COMP2400/6240 - Relational Databases

Assignment 1 (SQL)

Due date: 23:55, 28th of March 2025

Instructions

- **This assignment must be done individually.**

  Do not share your solutions, or partial solutions, with anyone. Do not post any idea/-partial solution/result related to the assignment on the course discussion forum, or anywhere else where it can be read by other students.

  You may of course ask clarification questions, but make sure you phrase your questions so that they do not give away what you think the answer or solution is. If you have technical problems with accessing the `moviedb` database then you should ask for help. If you are not sure if a question is ok to post to the forum then ask your tutor during a lab or come to a drop-in session instead.

  Your answers to the assignment must be *your* answers. You can (and indeed should) use documentation, tutorials and other resources you can find on the web to learn SQL, and use what you learn to answer the assignment questions, but you should not obtain answers to the assignment questions from anywhere, anyone or anything. Your mark will be based on your original contribution to the answers only. Reference all resources (other than course material) that you have used in completing this assignment (by writing comments in your submitted SQL file).

- **You must submit your solutions through Wattle before the assignment deadline.** There will be an Assignment activity on the course Wattle page where you should upload your solution file. You can submit more than once, but we can only see, and will only mark, the last file that you submit.

  Late submissions will *not* be accepted. Submissions made outside the Assignment activity on Wattle (for example, files sent by email) will *not* be accepted under any circumstance. You will be marked on what you have submitted at the time of the deadline.

  If you find yourself in an unforeseeable situation beyond your control that you believe significantly affects your ability to complete the assessment on time, you should submit an ECA (`https://www.anu.edu.au/students/program-administration/assessments-exams/extenuating-circumstances-application`).

- This assignment will count for 18% of the final course mark.

- A copy of the `moviedb` database is available in the PostgreSQL DBMS in the CSIT lab environment (including via `partch`). You can connect to the `moviedb` database by entering the following in a Terminal:

  ```
  psql moviedb
  ```

- (*Optional*) If you wish to set up your own copy of the movie database, you can use the `pg_dump` utility on a lab computer (or `partch`) to create an SQL file that will create and fill all tables in the database. You can then execute that script in your own database.

- You must submit one file: `myqueries.sql` containing your solutions to all the questions. The file must be submitted through Wattle before the due date. You can download the template file from the folder "Assignment 1 (SQL)" on Wattle. You must enter your queries into the template file, and your submitted file `myqueries.sql` must be executable in the given database `moviedb`.

  ```
  moviedb=> \i myqueries.sql
  ```

  You can also test your queries against the `moviedb` database one by one following previous lab instructions.

  Please write each of your answers below the comment indicating the corresponding question (`-- Q1` through `-- Q10`), and leave these comments unchanged.

- The correctness of queries should not depend on the database state, i.e., the content of the tables in the database. The current content in `moviedb` is provided to help you to get familiar with the database schema. To be fully correct, your SQL queries should produce correct answers when tested against other databases that have *the same schema, but different content*. Partial marks may be given if a query only has minor issues, but *any query that is not runnable* (i.e., that has syntax errors or is not correct w.r.t. the database schema) *will receive zero marks*.
- Please write your SQL queries in a clear, well formatted style, and use SQL comments (lines starting with `--`) to explain your thinking behind the crafting of a complex query. We may inspect queries that are not fully correct to determine if the flaw is minor. If your queries are well documented and easy to read and understand, this increases the chances that we will be able to find it partially correct.
- Sample SQL questions and solutions on `moviedb` will be available on Wattle, which will be helpful for you to work on your assignment.
- **Plagiarism, collusion, and the use of disallowed tools will attract academic penalties in accordance with the ANU guidelines. Every student in this course is expected to be able to explain and defend any submitted assessment item. The course conveners can conduct or initiate an additional interview about any submitted assessment item for any student. If there is a significant discrepancy between the two forms of assessment, this will be seen as a case of potential academic misconduct.**

The Movie Database

The schema of the relational database `moviedb` is shown in Figure 1.

There are five different categories of awards: movie awards, crew awards, director awards, writer awards and actor awards. The `result` attribute in the relations listing each type of award indicates if the move/actor/director/writer/crew won the award, or was only nominated. Note that values of this attribue are not consistently capitalised (for example, both `'won'` and `'Won'` may appear). A movie that won an award was, of course, also nominated for that award, but the `result` attribute will only show that it won.

MOVIE(title, production_year, country, run_time, major_genre)
`primary key` : {title, production_year}

PERSON(id, first_name, last_name, year_born)
`primary key` : {id}

AWARD(award_name, institution, country)
`primary key` : {award_name}

RESTRICTION_CATEGORY(description, country)
`primary key` : {description, country}

DIRECTOR(id, title, production_year)
`primary key` : {title, production_year}
`foreign keys` : [title, production_year] ⊆ MOVIE[title, production_year]
[id] ⊆ PERSON[id]

WRITER(id, title, production_year, credits)
`primary key` : {id, title, production_year}
`foreign keys` : [title, production_year] ⊆ MOVIE[title, production_year]
[id] ⊆ PERSON[id]

CREW(id, title, production_year, contribution)
`primary key` : {id, title, production_year}
`foreign keys` : [title, production_year] ⊆ MOVIE[title, production_year]
[id] ⊆ PERSON[id]

SCENE(title, production_year, scene_no, description)
`primary key` : {title, production_year, scene_no}
`foreign keys` : [title, production_year] ⊆ MOVIE[title, production_year]

ROLE(id, title, production_year, description, credits)
`primary key` : {title, production_year, description}
`foreign keys` : [title, production_year] ⊆ MOVIE[title, production_year]
[id] ⊆ PERSON[id]

RESTRICTION(title, production_year, description, country)
`primary key` : {title, production_year, description, country}
`foreign keys` : [title, production_year] ⊆ MOVIE[title, production_year]
[description, country] ⊆ RESTRICTION_CATEGORY[description, country]

APPEARANCE(title, production_year, description, scene_no)
`primary key` : {title, production_year, description, scene_no}
`foreign keys` : [title, production_year, scene_no] ⊆ SCENE[title, production_year, scene_no]
[title, production_year, description] ⊆ ROLE[title, production_year, description]

MOVIE_AWARD(title, production_year, award_name, year_of_award, category, result)
`primary key` : {title, production_year, award_name, year_of_award, category}
`foreign keys` : [title, production_year] ⊆ MOVIE[title, production_year]
[award_name] ⊆ AWARD[award_name]

CREW_AWARD(id, title, production_year, award_name, year_of_award, category, result)
`primary key` : {id, title, production_year, award_name, year_of_award, category}
`foreign keys` : [id, title, production_year] ⊆ CREW[id, title, production_year]
[award_name] ⊆ AWARD[award_name]

DIRECTOR_AWARD(title, production_year, award_name, year_of_award, category, result)
`primary key` : {title, production_year, award_name, year_of_award, category}
`foreign keys` : [title, production_year] ⊆ DIRECTOR[title, production_year]
[award_name] ⊆ AWARD[award_name]

WRITER_AWARD(id, title, production_year, award_name, year_of_award, category, result)
`primary key` : {id, title, production_year, award_name, year_of_award, category}
`foreign keys` : [id, title, production_year] ⊆ WRITER[id, title, production_year]
[award_name] ⊆ AWARD[award_name]

ACTOR_AWARD(title, production_year, description, award_name, year_of_award, category, result)
`primary key` : {title, production_year, description, award_name, year_of_award, category}
`foreign keys` : [title, production_year, description] ⊆ ROLE[title, production_year, description]
[award_name] ⊆ AWARD[award_name]

Figure 1: The `moviedb` schema.

Questions

Your task is to write SQL queries that answer the following questions. For each question, your answer must be a *single SQL query* (that may contain subqueries). Remember that your answers should not depend on the example database state.

You must write all your answers into the template file `myqueries.sql`.

1. List all countries in which one or more movies in the database have been produced, in alphabetical order and without repetitions. The query result should have a single column.
2. List each country and how many movies were produced there, sorted by movie count descending.
3. List all actors who have played more than one role in the same movie. (For example, in "Psycho", 1960, Anthony Perkins plays both "Norman Bates" and "Mother".) The query result should have four columns: The actor's first name, last name, and the title and production year of the movie.
4. List the award names (in the `award` table) that have *not* been given to any actor (i.e., that do not appear in the `actor_award` table). The query result should have a single column, with the award names in alphabetical order and without repetitions.
5. What is the maximum number of roles for any movie listed in the database? Note: different roles played by the same actor in a movie still count as different roles. The query result should be a single row with a single column, containing the maximum number.
6. List the first and last name of all persons who have won a director award but never won a writer award. The query result should have two columns, with the person's first and last name, and no duplicate rows.
7. List the title and production year of all movies that have won either a movie award or a director's award, along with the director's first and last name. That is, the query result should have four columns: title, production year, director's first name and director's last name, and no duplicate rows.
8. List all actors (first name and last name) who have played a role in a movie that has won less than 2 movie awards (including none), together with the title and production year of the movie. The query result should have four columns: The actor's first and last names, and the movie's title and production year.
9. List all actors who have had roles in the largest number of different movie genres (as given by the attribute `major_genre` of the `movie` relation). (For example, according to our example database, Meg Ryan has had a role in two different genres of movie: one drama, and three comedies.) The query result should have two columns, with the actors' first and last name.

All done? Well done! Before you submit:

- Check that your `myqueries.sql` file is *syntatically correct SQL*, by running it in `psql`:

```
moviedb=> \i myqueries.sql
```

- Check that your solution (query) for each question is written below the comment line indicating the corresponding question ("`-- Qi`") in your `myqueries.sql` file.