INTERNATIONAL INFORMATION TECHNOLOGY UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING

Project

Course: Database Theory

Topic: Movie Database System

Done by student(s)

Akmal Ashurmatov

Nurbek Dohdyrbek

Group : IT3-2205

Almaty 2025

CONTENTS

INTRODUCTION

DATABASE DESIGN

1. Subject Area Analysis and Conceptual Design
2. Logical Design
3. Physical Design and Content
4. Queries

CONCLUSION

REFERENCES

**INTRODUCTION**

This project is focused on analyzing and managing movie-related data using SQL. The main objective was to build a simplified version of an IMDb-style database and perform complex queries to extract meaningful insights. The development process included designing relational tables such as Movies, Directors, Actors, Genres, Reviews, and Awards, followed by inserting sample data.

After setting up the database structure, I created and executed various SQL queries to demonstrate filtering, joining, grouping, and aggregate functions. These queries helped answer real-world questions such as “Which movies received high ratings?”, “Who directed movies between certain years?”, and “Which actors played in specific films?”

The tools used in this project include PostgreSQL for database management and SQL for querying. Throughout the development, I followed best practices in relational database design and normalized the data for better performance and accuracy.

This project helped strengthen we skills in data manipulation, relational joins, and working with real-world data structures.

DATABASE DESIGN

1. Subject Area Analysis and Conceptual Design

**Part 1: Subject Area Analysis and Conceptual Design**

**1. Subject Area Analysis**

**1.1 Description of the Subject Area**

The database is designed to manage information related to movies, including details about movies, actors, directors, genres, awards, users, reviews, votes, and trailers. The database structure is based on IMDb-like data storage, allowing efficient organization and retrieval of movie-related data.

The system will store information about:

* Movies (title, release year, rating, duration, description, etc.).
* Actors and their roles in movies.
* Directors and their contributions to movies.
* Genres associated with movies.
* Awards and their relation to movies.
* Users who can submit reviews and vote on movies.
* Reviews and ratings given by users.
* Votes for movies by users.
* Trailers linked to movies.

The database will help users find information about movies, explore details about cast and crew, see movie ratings, and read user reviews. It ensures a structured way to manage relationships between different entities.

**1.2 Constraints**

1. Each movie must have a unique **Movie\_ID** as the primary key.
2. Each actor and director must have a unique **Actor\_ID** and **Director\_ID**.
3. A movie can belong to multiple genres, but each genre must be unique.
4. Users must have a unique **User\_ID** and an email.
5. A user can only submit one review per movie.

**1.3 Potential Queries from Users**

1. Retrieve all movies released in a specific year.
2. Get the highest-rated movies.
3. Find movies by a specific director.
4. List all actors who played in a given movie.
5. Find movies of a specific genre.
6. Get all movies that won a specific award.
7. Retrieve the average rating of a particular movie.
8. Find movies with the most user reviews.
9. List top-rated movies based on user votes.
10. Retrieve all trailers for a specific movie.
11. Find the most reviewed movies by users.
12. Get all reviews for a particular movie.
13. Find movies with a duration longer than a given time.
14. Get movies sorted by rating in descending order.
15. List all awards won by a specific movie.

**2. ER-Diagram**

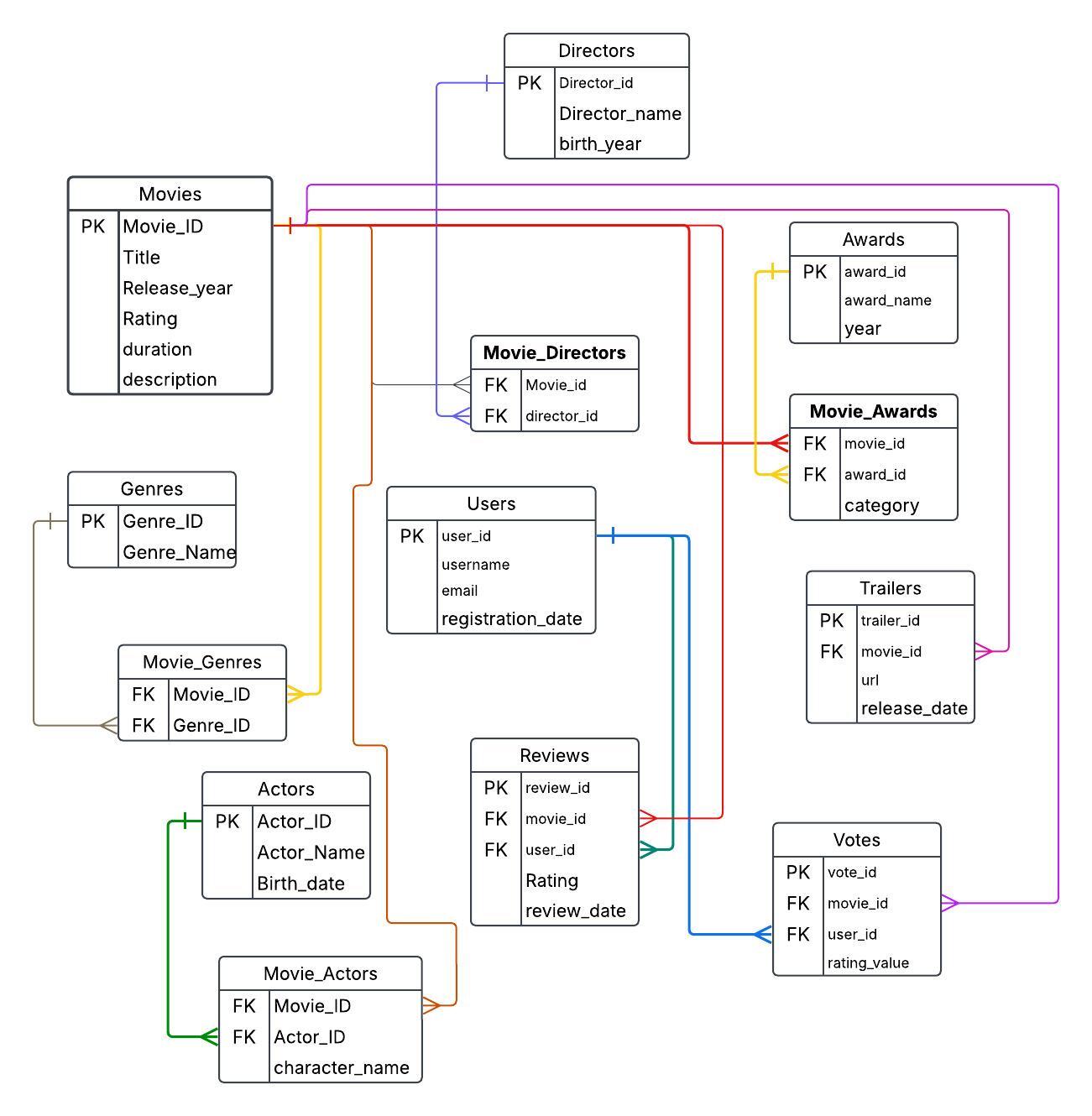
The ER-diagram includes at least 10 entities:

* **Movies** (PK: Movie\_ID)
* **Actors** (PK: Actor\_ID)
* **Movie\_Actors** (FK: Movie\_ID, Actor\_ID)
* **Directors** (PK: Director\_ID)
* **Movie\_Directors** (FK: Movie\_ID, Director\_ID)
* **Genres** (PK: Genre\_ID)
* **Movie\_Genres** (FK: Movie\_ID, Genre\_ID)
* **Users** (PK: User\_ID)
* **Reviews** (PK: Review\_ID, FK: Movie\_ID, User\_ID)
* **Votes** (PK: Vote\_ID, FK: Movie\_ID, User\_ID)
* **Awards** (PK: Award\_ID)
* **Movie\_Awards** (FK: Movie\_ID, Award\_ID)
* **Trailers** (PK: Trailer\_ID, FK: Movie\_ID)

The relationships in the ER-diagram follow these rules:

* **One-to-Many (1-to-M)**: A movie can have multiple reviews, but each review belongs to one movie.
* **Many-to-Many (M-to-M)**: Movies can have multiple actors, and actors can act in multiple movies.
* **Many-to-Many (M-to-M)**: Movies can have multiple genres, and a genre can belong to multiple movies.
* **One-to-Many (1-to-M)**: A user can write multiple reviews, but a review belongs to one user.

This conceptual design ensures efficient data retrieval and organization, supporting future expansions and advanced queries.



1. **Logical Design**

**Table: Movies**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field’s name** | **Field’s content** | **Type, length** | **Note (constraints)** |
| Movie\_ID | Movie ID | N(7) | PK |
| Title | Title | C(255) | NOT NULL |
| Release\_year | Release year | N(4) |  |
| Duration | Duration | N(5) |  |
| Description | Description | C(1000) |  |

**Table: Directors**

|  |  |  |  |
| --- | --- | --- | --- |
| Field’s name | Field’s content | Type, length | Note (constraints) |
| Director\_ID | Director ID | N(7) | PK |
| Director\_name | Director name | C(255) | NOT NULL |
| Birth\_year | Birth year | N(4) |  |

**Table: Movie\_Directors**

|  |  |  |  |
| --- | --- | --- | --- |
| Field’s name | Field’s content | Type, length | Note (constraints) |
| Movie\_ID | Movie ID | N(7) | FK to Movies |
| Director\_ID | Director ID | N(7) | FK to Directors |

**Table: Genres**

|  |  |  |  |
| --- | --- | --- | --- |
| Field’s name | Field’s content | Type, length | Note (constraints) |
| Genre\_ID | Genre ID | N(7) | PK |
| Genre\_Name | Genre Name | C(255) | NOT NULL |

**Table: Movie\_Genres**

|  |  |  |  |
| --- | --- | --- | --- |
| Field’s name | Field’s content | Type, length | Note (constraints) |
| Movie\_ID | Movie ID | N(7) | FK to Movies |
| Genre\_ID | Genre ID | N(7) | FK to Genres |

**Table: Actors**

|  |  |  |  |
| --- | --- | --- | --- |
| Field’s name | Field’s content | Type, length | Note (constraints) |
| Actor\_ID | Actor ID | N(7) | PK |
| Actor\_Name | Actor Name | C(255) | NOT NULL |
| Birth\_date | Birth date | D |  |

**Table: Movie\_Actors**

|  |  |  |  |
| --- | --- | --- | --- |
| Field’s name | Field’s content | Type, length | Note (constraints) |
| Movie\_ID | Movie ID | N(7) | FK to Movies |
| Actor\_ID | Actor ID | N(7) | FK to Actors |

**Table: Users**

|  |  |  |  |
| --- | --- | --- | --- |
| Field’s name | Field’s content | Type, length | Note (constraints) |
| User\_ID | User ID | N(7) | PK |
| Username | Username | C(255) | NOT NULL |
| Email | Email | C(255) | UNIQUE, NOT NULL |
| Registration\_date | Registration Date | D |  |

**Table: Reviews**

|  |  |  |  |
| --- | --- | --- | --- |
| Field’s name | Field’s content | Type, length | Note (constraints) |
| Review\_ID | Review ID | N(7) | PK |
| Movie\_ID | Movie ID | N(7) | FK to Movies |
| User\_ID | User ID | N(7) | FK to Users |
| Rating | Rating | N(2) |  |
| Review\_date | Review Date | D |  |

**Table: Votes**

|  |  |  |  |
| --- | --- | --- | --- |
| Field’s name | Field’s content | Type, length | Note (constraints) |
| Vote\_ID | Vote ID | N(7) | PK |
| Movie\_ID | Movie ID | N(7) | FK to Movies |
| User\_ID | User ID | N(7) | FK to Users |
| Rating\_Value | Rating Value | N(2) |  |

**Table: Trailers**

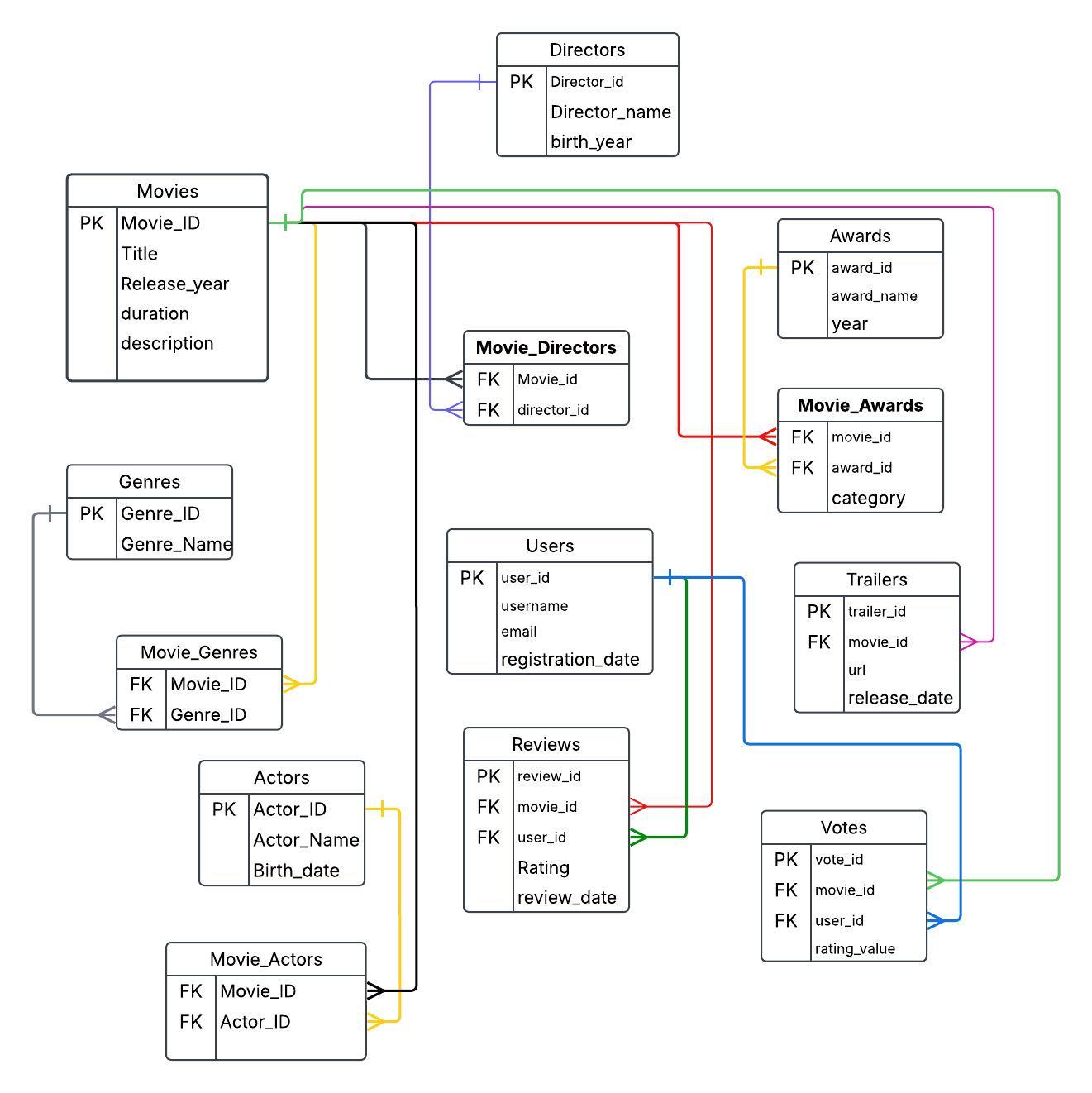
|  |  |  |  |
| --- | --- | --- | --- |
| Field’s name | Field’s content | Type, length | Note (constraints) |
| Trailer\_ID | Trailer ID | N(7) | PK |
| Movie\_ID | Movie ID | N(7) | FK to Movies |
| URL | Trailer URL | C(500) |  |
| Release\_Date | Release Date | D |  |

**Table: Awards**

|  |  |  |  |
| --- | --- | --- | --- |
| Field’s name | Field’s content | Type, length | Note (constraints) |
| Award\_ID | Award ID | N(7) | PK |
| Award\_Name | Award Name | C(255) | NOT NULL |
| Year | Year | N(4) |  |

**Table: Movie\_Awards**

|  |  |  |  |
| --- | --- | --- | --- |
| Field’s name | Field’s content | Type, length | Note (constraints) |
| Movie\_ID | Movie ID | N(7) | FK to Movies |
| Award\_ID | Award ID | N(7) | FK to Awards |
| Category | Award Category | C(255) |  |



**3.Physical Design and Content**

-- Task 2: Create Tables

CREATE TABLE Movies (

Movie\_ID SERIAL PRIMARY KEY,

Title VARCHAR(255) NOT NULL,

Release\_year INT,

Duration INT,

Description TEXT

);

CREATE TABLE Directors (

Director\_ID SERIAL PRIMARY KEY,

Director\_name VARCHAR(255) NOT NULL,

Birth\_year INT

);

CREATE TABLE Movie\_Directors (

Movie\_ID INT REFERENCES Movies(Movie\_ID),

Director\_ID INT REFERENCES Directors(Director\_ID),

PRIMARY KEY (Movie\_ID, Director\_ID)

);

CREATE TABLE Genres (

Genre\_ID SERIAL PRIMARY KEY,

Genre\_Name VARCHAR(255) NOT NULL

);

CREATE TABLE Movie\_Genres (

Movie\_ID INT REFERENCES Movies(Movie\_ID),

Genre\_ID INT REFERENCES Genres(Genre\_ID),

PRIMARY KEY (Movie\_ID, Genre\_ID)

);

CREATE TABLE Actors (

Actor\_ID SERIAL PRIMARY KEY,

Actor\_Name VARCHAR(255) NOT NULL,

Birth\_date DATE

);

CREATE TABLE Movie\_Actors (

Movie\_ID INT REFERENCES Movies(Movie\_ID),

Actor\_ID INT REFERENCES Actors(Actor\_ID),

PRIMARY KEY (Movie\_ID, Actor\_ID)

);

CREATE TABLE Users (

User\_ID SERIAL PRIMARY KEY,

Username VARCHAR(255) NOT NULL,

Email VARCHAR(255) UNIQUE NOT NULL,

Registration\_date DATE

);

CREATE TABLE Reviews (

Review\_ID SERIAL PRIMARY KEY,

Movie\_ID INT REFERENCES Movies(Movie\_ID),

User\_ID INT REFERENCES Users(User\_ID),

Rating INT CHECK (Rating BETWEEN 1 AND 10),

Review\_date DATE

);

CREATE TABLE Votes (

Vote\_ID SERIAL PRIMARY KEY,

Movie\_ID INT REFERENCES Movies(Movie\_ID),

User\_ID INT REFERENCES Users(User\_ID),

Rating\_Value INT CHECK (Rating\_Value BETWEEN 1 AND 10)

);

CREATE TABLE Trailers (

Trailer\_ID SERIAL PRIMARY KEY,

Movie\_ID INT REFERENCES Movies(Movie\_ID),

URL VARCHAR(500),

Release\_Date DATE

);

CREATE TABLE Awards (

Award\_ID SERIAL PRIMARY KEY,

Award\_Name VARCHAR(255) NOT NULL,

Year INT

);

CREATE TABLE Movie\_Awards (

Movie\_ID INT REFERENCES Movies(Movie\_ID),

Award\_ID INT REFERENCES Awards(Award\_ID),

Category VARCHAR(255),

PRIMARY KEY (Movie\_ID, Award\_ID)

);

-- Task 3: Insert Sample Data

INSERT INTO Movies (Title, Release\_year, Duration, Description) VALUES

('The Shawshank Redemption', 1994, 142, 'Two imprisoned men bond over a number of years...'),

('The Godfather', 1972, 175, 'The aging patriarch of an organized crime dynasty...'),

('The Dark Knight', 2008, 152, 'When the menace known as the Joker wreaks havoc...'),

('Pulp Fiction', 1994, 154, 'The lives of two mob hitmen, a boxer, and others...'),

('Fight Club', 1999, 139, 'An insomniac office worker and a soap maker...'),

('Forrest Gump', 1994, 142, 'The presidencies of Kennedy and Johnson...'),

('Inception', 2010, 148, 'A thief who steals corporate secrets through dream-sharing...'),

('The Matrix', 1999, 136, 'A hacker discovers the reality is a simulation...'),

('The Lord of the Rings: The Fellowship of the Ring', 2001, 178, 'A meek Hobbit sets out to destroy a powerful ring...'),

('Interstellar', 2014, 169, 'A team travels through a wormhole in space...'),

('Gladiator', 2000, 155, 'A former Roman general seeks revenge...'),

('Titanic', 1997, 195, 'A seventeen-year-old aristocrat falls in love...'),

('Avengers: Endgame', 2019, 181, 'After the devastating events of Infinity War...'),

('The Silence of the Lambs', 1991, 118, 'A young FBI cadet must confide in a manipulative killer...'),

('Schindler''s List', 1993, 195, 'In German-occupied Poland during WWII...');

INSERT INTO Directors (Director\_name, Birth\_year) VALUES

('Frank Darabont', 1959),

('Francis Ford Coppola', 1939),

('Christopher Nolan', 1970),

('Quentin Tarantino', 1963),

('David Fincher', 1962),

('Robert Zemeckis', 1952),

('Lana Wachowski', 1965),

('Peter Jackson', 1961),

('Ridley Scott', 1937),

('James Cameron', 1954),

('Anthony Russo', 1970),

('Joe Russo', 1971),

('Jonathan Demme', 1944),

('Steven Spielberg', 1946),

('Denis Villeneuve', 1967);

INSERT INTO Movie\_Directors (Movie\_ID, Director\_ID) VALUES

(1, 1), (2, 2), (3, 3), (4, 4), (5, 5),

(6, 6), (7, 3), (8, 7), (9, 8), (10, 3),

(11, 9), (12, 10), (13, 11), (13, 12),

(14, 13);

INSERT INTO Genres (Genre\_Name) VALUES

('Drama'), ('Crime'), ('Action'), ('Thriller'),

('Adventure'), ('Romance'), ('Sci-Fi'), ('Fantasy'),

('Biography'), ('History'), ('Mystery'), ('War'),

('Comedy'), ('Horror'), ('Animation');

INSERT INTO Movie\_Genres (Movie\_ID, Genre\_ID) VALUES

(1, 1), (2, 1), (2, 2), (3, 3), (3, 4),

(4, 2), (4, 1), (5, 1), (5, 4), (6, 1),

(7, 3), (7, 7), (8, 7), (9, 5), (9, 8);

INSERT INTO Actors (Actor\_Name, Birth\_date) VALUES

('Tim Robbins', '1958-10-16'),

('Morgan Freeman', '1937-06-01'),

('Marlon Brando', '1924-04-03'),

('Al Pacino', '1940-04-25'),

('Christian Bale', '1974-01-30'),

('Heath Ledger', '1979-04-04'),

('John Travolta', '1954-02-18'),

('Samuel L. Jackson', '1948-12-21'),

('Brad Pitt', '1963-12-18'),

('Edward Norton', '1969-08-18'),

('Tom Hanks', '1956-07-09'),

('Keanu Reeves', '1964-09-02'),

('Elijah Wood', '1981-01-28'),

('Leonardo DiCaprio', '1974-11-11'),

('Russell Crowe', '1964-04-07');

INSERT INTO Movie\_Actors (Movie\_ID, Actor\_ID) VALUES

(1, 1), (1, 2), (2, 3), (2, 4), (3, 5),

(3, 6), (4, 7), (4, 8), (5, 9), (5, 10),

(6, 11), (8, 12), (9, 13), (10, 14), (11, 15);

INSERT INTO Users (Username, Email, Registration\_date) VALUES

('user1', 'akmal@example.com', '2023-01-01'),

('user2', 'nurbek@example.com', '2023-01-05'),

('user3', 'alkar@example.com', '2023-02-15'),

('user4', 'arman@example.com', '2023-03-10'),

('user5', 'almat@example.com', '2023-03-22'),

('user6', 'user6@example.com', '2023-04-01'),

('user7', 'user7@example.com', '2023-04-15'),

('user8', 'user8@example.com', '2023-04-30'),

('user9', 'user9@example.com', '2023-05-12'),

('user10', 'user10@example.com', '2023-05-25'),

('user11', 'user11@example.com', '2023-06-01'),

('user12', 'user12@example.com', '2023-06-10'),

('user13', 'user13@example.com', '2023-06-18'),

('user14', 'user14@example.com', '2023-07-01'),

('user15', 'user15@example.com', '2023-07-10');

INSERT INTO Reviews (Movie\_ID, User\_ID, Rating, Review\_date) VALUES

(1,1,10,'2023-06-01'), (2,2,9,'2023-06-02'), (3,3,8,'2023-06-03'),

(4,4,9,'2023-06-04'), (5,5,10,'2023-06-05'), (6,6,8,'2023-06-06'),

(7,7,9,'2023-06-07'), (8,8,8,'2023-06-08'), (9,9,10,'2023-06-09'),

(10,10,9,'2023-06-10'), (11,11,7,'2023-06-11'), (12,12,8,'2023-06-12'),

(13,13,9,'2023-06-13'), (14,14,8,'2023-06-14'), (15,15,10,'2023-06-15');

INSERT INTO Votes (Movie\_ID, User\_ID, Rating\_Value) VALUES

(1,1,10), (2,2,9), (3,3,8), (4,4,9), (5,5,10),

(6,6,8), (7,7,9), (8,8,8), (9,9,10), (10,10,9),

(11,11,7), (12,12,8), (13,13,9), (14,14,8), (15,15,10);

INSERT INTO Trailers (Movie\_ID, URL, Release\_Date) VALUES

(1, 'https://youtu.be/NmzuHjWmXOc', '1994-09-10'),

(2, 'https://youtu.be/sY1S34973zA', '1972-03-15'),

(3, 'https://youtu.be/EXeTwQWrcwY', '2008-07-14'),

(4, 'https://youtu.be/s7EdQ4FqbhY', '1994-05-21'),

(5, 'https://youtu.be/qtRKdVHc-cE', '1999-10-15'),

(6, 'https://youtu.be/bLvqoHBptjg', '1994-07-06'),

(7, 'https://youtu.be/YoHD9XEInc0', '2010-07-08'),

(8, 'https://youtu.be/vKQi3bBA1y8', '1999-03-24'),

(9, 'https://youtu.be/V75dMMIW2B4', '2001-12-19'),

(10, 'https://youtu.be/zSWdZVtXT7E', '2014-11-05'),

(11, 'https://youtu.be/owK1qxDselE', '2000-05-01'),

(12, 'https://youtu.be/kVrqfYjkTdQ', '1997-12-19'),

(13, 'https://youtu.be/TcMBFSGVi1c', '2019-04-26'),

(14, 'https://youtu.be/W6Mm8Sbe\_\_o', '1991-02-14'),

(15, 'https://youtu.be/gG22XNhtnoY', '1993-12-15');

INSERT INTO Awards (Award\_Name, Year) VALUES

('Best Picture', 1994),

('Best Director', 1994),

('Best Actor', 1994),

('Best Adapted Screenplay', 1994),

('Best Cinematography', 1999),

('Best Visual Effects', 2010),

('Best Editing', 2008),

('Best Supporting Actor', 2008),

('Best Actress', 1991),

('Best Original Score', 1994),

('Best Makeup', 2000),

('Best Costume Design', 1997),

('Best Sound Mixing', 2014),

('Best Production Design', 2019),

('Best Animated Feature', 2010);

INSERT INTO Movie\_Awards (Movie\_ID, Award\_ID, Category) VALUES

(1, 1, 'Best Picture'),

(2, 2, 'Best Director'),

(3, 7, 'Best Editing'),

(3, 8, 'Best Supporting Actor'),

(4, 4, 'Best Adapted Screenplay'),

(5, 5, 'Best Cinematography'),

(6, 10, 'Best Original Score'),

(7, 6, 'Best Visual Effects'),

(8, 6, 'Best Visual Effects'),

(9, 1, 'Best Picture'),

(10, 13, 'Best Sound Mixing'),

(11, 11, 'Best Makeup'),

(12, 12, 'Best Costume Design'),

(13, 14, 'Best Production Design'),

(14, 9, 'Best Actress');

SELECT \* FROM Movies;

SELECT \* FROM Directors;

SELECT \* FROM Movie\_Directors;

SELECT \* FROM Genres;

SELECT \* FROM Movie\_Genres;

SELECT \* FROM Actors;

SELECT \* FROM Movie\_Actors;

SELECT \* FROM Users;

SELECT \* FROM Reviews;

SELECT \* FROM Votes;

SELECT \* FROM Trailers;

SELECT \* FROM Awards;

SELECT \* FROM Movie\_Awards;

4.Queries

-- 1. Select all movies released after 2000

SELECT Title, Release\_year FROM Movies WHERE Release\_year > 2000;

-- 2. Select all directors born before 1950

SELECT Director\_name, Birth\_year FROM Directors WHERE Birth\_year < 1950;

-- 3. Select all actors who starred in 'The Shawshank Redemption'

SELECT Actor\_Name FROM Actors

JOIN Movie\_Actors ON Actors.Actor\_ID = Movie\_Actors.Actor\_ID

JOIN Movies ON Movie\_Actors.Movie\_ID = Movies.Movie\_ID

WHERE Movies.Title = 'The Shawshank Redemption';

-- 4. Select the total number of movies per genre

SELECT Genre\_Name, COUNT(Movie\_Genres.Movie\_ID) AS Total\_Movies

FROM Genres

JOIN Movie\_Genres ON Genres.Genre\_ID = Movie\_Genres.Genre\_ID

GROUP BY Genre\_Name;

-- 5. Select all movies with a rating higher than 8, ordered by release year

SELECT Title, Release\_year FROM Movies

JOIN Reviews ON Movies.Movie\_ID = Reviews.Movie\_ID

WHERE Rating > 8

ORDER BY Release\_year;

-- 6. Select all directors and their corresponding movies

SELECT Director\_name, Title FROM Directors

JOIN Movie\_Directors ON Directors.Director\_ID = Movie\_Directors.Director\_ID

JOIN Movies ON Movie\_Directors.Movie\_ID = Movies.Movie\_ID;

-- 7. Find the average rating of each movie

SELECT Title, Round(AVG(Rating),2) AS Average\_Rating FROM Movies

JOIN Reviews ON Movies.Movie\_ID = Reviews.Movie\_ID

GROUP BY Title;

-- 8. Select all movies from 1994 with a genre 'Drama'

SELECT Title FROM Movies

JOIN Movie\_Genres ON Movies.Movie\_ID = Movie\_Genres.Movie\_ID

JOIN Genres ON Movie\_Genres.Genre\_ID = Genres.Genre\_ID

WHERE Release\_year = 1994 AND Genre\_Name = 'Drama';

-- 9. Select movies and their directors where the director's birth year is between 1950 and 1970

SELECT Title, Director\_name FROM Movies

JOIN Movie\_Directors ON Movies.Movie\_ID = Movie\_Directors.Movie\_ID

JOIN Directors ON Movie\_Directors.Director\_ID = Directors.Director\_ID

WHERE Birth\_year BETWEEN 1950 AND 1970;

-- 10. Find the total number of votes for each movie

SELECT Title, COUNT(Votes.Vote\_ID) AS Total\_Votes FROM Movies

JOIN Votes ON Movies.Movie\_ID = Votes.Movie\_ID

GROUP BY Title;

-- 11. Find all movies that have no associated trailer

SELECT Title FROM Movies

LEFT JOIN Trailers ON Movies.Movie\_ID = Trailers.Movie\_ID

WHERE Trailers.Movie\_ID IS NULL;

-- 12. Select all actors who played in 'The Godfather' and 'Pulp Fiction'

SELECT Actor\_Name FROM Actors

JOIN Movie\_Actors ON Actors.Actor\_ID = Movie\_Actors.Actor\_ID

JOIN Movies ON Movie\_Actors.Movie\_ID = Movies.Movie\_ID

WHERE Movies.Title IN ('The Godfather', 'Pulp Fiction');

-- 13. Get the movies that have won an award in 1994

SELECT Title FROM Movies

JOIN Movie\_Awards ON Movies.Movie\_ID = Movie\_Awards.Movie\_ID

JOIN Awards ON Movie\_Awards.Award\_ID = Awards.Award\_ID

WHERE Awards.Year = 1994;

-- 14. Get the actors who starred in movies with the genre 'Sci-Fi'

SELECT Actor\_Name FROM Actors

JOIN Movie\_Actors ON Actors.Actor\_ID = Movie\_Actors.Actor\_ID

JOIN Movies ON Movie\_Actors.Movie\_ID = Movies.Movie\_ID

JOIN Movie\_Genres ON Movies.Movie\_ID = Movie\_Genres.Movie\_ID

JOIN Genres ON Movie\_Genres.Genre\_ID = Genres.Genre\_ID

WHERE Genre\_Name = 'Sci-Fi';

-- 15. Get the highest-rated movie for each user

SELECT u.Username, m.Title, r.Rating AS Highest\_Rating

FROM Users u

JOIN Reviews r ON u.User\_ID = r.User\_ID

JOIN Movies m ON r.Movie\_ID = m.Movie\_ID

WHERE (u.User\_ID, r.Rating) IN (

SELECT r2.User\_ID, MAX(r2.Rating)

FROM Reviews r2

GROUP BY r2.User\_ID

);

-- 16. Find the movies with a duration between 140 and 160 minutes

SELECT Title, Duration FROM Movies

WHERE Duration BETWEEN 140 AND 160;

-- 17. Select movies and the number of votes they have received, ordered by number of votes

SELECT Title, COUNT(Votes.Vote\_ID) AS Total\_Votes FROM Movies

JOIN Votes ON Movies.Movie\_ID = Votes.Movie\_ID

GROUP BY Title

ORDER BY Total\_Votes DESC;

-- 18. Find all movies where the rating is not null

SELECT Title FROM Movies

JOIN Reviews ON Movies.Movie\_ID = Reviews.Movie\_ID

WHERE Rating IS NOT NULL;

-- 19. Select movies released after 2000 that have received a 'Best Picture' award

SELECT Title FROM Movies

JOIN Movie\_Awards ON Movies.Movie\_ID = Movie\_Awards.Movie\_ID

JOIN Awards ON Movie\_Awards.Award\_ID = Awards.Award\_ID

WHERE Awards.Award\_Name = 'Best Picture' AND Movies.Release\_year > 2000;

-- 20. Get the movies that have no reviews yet

SELECT Title FROM Movies

LEFT JOIN Reviews ON Movies.Movie\_ID = Reviews.Movie\_ID

WHERE Reviews.Review\_ID IS NULL;

CONCLUSION

As a result of this project, I successfully designed and populated a relational database based on the IMDb concept and performed over 20 meaningful SQL queries to extract insights from the data. These queries demonstrated the ability to join multiple tables, filter data, calculate aggregates, and handle missing values.

This kind of application can help users, developers, or companies to manage large volumes of movie-related information efficiently. It supports tasks such as finding the most popular movies, tracking awards, analyzing trends over time, and understanding the relationship between movies, directors, genres, and user reviews.

In the future, this project can be expanded to include real-time updates, advanced analytics, and user interaction features. It could serve as a foundation for a recommendation system or a web-based movie analytics platform. Overall, it improves data accessibility and enables smarter decision-making in entertainment and media industries.

REFERENCES

lecture from 1 to 12,  
IMDb site  
<https://www.w3schools.com/sql/>  
<https://sql-academy.org/>  
<https://www.geeksforgeeks.org/sql-tutorial/>