Movie

Database

System

Project

**INDEX**

1. **Introduction**

* Overview of the Movie Database System
* Objectives and Goals

2. **SQL Structure for Database**

* Database Schema
* Table Definitions
* Creating Tables in MySQL

3. **UML Diagrams**

* Class Diagram
* Activity Diagram
* Use Case Diagram
* Sequence Diagram
* Object Diagram

4. **Inserting Data to Database**

* SQL Queries for Inserting Data
* Example Data for Movies, Actors, Directors, Genres, Users, and Ratings

5. **References**

* Sources and Resources Used for Project Development

**INTRODUCTION**

In the realm of data management, creating an efficient and comprehensive database is crucial for organizing and retrieving information seamlessly. This project focuses on developing a robust Movie Database System using MySQL, designed to manage a vast array of movie-related data. The system aims to provide an intuitive and efficient way to store, manage, and analyze information about movies, actors, directors, genres, and user ratings.

The primary objective of this project is to design and implement a relational database that supports complex queries and analytical functions, facilitating the retrieval and manipulation of movie data. The database schema includes several interconnected tables: Movies, Actors, Directors, Genres, Users, and Ratings. This design allows for a detailed and organized structure, where each table serves a specific purpose. The Movies table captures essential details about each film, including its title, release year, director, and genre. The Actors and Directors tables store information about the individuals involved in the filmmaking process, while the Genres table categorizes films into different types.

To enhance user interaction, the Users table tracks user profiles, and the Ratings table records the ratings given by users to various movies. This setup not only supports basic operations such as data insertion, update, and deletion but also enables advanced queries for data analysis. For instance, users can retrieve information about the top-rated movies, analyze genre popularity, and explore actor and director contributions.

**Purpose:** Develop a comprehensive Movie Database System using MySQL to efficiently manage and analyze movie-related data.

**Objectives:**

* Design a relational database to store detailed information about movies, actors, directors, genres, and user ratings.
* Implement a structure that supports complex queries and data manipulation.

**Database Structure:**

* **Movies Table:** Contains details about each movie, including title, release year, director, and genre.
* **Actors Table:** Stores information about actors, including their names and birthdates.
* **Directors Table:** Captures details about directors, including names and birthdates.
* **Genres Table:** Categorizes movies into various genres.
* **Users Table:** Tracks user profiles and contact information.
* **Ratings Table:** Records user ratings for different movies, allowing for analysis of movie popularity and user preferences.

**Features:**

* Supports basic operations like data insertion, updating, and deletion.
* Enables advanced querying for retrieving top-rated movies, analyzing genre popularity, and exploring contributions by actors and directors.

**Benefits:**

* Facilitates seamless data retrieval and management for movie-related queries.
* Provides a platform for generating insights into movie trends and user preferences.

**Goal:** Create a dynamic and scalable database system that enhances data management capabilities and supports various applications, including movie recommendation systems and data analytics.

**SQL Structure for Database**

Create Database movie\_project;

CREATE TABLE Movies (

movie\_id INT PRIMARY KEY,

title VARCHAR(255) NOT NULL,

release\_year YEAR NOT NULL,

director\_id INT,

genre\_id INT,

FOREIGN KEY (director\_id) REFERENCES Directors(director\_id),

FOREIGN KEY (genre\_id) REFERENCES Genres(genre\_id)

);

CREATE TABLE Actors (

actor\_id INT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

birthdate DATE

);

CREATE TABLE Directors (

director\_id INT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

birthdate DATE

);

CREATE TABLE Genres (

genre\_id INT PRIMARY KEY,

genre\_name VARCHAR(255) NOT NULL

);

CREATE TABLE Users (

user\_id INT PRIMARY KEY,

username VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL

);

CREATE TABLE Ratings (

rating\_id INT PRIMARY KEY,

movie\_id INT,

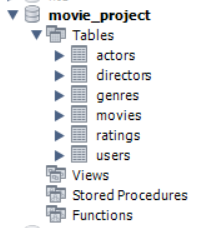
user\_id INT,

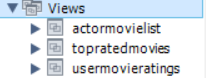
rating INT CHECK (rating BETWEEN 1 AND 10),

FOREIGN KEY (movie\_id) REFERENCES Movies(movie\_id),

FOREIGN KEY (user\_id) REFERENCES Users(user\_id)

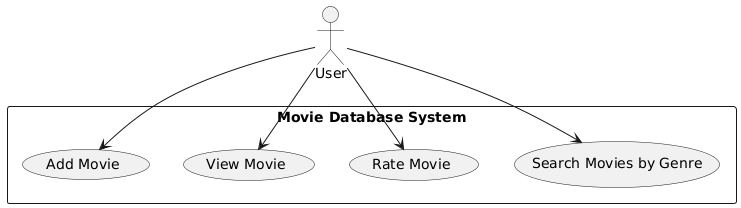
);



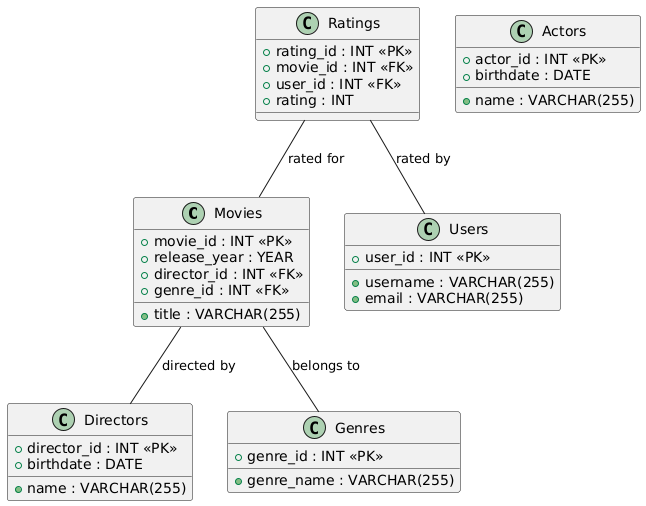


**UML**

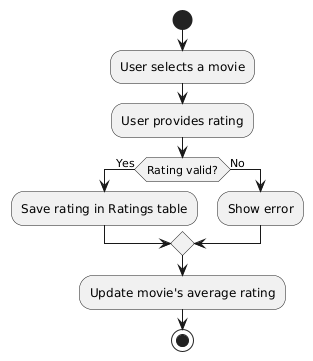
**Usecase**

****

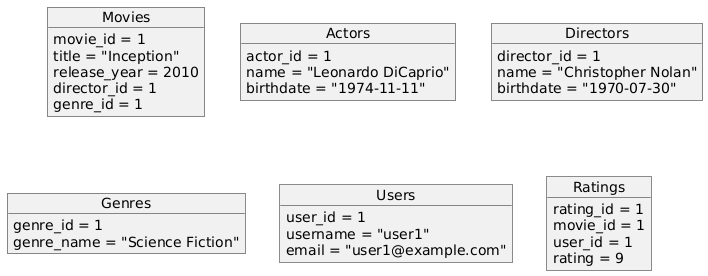
**Class diagram**

****

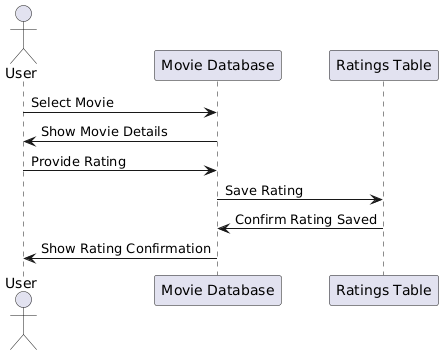
**Activity**

****

**Object**

****

**Sequence**

****

**Inserting Data to Database**

INSERT INTO Movies (movie\_id, title, release\_year, director\_id, genre\_id) VALUES

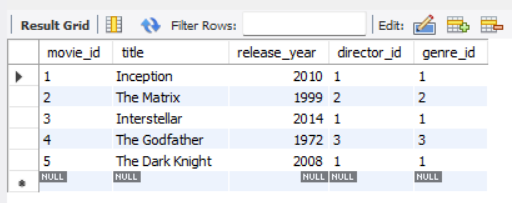
(1, 'Inception', 2010, 1, 1),

(2, 'The Matrix', 1999, 2, 2),

(3, 'Interstellar', 2014, 1, 1),

(4, 'The Godfather', 1972, 3, 3),

(5, 'The Dark Knight', 2008, 1, 1);



INSERT INTO Actors (actor\_id, name, birthdate) VALUES

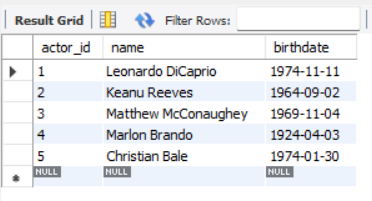
(1, 'Leonardo DiCaprio', '1974-11-11'),

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

(3, 'Matthew McConaughey', '1969-11-04'),

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

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

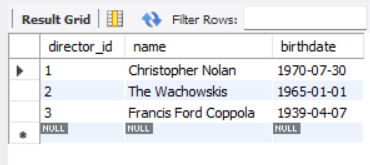


INSERT INTO Directors (director\_id, name, birthdate) VALUES

(1, 'Christopher Nolan', '1970-07-30'),

(2, 'The Wachowskis', '1965-01-01'),

(3, 'Francis Ford Coppola', '1939-04-07');

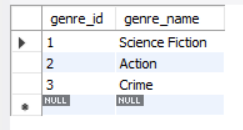


INSERT INTO Genres (genre\_id, genre\_name) VALUES

(1, 'Science Fiction'),

(2, 'Action'),

(3, 'Crime');

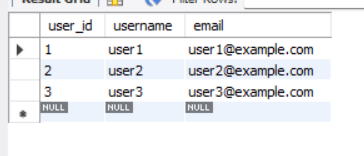


INSERT INTO Users (user\_id, username, email) VALUES

(1, 'user1', 'user1@example.com'),

(2, 'user2', 'user2@example.com'),

(3, 'user3', 'user3@example.com');



INSERT INTO Ratings (rating\_id, movie\_id, user\_id, rating) VALUES

(1, 1, 1, 9),

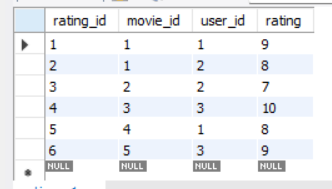
(2, 1, 2, 8),

(3, 2, 2, 7),

(4, 3, 3, 10),

(5, 4, 1, 8),

(6, 5, 3, 9);



**Views**

1. View: Top Rated Movies

This view will show movies with an average rating of 8 or above, along with the director and genre.

sql

CREATE VIEW TopRatedMovies AS

SELECT

M.title AS MovieTitle,

AVG(R.rating) AS AverageRating,

D.name AS DirectorName,

G.genre\_name AS Genre

FROM

Movies M

JOIN Ratings R ON M.movie\_id = R.movie\_id

JOIN Directors D ON M.director\_id = D.director\_id

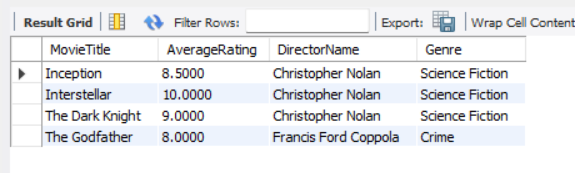
JOIN Genres G ON M.genre\_id = G.genre\_id

GROUP BY

M.movie\_id, M.title, D.name, G.genre\_name

HAVING

AVG(R.rating) >= 8;



**2. View: Actor Movie List**

This view will show a list of all movies along with the actors that starred in them. You can modify this further if you have an actor-movie relationship table.

sql

CREATE VIEW ActorMovieList

SELECT

A.name AS ActorName,

M.title AS MovieTitle,

M.release\_year AS ReleaseYear

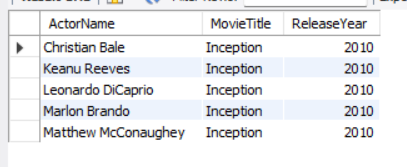
FROM

actors A

JOIN Movies M ON M.movie\_id = 1

ORDER BY

A.name, M.release\_year;



**3. View: User Movie Ratings**

This view will show a list of users and the movies they rated, along with their rating.

sql

CREATE VIEW UserMovieRatings AS

SELECT

U.username AS Username,

M.title AS MovieTitle,

R.rating AS UserRating

FROM

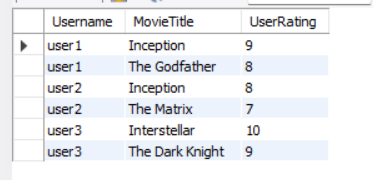
Users U

JOIN Ratings R ON U.user\_id = R.user\_id

JOIN Movies M ON R.movie\_id = M.movie\_id

ORDER BY

U.username, M.title;

****

**Reference**

|  |  |
| --- | --- |
| **1. Platform used** : | [mySQL workbench](https://www.mysql.com/) community edition 8.0 CE |
| **2. UML Tool** : | [StarUml](https://staruml.io/) |
| **3. queries reference** : | [W3Schools](https://www.w3schools.com/sql/) |