Assignment 3 Embedded SQL Report

Nicholas Wengel, Aida Radu, Hugh Bagan

Overview

Our application provides an interface for obtaining information about a database. To use the application, the following dependencies are required:

- `Python 3`, `pip`, `pandas`, and `matplotlib`. SQLite3 is also used, but comes bundled with Python 3.

There is a `setup.sh` provided that will inexhaustively install these dependencies on Debian-based Linux OSs.

If a database does not already exist, one can be created by running a provided script:

```
$ ./rebuild_modded_db.sh
```

To start the application, run the following command:

```
$ python3 main.py database.db
```

(Note: This assumes that your database is named database.db and is located in the same directory as main.py.)

A menu will appear with six options as well as the option to quit. Each option will return certain information from the database. Beside each task is a character in square brackets. Type this character to run the corresponding task, and follow any next instructions.

(Note: When a task displays a graph, it must be closed before the program can continue.)

Task 1

Type N or P to move to the next and previous pages (respectively) or Q to quit. To select a paper, type the number in the square brackets beside its title. The list of reviewers that have reviewed the selected paper will be displayed.

Task 2

Use the same controls as Task 1 to switch pages. Each paper's title is listed along with its area of expertise. The potential reviewers for the selected paper will be displayed (excluding reviewers that have already reviewed the selected paper).

Task 3

The program will prompt you to enter two numbers, `a` and `b`. The reviewers that have reviewed a number of papers within the inclusive range `[a, b]` will be displayed.

Task 4

There are two options: type `1` to display a bar graph with the number of sessions that authors participate in. Alternatively, type `2` to display the authors that participate in sessions, and type the number beside an author's name to see how many sessions they participate in (without a graph).

Task 5

The top 5 most popular areas of expertise for papers is displayed. The exact counts are displayed in the terminal, and the percentages are in a pie chart.

Task 6

A bar chart is displayed that contains the average scores that each reviewer has given to papers (for each scoring category).

Application Design

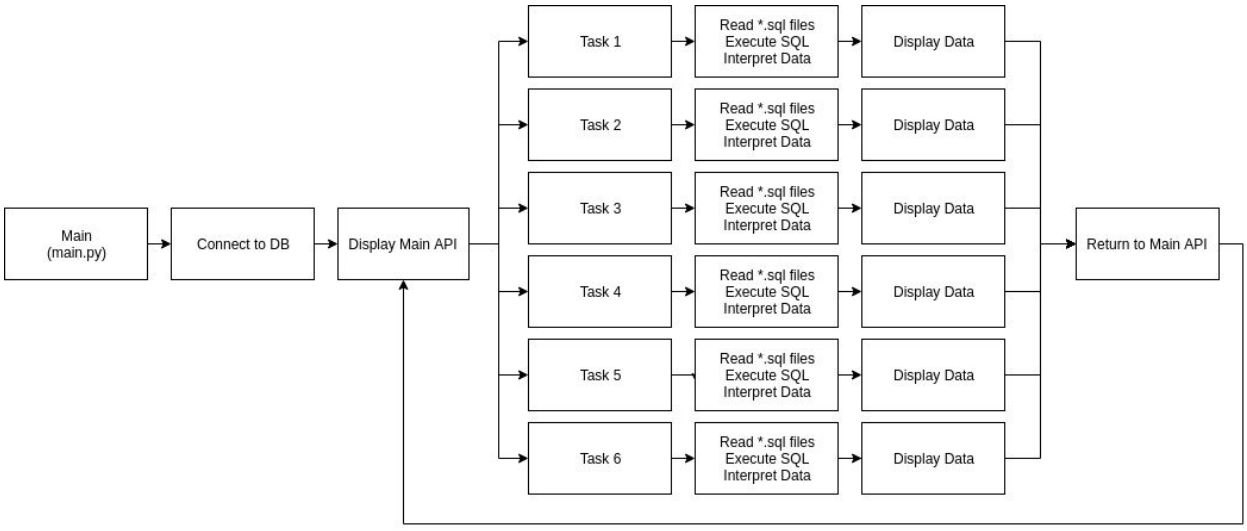
Python was used as the project's programming language, since it is high-level enough for fast software development and does not require a compile step. Python has support from libraries such as `pandas` and `matplotlib` to ease data processing and visualization.

The database name is passed to the program as a command line argument so it doesn't have to be retyped at a dialog every time the program starts.

The core program is contained in `main.py`.The `main()` function sets up a database connection then enters a main loop that receives keyboard input from the user. The program API is displayed as a list of labelled tasks, which are executed as prompted by the user. Task selection is done using a dictionary of functions since Python does not support switch statements. Each task function receives only the connection to the database, making them pure.

SQL queries are stored as individual files to avoid cluttering `main.py`. They are not embedded, but are loaded as needed by the task functions.

Control flow diagram (made with draw.io):



Testing

Git and GitHub were used for developing and versioning the project. GitHub's pull request system was used to ensure that new code was tested before being merged from feature branches onto our repository's master branch. To enforce this, merging without an approving review was disabled, as was commiting directly to the master branch.

We did not create automated tests due to time constraints. The strategy was to ensure that bugs did not make it into the master branch to begin with thanks to peer reviews.

In our peer reviews, we tested incorrect inputs such as: negative numbers, invalid letters and strings, and empty input. We also tested options that would result in a query returning no rows (for example, selecting a paper with no reviewers in task 1). We initially encountered a NoneType error when using the `pandas` API with this last testcase. Upon investigation, we found that `pandas` has the following complication when were no reviewers returned:

- "sql will return the rowcount for the call, which will come back without a column name, causing the NoneType error" (https://stackoverflow.com/questions/30534095/nonetype-object-is-not-iterable-error-in-pandas)

We fixed this by using a cursors instead of directly accessing dataframes. Since the provided database did not contain enough data to complete peer reviews, the following additional data was used:

- A dataset that extends the data provided in vanilla_data.sql, which was called `modded_data.sql`
- A dataset that extends the modded dataset, purely for the purpose of testing task 5: `more_papers.sql` (since there were less than 5 regions in the vanilla and modded datasets)
- An empty database called `emptydbtest.db`

We used the provided database schema to build our databases. It is contained in `tables.sql`

Group Work Break-Down

Git was used for project versioning, allowing each project member to work on their own feature branches, and merge those features to a master branch later. This enabled us to keep track of who did what in a very exact manner.

To partition work, our Git project was hosted on GitHub, where tasks were made into Github issues. These issues were assigned to project members at random, but evenly.

Communication was facilitated through Slack and Discord for group calls.

This report was created using Google Docs to enable easy, equal collaboration.

Work distribution:

- Nicholas Wengel:
    - Q3 and Q6 SQL and Python (20 hrs)
        - 3.sql, 6.sql, functions task3(), task6() in main.py
- Hugh Bagan:
    - Q2 and Q5 SQL and Python (20 hrs)
        - 2a.sql, 2b.sql, q5.sql, functions task2(), task5() in main.py
- Aida Radu:
    - Q1 and Q4 SQL and Python (20 hrs)
        - 1a.sql, 1b.sql, 4a.sql, functions task1(), task4() in main.py
- All group members worked on:
    - main() function in main.py
    - Report
    - Testing (peer reviews), bug fixes

It’s unclear how time spent on tasks could be measured since the group did all work remotely. As well, it is likely that most tasks were not completed in one sitting, but were attacked slowly over the assignment period. As such, hour estimates are approximate.