CS2102: Database System

Assignment 01: SQL Query AY2023/24 Sem 2: Week 07 (Wednesday, 6 March 2024)

Instructions

1. Please read the **Instructions** and **Preliminary** before continuing.
2. This is an **individual** assignment.
   - You must submit your own work.
   - You are **NOT** allowed to discuss with friends and/or using AI.
3. This assignment consists of **TEN (10)** half-mark SQL questions. The total is 5 marks.
   - Answer **ALL** questions.
4. The deadline for the submission is on Week 07 (Wednesday, 6 March 2024) at 12:00
5. **Late submission penalty:**
   - Two marks will be deducted for each late day *up to one late days*.
   - Submissions *after* the first late day will receive **zero** marks and will not be graded.
   - The solution will be released by **Friday, 8 March 2024** at 12:00 (Noon) to help revision before Midterm.
6. The assignment is to be submitted on Canvas.
7. The assignment will be **auto-graded**.
   - Additional *hidden* test cases may be added to ensure no hard-coding.
   - The SQL query will be run on PostgreSQL 16.
   - You **must** follow the requirement given in **Instructions** (Section **??**).

Good Luck!

Preliminary

In this assignment, you will formulate **10** SQL queries and submit your solutions into canvas assignment Assignment 01 (https://canvas.nus.edu.sg/courses/52825/assignments/105234). As the assignment will be auto-graded, it is very important that your submitted answers conform to the following requirements:

- Submit each question into its own individual answer box on Canvas.
- Each question has a specification on the output schema. You **MUST** follow the expected column name, otherwise there will be penalty even if you have the correct result. Consider the following template below using `AS` keyword for column renaming.

```
SELECT ... AS name, ... AS year
...
;
```

- Each answer must be a syntactically valid SQL query without any typographical errors: an SQL answer that is syntactically invalid or contains very minor typographical error will receive **ZERO (0)** marks even if the answer is semantically correct.
- For each question, your answer view must not contain any duplicate records. You may use *`DISTINCT`* as you see fit.
- Each question must be answered independently of other questions (*i.e.*, the answer for a question must not refer to any other view that is created for another question).
- You are **NOT** allowed to do any of the following for this assignment
  - Creating `VIEW`.
  - Using Common Table Expression (**CTE**).
  - Creating a new table, temporary or otherwise.
- Unless a value is explicitly specified in the question (*e.g.*, `'James Dean'`), you are not allowed to hardcode any value.
- You are **NOT** allowed to import any libraries and your answers must be executable on PostgreSQL.

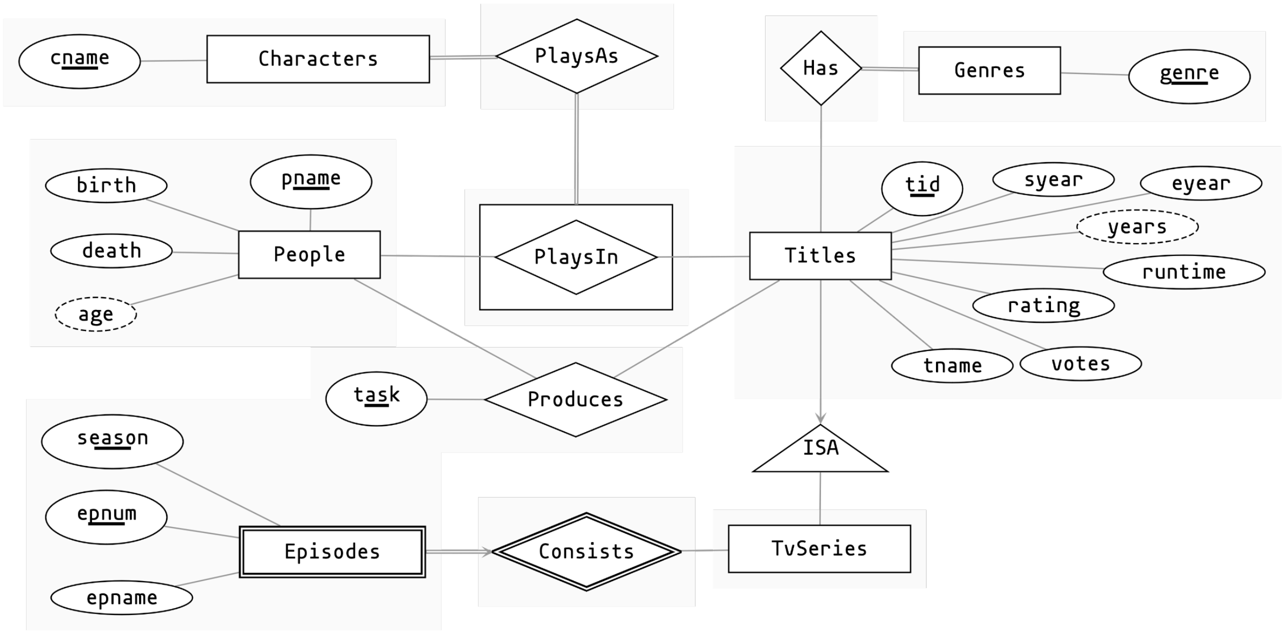
Database

The database is available on Canvas. The files are available in "Canvas > Files > Assignment 01" (https://canvas.nus.edu.sg/courses/52825/files/folder/Assignment%2001).

- Clean.sql: Contains the series of `DROP TABLE` statement to clean your database.
- DDL.sql: Contains the series of `CREATE TABLE` statement to (re-)initialize your database.
- Data.sql: Contains the series of `INSERT INTO` statement to (re-)initialize your tables.

For completeness, the approximate ER diagram for the schema is given below. You may assume that the data will be consistent with both the schema and the ER diagram. In particular, assume that total participation constraints are always enforced.

Do note that unless specified in the schema with NOT NULL, the attributes may contain NULL value. You should try to make your code work with NULL value whenever possible.



Additional Constraints

- We refer to entries in `Titles` as "film title" even when the title may also be a TV series (*i.e.*, an entry in `TvSeries`).
- If a film title is in `TvSeries`, we refer to them as "TV title", "Series title", "TV series title", or simply "TV series".
- If a film title is in `Titles` but **NOT** in `TvSeries`, we refer to them as "movie title" or simply "movies".
- We refer to "actor" as entries in `People` who are also entries in `PlaysIn`.
- The derived attribute `age` in `People` is derived purely from `death - birth` regardless of the date.
- If the person (*i.e.*, entries in `People`) is still alive, the value of `death` is NULL. Otherwise, the value of `death` is recorded and it must be greater than `birth` (*i.e.*, no person with age less than 1 year is recorded).

- The derived attribute years in Titles is derived purely from eyear - syear regardless of the date.
- If the film title (*i.e.*, entries in Title) is still ongoing, the value of eyear is NULL. Otherwise, the value of eyear is recorded.
- The runtime in Titles is specified in minutes.
- The rating in Titles must be between 0 and 10 (*inclusive of both*).
- For simplicity, the type of birth, death, syear (*i.e.*, start year), eyear (*i.e.*, end year) are *INT*.
- The people who produces film title (*i.e.*, entries in Produces) may be 'director', 'composer', or other task (*e.g.*, 'producer', *etc*).

Questions

1. We will start with a simple question first.

   Find all the names as well as the age of the people who are still alive. We define the age as of 31 December 2023 at 23:59. So if the person is born in the year 2000, the age is 23 regardless of the actual date and time of birth.

   The output schema should be

name	**age**

   You should see "Brigitte Bardot" with the age of 89.

2. Now we will have a slightly more complex conditions.

   For each movie title, find the movie name and the start year such that the start year from 1960 or above.

   The output schema should be

name	**age**

   There should at least be three movies from the year 1960 namely "La Dolce Vita", "Spartacus", and "The Truth". In total, there should be 26 rows in the output.

3. Let us now start joining tables.

   For each director name, find the film title that has been produced by the director with the genre of 'drama'. Recap that a director is the person that has a task of 'director'.

   The output schema should be

director	**title**

   There should be 7 rows in the output.

4. Please be careful with the following question as it involves dealing with `NULL` values.

   For the people who is not known to have played in a film title, find their name, birth year, and death year (if any, otherwise keep it as `NULL`).

   Note that these people may actually be director.

   The output schema should be

name	**birth**	**death**

   There should be 3 rows in the output.

5. And now we arrive at queries that may be solved via aggregation.

   For each person who plays in a film title, find their name, the total number of distinct films they have played in, the earliest starting year of the film they have played in, and the longest runtime for the film they have played in.

   Exclude people who have not played in any films at all.

   The output schema should be

name	**count**	**earliest**	**longest**

   There should be 17 rows in the output.

6. Recap that aggregation can also be filtered. Also remember that you should not hardcode any values unless the value is specified in the question.

   Find the name of the person as well as the number of films the person has played in where the person has played more than or equal number of films as `'Marlene Dietrich'` in her entire known career (*i.e.*, only based on the values in the database).

   The output schema should be

name	**count**

   There should be 5 rows in the output. Note that `'Marlene Dietrich'` has played a total of 3 films known in the database but your code should work for other known numbers too.

7. Here is a slight switch to something shorter, but requires some additional thinking.

> Find all the information about the people with at least 3 parts to their name. In other words, their name is at least of the form ”first_name middle_name last_name” and potentially more (*e.g.*, two middle names). Note that there should be only a single space between first_name, middle_name, and last_name.

The output schema should be

pname	**birth**	**death**

There should be only `'Olivia de Havilland'` in the current database instance.

8. For this question, the **order** of the rows matter.

> Find all the known episode names, season number, episode number, and episode name for TV series that has ended.
>
> Arrange the result in ascending order of TV title, followed by descending order of season number, and lastly in ascending order of episode number.

The output schema should be

title	**season**	**episode**	**name**

There should be 330 rows in the output. All from Schlitz Playhouse.

9. Here is another question where the order matters.

> For each genre, find the total number of film title with that genre. Display only genre with at least 5 film titles. Furthermore, display only the bottom 3 genres in terms of the number of film titles and display them in descending order of total number of film title. For genre with the same total number of film title, sort them in ascending order of genre.
>
> Note that the bottom 3 genre is
>
> - the bottom among the genre with at least 5 film titles, and
> - we must have 3 unique count of film titles regardless of the number of genres.

The output schema should be

genre	**count**

There should be 7 rows in the output.

For genres with at least 5 film titles, the bottom 3 counts are 6, 7, and 9 film titles. There is only one genre with 6 and 9 film titles but there are five genres with 7 film titles. This gives the total number of rows as 7.

To help you, the results should be the following:

genre	count
Film-Noir	9
Adventure	7
Biography	7
Mystery	7
Thriller	7
War	7
Action	6

10. James Dean is one of the most famous icon of western drama films. He is very focused on both drama and western genre. We want to find out if there are other actors with more varied genre than James Dean.

    Find the name of the person as well as the film genre of people who have played in a film genre that has been played by `'James Dean'`. Since James Dean has only played in drama and western (*in our current database*), we want to find the people who have played in both drama and western movie but possibly more genre.

    Please be reminded that your code should work in all instances even when James Dean has played in more genre or we have no data on James Dean at all.

    The output schema should be

name	genre

    There should be 23 rows in the output.

    Doris Day has played in 8 genres: Biography, Comedy, Drama, Music, Musical, Romance, Thriller, and Western.

    Gary Cooper has played in 8 genres: Adventure, Biography, Comedy, Drama, History, Romance, Thriller, and Western.

    Lauren Bacall has played in 7 genres: Comedy, Crime, Drama, Film-Noir, Mystery, Romance, and Western.