Movie Database Management System

Name: Sakshi Paralekar

Roll No: 459

College: Patkar Varde College, Mumbai

Date: 14-06-2025

Project Overview

The Movie Database Management System is a SQL-based project designed to manage information related to movies, actors, genres, and reviews.
The goal is to analyze the most popular movies, top actors, and genre performance through structured SQL queries.

ER Diagram (Text Representation)

[Movies]
- movie_id (PK)
- title
- release_year
- genre_id (FK)
- duration

[Genres]
- genre_id (PK)
- genre_name

[Actors]
- actor_id (PK)

- actor_name
- birth_year

[Movie_Actors]
- movie_id (PK, FK)
- actor_id (PK, FK)
- role

[Reviews]
- review_id (PK)
- movie_id (FK)
- rating
- review_text
- review_date

SQL Table Creation

Genres Table

```
CREATE TABLE Genres (
    genre_id INT PRIMARY KEY AUTO_INCREMENT,
    genre_name VARCHAR(50)
);
```

Movies Table

```
CREATE TABLE Movies (
    movie_id INT PRIMARY KEY AUTO_INCREMENT,
    title VARCHAR(100),
    release_year INT,
    genre_id INT,
```

```
    duration INT,
    FOREIGN KEY (genre_id) REFERENCES Genres(genre_id)
);
```

Actors Table

```
CREATE TABLE Actors (
    actor_id INT PRIMARY KEY AUTO_INCREMENT,
    actor_name VARCHAR(100),
    birth_year INT
);
```

Movie_Actors Table

```
CREATE TABLE Movie_Actors (
    movie_id INT,
    actor_id INT,
    role VARCHAR(100),
    PRIMARY KEY (movie_id, actor_id),
    FOREIGN KEY (movie_id) REFERENCES Movies(movie_id),
    FOREIGN KEY (actor_id) REFERENCES Actors(actor_id)
);
```

Reviews Table

```
CREATE TABLE Reviews (
    review_id INT PRIMARY KEY AUTO_INCREMENT,
    movie_id INT,
    rating DECIMAL(2,1),
    review_text TEXT,
    review_date DATE,
    FOREIGN KEY (movie_id) REFERENCES Movies(movie_id)
);
```

Sample SQL Queries

Top 5 Rated Movies

```
SELECT m.title, AVG(r.rating) AS avg_rating
FROM Movies m
JOIN Reviews r ON m.movie_id = r.movie_id
GROUP BY m.title
ORDER BY avg_rating DESC
LIMIT 5;
```

Most Popular Genre

```
SELECT g.genre_name, COUNT(*) AS movie_count
FROM Movies m
JOIN Genres g ON m.genre_id = g.genre_id
GROUP BY g.genre_name
ORDER BY movie_count DESC
LIMIT 1;
```

All Movies of Actor 'Amitabh Bachchan'

```
SELECT m.title, m.release_year
FROM Movies m
JOIN Movie_Actors ma ON m.movie_id = ma.movie_id
JOIN Actors a ON ma.actor_id = a.actor_id
WHERE a.actor_name = 'Amitabh Bachchan';
```

Conclusion

This project demonstrates the use of SQL in managing and analyzing data in a movie database. It helps track actor participation, genre performance,
and user feedback through reviews. The relational model improves query performance and data integrity.