Movie Analytics & Insights Platform

Database Name: ANTIK

SQL Database + Qlik Sense Dashboard Project

Prepared by

Md Anamul Haque

Bachelor of Science Computer Engineering(Software)

University of Duisburg-Essen

Date: May 2024

Project Overview

- The Movie Analytics & Insights Platform is a complete data analytics project built to explore movie performance, genres, ratings, cast involvement, and country-level patterns.
- It uses a relational database to store detailed movie information and an interactive Qlik Sense dashboard for insights.

Goals of the Project

- - Build a clean relational SQL database for movie data.
- - Load and transform movie metadata.
- - Build an interactive Qlik Sense dashboard.
- - Create KPIs and visualizations for data insights.
- - Deliver an end-to-end BI project workflow

Database Architecture

- The database is based on a normalized schema consisting of movies, actors, directors, genres, ratings, countries, production companies, and business data.
- The design ensures referential integrity and supports analytical queries.

Data Processing Workflow

- 1. Data Collection: IMDb, CSV files, Excel.
- 2. Data Cleaning: Standardization, duplicates removal, formatting.
- 3. Data Loading: SQL DDL, inserts, bulk loaders.

Qlik Sense Dashboards

- The project delivers four analytical dashboards:
- - Movie Overview
- - Ratings & Business Analysis
- - Actors & Directors
- - Geography & Production

Technologies Used

- - MySQL
- - SQL
- - Qlik Sense
- - Excel / Python

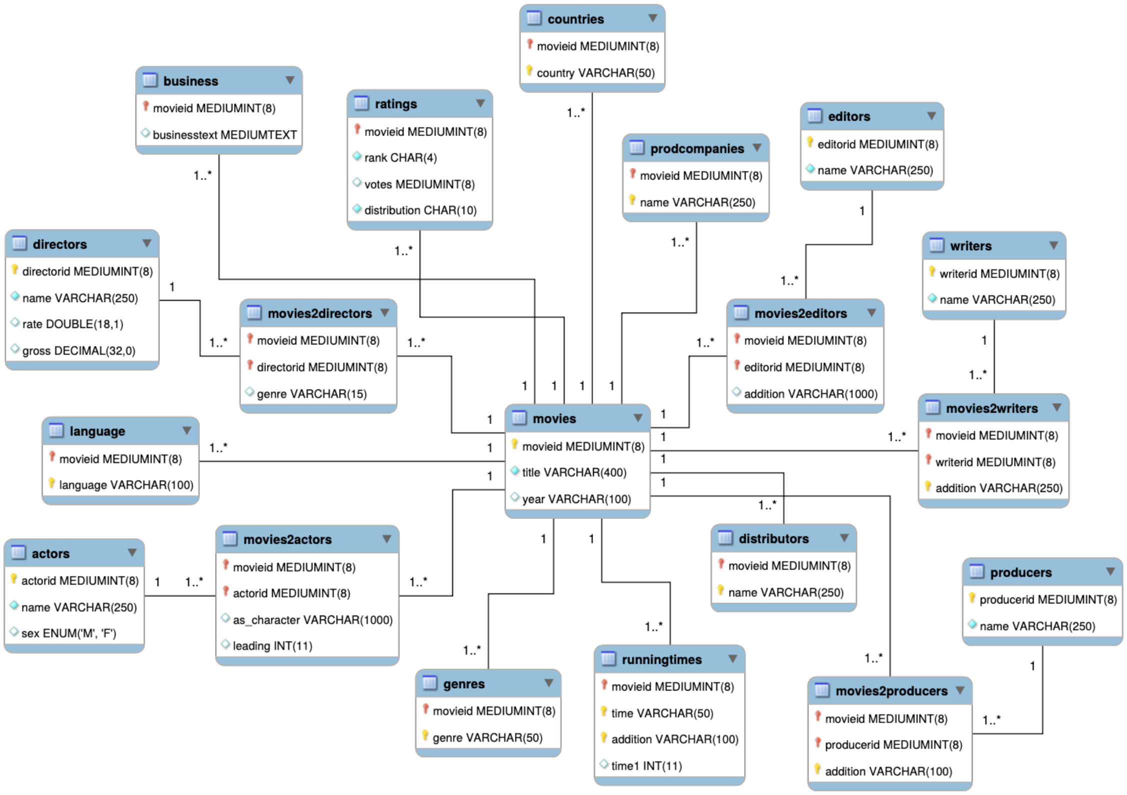
UML Class Diagram

- The UML class diagram used in this project is sourced from the University's course material (UDE). The database design is based on this diagram.

Build a clean relational SQL database for movie data.

Objective:
The goal of this phase is to design and implement a well-structured relational SQL database that stores movie-related data efficiently, ensures data integrity, and supports querying for business intelligence and analytics purposes.

Description:
Based on the UML class diagram sourced from the university (UDE), the database is structured into multiple tables such as **Movie, Actor, Director**, and **MovieActor** (junction table). Each table includes appropriate primary keys, foreign keys, and constraints to maintain relationships and enforce data consistency.



Key Steps:

1. Designing tables according to the UML class diagram.
2. Defining primary keys and foreign keys for relationships.
3. Implementing the tables using SQL `CREATE TABLE` statements.
4. Inserting sample data using `INSERT INTO` statements to validate the database.
5. Ensuring referential integrity and normalization for optimal performance.

SQL Table Creation

```
CREATE DATABASE ANTIK;
SHOW DATABASES;
USE ANTIK;
CREATE TABLE movies(
            movieid INT PRIMARY KEY,
  title VARCHAR(400),
  year VARCHAR(100)
);

INSERT INTO movies(movieid,title,year)
 VALUES   (101,'pother pachali',1920),
                        (102, 'The Shawshank Redemption',
1994),
              (103, 'Pulp Fiction', 1994)……..
```

```
SELECT * FROM movies;
CREATE TABLE directors(
            directorid INT PRIMARY KEY,
  name VARCHAR(250),
  rate DOUBLE(18,1),
  gross DECIMAL(32,0)
);

INSERT INTO
directors(directorid,name,rate,gross)
VALUES(1001, 'Christopher Nolan', 185.2, 5800),
            (1002, 'Steven Spielberg', 172.5,
10500),
            (1003, 'Martin Scorsese', 163.4, 3700),
            (1004, 'Quentin Tarantino', 159.7,
2200)…..
```

```
CREATE TABLE actors(
            actorid INT PRIMARY KEY,
  name VARCHAR(250),
  sex ENUM('M','F')
);

INSERT INTO actors(actorid,name,sex)
VALUES(2001, 'Leonardo DiCaprio', 'M'),
(2002, 'Meryl Streep', 'F'),
(2003, 'Denzel Washington', 'M'),
(2004, 'Natalie Portman', 'F'),…..
```

```
CREATE TABLE editors(
            editorid INT PRIMARY KEY,
  name VARCHAR(250)
);
SELECT * FROM editors;
INSERT INTO editors(editorid,name)
VALUES(3001, 'Thelma Schoonmaker'),
(3002, 'Walter Murch'),
(3003, 'Dody Dorn'),
(3004, 'Lee Smith'),……
```

```
CREATE TABLE writers(
            writerid INT PRIMARY KEY,
  name VARCHAR(250)
);
SELECT * FROM writers;
INSERT INTO writers(writerid,name)
VALUES(4001, 'Aaron Sorkin'),
(4002, 'Quentin Tarantino'),
(4003, 'Charlie Kaufman'),
(4004, 'Nora Ephron'),…..
```

```
CREATE TABLE producers(
            producerid INT PRIMARY KEY,
  name VARCHAR(250)
);
SELECT * FROM producers;
INSERT INTO producers(producerid,name)
VALUES(5001, 'Kathleen Kennedy'),
(5002, 'Kevin Feige'),
(5003, 'Jerry Bruckheimer'),….
```

```
CREATE TABLE language(
            movie_id INT,
            language VARCHAR(100),
            FOREIGN KEY(movie_id) REFERENCES
movies(movieid)
);
INSERT INTO language(movie_id, language)
VALUES(101, 'English'),
            (102, 'Spanish'),
            (103, 'French'),
            (104, 'German')
```

```
CREATE TABLE genres(
            movieid INT,
            genre VARCHAR(250),
  FOREIGN KEY(movieid) REFERENCES
movies(movieid)
);
SELECT * FROM genres;
INSERT INTO genres(movieid,genre)
VALUES(101, 'Action'),
            (102, 'Drama'),
            (103, 'Comedy'),
            (104, 'Romance'),
```

```
CREATE TABLE business(
            movieid INT,
  businesstext TEXT,
  FOREIGN KEY(movieid) REFERENCES
movies(movieid)
);
```

```
CREATE TABLE ratings(
            movieid INT,
            votes INT,
            ranking CHAR(4),
  distribution CHAR(4),
            FOREIGN KEY(movieid) REFERENCES
movies(movieid)
);
INSERT INTO ratings(movieid,votes,ranking,distribution)
VALUES(101, 1500, '8.5', 'A'),
(102, 2300, '7.9', 'B'),
(103, 1800, '8.2', 'A'),
```

```
CREATE TABLE countries(
            movieid INT,
  country VARCHAR(250),
  FOREIGN KEY(movieid) REFERENCES
movies(movieid)
);
```

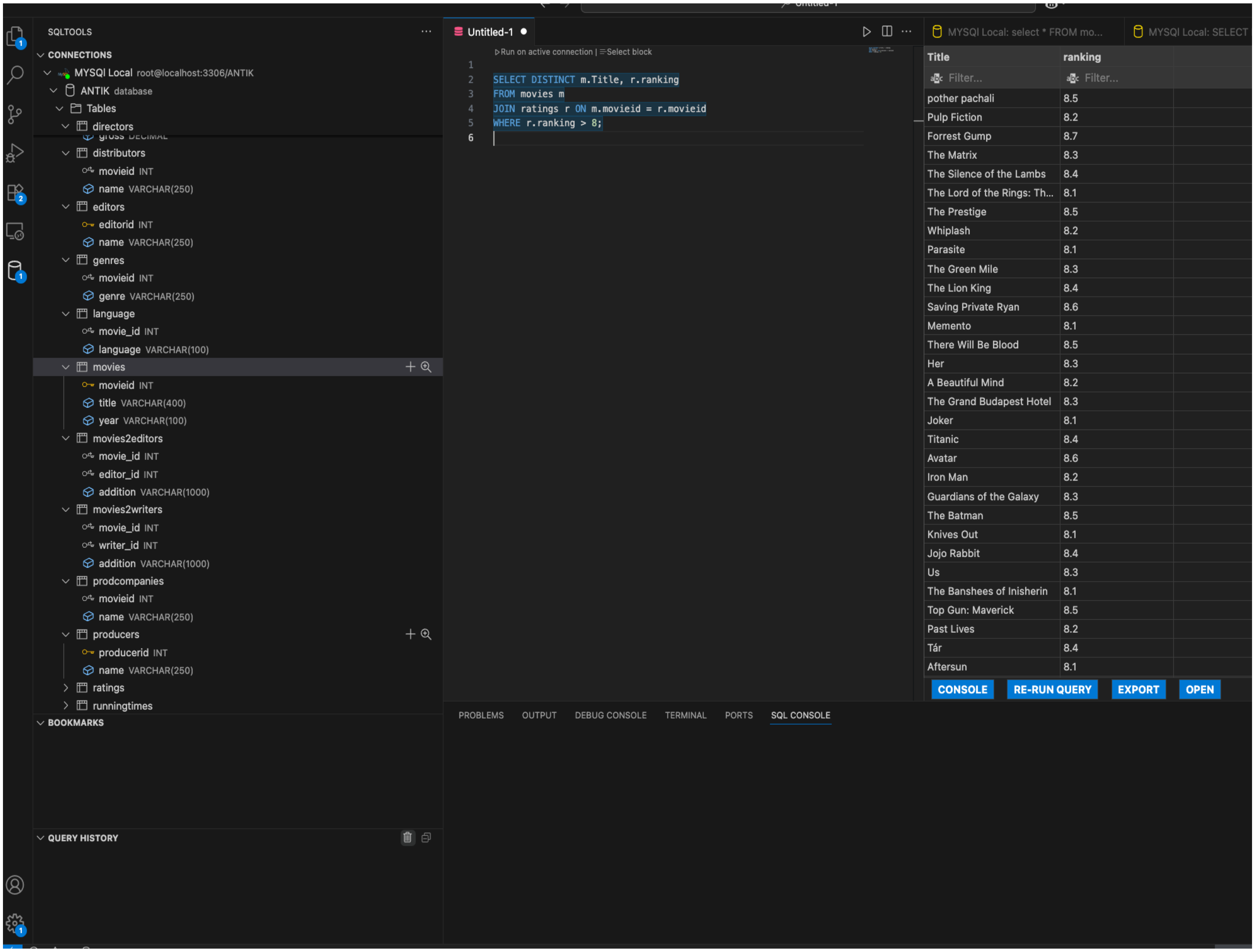
```
CREATE TABLE prodcompanies(
            movieid INT,
  name VARCHAR(250),
  FOREIGN KEY(movieid) REFERENCES
movies(movieid)
);
```

```
CREATE TABLE runningtimes(
        movietimeid INT,
  time VARCHAR(50),
  addition VARCHAR(100),
  time1 INT,
  FOREIGN KEY(movietimeid) REFERENCES
movies(movieid)
);
```

```
CREATE TABLE movies2editors(
    movie_id INT,
    editor_id INT,
    FOREIGN KEY(movie_id) REFERENCES
movies(movieid),
 FOREIGN KEY(editor_id) REFERENCES
editors(editorid),
 addition VARCHAR(1000)
);
INSERT INTO
movies2editors(movie_id,editor_id,addition)
VALUES(101, 3001, 'Initial edit'),
(102, 3002, 'Color correction'),
(103, 3003, 'Sound editing'),....
```

```
CREATE TABLE movies2writers(
 movie_id INT,
 writer_id INT,
 FOREIGN KEY(movie_id) REFERENCES
movies(movieid),
 FOREIGN KEY(writer_id) REFERENCES
writers(writerid),
 addition VARCHAR(1000)
);
SELECT * FROM movies2writers;
INSERT INTO
movies2writers(movie_id,writer_id,addition)
VALUES
(101, 4001, 'Original screenplay'),
(102, 4002, 'Adapted from novel'),
(103, 4003, 'Inspired by true events'),.....
```

Here is an overview of VS Code for demonstration.



```
MYSQl Local root@localhost:3306/ANTIK
  ANTIK database
    Tables
      directors
        gross DECIMAL
      distributors
        movieid INT
        name VARCHAR(250)
      editors
        editorid INT
        name VARCHAR(250)
      genres
        movieid INT
        genre VARCHAR(250)
      language
        movie_id INT
        language VARCHAR(100)
      movies
        movieid INT
        title VARCHAR(400)
        year VARCHAR(100)
      movies2editors
        movie_id INT
        editor_id INT
        addition VARCHAR(1000)
      movies2writers
        movie_id INT
        writer_id INT
        addition VARCHAR(1000)
      prodcompanies
        movieid INT
        name VARCHAR(250)
      producers
        producerid INT
        name VARCHAR(250)
      ratings
      runningtimes
```

▷Run on active connection | ≡Select block

```
SELECT DISTINCT m.Title, r.ranking
FROM movies m
JOIN ratings r ON m.movieid = r.movieid
WHERE r.ranking > 8;
```

Title	ranking
pother pachali	8.5
Pulp Fiction	8.2
Forrest Gump	8.7
The Matrix	8.3
The Silence of the Lambs	8.4
The Lord of the Rings: Th...	8.1
The Prestige	8.5
Whiplash	8.2
Parasite	8.1
The Green Mile	8.3
The Lion King	8.4
Saving Private Ryan	8.6
Memento	8.1
There Will Be Blood	8.5
Her	8.3
A Beautiful Mind	8.2
The Grand Budapest Hotel	8.3
Joker	8.1
Titanic	8.4
Avatar	8.6
Iron Man	8.2
Guardians of the Galaxy	8.3
The Batman	8.5
Knives Out	8.1
Jojo Rabbit	8.4
Us	8.3
The Banshees of Inisherin	8.1
Top Gun: Maverick	8.5
Past Lives	8.2
Tár	8.4
Aftersun	8.1

Creating KPIs and Dashboard Using Qlik Sense

In this phase of the project, we leveraged **Qlik Sense** to transform the movie database into an **interactive analytics dashboard**. The goal was to create meaningful **Key Performance Indicators (KPIs)** and visualizations that provide insights into movie performance, genre trends, and audience ratings.

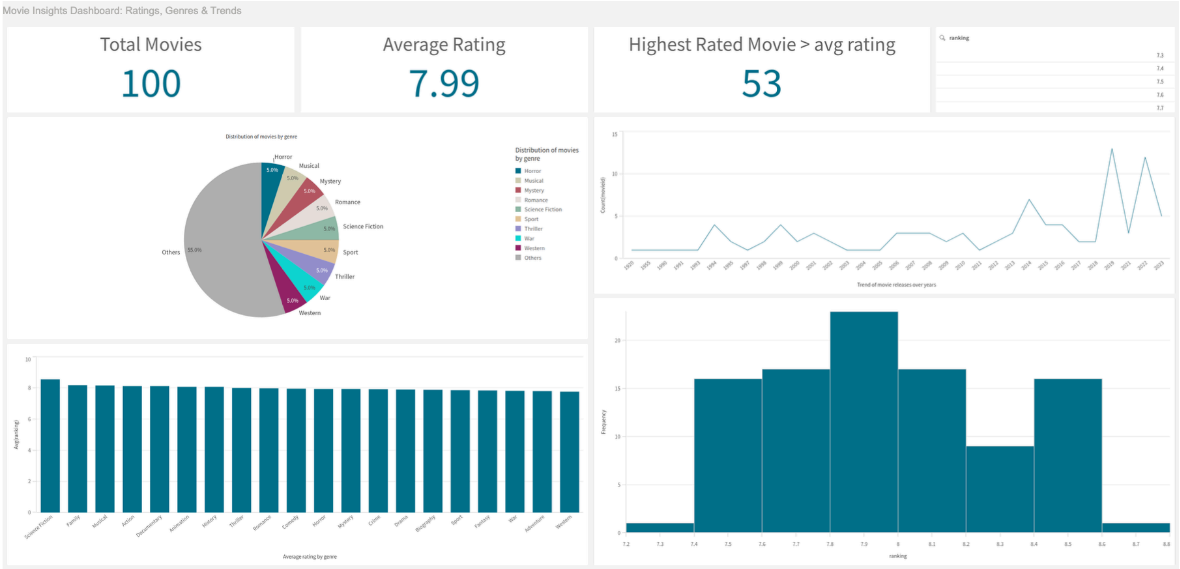
Key KPIs Created:

1. **Total Movies:** Shows the total number of movies in the dataset.
2. **Average Rating:** Displays the average rating across all movies.
3. **Highest Rated Movie:** Highlights the top-rated movie based on user ratings.
4. **Most Popular Genre:** Indicates the genre with the highest number of movies.

Dashboard Features:

- **Interactive Filters:** Users can filter movies by **genre, year, or rating range**, and all visualizations update dynamically.
- **Visualizations:**
  - **Bar Chart:** Average rating by genre.
  - **Pie Chart:** Distribution of movies by genre.
  - **Line Chart:** Trend of movie releases over years.
- **Interactivity:** Selecting a genre or year highlights relevant data across all charts and KPIs, enabling easy comparison and trend analysis.

The Qlik Sense dashboard effectively turns raw movie data into **actionable insights**, allowing users to analyze trends, identify top-performing movies and genres, and explore audience preferences in a visually intuitive



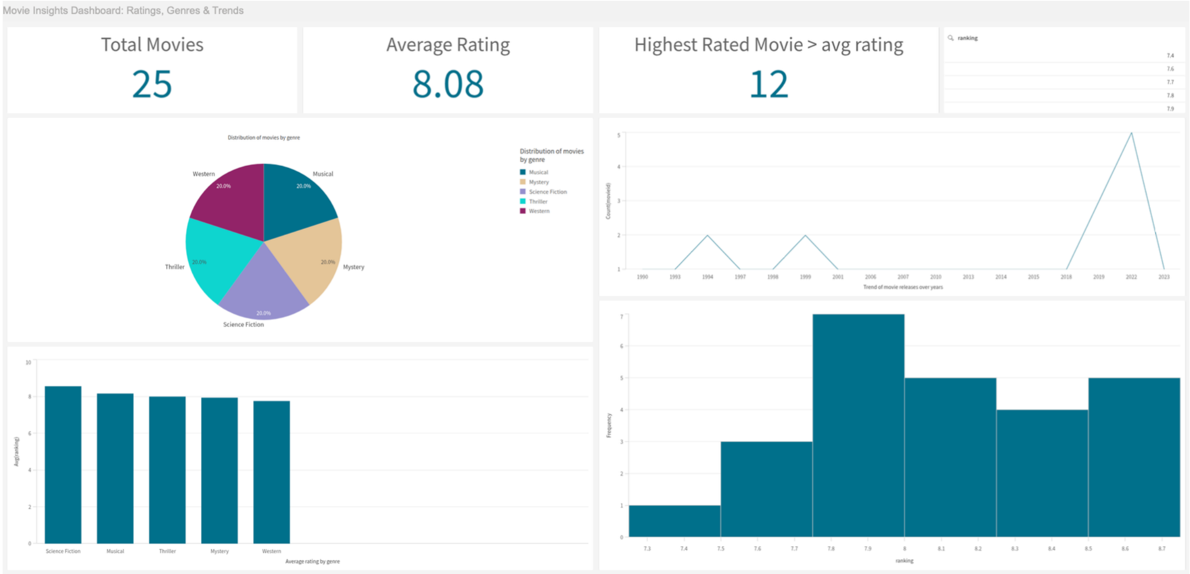
Movie Insights Dashboard: Ratings, Genres & Trends

Total Movies
100

Average Rating
7.99

Highest Rated Movie > avg rating
53

Filter movies by genre to explore trends and ratings in Western, Musical, Thriller, Science Fiction, and Mystery films.

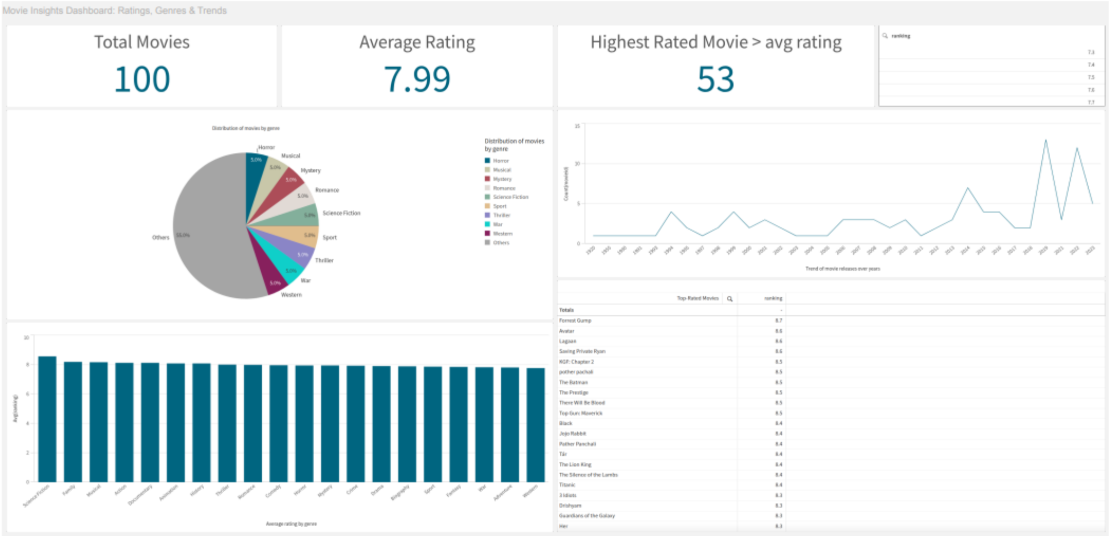


Movie Insights Dashboard: Ratings, Genres & Trends

Total Movies	Average Rating	Highest Rated Movie > avg rating
25	8.08	12

ranking
7.4
7.6
7.7
7.8
7.9

This section highlights the top-rated movies based on user ratings and reviews. It helps the audience quickly discover critically acclaimed films across different genres, making it easier to choose high-quality movies to watch. The list updates dynamically based on filters like genre, year, and rating, ensuring personalized recommendations for every viewer.



Movie Insights Dashboard: Ratings, Genres & Trends

Total Movies	Average Rating	Highest Rated Movie > avg rating
100	7.99	53

ranking
7.3
7.4
7.5
7.6
7.7

Top Rated Movies	ranking
Totals	-
Forrest Gump	8.7
Avatar	8.6
Lagaan	8.6
Saving Private Ryan	8.6
KGF: Chapter 2	8.5
pother pachali	8.5
The Batman	8.5
The Prestige	8.5
There Will Be Blood	8.5
Top Gun: Maverick	8.5
Black	8.4
Jojo Rabbit	8.4
Pather Panchali	8.4
Tár	8.4
The Lion King	8.4
The Silence of the Lambs	8.4
Titanic	8.4
3 Idiots	8.3
Drishyam	8.3
Guardians of the Galaxy	8.3
Her	8.3

Conclusion

The Movie Insights Dashboard successfully demonstrates the end-to-end process of **data collection, modeling, and visualization** using Qlik Sense. By integrating movie, genre, and rating data, the dashboard provides interactive insights into **movie performance, genre trends, and audience preferences**. Users can easily explore top-rated movies, identify the most popular genres, and analyze patterns over time through dynamic KPIs, charts, and filters. This project highlights the power of **associative data models and interactive dashboards** to turn raw data into actionable insights, making it an effective tool for both learning and professional analytics. Overall, it showcases practical skills in **data analytics, visualization, and business intelligence**, and serves as a strong portfolio project for demonstrating Qlik Sense proficiency.