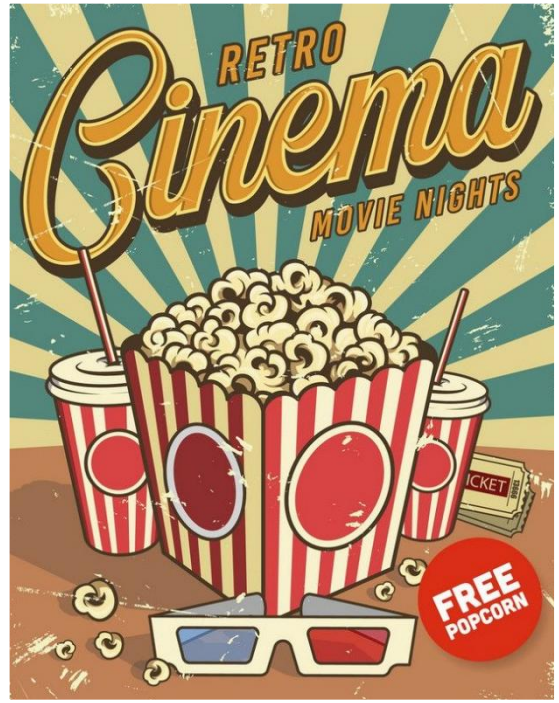
MOVIE STREAMING PLATFORM

CINEMA THEME

PRESENTED BY SANYA SINGH

SQL PROJECT

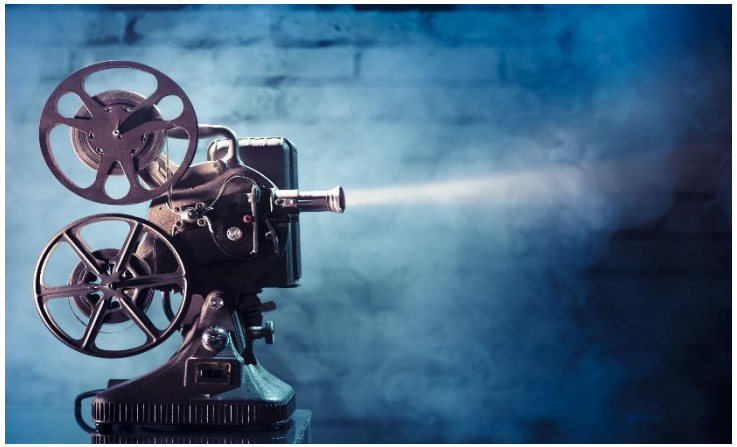


Movie Streaming Platform

Behind the Screens: A Movie Streaming Platform in SQL

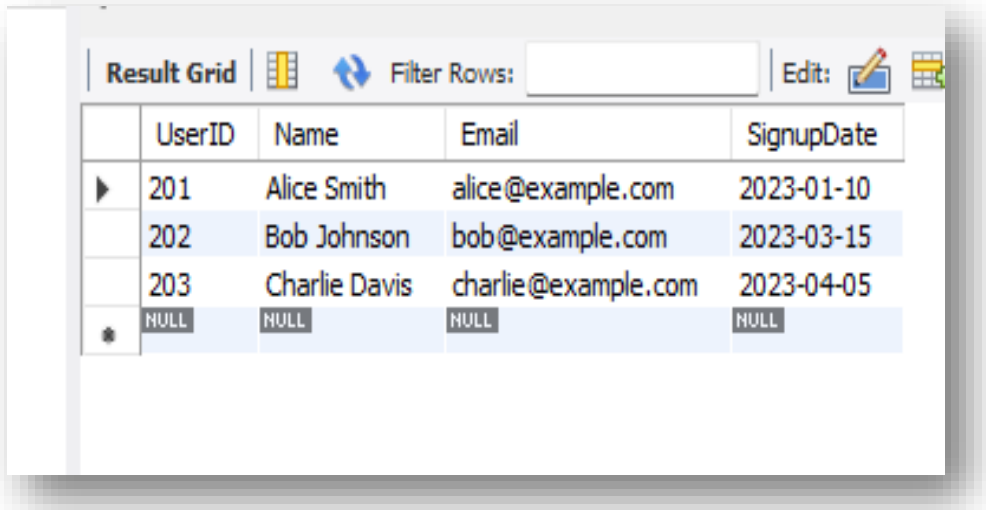
ABSTRACT

- This project presents the design and implementation of a relational database system for a movie streaming platform. The objective is to model key components such as users, movies, genres, watch history, ratings, and subscription plans using structured SQL tables and relationships. The database is designed to efficiently store and manage data related to streaming activity, user preferences, and content metadata. An Entity-Relationship (ER) diagram was created to visualize the relationships between entities, ensuring data integrity and normalization. The project demonstrates how SQL can be used to support common functionalities of a streaming platform, such as tracking user watch history, managing subscriptions, and analyzing viewer engagement through ratings. This system provides a foundation that could be extended into a fully functional backend for an entertainment service, offering insights into real-world database modeling and management.



ER DIAGRAM

Structure of Table

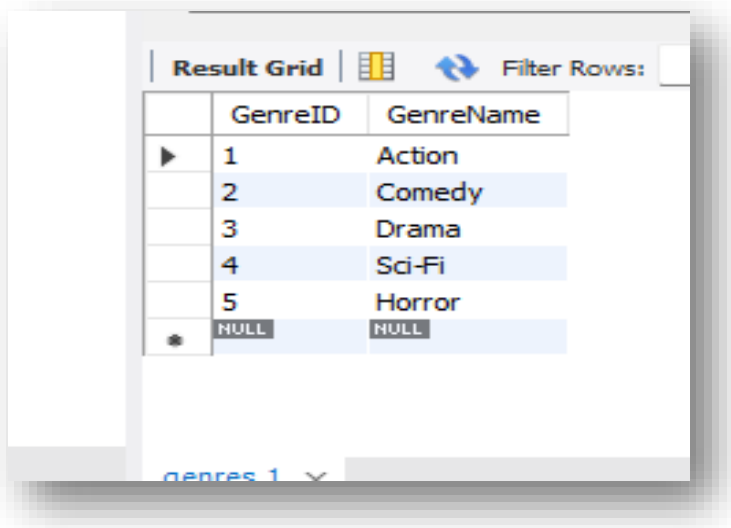


- The Users table holds data about the platform's users, including a unique UserID, their name, email address, and the date they signed up. With just a few users shown in the sample data, it illustrates how the platform tracks individual users and their activity history. This is essential for user-based analytics, targeted recommendations, and engagement tracking.
- **SYNTAX:-**

```
4  CREATE TABLE Users (
5      UserID INT PRIMARY KEY,
6      Name VARCHAR(100),
7      Email VARCHAR(100),
8      SignupDate DATE
9  );
```

```
57  INSERT INTO Users (UserID, Name, Email, SignupDate) VALUES
58  (201, 'Alice Smith', 'alice@example.com', '2023-01-10'),
59  (202, 'Bob Johnson', 'bob@example.com', '2023-03-15'),
60  (203, 'Charlie Davis', 'charlie@example.com', '2023-04-05');
```

Structure of Table

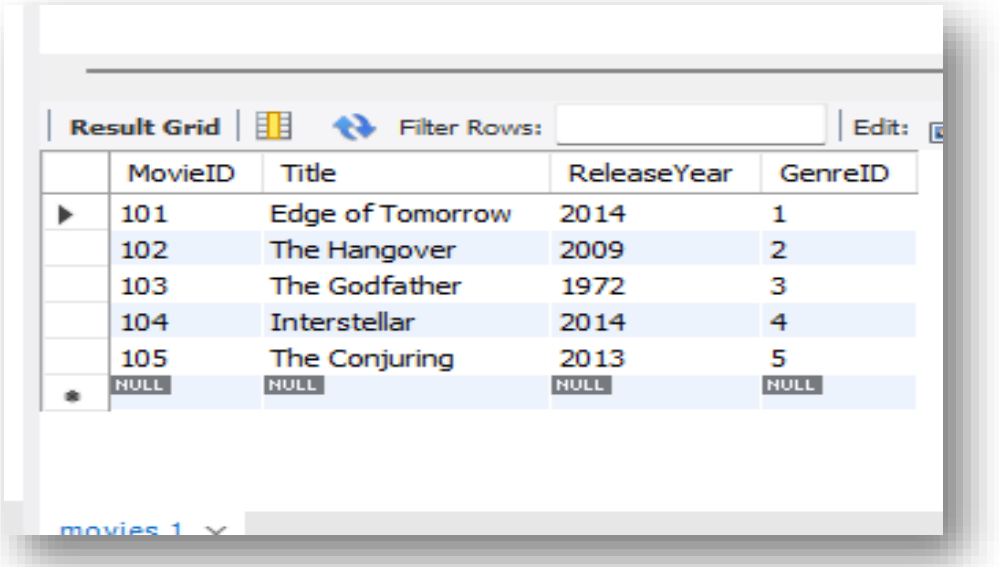


- The Genres table defines the different categories of movies available on the platform. Each genre is identified by a unique GenreID and a descriptive GenreName, such as Action, Comedy, Drama, Sci-Fi, and Horror. This table plays a crucial role in organizing movies for easier classification and genre-based filtering.
- **SYNTAX:-**

```
CREATE TABLE Genres (
    GenreID INT PRIMARY KEY,
    GenreName VARCHAR(50)
);
```

```
45 •  INSERT INTO Genres (GenreID, GenreName) VALUES
46    (1, 'Action'),
47    (2, 'Comedy'),
48    (3, 'Drama'),
49    (4, 'Sci-Fi'),
50    (5, 'Horror');
```

Structure of table

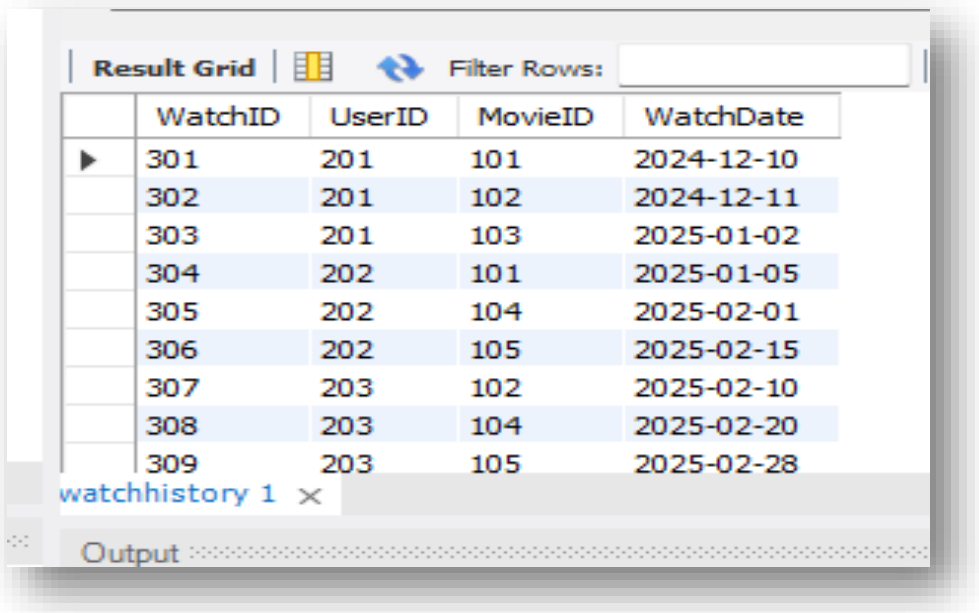


- The Movies table stores information about the films offered on the platform. It includes each movie's unique MovieID, title, year of release, and an associated GenreID linking it to the appropriate genre. This table acts as a central node in the database, connecting to multiple other tables including WatchHistory and Ratings. Sample data includes well-known titles like Edge of Tomorrow and The Godfather, spanning a range of genres and release years.
- **SYNTAX:-**

```
10  CREATE TABLE Movies (
11      MovieID INT PRIMARY KEY,
12      Title VARCHAR(150),
13      ReleaseYear INT,
14      GenreID INT
15      );
```

```
51  INSERT INTO Movies (MovieID, Title, ReleaseYear, GenreID) VALUES
52  (101, 'Edge of Tomorrow', 2014, 1),
53  (102, 'The Hangover', 2009, 2),
54  (103, 'The Godfather', 1972, 3),
55  (104, 'Interstellar', 2014, 4),
56  (105, 'The Conjuring', 2013, 5);
```

Structure Of Table



Result Grid | Filter Rows:

WatchID	UserID	MovieID	WatchDate
301	201	101	2024-12-10
302	201	102	2024-12-11
303	201	103	2025-01-02
304	202	101	2025-01-05
305	202	104	2025-02-01
306	202	105	2025-02-15
307	203	102	2025-02-10
308	203	104	2025-02-20
309	203	105	2025-02-28

watchhistory 1

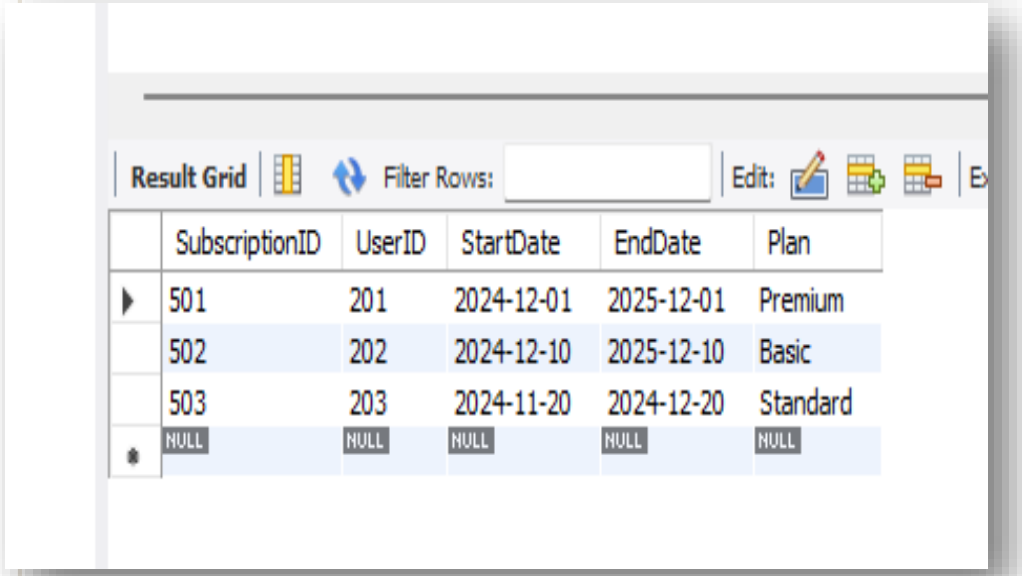
Output

```sql
CREATE TABLE WatchHistory (
    WatchID INT PRIMARY KEY,
    UserID INT,
    MovieID INT,
    WatchDate DATE,
    FOREIGN KEY (UserID) REFERENCES Users(UserID),
    FOREIGN KEY (MovieID) REFERENCES Movies(MovieID)
);
```

- The WatchHistory table logs each instance of a user watching a movie. It includes a unique WatchID, references to UserID and MovieID, and the WatchDate. This table reveals patterns in user behavior, such as which movies are most viewed and when. For example, user Alice watched three movies in December 2024 and January 2025, showing consistent engagement.
- **SYNTAX:-**

```sql
INSERT INTO WatchHistory (WatchID, UserID, MovieID, WatchDate) VALU
    (301, 201, 101, '2024-12-10'),
    (302, 201, 102, '2024-12-11'),
    (303, 201, 103, '2025-01-02'),
    (304, 202, 101, '2025-01-05'),
    (305, 202, 104, '2025-02-01'),
    (306, 202, 105, '2025-02-15'),
    (307, 203, 102, '2025-02-10'),
    (308, 203, 104, '2025-02-20'),
    (309, 203, 105, '2025-02-28');
```

Structure Of table



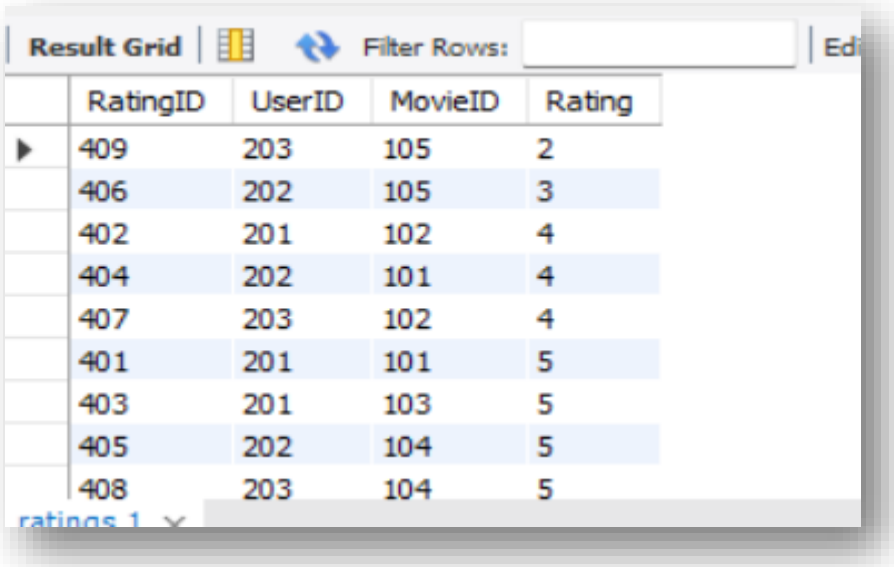
SubscriptionID	UserID	StartDate	EndDate	Plan
501	201	2024-12-01	2025-12-01	Premium
502	202	2024-12-10	2025-12-10	Basic
503	203	2024-11-20	2024-12-20	Standard
NULL	NULL	NULL	NULL	NULL

- the Subscription table tracks the financial and access status of users. It includes the user ID, subscription type (e.g., Basic, Premium), start and end dates, and payment status. This table is critical for managing platform revenue, identifying active subscribers, and ensuring content access is aligned with each user's subscription plan.
- **SYNTAX:-**

```
37  CREATE TABLE Subscriptions (
38      SubscriptionID INT PRIMARY KEY,
39      UserID INT,
40      StartDate DATE,
41      EndDate DATE,
42      Plan VARCHAR(50),
43      FOREIGN KEY (UserID) REFERENCES Users(UserID)
44  );
```

```
1  INSERT INTO Subscriptions (SubscriptionID, UserID, StartDate, EndDate, Plan) VALUES
2      (501, 201, '2024-12-01', '2025-12-01', 'Premium'),
3      (502, 202, '2024-12-10', '2025-12-10', 'Basic'),
4      (503, 203, '2024-11-20', '2024-12-20', 'Standard');
```

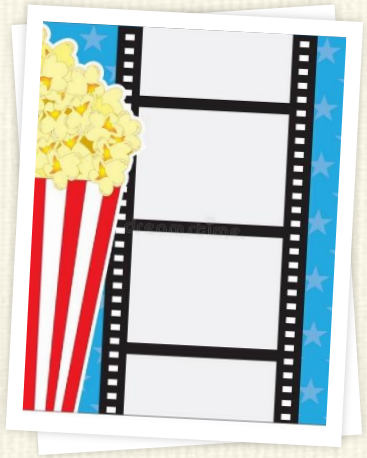
Structure of Table




```sql
CREATE TABLE Ratings (
    RatingID INT PRIMARY KEY,
    UserID INT,
    MovieID INT,
    Rating INT CHECK (Rating BETWEEN 1 AND 5),
    FOREIGN KEY (UserID) REFERENCES Users(UserID),
    FOREIGN KEY (MovieID) REFERENCES Movies(MovieID)
);
```

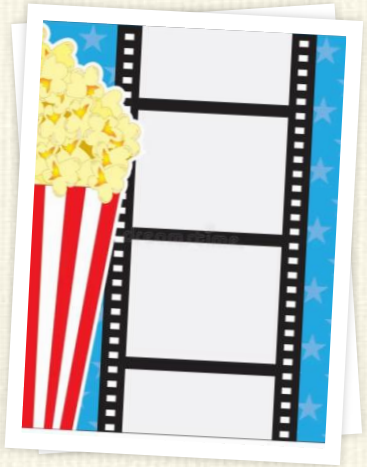
- The Ratings table collects user feedback in the form of numerical ratings for movies, on a scale from 1 to 5. This supports both quality assessment and recommendation algorithms by showing which movies resonate best with the audience.
- **SYNTAX:-**

```sql
INSERT INTO Ratings (RatingID, UserID, MovieID, Rating) VALUES
(401, 201, 101, 5),
(402, 201, 102, 4),
(403, 201, 103, 5),
(404, 202, 101, 4),
(405, 202, 104, 5),
(406, 202, 105, 3),
(407, 203, 102, 4),
(408, 203, 104, 5),
(409, 203, 105, 2);
```



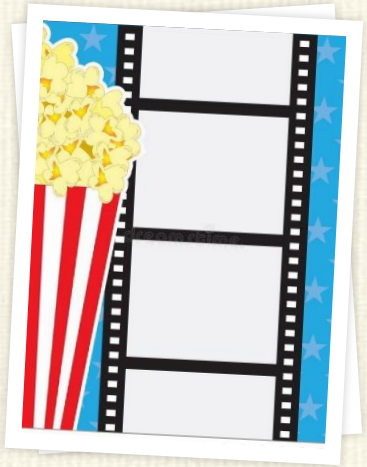
Contents Of the table (Genres Table)

GenreID	GenreName
1	Action
2	Comedy
3	Drama
4	Sci-Fi
5	Horror



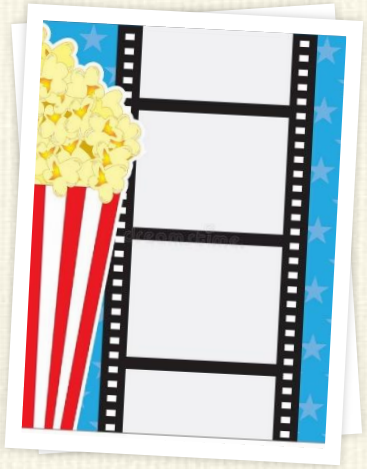
Contents Of the table (Movies Table)

MovieID	Title	ReleaseYear	GenreID
101	Edge of Tomorrow	2014	1
102	The Hangover	2009	2
103	The Godfather	1972	3
104	Interstellar	2014	4
105	The Conjuring	2013	5



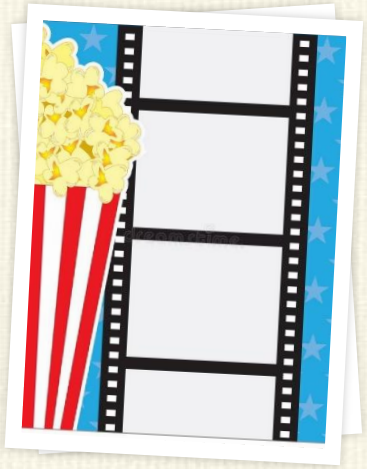
Contents Of the table (Users Table)

UserID	Name	Email	SignupDate
201	Alice Smith	alice@example.com	2023-01-10
202	Bob Johnson	bob@example.com	2023-03-15
203	Charlie Davis	charlie@example.com	2023-04-05



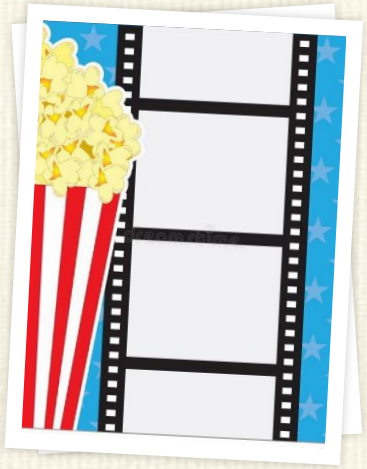
Contents Of the table (WatchHistory Table)

WatchID	UserID	MovieID	WatchDate
301	201	101	2024-12-10
302	201	102	2024-12-11
303	202	104	2025-02-01



Contents Of the table (Ratings Table)

RatingID	UserID	MovieID	Rating
401	201	101	5
402	201	102	4
403	202	104	5



Contents Of the table (Subscription Table)

SubscriptionID	UserID	PlanType	StartDate	EndDate	PaymentStatus
501	201	Premium	2023-01-10	2024-01-09	Paid
502	202	Basic	2023-03-15	2024-03-14	Paid
503	203	Premium	2023-04-05	2024-04-04	Pending

SUBQUERY

1.Users who watched more than the average number of movies:

```
35 •  SELECT Name
36    FROM Users
37    WHERE UserID IN (
38        SELECT UserID
39        FROM WatchHistory
40        GROUP BY UserID
41        HAVING COUNT(*) > (
42            SELECT AVG(watch_count)
43            FROM (
44                SELECT COUNT(*) AS watch_count
45                FROM WatchHistory
46                GROUP BY UserID
47            ) AS sub
48        )
49    );
```

Name
Alice Smith

2.Movies with average rating above 4.5:

```
100 •   SELECT Title
101     FROM Movies
102   WHERE MovieID IN (
103         SELECT MovieID
104         FROM Ratings
105         GROUP BY MovieID
106         HAVING AVG(Rating) > 4.5
107     );
```

Result Grid | Filter Rows:

Title
The Godfather
Interstellar

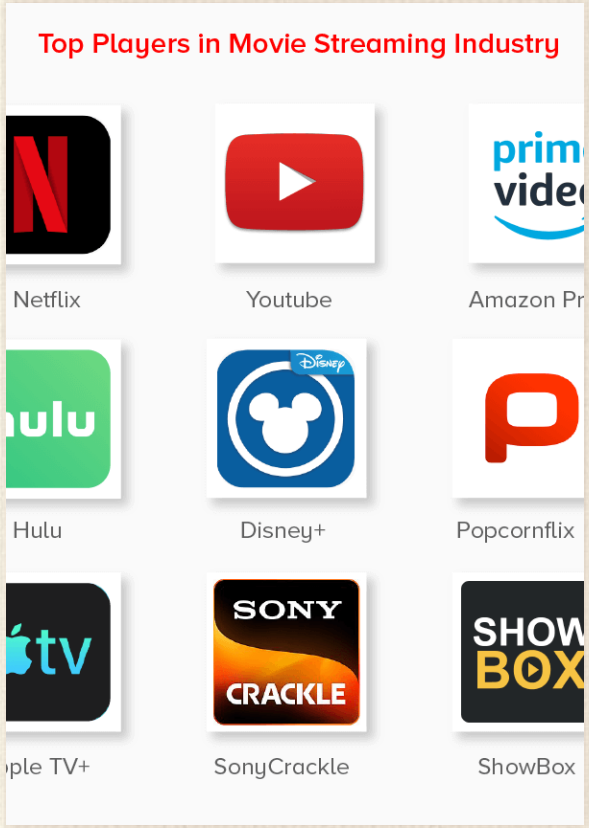
3.Genres with the highest number of movies

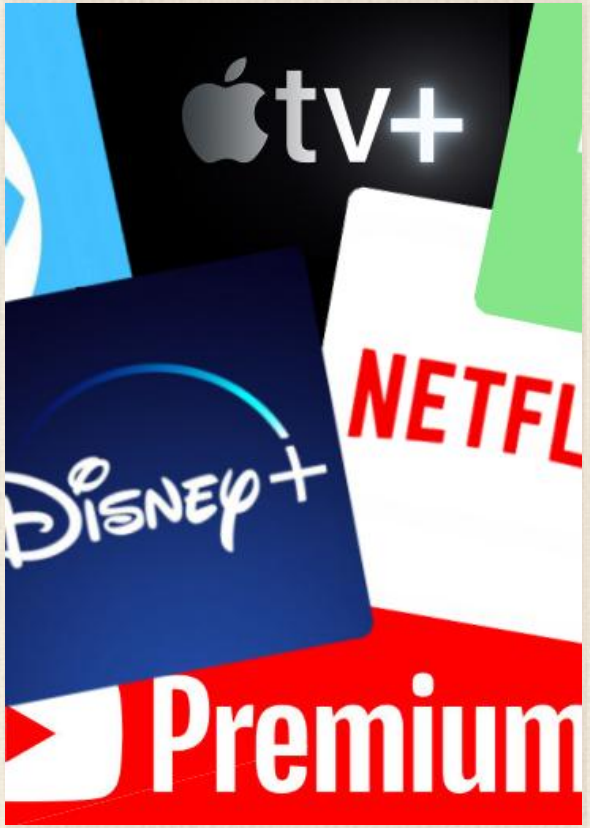
```
108 •    SELECT GenreName
109      FROM Genres
110      WHERE GenreID = (
111          SELECT GenreID
112          FROM Movies
113          GROUP BY GenreID
114          ORDER BY COUNT(*) DESC
115          LIMIT 1
116      );
```

Result Grid | Filter

	GenreName
▶	Action


tv+
me video


Top Players in Movie Streaming Industry
Netflix
Youtube
Amazon Pr
Hulu
Disney+
Popcornflix
SONY
CRACKLE
SonyCrackle
SHOW
BOX
ShowBox


tv+
Premium

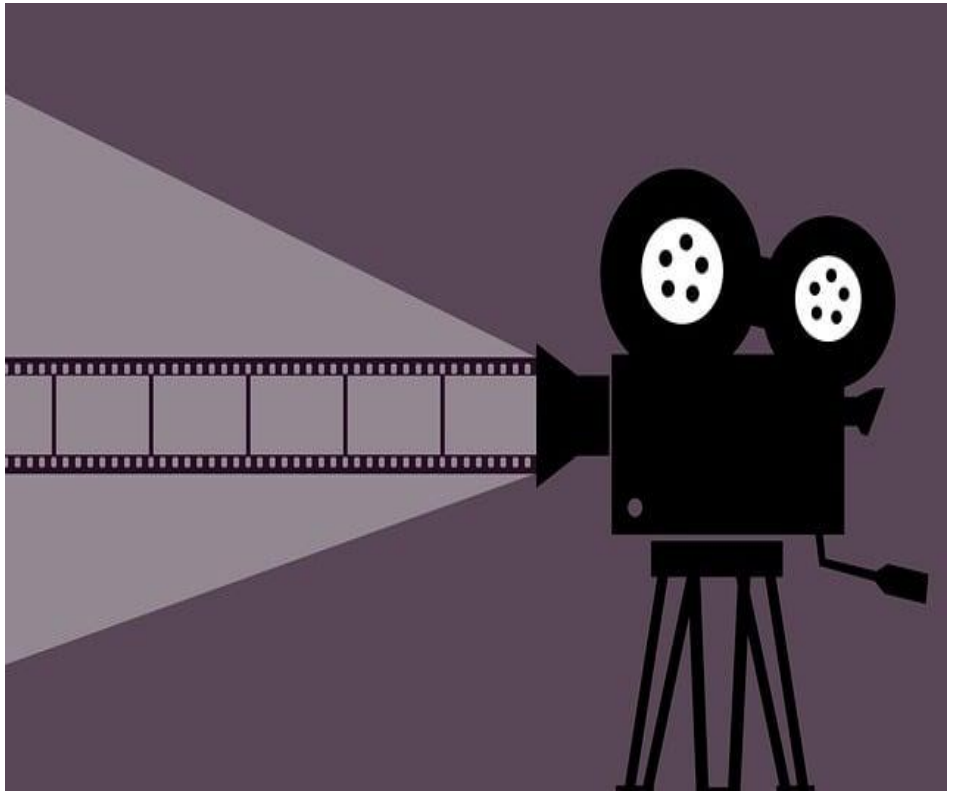
JOINS

List each user's watch history with movie titles and watch dates:

```
117 •   SELECT u.Name, m.Title, w.WatchDate
118     FROM WatchHistory w
119     JOIN Users u ON w.UserID = u.UserID
120     JOIN Movies m ON w.MovieID = m.MovieID;
```

Result Grid | Filter Rows: | Expo

Name	Title	WatchDate
Alice Smith	Edge of Tomorrow	2024-12-10
Alice Smith	The Hangover	2024-12-11
Alice Smith	The Godfather	2025-01-02
Bob Johnson	Edge of Tomorrow	2025-01-05
Bob Johnson	Interstellar	2025-02-01
Bob Johnson	The Conjuring	2025-02-15
Charlie Davis	The Hangover	2025-02-10
Charlie Davis	Interstellar	2025-02-20
Charlie Davis	The Conjuring	2025-02-28

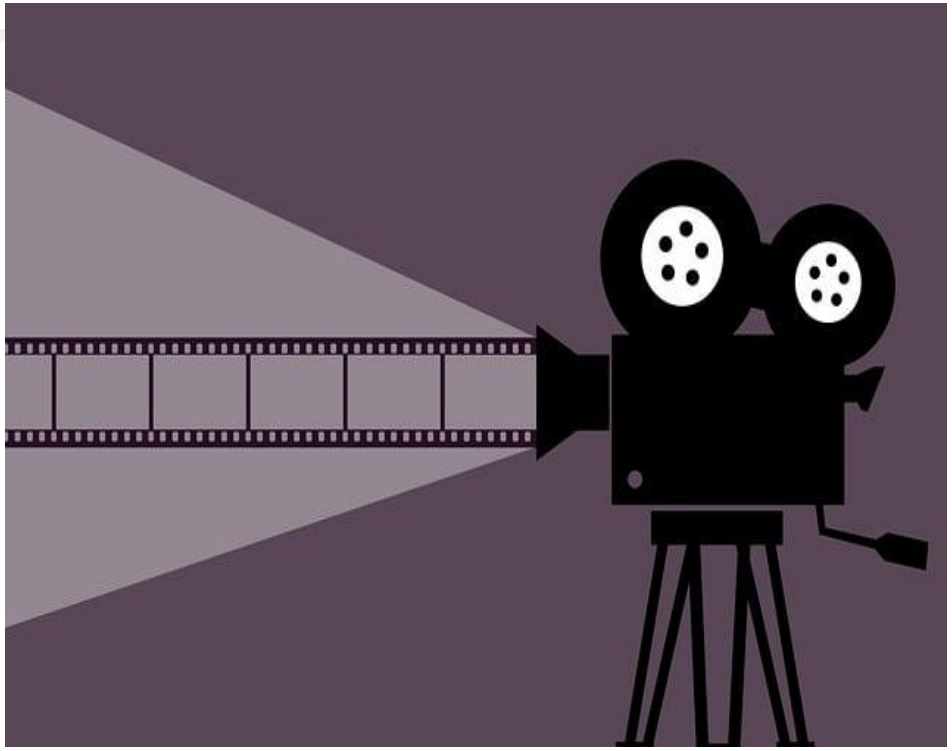


Movies with their average rating and genre:

```
121 •   SELECT m.Title, g.GenreName, AVG(r.Rating) AS AvgRating
122     FROM Movies m
123     JOIN Genres g ON m.GenreID = g.GenreID
124     JOIN Ratings r ON m.MovieID = r.MovieID
125     GROUP BY m.MovieID, m.Title, g.GenreName;
```

Result Grid | Filter Rows: | Ex

Title	GenreName	AvgRating
Edge of Tomorrow	Action	4.5000
The Hangover	Comedy	4.0000
The Godfather	Drama	5.0000
Interstellar	Sci-Fi	5.0000
The Conjuring	Horror	2.5000

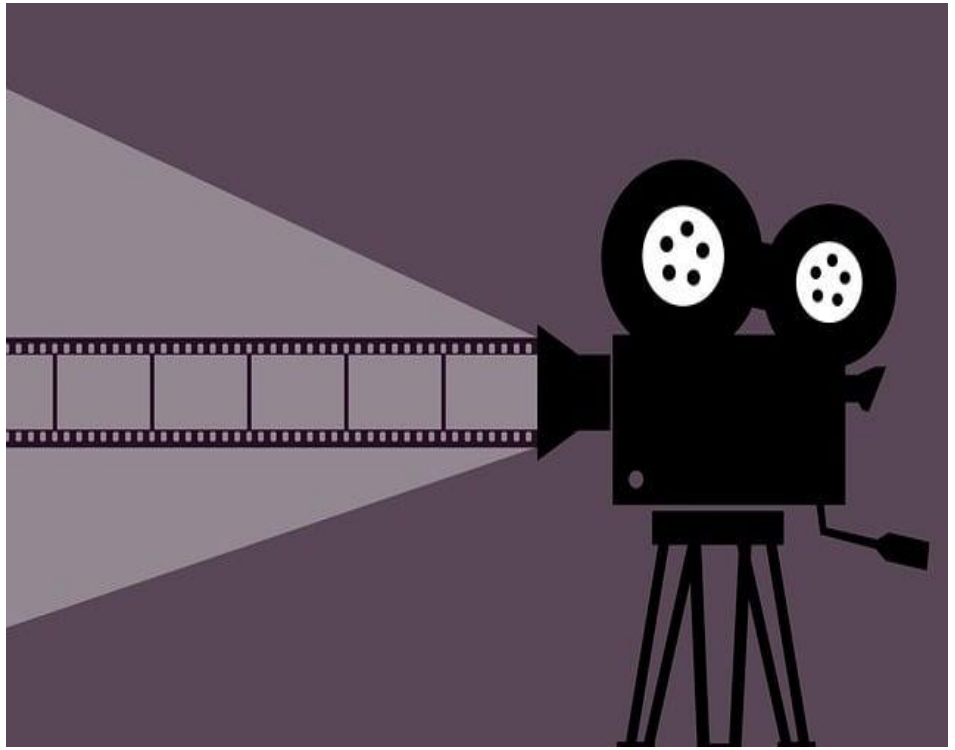


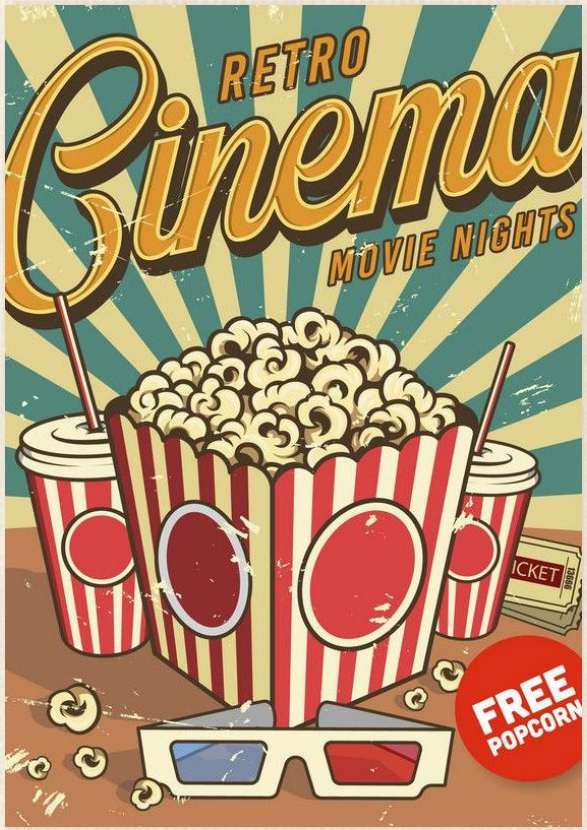
Users with active subscriptions:

```
26 •  SELECT u.Name, s.Plan, s.StartDate, s.EndDate
27    FROM Users u
28    JOIN Subscriptions s ON u.UserID = s.UserID
29    WHERE s.EndDate > CURRENT_DATE;
```

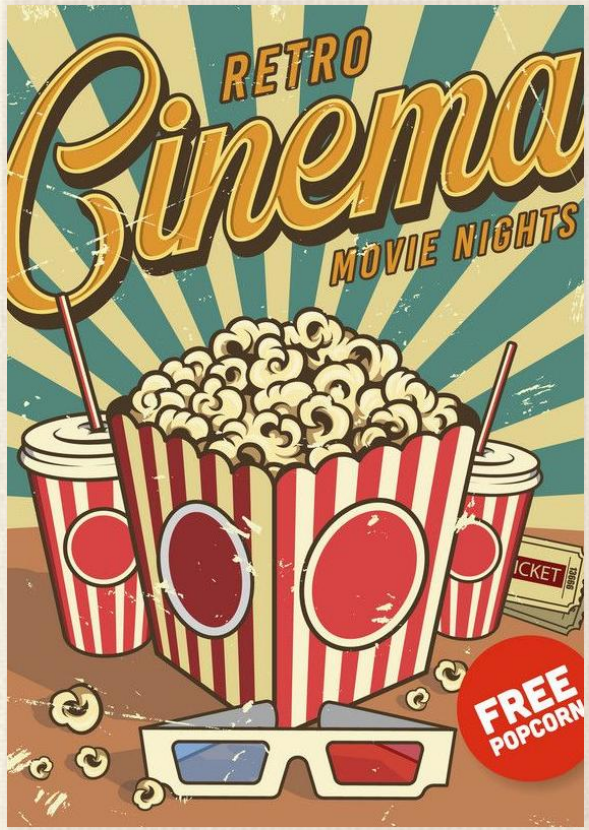
Result Grid | Filter Rows: | Expo

Name	Plan	StartDate	EndDate
Alice Smith	Premium	2024-12-01	2025-12-01
Bob Johnson	Basic	2024-12-10	2025-12-10




RETRO
Cinema
MOVIE NIGHTS
FREE
POPCORN

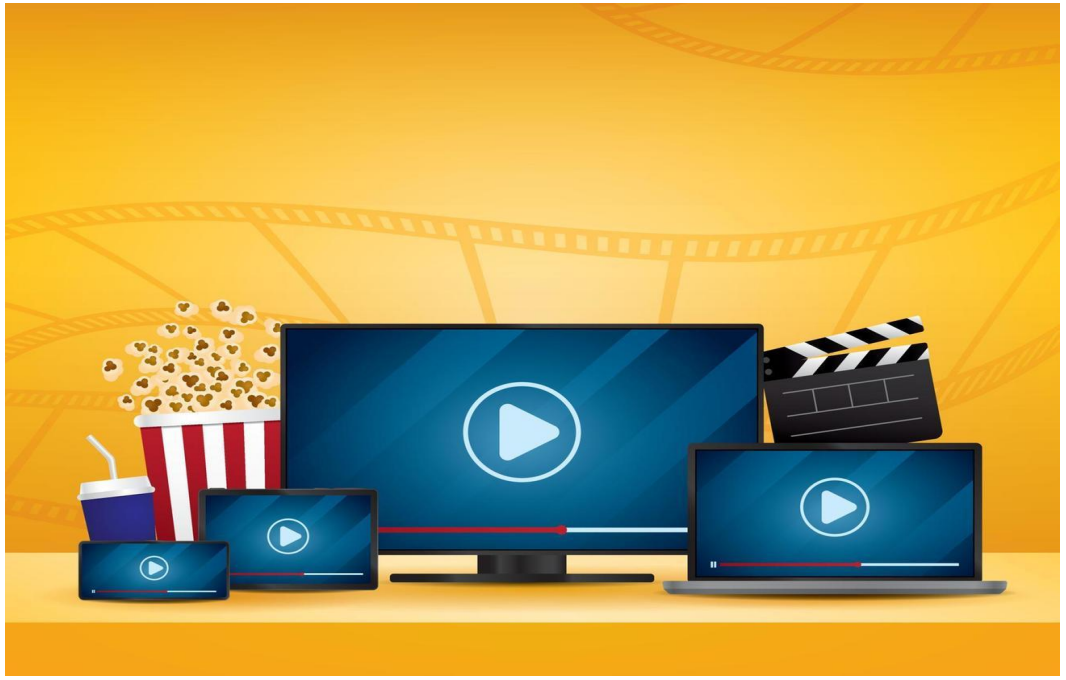

tv


RETRO
Cinema
MOVIE NIGHTS
FREE
POPCORN

VIEWS

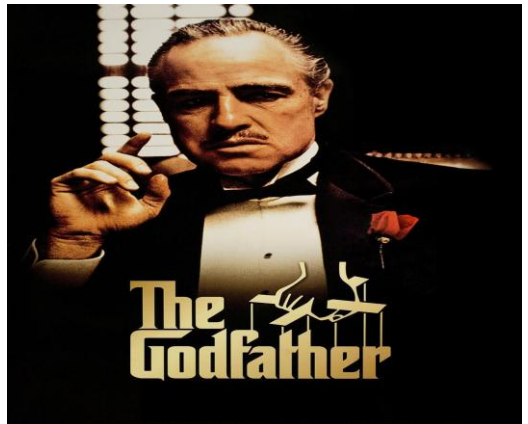
TopRatedMovies – Movies with rating > 4:

```
130 •   CREATE VIEW TopRatedMovies AS
131     SELECT m.Title, AVG(r.Rating) AS AvgRating
132     FROM Movies m
133     JOIN Ratings r ON m.MovieID = r.MovieID
134     GROUP BY m.MovieID
135     HAVING AVG(r.Rating) > 4;
```



Result Grid | Filter Rows:

Title	AvgRating
Edge of Tomorrow	4.5000
The Godfather	5.0000
Interstellar	5.0000



UserActivitySummary – Number of movies watched per user:

```
136 •   CREATE VIEW UserActivitySummary AS
137     SELECT u.UserID, u.Name, COUNT(w.WatchID) AS TotalWatched
138     FROM Users u
139     JOIN WatchHistory w ON u.UserID = w.UserID
140     GROUP BY u.UserID, u.Name;
```

Result Grid | Filter Rows:

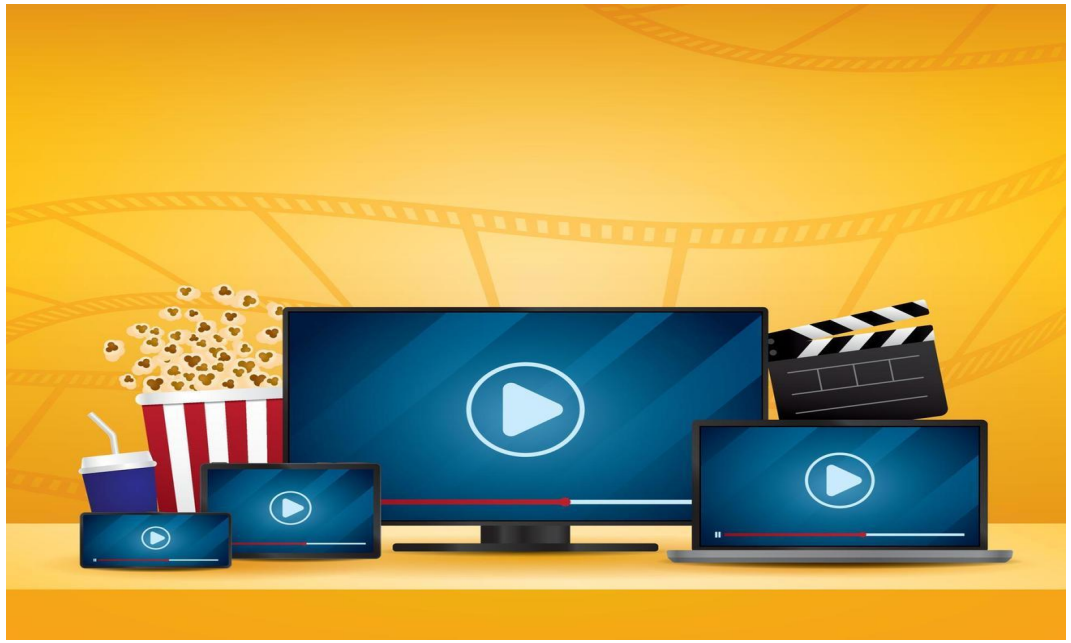
UserID	Name	TotalWatched
201	Alice Smith	3
202	Bob Johnson	3
203	Charlie Davis	3

GenrePopularity – Total views per genre:

```
141 •   CREATE VIEW GenrePopularity AS
142     SELECT g.GenreName, COUNT(*) AS Views
143     FROM WatchHistory w
144     JOIN Movies m ON w.MovieID = m.MovieID
145     JOIN Genres g ON m.GenreID = g.GenreID
146     GROUP BY g.GenreName
147     ORDER BY Views DESC;
```

Result Grid | Filter Rows

GenreName	Views
Action	2
Horror	2
Sci-Fi	2
Comedy	2
Drama	1



CONCLUSION

This SQL project successfully demonstrates how a well-structured relational database can support the core functionalities of a movie streaming platform. By organizing data into meaningful tables—such as users, movies, genres, watch history, ratings, and subscriptions—we've created a system that enables efficient data storage, user behavior tracking, personalized recommendations, and business analytics. The use of SQL queries, including subqueries and aggregations, allowed us to extract valuable insights like active user engagement, popular genres, and user preferences. This project not only enhances understanding of database design and querying but also reflects real-world applications in digital entertainment platforms.

NETFLIX

prime video

HBO NOW

YouTube TV

Disney+

tv+