“Public Libraries”: More Sophisticated Queries Against Relational Databases

Avraham Leff

COM3580: Spring 2024

1 Assignment-Specific Packaging

The general packaging is unchanged from the basic “Homework Requirements” (see slides from first lecture and “Homework Policies for COM 3580” on Piazza).

This assignment’s “DIR” **must be named** *PublicLibraries*. Your report **must be named** *$DIR/PublicLibraries.pdf*.

2 Motivation

This assignment requires you to use your knowledge of “more advanced SQL” to formulate SQL queries that provide the specified semantics and to execute them against a large database. You’ll find that with a rich data-set, once you get beyond “select *”, you can use SQL to formulate some very interesting queries.

Other benefits from the assignment:

- More experience understanding what a data-set contains and importing that data into an SQL database.

3 Background

The *Institute of Museum and Library Services* (IMLS) maintains data-sets about Public Libraries Surveys. When I last checked, you can download data-sets in e.g., csv format, spanning the years 1992-2021.

The meaning of the csv fields is supplied in documentation bundled with the data-set, e.g., Appendix A here.

To make the assignment more manageable, the data-sets you'll be using are a subset of the columns supplied in the IMLS data-set (but are otherwise unchanged).

4 Getting Started

1. Download the csv data from here, here, and here.

   Note: there are three files, each containing data for a given year. You will create one database table per downloaded file.

2. Create the three database tables, one per csv file. Note: the three files have the same structure.

   You may name the tables and columns as you choose, but I'd suggest that you name the columns identically to the csv field names. Similarly, you may "type" the table columns as you choose, but I urge you to keep the semantics as close to the documented semantics (and to keep things as simple as possible).

   You're responsible for specifying integrity constraints, if any.

   You may create as many indexes as you like (including "no indexes at all").

   Some important tips:

   - Type all *date related* fields as `text` to deal with csv data whose values can't be converted into SQL date. (This assignment is not about "data cleansing".)
   - Consider using an *identification code* attribute (Appendix A) as the *primary key* attribute.

3. Import the csv data into the POSTGRESQL database tables that you created in the previous step (read the POSTGRESQL documentation, and use whatever technique works for you).

5 Points To Consider (Extracted From The Documentation)

> Records for public libraries that were closed for the current year are included in the file for that year only. The closed records are not included in the appendix tables of this document or the Supplementary Tables.3 Data elements for the closed records are set to a value of -3 (not applicable), with flag U_16. Each library's data consist of one record. Appendix A contains the record layout.

And

> Alphanumeric fields that contain "M" and numeric fields that contain "-1" indicate nonresponse.

5.1 Sanity Check

After loading the csv data into the tables, you should have the following number of tuples.

- **2018**: 9261
- **2017**: 9245
- **2016**: 9252

6 Requirements

Note: you are *are allowed/encouraged* to use resources such as the Internet or books to get learn the SQL and POSTGRESQL material (and whatever other "how-to-do" information that you need). You are **not allowed** to search for information specific to this assignment, nor are you allowed to discuss the assignment with anyone else.

- When a query specifies multiple attributes be sure that your result set returns those attributes in the **named as, and in the order specified**!
- Return **only those attributes** that I've specified in the query.
- For each of the following queries, in the **specified order**, create a separate section in your write-up file. Each section should contain:

  1. The original natural-language query
  2. Your translation of the query into SQL
  3. A screenshot of the results

- Points will be deducted if you don't follow the above instructions.

6.1 The Queries

1. *"What is the most common library name in the 2018 data-set?"*

   Your answer should be a tuple with two columns, in this order: `libname`, `count`.

2. *"Which state has the most libraries?"*

   Your result set should contain seven tuples, in descending order of "number of libraries". Each tuple should contain (and be labelled), in this order: `state`, `number of libraries`

3. *"For each state, in 2018, how many libraries changed their address in any way?"*

   Your result set should contain, in descending order of "number libraries moved in that state in 2018". Each tuple should contain (and be labeled as) in this order:

   `state`, `number moved`.

4. *"Return the number of visits to libraries in 2018, 2017, and 2016."*

   The query is intended to get a sense of whether library usage has increased, decreased, or stayed roughly the same over this time period. Therefore: you need to do an "apples-to-apples" comparison, such that only libraries that were open in each of these years are used in the computation.

   Be sure to only include "valid survey responses/data" (see above) in the computation.

   Note: you're aggregating data across the United States as a whole.

   The result set should be a single tuple, containing (and labelled), in this order: `V2018`, `V2017`, `V2016`.

5. *"Do a by-state analysis of the above query (details below)."*

   Instead of reporting data aggregated across the United States (the previous query), you're aggregating data by state. You'll report the "raw number of visits" for each of the three years. In addition: you'll report (by state), the *trends* of visits: specifically, the *percentage change* (whether positive or negative) in 2018 relative to 2017, and the percentage change in 2017 relative to 2016.

   Your answer should be a tuple containing (and labeled) in this order:

   (a) `state`

   (b) `V2018`

   (c) `V2017`

   (d) `V2016`

   (e) `CHANGE_2018_17`

   (f) `CHANGE_2017_16`

   Report only the first ten tuples when the result set is ordered by descending "percentage change from 2018 to 2017" values.

   Note: this query uses the previous queries' semantics with respect to e.g., library usage.

   Suggestion: be careful about integer division!