1. Overview

When the application is first opened, the user is prompted to enter the database file. A function\_count array is initialized to keep track the number of times a function is executed. Then the user is prompted to enter a number between 1-4 to perform the corresponding task. Once the task is preformed, the program exits the function and prompts the user again. This loops until the user enters ‘e’ or ‘exit’ to stop the program.

1. Design

Task 1  
Task1function()

Implementation

Task 2   
Task2function()

Implementation

Task 3  
Task3function()

This tasks prompts the user to enter a range for the starting year, end year, crime type and number of neighbourhoods. It then creates and saves an html map page with markers of the top-N neighbourhoods with their crime count where the given crime type occurred most within the given range.

Implementation

First, the map is instantiated. Then, user input is retrieved for the years, crime type, and number of neighbourhoods. These values are used in an SQL query that returns the top-N neighbourhoods with their crime count where the given crime type occurred most within the given range (accounting for ties). Then for each row in the returned data, a circle is created in the folium map. Lastly, This map is saved.

Task 4  
Task4 function()

This tasks prompts the user to enter a range for the starting year, end year, and then asks for the number of N neighbourhoods. It then creates an html map page with markers of the top-N neighbourhoods with the highest crime to population ratio.

Implementation:

First, the map is instantiated. Then, use the python input() function to retrieve the starting year, end year, and N number of neighbourhoods and convert it to an integer. Then, use the sqlite3 component to execute an SQL query that fetches neighbourhoods and the number of crime incidents that occurred within the range specified. It is sorted with the highest ratio descending. Then, depending of the N specified, neighbourhoods = rows[0:N] will retrieve the top-N neighbourhoods. Then, a for loop is used for every x in neighbourhood. For every x, execute an SQL query to fetch the most frequent crime in that neighbourhood, then use the folium component to create a marker on the map. Finally, after the for loop is finished, save the map and exit the function.

Main function

main()

This function loads the database from the inputted name and creates an array to store the run counts of each function. It then prompts users to choose a task and executes the respective function while also updating the run count. When the user chooses to exit, the connection is closed and the program ends.

1. Testing Strategy

Task 3 was tested using the sample database provided on eclass. The code was tested with various values for years and number of areas. The results of these were compared to the results from DBBrowser. A bug was encountered when N exceeded the number of neighbourhoods and was fixed.

To test whether Task 4 works, used the sample database provided to test the SQL queries using DBBrowser for SQlite and compared the results with the program to check if they matched. Then we checked the folium map created to see if the markers matched with the results.

1. Group-Work Strategy

Split Tasks 2, 3, & 4 among the group members as they were the harder tasks. Li Shang is responsible for Task 2, William Wei is responsible for task 3, Gustavo Ortega is responsible for task 4. Everyone contributed to the report file. Li Shang finished Task 1.

1. Time Spent and Progress

Gustavo Ortega spent around 4-5 hours on Task 4 and the skeletal framework of the code. Started on Wednesday March 27th and finished on Friday March 29th.

William spent around 3 hours on Task 3. Started Wednesday March 27th, finished Thursday Math 28th

The design document was worked on by all three members and finished on Saturday March 30th.

Method of Communication:

Github was used to keep track of progress while Discord was used for discussion between group members.