COMP 430/533 Assignment #2

1 Description

The goal of this assignment is to write several complex SQL queries that will answer questions about clinical trials registered with the US government. The data is a subset of the data made available through the Clinical Trials Transformation Initiative. The data provided is a subset of the full data set, so be sure to use the version of the data available on the course Canvas site. It is a different subset than what was used in A1, so be sure to get a new copy. If you are interested, you can learn more about the dataset at http://aact.ctti-clinicaltrials.org.

1.0.1 What's In and Out of Scope

This is intended to be a SQL query assignment. Therefore, you must write queries in SQL (not functions). All answers must be computed by a self contained query (you may not use variables outside the query to store values). You may use VIEWs as needed and you may use standard built-in Postgres functions (e.g. ROUND or CASE statements). If you're not sure if something is allowed, ask!

2 Getting Started

First, go to your database, and create the tables found in the tablesA2.sql file.

2.1 Load the data

Load the data needed for the assignment. You should do this in pgAdmin or psql. The files are provided in the Canvas assignment. They are .sql files. You can copy & paste the contents into a pgAdmin window and run them or use the following command in psql:

```
\i <filename.sql>
```

Where <filename.sql> is the name of the file you want to run. For example:

```
\i conditions.sql
```

You must use the table and attribute names provided. Do not rename anything.

3 Queries

Answer all of the questions below by writing and executing SQL queries. The queries must contain ONLY the answer to the question (no extra rows or columns). You may need to explore the database a bit prior to generating your final solutions.

1. Suppose we want a master list of all the contacts in the database.

   (a) (*5 points*) Write a query that returns the all of the name, email, phone number for every entry in the central_contacts, result_contacts, and facility_contacts tables.

   Next, modify your query to count the number of rows returned. What is the count? (submit both queries and the count value)

   (b) (*3 points*) Modify your query to eliminate all duplicates. What is the new count? (submit the query and the count)

(c) (*2 points*) Contact Efthymios Avgerinos appears multiple times in the results. Why? Propose a way to change the database that would reduce these types of duplications.

2. There are numerous scores and indexes used to predict mortality or other dire outcomes. One such is a modification of the Elixhauser comorbidity measure into a score by van Walraven, et al. (See paper in file vanWalraven-2009.pdf). Typically, this score is computed for individuals. Instead, we are going to use it to identify the studies that appear to focus on patients with the highest mortality risk as determined by this score.

   Table scoreTerms(name, term) contains pairs of condition names and terms relevant to our data. The conditions are gleaned from the conditions table, where there are matches for the Elixhauser / van Walraven components. The attribute "term" groups the different conditions. In other words, there may be multiple "name" values that map to a single "term". This would occur if more that one condition name falls under the criteria for that element of the score.

   Table scorePoints(term, points) contains pairs of terms and point values. The term matches the term value in the scoreTerm table. The attribute "points" contains the number of points added to the score for the study if the term is present in the study.

   For each study in the studies table, compute the van Walraven score as follows:
   If any of the study's conditions appear in the scoreTerms table, include the points associated with that term into the score for that study. Add together all of the points from each term to compute the final score.

   Then provide queries that answer the following questions, and give the answer:

   (a) (*5 points*) What is the highest possible score value based on the data in scorePoints?

   (b) (*15 points*) What the nct_id and the score value for the study with the highest score in the data provided?

   (c) (*10 points*) How many studies have a score of 6 and have conditions that meet the 'Neurodegenerative disorders' term criteria?

   (d) (*10 points*) How many studies have no risk terms? That is, none of the conditions that are listed in the ScoreTerms table are included in the study.

   (e) (*10 points*) What is the average number of contributing conditions and average number of terms per study that has a non-zero score? Round to 2 decimal places
   In other words, if you consider only studies where the score is not equal to zero, what is the average number of "conditions" present that are of interest? On average, how many unique terms do they compose? Provide your query and the results.

3. (*5 points*) Suppose we want to add a similar, but different score. That is, a score that also assigns points to certain conditions. Suggest a change to the database schema to easily accommodate additional scores. We want to minimize changes to the schema and to the queries we have already written to compute a score based on conditions and points.

4. (*5 points*) Some of the point values in scorePoints are negative. Give one reason why that might be.

5. (*10 points*) Which other studies have all of the conditions in study NCT02789800? Note: you may not hard-code the condition names from study NCT02789800 in your query.

6. For this question, consider ONLY the data from studies with start_date of '2016-05-01'.

   The Jaccard Index (`https://en.wikipedia.org/wiki/Jaccard_index`) provides a similarity measure over sets. Compute the Jaccard Index for every pair of studies. Note that these queries may take a while to run. Let's compute the Jaccard Index for studies using conditions. Depending on how you write your queries, they can take a very long time to run. So, you might give some thought as to how to make them more efficient. Note that there is a base case for computing the Jaccard Index, where if the number of conditions you are comparing from both studies is 0, the Jaccard index is defined to be 1.

   Note: Be careful about integer division here. If you divide two integers, you will get another integer. In this case, we want a decimal value. I recommend using the NUMERIC data type when calculating the Jaccard Index.

   (a) (*15 points*) What is the average non-zero Jaccard index value in our set of studies? In other words, if you exclude pairs for which the Jaccard index value is 0, what is the average score?

   (b) (*5 points*) What percentage of study pairs have a Jaccard Index of 1? Provide the percentage to 2 decimal places.

4 Turnin

Create a document that contains your SQL code, as well as the results from running your code. By 11:55P on the due date, submit this document electronically to Canvas. You must submit a text file with a .txt or .sql extension. Other formats (such as Microsoft Word or PDF) are not acceptable. Your file should be "executable". That is, the TA should be able to run your code without any errors. This means that any non-code in your file (e.g. query results) should be in comments. In Postgres, comment blocks are denoted with a starting /* and an ending */. A single line comment is also possible, using a double dash.

5 Grading

The number of points for each query is indicated in the question. If you don't get the right answer or your code is not correct, you won't get all of the points; partial credit may be given at the discretion of the grader.

6 Academic Honesty

The following level of collaboration is allowed on this assignment: You may discuss the assignment with your classmates at a high level. Any issues getting Postgres running is totally fine. What is not allowed is direct examination of anyone else's SQL code (on a computer, email, whiteboard, etc.) or allowing anyone else to see your SQL code. You MAY post and discuss query results with your classmates.

You may use the search engine of your choice to lookup the syntax for SQL commands, but may not use it to find answers to queries.