STAT 19000 Project 11

Topics: MySQL, R & SQL

Motivation: It is important to note that there are a variety of relational databases, and even more databases that solve different problems. In this project you will see that it is just as easy to connect to and use a different relational database, MySQL, as it is to use SQLite. In addition, not everything needs to be solved at the database level. In this project we will use SQL in conjunction with R to solve data driven problems.

Context: In projects 9 & 10 we've learned about SQL by using SQLite. In this project we will use a different database engine, MySQL. In addition, we will use a database in conjunction with R to solve data driven problems.

Scope: SQL, MySQL, RMariaDB, and RSQLite.

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

`/class/datamine/data/spring2020/stat19000project11examples.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/stat19000project11template.Rmd`

to submit your solutions.

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.

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:

For the all of the following questions log in to scholar, open a terminal, and use the following command to login to the MySQL server:

`mysql -u <username> -h scholar-db.rcac.purdue.edu -p`.

Enter the password when prompted. Username and password information is available on scholar:

`/class/datamine/data/spring2020/mysql_credentials.txt`.

Use the crendentials that are for the `imdb` database.

You should be presented with a prompt that resembles:

```
MariaDB [(none)]>
```

Please note that the credentials we've provided you with are `read-only`. You do not have to worry about accidentally breaking anything.

In this series of question we will gain familiarity with MySQL, and run queries directly within MySQL.

1a. *(.5 pts)* Create an unordered list of the names of all databases that the user has access to.

Item(s) to submit:
- The SQL statement you used to find the result, typed in the provided code chunk.
- An unordered list with the names of the databases.

1b. *(.5 pts)* Connect to, or make the `imdb` database your active database. Verify you are connected by running a query that displays all of the tables.

Item(s) to submit:
- The 2 SQL statements you used to connect to or make the `imdb` database your active database, and that you used to display all of the tables.

Great. As you can see, its fairly straightforward to learn how to navigate a relational database, regardless of the engine you are using. Continue to use MySQL to solve the following problems.

1c. *(1 pt)* Get the average `runtime_minutes` of titles of `type` "movie" that premiered after 2010 (inclusive) for each premier year. Include `premiered` in your query. There might be an oddity in the query result. Investigate the query and find the oddity. Name the oddity.

Hint: *The result should contain 19 rows.*

Hint: *The average runtime should not be too long...*

Item(s) to submit:
- The SQL statement you used to find the result, typed in the provided code chunk.
- A sentence explaining the obvious oddity in the result.

1d. *(1 pt)* Explore the oddity in (1c). Write a query to get all of the title information about the *potentially* erroneous data. Do a search in https://imdb.com and explain what is going on.

Hint: *The result should contain 1 row.*

Item(s) to submit:
- The SQL statement you used to find the title info, typed in the provided code chunk.
- A sentence explaining what is going on with the oddity.

1e. *(1 pt)* Write a query to find whether or not titles of `type` "movie" were longer or shorter on average in the 80's (1980-1989 inclusive) or in the 2010's (2010-2019 inclusive).

Item(s) to submit:
- The SQL statement you used to find the result, typed in the provided code chunk.
- In which decade were movies (on average) longer?

Question 2:

In this next series of questions we are going to use the package `RMariaDB` to connect to our database, and run queries from within R.

Important note: It is highly recommended that you use https://rstudio.scholar.rcac.purdue.edu/. Use another system at your own risk.

First, install and load `RMariaDB`:

```
install.packages("RMariaDB")
library(RMariaDB)
```

Then, like we did in the examples, create a connection to the database using the credentials we've provided for you:

`/class/datamine/data/spring2020/mysql_credentials.txt`.

Use that connection, and your knowledge of SQL to solve the following problems from within R.

Important note: We've switched eval=T in your RMarkdown templates. Your output should be displayed automatically now.

2a. *(1 pt)* Get the `primary_title` from the `titles` table, `title` from the `akas` table, and `region` from the `akas` table when the row in the `akas` table `is_original_title` is 1, `region` has a value, and `title_id` from the `akas` and `titles` tables match.

Hint: *You can check if a column has a NULL value by using IS NOT NULL or IS NULL.*

Hint: *The result should contain 19 rows.*

Item(s) to submit:
- R code used to find the solution in a RMarkdown code chunk.

2b. *(1 pt)* The `episodes` table has both a `episode_title_id` and a `show_title_id`. The `titles` table contains not only the series title, but also individual episode titles. Go to https://imdb.com and search for your favorite TV series, and examine the url to find the tv show's `title_id`. For example, Friends is the best tv show. If you go to the friends page on imdb https://www.imdb.com/title/tt0108778/?ref_=ttep_ep_tt, you can see the `title_id` is "tt0108778".

What is your favorite show and it's `title_id`?

Use the `title_id` and write a query that results in a table containing every episode from you favorite tv show in the order first by season and then by episode numbers.

Each row should contain all of the `episodes` table variables.

Each row should contain the following variables from the `titles` table: `title_id`, `primary_title`, `premiered`, `runtime_minutes`, and `genres`.

Hint: *The result should contain 236 rows (if you chose "tt0108778").*

Item(s) to submit:
- The name of your favorite show and it's `title_id`.
- R code used to find the solution in a RMarkdown code chunk.

2c: *(1 pt)* What is the min and max `runtime_minutes` by season of your favorite show?

Hint: *The result should contain 10 rows (if you chose "tt0108778").*

Item(s) to submit:
- R code used to find the solution in a RMarkdown code chunk.

Now, let's continue with the same line of thought, but rather than use R and MySQL, let's experiment with R and SQLite. Like we did in the examples, create a connection to the database located:

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

Use that connection, and your knowledge of SQL to solve the following problems from within R.

2d. *(1 pt)* What is the average rating and total number of votes per season of your favorite show?

Hint: *The result should contain 10 rows (if you chose "tt0108778").*

Item(s) to submit:
- R code used to find the solution in a RMarkdown code chunk.

2e. *(2 pt)* Write a query to get the name of each crew member for your favorite episode of the tv series.

Be sure to include all info from the `episodes` table, the `primary_title` and `runtime_minutes` from the `titles` table, and the `name` from the `people` table.

Hint: *The result should contain 10 rows (if you chose "tt0583450").*

Item(s) to submit:
- R code used to find the solution in a RMarkdown code chunk.

Project Submission:

Submit your solutions for the project at this URL: https://classroom.github.com/a/asbfSNax 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.