

# Data analysis, fragmentation and aggregation for 311 calls in Python & SQL using localhost

Zak Graham Rackham

**Abstract**—Methods used for data analysis considering a data set of 311 calls in NYC for non-emergency government services and information. Python and SQL are applied to conduct statistical analysis using functional and declarative approaches respectively considering performance, ease of use and readability. Hypotheses are drawn to direct the analysis toward meaningful conclusions and correlations, causation's are considered within the data context.

**Index Terms**—Class, IEEEtran, L<sup>A</sup>T<sub>E</sub>X, paper, style, template, typesetting.

## I. INTRODUCTION

THE main analytic approach reviewed is python .csv evaluation using a linear time algorithm, its time complexity is proportional to the data set  $O(n)$ , this method looks at each row of data as a single entity. A constant  $K$  is introduced for each search condition  $O(Kn)$ .

To work with large sets of data fragmenting will be used to process queries, this will allow greater access speed and makes analysis more manageable from a computational perspective.

More optimised searching is introduced for faster set returns. Data is collated and visualised.

## II. HYPOTHESES

The data set reviewed is likely to contain null values, the set reviewed is sourced from a governing body second hand via a third party that catalogues this data making it available to the public. An API is provided by Socrata Open Data API (SODA) for filtering querying and aggregation. Data types are specified relative to this API, therefore conclusions on data types to be implemented locally must be made.

Corrupted data is unlikely but viable. Integrity of the set can only be assumed in this case. Data is not fragmented since one source is provided.

Full analysis will be possible applying SQL & Python. Since corruption is possible edge cases must be considered. SQL can be used to retrieve data from the database faster, but Python offers more flexibility by allowing you to manipulate and perform computations with the retrieved data.

The number of calls increases from March to August.

- $H0$  Total number of calls from summer month A is less than spring month B.  $A < B$ .
- $H1$  Total number of calls from summer month A is greater than spring month B,  $A > B$ .

## III. COMPARISON

### A. Performance

Python is generally slower for extensive computations. Multi-threading can be used to mitigate this to an extent, since

python is semi multi threaded due to a global interpreter lock (GIL) only certain library's using C-based implementations.

SQL has faster performance for simple queries and aggregations, google analytics can utilise multiple SQL data sources for aggregation.

### B. Functionality

Extensive functionality due to its integration with a wide variety of libraries. SQL less so due to ecosystem lock-ins.

### C. Testing

Python offers extensive unit and integration testing through the pipeline and code process. SQL is tested during production, and there are no extensive unit tests

### D. Ease of Use

Python has an easy-to-use syntax; however, there are multiple concepts to learn, which may increase difficulty. SQL is very beginner-friendly, with fewer concepts to learn

### E. Debugging

Debugging in Python is easier with breakpoints to help halt execution on encountering bugs. SQL is split into multiple files to help with debugging, but execution occurs at once with no breakpoints.

## IV. JUSTIFICATION

Python is crucial for roles like data scientists as it contains a range of libraries required to perform multiple tasks like data manipulation, wrangling, and exploration. Data engineers need extensive SQL skills for data modeling and ETL tasks. SQL also requires hosting to use tools such as Google Analytics.

## V. PYTHON

### A. File System

All files associated with the project are under the parent folder of Panaseer. Root is reserved for Python source files, data contains the subject and fragments of the subject.

## B. Coding Methods

The first function 0 (Read Data) is used to return .csv rows to the console. Logic is then used to loop through the entire set or a portion thereof. GUI is handled in the main function and thread.

Separate Query functions are created for various purposes. These functions store fragments of the data in associated .csv files.

Figure 1 shows the selection cases for finding the winter month and summer month in question. Each of these return data fragments to their own respective .csv files and append the UIN for each entry to an array for searching and counting.

```
#cases for data selection
match select:
    case 0:
        print ('no query')
    case 1:
        x = re.search("^03.*$", str(row[1]))
        if x:
            print('HIT')
            fragment1.writerow([row[0], row[1]])
            arr_f1.append(int(row[0]))
    case 2:
        x = re.search("^08.*$", str(row[1]))
        if x:
            print('HIT')
            fragment2.writerow([row[0], row[1]])
            arr_f2.append(int(row[0]))
    case 3:
        csv_writer.writerow(['0', '1', '0'])
```

Fig. 1. Selection cases using regular expressions

Binary search is implemented to search by UIN as required by the user.

GUI is handled by the tkinter library in the main thread, separate threads are used for running count functions, and the main read function 0.

Pyplot is used for data aggregation and displays a graph at the end of execution.

## VI. SQL

### A. Hosting

MS SQL Server 2022 is hosted locally using docker virtualization. This uses a local host for connecting to run queries.

Azure Data Studio is used to connect to the SQL server self-contained instance, this is used so the server does not have direct access to the filesystem of the host machine.

```
zak@Zaks-MacBook-Pro ~ % docker run -e "ACCEPT_EULA=Y" -e "MSSQL_SA_PASSWORD=Password150132" \
-p 1433:1433 --name sa --hostname sa \
-d \
mcr.microsoft.com/mssql/server:2022-latest
```

Fig. 2. Command to install the SQL Server image and container

### B. Database Management System

The Azure Data Studio is used to manage the SQL Server database. First, the database is created, then the table for the data to be inserted is created. The BULK INSERT command can be used to insert .csv formatted data.

```
1 BULK INSERT dbo.db0
2 FROM '/311_Service_Requests_from_2010_to_Present_20231023.csv'
3 WITH
4 (
5     FORMAT='CSV',
6     FIRSTROW=2,
7     ROWTERMINATOR = '0x0a',
8     FIELDTERMINATOR = ',',
9     TABLOCK --Requirement for minimal logging
10    --LASTROW = 102
11 )
12 GO
```

Fig. 3. Command to import the dataset

As this database is being hosted locally running into a storage limit can happen quickly when importing large sets in full recovery mode. In order to keep the load on storage as low as possible recovery needs to be set to simple.

## VII. RESULTS

A smaller set is considered for a faster data return to test the algorithms and to check for validity. To check the validity a comparison can be made via a SQL query on the same set.

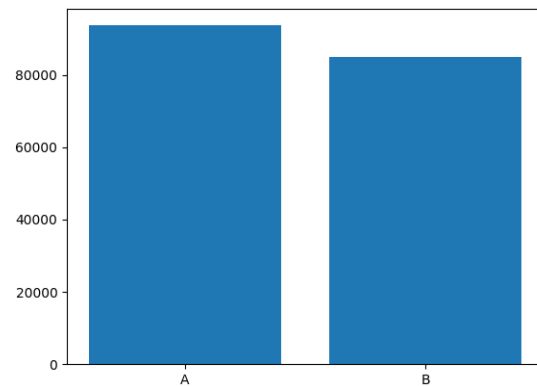


Fig. 4. Number of calls (reduced set of one million)

The set clearly shows that there is an increase in calls in August over March

## VIII. EVALUATION

The final artifact developed shows promise of being able to grow to complete analysis to achieve a larger set of hypotheses. Python is an excellent tool for data analysis however managing large sets can be a challenge, especially in terms of development time. When embarking on a larger data science problem in the future SQL will be used instead where the data is well organized.

Python allows flexibility, this is reliant on the architecture being well-designed before the code implementation. Making justified assumptions will aid this process. Moving forward with the project deliverable and aim need to be considered within the context with greater rigor in order to develop an environment in Python that allows the flexibility required to tackle more complex aggregation.

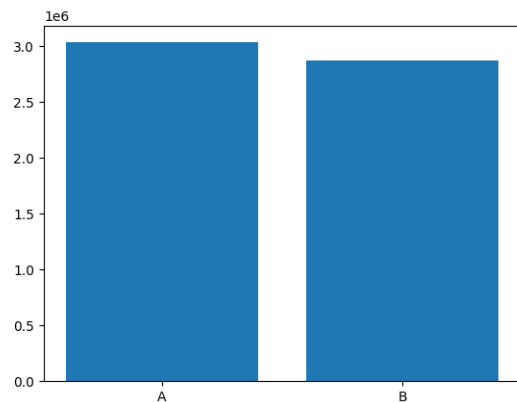


Fig. 5. Number of calls (entire set)

## IX. CONCLUTIONS

In the context of this data set the hypothesis  $H1$  was correct that calls increased during March compared to August. Further analysis is necessary to draw a complete conclusion and correlations. Employing a more object-orientated approach would be beneficial for recovery and other improvement aspects. Optimization is necessary for faster computation using pandas.

Since the data is unordered, ordering the data by UIN or date would be the next step for a stronger analysis.