M.Sc. (Five Year Integrated) in Computer Science (Artificial Intelligence & Data Science)

Third Semester

Laboratory Record

23-813-0307: DATABASE SYSTEMS LAB

Submitted in partial fulfillment
of the requirements for the award of degree in
Master of Science (Five Year Integrated)
in Computer Science (Artificial Intelligence & Data Science) of
Cochin University of Science and Technology (CUSAT)
Kochi

COCHIN UNIVERSITY OF
SCIENCE AND TECHNOLOGY

Submitted by

NANDAKISHORE V J
(81323017)

DEPARTMENT OF COMPUTER SCIENCE
COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY (CUSAT)
KOCHI-682022

APRIL 2024

DEPARTMENT OF COMPUTER SCIENCE

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY (CUSAT)

KOCHI, KERALA-682022

This is to certify that the software laboratory record for ***23-813-0307: DATABASE SYSYTEMS Lab*** *is a record of work carried out by* ***NANDAKISHORE V J(81323017)****, in partial fulfillment of the requirements for the award of degree in* ***Master of Science (Five Year Integrated)*** *in* ***Computer Science (Artificial Intelligence & Data Science)*** *of Cochin University of Science and Technology (CUSAT), Kochi. The lab record has been approved as it satisfies the academic requirements in respect of the second semester laboratory prescribed for the Master of Science (Five Year Integrated) in Computer Science degree.*

Faculty Member in-charge

Dr. Ajees A P	Dr. Madhu S Nair
Assistant Professor	Professor and Head
Department of Computer Science	Department of Computer Science
CUSAT	CUSAT

Contents

1.DDL & DQL COMMANDS

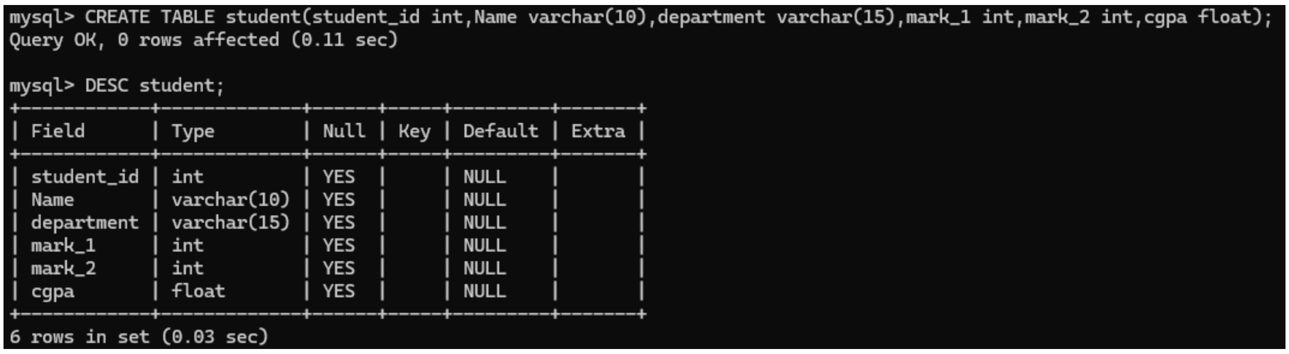
AIM

Creation of a database using DDL commands and writes DQL queries to retrieve information from the database.

QUERY-OUTPUT

· Create a Table STUDENT with following attributes:
std_id, stud_name, block, roomno and insert 5 rows for the table

```
mysql> CREATE TABLE student(student_id int,Name varchar(10),department varchar(15),
mark_1 int,mark_2 int,cgpa float);
 Query OK, 0 rows affected (0.11 sec)

 mysql> INSERT INTO student values (1,'RAVI','CS',92,94,9.2);
Query OK, 1 row affected (0.05 sec)

mysql> INSERT INTO student values (2,'ANU','SMS',97,91,9.3),
    -> (3,'AKHIL','CA',88,93,8.9),
    -> (4,'AMAL','CS',93,95,9.4),
    -> (5,'ASWIN','CIS',97,96,9.5);
Query OK, 4 rows affected (0.02 sec)
Records: 4  Duplicates: 0  Warnings: 0
```



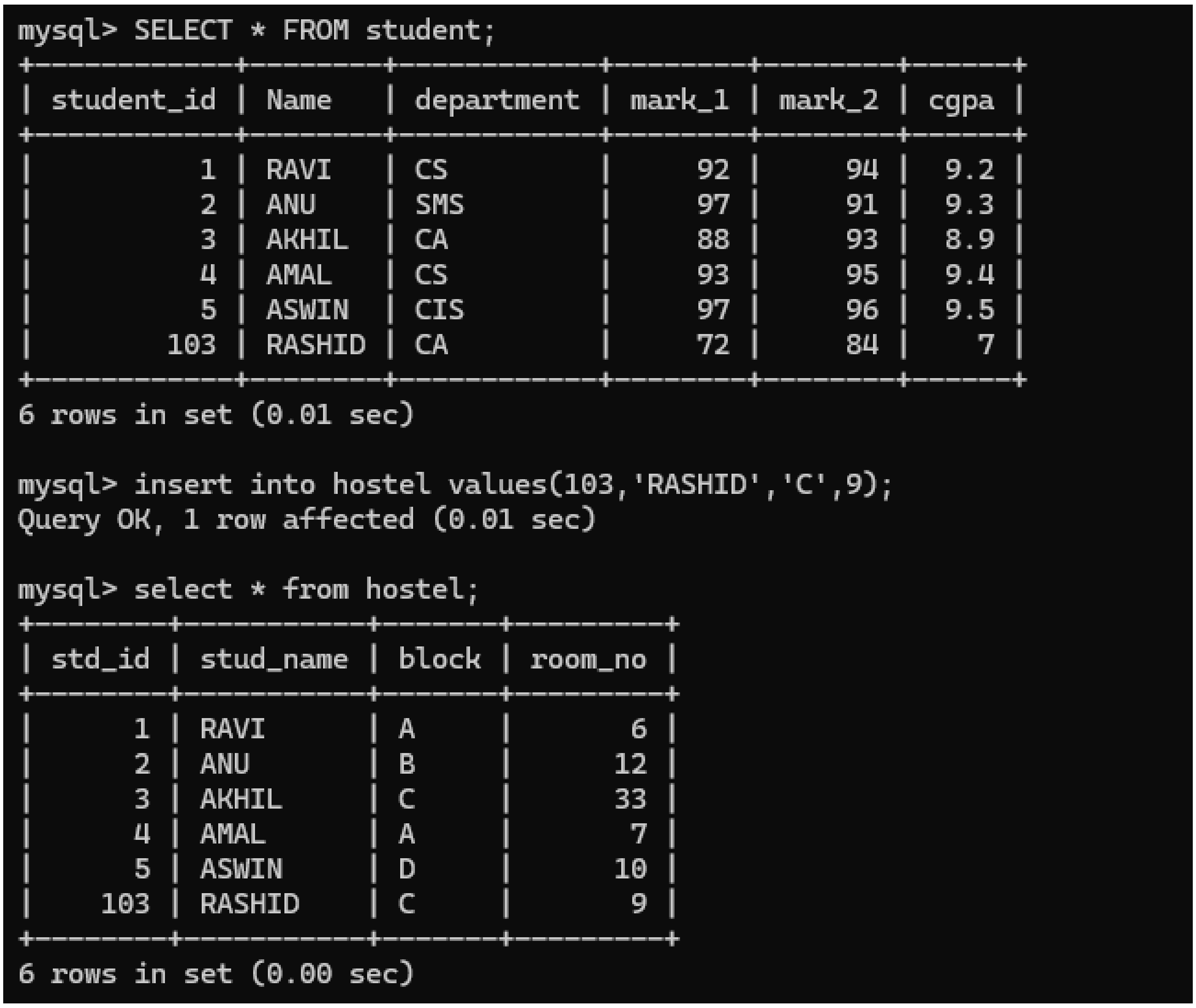
```
mysql> CREATE TABLE student(student_id int,Name varchar(10),department varchar(15),mark_1 int,mark_2 int,cgpa float);
Query OK, 0 rows affected (0.11 sec)

mysql> DESC student;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| student_id | int         | YES  |     | NULL    |       |
| Name       | varchar(10) | YES  |     | NULL    |       |
| department | varchar(15) | YES  |     | NULL    |       |
| mark_1     | int         | YES  |     | NULL    |       |
| mark_2     | int         | YES  |     | NULL    |       |
| cgpa       | float       | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
6 rows in set (0.03 sec)
```

· Create table hostel_details with following attributes: std_id, stud_name, block, roomno and insert 5 rows for the table

```
mysql> CREATE TABLE hostel_details(std_id int,stud_name varchar(10),
block char(1),room_no int);
Query OK, 0 rows affected (0.07 sec)

mysql> INSERT INTO hostel_details values (1,'RAVI','A',06),
    -> (2,'ANU','B',12),
    -> (3,'AKHIL','C',33),
    -> (4,'AMAL','A',07),
    -> (5,'ASWIN','D',10);
Query OK, 5 rows affected (0.01 sec)
```

· Display the details of student and hostel_details table

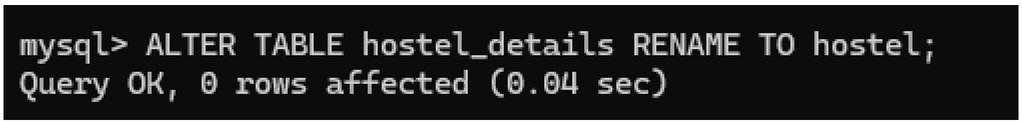
```
mysql> SELECT * FROM student;
mysql> SELECT * FROM hostel;
```



```
mysql> SELECT * FROM student;
+------------+--------+------------+--------+--------+------+
| student_id | Name   | department | mark_1 | mark_2 | cgpa |
+------------+--------+------------+--------+--------+------+
|          1 | RAVI   | CS         |     92 |     94 |  9.2 |
|          2 | ANU    | SMS        |     97 |     91 |  9.3 |
|          3 | AKHIL  | CA         |     88 |     93 |  8.9 |
|          4 | AMAL   | CS         |     93 |     95 |  9.4 |
|          5 | ASWIN  | CIS        |     97 |     96 |  9.5 |
|        103 | RASHID | CA         |     72 |     84 |    7 |
+------------+--------+------------+--------+--------+------+
6 rows in set (0.01 sec)

mysql> insert into hostel values(103,'RASHID','C',9);
Query OK, 1 row affected (0.01 sec)

mysql> select * from hostel;
+--------+-----------+-------+---------+
| std_id | stud_name | block | room_no |
+--------+-----------+-------+---------+
|      1 | RAVI      | A     |       6 |
|      2 | ANU       | B     |      12 |
|      3 | AKHIL     | C     |      33 |
|      4 | AMAL      | A     |       7 |
|      5 | ASWIN     | D     |      10 |
|    103 | RASHID    | C     |       9 |
+--------+-----------+-------+---------+
6 rows in set (0.00 sec)
```

· Rename table hostel_details to hostel

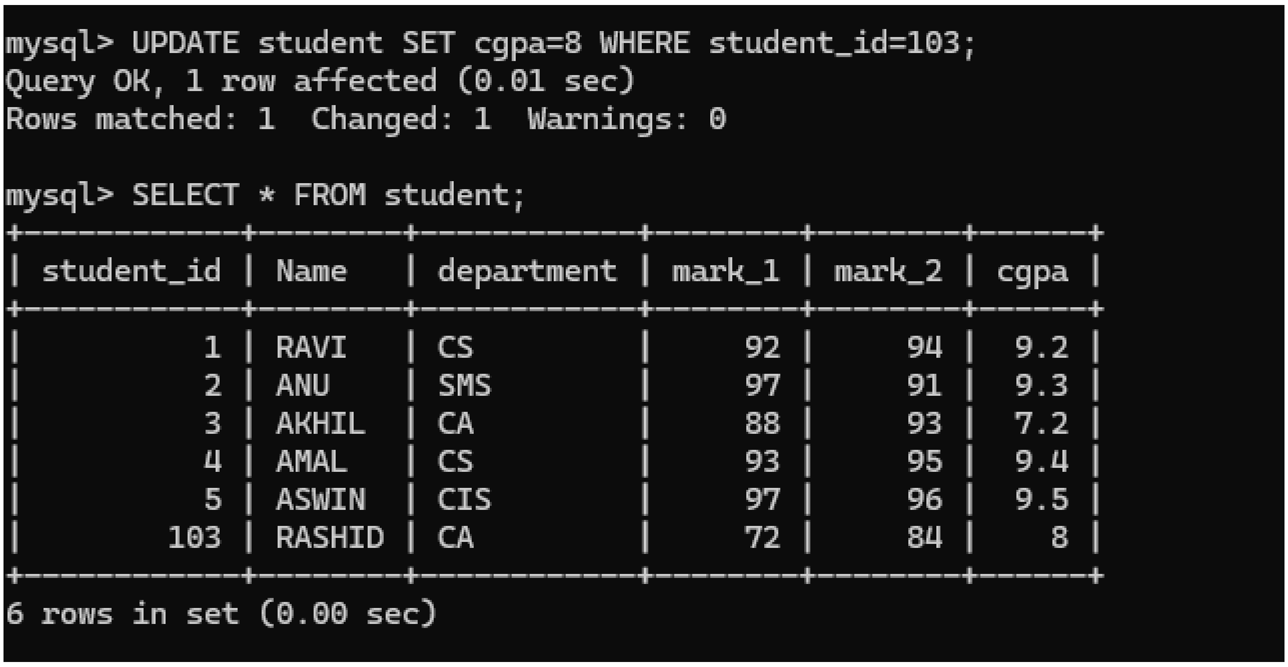
```
mysql> ALTER TABLE hostel_details RENAME TO hostel;
Query OK, 0 rows affected (0.04 sec)
```



```
mysql> ALTER TABLE hostel_details RENAME TO hostel;
Query OK, 0 rows affected (0.04 sec)
```

· Update the value of CGPA whose student_id is 103 from cgpa 7 to 8

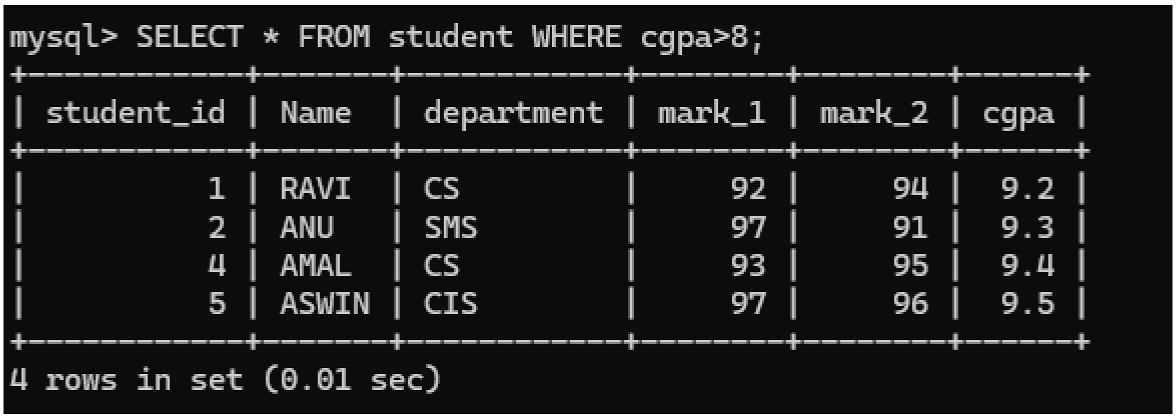
```
mysql> UPDATE student SET cgpa=8 WHERE student_id=103;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```



```
mysql> UPDATE student SET cgpa=8 WHERE student_id=103;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM student;
+------------+--------+------------+--------+--------+------+
| student_id | Name   | department | mark_1 | mark_2 | cgpa |
+------------+--------+------------+--------+--------+------+
|          1 | RAVI   | CS         |     92 |     94 |  9.2 |
|          2 | ANU    | SMS        |     97 |     91 |  9.3 |
|          3 | AKHIL  | CA         |     88 |     93 |  7.2 |
|          4 | AMAL   | CS         |     93 |     95 |  9.4 |
|          5 | ASWIN  | CIS        |     97 |     96 |  9.5 |
|        103 | RASHID | CA         |     72 |     84 |    8 |
+------------+--------+------------+--------+--------+------+
6 rows in set (0.00 sec)
```

· Display the name of the students whose cgpa is more than 8

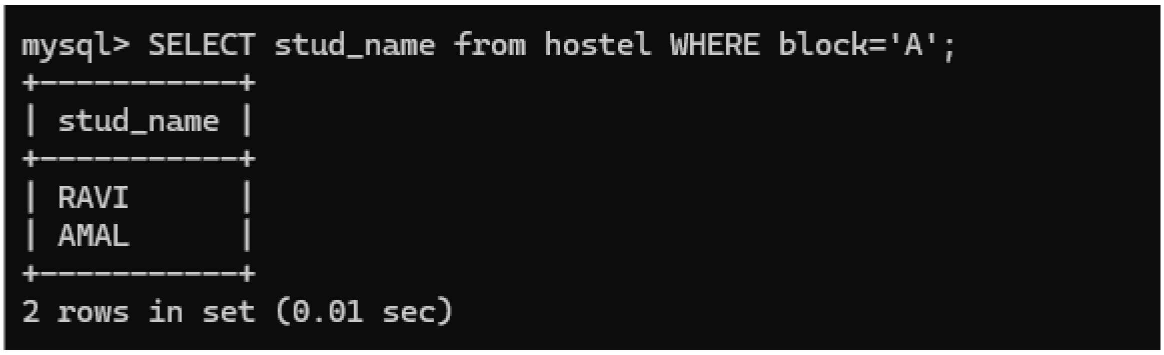
```
mysql> SELECT * FROM student WHERE cgpa>8;
```



```
mysql> SELECT * FROM student WHERE cgpa>8;
+------------+-------+------------+--------+--------+------+
| student_id | Name  | department | mark_1 | mark_2 | cgpa |
+------------+-------+------------+--------+--------+------+
|          1 | RAVI  | CS         |     92 |     94 |  9.2 |
|          2 | ANU   | SMS        |     97 |     91 |  9.3 |
|          4 | AMAL  | CS         |     93 |     95 |  9.4 |
|          5 | ASWIN | CIS        |     97 |     96 |  9.5 |
+------------+-------+------------+--------+--------+------+
4 rows in set (0.01 sec)
```

· Display the name of student staying in block A of the table hostel

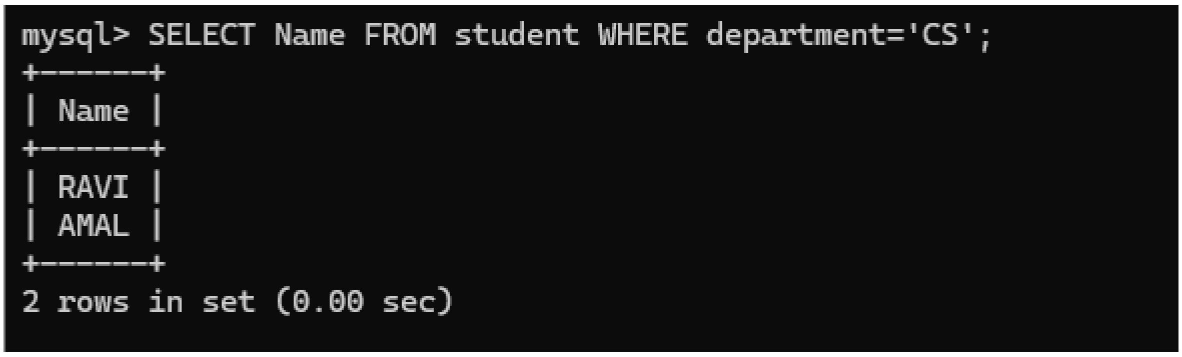
```
mysql> SELECT stud_name from hostel WHERE block='A';
```



```
mysql> SELECT stud_name from hostel WHERE block='A';
+-----------+
| stud_name |
+-----------+
| RAVI      |
| AMAL      |
+-----------+
2 rows in set (0.01 sec)
```

· Display name of student who belong to cs dept.

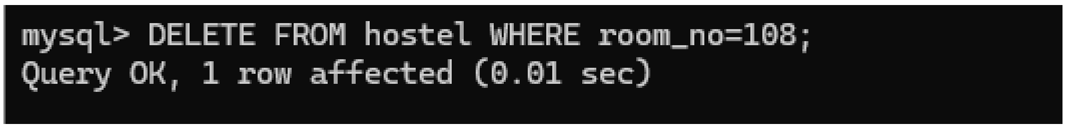
```
 mysql> SELECT Name FROM student WHERE department='CS';
```



```
mysql> SELECT Name FROM student WHERE department='CS';
+------+
| Name |
+------+
| RAVI |
| AMAL |
+------+
2 rows in set (0.00 sec)
```

· Delete the name of student whose room number is 108

```
mysql> DELETE FROM hostel_details WHERE room_no= 108;
Query OK, 1 row affected (0.01 sec)
```

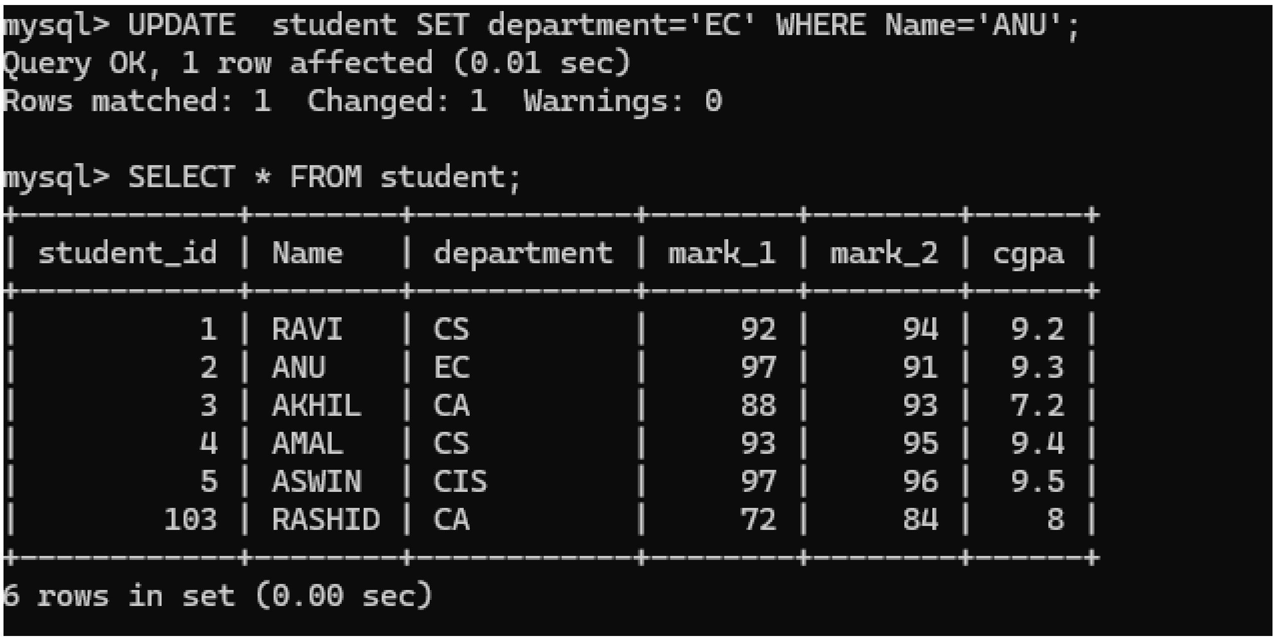


```
mysql> DELETE FROM hostel WHERE room_no=108;
Query OK, 1 row affected (0.01 sec)
```

.

Modify the department of the student' anu' from cs to ec

```
mysql> UPDATE  student SET department='EC' WHERE Name='ANU';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```



```
mysql> UPDATE  student SET department='EC' WHERE Name='ANU';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM student;
+------------+--------+------------+--------+--------+------+
| student_id | Name   | department | mark_1 | mark_2 | cgpa |
+------------+--------+------------+--------+--------+------+
|          1 | RAVI   | CS         |     92 |     94 |  9.2 |
|          2 | ANU    | EC         |     97 |     91 |  9.3 |
|          3 | AKHIL  | CA         |     88 |     93 |  7.2 |
|          4 | AMAL   | CS         |     93 |     95 |  9.4 |
|          5 | ASWIN  | CIS        |     97 |     96 |  9.5 |
|        103 | RASHID | CA         |     72 |     84 |    8 |
+------------+--------+------------+--------+--------+------+
6 rows in set (0.00 sec)
```

2.DML COMMANDS

AIM

Implement DML commands like Insertion, Deletion, Modifying, Altering, and Updating records based on conditions.

QUERY-OUTPUT

· Consider the employee database given below

· emp (emp_id,emp_name, Street_No, city)

· works (emp_id, company name, salary)

· company (company name, city)

· manages (emp_id, manager_id) Note: Emp_id starts with 'E' in Emp table and emp_id in works table is the emp_id from emptable .emp_id and manager_id in manages table is the emp_id from emp table.

· Add these four tables with sufficient constraints.

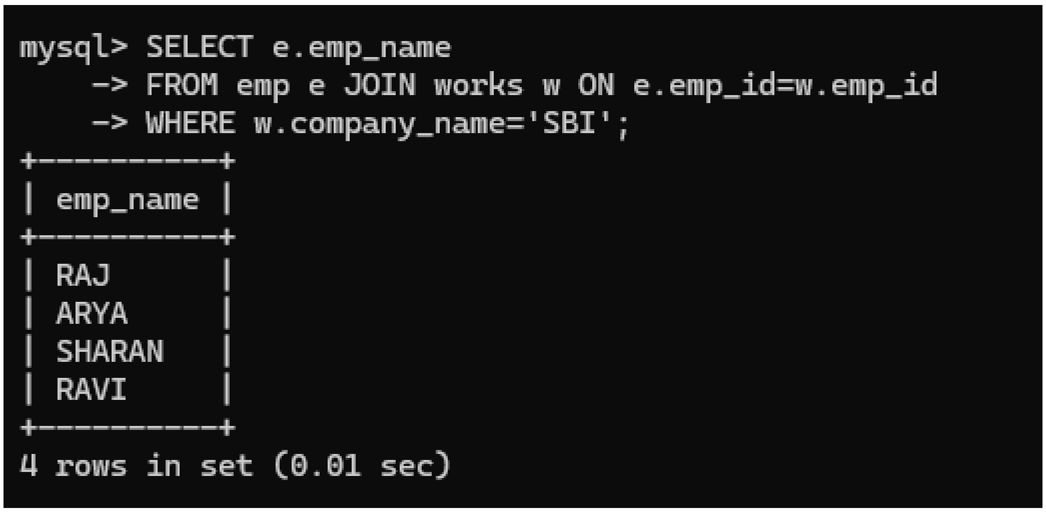
```
mysql> CREATE TABLE emp (
    -> emp_id varchar(10) PRIMARY KEY,
    -> emp_name varchar(10) NOT NULL,
    -> city varchar(10),
    -> CHECK (emp_id LIKE 'E%'));

mysql> CREATE TABLE works(
    -> emp_id varchar(10),
    -> company_name varchar(20),
    -> salary int,
    -> PRIMARY KEY(emp_id,company_name),
    -> FOREIGN KEY (emp_id) REFERENCES emp(emp_id));
Query OK, 0 rows affected (0.06 sec)

mysql> CREATE TABLE company(
    -> company_name varchar(20) PRIMARY KEY,
    -> city varchar(10));
Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TABLE manages (
    -> emp_id varchar(10),
    -> manager_id varchar(10),
    -> PRIMARY KEY (emp_id,manager_id),
    -> FOREIGN KEY (emp_id) REFERENCES emp(emp_id),
    -> FOREIGN KEY (manager_id) REFERENCES emp(emp_id));
Query OK, 0 rows affected (0.05 sec)
```

· Alter table emp add a constraint that emp_name cannot be null

```
mysql> ALTER TABLE emp MODIFY COLUMN emp_name varchar(10) NOT NULL;
Query OK, 0 rows affected (0.11 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

· Find the names of all employees who work for SBI.

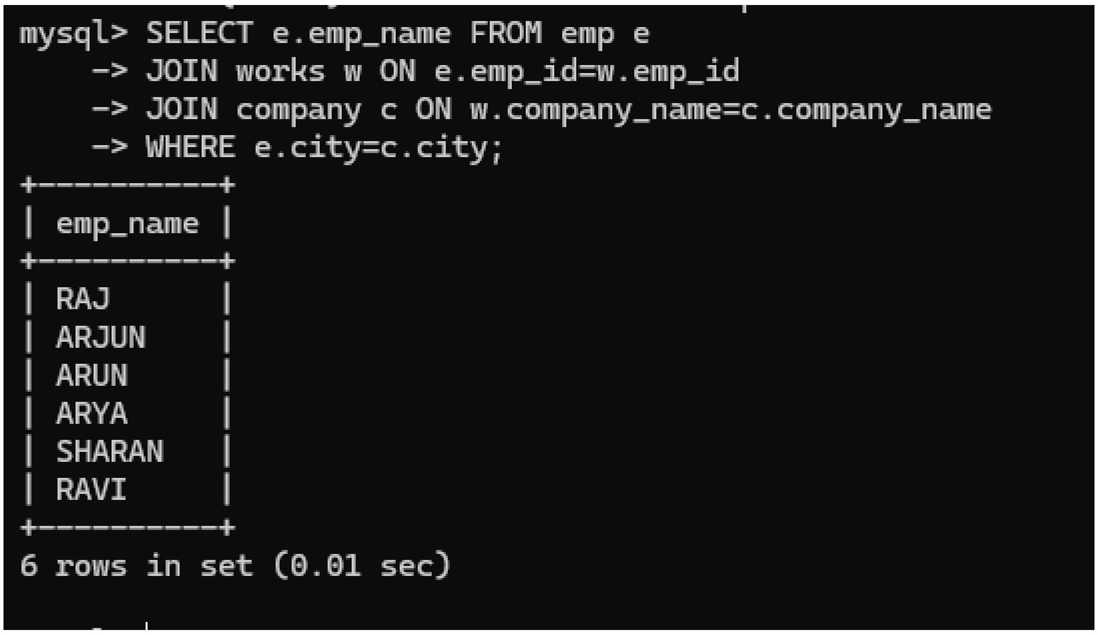
```
mysql> SELECT e.emp_name
    -> FROM emp e JOIN works w ON e.emp_id=w.emp_id
    -> WHERE w.company_name='SBI';
```



```
mysql> SELECT e.emp_name
    -> FROM emp e JOIN works w ON e.emp_id=w.emp_id
    -> WHERE w.company_name='SBI';
+----------+
| emp_name |
+----------+
| RAJ      |
| ARYA     |
| SHARAN   |
| RAVI     |
+----------+
4 rows in set (0.01 sec)
```

· Find all employees in the database who live in the same cities as the companies for which they work.

```
mysql> SELECT e.emp_name FROM emp e
    -> JOIN works w ON e.emp_id=w.emp_id
    -> JOIN company c ON w.company_name=c.company_name
    -> WHERE e.city=c.city;
```

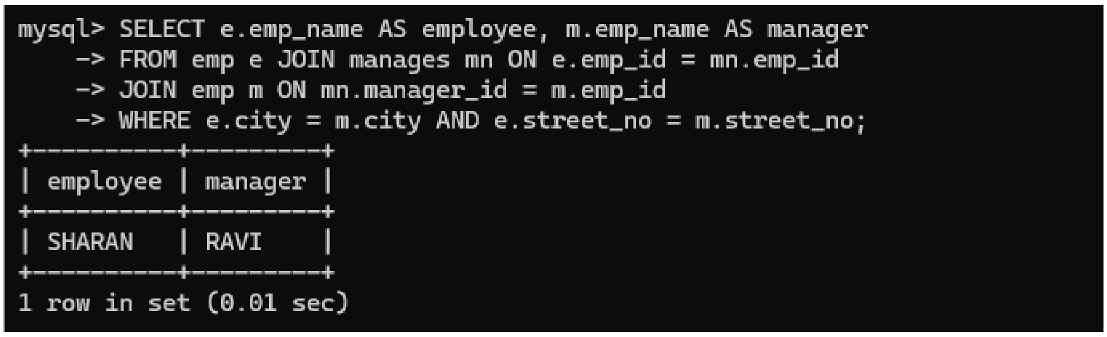


```
mysql> SELECT e.emp_name FROM emp e
    -> JOIN works w ON e.emp_id=w.emp_id
    -> JOIN company c ON w.company_name=c.company_name
    -> WHERE e.city=c.city;
+----------+
| emp_name |
+----------+
| RAJ      |
| ARJUN    |
| ARUN     |
| ARYA     |
| SHARAN   |
| RAVI     |
+----------+
6 rows in set (0.01 sec)
```

.

Find all employees and their managers in the database who live in the same cities and on the same street number as do their managers.

```
mysql> SELECT e.emp_name AS employee, m.emp_name AS manager
    -> FROM emp e JOIN manages mn ON e.emp_id = mn.emp_id
    -> JOIN emp m ON mn.manager_id = m.emp_id
    -> WHERE e.city = m.city AND e.street_no = m.street_no;
```

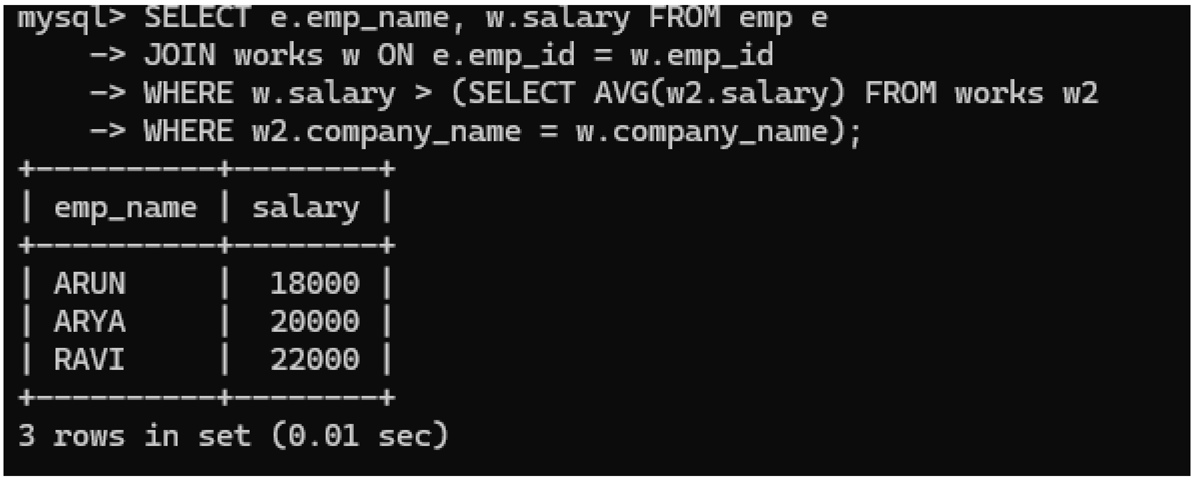


```
mysql> SELECT e.emp_name AS employee, m.emp_name AS manager
    -> FROM emp e JOIN manages mn ON e.emp_id = mn.emp_id
    -> JOIN emp m ON mn.manager_id = m.emp_id
    -> WHERE e.city = m.city AND e.street_no = m.street_no;
+----------+---------+
| employee | manager |
+----------+---------+
| SHARAN   | RAVI    |
+----------+---------+
1 row in set (0.01 sec)
```

.

Find all employees who earn more than the average salary of all employees of their company.

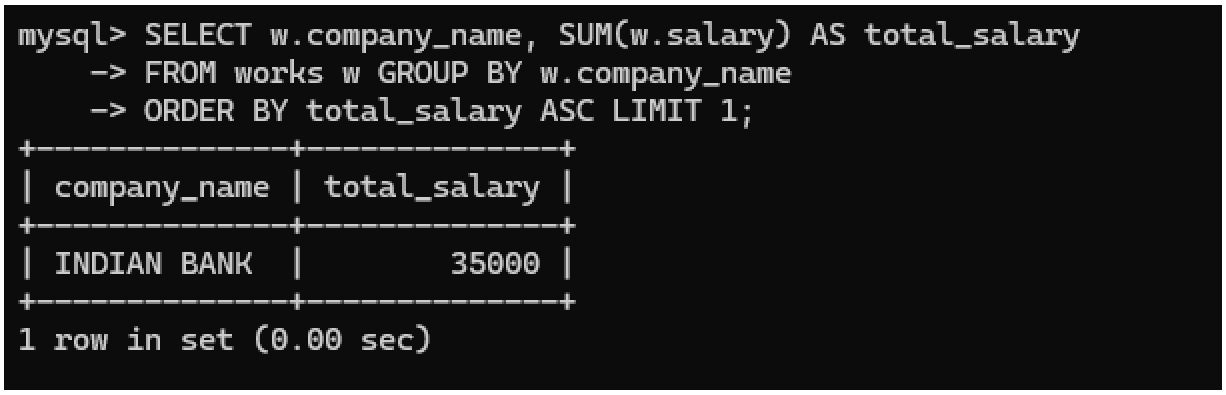
```
mysql> SELECT e.emp_name, w.salary FROM emp e
    -> JOIN works w ON e.emp_id = w.emp_id
    -> WHERE w.salary > (SELECT AVG(w2.salary) FROM works w2
    -> WHERE w2.company_name = w.company_name);
```



```
mysql> SELECT e.emp_name, w.salary FROM emp e
    -> JOIN works w ON e.emp_id = w.emp_id
    -> WHERE w.salary > (SELECT AVG(w2.salary) FROM works w2
    -> WHERE w2.company_name = w.company_name);
+----------+--------+
| emp_name | salary |
+----------+--------+
| ARUN     |  18000 |
| ARYA     |  20000 |
| RAVI     |  22000 |
+----------+--------+
3 rows in set (0.01 sec)
```

· Find the company that pay least total salary along with the salary paid.

```
mysql> SELECT w.company_name, SUM(w.salary) AS total_salary
    -> FROM works w GROUP BY w.company_name
    -> ORDER BY total_salary ASC LIMIT 1;
```



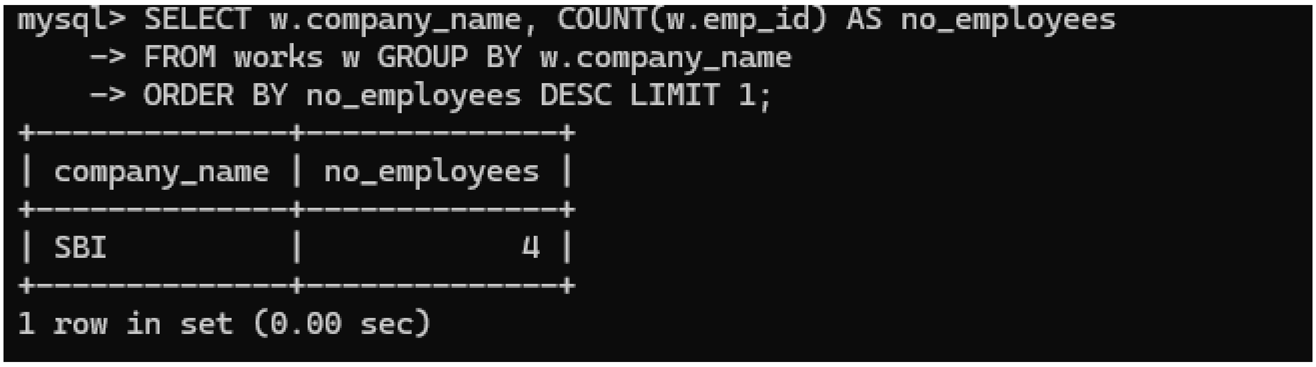
```
mysql> SELECT w.company_name, SUM(w.salary) AS total_salary
    -> FROM works w GROUP BY w.company_name
    -> ORDER BY total_salary ASC LIMIT 1;
+--------------+--------------+
| company_name | total_salary |
+--------------+--------------+
| INDIAN BANK  |        35000 |
+--------------+--------------+
1 row in set (0.00 sec)
```

· Give all managers of SBI a 10 percent raise.

```
mysql> UPDATE works w JOIN
    -> (SELECT m.emp_id FROM manages m JOIN works w ON m.emp_id = w.emp_id
    -> WHERE w.company_name = 'SBI')
    -> AS managers ON w.emp_id = managers.emp_id
    -> SET w.salary = w.salary * 1.10;
Query OK, 4 rows affected (0.01 sec)
Rows matched: 4  Changed: 4  Warnings: 0
```

· Find the company that has the most employees

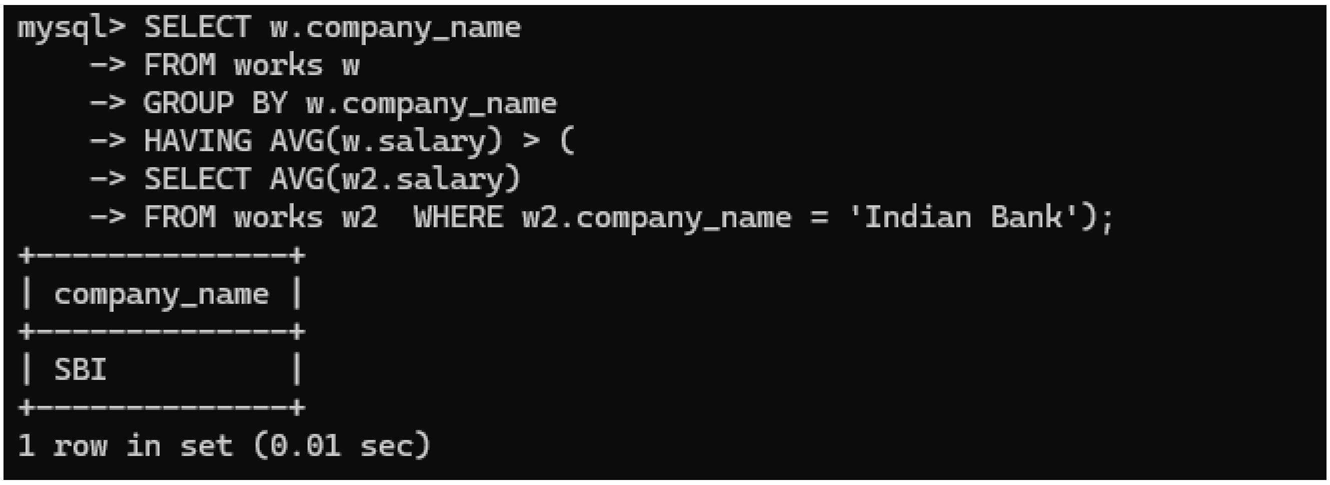
```
mysql> SELECT w.company_name, COUNT(w.emp_id) AS no_employees
    -> FROM works w GROUP BY w.company_name
    -> ORDER BY no_employees DESC LIMIT 1;
```



```
mysql> SELECT w.company_name, COUNT(w.emp_id) AS no_employees
    -> FROM works w GROUP BY w.company_name
    -> ORDER BY no_employees DESC LIMIT 1;
+--------------+--------------+
| company_name | no_employees |
+--------------+--------------+
| SBI          |            4 |
+--------------+--------------+
1 row in set (0.00 sec)
```

· Find those companies whose employees earn a higher salary, on average than the average salary at Indian Bank.

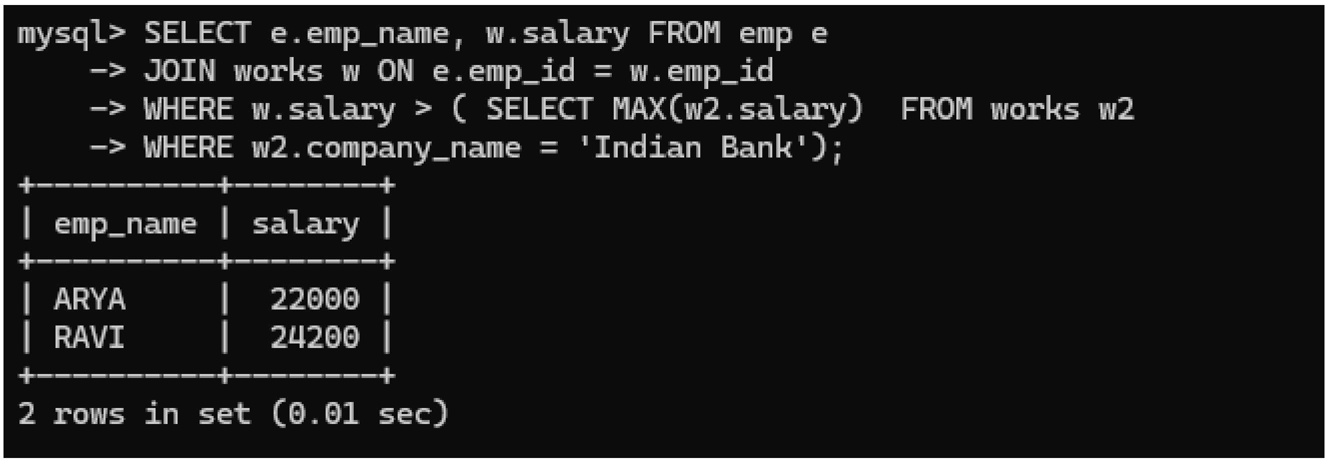
```
mysql> SELECT w.company_name
    -> FROM works w
    -> GROUP BY w.company_name
    -> HAVING AVG(w.salary) > (
    -> SELECT AVG(w2.salary)
    -> FROM works w2  WHERE w2.company_name = 'Indian Bank');
```



```
mysql> SELECT w.company_name
    -> FROM works w
    -> GROUP BY w.company_name
    -> HAVING AVG(w.salary) > (
    -> SELECT AVG(w2.salary)
    -> FROM works w2  WHERE w2.company_name = 'Indian Bank');
+--------------+
| company_name |
+--------------+
| SBI          |
+--------------+
1 row in set (0.01 sec)
```

· Query to find the name and salary of all employees who earn more than each employee of 'Indian Bank'

```
mysql> SELECT e.emp_name, w.salary FROM emp e
    -> JOIN works w ON e.emp_id = w.emp_id
    -> WHERE w.salary > ( SELECT MAX(w2.salary)  FROM works w2
    -> WHERE w2.company_name = 'Indian Bank');
```



```
mysql> SELECT e.emp_name, w.salary FROM emp e
    -> JOIN works w ON e.emp_id = w.emp_id
    -> WHERE w.salary > ( SELECT MAX(w2.salary)  FROM works w2
    -> WHERE w2.company_name = 'Indian Bank');
+----------+--------+
| emp_name | salary |
+----------+--------+
| ARYA     |  22000 |
| RAVI     |  24200 |
+----------+--------+
2 rows in set (0.01 sec)
```

3.BUILDING FUNCTIONS IN RDBMS

AIM

· Implementation of Building functions in RDBMS*

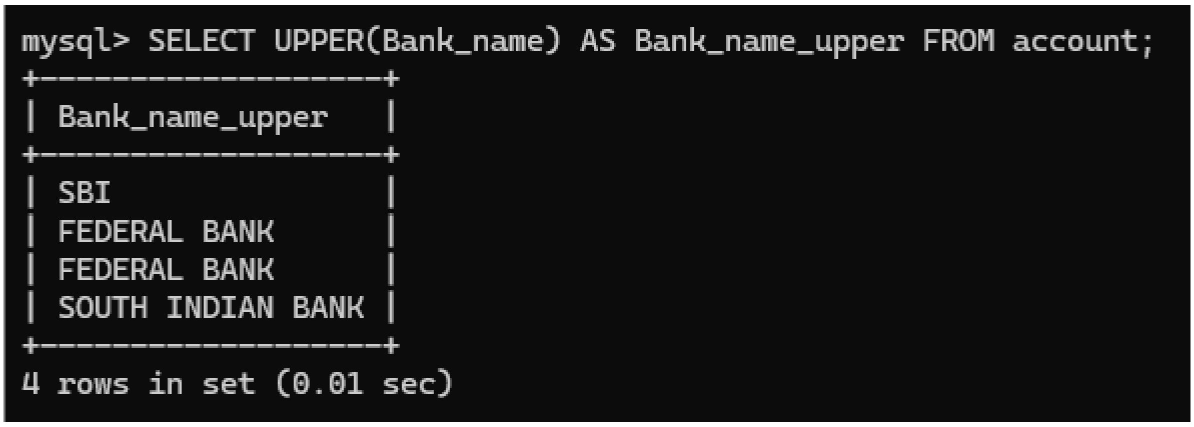
QUERY-OUTPUT

Create database bank.create table account with fields [Account number, IFSC code, Bank name, Bank branch, Account type, Account balance].

Do the following queries:

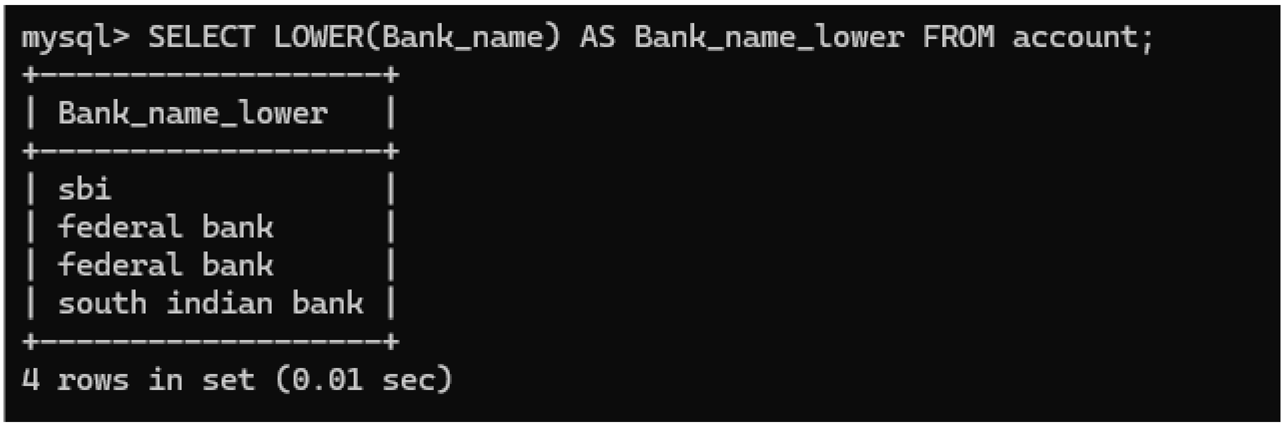
· Implement character string functions like upper,lower,length,replace with value of attribute "Bank-name"

```
UPPER()
 mysql> SELECT UPPER(Bank_name) AS Bank_name_upper FROM account;
```



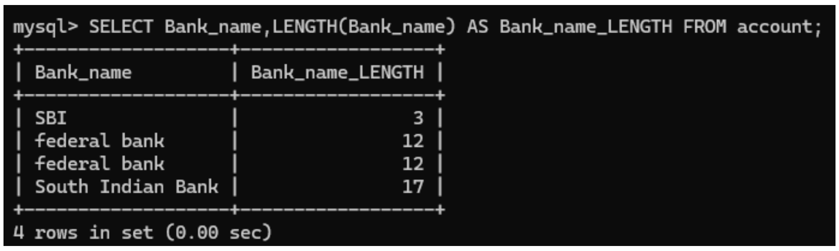
```
mysql> SELECT UPPER(Bank_name) AS Bank_name_upper FROM account;
+-------------------+
| Bank_name_upper   |
+-------------------+
| SBI               |
| FEDERAL BANK      |
| FEDERAL BANK      |
| SOUTH INDIAN BANK |
+-------------------+
4 rows in set (0.01 sec)
```

```
LOWER()
 mysql> SELECT LOWER(Bank_name) AS Bank_name_lower FROM account;
```



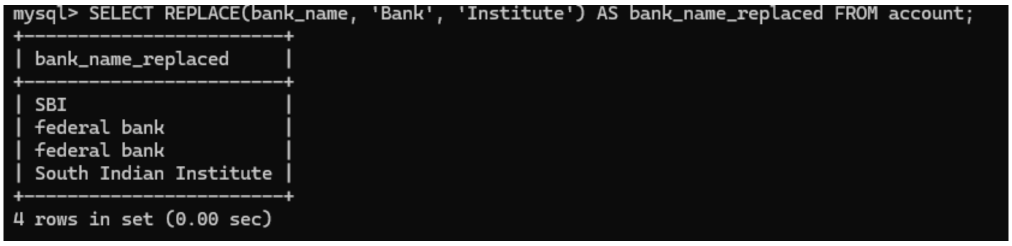
```
mysql> SELECT LOWER(Bank_name) AS Bank_name_lower FROM account;
+-------------------+
| Bank_name_lower   |
+-------------------+
| sbi               |
| federal bank      |
| federal bank      |
| south indian bank |
+-------------------+
4 rows in set (0.01 sec)
```

```
LENGTH()
mysql> SELECT Bank_name,LENGTH(Bank_name) AS Bank_name_LENGTH
FROM account;
```



```
mysql> SELECT Bank_name,LENGTH(Bank_name) AS Bank_name_LENGTH FROM account;
+-------------------+------------------+
| Bank_name         | Bank_name_LENGTH |
+-------------------+------------------+
| SBI               |                3 |
| federal bank      |               12 |
| federal bank      |               12 |
| South Indian Bank |               17 |
+-------------------+------------------+
4 rows in set (0.00 sec)
```

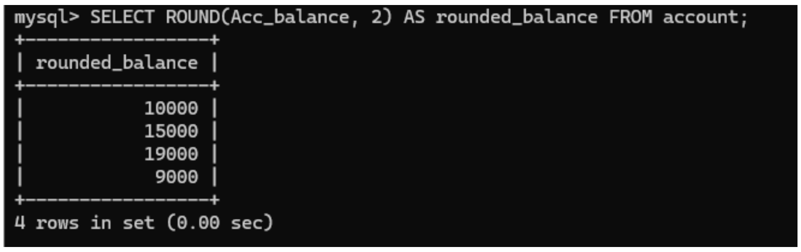
```
REPLACE()
mysql> SELECT REPLACE(bank_name, 'Bank', 'Institute')
AS bank_name_replaced FROM account;
```



```
mysql> SELECT REPLACE(bank_name, 'Bank', 'Institute') AS bank_name_replaced FROM account;
+------------------------+
| bank_name_replaced     |
+------------------------+
| SBI                    |
| federal bank           |
| federal bank           |
| South Indian Institute |
+------------------------+
4 rows in set (0.00 sec)
```

· Implement numeric functions like round,ceil,floor,sign with values of attribute "Account number" or "Account balance"

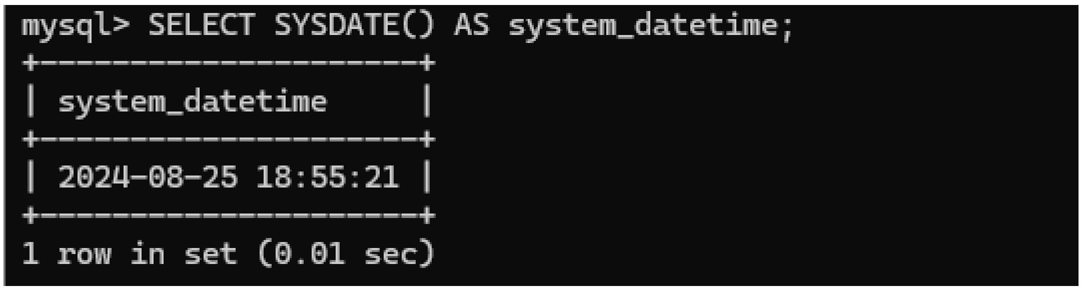
```
mysql> SELECT ROUND(Acc_balance, 2) AS rounded_balance FROM account;
```



```
mysql> SELECT ROUND(Acc_balance, 2) AS rounded_balance FROM account;
+-----------------+
| rounded_balance |
+-----------------+
|           10000 |
|           15000 |
|           19000 |
|            9000 |
+-----------------+
4 rows in set (0.00 sec)
```

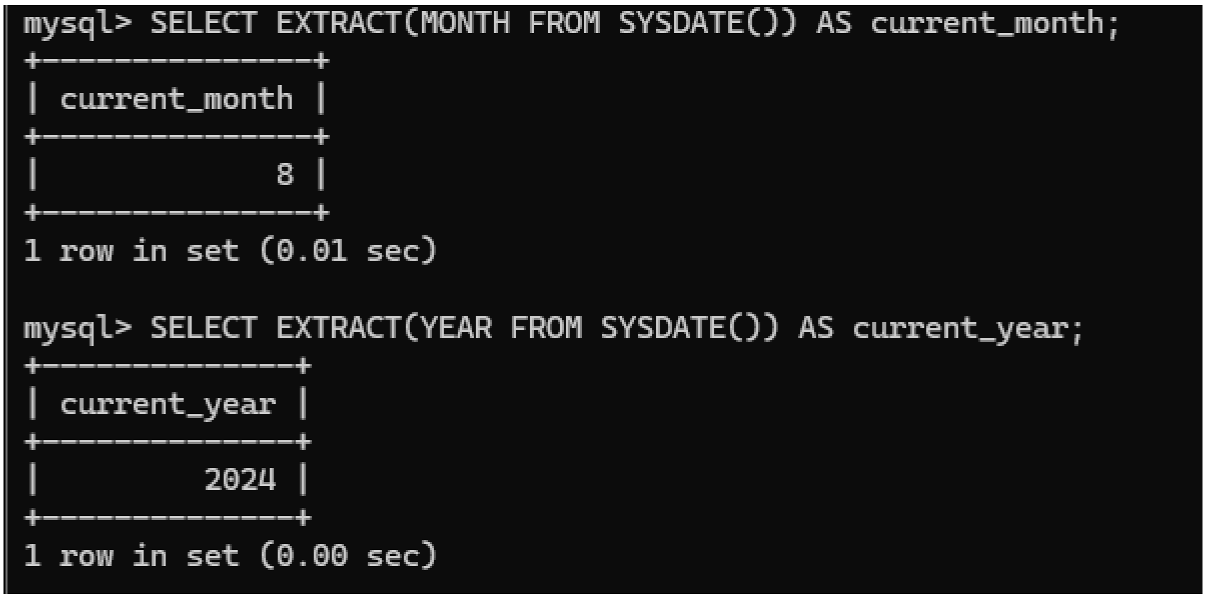
· Implement data functions like current date and sysdate, extract month from date, extract year from date.

```
mysql> SELECT SYSDATE() AS system_datetime;
```



```
mysql> SELECT SYSDATE() AS system_datetime;
+---------------------+
| system_datetime     |
+---------------------+
| 2024-08-25 18:55:21 |
+---------------------+
1 row in set (0.01 sec)
```

```
mysql> SELECT EXTRACT(MONTH FROM SYSDATE()) AS current_month;
 mysql> SELECT EXTRACT(YEAR FROM SYSDATE()) AS current_year;
```



```
mysql> SELECT EXTRACT(MONTH FROM SYSDATE()) AS current_month;
+---------------+
| current_month |
+---------------+
|             8 |
+---------------+
1 row in set (0.01 sec)

mysql> SELECT EXTRACT(YEAR FROM SYSDATE()) AS current_year;
+--------------+
| current_year |
+--------------+
|         2024 |
+--------------+
1 row in set (0.00 sec)
```

· Implement string function ”ascii” to any english alphabets. compute the ascii value of minimum of 5 alphabets.

```
 mysql> SELECT
        ASCII('A') AS ASCII_A,
        ASCII('B') AS ASCII_B,
        ASCII('C') AS ASCII_C,
        ASCII('D') AS ASCII_D,
        ASCII('E') AS ASCII_E;
+-------------+-------------+-------------+-------------+------------+
|  ASCII_A    |    ASCII_B  |  ASCII_C    |    ASCII_D  |  ASCII_E   |
+-------------+-------------+-------------+-------------+------------+
|     65      |    66       |    67       |    68       |    69      |
+-------------+-------------+-------------+-------------+------------+
2 rows in set (0.00 sec)
```

4.AGGREGTE FUNCTIONS

AIM

Implementation of various aggregate functions in SQL

create database college

create the tables with following fields

Faculty(Faculty code,Faculty name)

subject(subject code, subjectname, maxmarks, faculty code)

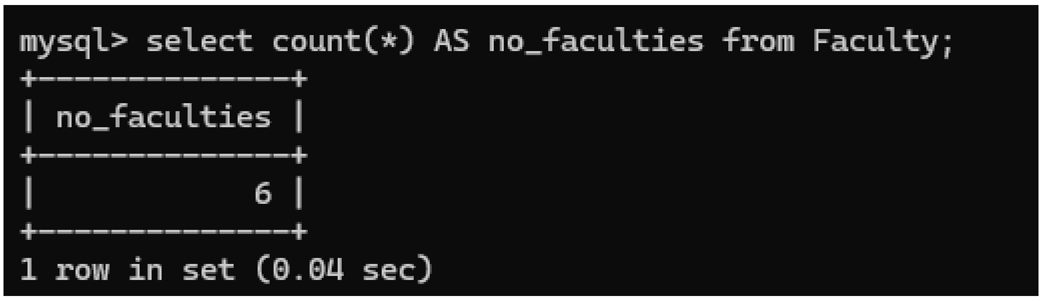
student(student code, studentname, DOB, studentbranch, admission date)

QUERY-OUTPUT

student(student code, studentname, DOB, studentbranch, admission date)

· Display the number of faculties

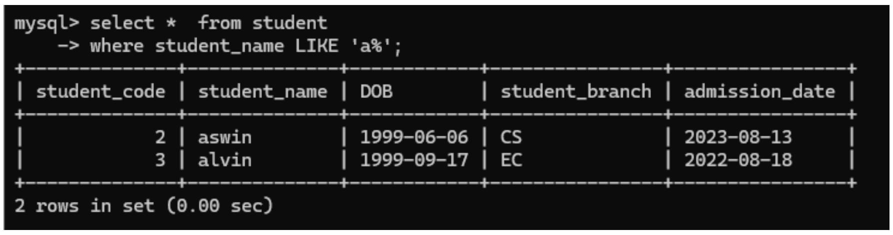
```
mysql> select count(*) AS no_faculties from Faculty;
```



```
mysql> select count(*) AS no_faculties from Faculty;
+--------------+
| no_faculties |
+--------------+
|            6 |
+--------------+
1 row in set (0.04 sec)
```

· Display the details of students with name starting with A.

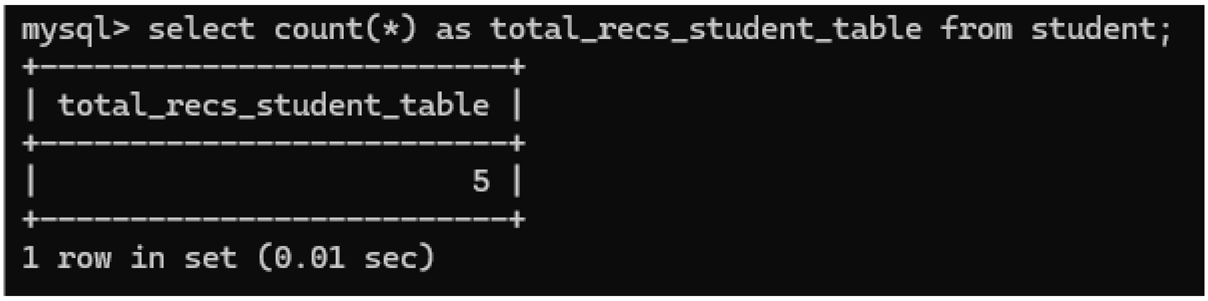
```
mysql> select *  from student
    -> where student_name LIKE 'a%';
```



```
mysql> select *  from student
    -> where student_name LIKE 'a%';
+--------------+--------------+------------+----------------+----------------+
| student_code | student_name | DOB        | student_branch | admission_date |
+--------------+--------------+------------+----------------+----------------+
|            2 | aswin        | 1999-06-06 | CS             | 2023-08-13     |
|            3 | alvin        | 1999-09-17 | EC             | 2022-08-18     |
+--------------+--------------+------------+----------------+----------------+
2 rows in set (0.00 sec)
```

· Display the total number of records in the student table.

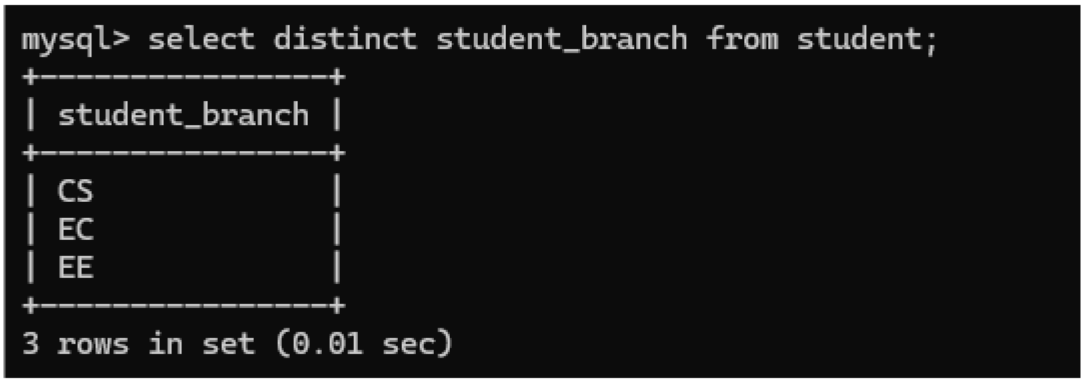
```
mysql> select count(*) as total_recs_student_table from student;
```



```
mysql> select count(*) as total_recs_student_table from student;
+--------------------------+
| total_recs_student_table |
+--------------------------+
|                        5 |
+--------------------------+
1 row in set (0.01 sec)
```

· Find the number of branches available in student table.

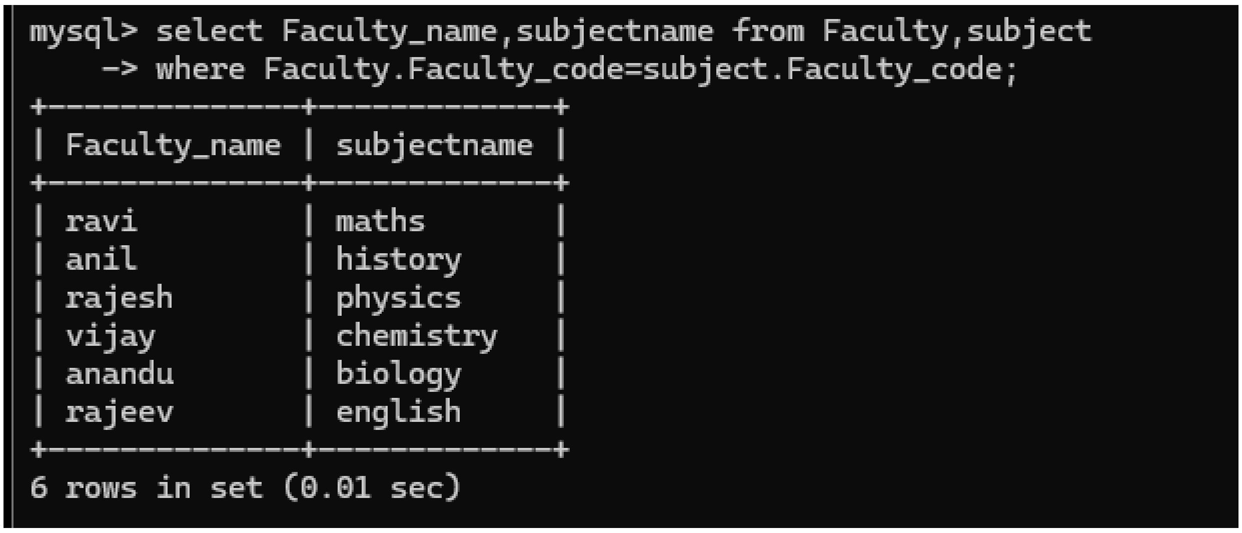
```
mysql> select distinct student_branch from student;
```



```
mysql> select distinct student_branch from student;
+----------------+
| student_branch |
+----------------+
| CS             |
| EC             |
| EE             |
+----------------+
3 rows in set (0.01 sec)
```

· Display the faculties and alloted subjects for each faculty.

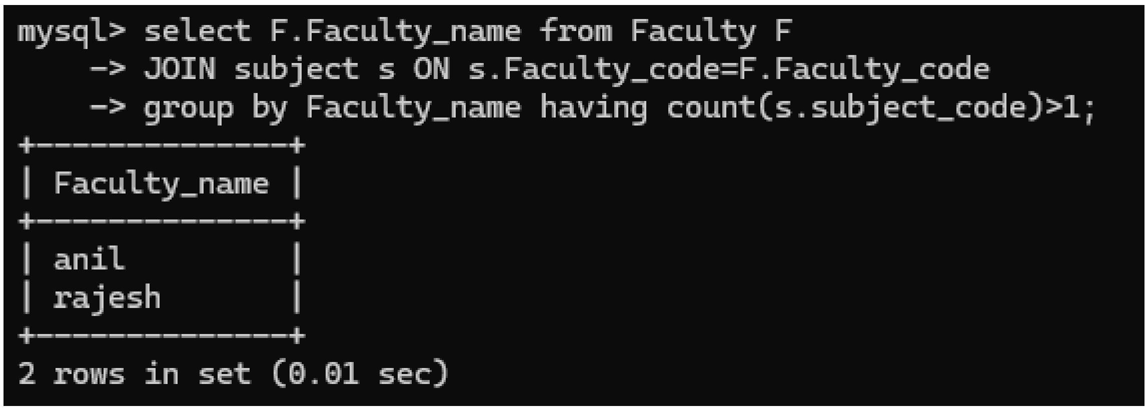
```
mysql> select Faculty_name,subjectname from Faculty,subject
    -> where Faculty.Faculty_code=subject.Faculty_code;
```



```
mysql> select Faculty_name,subjectname from Faculty,subject
    -> where Faculty.Faculty_code=subject.Faculty_code;
+--------------+-------------+
| Faculty_name | subjectname |
+--------------+-------------+
| ravi         | maths       |
| anil         | history     |
| rajesh       | physics     |
| vijay        | chemistry   |
| anandu       | biology     |
| rajeev       | english     |
+--------------+-------------+
6 rows in set (0.01 sec)
```

· Display the names of faculties who take more than one subject.

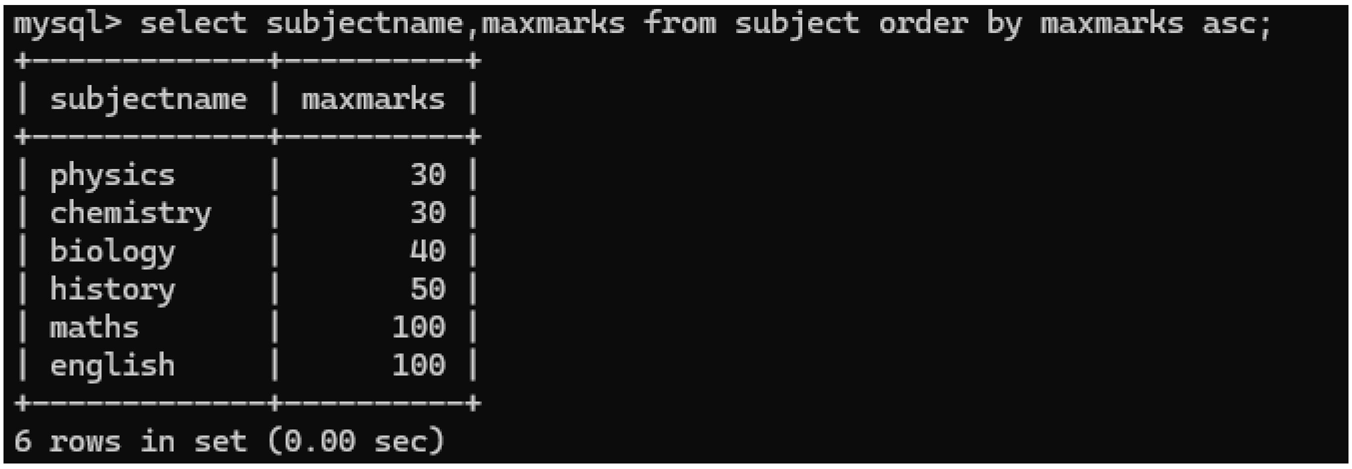
```
mysql> select F.Faculty_name from Faculty F
    -> JOIN subject s ON s.Faculty_code=F.Faculty_code
    -> group by Faculty_name having count(s.subject_code)>1;
```



```
mysql> select F.Faculty_name from Faculty F
    -> JOIN subject s ON s.Faculty_code=F.Faculty_code
    -> group by Faculty_name having count(s.subject_code)>1;
+--------------+
| Faculty_name |
+--------------+
| anil         |
| rajesh       |
+--------------+
2 rows in set (0.01 sec)
```

· Display the subject name and marks in ascending order of marks.

```
mysql> select subjectname,maxmarks from subject order by maxmarks asc;
```



```
mysql> select subjectname,maxmarks from subject order by maxmarks asc;
+-------------+----------+
| subjectname | maxmarks |
+-------------+----------+
| physics     |       30 |
| chemistry   |       30 |
| biology     |       40 |
| history     |       50 |
| maths       |      100 |
| english     |      100 |
+-------------+----------+
6 rows in set (0.00 sec)
```

5.Order By,Group By & Having clauses

AIM

Implementation of Order By,Group By & Having clause & implementation of constraints

QUERY-OUTPUT

Create two tables

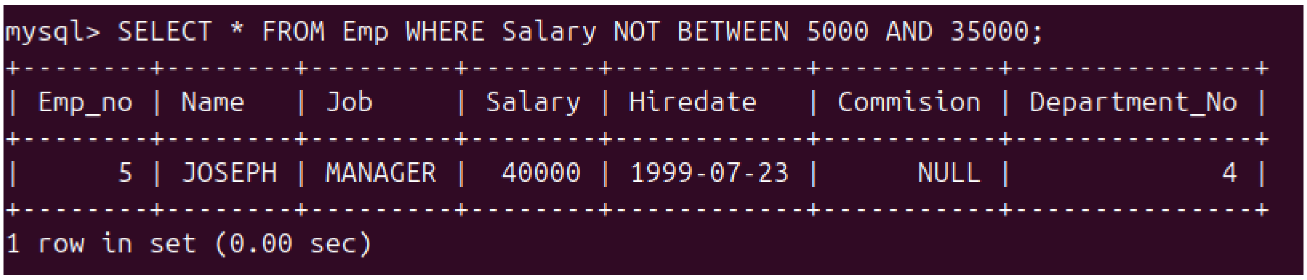
Depart(Department ID,Departmentname,managerid,loc)

Emp(emp-no,name,job,salary,hiredate,commission,Department no.)

manager id is the emp-no of the employee whom the emplyee reports to. Department no. is a foreign key. Insert values into the tables.

· Display the name and salary for all employee whose salary is not in the range of 5000 and 35000;

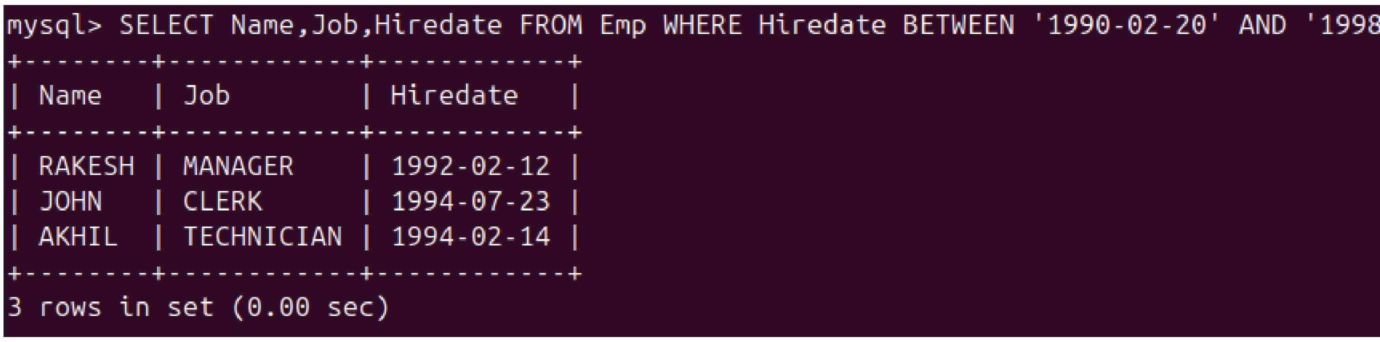
```
mysql> SELECT * FROM Emp WHERE Salary NOT BETWEEN 5000 AND 35000;
```



```
mysql> SELECT * FROM Emp WHERE Salary NOT BETWEEN 5000 AND 35000;
+--------+--------+---------+--------+------------+-----------+---------------+
| Emp_no | Name   | Job     | Salary | Hiredate   | Commision | Department_No |
+--------+--------+---------+--------+------------+-----------+---------------+
|      5 | JOSEPH | MANAGER |  40000 | 1999-07-23 |      NULL |             4 |
+--------+--------+---------+--------+------------+-----------+---------------+
1 row in set (0.00 sec)
```

· Display the employee name, job ID and start date of employees hired between february 20,1990 and May 1,1998. Order the query in ascending order by start date

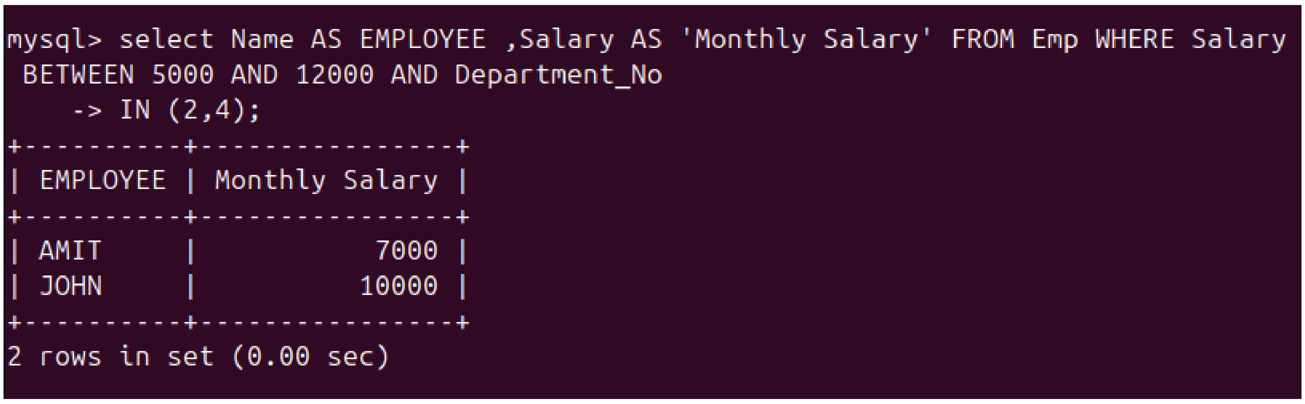
```
mysql> SELECT Name,Job,Hiredate FROM Emp WHERE Hiredate
    ->BETWEEN ’1990-02-20’ AND ’1998-05-01’;
```



```
mysql> SELECT Name,Job,Hiredate FROM Emp WHERE Hiredate BETWEEN '1990-02-20' AND '1998
+--------+------------+------------+
| Name   | Job        | Hiredate   |
+--------+------------+------------+
| RAKESH | MANAGER    | 1992-02-12 |
| JOHN   | CLERK      | 1994-07-23 |
| AKHIL  | TECHNICIAN | 1994-02-14 |
+--------+------------+------------+
3 rows in set (0.00 sec)
```

· List the name and salary of employees who earn between 5000 and 12000 and are in department 2 0r 4. Label the columns Employee and Monthly Salary respectively.

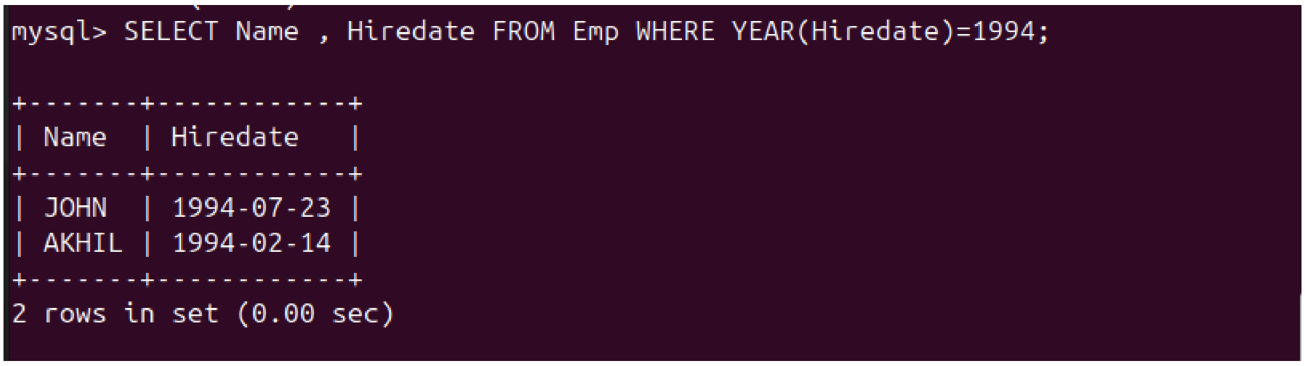
```
mysql> SELECT Name AS EMPLOYEE ,Salary AS ’Monthly Salary’ FROM Emp WHERE
    ->Salary BETWEEN 5000 AND 12000 AND Department_No IN (2,4);
```



```
mysql> select Name AS EMPLOYEE ,Salary AS 'Monthly Salary' FROM Emp WHERE Salary
 BETWEEN 5000 AND 12000 AND Department_No
    -> IN (2,4);
+----------+----------------+
| EMPLOYEE | Monthly Salary |
+----------+----------------+
| AMIT     |           7000 |
| JOHN     |          10000 |
+----------+----------------+
2 rows in set (0.00 sec)
```

· Display names and hire date of every employee who was hired in 1994.

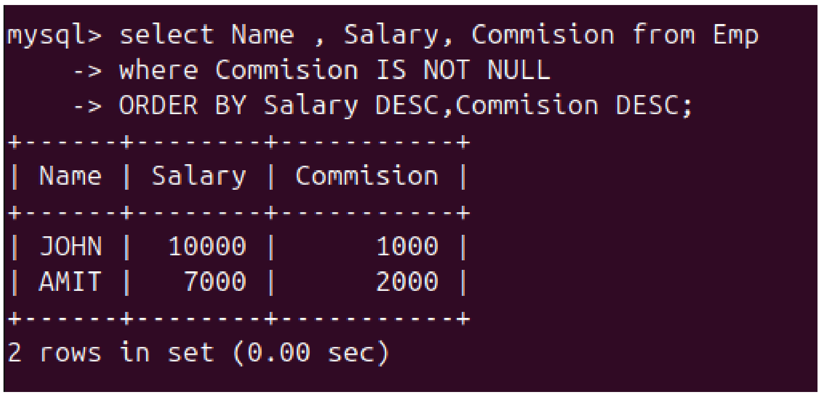
```
mysql> SELECT Name , Hiredate FROM Emp WHERE YEAR(Hiredate)=1994;
```



```
mysql> SELECT Name , Hiredate FROM Emp WHERE YEAR(Hiredate)=1994;

+-------+------------+
| Name  | Hiredate   |
+-------+------------+
| JOHN  | 1994-07-23 |
| AKHIL | 1994-02-14 |
+-------+------------+
2 rows in set (0.00 sec)
```

· Display the name,salary and commission for all employees who earn commissions. Sort data in descending order of salary and commissions.

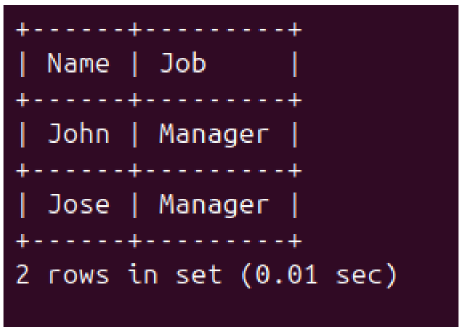
```
mysql> select Name , Salary, Commision from Emp
    -> where Commision IS NOT NULL
    -> ORDER BY Salary DESC,Commision DESC;
```



```
mysql> select Name , Salary, Commision from Emp
    -> where Commision IS NOT NULL
    -> ORDER BY Salary DESC,Commision DESC;
+------+--------+-----------+
| Name | Salary | Commision |
+------+--------+-----------+
| JOHN |  10000 |      1000 |
| AMIT |   7000 |      2000 |
+------+--------+-----------+
2 rows in set (0.00 sec)
```

· Display the name and job title of all employees who do not have a manager.

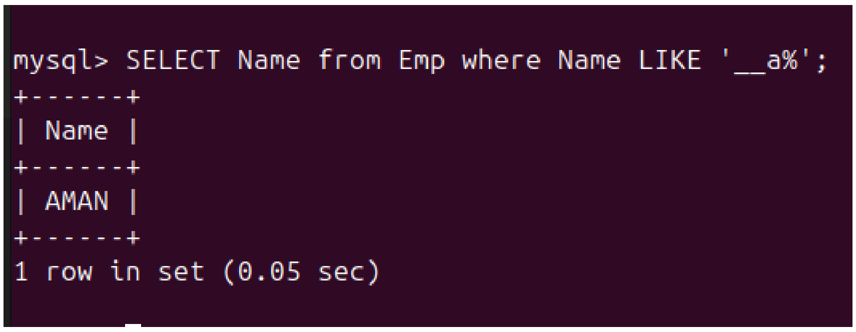
```
mysql>SELECT Name, Job
    ->FROM Emp
    ->WHERE ManagerID IS NULL;
```



```
+------+---------+
| Name | Job     |
+------+---------+
| John | Manager |
+------+---------+
| Jose | Manager |
+------+---------+
2 rows in set (0.01 sec)
```

· Display the name of all employee where the third letter of the name is an ‘a’.

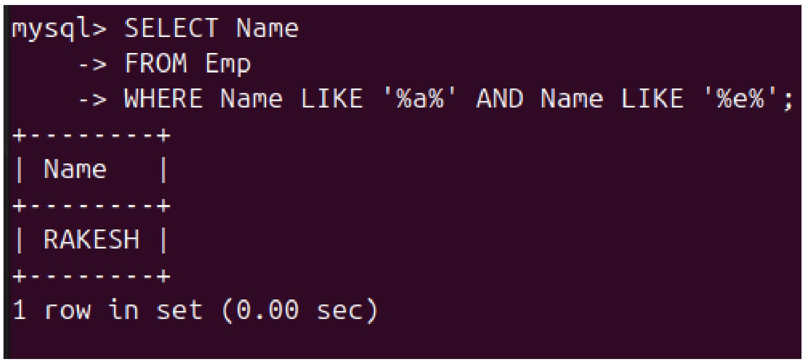
```
 mysql> SELECT Name from Emp where Name LIKE '__a%';
```



```
mysql> SELECT Name from Emp where Name LIKE '__a%';
+------+
| Name |
+------+
| AMAN |
+------+
1 row in set (0.05 sec)
```

· Display the name of all employees who have an a and an e in their name.

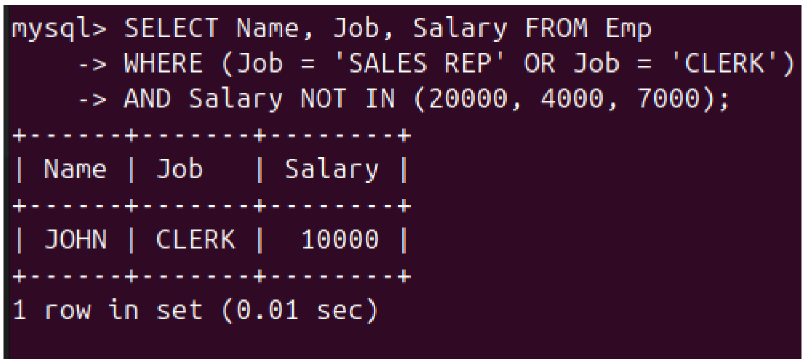
```
mysql> SELECT Name
    -> FROM Emp
    -> WHERE Name LIKE '%a%' AND Name LIKE '%e%';
```



```
mysql> SELECT Name
    -> FROM Emp
    -> WHERE Name LIKE '%a%' AND Name LIKE '%e%';
+--------+
| Name   |
+--------+
| RAKESH |
+--------+
1 row in set (0.00 sec)
```

· Display the name, job, and salary for all employees whose job is sales representative or stock clerk and whose salary is not equal to 2,0000, 4000, or 7,000.

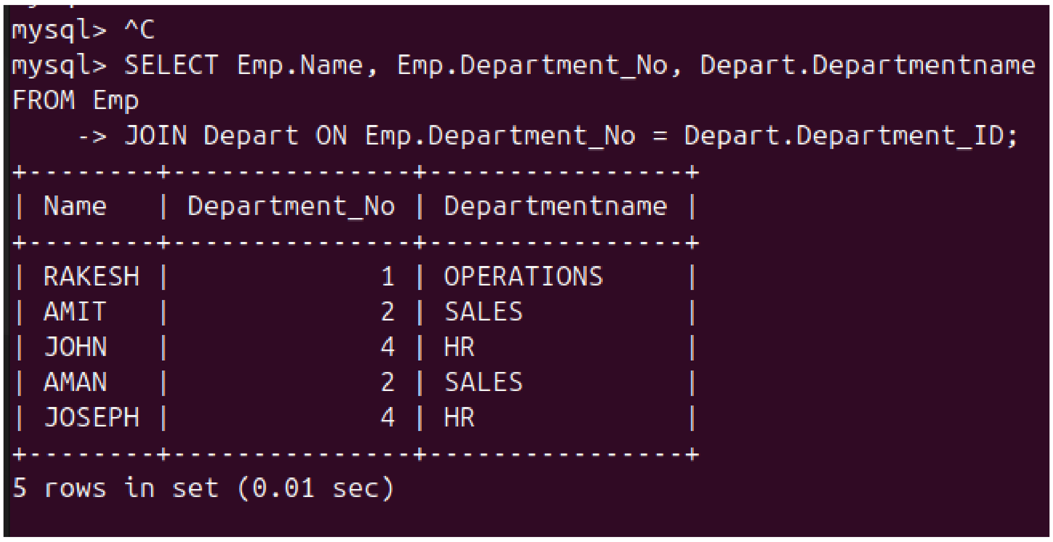
```
mysql> SELECT Name, Job, Salary FROM Emp
    -> WHERE (Job = 'SALES REP' OR Job = 'CLERK')
    -> AND Salary NOT IN (20000, 4000, 7000);
```



```
mysql> SELECT Name, Job, Salary FROM Emp
    -> WHERE (Job = 'SALES REP' OR Job = 'CLERK')
    -> AND Salary NOT IN (20000, 4000, 7000);
+------+-------+--------+
| Name | Job   | Salary |
+------+-------+--------+
| JOHN | CLERK |  10000 |
+------+-------+--------+
1 row in set (0.01 sec)
```

· Write a query to display the name, department number, and department name for all employees.

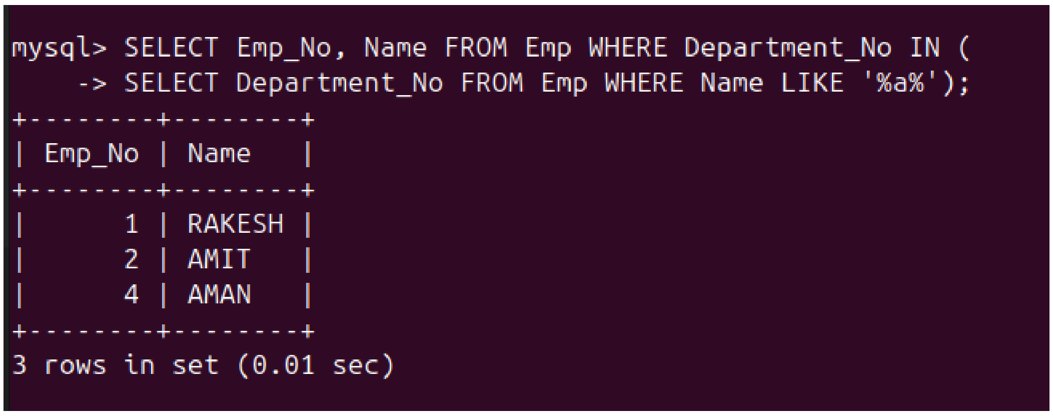
```
mysql> SELECT Emp.Name, Emp.Department_No, Depart.Departmentname FROM Emp
    -> JOIN Depart ON Emp.Department_No = Depart.Department_ID;
```



```
mysql> ^C
mysql> SELECT Emp.Name, Emp.Department_No, Depart.Departmentname
FROM Emp
    -> JOIN Depart ON Emp.Department_No = Depart.Department_ID;
+--------+---------------+----------------+
| Name   | Department_No | Departmentname |
+--------+---------------+----------------+
| RAKESH |             1 | OPERATIONS     |
| AMIT   |             2 | SALES          |
| JOHN   |             4 | HR             |
| AMAN   |             2 | SALES          |
| JOSEPH |             4 | HR             |
+--------+---------------+----------------+
5 rows in set (0.01 sec)
```

· Write a query that displays the employee numbers names of all employees who work in a department with any employee whose name contains a ’a’.

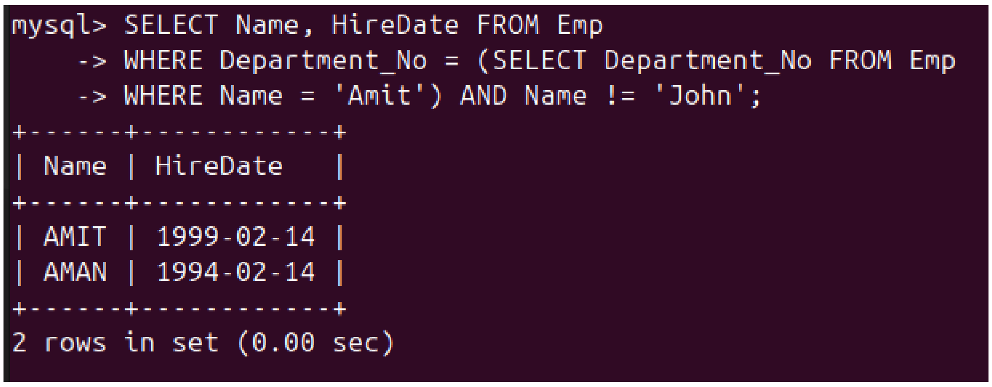
```
mysql> SELECT Emp_No, Name FROM Emp WHERE Department_No IN (
    -> SELECT Department_No FROM Emp WHERE Name LIKE ’%a%’);
```



```
mysql> SELECT Emp_No, Name FROM Emp WHERE Department_No IN (
    -> SELECT Department_No FROM Emp WHERE Name LIKE '%a%');
+--------+--------+
| Emp_No | Name   |
+--------+--------+
|      1 | RAKESH |
|      2 | AMIT   |
|      4 | AMAN   |
+--------+--------+
3 rows in set (0.01 sec)
```

· Write a query to display the name and hire date of any employee in the same department as amit. Exclude JOHN.

```
mysql> SELECT Name, HireDate FROM Emp
    -> WHERE Department_No = (SELECT Department_No FROM Emp
    -> WHERE Name = 'Amit') AND Name != 'John';
```



```
mysql> SELECT Name, HireDate FROM Emp
    -> WHERE Department_No = (SELECT Department_No FROM Emp
    -> WHERE Name = 'Amit') AND Name != 'John';
+------+------------+
| Name | HireDate   |
+------+------------+
| AMIT | 1999-02-14 |
| AMAN | 1994-02-14 |
+------+------------+
2 rows in set (0.00 sec)
```

6.SET OPERATORS & NESTED, JOIN QUERIES

AIM

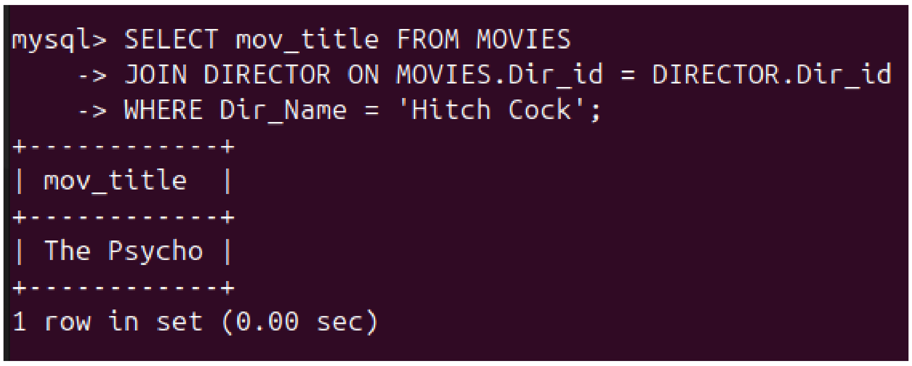
Implementation of set operators,nested queries and Join queries and verify the relationship

QUERY-OUTPUT

Consider the schema for the Movie Database.

ACTOR(Act_id, Act_Name, Act_Geander)
DIRECTOR(Dir_id, Dir_Name, Dir_Phone)
MOVIES(Mov_id, Mov_Title, Mov_Year, Mov_Lang, Dir_id)
MOVIE_CAST(Act_id, Mov_id, Role)
RATING(Mov_id, Rev_Stars)
Write queries to:
· List the title of all movies directed by 'Hitch Cock'.

```
mysql> SELECT mov_title FROM MOVIES
    -> JOIN DIRECTOR ON MOVIES.Dir_id = DIRECTOR.Dir_id
    -> WHERE Dir_Name = 'Hitch Cock';
```



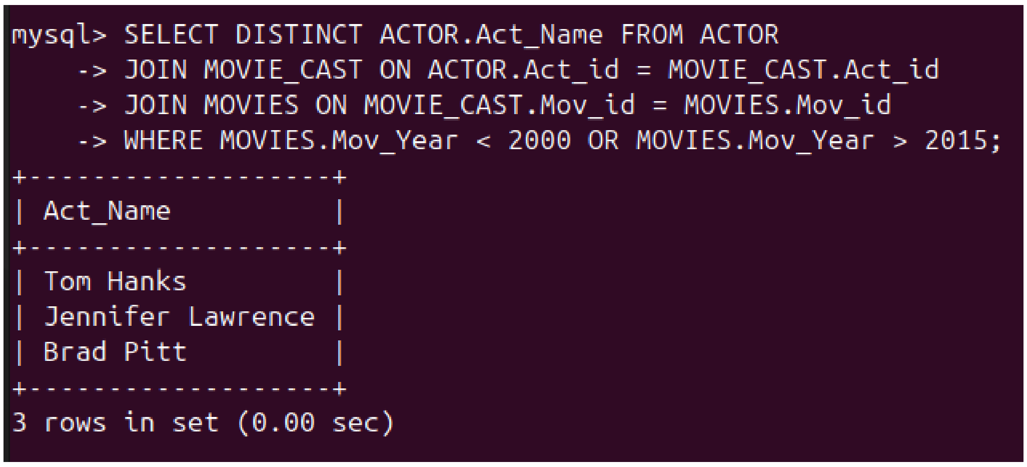
```
mysql> SELECT mov_title FROM MOVIES
    -> JOIN DIRECTOR ON MOVIES.Dir_id = DIRECTOR.Dir_id
    -> WHERE Dir_Name = 'Hitch Cock';
+------------+
| mov_title  |
+------------+
| The Psycho |
+------------+
1 row in set (0.00 sec)
```

· Find the movie names where one or more actors acted in two or more movies.

```
 mysql> SELECT Mov_Title FROM MOVIES
    -> WHERE Mov_id IN (SELECT Mov_id FROM MOVIE_CAST
    -> WHERE Act_id IN (SELECT Act_id FROM MOVIE_CAST
    -> GROUP BY Act_id HAVING COUNT(Mov_id) >= 2));
Empty set (0.01 sec)
```

· List all actors who acted in a movie before 2000 or in a movie after 2015 (use JOIN operation)

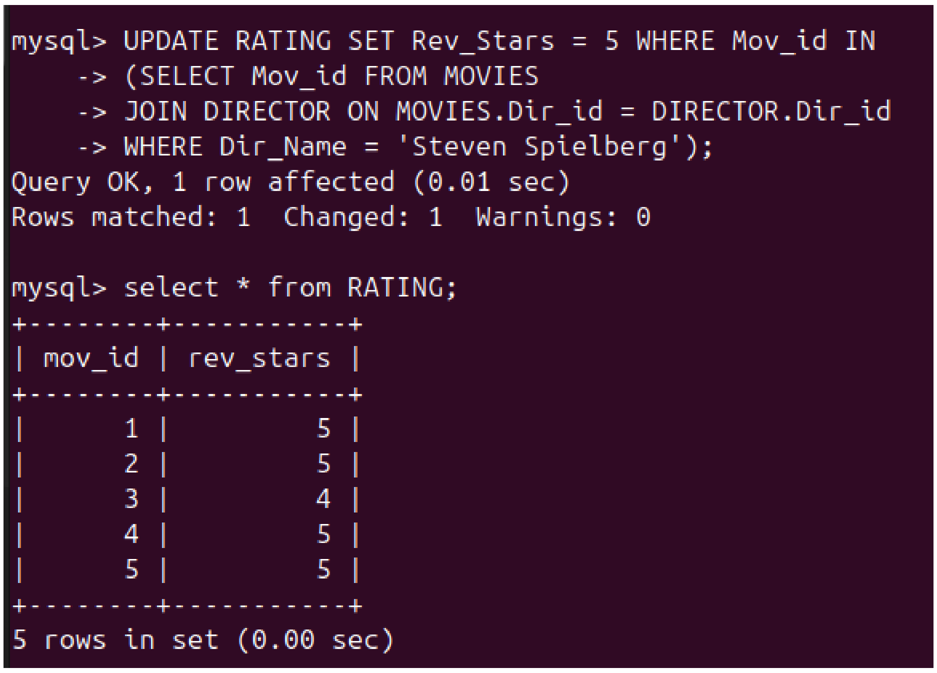
```
mysql> SELECT DISTINCT ACTOR.Act_Name FROM ACTOR
    -> JOIN MOVIE_CAST ON ACTOR.Act_id = MOVIE_CAST.Act_id
    -> JOIN MOVIES ON MOVIE_CAST.Mov_id = MOVIES.Mov_id
    -> WHERE MOVIES.Mov_Year < 2000 OR MOVIES.Mov_Year > 2015;
```



```
mysql> SELECT DISTINCT ACTOR.Act_Name FROM ACTOR
    -> JOIN MOVIE_CAST ON ACTOR.Act_id = MOVIE_CAST.Act_id
    -> JOIN MOVIES ON MOVIE_CAST.Mov_id = MOVIES.Mov_id
    -> WHERE MOVIES.Mov_Year < 2000 OR MOVIES.Mov_Year > 2015;
+-------------------+
| Act_Name          |
+-------------------+
| Tom Hanks         |
| Jennifer Lawrence |
| Brad Pitt         |
+-------------------+
3 rows in set (0.00 sec)
```

· Update rating of all movies directed by ’Steven Spielberg’ to 5.

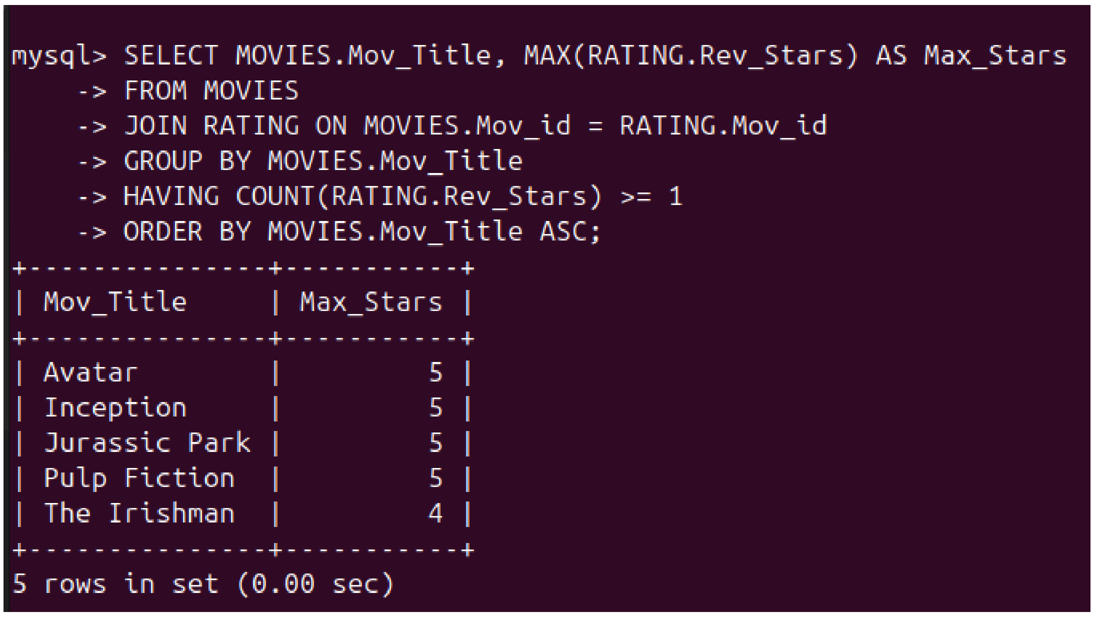
```
mysql> UPDATE RATING SET Rev_Stars = 5 WHERE Mov_id IN
    -> (SELECT Mov_id FROM MOVIES
    -> JOIN DIRECTOR ON MOVIES.Dir_id = DIRECTOR.Dir_id
    -> WHERE Dir_Name = ’Steven Spielberg’);
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```



```
mysql> UPDATE RATING SET Rev_Stars = 5 WHERE Mov_id IN
    -> (SELECT Mov_id FROM MOVIES
    -> JOIN DIRECTOR ON MOVIES.Dir_id = DIRECTOR.Dir_id
    -> WHERE Dir_Name = 'Steven Spielberg');
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from RATING;
+--------+-----------+
| mov_id | rev_stars |
+--------+-----------+
|      1 |         5 |
|      2 |         5 |
|      3 |         4 |
|      4 |         5 |
|      5 |         5 |
+--------+-----------+
5 rows in set (0.00 sec)
```

· Find the title of movies and number of stars for each movie that has atleast one rating and find the highest number of stars that movie received. Sort the result by movie title.

```
mysql> SELECT MOVIES.Mov_Title, MAX(RATING.Rev_Stars) AS Max_Stars
    -> FROM MOVIES
    -> JOIN RATING ON MOVIES.Mov_id = RATING.Mov_id
    -> GROUP BY MOVIES.Mov_Title
    -> HAVING COUNT(RATING.Rev_Stars) >= 1
    -> ORDER BY MOVIES.Mov_Title ASC;
```



```
mysql> SELECT MOVIES.Mov_Title, MAX(RATING.Rev_Stars) AS Max_Stars
    -> FROM MOVIES
    -> JOIN RATING ON MOVIES.Mov_id = RATING.Mov_id
    -> GROUP BY MOVIES.Mov_Title
    -> HAVING COUNT(RATING.Rev_Stars) >= 1
    -> ORDER BY MOVIES.Mov_Title ASC;
+---------------+-----------+
| Mov_Title     | Max_Stars |
+---------------+-----------+
| Avatar        |         5 |
| Inception     |         5 |
| Jurassic Park |         5 |
| Pulp Fiction  |         5 |
| The Irishman  |         4 |
+---------------+-----------+
5 rows in set (0.00 sec)
```

7.VIEWS & ASSERTIONS

AIM

Creation and implementation of Views and Assertions

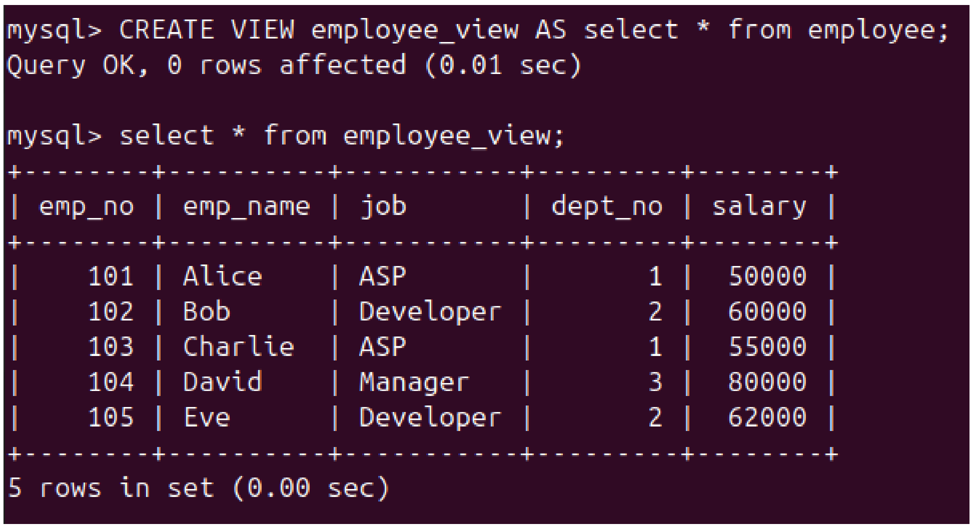
QUERY-OUTPUT

· Create a table employee with attributes employee no, employee name, job, department no, salary. Insert 5 values into it and display.

```
mysql> CREATE TABLE employee (emp_no INT PRIMARY KEY,emp_name VARCHAR(100), job
    ->VARCHAR(50),dept_no INT,salary INT);
Query OK, 0 rows affected (0.03 sec)

mysql> insert into employee values(101, 'Alice', 'ASP', 1, 50000),
    -> (102, 'Bob', 'Developer', 2, 60000),
    -> (103, 'Charlie', 'ASP', 1, 55000),
    -> (104, 'David', 'Manager', 3, 80000),
    -> (105, 'Eve', 'Developer', 2, 62000);
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

· Create a view for the above created table and display it.

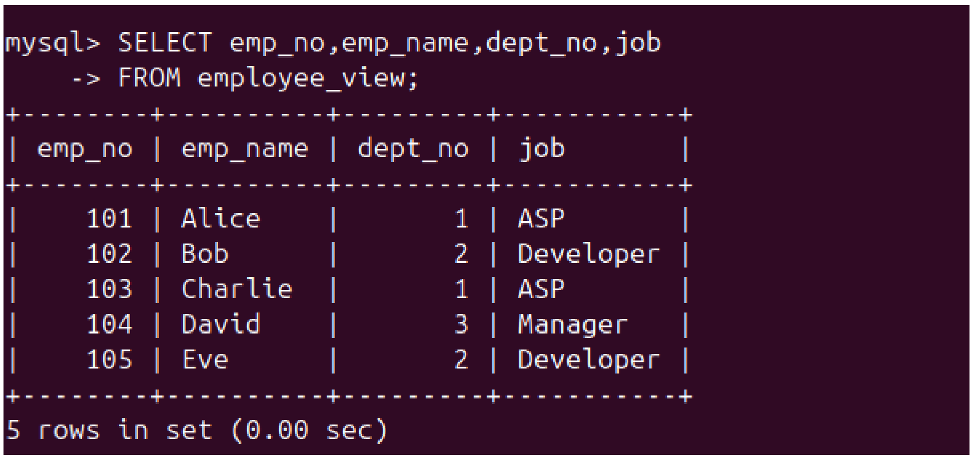
```
mysql> CREATE VIEW employee_view AS select * from employee;
Query OK, 0 rows affected (0.01 sec)

mysql> select * from employee_view;
```



```
mysql> CREATE VIEW employee_view AS select * from employee;
Query OK, 0 rows affected (0.01 sec)

mysql> select * from employee_view;
+--------+----------+-----------+---------+--------+
| emp_no | emp_name | job       | dept_no | salary |
+--------+----------+-----------+---------+--------+
|    101 | Alice    | ASP       |       1 |  50000 |
|    102 | Bob      | Developer |       2 |  60000 |
|    103 | Charlie  | ASP       |       1 |  55000 |
|    104 | David    | Manager   |       3 |  80000 |
|    105 | Eve      | Developer |       2 |  62000 |
+--------+----------+-----------+---------+--------+
5 rows in set (0.00 sec)
```

· Display only the details like employee no, employee name, department no and job of the employee view table (vertical partitioning)

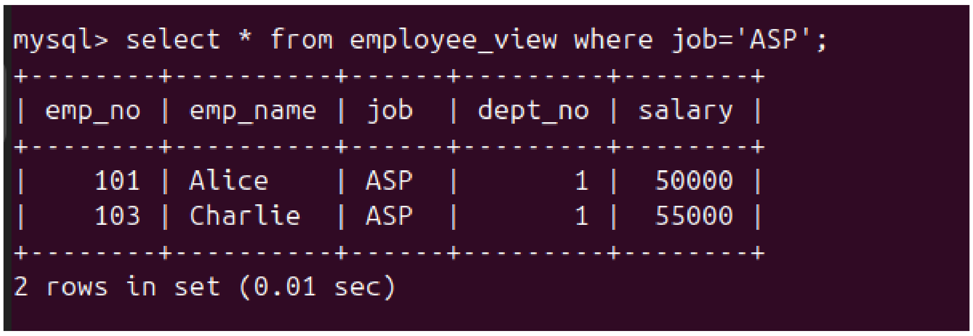
```
mysql> SELECT emp_no,emp_name,dept_no,job
    -> FROM employee_view;
```



```
mysql> SELECT emp_no,emp_name,dept_no,job
    -> FROM employee_view;
+--------+----------+---------+-----------+
| emp_no | emp_name | dept_no | job       |
+--------+----------+---------+-----------+
|    101 | Alice    |       1 | ASP       |
|    102 | Bob      |       2 | Developer |
|    103 | Charlie  |       1 | ASP       |
|    104 | David    |       3 | Manager   |
|    105 | Eve      |       2 | Developer |
+--------+----------+---------+-----------+
5 rows in set (0.00 sec)
```

· Display only the details of Employees those who are working under ASP job title from the view table (horizontal partitioning)

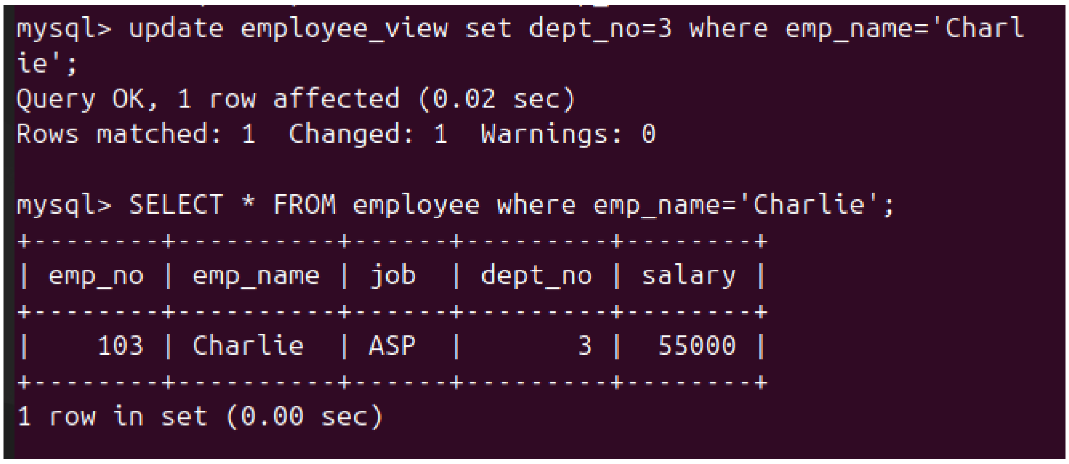
```
mysql> select * from employee_view where job='ASP';
```



```
mysql> select * from employee_view where job='ASP';
+--------+----------+------+---------+--------+
| emp_no | emp_name | job  | dept_no | salary |
+--------+----------+------+---------+--------+
|    101 | Alice    | ASP  |       1 |  50000 |
|    103 | Charlie  | ASP  |       1 |  55000 |
+--------+----------+------+---------+--------+
2 rows in set (0.01 sec)
```

· Update the view table, change the department name of a specified employee and check whether the changed are displayed on the original table.

```
mysql> update employee_view set dept_no=3 where emp_name='Charlie';
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```



```
mysql> update employee_view set dept_no=3 where emp_name='Charl
ie';
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM employee where emp_name='Charlie';
+--------+----------+------+---------+--------+
| emp_no | emp_name | job  | dept_no | salary |
+--------+----------+------+---------+--------+
|    103 | Charlie  | ASP  |       3 |  55000 |
+--------+----------+------+---------+--------+
1 row in set (0.00 sec)
```

· Drop a view.

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

8.TCL COMMANDS

AIM

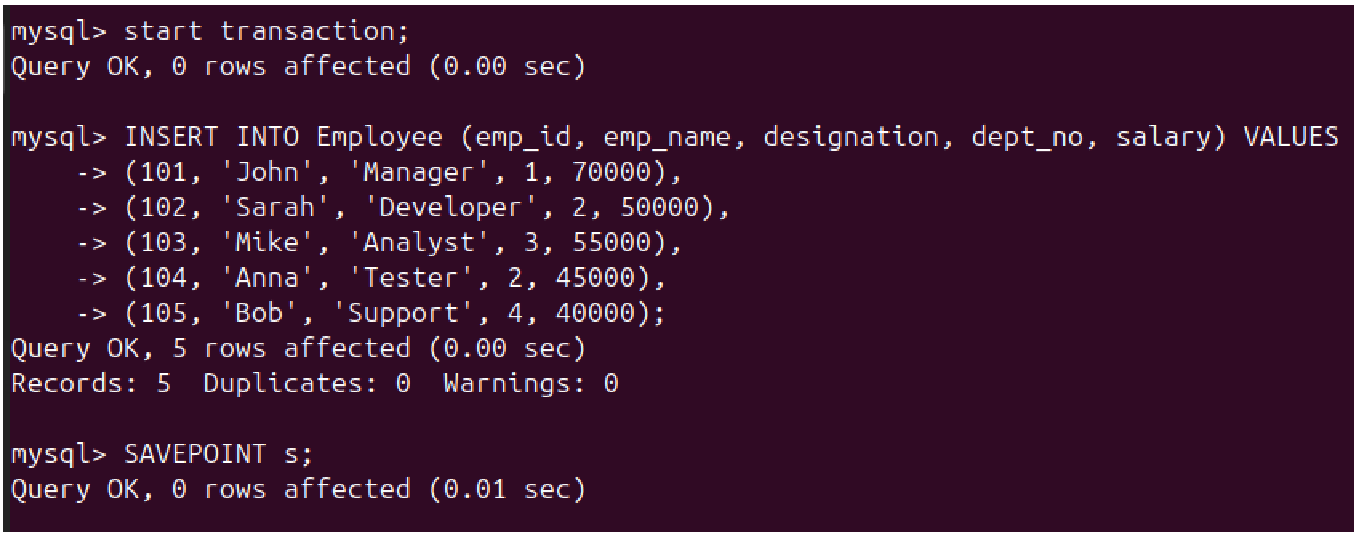
Implementation of TCL Commands

QUERY-OUTPUT

Implementation of TCL Commands* Consider the following table namely ”Employee” with schema as follows. Attributes of the table - Employee id, employee name, designation, department-no, employee salary.

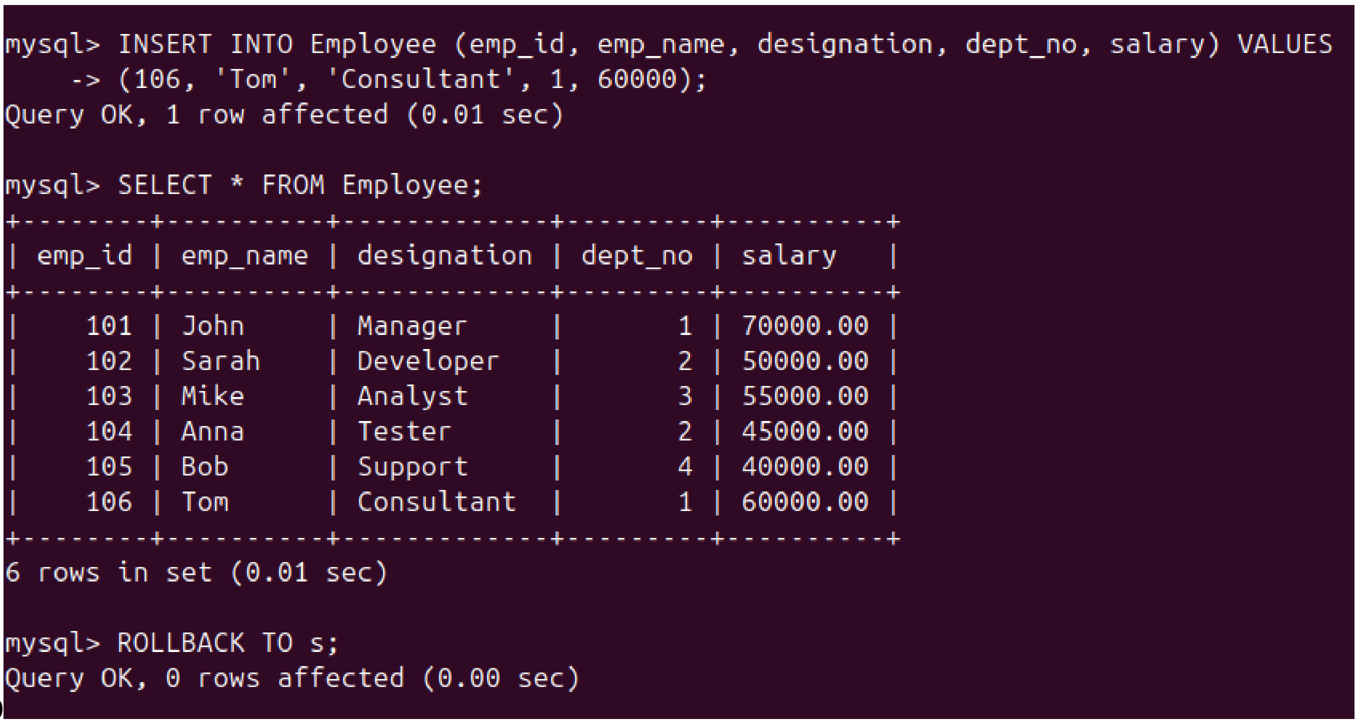
• Insert 5 values into database and create a savepointnamed ’s’. In the existing database add 1 row extra and then display the results on execution of rollback and commit commands.

```
mysql> INSERT INTO Employee (emp_id, emp_name, designation, dept_no, salary) VALUES
    -> (101, ’John’, ’Manager’, 1, 70000),
    -> (102, ’Sarah’, ’Developer’, 2, 50000),
    -> (103, ’Mike’, ’Analyst’, 3, 55000),
    -> (104, ’Anna’, ’Tester’, 2, 45000),
    -> (105, ’Bob’, ’Support’, 4, 40000);
Query OK, 5 rows affected (0.00 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> SAVEPOINT s;
Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO Employee (emp_id, emp_name, designation, dept_no, salary) VALUES
    -> (106, ’Tom’, ’Consultant’, 1, 60000);
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM Employee;
```



```
mysql> start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO Employee (emp_id, emp_name, designation, dept_no, salary) VALUES
    -> (101, 'John', 'Manager', 1, 70000),
    -> (102, 'Sarah', 'Developer', 2, 50000),
    -> (103, 'Mike', 'Analyst', 3, 55000),
    -> (104, 'Anna', 'Tester', 2, 45000),
    -> (105, 'Bob', 'Support', 4, 40000);
Query OK, 5 rows affected (0.00 sec)
Records: 5  Duplicates: 0  Warnings: 0

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

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

mysql> SELECT * FROM Employee;
```



```
mysql> INSERT INTO Employee (emp_id, emp_name, designation, dept_no, salary) VALUES
    -> (106, 'Tom', 'Consultant', 1, 60000);
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM Employee;
+--------+----------+-------------+---------+----------+
| emp_id | emp_name | designation | dept_no | salary   |
+--------+----------+-------------+---------+----------+
|    101 | John     | Manager     |       1 | 70000.00 |
|    102 | Sarah    | Developer   |       2 | 50000.00 |
|    103 | Mike     | Analyst     |       3 | 55000.00 |
|    104 | Anna     | Tester      |       2 | 45000.00 |
|    105 | Bob      | Support     |       4 | 40000.00 |
|    106 | Tom      | Consultant  |       1 | 60000.00 |
+--------+----------+-------------+---------+----------+
6 rows in set (0.01 sec)

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

9

```
mysql> COMMIT;
Query OK, 0 rows affected (0.02 sec)
```

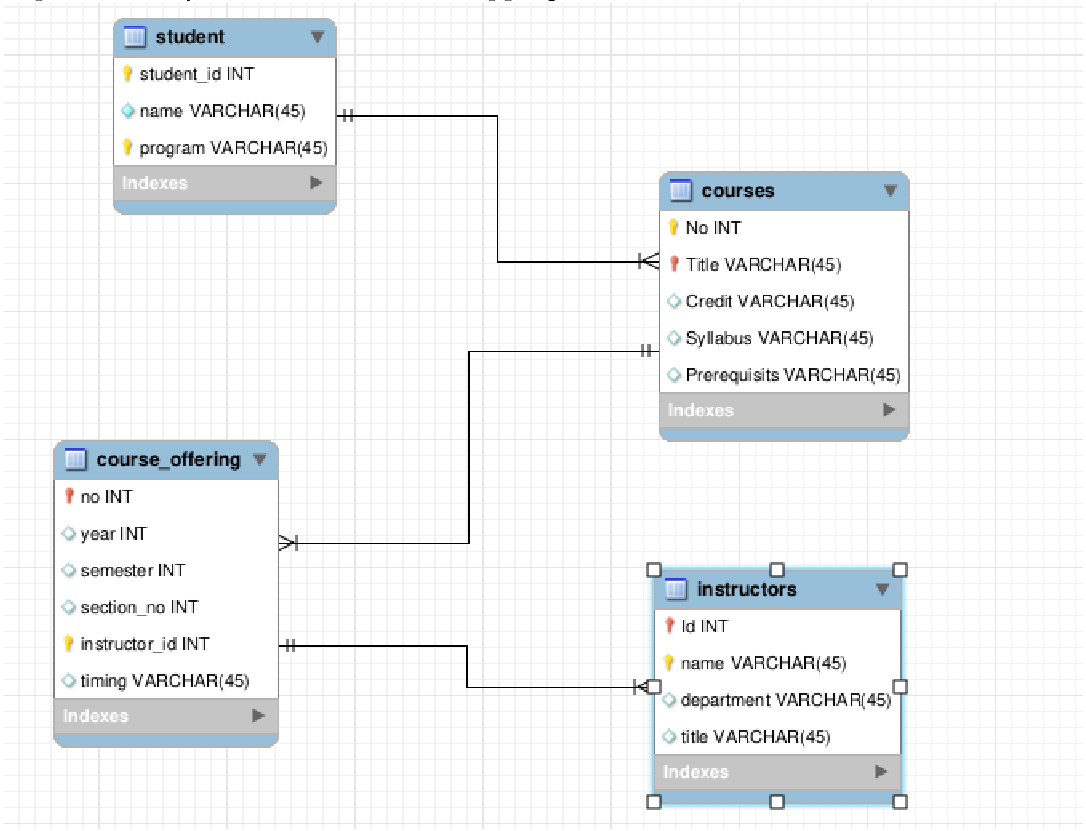
9. ER DIAGRAM & SCHEMA

AIM

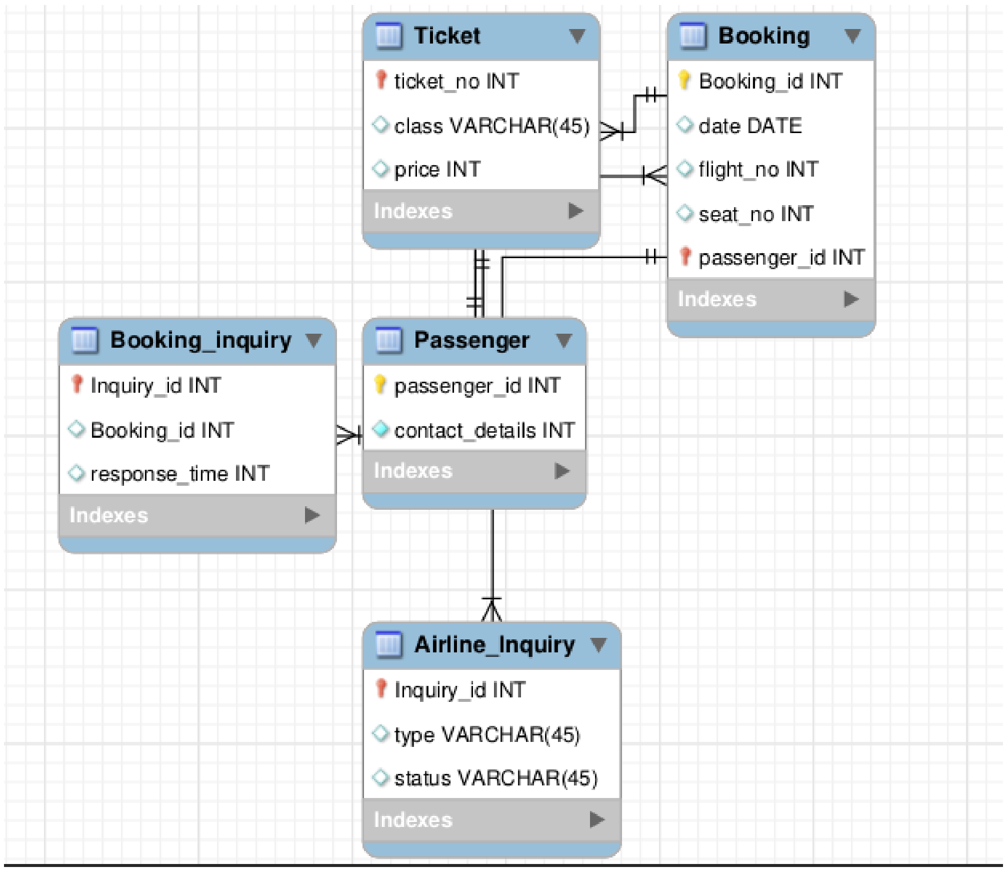
Implementation of ER Diagram and export diagram from the database and verify relationships NOTE: The following ER digram is created by reverse Engineering the schema using MySQL workbench.

QUERY-OUTPUT

A university registrar's office maintains data about the following entities (a) courses, including number, title, credits, syllabus, and prerequisites; (b) course offerings, including course number, year, semester, section number, instructor timings, and classroom; (c) students, including student-id, name, and program; an (d) instructors, including identification number, name, department, and title. Further, the enrollment of students in courses and grades awarded to students in each course they are enrolled for must be appropriately modeled. Construct an E-R diagram for the registrar's office. Document all assumptions that you make about the mapping constraints.



Construct an ER Diagram to represent a model an Airline Booking System. The ER diagramshould show all relations between Airline booking, Ticket, Airline Enquiry. The main entitiesof this system are Ticket, Airline Booking, Passenger, Ticket, Bocking Enquiry, and AirlineEnquiry.


Ticket
ticket_no INT
class VARCHAR(45)
price INT
Indexes
Booking
Booking_id INT
date DATE
flight_no INT
seat_no INT
passenger_id INT
Indexes
Booking_inquiry
Inquiry_id INT
Booking_id INT
response_time INT
Indexes
Passenger
passenger_id INT
contact_details INT
Indexes
Airline_Inquiry
Inquiry_id INT
type VARCHAR(45)
status VARCHAR(45)
Indexes