SVKM's NMIMS
School of Technology Management & Engineering, Chandigarh
A.Y. 2023 - 24
Course: Database Management Systems

Project Report

Program	Btech Integrated	
Semester	8	
Name of the Project:	Movie Database System	
Details of Project Members		
Batch	Roll No.	Name
C1	C089	Raj Parmar
C1	C092	Karan Dave
C1	C096	Vedant Jadhav
Date of Submission:		

Contribution of each project Members:

Roll No.	Name:	Contribution
C089	Raj Parmar	Creating DataBase, ER-model, Relational Model
C092	Karan Dave	SQL Queries, Normalization
C096	Vedant Jadhav	Creating DataBase, SQL Queries, Storyline of Project

Github link of your project: https://github.com/KaranDave31/Movie-Database-System

Project Report

Movie Database System

by

Raj Parmar, Roll number: C089

Karan Dave, Roll number: C092

Vedant Jadhav, Roll number: C096

Course: DBMS

AY: 2023-24

Table of Contents

I. Storyline

In the ever-changing hobby of bringing stories to life on silver screen, the need for a comprehensive film database becomes paramount.

Our journey begins with a vision – to create a centralized hub where users can explore, discover and engage with films across genres, eras and cultures.

The first step to realizing this vision is to develop a robust database system that captures every aspect of the movie experience.

We carefully create tables of actors, directors, movies, movies, and of course the movies themselves. Each table is carefully designed to preserve important information while making it easy to expand and customize in the future.

Then explain the relationships between these entities, ensuring data integrity and robust questions.
We then define the appropriate primary key, foreign key, and data attribute for each attribute to ensure data consistency and integrity.

As the database replicates, we explore the complexities of SQL queries, holding the language while a maestro leads an orchestra. From simple SELECT statements to complex JOIN functions, each query is perfectly designed to elicit meaningful insights and suggestions from users.

But our purposes extend beyond just data recovery. Recognizing the power of user engagement, we incorporate elements such as user review ratings into the database design. This not only allows users to consume content but also contribute their thoughts and ideas, creating a sense of community within the platform.

II. Components of Database Design

Entities and Attributes:

- Movies
  1) movie_id (Primary Key)
  2) title
  3) release_year
  4) genre
  5) director
  6) runtime
  7) plot
  8) poster_url

- Actors
  1) actor_id (Primary Key)
  2) name
  3) birthdate
  4) nationality

- Reviews
  1) review_id (Primary Key)
  2) movie_id (Foreign Key to Movies table)
  3) user_id (Foreign Key to Users table, if available)
  4) review_text
  5) review_date

- Users (Optional, for user management)
  1) user_id (Primary Key)
  2) username
  3) email
  4) password

- Ratings
  1) rating_id (Primary Key)
  2) movie_id (Foreign Key to Movies table)
  3) user_id (Foreign Key to Users table, if available)
  4) rating_value
  5) rating_date

- Genres
  1) genre_id (Primary Key)
  2) genre_name

Relationships:

- Movies - Actors (Many-to-Many):
  1) movie_id (Foreign Key to Movies table)
  2) actor_id (Foreign Key to Actors table)

- Movies - Reviews (One-to-Many):
  1) Each movie can have multiple reviews.
  2) Foreign Key: movie_id in Reviews table references movie_id in Movies table.

- Users - Reviews (One-to-Many): (Optional, if user management is implemented)
  1) Each user can write multiple reviews.
  2) Foreign Key: user_id in Reviews table references user_id in Users table.

- Movies - Ratings (One-to-Many):
  1) Each movie can have multiple ratings.
  2) Foreign Key: movie_id in Ratings table references movie_id in Movies table.

- Users - Ratings (One-to-Many): (Optional, if user management is implemented)
  1) Each user can rate multiple movies.
  2) Foreign Key: user_id in Ratings table references user_id in Users table.

- Movies - Genres (Many-to-Many):
  1) MovieGenres Table
  2) movie_id (Foreign Key to Movies table)
  3) genre_id (Foreign Key to Genres table)

Cardinality and Participation:

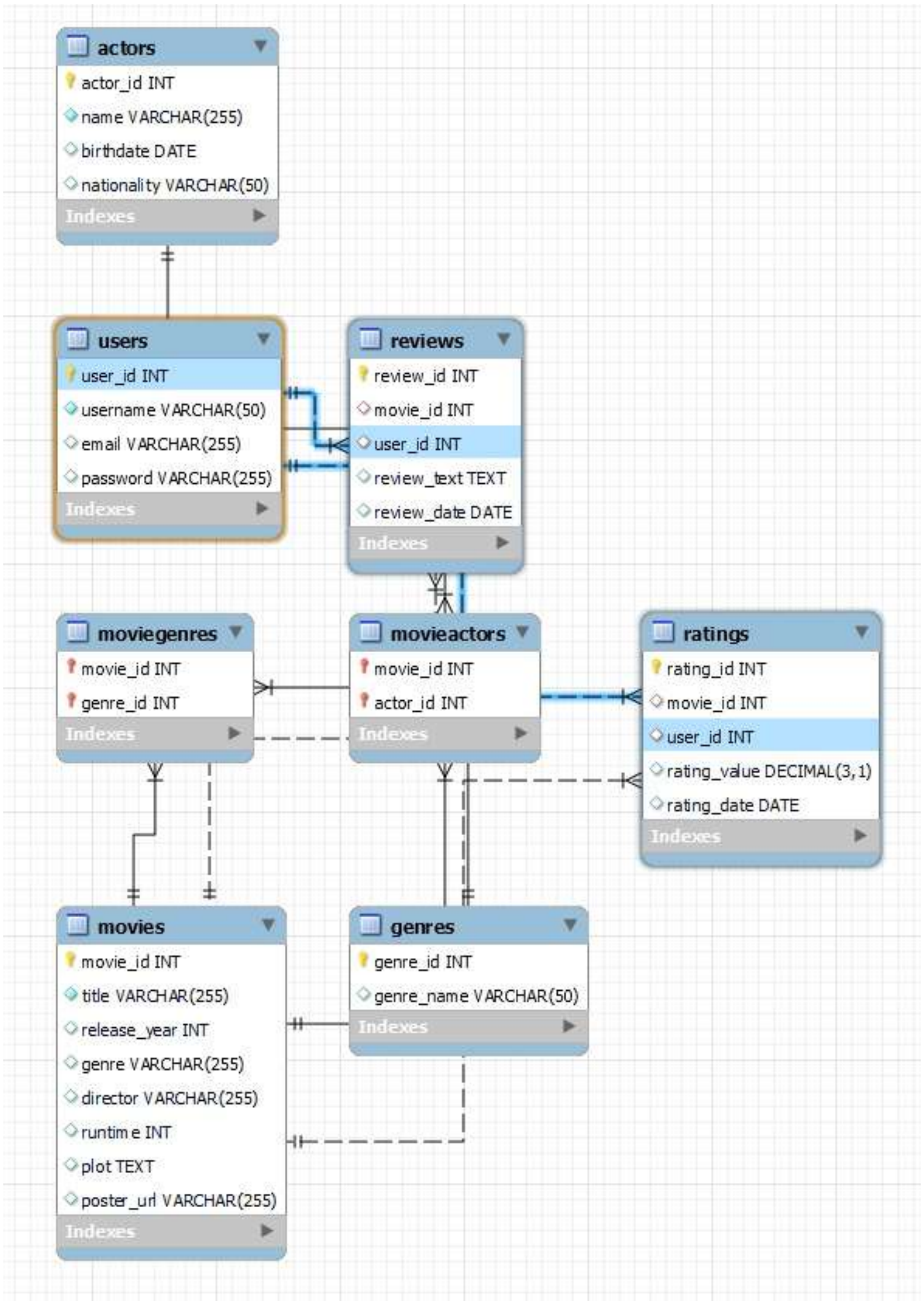
- Movies - Actors (Many-to-Many):
  1) Participation: Total (Both movies and actors must exist).
  2) Cardinality: Each movie can have many actors, and each actor can be in many movies.

- Movies - Reviews (One-to-Many):
  1) Participation: Partial (A movie may have zero or more reviews).
  2) Cardinality: Each movie can have multiple reviews, but each review belongs to only one movie.

- Movies - Ratings (One-to-Many):
  1) Participation: Partial (A movie may have zero or more ratings).
  2) Cardinality: Each movie can have multiple ratings, but each rating is for only one movie.

- Movies - Genres (Many-to-Many):
  1) Participation: Total (Both movies and genres must exist).
  2) Cardinality: Each movie can belong to multiple genres, and each genre can have multiple movies.

III. Entity Relationship Diagram



IV. Relational Model

Movies

Movie_id	Title	Release_year	genre	Director	Runtime	Plot	Poster_url

Actors

Actor_id	Name	Birthdate	nationality

MovieActors

Movie_id	Actor_id

Reviews

Review_id	Movie_id	User_id	Review_text	Review_date

Users

User_id	Username	Email	password

Ratings

Rating_id	Movie_id	User_id	Rating_value	Rating_date

Genres

Genre_id	Genre_name

MovieGenres

Movie_id	Genre_id

V. Normalization

1. 1NF (First Normal Form):

   - Movies Table: Already in 1NF, as each attribute contains atomic values.
   - Actors Table: Already in 1NF.
   - Reviews Table: Already in 1NF.
   - Users Table: Already in 1NF.
   - Ratings Table: Already in 1NF.
   - Genres Table: Already in 1NF.

2. 2NF (Second Normal Form):

   - Movies Table: No partial dependencies exist, so it's already in 2NF.

3. 3NF (Third Normal Form):

   - Movies Table: director and runtime are directly related to movies and do not depend on each other or any other non-key attribute. Hence, it's already in 3NF.

4. BCNF (Boyce-Codd Normal Form):

   - Movies Table: All non-key attributes are directly dependent on the primary key (movie_id), so it's already in BCNF.

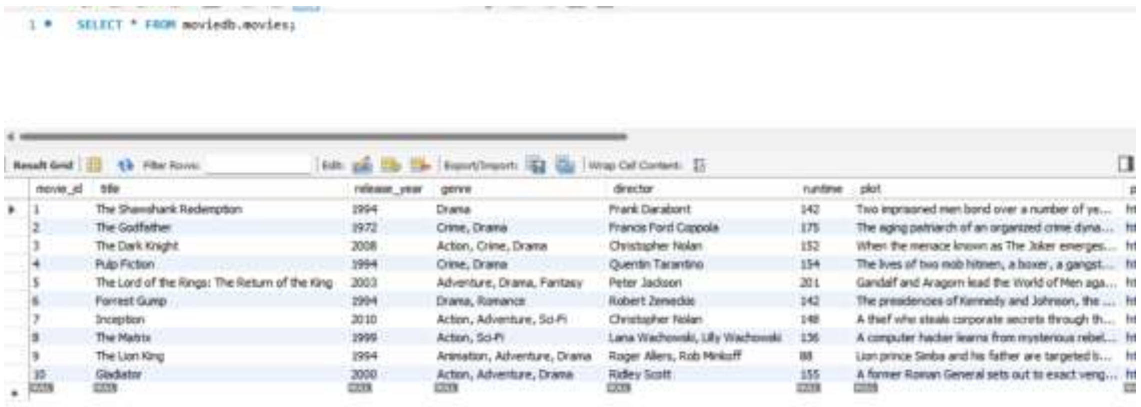
Since all tables are already in 1NF, 2NF, 3NF, and BCNF, there are no further adjustments needed in the database schema to achieve these normal forms. The design is already normalized and supports data integrity and efficiency.

VI. SQL Queries

1. Insert data into the Movies table

```
INSERT INTO Movies (title, release_year, genre, director, runtime, plot, poster_url) VALUES
('The Shawshank Redemption', 1994, 'Drama', 'Frank Darabont', 142, 'Two imprisoned men bond over
a number of years, finding solace and eventual redemption through acts of common decency.',
'https://example.com/poster1.jpg'),
```

('The Godfather', 1972, 'Crime, Drama', 'Francis Ford Coppola', 175, 'The aging patriarch of an organized crime dynasty transfers control of his clandestine empire to his reluctant son.', 'https://example.com/poster2.jpg'),
('The Dark Knight', 2008, 'Action, Crime, Drama', 'Christopher Nolan', 152, 'When the menace known as The Joker emerges from his mysterious past, he wreaks havoc and chaos on the people of Gotham.', 'https://example.com/poster3.jpg'),
('Pulp Fiction', 1994, 'Crime, Drama', 'Quentin Tarantino', 154, 'The lives of two mob hitmen, a boxer, a gangster and his wife, and a pair of diner bandits intertwine in four tales of violence and redemption.', 'https://example.com/poster4.jpg'),
('The Lord of the Rings: The Return of the King', 2003, 'Adventure, Drama, Fantasy', 'Peter Jackson', 201, 'Gandalf and Aragorn lead the World of Men against Sauron\'s army to draw his gaze from Frodo and Sam as they approach Mount Doom with the One Ring.', 'https://example.com/poster5.jpg'),
('Forrest Gump', 1994, 'Drama, Romance', 'Robert Zemeckis', 142, 'The presidencies of Kennedy and Johnson, the events of Vietnam, Watergate, and other historical events unfold from the perspective of an Alabama man with an IQ of 75, whose only desire is to be reunited with his childhood sweetheart.', 'https://example.com/poster6.jpg'),
('Inception', 2010, 'Action, Adventure, Sci-Fi', 'Christopher Nolan', 148, 'A thief who steals corporate secrets through the use of dream-sharing technology is given the inverse task of planting an idea into the mind of a C.E.O.', 'https://example.com/poster7.jpg'),
('The Matrix', 1999, 'Action, Sci-Fi', 'Lana Wachowski, Lilly Wachowski', 136, 'A computer hacker learns from mysterious rebels about the true nature of his reality and his role in the war against its controllers.', 'https://example.com/poster8.jpg'),
('The Lion King', 1994, 'Animation, Adventure, Drama', 'Roger Allers, Rob Minkoff', 88, 'Lion prince Simba and his father are targeted by his bitter uncle, who wants to ascend the throne himself.', 'https://example.com/poster9.jpg'),
('Gladiator', 2000, 'Action, Adventure, Drama', 'Ridley Scott', 155, 'A former Roman General sets out to exact vengeance against the corrupt emperor who murdered his family and sent him into slavery.', 'https://example.com/poster10.jpg');

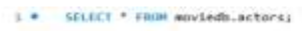


```
1 •   SELECT * FROM moviedb.movies;
```

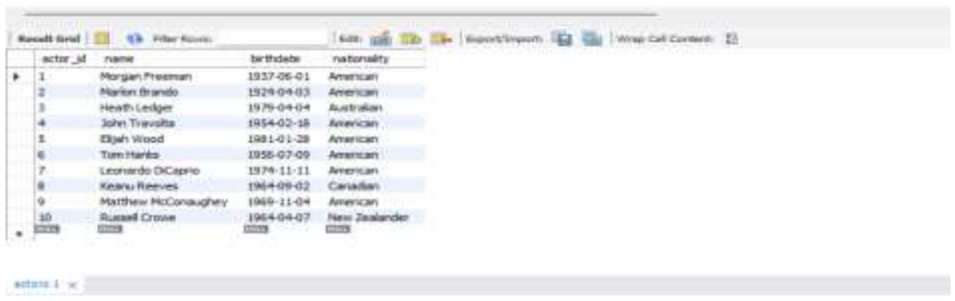
movie_id	title	release_year	genre	director	runtime	plot	p
1	The Shawshank Redemption	1994	Drama	Frank Darabont	142	Two imprisoned men bond over a number of ye...	ht
2	The Godfather	1972	Crime, Drama	Francis Ford Coppola	175	The aging patriarch of an organized crime dyna...	ht
3	The Dark Knight	2008	Action, Crime, Drama	Christopher Nolan	152	When the menace known as The Joker emerges...	ht
4	Pulp Fiction	1994	Crime, Drama	Quentin Tarantino	154	The lives of two mob hitmen, a boxer, a gangst...	ht
5	The Lord of the Rings: The Return of the King	2003	Adventure, Drama, Fantasy	Peter Jackson	201	Gandalf and Aragorn lead the World of Men aga...	ht
6	Forrest Gump	1994	Drama, Romance	Robert Zemeckis	142	The presidencies of Kennedy and Johnson, the ...	ht
7	Inception	2010	Action, Adventure, Sci-Fi	Christopher Nolan	148	A thief who steals corporate secrets through th...	ht
8	The Matrix	1999	Action, Sci-Fi	Lana Wachowski, Lilly Wachowski	136	A computer hacker learns from mysterious rebel...	ht
9	The Lion King	1994	Animation, Adventure, Drama	Roger Allers, Rob Minkoff	88	Lion prince Simba and his father are targeted b...	ht
10	Gladiator	2000	Action, Adventure, Drama	Ridley Scott	155	A former Roman General sets out to exact veng...	ht

2. Insert data into the Actors table
   INSERT INTO Actors (name, birthdate, nationality) VALUES
   ('Morgan Freeman', '1937-06-01', 'American'),
   ('Marlon Brando', '1924-04-03', 'American'),

```
('Heath Ledger', '1979-04-04', 'Australian'),
('John Travolta', '1954-02-18', 'American'),
('Elijah Wood', '1981-01-28', 'American'),
('Tom Hanks', '1956-07-09', 'American'),
('Leonardo DiCaprio', '1974-11-11', 'American'),
('Keanu Reeves', '1964-09-02', 'Canadian'),
('Matthew McConaughey', '1969-11-04', 'American'),
('Russell Crowe', '1964-04-07', 'New Zealander');
```



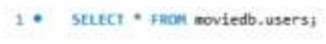
```
SELECT * FROM moviedb.actors;
```



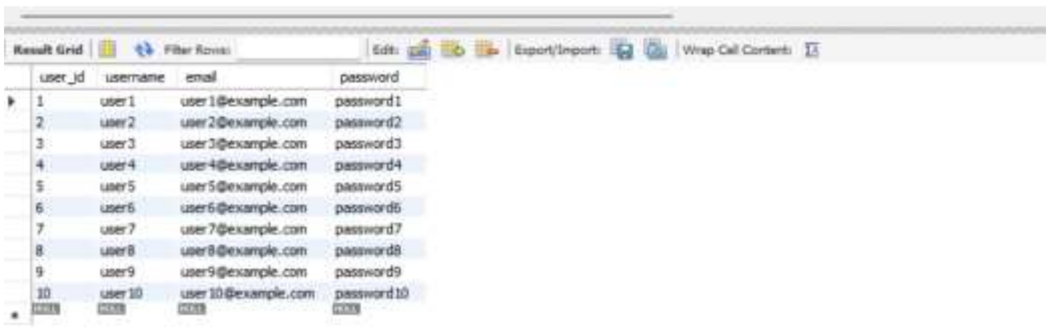
```
actor_id  name                 birthdate   nationality
1         Morgan Freeman       1937-06-01  American
2         Marlon Brando        1924-04-03  American
3         Heath Ledger         1979-04-04  Australian
4         John Travolta        1954-02-18  American
5         Elijah Wood          1981-01-28  American
6         Tom Hanks            1956-07-09  American
7         Leonardo DiCaprio    1974-11-11  American
8         Keanu Reeves         1964-09-02  Canadian
9         Matthew McConaughey  1969-11-04  American
10        Russell Crowe        1964-04-07  New Zealander
NULL      NULL                 NULL        NULL
```

3. Insert data into the Users table

```
INSERT INTO Users (username, email, password) VALUES
('user1', 'user1@example.com', 'password1'),
('user2', 'user2@example.com', 'password2'),
('user3', 'user3@example.com', 'password3'),
('user4', 'user4@example.com', 'password4'),
('user5', 'user5@example.com', 'password5'),
('user6', 'user6@example.com', 'password6'),
('user7', 'user7@example.com', 'password7'),
('user8', 'user8@example.com', 'password8'),
('user9', 'user9@example.com', 'password9'),
('user10', 'user10@example.com', 'password10');
```



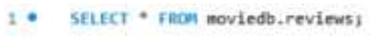
```
SELECT * FROM moviedb.users;
```



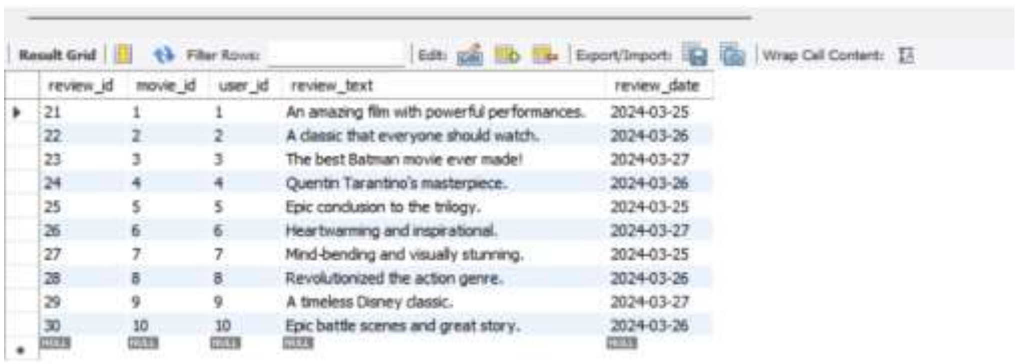
user_id	username	email	password
1	user1	user1@example.com	password1
2	user2	user2@example.com	password2
3	user3	user3@example.com	password3
4	user4	user4@example.com	password4
5	user5	user5@example.com	password5
6	user6	user6@example.com	password6
7	user7	user7@example.com	password7
8	user8	user8@example.com	password8
9	user9	user9@example.com	password9
10	user10	user10@example.com	password10
NULL	NULL	NULL	NULL

4. Insert data into the Reviews table

INSERT INTO Reviews (movie_id, user_id, review_text, review_date) VALUES
(1, 1, 'An amazing film with powerful performances.', '2024-03-25'),
(2, 2, 'A classic that everyone should watch.', '2024-03-26'),
(3, 3, 'The best Batman movie ever made!', '2024-03-27'),
(4, 4, 'Quentin Tarantino\'s masterpiece.', '2024-03-26'),
(5, 5, 'Epic conclusion to the trilogy.', '2024-03-25'),
(6, 6, 'Heartwarming and inspirational.', '2024-03-27'),
(7, 7, 'Mind-bending and visually stunning.', '2024-03-25'),
(8, 8, 'Revolutionized the action genre.', '2024-03-26'),
(9, 9, 'A timeless Disney classic.', '2024-03-27'),
(10, 10, 'Epic battle scenes and great story.', '2024-03-26')



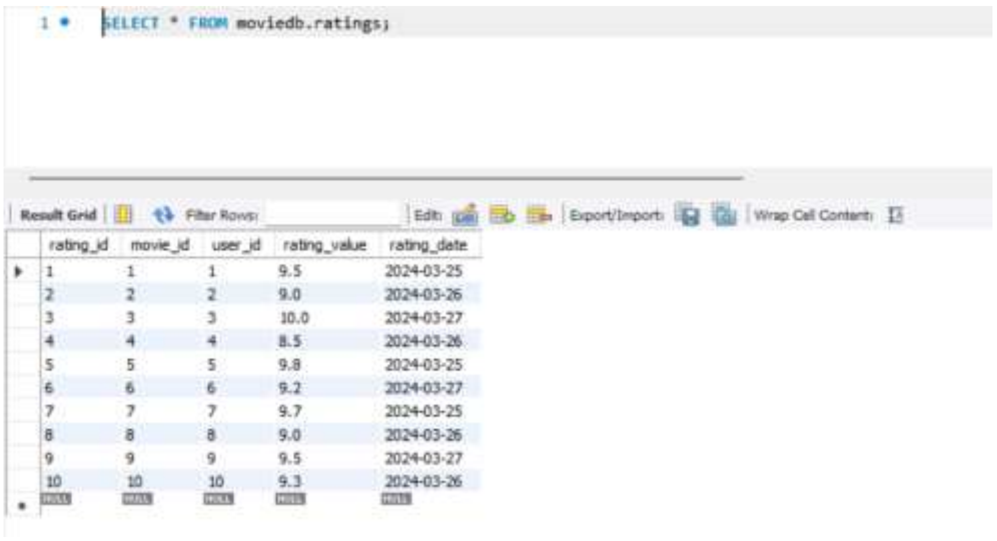
```
SELECT * FROM moviedb.reviews;
```



review_id	movie_id	user_id	review_text	review_date
21	1	1	An amazing film with powerful performances.	2024-03-25
22	2	2	A classic that everyone should watch.	2024-03-26
23	3	3	The best Batman movie ever made!	2024-03-27
24	4	4	Quentin Tarantino's masterpiece.	2024-03-26
25	5	5	Epic conclusion to the trilogy.	2024-03-25
26	6	6	Heartwarming and inspirational.	2024-03-27
27	7	7	Mind-bending and visually stunning.	2024-03-25
28	8	8	Revolutionized the action genre.	2024-03-26
29	9	9	A timeless Disney classic.	2024-03-27
30	10	10	Epic battle scenes and great story.	2024-03-26
NULL	NULL	NULL	NULL	NULL

5. Insert data into the Ratings table

```
INSERT INTO Ratings (movie_id, user_id, rating_value, rating_date) VALUES
(1, 1, 9.5, '2024-03-25'),
(2, 2, 9.0, '2024-03-26'),
(3, 3, 10.0, '2024-03-27'),
(4, 4, 8.5, '2024-03-26'),
(5, 5, 9.8, '2024-03-25'),
(6, 6, 9.2, '2024-03-27'),
(7, 7, 9.7, '2024-03-25'),
(8, 8, 9.0, '2024-03-26'),
(9, 9, 9.5, '2024-03-27'),
(10, 10, 9.3, '2024-03-26');
```

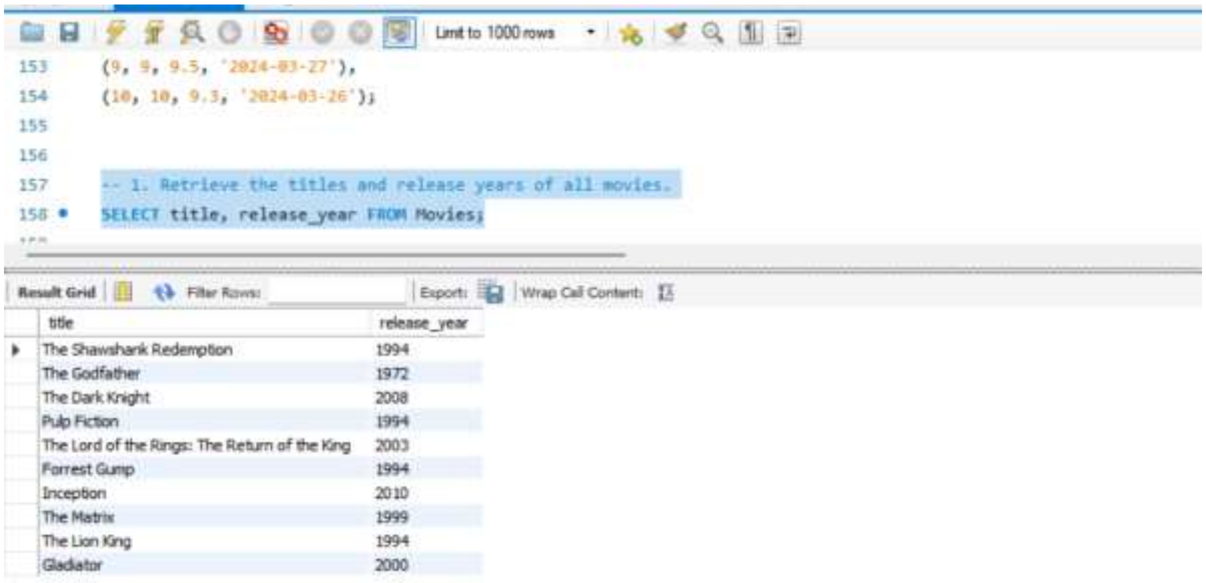


```
SELECT * FROM moviedb.ratings;
```

rating_id	movie_id	user_id	rating_value	rating_date
1	1	1	9.5	2024-03-25
2	2	2	9.0	2024-03-26
3	3	3	10.0	2024-03-27
4	4	4	8.5	2024-03-26
5	5	5	9.8	2024-03-25
6	6	6	9.2	2024-03-27
7	7	7	9.7	2024-03-25
8	8	8	9.0	2024-03-26
9	9	9	9.5	2024-03-27
10	10	10	9.3	2024-03-26
NULL	NULL	NULL	NULL	NULL

6. Retrieve the titles and release years of all movies.

```
SELECT title, release_year FROM Movies;
```

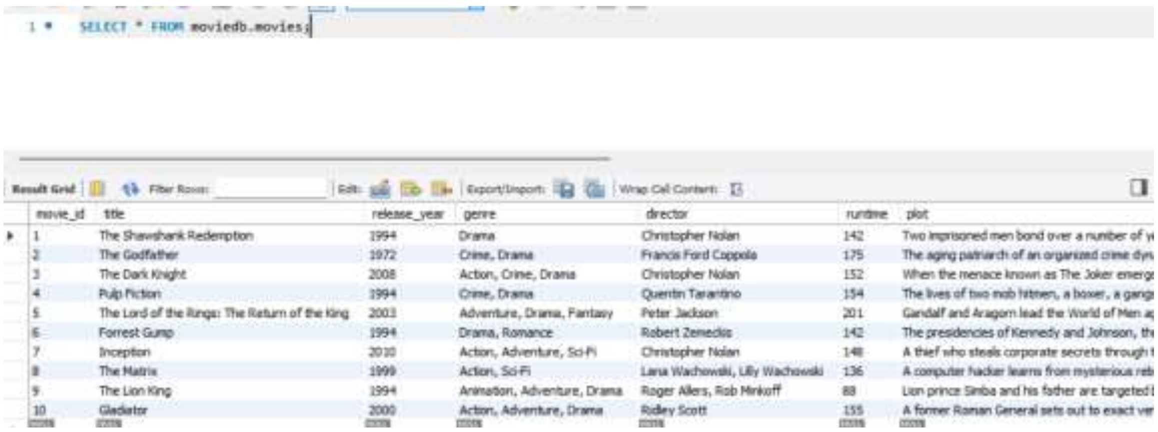


```
(9, 9, 9.5, '2024-03-27'),
(10, 10, 9.3, '2024-03-26');


-- 1. Retrieve the titles and release years of all movies.
SELECT title, release_year FROM Movies;
```

title	release_year
The Shawshank Redemption	1994
The Godfather	1972
The Dark Knight	2008
Pulp Fiction	1994
The Lord of the Rings: The Return of the King	2003
Forrest Gump	1994
Inception	2010
The Matrix	1999
The Lion King	1994
Gladiator	2000

7. Update the director of the movie with movie_id 1 to 'Christopher Nolan'.

UPDATE Movies SET director = 'Christopher Nolan' WHERE movie_id = 1;

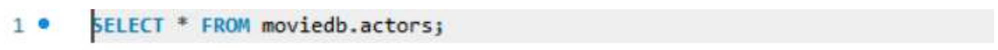


```
SELECT * FROM moviedb.movies;
```

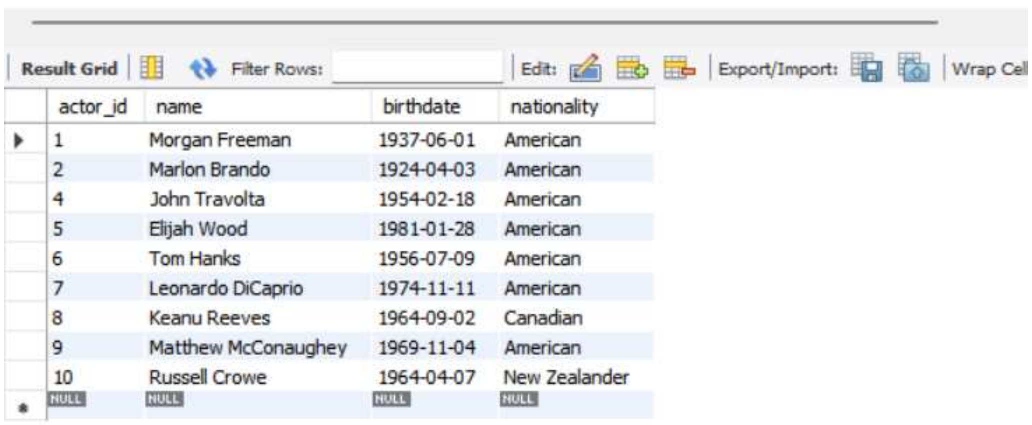
movie_id	title	release_year	genre	director	runtime	plot
1	The Shawshank Redemption	1994	Drama	Christopher Nolan	142	Two imprisoned men bond over a number of y
2	The Godfather	1972	Crime, Drama	Francis Ford Coppola	175	The aging patriarch of an organized crime dyn
3	The Dark Knight	2008	Action, Crime, Drama	Christopher Nolan	152	When the menace known as The Joker energ
4	Pulp Fiction	1994	Crime, Drama	Quentin Tarantino	154	The lives of two mob hitmen, a boxer, a gang
5	The Lord of the Rings: The Return of the King	2003	Adventure, Drama, Fantasy	Peter Jackson	201	Gandalf and Aragorn lead the World of Men ag
6	Forrest Gump	1994	Drama, Romance	Robert Zemeckis	142	The presidencies of Kennedy and Johnson, th
7	Inception	2010	Action, Adventure, Sci-Fi	Christopher Nolan	148	A thief who steals corporate secrets through t
8	The Matrix	1999	Action, Sci-Fi	Lana Wachowski, Lilly Wachowski	136	A computer hacker learns from mysterious reb
9	The Lion King	1994	Animation, Adventure, Drama	Roger Allers, Rob Minkoff	88	Lion prince Simba and his father are targeted b
10	Gladiator	2000	Action, Adventure, Drama	Ridley Scott	155	A former Roman General sets out to exact ver

8. Delete the actor with actor_id 3 from the Actors table.

DELETE FROM Actors WHERE actor_id = 3;

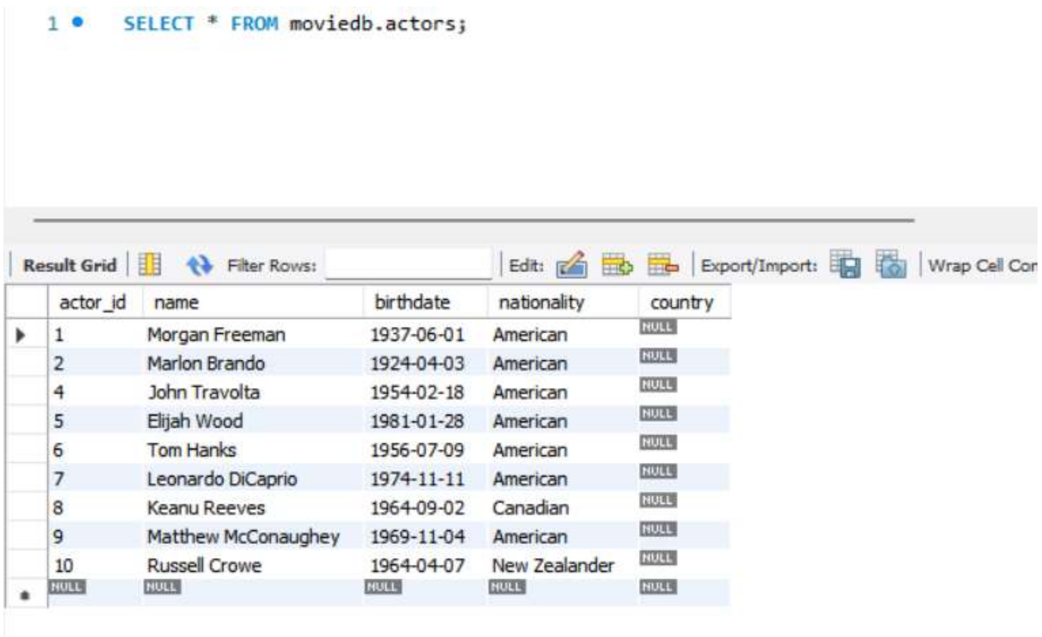


```
SELECT * FROM moviedb.actors;
```



actor_id	name	birthdate	nationality
1	Morgan Freeman	1937-06-01	American
2	Marlon Brando	1924-04-03	American
4	John Travolta	1954-02-18	American
5	Elijah Wood	1981-01-28	American
6	Tom Hanks	1956-07-09	American
7	Leonardo DiCaprio	1974-11-11	American
8	Keanu Reeves	1964-09-02	Canadian
9	Matthew McConaughey	1969-11-04	American
10	Russell Crowe	1964-04-07	New Zealander
NULL	NULL	NULL	NULL

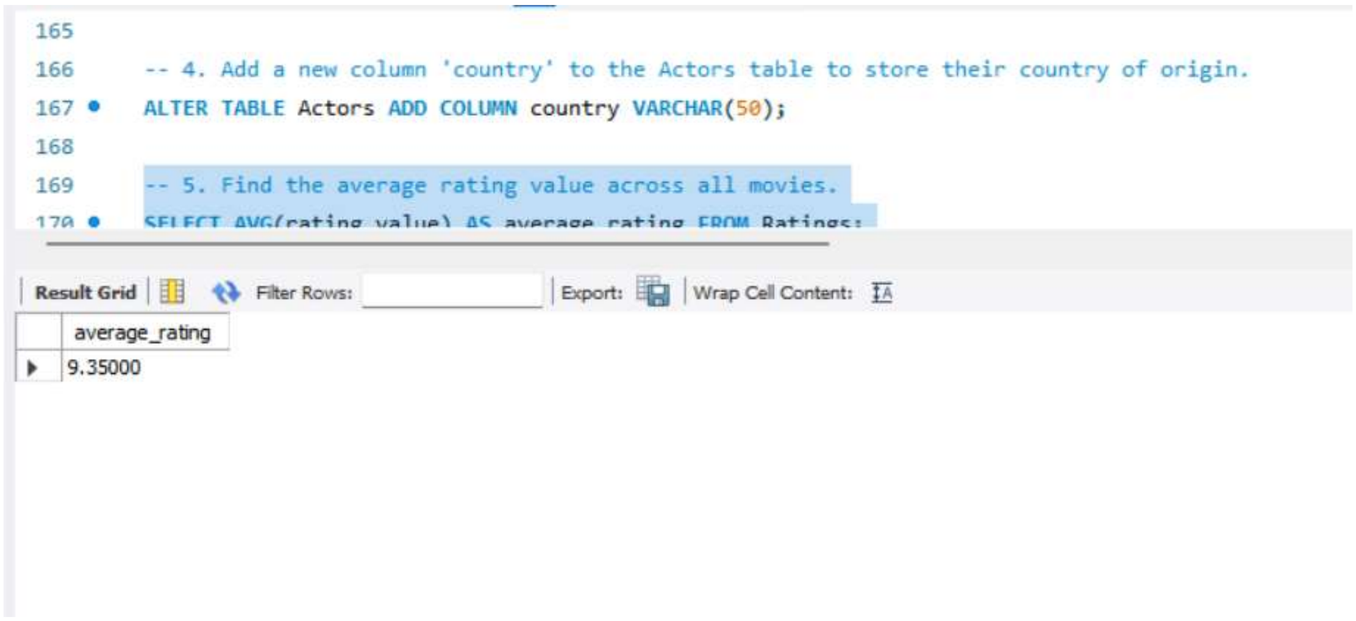
9. Add a new column 'country' to the Actors table to store their country of origin.
ALTER TABLE Actors ADD COLUMN country VARCHAR(50);



```
SELECT * FROM moviedb.actors;
```

actor_id	name	birthdate	nationality	country
1	Morgan Freeman	1937-06-01	American	NULL
2	Marlon Brando	1924-04-03	American	NULL
4	John Travolta	1954-02-18	American	NULL
5	Elijah Wood	1981-01-28	American	NULL
6	Tom Hanks	1956-07-09	American	NULL
7	Leonardo DiCaprio	1974-11-11	American	NULL
8	Keanu Reeves	1964-09-02	Canadian	NULL
9	Matthew McConaughey	1969-11-04	American	NULL
10	Russell Crowe	1964-04-07	New Zealander	NULL
NULL	NULL	NULL	NULL	NULL

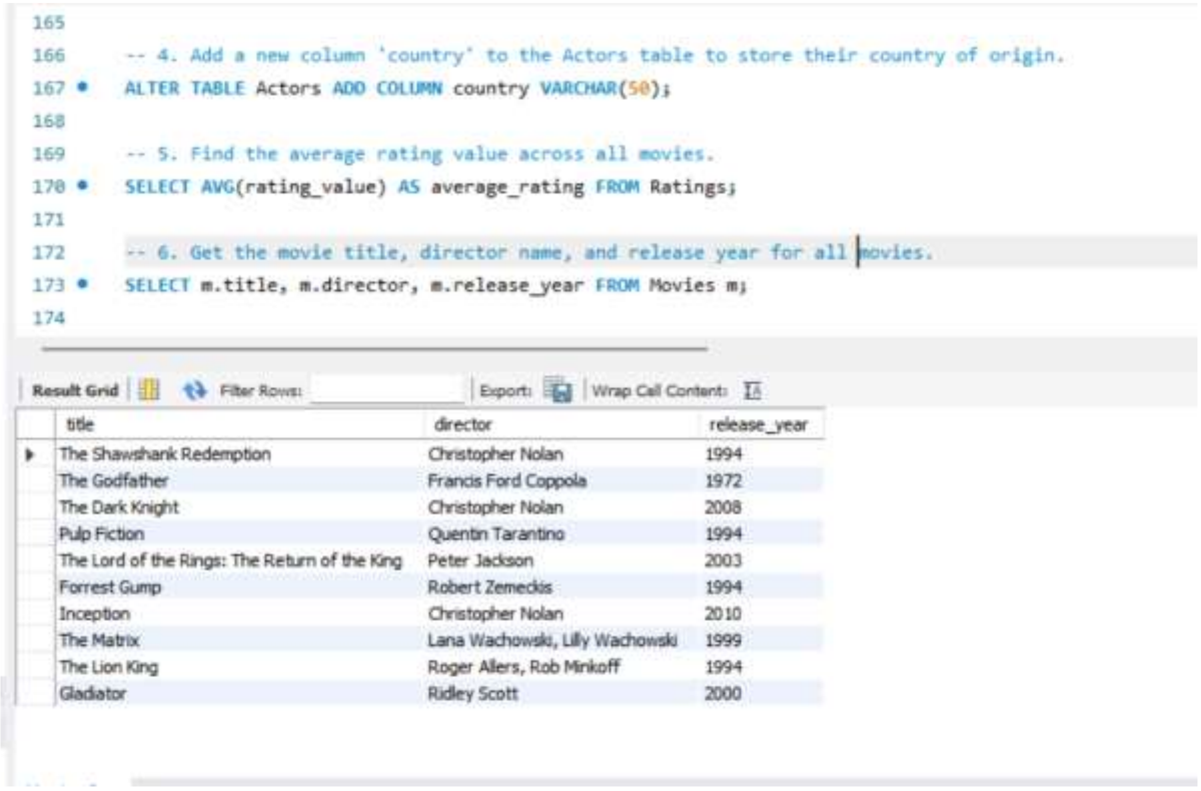
10. Find the average rating value across all movies.
SELECT AVG(rating_value) AS average_rating FROM Ratings;



```
-- 4. Add a new column 'country' to the Actors table to store their country of origin.
ALTER TABLE Actors ADD COLUMN country VARCHAR(50);

-- 5. Find the average rating value across all movies.
SELECT AVG(rating_value) AS average_rating FROM Ratings;
```

average_rating
9.35000

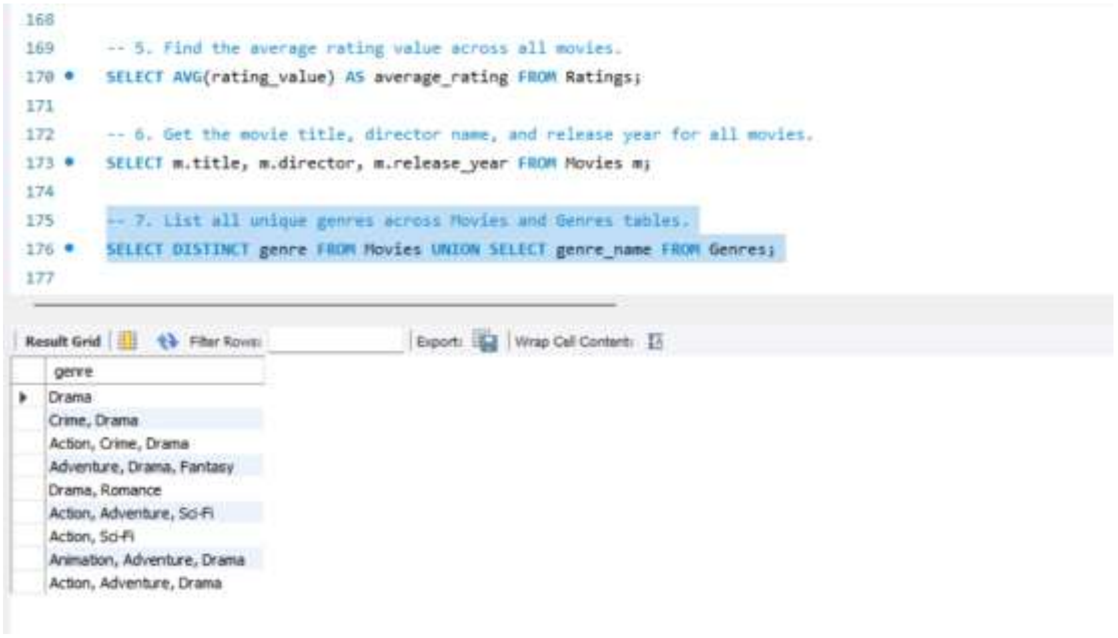
11. Get the movie title, director name, and release year for all movies.
SELECT m.title, m.director, m.release_year FROM Movies m;



```
-- 4. Add a new column 'country' to the Actors table to store their country of origin.
ALTER TABLE Actors ADD COLUMN country VARCHAR(50);

-- 5. Find the average rating value across all movies.
SELECT AVG(rating_value) AS average_rating FROM Ratings;

-- 6. Get the movie title, director name, and release year for all movies.
SELECT m.title, m.director, m.release_year FROM Movies m;
```

title	director	release_year
The Shawshank Redemption	Christopher Nolan	1994
The Godfather	Francis Ford Coppola	1972
The Dark Knight	Christopher Nolan	2008
Pulp Fiction	Quentin Tarantino	1994
The Lord of the Rings: The Return of the King	Peter Jackson	2003
Forrest Gump	Robert Zemeckis	1994
Inception	Christopher Nolan	2010
The Matrix	Lana Wachowski, Lilly Wachowski	1999
The Lion King	Roger Allers, Rob Minkoff	1994
Gladiator	Ridley Scott	2000

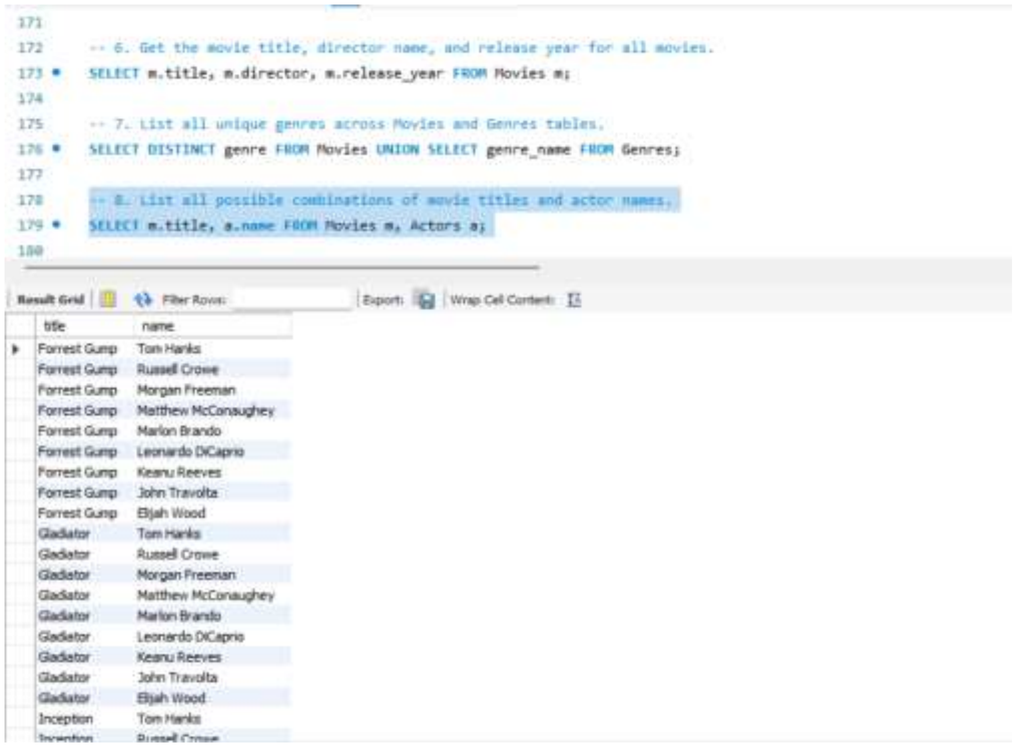
12. List all unique genres across Movies and Genres tables.
SELECT DISTINCT genre FROM Movies UNION SELECT genre_name FROM Genres;



```
-- 5. Find the average rating value across all movies.
SELECT AVG(rating_value) AS average_rating FROM Ratings;

-- 6. Get the movie title, director name, and release year for all movies.
SELECT m.title, m.director, m.release_year FROM Movies m;

-- 7. List all unique genres across Movies and Genres tables.
SELECT DISTINCT genre FROM Movies UNION SELECT genre_name FROM Genres;
```

genre
Drama
Crime, Drama
Action, Crime, Drama
Adventure, Drama, Fantasy
Drama, Romance
Action, Adventure, Sci-Fi
Action, Sci-Fi
Animation, Adventure, Drama
Action, Adventure, Drama

13. List all possible combinations of movie titles and actor names.
SELECT m.title, a.name FROM Movies m, Actors a;

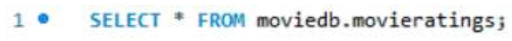


```
-- 6. Get the movie title, director name, and release year for all movies.
SELECT m.title, m.director, m.release_year FROM Movies m;

-- 7. List all unique genres across Movies and Genres tables.
SELECT DISTINCT genre FROM Movies UNION SELECT genre_name FROM Genres;

-- 8. List all possible combinations of movie titles and actor names.
SELECT m.title, a.name FROM Movies m, Actors a;
```

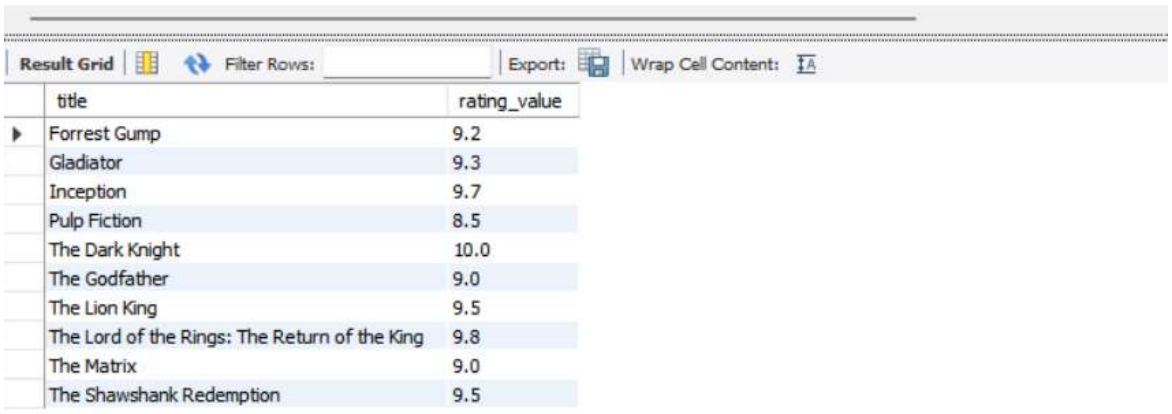
title	name
Forrest Gump	Tom Hanks
Forrest Gump	Russell Crowe
Forrest Gump	Morgan Freeman
Forrest Gump	Matthew McConaughey
Forrest Gump	Marlon Brando
Forrest Gump	Leonardo DiCaprio
Forrest Gump	Keanu Reeves
Forrest Gump	John Travolta
Forrest Gump	Elijah Wood
Gladiator	Tom Hanks
Gladiator	Russell Crowe
Gladiator	Morgan Freeman
Gladiator	Matthew McConaughey
Gladiator	Marlon Brando
Gladiator	Leonardo DiCaprio
Gladiator	Keanu Reeves
Gladiator	John Travolta
Gladiator	Elijah Wood
Inception	Tom Hanks
Inception	Russell Crowe

14. Create a view named "MovieRatings" to display movie titles and their corresponding ratings.
CREATE VIEW MovieRatings AS

SELECT m.title, r.rating_value FROM Movies m
JOIN Ratings r ON m.movie_id = r.movie_id;

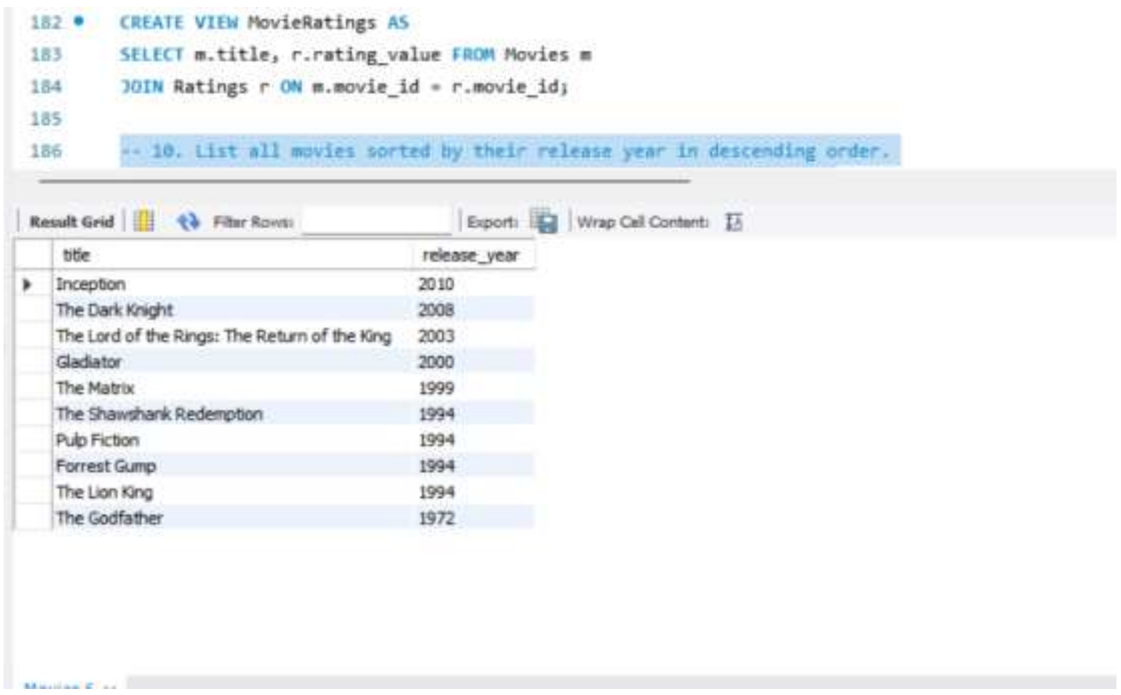


```
SELECT * FROM moviedb.movieratings;
```



title	rating_value
Forrest Gump	9.2
Gladiator	9.3
Inception	9.7
Pulp Fiction	8.5
The Dark Knight	10.0
The Godfather	9.0
The Lion King	9.5
The Lord of the Rings: The Return of the King	9.8
The Matrix	9.0
The Shawshank Redemption	9.5

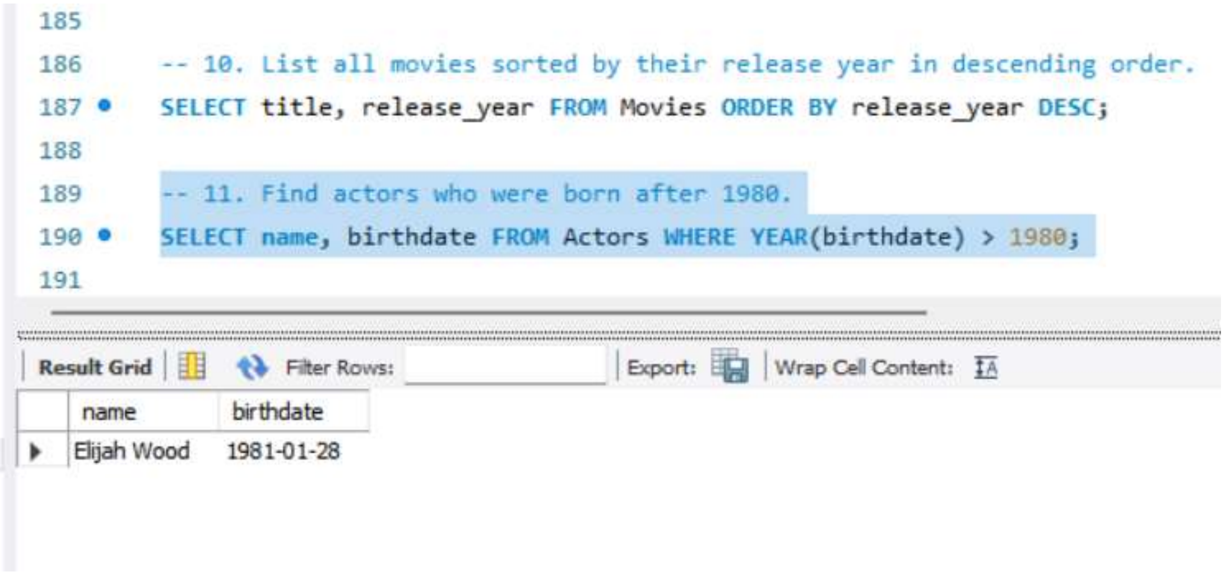
15. List all movies sorted by their release year in descending order.
SELECT title, release_year FROM Movies ORDER BY release_year DESC;



```
CREATE VIEW MovieRatings AS
SELECT m.title, r.rating_value FROM Movies m
JOIN Ratings r ON m.movie_id = r.movie_id;

-- 10. List all movies sorted by their release year in descending order.
```

title	release_year
Inception	2010
The Dark Knight	2008
The Lord of the Rings: The Return of the King	2003
Gladiator	2000
The Matrix	1999
The Shawshank Redemption	1994
Pulp Fiction	1994
Forrest Gump	1994
The Lion King	1994
The Godfather	1972

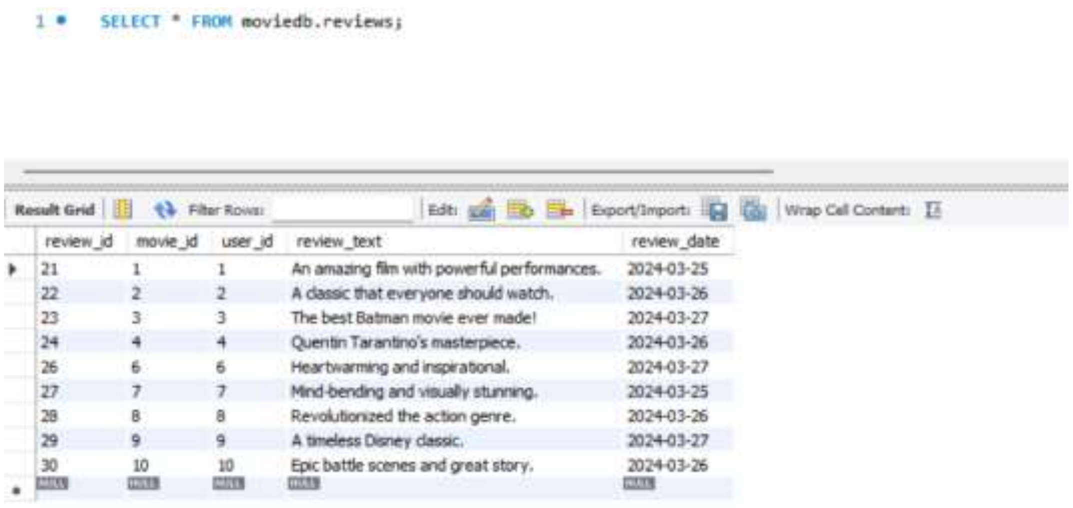
16. Find actors who were born after 1980.
SELECT name, birthdate FROM Actors WHERE YEAR(birthdate) > 1980;



```

-- 10. List all movies sorted by their release year in descending order.
SELECT title, release_year FROM Movies ORDER BY release_year DESC;

-- 11. Find actors who were born after 1980.
SELECT name, birthdate FROM Actors WHERE YEAR(birthdate) > 1980;

```

name	birthdate
Elijah Wood	1981-01-28

17. Delete the review with review_id 5 from the Reviews table.
DELETE FROM Reviews WHERE review_id = 25;

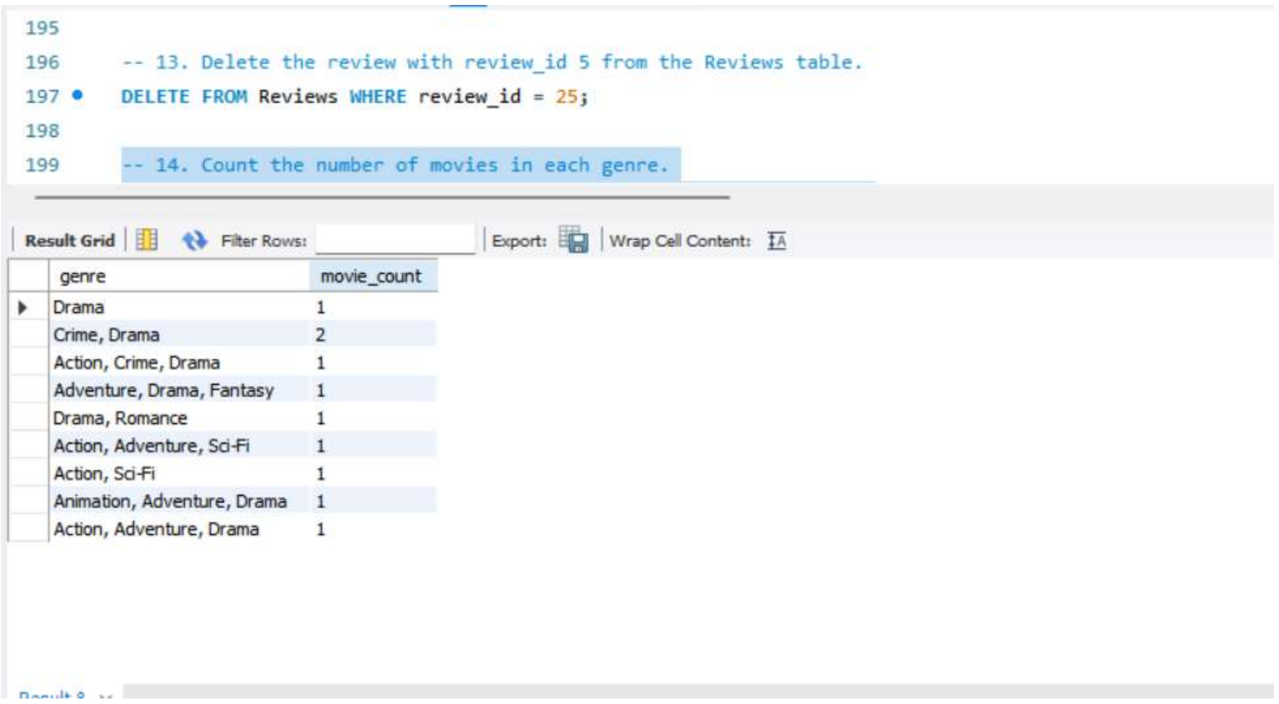


```
SELECT * FROM moviedb.reviews;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

review_id	movie_id	user_id	review_text	review_date
21	1	1	An amazing film with powerful performances.	2024-03-25
22	2	2	A classic that everyone should watch.	2024-03-26
23	3	3	The best Batman movie ever made!	2024-03-27
24	4	4	Quentin Tarantino's masterpiece.	2024-03-26
26	6	6	Heartwarming and inspirational.	2024-03-27
27	7	7	Mind-bending and visually stunning.	2024-03-25
28	8	8	Revolutionized the action genre.	2024-03-26
29	9	9	A timeless Disney classic.	2024-03-27
30	10	10	Epic battle scenes and great story.	2024-03-26
NULL	NULL	NULL	NULL	NULL

18. Count the number of movies in each genre.
SELECT genre, COUNT(*) AS movie_count FROM Movies GROUP BY genre;

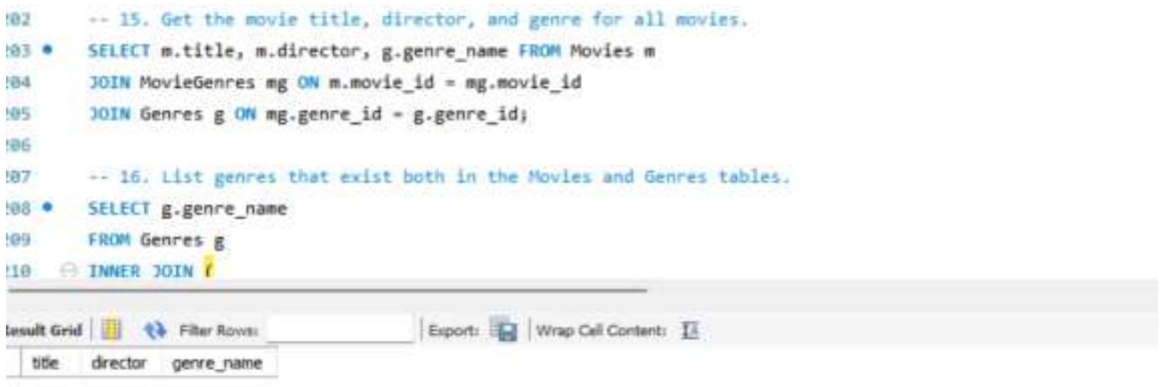


```

-- 13. Delete the review with review_id 5 from the Reviews table.
DELETE FROM Reviews WHERE review_id = 25;

-- 14. Count the number of movies in each genre.
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

genre	movie_count
Drama	1
Crime, Drama	2
Action, Crime, Drama	1
Adventure, Drama, Fantasy	1
Drama, Romance	1
Action, Adventure, Sci-Fi	1
Action, Sci-Fi	1
Animation, Adventure, Drama	1
Action, Adventure, Drama	1

19. Get the movie title, director, and genre for all movies.
SELECT m.title, m.director, g.genre_name FROM Movies m
JOIN MovieGenres mg ON m.movie_id = mg.movie_id
JOIN Genres g ON mg.genre_id = g.genre_id;

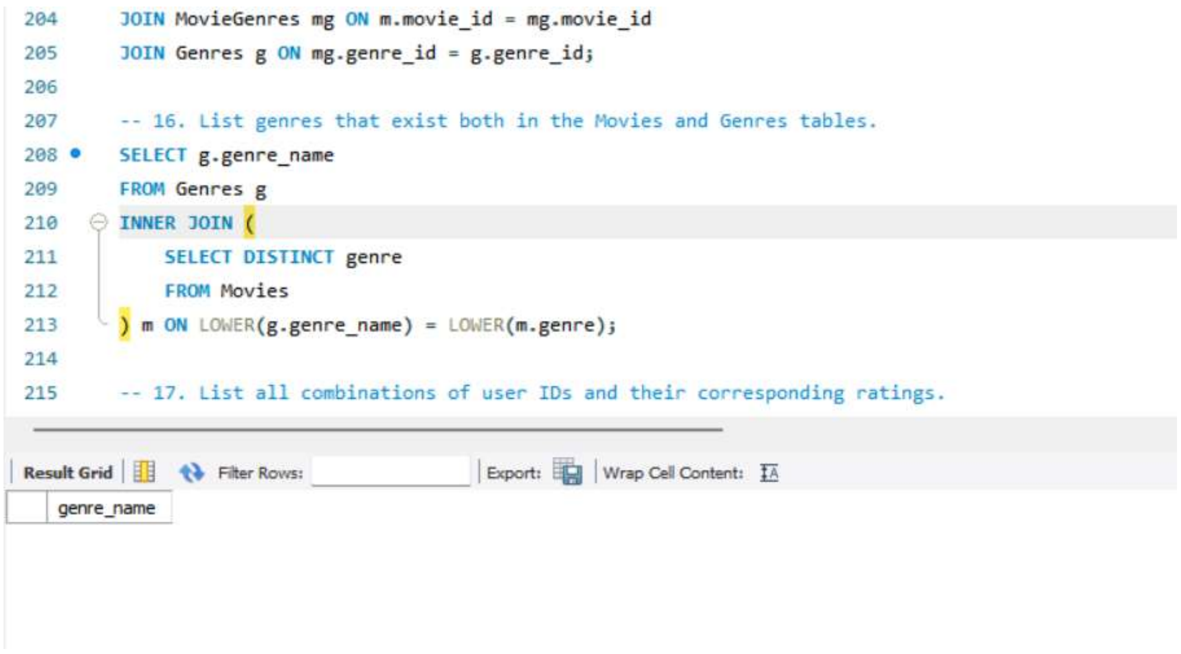


```
-- 15. Get the movie title, director, and genre for all movies.
SELECT m.title, m.director, g.genre_name FROM Movies m
JOIN MovieGenres mg ON m.movie_id = mg.movie_id
JOIN Genres g ON mg.genre_id = g.genre_id;

-- 16. List genres that exist both in the Movies and Genres tables.
SELECT g.genre_name
FROM Genres g
INNER JOIN (
```

title	director	genre_name

20. List genres that exist both in the Movies and Genres tables.

```
SELECT g.genre_name
FROM Genres g
INNER JOIN (
    SELECT DISTINCT genre
    FROM Movies
) m ON LOWER(g.genre_name) = LOWER(m.genre);
```

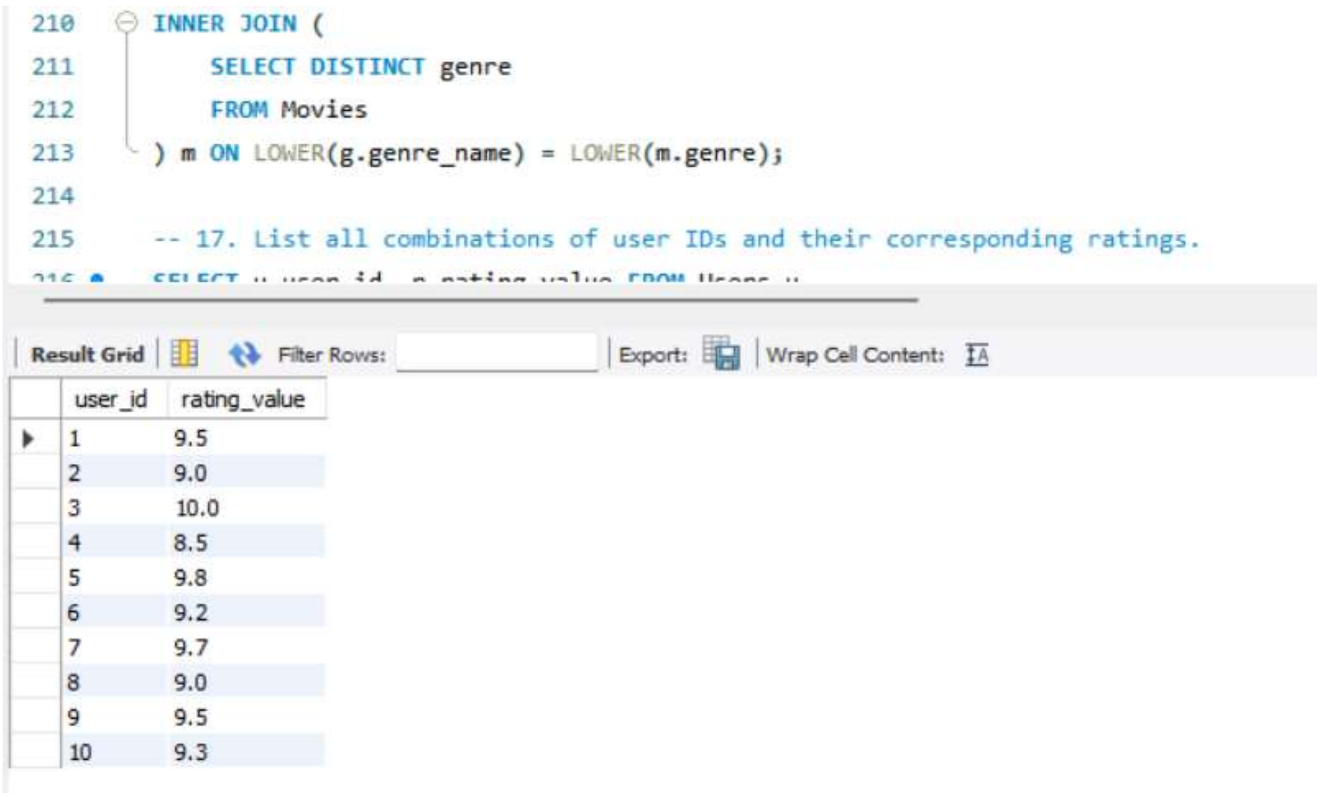


```
JOIN MovieGenres mg ON m.movie_id = mg.movie_id
JOIN Genres g ON mg.genre_id = g.genre_id;

-- 16. List genres that exist both in the Movies and Genres tables.
SELECT g.genre_name
FROM Genres g
INNER JOIN (
    SELECT DISTINCT genre
    FROM Movies
) m ON LOWER(g.genre_name) = LOWER(m.genre);

-- 17. List all combinations of user IDs and their corresponding ratings.
```

genre_name

21. List all combinations of user IDs and their corresponding ratings.

```
SELECT u.user_id, r.rating_value FROM Users u
JOIN Ratings r ON u.user_id = r.user_id;
```

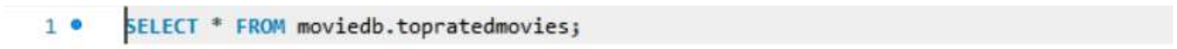


```
INNER JOIN (
    SELECT DISTINCT genre
    FROM Movies
) m ON LOWER(g.genre_name) = LOWER(m.genre);

-- 17. List all combinations of user IDs and their corresponding ratings.
```

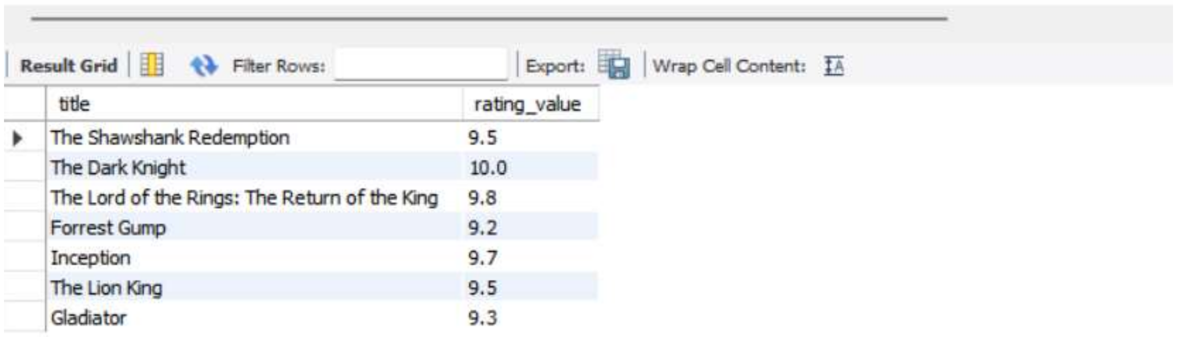
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

user_id	rating_value
1	9.5
2	9.0
3	10.0
4	8.5
5	9.8
6	9.2
7	9.7
8	9.0
9	9.5
10	9.3

22. Create a view named "TopRatedMovies" to display movie titles with ratings above 9.0.
CREATE VIEW TopRatedMovies AS
SELECT m.title, r.rating_value FROM Movies m
JOIN Ratings r ON m.movie_id = r.movie_id
WHERE r.rating_value > 9.0;



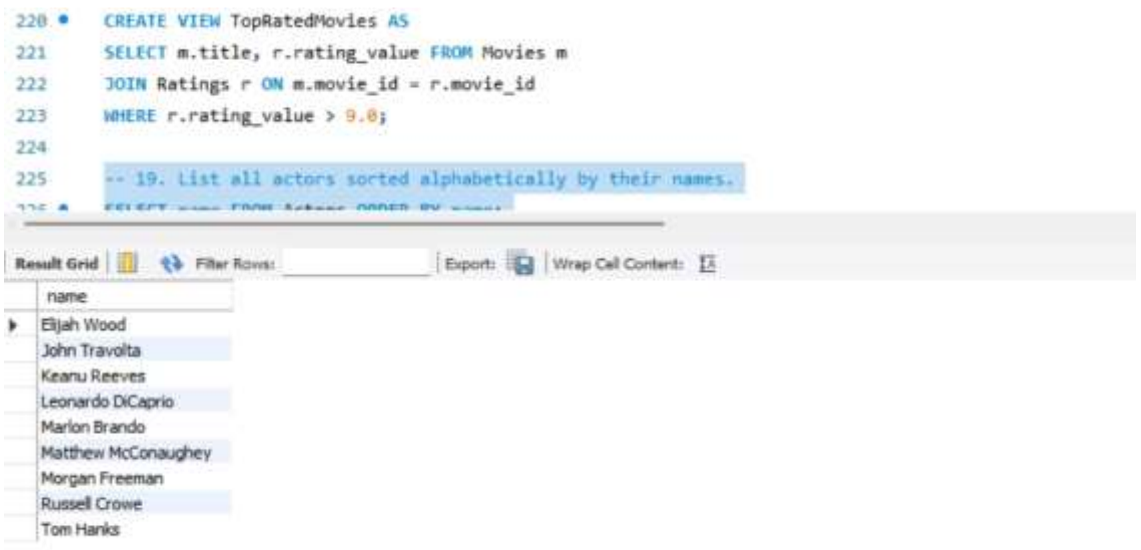
```
SELECT * FROM moviedb.topratedmovies;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

title	rating_value
The Shawshank Redemption	9.5
The Dark Knight	10.0
The Lord of the Rings: The Return of the King	9.8
Forrest Gump	9.2
Inception	9.7
The Lion King	9.5
Gladiator	9.3

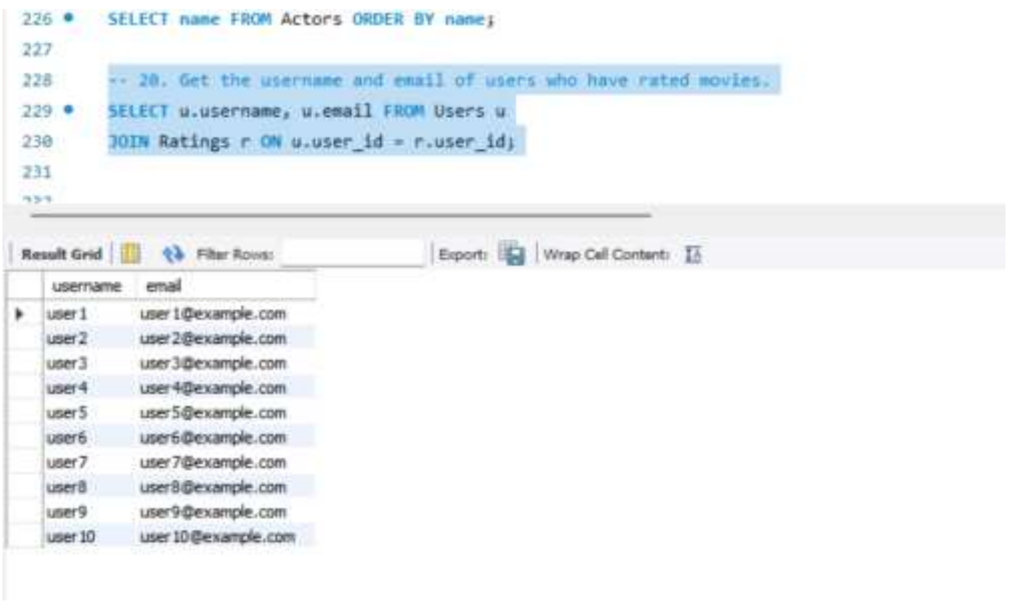
23. List all actors sorted alphabetically by their names.
SELECT name FROM Actors ORDER BY name;



```
CREATE VIEW TopRatedMovies AS
SELECT m.title, r.rating_value FROM Movies m
JOIN Ratings r ON m.movie_id = r.movie_id
WHERE r.rating_value > 9.0;

-- 19. List all actors sorted alphabetically by their names.
```

name
Elijah Wood
John Travolta
Keanu Reeves
Leonardo DiCaprio
Marlon Brando
Matthew McConaughey
Morgan Freeman
Russell Crowe
Tom Hanks

24. Get the username and email of users who have rated movies.
SELECT u.username, u.email FROM Users u
JOIN Ratings r ON u.user_id = r.user_id;

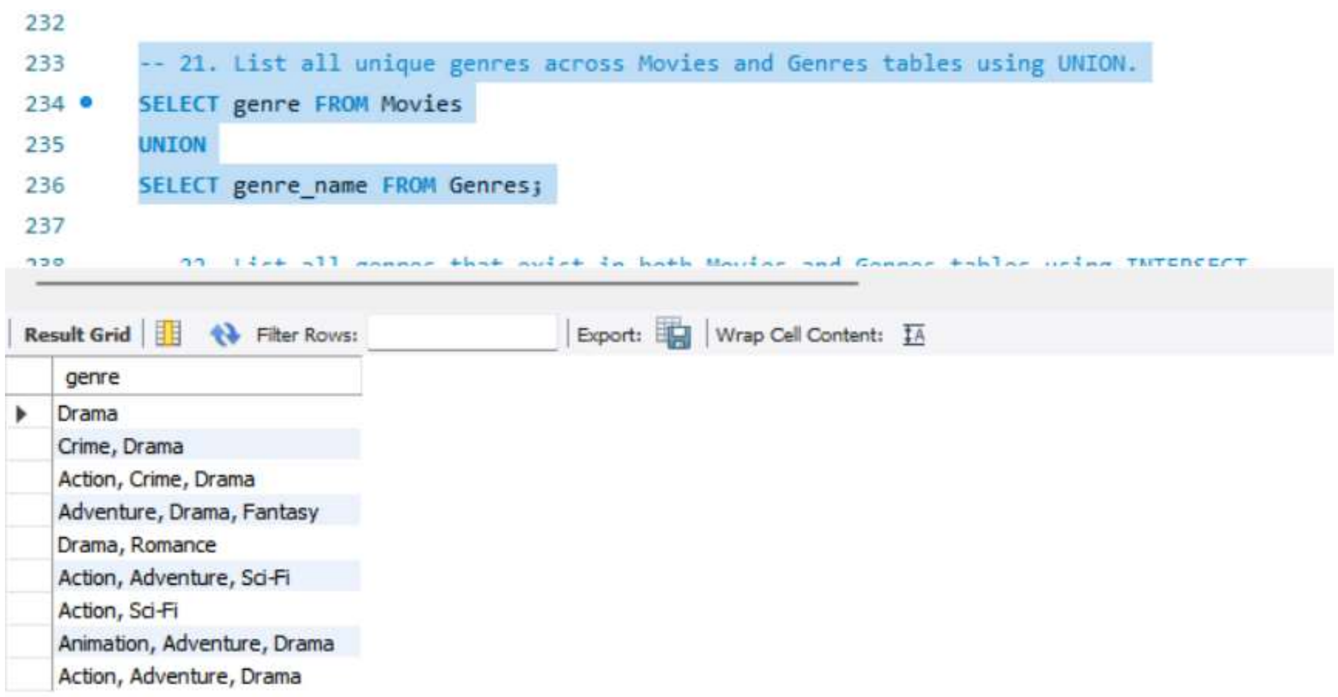


```
SELECT name FROM Actors ORDER BY name;

-- 20. Get the username and email of users who have rated movies.
SELECT u.username, u.email FROM Users u
JOIN Ratings r ON u.user_id = r.user_id;
```

username	email
user1	user1@example.com
user2	user2@example.com
user3	user3@example.com
user4	user4@example.com
user5	user5@example.com
user6	user6@example.com
user7	user7@example.com
user8	user8@example.com
user9	user9@example.com
user10	user10@example.com

25. List all unique genres across Movies and Genres tables using UNION.
SELECT genre FROM Movies

UNION
SELECT genre_name FROM Genres;

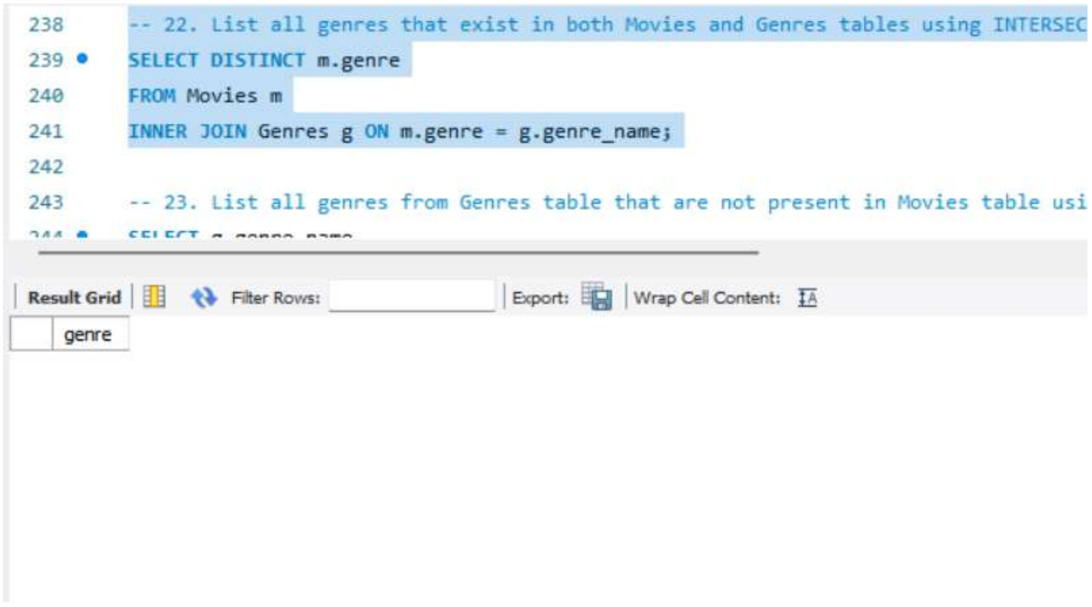


```
-- 21. List all unique genres across Movies and Genres tables using UNION.
SELECT genre FROM Movies
UNION
SELECT genre_name FROM Genres;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

```
genre
Drama
Crime, Drama
Action, Crime, Drama
Adventure, Drama, Fantasy
Drama, Romance
Action, Adventure, Sci-Fi
Action, Sci-Fi
Animation, Adventure, Drama
Action, Adventure, Drama
```

26. List all genres that exist in both Movies and Genres tables using INTERSECT
SELECT DISTINCT m.genre
FROM Movies m
INNER JOIN Genres g ON m.genre = g.genre_name;

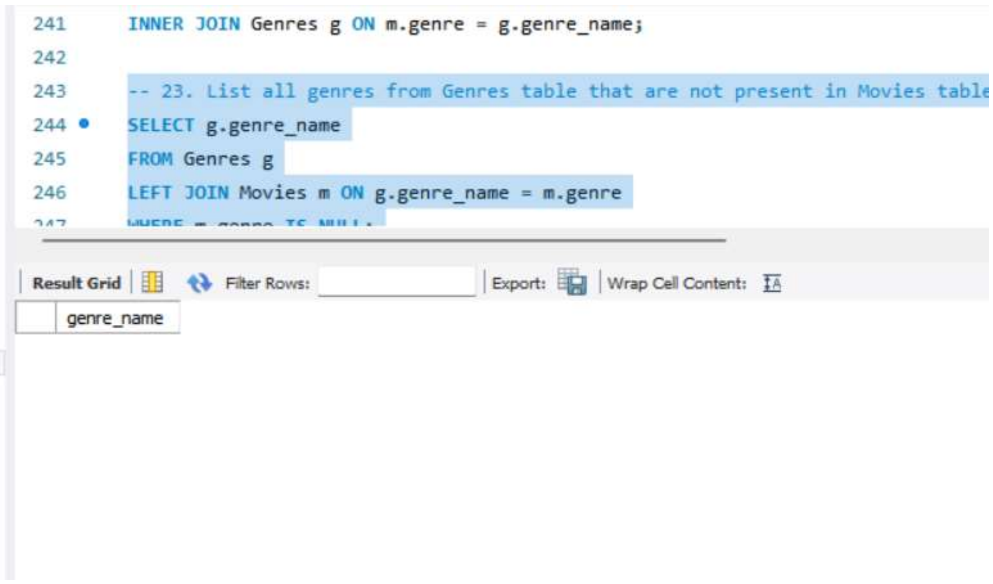


```
-- 22. List all genres that exist in both Movies and Genres tables using INTERSEC
SELECT DISTINCT m.genre
FROM Movies m
INNER JOIN Genres g ON m.genre = g.genre_name;

-- 23. List all genres from Genres table that are not present in Movies table usi
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

```
genre
```

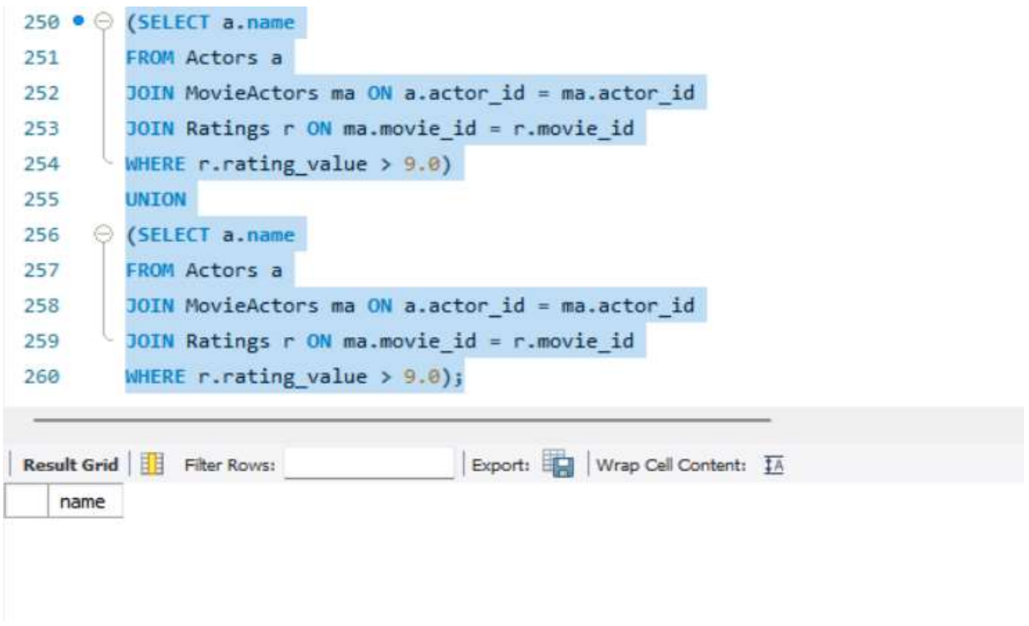
27. List all genres from Genres table that are not present in Movies table using EXCEPT
SELECT g.genre_name
FROM Genres g
LEFT JOIN Movies m ON g.genre_name = m.genre
WHERE m.genre IS NULL;



```
INNER JOIN Genres g ON m.genre = g.genre_name;

-- 23. List all genres from Genres table that are not present in Movies table
SELECT g.genre_name
FROM Genres g
LEFT JOIN Movies m ON g.genre_name = m.genre
```

```
genre_name
```

28. List all actors who have acted in movies with ratings above 9.0
(SELECT a.name
FROM Actors a
JOIN MovieActors ma ON a.actor_id = ma.actor_id
JOIN Ratings r ON ma.movie_id = r.movie_id
WHERE r.rating_value > 9.0)
UNION
(SELECT a.name
FROM Actors a
JOIN MovieActors ma ON a.actor_id = ma.actor_id
JOIN Ratings r ON ma.movie_id = r.movie_id
WHERE r.rating_value > 9.0);



```
(SELECT a.name
FROM Actors a
JOIN MovieActors ma ON a.actor_id = ma.actor_id
JOIN Ratings r ON ma.movie_id = r.movie_id
WHERE r.rating_value > 9.0)
UNION
(SELECT a.name
FROM Actors a
JOIN MovieActors ma ON a.actor_id = ma.actor_id
JOIN Ratings r ON ma.movie_id = r.movie_id
WHERE r.rating_value > 9.0);
```

	name

VI. Project demonstration

Tools used:
MySQL Workbench

VII. Self -Learning beyond classroom

Outside the formal confines of the classroom, we embarked on a journey of self-directed learning, exploring topics that piqued our curiosity and expanded our understanding of database management systems. We participated in online resources, such as tutorials, seminars, and documentation, delved into advanced SQL techniques, developed my skills in query optimization and database performance manipulation as well as hands-on activities such as building a mobile database system for personal use provided a rewarding practical experience Through directed learning efforts, we new not only gained technical skills but also a deeper appreciation for the real world in the application of database management.

VIII. Learning from the Project

This project was necessary to increase our understanding of database management principles by providing hands-on experience in real-world applications. This allowed us to apply theoretical skills in practical situations, honing our skills in programming, SQL execution, and user interface features. Collaborating with peers fostered teamwork and innovation, and managing project deadlines improved our time management. Overall, this project served as a catalyst for personal and professional growth, giving us valuable skills and confidence to successfully navigate the complexities of database management

IX. Challenges Faced

Several challenges arose in the development of a film database, each providing opportunities for growth and learning. An important challenge was ensuring the accuracy and precision of data in tables, especially in integrating applications such as analysis and presentation, which required careful consideration and strategizing does verify use to balance the need for flexibility with the need to maintain data integrity.

In addition, optimizing SQL queries for efficiency proved to be a recurring challenge. As database size and complexity grew, identifying and solving challenges in query execution became paramount. This required detailed engagement with query optimization techniques such as index manipulation and query rewriting to increase the responsiveness and scalability of the system.

X. Conclusion

In conclusion, the film database project has been a journey full of discoveries, collaborations and developments. Several key takeaways have resulted from this effort.

First, the project emphasized the importance of practical application in strengthening theoretical skills. Applying the concept of database management in a real-world setting gave us a deeper understanding of its complexities and nuances.

Second, cooperation proved necessary to overcome challenges and achieve success. Working closely with peers gave us the benefit of different perspectives and skill sets, leading to innovation and problem solving.

Third, they highlighted the importance of a user-centered design in creating an engaging and intuitive project experience. By prioritizing content priorities and repeatedly revising the user interface, we were able to increase engagement and satisfaction.

The latter work highlighted the iterative nature of software development. From initial design to final implementation, constant adaptation and modification was required to meet evolving needs and meet emerging challenges.

Overall, the film database project has been a rewarding and enriching experience, providing valuable skills, insights, and lessons that will serve us well in future endeavors.
Overall, the movie database project has been a rewarding and enriching experience, equipping us with valuable skills, insights, and lessons that will serve us well in future endeavors.