SI507 Project Document

Siyuan He (hesy)

Project code

Code Repo Url: https://github.com/sweetsinpackets/si507_project

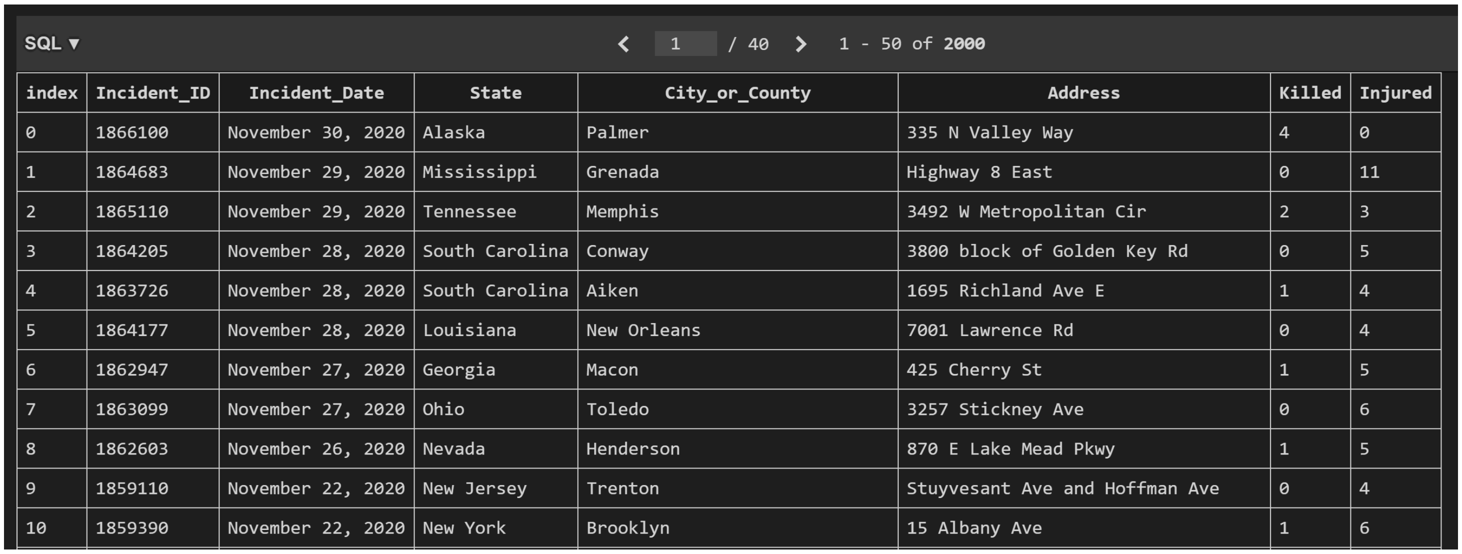
The readme is at the root directory of the code base, here is meaningless to copy it here.

Data Source

1. **Main Page**, GVA Reports, HTML format
   The page contains a list of href to subpages. The program will first crawl the basic page and list all the available reports for user to choose.
   The function is `data_fetch.crawl_main_page` , returns a `dict` of `{page name: page link}` .
2. **Sub-report pages**, all pages listed in basic page, HTML format
   For example, the biggest subpage massive shooting reports contains 2000 records, while other subpages have less records (still several hundred). We will retrieve all records on-demand, so the requesting time will be quite long (1 minute).
   These pages will be crawled **on-demand** by user selection. Note that each subpage contains a sequence of pages, we will recursively crawl and combine all these sequential pages.
   The function is `data.fetch.multiple_scrape` , return a dataframe of all these records.
   Side Note: The retrieved records are originally organized as `class_definition.shooting_record` , you can directly check the class definition. However, since we have 2000 records, and it will take 2 minutes to execute the request and instance generating, so I turned to use pandas dataframe because it's faster to use `.apply` function. The organization is similar to the class (just as the class initialization by a list), please refer to `class_definition.transfer` .
3. **MapQuest API request**
   We use MapQuest API to search for the lat and lng by the address.
4. Cache: all the requests are cached.
   Note that since the website will update, so we refresh our cache every month. The API request cache won't update because they're stable.

Database

All records of a selected report will be saved to a library. For example, the screenshot is a table of massive shooting record in 2020.



index	Incident_ID	Incident_Date	State	City_or_County	Address	Killed	Injured
0	1866100	November 30, 2020	Alaska	Palmer	335 N Valley Way	4	0
1	1864683	November 29, 2020	Mississippi	Grenada	Highway 8 East	0	11
2	1865110	November 29, 2020	Tennessee	Memphis	3492 W Metropolitan Cir	2	3
3	1864205	November 28, 2020	South Carolina	Conway	3800 block of Golden Key Rd	0	5
4	1863726	November 28, 2020	South Carolina	Aiken	1695 Richland Ave E	1	4
5	1864177	November 28, 2020	Louisiana	New Orleans	7001 Lawrence Rd	0	4
6	1862947	November 27, 2020	Georgia	Macon	425 Cherry St	1	5
7	1863099	November 27, 2020	Ohio	Toledo	3257 Stickney Ave	0	6
8	1862603	November 26, 2020	Nevada	Henderson	870 E Lake Mead Pkwy	1	5
9	1859110	November 22, 2020	New Jersey	Trenton	Stuyvesant Ave and Hoffman Ave	0	4
10	1859390	November 22, 2020	New York	Brooklyn	15 Albany Ave	1	6

The fields of the table is corresponding to the fields of the reports on GVA. Except for `Incident_ID, Killed, Injured` which are integer, other fields are stored as string.
No foreign keys among the tables. It's because each report contains a different set of records, so each table is parallel to other tables. We don't need to connect them.
Side Note: In the program we need all these structures, so I used `pandas` to organize the records. For convenience, the database is created and loaded by the build-in functions of dataframe and sqlite3.

Interaction and Presentation Plans

We provide both command line interaction and a map plot for presentation. The users will see a list of record information printed (surely human-friendly) in the command line interface as text output. Meanwhile, user can require a map plot to clearly see the distribution of the records on map. When executing the program, user will select the records they want by typing commands in the command line interface.

1. **Command Line Prompt**:
   It's designed as every common command line interface program. It will ask the user to input command and check the user's input. The records will be displayed as a list of human readable string. If there are more than 10 records, we only show the top 10 records.
2. **Map Plot**:
   We offer a map plot with dots representing shooting cases on the US map. The plot is implemented by `Plotly`, while the lat and lng are accessed from `MapQuest`. The plot is optional and only generated on-demand. Meanwhile, the plot will be saved as an html file in the root directory and automatically opened.

Demo Video

The demo video can be accessed by the line:
https://drive.google.com/file/d/1BHqyXvqiHKRn0CABAZ08dNKZA4HOA_Q8/view?usp=sharing
Note that only UM accounts can access the link.