Programming Assignment 1.1

Due: Saturday 1st March, 2025 at 23:59 on **Gradescope**.

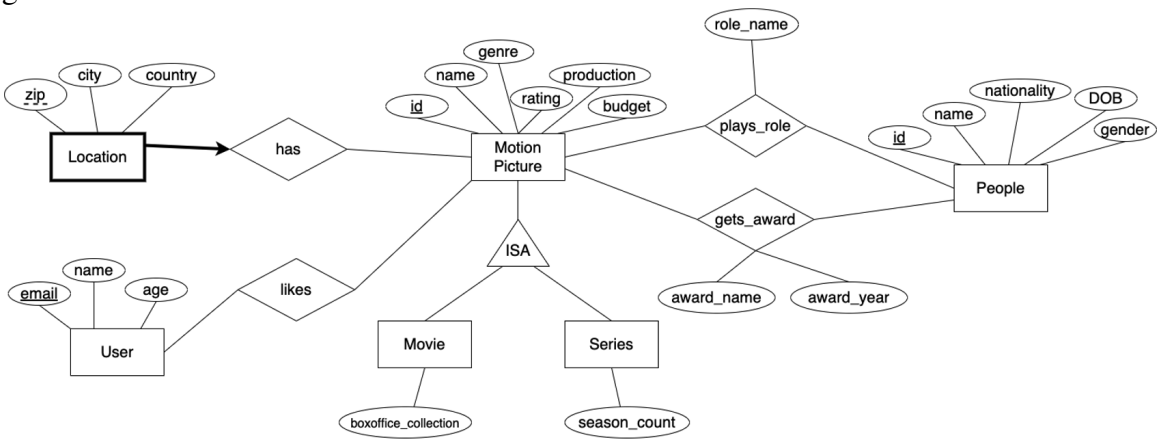
In Problem 3 of Written Assignment 1, you have designed an Entity-Relationship Diagram (ER Diagram) for an IMDb-like movie database. Project 1.1 is an extension to that problem where you will be working on all database-related stuff to build a complete functional website.

You have already submitted an ER Diagram for the application. The deliverable for this project involves **database schema creation**, **basic UI setup**, and simple **query implementations**. For this and the rest of the project, you will use the below given ER Diagram and schema description. For helpful links and setup, please look at the Section 5.

We recommend that you start as early as possible on this project.

1 Schema Description

An infotainment startup plans to create a website that will allow its users to view details of various motion pictures. Registered users will be able to view the list of all movies and search movies by name, genre, actor, etc. Besides users will be able to 'like' movies. The required ER Diagram for the database is as follows:



Below, we list the tables you would have in your database alongside their schema and primary and foreign keys.

MotionPicture (id, name, rating, production, budget)
User (email, name, age)
Likes (uemail, mpid)
Movie (mpid, boxoffice_collection)
Series (mpid, season_count)
People (id, name, nationality, dob, gender)
Role (mpid, pid, role_name)
Award (mpid, pid, award_name, award_year)
Genre (mpid, genre_name)
Location (mpid, zip, city, country)

Note: Primary keys are underlined and foreign keys are in blue.

2 Part 1: Building the Relational Schema

(a) Implement the schema in **MySQL (MariaDB)** using **`CREATE TABLE`** statements.

(b) Ensure correct **`PRIMARY KEY`** and **`FOREIGN KEY`** constraints with **`ON DELETE CASCADE`**.

(c) Add the following **`CHECK CONSTRAINT`**:

- The **rating** must be a floating-point number between **0 and 10**.
- The **budget** must be a **positive integer** (greater than 0).
- The **box office collection** must be a **non-negative float** (≥ 0).
- The **season count** must be at least **1**.
- Pay attention to participation constraints, especially with weak entities.

3 Part 2: Implementing Queries

A simple *Flask*-based UI has been built to connect the application's frontend with the backend database. Before executing queries, you need to configure database credentials in the `ini.env` file by providing values for `DB_USER` and `DB_PASSWORD`. This will enable the application to communicate with the database.

The UI provided comes with the support for the following functionalities.

(a) Viewing a **list of all movies** along with details such as name, rating, production, and budget.

(b) Viewing a **list of all actors** along with details such as name, nationality, date of birth, and gender.

(c) Allowing users to **like a movie** by selecting it and submitting their email address.

Your task in this part of the project is to implement SQL queries to retrieve and modify data in the database. Each query must be written inside the appropriate file located in the `routes` directory.

3.1 Task: Writing SQL Queries

The project includes a set of functions that interact with the database using raw SQL queries. Some functions already contain an empty query placeholder (`query = """ """`), which you need to replace with the correct SQL statements.

3.1.1 TODO 1: Retrieve All Actors

You will implement a query to fetch all actors from the database and display them in the UI.

- Open the file `actors.py` located in the `routes` directory.
- Locate the function `view_all_actors()`.

- Replace the placeholder `query = """ """` with a valid SQL statement to retrieve actor details from the `People` table.
- Save the file and restart the Flask application.
- Test your implementation by clicking the **"View All Actors"** button in the UI. Ensure that actor details are correctly retrieved and displayed.

3.1.2 TODO 2: Allow Users to Like a Movie

In this, you will modify the function for inserting a user's movie preference into the database.

- Open the file `movies.py` located in the `routes` directory.
- Locate the function `like_movie()`.
- Replace the placeholder `query = """ """` with a valid SQL statement to insert a new record into the `Likes` table. The query should:
  - Insert the selected `movie_id`.
  - Insert the `user_email` provided by the user.
- Save the file and restart the Flask application.
- Test your implementation by selecting a movie and clicking the **"Like a Movie"** button. Verify that the movie is successfully added to the user's liked list in the database.

These tasks will ensure that your application correctly interacts with the database using raw SQL queries. Make sure to test your queries thoroughly and debug any issues before proceeding to the next phase of the project.

4 Logistics

4.1 Collaboration

This is to be a group project with maximum size of 2.

4.2 Submitting Your Assignment

You must submit your assignment via Gradescope. Please ensure that all relevant project files are properly organized within a directory named `Project 1.1`. The submission must include all necessary Python and HTML files to allow instructors to run and evaluate your project.

For this assignment, you have two submission options:

1. **Git Repository Submission**:
   Upload your `Project 1.1` folder to a GitHub or Bitbucket repository. Gradescope allows you to submit a link to your repository. Please ensure that the repository is **public** or that appropriate access permissions are granted.

2. **Zipped File Submission**:
   Alternatively, you can create a zip archive of your `Project 1.1` folder and submit it on Gradescope. Before zipping, remove all irrelevant files (such as temporary files or IDE-specific folders like `.vscode/` or `.Docstore`).

The due date for this submission is Saturday 1st March, 2025 at 23:59. Late submissions may incur penalties as per course policies.

4.2.1 Project Folder Structure

Your project must follow the structure outlined below. Ensure that all required files are included to avoid submission issues.

- 'app/' - contains Flask routes and database logic
- 'templates/' - contains HTML templates for the UI
- 'static/' - contains CSS files
- 'data/' - contains 'data.sql' for initial database setup
- 'run.py' - main entry point for running the Flask application
- 'requirements.txt' - contains required Python dependencies

Before submission, ensure that:

- Your application runs without errors.
- The database is correctly initialized and populated.
- The required queries are implemented and functional.
- The project follows the expected directory structure.

Failure to follow these submission guidelines may result in a loss of points. If you encounter any issues, show up in TAs' office hours before the deadline.

5 Setup Instructions

Follow the steps below to set up the project on your local machine. For more detailed instructions, refer to the official project repository on GitHub:
https://github.com/SSD-Brandeis/YourOwnIMDb.

Step 1: **Install MariaDB**: Ensure MariaDB is installed on your system.

- **MacOS:** `brew install mariadb`
- **Ubuntu/Debian:** `sudo apt install mariadb-server`
- **Windows:** Download and install from https://mariadb.org/

Step 2: **Clone the Project Repository**: Clone the project from GitHub using the following command:

```
git clone https://github.com/SSD-Brandeis/YourOwnIMDb.git
```

Step 3: **Configure Database Credentials**: Open the `ini.env` file and update the credentials:

```
DB_USER=your_username
DB_PASSWORD=your_password
DB_HOST=localhost
DB_DATABASE=moviedb
```

Step 4: **Create the Database and Import Sample Data**: Open a terminal and execute the provided SQL script to initialize the database. You can also do copy paste from the data.sql file and insert your sample data.

```
mysql -u root -p < data/data.sql
```

Step 5: **Install Project Dependencies**: Navigate to the project directory and install the required dependencies:

```
pip install -r requirements.txt
```

Step 6: **Run the Flask Application**: Start the Flask web server:

```
python run.py
```

Step 7: **Access the Web Application**: Open your browser and go to:

```
http://127.0.0.1:5000/
```

Following these steps will set up the project correctly on your local system.