Object Oriented Analysis and Design using Java - UE21CS352B

Self Learning Hands-on Assignment: MVC Framework

Name: Nagaveni L G

SRN:PES2UG21CS315

Q) Design a simple movie database website allows users to browse and rate movies (think an extremely simplified version of letterboxd). It follows the MVC pattern, separating the application into three main components: the Model, the View, and the Controller.

1. A write up on how you have used MVC within your project, highlighting the respective concepts.

Ans: In the movie database project, I utilized the MVC (Model-View-Controller) architectural pattern to separate concerns and maintain a structured codebase. The Movie class represents the Model, encapsulating movie data. The MovieController serves as the Controller, handling HTTP requests, interacting with the Model (via the MovieRepository), and providing data to the View. The View, represented by the movies.html template, displays movie information to users. This separation enables modularity, easier maintenance, and scalability of the application, adhering to the principles of MVC design.

2. Complete screenshot of code written by you

MovieController.java

```java
package com.movie.demo;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import com.movie.demo.repository.MovieRepository;
// import com.movie.demo.Movie;
import java.util.List;
```

```java
@Controller
public class MovieController {

    @Autowired
    private MovieRepository movieRepository;

    @GetMapping("/movies")
    public String getAllMovies(Model model) {
        List<Movie> movies =
movieRepository.findAll();
        model.addAttribute("movies", movies);
        return "movies";
    }
}
```

Movie.java

```java
package com.movie.demo;

import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
// import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
// import jakarta.persistence.Table;

@Entity
public class Movie {

    @Id
    @GeneratedValue(strategy =
GenerationType.IDENTITY)
    private Long id;

    private  String  title;
    private int releaseYear;
    private  String  genre;
    private String director;
    private double averageRating;

    public Movie() {
    }

    public Movie(String title, int releaseYear, String
genre, String director, double averageRating) {
        this.title = title;
        this.releaseYear = releaseYear;
        this.genre = genre;
        this.director = director;
        this.averageRating = averageRating;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }

    public int getReleaseYear() {
        return releaseYear;
    }

    public void setReleaseYear(int releaseYear) {
        this.releaseYear = releaseYear;
    }

    public String getGenre() {
        return genre;
    }

    public void setGenre(String genre) {
        this.genre = genre;
    }

    public String getDirector() {
        return director;
    }

    public void setDirector(String director) {
        this.director = director;
    }

    public double getAverageRating() {
        return averageRating;
    }

    public void setAverageRating(double averageRating)
{
```

```java
        this.averageRating = averageRating;
    }
}
```

Review.java

```java
package com.movie.demo;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class Review {

    @Id
    @GeneratedValue(strategy =
GenerationType.IDENTITY)
    private Long id;

    private Long movieId;
    private Long userId;
    private String comment;

    public Review() {
    }

    public Review(Long movieId, Long userId, String
comment) {
        this.movieId = movieId;
        this.userId = userId;
        this.comment = comment;
    }
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public Long getMovieId() {
        return movieId;
    }

    public void setMovieId(Long movieId) {
        this.movieId = movieId;
    }

    public Long getUserId() {
        return userId;
    }

    public void setUserId(Long userId) {
        this.userId = userId;
    }

    public String getComment() {
        return comment;
    }

    public void setComment(String comment) {
        this.comment = comment;
    }
}
```

User.java

```java
package com.movie.demo;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class User {

    @Id
    @GeneratedValue(strategy =
GenerationType.IDENTITY)
    private Long id;

    private String username;
    private String password;

    public User() {
    }

    public User(String username, String password) {
        this.username = username;
        this.password = password;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }
    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

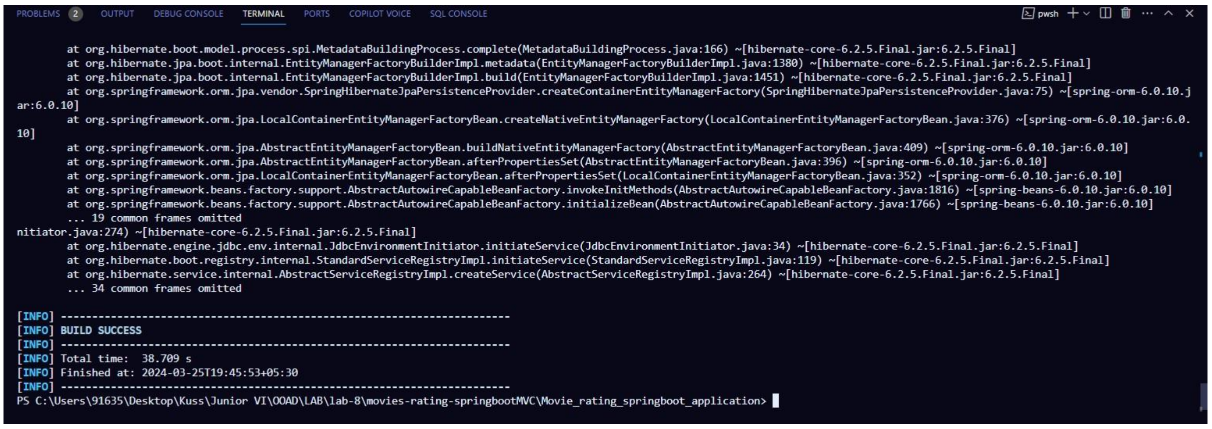
UserRating.java

```java
package com.movie.demo;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class UserRating {

    @Id
    @GeneratedValue(strategy =
GenerationType.IDENTITY)
    private Long id;
    private Long movieId;
    private Long userId;
    private double rating;

    public UserRating() {
    }

    public UserRating(Long movieId, Long userId,
double rating) {
        this.movieId = movieId;
        this.userId = userId;
        this.rating = rating;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public Long getMovieId() {
        return movieId;
    }

    public void setMovieId(Long movieId) {
        this.movieId = movieId;
    }

    public Long getUserId() {
        return userId;
    }
    public void setUserId(Long userId) {
        this.userId = userId;
    }

    public double getRating() {
        return rating;
    }

    public void setRating(double rating) {
        this.rating = rating;
    }
}
```

Movie.html

```html
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
    <title>Movies</title>
  </head>
  <body>
    <h1>Movie Catalog</h1>
    <table>
      <thead>
        <tr>
          <th>Title</th>
          <th>Release Year</th>
          <th>Genre</th>
          <th>Director</th>
          <th>Average Rating</th>
        </tr>
      </thead>
      <tbody>
        <tr th:each="movie : ${movies}">
          <td th:text="${movie.title}"></td>
          <td th:text="${movie.releaseYear}"></td>
          <td th:text="${movie.genre}"></td>
          <td th:text="${movie.director}"></td>
          <td th:text="${movie.averageRating}"></td>
        </tr>
      </tbody>
    </table>
  </body>
</html>
```

3. Screenshot of console with application running

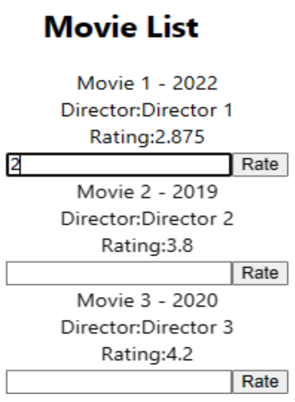


```
        at org.hibernate.boot.model.process.spi.MetadataBuildingProcess.complete(MetadataBuildingProcess.java:166) ~[hibernate-core-6.2.5.Final.jar:6.2.5.Final]
        at org.hibernate.jpa.boot.internal.EntityManagerFactoryBuilderImpl.metadata(EntityManagerFactoryBuilderImpl.java:1380) ~[hibernate-core-6.2.5.Final.jar:6.2.5.Final]
        at org.hibernate.jpa.boot.internal.EntityManagerFactoryBuilderImpl.build(EntityManagerFactoryBuilderImpl.java:1451) ~[hibernate-core-6.2.5.Final.jar:6.2.5.Final]
        at org.springframework.orm.jpa.vendor.SpringHibernateJpaPersistenceProvider.createContainerEntityManagerFactory(SpringHibernateJpaPersistenceProvider.java:75) ~[spring-orm-6.0.10.j
ar:6.0.10]
        at org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean.createNativeEntityManagerFactory(LocalContainerEntityManagerFactoryBean.java:376) ~[spring-orm-6.0.10.jar:6.0.
10]
        at org.springframework.orm.jpa.AbstractEntityManagerFactoryBean.buildNativeEntityManagerFactory(AbstractEntityManagerFactoryBean.java:409) ~[spring-orm-6.0.10.jar:6.0.10]
        at org.springframework.orm.jpa.AbstractEntityManagerFactoryBean.afterPropertiesSet(AbstractEntityManagerFactoryBean.java:396) ~[spring-orm-6.0.10.jar:6.0.10]
        at org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean.afterPropertiesSet(LocalContainerEntityManagerFactoryBean.java:352) ~[spring-orm-6.0.10.jar:6.0.10]
        at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.invokeInitMethods(AbstractAutowireCapableBeanFactory.java:1816) ~[spring-beans-6.0.10.jar:6.0.10]
        at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.initializeBean(AbstractAutowireCapableBeanFactory.java:1766) ~[spring-beans-6.0.10.jar:6.0.10]
        ... 19 common frames omitted
nitiator.java:274) ~[hibernate-core-6.2.5.Final.jar:6.2.5.Final]
        at org.hibernate.engine.jdbc.env.internal.JdbcEnvironmentInitiator.initiateService(JdbcEnvironmentInitiator.java:34) ~[hibernate-core-6.2.5.Final.jar:6.2.5.Final]
        at org.hibernate.boot.registry.internal.StandardServiceRegistryImpl.initiateService(StandardServiceRegistryImpl.java:119) ~[hibernate-core-6.2.5.Final.jar:6.2.5.Final]
        at org.hibernate.service.internal.AbstractServiceRegistryImpl.createService(AbstractServiceRegistryImpl.java:264) ~[hibernate-core-6.2.5.Final.jar:6.2.5.Final]
        ... 34 common frames omitted

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  38.709 s
[INFO] Finished at: 2024-03-25T19:45:53+05:30
[INFO] ------------------------------------------------------------------------
PS C:\Users\91635\Desktop\Kuss\Junior VI\OOAD\LAB\lab-8\movies-rating-springbootMVC\Movie_rating_springboot_application>
```

4. Screenshot of UIs related to the two scenarios with values, outputs, errors (if any)

Website

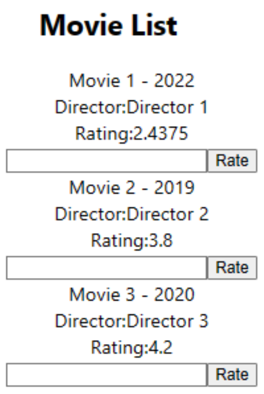
Before Rating



Movie List

Movie 1 - 2022
Director:Director 1
Rating:2.875
2 Rate
Movie 2 - 2019
Director:Director 2
Rating:3.8
Rate
Movie 3 - 2020
Director:Director 3
Rating:4.2
Rate

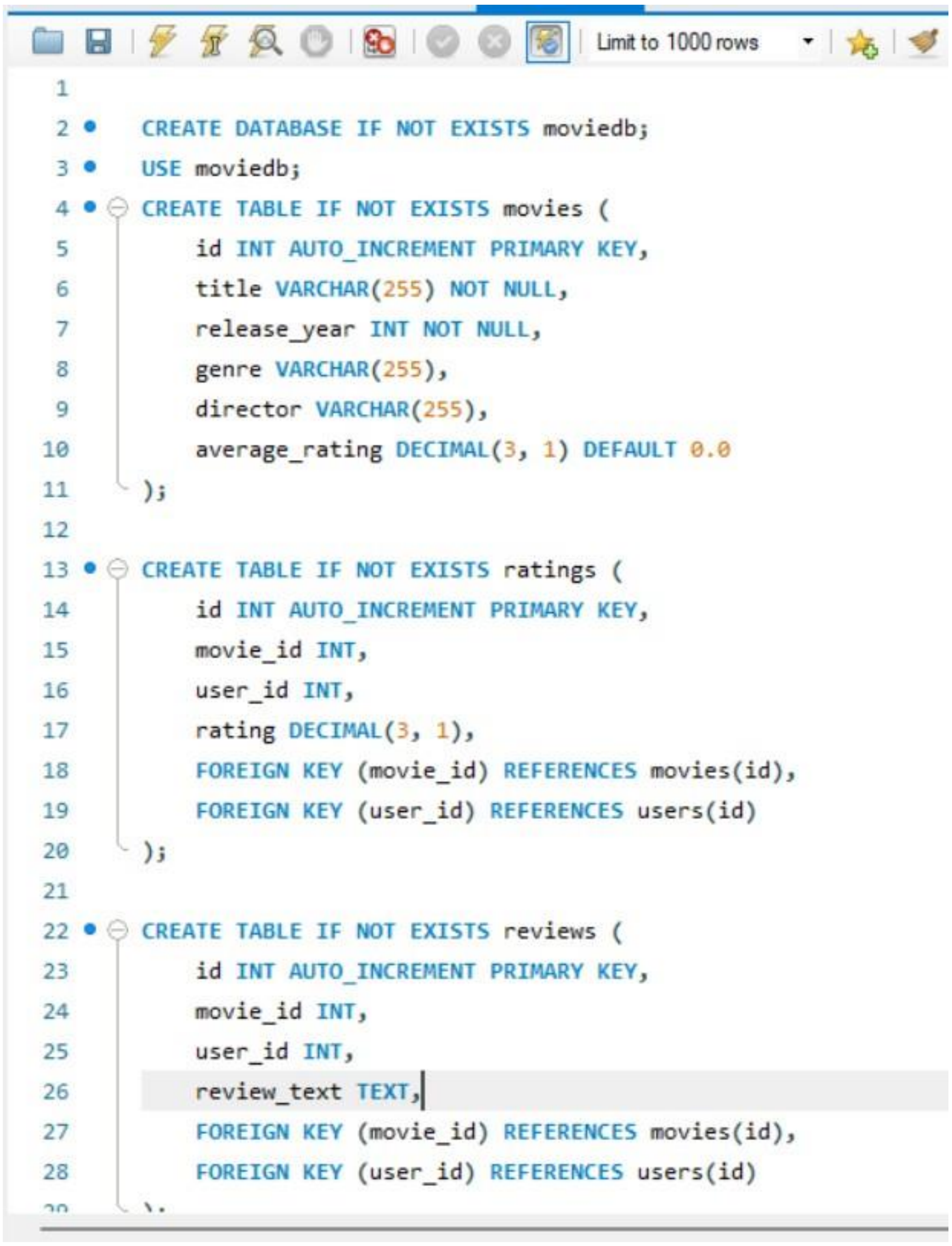
After Rating



Movie List

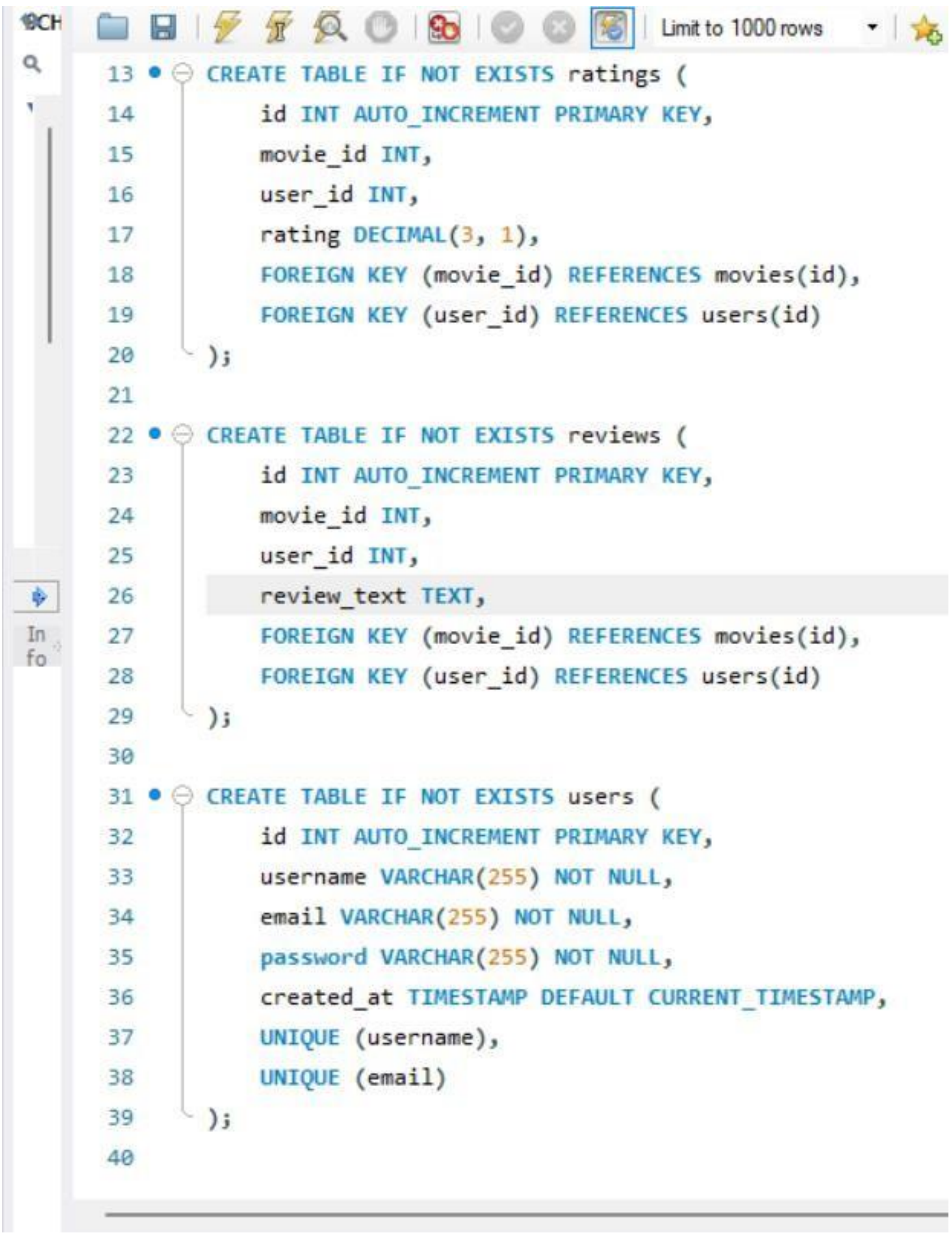
Movie 1 - 2022
Director:Director 1
Rating:2.4375
Rate
Movie 2 - 2019
Director:Director 2
Rating:3.8
Rate
Movie 3 - 2020
Director:Director 3
Rating:4.2
Rate

5. Screenshot of database with data items



Limit to 1000 rows

```sql

CREATE DATABASE IF NOT EXISTS moviedb;
USE moviedb;
CREATE TABLE IF NOT EXISTS movies (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    release_year INT NOT NULL,
    genre VARCHAR(255),
    director VARCHAR(255),
    average_rating DECIMAL(3, 1) DEFAULT 0.0
);

CREATE TABLE IF NOT EXISTS ratings (
    id INT AUTO_INCREMENT PRIMARY KEY,
    movie_id INT,
    user_id INT,
    rating DECIMAL(3, 1),
    FOREIGN KEY (movie_id) REFERENCES movies(id),
    FOREIGN KEY (user_id) REFERENCES users(id)
);

CREATE TABLE IF NOT EXISTS reviews (
    id INT AUTO_INCREMENT PRIMARY KEY,
    movie_id INT,
    user_id INT,
    review_text TEXT,
    FOREIGN KEY (movie_id) REFERENCES movies(id),
    FOREIGN KEY (user_id) REFERENCES users(id)
```



```sql
CREATE TABLE IF NOT EXISTS ratings (
    id INT AUTO_INCREMENT PRIMARY KEY,
    movie_id INT,
    user_id INT,
    rating DECIMAL(3, 1),
    FOREIGN KEY (movie_id) REFERENCES movies(id),
    FOREIGN KEY (user_id) REFERENCES users(id)
);

CREATE TABLE IF NOT EXISTS reviews (
    id INT AUTO_INCREMENT PRIMARY KEY,
    movie_id INT,
    user_id INT,
    review_text TEXT,
    FOREIGN KEY (movie_id) REFERENCES movies(id),
    FOREIGN KEY (user_id) REFERENCES users(id)
);

CREATE TABLE IF NOT EXISTS users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    UNIQUE (username),
    UNIQUE (email)
);
```