CS 564, Spring 2017: SQL Assignment

Due Date: April 14, 2017 by 2:00PM. No late days.
Project Grade Weight: 6% of the total grade

[This is an individual assignment, do not work in groups!]

Introduction

In this assignment, you will use SQLlite3 to build a simple database application. You can find documentation on SQLite3 at http://www.sqlite.org/. SQLite3 is not as functional as PostgreSQL or mySQL or commercial relational DBMSs, but it is much easier to use and program. That is why there are many more installations of SQLite than PostgreSQL or any other traditional database management system (DBMS). Also, you can easily install SQLite3 on your own machine, but make sure your final code runs on SQLite3 installed on the CS **snares-XX.cs.wisc.edu** machines.

For this assignment, you will use the USDA National Nutrient Database. Most substantial datasets are published with a bulky document describing the data. Here is what that looks like for this dataset: https://www.ars.usda.gov/ARSUserFiles/80400525/Data/SR/SR28/sr28_doc.pdf. Starting on page 29 (page 36 in the pdf) of this document, you will find the schema and file format descriptions for this dataset.

Your first task is to load the data into SQLite3 using SQL DDL. USDA provides the data at: https://www.ars.usda.gov/ARSUserFiles/80400525/Data/SR/SR28/dnload/sr28asc.zip. Load this data by writing a C++ program that embeds SQL calls to create the appropriate tables. Then, load the data from the data files into each of the tables that you create. Before creating the tables in your program, make sure that you drop the tables if they already exist. Your program should be a single C++ file called `load.cpp`. You are allowed to have an additional header file, called `load.h`. This load program should create the database in a file called "`nutrients.db`". For this part, you will need to use the C/C++ interface to SQLite.

Your second task is to write the following queries in separate files, as described below:

File name	Query
`Query1.txt`	List the nutrition value of the nutrient 205 (205 is the unique 3-digit identifier code for a nutrient) in "`McDONALD'S, Hamburger`".
`Query2.txt`	Print the name of the food item(s) with the highest nutrition value per 100 grams.
`Query3.txt`	Write a query to generate Table 1 in the `sr28_doc.pdf` (you don't need to sort it in exactly the same way as Table 1).
`Query4.txt`	Print the name of the food item(s) with the highest number of nutrients in it and also print the count of the nutrients.
`Query5.txt`	List all the food items whose average nutrition value (total nutrition value per 100 grams/ number of nutrients) is greater than that of "`McDONALD'S, Hamburger`".

Query6.txt	List all the food item(s) that contain "water" but do not contain "calcium". Sort the result in descending order of food group code. For food items belonging to the same food group, sort them in ascending order of short description of the food item.
Query7.txt	Find all the food items that contain both "Caffeine" and "Alcohol, ethyl" and then list both these ingredients per 100 grams of the food item.

If you can't write any of the queries above, provide an explanation of the features that is lacking in SQLite3 that prevents you from being able to write that query. Write these explanations in a file called `Queries-readme.txt`.

Note: To check the query output, we will first load the data using your load program, and then run:

```
sqlite3 nutrients.db < Query3.txt
```

Thus, your text file can have multiple queries, but it must only output your final desired output, and clean up any temporary tables that it creates. You should try to minimize the # of distinct SQL query blocks in each query file – **try to write a single query for each of the queries above, whenever possible**.

Copy your query results to a file called `results.txt`.

Compiling C++ file for step 1

1) Name your c/c++ file as load.cpp

2) Download cs564_ass3.tar.gz. The file can be downloaded from

http://pages.cs.wisc.edu/~jignesh/cs564/projects/SQL/cs564_ass3.tar.gz

3) Uncompress the tar file (run: `tar –xzvf` cs564_ass3.tar.gz). This folder has the makefile and the necessary files to run sqlite3 api's.

4) Place load.cpp in this folder. (Replace if already a file with the same name exists).

5) To compile the code, Run: make

Submission Instructions (Updated- 3rd April)

Only the following files are required for this assignment: `load.h`, `load.cpp`, `Query1.txt`, `Query2.txt`, `Query3.txt`, `Query4.txt`, `Query5.txt`, `Query6.txt`, `Query7.txt`, `results.txt`, `Queries-readme.txt`. Please use comments in all your code (including in your SQL file). Please follow these instructions to submit the project:

1) Place all the required files in the "cs564_ass3" folder. (described in the previous section "Compiling C++ file for step 1")

2) Rename this directory using the format: `<lastname>_<firstname>_P4` (e.g. `Cook_Tim_P4`).

3) Run: make clean, from inside the directory (so that the submitted file size is small)

2) Run:

```
tar –czvf <lastname>_<firstname>_P4.tar.gz /path-to-project/lastname_firstname_P4
```

3) Submit the tar file.

4) To check you can uncompress the tar file. (run: `tar –xzvf <lastname>_<firstname>_P4.tar.gz`).

To hand in your work, please go to the Canvas: Assignment 4 (SQL) to upload your files. Your files must be uploaded by the deadline stated on the first page.