Lab Assignment 1

(due Friday April 17, 2020 before midnight)

1 Introduction

Consider a database that is used by your favorite streaming service to access information related to movies. This database keeps information about movies, famous actors, character roles played by these actors, customer reviews of the movies, and IMDb information associated with each movie.

2 The Schemas

This information is held in five separate relations conforming to the following schemas:

Movies (movieid, title, genre, year, director)
Actors (actorid, name, gender, birthyear)
MovieRoles (actorid, movieid, role)
Reviews (customerid, movieid, rating, review_text)
Imdb (movieid, imdb_rating, imdbid)

The `movieid` attribute is the primary key of the Movies relation, and is referenced as a foreign key in the Reviews, MovieRoles, and Imdb relations (see the last section of this document for the definitions of a primary key and a foreign key). `actorid` is the primary key of the Actors relation, and is referenced as a foreign key in the MovieRoles relation. The primary key of the MovieRoles relation consists of `actorid` and `movieid`. The primary key of the Reviews relation consists of `customerid` and the `movieid`.

In the Movies relation, attribute `title` stores the name of the movie, `genre` stores a list of the associated genres, and `year` stores the four digit release year of the movie. `director` stores the name of the person who directed the movie.

The Actors relation stores the `name` of the actor, the `gender` of the actor, specified with the values 'Male', 'Female', or 'Non-binary' and the four digit `birthyear` of the actor.

The MovieRoles relation stores the name of the `role` portrayed by a given actor in a given movie. Here, we assume an actor can only play one role in each movie.

In the Reviews relation, the attribute *rating* for a movie is a float in the range 1 to 10 (10 means perfect). Customers have the opportunity to provide a detailed review up to 2,000 characters long, which is stored in the `review_text` attribute. Here, we assume each customer can only review a given movie once.

Finally, the Imdb relation stores the aggregated `imdb_rating` for each movie, which is also a float in the range 1 to 10. Each `movieid` in our database has a 1-1 mapping with an `imdbid`, retrieved from the IMDb website[1].

[1]If you're curious to know what the real IMDb id is for your favorite movie, you can see it right in the url of the associated webpage, e.g., https://www.imdb.com/title/**tt6320628**/.

3 Submission

In this assignment, you are asked to do the following:

1. Create the above database schema in the PostgreSQL server by issuing the proper data definition commands. Use appropriate data types for the attributes (e.g., integer, string, char and so on). Primary and foreign key constraints need to be defined.
2. Populate the relations of the schema with tuples using the INSERT INTO command. The detailed dataset will be given as CSV files (see the data.zip on piazza); all the data should be inserted. Note, you can easily convert a CSV file into an Excel or Google Sheets file if you want to more easily visualize the data.
3. Save the commands issued in a file named Lab1-schema.txt. Upload this file to the Lab Assignment 1 submission page on Canvas by 11:59pm on Friday April 17th.

One way to create the Lab1-schema.txt is to cut-and-paste your terminal output and then save it on a file. A more convenient way is to record your entire session by using the Unix command:

```
script Lab1-schema.txt
```

After the above command is issued, the rest of your session will be recorded at Lab1-schema.txt file. If there is already a Lab1-schema.txt file at your working folder, it will be overwritten. If you want to append the session to an already existing file you should use the -a option as follows:

```
script -a Lab1-schema.txt
```

For more information about the command, check out its man page by typing: `man script`

4 Definitions

Primary Key: As discussed in the first class meeting, a *key* of a relation schema **R** is a set K consisting of attributes of R such that for every relation R that is an instance of **R**, there are no two distinct tuples s and t in R that have identical coordinates on the attributes in K. Moreover, no proper subset of K has this property.

In general, a relation schema **R** may have more than one keys. In that case, one of the keys of **R** is designated as the *primary key* of **R**.

Foreign Keys: In SQL an attribute may be declared to be a *foreign* key referencing some attribute of another relation to ascertain that the value of the attribute makes sense. The referenced attribute in the other relation must be declared UNIQUE or the PRIMARY KEY of the relation. Any value of the attribute declared as a foreign key should also appear in the referenced relation. For instance, using foreign key reference in the MovieRoles relation we make sure that a role is always associated with an existing movie and that the actor is also in the database. More details about the foreign key constraint declarations can be found in section 7.1.1 of the textbook.