PSTAT 10: Homework 7

Solutions; DO NOT DISTRIBUTE

- Instructions for Submission
- Introduction
    - Fall 2022 Course Offerings Database
- Exercises
    - Exercise 0: YAML
    - Exercise 1: Creating the Database
    - Exercise 2: Exploring the Database
    - Exercise 3: Exploring the Database Further
    - Exercise 4: Aggregations and Explorations
    - Exercise 5: Fixing a Mistake
    - Exercise 6: Probabilities
    - Exericse 7: Closing the Connection

A NOTE ON THESE SOLUTIONS: As with many problems in this class, there are often many ways to solve these problems! What this means is: just because your method doesn't match the method outlined in the solutions, doesn't mean your solution isn't correct!

Instructions for Submission

- As usual, 5 Multiple Choice Questions also appear on Canvas and are a required part of this homework. **Please note** that the multiple choice questions for this homework cover material from throughout the course; not just material tested on this homework!
- Remember to save your `.Rmd` file as `hw07-yournetid.Rmd`

Introduction

Fall 2022 Course Offerings Database

We will be exploring the `F22_Offerings` database, which contains information on a portion of the PSTAT courses being offerred this quarter (Fall 2022). A complete database description is included in a file called `Database_Description.pdf`.

Exercises

Exercise 0: YAML

Update the YAML to reflect your own information (i.e. name, collaborators, etc.). If you did not collaborate with anyone, note that down in the YAML.

Exercise 1: Creating the Database

You will notice that we have provided the database to you in the form of several `.csv` files. Our first task will be to combine these into a single `.sqlite` database. This will very closely mirror the process we used in Lecture 16.

(a) Load the required libraries for connecting to an `RSQLite` database.

Solutions:

```
## Add your code here
```

```
library(DBI)
library(RSQLite)
```

(b) You should see a file called `f22_offerings.sqlite` included in the `data` folder of this homework. (It should be blank at first!) Establish a connection to the database and assign this connection to a variable called `f22_offerings_db`.

Solutions:

```
## Add your code here
```

```
## as stated on slide 7, we supply the name to dbConnect()
drv <- dbDriver("SQLite")
f22_offerings_db <- dbConnect(RSQLite::SQLite(),
                              "./data/f22_offerings_solns.sqlite")
```

(c) In the `data` folder, you should see a subfolder called `f22_offerings`. In this subfolder, you should see a series of `.csv` files. Create appropriately-named data frames from these `.csv` files. (Again: Lecture 16 will be very useful!)

Solutions:

```
## Add your code here
```

```
course_info <- read.csv("data/f22_offerings/course_info.csv")
offerings <- read.csv("data/f22_offerings/offerings.csv")
professors <- read.csv("data/f22_offerings/professors.csv")
teaching_assistants <- read.csv("data/f22_offerings/teaching_assistants.cs
v")
```

(d) Write the data frames you created in part (c) to the `f22_offerings.sqlite` database.

Solutions:

```
## Add your code here
```

```
dbWriteTable(f22_offerings_db,
             "COURSE_INFO", course_info, overwrite = T)
dbWriteTable(f22_offerings_db,
             "OFFERINGS", offerings, overwrite = T)
dbWriteTable(f22_offerings_db,
             "PROFESSORS", professors, overwrite = T)
dbWriteTable(f22_offerings_db,
             "TEACHING_ASSISTANTS", teaching_assistants, overwrite = T)
```

(e) Check that your tables have been successfully written to the database.

Solutions:

```
## Add your code here
```

```
dbListTables(f22_offerings_db)
```

```
## [1] "COURSE_INFO"         "JOINED_TABLE"        "OFFERINGS"
## [4] "PROFESSORS"          "TEACHING_ASSISTANTS"
```

Exercise 2: Exploring the Database

(a) Write code to ensure foreign keys constraints are enforced. (This was done several times in Lecture.)

Solutions:

```
## Add your code here
```

```
dbSendQuery(f22_offerings_db,'PRAGMA foreign_keys = ON')
```

```
## <SQLiteResult>
##   SQL  PRAGMA foreign_keys = ON
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

(b) List the fields in the `OFFERINGS` relation, and check that they match the fields listed in the database description.

Solutions:

```
## Add your code here
```

```
dbListFields(f22_offerings_db, "offerings")
```

```
## Warning: Closing open result set, pending rows
```

```
## [1] "CourseID"  "Days"      "Times"     "Location"  "Professor"
```

(c) How many courses is Professor Wainwright teaching? **Answer using CODE;** don't just manually count!

Solutions:

```
## Add your code here
```

```
dbGetQuery(f22_offerings_db,
           "SELECT COUNT(*) FROM Offerings
           WHERE Professor = 'Wainwright'")
```

```
##   COUNT(*)
## 1        2
```

(d) How many lecture sections of PSTAT 5A are there? **Answer using CODE;** don't just manually count!

Solutions:

```
## Add your code here
```

```
dbGetQuery(f22_offerings_db,
           "SELECT COUNT(*) FROM Offerings
           WHERE CourseID = 'PSTAT 5A'")
```

```
##   COUNT(*)
## 1        2
```

(e) How many classes have a lecture that meet on Mondays (not necessarily *just* on Mondays)? **Answer using CODE;** don't just manually count!

Solutions:

```
## Add your code here
```

```
dbGetQuery(f22_offerings_db,
           "SELECT COUNT(*)
           FROM Offerings
           WHERE Days LIKE 'M%'")
```

```
##   COUNT(*)
## 1        4
```

(f) Which professors have offices in South Hall? Display the professors' last and first names, along with their office number. **Answer using CODE!**

Solutions:

```
## Add your code here
```

```
dbGetQuery(f22_offerings_db,
           "SELECT Last_Name, First_Name, Office
           FROM Professors
           WHERE Office LIKE 'SH%'")
```

```
##   Last_Name First_Name  Office
## 1   Swenson      Julie SH 5524
## 2   Meiring      Wendy SH 5510
## 3     Ravat        Uma SH 5503
## 4    Coburn      Katie SH 5524
```

Exercise 3: Exploring the Database Further

(a) How many TA's are hosting a Section of PSTAT 5A?

Solutions:

```
## Add your code here
```

```
dbGetQuery(f22_offerings_db,
           "SELECT COUNT(*)
           FROM Teaching_Assistants
           WHERE Course == 'PSTAT 5A'")
```

```
##   COUNT(*)
## 1       12
```

(b) How many TA's are hosting a Section of PSTAT 10?

Solutions:

```
## Add your code here
```

```
dbGetQuery(f22_offerings_db,
           "SELECT COUNT(*)
           FROM Teaching_Assistants
           WHERE Course == 'PSTAT 10'")
```

```
##   COUNT(*)
## 1        8
```

(c) Which TA's have a first initial between E and I, inclusive? Display the associated TA's Last name as well as their first initial and the class they are TA'ing. **Hint:** Look up how to use the `BETWEEN` clause in `SQL`.

Solutions:

```
## Add your code here
```

```
dbGetQuery(f22_offerings_db,
           "SELECT *
           FROM Teaching_Assistants
           WHERE First_Initial BETWEEN 'E' and 'I'")
```

```
##       Course Last_Name First_initial
## 1   PSTAT 5A    Deamos             E
## 2   PSTAT 5A       Mok             H
## 3   PSTAT 10    Castro             E
## 4   PSTAT 10        Yu             H
## 5   PSTAT 10   Marzban             E
## 6  PSTAT 105      Wang             E
## 7 PSTAT 120A        Lu             H
## 8 PSTAT 120A      Katz             I
## 9 PSTAT 120A     Zhang             H
```

(d) For each course being offered, identify the TA with the longest last name. Display the course, last name, and first initial of the associated TA's. **Hint:** Look up how to use the `LENGTH` clause in `SQL`.

Solutions:

```
## Add your code here
```

```
dbGetQuery(f22_offerings_db,
           "SELECT Course, Last_name, First_initial
           FROM Teaching_Assistants
           GROUP BY Course
           HAVING MAX(LENGTH(Last_name))")
```

```
##       Course       Last_Name First_initial
## 1   PSTAT 10 Tabatabai-Yazdi             C
## 2  PSTAT 105            Wang             E
## 3  PSTAT 115          Hughes             L
## 4 PSTAT 120A      Babichenko             S
## 5   PSTAT 5A     Lencevicius             R
## 6    PSTAT 8      Babichenko             S
```

(e) Which classes have Calculus as a prerequisite?

Solutions:

```
## Add your code here
```

```
dbGetQuery(f22_offerings_db,
           "SELECT CourseID, Title
           FROM Course_Info
           WHERE
              Prereqs like 'Calculus%'
              OR Prereqs like '%Calculus%'
              OR Prereqs like '%Calculus'")
```

```
##     CourseID                                                  Title
## 1    PSTAT 8 Transition to Data Science, Probability and Statistics
## 2   PSTAT 10                      Principles of Data Science with R
## 3  PSTAT 115                 Introduction to Bayesian Data Analysis
## 4 PSTAT 120A                                Probability and Statistics
```

(f) Create a barplot to visualize the distribution of prerequisites. That is, your x-axis should include the names of all prerequisite subjects (e.g. Calculus, Linear Algebra, etc.) and your y-axis should count the number of courses that require each subject as a prerequisite. **Hint:** Consider extracting all of the prerequisites, using the `strsplit()` and `unlist()` functions, and then calling `barplot()` as we did before.

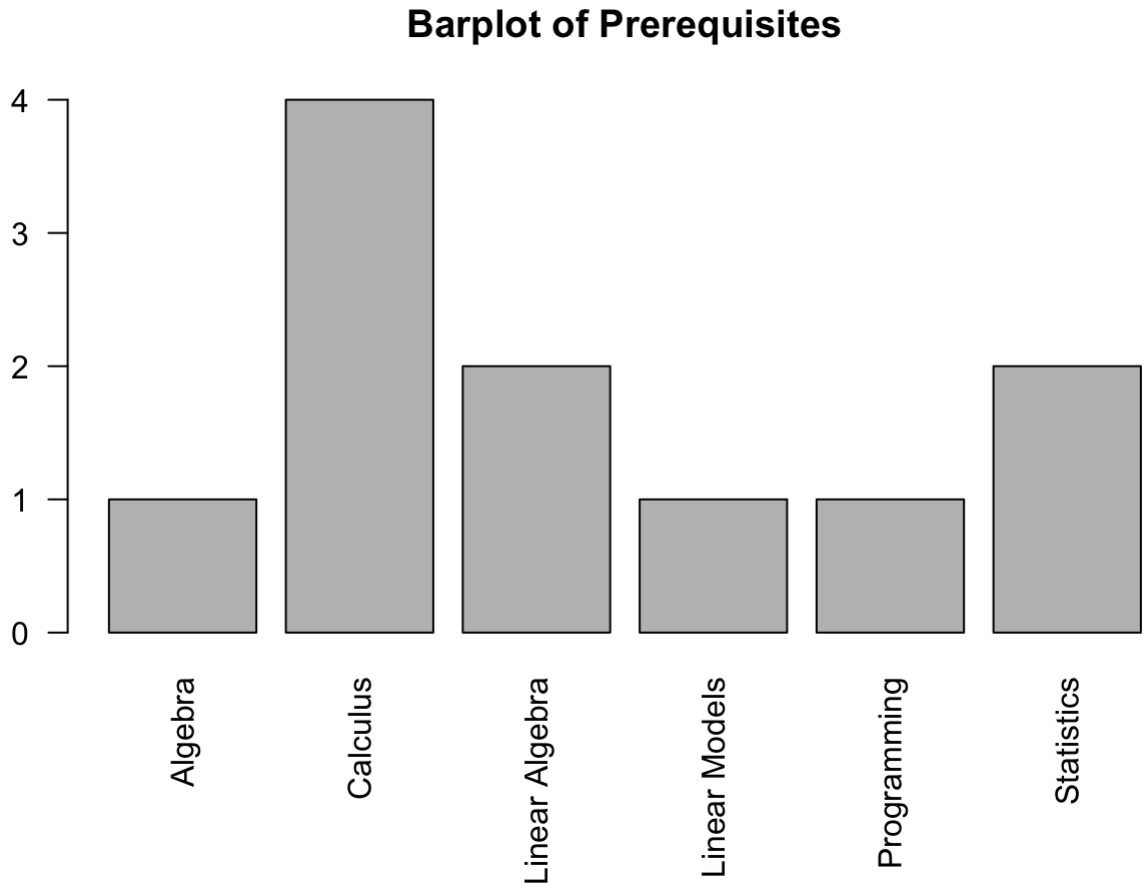
Solutions:

```
## Add your code here
```

```
prereqs <- dbGetQuery(f22_offerings_db,
           "SELECT Prereqs
           FROM Course_Info")

par(mar=c(7,4,4,2)) # optional; adjusts figure margins

barplot(table(unlist(strsplit(prereqs$Prereqs, split = "; "))),
        las = 2,
        main = "Barplot of Prerequisites")
```



NOTE: The space inside the `split` argument of our call to `strsplit()` is very important; otherwise, there will be a prereq called `"Statistics"` and another called `" Statistics"` (and similarly for other subjects), which will be counted as two separate prerequisites in our call to `table()`.

Exercise 4: Aggregations and Explorations

(a) Produce the following table using a suitable join.

CourseID	Days	Times	Lecture_Location	Prof_Surname	Prof_Forename	Office
PSTAT 5A	TR	8-915	CHEMISTRY 1179	Swenson	Julie	SH 5524
PSTAT 5A	MWF	2-250	EMBARCADERO HALL	Wainwright	Brian	UNKNOWN
PSTAT 5H	M	11-1150	GIRVETZ 2127	Meiring	Wendy	SH 5510
PSTAT 8	TR	11-1215	HSSB 1173	Solis	Sharon	ETR 103B
PSTAT 10	TR	8-915	IV THEATRE 1	Ravat	Uma	SH 5503
PSTAT 115	TR	8-915	HSSB 1173	Targino	Rodrigo	OG 1203
PSTAT 120A	MWF	12-1250	LOTTE LEHMANN CONCERT HALL	Wainwright	Brian	UNKNOWN
PSTAT 120A	TR	2-315	TD-W 1701	Coburn	Katie	SH 5524

Solutions:

```
## Add your code here
```

```
dbGetQuery(f22_offerings_db,
           "SELECT CourseID, Days, Times,
                  Location as Lecture_Location,
                  Last_Name as Prof_Surname,
                  First_Name as Prof_Forename,
                  Office
            FROM Offerings
           INNER JOIN Professors on Offerings.Professor = Professors.Last_Na
me")
```

```
##     CourseID Days   Times           Lecture_Location Prof_Surname Prof_Fo
rename
## 1   PSTAT 5A   TR   8-915             CHEMISTRY 1179      Swenson
Julie
## 2   PSTAT 5A  MWF   2-250           EMBARCADERO HALL   Wainwright
Brian
## 3   PSTAT 5H    M 11-1150               GIRVETZ 2127      Meiring
Wendy
## 4    PSTAT 8   TR 11-1215                  HSSB 1173        Solis
Sharon
## 5   PSTAT 10   TR   8-915                IV THEATRE 1        Ravat
Uma
## 6  PSTAT 115   TR   8-915                   HSSB 1173      Targino       R
odrigo
## 7 PSTAT 120A  MWF 12-1250 LOTTE LEHMANN CONCERT HALL   Wainwright
Brian
## 8 PSTAT 120A   TR   2-315                   TD-W 1701       Coburn
Katie
##     Office
## 1  SH 5524
## 2  UNKNOWN
## 3  SH 5510
## 4 ETR 103B
## 5  SH 5503
## 6  OG 1203
## 7  UNKNOWN
## 8  SH 5524
```

(b) Are any professors sharing an office? **Do NOT simply look up the answer from the table you just created in part (a).** Rather, think about how you can answer this question using code. As a hint: think about how you can use `COUNT` and uniqueness to approach this problem. Once you're done, you can check your answer (visually) using the table you created in part (a). **Note:** you will need to examine the result of two separate queries to answer this problem.

Solutions:

```
## Add your code here
```

```
## first, count the distinct number of offices:
dbGetQuery(f22_offerings_db,
           "SELECT COUNT(DISTINCT Office)
           FROM Professors"
)
```

```
##   COUNT(DISTINCT Office)
## 1                      6
```

```
## then, count the number of rows
## (equivalent to counting distinct number of profs)
dbGetQuery(f22_offerings_db,
           "SELECT COUNT(*)
           FROM Professors"
)
```

```
##   COUNT(*)
## 1        7
```

Because the number of distinct tuples (as measured by the `OFFICE` field) is different from the total number of tuples, this means that there is at least one pair of professors sharing an office. From the table in part (a), we can see that these two professors are Drs. Swenson and Coburn, who both share South Hall 5524.

An Alternate way: If we wanted to actually display which professors are sharing an office, we can first use a combination of a `GROUP BY` and a `HAVING` clause to identify which offices have more than one professor associated with them; then, we can filter out professors in only those offices using another query (note that we need to use `sQuote` to display the string with single quotes; we also need to set `options(useFancyQuotes = FALSE)` to ensure `R` doesn't use the unicode single quotes):

```
## alternate way, to select which prof's are sharing an office:

options(useFancyQuotes = FALSE)

more_than_one <- dbGetQuery(f22_offerings_db,
           "SELECT COUNT(Last_Name), OFFICE
           FROM Professors
           GROUP BY Office
           HAVING COUNT(Last_Name) > 1"
)

dbGetQuery(f22_offerings_db,
           paste('SELECT Last_Name, First_Name, Office
           FROM Professors
           WHERE Office == ',
           sQuote(more_than_one$Office))
)
```

```
##   Last_Name First_Name  Office
## 1   Swenson      Julie SH 5524
## 2    Coburn      Katie SH 5524
```

Exercise 5: Fixing a Mistake

You may notice that the `OFFERINGS` relation includes a tuple on PSTAT 105, taught by Dr. Carter, whereas the `PROFESSORS` relation is missing Dr. Carter's information. Let's fix that!

(a) Add the following tuple to the `PROFESSORS` relation:

- <Carter, Andrew, SH 5507>

Solutions:

```
## Add your code here
```

```
dbSendQuery(f22_offerings_db,
            'INSERT INTO PROFESSORS
            VALUES ("Carter", "Andrew", "SH 5507")')
```

```
## <SQLiteResult>
##   SQL  INSERT INTO PROFESSORS
##             VALUES ("Carter", "Andrew", "SH 5507")
##   ROWS Fetched: 0 [complete]
##        Changed: 1
```

(b) Check that your tuple has been successfully added.

Solutions:

```
## Add your code here
```

```
dbGetQuery(f22_offerings_db,
           "SELECT * FROM PROFESSORS")
```

```
## Warning: Closing open result set, pending rows
```

```
##    Last_Name First_Name   Office
## 1    Swenson      Julie  SH 5524
## 2 Wainwright      Brian  UNKNOWN
## 3    Meiring      Wendy  SH 5510
## 4      Solis     Sharon ETR 103B
## 5      Ravat        Uma  SH 5503
## 6    Targino    Rodrigo  OG 1203
## 7     Coburn      Katie  SH 5524
## 8     Carter     Andrew  SH 5507
```

(c) Now, recount the number of professors that have an office in South Hall.

Solutions:

```
## Add your code here
```

```
dbGetQuery(f22_offerings_db,
           "SELECT COUNT (*)
           FROM PROFESSORS
           WHERE OFFICE LIKE 'SH%'")
```

```
##   COUNT (*)
## 1         5
```

Exercise 6: Probabilities

For each of these parts, use a combination of `SQL` queries and computations done in `R`.

(a) Suppose a Teaching Assistant is selected at random. Given that they are a TA for PSTAT 120A, what is the probability that their first name begins with the letter "S"?

Solutions:

```
## Add your code here
```

```
num <- dbGetQuery(f22_offerings_db,
                  "SELECT COUNT(*)
                  FROM Teaching_Assistants
                  WHERE Course == 'PSTAT 120A'
                  AND First_Initial == 'S'")

denom <- dbGetQuery(f22_offerings_db,
                  "SELECT COUNT(*)
                  FROM Teaching_Assistants
                  WHERE Course == 'PSTAT 120A'")

num/denom
```

```
##    COUNT(*)
## 1 0.2222222
```

(b) In this problem, we will work toward answering the question:

> From the pool of professors included in the dataset, a single professor is to be selected at random. What is the probability that this professor is either teaching PSTAT 5A or has an office in South Hall?

We will work toward our answer using a combination of the probability theory and `SQL` knowledge we've learnt in this class.

(i) Define events, and identify the probability rules you'll use to calculate the desired probability.

Solutions:

Replace this line with your answer.

Define:

- A = "a randomly selected professor is teaching PSTAT 5A"
- B = "a randomly selected professor has an office in South Hall"

We seek $\mathbb{P}(A \cup B)$; by the **Addition Rule** this is computed as $\mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(A \cap B)$, where we can use the **Classical Approach to Probability** to compute each of these probabilities, since we are told that the professor is selected *at random*. What this means is, for example,

$$\mathbb{P}(A) = \frac{\text{number of professors teaching PSTAT 5A}}{\text{total number of professors}}$$

and so on and so forth.

(ii) Notice that the professors' offices and the courses they are teaching are located in two separate relations. Join these two relations, and return all rows; save this resulting table in a data framed called `table1_join_table2`.

Solutions:

```
## Add your code here
```

```
table1_join_table2 <- dbGetQuery(f22_offerings_db,
                  "SELECT CourseID, Professor as Prof_Surname, Office
                  FROM Offerings
                  INNER JOIN Professors on Offerings.Professor = Professors.
Last_Name")
```

(iii) Now, write your `table1_join_table2` data frame from part (ii) above to a new table in the `F22_Offerings` database, called `JOINED_TABLE`.

Solutions:

```
## Add your code here
```

```
dbWriteTable(f22_offerings_db,
             "JOINED_TABLE", table1_join_table2, overwrite = T)
```

(iv) Finally, use your newly written table to obtain the desired probability by writing as many `SQL` queries are necessary. Keep in mind you may need to perform some computations in `R` (i.e. outside of `SQL` queries).

Solutions:

```
## Add your code here
```

```
num_5a <- dbGetQuery(f22_offerings_db,
                  "SELECT COUNT(*)
                  FROM JOINED_TABLE
                  WHERE CourseID == 'PSTAT 5A'")

num_sh <- dbGetQuery(f22_offerings_db,
                  "SELECT COUNT(*)
                  FROM JOINED_TABLE
                  WHERE Office Like 'SH%'")

num_5a_and_sh <- dbGetQuery(f22_offerings_db,
                  "SELECT COUNT(*)
                  FROM JOINED_TABLE
                  WHERE CourseID == 'PSTAT 5A'
                  AND Office Like 'SH%'")

tot_num <- dbGetQuery(f22_offerings_db,
                  "SELECT COUNT(DISTINCT Prof_Surname)
                  FROM JOINED_TABLE")

dbGetQuery(f22_offerings_db,
                  "SELECT COUNT(DISTINCT Prof_Surname)
                  FROM JOINED_TABLE")
```

```
##   COUNT(DISTINCT Prof_Surname)
## 1                            8
```

```
(num_5a + num_sh - num_5a_and_sh) / (tot_num)
```

```
##   COUNT(*)
## 1     0.75
```

NOTE: There was perhaps some ambiguity about what the "pool of professors included in the dataset" represents. The original intent of the question was to count *distinct* professors, of which there are 8. We can, however, see that some students may interpret this as simply counting all professors in the original `OFFERINGS` table, which would yield 9 professors (since Dr. Wainwright is teaching two classes).

Exericse 7: Closing the Connection

Don't forget to close your connection to the database!

Solutions:

```
## Add your code here
```

```
dbDisconnect(f22_offerings_db)
```
