SELECT actor FROM Actors;

→ $[\pi_{actor}(Actors) \cup \pi_{director}(Directors)]$

Q5 = Find Coen's movies with McDormand

→ SELECT DISTINCT Acts.title, ~~Acts.actor~~, ~~Movies.director~~
FROM Acts, Movies
WHERE Acts.actor = 'McDormand' AND
Movies.director = 'Coen' AND
Movies.title = Acts.title;

$e_1 = \pi_{title}(\sigma_{actor = 'McDormand'}(Acts))$

$e_2 = \pi_{title}(\sigma_{director = 'Coen'}(Movies))$

$e_1 \cap e_2$

Experiment #1

⊛ Consider the MOVIE DATABASE

- Movies (title, director, myear, rating)
- Actors (actor, ayear)
- Acts (actor, title)
- Directors (director, dyear)

* Write following relational algebra queries for a given set of relations.

1. Find movies made after 1997.

→ SELECT * FROM Movies
WHERE myear > 1997;

$[\sigma_{myear > 1997}(Movies)]$

2. Find movie made by Hanson after 1997

→ SELECT * FROM Movies
WHERE myear = '1997' AND
director = 'Hanson';

→ $[\sigma_{myear > 1997 \wedge director = 'Hanson'}(Movies)]$

3. Find all movies and their ratings.

→ SELECT title, rating FROM Movies;

$[\pi_{title, rating}(Movies)]$

4. Find all actors and directors.

→ SELECT director AS total_Director_Actor FROM Directors.
UNION

```
mysql> select m.title from Acts a,Movies m
    -> Where m.title = a.title AND
    -> m.director='Coen' AND a.actor='McDormand';
+----------------+
| title          |
+----------------+
| Fargo          |
| Rising Arizona |
+----------------+
2 rows in set (0.00 sec)

mysql>
```

```
mysql> select director AS total_Director_Actor from Directors
    -> UNION
    -> Select actor from Actors;
+----------------------+
| total_Director_Actor |
+----------------------+
| Nolan                |
| Steven               |
| Coen                 |
| Raimi                |
| Hanson               |
| Cage                 |
| Hanks                |
| Maguire              |
| McDormand            |
| DCaprio              |
| Rohan                |
| Jude Law             |
| Anna Hathaway        |
+----------------------+
13 rows in set (0.00 sec)
```

```
mysql> select title,rating From Movies;
+--------------------+--------+
| title              | rating |
+--------------------+--------+
| Inception          |    8.8 |
| Hacked             |    7.3 |
| AI                 |    6.3 |
| Interstellar       |    8.6 |
| Fargo              |    8.2 |
| Rising Arizona     |    7.4 |
| Amazing Spiderman  |    7.6 |
| Wonder Boys        |    8.3 |
+--------------------+--------+
8 rows in set (0.00 sec)
```

```
mysql> select * from Movies
    -> where myear > 1997;
+------------------+----------+-------+--------+
| title            | director | myear | rating |
+------------------+----------+-------+--------+
| Inception        | Nolan    |  2010 |    8.8 |
| Hacked           | Steven   |  2019 |    7.3 |
| AI               | Steven   |  2001 |    6.3 |
| Interstellar     | Nolan    |  2014 |    8.6 |
| Amazing Spiderman | Raimi   |  2008 |    7.6 |
| Wonder Boys      | Hanson   |  2016 |    8.3 |
+------------------+----------+-------+--------+
6 rows in set (0.00 sec)

mysql>  select * from Movies
    -> where myear>1997 AND
    -> director='Hanson';
+-------------+----------+-------+--------+
| title       | director | myear | rating |
+-------------+----------+-------+--------+
| Wonder Boys | Hanson   |  2016 |    8.3 |
+-------------+----------+-------+--------+
1 row in set (0.00 sec)

mysql> select * from Movies where myear>1997 ^ director='Hanson';
+---------------+----------+-------+--------+
| title         | director | myear | rating |
+---------------+----------+-------+--------+
| Fargo         | Coen     |  1996 |    8.2 |
| Rising Arizona | Coen    |  1987 |    7.4 |
+---------------+----------+-------+--------+
2 rows in set, 9 warnings (0.00 sec)
```

```
mysql> select * from Acts;
+--------------+-------------------+
| actor        | title             |
+--------------+-------------------+
| Cage         | Rising Horizon    |
| Hanks        | Interstellar      |
| Maguire      | Wonder Boys       |
| Maguire      | Amazing Spiderman |
| McDormand    | Fargo             |
| McDormand    | Rising Arizona    |
| McDormand    | Wonder Boys       |
| DCaprio      | Inception         |
| Rohan        | Hacked            |
| Jude Law     | AI                |
| Anna Hathaway | AI               |
+--------------+-------------------+
11 rows in set (0.00 sec)

mysql> select * from Movies;
+-------------------+----------+-------+--------+
| title             | director | myear | rating |
+-------------------+----------+-------+--------+
| Inception         | Nolan    |  2010 |    8.8 |
| Hacked            | Steven   |  2019 |    7.3 |
| AI                | Steven   |  2001 |    6.3 |
| Interstellar      | Nolan    |  2014 |    8.6 |
| Fargo             | Coen     |  1996 |    8.2 |
| Rising Arizona    | Coen     |  1987 |    7.4 |
| Amazing Spiderman | Raimi    |  2008 |    7.6 |
| Wonder Boys       | Hanson   |  2016 |    8.3 |
+-------------------+----------+-------+--------+
8 rows in set (0.00 sec)
```

```
+-------------------+
| Tables_in_movies  |
+-------------------+
| Actors            |
| Acts              |
| Directors         |
| Movies            |
+-------------------+
4 rows in set (0.01 sec)

mysql> select * from Actors;
+-----------------+-------+
| actor           | ayear |
+-----------------+-------+
| Cage            |  1964 |
| Hanks           |  1956 |
| Maguire         |  1975 |
| McDormand       |  1957 |
| DCaprio         |  1974 |
| Rohan           |  1998 |
| Jude Law        |  1964 |
| Anna Hathaway   |  1972 |
+-----------------+-------+
8 rows in set (0.00 sec)

mysql> select * from Directors;
+----------+-------+
| director | dyear |
+----------+-------+
| Nolan    | 1945  |
| Steven   | 1980  |
| Coen     | 1954  |
| Raimi    | 1945  |
| Hanson   | 1959  |
+----------+-------+
5 rows in set (0.00 sec)
```

(b) Deletion:

```
DELETE FROM Employee
WHERE fname='Simon';

DELETE FROM Employee
where bdate='1979-08-15';
```

(c) Update:

```
UPDATE Employee
SET fname='Tata', address='Bholapur'
WHERE lname='Belister';
```

Qs

(i) Inserting:

Insert INTO Department (dname, mgrssn, mgrstartdate)

Values ('Cashier', 02, '2010-02-16'),

('Collector', 03, '2007-08-20'),

('Decorator', 04, '2011-05-15'),

('Lumbser', 05, '2004-09-17');

('Digger', 06, '2008-03-26'),

('Vice-Assistant', 07, '2008-03-26'),

('Director', 08, '2008-09-05'),

('Coordinator', 09, '2020-01-07');

(ii) Insert INTO Employee (fname, lname, bdate, address, gender, salary, superssn, dno)

VALUES ('SHUBHAM', 'Prince', '1995-09-15', 'Kahalgaon', 'M', 25000.00, 01, 1),

('Simon', 'GOS', '1991-06-21', 'Bangalore', 'M', 24982.43, 02, 2),

('Timon', 'Bola', '1998-11-19', 'Shindi', 'M', 24002.43, 03, 3),

('Limbra', 'Tripper', '1984-06-02', 'Delhi', 'F', 50000.43, 04, 4);

('Sunny', 'Chupa', '1979-08-15', 'Bangkok', 'M', 50000.43, 05, 5),

('Naive', 'Grounda', '1990-11-19', 'Howrah', 'F', 24000.43, 06, 6),

('Tiscon', 'Delister', '1985-01-01', 'Bolpur', 'F', 44000.43, 07, 7);

(i) CREATE TABLE Department

```
( dno INT PRIMARY KEY AUTO_INCREMENT,
dname VARCHAR (30),
mgrssn VARCHAR (30) UNIQUE,
mgrstartdate DATE );
```

(ii) CREATE TABLE Employee

```
( ssn INT PRIMARY KEY AUTO_INCREMENT,
fname VARCHAR (30) NOT NULL,
lname VARCHAR (30),
bdate Date ,
address VARCHAR (50),
gender ENUM ('M', 'F'),
salary DOUBLE (8,2),
superssn INT,
dno INT , );
```

Experiment #2

* Consider the company database with following tables.

Employee (fname, lname, ssn, bdate, address, gender, salary, superssn, dno)

Department (dname, dnumber, mgrssn, mgrstartdate)

Perform the following

Q1. Create Company database

```
CREATE DATABASE Company;
```

Q2. Viewing all databases.

```
SHOW DATABASES;
```

Q3. Viewing all Tables in a Database.

```
SHOW Tables;
```

Q4. Creating tables (With or Without Dependencies)

* Table - Department

```
mysql> update Employee
    -> set fname='Tata',address='Bholapur'
    -> where lname='Belister';
Query OK, 1 row affected (0.19 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> delete from Employee
    -> Where fname='Simon';
Query OK, 1 row affected (0.20 sec)

mysql> delete from Employee
    -> Where bdate='1979-08-15';

Query OK, 1 row affected (0.13 sec)

mysql>
mysql> update Employee
    -> set fname='Tata',address='Bholapur'
    -> where lname='Belister';
Query OK, 1 row affected (0.19 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> select * from department;
ERROR 1146 (42S02): Table 'Company.department' doesn't exist
mysql> select * from Department;
+-----+----------------+--------+--------------+
| dno | dname          | mgrssn | mgrstartdate |
+-----+----------------+--------+--------------+
|   1 | Assistant      |      1 | 2010-02-12   |
|   3 | Cashier        |      2 | 2010-02-16   |
|   4 | Collector      |      3 | 2007-08-20   |
|   5 | Decorator      |      4 | 2011-05-15   |
|   6 | Lumberer       |      5 | 2004-09-17   |
|   7 | Digger         |      6 | 2008-03-26   |
|   8 | Vice-Assistant |      7 | 2008-03-26   |
|   9 | Director       |      8 | 2008-09-05   |
|  10 | Coordinator    |      9 | 2020-01-07   |
+-----+----------------+--------+--------------+
9 rows in set (0.00 sec)

mysql> Select * From Employee;
+-----+---------+----------+------------+--------------+--------+----------+----------+------+
| ssn | fname   | lname    | bdate      | address      | gender | salary   | superssn | dno  |
+-----+---------+----------+------------+--------------+--------+----------+----------+------+
|   1 | Shubham | Prince   | 1995-09-15 | Kahalgaon    | M      | 25000.00 |        1 |    1 |
|  20 | Simon   | Gos      | 1991-06-21 | Bangalore    | M      | 24002.43 |        2 |    3 |
|  21 | Timon   | Bola     | 1998-11-19 | Shindi Bazar | M      | 24982.43 |        3 |    4 |
|  22 | Limbra  | Tripper  | 1984-06-02 | Delhi        | F      | 20000.43 |        4 |    5 |
|  23 | Sunny   | Chupa    | 1979-08-15 | Bangkok      | M      | 50000.43 |        5 |    6 |
|  24 | Naive   | Grounda  | 1990-11-19 | Howrah       | F      | 24000.43 |        6 |    7 |
|  25 | Tiscon  | Belister | 1985-01-01 | Bolpur       | F      | 44000.43 |        7 |    8 |
+-----+---------+----------+------------+--------------+--------+----------+----------+------+
7 rows in set (0.00 sec)
```

```
Alter table Employee
modify salary DOUBLE(7,2);
  INSERT INTO Department(dname,mgrssn,mgrstartdate)
  Values('Cashier',02,'2010-02-16'),
        ('Collector',03,'2007-08-20'),
        ('Decorator',04,'2011-05-15'),
        ('Lumberer',05,'2004-09-17'),
        ('Digger',06,'2008-03-26'),
        ('Vice-Assistant',07,'2008-03-26'),
        ('Director',08,'2008-09-05'),
        ('Coordinator',09,'2020-01-07');
INSERT INTO Employee(fname,lname,bdate,address,gender,salary,superssn,dno)
VALUES('Simon', 'Gos' , '1991-06-21' , 'Bangalore' , 'M' , 24002.43 , 2 , 3),
     ('Timon', 'Bola', '1998-11-19', 'Shindi Bazar', 'M', 24982.43, 3, 4),
     ('Limbra' , 'Tripper' , '1984-06-02' , 'Delhi' , 'F' , 20000.43, 4 , 5),
     ('Sunny', 'Chupa', '1979-08-15' , 'Bangkok', 'M', 50000.43, 5, 6),
     ('Naive', 'Grounda', '1990-11-19' , 'Howrah' , 'F' , 24000.43 , 6 , 7),
     ('Tiscon' , 'Belister' , '1985-01-01' , 'Bolpur' , 'F' , 44000.43 ,7 ,8);
```

```
create table Employee
(
   ssn INT PRIMARY KEY AUTO_INCREMENT,
   fname VARCHAR(30) NOT NULL,
   lname VARCHAR(30),
   bdate DATE,
   address VARCHAR(50),
   gender ENUM('M','F'),
   salary DOUBLE(4,2),
   superssn INT,
   dno INT
);
```

```sql
Create Database Company;
use Company;
create table Department
(
  dno INT Primary Key AUTO_INCREMENT,
  dname VARCHAR(30),
  mgrssn INT,
  mgrstartdate DATE
  );
```

(7) Retrive total salary of Employee which is greater than 120000

```
SELECT emp_name , sum(salary) FROM Employee,
group By (emp_name)
Having sum(salary) > 120000;
```

(8) Display name of employee in descending order.

```
SELECT emp_name from Employee
ORDER BY emp_name desc;
```

(9) Display details of employee whose name is Amit and salary greater than 50000;

```
SELECT * from Employee
WHERE emp_name = 'Amit' AND
Salary > 50000;
```

Experiment - 3.

→ Consider Employee table

→ Employee (EmpNo, Ename, dept, salary, DOJ, branch)

Perform the following:

(Q1) Display all the fields of employee table.

→ SELECT * from Employee;

(Q2) Retrive employee number and their salary

→ SELECT empno, salary from Employee;

(Q3) Retrive average salary of all employee

→ SELECT avg(salary) from employee;

(4) Retrive number of Employee

SELECT count(*) from employee;

(5) Retrive distinct number of employee

SELECT count (DISTINCT emp_name) from Employee;

(6) Retrive total salary of Employee group by employee numbers and count similar names

SELECT emp_name, sum(salary) FROM Employee
GROUP BY (emp_name)
HAVING sum(salary) > 20000

```
mysql> select emp_name from Employee
    -> ORDER BY emp_name desc;
+-----------+
| emp_name  |
+-----------+
| Sunita    |
| Sunita    |
| Mahesh    |
| Amit      |
| Amit      |
+-----------+
5 rows in set (0.00 sec)

mysql> Select * from Employee
    -> WHERE emp_name='Amit' AND
    -> salary>50000;
+-------+----------+------+--------+------------+-----------+
| empno | emp_name | dept | salary | doj        | branch    |
+-------+----------+------+--------+------------+-----------+
| 102   | Amit     | HR   |  70000 | 2002-07-02 | Bangalore |
+-------+----------+------+--------+------------+-----------+
1 row in set (0.01 sec)

mysql>
```

```
mysql> select count(Distinct emp_name) from Employee;
+---------------------------+
| count(Distinct emp_name) |
+---------------------------+
|                         3 |
+---------------------------+
1 row in set (0.00 sec)

mysql> select emp_name,sum(salary) from Employee
    -> GROUP BY(emp_name);
+----------+-------------+
| emp_name | sum(salary) |
+----------+-------------+
| Amit     |      115000 |
| Sunita   |      187000 |
| Mahesh   |      145000 |
+----------+-------------+
3 rows in set (0.00 sec)

mysql> select emp_name,sum(salary) from Employee
    -> GROUP BY(emp_name)
    -> HAVING
    -> sum(salary)>120000;
+----------+-------------+
| emp_name | sum(salary) |
+----------+-------------+
| Sunita   |      187000 |
| Mahesh   |      145000 |
+----------+-------------+
2 rows in set (0.00 sec)
```

```
mysql> select * from Employee;
+-------+----------+------------+--------+------------+-----------+
| empno | emp_name | dept       | salary | doj        | branch    |
+-------+----------+------------+--------+------------+-----------+
| 101   | Amit     | Production |  45000 | 2000-03-12 | Bangalore |
| 102   | Amit     | HR         |  70000 | 2002-07-02 | Bangalore |
| 103   | Sunita   | Manager    | 120000 | 2001-01-11 | Mysore    |
| 105   | Sunita   | IT         |  67000 | 2001-08-01 | Mysore    |
| 106   | Mahesh   | Civil      | 145000 | 2003-09-20 | Mumbai    |
+-------+----------+------------+--------+------------+-----------+
5 rows in set (0.00 sec)

mysql> select empno,salary from Employee;
+-------+--------+
| empno | salary |
+-------+--------+
| 101   |  45000 |
| 102   |  70000 |
| 103   | 120000 |
| 105   |  67000 |
| 106   | 145000 |
+-------+--------+
5 rows in set (0.01 sec)

mysql> select avg(salary) from Employee;
+-------------+
| avg(salary) |
+-------------+
|  89400.0000 |
+-------------+
1 row in set (0.00 sec)

mysql> select count(*) from Employee;
+----------+
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)
```

Q5. Retrive the name of employees who born in the year 1990's

```
→ SELECT e.fname, e.lname, e.bdate
→ FROM Employee e
→ WHERE e.bdate LIKE '199%';
```

Q6. Retrive the name of employees and their Depatonent name (Using JOIN)

```
→ SELECT e.fname, e.lname, d.dname
→ FROM Employee e JOIN
→ Department d ON
→ e.dno = d.dnumber;
```

```
→ SELECT  SUM(e.salary), MAX(e.salary), MIN(e.salary),
          AVG(e.salary)
→ FROM  Employee e, Department d
→   WHERE  d.dnumber = e.dno  AND
      → d.dname = 'Accounts';
```

Q3. Retrive the name of each Employee controlled by department number 5

```
→ SELECT  e.fname, e.lname FROM Employee e
→   WHERE EXISTS (SELECT dno FROM Employee
                    WHERE e.dno = 5);
→
```

Q4. Retrive the name of each department and number of employees working in each department which has at least 2 Employees.

```
→ SELECT dname, COUNT(*)
→ FROM Employee e, Department d
→ WHERE d.dnumber = e.dno
→ AND
→ d.dno IN (SELECT e.dno
→   FROM Employee e
→   GROUP BY e.dno
→   HAVING COUNT(*) >2.)
→   ORDER BY Dno;
```

Experiment = 4.

Consider the Company database with following tables

Employee (fname, lname, ssn, bdate, address, gender, salary, superssn, dno)

Department (dname, dnumber, mgrssn, mgrstartdate)

→ Create tables and perform the following:

Q.1 How the resulting salaries if every employee working on the 'Research' Departments is given a 10% raise.

```
→ SELECT e.fname, e.lname, e.salary,
→   1.1 * e.salary AS Hike_Salary
→ FROM Employee e, Department d
→ WHERE d.dnumber = e.dno
→ AND
→ d.dname = 'Research';
```

2. Find the SUM of the Salaries of all employees of the 'Accounts' department, as well as the maximum Salary, the minimum Salary, and the Average Salary in this department.

```
mysql> select e.fname,e.lname,d.dname
    -> FROM Employee e JOIN
    -> Department d ON
    -> e.dno=d.dnumber;
+----------+--------+-----------------+
| fname    | lname  | dname           |
+----------+--------+-----------------+
| AHANA    | K      | Research        |
| SANTHOSH | KUMAR  | Research        |
| NEHA     | SN     | Research        |
| PAVAN    | HEGDE  | Administrations |
| NAGESH   | HR     | Administrations |
| JOHN     | SCOTT  | Headquarters    |
| HEARN    | BAKER  | Accounts        |
| EDWARD   | SCOTT  | Accounts        |
| JAMES    | SMITH  | IT              |
| GIRISH   | MALYA  | IT              |
+----------+--------+-----------------+
10 rows in set (0.00 sec)
```

```
mysql> select sum(e.salary),MAX(e.salary),MIN(e.salary),AVG(e.salary)
    -> FROM Employee e, Department d
    -> WHERE d.dnumber=e.dno AND
    -> d.dname='Accounts';
+---------------+---------------+---------------+---------------+
| sum(e.salary) | MAX(e.salary) | MIN(e.salary) | AVG(e.salary) |
+---------------+---------------+---------------+---------------+
|       1200000 |        700000 |        500000 |   600000.0000 |
+---------------+---------------+---------------+---------------+
1 row in set (0.00 sec)

mysql> Select e.fname,e.lname FROM Employee e
    -> WHERE exists( Select dno From Employee Where
    -> e.dno=5);
+--------+-------+
| fname  | lname |
+--------+-------+
| JAMES  | SMITH |
| GIRISH | MALYA |
+--------+-------+
2 rows in set (0.00 sec)

mysql> select e.fname,e.lname,e.bdate
    -> FROM Employee e
    -> WHERE e.bdate LIKE '199%';
+----------+-------+------------+
| fname    | lname | bdate      |
+----------+-------+------------+
| AHANA    | K     | 1992-05-19 |
| SANTHOSH | KUMAR | 1993-02-20 |
| JAMES    | SMITH | 1992-04-19 |
| HEARN    | BAKER | 1995-05-23 |
| EDWARD   | SCOTT | 1994-03-04 |
| PAVAN    | HEGDE | 1993-06-28 |
| GIRISH   | MALYA | 1991-04-24 |
| NEHA     | SN    | 1990-08-09 |
| JOHN     | SCOTT | 1995-02-12 |
| VEENA    | M     | 1990-09-03 |
| NAGESH   | HR    | 1993-04-17 |
+----------+-------+------------+
11 rows in set (0.00 sec)
```

```
25
26 •     alter table Department
27      add mgrssn varchar(10);
28
29 •     ALTER TABLE Department
30      ADD FOREIGN KEY(mgrssn) REFERENCES Employee (SSN);
31
32 •     alter table Employee
33       add dno varchar(10);
34
35 •   alter table Employee
36    add foreign key(dno) references Department(dnumber);
37
```

```sql
create table Department(
    dnumber Varchar(10) PRIMARY KEY NOT NULL,
    dname VARCHAR(30),
    mgrstartdate DATE);

create table Employee(
    ssn varchar(10) primary key not null,
    fname varchar(30),
    lname varchar(30),
    address varchar(50),
    gender enum('M','F'),
    salary integer,
    superssn varchar(10),
   foreign key(superssn) references Employee(ssn));

  alter table Employee
  add bdate DATE;
```

```sql
INSERT INTO Employee(ssn,fname,lname,address,gender,salary,bdate)
VALUES ('RNSECE01','JOHN','SCOTT','BANGALORE','M', 450000,'1995-02-12'),
       ('RNSCSE01','JAMES','SMITH','BANGALORE','M', 500000,'1992-04-19'),
       ('RNSCSE02','HEARN','BAKER','BANGALORE','M', 700000,'1995-05-23'),
       ('RNSCSE03','EDWARD','SCOTT','MYSORE','M', 500000,'1994-03-04'),
       ('RNSCSE04','PAVAN','HEGDE','MANGALORE','M', 650000,'1993-06-28'),
       ('RNSCSE05','GIRISH','MALYA','MYSORE','M', 450000,'1991-04-24'),
       ('RNSCSE06','NEHA','SN','BANGALORE','F', 800000,'1990-08-09'),
       ('RNSACC01','AHANA','K','MANGALORE','F', 350000,'1992-05-19'),
       ('RNSACC02','SANTHOSH','KUMAR','MANGALORE','M', 300000,'1993-02-20'),
       ('RNSISE01','VEENA','M','MYSORE','M', 600000,'1990-09-03'),
       ('RNSIT01','NAGESH','HR','BANGALORE','M', 500000,'1993-04-17');

INSERT INTO Department
VALUES  (1,'Research','2001-01-01','RNSECE01'),
       (2,'Administrations','2016-08-01','RNSISE01'),
       (3,'Headquarters','1908-06-08','RNSCSE05'),
       (4,'Accounts','1998-01-01','RNSACC02'),
       (5,'IT','2016-09-03','RNSIT01');
```

```
mysql> select * from Employee;
+----------+----------+-------+-----------+--------+--------+----------+------+------------+
| ssn      | fname    | lname | address   | gender | salary | superssn | dno  | bdate      |
+----------+----------+-------+-----------+--------+--------+----------+------+------------+
| RNSACC01 | AHANA    | K     | MANGALORE | F      | 350000 | RNSACC02 | 1    | 1992-05-19 |
| RNSACC02 | SANTHOSH | KUMAR | MANGALORE | M      | 300000 | NULL     | 1    | 1993-02-20 |
| RNSCSE01 | JAMES    | SMITH | BANGALORE | M      | 500000 | RNSCSE02 | 5    | 1992-04-19 |
| RNSCSE02 | HEARN    | BAKER | BANGALORE | M      | 700000 | RNSCSE03 | 4    | 1995-05-23 |
| RNSCSE03 | EDWARD   | SCOTT | MYSORE    | M      | 500000 | RNSCSE04 | 4    | 1994-03-04 |
| RNSCSE04 | PAVAN    | HEGDE | MANGALORE | M      | 650000 | RNSCSE05 | 2    | 1993-06-28 |
| RNSCSE05 | GIRISH   | MALYA | MYSORE    | M      | 450000 | NULL     | 5    | 1991-04-24 |
| RNSCSE06 | NEHA     | SN    | BANGALORE | F      | 800000 | NULL     | 1    | 1990-08-09 |
| RNSECE01 | JOHN     | SCOTT | BANGALORE | M      | 450000 | NULL     | 3    | 1995-02-12 |
| RNSISE01 | VEENA    | M     | MYSORE    | M      | 600000 | NULL     | NULL | 1990-09-03 |
| RNSIT01  | NAGESH   | HR    | BANGALORE | M      | 500000 | NULL     | 2    | 1993-04-17 |
+----------+----------+-------+-----------+--------+--------+----------+------+------------+
11 rows in set (0.00 sec)

mysql> select * from Department;
+---------+----------------+--------------+----------+
| dnumber | dname          | mgrstartdate | mgrssn   |
+---------+----------------+--------------+----------+
| 1       | Research       | 2001-01-01   | RNSECE01 |
| 2       | Administrations | 2016-08-01   | RNSIT01  |
| 3       | Headquarters   | 1908-06-08   | RNSISE01 |
| 4       | Accounts       | 1998-01-01   | RNSACC02 |
| 5       | IT             | 2016-09-03   | RNSIT01  |
+---------+----------------+--------------+----------+
5 rows in set (0.00 sec)

mysql>
```

Q2 Selecting from a view

```
Select * from sales_view;
```

Q3. Drop view

```
Drop view sales_staff;
```

Experiment 5.

Consider the Company database with following tables.

Employee (fname, lname, ssn, bdate, address, gender, salary, superssn, dno)

Perform the following.

Q1 Creating Views (With or Without Check Option).

Solution:

```
CREATE TABLE Employee (
    SSN VARCHAR (10) PRIMARY KEY NOT NULL,
    fname Varchar(20),
    lname VARCHAR(20),
    bdate DATE,
    addres VARCHAR(50),
    gender enum('M','F'),
    salary INTEGER,
    superssn VARCHAR(10)
    dno VARCHAR(10)
    FOREIGN KEY (superssn) RERERENCES Department(ssn),
    FOREIGN KEY (dno) REFERENCES Department(dnumber)
);
```

→ Create View:

```
CREATE VIEW sales_staff AS
    SELECT fname, ssn, dno
    FROM Employee
    WHERE dno = 5;
WITH CHECK OPTION CONSTRAINT sales_staff_cnst;
```

```
mysql> CREATE VIEW sales_staff AS
    -> SELECT fname,lname,dno
    -> FROM Employee
    -> WHERE dno=5;
Query OK, 0 rows affected (0.21 sec)

mysql> Select * from sales_staff;
+---------+-------+------+
| fname   | lname | dno  |
+---------+-------+------+
| JAMES   | SMITH | 5    |
| GIRISH | MALYA | 5    |
+---------+-------+------+
2 rows in set (0.00 sec)

mysql> drop view sales_staff;
Query OK, 0 rows affected (0.18 sec)

mysql>
```

```
SELECT CARD_NO .FROM BOOK_LENDING
→. WHERE DATE_OUT .BETWEEN '2017-01-01' AND
                           '2017-07-01'
   GROUP BY CARD_NO
   HAVING COUNT(*) >3;
```

Q3. Delete a book in Book table. Update the contents of other tables to reflect this data Manipulation operation.

```
DELETE FROM BOOK
WHERE BOOK_ID = 3;
```

Q4 Partition the BOOK table based on year of publication. Demonstrate its working using simple query.

```
CREATE VIEW V_PUBLIC AS
SELECT PUB_YEAR
FROM BOOK;
```

Q5. Create a view of all books and its number of copies that are currently available in the library.

```
CREATE VIEW V_BOOKS AS
SELECT b.BOOK, COPIES.
       BOOK b, BOOK_COPIES C, LIBRARY_Branch l
WHERE b.BOOK_ID = C.BOOK_ID.
  AND
C.BRANCH_ID = l.BRANCH_ID;
```

Experiment 6.

BOOK (Book_id, Title, Publisher_Name, Pub_Year)

BOOK_AUTHORS (Book_id, Author_Name)

PUBLISHER (Name, Address, Phone)

BOOK_COPIES (Book_id, Branch_id, No_of_Copies)

BOOK_LENDING (Book_id, Branch_id, Card_No, Date_Out, Due_Date)

LIBRARY_BRANCH (Branch_id, Branch_Name, Address)

Write SQL queries to

1. Retrive details of all books in the library_id, title, name of publisher, authors, number of copies in each branch, etc.

```
→ SELECT b.BOOK_ID, b.TITLE, b.PUBLISHER, a.AUTHOR_NAME
    → c.NO_OF_COPIES, l.BRANCH_ID
→ FROM BOOK b, BOOK_AUTHORS a, BOOK_COPIES c,
→ LIBRARY BRANCH l
→ WHERE b.BOOK_ID = a.BOOK_ID
→ AND
→ l.BRANCH_ID = c.BRANCH_ID;
```

Q.2 Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to JUN 2017;

```
mysql> CREATE VIEW V_BOOKS
    -> AS
    -> SELECT b.BOOK_ID,b.TITLE,c.NO_OF_COPIES
    -> FROM BOOK b,BOOK_COPIES c,LIBRARY_BRANCH l
    -> WHERE b.BOOK_ID=c.BOOK_ID
    -> AND c.BRANCH_ID=l.BRANCH_ID;
Query OK, 0 rows affected (0.17 sec)

mysql> SELECT * FROM V_BOOKS;
+---------+-------+--------------+
| BOOK_ID | TITLE | NO_OF_COPIES |
+---------+-------+--------------+
|       1 | DBMS  |           10 |
|       1 | DBMS  |            5 |
|       2 | ADBMS |            2 |
|       2 | ADBMS |            5 |
|       4 | CG    |            3 |
|       5 | OS    |            1 |
+---------+-------+--------------+
6 rows in set (0.00 sec)
```

```
mysql> select CARD_NO FROM BOOK_LENDING
    ->      WHERE DATE_OUT BETWEEN '2017-01-01' AND '2017-07-01'
    ->      GROUP BY CARD_NO
    ->      HAVING COUNT(*)>3;
+---------+
| CARD_NO |
+---------+
|     101 |
+---------+
1 row in set (0.00 sec)

mysql> DELETE FROM BOOK
    -> WHERE BOOK_ID=3;
Query OK, 1 row affected (0.15 sec)

mysql> SELECT * FROM BOOK;
+---------+-------+----------+----------------+
| BOOK_ID | TITLE | PUB_YEAR | PUBLISHER_NAME |
+---------+-------+----------+----------------+
|       1 | DBMS  | JAN-2017 | MCGRAW-HILL    |
|       2 | ADBMS | JUN-2016 | MCGRAW-HILL    |
|       4 | CG    | SEP-2015 | GRUPOPLANETA   |
|       5 | OS    | MAY-2016 | PEARSON        |
+---------+-------+----------+----------------+
4 rows in set (0.00 sec)

mysql> CREATE VIEW V_PUBLIC AS
    -> SELECT PUB_YEAR
    -> FROM BOOK;
Query OK, 0 rows affected (0.32 sec)

mysql> SELECT * FROM V_PUBLIC;
+----------+
| PUB_YEAR |
+----------+
| JAN-2017 |
| JUN-2016 |
| SEP-2015 |
| MAY-2016 |
+----------+
4 rows in set (0.00 sec)
```

```
mysql> select b.BOOK_ID,b.TITLE,b.PUBLISHER_NAME,a.AUTHOR_NAME,c.NO_OF_COPIES,l.BRANCH_ID
    -> FROM BOOK b,BOOK_AUTHORS a,BOOK_COPIES c,LIBRARY_BRANCH l
    -> where b.BOOK_ID=a.BOOK_ID
    -> AND
    -> b.BOOK_ID=c.BOOK_ID
    -> AND
    -> l.BRANCH_ID=c.BRANCH_ID;
+---------+-------+----------------+-------------+--------------+-----------+
| BOOK_ID | TITLE | PUBLISHER_NAME | AUTHOR_NAME | NO_OF_COPIES | BRANCH_ID |
+---------+-------+----------------+-------------+--------------+-----------+
|       1 | DBMS  | MCGRAW-HILL    | NAVATHE     |           10 |        10 |
|       1 | DBMS  | MCGRAW-HILL    | NAVATHE     |            5 |        11 |
|       2 | ADBMS | MCGRAW-HILL    | NAVATHE     |            2 |        12 |
|       2 | ADBMS | MCGRAW-HILL    | NAVATHE     |            5 |        13 |
|       3 | CN    | PEARSON        | TANENBAUM   |            7 |        14 |
|       4 | CG    | GRUPOPLANETA   | EDWARD ANGEL|            3 |        11 |
|       5 | OS    | PEARSON        | GALVIN      |            1 |        10 |
+---------+-------+----------------+-------------+--------------+-----------+
7 rows in set (0.01 sec)
```

```
93 •   INSERT INTO BOOK_LENDING
94     VALUES('2017-01-01','17-06-01', 1, 10, 101),
95           ('2017-01-11','2017-03-11', 3, 14, 101),
96           ('2017-02-21','2017-04-21', 2, 13, 101),
97           ('2017-03-15','2017-07-15', 4, 11, 101),
98           ('2017-04-12','2017-05-12', 1, 11, 104);
99
100
101
102
103
104
105
106
```

```
INSERT INTO LIBRARY_BRANCH
VALUES (10,'RR NAGAR','BANGALORE'),
       (11,'RNSIT','BANGALORE'),
       (12,'RAJAJI NAGAR', 'BANGALORE'),
       (13,'NITTE','MANGALORE'),
       (14,'MANIPAL','UDUPI');

INSERT INTO BOOK_COPIES
VALUES (10, 1, 10),
       (5, 1,11),
       (2, 2,12),
       (5, 2,13),
       (7, 3,14),
       (1, 5,10),
       (3, 4,11);
    INSERT  INTO  CARD
     VALUES(100),(101),(102),(103),(104);
```

```
51
52 •    INSERT INTO PUBLISHER
53     VALUES ('MCGRAW-HILL','9989076587','BANGALORE'),
54            ('PEARSON','9889076565','NEWDELHI'),
55            ('RANDOM HOUSE','7455679345','HYDRABAD'),
56            ('HACHETTE LIVRE','8970862340','CHENNAI'),
57            ('GRUPOPLANETA','7756120238','BANGALORE');
58
59 •    select * from PUBLISHER;
60
61 •    INSERT INTO BOOK
62     VALUES (1,'DBMS','JAN-2017','MCGRAW-HILL'),
63            (2,'ADBMS','JUN-2016','MCGRAW-HILL'),
64            (3,'CN','SEP-2016','PEARSON'),
65            (4,'CG','SEP-2015','GRUPOPLANETA');
66 •    INSERT INTO BOOK VALUES(5,'OS','MAY-2016','PEARSON');
67
```

```sql
CREATE TABLE BOOK_LENDING (
DATE_OUT DATE,
DUE_DATE DATE,
BOOK_ID INTEGER,
FOREIGN KEY(BOOK_ID)REFERENCES BOOK(BOOK_ID) ON DELETE CASCADE,
BRANCH_ID INTEGER,
FOREIGN KEY(BRANCH_ID)REFERENCES LIBRARY_BRANCH (BRANCH_ID)ON DELETE CASCADE,
CARD_NO INTEGER,
FOREIGN KEY(CARD_NO)REFERENCES CARD(CARD_NO)ON DELETE CASCADE,
PRIMARY KEY (BOOK_ID, BRANCH_ID, CARD_NO));


INSERT INTO PUBLISHER
VALUES ('MCGRAW-HILL','9989076587','BANGALORE'),
       ('PEARSON','9889076565','NEWDELHI'),
       ('RANDOM HOUSE','7455679345','HYDRABAD'),
       ('HACHETTE LIVRE','8970862340','CHENNAI'),
       ('GRUPOPLANETA','7756120238','BANGALORE');

```

```
CREATE TABLE LIBRARY_BRANCH (
BRANCH_ID INTEGER PRIMARY KEY,
 BRANCH_NAME VARCHAR(50),
 ADDRESS VARCHAR(50));

 CREATE TABLE BOOK_COPIES (
 NO_OF_COPIES INTEGER,
 BOOK_ID INTEGER,
 FOREIGN KEY(BOOK_ID)REFERENCES BOOK (BOOK_ID) ON DELETE CASCADE,
 BRANCH_ID INTEGER,
 FOREIGN KEY(BRANCH_ID)REFERENCES LIBRARY_BRANCH(BRANCH_ID) ON DELETE CASCADE,
 PRIMARY KEY (BOOK_ID, BRANCH_ID));


 CREATE TABLE CARD (
 CARD_NO INTEGER PRIMARY KEY);

```

```sql
create database experiment6;
use experiment6;
show tables;

CREATE TABLE PUBLISHER (
NAME VARCHAR(20) PRIMARY KEY,
PHONE VARCHAR(12),
ADDRESS VARCHAR(20));

CREATE TABLE BOOK (
BOOK_ID INTEGER PRIMARY KEY,
TITLE VARCHAR(20),
PUB_YEAR VARCHAR(20),
PUBLISHER_NAME VARCHAR(20),
FOREIGN KEY(PUBLISHER_NAME)REFERENCES PUBLISHER (NAME) ON DELETE CASCADE);

CREATE TABLE BOOK_AUTHORS (
AUTHOR_NAME VARCHAR(20),
BOOK_ID INTEGER,
FOREIGN KEY(BOOK_ID) REFERENCES BOOK (BOOK_ID) ON DELETE CASCADE,
PRIMARY KEY (BOOK_ID, AUTHOR_NAME));

```

Experiment # 7.

SALESMAN (Salesman_id, Name, City, Commission)
CUSTOMER (Customer_id, Cust_Name, City, Grade, Salesman_id)
ORDERS (Ord_No, Purchase_Amt, Ord_Date, Customer_id, Salesma_id)

Write SQL Queries.

Q1 Count the customers with grades above Bangalore's average.

```
SELECT GRADE, COUNT(DISTINCT CUSTOMER_ID)
    FROM CUSTOMER
GROUP BY GRADE
HAVING GRADE > (SELECT AVG(GRADE)
                FROM CUSTOMER
                WHERE CITY = 'BANGALORE');
```

Q2 Find the name and number of all salesman who had more than one customer.

```
SELECT SALESMAN_ID, NAME
FROM SALESMAN S
WHERE 1 < (SELECT COUNT(*)
           FROM CUSTOMER
           WHERE SALESMAN_ID = S.SALESMAN_ID);
```

Q3 List of all salesmen and indicate those who have and don't have customers in their city.
(USE UNION OPERATION)

```
SELECT S.SALESMAN_ID, NAME, CUST_NAME,
       COMMISSION
FROM SALESMAN S, CUSTOMER C
WHERE S.CITY = C.CITY
UNION.
SELECT SALESMAN_ID, NAME, 'NO MATCH', COMMISSION
  FROM SALESMAN,
WHERE NOT CITY = ANY
(SELECT CITY FROM CUSTOMER)
  ORDER BY 2 DESC;
```

Q4 Create a view that finds the salesman who has the customer with highest order of a day.

```
CREATE VIEW ELITE_SALESMAN AS
SELECT b.ord_DATE, a.SALESMAN_ID, a.name
FROM
SALESMAN a, ORDERS b
WHERE a.SALESMAN_ID = b.SALESMAN_ID
  AND
b.PURCHASE_AMT = (SELECT MAX(PURCHASE_AMT)
                  FROM ORDERS c
                  WHERE c.ORD_DATE = b.ORD_DATE);
```

Q5:

Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted.

```
DELETE FROM SALESMAN
WHERE SALESMAN_ID =1000;
```

```
    -> s.SALESMAN_ID = b.SALESMAN_ID AND b.PURCHASE_AMT=(SELECT MAX (PURCHASE_AMT) FROM ORDERS o
    ->  WHERE o.ORD_DATE = b.ORD_DATE);
ERROR 1630 (42000): FUNCTION example7.MAX does not exist. Check the 'Function Name Parsing and Resolution' section in the Reference Manual
mysql> CREATE VIEW ELITE_SALESMAN
    -> AS
    -> SELECT b.ORD_DATE, s.SALESMAN_ID, s.NAME FROM SALESMAN s, ORDERS b
    -> WHERE
    -> s.SALESMAN_ID = b.SALESMAN_ID AND b.PURCHASE_AMT=(SELECT MAX(PURCHASE_AMT) FROM ORDERS o
    ->  WHERE o.ORD_DATE = b.ORD_DATE);
Query OK, 0 rows affected (0.20 sec)

mysql> select * FROM ELITE_SALESMAN;
+------------+-------------+-------+
| ORD_DATE   | SALESMAN_ID | NAME  |
+------------+-------------+-------+
| 2017-05-04 | 1000        | JOHN  |
| 2017-01-20 | 2000        | RAVI  |
| 2017-02-24 | 2000        | RAVI  |
| 2017-04-13 | 3000        | KUMAR |
| 2017-03-09 | 2000        | RAVI  |
+------------+-------------+-------+
5 rows in set (0.01 sec)

mysql> DELETE FROM SALESMAN WHERE SALESMAN_ID=1000;
Query OK, 1 row affected (0.12 sec)

mysql> select * from salesman;
ERROR 1146 (42S02): Table 'example7.salesman' doesn't exist
mysql> select * from SALESMAN;
+-------------+--------+-----------+------------+
| SALESMAN_ID | NAME   | CITY      | COMMISSION |
+-------------+--------+-----------+------------+
| 2000        | RAVI   | BANGALORE | 20%        |
| 3000        | KUMAR  | MYSORE    | 15%        |
| 4000        | SMITH  | DELHI     | 30%        |
| 5000        | HARSHA | HYDRABAD  | 15%        |
+-------------+--------+-----------+------------+
4 rows in set (0.00 sec)

mysql>
```


Consider the schema for MovieDatabase:

ACTOR (*Act_Id, Act_Name, Act_Gender*)
DIRECTOR (*Dir_Id, Dir_Name,*
ne)
(*Mov_Id, Mov_Title, Mov_Year, Mov_Lang,*
Dir_Id) MOVIE_CAST (*Act_Id, Mov_Id, Role*)
RATING (*Mov_Id,*
Rat_Stars) Write SQL queries to

1. List the titles of all movies directed by'Hitchcock'.
2. Find the movie names where one or more actors acted in two or moremovies.
3. List all actors who acted in a movie before 2000 and also in a

```
mysql> SELECT GRADE, COUNT(DISTINCT CUSTOMER_ID) FROM
    ->     CUSTOMER GROUP BY GRADE HAVING
    ->     GRADE > (SELECT AVG(GRADE) FROM CUSTOMER WHERE CITY='BANGALORE');
+-------+-----------------------------+
| GRADE | COUNT(DISTINCT CUSTOMER_ID) |
+-------+-----------------------------+
| 300   |                           1 |
| 400   |                           2 |
+-------+-----------------------------+
2 rows in set (0.00 sec)

mysql> SELECT SALESMAN_ID, NAME FROM SALESMAN s
    ->     WHERE 1 < (SELECT COUNT(*) FROM CUSTOMER WHERE SALESMAN_ID=s.SALESMAN_ID);
+-------------+------+
| SALESMAN_ID | NAME |
+-------------+------+
| 1000        | JOHN |
| 2000        | RAVI |
+-------------+------+
2 rows in set (0.00 sec)

mysql> SELECT s.SALESMAN_ID, NAME, CUST_NAME, COMMISSION FROM SALESMAN s,
    ->            CUSTOMER c WHERE s.CITY = c.CITY
    ->            UNION
    ->            SELECT SALESMAN_ID, NAME, 'NO MATCH',COMMISSION FROM SALESMAN
    ->            WHERE NOT CITY = ANY(SELECT CITY FROM CUSTOMER) ORDER BY 2 DESC;
+-------------+--------+-----------+------------+
| SALESMAN_ID | NAME   | CUST_NAME | COMMISSION |
+-------------+--------+-----------+------------+
| 4000        | SMITH  | NO MATCH  | 30%        |
| 2000        | RAVI   | PREETHI   | 20%        |
| 2000        | RAVI   | CHETHAN   | 20%        |
| 2000        | RAVI   | MAMATHA   | 20%        |
| 3000        | KUMAR  | NO MATCH  | 15%        |
| 1000        | JOHN   | PREETHI   | 25%        |
| 1000        | JOHN   | CHETHAN   | 25%        |
| 1000        | JOHN   | MAMATHA   | 25%        |
| 5000        | HARSHA | NO MATCH  | 15%        |
+-------------+--------+-----------+------------+
9 rows in set (0.00 sec)
```

```
36 •    INSERT INTO CUSTOMER VALUES
37      ('10','PREETHI','BANGALORE','100','1000'),
38      ('11','VIVEK','MANGALORE','300','1000'),
39      ('12','BHASKAR','CHENNAI','400','2000'),
40      ('13','CHETHAN','BANGALORE','200','2000'),
41      ('14','MAMATHA','BANGALORE','400','3000');
42
43
44 •    INSERT INTO ORDERS VALUES
45      ('50', 5000,'2017-05-04','10','1000'),
46      ('51', 450,'2017-01-20','10','2000'),
47      ('52',1000,'2017-02-24','13','2000'),
48      ('53',3500,'2017-04-13','14','3000'),
49      ('54',550,'2017-03-09','12','2000');
```

```sql
CREATE TABLE ORDERS (
        ORD_NO VARCHAR(5),
        PURCHASE_AMT DECIMAL(10, 2),
        ORD_DATE DATE,
        PRIMARY KEY (ORD_NO),
        CUSTOMER_ID VARCHAR(4)REFERENCES CUSTOMER(CUSTOMER_ID) ON DELETE CASCADE,
        SALESMAN_ID VARCHAR(4)REFERENCES SALESMAN (SALESMAN_ID) ON DELETE CASCADE);


    INSERT INTO SALESMAN
    VALUES ('1000','JOHN','BANGALORE','25%'),
           ('2000','RAVI','BANGALORE','20%'),
           ('3000','KUMAR','MYSORE','15%'),
           ('4000','SMITH','DELHI','30%'),
           ('5000','HARSHA','HYDRABAD','15%');
```

```sql
CREATE DATABASE example7;
use example7;

CREATE TABLE SALESMAN (
      SALESMAN_ID VARCHAR(4),
      NAME VARCHAR(20),
      CITY VARCHAR(20),
      COMMISSION VARCHAR(20),
      PRIMARY KEY(SALESMAN_ID));

CREATE TABLE CUSTOMER (
    CUSTOMER_ID VARCHAR(4),
    CUST_NAME VARCHAR(20),
    CITY VARCHAR(20),
    GRADE VARCHAR(3),
    PRIMARY KEY (CUSTOMER_ID),
    SALESMAN_ID VARCHAR(4),
    FOREIGN KEY(SALESMAN_ID)REFERENCES SALESMAN(SALESMAN_ID) ON DELETE SET NULL);

```

④ Find the title of movies and number of stars for each movie that has at least one rating and find the highest number of stars that movies recieved. Sort the result by movie title.

```
SELECT MOV_TITLE, MAX(REV_STARS)
 FROM MOVIES
JOIN RATING USING (MOV_ID)
  GROUP BY MOV_TITLE
HAVING MAX(REV_STARS)>0
  ORDER BY MOV_TITLE;
```

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

```
UPDATE RATING
SET REV_STARS = 5
WHERE MOV_ID IN( SELECT MOV_ID FROM MOVIES
                  WHERE DIR_ID
                IN(SELECT DIR_ID FROM
                   DIRECTOR
                  WHERE DIR_NAME =
                    'STEVEN SPIELBERG'));
```

```
SELECT MOV_TITLE FROM
MOVIES m, MOVIE_CAST mc
WHERE
   m.MOV_ID = mc.MOV_ID
AND
ACT_ID IN (SELECT ACT_ID FROM MOVIE_CAST
            GROUP BY ACT_ID HAVING
              COUNT(ACT_ID)>1)
GROUP BY MOV_TITLE HAVING
    COUNT(*)>1;
```

Q3. List all actors who acted in a movie before 2000 and also in movie after 2015 (use JOIN OPERATION)

```
SELECT ACT_NAME, MOV_TITLE, MOV_YEAR
FROM ACTORS a JOIN
   MOVIE_CAST c
      ON
   a.ACT_ID = c.ACT_ID
JOIN
   MOVIES m
   ON
      m.MOV_ID = c.MOV_ID
WHERE
   m.MOV_YEAR NOT BETWEEN 2000 AND
                          2015;
```

Experiment #8

ACTOR (Act_id, Act_Name, Act_Gender)
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 SQL Queries.

1. List the titles of all movies directed by 'HITCHCOCK'

```
SELECT MOV_TITLE
FROM MOVIES
WHERE DIR_ID IN(
  SELECT DIRID FROM DIRECTOR
    WHERE DIR_NAME = 'HITCHCOK');
```

OR

```
SELECT MOV_TITLE FROM
Movies m, DIRECTOR d
WHERE
  m.DIR_ID = d.DIR_ID
      AND
  d.DIR_NAME = 'HITCHCOCK';
```

Q2 List all the movie names where one or more actors acted in two or more movies?

Q3.

```
mysql> SELECT MOV_TITLE, MAX(REV_STARS)
    -> FROM MOVIES
    -> INNER JOIN RATING USING (MOV_ID) GROUP
    -> BY MOV_TITLE
    -> HAVING MAX(REV_STARS)>0
    -> ORDER BY MOV_TITLE;
+------------+-----------------+
| MOV_TITLE  | MAX(REV_STARS) |
+------------+-----------------+
| AKASH      | 5               |
| BAHUBALI-1 | 2               |
| BAHUBALI-2 | 4               |
| WAR HORSE  | 5               |
+------------+-----------------+
4 rows in set (0.00 sec)

mysql> UPDATE RATING
    -> SET REV_STARS=5
    -> WHERE MOV_ID IN (SELECT MOV_ID FROM MOVIES
    -> WHERE DIR_ID IN (SELECT DIR_ID
    -> FROM DIRECTOR
    -> WHERE DIR_NAME ='STEVEN SPIELBERG'));
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0

mysql> select * from RATING;
+--------+-----------+
| MOV_ID | REV_STARS |
+--------+-----------+
|   1001 | 4         |
|   1002 | 2         |
|   1003 | 5         |
|   1004 | 5         |
+--------+-----------+
4 rows in set (0.00 sec)
```

```
mysql> select MOV_TITLE FROM MOVIES m, DIRECTOR d
    ->     where m.DIR_ID=d.DIR_ID
    ->     AND d.DIR_NAME='HITCHCOCK';
+-----------+
| MOV_TITLE |
+-----------+
| AKASH     |
+-----------+
1 row in set (0.01 sec)

mysql> SELECT MOV_TITLE
    -> FROM MOVIES m, MOVIE_CAST mc
    -> WHERE m.MOV_ID=mc.MOV_ID AND
    -> ACT_ID IN
    -> (SELECT ACT_ID FROM MOVIE_CAST
    -> GROUP BY ACT_ID HAVING COUNT(ACT_ID)>1)
    -> GROUP BY MOV_TITLE HAVING
    -> COUNT(*)>1;
+-------------+
| MOV_TITLE   |
+-------------+
| BAHUBALI-1 |
+-------------+
1 row in set (0.00 sec)

mysql> SELECT ACT_NAME, MOV_TITLE, MOV_YEAR
    -> FROM ACTOR A JOIN
    -> MOVIE_CAST C
    -> ON A.ACT_ID=C.ACT_ID
    -> JOIN MOVIES M
    -> ON C.MOV_ID=M.MOV_ID
    -> WHERE M.MOV_YEAR NOT BETWEEN 2000 AND 2015;
+----------+-------------+----------+
| ACT_NAME | MOV_TITLE   | MOV_YEAR |
+----------+-------------+----------+
| ANUSHKA  | BAHUBALI-2 |     2017 |
+----------+-------------+----------+
1 row in set (0.00 sec)
```

```sql
INSERT  INTO  DIRECTOR
VALUES  (60,'RAJAMOULI','8751611001'),
        (61,'HITCHCOCK','7766138911'),
        (62,'FARAN','9986776531'),
        (63,'STEVEN SPIELBERG','8989776530');

INSERT INTO MOVIES
VALUES (1001,'BAHUBALI-2', 2017,'TELAGU', 60),
       (1002,'BAHUBALI-1', 2015,'TELAGU', 60),
       (1003,'AKASH', 2008,'KANNADA', 61),
       (1004,'WAR HORSE', 2011,'ENGLISH', 63);


INSERT INTO MOVIE_CAST
VALUES (301, 1002,'HEROINE'),
       (301, 1001,'HEROINE'),
       (303, 1003,'HERO'),
       (303, 1002,'GUEST'),
       (304, 1004,'HERO');

```

```sql
CREATE TABLE MOVIE_CAST (
ACT_ID INTEGER,
MOV_ID INTEGER,
ROLE  VARCHAR(10),
PRIMARY KEY(ACT_ID, MOV_ID),
FOREIGN KEY (ACT_ID) REFERENCES ACTOR(ACT_ID),
FOREIGN KEY (MOV_ID) REFERENCES MOVIES (MOV_ID));

CREATE TABLE RATING (
MOV_ID INTEGER,
REV_STARS VARCHAR(25),
PRIMARY KEY(MOV_ID),
FOREIGN KEY (MOV_ID) REFERENCES MOVIES (MOV_ID));

show tables;

INSERT INTO ACTOR
VALUES(301,'ANUSHKA','F'),
      (302,'PRABHAS','M'),
      (303,'PUNITH','M'),
      (304,'JERMY','M');
```

```
3
4  CREATE TABLE ACTOR (
5      ACT_ID INTEGER,
6      ACT_NAME VARCHAR(20),
7      ACT_GENDER ENUM('M','F'),
8      PRIMARY KEY (ACT_ID));
9
10     CREATE TABLE DIRECTOR (
11       DIR_ID INTEGER,
12       DIR_NAME VARCHAR(20),
13       DIR_PHONE VARCHAR(10),
14       PRIMARY KEY (DIR_ID));
15
16       CREATE TABLE MOVIES (
17             MOV_ID INTEGER,
18             MOV_TITLE VARCHAR(25),
19             MOV_YEAR INTEGER,
20             MOV_LANG VARCHAR(12),
21             DIR_ID INTEGER,
22             PRIMARY KEY (MOV_ID),
23             FOREIGN KEY (DIR_ID) REFERENCES DIRECTOR(DIR_ID));
```

```
WHERE FINALIA IS NULL
FOR UPDATE;
OPEN C_I MARKS;
LOOP
FETCH C_IAMARKS INTO C_A, C_B, C_C, C_USN, C_SUBNODE,
      C_SSID;
IF (C_A != C_B) THEN
 SET C_SUM = C_A + C_B;
  ELSE
  SET C_SUM = C_A + C_C;
  END IF;
SET C_AVG := C_SUM/2;
UPDATE IAMARKS SET FINALIA = C_AVG
WHERE USN = C_USN AND SUBCODE = C_SUBCODE AND
  SSID = C_SSID;
END LOOP;
CLOSE C_IAMARKS;
END
//
```

3. Create a view of Test1 marks of students USN '1BI15CS101' in all subjects.

```
CREATE VIEW STU_TEST1_MARKS_VIEW AS
SELECT TEST1, SUBCODE
FROM IAMARKS
WHERE USN = '1RN13CS091';
```

(4) Calculate the FinalIA (average of best two test marks) and update the corresponding table for all students.

```
Delimiter //
CREATE PROCEDURE AVG_MARKS()
BEGIN
DECLARE C_A NUMBER;
DECLARE C_B NUMBER;
DECLARE C_C INTEGER;
DECLARE C_SUM INTEGER;
DECLARE C_AVG INTEGER;
DECLARE C_USN VARCHAR(10);
DECLARE C_SSID VARCHAR(5);
DECLARE C_IAMARKS CURSOR FOR
DECLARE C_SUBCODE VARCHAR(8);
SELECT GREATEST(Test1, Test2) AS A
       GREATEST(Test1, TEST3) AS B
       GREATEST(Test2, TEST3) AS C
FROM IAMARKS;
```

Experiment #9.

STUDENT (USN, SName, Address, Phone, Gender)
SEMSEC (SSID, Sem, Sec)
CLASS (USN, SSID)
SUBJECT (Subcode, Title, Sem, Credits)
IAMARKS (USN, Subcode, SSID, Test1, Test2, Test3, Final1A)

Write SQL Queries

01. List all the student details studying in fourth semester 'C' section.

```
SELECT S.*, SS.Sem, SS.Sec
FROM STUDENT S, SEMSEC SS, CLASS C.
WHERE S.USN = C.USN
 AND
SS.SSID = C.SSID
 AND
SS.SEM = 4 AND
SS.SEC;
```

2 Compute the total numbes of male and female students in each semester and in each section.

```
SELECT SS.SEM, SS.SEC, S.GENDER, COUNT(S.GENDER)
AS COUNT FROM STUDENT S, SECSEM SS, CLASS C
WHERE S.USN = C.USN
 AND
SS.SSID = C.SSID
GROUP BY SS.SEM, SS.SEC, S.GENDER
ORDER BY SEM;
```