STAT 19000 Project 9

Topics: Introduction to SQL using sqlite3

Motivation: Some form of database backs any production-level system. SQL or Structured Query Language is a language used to manage many popular databases including but not limited to: MySQL, Postgresql, and sqlite3. We'd be remiss to not touch on a subject so important for organizing large amounts of data.

Context: In previous projects we've learned about XML and touched on scraping data. But what would be the next step? What do we do with large amounts of gathered data? In this project we will work with a database to learn some SQL basics.

Scope: Understand the need for databases, get familiar with SQL and sqlite3.

You can find useful examples that walk you through relevant material here or on scholar:

`/class/datamine/data/spring2020/stat19000project09examples.pdf`.

It is highly recommended to read through these to help solve problems.

Use the template found here or on scholar:

`/class/datamine/data/spring2020/stat19000project09template.Rmd`

to submit your solutions.

Using a terminal within RStudio: If you navigate to https://rstudio.scholar.rcac.purdue.edu, login, and pay close attention, you can see that you can access and use a terminal directly within RStudio!

Important note: From this point on, unless specified otherwise, submit your solutions as an RMarkdown (`.Rmd`) file. Place code in the provided code chunks. Submit both the raw `.Rmd` file as well as the knitted PDF. *Note that you do not need to run anything from within RMarkdown. Simply use RMarkdown to create a better looking set of solutions.*

After each problem, we've provided you with a list of keywords. These keywords could be package names, functions, or important terms that will point you in the right direction when trying to solve the problem, and give you accurate terminology that you can further look into online. You are not required to utilize all the given keywords. You will receive full points as long as your code gives the correct result.

Don't forget the very useful documentation shortcut `?`. To use, simply type `?` in the console, followed by the name of the function you are interested in.

You can also look for package documentation by using `help(package=PACKAGENAME)`.

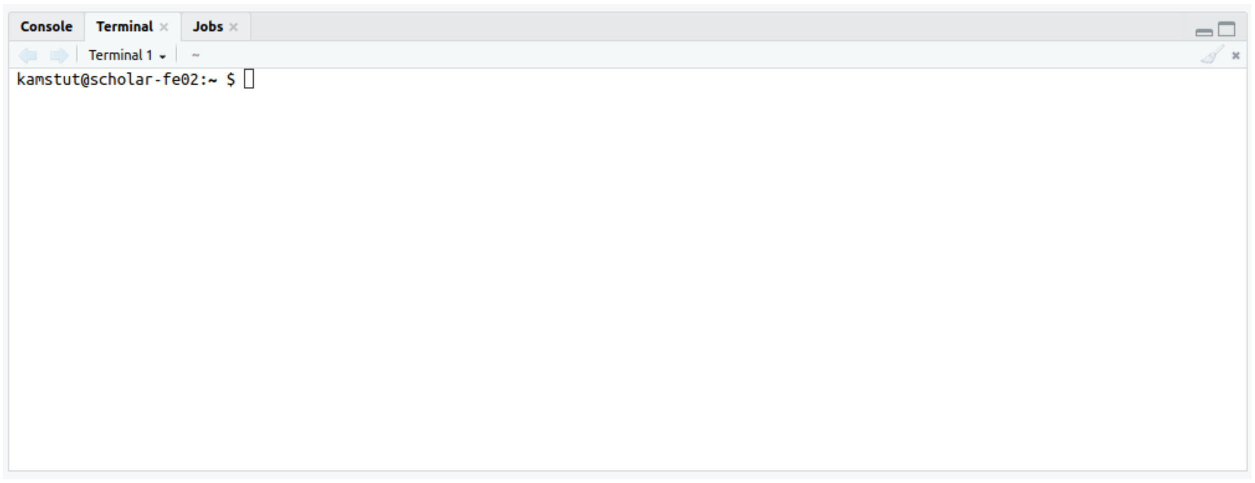
Sometimes it can be helpful to see the source code of a defined function. To do so, type the function's name without the `()`.

Question 1:

1a. *(1 pt)* Describe two advantages a database has over a tool like a spreadsheet.

Item(s) to submit:
- 1-2 brief sentences (3-4 total) describing each advantage.

1b. *(1 pt)* SQLite is an example of a relational database management system (RDBMS). Name 5 other RDBMS's.



```
kamstut@scholar-fe02:~ $
```

Figure 1: Just click the terminal tab.

Item(s) to submit:
- An unordered list (in markdown) of 5 other RDMS's.

1c. *(1 pt)* Name 1 advantage and 1 disadvantage of indexing a column.

Item(s) to submit:
- 1 sentence explaining at least one advantage of indexing a column.
- 1 sentence explaining at least one disadvantage of indexing a column.

Question 2:

For the all of the following questions log in to scholar, open a terminal, and use `sqlite3` to open a connection to the database located:

`/class/datamine/data/spring2020/imdb.db`.

To get started from your terminal session, run the following:

`sqlite3 /class/datamine/data/spring2020/imdb.db`.

You should be presented with an sqlite3 prompt that resembles:

`sqlite>`

Please note that this database is `read-only`. You do not have to worry about accidentally breaking anything.

2a. *(.5 pts)* What are the names of the tables in our database?

Item(s) to submit:
- The sqlite3 dot command used to find the answer.
- An unordered list of 6 table names.

2b. *(.5 pts)* Turn the headers on using the `.headers` dot command. For each of our 6 tables, list the table's headers.

Item(s) to submit:
- The SQL statement(s) and sqlite3 dot command used to find the answer.
- An unordered list containing the table name and the names of all the table's headers for all 6 tables.

2c. *(1 pt)* How many rows of data do each of our 6 tables have?

Item(s) to submit:
- The SQL statement(s) used to find the answer.
- An unordered list containing the table name and the number of rows for all 6 tables.

Question 3:

3a. *(1 pt)* Get the rows of the 10 longest titles in the `titles` table by `runtime_minutes`.

Keywords: *SELECT, ORDER BY, LIMIT, DESC*

Item(s) to submit:
- The SQL statement(s) used to find the answer.

3b. *(1 pt)* Get a list of unique values from the `genres` column in the `titles` table. Note that any new combination or order of comma separated values are to be considered unique. In other words, no need to use any sort of string manipulation to solve this problem.

Item(s) to submit:
- The SQL statement(s) used to find the answer.

3c. *(1 pt)* How many titles in the `titles` table don't have the same `primary_title` and `original_title`?

Keywords: *SELECT, COUNT, WHERE*

Item(s) to submit:
- The SQL statement(s) used to find the answer.
- The number of titles where the primary and original titles don't match.

3d. *(2 pts)* In a single query/statement, get the title information from the `titles` table and the episode information from the `episodes` table for the title with `title_id` "tt0108778". The result should be in ascending order first by `season_number` and then by `episode_number`. Export your result to a csv file called friends.csv.

Hint: *Note that the `show_title_id` column in the `episodes` table is the id for the show itself. The `episode_title_id` is the id for a single episode of a show.*

Hint: *From the previous hint, it makes logical sense that (in addition to the requirement explicitly mentioned in the question, where the `title_id` is "tt0108778") we would want to get the data where the `show_title_id` from the `episodes` table matches the `title_id` from the `titles` table (e.show_title_id=t.title_id), otherwise, we will get information for all shows, not just the show we want.*

Keywords: *SELECT, FROM, AS, WHERE, AND, ORDER BY*

Item(s) to submit:

- The SQL statement(s) used to find the answer.

Project Submission:

Submit your solutions for the project at this URL: https://classroom.github.com/a/-I8VGeyt using the instructions found in the GitHub Classroom instructions folder on Blackboard.

Important note: Make sure you submit your solutions in both .Rmd and .pdf formats. The PDF should be the knitted result of the .Rmd. Make sure and double check that the PDF looks okay and displays your solutions correctly. *Note that you do not need to run anything from within RMarkdown. Simply use RMarkdown to create a better looking set of solutions.*