Homework 9: SQL

Due April 9, 11:59 pm

Worth 10 points

Instructions on writing and submitting your homework can be found on the course webpage at `http://pages.stat.wisc.edu/~kdlevin/teaching/Spring2021/STAT679/hw_instructions.html`. *Failure to follow these instructions will result in lost points.* Please direct any questions the instructor.

1 Warmup: `sqlite3` (3 points)

Here is a table similar to the ones that we saw in the lecture slides, describing some information about the colleges in the West Division of the Big 10 conference. [1]

ID	University	City	State	Founded
101	University of Illinois	Urbana	Illinois	1867
202	University of Iowa	Iowa City	Iowa	1847
303	University of Minnesota	Minneapolis	Minnesota	1851
404	University of Nebraska	Lincoln	Nebraska	1869
505	Northwestern University	Evanston	Illinois	1851
606	Purdue University	West Lafayette	Indiana	1869
707	University of Wisconsin	Madison	Wisconsin	1849

1. Use `sqlite3` to create a database with a single table (in addition to the standard metainformation tables) called `t_big10west` that recreates the table in the figure above. That is, `t_big10west` should have five columns (`ID`, `University`, `City`, `State` and `Founded`), and seven rows corresponding to the seven universities in the table. Save the database in a file called `big10.db`, and include this file in your submission.

2. Oops! There's a typo in that table. The University of Wisconsin was founded in 1848, not 1849. Write a SQL command to correct the corresponding entry of the table, and save it in a string-valued variable called `big10_correction`. You **do not** need to run this command (but you probably should, to check that it's correct, and if you do, and you change the entry in the table in `big10.db`, that's okay).

[1] `https://en.wikipedia.org/wiki/Big_Ten_Conference`

2 Relational Databases and SQL (7 points)

In this problem, you'll interact with a toy SQL database using Python's built-in `sqlite3` package. Documentation can be found at `https://docs.python.org/3/library/sqlite3.html`. For this problem, we'll use a popular toy SQLite database, called Chinook, which represents a digital music collection. See the documentation at

> `https://github.com/lerocha/chinook-database/blob/master/ChinookDatabase/DataSources/Chinook_Sqlite.sqlite`

for a more detailed explanation. We'll use the `.sqlite` file `Chinook_Sqlite.sqlite`, which you should download from the GitHub page above. **Note:** Don't forget to save the file in the directory that you're going to compress and hand in, and make sure that you use a relative path when referring to the file, so that when we try to run your code on our machines the file path will still work!

1. Load the database using the Python `sqlite3` package. How many tables are in the database? Save the answer in the variable `n_tables`.

2. What are the names of the tables in the database? Save the answer as a list of strings, `table_names`. **Note:** you should write Python `sqlite3` code to answer this; don't just look up the answer in the documentation!

3. Write a function `list_album_ids_by_letter` that takes as an argument a single character (i.e., a string of length one) and returns a list of the primary keys of all the albums whose titles start with that character. Your function should ignore case, so that the inputs "a" and "A" yield the same results. Include error checking that raises an appropriate error in the event that the input is of the wrong type or if it is not a single character.

4. Write a function `list_song_ids_by_album_letter` that takes as an argument a single character and returns a list of the primary keys of all the songs whose album names begin with that letter (again ignoring case). As in `list_album_ids_by_letter`, your function should ignore case and perform error checking as appropriate. **Hint:** you'll need a JOIN statement here. You can use the `cursor.description` attribute to find out about tables and the names of their columns.

5. Write a function `total_cost_by_album_letter` that takes as an argument a single character and returns the cost of buying every song whose album begins with that letter. This cost should be based on the tracks' unit prices, so that the cost of buying a set of tracks is simply the sum of the unit prices of all the tracks in the set. Again your function should ignore case and perform appropriate error checking.