CMPUT 291 Assignment 4 Design Document

Nomar Chavez
Adrian Leung
Nathan Vandermolen-Pater

March 26, 2019

Table of Contents

1. System overview and user guide

This system guides users, using a command line interface, through different tasks that parse data from City of Edmonton's Open Data initiative, a platform used to promote transparency and innovation by providing public access to hundreds of datasets, in a database and provides services to users. This design document will provide a brief overview of the layout of the system and will guide users through each individual tasks.

The system consisted of many different driving components, most notably, the libraries used. The libraries used in this system consists of pandas, matplotlib, sqlite3, os, webbrowser and folium. Pandas provide high-performance, easy to use data structures and data analysis tools. Pandas was used to analyze the data received from the SQLite database for Task 1. Matplotlib is a 2D plotting library that produces quality figures from data; it was used to create the plot in Task 1 as well. Sqlite3 is a small and versatile SQL database engine. Sqlite3 was used to interpret data from a database. Folium supports visualization of data on interactive Leaflet maps. Os and webbrowser support opening of created html file maps in a web browser.

In order to be user and developer friendly, the data flow between each individual function is clean, well documented, and concise. Each individual functions were kept in individual files to allow easy editing, testing, and collaboration. Functions were then imported into the main program for further use. The menu is kept in the main program but isolated into its own function for easy editing and testing. The main function creates a Sqlite3 connection and cursor to the database and passes these to each task function.

Before executing the program, ensure that the database is in the same directory as the program. To execute the program, run `python3 main.py <database>`, where `<database>` is the name of the database, or `python3 main.py` to connect to the default database, `a4-sampled.db`, in the command line. Example: `python3 main.py a4-sampled.db`. Afterward, a menu will show with the following options:

```
=== Assignment 4 Project ===
(1) Month-wise total count of certain crime type
(2) Map most/least populous neighbourhoods
(3) Top neighbourhoods and their crime count of certain crime type
(4) Top neighbourhoods with the highest crimes to population ratio

(q) Quit

Enter command:
```

Using a keyboard, the user can choose from the options provided. Each individual functions will have their own implementation of this menu. Follow the instructions provided in each menu. For tasks that open a chart, the user can close the chart by clicking the ‘X’ in the top left or right (depending on operating system) corner of the window and return back to the menu. For tasks that create a map, the map is opened in the default web browser. Maps include bubbles to visualize neighbourhoods and various neighbourhood properties, according to the specifications of the task implemented. These bubbles can be selected to display pop-up information about the neighbourhood, according to the requirements of the task.

2. Software design

2.1. Main function

The main function connects and disconnects the database, prints out a menu, and calls each individual functions. The path of the database is assumed to be in the same directory as the program. If database was provided as a command line argument, the program will use that database. If not, the program will use the default database a4-sampled.db. The function displays the menu with all possible options and accepts input from 1 to 4 and the letter ‘q’. All other inputs are rejected and the menu is displayed again. After quitting the menu, the main function commits any changes to the database and then terminates the connection with the database before exiting the program.

2.2. Task 1

Given a range of years and crime type, task 1 will generate and display a bar graph of the total number of the given crime type per month in the given range of years in Edmonton. If the start year is greater than the end year, the user will be prompted by a message and asked to try again. If an error occurred while querying or there was no data to plot on the bar graph, an error message will show and ask the user to try again or quit the task. If all successful, a bar graph will be displayed and a figure of the bar graph will be saved in the current directory as Q1-n.png, where n is the number of times this task was called.

2.3. Task 2

Given an integer N, task 2 will generate and display a map of the N-most populous and N-least populous neighbourhoods by their population count in Edmonton. All neighbourhoods will no latitude, longitude, and population are ignored. If the given integer N is greater than the number of neighbourhoods or the sum of the N-most populous and N-least populous neighbourhoods is greater than the number of neighbourhoods, the map will display all neighbourhoods. If the above condition is not met, the map will display the N-most populous neighbourhoods in red and the N-least

populous neighbourhoods in blue. Clicking on the bubbles will reveal the neighbourhood name and population count. If an error occurred while executing the task, an error message will show and ask the user to try again or quit the task. If all successful, a map will be opened in the browser and saved in the current directory as Q2-n.html, where n is the number of times this task was called.

2.4. Task 3

Task 3 takes 4 inputs: two years, representing a range; an integer N; and a string representing crime type. This task creates and displays a map of the top-N neighbourhoods in terms of the number of the specified type of crime in the range of years specified. This map is saved in the current path according to the naming convention Q3-*n*.html, where *n* is the number of times this task was called. Upon any erroneous input, an error message is displayed and the option to try different input is provided.

2.5. Task 4

Task 4 takes 3 inputs from the user: two different years (representing a range), and an integer N. When the user's input is valid, the task creates and displays a map of the top-N neighbourhoods with the highest crimes to population ratio within the provided range. Also, the task shows the most frequent crime type in each of these respective neighbourhoods. If there is an error when receiving user input and/or processing the data, the program prompts the user that an error has occurred and whether they want to continue with the current task or quit. The map will be displayed through a browser that pops up and saved in the current directory as Q4-nhtml, where n is the number of times this task was called.

3. Testing Strategy

All testing involved the sample database provided. Testing for individual tasks was done as the code was being written; this involved, in some cases, checking the output of SQL queries before outputting the result as a map visualization or plotting the results on a graph. Once tasks were completed, testing was done using the sample database on each task individually; this involved manually inspecting the database contents using a database visualizer such as DB Browser for SQLite, examining the expected output of the task, running the task's code to ensure accuracy, and error checking with improper inputs. When all tasks were completed and corroborated into the same program, final testing was done on each task to ensure accuracy.

4. Group work break-down

The group members used GitHub, a web-based hosting service for version control, to efficiently distribute and collaborate with one another. Nathan completed the code for task 3, created the layout for this design document and wrote sections 2.4 and 3. Adrian completed the code for tasks 1 and 2, implemented the main function, menu, and dataflow between functions and the main function, and wrote sections 1, 2.1, 2.2, and 2.3. Nomar completed the code for task 4 and wrote section 2.5. Each group member looked over the others' code to ensure accuracy and performed final editing on this design document. The group met twice for an hour each time, to discuss the project and divide responsibilities. Each group member spent approximately 3 hours completing their portions.