📽️ HeartfeltCinema – SQL Project

👷 **Developed by:** Akshat Patel
🗂️ **Project Type:** SQL Database Project
🗓️ **Year:** 2025

📌 Project Overview

HeartfeltCinema is a SQL-based relational database designed to manage a romantic movie review and rating platform. This project stores data related to movies, genres, directors, actors, users, ratings, and reviews, offering efficient querying, analytics, and automation using advanced SQL features.

🛠️ Tech Stack

- **Database:** MySQL
- **Language:** SQL
- **Tools:** MySQL Workbench / phpMyAdmin / Command Line (any interface)

🧱 Database Schema Includes:

📑 Database Schema 🎬 Core Tables & Relationships Each table is interconnected to maintain data integrity and optimize data retrieval:

- **Genres (genre_id)** – Categorizes movies (e.g., Romantic Fantasy, Love Triangle).
- **Directors (director_id)** – Stores information about film directors.
- **Actors (actor_id)** – – Contains details of cast members.
- **Movies (movie_id)** – Core movie details with relationships to genres & directors. 🎞️ Each movie belongs to one genre (genre_id → FK). 🎥 Each movie has one director (director_id → FK).
- **Cast(movie_id,actor_id)** – Connects actors to their roles in movies.
- **Users(user_id)** – Stores information about Platform users with email and username.
- **Ratings(rating_id)** – Allows users to rate movies (0-10).
- **Reviews(review_id)** – Stores user-submitted text reviews.

- **Favorites(user_id,movie_id)** – Tracks favorite movies for each user.

🚀 Key Features

- Normalized schema with foreign key constraints.
- Meaningful Sample data insertion for real-world testing.
- Advanced SQL objects:
  - ✅ Views (**TopRatedRomance**) – Quickly retrieve top-rated romantic movies.
  - ✅ Triggers (auto-fill `rating_date` and `review_date`) – Auto-fill missing review/rating dates.
  - ✅ Stored Procedures (**AddFavorite**) – Simplify user interactions.
  - ✅ Indexes for performance) – Enhance query performance.

🔍 Index Optimization

- Indexes speed up common queries and database searches:
  - idx_movies_genre – Accelerates searches by movie genre.
  - idx_users_email – Improves lookup efficiency by user email.

📊 Analytical queries to find:

- **Top-rated movies**
- **Average ratings by genre**
- **Most loved directors**
- **User engagement** (favorites, reviews)

📊 Sample Query

```
-- Top-rated romantic movies ---

SELECT M.title, AVG(R.rating) AS avg_rating
FROM Movies M
JOIN Ratings R ON M.movie_id = R.movie_id
GROUP BY M.title
ORDER BY avg_rating DESC
LIMIT 10;

--- Most loved directors ---
```

```sql
SELECT D.name, COUNT(DISTINCT R.movie_id) AS movie_count
FROM Directors D
JOIN Movies M ON D.director_id = M.director_id
JOIN Ratings R ON M.movie_id = R.movie_id
GROUP BY D.name
ORDER BY movie_count DESC
LIMIT 5;
```

🔄 Stored Procedure Usage

The `AddFavorite(uid, mid)` stored procedure helps users add movies to their favorites list effortlessly.

📌 Execution Example

CALL AddFavorite(1, 3); -- Adds movie with ID 3 as a favorite for user ID 1

Execution Example:

sql

CALL AddFavorite(1, 3); -- Adds movie with ID 3 as a favorite for user ID 1

⚡ SQL Triggers Trigger to Auto-Fill rating_date if NULL

Sql

DELIMITER $$

CREATE TRIGGER set_rating_date BEFORE INSERT ON Ratings
FOR EACH ROW

BEGIN
IF NEW.rating_date IS NULL THEN
SET NEW.rating_date = CURDATE();
END IF;
END $$

DELIMITER ;

Trigger to Auto-Fill review_date if NULL

sql DELIMITER $$

CREATE TRIGGER set_review_date BEFORE INSERT ON Reviews
FOR EACH ROW

```
BEGIN
IF NEW.review_date IS NULL THEN
SET NEW.review_date = CURDATE();
END IF;
END $$

DELIMITER ;
```

🏁 Conclusion

HeartfeltCinema's database efficiently manages movie details, ratings, and user engagement using optimized SQL techniques. With views, triggers, stored procedures, and indexes, the project ensures seamless data processing and enhances analytics for a robust movie-review platform.