Report on Lab-04

DATABASE MANAGEMENT SYSTEMS LAB

Submitted by

Adid-Al-Mahamud Shazid

Student Id: 210042172

Department: CSE

Programme: SWE

Course Title: CSE 4308

Submitted to

Zannatun Naim Sristy

Lecturer, Department of CSE

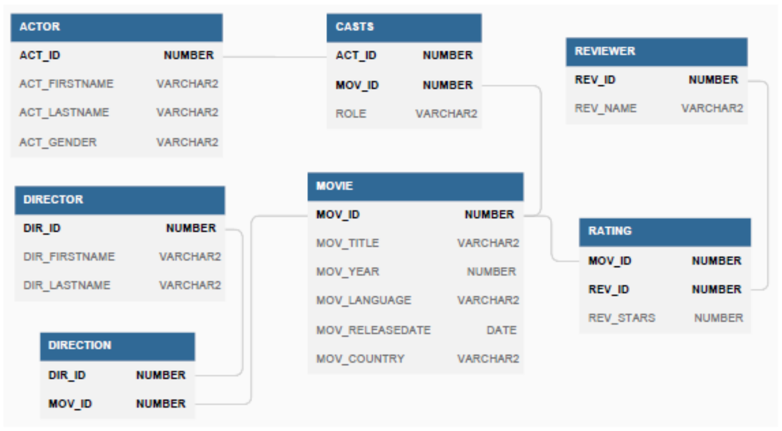
September 07, 2023

Introduction

In this lab class, we were given tasks based on advanced data manipulation techniques to solve using SQL command line to understand the basics of data definition and data manipulation.

Task

Execute the `movie.sql` script using command. It creates a set of tables along with values that maintain the following schema:



Here, the boldfaces denote the primary keys and the arcs denote the foreign key relationships.
In this lab, you have to write all SQL statements in an editor first and save them with .sql extension. Then execute the SQL script.
Write SQL statements for the following queries:

1. Find the name of the actors/actresses that are also directors (with and without set operator).
2. Find the actresses with the same first name.
3. Find the list of all the full names stored in the database.
4. Find the movie titles that did not receive any ratings.
5. Find the average rating of all movies.
6. Find the minimum rating for each movie and display them in descending order of rating.
7. Find the title of the movie having an average rev_star higher than the average rev_star of all the movies.
8. Find the name of actors/actresses and the number of ratings received by the movies in which they played a role.
9. Find the name of the director of the movie having the highest average rev_star.
10. Find all the movie-related information of movies acted and directed by the same person.
11. Find the title and average rating of the movies that have an average rev_star of more than 7.
12. Find the reviewer who gives the highest number of lowest rev_star.
13. Find the name and average runtime of movies of different actors/actresses. Do not include any actor/actress who worked with 'James Cameron'.

Solution

```sql
--TASK 1
SELECT ACT_FIRSTNAME|| ' ' || ACT_LASTNAME AS NAME FROM ACTOR, DIRECTOR WHERE ACT_FIRSTNAME=DIR_FIRSTNAME AND ACT_LASTNAME=DIR_LASTNAME;
SELECT ACT_FIRSTNAME || ' ' || ACT_LASTNAME AS NAME FROM ACTOR INTERSECT SELECT DIR_FIRSTNAME || ' ' || DIR_LASTNAME AS NAME FROM DIRECTOR;

--TASK 2
SELECT ACT_FIRSTNAME FROM ACTOR WHERE ACT_GENDER = 'F' GROUP BY ACT_FIRSTNAME HAVING COUNT(*)>1;

--TASK 3
SELECT ACT_LASTNAME|| ' ' || ACT_LASTNAME AS NAME FROM ACTOR UNION SELECT DIR_FIRSTNAME|| ' ' || DIR_LASTNAME FROM DIRECTOR;

--TASK 4
SELECT MOV_TITLE FROM MOVIE WHERE MOV_ID NOT IN (SELECT MOV_ID FROM RATING);

--TASK 5
SELECT AVG(REV_STARS) FROM RATING;

--TASK 6
SELECT MOV_TITLE, MIN(REV_STARS) AS MIN_RATE FROM RATING NATURAL JOIN MOVIE GROUP BY MOV_TITLE ORDER BY MIN_RATE DESC;

--TASK 7
SELECT MOV_TITLE FROM MOVIE WHERE MOV_ID IN(SELECT MOV_ID FROM RATING GROUP BY MOV_ID HAVING AVG(REV_STARS) > (SELECT AVG(REV_STARS) FROM RATING));

--TASK 8
SELECT ACT_FIRSTNAME, ACT_LASTNAME, COUNT(*) AS number_of_ratings FROM ACTOR, CASTS, RATING WHERE ACTOR.ACT_ID = CASTS.ACT_ID AND CASTS.MOV_ID = RATING.MOV_ID
GROUP BY ACT_FIRSTNAME , ACT_LASTNAME ORDER BY number_of_ratings DESC;

--TASK 9
SELECT DIR_FIRSTNAME,DIR_LASTNAME FROM DIRECTOR WHERE DIR_ID IN (SELECT DIR_ID FROM MOVIE JOIN RATING USING (MOV_ID) GROUP BY DIR_ID
HAVING AVG(REV_STARS) = (SELECT MAX(AVG(REV_STARS)) FROM MOVIE JOIN RATING USING (MOV_ID) GROUP BY DIR_ID));

--TASK 10
SELECT *
```

```sql
FROM MOVIE
WHERE MOV_ID
IN
(
    SELECT DN.MOV_ID
    FROM DIRECTION DN
    WHERE DN.DIR_ID
    IN
    (
        SELECT D.DIR_ID
        FROM DIRECTOR D
        WHERE D.DIR_FIRSTNAME || ' ' || D.DIR_LASTNAME
        IN
        (
            SELECT D1.DIR_FIRSTNAME || ' ' || D1.DIR_LASTNAME AS DIR_NAME1
            FROM DIRECTOR D1
            INTERSECT
            SELECT A.ACT_FIRSTNAME || ' ' || A.ACT_LASTNAME AS ACT_NAME
            FROM ACTOR A
        )
    )
);

--TASK 11
SELECT M.MOV_TITLE,
(
    SELECT avg(R.REV_STARS)
    FROM RATING R
    WHERE R.MOV_ID = M.MOV_ID
    AND R.REV_STARS IS NOT NULL
) AS AVG_RATE
FROM MOVIE M
WHERE M.MOV_ID
IN
(
    SELECT R1.MOV_ID
    FROM RATING R1
    WHERE R1.REV_STARS IS NOT NULL
    GROUP BY R1.MOV_ID
    HAVING avg(R1.REV_STARS) > 7
)
ORDER BY AVG_RATE DESC;

--TASK 12
SELECT R.REV_NAME
```

```sql
FROM REVIEWER R
WHERE R.REV_ID
IN
(
    SELECT RT.REV_ID
    FROM RATING RT
    WHERE RT.REV_STARS =
    (
        SELECT min(REV_STARS)
        FROM RATING
    )
);

--TASK 13
SELECT A.ACT_FIRSTNAME || ' ' ||A.ACT_LASTNAME AS ACT_NAME,
(
    SELECT avg(M.MOV_TIME)
    FROM MOVIE M
    WHERE M.MOV_ID
    IN
    (
        SELECT C.MOV_ID
        FROM CASTS C
        WHERE C.ACT_ID = A.ACT_ID
    )
    AND M.MOV_TIME IS NOT NULL
) AS AVG_RUNTIME
FROM ACTOR A
WHERE A.ACT_ID
IN
(
    SELECT ACT_ID FROM CASTS
)
AND A.ACT_ID
NOT IN
(
    SELECT C2.ACT_ID
    FROM CASTS C2
    WHERE C2.MOV_ID
    IN
    (
        SELECT DN.MOV_ID
        FROM DIRECTION DN
        WHERE DN.DIR_ID =
        (
            SELECT DR.DIR_ID
            FROM DIRECTOR DR
            WHERE DR.DIR_FIRSTNAME = 'James' AND DR.DIR_LASTNAME = 'Cameron'
        )
    )
)
ORDER BY ACT_NAME;
```

Analysis and Explanation

From this task I learnt to use sub queries properly. Also got to know some new functionalities as well.

Difficulties

I faced difficulties while I was trying to use nested queries. It was bit confusing. Thus, it took much time and tries for me to complete the tasks.