

# CSCI-5408

## DATA MANAGEMENT, WAREHOUSING, & ANALYTICS

### LAB ASSIGNMENT - 1

Banner ID: B00977669

GitLab Assignment Link:

[https://git.cs.dal.ca/saji/csci5408\\_w24\\_b00977669\\_christin\\_saji](https://git.cs.dal.ca/saji/csci5408_w24_b00977669_christin_saji)

## Table of Contents

Problem Statement 1 .....	3
Problem Statement 2 .....	5
Problem Statement 3 .....	7
Problem Statement 4 .....	9
Problem Statement 5 .....	12
Problem Statement 6 .....	15

## Problem Statement 1

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

### Query:

```
SELECT COUNT(id) AS unique_actor_count  
FROM imdb.actors;
```

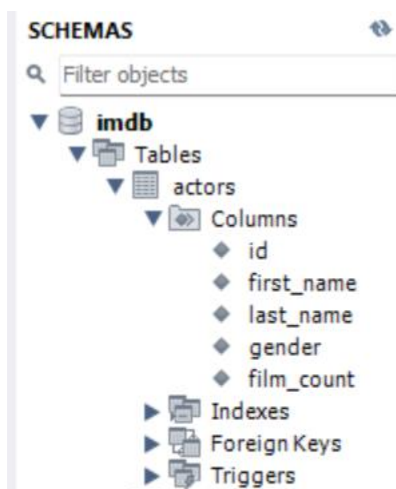


Figure 1 IMDB Schema: Columns of actors Table

**Step 1:** To obtain the count of unique actors, one can simply count the “id” which serves as the primary key in the “actors” table, even when there are multiple actors with the same name.

## Output:

The screenshot shows a database query tool interface. On the left is a 'Navigator' pane with a tree view of the 'imdb' schema. The tree includes 'Tables' (actors, directors, directors\_genres, movies, movies\_directors, movies\_genres, roles), 'Indexes', 'Foreign Keys', 'Triggers', 'Views', 'Stored Procedures', and 'Functions'. The 'actors' table is selected, showing its columns: id, first\_name, last\_name, gender, and film\_count. The main pane displays a SQL query in a text editor:

```
34 • SELECT COUNT(id) AS unique_actor_count
35 FROM imdb.actors;
36
37
38
```

Below the query editor is a 'Result Grid' showing the output of the query. The grid has one column labeled 'unique\_actor\_count' and one row with the value '1907'.

unique_actor_count
1907

Figure 2 Problem Statement 1 Query with its output

## Problem Statement 2

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

### Query:

```
SELECT COUNT(*) AS movie_count_between_1999_till_2000  
FROM imdb.movies  
WHERE year BETWEEN 1999 AND 2000;
```

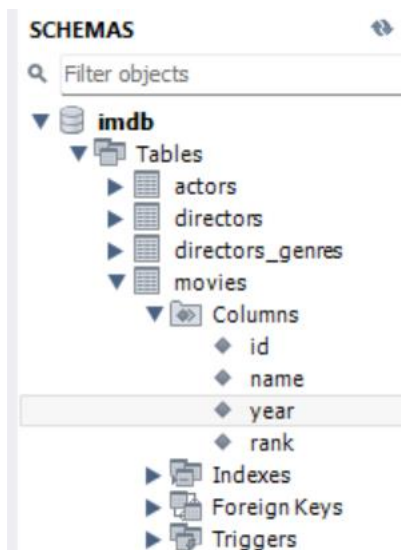


Figure 3 IMDB Schema: Columns of movies Table

**Step 1:** Using “COUNT(\*)”, we can determine the number of rows in a table.

**Step 2:** By incorporating the “WHERE” clause, we can retrieve only the desired rows, i.e., those between the 1990s and 2000.

## Output:

The screenshot shows a database query editor interface. On the left is a 'Navigator' pane with a 'SCHEMAS' tree. Under the 'imdb' schema, the 'movies' table is selected, showing its columns: 'id', 'name', 'year', and 'rank'. The main editor area, titled 'Query 1', contains the following SQL query:

```
36  
37 • SELECT COUNT(*) AS movie_count_between_1999_till_2000  
38 FROM imdb.movies  
39 WHERE year BETWEEN 1999 AND 2000;  
40  
41  
42  
43  
44
```

Below the query editor is a 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The result grid displays the following data:

movie_count_between_1999_till_2000
8

Figure 4 Problem Statement 2 Query with its output

### Problem Statement 3

Find the list of **genres** of movies directed by **Christopher Nolan**.

#### Query:

```
SELECT DISTINCT genre
FROM imdb.directors_genres dg
JOIN imdb.directors d ON dg.director_id = d.id
WHERE d.first_name = 'Christopher' AND d.last_name = 'Nolan';
```

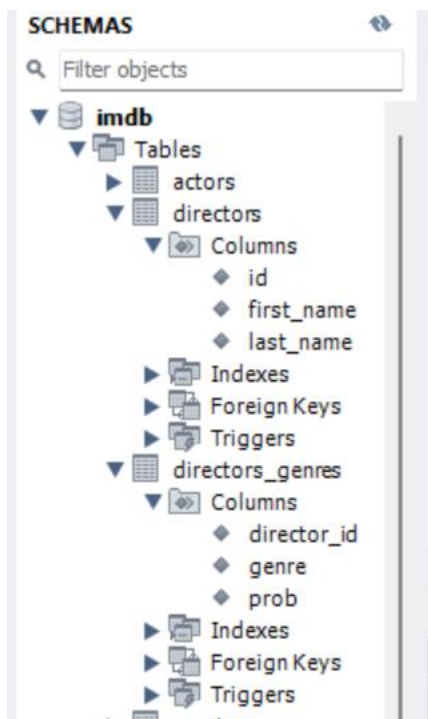
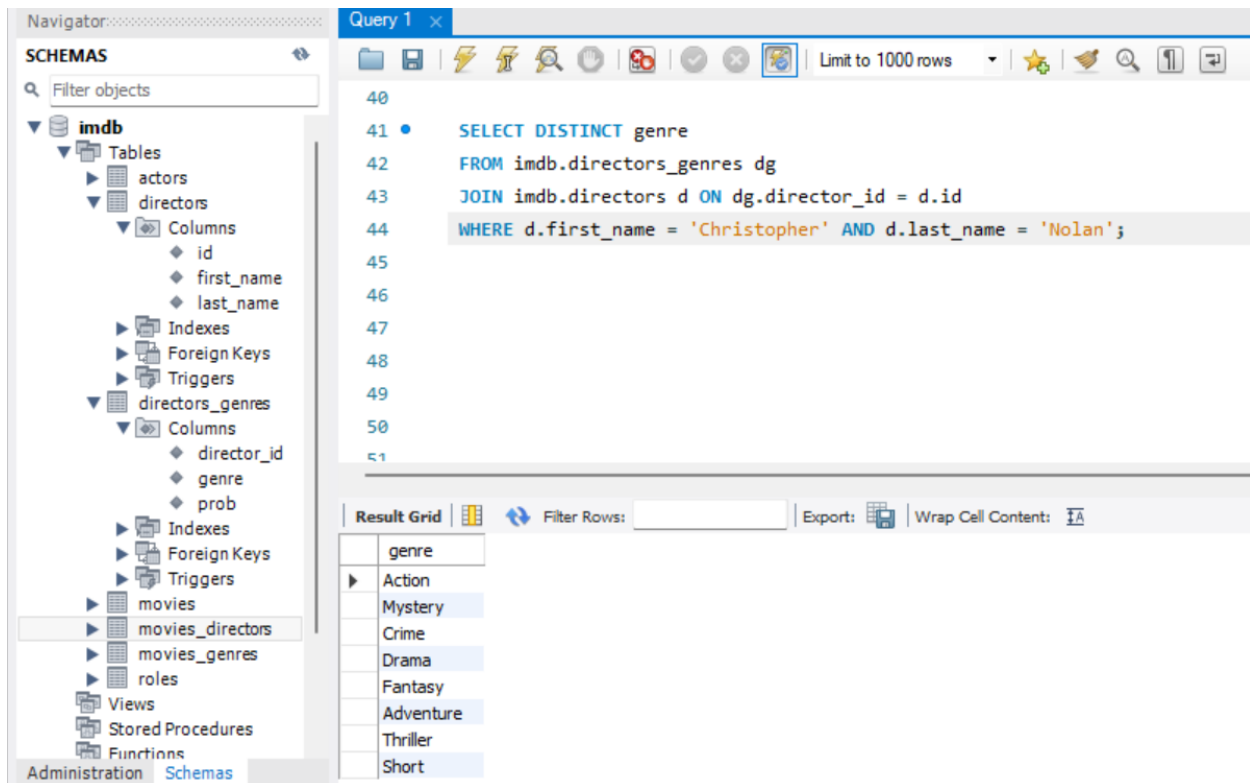


Figure 5 IMDB Schema: Columns of directors and directors\_genres Table

**Step 1:** To obtain the list of genres for movies directed by Christopher Nolan, we need to join the two tables, “directors” and “directors\_genres”. The “directors” table contains details about the director’s name, while the genre details are stored in the “directors\_genres” table.

**Step 2:** Using the “WHERE” clause, we can retrieve only the rows directed by “Christopher Nolan”. Employing “DISTINCT” helps avoid any duplicate genres that may be present in the database.

## Output:



The screenshot shows a database query editor interface. On the left is a 'Navigator' pane with a tree view of the 'imdb' schema, including tables like 'directors' and 'directors\_genres'. The main area displays a SQL query in 'Query 1' tab:

```
40
41 • SELECT DISTINCT genre
42 FROM imdb.directors_genres dg
43 JOIN imdb.directors d ON dg.director_id = d.id
44 WHERE d.first_name = 'Christopher' AND d.last_name = 'Nolan';
45
46
47
48
49
50
51
```

Below the query editor is a 'Result Grid' showing the output of the query. The grid has a single column labeled 'genre' and lists the following genres: Action, Mystery, Crime, Drama, Fantasy, Adventure, Thriller, and Short.

genre
Action
Mystery
Crime
Drama
Fantasy
Adventure
Thriller
Short

Figure 6 Problem Statement 3 Query with its output



#### Problem Statement 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**.

#### Query:

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

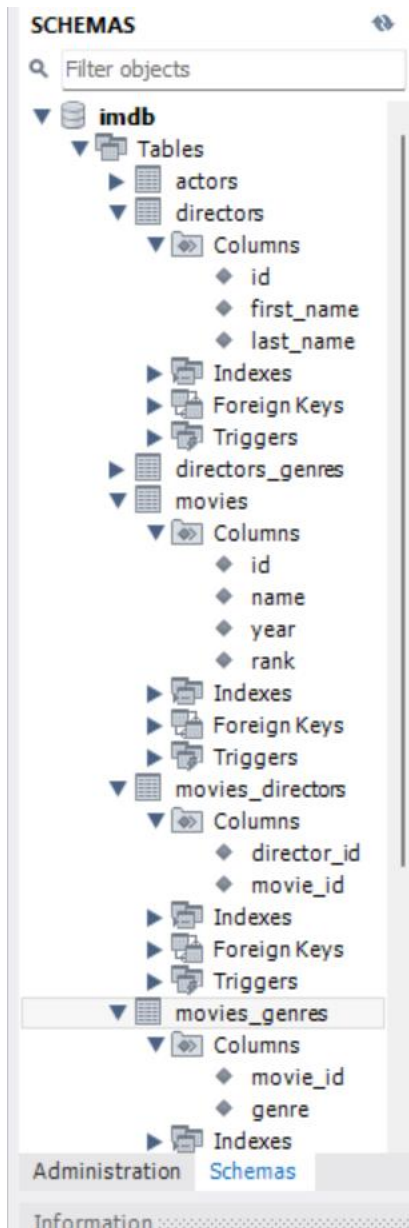


Figure 7 IMDB Schema: Columns of directors, movies, movies\_director and movies\_genres Table

**Step 1:** By joining the tables “directors”, “movies\_directors”, “movies”, and “movies\_genres”, we can retrieve information about directors, movies, ratings, and genres.

**Step 2:** Filtering the results based on rating and genre range will list all the movies with either “Sci-Fi” or “Action” genre.

**Step 3:** Grouping the result by director's name and movie name, we can then count the genres that occur together. If the count is 2, it indicates that movie has both the “Sci-Fi” and “Action” genres. If the count is 1, it means only one of the genres was present.

## Output:

The screenshot shows a database query editor with a schema tree on the left and a query window on the right. The schema tree shows the 'imdb' database with tables 'actors', 'directors', 'directors\_genres', 'movies', 'movies\_directors', and 'movies\_genres'. The query window shows a SQL query that selects director names and movie names, joins the 'directors', 'movies\_directors', 'movies', and 'movies\_genres' tables, filters for movies with a rank between 8 and 9 and genres 'Sci-fi' or 'Action', groups by director name and movie name, and filters for movies with exactly 2 genres.

```
45
46 • SELECT d.first_name, d.last_name, m.name AS movie_name
47 FROM imdb.directors d
48 JOIN imdb.movies_directors md ON d.id = md.director_id
49 JOIN imdb.movies m ON md.movie_id = m.id
50 JOIN imdb.movies_genres mg ON m.id = mg.movie_id
51 WHERE m.rank BETWEEN 8 AND 9
52 AND mg.genre IN ('Sci-fi', 'Action')
53 GROUP BY d.first_name, d.last_name, movie_name
54 HAVING COUNT(mg.genre) = 2;
55
56
57
58
59
```

The result grid shows the following data:

first_name	last_name	movie_name
James (I)	Cameron	Aliens
Andy	Wachowski	Matrix, The
Larry	Wachowski	Matrix, The
George	Lucas	Star Wars

Figure 8 Problem Statement 4 Query with its output

### Problem Statement 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**.

#### Query:

```
SELECT m.name AS movie_name
FROM imdb.roles r
JOIN imdb.movies m ON r.movie_id = m.id
WHERE r.role LIKE 'Dr.%' OR r.role LIKE '%doctor%'
GROUP BY m.id, movie_name
ORDER BY COUNT(*) DESC
LIMIT 1;
```

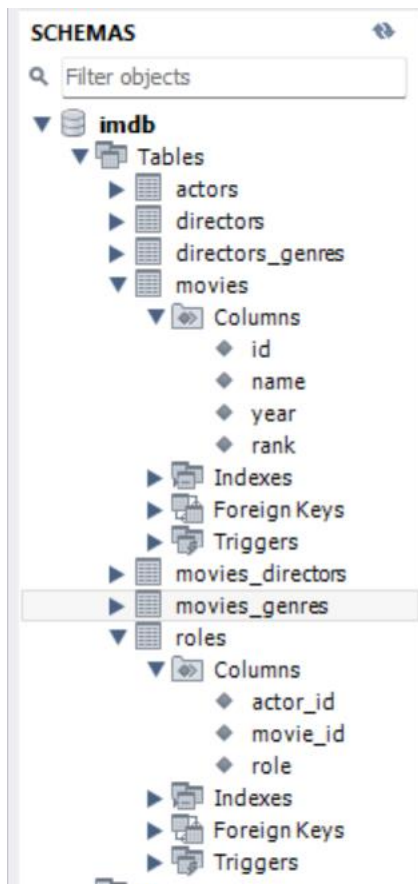


Figure 9 IMDB Schema: Columns of movies and roles Table

Result Grid		Filter Rows:
role		
Man missing guard		
Marine		
Vernon Bundy		
Himself (in Zapruder, Muchmor		
Mullet		
Himself (behind Khrushchev)		
Dr. Peters		
Al Neri		
Quartermaster Robert Hitch...		
Roy Truly (credited on Direct		
Mayor Dooley		
Uncle Harvey		

Figure 10 Output of role data

**Step 1:** Since doctors in the role database are mentioned as “Dr.”, we need to filter roles that start with “Dr.”

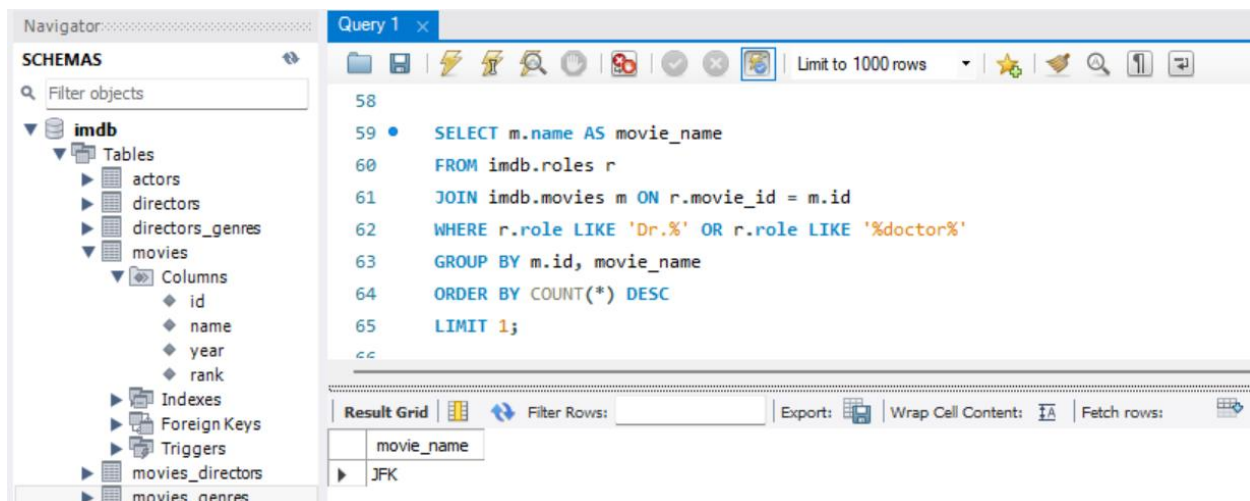
**Step 2:** We join the “roles” with the “movies” to obtain movie and role details.

**Step 3:** Using the “LIKE” operator, we can retrieve role names that start with “Dr.”, and in case any role contains the term “doctor”, we can add “%doctor%” for additional coverage.

**Step 4:** We then group the results by movie ID and movie name, ordering the groups in descending order based on the count of roles (doctors) in each movie. This ensures that the top result will have the highest number of doctor roles.

**Step 5:** Finally, by using “LIMIT 1”, we retrieve only the movie with the highest count.

## Output:



The screenshot displays a database query editor interface. On the left, a 'Navigator' pane shows the 'imdb' schema with tables like 'actors', 'directors', 'directors\_genres', 'movies', 'movies\_directors', and 'movies\_genres'. The 'movies' table is selected, showing columns: id, name, year, rank, and indexes. The main area, titled 'Query 1', contains the following SQL query:

```
58
59 • SELECT m.name AS movie_name
60 FROM imdb.roles r
61 JOIN imdb.movies m ON r.movie_id = m.id
62 WHERE r.role LIKE 'Dr.%' OR r.role LIKE '%doctor%'
63 GROUP BY m.id, movie_name
64 ORDER BY COUNT(*) DESC
65 LIMIT 1;
```

Below the query, the 'Result Grid' shows the output:

movie_name
JFK

Figure 11 Problem Statement 5 Query with its output

## Problem Statement 6

Find the list of the movies that start with the letter ‘f’.

### Query:

```
SELECT name AS movie_name
```

```
FROM imdb.movies
```

```
WHERE name LIKE 'F%';
```

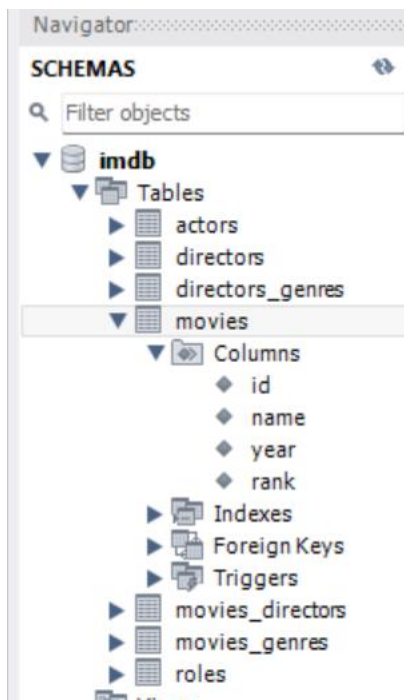


Figure 12 IMDB Schema: Columns of movies Table

**Step 1:** Simply by checking the names that start with “F” using the “LIKE” operator in the movies table.

## Output:

The screenshot shows a database query tool interface. On the left is a 'Navigator' pane with a tree view of the 'imdb' schema. The 'movies' table is selected, and its columns (id, name, year, rank) are visible. The main area is titled 'Query 1' and contains the following SQL query:

```
66  
67 • SELECT name AS movie_name  
68 FROM imdb.movies  
69 WHERE name LIKE 'F%';  
70
```

Below the query editor is a 'Result Grid' showing the output of the query. The grid has a header row 'movie\_name' and four data rows:

movie_name
Fargo
Few Good Men, A
Fight Club
Footloose

Figure 13 Problem Statement 6 Query with its output