# CS2207PC:DATABASEMANAGEMENT SYSTEMSLAB

| B.Tech. II Year II Sem. | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Course Code | Category | Hours / Week | | | Credits | Maxumum Marks | | |
| | Core | L | T | P | C | CIA | SEE | Total |
| | | 0 | 0 | 3 | 1.5 | 25 | 75 | 100 |
| Contact classes: NIL | Tutorial Classes : NIL | Practical classes : 45 | | | | Total Classes :45 | | |
| Co-requisites: • Co-requisiteofcourse“DatabaseManagementSystems” | | | | | | | | |

**Course Overview:**

The purpose of this course is to provide a clear understanding of fundamentals with emphasis on their applications to create and manage large data sets. It highlights on technical overview of database software to retrieve data from n database. The course includes database design principles, normalization, concurrent transaction processing,, security, recovery and file organization techniques.

**CourseObjectives:**

- IntroduceERdatamodel,database designandnormalization
- LearnSQL basicsfordata definition and datamanipulation

**CourseOutcomes:**

- Designdatabaseschemafor agivenapplicationandapply normalization
- Acquireskillsinusing SQLcommandsfordata definitionand datamanipulation.
- Developsolutionsfordatabaseapplicationsusingprocedures,cursorsandtriggers

**ListofExperiments:**

1. ConceptdesignwithE-RModel
2. Relational Model
3. Normalization
4. PracticingDDLcommands
5. PracticingDMLcommands
6. Querying(usingANY,ALL,IN,Exists,NOTEXISTS,UNION,INTERSECT,Constraintsetc.)
7. QueriesusingAggregatefunctions,GROUPBY,HAVINGandCreationanddroppingofViews.
8. Triggers(Creationof inserttrigger,deletetrigger,updatetrigger)
9. Procedures
10. UsageofCursors

**TEXTBOOKS:**

1. DatabaseManagementSystems,RaghuramaKrishnan,JohannesGehrke,TataMcGrawHill,3rd Edition
2. DatabaseSystemConcepts,Silberschatz,Korth,McGrawHill,Vedition.

**REFERENCESBOOKS:**

1. DatabaseSystemsdesign,Implementation,andManagement,PeterRob&CarlosCoronel7thEdition.
2. FundamentalsofDatabaseSystems,ElmasriNavrate,*PearsonEducation*
3. IntroductiontoDatabaseSystems, C.J.Date,*PearsonEducation*
4. OracleforProfessionals,The XTeam,S.Shah andV.Shah,*SPD.*
5. DatabaseSystemsUsingOracle:ASimplifiedguide toSQLandPL/SQL,Shah,*PHI.*
6. FundamentalsofDatabaseManagementSystems, M.L. Gillenson,*WileyStudentEdition.*

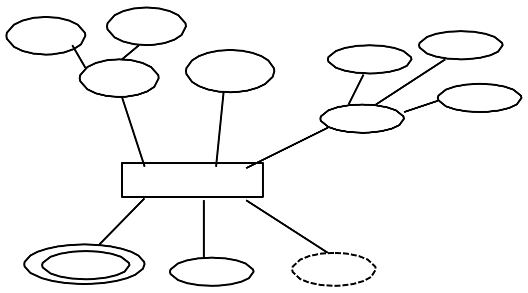
EXPERIMENT- 1

## CONCEPT DESIGN WITH E-R MODEL

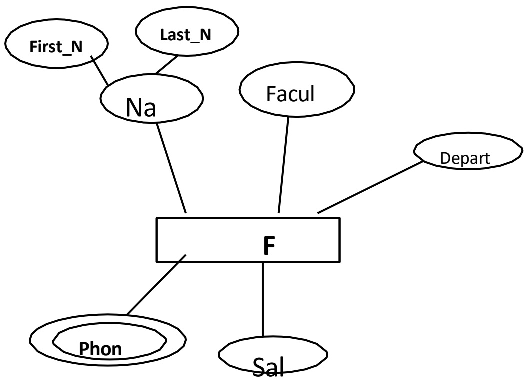
**AIM:** To Relate the entities appropriately. Apply cardinalities for each relationship. Identify strong and weak entities. Indicate the type of relationships (total/partial). Incorporate generalization, aggregation and specialization etc wherever required.

### E-R Model

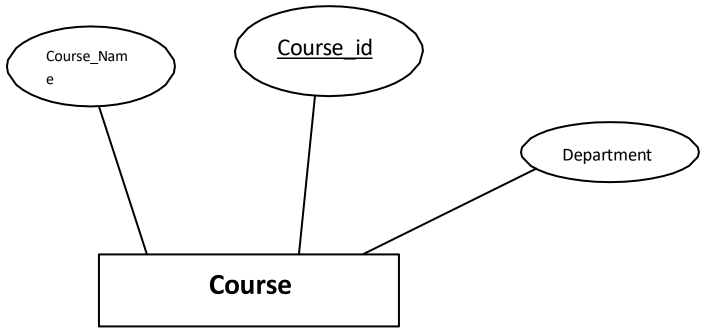
### Student (Entity)



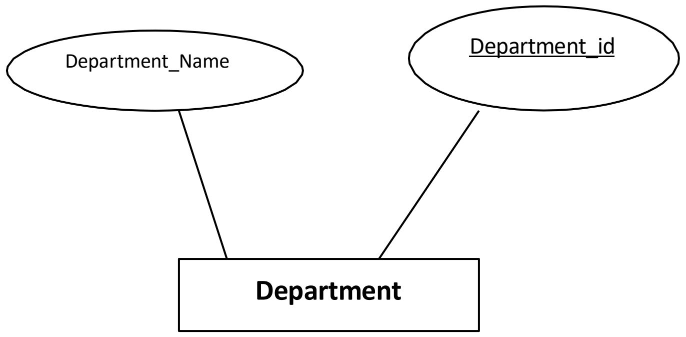
### Faculty : (Entity)



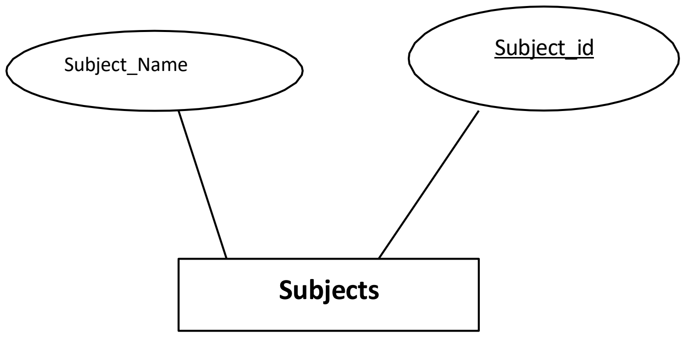
## Course : (Entity)



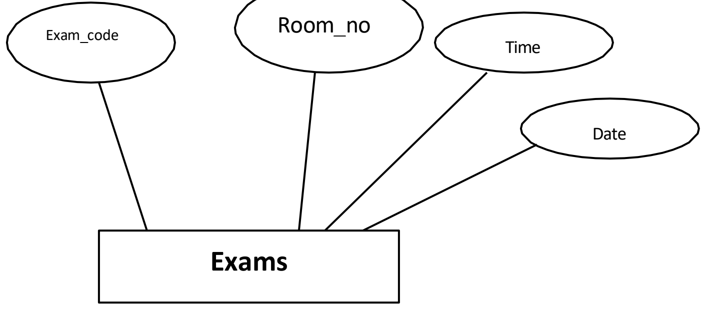
## Department : (Entity)



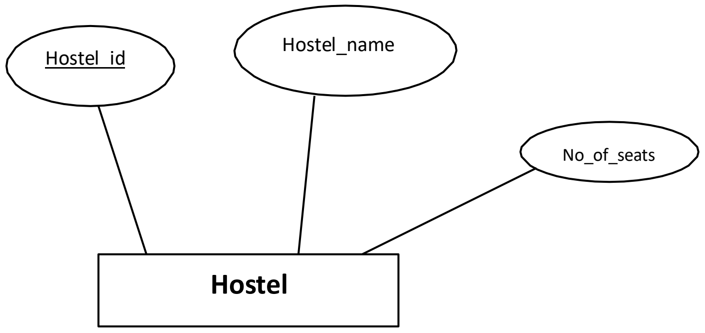
## Subjects : (Entity)



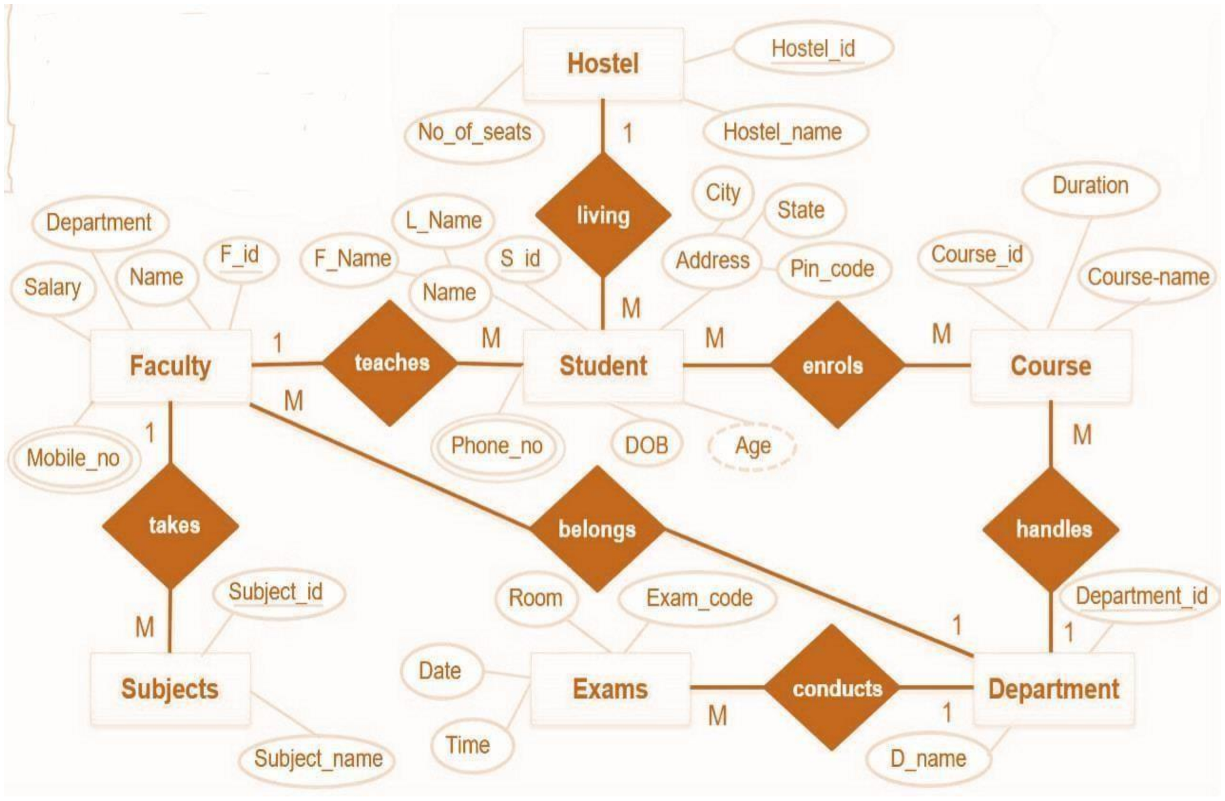
## Exams : (Entity)



## Hostel : (Entity)



## CONCEPT DESIGN WITH E-R MODEL



## EXPERIMENT – 2
## RELATIONAL MODEL

**AIM:** To Represent all the entities (Strong, Weak) in tabular fashion. Represent relationships in a tabular fashion.

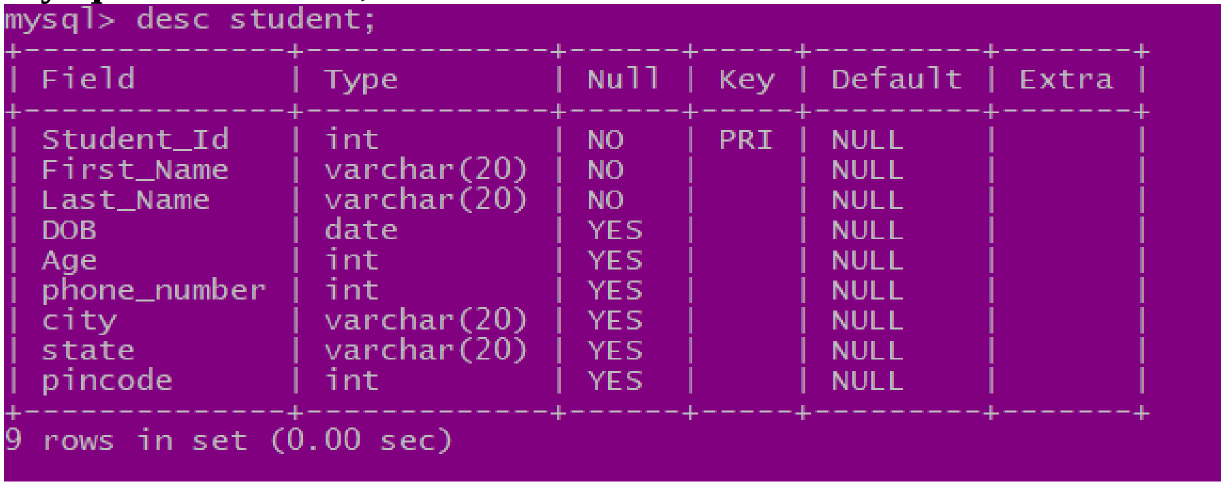
**Student:** student(S_ID : INTEGER,S_NAME : STRING, ADDRESS:STRING)

| Column Name | DataType | Constraints | Type of Attributes |
|---|---|---|---|
| S_ID | INTEGER | PRIMARY KEY | Single value |
| S_NAME | VARCHAR(20) | | Multi value |
| D.O.B | DATE | | |
| AGE | INT | | Single value |
| ADDRESS | VARCHAR (255) | | Multi value |

**SCHEMA:**

**Mysql>create table student(Student_Id integer primary key,First_Name Varchar(20) not null,Last_Name varchar(20) not null,DOB date,Age int,phone_number int,city varchar(20),state varchar(20),pincode int);**

**Mysql>desc student;**



```
mysql> desc student;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| Student_Id   | int         | NO   | PRI | NULL    |       |
| First_Name   | varchar(20) | NO   |     | NULL    |       |
| Last_Name    | varchar(20) | NO   |     | NULL    |       |
| DOB          | date        | YES  |     | NULL    |       |
| Age          | int         | YES  |     | NULL    |       |
| phone_number | int         | YES  |     | NULL    |       |
| city         | varchar(20) | YES  |     | NULL    |       |
| state        | varchar(20) | YES  |     | NULL    |       |
| pincode      | int         | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
9 rows in set (0.00 sec)
```

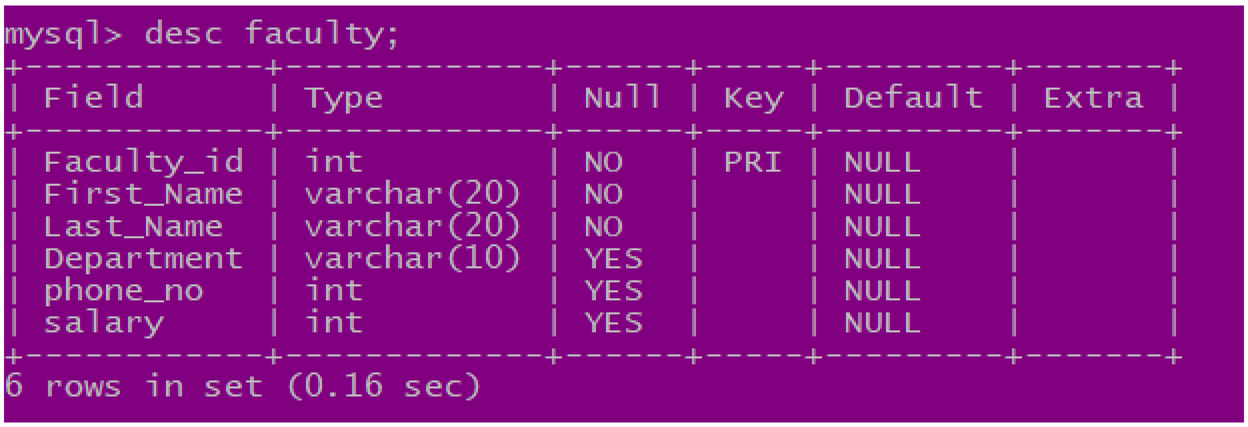
**FACULTY:**

| Column Name | DataType | Constraints | Type of Attributes |
|---|---|---|---|
| FACULTY_ID | INTEGER | PRIMARY KEY | Single value |
| FACULTY_NAME | VARCHAR(20) | | Multi value |
| DEPARTMENT | VARCHAR(10) | | Single value |
| PHONE NO | INT | | Multi value |
| SALARY | INT | | Single value |

**<u>SCHEMA:</u>**

**mysql> create table faculty(Faculty_id int primary key,First_Name varchar(20) not null,Last_Name varchar(20) not null,Department varchar(10),phone_no int,salary int);**

**Mysql>desc faculty;**



```
mysql> desc faculty;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| Faculty_id | int         | NO   | PRI | NULL    |       |
| First_Name | varchar(20) | NO   |     | NULL    |       |
| Last_Name  | varchar(20) | NO   |     | NULL    |       |
| Department | varchar(10) | YES  |     | NULL    |       |
| phone_no   | int         | YES  |     | NULL    |       |
| salary     | int         | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
6 rows in set (0.16 sec)
```

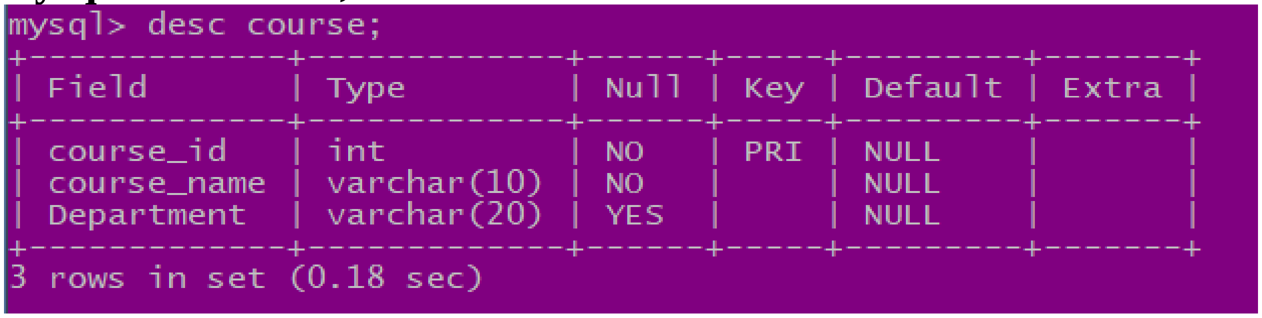
**COURSE:**

| Column Name | DataType | Constraints | Type of Attributes |
|---|---|---|---|
| COURSE_ID | INTEGER | PRIMARY KEY | Single value |
| COURSE_NAME | VARCHAR(10) | | Single value |
| DEPARTMENT | VARCHAR(20) | | Single value |

**SCHEMA:**

**mysql> create table course(course_id int primary key,course_name varchar(10) not null,Department varchar(20));**

**mysql>desc course;**



```
mysql> desc course;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| course_id   | int         | NO   | PRI | NULL    |       |
| course_name | varchar(10) | NO   |     | NULL    |       |
| Department  | varchar(20) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set (0.18 sec)
```

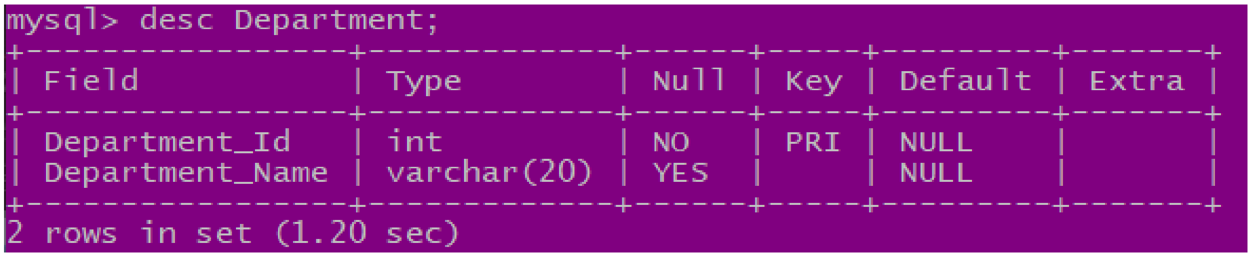
**DEPARTMENT:**

| Column Name | DataType | Constraints | Type of Attributes |
|---|---|---|---|
| DEPARTMENT_ID | INTEGER | PRIMARY KEY | Single value |
| DEPARTMENT_NAME | VARCHAR(20) | | Single value |

**SCHEMA:**

**mysql> create table Department(Department_Id int primary key,Department_Name varchar(20));**

**mysql>desc Department;**



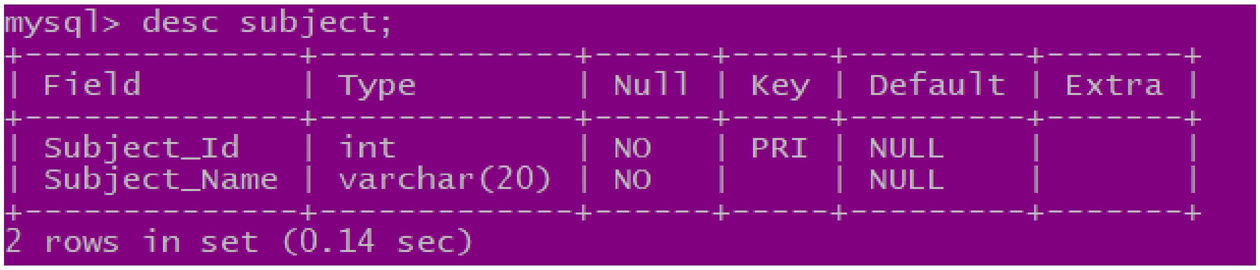
```
mysql> desc Department;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| Department_Id   | int         | NO   | PRI | NULL    |       |
| Department_Name | varchar(20) | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
2 rows in set (1.20 sec)
```

**SUBJECT:**

| Column Name | DataType | Constraints | Type of Attributes |
|---|---|---|---|
| SUBJECT_ID | INTEGER | PRIMARY KEY | Single value |
| SUBJECT_NAME | VARCHAR(20) | NOT NULL | Single value |

**<u>SCHEMA:</u>**

**mysql> create table Subject(Subject_Id int primary key,Subject_Name varchar(20) not null);**



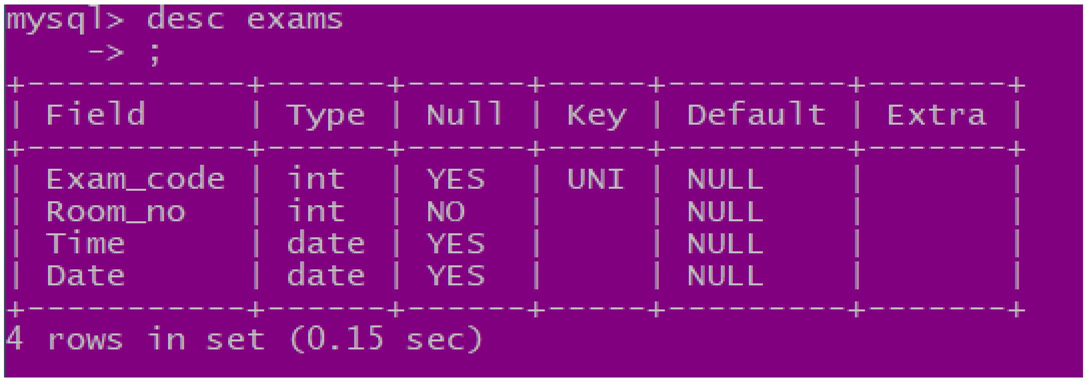
```
mysql> desc subject;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| Subject_Id   | int         | NO   | PRI | NULL    |       |
| Subject_Name | varchar(20) | NO   |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
2 rows in set (0.14 sec)
```

**EXAMS:**

| Column Name | DataType | Constraints | Type of Attributes |
|---|---|---|---|
| EXAM_CODE | INTEGER | UNIQUE | Single value |
| ROOM_NO | INTEGER | NOT NULL | Single value |
| TIME | DATE |  | Single value |
| DATE | DATE |  |  |

## SCHEMA:

**mysql> create table Exams(Exam_code int unique,Room_no int not null,Time date, Date date);**



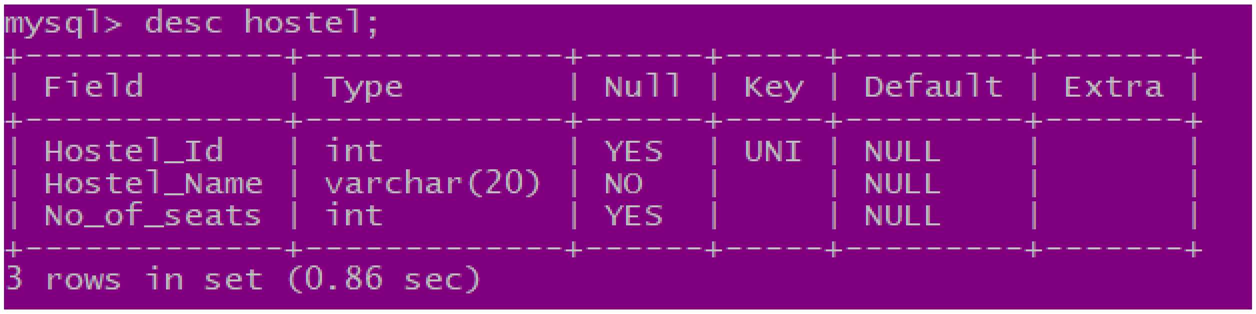
```
mysql> desc exams
    -> ;
+-----------+------+------+-----+---------+-------+
| Field     | Type | Null | Key | Default | Extra |
+-----------+------+------+-----+---------+-------+
| Exam_code | int  | YES  | UNI | NULL    |       |
| Room_no   | int  | NO   |     | NULL    |       |
| Time      | date | YES  |     | NULL    |       |
| Date      | date | YES  |     | NULL    |       |
+-----------+------+------+-----+---------+-------+
4 rows in set (0.15 sec)
```

**HOSTEL:**

| **Column Name** | **DataType** | **Constraints** | **Type of Attributes** |
|---|---|---|---|
| **HOSTEL_ID** | **INTEGER** | **UNIQUE** | **Single value** |
| **HOSTEL_NAME** | **VARCHAR(20)** | **NOT NULL** | **Single value** |
| **NO_OF_SEATS** | **INT** | | **Single value** |

## SCHEMA:

**mysql> create table Hostel(Hostel_Id int unique,Hostel_Name varchar(20) not null,No_of_seats int);**



```
mysql> desc hostel;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| Hostel_Id   | int         | YES  | UNI | NULL    |       |
| Hostel_Name | varchar(20) | NO   |     | NULL    |       |
| No_of_seats | int         | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set (0.86 sec)
```

# EXPERIMENT - 3

## NORMALIZATION

**AIM:** Apply the database Normalization techniques for designing relational database tables to minimize duplication of information like 1NF, 2NF, 3NF, BCNF.

Normalization is a process of converting a relation to be standard form by decomposition a larger relation into smaller efficient relation that depicts a good database design.

**1NF:** A Relation scheme is said to be in 1NF if the attribute values in the relation are atomic .i.e., Mutlivalued attributes are not permitted.

**2NF:** A Relation scheme is said to be in 2NF,iff and every Non-key attribute is fully functionally dependent on primary Key.

**3NF:** A Relation scheme is said to be in 3NF,iff and does not have transitivity dependencies. A Relation is said to be 3NF if every determinant is a key for each & every functional dependency.

**BCNF:** A Relation scheme is said to be BCNF if the following statements are true for eacg FD P->Q in set F of FDs that holds for each FD. P->Q in set F of FD's that holds over R. Here P is the subset of attributes of R & Q is a single attribute of R.
The given FD is a trival
P is a super key.

### 1. Exercise 1: 1st Normal Form (1NF)

Consider the Faculty table, with the primary key underlined, and the following data:

**Faculty:**

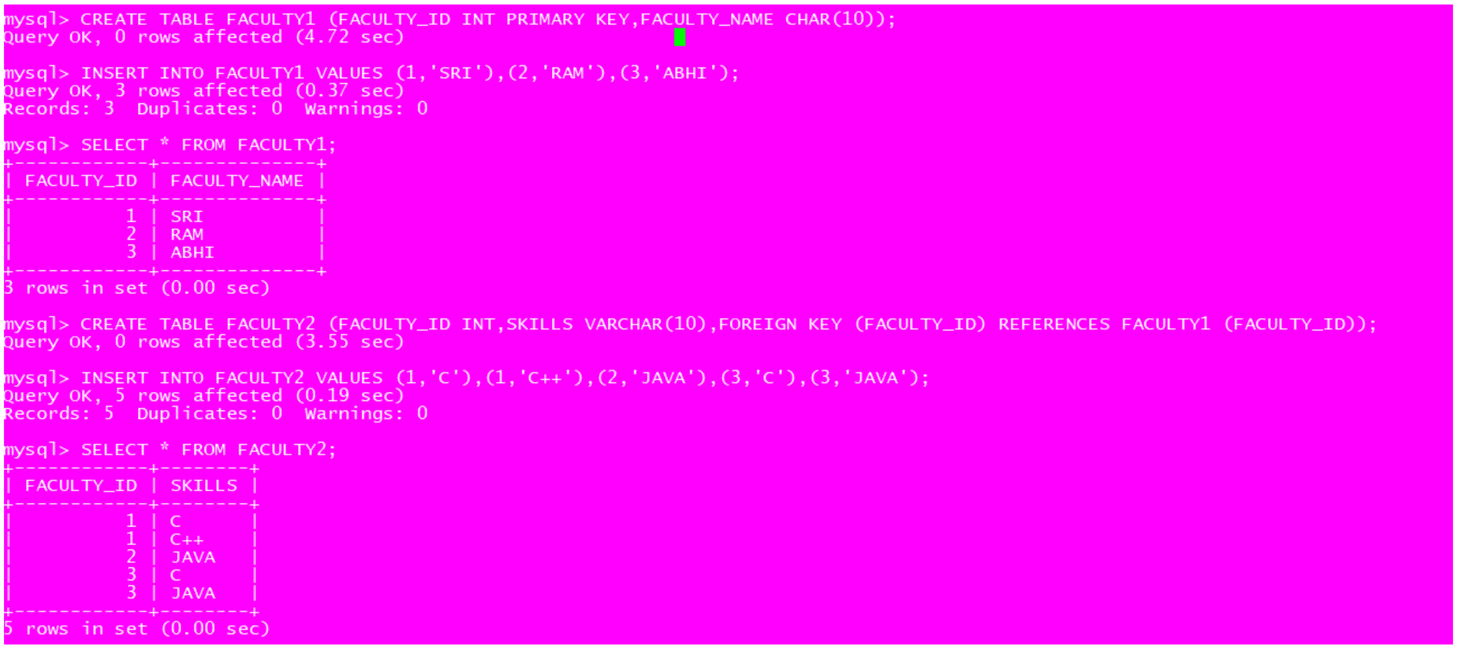
| **Faculty_Id** | **Faculty_Name** | **Skills** |
|---|---|---|
| 1 | Sri | C,C++ |
| 2 | Ram | Java |
| 3 | Abhi | C,Java |

a) Is the Faculty table in 1NF? Why?

A. No it is not in 1NF,As skills attribute in above table has multi value attributes. In 1NF only atomic i.e. single value are allowed.

b) If the Faculty table is not in 1NF, redesign the tables such that all the information currently in the Faculty table is found in the resulting tables, and the resulting tables are in 1NF. For each of the resulting tables, give the table name, column names, primary keys.

A. So solution to bring table to 1NF, here we need to decompose the table as following.



```
mysql> CREATE TABLE FACULTY1 (FACULTY_ID INT PRIMARY KEY,FACULTY_NAME CHAR(10));
Query OK, 0 rows affected (4.72 sec)

mysql> INSERT INTO FACULTY1 VALUES (1,'SRI'),(2,'RAM'),(3,'ABHI');
Query OK, 3 rows affected (0.37 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM FACULTY1;
+------------+--------------+
| FACULTY_ID | FACULTY_NAME |
+------------+--------------+
|          1 | SRI          |
|          2 | RAM          |
|          3 | ABHI         |
+------------+--------------+
3 rows in set (0.00 sec)

mysql> CREATE TABLE FACULTY2 (FACULTY_ID INT,SKILLS VARCHAR(10),FOREIGN KEY (FACULTY_ID) REFERENCES FACULTY1 (FACULTY_ID));
Query OK, 0 rows affected (3.55 sec)

mysql> INSERT INTO FACULTY2 VALUES (1,'C'),(1,'C++'),(2,'JAVA'),(3,'C'),(3,'JAVA');
Query OK, 5 rows affected (0.19 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM FACULTY2;
+------------+--------+
| FACULTY_ID | SKILLS |
+------------+--------+
|          1 | C      |
|          1 | C++    |
|          2 | JAVA   |
|          3 | C      |
|          3 | JAVA   |
+------------+--------+
5 rows in set (0.00 sec)
```

## 2. Exercise 2: 2$^{ND}$ Normal Form (2NF)

Consider the Students table, with the primary key underlined, and the following data:

**Students:**

| **Student_Id** | **Student_Name** | **Student_Address** | **Course_Name** | **Date of completion** |
|---|---|---|---|---|
| 1001 | John | Hyd | C | 20/4/2022 |
| 1002 | Abhi | Chennai | C++ | 15/4/2022 |
| 1003 | Alex | Hyd | Java | 25/04/2022 |
| 1004 | Bob | Bangalore | DBMS | 16/04/2022 |

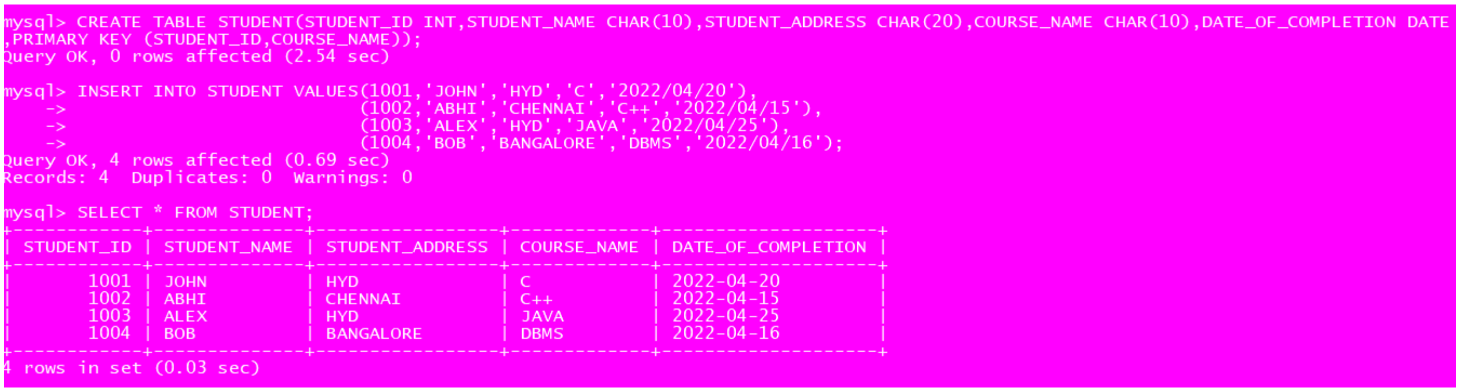
a) Is the Students table in 2 NF? If yes Why? If No Why not?

A. The above given table is in 1NF but not in 2NF. Because STUDENT_NAME, STUDENT_ADDRESS depends on STUDENT_ID but not on COURSE_NAME that makes a partial dependency and partial dependency are not allowed in 2NF .

b) If the Students table is not in 2NF, redesign or decompose the tables such that all the information currently in the Students table is found in the resulting tables, and the resulting tables are in 2 NF. For each of the resulting tables, give the table name, column names, primary keys, and foreign keys.

A. The solution for this is to decompose the table into 2 tables.as

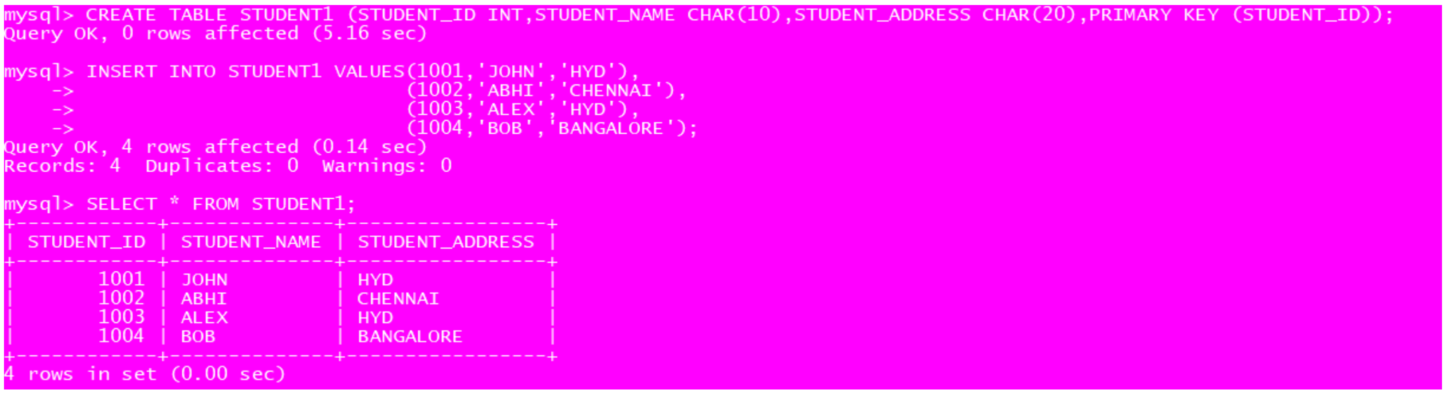
| <u>STUDENT_ID</u> | STUDENT_NAME | STUDENT_ADDRESS | <u>COURSE_NAME</u> | DATEOFCOMPLETION |
|---|---|---|---|---|



```
mysql> CREATE TABLE STUDENT(STUDENT_ID INT,STUDENT_NAME CHAR(10),STUDENT_ADDRESS CHAR(20),COURSE_NAME CHAR(10),DATE_OF_COMPLETION DATE
,PRIMARY KEY (STUDENT_ID,COURSE_NAME));
Query OK, 0 rows affected (2.54 sec)

mysql> INSERT INTO STUDENT VALUES(1001,'JOHN','HYD','C','2022/04/20'),
    ->                          (1002,'ABHI','CHENNAI','C++','2022/04/15'),
    ->                          (1003,'ALEX','HYD','JAVA','2022/04/25'),
    ->                          (1004,'BOB','BANGALORE','DBMS','2022/04/16');
Query OK, 4 rows affected (0.69 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM STUDENT;
+------------+--------------+-----------------+-------------+--------------------+
| STUDENT_ID | STUDENT_NAME | STUDENT_ADDRESS | COURSE_NAME | DATE_OF_COMPLETION |
+------------+--------------+-----------------+-------------+--------------------+
|       1001 | JOHN         | HYD             | C           | 2022-04-20         |
|       1002 | ABHI         | CHENNAI         | C++         | 2022-04-15         |
|       1003 | ALEX         | HYD             | JAVA        | 2022-04-25         |
|       1004 | BOB          | BANGALORE       | DBMS        | 2022-04-16         |
+------------+--------------+-----------------+-------------+--------------------+
4 rows in set (0.03 sec)
```

STUDENT1

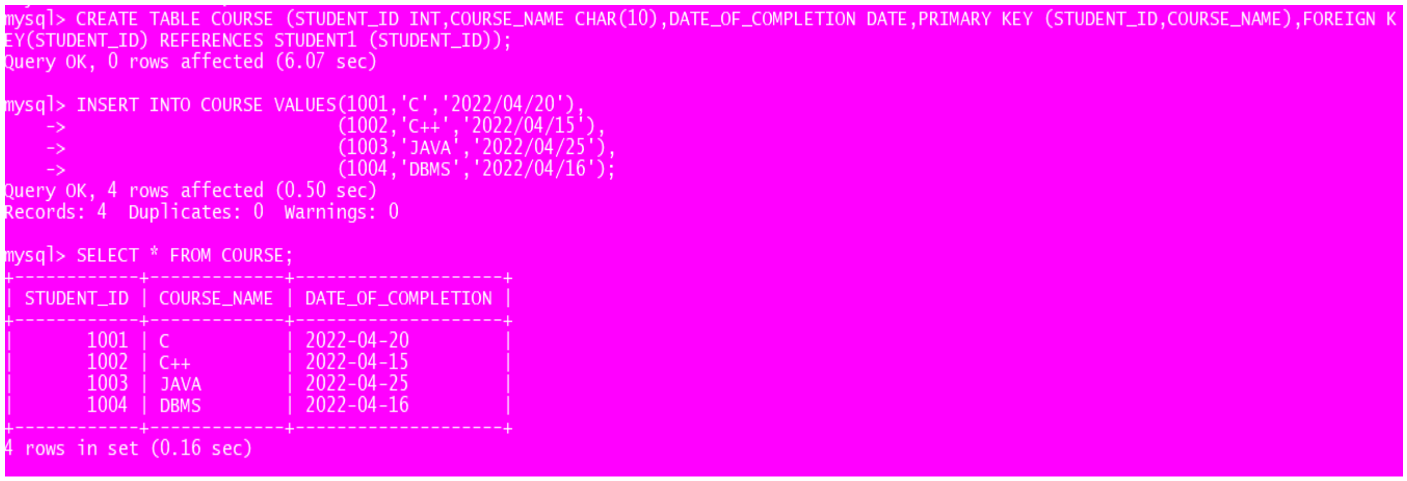
| STUDENT_ID | STUDENT_NAME |
|---|---|



```
mysql> CREATE TABLE STUDENT1 (STUDENT_ID INT,STUDENT_NAME CHAR(10),STUDENT_ADDRESS CHAR(20),PRIMARY KEY (STUDENT_ID));
Query OK, 0 rows affected (5.16 sec)

mysql> INSERT INTO STUDENT1 VALUES(1001,'JOHN','HYD'),
    ->                          (1002,'ABHI','CHENNAI'),
    ->                          (1003,'ALEX','HYD'),
    ->                          (1004,'BOB','BANGALORE');
Query OK, 4 rows affected (0.14 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM STUDENT1;
+------------+--------------+-----------------+
| STUDENT_ID | STUDENT_NAME | STUDENT_ADDRESS |
+------------+--------------+-----------------+
|       1001 | JOHN         | HYD             |
|       1002 | ABHI         | CHENNAI         |
|       1003 | ALEX         | HYD             |
|       1004 | BOB          | BANGALORE       |
+------------+--------------+-----------------+
4 rows in set (0.00 sec)
```

COURSE

| <u>STUDENT_ID</u> | <u>COURSE_NAME</u> |
|---|---|



```
mysql> CREATE TABLE COURSE (STUDENT_ID INT,COURSE_NAME CHAR(10),DATE_OF_COMPLETION DATE,PRIMARY KEY (STUDENT_ID,COURSE_NAME),FOREIGN K
EY(STUDENT_ID) REFERENCES STUDENT1 (STUDENT_ID));
Query OK, 0 rows affected (6.07 sec)

mysql> INSERT INTO COURSE VALUES(1001,'C','2022/04/20'),
    ->                          (1002,'C++','2022/04/15'),
    ->                          (1003,'JAVA','2022/04/25'),
    ->                          (1004,'DBMS','2022/04/16');
Query OK, 4 rows affected (0.50 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM COURSE;
+------------+-------------+--------------------+
| STUDENT_ID | COURSE_NAME | DATE_OF_COMPLETION |
+------------+-------------+--------------------+
|       1001 | C           | 2022-04-20         |
|       1002 | C++         | 2022-04-15         |
|       1003 | JAVA        | 2022-04-25         |
|       1004 | DBMS        | 2022-04-16         |
+------------+-------------+--------------------+
4 rows in set (0.16 sec)
```

## 3. Exercise 3 : 3rd Normal Form (3NF)

Consider the Employee table, with the primary key underlined, and the following data:

**Employee:**

| <u>Emp_no</u> | Emp_Name | Address | Salary | Company_Name | Location |
|---|---|---|---|---|---|
| 101 | Sri | TS | 35000 | C1 | Hyd |
| 102 | Ram | KA | 40000 | C2 | Bangalore |
| 103 | Abhi | TS | 48000 | C3 | Hyd |
| 104 | Hari | TS | 50000 | C1 | Hyd |

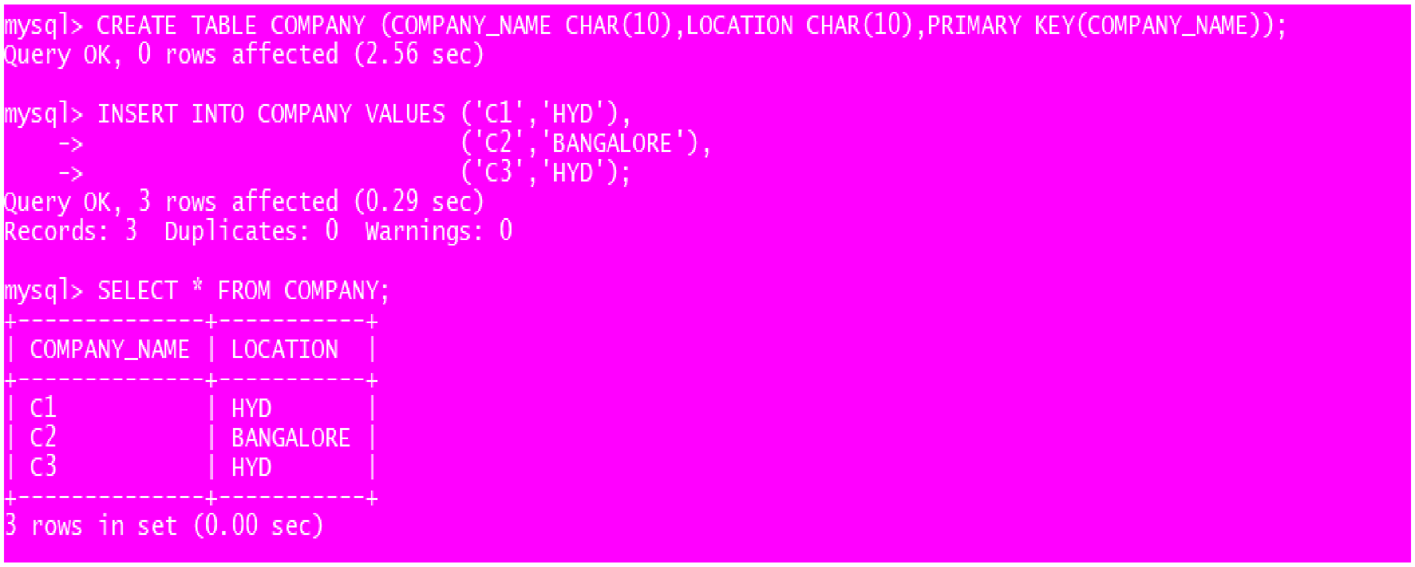
a) Is the Employee table in 3 NF? If yes Why? If No Why not?

A. NO it is not in 3NF, it is in 1NF and also in 2NF.
Here "Location" attribute depends on "Company_name" attribute which makes transitive dependency. Transitive dependency are not allowed in 3NF.

b) If the Employee table is not in 3NF, redesign or decompose the tables such that all the information currently in the Employee table is found in the resulting tables, and the resulting tables are in 3 NF. For each of the resulting tables, give the table name, column names, primary keys, and foreign keys.
A. To covert relation to 3NF,we should decompose as following.

COMPANY

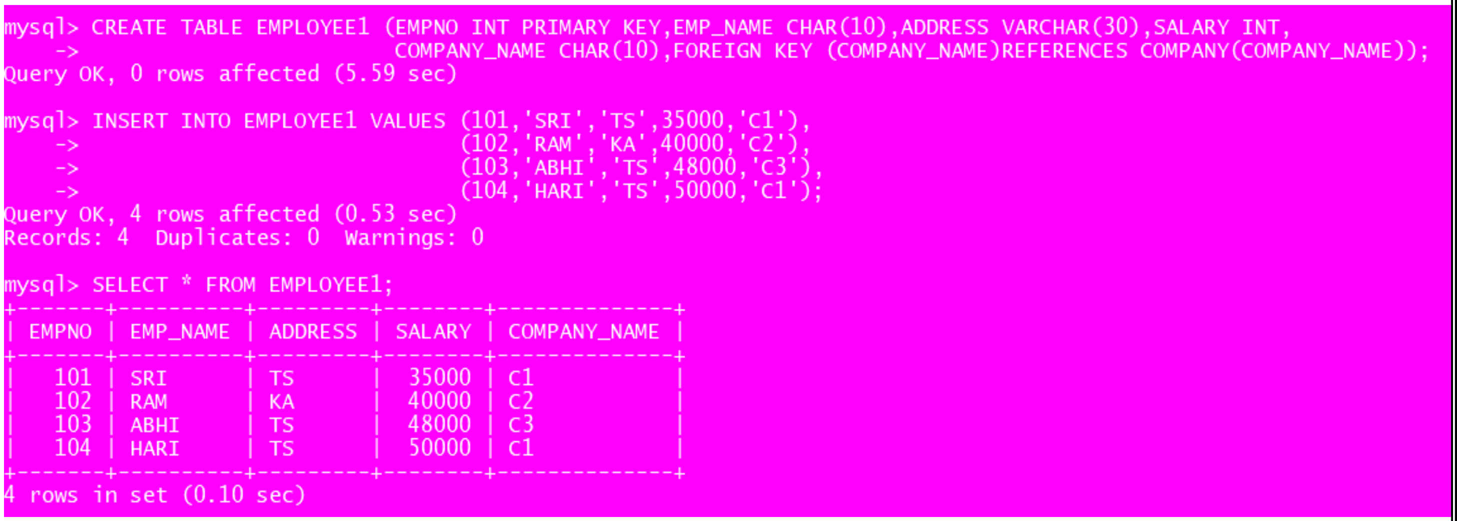
| <u>COMPANY_NAME</u> | LOCATION |
|---|---|



```
mysql> CREATE TABLE COMPANY (COMPANY_NAME CHAR(10),LOCATION CHAR(10),PRIMARY KEY(COMPANY_NAME));
Query OK, 0 rows affected (2.56 sec)

mysql> INSERT INTO COMPANY VALUES ('C1','HYD'),
    ->                            ('C2','BANGALORE'),
    ->                            ('C3','HYD');
Query OK, 3 rows affected (0.29 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM COMPANY;
+--------------+-----------+
| COMPANY_NAME | LOCATION  |
+--------------+-----------+
| C1           | HYD       |
| C2           | BANGALORE |
| C3           | HYD       |
+--------------+-----------+
3 rows in set (0.00 sec)
```

EMPLOYEE1

| EMPNO | EMP_NAME | ADDRESS |
|---|---|---|



```
mysql> CREATE TABLE EMPLOYEE1 (EMPNO INT PRIMARY KEY,EMP_NAME CHAR(10),ADDRESS VARCHAR(30),SALARY INT,
    ->                        COMPANY_NAME CHAR(10),FOREIGN KEY (COMPANY_NAME)REFERENCES COMPANY(COMPANY_NAME));
Query OK, 0 rows affected (5.59 sec)

mysql> INSERT INTO EMPLOYEE1 VALUES (101,'SRI','TS',35000,'C1'),
    ->                             (102,'RAM','KA',40000,'C2'),
    ->                             (103,'ABHI','TS',48000,'C3'),
    ->                             (104,'HARI','TS',50000,'C1');
Query OK, 4 rows affected (0.53 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM EMPLOYEE1;
+-------+----------+---------+--------+--------------+
| EMPNO | EMP_NAME | ADDRESS | SALARY | COMPANY_NAME |
+-------+----------+---------+--------+--------------+
|   101 | SRI      | TS      |  35000 | C1           |
|   102 | RAM      | KA      |  40000 | C2           |
|   103 | ABHI     | TS      |  48000 | C3           |
|   104 | HARI     | TS      |  50000 | C1           |
+-------+----------+---------+--------+--------------+
4 rows in set (0.10 sec)
```

## 4. Exercise 4 : Boyce-Codd Normal Form (BCNF)

| **Student_Id** | **Course** | **Teacher** |
|---|---|---|
| 101 | DBMS | MR.A |
| 101 | JAVA | MR.B |
| 102 | DBMS | MR.C |
| 103 | C++ | MR.D |
| 104 | DBMS | MR.A |

a) Is the Above table in BCNF? If yes Why? If No Why not?

A. NO It is not in BCNF.Here {student_id,course}determines Teacher

and teacher(non key attribute)-------- > course(key attribute)

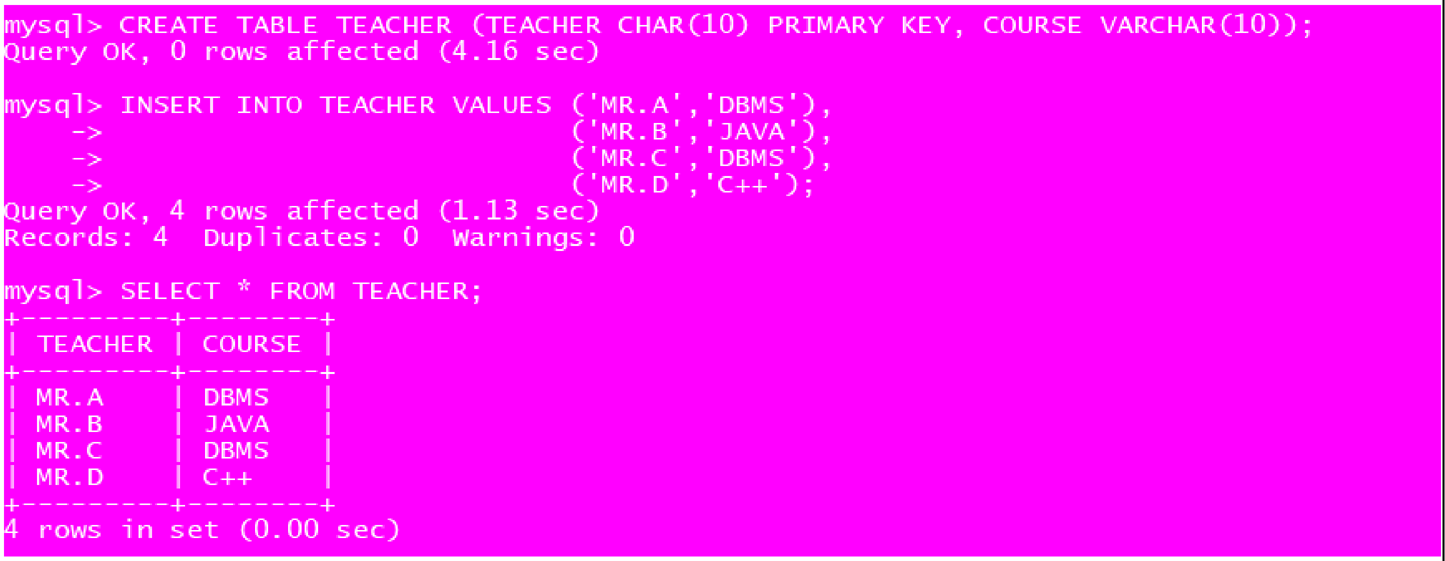
The determinant attribute(here Teacher) has to be a super key.But here Teacher is not a superkey.

b) If the above table is not in BCNF, redesign or decompose the tables such that all the information currently in the above table is found in the resulting tables, and the resulting tables are in BCNF. For eachof the resulting tables, give the table name, column names, primary keys, and foreign keys.

A. So we can get it into BCNF by decomposing as

following tables TEACHER

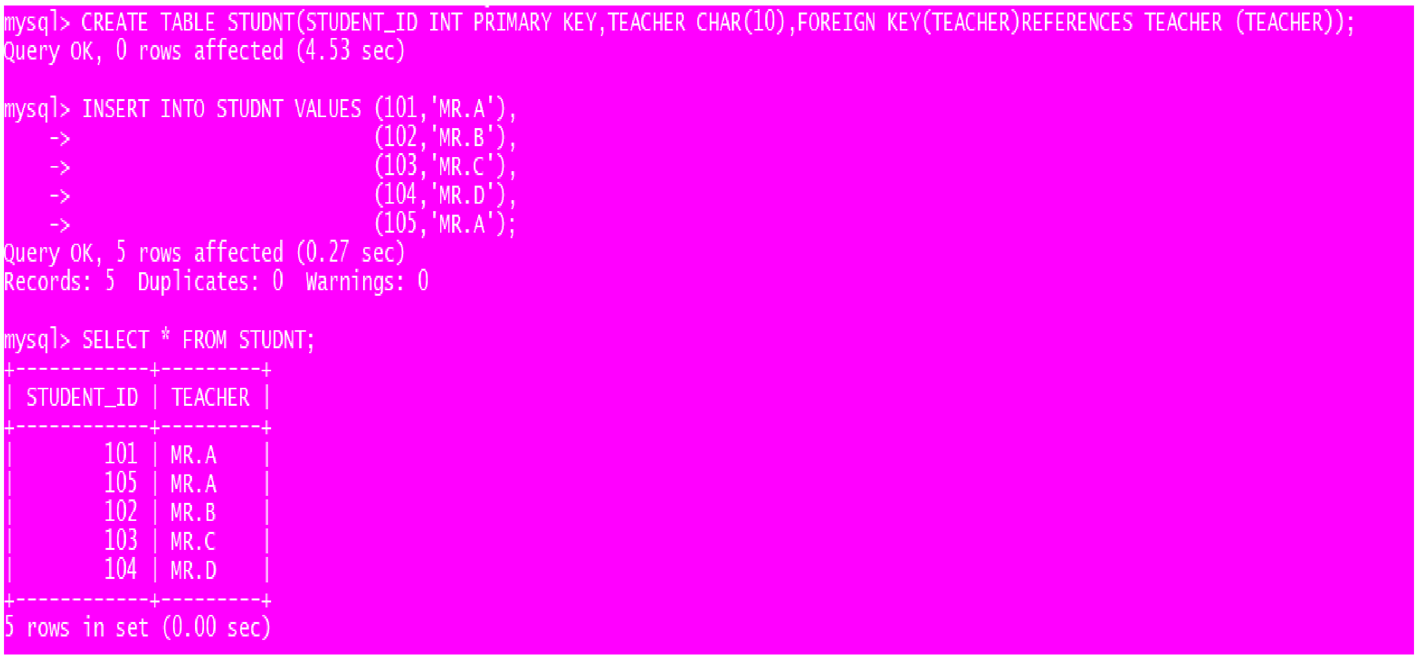
| TEACHER | COURSE |
|---|---|



```
mysql> CREATE TABLE TEACHER (TEACHER CHAR(10) PRIMARY KEY, COURSE VARCHAR(10));
Query OK, 0 rows affected (4.16 sec)

mysql> INSERT INTO TEACHER VALUES ('MR.A','DBMS'),
    ->                            ('MR.B','JAVA'),
    ->                            ('MR.C','DBMS'),
    ->                            ('MR.D','C++');
Query OK, 4 rows affected (1.13 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM TEACHER;
+---------+--------+
| TEACHER | COURSE |
+---------+--------+
| MR.A    | DBMS   |
| MR.B    | JAVA   |
| MR.C    | DBMS   |
| MR.D    | C++    |
+---------+--------+
4 rows in set (0.00 sec)
```

STUDENT

| STUDENT_ID | TEACHER |
|---|---|

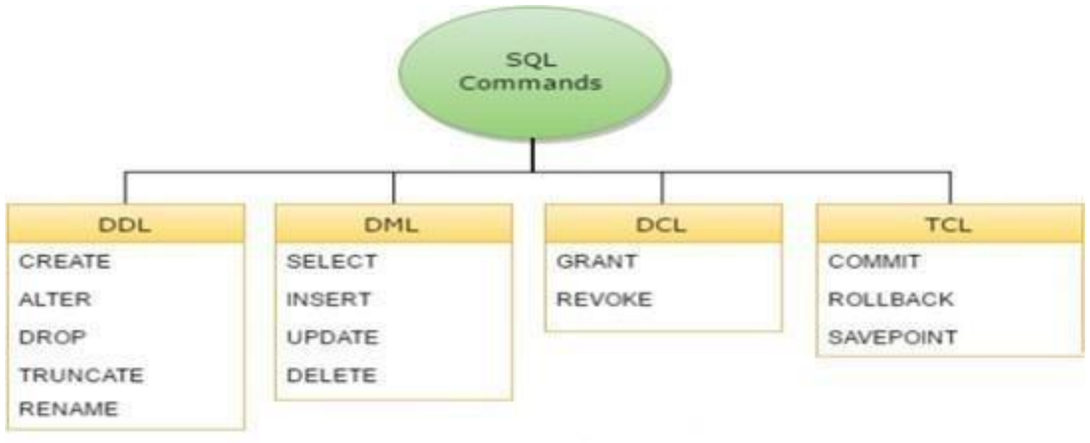


```
mysql> CREATE TABLE STUDNT(STUDENT_ID INT PRIMARY KEY,TEACHER CHAR(10),FOREIGN KEY(TEACHER)REFERENCES TEACHER (TEACHER));
Query OK, 0 rows affected (4.53 sec)

mysql> INSERT INTO STUDNT VALUES (101,'MR.A'),
    ->                           (102,'MR.B'),
    ->                           (103,'MR.C'),
    ->                           (104,'MR.D'),
    ->                           (105,'MR.A');
Query OK, 5 rows affected (0.27 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM STUDNT;
+------------+---------+
| STUDENT_ID | TEACHER |
+------------+---------+
|        101 | MR.A    |
|        105 | MR.A    |
|        102 | MR.B    |
|        103 | MR.C    |
|        104 | MR.D    |
+------------+---------+
5 rows in set (0.00 sec)
```

P is a super key.

## EXPERIMENT- 4
## PRACTICING DDL COMMANDS

**AIM: To Implement DDL commands**



### DDL Commands:

DDL means Data Definition Language. It is used to create an object, alter the structure of an object and also drop already created object.

The Data Definition Languages used for table definition can be classified into following:

• Create table command

• Alter table command

• Truncate table command

• Drop table command

• Rename

| Data Definition Language | |
|---|---|
| **COMMAND** | **DESCRIPTION** |
| CREATE | Create an object. means, create a database, table, triggers,index, functions, stored procedures, etc. |
| ALTER | Used to alter the existing database or its object structures.i.e., tables |
| DROP | This SQL DDL command helps to delete objects. Forexample, delete tables, delete a database, etc. |
| TRUNCATE | This SQL DDL command removes records from tables |
| RENAME | Renaming the database objects |

## 1. CREATION OF TABLES:

SQL - CREATE TABLE: Table is a primary object of database, used to store data in form of rows and columns.

It is created using following command:

### Syntax:

CREATE TABLE tablename (column_name data_ type constraints, …)

### Example:

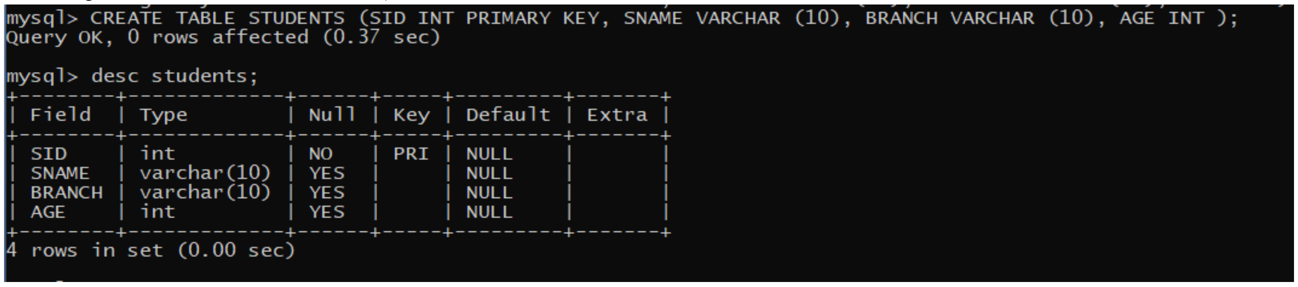
MYSQL > CREATE TABLE STUDENTS ((SID INT PRIMARY KEY, SNAME VARCHAR (10), BRANCH VARCHAR (10), AGE INT );

Table Created.

**Desc command**

The DESCRIBE command is used to view the structure of a table as follows.

MYSQL>DESC STUDENTS;

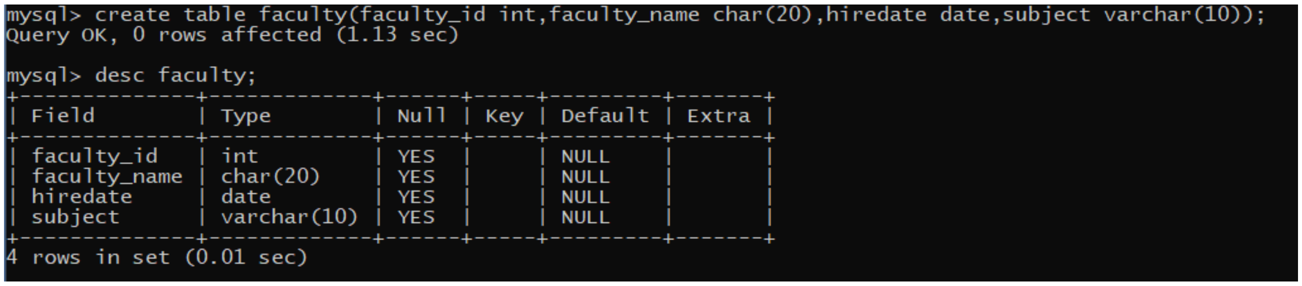


```
mysql> CREATE TABLE STUDENTS (SID INT PRIMARY KEY, SNAME VARCHAR (10), BRANCH VARCHAR (10), AGE INT );
Query OK, 0 rows affected (0.37 sec)

mysql> desc students;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| SID    | int         | NO   | PRI | NULL    |       |
| SNAME  | varchar(10) | YES  |     | NULL    |       |
| BRANCH | varchar(10) | YES  |     | NULL    |       |
| AGE    | int         | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

**EXERCISE:**

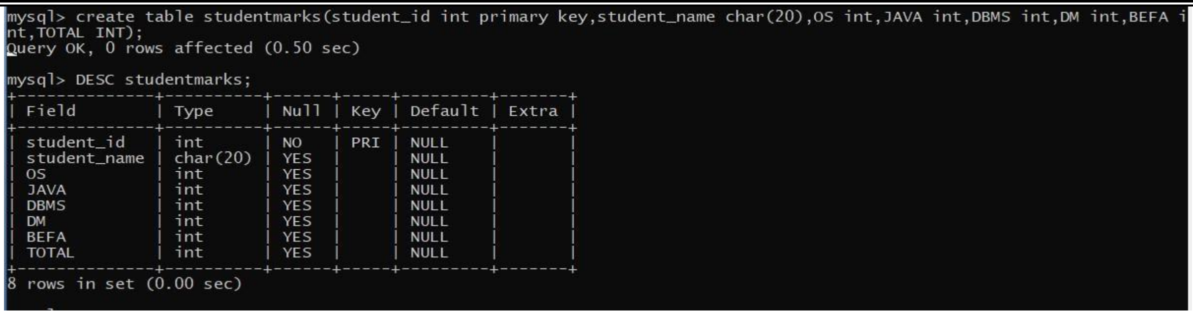
1. Create an FACULTY table with fields (FACULTY_ID , FACULTY_NAME , HIREDATE,SUBJECT) and display using DESCRIBE command..

Sol:



```
mysql> create table faculty(faculty_id int,faculty_name char(20),hiredate date,subject varchar(10));
Query OK, 0 rows affected (1.13 sec)

mysql> desc faculty;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| faculty_id   | int         | YES  |     | NULL    |       |
| faculty_name | char(20)    | YES  |     | NULL    |       |
| hiredate     | date        | YES  |     | NULL    |       |
| subject      | varchar(10) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

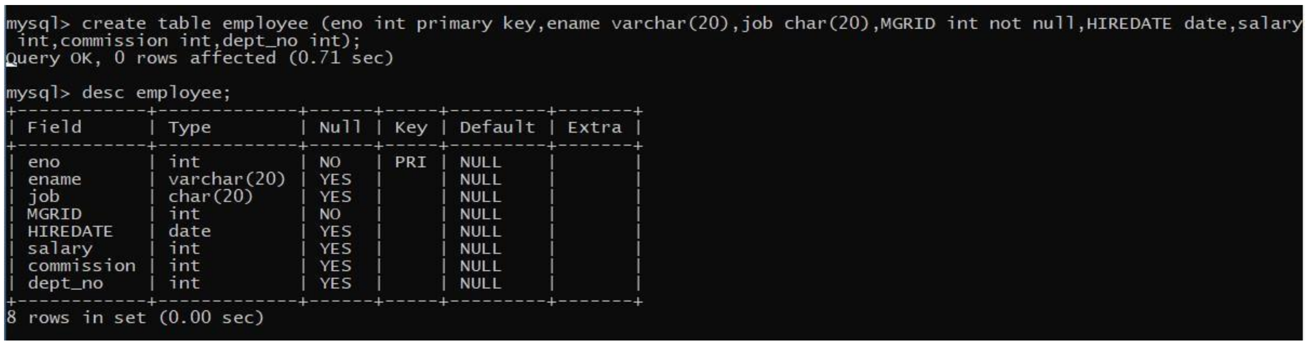
2. Create an STUDENTMARKS table with fields (STUDENT_ID , STUDENT_NAME , OS,JAVA,DBMS,DM,BEFA,TOTAL) and display using DESCRIBE command..



```
mysql> create table studentmarks(student_id int primary key,student_name char(20),OS int,JAVA int,DBMS int,DM int,BEFA i
nt,TOTAL INT);
Query OK, 0 rows affected (0.50 sec)

mysql> DESC studentmarks;
+--------------+----------+------+-----+---------+-------+
| Field        | Type     | Null | Key | Default | Extra |
+--------------+----------+------+-----+---------+-------+
| student_id   | int      | NO   | PRI | NULL    |       |
| student_name | char(20) | YES  |     | NULL    |       |
| OS           | int      | YES  |     | NULL    |       |
| JAVA         | int      | YES  |     | NULL    |       |
| DBMS         | int      | YES  |     | NULL    |       |
| DM           | int      | YES  |     | NULL    |       |
| BEFA         | int      | YES  |     | NULL    |       |
| TOTAL        | int      | YES  |     | NULL    |       |
+--------------+----------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

3. Create an EMPLOYEE table with field (ENO,ENAME,JOB,MGRID,HIREDATE,SALARY,COMMISION,DAPRTMENTNO)

Sol:



```
mysql> create table employee (eno int primary key,ename varchar(20),job char(20),MGRID int not null,HIREDATE date,salary
 int,commission int,dept_no int);
Query OK, 0 rows affected (0.71 sec)

mysql> desc employee;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| eno       | int         | NO   | PRI | NULL    |       |
| ename     | varchar(20) | YES  |     | NULL    |       |
| job       | char(20)    | YES  |     | NULL    |       |
| MGRID     | int         | NO   |     | NULL    |       |
| HIREDATE  | date        | YES  |     | NULL    |       |
| salary    | int         | YES  |     | NULL    |       |
| commission | int        | YES  |     | NULL    |       |
| dept_no   | int         | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

## 2. ALTER TABLE :

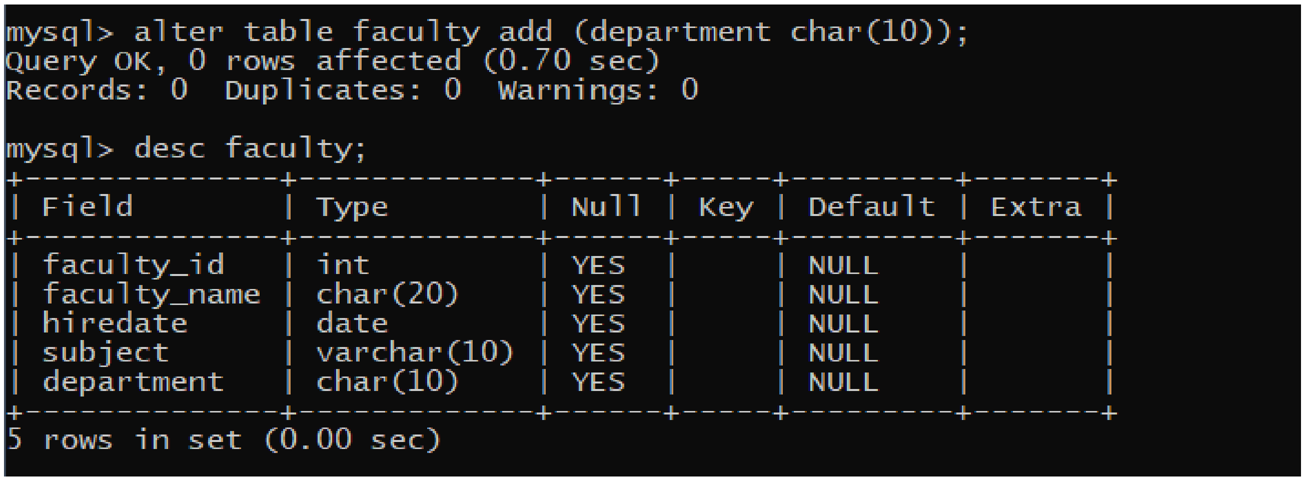
**To ADD a column:**

SYNTAX: ALTER TABLE <TBLE_NAME> ADD (<NEW_COLUMN_NAME> <DATATYPE> (<SIZE>) .................);

**EXERCISE:**

1. Add A Column '**Department**' To FACULTY Table.

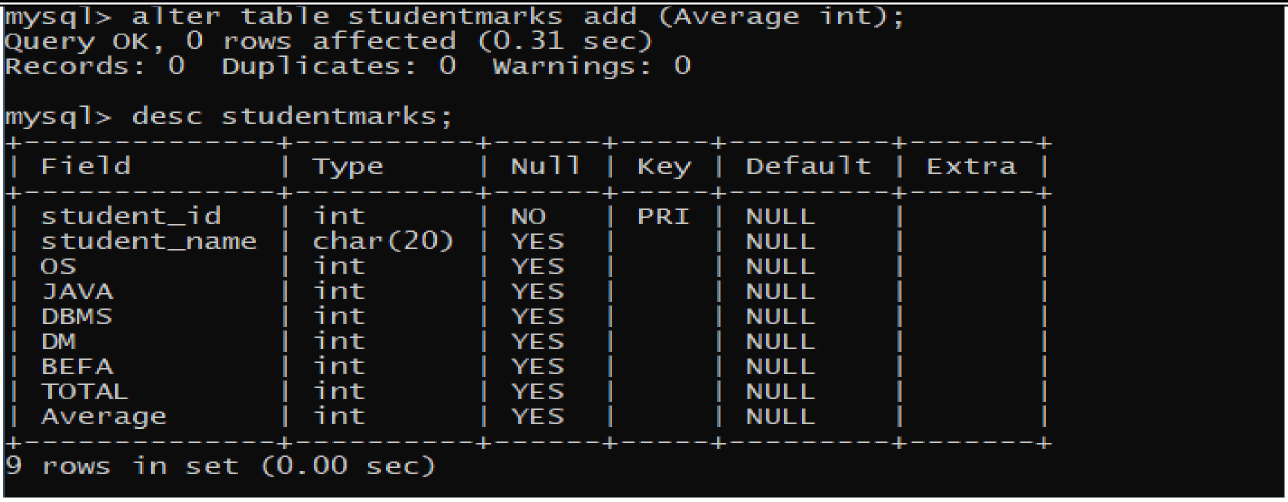
Sol:



```
mysql> alter table faculty add (department char(10));
Query OK, 0 rows affected (0.70 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc faculty;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| faculty_id   | int         | YES  |     | NULL    |       |
| faculty_name | char(20)    | YES  |     | NULL    |       |
| hiredate     | date        | YES  |     | NULL    |       |
| subject      | varchar(10) | YES  |     | NULL    |       |
| department   | char(10)    | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

2. Add a column **"Average"** to STUDENTMARKS Table.

Sol:



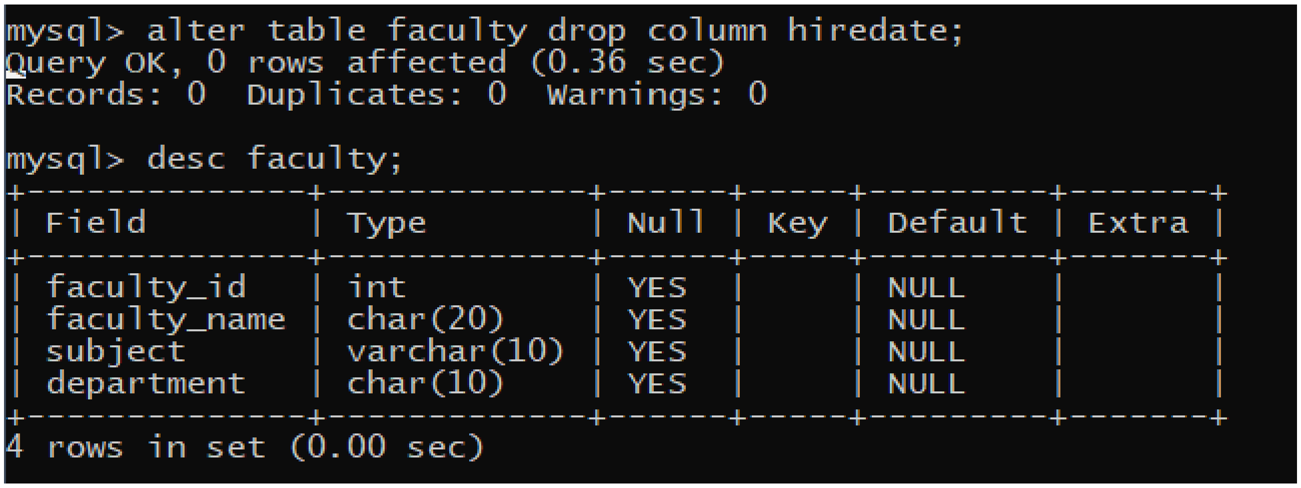
```
mysql> alter table studentmarks add (Average int);
Query OK, 0 rows affected (0.31 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc studentmarks;
+--------------+----------+------+-----+---------+-------+
| Field        | Type     | Null | Key | Default | Extra |
+--------------+----------+------+-----+---------+-------+
| student_id   | int      | NO   | PRI | NULL    |       |
| student_name | char(20) | YES  |     | NULL    |       |
| OS           | int      | YES  |     | NULL    |       |
| JAVA         | int      | YES  |     | NULL    |       |
| DBMS         | int      | YES  |     | NULL    |       |
| DM           | int      | YES  |     | NULL    |       |
| BEFA         | int      | YES  |     | NULL    |       |
| TOTAL        | int      | YES  |     | NULL    |       |
| Average      | int      | YES  |     | NULL    |       |
+--------------+----------+------+-----+---------+-------+
9 rows in set (0.00 sec)
```

**To DROP a column:**

SYNTAX: ALTER TABLE <TABLE_NAME>DROP COLUMN<COLUMN_NAME> ;.

**EXERCISE:**

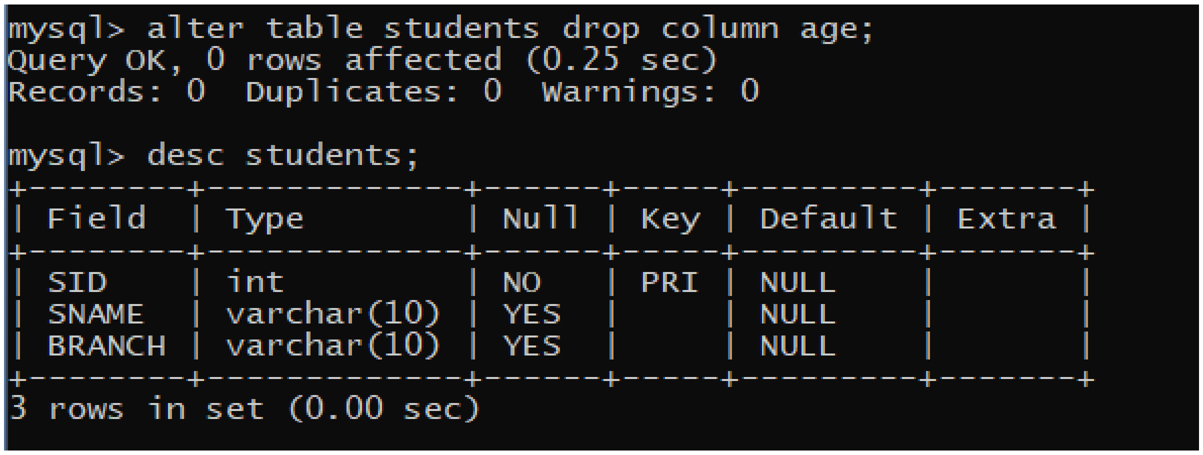
1. Drop Column **'HIREDATE'** From FACULTY Table.



```
mysql> alter table faculty drop column hiredate;
Query OK, 0 rows affected (0.36 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc faculty;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| faculty_id   | int         | YES  |     | NULL    |       |
| faculty_name | char(20)    | YES  |     | NULL    |       |
| subject      | varchar(10) | YES  |     | NULL    |       |
| department   | char(10)    | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

2. Drop column **'AGE'** from STUDENTS Table.

Sol:



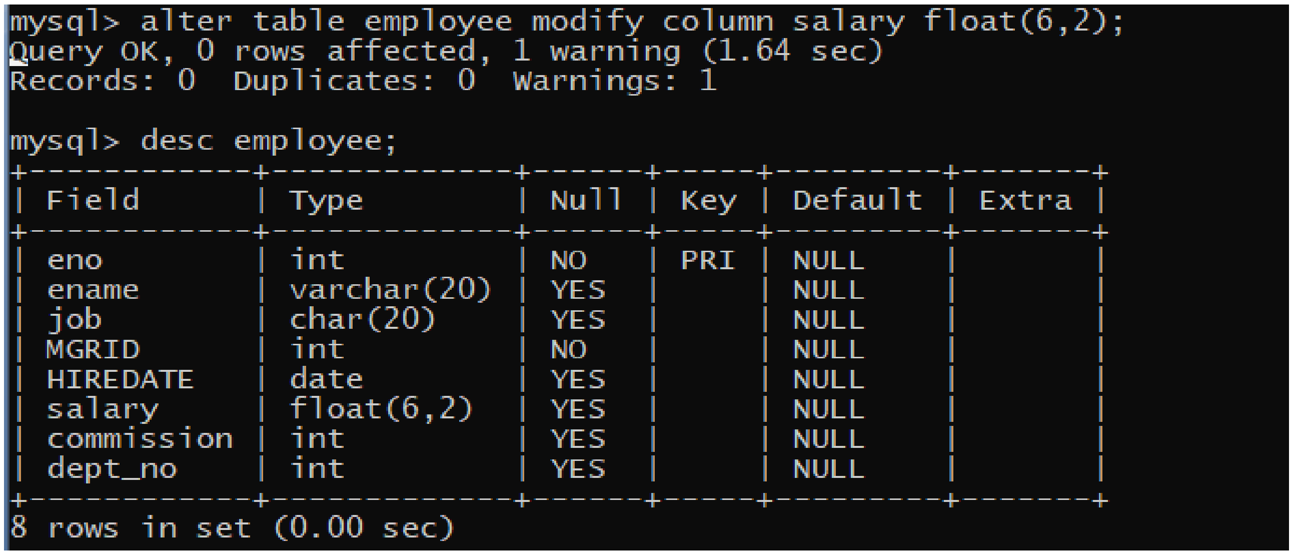
```
mysql> alter table students drop column age;
Query OK, 0 rows affected (0.25 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc students;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| SID    | int         | NO   | PRI | NULL    |       |
| SNAME  | varchar(10) | YES  |     | NULL    |       |
| BRANCH | varchar(10) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

**To MODIFY a column:**

SYNTAX: ALTER TABLE <TABLE_NAME>MODIFY COLUMN<COLUMN_NAME><NEW_DATATYPE> (<NEWSIZE>);

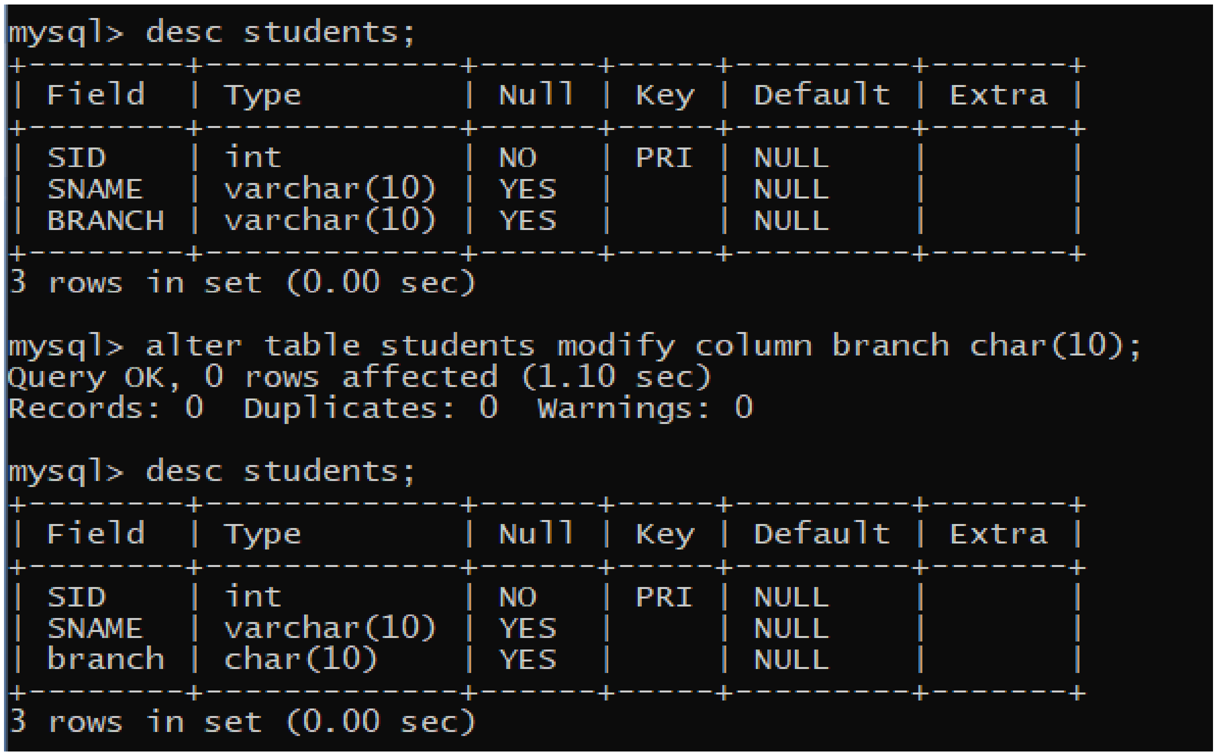
**EXERCISE:**

1. Modify Column **'SALARY' DATATYPE** of EMPLOYEE Table From INT to FLOAT.



```
mysql> alter table employee modify column salary float(6,2);
Query OK, 0 rows affected, 1 warning (1.64 sec)
Records: 0  Duplicates: 0  Warnings: 1

mysql> desc employee;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| eno        | int         | NO   | PRI | NULL    |       |
| ename      | varchar(20) | YES  |     | NULL    |       |
| job        | char(20)    | YES  |     | NULL    |       |
| MGRID      | int         | NO   |     | NULL    |       |
| HIREDATE   | date        | YES  |     | NULL    |       |
| salary     | float(6,2)  | YES  |     | NULL    |       |
| commission | int         | YES  |     | NULL    |       |
| dept_no    | int         | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

2. Modify column **'BRANCH' DATATYPE** of STUDENTS Table from VARCHAR TO CHAR with different size.



```
mysql> desc students;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| SID    | int         | NO   | PRI | NULL    |       |
| SNAME  | varchar(10) | YES  |     | NULL    |       |
| BRANCH | varchar(10) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> alter table students modify column branch char(10);
Query OK, 0 rows affected (1.10 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc students;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| SID    | int         | NO   | PRI | NULL    |       |
| SNAME  | varchar(10) | YES  |     | NULL    |       |
| branch | char(10)    | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

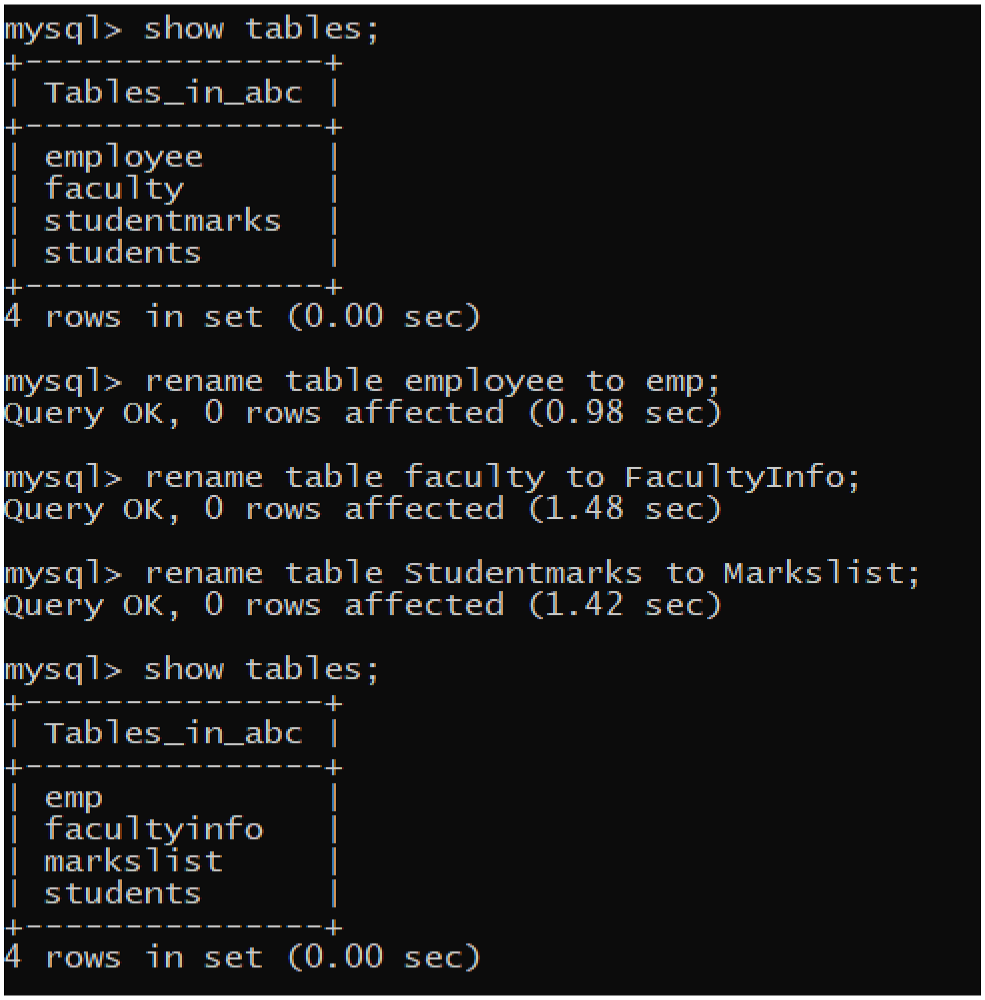
## 3. RENAME A TABLE

Rename command is used to give new names for existing tables.

SYNTAX:
MYSQL> RENAME table oldtablename TO newtablename;

**EXERCISE:**

1. PRACTICE THE COMMAND BY RENAMING THE ALREADY CREATED TABLES



```
mysql> show tables;
+---------------+
| Tables_in_abc |
+---------------+
| employee      |
| faculty       |
| studentmarks  |
| students      |
+---------------+
4 rows in set (0.00 sec)

mysql> rename table employee to emp;
Query OK, 0 rows affected (0.98 sec)

mysql> rename table faculty to FacultyInfo;
Query OK, 0 rows affected (1.48 sec)

mysql> rename table Studentmarks to Markslist;
Query OK, 0 rows affected (1.42 sec)

mysql> show tables;
+---------------+
| Tables_in_abc |
+---------------+
| emp           |
| facultyinfo   |
| markslist     |
| students      |
+---------------+
4 rows in set (0.00 sec)
```

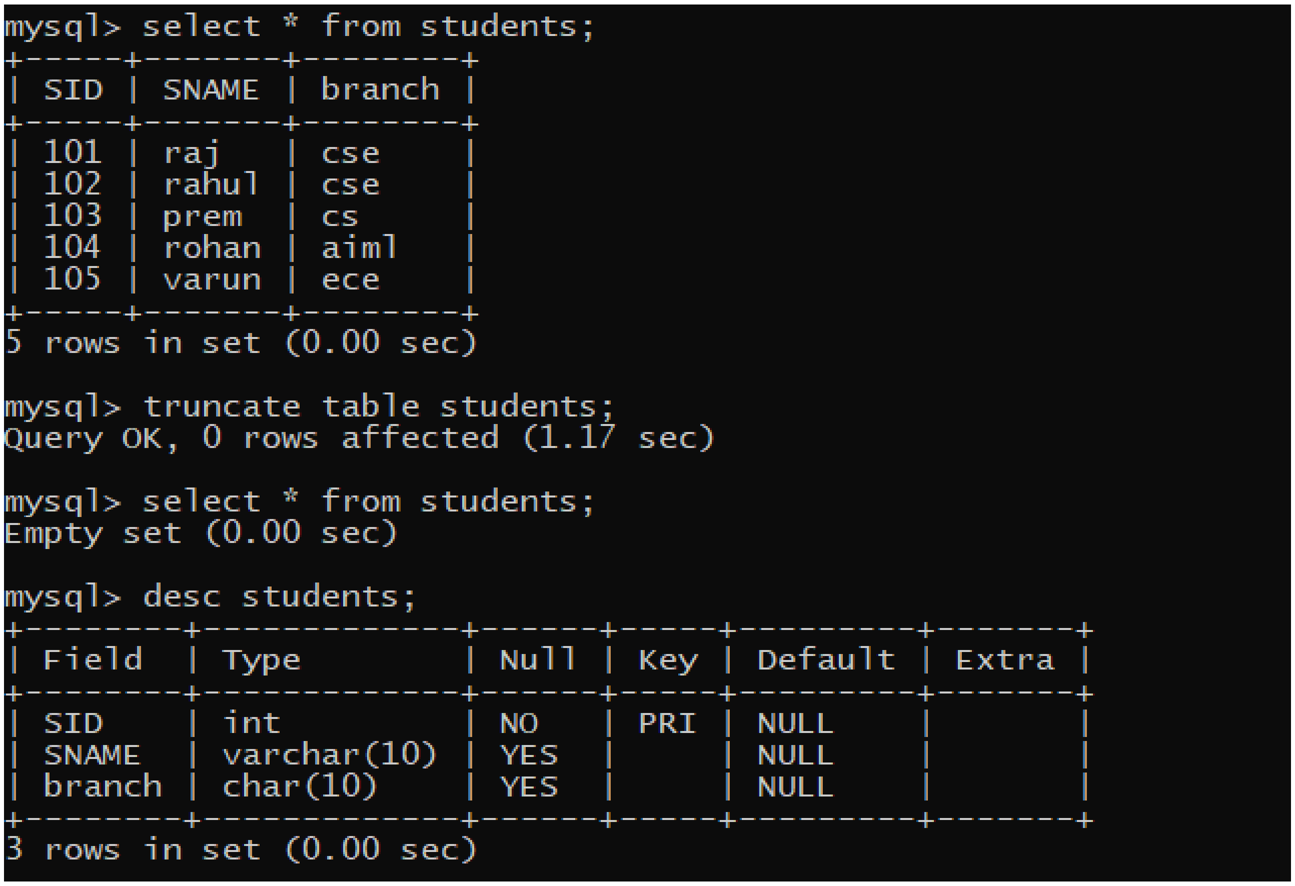
## 4. TRUNCATE A TABLE

Truncate command is used to delete all records from a table.

SYNTAX:
MYSQL> TRUNCATE TABLE tablename;

**EXERCISE:**

1. PRACTICE THE COMMAND ON THE ALREADY CREATED TABLES



```
mysql> select * from students;
+-----+-------+--------+
| SID | SNAME | branch |
+-----+-------+--------+
| 101 | raj   | cse    |
| 102 | rahul | cse    |
| 103 | prem  | cs     |
| 104 | rohan | aiml   |
| 105 | varun | ece    |
+-----+-------+--------+
5 rows in set (0.00 sec)

mysql> truncate table students;
Query OK, 0 rows affected (1.17 sec)

mysql> select * from students;
Empty set (0.00 sec)

mysql> desc students;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| SID    | int         | NO   | PRI | NULL    |       |
| SNAME  | varchar(10) | YES  |     | NULL    |       |
| branch | char(10)    | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

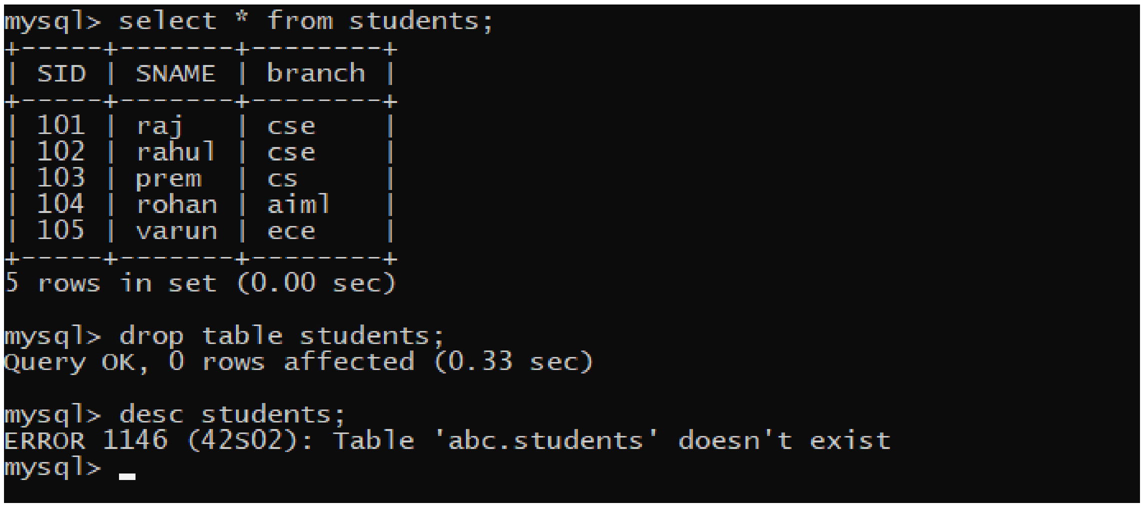
**5. DROP A TABLE**

Drop command is used to remove an existing table permanently from database.

SYNTAX:
MYSQL> DROP TABLE tablename;

**EXERCISE:**

1. PRACTICE THE COMMAND BY DROPING THE ALREADY CREATED TABLES



```
mysql> select * from students;
+-----+-------+--------+
| SID | SNAME | branch |
+-----+-------+--------+
| 101 | raj   | cse    |
| 102 | rahul | cse    |
| 103 | prem  | cs     |
| 104 | rohan | aiml   |
| 105 | varun | ece    |
+-----+-------+--------+
5 rows in set (0.00 sec)

mysql> drop table students;
Query OK, 0 rows affected (0.33 sec)

mysql> desc students;
ERROR 1146 (42S02): Table 'abc.students' doesn't exist
mysql> _
```

# EXPERIMENT- 5
# PRACTICING DML COMMANDS

**AIM: To Implement DML commands**

**DML: Data Manipulation Language (DML)** statements are used for managing data within schema objects and to manipulate data of a database objects.

DML Commands: Insert, Update, Delete, Select

INSERT - insert data into a table

UPDATE - updates existing data within a table

DELETE - deletes all records from a table, the space for the records remain

SELECT - retrieve data from the a database

1. **Insert:** Inserting data into tables.

**Syntax**: INSERT INTO TABLE_NAME (column1, column2, column3,...columnN)

VALUES (value1, value2, value3,...valueN);

OR

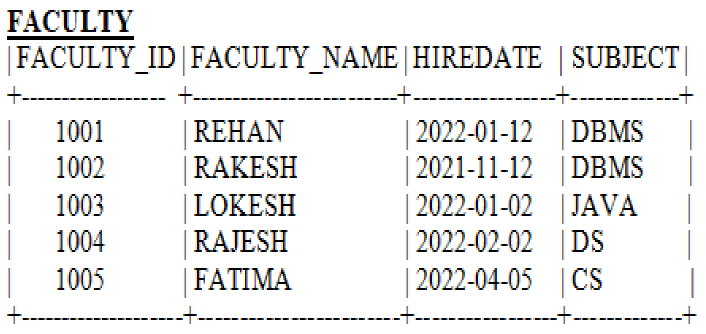
INSERT INTO TABLE_NAME VALUES(value1, value2, value3,...valueN);

## Exercise:

Insert any five records into the tables STUDENTS, FACULTY, EMPLOYEE.

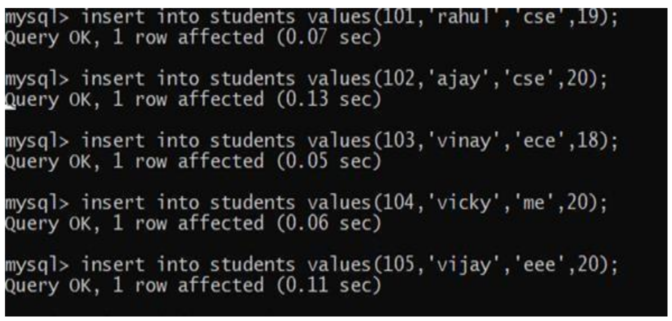
**STUDENTS**

| SID | SNAME | BRANCH | AGE |
|---|---|---|---|
| 101 | RAHUL | CSE | 19 |
| 102 | AJAY | CSE | 20 |
| 103 | VINAY | ECE | 18 |
| 104 | VICKY | ME | 20 |
| 105 | VIJAY | EEE | 20 |

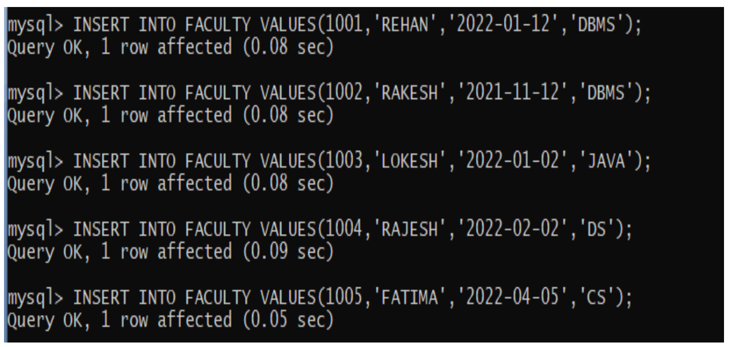


**FACULTY**

| FACULTY_ID | FACULTY_NAME | HIREDATE | SUBJECT |
|---|---|---|---|
| 1001 | REHAN | 2022-01-12 | DBMS |
| 1002 | RAKESH | 2021-11-12 | DBMS |
| 1003 | LOKESH | 2022-01-02 | JAVA |
| 1004 | RAJESH | 2022-02-02 | DS |
| 1005 | FATIMA | 2022-04-05 | CS |

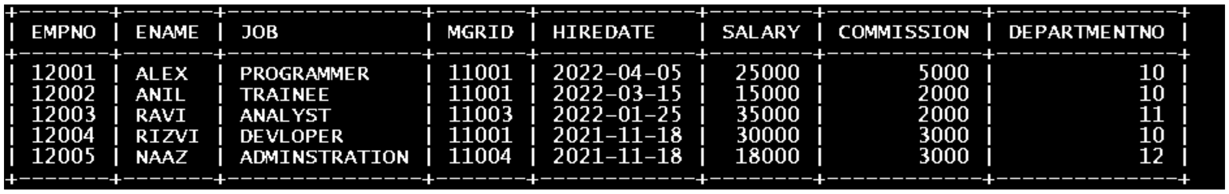


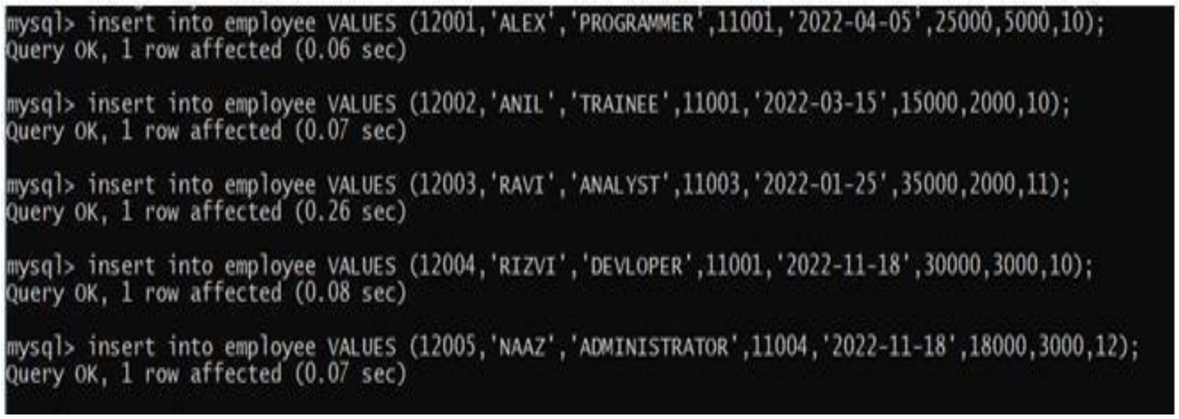
```
mysql> insert into students values(101,'rahul','cse',19);
Query OK, 1 row affected (0.07 sec)

mysql> insert into students values(102,'ajay','cse',20);
Query OK, 1 row affected (0.13 sec)

mysql> insert into students values(103,'vinay','ece',18);
Query OK, 1 row affected (0.05 sec)

mysql> insert into students values(104,'vicky','me',20);
Query OK, 1 row affected (0.06 sec)

mysql> insert into students values(105,'vijay','eee',20);
Query OK, 1 row affected (0.11 sec)
```



```
mysql> INSERT INTO FACULTY VALUES(1001,'REHAN','2022-01-12','DBMS');
Query OK, 1 row affected (0.08 sec)

mysql> INSERT INTO FACULTY VALUES(1002,'RAKESH','2021-11-12','DBMS');
Query OK, 1 row affected (0.08 sec)

mysql> INSERT INTO FACULTY VALUES(1003,'LOKESH','2022-01-02','JAVA');
Query OK, 1 row affected (0.08 sec)

mysql> INSERT INTO FACULTY VALUES(1004,'RAJESH','2022-02-02','DS');
Query OK, 1 row affected (0.09 sec)

mysql> INSERT INTO FACULTY VALUES(1005,'FATIMA','2022-04-05','CS');
Query OK, 1 row affected (0.05 sec)
```

**EMPLOYEE**


```
+-------+-------+---------------+-------+------------+--------+------------+--------------+
| EMPNO | ENAME | JOB           | MGRID | HIREDATE   | SALARY | COMMISSION | DEPARTMENTNO |
+-------+-------+---------------+-------+------------+--------+------------+--------------+
| 12001 | ALEX  | PROGRAMMER    | 11001 | 2022-04-05 |  25000 |       5000 |           10 |
| 12002 | ANIL  | TRAINEE       | 11001 | 2022-03-15 |  15000 |       2000 |           10 |
| 12003 | RAVI  | ANALYST       | 11003 | 2022-01-25 |  35000 |       2000 |           11 |
| 12004 | RIZVI | DEVLOPER      | 11001 | 2021-11-18 |  30000 |       3000 |           10 |
| 12005 | NAAZ  | ADMINSTRATION | 11004 | 2021-11-18 |  18000 |       3000 |           12 |
+-------+-------+---------------+-------+------------+--------+------------+--------------+
```

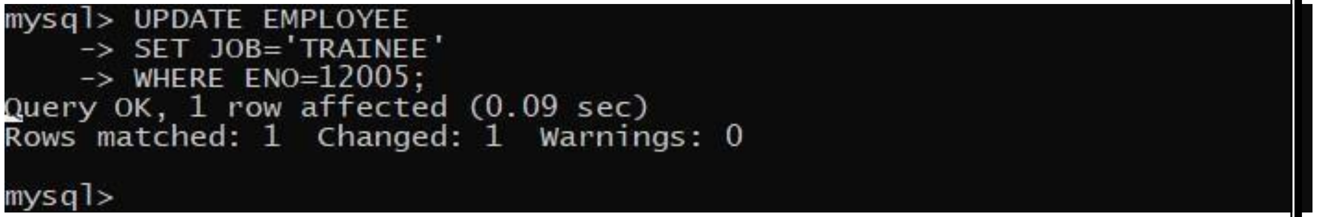

```
mysql> insert into employee VALUES (12001,'ALEX','PROGRAMMER',11001,'2022-04-05',25000,5000,10);
Query OK, 1 row affected (0.06 sec)

mysql> insert into employee VALUES (12002,'ANIL','TRAINEE',11001,'2022-03-15',15000,2000,10);
Query OK, 1 row affected (0.07 sec)

mysql> insert into employee VALUES (12003,'RAVI','ANALYST',11003,'2022-01-25',35000,2000,11);
Query OK, 1 row affected (0.26 sec)

mysql> insert into employee VALUES (12004,'RIZVI','DEVLOPER',11001,'2022-11-18',30000,3000,10);
Query OK, 1 row affected (0.08 sec)

mysql> insert into employee VALUES (12005,'NAAZ','ADMINISTRATOR',11004,'2022-11-18',18000,3000,12);
Query OK, 1 row affected (0.07 sec)
```

**2. Update**: Updating the column details of tables

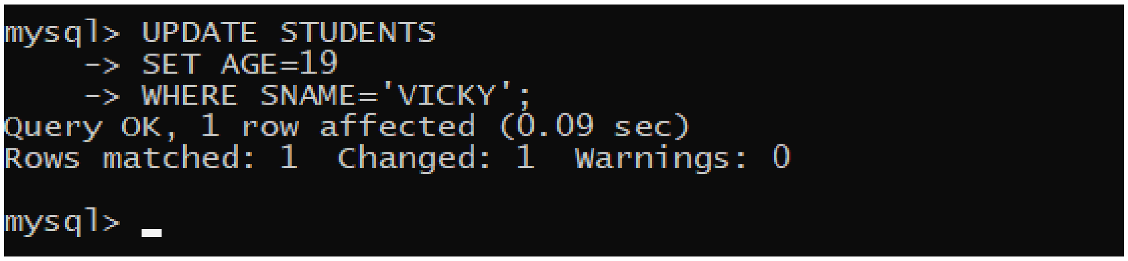
**Syntax:** UPDATE table_name
SET column1 = value1, column2 = value2. .. , columnN = valueN
WHERE [condition];

**Exercise:**

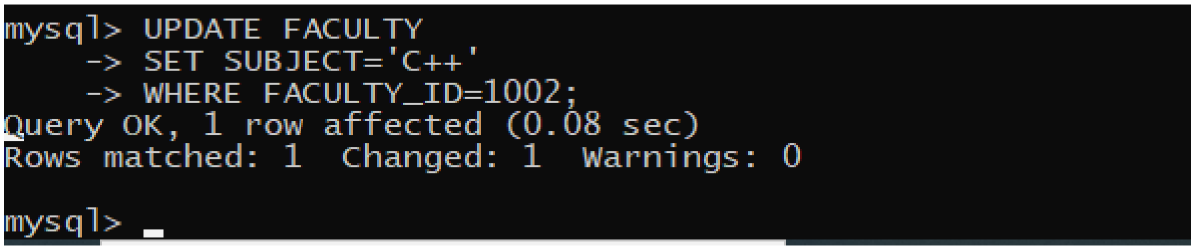
1. Update the job = trainee of employee with empno=12005.


```
mysql> UPDATE EMPLOYEE
    -> SET JOB='TRAINEE'
    -> WHERE ENO=12005;
Query OK, 1 row affected (0.09 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
```

2. Update the age = 19 of student vicky.

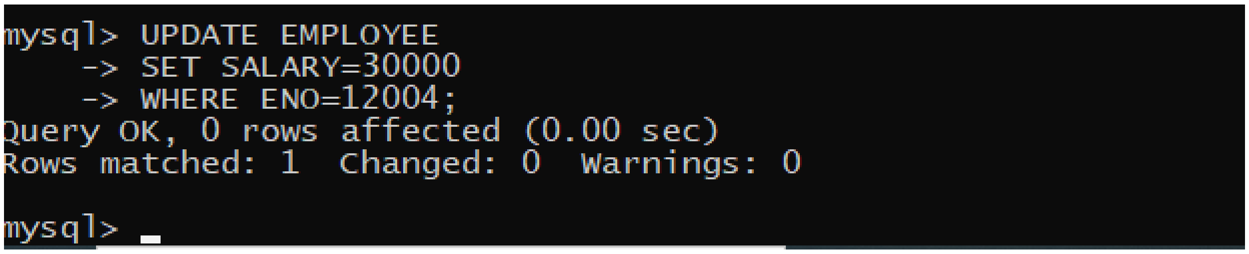

```
mysql> UPDATE STUDENTS
    -> SET AGE=19
    -> WHERE SNAME='VICKY';
Query OK, 1 row affected (0.09 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
```

3. Update the subject = C++ of faculty with facultyid=1002.



```
mysql> UPDATE FACULTY
    -> SET SUBJECT='C++'
    -> WHERE FACULTY_ID=1002;
Query OK, 1 row affected (0.08 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> _
```

4. Update the salary = 30000 of employee with empno=12003.

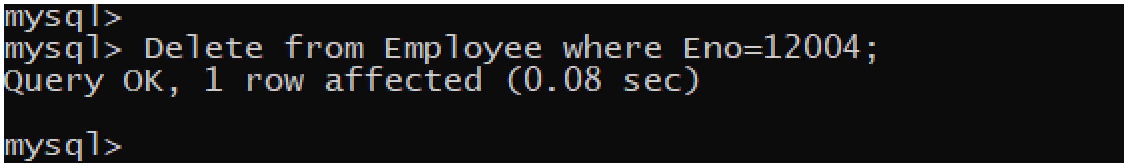


```
mysql> UPDATE EMPLOYEE
    -> SET SALARY=30000
    -> WHERE ENO=12004;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0

mysql> _
```

**3. DELETE:**Deleting records from tables

**Syntax:** DELETE FROM table_name WHERE condition;

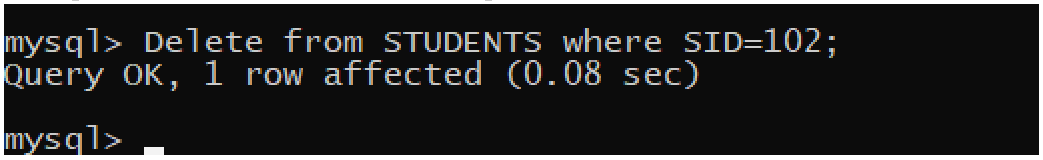
**<u>Example:</u>** Delete from Employee where Empno=12004;



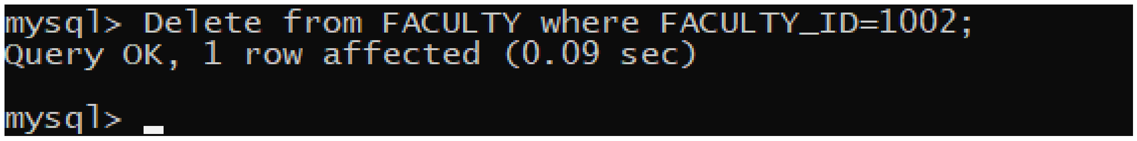
```
mysql>
mysql> Delete from Employee where Eno=12004;
Query OK, 1 row affected (0.08 sec)

mysql>
```

**<u>Exercise:</u>**

practice the delete command on previous created table.



```
mysql> Delete from STUDENTS where SID=102;
Query OK, 1 row affected (0.08 sec)

mysql> _
```



```
mysql> Delete from FACULTY where FACULTY_ID=1002;
Query OK, 1 row affected (0.09 sec)

mysql> _
```

**4. SELECT:**

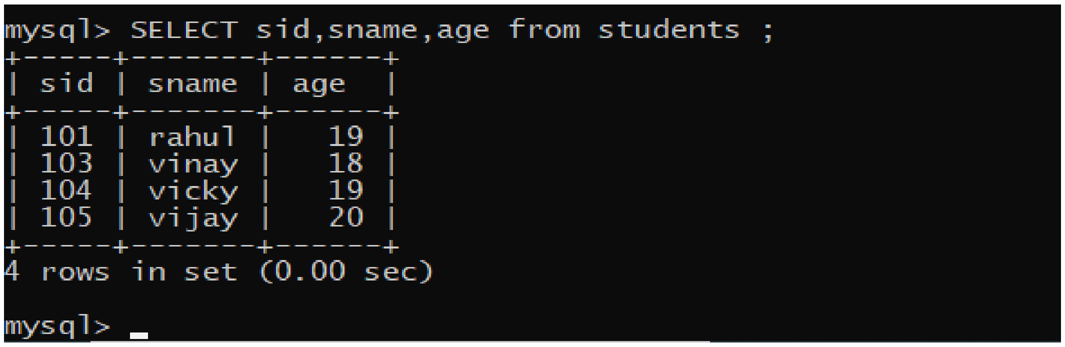
**Syntax:** SELECT column1, column2, ... FROM table_name;

or

SELECT * FROM table_name;

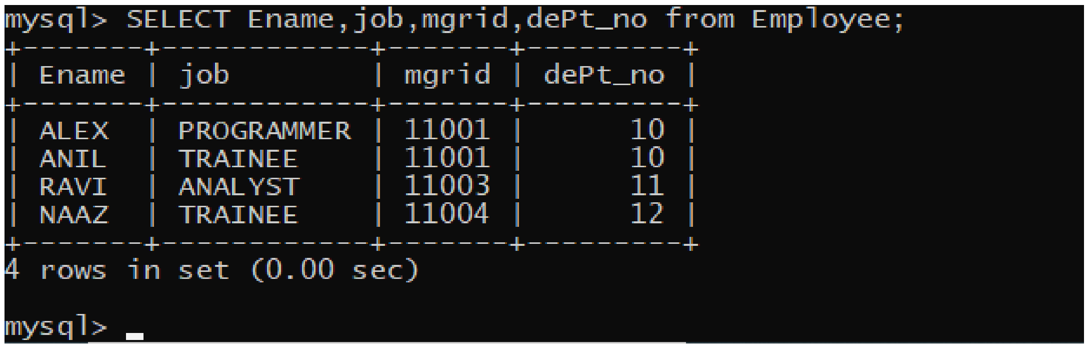
**Exercise:**

1. Display sid,sname and age from students table;



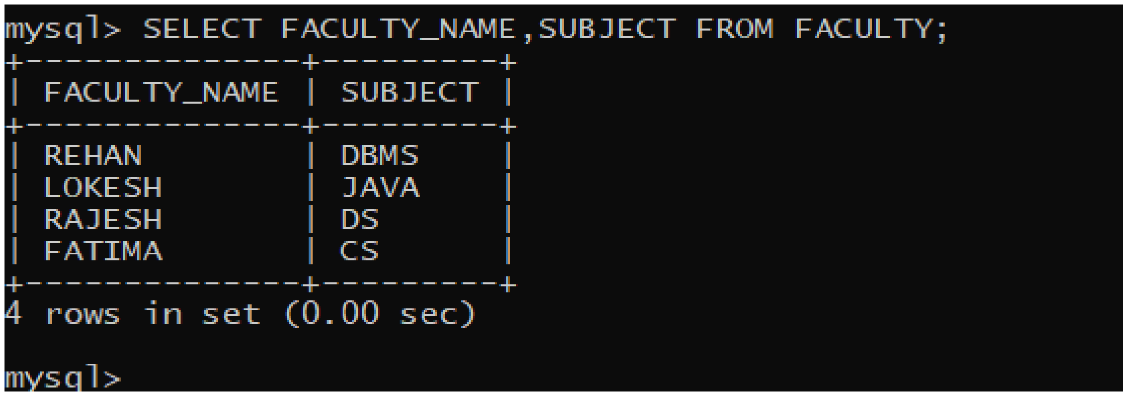
```
mysql> SELECT sid,sname,age from students ;
+-----+-------+------+
| sid | sname | age  |
+-----+-------+------+
| 101 | rahul |   19 |
| 103 | vinay |   18 |
| 104 | vicky |   19 |
| 105 | vijay |   20 |
+-----+-------+------+
4 rows in set (0.00 sec)

mysql> _
```

2. Dispaly Ename,job,mgrid,departmentno from Employee table.



```
mysql> SELECT Ename,job,mgrid,dePt_no from Employee;
+-------+------------+-------+---------+
| Ename | job        | mgrid | dePt_no |
+-------+------------+-------+---------+
| ALEX  | PROGRAMMER | 11001 |      10 |
| ANIL  | TRAINEE    | 11001 |      10 |
| RAVI  | ANALYST    | 11003 |      11 |
| NAAZ  | TRAINEE    | 11004 |      12 |
+-------+------------+-------+---------+
4 rows in set (0.00 sec)

mysql> _
```

3. Dispaly Facultyname and subject from Faculty table.



```
mysql> SELECT FACULTY_NAME,SUBJECT FROM FACULTY;
+--------------+---------+
| FACULTY_NAME | SUBJECT |
+--------------+---------+
| REHAN        | DBMS    |
| LOKESH       | JAVA    |
| RAJESH       | DS      |
| FATIMA       | CS      |
+--------------+---------+
4 rows in set (0.00 sec)

mysql>
```

**VIVA QUESTIONS:**

1. What is the full form of DML?

A. DML is an abbreviation of **Data Manipulation Language**.

2. Why is DML provided?

A. Data Manipulation Language which deals with data manipulation and it is used to store, modify, retrieve, delete and update data in a database.

3. What are the 3 DML commands?

A. DML commands include **SELECT, INSERT, UPDATE, and DELETE**.

4. Difference between DDL and DML?

A. DDL is Data Definition Language which is used to define data structures. For example: create table, alter table are instructions in MYSQL.

DML is Data Manipulation Language which is used to manipulate data itself. For example: insert, update, delete are instructions in MYSQL.

5. What is the syntax for Update command?

```
AUPDATE table_name
            SET column1 = value1, column2 = value2. .. , columnN = valueN
            WHERE [condition];
```

6. What happens when we give delete command on table?

A. It deletes the record from given table name with help of where condition.

7. Update command are used along with which conditions?

A. Update command are used along with SET and WHERE conditions.

8. Which is the most commonly used Dml command?

A.SELECT.

9. For what Insert command is used for?

A. To Insert the new data into the table.

10. What is the difference between Delete and drop command?

A. DELETE is a Data Manipulation Language command, DML command and is used to remove tuple /records from a relation/table. Whereas DROP is a Data Definition Language, DDL command and is used to remove named elements of schema like relations/table, constraints or entire schema

**EXPERIMENT – 6**

**Querying (using ANY, ALL, IN, Exists, NOT EXISTS, UNION, Constraints etc.)**

Aim: Practice the following Queries:

**ANY SYNTAX:**

SELECT *column_name(s)* FROM *table_name* WHERE *column_name operator* ANY (SELECT *column_name* FROM *table_name* WHERE *condition*);

**ALL SYNTAX WITH SELECT:**

SELECT *column_name(s)* FROM *table_name* WHERE *column_name operator* ALL (SELECT *column_name* FROM *table_name* WHERE *condition*);

**IN SYNTAX**

SELECT column_name(s) FROM table_name WHERE column_name IN (value1, value2, ...);

**EXISTS SYNTAX:**

SELECT column_names FROM table_name WHERE EXISTS (SELECT column_names FROM table_name WHERE condition);

**NOT EXISTS SYNTAX:**

SELECT col1, col2, ... FROM tablename WHERE NOT EXISTS (SELECT col1 FROM tablename WHERE condition);

**UNION SYNTAX:**

SELECT *column_name(s)* FROM *table1* UNION SELECT *column_name(s)* FROM *table2*;

**INTERSECT SYNTAX:**

SELECT *column_name(s)* FROM *table1* INTERSECT SELECT *column_name(s)* FROM *table2*;

**P.K**

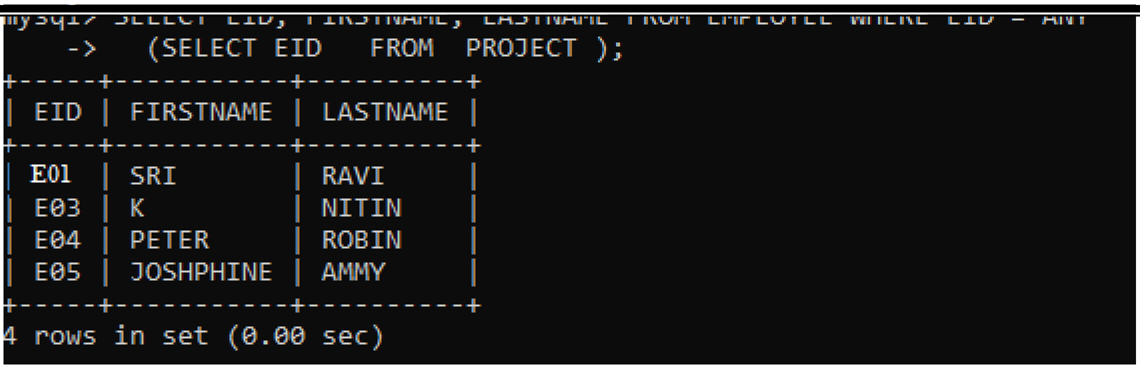
| **EID** | **FIRSTNAME** | **LASTNAME** | **JOB** | **SALARY** | **ADDRESS** |
|---|---|---|---|---|---|
| E01 | SRI | RAVI | MANAGER | 55000 | CHANDIGARH |
| E02 | ABHI | VARUN | ADMIN | 20000 | DELHI |
| E03 | K | NITIN | ASSOCIATE | 28000 | PUNE |
| E04 | PETER | ROBIN | ASSOCIATE | 28000 | BANGALORE |
| E05 | JOSEPHINE | AMMY | DEVELOPER | 38000 | HYDERABAD |

**F.K** **P.K**

| **EID** | **PID** | **PNAME** | **LOCATION** |
|---|---|---|---|
| E01 | P1 | IOT | BANGALORE |
| E03 | P3 | BIG DATA | DELHI |
| E04 | P4 | RETAIL | MUMBAI |
| E05 | P2 | ANDROID | HYDERABAD |

1. Find the detail of the employees who is working on at least one project.

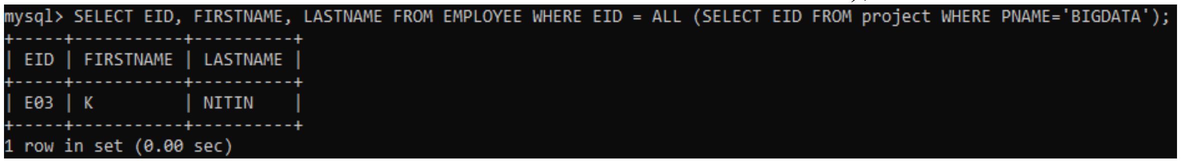
   SELECT EID, FIRSTNAME, LASTNAME FROM EMPLOYEE WHERE EID = ANY (SELECT EID FROM PROJECT);



```
mysql> SELECT EID, FIRSTNAME, LASTNAME FROM EMPLOYEE WHERE EID = ANY
    ->   (SELECT EID   FROM  PROJECT );
+-----+-----------+----------+
| EID | FIRSTNAME | LASTNAME |
+-----+-----------+----------+
| E01 | SRI       | RAVI     |
| E03 | K         | NITIN    |
| E04 | PETER     | ROBIN    |
| E05 | JOSHPHINE | AMMY     |
+-----+-----------+----------+
4 rows in set (0.00 sec)
```

2. Find the detail of the employees who is working on project IOT.

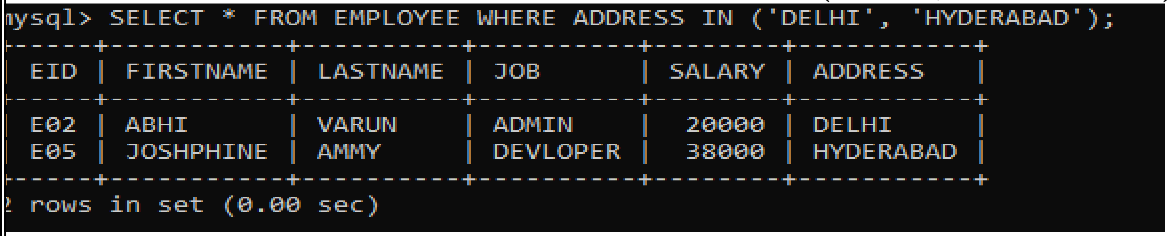
SELECT EID, FIRSTNAME, LASTNAME FROM EMPLOYEE WHERE EID = ALL (SELECT EID  FROM PROJECT WHERE PNAME = ‘BIGDATA’);



```
mysql> SELECT EID, FIRSTNAME, LASTNAME FROM EMPLOYEE WHERE EID = ALL (SELECT EID FROM project WHERE PNAME='BIGDATA');
+-----+-----------+----------+
| EID | FIRSTNAME | LASTNAME |
+-----+-----------+----------+
| E03 | K         | NITIN    |
+-----+-----------+----------+
1 row in set (0.00 sec)
```

3. Find the details of the employees belonging to Delhi or Hyderabad.

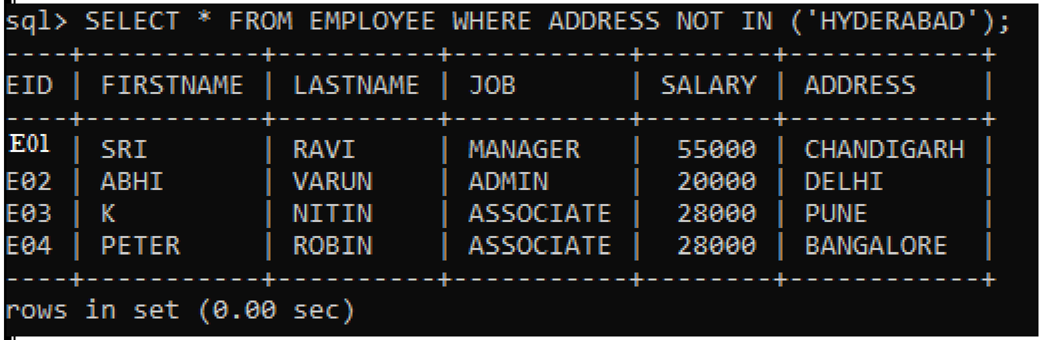
SELECT * FROM EMPLOYEE WHERE ADDRESS IN ('DELHI', 'HYDERABAD');



```
mysql> SELECT * FROM EMPLOYEE WHERE ADDRESS IN ('DELHI', 'HYDERABAD');
-----+-----------+----------+----------+--------+-----------+
 EID | FIRSTNAME | LASTNAME | JOB      | SALARY | ADDRESS   |
-----+-----------+----------+----------+--------+-----------+
 E02 | ABHI      | VARUN    | ADMIN    |  20000 | DELHI     |
 E05 | JOSHPHINE | AMMY     | DEVLOPER |  38000 | HYDERABAD |
-----+-----------+----------+----------+--------+-----------+
2 rows in set (0.00 sec)
```

4. Find the details of all Employees apart from location Hyderabad.

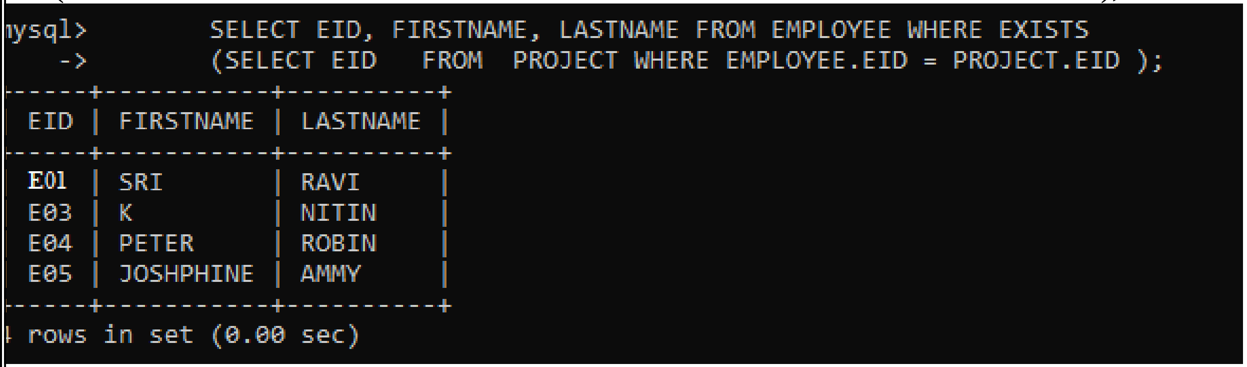
SELECT * FROM EMPLOYEE WHERE ADDRESS NOT IN ('HYDERABAD');



```
sql> SELECT * FROM EMPLOYEE WHERE ADDRESS NOT IN ('HYDERABAD');
----+-----------+----------+-----------+--------+------------+
EID | FIRSTNAME | LASTNAME | JOB       | SALARY | ADDRESS    |
----+-----------+----------+-----------+--------+------------+
E01 | SRI       | RAVI     | MANAGER   |  55000 | CHANDIGARH |
E02 | ABHI      | VARUN    | ADMIN     |  20000 | DELHI      |
E03 | K         | NITIN    | ASSOCIATE |  28000 | PUNE       |
E04 | PETER     | ROBIN    | ASSOCIATE |  28000 | BANGALORE  |
----+-----------+----------+-----------+--------+------------+
rows in set (0.00 sec)
```

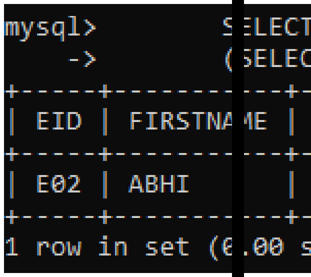
5. Find the detail of the employees who is working on at least one project.

SELECT EID, FIRSTNAME, LASTNAME FROM EMPLOYEE WHERE EXISTS (SELECT EID  FROM PROJECT WHERE EMPLOYEE.EID = PROJECT.EID );



```
mysql>       SELECT EID, FIRSTNAME, LASTNAME FROM EMPLOYEE WHERE EXISTS
    ->       (SELECT EID   FROM  PROJECT WHERE EMPLOYEE.EID = PROJECT.EID );
-----+-----------+----------+
 EID | FIRSTNAME | LASTNAME |
-----+-----------+----------+
 E01 | SRI       | RAVI     |
 E03 | K         | NITIN    |
 E04 | PETER     | ROBIN    |
 E05 | JOSHPHINE | AMMY     |
-----+-----------+----------+
4 rows in set (0.00 sec)
```

6. Find the detail of the employees who is not working on any project.
SELECT EID, FIRSTNAME, LASTNAME FROM EMPLOYEE WHERE NOT EXISTS
(SELECT EID  FROM  PROJECT WHERE EMPLOYEE.EID = PROJECT.EID );



```
mysql>        SELECT
    ->        (SELEC
+-----+-----------+-
| EID | FIRSTNAME |
+-----+-----------+-
| E02 | ABHI      |
+-----+-----------+-
1 row in set (0.00 s
```

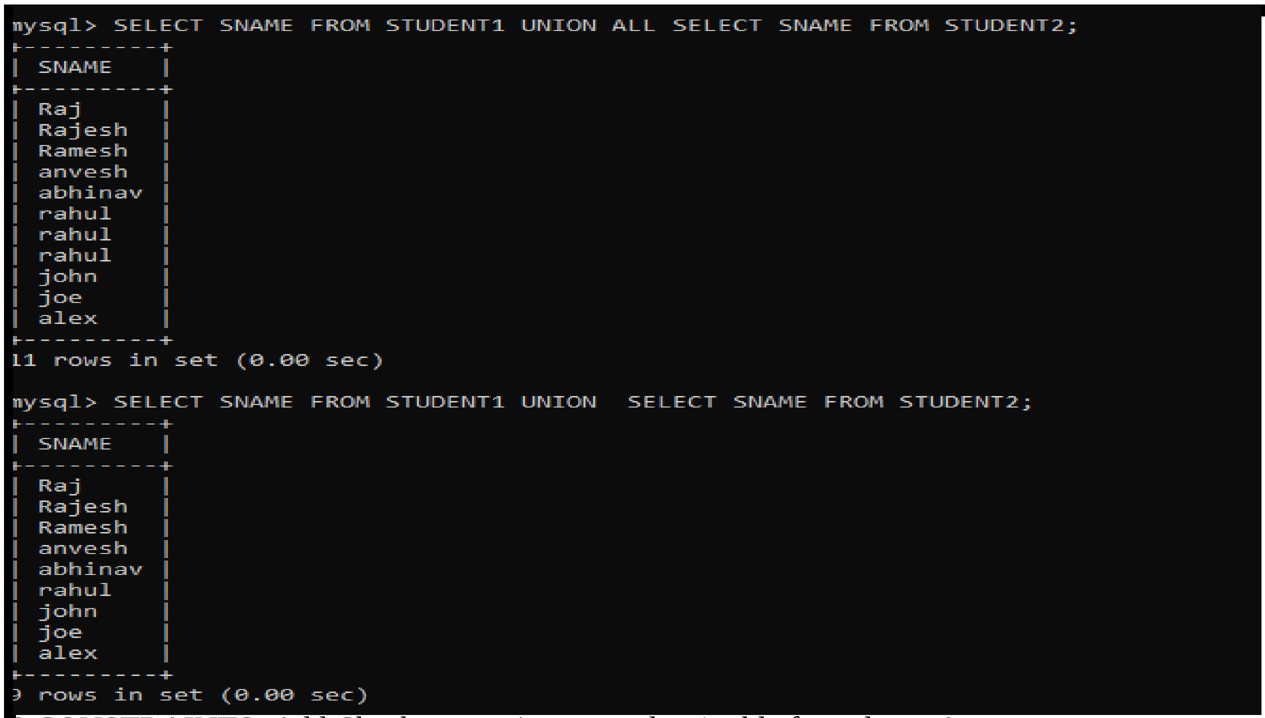
| **SID** | **SNAME** | **AGE** | **ADDRESS** |
|---|---|---|---|
| 100 | Raj | 19 | Hyd |
| 101 | Rajesh | 19 | Hyd |
| 102 | Ramesh | 20 | Hyd |
| 103 | Anvesh | 20 | Hyd |
| 104 | Abhinav | 20 | tn |
| 104 | Rahul | 20 | Bangalore |
| 105 | Rahul | 20 | Bangalore |

| **SID** | **SNAME** | **AGE** | **ADDRESS** |
|---|---|---|---|
| 201 | Rahul | 18 | Bangalore |
| 202 | John | 19 | Bangalore |
| 203 | Joe | 20 | Hyd |
| 204 | Alex | 20 | Che |

TABLE: STUDENT1

TABLE: STUDENT2

1.UNION:
SELECT SNAME FROM STUDENT1 UNION SELECT SNAME FROM STUDENT2;
SELECT SNAME FROM STUDENT1 UNION ALL SELECT SNAME FROM STUDENT2;
By giving union we won’t get any duplicate values in result. while with union all it gives duplicate values also in results.

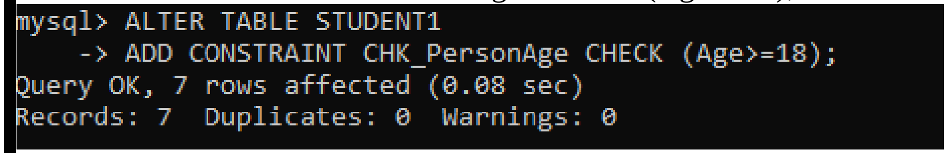


```
mysql> SELECT SNAME FROM STUDENT1 UNION ALL SELECT SNAME FROM STUDENT2;
+---------+
| SNAME   |
+---------+
| Raj     |
| Rajesh  |
| Ramesh  |
| anvesh  |
| abhinav |
| rahul   |
| rahul   |
| rahul   |
| john    |
| joe     |
| alex    |
+---------+
11 rows in set (0.00 sec)

mysql> SELECT SNAME FROM STUDENT1 UNION  SELECT SNAME FROM STUDENT2;
+---------+
| SNAME   |
+---------+
| Raj     |
| Rajesh  |
| Ramesh  |
| anvesh  |
| abhinav |
| rahul   |
| john    |
| joe     |
| alex    |
+---------+
9 rows in set (0.00 sec)
```

2.CONSTRAINTS: Add Check constraints to student1 table for column Age.

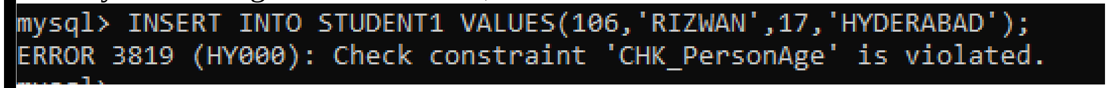
CHECK CONSTRAINT:

ALTER TABLE STUDENT1
ADD CONSTRAINT CHK_PersonAge CHECK (Age>=18);



```
mysql> ALTER TABLE STUDENT1
    -> ADD CONSTRAINT CHK_PersonAge CHECK (Age>=18);
Query OK, 7 rows affected (0.08 sec)
Records: 7  Duplicates: 0  Warnings: 0
```

Now try to insert age less than 18 i.e.,17 in student1 table. It won't allow user to enter age of student less than 18



```
mysql> INSERT INTO STUDENT1 VALUES(106,'RIZWAN',17,'HYDERABAD');
ERROR 3819 (HY000): Check constraint 'CHK_PersonAge' is violated.
```

**EXPERIMENT – 7**

**Queries using Aggregate functions, GROUP BY, HAVING and Creation and dropping of Views.**

Aim: To Practice Queries using Aggregate functions for the following

MySQL supports the following aggregate functions:

| Function | Description |
|---|---|
| AVG() | Returns the average of the values in the selected column |
| COUNT() | Returns the number of rows returned for a selection |
| MAX() | Returns the maximum value for a column |
| MIN() | Returns the minimum value of a column |
| SUM() | Returns the sum of the values in a specified column |

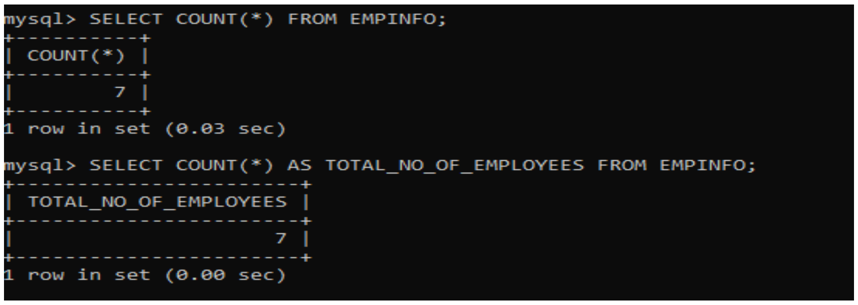
TABLE: **EMPINFO**

| EID | EMPLOYEENAME | JOB | MGR ID | HIREDATE | SALARY | COMMISSION | DEPTNO |
|---|---|---|---|---|---|---|---|
| **1001** | **ANIL** | **MANAGER** | **NULL** | **03/03/2010** | **35000** | **NULL** | **10** |
| 1002 | AKHIL | CLERK | 1001 | 02/04/2015 | 25000 | NULL | 10 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1003 | VINOD | SALES | 1001 | 05/06/2016 | 18000 | 1800 | 10 |
| 1004 | VIKAS | SALES | 1001 | 06/07/2016 | 16000 | 1600 | 10 |
| **1005** | **SUNIL** | **MANAGER** | **NULL** | **03/04/2011** | **30000** | **NULL** | **20** |
| 1006 | KIRAN | CLERK | 1005 | 05/06/2016 | 20000 | NULL | 20 |
| 1007 | AREEB | SALES | 1005 | 10/05/2016 | 15000 | 1500 | 20 |

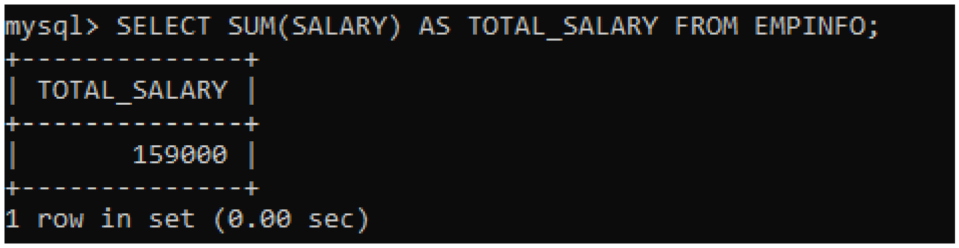
**AGGREGRATE FUNCTIONS:**

1. Write a query to get total number of employees working in organization.



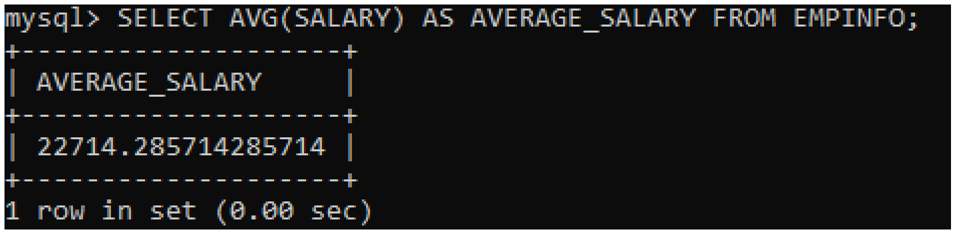
```
mysql> SELECT COUNT(*) FROM EMPINFO;
+----------+
| COUNT(*) |
+----------+
|        7 |
+----------+
1 row in set (0.03 sec)

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

2. Write a query to get total of salary paid to all employees.



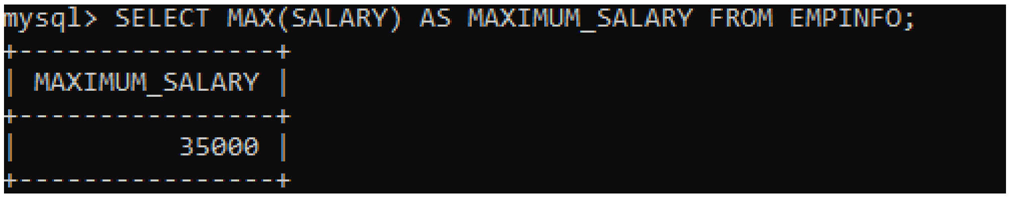
```
mysql> SELECT SUM(SALARY) AS TOTAL_SALARY FROM EMPINFO;
+--------------+
| TOTAL_SALARY |
+--------------+
|       159000 |
+--------------+
1 row in set (0.00 sec)
```

3. Write a query to get the average of the salary paid.



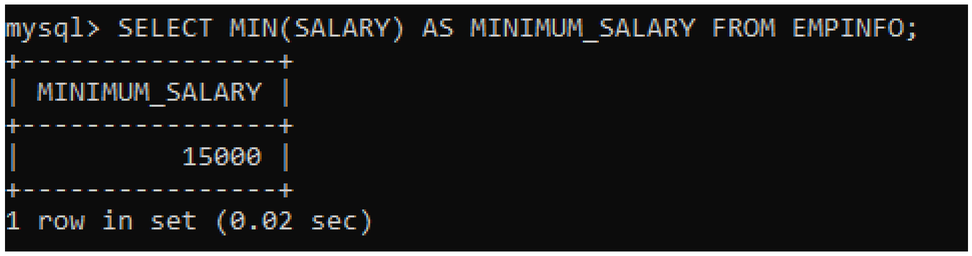
```
mysql> SELECT AVG(SALARY) AS AVERAGE_SALARY FROM EMPINFO;
+--------------------+
| AVERAGE_SALARY     |
+--------------------+
| 22714.285714285714 |
+--------------------+
1 row in set (0.00 sec)
```

4. Write a query to get details of employee whose salary is maximum amongst all employees.



```
mysql> SELECT MAX(SALARY) AS MAXIMUM_SALARY FROM EMPINFO;
+----------------+
| MAXIMUM_SALARY |
+----------------+
|          35000 |
+----------------+
```

5. Write a query to get details of employee whose salary is minimum amongst all employees.

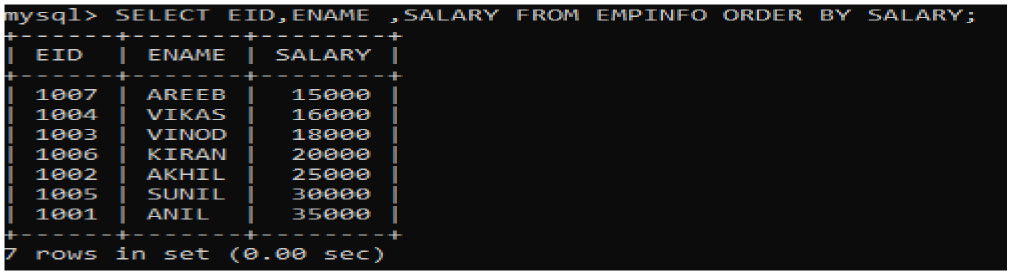


```
mysql> SELECT MIN(SALARY) AS MINIMUM_SALARY FROM EMPINFO;
+----------------+
| MINIMUM_SALARY |
+----------------+
|          15000 |
+----------------+
1 row in set (0.02 sec)
```

**ORDERBY, GROUPBY, HAVING CLAUSES:**

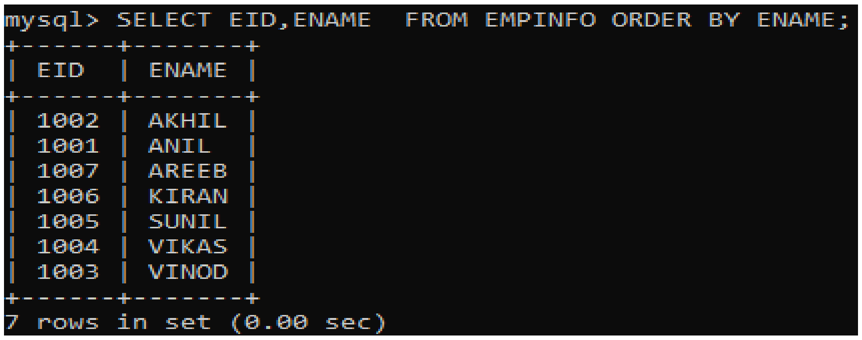
ORDERBY:

1. WAQ to get salary in ascending order:(Min to Max).



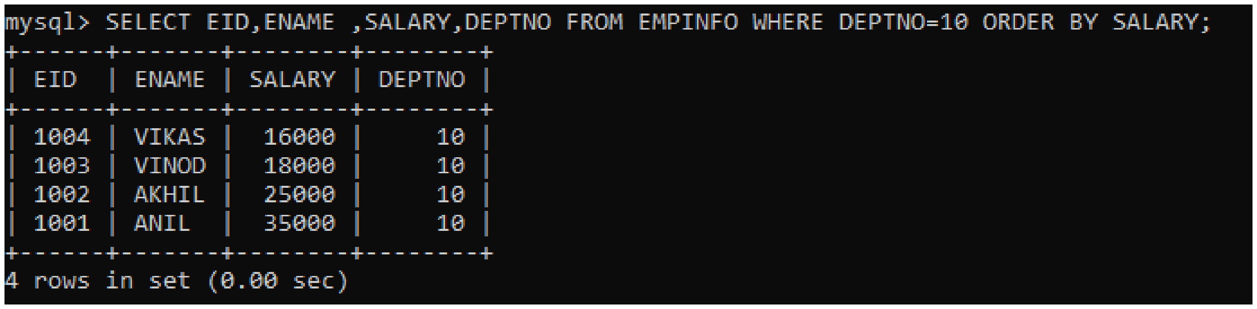
```
mysql> SELECT EID,ENAME ,SALARY FROM EMPINFO ORDER BY SALARY;
+------+-------+--------+
| EID  | ENAME | SALARY |
+------+-------+--------+
| 1007 | AREEB |  15000 |
| 1004 | VIKAS |  16000 |
| 1003 | VINOD |  18000 |
| 1006 | KIRAN |  20000 |
| 1002 | AKHIL |  25000 |
| 1005 | SUNIL |  30000 |
| 1001 | ANIL  |  35000 |
+------+-------+--------+
7 rows in set (0.00 sec)
```

2. WAQ to get employee names in Alphabetic order.



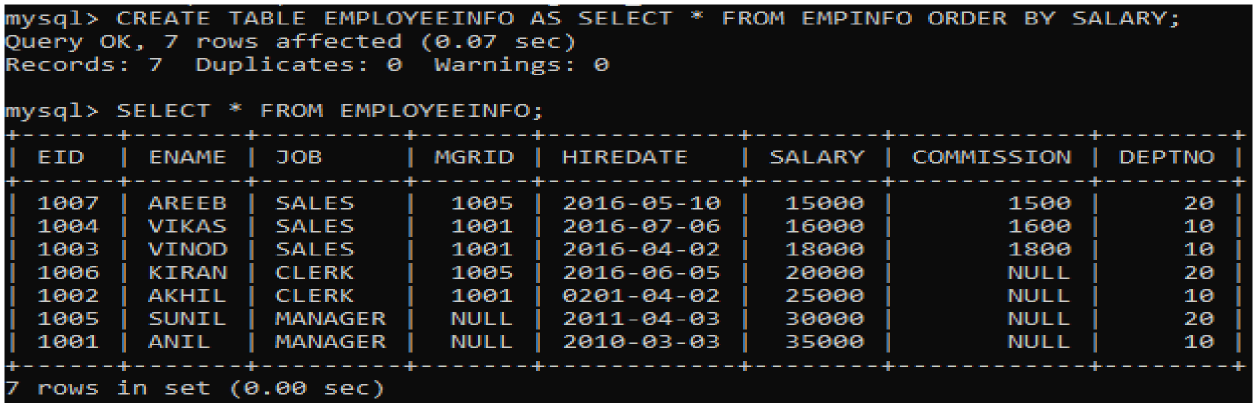
```
mysql> SELECT EID,ENAME  FROM EMPINFO ORDER BY ENAME;
+------+-------+
| EID  | ENAME |
+------+-------+
| 1002 | AKHIL |
| 1001 | ANIL  |
| 1007 | AREEB |
| 1006 | KIRAN |
| 1005 | SUNIL |
| 1004 | VIKAS |
| 1003 | VINOD |
+------+-------+
7 rows in set (0.00 sec)
```

3. WAQ to get salary in ascending order for the employees belonging to department number 10.



```
mysql> SELECT EID,ENAME ,SALARY,DEPTNO FROM EMPINFO WHERE DEPTNO=10 ORDER BY SALARY;
+------+-------+--------+--------+
| EID  | ENAME | SALARY | DEPTNO |
+------+-------+--------+--------+
| 1004 | VIKAS |  16000 |     10 |
| 1003 | VINOD |  18000 |     10 |
| 1002 | AKHIL |  25000 |     10 |
| 1001 | ANIL  |  35000 |     10 |
+------+-------+--------+--------+
4 rows in set (0.00 sec)
```

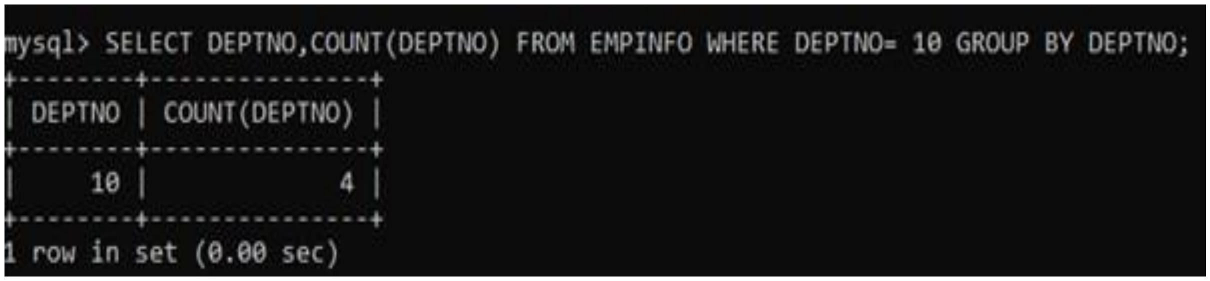
4. WAQ to create duplicate table and store the records based on salary wise.



```
mysql> CREATE TABLE EMPLOYEEINFO AS SELECT * FROM EMPINFO ORDER BY SALARY;
Query OK, 7 rows affected (0.07 sec)
Records: 7  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM EMPLOYEEINFO;
+------+-------+---------+-------+------------+--------+------------+--------+
| EID  | ENAME | JOB     | MGRID | HIREDATE   | SALARY | COMMISSION | DEPTNO |
+------+-------+---------+-------+------------+--------+------------+--------+
| 1007 | AREEB | SALES   |  1005 | 2016-05-10 |  15000 |       1500 |     20 |
| 1004 | VIKAS | SALES   |  1001 | 2016-07-06 |  16000 |       1600 |     10 |
| 1003 | VINOD | SALES   |  1001 | 2016-04-02 |  18000 |       1800 |     10 |
| 1006 | KIRAN | CLERK   |  1005 | 2016-06-05 |  20000 |       NULL |     20 |
| 1002 | AKHIL | CLERK   |  1001 | 0201-04-02 |  25000 |       NULL |     10 |
| 1005 | SUNIL | MANAGER |  NULL | 2011-04-03 |  30000 |       NULL |     20 |
| 1001 | ANIL  | MANAGER |  NULL | 2010-03-03 |  35000 |       NULL |     10 |
+------+-------+---------+-------+------------+--------+------------+--------+
7 rows in set (0.00 sec)
```

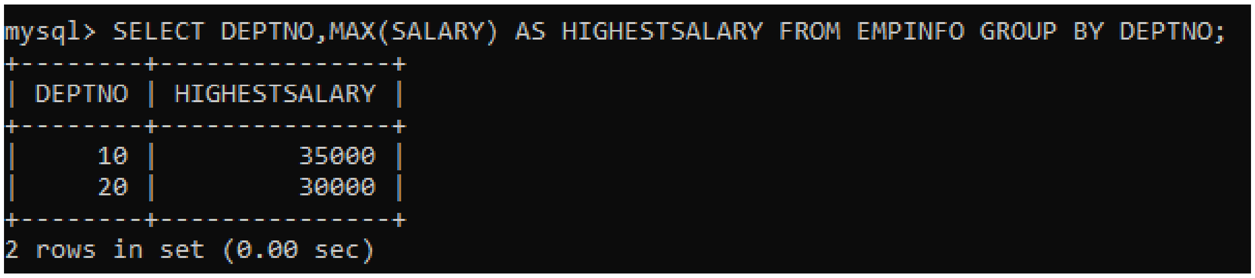
<u>GROUPBY, HAVING:</u>

1. WAQ to get count of employees from each department.



```
mysql> SELECT DEPTNO,COUNT(DEPTNO) FROM EMPINFO WHERE DEPTNO= 10 GROUP BY DEPTNO;
+--------+---------------+
| DEPTNO | COUNT(DEPTNO) |
+--------+---------------+
|     10 |             4 |
+--------+---------------+
1 row in set (0.00 sec)
```

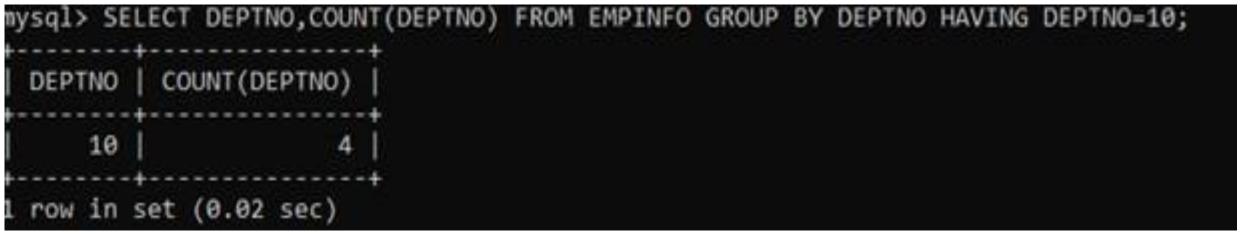
2. WAQ to get highest salary from each department.



```
mysql> SELECT DEPTNO,MAX(SALARY) AS HIGHESTSALARY FROM EMPINFO GROUP BY DEPTNO;
+--------+---------------+
| DEPTNO | HIGHESTSALARY |
+--------+---------------+
|     10 |         35000 |
|     20 |         30000 |
+--------+---------------+
2 rows in set (0.00 sec)
```

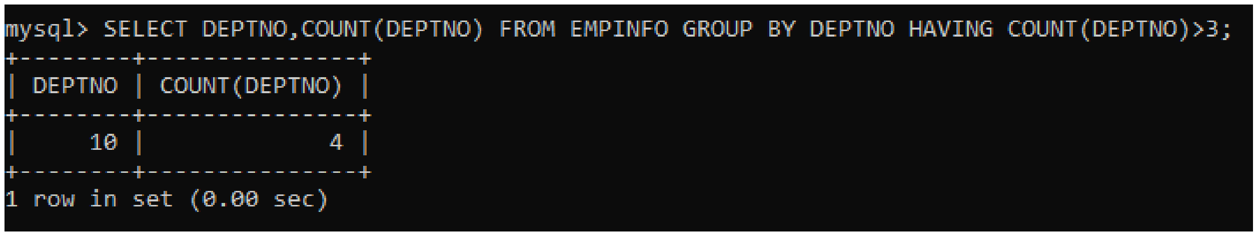
3. WAQ to get highest salary from each job.

4. WAQ to get a count of employees with department number 10.



```
mysql> SELECT DEPTNO,COUNT(DEPTNO) FROM EMPINFO GROUP BY DEPTNO HAVING DEPTNO=10;
+--------+---------------+
| DEPTNO | COUNT(DEPTNO) |
+--------+---------------+
|     10 |             4 |
+--------+---------------+
1 row in set (0.02 sec)
```

5. WAQ to get count of employees from each department only when the count is greater than 3.



```
mysql> SELECT DEPTNO,COUNT(DEPTNO) FROM EMPINFO GROUP BY DEPTNO HAVING COUNT(DEPTNO)>3;
+--------+---------------+
| DEPTNO | COUNT(DEPTNO) |
+--------+---------------+
|     10 |             4 |
+--------+---------------+
1 row in set (0.00 sec)
```

**CREATION AND DROPPING OF VIEWS:**

SYNTAX FOR CREATION:

CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;

SYNTAX FOR DELETION:

DROP VIEW view_name;

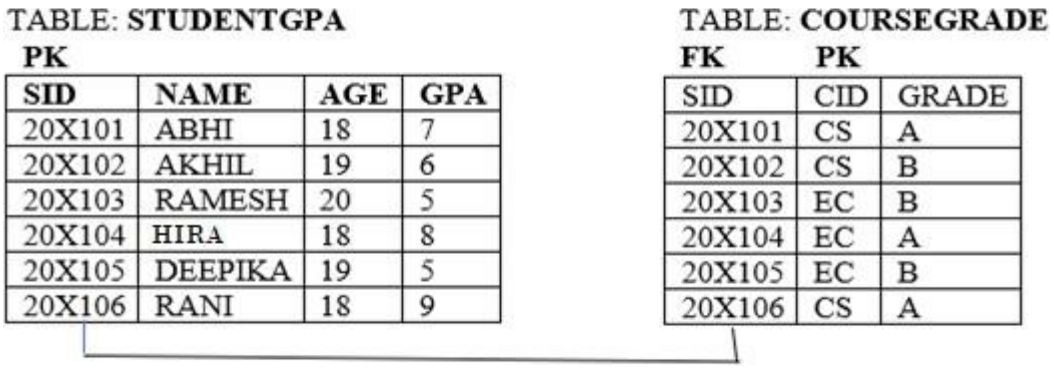


TABLE: **STUDENTGPA**

**PK**

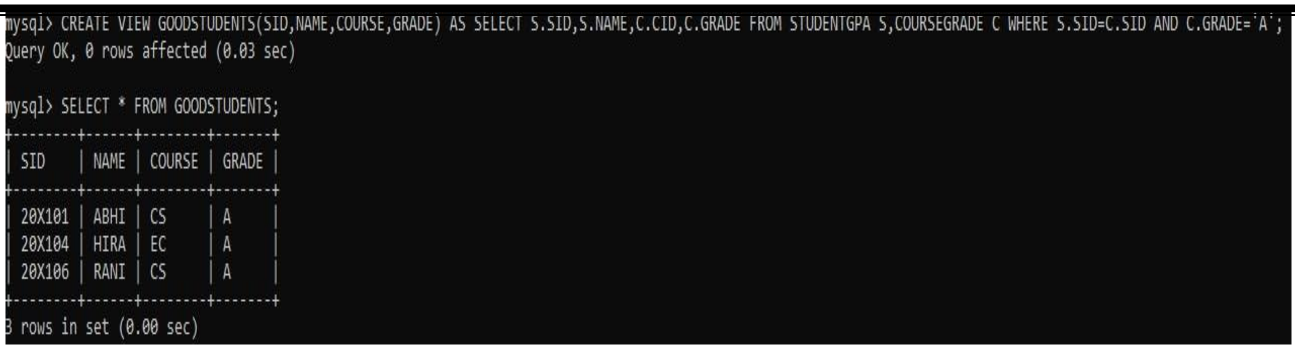
| **SID** | **NAME** | **AGE** | **GPA** |
|---|---|---|---|
| 20X101 | ABHI | 18 | 7 |
| 20X102 | AKHIL | 19 | 6 |
| 20X103 | RAMESH | 20 | 5 |
| 20X104 | HIRA | 18 | 8 |
| 20X105 | DEEPIKA | 19 | 5 |
| 20X106 | RANI | 18 | 9 |

TABLE: **COURSEGRADE**

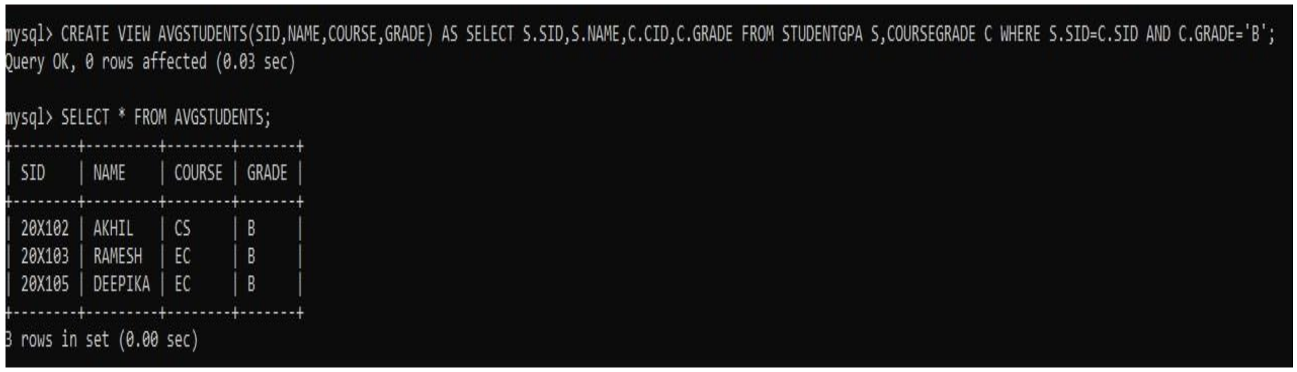
**FK** **PK**

| SID | CID | GRADE |
|---|---|---|
| 20X101 | CS | A |
| 20X102 | CS | B |
| 20X103 | EC | B |
| 20X104 | EC | A |
| 20X105 | EC | B |
| 20X106 | CS | A |

1. Create views as GOODSTUDENTS whose grade is ‘A’ and view as average AVGSTUDENTS whose grade is ‘B’.

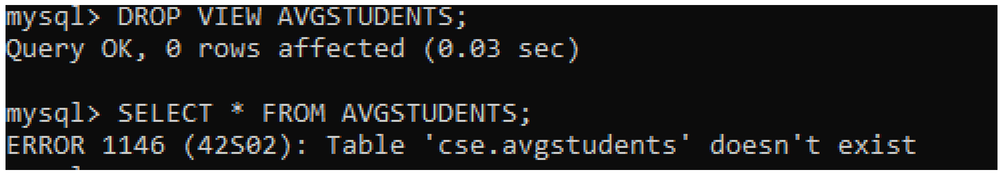


```
mysql> CREATE VIEW GOODSTUDENTS(SID,NAME,COURSE,GRADE) AS SELECT S.SID,S.NAME,C.CID,C.GRADE FROM STUDENTGPA S,COURSEGRADE C WHERE S.SID=C.SID AND C.GRADE='A';
Query OK, 0 rows affected (0.03 sec)

mysql> SELECT * FROM GOODSTUDENTS;
+--------+------+--------+-------+
| SID    | NAME | COURSE | GRADE |
+--------+------+--------+-------+
| 20X101 | ABHI | CS     | A     |
| 20X104 | HIRA | EC     | A     |
| 20X106 | RANI | CS     | A     |
+--------+------+--------+-------+
3 rows in set (0.00 sec)
```



```
mysql> CREATE VIEW AVGSTUDENTS(SID,NAME,COURSE,GRADE) AS SELECT S.SID,S.NAME,C.CID,C.GRADE FROM STUDENTGPA S,COURSEGRADE C WHERE S.SID=C.SID AND C.GRADE='B';
Query OK, 0 rows affected (0.03 sec)

mysql> SELECT * FROM AVGSTUDENTS;
+--------+---------+--------+-------+
| SID    | NAME    | COURSE | GRADE |
+--------+---------+--------+-------+
| 20X102 | AKHIL   | CS     | B     |
| 20X103 | RAMESH  | EC     | B     |
| 20X105 | DEEPIKA | EC     | B     |
+--------+---------+--------+-------+
3 rows in set (0.00 sec)
```

2. Practice drop view command on already created views



```
mysql> DROP VIEW AVGSTUDENTS;
Query OK, 0 rows affected (0.03 sec)

mysql> SELECT * FROM AVGSTUDENTS;
ERROR 1146 (42S02): Table 'cse.avgstudents' doesn't exist
```

## EXPERIMENT – 8

## TRIGGERS

Aim: Creation of insert trigger, delete trigger and update trigger.

A **trigger** is a procedure that is automatically invoked by the DBMS in response to specified changes to the database.

Six types of actions or events in the form of triggers:

- **Before Insert:** It is activated before the insertion of data into the table.
- **After Insert:** It is activated after the insertion of data into the table.
- **Before Update:** It is activated before the update of data in the table.
- **After Update:** It is activated after the update of the data in the table.
- **Before Delete:** It is activated before the data is removed from the table.

- **After Delete:** It is activated after the deletion of data from the table.

## Syntax of Creating a Trigger in MySQL:

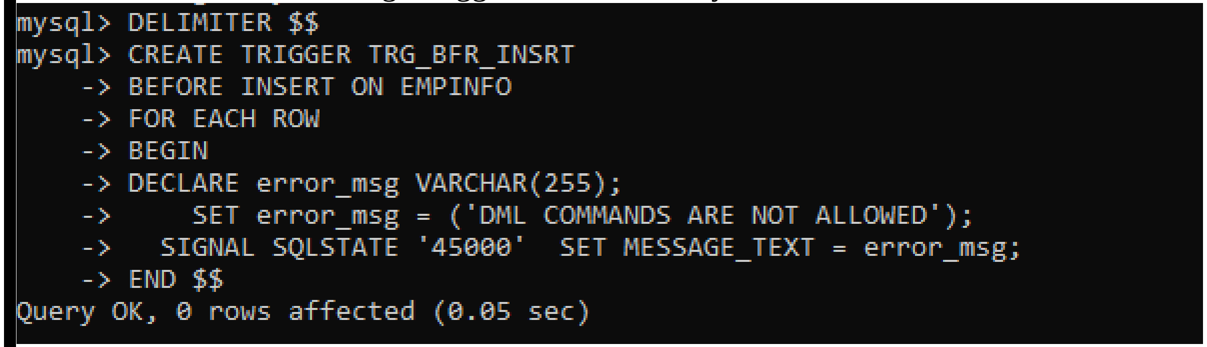
```
CREATE TRIGGER trigger_name
    (AFTER | BEFORE) (INSERT | UPDATE | DELETE)                  /*Event/
    ON table_name FOR EACH ROW
    BEGIN                                                        /*Action/
    --variable declarations
    --trigger code
    END;
```

TABLE: **EMPINFO**

**PK**

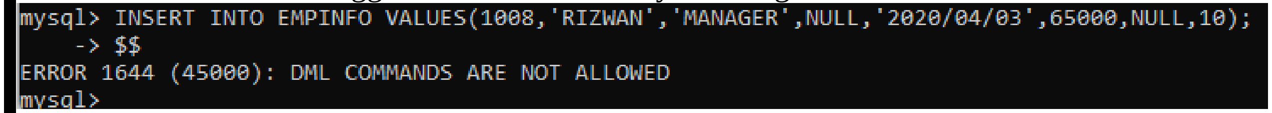
| EID | EMPLOYEENAME | JOB | MGR ID | HIREDATE | SALARY | COMMISSION | DEPTNO |
|---|---|---|---|---|---|---|---|
| **1001** | **ANIL** | **MANAGER** | **NULL** | **03/03/2010** | **35000** | **NULL** | **10** |
| 1002 | AKHIL | CLERK | 1001 | 02/04/2015 | 25000 | NULL | 10 |
| 1003 | VINOD | SALES | 1001 | 05/06/2016 | 18000 | 1800 | 10 |
| 1004 | VIKAS | SALES | 1001 | 06/07/2016 | 16000 | 1600 | 10 |
| **1005** | **SUNIL** | **MANAGER** | **NULL** | **03/04/2011** | **30000** | **NULL** | **20** |
| 1006 | KIRAN | CLERK | 1005 | 05/06/2016 | 20000 | NULL | 20 |
| 1007 | AREEB | SALES | 1005 | 10/05/2016 | 15000 | 1500 | 20 |

**BEFORE INSERT**: Creating a trigger not to allow any insertion in **EMPINFO** table:



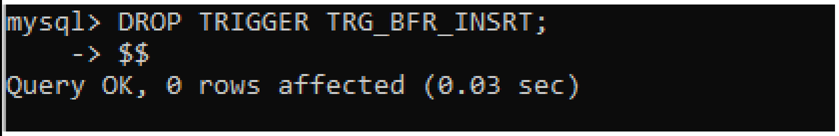
```
mysql> DELIMITER $$
mysql> CREATE TRIGGER TRG_BFR_INSRT
    -> BEFORE INSERT ON EMPINFO
    -> FOR EACH ROW
    -> BEGIN
    -> DECLARE error_msg VARCHAR(255);
    ->     SET error_msg = ('DML COMMANDS ARE NOT ALLOWED');
    ->   SIGNAL SQLSTATE '45000'  SET MESSAGE_TEXT = error_msg;
    -> END $$
Query OK, 0 rows affected (0.05 sec)
```

Now check whether the trigger is invoked or not by inserting values in EMPINFO table



```
mysql> INSERT INTO EMPINFO VALUES(1008,'RIZWAN','MANAGER',NULL,'2020/04/03',65000,NULL,10);
    -> $$
ERROR 1644 (45000): DML COMMANDS ARE NOT ALLOWED
mysql>
```

DROP THE TRIGGER TO PERFORM OTHER OPERATIONS

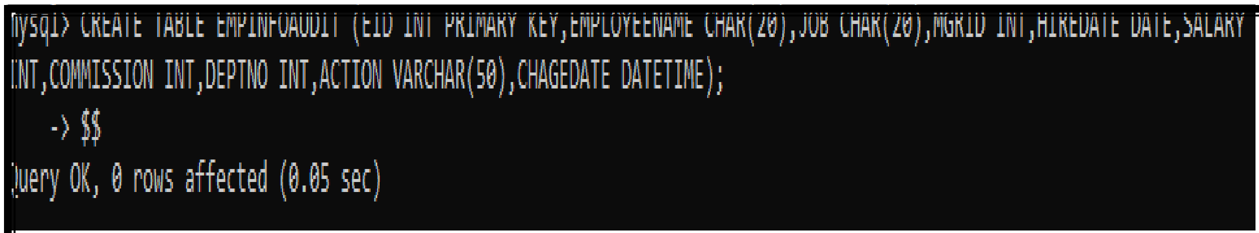


```
mysql> DROP TRIGGER TRG_BFR_INSRT;
    -> $$
Query OK, 0 rows affected (0.03 sec)
```

-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

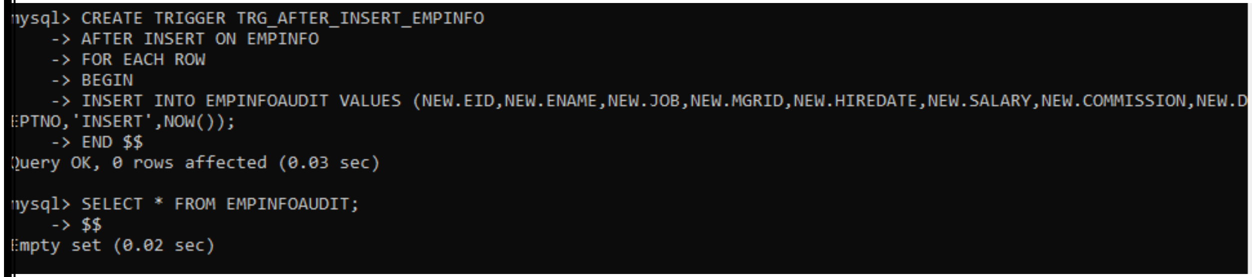
**AFTER INSERT:** creating a trigger FOR any insertion in **EMPINFO** table copies the data to another table **EMPINFOAUDIT**.

CREATE A TABLE **EMPINFOAUDIT**:



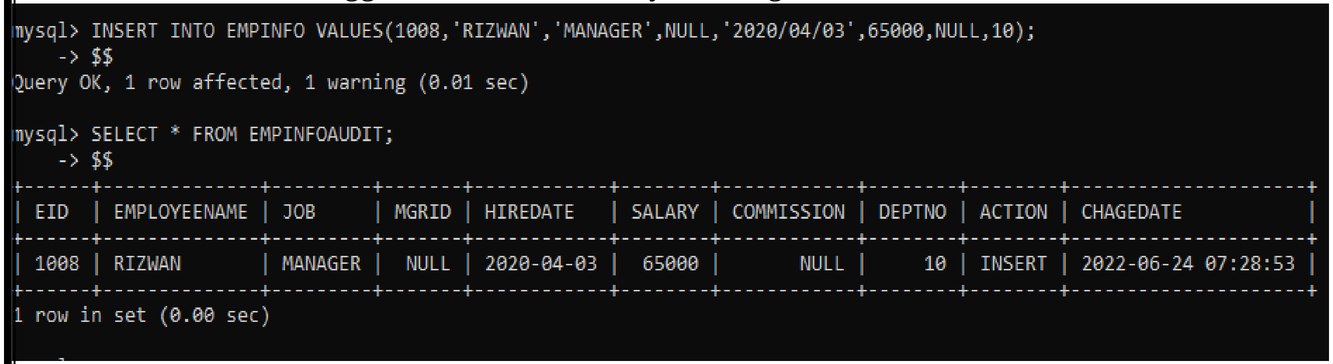
```
ysql> CREATE TABLE EMPINFOAUDIT (EID INT PRIMARY KEY,EMPLOYEENAME CHAR(20),JOB CHAR(20),MGRID INT,HIREDATE DATE,SALARY
INT,COMMISSION INT,DEPTNO INT,ACTION VARCHAR(50),CHAGEDATE DATETIME);
    -> $$
Query OK, 0 rows affected (0.05 sec)
```

NOW CREATE A TRIGGER:

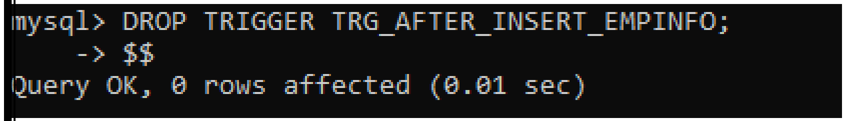


```
mysql> CREATE TRIGGER TRG_AFTER_INSERT_EMPINFO
    -> AFTER INSERT ON EMPINFO
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO EMPINFOAUDIT VALUES (NEW.EID,NEW.ENAME,NEW.JOB,NEW.MGRID,NEW.HIREDATE,NEW.SALARY,NEW.COMMISSION,NEW.D
EPTNO,'INSERT',NOW());
    -> END $$
Query OK, 0 rows affected (0.03 sec)

mysql> SELECT * FROM EMPINFOAUDIT;
    -> $$
Empty set (0.02 sec)
```

Now check whether the trigger is invoked or not by inserting values in **EMPINFO** table



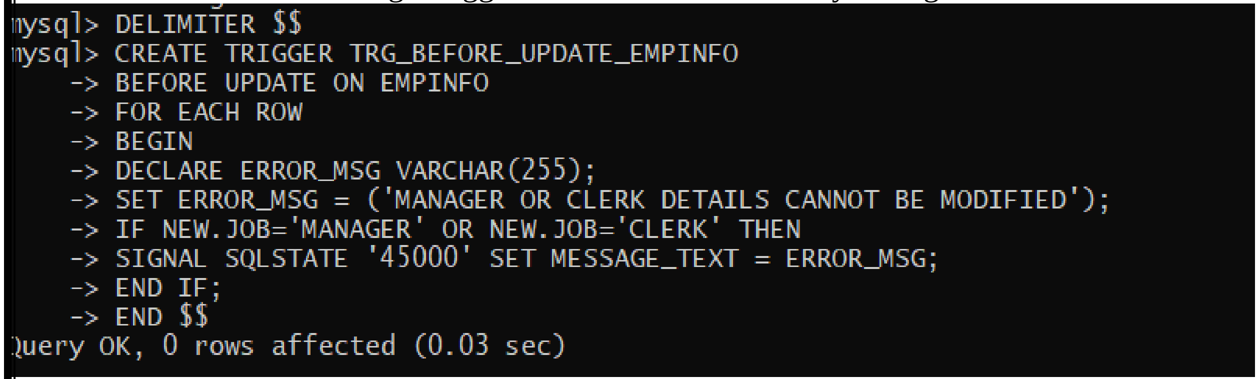
```
mysql> INSERT INTO EMPINFO VALUES(1008,'RIZWAN','MANAGER',NULL,'2020/04/03',65000,NULL,10);
    -> $$
Query OK, 1 row affected, 1 warning (0.01 sec)

mysql> SELECT * FROM EMPINFOAUDIT;
    -> $$
+------+--------------+---------+-------+------------+--------+------------+--------+--------+---------------------+
| EID  | EMPLOYEENAME | JOB     | MGRID | HIREDATE   | SALARY | COMMISSION | DEPTNO | ACTION | CHAGEDATE           |
+------+--------------+---------+-------+------------+--------+------------+--------+--------+---------------------+
| 1008 | RIZWAN       | MANAGER |  NULL | 2020-04-03 |  65000 |       NULL |     10 | INSERT | 2022-06-24 07:28:53 |
+------+--------------+---------+-------+------------+--------+------------+--------+--------+---------------------+
1 row in set (0.00 sec)
```



```
mysql> DROP TRIGGER TRG_AFTER_INSERT_EMPINFO;
    -> $$
Query OK, 0 rows affected (0.01 sec)
```

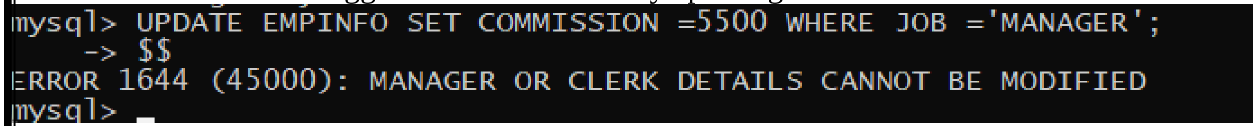
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**BEFORE UPDATE**: creating a trigger that not allows to modify manager or clerk details:

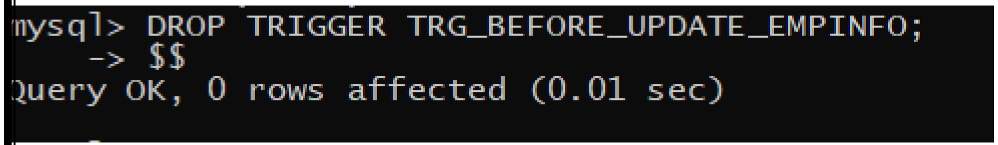


```
mysql> DELIMITER $$
mysql> CREATE TRIGGER TRG_BEFORE_UPDATE_EMPINFO
    -> BEFORE UPDATE ON EMPINFO
    -> FOR EACH ROW
    -> BEGIN
    -> DECLARE ERROR_MSG VARCHAR(255);
    -> SET ERROR_MSG = ('MANAGER OR CLERK DETAILS CANNOT BE MODIFIED');
    -> IF NEW.JOB='MANAGER' OR NEW.JOB='CLERK' THEN
    -> SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = ERROR_MSG;
    -> END IF;
    -> END $$
Query OK, 0 rows affected (0.03 sec)
```

Now check whether the trigger is invoked or not by updating values in **EMPINFO** table



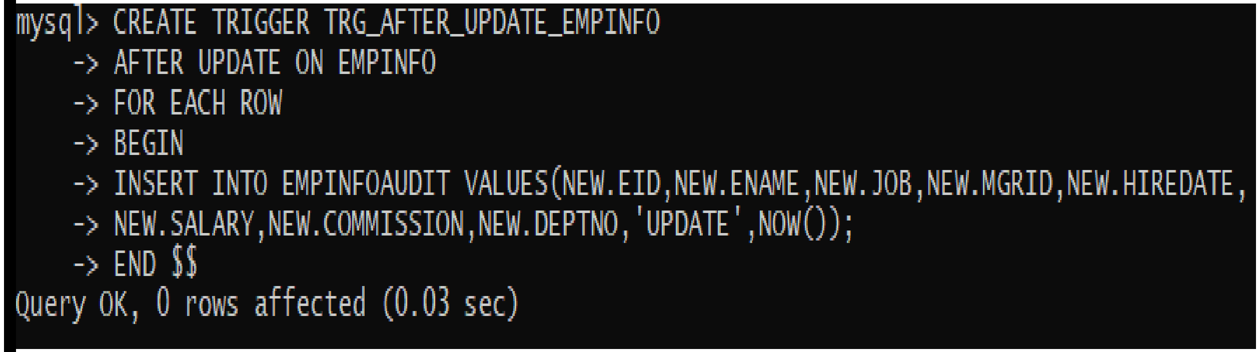
```
mysql> UPDATE EMPINFO SET COMMISSION =5500 WHERE JOB ='MANAGER';
    -> $$
ERROR 1644 (45000): MANAGER OR CLERK DETAILS CANNOT BE MODIFIED
mysql> _
```



```
mysql> DROP TRIGGER TRG_BEFORE_UPDATE_EMPINFO;
    -> $$
Query OK, 0 rows affected (0.01 sec)
```

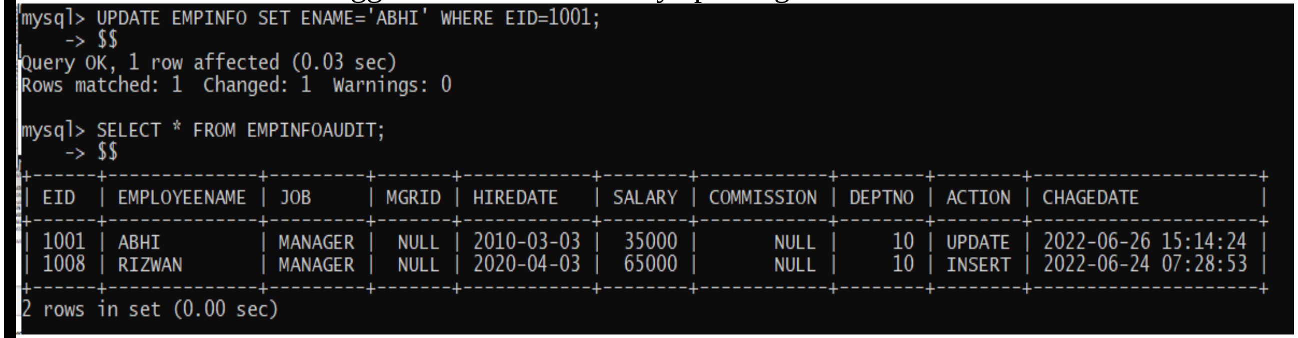
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**AFTER UPDATE:** creating a trigger FOR any updating in **EMPINFO** table copies the data to another table **EMPINFOAUDIT**.

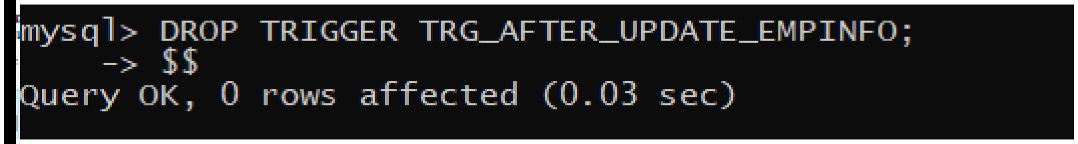


```
mysql> CREATE TRIGGER TRG_AFTER_UPDATE_EMPINFO
    -> AFTER UPDATE ON EMPINFO
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO EMPINFOAUDIT VALUES(NEW.EID,NEW.ENAME,NEW.JOB,NEW.MGRID,NEW.HIREDATE,
    -> NEW.SALARY,NEW.COMMISSION,NEW.DEPTNO,'UPDATE',NOW());
    -> END $$
Query OK, 0 rows affected (0.03 sec)
```

Now check whether the trigger is invoked or not by updating values in **EMPINFO** table



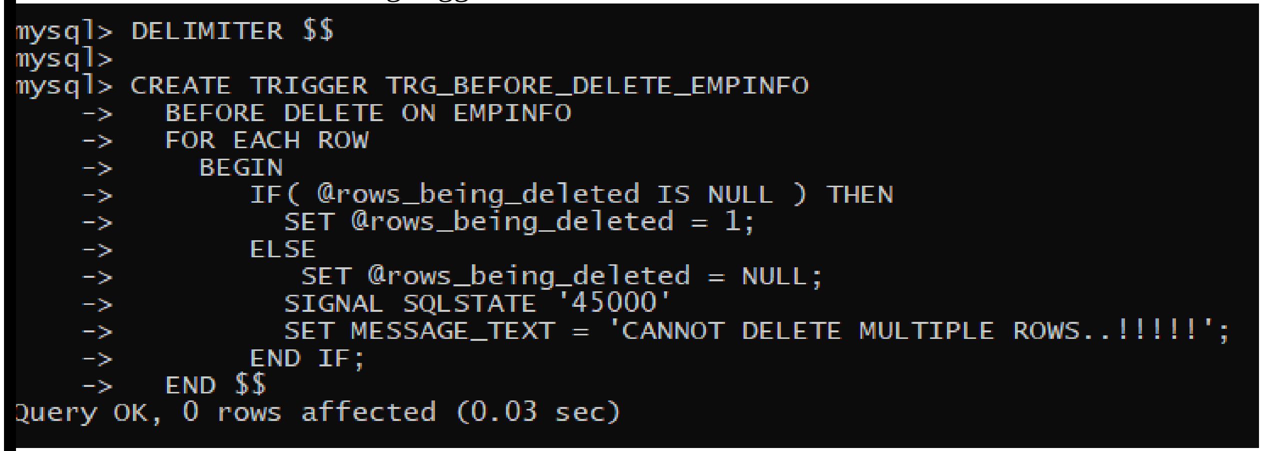
```
mysql> UPDATE EMPINFO SET ENAME='ABHI' WHERE EID=1001;
    -> $$
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM EMPINFOAUDIT;
    -> $$
+------+--------------+---------+-------+------------+--------+------------+--------+--------+---------------------+
| EID  | EMPLOYEENAME | JOB     | MGRID | HIREDATE   | SALARY | COMMISSION | DEPTNO | ACTION | CHAGEDATE           |
+------+--------------+---------+-------+------------+--------+------------+--------+--------+---------------------+
| 1001 | ABHI         | MANAGER |  NULL | 2010-03-03 |  35000 |       NULL |     10 | UPDATE | 2022-06-26 15:14:24 |
| 1008 | RIZWAN       | MANAGER |  NULL | 2020-04-03 |  65000 |       NULL |     10 | INSERT | 2022-06-24 07:28:53 |
+------+--------------+---------+-------+------------+--------+------------+--------+--------+---------------------+
2 rows in set (0.00 sec)
```



```
mysql> DROP TRIGGER TRG_AFTER_UPDATE_EMPINFO;
    -> $$
Query OK, 0 rows affected (0.03 sec)
```

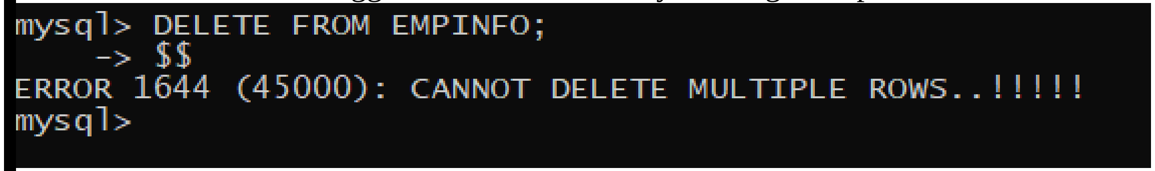
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**BEFORE DELETE**: creating trigger which not allows to delete more than one row from table.



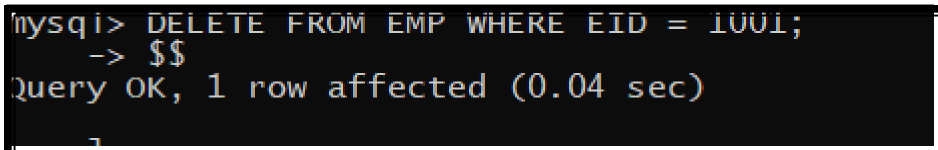
```
mysql> DELIMITER $$
mysql>
mysql> CREATE TRIGGER TRG_BEFORE_DELETE_EMPINFO
    ->   BEFORE DELETE ON EMPINFO
    ->   FOR EACH ROW
    ->     BEGIN
    ->        IF( @rows_being_deleted IS NULL ) THEN
    ->          SET @rows_being_deleted = 1;
    ->        ELSE
    ->           SET @rows_being_deleted = NULL;
    ->          SIGNAL SQLSTATE '45000'
    ->          SET MESSAGE_TEXT = 'CANNOT DELETE MULTIPLE ROWS..!!!!!';
    ->        END IF;
    ->   END $$
Query OK, 0 rows affected (0.03 sec)
```

Now check whether the trigger is invoked or not by deleting multiple rows in **EMPINFO** table



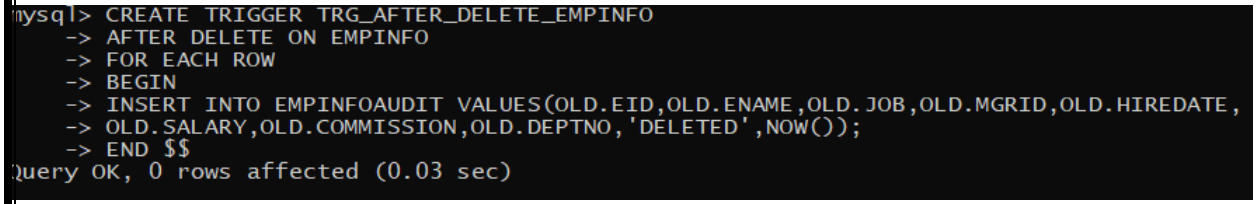
```
mysql> DELETE FROM EMPINFO;
    -> $$
ERROR 1644 (45000): CANNOT DELETE MULTIPLE ROWS..!!!!!
mysql>
```

Now check by deleting single row.it should delete a single row as we written trigger for multiple rows i.e., more than 1 row.



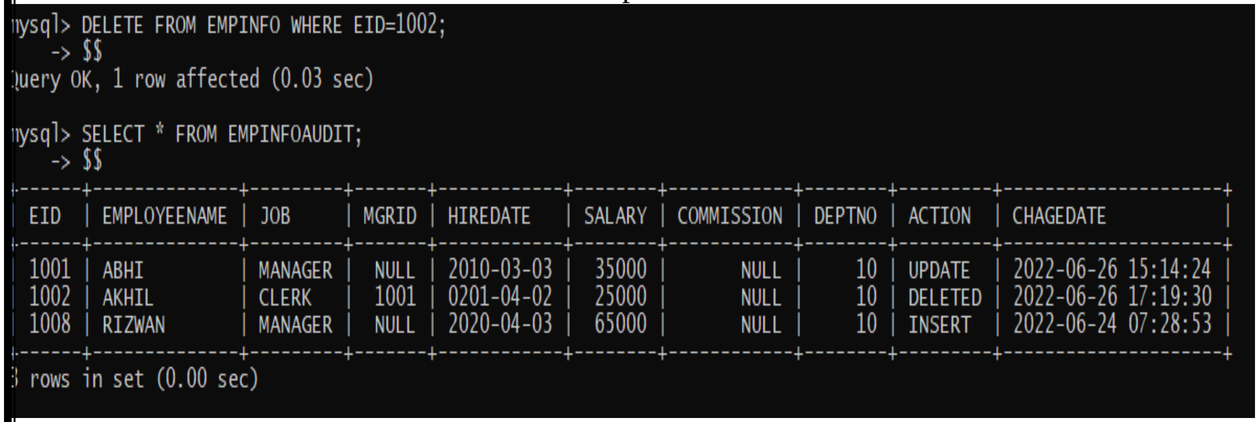
```
mysql> DELETE FROM EMP WHERE EID = 1001;
    -> $$
Query OK, 1 row affected (0.04 sec)
```

**AFTER DELETE:** creating a trigger for deleting any row in **EMPINFO** table copies the data to another table **EMPINFOAUDIT**.



```
nysql> CREATE TRIGGER TRG_AFTER_DELETE_EMPINFO
    -> AFTER DELETE ON EMPINFO
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO EMPINFOAUDIT VALUES(OLD.EID,OLD.ENAME,OLD.JOB,OLD.MGRID,OLD.HIREDATE,
    -> OLD.SALARY,OLD.COMMISSION,OLD.DEPTNO,'DELETED',NOW());
    -> END $$
Query OK, 0 rows affected (0.03 sec)
```

Now check whether the trigger is invoked or not by deleting rows in **EMPINFO** table and also check the **EMPINFOAUDIT** table to check whether it copied the deleted row.



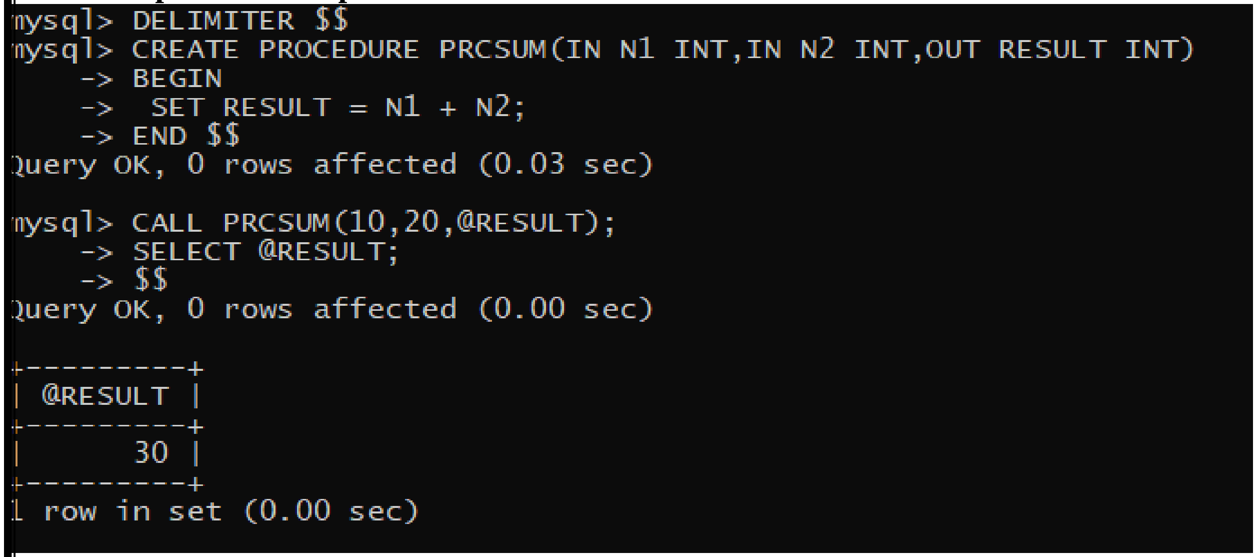
```
nysql> DELETE FROM EMPINFO WHERE EID=1002;
    -> $$
Query OK, 1 row affected (0.03 sec)

nysql> SELECT * FROM EMPINFOAUDIT;
    -> $$
+------+-------------+---------+-------+------------+--------+------------+--------+---------+---------------------+
| EID  | EMPLOYEENAME | JOB     | MGRID | HIREDATE   | SALARY | COMMISSION | DEPTNO | ACTION  | CHAGEDATE           |
+------+-------------+---------+-------+------------+--------+------------+--------+---------+---------------------+
| 1001 | ABHI        | MANAGER |  NULL | 2010-03-03 |  35000 |       NULL |     10 | UPDATE  | 2022-06-26 15:14:24 |
| 1002 | AKHIL       | CLERK   |  1001 | 0201-04-02 |  25000 |       NULL |     10 | DELETED | 2022-06-26 17:19:30 |
| 1008 | RIZWAN      | MANAGER |  NULL | 2020-04-03 |  65000 |       NULL |     10 | INSERT  | 2022-06-24 07:28:53 |
+------+-------------+---------+-------+------------+--------+------------+--------+---------+---------------------+
3 rows in set (0.00 sec)
```

.

## EXPERIMENT – 9
## PROCEDURES

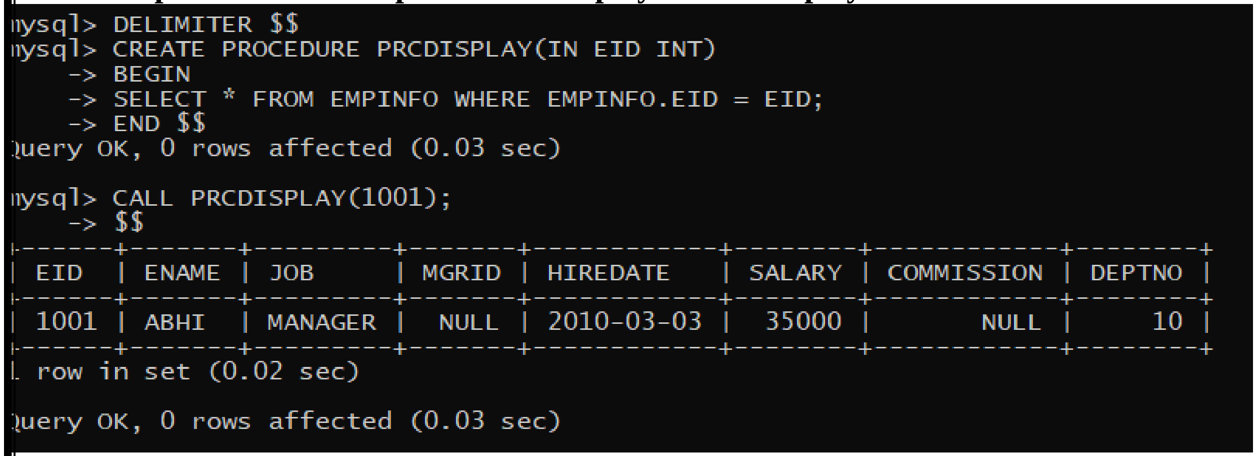
**Aim: Creation of stored Procedures and Execution of Procedures and Modification of Procedures.**

**1.Create a procedure to print sum of two values.**



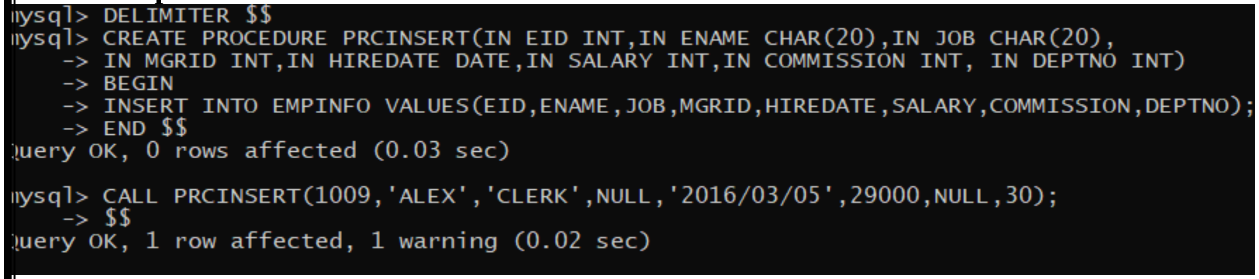
```
ysql> DELIMITER $$
ysql> CREATE PROCEDURE PRCSUM(IN N1 INT,IN N2 INT,OUT RESULT INT)
    -> BEGIN
    ->  SET RESULT = N1 + N2;
    -> END $$
uery OK, 0 rows affected (0.03 sec)

ysql> CALL PRCSUM(10,20,@RESULT);
    -> SELECT @RESULT;
    -> $$
uery OK, 0 rows affected (0.00 sec)

+---------+
| @RESULT |
+---------+
|      30 |
+---------+
1 row in set (0.00 sec)
```

**2.Create a procedure to accept EID and display info of Employee from EMPINFO table.**



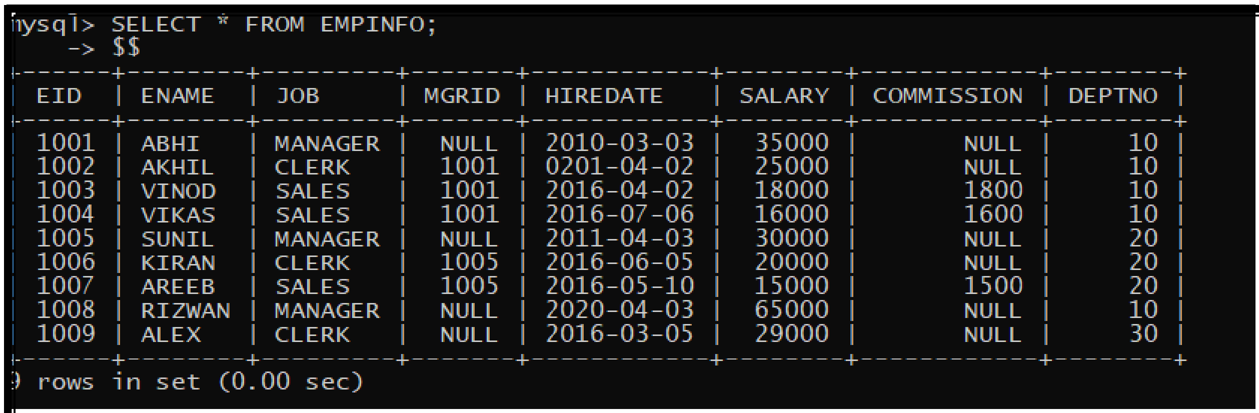
```
ysql> DELIMITER $$
ysql> CREATE PROCEDURE PRCDISPLAY(IN EID INT)
    -> BEGIN
    -> SELECT * FROM EMPINFO WHERE EMPINFO.EID = EID;
    -> END $$
uery OK, 0 rows affected (0.03 sec)

ysql> CALL PRCDISPLAY(1001);
    -> $$
+------+-------+---------+-------+------------+--------+------------+--------+
| EID  | ENAME | JOB     | MGRID | HIREDATE   | SALARY | COMMISSION | DEPTNO |
+------+-------+---------+-------+------------+--------+------------+--------+
| 1001 | ABHI  | MANAGER |  NULL | 2010-03-03 |  35000 |       NULL |     10 |
+------+-------+---------+-------+------------+--------+------------+--------+
1 row in set (0.02 sec)

uery OK, 0 rows affected (0.03 sec)
```

**3.Create a procedure to add records in EMPINFO table.**



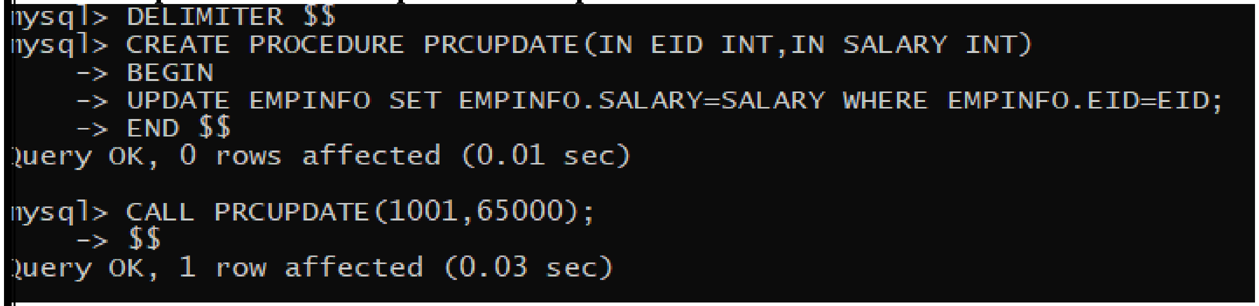
```
ysql> DELIMITER $$
ysql> CREATE PROCEDURE PRCINSERT(IN EID INT,IN ENAME CHAR(20),IN JOB CHAR(20),
    -> IN MGRID INT,IN HIREDATE DATE,IN SALARY INT,IN COMMISSION INT, IN DEPTNO INT)
    -> BEGIN
    -> INSERT INTO EMPINFO VALUES(EID,ENAME,JOB,MGRID,HIREDATE,SALARY,COMMISSION,DEPTNO);
    -> END $$
uery OK, 0 rows affected (0.03 sec)

ysql> CALL PRCINSERT(1009,'ALEX','CLERK',NULL,'2016/03/05',29000,NULL,30);
    -> $$
uery OK, 1 row affected, 1 warning (0.02 sec)
```

CHECK THE TABLE



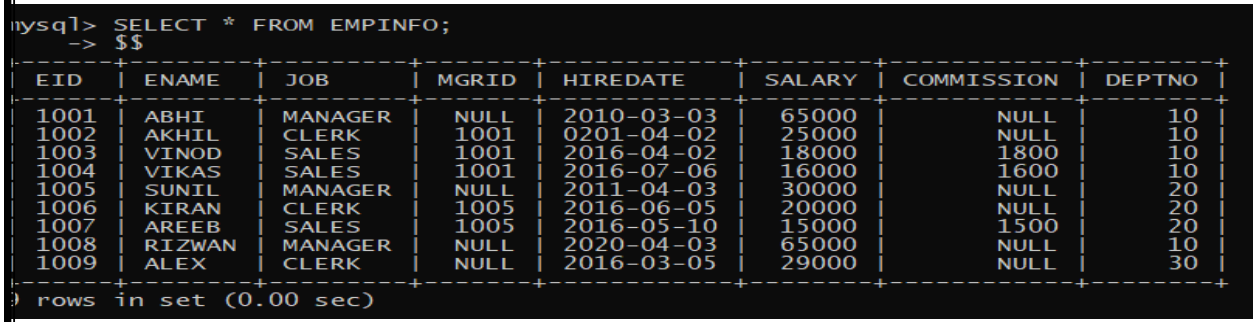
```
mysql> SELECT * FROM EMPINFO;
    -> $$
+------+--------+---------+-------+------------+--------+------------+--------+
| EID  | ENAME  | JOB     | MGRID | HIREDATE   | SALARY | COMMISSION | DEPTNO |
+------+--------+---------+-------+------------+--------+------------+--------+
| 1001 | ABHI   | MANAGER |  NULL | 2010-03-03 |  35000 |       NULL |     10 |
| 1002 | AKHIL  | CLERK   |  1001 | 0201-04-02 |  25000 |       NULL |     10 |
| 1003 | VINOD  | SALES   |  1001 | 2016-04-02 |  18000 |       1800 |     10 |
| 1004 | VIKAS  | SALES   |  1001 | 2016-07-06 |  16000 |       1600 |     10 |
| 1005 | SUNIL  | MANAGER |  NULL | 2011-04-03 |  30000 |       NULL |     20 |
| 1006 | KIRAN  | CLERK   |  1005 | 2016-06-05 |  20000 |       NULL |     20 |
| 1007 | AREEB  | SALES   |  1005 | 2016-05-10 |  15000 |       1500 |     20 |
| 1008 | RIZWAN | MANAGER |  NULL | 2020-04-03 |  65000 |       NULL |     10 |
| 1009 | ALEX   | CLERK   |  NULL | 2016-03-05 |  29000 |       NULL |     30 |
+------+--------+---------+-------+------------+--------+------------+--------+
9 rows in set (0.00 sec)
```

**4.Create a procedure to accept EID and update the SALARY in EMPINFO Table.**



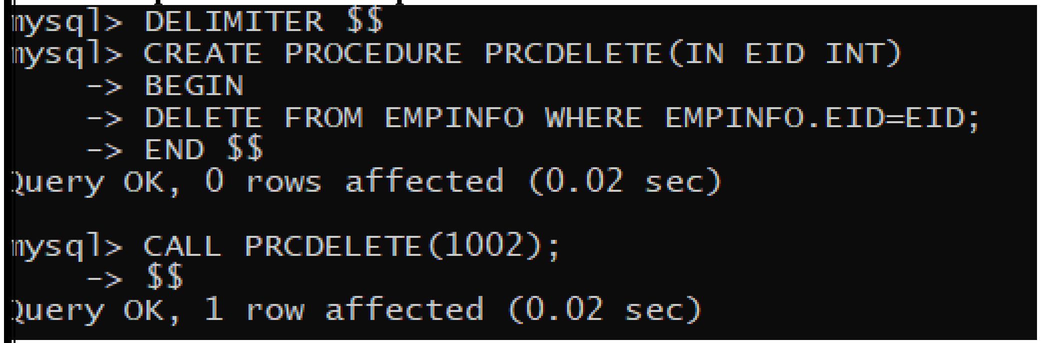
```
mysql> DELIMITER $$
mysql> CREATE PROCEDURE PRCUPDATE(IN EID INT,IN SALARY INT)
    -> BEGIN
    -> UPDATE EMPINFO SET EMPINFO.SALARY=SALARY WHERE EMPINFO.EID=EID;
    -> END $$
Query OK, 0 rows affected (0.01 sec)

mysql> CALL PRCUPDATE(1001,65000);
    -> $$
Query OK, 1 row affected (0.03 sec)
```

CHECK THE TABLE



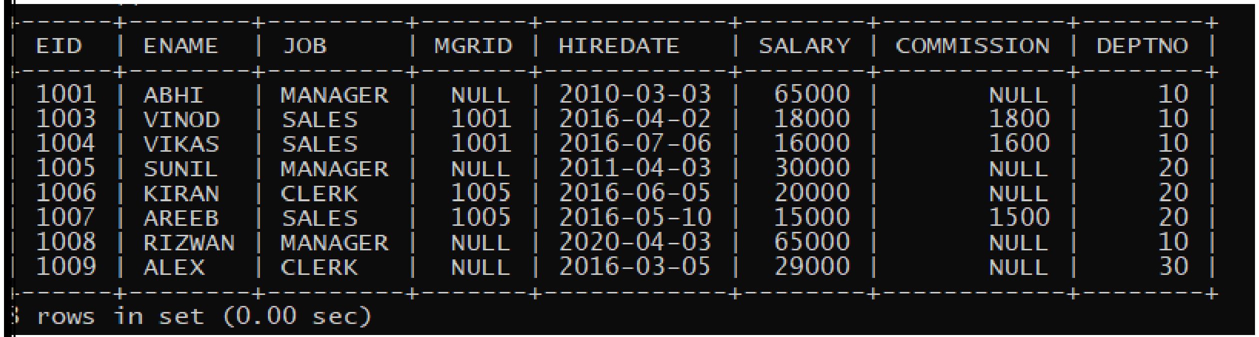
```
mysql> SELECT * FROM EMPINFO;
    -> $$
+------+--------+---------+-------+------------+--------+------------+--------+
| EID  | ENAME  | JOB     | MGRID | HIREDATE   | SALARY | COMMISSION | DEPTNO |
+------+--------+---------+-------+------------+--------+------------+--------+
| 1001 | ABHI   | MANAGER |  NULL | 2010-03-03 |  65000 |       NULL |     10 |
| 1002 | AKHIL  | CLERK   |  1001 | 0201-04-02 |  25000 |       NULL |     10 |
| 1003 | VINOD  | SALES   |  1001 | 2016-04-02 |  18000 |       1800 |     10 |
| 1004 | VIKAS  | SALES   |  1001 | 2016-07-06 |  16000 |       1600 |     10 |
| 1005 | SUNIL  | MANAGER |  NULL | 2011-04-03 |  30000 |       NULL |     20 |
| 1006 | KIRAN  | CLERK   |  1005 | 2016-06-05 |  20000 |       NULL |     20 |
| 1007 | AREEB  | SALES   |  1005 | 2016-05-10 |  15000 |       1500 |     20 |
| 1008 | RIZWAN | MANAGER |  NULL | 2020-04-03 |  65000 |       NULL |     10 |
| 1009 | ALEX   | CLERK   |  NULL | 2016-03-05 |  29000 |       NULL |     30 |
+------+--------+---------+-------+------------+--------+------------+--------+
9 rows in set (0.00 sec)
```

**5.Create a procedure to accept EID and delete the record from EMPINFO Table.**



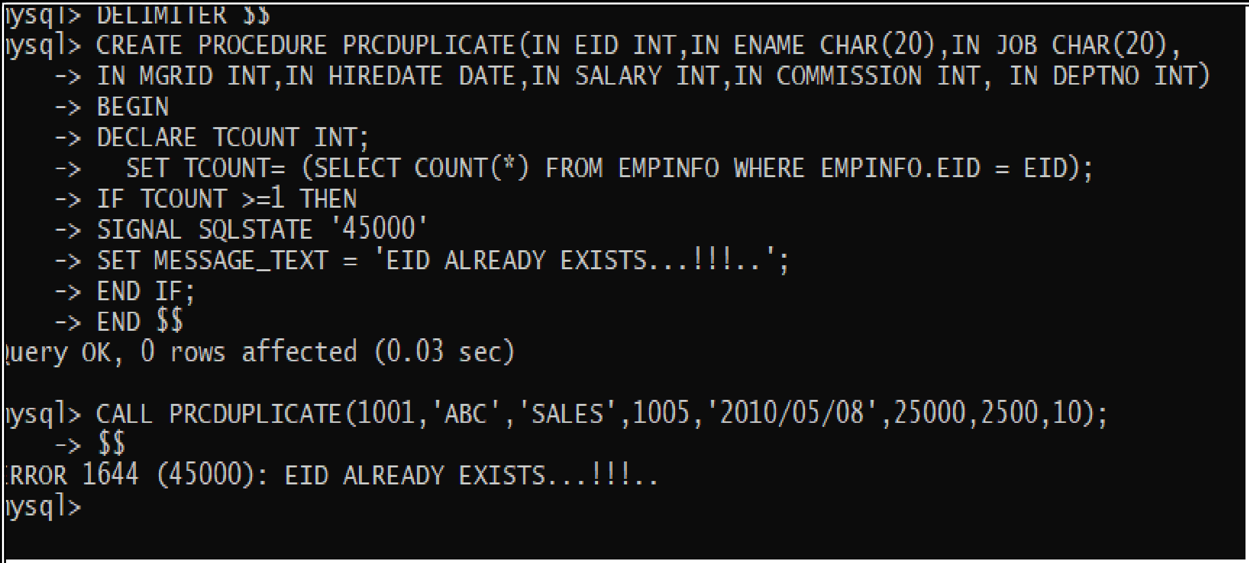
```
mysql> DELIMITER $$
mysql> CREATE PROCEDURE PRCDELETE(IN EID INT)
    -> BEGIN
    -> DELETE FROM EMPINFO WHERE EMPINFO.EID=EID;
    -> END $$
Query OK, 0 rows affected (0.02 sec)

mysql> CALL PRCDELETE(1002);
    -> $$
Query OK, 1 row affected (0.02 sec)
```

CHECK THE TABLE



```
+------+--------+---------+-------+------------+--------+------------+--------+
| EID  | ENAME  | JOB     | MGRID | HIREDATE   | SALARY | COMMISSION | DEPTNO |
+------+--------+---------+-------+------------+--------+------------+--------+
| 1001 | ABHI   | MANAGER |  NULL | 2010-03-03 |  65000 |       NULL |     10 |
| 1003 | VINOD  | SALES   |  1001 | 2016-04-02 |  18000 |       1800 |     10 |
| 1004 | VIKAS  | SALES   |  1001 | 2016-07-06 |  16000 |       1600 |     10 |
| 1005 | SUNIL  | MANAGER |  NULL | 2011-04-03 |  30000 |       NULL |     20 |
| 1006 | KIRAN  | CLERK   |  1005 | 2016-06-05 |  20000 |       NULL |     20 |
| 1007 | AREEB  | SALES   |  1005 | 2016-05-10 |  15000 |       1500 |     20 |
| 1008 | RIZWAN | MANAGER |  NULL | 2020-04-03 |  65000 |       NULL |     10 |
| 1009 | ALEX   | CLERK   |  NULL | 2016-03-05 |  29000 |       NULL |     30 |
+------+--------+---------+-------+------------+--------+------------+--------+
8 rows in set (0.00 sec)
```

**6.Create a procedure to check duplicate EID while ADDING Record in EMPINFO Table.**



```
ysql> DELIMITER $$
ysql> CREATE PROCEDURE PRCDUPLICATE(IN EID INT,IN ENAME CHAR(20),IN JOB CHAR(20),
    -> IN MGRID INT,IN HIREDATE DATE,IN SALARY INT,IN COMMISSION INT, IN DEPTNO INT)
    -> BEGIN
    -> DECLARE TCOUNT INT;
    ->   SET TCOUNT= (SELECT COUNT(*) FROM EMPINFO WHERE EMPINFO.EID = EID);
    -> IF TCOUNT >=1 THEN
    -> SIGNAL SQLSTATE '45000'
    -> SET MESSAGE_TEXT = 'EID ALREADY EXISTS...!!!..';
    -> END IF;
    -> END $$
uery OK, 0 rows affected (0.03 sec)

ysql> CALL PRCDUPLICATE(1001,'ABC','SALES',1005,'2010/05/08',25000,2500,10);
    -> $$
RROR 1644 (45000): EID ALREADY EXISTS...!!!..
ysql>
```

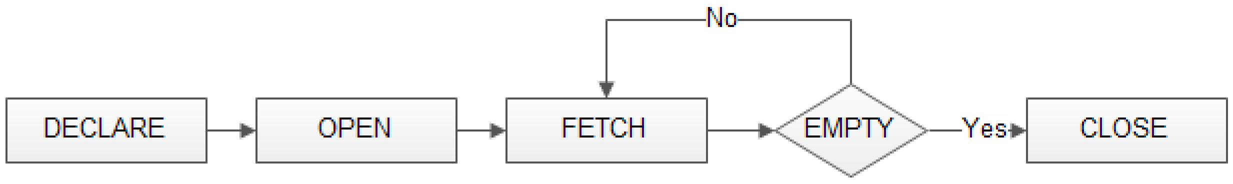
**EXPERIMENT – 10**
**CURSORS**

Aim: Declare a cursor that defines a result set. Open the cursor to establish the result set. Fetch the data into local variables as needed from the cursor, one row at a time. Close the cursor when done.

## MySQL Cursor

- Declare Cursor. A cursor is a select statement, defined in the declaration section in MySQL.
- Open Cursor. After declaring the cursor, the next step is to open the cursor using open statement.
- Fetch Cursor. After declaring and opening the cursor, the next step is to fetch the cursor. ...
- Close Cursor. The final step is to close the cursor.

The following diagram illustrates how MySQL cursor works.

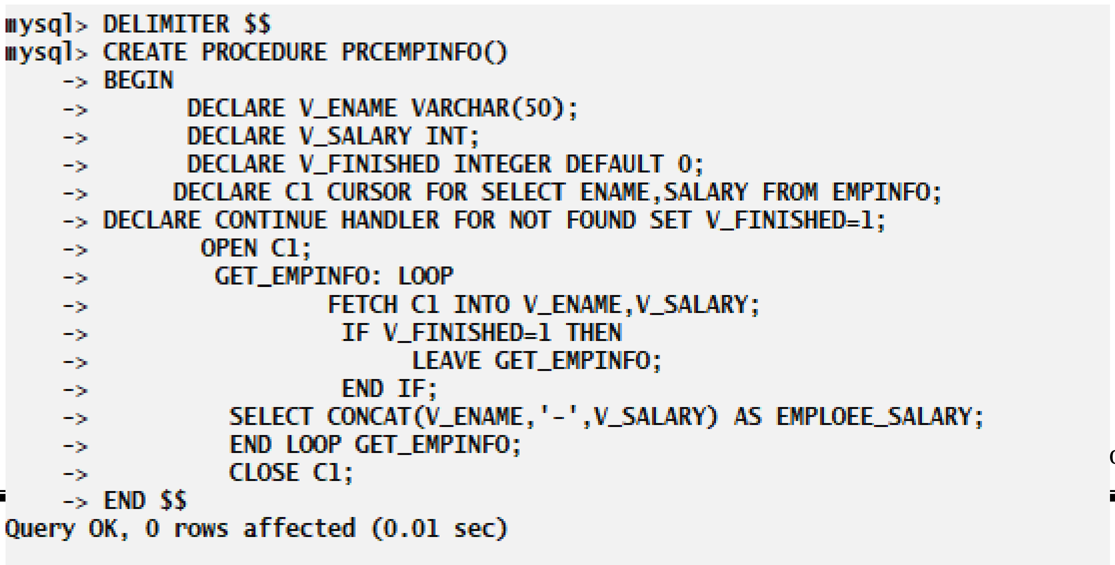


Declare Cursor
Syntax
**DECLARE** cursor_name **CURSOR FOR** Select statement;

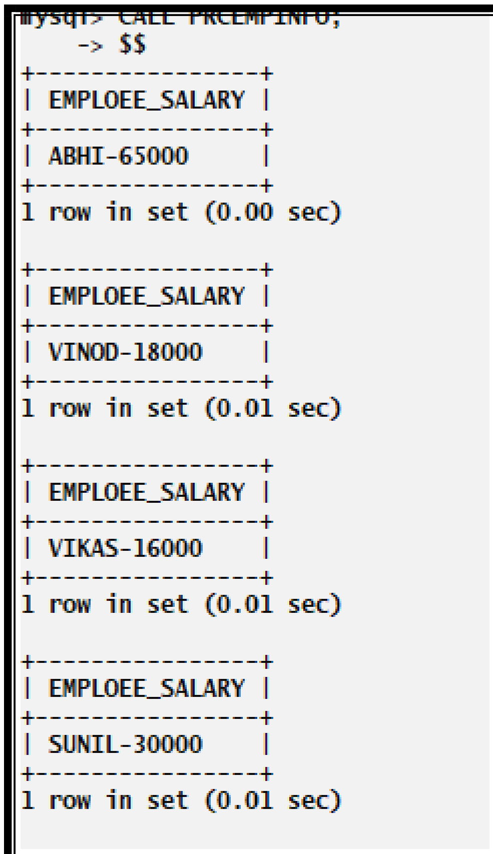
Open Cursor
Syntax
**Open** cursor_name;

Fetch Cursor
Syntax
**FETCH <cursor_name> INTO <variable_list>;**
Close Cursor
Syntax
**Close** cursor_name;

**1.Create a CURSOR to display records i.e., Empname and salary from EMPINFO table.**

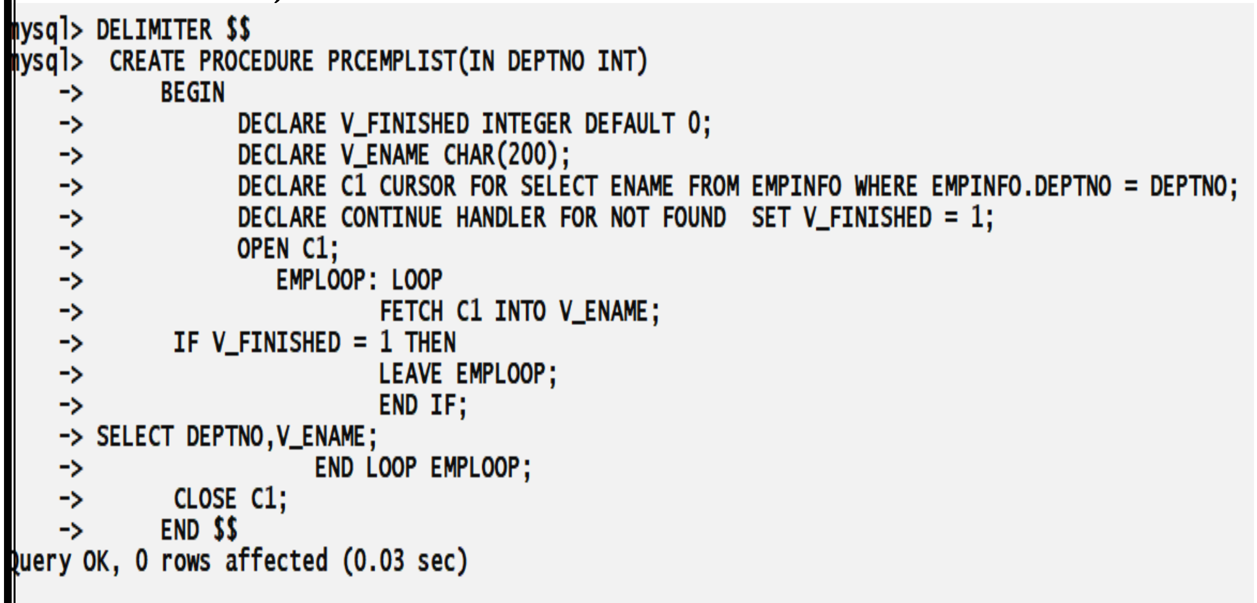


```
mysql> DELIMITER $$
mysql> CREATE PROCEDURE PRCEMPINFO()
    -> BEGIN
    ->       DECLARE V_ENAME VARCHAR(50);
    ->       DECLARE V_SALARY INT;
    ->       DECLARE V_FINISHED INTEGER DEFAULT 0;
    ->      DECLARE C1 CURSOR FOR SELECT ENAME,SALARY FROM EMPINFO;
    -> DECLARE CONTINUE HANDLER FOR NOT FOUND SET V_FINISHED=1;
    ->        OPEN C1;
    ->         GET_EMPINFO: LOOP
    ->                 FETCH C1 INTO V_ENAME,V_SALARY;
    ->                  IF V_FINISHED=1 THEN
    ->                       LEAVE GET_EMPINFO;
    ->                  END IF;
    ->          SELECT CONCAT(V_ENAME,'-',V_SALARY) AS EMPLOEE_SALARY;
    ->          END LOOP GET_EMPINFO;
    ->          CLOSE C1;
    -> END $$
Query OK, 0 rows affected (0.01 sec)
```



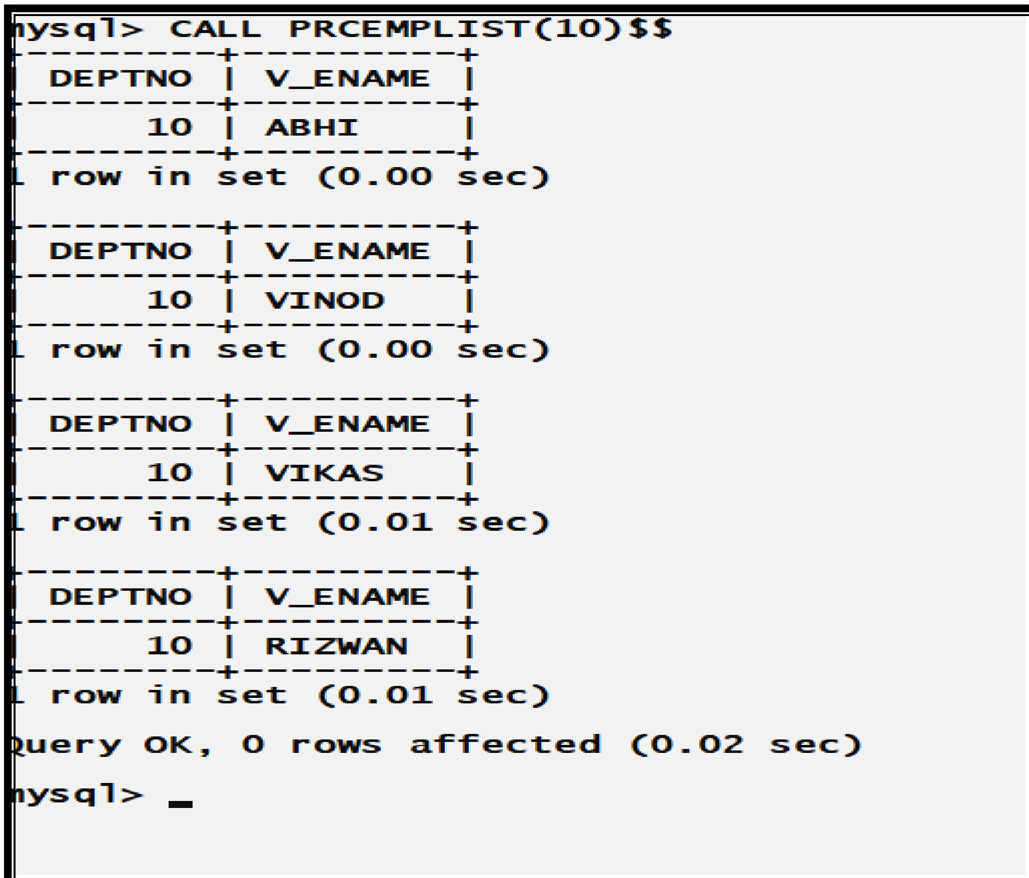
```
mysql> CALL PRCEMPINFO;
    -> $$
+----------------+
| EMPLOEE_SALARY |
+----------------+
| ABHI-65000     |
+----------------+
1 row in set (0.00 sec)

+----------------+
| EMPLOEE_SALARY |
+----------------+
| VINOD-18000    |
+----------------+
1 row in set (0.01 sec)

+----------------+
| EMPLOEE_SALARY |
+----------------+
| VIKAS-16000    |
+----------------+
1 row in set (0.01 sec)

+----------------+
| EMPLOEE_SALARY |
+----------------+
| SUNIL-30000    |
+----------------+
1 row in set (0.01 sec)
```

**2. Create procedure with cursor to accept deptno and display the employee names in that department.(USE EMPINFO TABLE)**



```
mysql> DELIMITER $$
mysql>  CREATE PROCEDURE PRCEMPLIST(IN DEPTNO INT)
    ->      BEGIN
    ->           DECLARE V_FINISHED INTEGER DEFAULT 0;
    ->           DECLARE V_ENAME CHAR(200);
    ->           DECLARE C1 CURSOR FOR SELECT ENAME FROM EMPINFO WHERE EMPINFO.DEPTNO = DEPTNO;
    ->           DECLARE CONTINUE HANDLER FOR NOT FOUND  SET V_FINISHED = 1;
    ->           OPEN C1;
    ->              EMPLOOP: LOOP
    ->                      FETCH C1 INTO V_ENAME;
    ->       IF V_FINISHED = 1 THEN
    ->                      LEAVE EMPLOOP;
    ->                      END IF;
    -> SELECT DEPTNO,V_ENAME;
    ->                 END LOOP EMPLOOP;
    ->       CLOSE C1;
    ->      END $$
Query OK, 0 rows affected (0.03 sec)
```

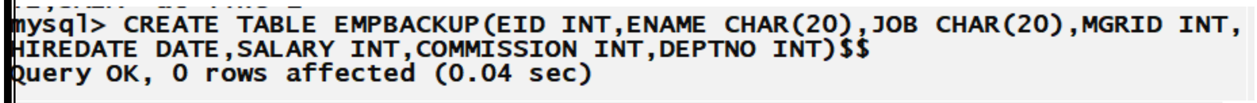
**Now call the procedure to execute the cursor**



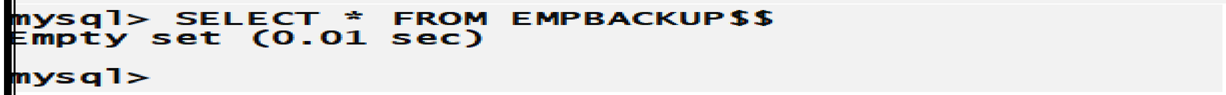
```
mysql> CALL PRCEMPLIST(10)$$
+---------+---------+
| DEPTNO  | V_ENAME |
+---------+---------+
|      10 | ABHI    |
+---------+---------+
1 row in set (0.00 sec)

+---------+---------+
| DEPTNO  | V_ENAME |
+---------+---------+
|      10 | VINOD   |
+---------+---------+
1 row in set (0.00 sec)

+---------+---------+
| DEPTNO  | V_ENAME |
+---------+---------+
|      10 | VIKAS   |
+---------+---------+
1 row in set (0.01 sec)

+---------+---------+
| DEPTNO  | V_ENAME |
+---------+---------+
|      10 | RIZWAN  |
+---------+---------+
1 row in set (0.01 sec)

Query OK, 0 rows affected (0.02 sec)

mysql> _
```

**3.Create a cursor to copy the EMPINFO table data into another table.**

First create an empty table as EMPBACKUP in which you want to copy data.

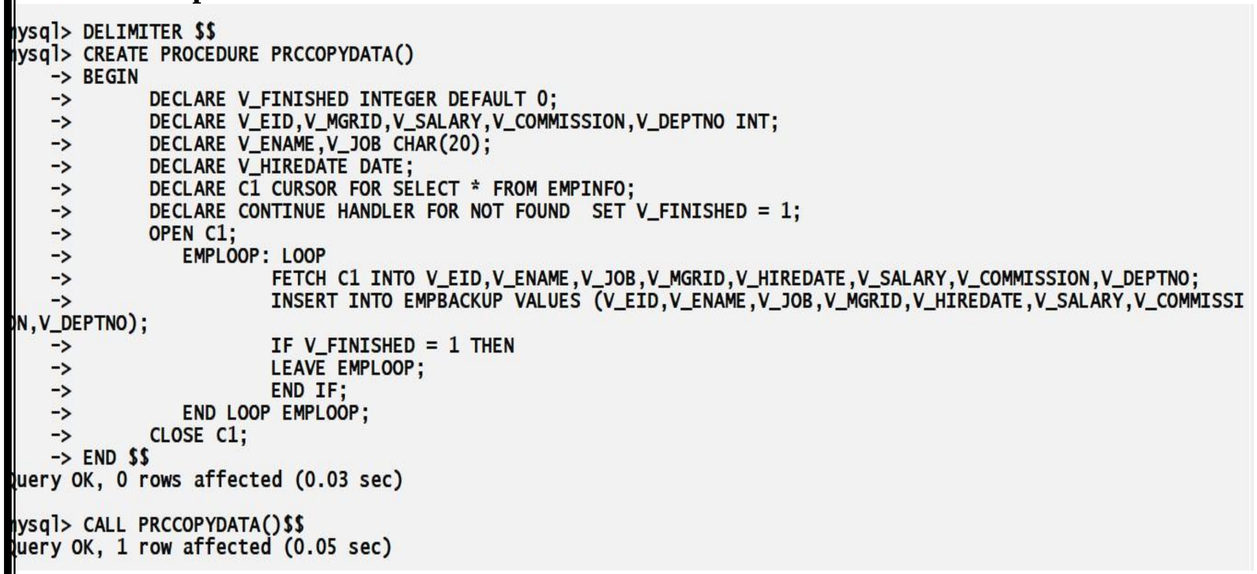


```
mysql> CREATE TABLE EMPBACKUP(EID INT,ENAME CHAR(20),JOB CHAR(20),MGRID INT,
HIREDATE DATE,SALARY INT,COMMISSION INT,DEPTNO INT)$$
Query OK, 0 rows affected (0.04 sec)
```



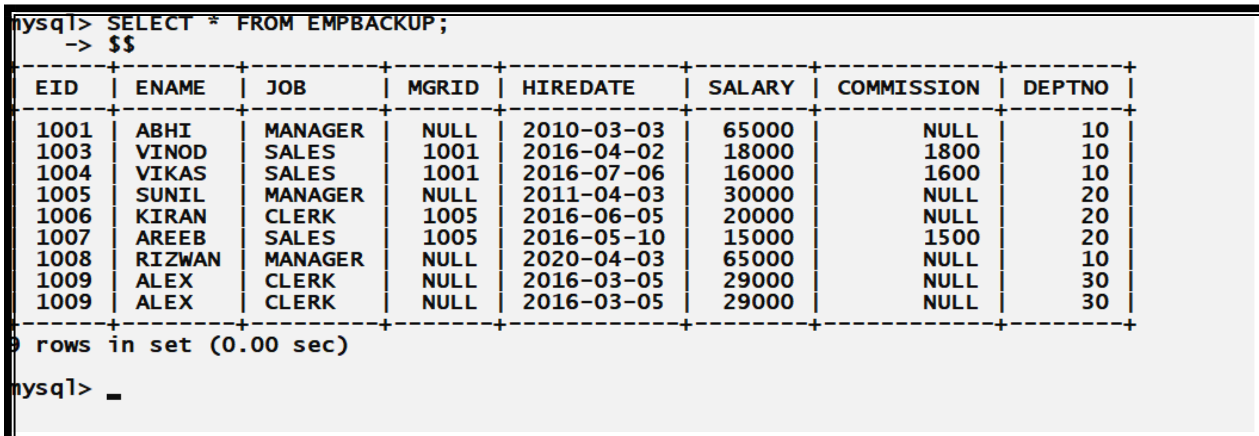
```
mysql> SELECT * FROM EMPBACKUP$$
Empty set (0.01 sec)

mysql>
```

**Now create a procedure with cursor**



```
mysql> DELIMITER $$
mysql> CREATE PROCEDURE PRCCOPYDATA()
    -> BEGIN
    ->       DECLARE V_FINISHED INTEGER DEFAULT 0;
    ->       DECLARE V_EID,V_MGRID,V_SALARY,V_COMMISSION,V_DEPTNO INT;
    ->       DECLARE V_ENAME,V_JOB CHAR(20);
    ->       DECLARE V_HIREDATE DATE;
    ->       DECLARE C1 CURSOR FOR SELECT * FROM EMPINFO;
    ->       DECLARE CONTINUE HANDLER FOR NOT FOUND  SET V_FINISHED = 1;
    ->       OPEN C1;
    ->          EMPLOOP: LOOP
    ->                  FETCH C1 INTO V_EID,V_ENAME,V_JOB,V_MGRID,V_HIREDATE,V_SALARY,V_COMMISSION,V_DEPTNO;
    ->                  INSERT INTO EMPBACKUP VALUES (V_EID,V_ENAME,V_JOB,V_MGRID,V_HIREDATE,V_SALARY,V_COMMISSI
ON,V_DEPTNO);
    ->                  IF V_FINISHED = 1 THEN
    ->                  LEAVE EMPLOOP;
    ->                  END IF;
    ->          END LOOP EMPLOOP;
    ->       CLOSE C1;
    -> END $$
Query OK, 0 rows affected (0.03 sec)

mysql> CALL PRCCOPYDATA()$$
Query OK, 1 row affected (0.05 sec)
```

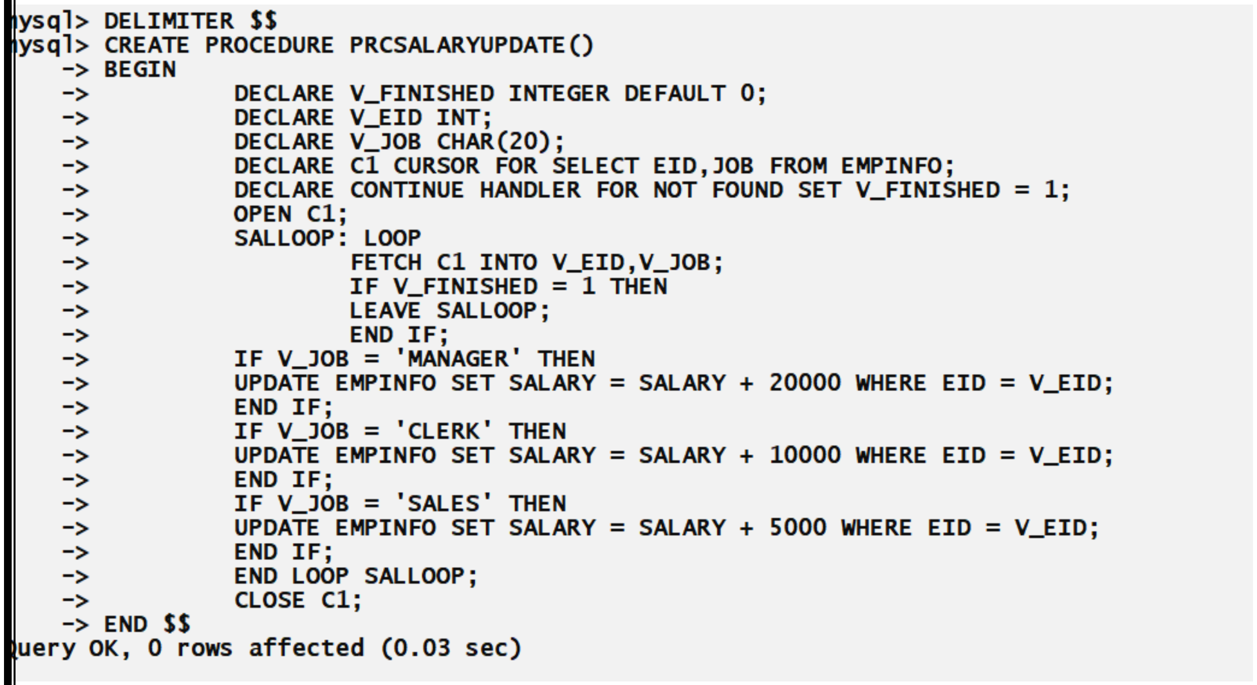
**Now check the table EMPBACKUP**



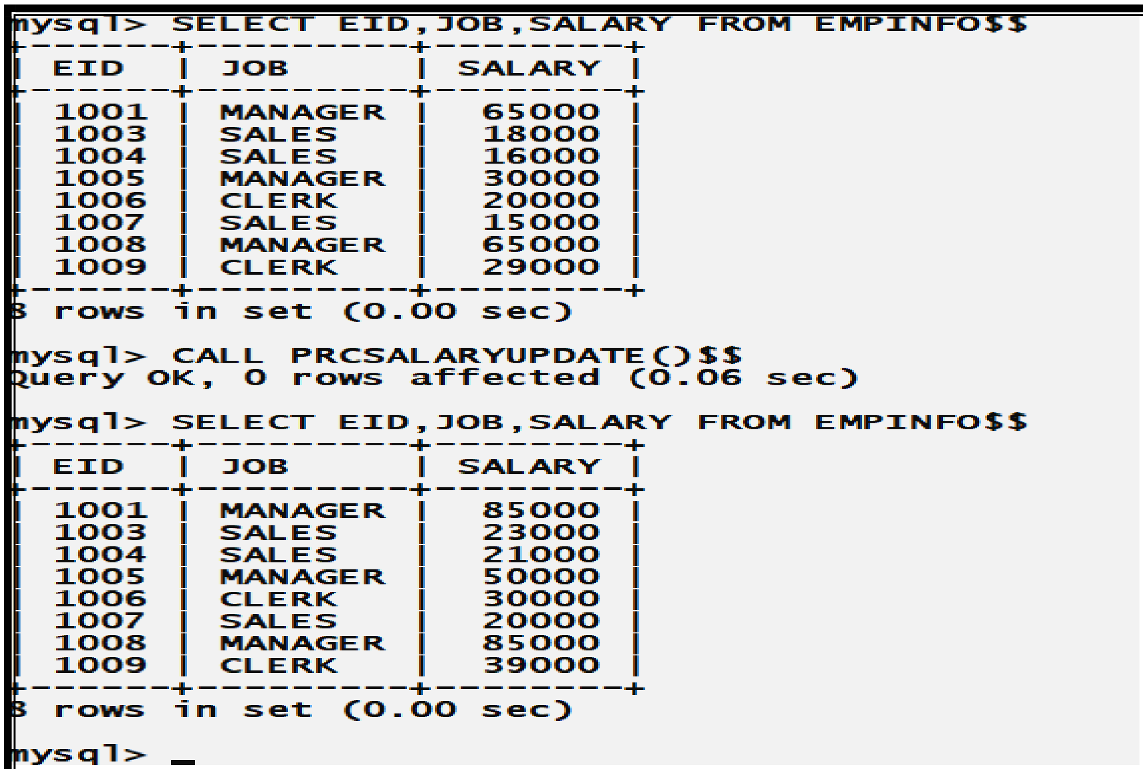
```
ysql> SELECT * FROM EMPBACKUP;
    -> $$
+------+--------+---------+-------+------------+--------+------------+--------+
| EID  | ENAME  | JOB     | MGRID | HIREDATE   | SALARY | COMMISSION | DEPTNO |
+------+--------+---------+-------+------------+--------+------------+--------+
| 1001 | ABHI   | MANAGER |  NULL | 2010-03-03 |  65000 |       NULL |     10 |
| 1003 | VINOD  | SALES   |  1001 | 2016-04-02 |  18000 |       1800 |     10 |
| 1004 | VIKAS  | SALES   |  1001 | 2016-07-06 |  16000 |       1600 |     10 |
| 1005 | SUNIL  | MANAGER |  NULL | 2011-04-03 |  30000 |       NULL |     20 |
| 1006 | KIRAN  | CLERK   |  1005 | 2016-06-05 |  20000 |       NULL |     20 |
| 1007 | AREEB  | SALES   |  1005 | 2016-05-10 |  15000 |       1500 |     20 |
| 1008 | RIZWAN | MANAGER |  NULL | 2020-04-03 |  65000 |       NULL |     10 |
| 1009 | ALEX   | CLERK   |  NULL | 2016-03-05 |  29000 |       NULL |     30 |
| 1009 | ALEX   | CLERK   |  NULL | 2016-03-05 |  29000 |       NULL |     30 |
+------+--------+---------+-------+------------+--------+------------+--------+
 rows in set (0.00 sec)

ysql> _
```

**4. Create a CURSOR to increment the salary based on Designation (here JOB).**

| If job = manager; | If job = clerk; | If job = sales; |
|---|---|---|
| Increment by20000 | Increment by 10000 | Increment by 5000 |

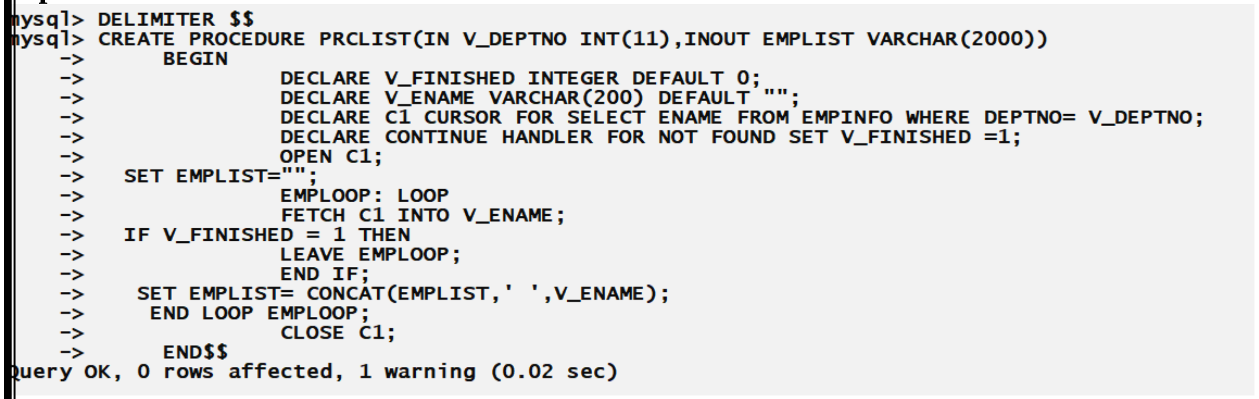


```
ysql> DELIMITER $$
ysql> CREATE PROCEDURE PRCSALARYUPDATE()
    -> BEGIN
    ->          DECLARE V_FINISHED INTEGER DEFAULT 0;
    ->          DECLARE V_EID INT;
    ->          DECLARE V_JOB CHAR(20);
    ->          DECLARE C1 CURSOR FOR SELECT EID,JOB FROM EMPINFO;
    ->          DECLARE CONTINUE HANDLER FOR NOT FOUND SET V_FINISHED = 1;
    ->          OPEN C1;
    ->          SALLOOP: LOOP
    ->                  FETCH C1 INTO V_EID,V_JOB;
    ->                  IF V_FINISHED = 1 THEN
    ->                  LEAVE SALLOOP;
    ->                  END IF;
    ->          IF V_JOB = 'MANAGER' THEN
    ->          UPDATE EMPINFO SET SALARY = SALARY + 20000 WHERE EID = V_EID;
    ->          END IF;
    ->          IF V_JOB = 'CLERK' THEN
    ->          UPDATE EMPINFO SET SALARY = SALARY + 10000 WHERE EID = V_EID;
    ->          END IF;
    ->          IF V_JOB = 'SALES' THEN
    ->          UPDATE EMPINFO SET SALARY = SALARY + 5000 WHERE EID = V_EID;
    ->          END IF;
    ->          END LOOP SALLOOP;
    ->          CLOSE C1;
    -> END $$
uery OK, 0 rows affected (0.03 sec)
```



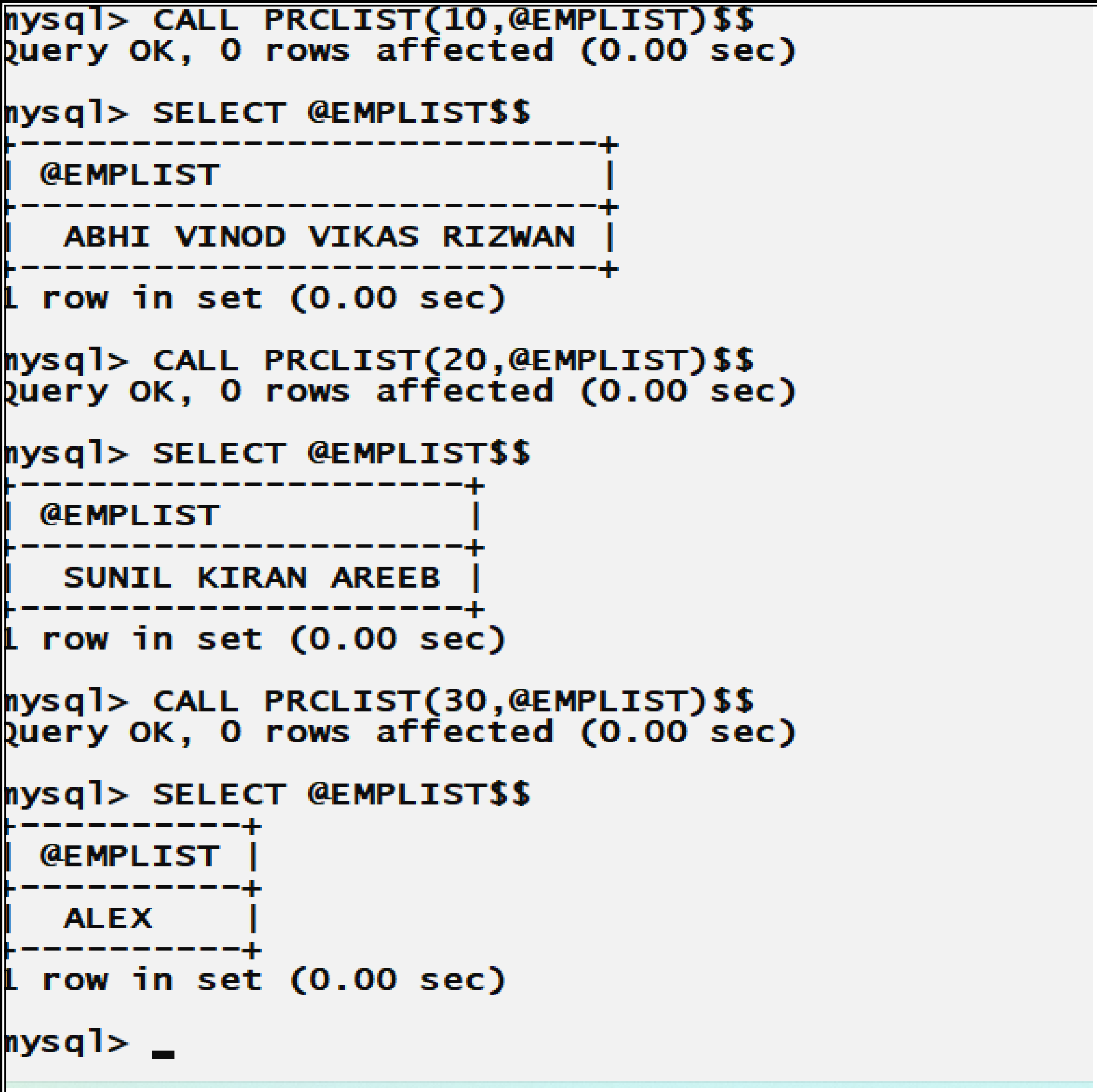
```
mysql> SELECT EID,JOB,SALARY FROM EMPINFO$$
+------+---------+--------+
| EID  | JOB     | SALARY |
+------+---------+--------+
| 1001 | MANAGER |  65000 |
| 1003 | SALES   |  18000 |
| 1004 | SALES   |  16000 |
| 1005 | MANAGER |  30000 |
| 1006 | CLERK   |  20000 |
| 1007 | SALES   |  15000 |
| 1008 | MANAGER |  65000 |
| 1009 | CLERK   |  29000 |
+------+---------+--------+
8 rows in set (0.00 sec)

mysql> CALL PRCSALARYUPDATE()$$
Query OK, 0 rows affected (0.06 sec)

mysql> SELECT EID,JOB,SALARY FROM EMPINFO$$
+------+---------+--------+
| EID  | JOB     | SALARY |
+------+---------+--------+
| 1001 | MANAGER |  85000 |
| 1003 | SALES   |  23000 |
| 1004 | SALES   |  21000 |
| 1005 | MANAGER |  50000 |
| 1006 | CLERK   |  30000 |
| 1007 | SALES   |  20000 |
| 1008 | MANAGER |  85000 |
| 1009 | CLERK   |  39000 |
+------+---------+--------+
8 rows in set (0.00 sec)

mysql> _
```

**5.Create a procedure with CURSOR to accept a deptno and display the employee's names as list belongs to that department.**



```
mysql> DELIMITER $$
mysql> CREATE PROCEDURE PRCLIST(IN V_DEPTNO INT(11),INOUT EMPLIST VARCHAR(2000))
    ->      BEGIN
    ->               DECLARE V_FINISHED INTEGER DEFAULT 0;
    ->               DECLARE V_ENAME VARCHAR(200) DEFAULT "";
    ->               DECLARE C1 CURSOR FOR SELECT ENAME FROM EMPINFO WHERE DEPTNO= V_DEPTNO;
    ->               DECLARE CONTINUE HANDLER FOR NOT FOUND SET V_FINISHED =1;
    ->               OPEN C1;
    ->   SET EMPLIST="";
    ->               EMPLOOP: LOOP
    ->               FETCH C1 INTO V_ENAME;
    ->   IF V_FINISHED = 1 THEN
    ->               LEAVE EMPLOOP;
    ->               END IF;
    ->    SET EMPLIST= CONCAT(EMPLIST,' ',V_ENAME);
    ->     END LOOP EMPLOOP;
    ->               CLOSE C1;
    ->      END$$
Query OK, 0 rows affected, 1 warning (0.02 sec)
```

**Now call the procedure**



```
mysql> CALL PRCLIST(10,@EMPLIST)$$
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @EMPLIST$$
+---------------------------+
| @EMPLIST                  |
+---------------------------+
|  ABHI VINOD VIKAS RIZWAN |
+---------------------------+
1 row in set (0.00 sec)

mysql> CALL PRCLIST(20,@EMPLIST)$$
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @EMPLIST$$
+---------------------+
| @EMPLIST            |
+---------------------+
|  SUNIL KIRAN AREEB |
+---------------------+
1 row in set (0.00 sec)

mysql> CALL PRCLIST(30,@EMPLIST)$$
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @EMPLIST$$
+----------+
| @EMPLIST |
+----------+
|  ALEX    |
+----------+
1 row in set (0.00 sec)

mysql> _
```