**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**
  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. ** **5.**

**Program Objectives:**

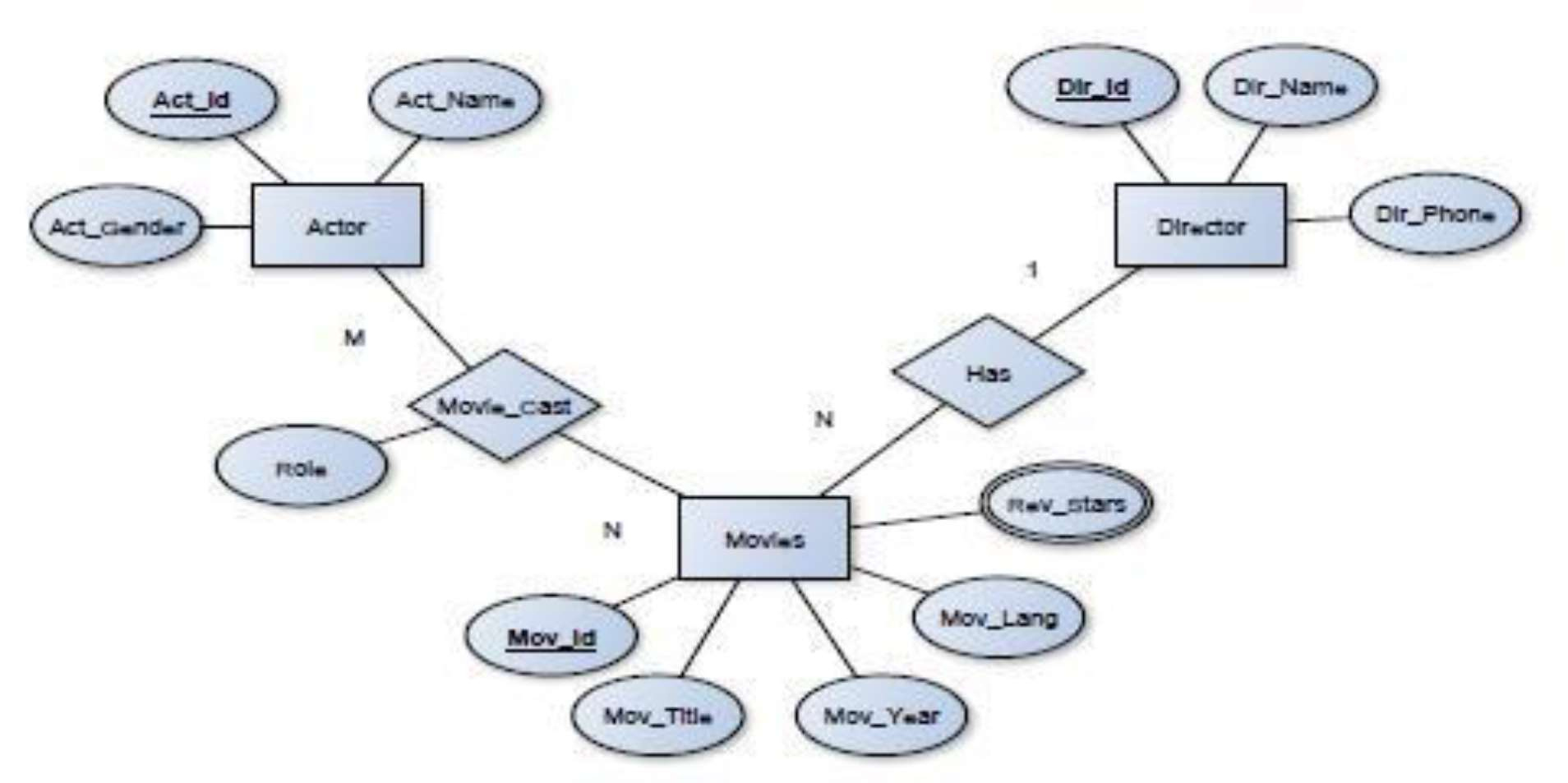
**This course will enable students to**

Foundation knowledge in database concepts, technology and practice to groom students into well-informed database application developers.

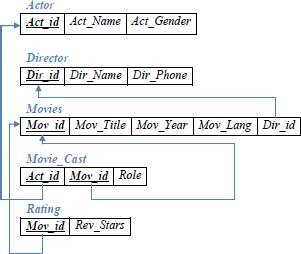
Strong practice in SQL programming through a variety of database problems. Develop database applications using front-end tools and back-end DBMS.

**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;



DESC DIRECTOR;



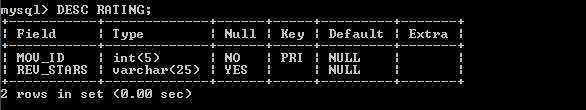
DESC MOVIES;



DESC MOVIES\_CAST;



DESC RATING;



#### Insertion of Values to Tables

INSERT INTO ACTOR VALUES (  INSERT INTO ACTOR VALUES ( INSERT INTO ACTOR VALUES ( INSERT INTO ACTOR VALUES (

 9563400156); INSERT INTO DIRECTOR VALUES(102,'ALAN TAYLOR',9971960035);

75); INSERT INTO DIRECTOR VALUES (105,'HITCHCOCK',7766138911); INSERT INTO DIRECTOR VALUES (106,'STEVEN SPIELBERG',9966138934);

25);

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 IN



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:

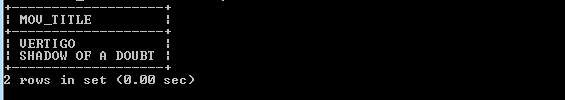
1. **List the titles of all movies directed by**

SELECT MOV\_TITLE FROM MOVIES WHERE DIR\_ID IN (SELECT DIR\_ID FROM



### OR

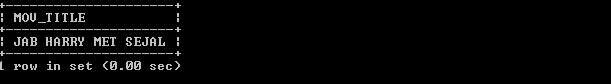
SELECT MOV\_TITLE FROM MOVIES M, DIRECTOR D WHERE M.DIR\_ID=D.DIR\_ID AND DIR\_NAME='HITCHCOCK';



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

SELECT MOV\_TITLE FROM MOVIES M,MOVIES\_CAST MV

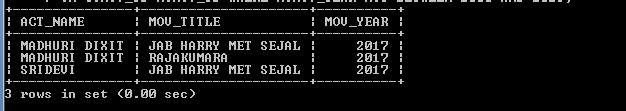
WHERE M.MOV\_ID=MV.MOV\_ID AND ACT\_ID IN(SELECT ACT\_ID FROM MOVIES\_CAST GROUP BY ACT\_ID HAVING COUNT(ACT\_ID)>1) GROUP BY MOV\_TITLE HAVING COUNT(\*)>1;



#### 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;



1. **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;



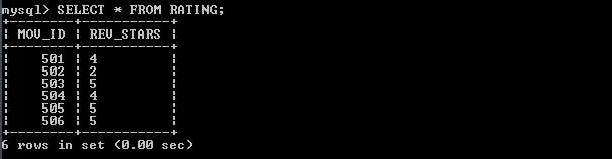
#### 5. 5

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';



**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.