)

);

```
mysql> CREATE TABLE Works(
    ->     name varchar(45) NOT NULL,
    ->     gender varchar(6) NOT NULL,
    ->     occupation varchar(35) NOT NULL,
    ->     working_date date,
    ->     working_hours varchar(10)
    -> );
Query OK, 0 rows affected (0.19 sec)

mysql> SELECT * FROM Works;
Empty set (0.02 sec)

mysql> DESC Works;
+--------------+------------+------+-----+---------+-------+
| Field        | Type       | Null | Key | Default | Extra |
+--------------+------------+------+-----+---------+-------+
| name         | varchar(45)| NO   |     | NULL    |       |
| gender       | varchar(6) | NO   |     | NULL    |       |
| occupation   | varchar(35)| NO   |     | NULL    |       |
| working_date | date       | YES  |     | NULL    |       |
| working_hours| varchar(10)| YES  |     | NULL    |       |
+--------------+------------+------+-----+---------+-------+
5 rows in set (0.03 sec)
```

Insert data as:

INSERT INTO Works VALUES

('Robin', ‘Male’, 'Scientist', '2020-10-04',12),
('Warner', ‘Male’, 'Engineer', '2020-10-04',10),
('Patricia', ‘Female’, 'Actor', '2020-10-04',13),
('Marco', ‘Male’, 'Doctor', '2020-10-04',14),
('Brayden', ‘Male’, 'Teacher', '2020-10-04',12),
('Anita', ‘Female’, 'Business', '2020-10-04',11),
('Roshani', ‘Female’, 'Analyst', '2021-12-14',10),
('Wishu', ‘Male’, 'Engineer', '2022-11-24',8),
('Preet', ’Male’, 'Anchor', '2020-04-01',9), ('Maria',
‘Female’, 'Doctor', '2023-01-01',8), ('Badri',
‘Male’, 'Teacher', '2021-06-01',16), ('Ananda',
‘Male’, 'Steward', '2022-08-05',12), (‘Roshan',
‘Male’, 'Engineer', '2020-03-23',10),
('Preesha', ’Female’, 'Actor', '2021-04-09',13),
('Madan', ‘Male’, 'Doctor', '2022-10-04',10),
('Brajesh', ‘Male’, 'Teacher', '2023-09-01',08)
('Anindita', ‘Female’, 'Business', '2020-09-01',10),
('Roshani', ‘Female’, 'Analyst', '2021-12-14',10),
('Watson',’Male’, 'Engineer', '2022-05-14',9),
('Prachi',’Female’, 'Anchor', '2020-04-01',9),
('Mehul',’Female’, 'Doctor', '2022-04-01',10);
Sql> SELECT * FROM works;

```
mysql> SELECT * FROM Works;
+----------+--------+------------+--------------+---------------+
| name     | gender | occupation | working_date | working_hours |
+----------+--------+------------+--------------+---------------+
| Robin    | Male   | Scientist  | 2020-10-04   | 12            |
| Warner   | Male   | Engineer   | 2020-10-04   | 10            |
| Patricia | Female | Actor      | 2020-10-04   | 13            |
| Marco    | Male   | Doctor     | 2020-10-04   | 14            |
| Brayden  | Male   | Teacher    | 2020-10-04   | 12            |
| Anita    | Female | Business   | 2020-10-04   | 11            |
| Roshani  | Female | Analyst    | 2021-12-14   | 10            |
| Wishu    | Male   | Engineer   | 2022-11-24   | 8             |
| Preet    | Male   | Anchor     | 2020-04-01   | 9             |
| Maria    | Female | Doctor     | 2023-01-01   | 8             |
| Badri    | Male   | Teacher    | 2021-06-01   | 16            |
| Ananda   | Male   | Steward    | 2022-08-05   | 12            |
| Roshan   | Male   | Engineer   | 2020-03-23   | 10            |
| Preesha  | Female | Actor      | 2021-04-09   | 13            |
| Madan    | Male   | Doctor     | 2022-10-04   | 10            |
| Brajesh  | Male   | Teacher    | 2023-09-01   | 08            |
| Anindita | Female | Business   | 2020-09-01   | 10            |
| Roshani  | Female | Analyst    | 2021-12-14   | 10            |
| Watson   | Male   | Engineer   | 2022-05-14   | 9             |
| Prachi   | Female | Anchor     | 2020-04-01   | 9             |
| Mehul    | Female | Doctor     | 2022-04-01   | 10            |
+----------+--------+------------+--------------+---------------+
21 rows in set (0.00 sec)
```

1) (COUNT)get the total number of employee in the works table

sql> select count(*) from works;

```
mysql> SELECT COUNT(*) FROM Works;
+----------+
| COUNT(*) |
+----------+
|       21 |
+----------+
1 row in set (0.05 sec)
```

2) (SUM)calculate the total number of working hours of all employees in the works table

sql> select sum(working_hours) as “Total Working Hours” from works;

```
mysql> SELECT SUM(working_hours) AS "Total Working Hours" FROM Works;
+---------------------+
| Total Working Hours |
+---------------------+
|                 224 |
+---------------------+
1 row in set (0.01 sec)
```

3) (AVG)get the average working hours of all employees in the works table

sql> select avg(working_hours) as “Average Working Hours” from works;

```
mysql> SELECT AVG(working_hours) AS "Average Working Hours" FROM Works;
+-----------------------+
| Average Working Hours |
+-----------------------+
|    10.666666666666666 |
+-----------------------+
1 row in set (0.00 sec)
```

4) (MIN)get minimum working hours of an employeeavailable in the table

sql>select min(working_hours) as “Minimum Working hour” from Works;

```
mysql> SELECT MIN(working_hours) AS "Minimum Working Hour" FROM Works;
+----------------------+
| Minimum Working Hour |
+----------------------+
| 08                   |
+----------------------+
1 row in set (0.01 sec)
```

5) (MAX)get maximum working hours of an employee available in the table

sql>select max(working_hours) as “Maximum Working hour” from Works;

```
mysql> SELECT MAX(working_hours) AS "Maximum Working Hour" FROM Works;
+----------------------+
| Maximum Working Hour |
+----------------------+
| 9                    |
+----------------------+
1 row in set (0.00 sec)
```

Queries with Aggregate functions (GROUP BY and HAVING clause)

1) Fetch the gender from works table.

Sql> SELECT gender FROM works GROUP BY gender;

```
mysql>  SELECT gender FROM works GROUP BY gender;
+--------+
| gender |
+--------+
| Male   |
| Female |
+--------+
2 rows in set (0.02 sec)
```

Fetch total employee per occupation in reverse alphabetic

1) order.

Sql> SELECT occupation, count (*) as “Total
Employee” FROM works GROUP BY occupation
ORDER BY occupation DESC

```
mysql> SELECT occupation, COUNT(*) AS "Total Employee"
    -> FROM Works
    -> GROUP BY occupation
    -> ORDER BY occupation DESC;
+------------+-----------------+
| occupation | Total Employee  |
+------------+-----------------+
| Teacher    |               3 |
| Steward    |               1 |
| Scientist  |               1 |
| Engineer   |               4 |
| Doctor     |               4 |
| Business   |               2 |
| Anchor     |               2 |
| Analyst    |               2 |
| Actor      |               2 |
+------------+-----------------+
9 rows in set (0.01 sec)
```

2) Fetch the average working hours of male and female employees.

Sql> SELECT gender, avg(working_hours) FROM works GROUP BY gender ORDER BY gender

```
mysql> SELECT gender, AVG(working_hours)
    -> FROM Works
    -> GROUP BY gender
    -> ORDER BY gender;
+--------+---------------------+
| gender | AVG(working_hours) |
+--------+---------------------+
| Female | 10.444444444444445 |
| Male   | 10.833333333333334 |
+--------+---------------------+
2 rows in set (0.01 sec)
```

3) Fetch the number of male and female workers FROM works table.

Sql> SELECT gender, count (*) as “Total” FROM works GROUP BY gender;

```
mysql> SELECT gender, COUNT(*) AS "Total"
    -> FROM Works
    -> GROUP BY gender;
+--------+-------+
| gender | Total |
+--------+-------+
| Male   |    12 |
| Female |     9 |
+--------+-------+
2 rows in set (0.00 sec)
```

4) Fetch the total working hours, avg working hours per occupation, sum total working hours for total working hours greater than 18 hrs

Sql>SELECT occupation,
AVG(working_hours) AS “Avg WH”,
SUM(working_hours) AS “Total WH”,
FROM works

GROUP BY working_hours
HAVING SUM(working_hours) > 18
ORDER BY MAX(working_hours) DESC;

```
mysql> SELECT occupation,
    -> AVG(working_hours) AS "Avg WH",
    -> SUM(working_hours) AS "Total WH"
    -> FROM Works
    -> GROUP BY occupation
    -> HAVING SUM(working_hours) > 18
    -> ORDER BY MAX(working_hours) DESC;
+-------------+--------+-----------+
| occupation | Avg WH | Total WH |
+-------------+--------+-----------+
| Engineer    |   9.25 |        37 |
| Doctor      |   10.5 |        42 |
| Teacher     |     12 |        36 |
| Actor       |     13 |        26 |
| Business    |   10.5 |        21 |
| Analyst     |     10 |        20 |
+-------------+--------+-----------+
6 rows in set (0.02 sec)
```

5) to retrieve the total working hours of All the Male employees who are engineers and doctors

Sql> SELECT occupation, working_hours
FROM works
WHERE gender =’Male’
GROUP BY occupation
HAVING occupation IN (‘Engineer’,’Doctor’);

```
mysql> SELECT occupation, SUM(working_hours) AS "Total Working Hours"
    -> FROM Works
    -> WHERE gender = 'Male' AND occupation IN ('Engineer', 'Doctor')
    -> GROUP BY occupation;
+------------+---------------------+
| occupation | Total Working Hours |
+------------+---------------------+
| Engineer   |                  37 |
| Doctor     |                  24 |
+------------+---------------------+
2 rows in set (0.01 sec)
```

6) using Aggregate function return the occupation and number of employees and apply having clause to filter the results to display occupation having more than 2 employees.

Sql> SELECT occupation,count(*) as “Total Employee”
FROM works

GROUP BY occupation
HAVING count(*) > 2;

```
mysql> SELECT occupation, COUNT(*) AS "Total Employee"
    -> FROM Works
    -> GROUP BY occupation
    -> HAVING COUNT(*) > 2;
+------------+----------------+
| occupation | Total Employee |
+------------+----------------+
| Engineer   |              4 |
| Doctor     |              4 |
| Teacher    |              3 |
+------------+----------------+
3 rows in set (0.00 sec)
```

Practical 5

1) Extracts the date part of the date or datetime expression expr. Returns NULL if expr is NULL.

Sql>SELECT DATE('2023-03-16 01:02:03');

```
mysql> SELECT DATE('2023-03-16 01:02:03');
+-----------------------------+
| DATE('2023-03-16 01:02:03') |
+-----------------------------+
| 2023-03-16                  |
+-----------------------------+
1 row in set (0.01 sec)
```

2) Extracts the date part of the date or datetime expression expr. Returns NULL if expr is NULL.

Sql>SELECT DATE('2023-03-03’);

```
mysql> SELECT DATE('2023-03-03');
+----------------------+
| DATE('2023-03-03') |
+----------------------+
| 2023-03-03         |
+----------------------+
1 row in set (0.01 sec)
```

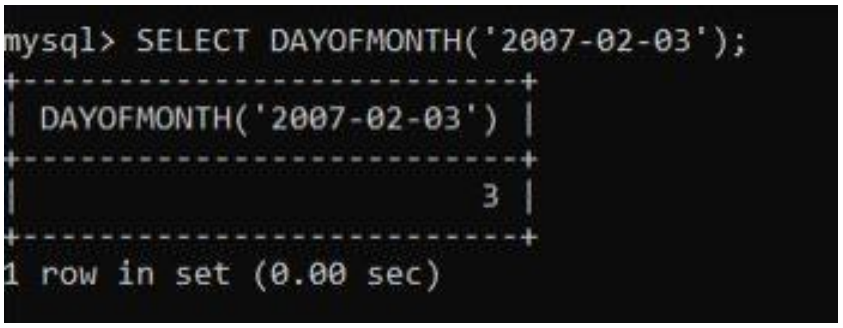
3) Return day name

sql> SELECT DAYNAME('2007-02-03');

```
mysql> SELECT DAYNAME('2007-02-03');
+------------------------+
| DAYNAME('2007-02-03') |
+------------------------+
| Saturday               |
+------------------------+
1 row in set (0.01 sec)
```

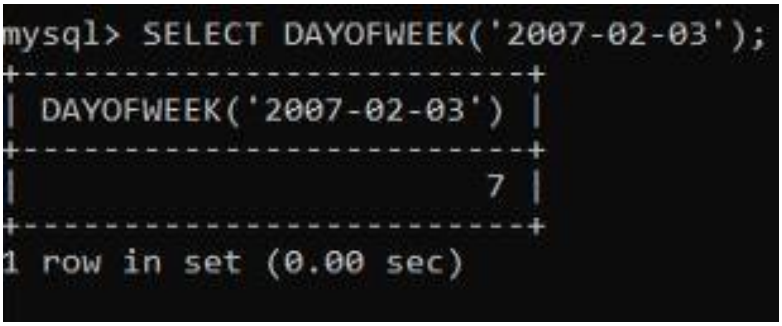
4) Return day of month

sql> SELECT DAYOFMONTH('2007-02-03');



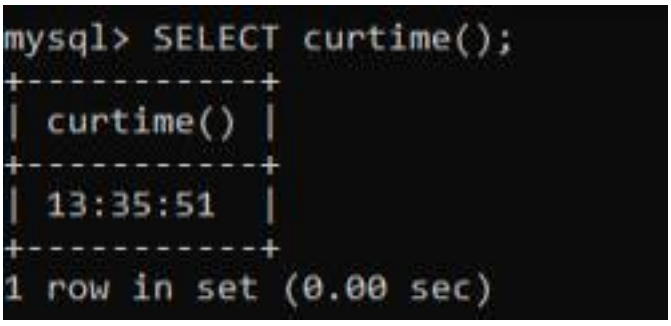
5) Return day of week

sql>SELECT DAYOFWEEK('2007-02-03');



6) Current time

Sql> SELECT curtime();



7) Current date

Sql> SELECT curdate()

```
mysql> SELECT curdate();
+-------------+
| curdate()   |
+-------------+
| 2024-02-18  |
+-------------+
1 row in set (0.00 sec)
```

8) Extract date part from works table's working_date

Sql>SELECT DATE(working_date) from works;

```
mysql> SELECT DATE ( working_date) from works;
+-----------------------+
| DATE ( working_date)  |
+-----------------------+
| 2020-10-04            |
| 2020-10-04            |
| 2020-10-04            |
| 2020-10-04            |
| 2020-10-04            |
| 2020-10-04            |
| 2021-12-14            |
| 2022-11-24            |
| 2020-04-01            |
| 2023-01-01            |
| 2021-06-01            |
| 2022-08-05            |
| 2020-03-23            |
| 2021-04-09            |
| 2022-10-04            |
| 2023-09-01            |
| 2020-09-01            |
| 2021-12-14            |
| 2022-05-14            |
| 2020-04-01            |
| 2022-04-01            |
+-----------------------+
21 rows in set (0.02 sec)
```

9) Extract the no of days between the two dates(Return the number of days between two date values:

Return the number of days between two date values) :

sql>SELECT DATEDIFF("2017-06-25 09:34:21", "2017-06-15 15:25:35");

```
mysql> SELECT DATEDIFF("2017-06-25 09:34:21", "2017-06-15 15:25:35");
+-----------------------------------------------------------+
| DATEDIFF("2017-06-25 09:34:21", "2017-06-15 15:25:35") |
+-----------------------------------------------------------+
|                                                        10 |
+-----------------------------------------------------------+
1 row in set (0.01 sec)
```

10) The DATE_ADD() function adds a time/date interval to a date and then returns the date.

sql>SELECT DATE_ADD("2017-06-15 09:34:21", INTERVAL 15 MINUTE);

```
mysql> SELECT DATE_ADD("2017-06-15 09:34:21", INTERVAL 15 MINUTE);
+-----------------------------------------------------+
| DATE_ADD("2017-06-15 09:34:21", INTERVAL 15 MINUTE) |
+-----------------------------------------------------+
| 2017-06-15 09:49:21                                 |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

sql>SELECT DATE_ADD("2017-06-15 09:34:21", INTERVAL -3 HOUR);

```
mysql> SELECT DATE_ADD("2017-06-15 09:34:21", INTERVAL -3 HOUR);
+-----------------------------------------------------+
| DATE_ADD("2017-06-15 09:34:21", INTERVAL -3 HOUR) |
+-----------------------------------------------------+
| 2017-06-15 06:34:21                                 |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

sql>SELECT DATE_ADD("2017-06-15", INTERVAL -2 MONTH);

```
mysql> SELECT DATE_ADD("2017-06-15", INTERVAL -2 MONTH);
+-------------------------------------------+
| DATE_ADD("2017-06-15", INTERVAL -2 MONTH) |
+-------------------------------------------+
| 2017-04-15                                |
+-------------------------------------------+
1 row in set (0.06 sec)
```

sql>SELECT DATE_ADD("2017-06-15", INTERVAL 10 DAY);

```
mysql> SELECT DATE_ADD("2017-06-15", INTERVAL 10 DAY);
+-----------------------------------------+
| DATE_ADD("2017-06-15", INTERVAL 10 DAY) |
+-----------------------------------------+
| 2017-06-25                              |
+-----------------------------------------+
1 row in set (0.00 sec)
```

11) Extract Current date and time:

sql>SELECT NOW();

```
mysql> SELECT NOW();
+---------------------+
| NOW()               |
+---------------------+
| 2024-02-18 13:54:09 |
+---------------------+
1 row in set (0.00 sec)
```

12) Extract the year part of the date:

Sql> SELECT YEAR("2024-02-12");

```
mysql> SELECT YEAR("2024-02-12");
+--------------------+
| YEAR("2024-02-12") |
+--------------------+
|               2024 |
+--------------------+
1 row in set (0.01 sec)
```

13) Extract the hour part of a datetime:

Sql>select hour(now());

```
mysql> Select hour(now());
+--------------+
| hour(now()) |
+--------------+
|           13 |
+--------------+
1 row in set (0.00 sec)
```

Sql>select hour(“2024-02-12 09:34:00");

```
mysql> select hour("2024-02-12 09:34:00");
+-----------------------------+
| hour("2024-02-12 09:34:00") |
+-----------------------------+
|                           9 |
+-----------------------------+
1 row in set (0.00 sec)
```

14) Extract the minutes part of the date

Sql> SELECT MINUTE("2017-06-20 09:34:00");

```
mysql>  SELECT MINUTE("2017-06-20 09:34:00");
+-------------------------------+
| MINUTE("2017-06-20 09:34:00") |
+-------------------------------+
|                            34 |
+-------------------------------+
1 row in set (0.00 sec)
```

Sql>Select minute(now());

```
mysql> Select minute(now());
+-----------------+
| minute(now()) |
+-----------------+
|               0 |
+-----------------+
1 row in set (0.00 sec)
```

15) Extract the Seconds part of the date

Sql> SELECT SECOND("2017-06-20 09:34:00.000023");

```
mysql>  SELECT SECOND("2017-06-20 09:34:00.000023");
+---------------------------------------+
| SECOND("2017-06-20 09:34:00.000023") |
+---------------------------------------+
|                                     0 |
+---------------------------------------+
1 row in set (0.00 sec)
```

Sql> SELECT SECOND(now());

```
mysql>  SELECT SECOND(now());
+----------------+
| SECOND(now()) |
+----------------+
|             52 |
+----------------+
1 row in set (0.00 sec)
```

16) Extract the reverse the date

Sql>SELECT REVERSE(now());

```
mysql> SELECT REVERSE(now());
+---------------------+
| REVERSE(now())      |
+---------------------+
| 80:50:41 81-20-4202 |
+---------------------+
1 row in set (0.00 sec)
```

Sql> SELECT REVERSE("2017-06-20 09:34:00.000023");

```
mysql> SELECT REVERSE("2017-06-20 09:34:00.000023");
+----------------------------------------+
| REVERSE("2017-06-20 09:34:00.000023") |
+----------------------------------------+
| 320000.00:43:90 02-60-7102             |
+----------------------------------------+
1 row in set (0.00 sec)
```

b. String Functions

1 ASCII() This function returns numeric value of left-most character. **SELECT ASCII('t');**

```
mysql>  SELECT ASCII('t');
+-------------+
| ASCII('t') |
+-------------+
|         116 |
+-------------+
1 row in set (0.00 sec)
```

2 BIN() This function returns a string representation of the argument. **SELECT BIN(225);**

```
mysql> SELECT BIN(225);
+------------+
| BIN(225)   |
+------------+
| 11100001   |
+------------+
1 row in set (0.01 sec)
```

3 BIT_LENGTH() This function returns length of argument in bits.

SELECT BIT_LENGTH('RDNational');

```
mysql> SELECT BIT_LENGTH('RDNational');
+--------------------------+
| BIT_LENGTH('RDNational') |
+--------------------------+
|                       80 |
+--------------------------+
1 row in set (0.01 sec)
```

4 CHAR_LENGTH() This function returns number of characters in argument

SELECT CHAR_LENGTH(' RDNational ');

```
mysql> SELECT CHAR_LENGTH(' RDNational ');
+----------------------------+
| CHAR_LENGTH(' RDNational ') |
+----------------------------+
|                         12 |
+----------------------------+
1 row in set (0.01 sec)
```

7 CONCAT() This function returns concatenated string.

SELECT CONCAT('RD', 'SH','National');

```
mysql> SELECT CONCAT('RD', 'SH','National');
+-------------------------------+
| CONCAT('RD', 'SH','National') |
+-------------------------------+
| RDSHNational                  |
+-------------------------------+
1 row in set (0.00 sec)
```

9 ELT() This function returns string at index number.

SELECT ELT(3, 'Java', 'JavaFX', 'OpenCV', 'WebGL');

```
mysql> SELECT ELT( 3, 'Java', 'JavaFX',
    -> 'OpenCV', 'WebGL');
+-----------------------------------------------+
| ELT( 3, 'Java', 'JavaFX',
'OpenCV', 'WebGL') |
+-----------------------------------------------+
| OpenCV                                        |
+-----------------------------------------------+
1 row in set (0.00 sec)
```

11 FIELD() This function returns the index (position) of the first argument in the subsequent arguments.

SELECT FIELD('JavaFX', 'Java', 'JavaFX', 'OpenCV', 'WebGL');

```
mysql> SELECT FIELD('JavaFX', 'Java',
    -> 'JavaFX', 'OpenCV', 'WebGL');
+-------------------------------------------------------+
| FIELD('JavaFX', 'Java',
'JavaFX', 'OpenCV', 'WebGL') |
+-------------------------------------------------------+
|                                                     2 |
+-------------------------------------------------------+
1 row in set (0.00 sec)
```

12 FIND_IN_SET() This function returns the index position of the first argument within the second argument

SELECT FIND_IN_SET('JavaFX', 'Java,JavaFX,OpenCV,WebGL');

```
mysql> SELECT FIND_IN_SET('JavaFX', 'Java,JavaFX,OpenCV,WebGL');
+-----------------------------------------------+
| FIND_IN_SET('JavaFX', 'Java,JavaFX,OpenCV,WebGL') |
+-----------------------------------------------+
|                                             2 |
+-----------------------------------------------+
1 row in set (0.01 sec)
```

13 FROM_BASE64() This function decodes a base-64 encoded string as a binary string

SELECT FROM_BASE64 ('VHV0b3JpYWxzcG9pbnQ=');

```
mysql> SELECT FROM_BASE64 ('VHV0b3JpYWxzcG9pbnQ=');
+-----------------------------------------------------------------------+
| FROM_BASE64 ('VHV0b3JpYWxzcG9pbnQ=')                                  |
+-----------------------------------------------------------------------+
| 0x5475746F7269616C73706F696E74                                        |
+-----------------------------------------------------------------------+
1 row in set (0.00 sec)
```

16 LCASE() Synonym for LOWER(). **SELECT LCASE('HELLOWORLD');**

```
mysql> SELECT LCASE('HELLOWORLD');
+---------------------+
| LCASE('HELLOWORLD') |
+---------------------+
| helloworld          |
+---------------------+
1 row in set (0.00 sec)
```

17 LEFT() This function returns the leftmost number of characters as specified

SELECT LEFT('GOODMORNING', 4);

```
mysql> SELECT LEFT('GOODMORNING',
    -> 4);
+-------------------------+
| LEFT('GOODMORNING',
4) |
+-------------------------+
| GOOD                    |
+-------------------------+
1 row in set (0.01 sec)
```

22 LPAD() This function returns the string argument, left-padded with the specified string.

SELECT LPAD('Fourier Series', 35,'$%*');

```
mysql> SELECT LPAD('Fourier Series',
    -> 35,'$%*');
+-------------------------------------+
| LPAD('Fourier Series',
35,'$%*')    |
+-------------------------------------+
| $%*$%*$%*$%*$%*$%*$%*Fourier Series |
+-------------------------------------+
1 row in set (0.01 sec)
```

c. Math Functions

1)number raised to power

Sql>Select pow(3,2);

```
mysql> Select pow(3,2);
+-----------+
| pow(3,2)  |
+-----------+
|         9 |
+-----------+
1 row in set (0.01 sec)
```

2) Value of pi

Sql>Select pi();

```
mysql> Select pi();
+-----------+
| pi()      |
+-----------+
| 3.141593  |
+-----------+
1 row in set (0.01 sec)
```

3) get the reminder (modulus operator)

Sql>Select 4%3;

```
mysql> Select 4%3;
+------+
| 4%3  |
+------+
|    1 |
+------+
1 row in set (0.01 sec)
```

Sql>Select mod(14,3);

```
mysql> Select mod(14,3);
+------------+
| mod(14,3)  |
+------------+
|          2 |
+------------+
1 row in set (0.00 sec)
```

4) to round off the given number
Sql>Select round(123.564);

```
mysql> Select round(123.564);
+-----------------+
| round(123.564) |
+-----------------+
|             124 |
+-----------------+
1 row in set (0.01 sec)
```

Sql>Select round(34.223);

```
mysql> Select round(34.223);
+----------------+
| round(34.223) |
+----------------+
|             34 |
+----------------+
1 row in set (0.01 sec)
```

5) to get the smallest number from the list.

Sql>Select least(4,1,9,0);

```
mysql> Select least(4,1,9,0);
+----------------+
| least(4,1,9,0) |
+----------------+
|              0 |
+----------------+
1 row in set (0.00 sec)
```

Sql>Select least ('m','Y','S','q','L');

```
mysql> Select least ('m', 'y', 's', 'q', 'l');
+---------------------------------+
| least ('m', 'y', 's', 'q', 'l') |
+---------------------------------+
| l                               |
+---------------------------------+
1 row in set (0.00 sec)
```

6) to find the greatest integer which is equal to or less than the given number.

Sql>Select floor(11.6);

```
mysql> Select floor(11.6);
+--------------+
| floor(11.6) |
+--------------+
|           11 |
+--------------+
1 row in set (0.01 sec)
```

Sql>Select floor(29.34);

```
mysql> Select floor(29.34);
+--------------+
| floor(29.34) |
+--------------+
|           29 |
+--------------+
1 row in set (0.00 sec)
```

Sql>Select floor(-12.5);

```
mysql> Select floor(-12.5);
+---------------+
| floor(-12.5) |
+---------------+
|           -13 |
+---------------+
1 row in set (0.00 sec)
```

7) returns the smallest value which is greater than or equal to the given number.

Sql>Select ceil(49.9);

```
mysql> Select ceil(49.9);
+------------+
| ceil(49.9) |
+------------+
|         50 |
+------------+
1 row in set (0.00 sec)
```

Sql>Select ceil(56.5);

```
mysql> Select ceil(56.5);
+------------+
| ceil(56.5) |
+------------+
|         57 |
+------------+
1 row in set (0.00 sec)
```

Sql>Select ceil(-45.5);

```
mysql> Select ceil(-45.5);
+-------------+
| ceil(-45.5) |
+-------------+
|         -45 |
+-------------+
1 row in set (0.00 sec)
```

Sql>Select ceil(-45);

```
mysql> Select ceil(-45);
+------------+
| ceil(-45) |
+------------+
|        -45 |
+------------+
1 row in set (0.00 sec)
```

8) truncates a number to the specified number of decimal places.

Sql>Select truncate(43.2134,3);

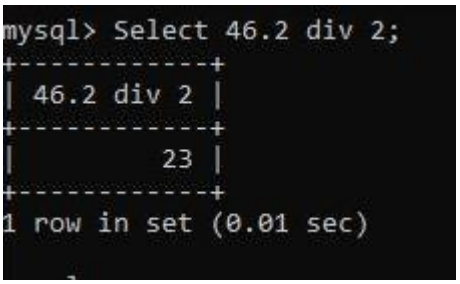
```
mysql> Select truncate(43.2134,3);
+----------------------+
| truncate(43.2134,3) |
+----------------------+
|               43.213 |
+----------------------+
1 row in set (0.00 sec)
```

Sql>Select truncate(243.52134,2);

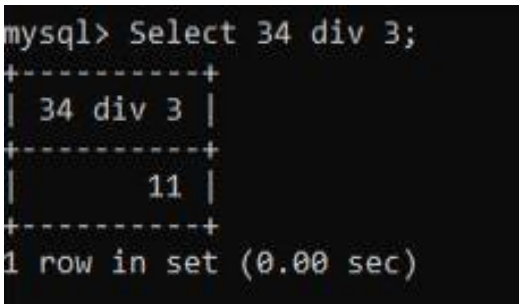
```
mysql> Select truncate(243.52134,2);
+-----------------------+
| truncate(243.52134,2) |
+-----------------------+
|                243.52 |
+-----------------------+
1 row in set (0.00 sec)
```

9) for integer division, in which n is the number to be divided by the number m.

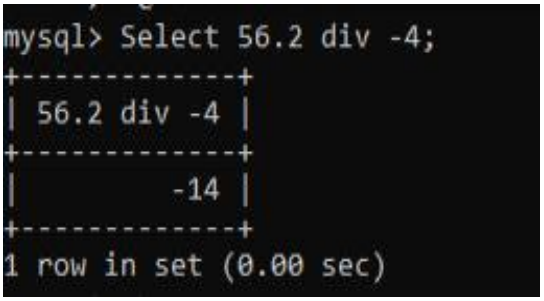
Sql>Select 46.2 div 2;



Sql>Select 34 div 3;



Sql>Select 56.2 div -4 ;



PRACTICAL NO: 6

Perform Queries Involving (JOIN QUERIES):

Retrieving Data FROM Multiple Table:

1) CREATE TABLE prod (Pid integer(4) NOT NULL,pname varchar(45) NOT NULL,pqty integer(8),pcost
decimal(10,2) NOT NULL,PRIMARY KEY (pid));

```
mysql> CREATE TABLE prod (
    ->   Pid INT(4) NOT NULL,
    ->   pname VARCHAR(45) NOT NULL,
    ->   pqty INT(8),
    ->   pcost DECIMAL(10,2) NOT NULL,
    ->   PRIMARY KEY (Pid)
    -> );
Query OK, 0 rows affected, 2 warnings (0.10 sec)
```

2) INSERT INTO prod (Pid, pname, pqty, pcost)
VALUES
(1001, 'Pen Drive', 100, 900),
(1002, 'Hard Disk', 200, 4000),
(1003, 'Headphone', 1000, 15000),
(1004, 'DVD', 200, 1000),
(1005, 'Speaker', 600, 5400),
(1006, 'Mouse', 1000, 10000),
(1007, 'ZMouse', 100, 10000),
(1008, 'ExternalHD', 50, 40000);

```
mysql> SELECT * FROM prod;
+------+------------+------+----------+
| Pid  | pname      | pqty | pcost    |
+------+------------+------+----------+
| 1001 | Pen Drive  |  100 |   900.00 |
| 1002 | Hard Disk  |  200 |  4000.00 |
| 1003 | Headphone  | 1000 | 15000.00 |
| 1004 | DVD        |  200 |  1000.00 |
| 1005 | Speaker    |  600 |  5400.00 |
| 1006 | Mouse      | 1000 | 10000.00 |
| 1007 | ZMouse     |  100 | 10000.00 |
| 1008 | ExternalHD |   50 | 40000.00 |
+------+------------+------+----------+
8 rows in set (0.00 sec)
```

3) CREATE TABLE sale (Sid number(4) NOT NULL, Pid intrger(4) NOT NULL, sqty integer(8) NOT NULL ,
scost decimal(10,2) NOT NULL , custname varchar(50) NOT NULL, PRIMARY KEY (sid) , CONSTRAINT
FK_Pid FOREIGN KEY(pid) REFERENCES prod(pid));

```
mysql> CREATE TABLE sale (
    ->   Sid INT(4) NOT NULL,
    ->   Pid INT(4) NOT NULL,
    ->   sqty INT(8) NOT NULL,
    ->   scost DECIMAL(10,2) NOT NULL,
    ->   custname VARCHAR(50) NOT NULL,
    ->   PRIMARY KEY (Sid),
    ->   CONSTRAINT FK_Pid FOREIGN KEY(Pid) REFERENCES prod(Pid)
    -> );
Query OK, 0 rows affected, 3 warnings (0.08 sec)
```

4) Insert data as:

INSERT INTO sale VALUES (2001,1001,50,900,'Shanti'),(2002,1004,10,1000,'Shanti'), (2003,1003,120,15000,'Shanti'),(2004,1005,420,2400,'Nandane'), (2005,1002,40,400,'Ramesh'),(2006,1006,100,10500,'Suresh');

```
mysql> SELECT * FROM sale;
+------+------+------+----------+----------+
| Sid  | Pid  | sqty | scost    | custname |
+------+------+------+----------+----------+
| 2001 | 1001 |   50 |   900.00 | Shanti   |
| 2002 | 1004 |   10 |  1000.00 | Shanti   |
| 2003 | 1003 |  120 | 15000.00 | Shanti   |
| 2004 | 1005 |  420 |  2400.00 | Nandane  |
| 2005 | 1002 |   40 |   400.00 | Ramesh   |
| 2006 | 1006 |  100 | 10500.00 | Suresh   |
+------+------+------+----------+----------+
6 rows in set (0.00 sec)
```

a. Joining Tables (Inner Joins, Outer-Joins)

1) Inner join – Matching values in both tables

SELECT prod.pid, sale.custname, prod.pname FROM prod INNER JOIN sale ON prod.pid = sale.pid;

```
mysql> SELECT prod.Pid, sale.custname, prod.pname
    -> FROM prod
    -> INNER JOIN sale
    -> ON prod.Pid = sale.Pid;
+------+----------+-----------+
| Pid  | custname | pname     |
+------+----------+-----------+
| 1001 | Shanti   | Pen Drive |
| 1004 | Shanti   | DVD       |
| 1003 | Shanti   | Headphone |
| 1005 | Nandane  | Speaker   |
| 1002 | Ramesh   | Hard Disk |
| 1006 | Suresh   | Mouse     |
+------+----------+-----------+
6 rows in set (0.01 sec)
```

2) left join – returns all records from the left table, even if there are no matches in the right table.

SELECT prod.pid, prod.pname, sale.custname FROM prod LEFT OUTER JOIN sale ON sale.pid = prod.pid;

```
mysql> SELECT prod.Pid, prod.pname, sale.custname
    -> FROM prod
    -> LEFT OUTER JOIN sale
    -> ON prod.Pid = sale.Pid;
+------+------------+-----------+
| Pid  | pname      | custname  |
+------+------------+-----------+
| 1001 | Pen Drive  | Shanti    |
| 1002 | Hard Disk  | Ramesh    |
| 1003 | Headphone  | Shanti    |
| 1004 | DVD        | Shanti    |
| 1005 | Speaker    | Nandane   |
| 1006 | Mouse      | Suresh    |
| 1007 | ZMouse     | NULL      |
| 1008 | ExternalHD | NULL      |
+------+------------+-----------+
8 rows in set (0.00 sec)
```

3) right join – returns all records from the right table , and the matching records from the left table
SELECT prod.pid, prod.pname, sale.custname FROM prod RIGHT OUTER JOIN sale ON sale.pid =
prod.pid;

```
mysql> SELECT prod.Pid, prod.pname, sale.custname
    -> FROM prod
    -> RIGHT OUTER JOIN sale
    -> ON prod.Pid = sale.Pid;
+------+-----------+----------+
| Pid  | pname     | custname |
+------+-----------+----------+
| 1001 | Pen Drive | Shanti   |
| 1004 | DVD       | Shanti   |
| 1003 | Headphone | Shanti   |
| 1005 | Speaker   | Nandane  |
| 1002 | Hard Disk | Ramesh   |
| 1006 | Mouse     | Suresh   |
+------+-----------+----------+
6 rows in set (0.00 sec)
```

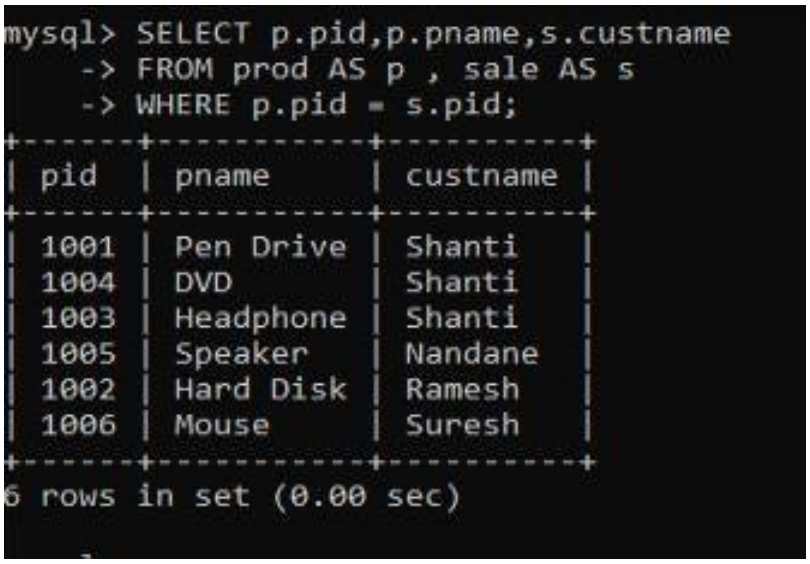
4)full Join – returns all records when there is a match in left or right table records.

SELECT prod.pid,prod.pname,sale.custname FROM sale FULL OUTER JOIN ON prod.pid = sale.pid

```
mysql> SELECT prod.Pid, prod.pname, sale.custname
    -> FROM prod
    -> LEFT JOIN sale
    -> ON prod.Pid = sale.Pid
    ->
    -> UNION
    ->
    -> SELECT prod.Pid, prod.pname, sale.custname
    -> FROM prod
    -> RIGHT JOIN sale
    -> ON prod.Pid = sale.Pid;
+------+------------+----------+
| Pid  | pname      | custname |
+------+------------+----------+
| 1001 | Pen Drive  | Shanti   |
| 1002 | Hard Disk  | Ramesh   |
| 1003 | Headphone  | Shanti   |
| 1004 | DVD        | Shanti   |
| 1005 | Speaker    | Nandane  |
| 1006 | Mouse      | Suresh   |
| 1007 | ZMouse     | NULL     |
| 1008 | ExternalHD | NULL     |
+------+------------+----------+
8 rows in set (0.02 sec)
```

b. Aliases for Table Names

```
SELECT p.pid,p.pname,s.custname
FROM prod AS p , sale AS s
WHERE p.pid = s.pid;
```



```
mysql> SELECT p.pid,p.pname,s.custname
    -> FROM prod AS p , sale AS s
    -> WHERE p.pid = s.pid;
+------+-----------+----------+
| pid  | pname     | custname |
+------+-----------+----------+
| 1001 | Pen Drive | Shanti   |
| 1004 | DVD       | Shanti   |
| 1003 | Headphone | Shanti   |
| 1005 | Speaker   | Nandane  |
| 1002 | Hard Disk | Ramesh   |
| 1006 | Mouse     | Suresh   |
+------+-----------+----------+
6 rows in set (0.00 sec)
```

PRACTICAL NO: 7

Subqueries:

With IN clause

1) get names of all people who run a business or are actors
from works table

SELECT name, working_hours
FROM Works
WHERE occupation IN

(SELECT occupation from Works where
occupation =’Business’ OR occupation = ‘Actor’);

```
mysql> SELECT name, working_hours
    -> FROM Works
    -> WHERE occupation IN
    -> (SELECT occupation FROM Works WHERE
    -> occupation ='Business' OR occupation = 'Actor');
+----------+---------------+
| name     | working_hours |
+----------+---------------+
| Patricia | 13            |
| Anita    | 11            |
| Preesha  | 13            |
| Anindita | 10            |
+----------+---------------+
4 rows in set (0.01 sec)
```

2) get the data all employees from works table whose name start with “P”
SELECT * FROM Works
WHERE name IN
(SELECT name from

Works where name LIKE ‘P%’);

```
mysql> SELECT * FROM Works
    -> WHERE name IN
    -> (SELECT name FROM Works WHERE name LIKE 'P%');
+----------+--------+------------+--------------+---------------+
| name     | gender | occupation | working_date | working_hours |
+----------+--------+------------+--------------+---------------+
| Patricia | Female | Actor      | 2020-10-04   | 13            |
| Preet    | Male   | Anchor     | 2020-04-01   | 9             |
| Preesha  | Female | Actor      | 2021-04-09   | 13            |
| Prachi   | Female | Anchor     | 2020-04-01   | 9             |
+----------+--------+------------+--------------+---------------+
4 rows in set (0.02 sec)
```

3)get the data of all the employees who have joined the organization from year 2021.
SELECT * FROM Works
WHERE working_date IN

(SELECT working_date from Works where working_date >= ‘2021-01-01’);

```
mysql> SELECT * FROM Works
    -> WHERE working_date IN
    -> (SELECT working_date FROM Works WHERE working_date >= '2021-01-01');
+---------+--------+------------+--------------+---------------+
| name    | gender | occupation | working_date | working_hours |
+---------+--------+------------+--------------+---------------+
| Roshani | Female | Analyst    | 2021-12-14   | 10            |
| Wishu   | Male   | Engineer   | 2022-11-24   | 8             |
| Maria   | Female | Doctor     | 2023-01-01   | 8             |
| Badri   | Male   | Teacher    | 2021-06-01   | 16            |
| Ananda  | Male   | Steward    | 2022-08-05   | 12            |
| Preesha | Female | Actor      | 2021-04-09   | 13            |
| Madan   | Male   | Doctor     | 2022-10-04   | 10            |
| Brajesh | Male   | Teacher    | 2023-09-01   | 08            |
| Roshani | Female | Analyst    | 2021-12-14   | 10            |
| Watson  | Male   | Engineer   | 2022-05-14   | 9             |
| Mehul   | Female | Doctor     | 2022-04-01   | 10            |
+---------+--------+------------+--------------+---------------+
11 rows in set (0.00 sec)
```

4) get the data of all the employees who have joined the organization before year 2021.

SELECT * FROM Works
WHERE working_date NOT IN
(SELECT working_date from Works where working_date >= ‘2021-01-01’);

```
mysql> SELECT * FROM Works
    -> WHERE working_date NOT IN
    -> (SELECT working_date FROM Works WHERE working_date >= '2021-01-01');
+----------+--------+------------+--------------+---------------+
| name     | gender | occupation | working_date | working_hours |
+----------+--------+------------+--------------+---------------+
| Robin    | Male   | Scientist  | 2020-10-04   | 12            |
| Warner   | Male   | Engineer   | 2020-10-04   | 10            |
| Patricia | Female | Actor      | 2020-10-04   | 13            |
| Marco    | Male   | Doctor     | 2020-10-04   | 14            |
| Brayden  | Male   | Teacher    | 2020-10-04   | 12            |
| Anita    | Female | Business   | 2020-10-04   | 11            |
| Preet    | Male   | Anchor     | 2020-04-01   | 9             |
| Roshan   | Male   | Engineer   | 2020-03-23   | 10            |
| Anindita | Female | Business   | 2020-09-01   | 10            |
| Prachi   | Female | Anchor     | 2020-04-01   | 9             |
+----------+--------+------------+--------------+---------------+
10 rows in set (0.01 sec)
```

b. With EXISTS clause

1) get the pid number and name from products file whose sale records are present in sale table.

Sql> SELECT pid,pname
FROM prod
WHERE EXISTS

(SELECT * FROM sale WHERE prod.pid = sale.pid);

```
mysql> SELECT pid, pname
    -> FROM prod
    -> WHERE EXISTS
    -> (SELECT * FROM sale WHERE prod.pid = sale.pid);
+------+------------+
| pid  | pname      |
+------+------------+
| 1001 | Pen Drive  |
| 1002 | Hard Disk  |
| 1003 | Headphone  |
| 1004 | DVD        |
| 1005 | Speaker    |
| 1006 | Mouse      |
+------+------------+
6 rows in set (0.01 sec)
```

2) get the pid number and name from products file whose sale records are present not in sale table.

Sql> SELECT pid,pname

FROM prod

WHERE NOT EXISTS

(SELECT * FROM sale WHERE prod.pid = sale.pid);

```
mysql> SELECT pid, pname
    -> FROM prod
    -> WHERE NOT EXISTS
    -> (SELECT * FROM sale WHERE prod.pid = sale.pid);
+------+------------+
| pid  | pname      |
+------+------------+
| 1007 | ZMouse     |
| 1008 | ExternalHD |
+------+------------+
2 rows in set (0.00 sec)
```

c. Handling NULL

Null values indicate an unknown or non-existent value and is different from an empty string
Sql>SELECT name , address from customers where address IS NULL;

```
mysql> SELECT name , address from customers where address IS NULL;
Empty set (0.06 sec)
```

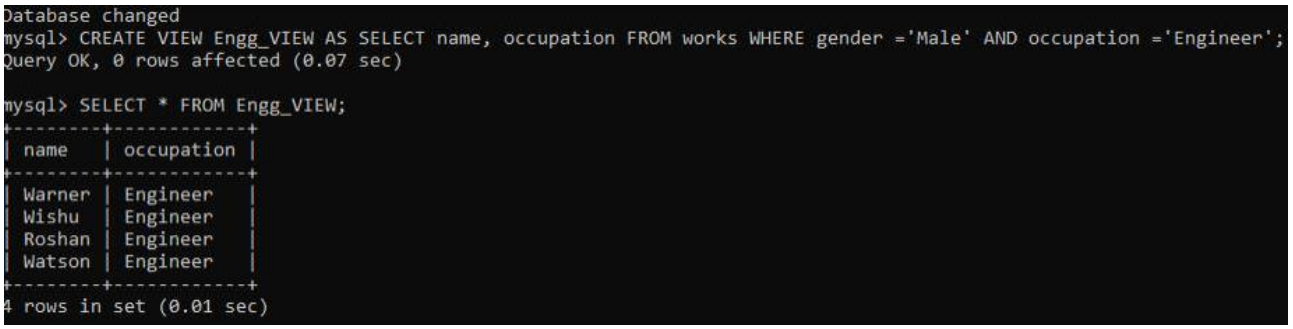
PRACTICAL NO: 8

Views:

a. Creating Views

1) view of all male engineers from works table.

CREATE VIEW Engg_VIEW AS SELECT name, ocupation FROM works WHERE gender =’Male’ and occupation =’Engineer’;



```
Database changed
mysql> CREATE VIEW Engg_VIEW AS SELECT name, occupation FROM works WHERE gender ='Male' AND occupation ='Engineer';
Query OK, 0 rows affected (0.07 sec)

mysql> SELECT * FROM Engg_VIEW;
+--------+------------+
| name   | occupation |
+--------+------------+
| Warner | Engineer   |
| Wishu  | Engineer   |
| Roshan | Engineer   |
| Watson | Engineer   |
+--------+------------+
4 rows in set (0.01 sec)
```

2) view of name and occupation of all female staff working in the enterprise.

CREATE VIEW FemaleStaff AS SELECT name, occupation FROM Works WHERE gender=’Female’;

```
mysql> CREATE VIEW FemaleStaff AS SELECT name, occupation FROM Works WHERE gender='Female';
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM FemaleStaff;
+----------+------------+
| name     | occupation |
+----------+------------+
| Patricia | Actor      |
| Anita    | Business   |
| Roshani  | Analyst    |
| Maria    | Doctor     |
| Preesha  | Actor      |
| Anindita | Business   |
| Roshani  | Analyst    |
| Prachi   | Anchor     |
| Mehul    | Doctor     |
+----------+------------+
9 rows in set (0.00 sec)
```

3) view of products having cost above the average cost

CREATE VIEW ProdAbvAvgView AS
SELECT pname,pcost
FROM prod
WHERE pcost > (SELECT AVG(pcost) FROM prod);

```
mysql> CREATE VIEW ProdAbvAvgView AS
    -> SELECT pname, pcost
    -> FROM prod
    -> WHERE pcost > (SELECT AVG(pcost) FROM prod);
Query OK, 0 rows affected (0.02 sec)

mysql> SELECT * FROM ProdAbvAvgView;
+------------+----------+
| pname      | pcost    |
+------------+----------+
| Headphone  | 15000.00 |
| ExternalHD | 40000.00 |
+------------+----------+
2 rows in set (0.01 sec)
```

b. Selecting from view

1)
Sql>Select * from Engg_View;

2)
Sql>Select * from FemaleStaff;

3)
Sql>Select * from ProdAbvAvgView;

c. Dropping Views

Sql> DROP VIEW FemaleStaff CASCADE;

OR **{In MySQL, the CASCADE and RESTRICT keywords are not used when dropping views }**

Sql> DROP VIEW FemaleStaff RESTRICT;

```
ysql> DROP VIEW IF EXISTS FemaleStaff;
uery OK, 0 rows affected, 1 warning (0.01 sec)

ysql> Select * FROM FemaleStaff;
RROR 1146 (42S02): Table 'arpitdb.femalestaff' doesn't exist
ysql>
```

DROP VIEW IF EXISTS Engg_View;

```
mysql> DROP VIEW IF EXISTS Engg_View;
Query OK, 0 rows affected (0.01 sec)
```

DROP VIEW IF EXISTS ProdAbvAvgView;

```
mysql> DROP VIEW IF EXISTS ProdAbvAvgView;
Query OK, 0 rows affected (0.01 sec)
```

Name: Kaustubh Anant Rane **DATABASE MANAGEMENT SYSTEM**
Roll No: CS23037

PRACTICAL NO :9

Granting and revoking permissions:

1. Grant:

   SQL Grant command is specifically used to provide privileges to database objects for a user. This command also allows users to grant permissions to other users too.

   Syntax: grant privilege_name on object_name to {user_name | public | role_name}
   Example: grant insert,select on accounts to user1;
   By the above command user1 has been granted permissions on accounts database object to query or insert into accounts.

2. Revoke: Revoke command withdraw user privileges on database objects if any granted. It does operations opposite to the Grant command. When a privilege is revoked from a particular user U, then the privileges granted to all other users by user U will be revoked.

   Syntax: revoke privilege_name on object_name from {user_name | public | role_name}
   Example: revoke insert,select on accounts from user1;
   By the above command user1's permissions like query or insert on accounts database object has been removed.

Grant and Revoke Commands

Practical 10

mysql> START TRANSACTION;

```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)
```

mysql>select * from emp1;

```
mysql> SELECT * FROM emp1;
+-------+---------+------------+----------+
| EmpNo | EmpName | JoinDate   | EmpSal   |
+-------+---------+------------+----------+
|   101 | Rajesh  | 2021-10-23 | 50000.00 |
|   102 | Raju    | 2021-10-23 | 15000.00 |
|   103 | Kamlesh | 2019-04-15 | 19000.00 |
+-------+---------+------------+----------+
```

mysql>insert into emp1 values (111,'Hello World','2023-03- 16',50000);

```
mysql> INSERT INTO emp1 VALUES (111,'Hello World','2023-03-16',50000);
Query OK, 1 row affected (0.02 sec)
```

mysql>select * from emp1;

```
mysql> SELECT * FROM emp1;
+-------+-------------+------------+----------+
| EmpNo | EmpName     | JoinDate   | EmpSal   |
+-------+-------------+------------+----------+
|   101 | Rajesh      | 2021-10-23 | 50000.00 |
|   102 | Raju        | 2021-10-23 | 15000.00 |
|   103 | Kamlesh     | 2019-04-15 | 19000.00 |
|   111 | Hello World | 2023-03-16 | 50000.00 |
+-------+-------------+------------+----------+
4 rows in set (0.00 sec)
```

mysql>SAVEPOINT my_savepoint1;

```
mysql> SAVEPOINT my_savepoint1;
Query OK, 0 rows affected (0.00 sec)
```

mysql> insert into emp1 values (112,'Hello All','2023-03- 15',52000);

```
mysql> INSERT INTO emp1 VALUES (112,'Hello All','2023-03-15',52000);
Query OK, 1 row affected (0.00 sec)
```

mysql>ROLLBACK TO SAVEPOINT my_savepoint1;

```
mysql> ROLLBACK TO SAVEPOINT my_savepoint1;
Query OK, 0 rows affected (0.00 sec)
```

mysql> insert into emp1 values (112,'Hello One','2023-03- 14',25000);

```
mysql> INSERT INTO emp1 VALUES (112,'Hello One','2023-03-14',25000);
Query OK, 1 row affected (0.00 sec)
```

mysql> COMMIT;

```
mysql> COMMIT;
Query OK, 0 rows affected (0.01 sec)
```

mysql>select * from emp1;

```
mysql> SELECT * FROM emp1;
+-------+-------------+------------+----------+
| EmpNo | EmpName     | JoinDate   | EmpSal   |
+-------+-------------+------------+----------+
|   101 | Rajesh      | 2021-10-23 | 50000.00 |
|   102 | Raju        | 2021-10-23 | 15000.00 |
|   103 | Kamlesh     | 2019-04-15 | 19000.00 |
|   111 | Hello World | 2023-03-16 | 50000.00 |
|   112 | Hello One   | 2023-03-14 | 25000.00 |
+-------+-------------+------------+----------+
5 rows in set (0.00 sec)
```

Practical.11

Creating /viewing / dropping Indexes on Databases

Syntax: CREATE UNIQUE INDEX index_name
ON table_name (column1, column2, ...);

Q) create index on last name attribute on employee table.

A) mysql> CREATE INDEX idx_lastname
ON emp1 (LastName);

```
mysql> CREATE INDEX idx_lastname
    -> ON emp1 (LastName);
Query OK, 0 rows affected (0.14 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select*from emp1;
+-------+-------------+------------+----------+----------+
| EmpNo | EmpName     | JoinDate   | EmpSal   | LastName |
+-------+-------------+------------+----------+----------+
|   101 | Rajesh      | 2021-10-23 | 50000.00 | NULL     |
|   102 | Raju        | 2021-10-23 | 15000.00 | NULL     |
|   103 | Kamlesh     | 2019-04-15 | 19000.00 | NULL     |
|   111 | Hello World | 2023-03-16 | 50000.00 | NULL     |
|   112 | Hello One   | 2023-03-14 | 25000.00 | NULL     |
+-------+-------------+------------+----------+----------+
5 rows in set (0.00 sec)
```

Q) create index on last name and first name attribute on employee table.

A) CREATE INDEX idx_pname
ON emp1 (LastName, FirstName);

```
mysql> ALTER TABLE emp1
    -> ADD FirstName VARCHAR(255);
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select*from emp1;
+--------+-------------+------------+----------+----------+------------+
| EmpNo  | EmpName     | JoinDate   | EmpSal   | LastName | FirstName  |
+--------+-------------+------------+----------+----------+------------+
|      1 | NULL        | NULL       |     NULL | Smith    | NULL       |
|    101 | Rajesh      | 2021-10-23 | 50000.00 | NULL     | NULL       |
|    102 | Raju        | 2021-10-23 | 15000.00 | NULL     | NULL       |
|    103 | Kamlesh     | 2019-04-15 | 19000.00 | NULL     | NULL       |
|    111 | Hello World | 2023-03-16 | 50000.00 | NULL     | NULL       |
|    112 | Hello One   | 2023-03-14 | 25000.00 | NULL     | NULL       |
+--------+-------------+------------+----------+----------+------------+
6 rows in set (0.00 sec)
```

TO view the indexes from the tables in the database

Syntax: SHOW INDEXES FROM table_name;
mysql> SHOW INDEXES FROM emp1;

```
mysql> SHOW INDEXES FROM emp1;
+-------+------------+--------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+
| Table | Non_unique | Key_name     | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible |
 Expression |
+-------+------------+--------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+
------------+
| emp1  |          0 | PRIMARY      |            1 | EmpNo       | A         |           6 |     NULL |   NULL |      | BTREE      |         |               | YES     |
 NULL       |
| emp1  |          1 | idx_lastname |            1 | LastName    | A         |           2 |     NULL |   NULL | YES  | BTREE      |         |               | YES     |
 NULL       |
+-------+------------+--------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+
------------+
2 rows in set (0.07 sec)
```

Syntax: SHOW INDEXES FROM table_name IN databasename;

mysql> SHOW INDEXES FROM fycs0102.emp1;

```
mysql> SHOW INDEXES FROM arpitdb.emp1;
+-------+------------+--------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+
| Table | Non_unique | Key_name     | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible |
 Expression |
+-------+------------+--------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+
------------+
| emp1  |          0 | PRIMARY      |            1 | EmpNo       | A         |           6 |     NULL |   NULL |      | BTREE      |         |               | YES     |
 NULL       |
| emp1  |          1 | idx_lastname |            1 | LastName    | A         |           2 |     NULL |   NULL | YES  | BTREE      |         |               | YES     |
 NULL       |
+-------+------------+--------------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+
------------+
```

TO DROP / Delete / Remove Index

Syntax: ALTER TABLE table_name

DROP INDEX index_name;

Q) drop index idx_lastname on employee table.

A) mysql> ALTER TABLE emp1
DROP INDEX idx_lastname;

```
mysql>  ALTER TABLE emp1
    ->
    -> DROP INDEX idx_lastname;
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select*from emp1;
+-------+-------------+------------+----------+----------+-----------+
| EmpNo | EmpName     | JoinDate   | EmpSal   | LastName | FirstName |
+-------+-------------+------------+----------+----------+-----------+
|     1 | NULL        | NULL       |     NULL | Smith    | John      |
|   101 | Rajesh      | 2021-10-23 | 50000.00 | NULL     | veer      |
|   102 | Raju        | 2021-10-23 | 15000.00 | NULL     | NULL      |
|   103 | Kamlesh     | 2019-04-15 | 19000.00 | NULL     | NULL      |
|   111 | Hello World | 2023-03-16 | 50000.00 | NULL     | NULL      |
|   112 | Hello One   | 2023-03-14 | 25000.00 | NULL     | NULL      |
+-------+-------------+------------+----------+----------+-----------+
6 rows in set (0.00 sec)
```

Q drop index idx_pname on employee table.

A) ALTER TABLE emp1
DROP INDEX idx_pname;

```
mysql>  ALTER TABLE emp1
    ->
    -> DROP INDEX idx_pname;
ERROR 1091 (42000): Can't DROP 'idx_pname'; check that column/key exists
mysql>
```

Name: Kaustubh Anant Rane
Roll No: CS23037

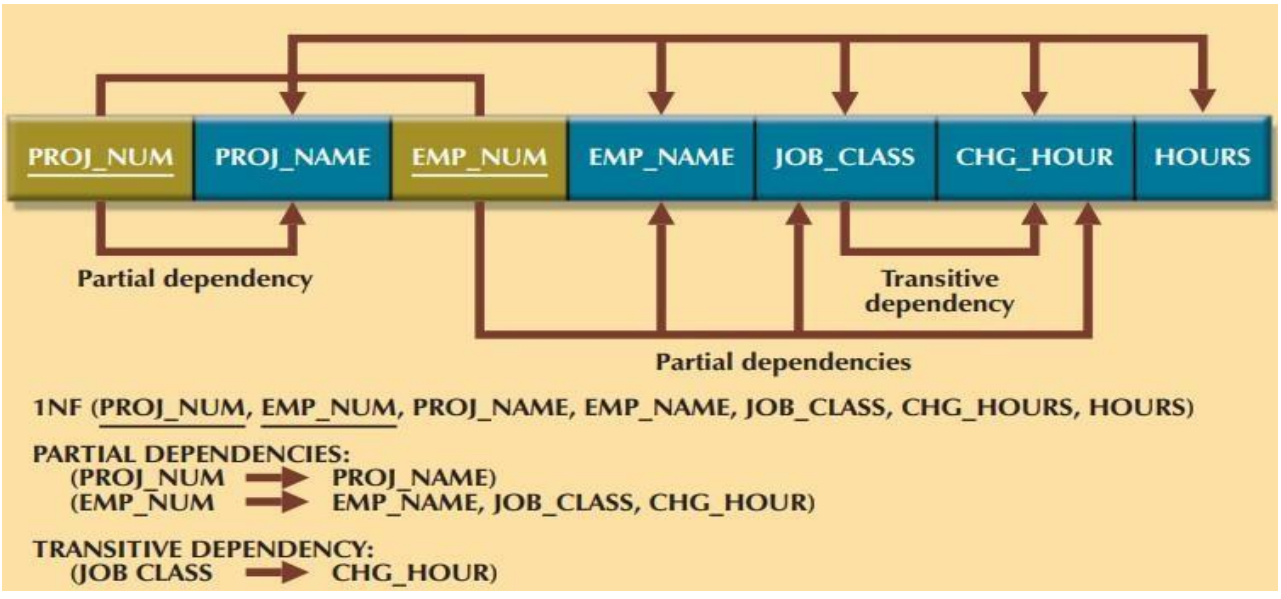
Practical 12

Normalization of data.

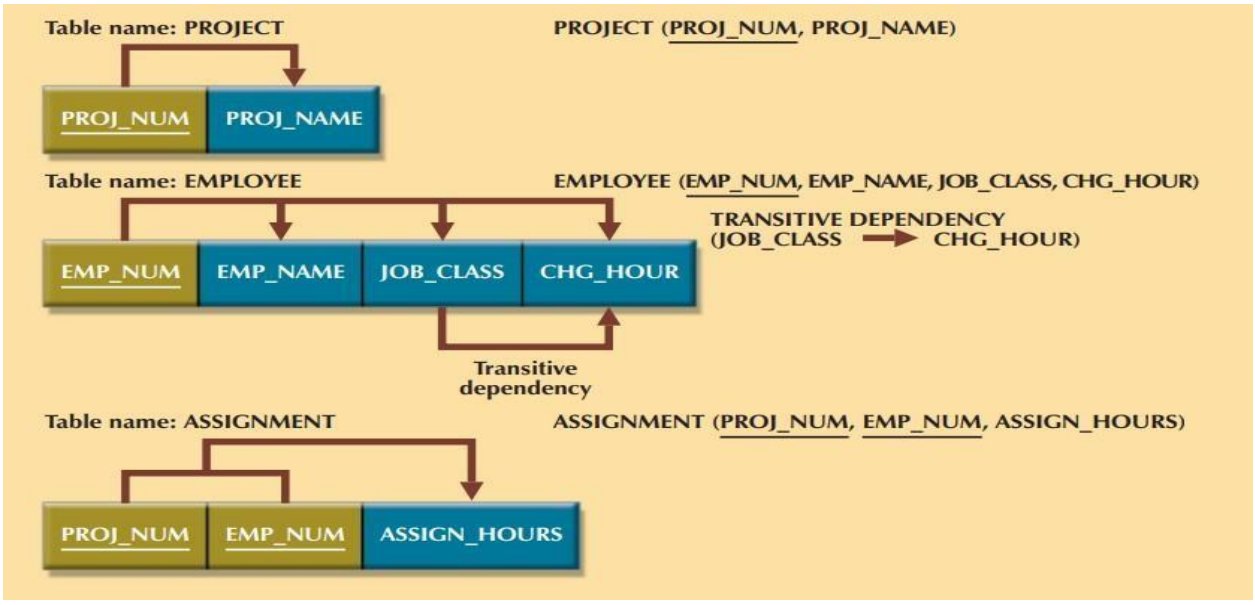
Problem 1] Consider the given scenario- the activities of a construction company that manages several building projects. Each project has its own project number, name, employees assigned to it, and so on. Each employee has an employee number, name, and job classification, such as engineer or computer technician. The company charges its clients by billing the hours spent on each contract. The hourly billing rate is dependent on the employee's position. For example, one hour of computer technician time is billed at a different rate than one hour of engineer time. The generated report is -

PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS
15	Evergreen	103	June E. Arbough	Elect. Engineer	84.50	23.8
		101	John G. News	Database Designer	105.00	19.4
		105	Alice K. Johnson *	Database Designer	105.00	35.7
		106	William Smithfield	Programmer	35.75	12.6
		102	David H. Senior	Systems Analyst	96.75	23.8
18	Amber Wave	114	Annelise Jones	Applications Designer	48.10	24.6
		118	James J. Frommer	General Support	18.36	45.3
		104	Anne K. Ramoras *	Systems Analyst	96.75	32.4
		112	Darlene M. Smithson	DSS Analyst	45.95	44.0
22	Rolling Tide	105	Alice K. Johnson	Database Designer	105.00	64.7
		104	Anne K. Ramoras	Systems Analyst	96.75	48.4
		113	Delbert K. Joenbrood *	Applications Designer	48.10	23.6
		111	Geoff B. Wabash	Clerical Support	26.87	22.0
		106	William Smithfield	Programmer	35.75	12.8
25	Starflight	107	Maria D. Alonzo	Programmer	35.75	24.6
		115	Travis B. Bawangi	Systems Analyst	96.75	45.8
		101	John G. News *	Database Designer	105.00	56.3
		114	Annelise Jones	Applications Designer	48.10	33.1
		108	Ralph B. Washington	Systems Analyst	96.75	23.6
		118	James J. Frommer	General Support	18.36	30.5
		112	Darlene M. Smithson	DSS Analyst	45.95	41.4

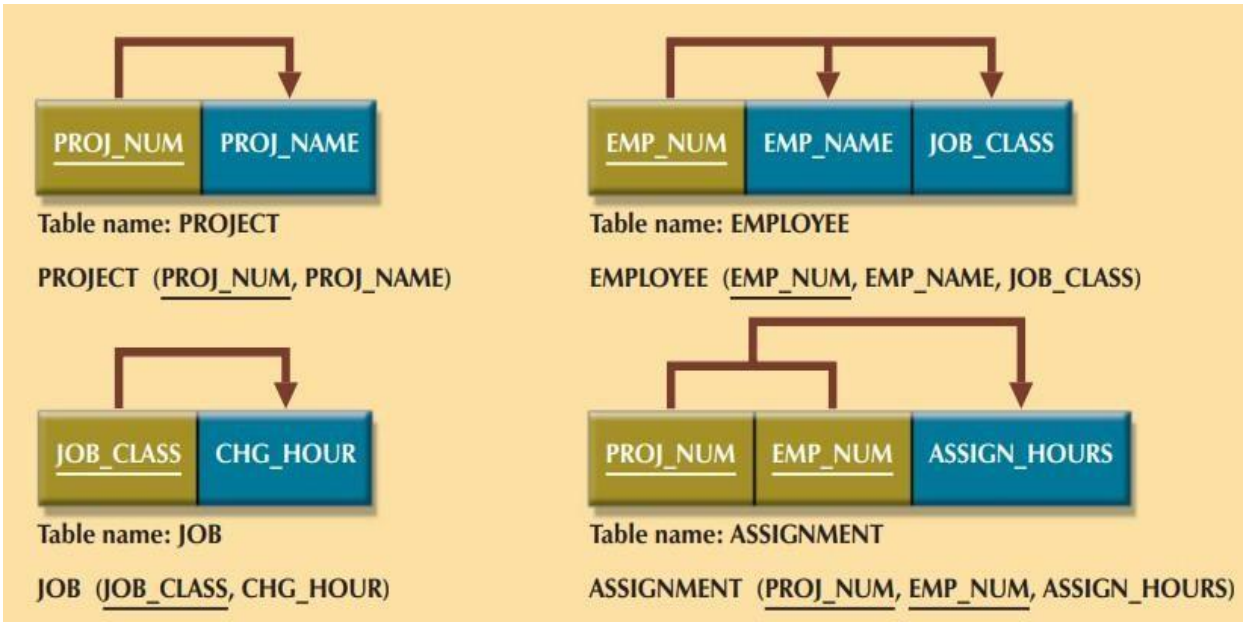
Conversion to 1NF



Conversion to 2NF



Conversion to 3NF



After the 3NF conversion has been completed, database contains four tables:
PROJECT (PROJ_NUM, PROJ_NAME)

EMPLOYEE (EMP_NUM, EMP_NAME, JOB_CLASS)

JOB (JOB_CLASS, CHG_HOUR)

ASSIGNMENT (PROJ_NUM, EMP_NUM, ASSIGN_HOURS)

Conversion to BCNF Problem

2]

Each CLASS_CODE identifies a class uniquely. The case in which a course might generate many classes. A student can take many classes. A staff member can teach many classes, but each class is taught by only one staff member.

STU_ID	STAFF_ID	CLASS_CODE	ENROLL_GRADE
125	25	21334	A
125	20	32456	C
135	20	28458	B
144	25	27563	C
144	20	32456	B

The structure shown in Table for conversion to BCNF is reflected in Panel A:
STU_ID + STAFF_ID → CLASS_CODE, ENROLL_GRADE CLASS_CODE → STAFF_ID