select \* from customer c, book b where c.book\_id = b.book\_idselect \* from customer c join book b on c.book\_id = b.book\_id

**Problem Statement →**

A Movie database has info about various movies,its actors,directors and ratings-

1. You are required to create five tables in the database for

**Actor,Director, Movies, Movie\_Cast and Rating**

**Actor**(**ACT\_ID**,ACT\_NAME,ACT\_GENDER)--3rd

**Director**(**DIR\_ID**,DIR\_NAME,DIR\_PHONE) ---1st

**Movies**(**MOV\_ID**,MOV\_TITLE,MOV\_YEAR,MOV\_LANG,**DIR\_ID**)--2nd

**Movie\_Cast**(ACT\_ID,MOV\_ID,ROLE) - 4th

**Rating**(MOV\_ID,REV\_STARS) --5th

**Director : Movies**

**1 : M**

**Actor : Movies**

**M : M**

**Movie : Rating**

**1 : M**

1. Insert the following data in the tables

Actor-

ACT\_ID ACT\_NAME ACT\_GENDER

1 Anisha f

2 Pavan m

3 Pradeep m

4 George m

Director-

DIR\_ID DIR\_NAME DIR\_PHONE

1 Rohit 1234567890

2 Akash 1345678901

3 Christopher 1456789012

4 Pamela 1567890123

Movies-

MOV\_ID MOV\_TITLE MOV\_YEAR MOV\_LANG DIR\_ID

1 ABCD 2020 Hindi 1

2 BCDA 2019 Telugu 1

3 CDAB 2021 English 2

4 DABC 2018 Kannada 3

5 HHHH 2021 Kannada 4

Movie\_Cast-

ACT\_ID MOV\_ID ROLE

1 1 Actress

1 2 Actress

3 2 Villain

3 3 Actor

4 4 actor

Rating-

MOV\_ID REV\_STARS

1 4

2 2

3 5

4 4

5 0

**Queries →**

Write queries for the following:

1. List the titles of all movies directed by ‘akash’.
2. 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.
3. Update rating of all movies directed by ‘Pamela’ to 5.
4. Create a stored Procedure which will display the movie title, year, language, director, rating of all the movies in their decreasing order of ratings.
5. Use case statement to list movies along with the verdict on the basis of their ratings like if rating is 4/5 then movie is good to watch, if 3 then movie is decent and if rating is 2/1 then not a good movie.

**Solutions →**

1)

CREATE database IF NOT EXISTS `movie\_directory` ;

USE `movie\_directory` ;

CREATE TABLE IF NOT EXISTS `actor` (

`ACT\_ID` INT NOT NULL,

`ACT\_NAME` VARCHAR(20) NULL DEFAULT NULL,

`ACT\_GENDER` CHAR(1) NULL DEFAULT NULL,

PRIMARY KEY (`ACT\_ID`));

CREATE TABLE IF NOT EXISTS `director` (

`DIR\_ID` INT NOT NULL,

`DIR\_NAME` VARCHAR(20) NULL DEFAULT NULL,

`DIR\_PHONE` INT NULL DEFAULT NULL,

PRIMARY KEY (`DIR\_ID`));

CREATE TABLE IF NOT EXISTS `movies` (

`MOV\_ID` INT NOT NULL,

`MOV\_TITLE` VARCHAR(25) NULL DEFAULT NULL,

`MOV\_YEAR` INT NULL DEFAULT NULL,

`MOV\_LANG` VARCHAR(12) NULL DEFAULT NULL,

`DIR\_ID` INT NULL DEFAULT NULL,

PRIMARY KEY (`MOV\_ID`),

FOREIGN KEY (`DIR\_ID`)

REFERENCES `director` (`DIR\_ID`));

CREATE TABLE IF NOT EXISTS `movie\_cast` (

`ACT\_ID` INT NOT NULL,

`MOV\_ID` INT NOT NULL,

`ROLE` VARCHAR(10) NULL DEFAULT NULL,

PRIMARY KEY (`ACT\_ID`, `MOV\_ID`),

FOREIGN KEY (`ACT\_ID`) REFERENCES `actor` (`ACT\_ID`),

FOREIGN KEY (`MOV\_ID`) REFERENCES `movies` (`MOV\_ID`));

CREATE TABLE IF NOT EXISTS `rating` (

`MOV\_ID` INT NOT NULL,

`REV\_STARS` VARCHAR(25) NULL DEFAULT NULL,

PRIMARY KEY (`MOV\_ID`),

FOREIGN KEY (`MOV\_ID`) REFERENCES `customer\_directory`.`movies` (`MOV\_ID`));

2)

INSERT INTO ACTOR VALUES (1,'Anisha','f');

INSERT INTO ACTOR VALUES (2,'Pavan','m');

INSERT INTO ACTOR VALUES (3,'Pradeep','m');

INSERT INTO ACTOR VALUES (4,'George','m');

INSERT INTO DIRECTOR VALUES (1,'Rohit', 1234567890);

INSERT INTO DIRECTOR VALUES (2,'Akash', 1345678901);

INSERT INTO DIRECTOR VALUES (3,'Christopher', 1456789012);

INSERT INTO DIRECTOR VALUES (4,'Pamela', 1567890123);

INSERT INTO MOVIES VALUES (1,'ABCD', 2020, 'Hindi', 1);

INSERT INTO MOVIES VALUES (2,'BCDA', 2019, 'Telugu', 1);

INSERT INTO MOVIES VALUES (3,'CDAB', 2021, 'English', 2);

INSERT INTO MOVIES VALUES (4,'DABC', 2018, 'Kannada', 3);

INSERT INTO MOVIES VALUES (5,'HHHH', 2021, 'Kannada', 4);

-- use to avoid foreign key constraint errors --

SET FOREIGN\_KEY\_CHECKS=0;

INSERT INTO MOVIE\_CAST VALUES (1, 2, 'Actress');

INSERT INTO MOVIE\_CAST VALUES (1, 1, 'Actress');

INSERT INTO MOVIE\_CAST VALUES (3, 3, 'Actor');

INSERT INTO MOVIE\_CAST VALUES (3, 2, 'Villain');

INSERT INTO MOVIE\_CAST VALUES (4, 4, 'actor');

INSERT INTO RATING VALUES (1,4);

INSERT INTO RATING VALUES (2,2);

INSERT INTO RATING VALUES (3, 5);

INSERT INTO RATING VALUES (4, 4);

-- delete from rating where mov\_id=5; --

INSERT INTO RATING VALUES (1,4);

INSERT INTO RATING VALUES (2,2);

INSERT INTO RATING VALUES (3, 5);

INSERT INTO RATING VALUES (4, 4);

INSERT INTO RATING VALUES (5, 0);

3)

-- List the titles of all movies directed by ‘akash’. --

SELECT MOV\_TITLE FROM MOVIES WHERE DIR\_ID IN (SELECT DIR\_ID FROM DIRECTOR WHERE DIR\_NAME = 'akash');

MOV\_TITLE

CDAB

4)

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

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)

ABCD 4

BCDA 2

CDAB 5

DABC 4

5)

-- Update rating of all movies directed by ‘Pamela’ to 5 --

SET SQL\_SAFE\_UPDATES = 0;

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 = 'pamela'));

# MOV\_ID REV\_STARS

1 4

2 2

3 5

4 4

**5 5**

6)

-- stored procedure on movies based on decreasing ratings movie title, year, language, director, rating --

DELIMITER &&

CREATE PROCEDURE lro()

BEGIN

SELECT movies.mov\_title,movies.mov\_year,movies.mov\_lang,director.\*,rating.rev\_stars from movies ,director,rating where movies.dir\_id=director.dir\_id and rating.mov\_id=movies.mov\_id order by rating.rev\_stars;

END &&

DELIMITER ;

call lpro();

mov\_title mov\_year mov\_lang DIR\_ID DIR\_NAME DIR\_PHONE rev\_stars

BCDA 2019 Telugu 1 Rohit 1234567890 2

ABCD 2020 Hindi 1 Rohit 1234567890 4

DABC 2018 Kannada 3 Christopher 1456789012 4

CDAB 2021 English 2 Akash 1345678901 5

HHHH 2021 Kannada 4 Pamela 1567890123 5

7)

-- case statement 4,5 good to watch, 3 decent , 2,1 not a good movie --

SELECT movies.mov\_title,rating.rev\_stars,

CASE

WHEN rating.rev\_stars >3 THEN 'Movie good to watch'

WHEN rating.rev\_stars>2 THEN 'Movie is decent'

ELSE 'Not a good movie'

END AS verdict

FROM movies,rating where rating.mov\_id=movies.mov\_id;

mov\_title rev\_stars verdict

ABCD 4 Movie good to watch

BCDA 2 Not a good movie

CDAB 5 Movie good to watch

DABC 4 Movie good to watch

HHHH 5 Movie good to watch