

CSCI 5408

**DATA MANAGEMENT AND
WAREHOUSING**

LAB - 1

Table of Contents

1. Check how many unique actors are present in IMDB dataset.....	3
2. Check how many movies are released between the year 1990s till 2000.....	4
3. Find the list of genres of movies directed by Christopher Nola.....	5
4. Find the list of all directors, and the movie name which are ranked between 8 to 9 and have a genre of Sci-Fi and Action.....	6
5. Find the name of the movie in which the actor's role is any doctor, and the movie has the highest number of roles of doctor.....	7
6. Find the list of the movies that start with the letter 'f'.....	8
7. Reference.....	9

Query 1

Problem Statement

Check how many unique actors are present in the IMDB dataset.

SQL Syntax

```
SELECT COUNT(DISTINCT CONCAT(first_name, ' ', last_name)) AS UniqueActors FROM actors;
```

Explanation

- Firstly, we get the first name and last name of the actors from the IMDB.
- After that, I combined 2 fields by using CONCAT function.
- At last, use the DISTINCT to return unique actors.

Output Screenshot

The screenshot shows a SQL query editor with the following code:

```
1  USE dmwa;  
2  
3  •  SELECT COUNT(DISTINCT CONCAT(first_name, ' ', last_name)) AS UniqueActors FROM actors;  
4
```

Below the editor is a toolbar with options: Result Grid, Filter Rows, Export, and Wrap Cell Content. The results are displayed in a table with one column, 'UniqueActors', and one row containing the value '1907'.

UniqueActors
1907

Query 2

Problem Statement

Check how many movies are released between the year 1990s till 2000.

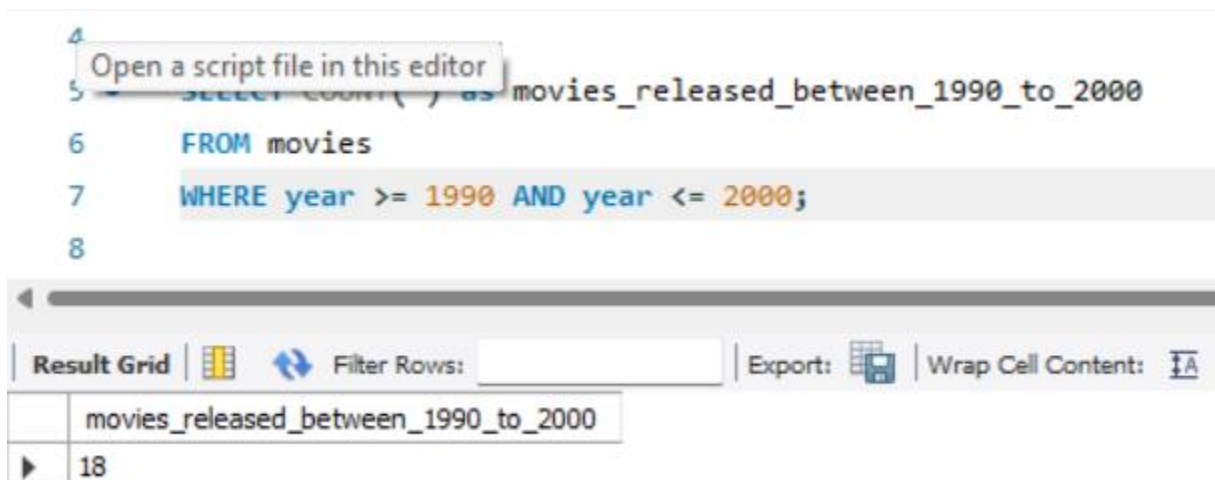
SQL Syntax

```
SELECT COUNT(*) as movies_released_between_1990_to_2000
FROM movies
WHERE year >= 1990 AND year <= 2000;
```

Explanation

- First, use the BETWEEN operator to get the record from the 1990 to 2000.
- Next, use the COUNT to get released movies.

Output Screenshot



Query 3

Problem Statement

Find the list of genres of movies directed by Christopher Nola.

SQL Syntax

```
SELECT movies_genres.genre FROM movies_directors
JOIN directors ON directors.id = movies_directors.director_id
JOIN movies_genres ON movies_genres.movie_id = movies_directors.movie_id
WHERE directors.first_name = 'Christopher' AND directors.last_name = 'Nolan';
```

Explanation

- First, specify the genre column from the genres of the movies table to return.
- Next, do the JOIN on the directors table based on the common field director_id to join the two tables.
- Finally, join the same result with the movies_genres table based on the movie-id and use the condition on the first name and last name field of the directors table to return output.

Output Screenshot

9

```
10 • SELECT movies_genres.genre
11 FROM movies_directors
12 JOIN directors ON directors.id = movies_directors.director_id
13 JOIN movies_genres ON movies_genres.movie_id = movies_directors.movie_id
14 WHERE directors.first_name = 'Christopher' AND directors.last_name = 'Nolan';
15
```

Result Grid

genre
Action
Adventure
Crime
Fantasy
Thriller
Drama
Mystery
Thriller

Query 4

Problem Statement

Find the list of all directors, and the movie name which are ranked between 8 to 9 and have a genre of Sci-Fi and Action

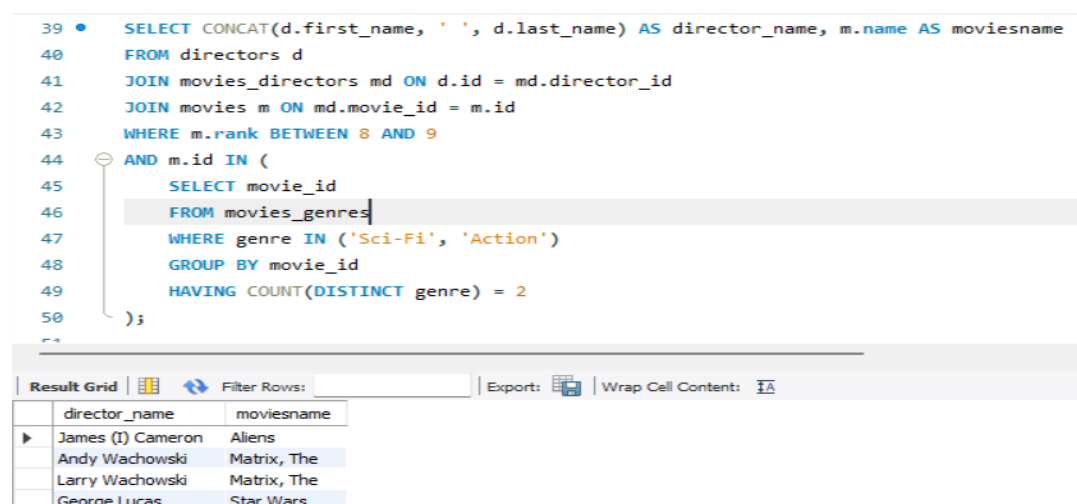
SQL Syntax

```
SELECT CONCAT(d.first_name, ' ', d.last_name) AS director_name, m.name AS moviesname
FROM directors d
JOIN movies_directors md ON d.id = md.director_id
JOIN movies m ON md.movie_id = m.id WHERE m.rank BETWEEN 8 AND 9
AND m.id IN (
    SELECT movie_id
    FROM movies_genres
    WHERE genre IN ('Sci-Fi', 'Action')
    GROUP BY movie_id
    HAVING COUNT(DISTINCT genre) = 2
);
```

Explanation

- First, I concatenate the first and last names of the directors and labeling it as a “moviesname”
- After, we join the table to get the movies names, directors and genres.
- After that, do filter by the given rank and the genres ‘Sci-Fi’ and ‘Action’ and connect movies along with the directors.
- Finally, we ensure the unique combination of the director name and movies name.

Output Screenshot



```
39 • SELECT CONCAT(d.first_name, ' ', d.last_name) AS director_name, m.name AS moviesname
40 FROM directors d
41 JOIN movies_directors md ON d.id = md.director_id
42 JOIN movies m ON md.movie_id = m.id
43 WHERE m.rank BETWEEN 8 AND 9
44 AND m.id IN (
45     SELECT movie_id
46     FROM movies_genres
47     WHERE genre IN ('Sci-Fi', 'Action')
48     GROUP BY movie_id
49     HAVING COUNT(DISTINCT genre) = 2
50 );
```

director_name	moviesname
James (I) Cameron	Aliens
Andy Wachowski	Matrix, The
Larry Wachowski	Matrix, The
George Lucas	Star Wars

Query 5

Problem Statement

Find the name of the movie in which the actor's role is any doctor, and the movie has the highest number of roles of doctor.

SQL Syntax

```
SELECT name FROM ( SELECT MAX(movie_id) AS max_movie_id FROM roles WHERE role
LIKE '%doctor%') AS roles
LEFT JOIN movies
ON movies.id = roles.max_movie_id;
```

Explanation

- First, the highest movie_id is determined from the “roles” table where the role contains the doctor and give alias “roles” to that.
- Then we left join the result with the movies table using movie_id.
- Finally, we retrieve the names of the movies from the movies table that have a matching movie_id.

Output Screenshot

```
27 • SELECT name
28 FROM (
29     SELECT MAX(movie_id) AS max_movie_id
30     FROM roles
31     WHERE role LIKE '%doctor%'
32 ) AS roles
33 LEFT JOIN movies
34 ON movies.id = roles.max_movie_id;
35
```

Result Grid

	name
▶	Vanilla Sky

Filter Rows: Export: Wrap Cell Content:

Query 6

Problem Statement

Find the list of the movies that start with the letter 'f'.

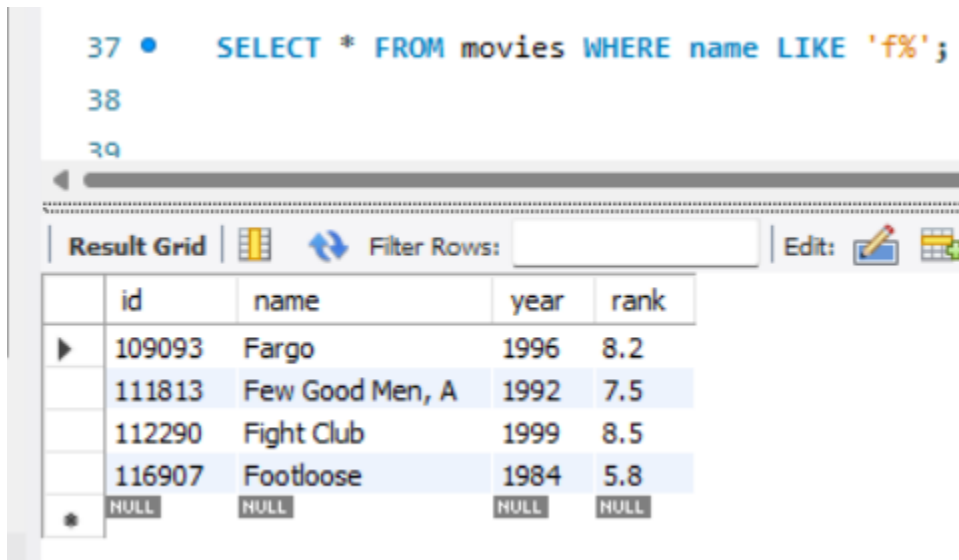
SQL Syntax

```
SELECT * FROM movies WHERE name LIKE 'f%';
```

Explanation

- Here, we get the all columns from movies table and filter rows where the name starts with the letter "f" using LIKE operator.

Output Screenshot



```
37 • SELECT * FROM movies WHERE name LIKE 'f%';
38
39
```

	id	name	year	rank
▶	109093	Fargo	1996	8.2
	111813	Few Good Men, A	1992	7.5
	112290	Fight Club	1999	8.5
	116907	Footloose	1984	5.8
*	NULL	NULL	NULL	NULL

Reference:

- [1] "MySQL DISTINCT Operator," JavaTPoint, [Online]. Available: <https://www.javatpoint.com/mysql-distinct>. [Accessed 9 May 2024]
- [2] "SQL JOIN Clause," W3Schools, [Online]. Available: https://www.w3schools.com/sql/sql_join.asp. [Accessed 9 May 2024]
- [3] "SQL CONCAT() Function," W3Schools, [Online]. Available: https://www.w3schools.com/sql/func_sqlserver_concat.asp. [Accessed 9 May 2024]