A. Consider the following schema for a Library Database:

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. Retrieve details of all books in the library – id, title, name of publisher, authors, numberof copies in each branch, etc.

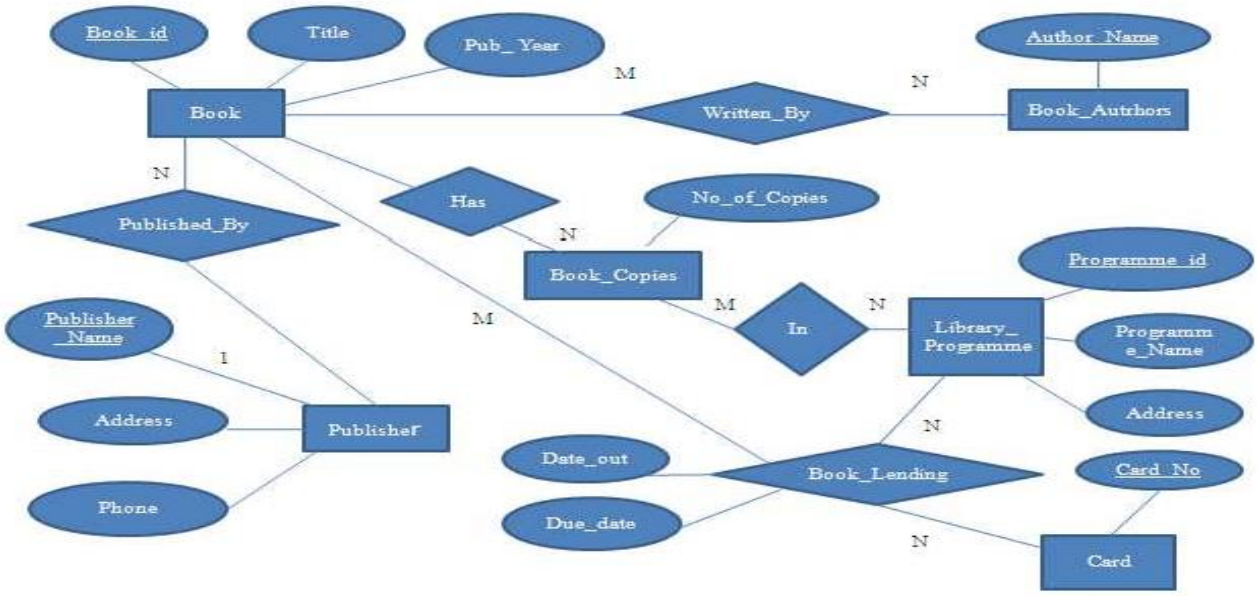
2. Get the particulars of borrowers who have borrowed more than 3 books, but from Jan2017 to Jun 2017

3. Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation.

4. Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.

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

ER Diagram:



Schema Diagram:

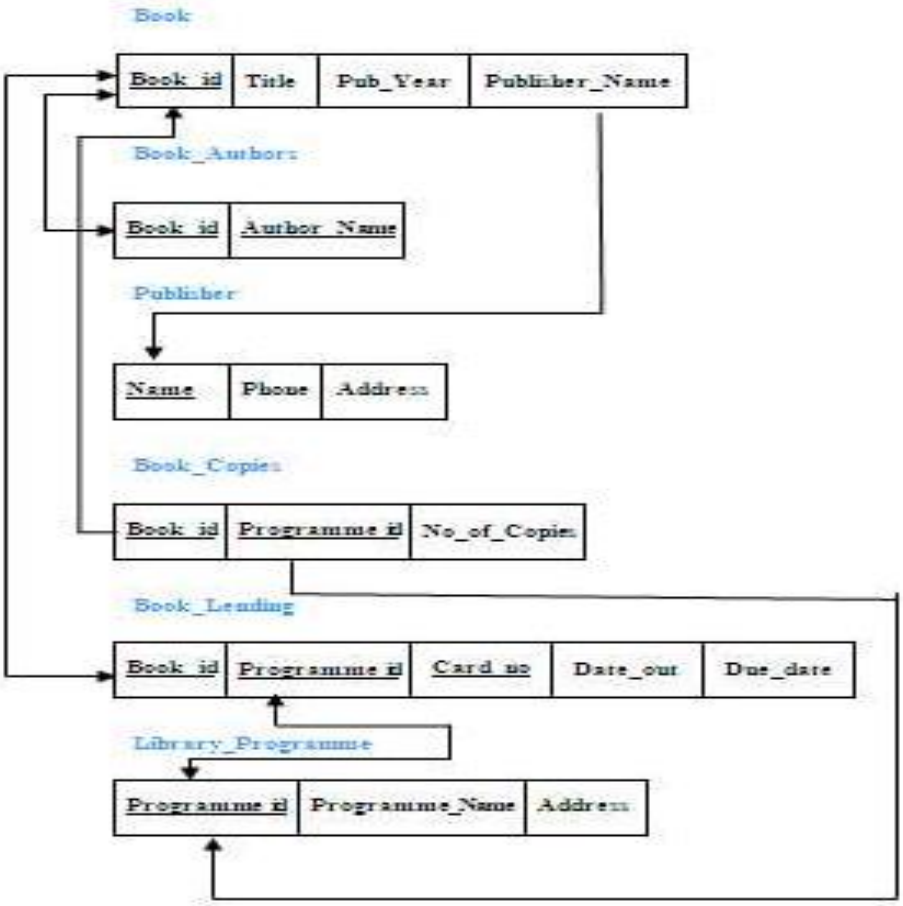


Table Creation

```
CREATE TABLE BOOK (
BOOK_ID INT (10) 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 INT (10),
PRIMARY KEY (BOOK_ID, AUTHOR_NAME),
FOREIGN KEY (BOOK_ID) REFERENCES BOOK (BOOK_ID) ON DELETE CASCADE);

CREATE TABLE PUBLISHER (
NAME VARCHAR (20) PRIMARY KEY,
PHONE BIGINT (20),
ADDRESS VARCHAR (100));

CREATE TABLE BOOK_COPIES (
NO_OF_COPIES INT (5),
BOOK_ID INT (10),
BRANCH _ID INT (10),
PRIMARY KEY (BOOK_ID, BRANCH _ID),
FOREIGN KEY (BOOK_ID) REFERENCES BOOK (BOOK_ID) ON DELETE CASCADE,
FOREIGN KEY (BRANCH_ID) REFERENCES LIBRARY_ BRANCH (BRANCH _ID) ON
DELETE CASCADE);

CREATE TABLE BOOK_LENDING (
DATE_OUT DATE,
DUE_DATE DATE,
BOOK_ID INT (10),
BRANCH_ID INT (10),
CARD_NO INT (10),
PRIMARY KEY (BOOK_ID,BRANCH_ID, CARD_NO),
FOREIGN KEY (BOOK_ID) REFERENCES BOOK (BOOK_ID) ON DELETE CASCADE,
FOREIGN KEY (BRANCH_ID) REFERENCES LIBRARY_BRANCH(BRANCH_ID) ON
DELETE CASCADE,
FOREIGN KEY (CARD_NO) REFERENCES CARD (CARD_NO) ON DELETE CASCADE);

CREATE TABLE CARD
 (CARD_NO INT (10) PRIMARY KEY);
```

CREATE TABLE LIBRARY_BRANCH (
BRANCH_ID INT (10) PRIMARY KEY,
BRANCH_NAME VARCHAR (50),
ADDRESS VARCHAR (100));

Table Descriptions

DESC BOOK;

```
mysql> DESC BOOK;
+----------------+--------------+------+-----+---------+-------+
| Field          | Type         | Null | Key | Default | Extra |
+----------------+--------------+------+-----+---------+-------+
| BOOK_ID        | int(10)      | NO   | PRI | NULL    |       |
| TITLE          | varchar(20)  | YES  |     | NULL    |       |
| PUB_YEAR       | varchar(20)  | YES  |     | NULL    |       |
| PUBLISHER_NAME | varchar(20)  | YES  | MUL | NULL    |       |
+----------------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

DESC BOOK_AUTHORS;

```
mysql> DESC BOOK_AUTHORS;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| AUTHOR_NAME | varchar(20)  | NO   | PRI |         |       |
| BOOK_ID     | int(10)      | NO   | PRI | 0       |       |
+-------------+--------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

DESC PUBLISHER;

```
mysql> DESC PUBLISHER;
+---------+---------------+------+-----+---------+-------+
| Field   | Type          | Null | Key | Default | Extra |
+---------+---------------+------+-----+---------+-------+
| NAME    | varchar(20)   | NO   | PRI | NULL    |       |
| PHONE   | bigint(20)    | YES  |     | NULL    |       |
| ADDRESS | varchar(100)  | YES  |     | NULL    |       |
+---------+---------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

DESC BOOK_COPIES;

```
mysql> DESC  BOOK_COPIES;
+-------------+---------+------+-----+---------+-------+
| Field       | Type    | Null | Key | Default | Extra |
+-------------+---------+------+-----+---------+-------+
| NO_OF_COPIES | int(5)  | YES  |     | NULL    |       |
| BOOK_ID     | int(10) | NO   | PRI | NULL    |       |
| PROGRAMME_ID | int(10) | NO   | PRI | NULL    |       |
+-------------+---------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql>
```

DESC BOOK_LENDING;

```
mysql>  DESC  BOOK_LENDING;
+---------------+----------+------+-----+----------+-------+
| Field         | Type     | Null | Key | Default  | Extra |
+---------------+----------+------+-----+----------+-------+
| DATE_OUT      | date     | YES  |     | NULL     |       |
| DUE_DATE      | date     | YES  |     | NULL     |       |
| BOOK_ID       | int(10)  | NO   | PRI | NULL     |       |
| PROGRAMME_ID  | int(10)  | NO   | PRI | NULL     |       |
| CARD_NO       | int(10)  | NO   | PRI | NULL     |       |
+---------------+----------+------+-----+----------+-------+
5 rows in set (0.03 sec)

mysql>
```

DESC CARD;

```
mysql> DESC  CARD;
+---------+----------+------+-----+----------+-------+
| Field   | Type     | Null | Key | Default  | Extra |
+---------+----------+------+-----+----------+-------+
| CARD_NO | int(10)  | NO   | PRI | NULL     |       |
+---------+----------+------+-----+----------+-------+
1 row in set (0.00 sec)

mysql> _
```

DESC LIBRARY_PROGRAMME

```
mysql> DESC  LIBRARY_PROGRAMME;
+----------------+--------------+------+-----+----------+-------+
| Field          | Type         | Null | Key | Default  | Extra |
+----------------+--------------+------+-----+----------+-------+
| PROGRAMME_ID   | int(10)      | NO   | PRI | NULL     |       |
| PROGRAMME_NAME | varchar(50)  | YES  |     | NULL     |       |
| ADDRESS        | varchar(100) | YES  |     | NULL     |       |
+----------------+--------------+------+-----+----------+-------+
3 rows in set (0.00 sec)

mysql> _
```

Insertion of Values to Tables

```
INSERT INTO BOOK VALUES (1,'DBMS','JAN-2017', 'MCGRAW-HILL');
INSERT INTO BOOK VALUES (2,'ADBMS','JUN-2016','MCGRAW-HILL');
INSERT INTO BOOK VALUES (3, 'CD','SEP-2016','PEARSON');
INSERT INTO BOOK VALUES (4,' ALGORITHMS ','SEP-2015',' MIT');
INSERT INTO BOOK VALUES (5,'OS','MAY-2016','PEARSON');

INSERT INTO  BOOK_AUTHORS  VALUES ('NAVATHE', 1);
INSERT INTO  BOOK_AUTHORS  VALUES ('NAVATHE', 2);
INSERT INTO  BOOK_AUTHORS  VALUES ('ULLMAN',3);
INSERT INTO  BOOK_AUTHORS  VALUES ('CHARLES', 4);
INSERT INTO  BOOK_AUTHORS  VALUES('GALVIN',  5);


INSERT INTO   PUBLISHER VALUES ('MCGRAW-HILL', 9989076587,'BANGALORE');
INSERT INTO   PUBLISHER VALUES ('PEARSON', 9889076565,'NEWDELHI');
INSERT INTO   PUBLISHER VALUES ('PRENTICE HALL', 7455679345,'HYEDRABAD');
INSERT INTO   PUBLISHER VALUES ('WILEY', 8970862340,'CHENNAI');
INSERT INTO    PUBLISHER VALUES ('MIT',7756120238,'BANGALORE');

INSERT INTO BOOK_COPIES  VALUES (10, 1, 10);
INSERT INTO BOOK_COPIES  VALUES (5, 1, 11);
INSERT INTO BOOK_COPIES  VALUES (2, 2, 12);
INSERT INTO BOOK_COPIES  VALUES (5, 2, 13);
INSERT INTO BOOK_COPIES  VALUES (7, 3, 14);
INSERT INTO BOOK_COPIES  VALUES (1, 5, 10);
INSERT INTO BOOK_COPIES  VALUES (3, 4, 11);

INSERT INTO BOOK_LENDING  VALUES ('2017-01-01','2017-06-01', 1, 10, 101);
INSERT INTO BOOK_LENDING  VALUES ('2017-01-11 ','2017-03-11', 3, 14, 101);
INSERT INTO BOOK_LENDING  VALUES ('2017-02-21','2017-04-21', 2, 13, 101);
INSERT INTO BOOK_LENDING  VALUES ('2017-03-15 ','2017-07-15', 4, 11, 101);
INSERT INTO BOOK_LENDING  VALUES ('2017-04-12','2017-05-12', 1, 11, 104);

INSERT INTO CARD VALUES (100);
INSERT INTO CARD VALUES (101);
INSERT INTO CARD VALUES (102);
INSERT INTO CARD VALUES (103);
INSERT INTO CARD VALUES (104);

INSERT INTO  LIBRARY_BRANCH   VALUES (10,'VIJAY NAGAR','MYSURU');
INSERT INTO  LIBRARY_ BRANCH  VALUES (11,'VIDYANAGAR','HUBLI');
INSERT INTO  LIBRARY_ BRANCH  VALUES(12,'KUVEMPUNAGAR','MYSURU');
INSERT INTO  LIBRARY_ BRANCH  VALUE(13,'RAJAJINAGAR','BANGALORE');
INSERT INTO  LIBRARY_ BRANCH VALUES (14,'MANIPAL','UDUPI');
```

SELECT * FROM BOOK;

BOOK_ID	TITLE	PUB_YEAR	PUBLISHER_NAME
1	DBMS	Jan-2017	MCGRAW-HILL
2	ADBMS	Jun-2017	MCGRAW-HILL
3	CD	Sep-2016	PEARSON
4	ALGORITHMS	Sep-2015	MIT
5	OS	May-2016	PEARSON

SELECT * FROM BOOK_AUTHORS;

AUTHOR_NAME	BOOK_ID
NAVATHE	1
NAVATHE	2
ULLMAN	3
CHARLES	4
GALVIN	5

SELECT * FROM PUBLISHER;

NAME	PHONE	ADDRESS
MCGRAW-HILL	9989076587	BANGALORE
MIT	7756120238	BANGALORE
PEARSON	9889076565	NEWDELHI
PRENTICE HALL	7455679345	HYEDRABAD
WILEY	8970862340	CHENNAI

SELECT * FROM BOOK_COPIES;

NO_OF_COPIES	BOOK_ID	BRANCH _ID
10	1	10
5	1	11
2	2	12
5	2	13
7	3	14
1	5	10
3	4	11

SELECT * FROM BOOK_LENDING;

DATEOUT	DUEDATE	BOOKID	BRANCH _ID	CARD_NO
2017-01-01	2017-06-01	1	10	
2017-01-11	2017-03-11	3	4	101
2017-02-21	2017-04-21	2	13	101
2017-03-15	2017-07-15	4	11	101
2017-04-12	2017-05-12	1	11	104

SELECT * FROM CARD;

CARDNO
101
102
103
104
105

SELECT * FROM LIBRARY_BRANCH;

BRANCH _ID	BRANCH _NAME	ADDRESS
10	VIJAY NAGAR	MYSURU
11	VIDYANAGAR	HUBLI
12	KUVEMPUNAGAR	MYSURU
13	RAJAJINAGAR	BANGALORE
14	MANIPAL	UDUPI

Queries:

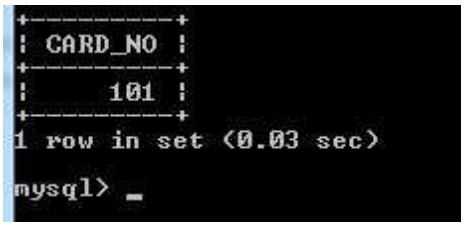
1. **Retrieve details of all books in the library– id, title, name of publisher, authors, number of copies in each branch, etc.**

SELECT B.BOOK_ID, B.TITLE, B.PUBLISHER_NAME, A.AUTHOR_NAME, C.NO_OF_COPIES, L.PROGRAMME_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.PROGRAMME_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	CD	PEARSON	ULLMAN	7	14
4	ALGORITHMS	MIT	CHARLES	1	11
5	OS	PEARSON	GALVIN	3	10

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

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



3. **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;

```
mysql> SELECT * FROM BOOK;
+---------+------------+----------+----------------+
| BOOK_ID | TITLE      | PUB_YEAR | PUBLISHER_NAME |
+---------+------------+----------+----------------+
|       1 | DBMS       | JAN-2017 | MCGRAW-HILL    |
|       2 | ADBMS      | JUN-2016 | MCGRAW-HILL    |
|       3 | CD         | SEP-2016 | PEARSON        |
|       4 | ALGORITHMS | SEP-2015 | MIT            |
|       5 | OS         | MAY-2016 | PEARSON        |
+---------+------------+----------+----------------+
5 rows in set (0.00 sec)

mysql> DELETE FROM BOOK WHERE BOOK_ID=3;
Query OK, 1 row affected (0.03 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 | ALGORITHMS | SEP-2015 | MIT            |
|       5 | OS         | MAY-2016 | PEARSON        |
+---------+------------+----------+----------------+
4 rows in set (0.00 sec)
```

4. **Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.**

CREATE VIEW VW_PUBLICATION AS
SELECT PUB_YEAR
FROM BOOK;
SELECT * FROM VW_PUBLICATION

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

5. **Create a view of all books and its number of copies that are currently available in the Library.**

CREATE VIEW VW_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;

SELECT * FROM VW_BOOKS;

```
mysql> SHOW TABLES;
+-------------------+
| Tables_in_library |
+-------------------+
| book              |
| book_authors      |
| book_copies       |
| book_lending      |
| card              |
| library_programme |
| publisher         |
| vw_books          |
+-------------------+
8 rows in set (0.00 sec)

mysql> SELECT * FROM  VW_BOOKS;
+---------+--------------+--------------+
| BOOK_ID | TITLE        | NO_OF_COPIES |
+---------+--------------+--------------+
|       1 | DBMS         |           10 |
|       1 | DBMS         |            5 |
|       2 | ADBMS        |            2 |
|       2 | ADBMS        |            5 |
|       3 | CD           |            7 |
|       5 | OS           |            1 |
|       4 |  ALGORITHMS  |            3 |
+---------+--------------+--------------+
7 rows in set (0.00 sec)

mysql>
```

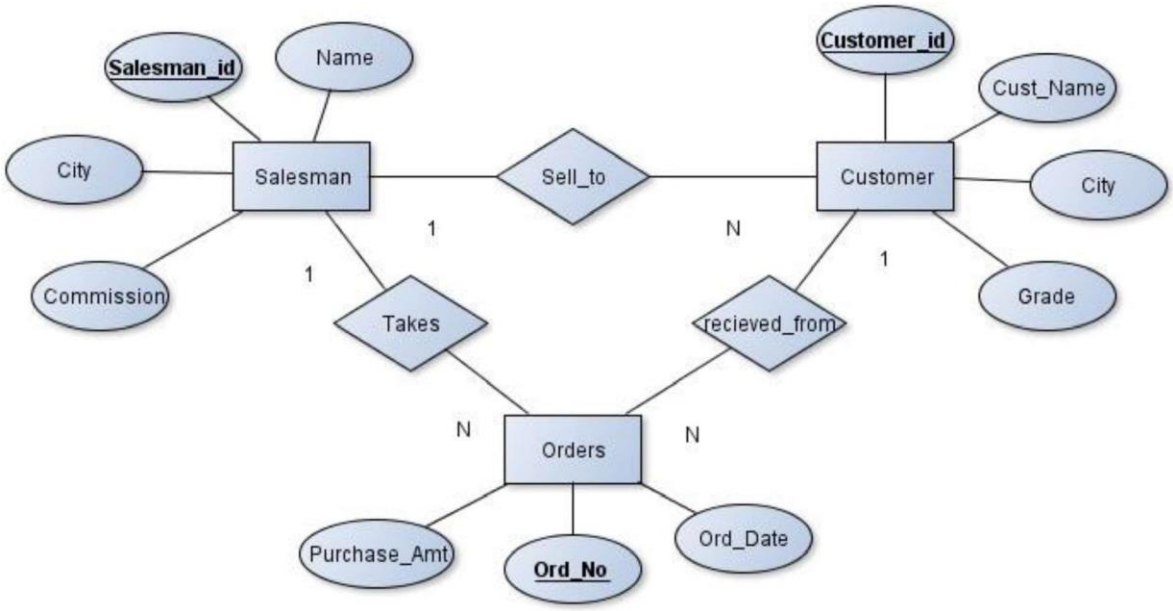
Consider the following schema for Order Database:

SALESMAN (<u>Salesman_id</u>, Name, City, Commission)
CUSTOMER (<u>Customer_id</u>, Cust_Name, City, Grade, Salesman_id)
ORDERS (<u>Ord_No</u>, Purchase_Amt, Ord_Date, Customer_id, Salesman_id)

Write SQL queries to

1. **Count the customers with grades above Bangalore's average.**
2. **Find the name and numbers of all salesmen who had more than one customer.**
3. **List all salesmen and indicate those who have and don't have customers in their cities (Use UNION operation.)**
4. **Create a view that finds the salesman who has the customer with the highest order of a day.**
5. **Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted.**

Solution:
Entity-Relationship Diagram



Schema Diagram

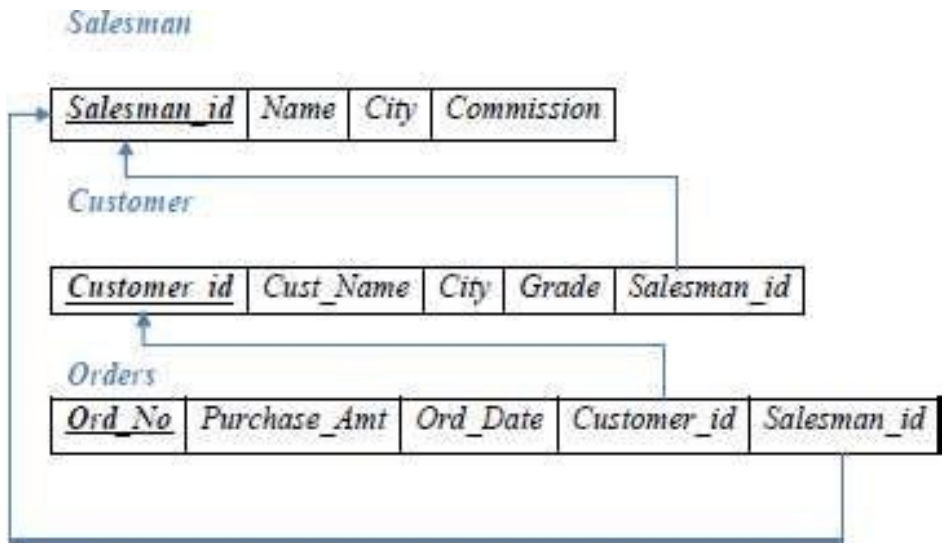


Table Creation

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

CREATE TABLE CUSTOMER (
CUSTOMER_ID INT (5) PRIMARY KEY,
CUST_NAME VARCHAR (20),
CITY VARCHAR (20), GRADE INT (4),
SALESMAN_ID INT (6),
FOREIGN KEY (SALESMAN_ID) REFERENCES SALESMAN (SALESMAN_ID) ON DELETE
SET NULL);

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

Table Descriptions

DESC SALESMAN;

```
mysql> DESC SALESMAN;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| SALESMAN_ID | int(4)       | NO   | PRI | NULL    |       |
| NAME        | varchar(20)  | YES  |     | NULL    |       |
| CITY        | varchar(20)  | YES  |     | NULL    |       |
| COMMISSION  | varchar(20)  | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

DESC CUSTOMER;

```
mysql> DESC CUSTOMER;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| CUSTOMER_ID | int(5)      | NO   | PRI | NULL    |       |
| CUST_NAME   | varchar(20) | YES  |     | NULL    |       |
| CITY        | varchar(20) | YES  |     | NULL    |       |
| GRADE       | int(4)      | YES  |     | NULL    |       |
| SALESMAN_ID | int(6)      | YES  | MUL | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

DESC ORDERS;

```
mysql> DESC ORDERS;
+--------------+---------------+------+-----+---------+-------+
| Field        | Type          | Null | Key | Default | Extra |
+--------------+---------------+------+-----+---------+-------+
| ORD_NO       | int(5)        | NO   | PRI | NULL    |       |
| PURCHASE_AMT | decimal(10,2) | YES  |     | NULL    |       |
| ORD_DATE     | date          | YES  |     | NULL    |       |
| CUSTOMER_ID  | int(4)        | YES  | MUL | NULL    |       |
| SALESMAN_ID  | int(4)        | YES  | MUL | NULL    |       |
+--------------+---------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

INSERT INTO SALESMAN VALUES(101,'RICHARD','LOS ANGELES','18%');
INSERT INTO SALESMAN VALUES(103,'GEORGE','NEWYORK','32%');
INSERT INTO SALESMAN VALUES(110,'CHARLES','BANGALORE','54%');
INSERT INTO SALESMAN VALUES(122,'ROWLING','PHILADELPHIA','46%');
INSERT INTO SALESMAN VALUES(126,'KURT','CHICAGO','52%');
INSERT INTO SALESMAN VALUES(132,'EDWIN','PHOENIX','41%');

INSERT INTO CUSTOMER VALUES(501,'SMITH','LOS ANGELES',10,103);
INSERT INTO CUSTOMER VALUES(510,'BROWN','ATLANTA',14,122);
INSERT INTO CUSTOMER VALUES(522,'LEWIS','BANGALORE',10,132);
INSERT INTO CUSTOMER VALUES(534,'PHILIPS','BOSTON',17,103);
INSERT INTO CUSTOMER VALUES(543,'EDWARD','BANGALORE',14,110);
INSERT INTO CUSTOMER VALUES(550,'PARKER','ATLANTA',19,126);

INSERT INTO ORDERS VALUES(1,1000, '2017-05-04',501,103);
INSERT INTO ORDERS VALUES(2,4000,'2017-01-20',522,132);
INSERT INTO ORDERS VALUES(3,2500, '2017-02-24',550,126);
INSERT INTO ORDERS VALUES(5,6000,'2017-04-13',522,103);
INSERT INTO ORDERS VALUES(6,7000, '2017-03-09',550,126);
INSERT INTO ORDERS VALUES (7,3400,'2017-01-20',501,122);

SELECT * FROM SALESMAN;

```
mysql> SELECT * FROM SALESMAN;
+-------------+----------+---------------+-------------+
| SALESMAN_ID | NAME     | CITY          | COMMISSION  |
+-------------+----------+---------------+-------------+
|         101 | RICHARD  | LOS ANGELES   | 18%         |
|         103 | GEORGE   | NEWYORK       | 32%         |
|         110 | CHARLES  | BANGALORE     | 54%         |
|         122 | ROWLING  | PHILADELPHIA  | 46%         |
|         126 | KURT     | CHICAGO       | 52%         |
|         132 | EDWIN    | PHOENIX       | 41%         |
+-------------+----------+---------------+-------------+
6 rows in set (0.00 sec)
```

SELECT * FROM CUSTOMER;

```
mysql> SELECT * FROM CUSTOMER;
+-------------+-----------+-------------+-------+-------------+
| CUSTOMER_ID | CUST_NAME | CITY        | GRADE | SALESMAN_ID |
+-------------+-----------+-------------+-------+-------------+
|         501 | SMITH     | LOS ANGELES |    10 |         103 |
|         510 | BROWN     | ATLANTA     |    14 |         122 |
|         522 | LEWIS     | BANGALORE   |    10 |         132 |
|         534 | PHILIPS   | BOSTON      |    17 |         103 |
|         543 | EDWARD    | BANGALORE   |    14 |         110 |
|         550 | PARKER    | ATLANTA     |    19 |         126 |
+-------------+-----------+-------------+-------+-------------+
6 rows in set (0.00 sec)
```

SELECT * FROM ORDERS;

```
mysql> select * from orders;
+--------+--------------+------------+-------------+-------------+
| ORD_NO | PURCHASE_AMT | ORD_DATE   | CUSTOMER_ID | SALESMAN_ID |
+--------+--------------+------------+-------------+-------------+
|      1 |      1000.00 | 2017-05-04 |         501 |         103 |
|      2 |      4000.00 | 2017-01-20 |         522 |         132 |
|      3 |      2500.00 | 2017-02-24 |         550 |         126 |
|      5 |      6000.00 | 2017-04-13 |         522 |         103 |
|      6 |      7000.00 | 2017-03-09 |         550 |         126 |
|      7 |      3400.00 | 2017-01-20 |         501 |         122 |
+--------+--------------+------------+-------------+-------------+
6 rows in set (0.00 sec)
```

Queries

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

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

GRADE	COUNT(DISTINCT CUSTOMER_ID)
14	2
17	1
19	1

3 rows in set (0.03 sec)

2. Find the name and numbers of all salesmen who had more than one customer.

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

OR

```
SELECT S.SALESMAN_ID,NAME, FROM CUSTOMER
C,SALESMAN S WHERE
S.SALESMAN_ID=C.SALESMAN_ID GROUP BY
C.SALESMAN_ID HAVING COUNT(*)>1
```

SALESMAN_ID	NAME
103	GEORGE

1 row in set (0.00 sec)

3. List all salesmen and indicate those who have and don't have customers in their cities (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;
```

SALESMAN_ID	NAME	CUST_NAME	COMMISSION
122	ROWLING	NO MATCH	46%
101	RICHARD	SMITH	18%
126	KURT	NO MATCH	52%
103	GEORGE	NO MATCH	32%
132	EDWIN	NO MATCH	41%
110	CHARLES	LEWIS	54%
110	CHARLES	EDWARD	54%

7 rows in set (0.03 sec)

4. **Create a view that finds the salesman who has the customer with the highest order of a day.**
   CREATE VIEW VW_ELITSALESMAN 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); SELECT *

```
mysql> SELECT * FROM  VW_ELITSALESMAN;
+-------------+---------------+---------+
| ORD_DATE    | SALESMAN_ID   | NAME    |
+-------------+---------------+---------+
| 2017-05-04  |           103 | GEORGE  |
| 2017-01-20  |           132 | EDWIN   |
| 2017-02-24  |           126 | KURT    |
| 2017-04-13  |           103 | GEORGE  |
| 2017-03-09  |           126 | KURT    |
+-------------+---------------+---------+
5 rows in set (0.00 sec)
```

   FROM

   VW_ELITSALESMAN

5. **Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted.**
   Use ON DELETE CASCADE at the end of foreign key definitions while creating child table orders and then execute the following:

   DELETE FROM SALESMAN WHERE SALESMAN_ID=101;

```
mysql> SELECT * FROM SALESMAN;
+-------------+---------+--------------+------------+
| SALESMAN_ID | NAME    | CITY         | COMMISSION |
+-------------+---------+--------------+------------+
|         101 | RICHARD | LOS ANGELES  | 18%        |
|         103 | GEORGE  | NEWYORK      | 32%        |
|         110 | CHARLES | BANGALORE    | 54%        |
|         122 | ROWLING | PHILADELPHIA | 46%        |
|         126 | KURT    | CHICAGO      | 52%        |
|         132 | EDWIN   | PHOENIX      | 41%        |
+-------------+---------+--------------+------------+
6 rows in set (0.02 sec)

mysql> DELETE FROM SALESMAN WHERE SALESMAN_ID=101;
Query OK, 1 row affected (0.02 sec)

mysql> SELECT * FROM SALESMAN;
+-------------+---------+--------------+------------+
| SALESMAN_ID | NAME    | CITY         | COMMISSION |
+-------------+---------+--------------+------------+
|         103 | GEORGE  | NEWYORK      | 32%        |
|         110 | CHARLES | BANGALORE    | 54%        |
|         122 | ROWLING | PHILADELPHIA | 46%        |
|         126 | KURT    | CHICAGO      | 52%        |
|         132 | EDWIN   | PHOENIX      | 41%        |
+-------------+---------+--------------+------------+
5 rows in set (0.00 sec)
```

Consider the schema for Movie Database:
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 to

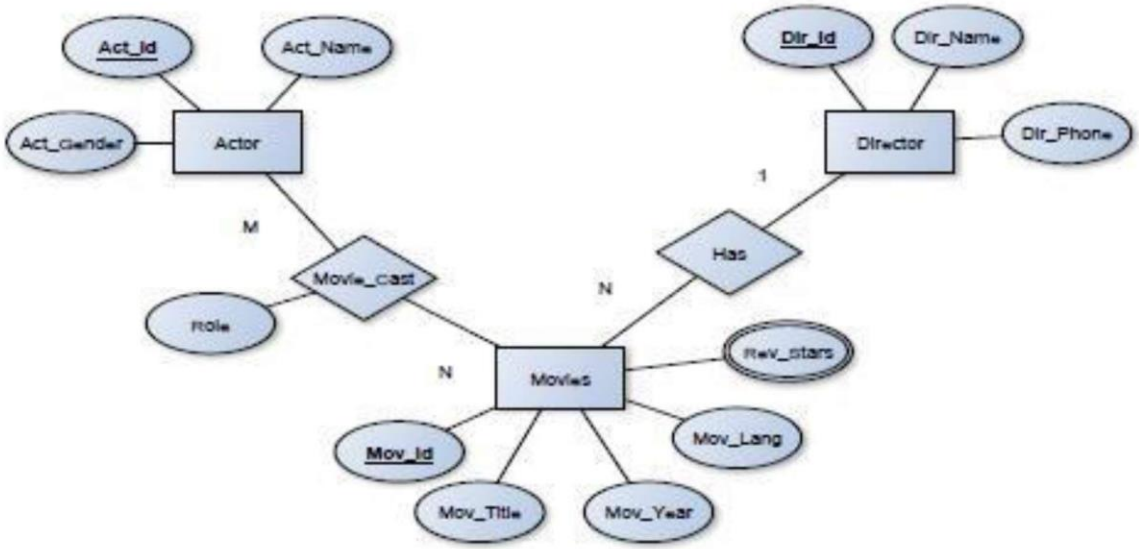
1. **List the titles of all movies directed by'Hitchcock'.**
2. **Find the movie names where one or more actors acted in two or more movies.**
3. **List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).**
4. **Find the title of movies and number of stars for each movie that has at least one rating and find the highest number of stars that movie received. Sort the result by movie title.**
5. **Update rating of all movies directed by 'Steven Spielberg' to 5.**

Program Objectives:

This course will enable students to

- Foundation knowledge in database concepts, technology and practice to groom students into well-informed database application developers.
- Strong practice in SQL programming through a variety of database problems.
- Develop database applications using front-end tools and back-end DBMS.

Solution:
Entity-Relationship Diagram



Schema Diagram

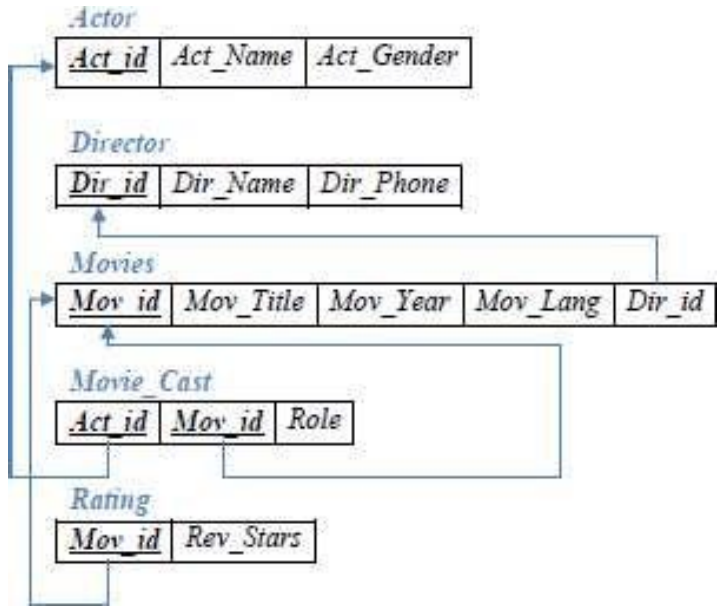


Table Creation

```
CREATE TABLE ACTOR (
ACT_ID INT (5) PRIMARY KEY,
ACT_NAME VARCHAR (20),
ACT_GENDER CHAR (1));

CREATE TABLE DIRECTOR (
DIR_ID INT (5) PRIMARY KEY,
DIR_NAME VARCHAR (20),
DIR_PHONE BIGINT);

CREATE TABLE MOVIES
(MOV_ID INT (4) PRIMARY KEY,
MOV_TITLE VARCHAR (50),
MOV_YEAR INT (4),
MOV_LANG VARCHAR (20),
DIR_ID INT (5),
FOREIGN KEY (DIR_ID) REFERENCES DIRECTOR(DIR_ID));

CREATE TABLE MOVIES_CAST (
ACT_ID INT (5),
MOV_ID INT (5),
ROLE VARCHAR (20),
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 INT (5) PRIMARY KEY,
REV_STARS VARCHAR (25),
FOREIGN KEY (MOV_ID) REFERENCES MOVIES (MOV_ID));
```

Table Descriptions

DESC ACTOR;

```
mysql> DESC ACTOR;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| ACT_ID     | int(5)       | NO   | PRI | NULL    |       |
| ACT_NAME   | varchar(20)  | YES  |     | NULL    |       |
| ACT_GENDER | char(1)      | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

DESC DIRECTOR;

```
mysql> DESC DIRECTOR;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| DIR_ID    | int(5)       | NO   | PRI | NULL    |       |
| DIR_NAME  | varchar(20)  | YES  |     | NULL    |       |
| DIR_PHONE | bigint(20)   | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

DESC MOVIES;

```
mysql> DESC MOVIES;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| MOV_ID   | int(4)       | NO   | PRI | NULL    |       |
| MOV_TITLE | varchar(50)  | YES  |     | NULL    |       |
| MOV_YEAR | int(4)       | YES  |     | NULL    |       |
| MOV_LANG | varchar(20)  | YES  |     | NULL    |       |
| DIR_ID   | int(5)       | YES  | MUL | NULL    |       |
+----------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

DESC MOVIES_CAST;

```
mysql> DESC MOVIES_CAST;
+--------+--------------+------+-----+---------+-------+
| Field  | Type         | Null | Key | Default | Extra |
+--------+--------------+------+-----+---------+-------+
| ACT_ID | int(5)       | NO   | PRI | 0       |       |
| MOV_ID | int(5)       | NO   | PRI | 0       |       |
| ROLE   | varchar(20)  | YES  |     | NULL    |       |
+--------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

DESC RATING;

```
mysql> DESC RATING;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| MOV_ID    | int(5)      | NO   | PRI | NULL    |       |
| REV_STARS | varchar(25) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Insertion of Values to Tables

INSERT INTO ACTOR VALUES (1,'MADHURI DIXIT','F');
INSERT INTO ACTOR VALUES (2,'AAMIR KHAN','M');
INSERT INTO ACTOR VALUES (3,'JUHI CHAWLA','F');
INSERT INTO ACTOR VALUES (4,'SRIDEVI','F');

INSERT INTO DIRECTOR VALUES (100,'SUBHASH KAPOOR', 9563400156);
INSERT INTO DIRECTOR VALUES(102,'ALAN TAYLOR',9971960035);
INSERT INTO DIRECTOR VALUES (103,'SANTHOSH ANANDDRAM', 9934611125);
INSERT INTO DIRECTOR VALUES (104,'IMTIAZ ALI', 8539920975);
INSERT INTO DIRECTOR VALUES (105,'HITCHCOCK',7766138911);
INSERT INTO DIRECTOR VALUES (106,'STEVEN SPIELBERG',9966138934);

INSERT INTO MOVIES VALUES (501,'JAB HARRY MET SEJAL',2017,'HINDI',104);
INSERT INTO MOVIES VALUES (502,'RAJAKUMARA',2017,'KANNADA',103);
INSERT INTO MOVIES VALUES (503,'JOLLY LLB 2', 2013,'HINDI', 100);
INSERT INTO MOVIES VALUES (504,'TERMINATOR GENESYS',2015,'ENGLISH',102);
INSERT INTO MOVIES VALUES (505,'JAWS',1975,'ENGLISH',106);
INSERT INTO MOVIES VALUES (506,'BRIDGE OF SPIES',2015,'ENGLISH', 106);
INSERT INTO MOVIES VALUES (507,'VERTIGO',1943,'ENGLISH',105);
INSERT INTO MOVIES VALUES (508,'SHADOW OF A DOUBT',1943,'ENGLISH', 105);

INSERT INTO MOVIES_CAST VALUES (1, 501,'HEROINE');
INSERT INTO MOVIES_CAST VALUES (1, 502,'HEROINE');
INSERT INTO MOVIES_CAST VALUES (3, 503,'COMEDIAN');
INSERT INTO MOVIES_CAST VALUES (4, 504,'GUEST');
INSERT INTO MOVIES_CAST VALUES (4, 501,'HERO');

INSERT INTO RATING VALUES (501, 4);
INSERT INTO RATING VALUES (502, 2);
INSERT INTO RATING VALUES (503, 5);
INSERT INTO RATING VALUES (504, 4);
INSERT INTO RATING VALUES (505, 3);
INSERT INTO RATING VALUES (506, 2);

SELECT * FROM ACTOR;

ACT_ID	ACT_NAME	ACT
1	MADHURI DIXIT	F
2	AAMIR KHAN	M
3	JUHI CHAWLA	F
4	SRIDEVI	F

SELECT * FROM DIRECTOR;

DIR_ID	DIR_NAME	DIR_PHONE
100	SUBHASH KAPOOR	56340015
102	ALAN TAYLOR	719600310
103	SANTHOSH ANANDDRAM	99346111
104	IMTIAZ ALI	85399209
105	HITCHCOCK	7766138911
106	STEVEN SPIELBERG	9966138934

SELECT * FROM MOVIES;

MOV_ID	MOV_TITLE	MOV_YEAR	MOV_LANG	DIR_ID
501	JAB HARRY MET SEJAL	2017	HINDI	104
502	RAJAKUMARA	2017	KANNADA	103
503	JOLLY LLB 2	2013	HINDI	100
504	TERMINATOR GENESYS	2015	ENGLISH	102
505	JAWS	1975	ENGLISH	106
506	BRIDGE OF SPIES	2015	ENGLISH	106
507	VERTIGO	1958	ENGLISH	105
508	SHADOW OF A DOUBT	1943	ENGLISH	105

SELECT * FROM MOVIE_CAST;

MOV_ID	MOV_TITLE	MOV_YEAR	MOV_LANG	DIR_ID
501	JAB HARRY MET SEJAL	2017	HINDI	104
502	RAJAKUMARA	2017	KANNADA	103
503	JOLLY LLB 2	2013	HINDI	100
504	TERMINATOR GENESYS	2015	ENGLISH	102
505	JAWS	1975	ENGLISH	106
506	BRIDGE OF SPIES	2015	ENGLISH	106
507	VERTIGO	1958	ENGLISH	105
508	SHADOW OF A DOUBT	1943	ENGLISH	105

SELECT * FROM RATING;

MOV_ID	REV_STARS
501	4
502	2
503	5
504	4
505	3
506	2
507	2
508	4

Queries:

1. **List the titles of all movies directed by'Hitchcock'.**
   SELECT MOV_TITLE FROM MOVIES WHERE DIR_ID IN (SELECT DIR_ID FROM DIRECTOR WHERE DIR_NAME = 'HITCHCOCK');
   OR
   SELECT MOV_TITLE FROM MOVIES M, DIRECTOR D WHERE M.DIR_ID=D.DIR_ID AND DIR_NAME='HITCHCOCK';

```
+--------------------+
| MOV_TITLE          |
+--------------------+
| VERTIGO            |
| SHADOW OF A DOUBT  |
+--------------------+
2 rows in set (0.00 sec)
```

2. **Find the movie names where one or more actors acted in two or more movies.**
   SELECT MOV_TITLE FROM MOVIES M,MOVIES_CAST MV
   WHERE M.MOV_ID=MV.MOV_ID AND ACT_ID IN(SELECT ACT_ID FROM MOVIES_CAST GROUP BY ACT_ID HAVING COUNT(ACT_ID)>1) GROUP BY MOV_TITLE HAVING COUNT(*)>1;

```
+----------------------+
| MOV_TITLE            |
+----------------------+
| JAB HARRY MET SEJAL  |
+----------------------+
1 row in set (0.00 sec)
```

3. **List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).**
   SELECT ACT_NAME, MOV_TITLE, MOV_YEAR FROM ACTOR A JOIN MOVIE_CAST C ON A.ACT_ID=C.ACT_ID INNER 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 |
+---------------+----------------------+----------+
| MADHURI DIXIT | JAB HARRY MET SEJAL  |     2017 |
| MADHURI DIXIT | RAJAKUMARA           |     2017 |
| SRIDEVI       | JAB HARRY MET SEJAL  |     2017 |
+---------------+----------------------+----------+
3 rows in set (0.00 sec)
```

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 movie received. Sort the result by movie title.** SELECT MOV_TITLE,MAX(REV_STARS) FROM MOVIES M ,RATING R WHERE M.MOV_ID=R.MOV_ID GROUP BY MOV_TITLE HAVING MAX(REV_STARS)>0 ORDER BY MOV_TITLE;

```
+----------------------+-----------------+
| MOV_TITLE            | MAX(REV_STARS)  |
+----------------------+-----------------+
| BRIDGE OF SPIES      | 2               |
| JAB HARRY MET SEJAL  | 4               |
| JAWS                 | 3               |
| JOLLY LLB 2          | 5               |
| RAJAKUMARA           | 2               |
| TERMINATOR GENESYS   | 4               |
+----------------------+-----------------+
6 rows in set (0.00 sec)
```

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'));

OR

UPDATE RATING R, MOVIES M, DIRECTOR D SET REV_STARS=5 WHERE R.MOV_ID=M.MOV_ID AND M.DIR_ID=D.DIR_ID AND DIR_NAME='STEVEN SPIELBERG';

```
mysql> SELECT * FROM RATING;
+--------+-----------+
| MOV_ID | REV_STARS |
+--------+-----------+
|    501 | 4         |
|    502 | 2         |
|    503 | 5         |
|    504 | 4         |
|    505 | 5         |
|    506 | 5         |
+--------+-----------+
6 rows in set (0.00 sec)
```

Program Outcomes:

The students are able to

- Create, Update and query on the database.
- Demonstrate the working of different concepts of DBMS
- Implement, analyze and evaluate the project developed for an application.