CS6401 Project: 20 marks = 20%

This is a group project. A project group must consist of either 4 or 5 students. Consider the relational database schema in **tv_series_schema.sql**.

Task 1. ERD

Draw an entity-relationship diagram (ERD) for the schema in **tv_series_schema.sql**. Save the ERD as a PNG file **tv_series_erd.png**.

Task 2. SQL

For the schema in **tv_series_schema.sql**:

1. Create view **top_series_cast(series_id, series_title, cast)**. This view should have a data row for each series with rating at least 4.00. Attribute **cast** is a comma-separated list of the names of all actors who play in at least one episode of this series. Avoid listing the same name multiple times in the cast of a series.

   Hint 1: Feel free to create more views and use them for constructing the two required views.

   Hint 2: Use an aggregate function not covered in class.

   Hint 3: Follow the examples in presentation **Lecture 06 (advanced examples)** published under content for week 4 on Brightspace.

2. Create view **actor_minutes(actor_id, actor_name, total_minutes_played)**. This view should have a data row for each **actor_id**. Attribute **total_minutes_played** is the total minutes played by all users of episodes featuring this actor (based on table **user_history**).

3. Write trigger **AdjustRating** that does the following. When a new data row is inserted into table **user_history** with **minutes_played = x**:
   a. Check if **x** is less than or equal to the length of the episode in table **episodes**. If **x** is greater than the length of the episode, change **x** to be equal to the length of the episode before it is inserted into table **user_history**.
   b. Add **0.0001*x** to the rating of the series (i.e., update attribute **rating** in table **series**) that the played episode belongs to. Do this only if the rating is less than 5.00, i.e., 5.00 is the maximum rating of a series.

4. Write stored procedure **AddEpisode(s_id, s_number, e_number, e_title, e_length)**. The procedure should check if the argument **s_id** is a **series_id** that already exists in table **series** and if this series does not already have an episode with number **e_number** for season **s_number**. If both conditions are satisfied, then the procedure should insert a new data row into table **episodes** with:

- episode_id auto generated
- series_id = s_id,
- season_number = s_number
- episode_number = e_number,
- episode_title = e_title,
- episode_length = e_length
- date_of_release = the current date

5. Write stored function **GetEpisodeList(s_id, s_number)** which returns a comma-separated list of the titles of all episodes of season **s_number** of series with **season_id = s_id**. The titles in the comma-separated list must be in ascending order by **episode_number**.

   Hint: Use an aggregate function not covered in class.

TEST

- While working on Task 2, insert some data into the database to test your code. You may use the provided **project_test_data.sql** file and the provided test sequence in the **Project Test.pdf** file.

IMPORTANT:

- Save all the code for creating the two views, the trigger, the stored procedure and the stored function in a text file named **tv_series_code.sql**. Please do not include code for inserting data into the database in the file **tv_series_code.sql**.
- The file **tv_series_code.sql** must contain only CREATE VIEW, CREATE TRIGGER, CREATE PROCEDURE and CREATE FUNCTION statements.
- If you have created other views as part of this task, make sure to include their code in **tv_series_code.sql** as well.
- The file **tv_series_code.sql** must be a simple text file that can be opened with Windows Notepad. Do not use MS Word!

Task 3. Cover Page

Create a PDF document called **tv_series_cp.pdf** which contains the list of names and student ID numbers of all group members.

Describe the platform you have used to test your code. For example, XAMPP on Windows 11. If you do not provide a description, then it will be assumed that your code must work with XAMPP on Windows 11.

Optionally, describe the contribution of each group member to the project. If you do not describe the contribution of each group member, it will be assumed that all group members contributed equally, and all will receive the same marks.

Submission

Submit the three files:

- **tv_series_erd.png**
- **tv_series_code.sql**
- **tv_series_cp.pdf**

as three attachments on Brightspace by 6 pm on Friday week 12.

Marking

1. Task 1: **4 marks**.
2. Task 2: **16 marks**
    - Task 2.1: 4 marks
    - Task 2.2: 3 marks
    - Task 2.3: 3 marks
    - Task 2.4: 3 marks
    - Task 2.5: 3 marks

Penalty of **-1 mark** for not providing a cover page with the list of group members and their ID numbers.

All group members are equally responsible for the quality of the project and will receive the same marks, unless the cover page suggests significant disbalance in contribution.

Late Submission Penalty

Late submissions will be accepted until the end of week 13.

Late submissions are subject to **-5 marks** penalty.