**COS 598: Introduction to Data Science**

**Fall 2021**

**Homework Assignment 2**

Assigned: October 7, 2021

Due: October 28, 2021

*Submission instructions*:

Task 0: There is nothing to submit.

Task 1: You must submit a writeup with your entity/relationship diagram and database schema. Briefly describe how you translate the entity/relationship diagram to the database schema.

Task 2: There is nothing to submit for part (a). You must submit your Python code for part (b) in a .py file.

Task 3: You must submit a text file named sql.txt containing all your SQL queries. Additionally, you must submit a text file named query\_results.txt containing the results of your SQL queries.

Task 4: You must submit your Python code in a .py file.

The file(s) must be uploaded to the Brightspace submission site for Homework 2.

Total: 100 points.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Task 0. PostgreSQL installation (0 point).

* Download PostgreSQL from <https://www.postgresql.org/download/>.
* Install the appropriate version of PostgreSQL using the EDB interactive installer.

Task 1. Database design using the entity/relationship diagram (10 points).

Suppose that you are building a movies database. You have the following information that you want to store in the database:

* information pertaining to each movie: movie id, name, year, and rank
* information pertaining to each person (actor): person id, first name, last name, gender
* information pertaining to each director: director id, first name, last name

You also know what persons (actors) act in what movies *in what roles*, and what directors direct what movies.

1. Draw an entity/relationship diagram to capture the information pertaining to movies, persons (actors), and directors, and the relationships among them. In particular, you should capture the idea that a person acts in a movie *in a particular role*, and the idea that a director directs a movie. Assume that a person may act in multiple movies; a movie may have multiple actors; a director may direct multiple movies; a movie may have multiple directors.
2. Translate the entity/relationship diagram you have drawn into a database schema.

Task 2. Data import (40 points).

On the Brightspace submission site for Homework 2, you will find a zip file named IMDB.zip. Download and decompress this file into a directory. Inside the directory, you will find 5 text (.txt) files:

* IMDBCast.txt: describing what persons (identified by pid’s) act in what movies (identified by mid’s) in what roles.
* IMDBDirectors.txt: describing information pertaining to directors: id, fname (first name), lname (last name).
* IMDBMovie\_Directors.txt: describing what directors (identified by did’s) direct what movies (identified by mid’s).
* IMDBMovie.txt: describing information pertaining to movies: id, name, year, rank.
* IMDBPerson.txt: describing information pertaining to persons (actors): id, fname (first name), lname (last name), gender.

1. Create a database named moviesdb in the PostgreSQL database system. You can do this on the PostgreSQL shell. There is nothing to submit for this part.
2. Write a Python program to create five tables named Movie, Person, ActsIn, Director, Directs in the moviesdb database. In particular:

* The Movie table should contain all information in IMDBMovie.txt, and should have columns: id, name, year, rank.
* The Person table should contain all information in IMDBPerson.txt, and should have columns: id, fname, lname, gender.
* The ActsIn table should contain all information in IMDBCast.txt, and should have columns: pid, mid, role.
* The Director table should contain all information in IMDBDirectors.txt, and should have columns: id, fname, lname.
* The Directs table should contain all information in IMDBMovie\_Directors.txt, and should have columns: did, mid.

You should declare appropriate **primary key** and **foreign key** constraints when constructing the tables.

Note: if you encounter an error while reading a text (.txt) file (e.g., unrecognized byte at …), it is because the text file is not encoded using the default utf-8 encoding. To fix this problem, when you call the open function in Python to open such a text file, in addition to the file path and mode of opening ('r' for reading), you should also specify encoding='latin-1'. For example, you can call the open function in the following way:

open(<file path>, 'r', encoding='latin-1')

if you encounter an error reading the file without specifying encoding.

Task 3. SQL queries (30 points).

For this task, you can execute your SQL queries in a PostgreSQL shell. You need to submit a text file named sql.txt containing all your SQL queries. Additionally, you need to submit a text file named query\_results.txt containing the results of your SQL queries. **Limit the result of each query to at most 10 rows.**

1. List (10 of) the persons (id, fname, lname, gender) who acted in at least one movie in the second half of the 19th century *and* in the first half of the 20th century.
2. List (10 of) the directors (id, fname, lname) who directed a film in a leap year.
3. List the top 10 (in terms of rank) movies (id, name, year, rank) that have the same year as the movie 'Shrek (2001)', but a better rank. (Note: bigger value of rank implies a better rank.)
4. List the top 10 directors in descending order of the number of films they directed.
5. Find the movie(s) with the largest number of actors (persons). Also, find the movie(s) with the smallest number of actors (persons). In both cases, also return the number of actors.
6. Find (10 of) the actors (persons) who acted in movies by at least 10 distinct directors (i.e. actors who worked with at least 10 distinct directors).

Task 4. SQL queries in a Python program (20 points).

1. Implement a Python function find\_best\_movies\_in\_years that finds the best *k* movies in years from start\_year to end\_year, in descending order of rank. Additionally, your function should save the best *k* movies in those years in a .csv file with delimiters semicolons (;) instead of commas.
2. Use your function to find the top 20 movies in years from 1995 to 2004. Submit the .csv file produced by the function call.