Problem Set 1: SQL

Assigned: 9/11/2017

Due: 9/18/2017 11:59 PM

Submit to the 6.814/6.830 Gradescope (`https://gradescope.com/`)

You may work in pairs on this problem set. Clearly indicate the name of your partner. Only one of you needs to submit on Gradescope.

1 Introduction

The purpose of this assignment is to give you hands-on experience with the SQL programming language. SQL is a declarative language in which you specify the data you want in terms of its properties. This assignment focuses on the `SELECT` subset of SQL, which is all about *querying* data rather than modifying it.

We will be using SQLite, which provides a standards-compliant SQL implementation. In reality, there are slight variations between the SQL dialects of different vendors (PostgreSQL, MySQL, SQLite, Oracle, Microsoft, etc.) —especially with respect to built-in functions. The SQL tutorial at `http://sqlzoo.net/`, provides a good introduction to the basic features of SQL. After following this tutorial you should be able to answer most of the problems in this problem set.

To install SQLite, you can simply use the command `apt install sqlite` on Debian-based Linux distributions like Ubuntu, or `brew install sqlite` on Mac. Downloads for the pre-compiled binaries can be found at `https://sqlite.org/download.html` for Windows (as well as Linux and Mac, if you'd prefer to install with the binaries). If you use a pre-compiled binary, you might have to make sure that the path to your installed directory is in the PATH environment variable.

The SQLite SELECT documentation at `https://sqlite.org/lang_select.html` will be helpful to you, and you can access all the other SQLite documentation on that site as well. You may also wish to refer to Chapter 5 of "Database Management Systems."

To access the SQLite shell for the database, download the provided SQLite database file, `cd` to the directory containing the file and run:

```
sqlite3 imdb.db
```

If the name of your downloaded `.db` file is not `imdb.db`, then replace `imdb.db` with whatever the name of your file is.

2 Dataset

The data for this assignment is a modified subset of the IMDb dataset. We've taken the original dataset and only kept the movies and any people related to the movies, and we've modified some of the names and types of some of the original attributes to make things simpler for you. Therefore, everything you need to understand the dataset is contained in your SQLite database and this pset. However, if you enjoyed this pset so much that you want to download the whole IMDb dataset, you can find details here: `http://www.imdb.com/interfaces`

The database tables include:

`movies`: contains the id, title, release year, and runtime in minutes of each movie

`ratings`: contains the movie id (which maps to an id from `movies`), average rating, and number of votes for each movie.

`people`: contains the id, name, birth year, and death year of each person

`directors`: contains a mapping of movies to directors who directed those movies. The `movie_id` corresponds to an id from movies, and the `person_id` corresponds to an id from people. In `directors` there is an n:n relationship between the movie ids and the person ids.

`cast_members`: contains a mapping of movies to cast members of those movies. The `movie_id` corresponds to an id from movies, and the `person_id` corresponds to an id from people. In `cast_members` there is an n:n relationship between the movie ids and the person ids.

3 Using the Database

Once in the SQLite shell, there are two kinds of commands useful to a database user. The first kind are the client meta-commands. The most important one, of course, is `.help`, which gives you help on meta-commands. There are two others that greatly help:

We can list all the table schemas in the database with `.tables`:

```
sqlite> .tables
cast_members  directors     movies         people         ratings
```

And we can check the schema (recall, that the "schema" of a database is like a class definition in an object oriented language) of a given table with `.schema` *table_name*:

```
sqlite> .schema movies
CREATE TABLE movies(
            id TEXT PRIMARY KEY,
            title TEXT,
            year INT,
            runtime INT
        );
```

The second class of useful commands are SQL commands. All SQL queries in SQLite must be terminated with a semi-colon. For example, to get a list of all records in the `page` table, you would type:

```
SELECT * FROM movies LIMIT 10;
```

This query requests a max 10 rows from the table. Using `LIMIT` in this manner one can explore the data small bits at a time. If you really wanted to produce all the records, though, the query is:

```
SELECT * FROM movies;
```

You can use `Ctrl+C` to end a query that is taking too long (it is very possible to write such bad queries even unintentionally). Note that using the `LIMIT` keyword, when used by itself, offers no guarantee on which 10 rows from the result are returned, so do not assume an ordering.

You can change the way the SQLite shell displays the result sets to suit you better. In particular, you may find the commands `.header on` and `.mode column` useful.

Finally, to exit the SQLite shell, you can use `.exit`

4 Questions

For each question, please include both the **SQL query** and the **result** in your answer. If the result is more than 10 rows, just include the first 10 rows. Your SQL queries do not have to be one-liners: you can save the results of a previous query, if that is convenient to you, using `create temp table`. Also, if the query is taking too long then try changing it. Most questions have solutions that run within seconds. A few of the questions Q10 and after may take up to about a minute to run.

Notes: It is possible for any value in a non-primary key column to be null, so pay attention to whether a query may require you to filter out null values.

Q1. Find all people with the first name John.

Q2. Find the title and year of the 5 oldest movies.

Q3. Find the average runtime in hours of all movies released in 1963.

Q4. Find the title of the movie with the highest rating

Q5. List the original title of movies that Daniel Craig has acted in.

Q6. Find the average runtime of movies with a rating higher than 9.0.

Q7. Find the total number of cast members for the movie with the longest runtime.

Q8. Find the name of the director who directed the most movies.

Q9. Find the names of the director and cast member who have directed/acted in the most movies together.

Q10. List the title of movies (if any) with at least 10 cast members where the entire movie cast has acted together in more than 1 movie. [If S is the set of cast members for movie A, list A if S has acted together in at least one other movie.]

Q11. Find the cast member whose career spanned the most years (where the career span of a cast member is the number of years between the first movie they appeared in and the last movie they appeared in).

N degrees of Kevin Bacon:

Q12. Find cast members who have acted in a movie with Kevin Bacon.

Q13. Now, define the ”Bacon Degree” of a cast member as 1 if a cast member has acted in a movie with Kevin Bacon, 2 if the cast member has acted in a movie with one of those actors, and so. Provide a SQL query to compute the Bacon Degree for an actor, and Compute the Bacon Degree for the following actors:

(a) Sean Connery
(b) Humphrey Bogart
(c) Spencer Tracey
(d) Shirley Temple

(You may assume that no actor has a Bacon degree higher than 6 – as they say ”Six Degrees of Kevin Bacon”)