|  |  |  |  |
| --- | --- | --- | --- |
| Relation name | Relation Type | Constraints | Description |
| Search | 1:M | Only registered user/admin can search | User/ admin searchs for a particular movie |
| Sort | 1:M | Only registered user/admin can sort | User/admin can sort movies alphabetically |
| View | 1:M | Only registered user/admin can view reviews | User/admin views movies reviews |
| Rate | 1:M | Only registered user can submit a rating | User submits ratings for movies |
| Review | 1:M | Only registered user can submit a review | User submits reviews for movies |
| add | 1:M | Only registered admin can add movies | Admin adds new movies to the database |
| remove | 1:M | Only registered admin can remove movies | Admin removes movies from the database |
| order | 1:M | Only registered admin can order a report | Admin orders a report that contains information about the database |
| View | 1:M | Only registered admin can view system logs | Admin view system logs |
| Login | 1:1 | Only registered user/admin can login | User/admin login to be able to use the application |
| Register | 1:1 | Unregistered users can create accounts | Users can register to create an account |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute name** | **Data Type** | **Format** | **Constraints** | **Description** |
| adminID | INT | 32-bit integer | Unique, not null | Value of admin’s ID |
| Admin\_name | VARCHAR | String | Maximum 45 characters | Value of admin’s name |
| password | VARCHAR | String | 3 to 6 characters | Value of admin’s password |
| ID | INT | 32-bit integer | Unique, not null | Value of user’s ID |
| Name | VARCHAR | String | 3 to 12 characters | Value of user’s name |
| password | VARCHAR | String | 3 to 6 characters | Value of user’s password |
| MovieID | INT | 32-bit integer | Not null | Value of movie’s ID |
| UserID | INT | 32-bit integer | Not null | Value of user’s ID |
| Review | VARCHAR | String | Not null | Set of characters forming the review |
| TimeStamp | VARCHAR | String | Not null | Date and time of the log creation |
| Operation | VARCHAR | String | Maximum 10 characters | Description of the performed operation |
| LogMessage | VARCHAR | String | Maximum 200 characters | Message of contain useful information |
| Genre | VARCHAR | String | Not null | Category of the movie |
| Rating | DOUBLE | 64-bit double precision floating-point number | Not null | Value of the movie rating |
| Report | VARCHAR | String | Maximum 45 characters | Contents of the report |
| Type | VARCHAR | String | Maximum 45 characters | Type of the report |
| Owner | INT | 32-bit integer | 32-bit integer | Value of admin’s ID |
|  |  |  |  |  |
|  |  |  |  |  |

The main purpose of the project:

The movies recommendation system aim is to create a community of movies lovers, that will be able to view movies, write reviews for them, give them ratings and view reviews made by other people on specific movie, so that they can decide whether this movie is worth watching or not. We can do all that by using a Database management system (DBMS) particularly MySQL, in which we will save the users records so that every user can have his own account that he/she will use to browse movies and do other stuff, add movies/actors, save ratings and reviews of movies so that another users can view them and save admins records so that they can have their own account from which they will be able to add/ delete movies/actors, view system logs to be able to debug the system and keep track of what is going on and view reports that can summarize information from various records.

Note: “?” This sign present in the SQL statements acts as a place holder that can be assigned later to java variables such as variables containing usernames, passwords, IDs, movie names, etc.

* INSERT INTO users (id,name,password) VALUES (DEFAULT,?,?);
  + This statement is used in user registration which takes the username and the password entered by the user and store them in the users table.
* SELECT ID, Name, password FROM users;
  + This statement is used in user login which extracts the username and the password of the users registered in the database and check whether a pair of them matches those that was entered by the user in the login screen.
* SELECT adminID, admin\_name, password FROM admins;
  + This statement is used in admin login which extracts the admin name and the password of the admins registered in the database and check whether a pair of them matches those that was entered by the admin in the login screen.
* INSERT INTO movies (id,name,genre,rating) VALUES(DEFAULT,?,?,?);
  + This statement is used to add new movies into database
* SELECT ID FROM movies ORDER BY ID DESC LIMIT 1;
  + This statement is used to select the last id inserted in the database.
* INSERT INTO system\_logs (ID, UserID, AdminID, movieID, TimeStamp, Operation, LogMessage) Values(DEFAULT, ?, ?, ?, ?, ?, ?);
  + This statement is used to insert a log message for the added movie in the database.
* SELECT \* FROM movies;
  + This statement is used to select all the columns from the movies table in the database.
* DELETE FROM movies WHERE id = ?;
  + This statement is used to delete the movie that has id matching the id entered by the admin.
* INSERT INTO system\_logs (ID, UserID, AdminID, movieID, TimeStamp, Operation, LogMessage) Values(DEFAULT, ?, ?, ?, ?, ?, ?);
  + This statement inserts log for the deleted movie from the database.
* SELECT userreview.review, userrating.userrating, userrating.userid" + " FROM userreview INNER JOIN userrating ON userrating.UserID = userreview.UserID" + " WHERE userrating.MovieID = ? AND userreview.MovieID = ?;
  + This statement selects users reviews and ratings of specific movie id.
* SELECT count(\*) FROM userreview" + " INNER JOIN userrating ON userrating.UserID = userreview.UserID" + " WHERE userrating.MovieID = ? AND userreview.MovieID = ?;
  + This statement ????
* SELECT name FROM users WHERE id = ?;
  + This statement selects a username from the database whose id matches the one entered.
* SELECT COUNT(\*) FROM userreview WHERE movieid = ? AND userid = ?;
  + This statement
* UPDATE userrating SET userrating = ? WHERE movieid = ? AND userid = ?;
  + This statements updates the user rating for a movie
* INSERT INTO system\_logs (ID, UserID, AdminID, movieID, TimeStamp, Operation, LogMessage) Values(DEFAULT, ?, ?, ?, ?, ?, ?);
  + This statement insert the log for updating a movie rating by a user.
* UPDATE userreview SET review = ? WHERE movieid = ? AND userid = ?;
  + This statement updates the user review for a movie.
* INSERT INTO system\_logs (ID, UserID, AdminID, movieID, TimeStamp, Operation, LogMessage) Values(DEFAULT, ?, ?, ?, ?, ?, ?);
  + This statement insert the log for updating a movie review by a user.
* INSERT INTO userrating (id,movieid,userid,userrating) VALUES(DEFAULT,?,?,?)
  + This statement insert a rating by a user for a movie in the database.
* INSERT INTO system\_logs (ID, UserID, AdminID, movieID, TimeStamp, Operation, LogMessage) Values(DEFAULT, ?, ?, ?, ?, ?, ?);
  + This statement inserts the log for submitting rating for a movie in the database
* INSERT INTO userreview (id,movieid,userid,review) VALUES(DEFAULT,?,?,?)
  + This statement inserts a review by a user for a movie in the database.
* INSERT INTO system\_logs (ID, UserID, AdminID, movieID, TimeStamp, Operation, LogMessage) Values(DEFAULT, ?, ?, ?, ?, ?, ?);
  + This statement inserts a log for submitting review for a movie in the database.
* SELECT \* FROM system\_logs;
  + This statement selects every column in the system logs table in the database.