Databases 2022 - Assignment 2

Deadline

Strict deadline **30 November 2022 at 23:59 CET**. The deadline is definitive and it already accounts for vacation days. No submission after the deadline will be considered.

Consider the following relational database schema (A script to create the schema is provided at the end)

Movies (title:string, year:int, director:string, budget:int, gross:int)
(director) is a Foreign Key to Directors
Directors(director:string, yearOfBirth:int)
DirectorAwards(director:string, year:int, award:string, result:string)
(director) is a Foreign Key to Directors
MovieAwards(title:string, year:int. award:string, result:string)
(title, year) is a Foreign Key to Movies

The underlying attributes indicate keys. All the attributes are considered to be NOT NULL. The attribute `result` can take only two values: 'won' or 'nominated'. The values of the `award` attribute in the DirectorAwards is the name of the event, e.g., 'Oscar' or 'Cannes'. The values of the `award` attribute in the MovieAwards consists of two parts, the event and the award type separated by coma. For example: 'Oscar, best film', 'Golden Globe, screenplay', 'Oscar, best director', etc.
All the awards recorded in DirectorAwards are of type best director. For example ‘Oscar’ in DirectorAwards is equivalent to 'Oscar, best director' in MovieAwards. It is possible that a director award appears also in the movie award. For example, a director award from Oscal will appear in DirectorAwards (as simply ‘Oscar’) and in MovieAwards as 'Oscar, best director'.

Write the following queries in SQL.

1. For each director that is older than 50, find the minimum, maximum, and average profit of his/her movies. (Output -1 for minimum, maximum, and average of the directors with no movies)
2. Find the latest movie(s) that won at least 3 awards.
3. Find the most profitable movie(s), and the least expensive movie(s). (use a column named “feature” as the first output column accepts values 'most profitable' and 'least expensive' respectively to distinguish mentioned cases) (note that profit means the revenue that remains after expenses)
4. Find awards received by directors who made a high-grossing movie in the past 5 years. (high-grossing movie is a movie that have made over 1 million gross) (Output award, year, and director)
5. For each movie, find its award "success-rate", that is, the probability of getting an award it was nominated for. (Output -1 for movies never even nominated) (success-rate is in a decimal representation in the last output column)
6. Find the movie(s) that won the largest number of Oscars.

7. Find the youngest and the oldest directors who won an Oscar for "best director". (recall that it can be recorded in MovieAwards or DirectorAwards) (Output director, and then feature accepts values 'youngest' and 'oldest') (youngest/oldest refers to their current age at the time of executing the query, not the age at the time of receiving the awards)
8. Find the percentage of movies that won an Oscar among all movies made in the 80s. (Output -1 if no movies were made in the 80s.)
9. Find directors who won the 'best director' award, although the movie for which they won the award was not profitable.
10. Find directors who won a 'best director' every year when a Spielberg movie won at least three awards. ("Spielberg" is the exact name of director not part of it)

Instructions

- The assignment must be done individually.
- Implement each query in its own **plain text file**. The file name must be **`query_<number>.sql`** . For example: `query_8.sql`. Every query in the file should be finished with the ";". The file should not contain anything else apart from the query.
- Each query will be evaluated by an automatic script that will compare its results with the ones from the reference implementation. Each query will then be either correct (+1 point) or wrong (+0 points). Make sure that the name is EXACTLY as described above. The marking is done by a script automatically and if the file has a different name (e.g. missing the _ or the .sql or uses uppercase letters instead of lowercase, it will be marked as non-working (0 points).
- The **attributes** in the results must be **ordered as they appear in the text**. Example: "return the beer name and its manufacturer" should be matched with a "`SELECT name, manf FROM ...`" Failure to properly order the columns leads to (0 points).
- When asked to return information about a relation, select the attributes that identifies it, *i.e.* the primary key. Example: Return all the Directors that …, would be: "`SELECT director FROM Directors`".
- Non-integer numbers must be rounded at the second decimal. Example 0.1234 → 0.12.
- The outputs that are numbers without fractional part do not need zero in their fractional part. For example output 11 and do not output 11.00.
- Unless otherwise specified, results must be **distinct**, meaning always use the distinct unless explicitly asked not to do it.
- Do not create additional tables or persistent views. You can use TEMP views if you think it is really necessary
- Do not modify in any way the existing schema.
- The queries will be tested against a PostgreSQL database provided by the university. You can install one on your computer if you want to work locally but always keep in mind that the queries have to work to the univ database. You have already been provided with the number (name, username and passwd) of the database that has been reserved for you.

- Test your queries before submitting! You need to make sure that your queries return the correct results always. You need to think if there are any special cases. If you would like to get some real data values to test your queries, if you want you can use the IMDB data.

Delivery

- Create a .zip archive with, and only with, the .sql files and upload it via this form: https://forms.gle/hkhbWZ9ATNBkfskF9
- The name of the archive must be **`SQL_<mat>.zip`**, where mat is your matricole. Example: `SQL_12345.zip`;
  Example command: `zip SQL_12345.zip query_*.sql`
- **The archive must not contain anything else (no subfolders, no files).** Extraction and evaluation are completely automated. Malformed archives, misspelled names, and malformed files will result in a total evaluation of 0 points. **CHECK your zip file that when it is extracted it does not create any subfolders but only the .sql files. Sometimes, windows creates a folder with all the files. This is wrong. When you unzip the zip, all the .sql files should become available in that folder and not in any subfolder.**

Script to create the database schema:

```
CREATE TYPE AwardResult AS ENUM
   ('won', 'nominated');
CREATE TABLE Directors
(
   director varchar(50) NOT NULL,
   yearOfBirth integer NOT NULL,
   CONSTRAINT "Directors_pkey" PRIMARY KEY (director)
);
CREATE TABLE Movies
(
   title varchar(100) NOT NULL,
   year integer NOT NULL,
   director varchar(50) NOT NULL,
   budget integer NOT NULL,
   gross integer NOT NULL,
   CONSTRAINT "Movies_pkey" PRIMARY KEY (title, year),
         CONSTRAINT "DirectorAwards_director_fkey" FOREIGN KEY (director)
      REFERENCES Directors(director) MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE NO ACTION
);
CREATE TABLE DirectorAwards
(
   director varchar(50) NOT NULL,
   year integer NOT NULL,
   award varchar(100) NOT NULL,
   result AwardResult NOT NULL,
         CONSTRAINT "DirectorAwards_pkey" PRIMARY KEY (director, year, award),
```

```
    CONSTRAINT "DirectorAwards_director_fkey" FOREIGN KEY (director)
        REFERENCES Directors(director) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
);
CREATE TABLE MovieAwards
(
    title varchar(100) NOT NULL,
    year integer NOT NULL,
    award varchar(100) NOT NULL,
    result AwardResult NOT NULL,
            CONSTRAINT "MovieAwards_pkey" PRIMARY KEY (title, year, award),
    CONSTRAINT "MovieAwards_year_title_fkey" FOREIGN KEY (title, year)
        REFERENCES Movies(title, year) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
);
```