CSC 370: Database Systems
Summer 2019

Assignment 4

Due: Thursday, August 1st, 2019, 11:55 pm, via git push

Important

All work for this assignment is to be done using PostgreSQL 10. Each student has been assigned their own account/database on a server instance of PostgreSQL10; the server is located at `studsql.csc.uvic.ca`.

As part of the previous assignment, you have already receive the information needed to access your git repository and PostgreSQL database instance. **Note: If you have already cloned a copy of the repo, you need not do this again.**

Please use the `linux.csc.uvic.ca` login server for your work (as was described in lecture).

Learning Outcomes of assignment

By the end of this assignment, you should be able to:

- Write a Python 3 program that connects to your Postgres DBMS schema.
- Execute A#3 queries from within the program and integrate the results from the queries used by the rest of your program.
- Execute modification statements (`update` , `delete` and `insert` operations) from within the Python script.
- Utilize exception handling to provide appropriate program behavior in the event of unexpected events when interacting with the DBMS.
- Interact – in a simple way – with a user (who is a hypothetical employee of the "Green-not-Greed" organization) by accepting the user's inputs and responding with appropriate results.

A continuation....

In the previous assignment you modelled the database needed for a hypothetical environmental activist organization named "Green-not-Greed" (GnG). In this assignment you will use those tables, queries, views, etc. from within a Python program that you will write. This program is intended to be used by employees of GnG who are going about the tasks of the organization. In order to keep our focus on interacting with the database (i.e., the classical "application-server"-tier level of coding) we will keep the user interface very simple (i.e., text based). However, the information provided by the program to the user should be clear – that is, cryptic prompts and confusing output must be avoided.

As you write the program you may decide that some of your tables and queries need to be changed in a significant way. You are allowed to do this, but please ensure that these changes are carefully documented in a `CHANGES.md` document stored within the `sql/` directory of your `git` repository.

In each of the phases below you must use exception handling where appropriate. Working with databases is fraught with peril – much can go wrong, and we want the user isolated (as best possible) from DBMS errors. If an exception is likely to occur, your code must handle it.

Phase 1: Queries from Assignment #3

Your program must allow the GnG user to select from the queries you prepared in Assignment 3, and to produce the output in a way suitable for the user to read and interpret. Note that your queries should be briefly described in the program's menu; simply allowing users to select from "query 1", "query 2", ..., "query n" is not enough information.

Phase 2: Setting up a campaign with volunteers, activities, etc.

Your program will enable a GnG user to enter the information needed to set up a campaign, including: adding new volunteers to the organization who have joined specifically to help with the campaign; scheduling events; etc. Existing volunteers can help with the campaign. The GnG user must be able to see the state of a campaign at suitable points during the steps needed for campaign setup.

Phase 3: Some accounting information

GnG does have donors and campaigns have costs. Reporting on these fund inflows and outflows is often needed as the group plans for future events. The GnG user must be able to obtain this kind of data in both a textual and a quasi-graphical format (e.g, ASCII bar charts) and at a suitable level of detail.

Phase 4: Membership history

Another useful tool when planning is to invite specific members/volunteers to take on tasks for campaigns, yet organizers are very mindful that time is finite. Member burnout is commonplace in many volunteer organizations, and we can help organizers here by showing them the way members have been involved. Some of this browsing through membership history may also suggest ideas or annotations to be added to campaigns or member records. The GnG user must be able to browse both membership history and add annotations to campaigns, member records, or any other piece of data you believe appropriate.

Phase 5: Your own ideas

Create at least one more way in which the GnG user can interact with the database – via the Python program– in a non-trivial way.

A word about Python 3

The default version of Python available on `linux.csc.uvic.ca` is Python 2.7 (i.e., what you get when you type `python` on the command line). I do request, however, that you write your solution using Python 3 (i.e, `python3` on the command line). As much as I would like to accommodate requests for extra additional modules to be installed via `pip`, I'm afraid I'm unable to do so. Therefore please write your solution with what is available (i.e., the bog-standard Python 3 distribution, which unfortunately does not have such things as `numpy` or `matplotlib`, etc.).

What to submit

All of your work is to be stored in your `git` repo (that same one you used for A#3) and submitted via `git push` – that is, diagrams, notes, Python code, markdown files, SQL files, etc. must be in the `git` project provided to you. **E-mailed submissions will not be accepted**.

- In a directory named `src/` your Python program/script with the name of `gng.py`. You are welcome to break down the whole application into your own modules, but please ensure all of this code is also committed into the `src/` directory.

- A file named `src/PHASE5.md` that lists the functionality you have invented for your GnG program in Phase 5 (above).

- The SQL file `sql/gng_dump.sql` containing the output produced by the `pg_dump` command on `linux.csc.uvic.ca` when used with your database. This file contains all of the SQL commands needed to reconstruct your database with all tables, table data, and views (i.e., your SQL queries).

You may create other directories as needed in your `git` project to support your work. Please ensure these directories are properly `added` and `committed` before your final `push` for the assignment.

Evaluation

As there are many possible correct solutions to the problem, evaluation will be done via a demonstration of work in front of a member of the CSC 370 teaching team. Information on demos (i.e., where, when, how to sign up) will be distributed shortly before the assignment due date.

The marking scheme below will be used:

- "A" grade: An exceptional submission demonstrating creativity and initiative. The program provided is thorough and shows insight, enables the user to interact with the data in a clear way, and is free of unexplained behavior. Phase 5 shows exceptional creativity.

- "B" grade: A submission completing the requirements of the assignment. The program provided is thorough, enables the user to interact with the data, and is free of unexplained behavior. Phase 5 has been completed and is non-trivial.

- "C" grade: A submission completing most of the requirements of the assignment. The program has some rough spots, but the user is able to interact with the data. There is some unexplained behavior. Phase 5 might not have been completed.

- "D" grade: A serious attempt at completing the requirements of the assignment. There are many problems with the submitted work.

- "F" grade: Either no submission is given, or submission represents very little work.