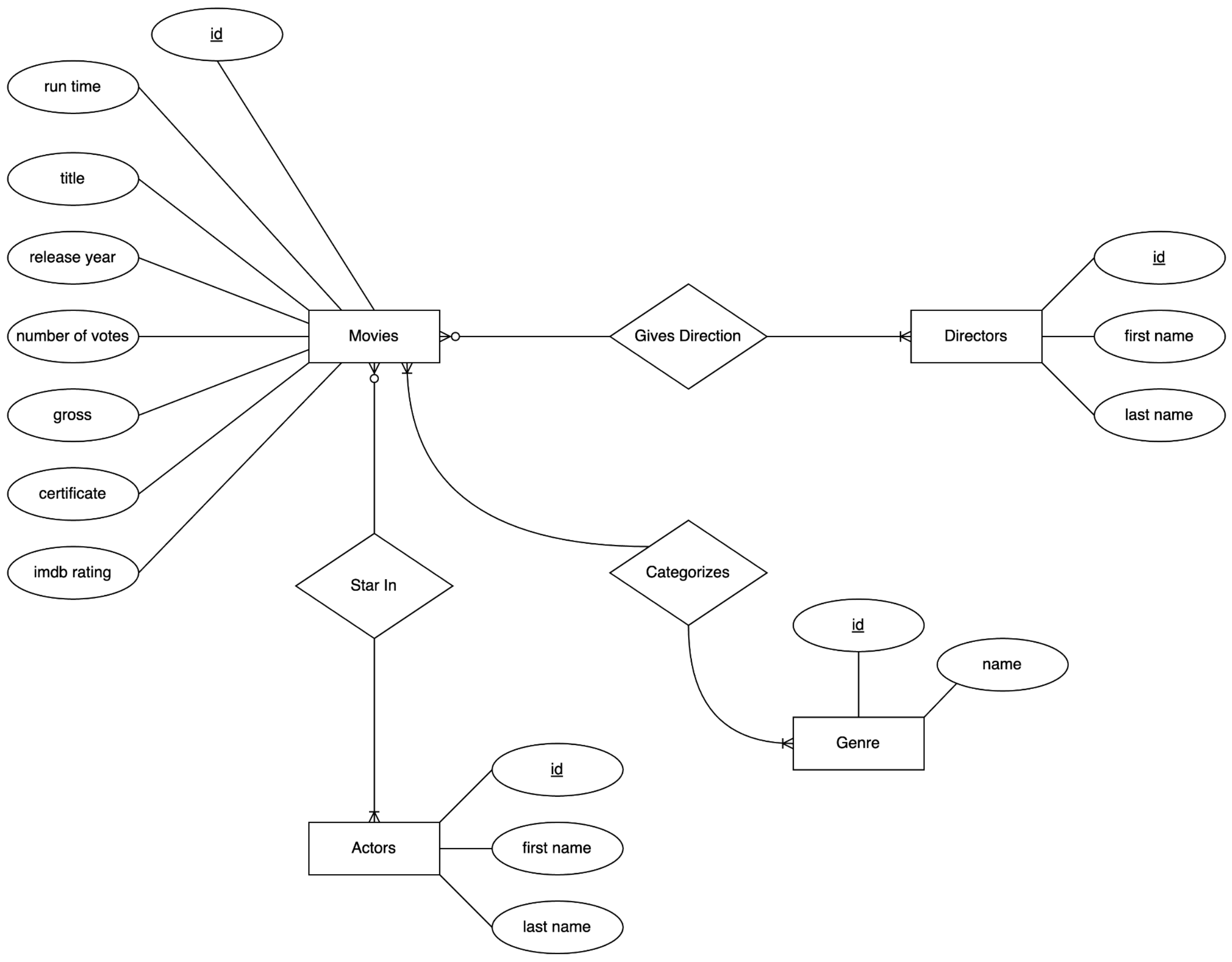
CS 586 Project Report

1. Dataset

I chose cinema as the topic for my final project because I have always been passionate about movies and enjoy exploring their impacts on culture and society. Having a movie database will help us gain valuable insights into various aspects of cinema, such as box office performance, reviews, and the people involved in making these films. This database will also be useful for researchers who want to analyze trends and understand how cinema changes over time. The database is based on this data source: IMDB Movies Dataset. The GitHub repository for this project is available here.

2. ER Diagram



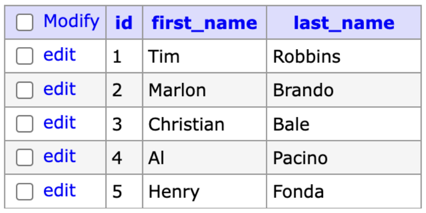
3. Relational Schema

The database consists of 7 tables:

- Movies (id, title, release_year, certificate, runtime, imdb_rating, number_of_votes, gross)
- Actors (id, first_name, last_name)
- Directors (id, first_name, last_name)
- Genres (id, name)
- Movie_Director_Rel(movie_id, director_id)
  - movie_id is a foreign key referencing Movies(id)
  - director_id is a foreign key referencing Directors(id)
- Movie_Actor_Rel (movie_id, actor_id)
  - movie_id is a foreign key referencing Movies(id)
  - actor_id is a foreign key referencing Actors(id)
- Movie_Genre_Rel (movie_id, genre_id)
  - movie_id is a foreign key referencing Movies(id)
  - genre_id is a foreign key referencing Genres(id)

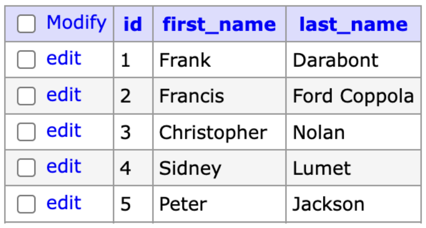
4. Table Creation

```
CREATE TABLE actors (
   id INT PRIMARY KEY,
   first_name VARCHAR(100),
   last_name VARCHAR(100)
);
```



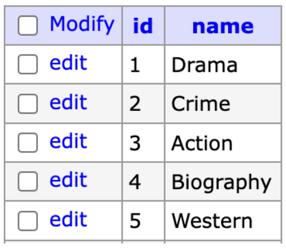
Modify	id	first_name	last_name
edit	1	Tim	Robbins
edit	2	Marlon	Brando
edit	3	Christian	Bale
edit	4	Al	Pacino
edit	5	Henry	Fonda

```
CREATE TABLE directors (
   id INT PRIMARY KEY,
   first_name VARCHAR(100),
   last_name VARCHAR(100)
);
```



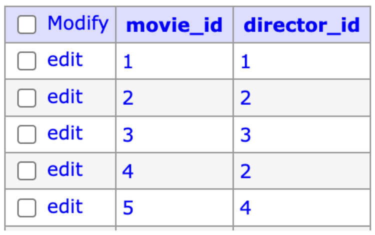
Modify	id	first_name	last_name
edit	1	Frank	Darabont
edit	2	Francis	Ford Coppola
edit	3	Christopher	Nolan
edit	4	Sidney	Lumet
edit	5	Peter	Jackson

```
CREATE TABLE genres (
   id INT PRIMARY KEY,
   name VARCHAR(100) NOT NULL
);
```



Modify	id	name
edit	1	Drama
edit	2	Crime
edit	3	Action
edit	4	Biography
edit	5	Western

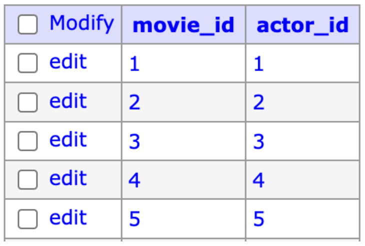
```
CREATE TABLE movie_director_rel (
   movie_id INT NOT NULL,
   director_id INT NOT NULL,
   PRIMARY KEY (movie_id, director_id),
```



Modify	movie_id	director_id
edit	1	1
edit	2	2
edit	3	3
edit	4	2
edit	5	4

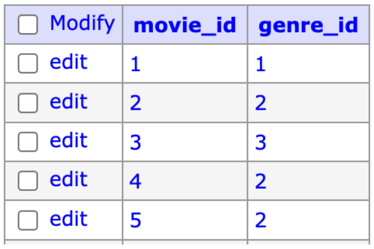
```
    FOREIGN KEY (movie_id) REFERENCES Movies(id),
    FOREIGN KEY (director_id) REFERENCES Directors(id)
);

CREATE TABLE movie_actor_rel (
    movie_id INT NOT NULL,
    actor_id INT NOT NULL,
    PRIMARY KEY (movie_id, actor_id),
    FOREIGN KEY (movie_id) REFERENCES Movies(id),
    FOREIGN KEY (actor_id) REFERENCES Actors(id)
);
```



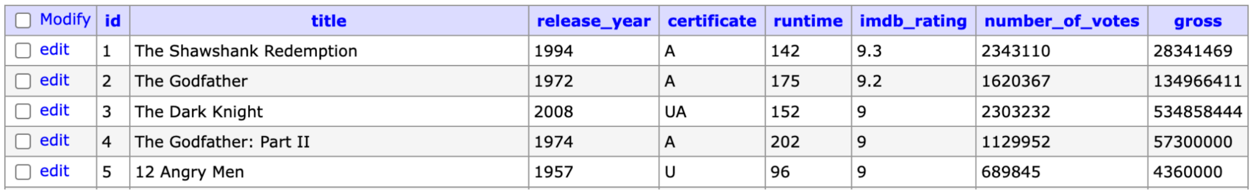
Modify	movie_id	actor_id
edit	1	1
edit	2	2
edit	3	3
edit	4	4
edit	5	5

```
CREATE TABLE movie_genre_rel (
    movie_id INT NOT NULL,
    genre_id INT NOT NULL,
    PRIMARY KEY (movie_id, genre_id),
    FOREIGN KEY (movie_id) REFERENCES Movies(id),
    FOREIGN KEY (genre_id) REFERENCES Genres(id)
);
```



Modify	movie_id	genre_id
edit	1	1
edit	2	2
edit	3	3
edit	4	2
edit	5	2

```
CREATE TABLE movies (
    id INT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    release_year INT NOT NULL,
    certificate VARCHAR(50),
    runtime INT,
    imdb_rating FLOAT,
    number_of_votes INT,
    gross INT
);
```



Modify	id	title	release_year	certificate	runtime	imdb_rating	number_of_votes	gross
edit	1	The Shawshank Redemption	1994	A	142	9.3	2343110	28341469
edit	2	The Godfather	1972	A	175	9.2	1620367	134966411
edit	3	The Dark Knight	2008	UA	152	9	2303232	534858444
edit	4	The Godfather: Part II	1974	A	202	9	1129952	57300000
edit	5	12 Angry Men	1957	U	96	9	689845	4360000

5. Data Cleaning, Data Verification, and Data Validation

I utilized Google Sheets and Pandas for cleaning the dataset. I used Google Sheets to identify any inconsistencies, missing values, and formatting issues across all fields. The original dataset contains multiple genre values for a movie, but I decided to include only the main genre for simplification. I also normalized numerical data like runtime, gross earnings, and ratings to ensure consistent measurement units throughout.

After cleaning the original dataset in Google Sheets, I utilized the Pandas library in Python to create the relationship tables that establish connections between movies, directors, actors, and genres. This tool helped me generate the appropriate foreign key relationships required for the many-to-many relationships (e.g. multiple actors with multiple movies) in the database. I also implemented checks to ensure referential integrity across all relationship tables to confirm that every movie_id, actor_id, director_id, and genre_id in the relationship tables corresponded to the right primary keys in their respective parent tables.

6. Database Population

I used Google Sheets and Pandas to segment the cleaned dataset into separate CSV files corresponding to each entity in the database. The schema consists of 7 tables: movies, actors, directors, genres, movie_director_rel, movie_actor_rel, and movie_genre_rel. The primary entity tables (Movies, Actors, Directors, and Genres) were populated with unique records.

- For the Movies table, each film would be assigned a unique ID along with its title, release year, certificate, runtime, IMDB rating, number of votes, and gross earnings.
- The Actors and Directors tables store unique individuals with IDs and their first and last names.
- The Genres table stores unique genre categories with corresponding IDs

After populating these primary tables, I used Pandas to create relationship tables to establish connections between entities.

- Movie_Director_Rel connects movies to their directors
- Movie_Actor_Rel connects movies with their cast members
- Movie_Genre_Rel connects movies with their genres

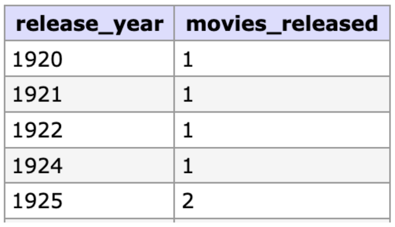
The final step was to create PostgreSQL tables with structures that matched the clean data files. After the tables were created, I imported the data using PostgreSQL's built-in import function.

7. Questions and SQL Queries

Here is a list of 20 questions that are frequently asked about cinema:

- Question 1: How many movies were released each year? (returns 99 rows)
  ```
  SELECT release_year, COUNT(*) AS movies_released
  FROM movies
  GROUP BY release_year
  ```

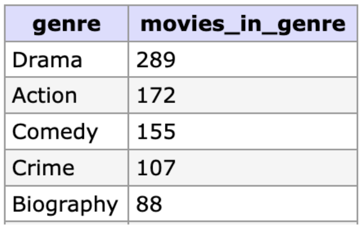
```
ORDER BY release_year;
```



release_year	movies_released
1920	1
1921	1
1922	1
1924	1
1925	2

- Question 2: Which movie genres are the most common? (returns 14 rows)

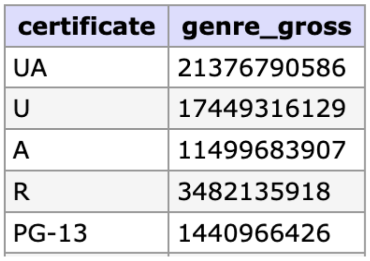
```
SELECT g.name AS genre, COUNT(*) AS movies_in_genre
FROM genres g
JOIN movie_genre_rel mgr ON g.id = mgr.genre_id
GROUP BY g.name
ORDER BY movies_in_genre DESC;
```



genre	movies_in_genre
Drama	289
Action	172
Comedy	155
Crime	107
Biography	88

- Question 3: Which certificate ratings (e.g. PG, R, G) earn the most money? (returns 14 rows)

```
SELECT certificate, SUM(gross) AS genre_gross
FROM movies
WHERE gross IS NOT NULL
GROUP BY certificate
ORDER BY genre_gross DESC
```

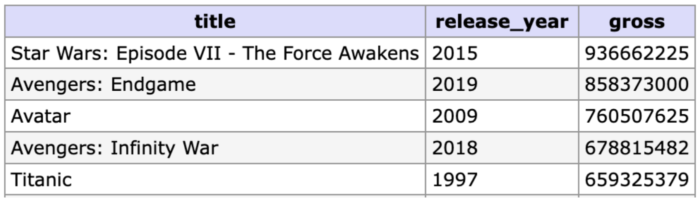


certificate	genre_gross
UA	21376790586
U	17449316129
A	11499683907
R	3482135918
PG-13	1440966426

- Question 4: What are the top 10 highest-grossing movies of all time? (returns 10 rows)

```
SELECT title, release_year, gross
FROM movies
WHERE gross IS NOT NULL
ORDER BY gross DESC
```

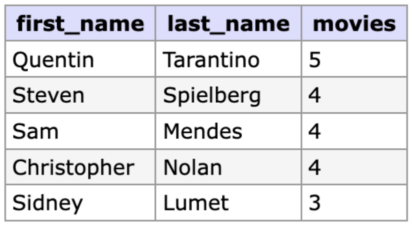
```
LIMIT 10;
```



title	release_year	gross
Star Wars: Episode VII - The Force Awakens	2015	936662225
Avengers: Endgame	2019	858373000
Avatar	2009	760507625
Avengers: Infinity War	2018	678815482
Titanic	1997	659325379

- Question 5: Who are the top 5 directors with the most films grossing over $100 million? (returns 5 rows)

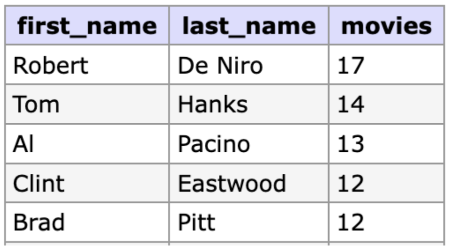
```
SELECT d.first_name, d.last_name, COUNT(*) movies
FROM directors d
JOIN movie_director_rel mdr ON d.id = mdr.director_id
JOIN movies m ON mdr.movie_id = m.id
WHERE m.gross > 100000000
GROUP BY d.id, d.first_name, d.last_name
ORDER BY movies DESC
LIMIT 5;
```



first_name	last_name	movies
Quentin	Tarantino	5
Steven	Spielberg	4
Sam	Mendes	4
Christopher	Nolan	4
Sidney	Lumet	3

- Question 6: Who are the top 10 actors that appear most frequently in movies with high ratings (e.g. above 7.5)? (returns 10 rows)

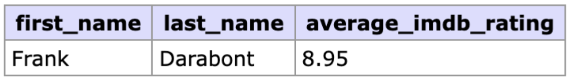
```
SELECT a.first_name, a.last_name, COUNT(*) AS movies
FROM actors a
JOIN movie_actor_rel mar ON a.id = mar.actor_id
JOIN movies m ON mar.movie_id = m.id
WHERE m.imdb_rating > 7.5
GROUP BY a.id, a.first_name, a.last_name
ORDER BY movies DESC
LIMIT 10;
```



first_name	last_name	movies
Robert	De Niro	17
Tom	Hanks	14
Al	Pacino	13
Clint	Eastwood	12
Brad	Pitt	12

- Question 7: Which director has the highest average rating for their movies? (returns 1 row)

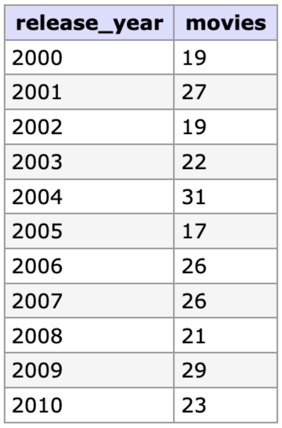
```
SELECT d.first_name, d.last_name, AVG(m.imdb_rating) AS
average_imdb_rating
FROM directors d
JOIN movie_director_rel mdr ON d.id = mdr.director_id
JOIN movies m ON mdr.movie_id = m.id
GROUP BY d.id, d.first_name, d.last_name
ORDER BY average_imdb_rating DESC
LIMIT 1;
```



first_name	last_name	average_imdb_rating
Frank	Darabont	8.95

- Question 8: How many movies were released each year in a certain period (e.g. 2000 - 2010)? (returns 11 rows)

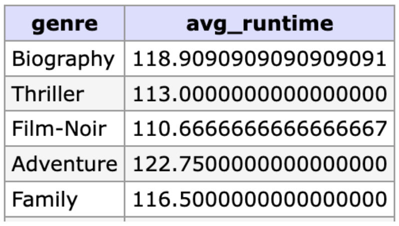
```
SELECT release_year, COUNT(*) AS movies
FROM movies
WHERE release_year >= 2000 AND release_year <= 2010
GROUP BY release_year
ORDER BY release_year;
```



release_year	movies
2000	19
2001	27
2002	19
2003	22
2004	31
2005	17
2006	26
2007	26
2008	21
2009	29
2010	23

- Question 9: What is the average runtime of movies in each genre? (returns 14 rows)

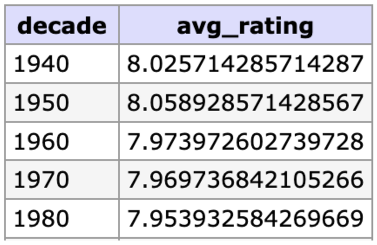
```
SELECT g.name AS genre, AVG(m.runtime) AS avg_runtime
FROM genres g
JOIN movie_genre_rel mgr ON g.id = mgr.genre_id
JOIN movies m ON mgr.movie_id = m.id
GROUP BY g.name;
```



genre	avg_runtime
Biography	118.9090909090909091
Thriller	113.0000000000000000
Film-Noir	110.6666666666666667
Adventure	122.7500000000000000
Family	116.5000000000000000

- Question 10: How has the average rating for movies changed in each decade since 1940? (returns 9 rows)

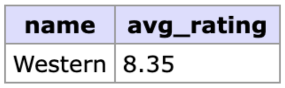
```
SELECT  (release_year / 10) * 10 AS decade, AVG(imdb_rating) AS avg_rating
FROM movies
WHERE release_year >= 1940
GROUP BY decade
ORDER BY decade;
```



decade	avg_rating
1940	8.025714285714287
1950	8.058928571428567
1960	7.973972602739728
1970	7.969736842105266
1980	7.953932584269669

- Question 11: Which movie genre has the highest average rating? (returns 1 row)

```
SELECT g.name, AVG(m.imdb_rating) AS avg_rating
FROM genres g
JOIN movie_genre_rel mgr ON g.id = mgr.genre_id
JOIN movies m ON mgr.movie_id = m.id
GROUP BY g.name
ORDER BY avg_rating DESC
LIMIT 1;
```

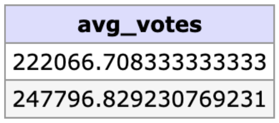


name	avg_rating
Western	8.35

- Question 12: How has the average number of votes for recently released movies (e.g. 2 years or less) compared to old movies (e.g. 15 years or more)? The latest release_year in this dataset is 2020. (returns 2 rows)

```
SELECT AVG(number_of_votes) AS avg_votes
FROM movies
WHERE (2020 - release_year) <= 2
UNION ALL
SELECT AVG(number_of_votes) AS avg_votes
```

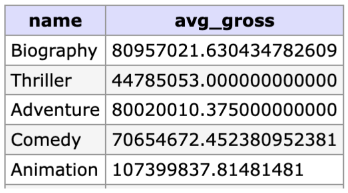
```
FROM movies
WHERE (2020 - release_year) >= 15;
```



avg_votes
222066.708333333333
247796.829230769231

- Question 13: What is the average box office gross for movies in each genre, considering only movies with a high number of votes (e.g. 100000 or more votes)?

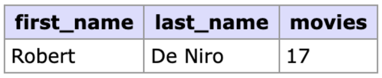
```
SELECT g.name, AVG(m.gross) AS avg_gross
FROM genres g
JOIN movie_genre_rel mgr ON g.id = mgr.genre_id
JOIN movies m ON mgr.movie_id = m.id
WHERE m.number_of_votes >= 100000
GROUP BY g.name;
```



name	avg_gross
Biography	80957021.630434782609
Thriller	44785053.000000000000
Adventure	80020010.375000000000
Comedy	70654672.452380952381
Animation	107399837.81481481

- Question 14: Which actor/actress has appeared in the most films?

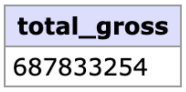
```
SELECT a.first_name, a.last_name, COUNT(*) AS movies
FROM actors a
JOIN movie_actor_rel mar ON a.id = mar.actor_id
GROUP BY a.id, a.first_name, a.last_name
ORDER BY movies DESC
LIMIT 1;
```



first_name	last_name	movies
Robert	De Niro	17

- Question 15: What is the total box office gross for all movies directed by a director (e.g. James Cameron)? (returns 1 row)

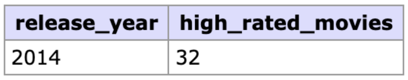
```
SELECT SUM(m.gross) AS total_gross
FROM movies m
JOIN movie_director_rel mdr ON m.id = mdr.movie_id
JOIN directors d ON mdr.director_id = d.id
WHERE d.first_name = 'James' AND d.last_name = 'Cameron';
```



total_gross
687833254

- Question 16: Which year had the highest number of movies with a high rating (e.g. above 7.5)?

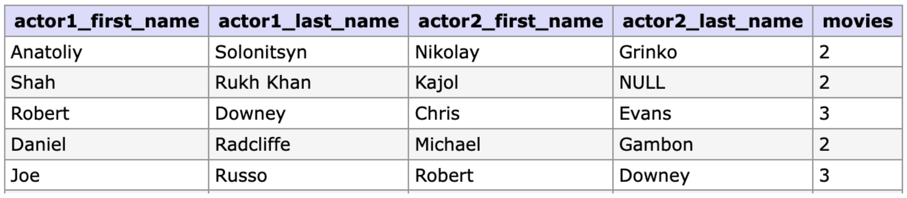
```
SELECT release_year, COUNT(*) AS high_rated_movies
FROM movies
WHERE imdb_rating > 7.5
GROUP BY release_year
ORDER BY high_rated_movies DESC
LIMIT 1;
```



release_year	high_rated_movies
2014	32

- Question 17: What actors appear together in more than one movie? (returns 121 rows)

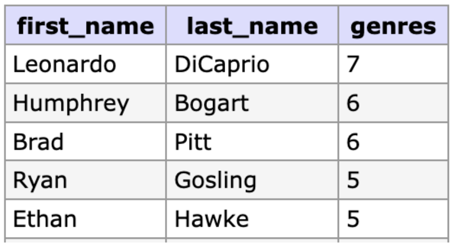
```
SELECT a1.first_name AS actor1_first_name,
            a1.last_name AS actor1_last_name,
            a2.first_name AS actor2_first_name,
            a2.last_name AS actor2_last_name,
            COUNT(*) AS movies
FROM movie_actor_rel mar1,
         movie_actor_rel mar2,
         actors a1,
         actors a2
WHERE mar1.movie_id = mar2.movie_id
   AND mar1.actor_id = a1.id
   AND mar2.actor_id = a2.id
   AND mar1.actor_id < mar2.actor_id
GROUP BY a1.id, a1.first_name, a1.last_name, a2.id, a2.first_name,
a2.last_name
HAVING COUNT(*) > 1;
```



actor1_first_name	actor1_last_name	actor2_first_name	actor2_last_name	movies
Anatoliy	Solonitsyn	Nikolay	Grinko	2
Shah	Rukh Khan	Kajol	NULL	2
Robert	Downey	Chris	Evans	3
Daniel	Radcliffe	Michael	Gambon	2
Joe	Russo	Robert	Downey	3

- Question 18: Which actors have appeared across most movie genres? (returns 10 rows)

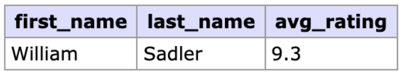
```
 SELECT a.first_name, a.last_name,
      (SELECT COUNT(DISTINCT g.id)
    FROM movie_actor_rel mar, movie_genre_rel mgr, genres g
    WHERE mar.actor_id = a.id
        AND mar.movie_id = mgr.movie_id
        AND mgr.genre_id = g.id) AS genres
FROM actors a
LIMIT 10;
```



first_name	last_name	genres
Leonardo	DiCaprio	7
Humphrey	Bogart	6
Brad	Pitt	6
Ryan	Gosling	5
Ethan	Hawke	5

- Question 19: Which actor has the highest average rating across all their movies? (returns 1 row)

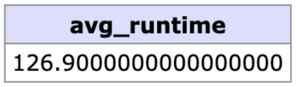
```
SELECT a.first_name, a.last_name,
    AVG(m.imdb_rating) AS avg_rating
FROM actors a
JOIN movie_actor_rel mar ON a.id = mar.actor_id
JOIN movies m ON mar.movie_id = m.id
GROUP BY a.id, a.first_name, a.last_name
ORDER BY avg_rating DESC
LIMIT 1;
```



first_name	last_name	avg_rating
William	Sadler	9.3

- Question 20: What is the average runtime of movies that earn the most money? (returns 1 row)

```
SELECT AVG(runtime) AS avg_runtime
FROM (
  SELECT runtime
  FROM movies
  ORDER BY gross DESC
  LIMIT 10
) AS top_movies;
```



avg_runtime
126.9000000000000000