Movie Database Application

Project Overview

This project, developed as part of the **Advanced Programming** course, implements a comprehensive Movie Database Application that interacts with a MySQL database to manage and search through a collection of movies, actors, directors, genres, and ratings. The application uses Tkinter for the GUI interface and MySQL for the backend, providing a user-friendly experience for searching and displaying movie-related data. The core functionality includes querying by multiple criteria such as year, genre, actor, director, nationality, and rating. This project demonstrates the ability to integrate database management, GUI programming, and SQL querying in a real-world application.

Technologies Used

- **Programming Language**: Python
- **Database**: MySQL
- **GUI Library**: Tkinter
- **Modules**:
    - mysql.connector for connecting to MySQL
    - tkinter for building the GUI
    - ttk for advanced widgets in Tkinter
    - os for handling file paths and managing the environment
    - SQL for creating the database schema, inserting records, and performing queries

Project Structure

The project is structured into several core components:

- **Database Setup**: Includes creating tables for actors, directors, movies, genres, ratings, and reviewers.
- **SQL Query Execution**: Handles dynamic querying based on user input.
- **GUI Interface**: Displays search results in a table format, and provides dropdowns for selecting search categories.
- **Search Functionality**: Allows searching movies by different attributes such as actor, director, rating, genre, year, etc.

Features & Functionality

1. **Database Structure**:
    - The database consists of eight tables: actor, director, movie, movie_cast, genres, movie_genres, rating, and reviewer.

- The movie_cast and movie_genres tables link movies to actors and genres, respectively, ensuring that the database can scale to include many-to-many relationships.
- Foreign key relationships between tables ensure referential integrity and consistency in the database.

2. **GUI Interface**:
    - The user is presented with a search bar and a dropdown menu to select the search category (e.g., actor, year, genre, rating).
    - Upon entering a search term, clicking the Search button retrieves relevant data from the database and displays it in a structured table using Treeview.
    - The table shows columns like ID, Title, Year, Time, Language, Release Date, Country, and Additional Info (e.g., actor names, genre, etc.).
3. **SQL Queries**:
    - The application supports searching by various attributes, including:
        - **Year**: Retrieves movies released in a specific year.
        - **Genre**: Lists movies of a given genre.
        - **Actor**: Finds movies where a specific actor has played a role.
        - **Director**: Displays movies directed by a specific director.
        - **Country**: Filters movies by the country of release.
        - **Rating**: Allows searching for movies based on their ratings.
4. **Error Handling**:
    - Validation is performed on user input to ensure that appropriate data is entered (e.g., numeric input for age and rating).
    - If no results are found for a query, an information message is displayed informing the user.

Conclusion

This project successfully demonstrated how Python, MySQL, and Tkinter can be combined to create a functional application for managing and querying movie data. It provided hands-on experience with database management, SQL querying, and building a GUI for real-time interaction with the user. Moving forward, I plan to refine the project by adding features like pagination and advanced filters, as well as improving the user interface and performance.