



**Module Code & Module Title**

**CC5061NI Applied Data Science**

**60% Individual Coursework**

**Submission : Final Submission**

**Academic Semester: Spring Semester 2025**

**Credit: 15 credit semester long module**

**Student Name: Ekata Baral**

**London Met ID: 23048589**

**College ID: NP01AI04A230015**

**Assignment Due Date: Thursday, May 15, 2025**

**Assignment Submission Date: Thursday, May 15, 2025**

**Submitted To: Alish KC**

*I confirm that I understand my coursework needs to be submitted online via MST Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.*

# 23048589-Ekata ADS Final.docx

 Islington College, Nepal

## Document Details

Submission ID

trn:oid::3618:95890185

Submission Date

May 14, 2025, 9:20 PM GMT+5:45

Download Date

May 14, 2025, 9:23 PM GMT+5:45

File Name

23048589-Ekata ADS Final.docx

File Size

36.9 KB

36 Pages

6,186 Words

30,828 Characters



Page 2 of 41 - Integrity Overview

Submission ID trn:oid::3618:95890185

## 9% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

### Match Groups



47 Not Cited or Quoted 9%

Matches with neither in-text citation nor quotation marks



1 Missing Quotations 0%

Matches that are still very similar to source material



0 Missing Citation 0%

Matches that have quotation marks, but no in-text citation




0 Cited and Quoted 0%

Matches with in-text citation present, but no quotation marks

### Top Sources

1%  Internet sources

1%  Publications

9%  Submitted works (Student Papers)

## Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

## Table of Contents

Table of Figure .....	4
Table of Table .....	5
1. Data Understanding .....	1
2. Data Preparation .....	6
2.1. Importing the Dataset.....	6
2.2. Provide your insight on the information and details that the provided dataset carries.....	7
2.3. Convert the columns “Created Date” and “Closed Date” to datetime datatype and create a new column “Request_Closing_Time” as the time elapsed between request creation and request closing.....	10
2.4. Write a python program to drop irrelevant Columns.....	13
2.5. Write a python program to remove the NaN missing values from updated dataframe. ....	15
2.6. Write a python program to see the unique values from all the columns in the data frame. ....	17
3. Data Analysis .....	20
3.1. Write a Python program to show summary statistics of sum, mean, standard deviation, skewness, and kurtosis of the data frame. ....	20
3.2. Write a Python program to calculate and show correlation of all variables. ....	25
4. Data Exploration.....	27
4.1. Provide four major insights through visualization after data mining.....	28
4.1.1. Frequency of Complaints.....	28
4.1.2. Complaints by Borough .....	31
4.1.3. Monthly Complaint Tracking .....	33
4.1.4. Geographical Distribution of Complaints.....	35
4.2. Arrange the complaint types according to their average 'Request_Closing_Time', categorized by various locations. Illustrate it through graph as well.....	37
5. Statistical Testing .....	41
5.1. Test 1 .....	41
5.2. Test 2 .....	43
6. Conclusion .....	45
References.....	46

## Table of Figure

Figure 1. Importing pandas and the csv file.....	6
Figure 2. List of columns in the data frame .....	8
Figure 3. Data in the data frame.....	9
Figure 4. To change the datatype and add Request_Closing_Time column .....	10
Figure 5. Changing data type of Created Date and Closed Date .....	11
Figure 6. Column "Return_Closing_Time" is added .....	12
Figure 7. Dropping irrelevant columns.....	13
Figure 8. Irrelevant columns dropped.....	14
Figure 9. Counting all the null values in the data.....	15
Figure 10. Dropping the missing values from the table .....	16
Figure 11. Recounting all the null values in the data .....	16
Figure 12. Finding unique values .....	18
Figure 13. Finding unique values (2).....	18
Figure 14. Finding unique values (3).....	19
Figure 15. Finding unique values (4).....	19
Figure 16. Data described with sum, mean and standard deviation.....	20
Figure 17. Skewness of all the columns with numerical data .....	22
Figure 18. Skew for Request Closing Time column .....	22
Figure 19. Kurtosis of all the columns with numerical data.....	23
Figure 20. Kurtosis of Request Closing Time column.....	24
Figure 21. Correlation between the variables.....	25
Figure 22. Code to find frequency of complaints .....	28
Figure 23. Graphs showing the frequency of complaints.....	29
Figure 24. Complaints made based on Borough .....	32
Figure 25. Code for monthly complaints tracking .....	33
Figure 26. Graph of Complaints based on month.....	34
Figure 27. Code for the scatter plot of Geographic Distribution of Complaints.....	35
Figure 28. Scatter plot of Geographical Distribution of Complaints .....	36
Figure 29. Code for Complaint Types based on Location and Closing time .....	37
Figure 30. Complaint Type Based on Location: Bronx .....	38
Figure 31. Complaint Type Based on Location: Brooklyn and Manhattan.....	39
Figure 32. Complaint Type Based on Location: Queens and Staten Island .....	40
Figure 33. Test 1 Code and Result .....	42
Figure 34. Chi square test to check the complaint and location are related .....	44

**Table of Table**

Table 1. Table with column description .....	5
----------------------------------------------	---

## 1. Data Understanding

Data understanding is the first step for data analysis. The step involves searching the relevant data through legitimate means to extract the data required for the analysis. The found data is then extracted into a database. Data understanding deals with identifying the data currently with you, determining the tools to be used for further data collection and where to extract the data from. It is crucial to get these four questions before any project involving data: what kinds of data you require, where to get the data, how you will fetch the data, and how much data you are going to acquire (Luna, 2021).

The data set for this project is fetched from the 311 service requests of the New York City from 2010 to the present time. This contains the data of the complaints made by the New York City residents based on the inconveniences faced on a daily basis. The data was provided to analyse the collected data and to test it against various cases. The data from a century ago may not be relevant for the analysis of the current situation but the data used in this project can represent the trend of these complaints in the near future, making it relevant for this project.

The given 53 columns contain every detail of the complaint from the complaint type, location type to the dates, like creation date, closed date, resolution action update date and due date. The majority of the columns in the data present the relative location of the place where the complaint is being registered about. Columns like incident address, street name, cross street name, intersection street name, city, school region, taxi pick up location, longitude, latitude, location and so on. Most of these columns are related data about the place of complaint making the data highly redundant.

The X coordinate and Y coordinate state plane gives the relative coordinate of the complaint place in a slightly different format than the latitude and longitude of the same. These columns store the exact coordinate details of the place of complaint in two different

measurements, one with the US-specific coordinates detailing it in a more accurate and precise manner (US Geological Survey, 2024) and the other in the standard internationalized format.

During various phases of data preparation, a lot of these columns are expected to be dropped to facilitate only relevant data to remain for further analysis. The data collected here will help analyse what system should be in place to reduce such future inconveniences. The data collected also contains the information on the community boards and agency taking care of the complaint, location of the places which are relevant in working out the mitigating measures for the future.

S.N.	Column Name	Description	Data type
1.	Unique Key	A unique identification for each complaint.	Int64
2.	Created Date	The initial date for when the complaint was created.	Object
3.	Closed Date	The date for when the complaint was closed.	Object
4.	Agency	The abbreviated name of agency handling the complaint.	Object
5.	Agency Name	The full name of the agency.	Object
6.	Complaint Type	The type of complaint made by the commoners.	Object
7.	Descriptor	The description of the complaint.	Object
8.	Location Type	The location type e.g.: street, sidewalk, etc.	Object
9.	Incident Zip	The ZIP code of the place of incident.	Float64
10.	Incident Address	The full address of the place of incident.	Object

11.	Street Name	The name of the street where the incident occurred.	Object
12.	Cross Street 1	The name of the cross street where the incident occurred.	Object
13.	Cross Street 2	The name of the cross street where the incident occurred.	Object
14.	Intersection Street 1	The name of the first intersection from the incident.	Object
15.	Intersection Street 2	The name of the second intersection from the incident.	Object
16.	Address Type	The type of address whether “address” or “intersection”.	Object
17.	City	The name of the city where the incident took place.	Object
18.	Landmark	One known landmark near the place of incident.	Object
19.	Facility Type	The facility type involved in the incident.	Object
20.	Status	The status of the complaint, whether opened or closed.	Object
21.	Due Date	The date by which the complaint was supposed to be resolved and closed.	Object
22.	Resolution Description	The description of the solution finalized.	Object
23.	Resolution Action Updated Date	The final date by which the resolution was last conducted.	Object
24.	Community Board	The information about the boards involved with running the community.	Object
25.	Borough	The information about the district where the incident took place.	Object



26.	X Coordinate (State Plane)	The x coordinate for the place of incident.	Float64
27.	Y Coordinate (State Plane)	The y coordinates for the place of incident.	Float64
28.	Park Facility Name	The name of the park located near the place of incident if any.	Object
29.	Park Borough	The borough name of the park near the place of incident if any.	Object
30.	School Name	The name of the school situated near the place of incident.	Object
31.	School Number	The number of the school situated near the place of incident.	Object
32.	School Region	The region of the school situated near the place of incident.	Object
33.	School Code	The code of the school situated near the place of incident.	Object
34.	School Phone Number	The phone number of the school situated near the place of incident.	Object
35.	School Address	The address of the school situated near the place of incident.	Object
36.	School City	The city where the school is situated.	Object
37.	School State	The state in which the school lies.	Object
38.	School Zip	The ZIP code of the school.	Object
39.	School Not Found	The information on whether a school was in the area or not.	Object
40.	School or Citywide Complaint	The complaint based on school or city.	Float64
41.	Vehicle Type	The type of the vehicle involved during the incident, if any.	Float64

42.	Taxi Company Borough	The name of the taxi company involved during the incident, if any.	Float64
43.	Taxi Pick Up Location	The location of pick-up for the taxi if any.	Float64
44.	Bridge Highway Name	The name of the bridge or the highway involved in the incident, if any.	Object
45.	Bridge Highway Direction	The direction of the bridge or the highway involved in the incident, if any.	Object
46.	Road Ramp	The ramp involved in the incident, if any.	Object
47.	Bridge Highway Segment	The segment of the bridge or the highway involved in the incident, if any.	Object
48.	Garage Lot Name	The garage name involved in the incident if any.	Float64
49.	Ferry Direction	The direction of the ferry involved in the incident, if any.	Object
50.	Ferry Terminal Name	The name of the ferry terminal involved in the incident, if any.	Object
51.	Latitude	The latitude of the place of incident.	Float64
52.	Longitude	The longitude of the place of incident.	Float64
53.	Location	Both the longitude and the latitude of the place of incident.	Object

*Table 1. Table with column description*

## 2. Data Preparation

Data Preparation involves preparing the extracted data for further data analysis. The data is then arranged according to the required columns for the analysis. The unnecessary columns and missing values are removed in this step. Some columns are merged together and some are distributed according to the requirements of the data and what is going to be analysed.

### 2.1. Importing the Dataset

The Python Pandas library has been imported to create a Pandas data frame to access and store the data from the given csv file for carrying the further process of preparing, analysing and exploring the data. Pandas is a powerful python library used to for data analysis of a large volume of data (Pandas, 2024).



```
[2]: import pandas as pd

[3]: df = pd.read_csv("Customer_Service_Requests_from_2010_to_Present.csv")
```

*Figure 1. Importing pandas and the csv file*

The data was imported by:

- Importing pandas library in the python kernel.
- The CSV file, was then stored in the data frame named df by accessing it through the pandas library using read\_csv function offered by the pandas library.

## **2.2. Provide your insight on the information and details that the provided dataset carries.**

The provided dataset contains the data of the customer service requests of New York City from 2010 to the present time, as per the heading but after going through the data as done below, it is clear that the data in our dataset only contains the data from the year 2015. It contains 53 columns storing various data related to the complaint. The data contains a lot of columns dealing with the location of the incident and other various information about the complaint. The presented data contains all the non-emergency complaints from New York City, varying from noise complaints to complaints about blocked driveway, parking sign violations, animal abuse and so on.

The `.info()` function has been used to describe the columns present in the dataset. This function shows all the columns in the dataset as well as their data types and the amount of non-null values in them.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300698 entries, 0 to 300697
Data columns (total 53 columns):
 #   Column                                     Non-Null Count  Dtype
---  -
 0   Unique Key                               300698 non-null  int64
 1   Created Date                             300698 non-null  object
 2   Closed Date                              298534 non-null  object
 3   Agency                                   300698 non-null  object
 4   Agency Name                             300698 non-null  object
 5   Complaint Type                           300698 non-null  object
 6   Descriptor                               294784 non-null  object
 7   Location Type                            300567 non-null  object
 8   Incident Zip                             298083 non-null  float64
 9   Incident Address                         256288 non-null  object
10   Street Name                             256288 non-null  object
11   Cross Street 1                           251419 non-null  object
12   Cross Street 2                           250919 non-null  object
13   Intersection Street 1                     43858 non-null  object
14   Intersection Street 2                     43362 non-null  object
15   Address Type                             297883 non-null  object
16   City                                     298084 non-null  object
17   Landmark                                 349 non-null    object
18   Facility Type                             298527 non-null  object
19   Status                                   300698 non-null  object
20   Due Date                                 300695 non-null  object
21   Resolution Description                    300698 non-null  object
22   Resolution Action Updated Date            298511 non-null  object
23   Community Board                          300698 non-null  object
24   Borough                                  300698 non-null  object
25   X Coordinate (State Plane)                297158 non-null  float64
26   Y Coordinate (State Plane)                297158 non-null  float64
27   Park Facility Name                       300698 non-null  object
28   Park Borough                             300698 non-null  object
29   School Name                              300698 non-null  object
30   School Number                           300698 non-null  object
31   School Region                            300697 non-null  object
32   School Code                             300697 non-null  object
33   School Phone Number                      300698 non-null  object
34   School Address                           300698 non-null  object

35   School City                               300698 non-null  object
36   School State                             300698 non-null  object
37   School Zip                               300697 non-null  object
38   School Not Found                         300698 non-null  object
39   School or Citywide Complaint             0 non-null      float64
40   Vehicle Type                             0 non-null      float64
41   Taxi Company Borough                     0 non-null      float64
42   Taxi Pick Up Location                    0 non-null      float64
43   Bridge Highway Name                       243 non-null    object
44   Bridge Highway Direction                  243 non-null    object
45   Road Ramp                                213 non-null    object
46   Bridge Highway Segment                    213 non-null    object
47   Garage Lot Name                           0 non-null      float64
48   Ferry Direction                           1 non-null      object
49   Ferry Terminal Name                       2 non-null      object
50   Latitude                                 297158 non-null  float64
51   Longitude                                297158 non-null  float64
52   Location                                 297158 non-null  object

dtypes: float64(10), int64(1), object(42)
memory usage: 121.6+ MB
```

Figure 2. List of columns in the data frame

The data contains a lot of location information representing the relative area the complaints are coming from. The location information vary from latitude and longitude to the school zip, bridge highway direction and so on. Majority of the irrelevant columns other than a few relevant ones indicating the location of the place where the complaint was being registered and the information on the complaint, its type, time constraints and the Agencies handling the request will be removed during the data preparation phase.

Though the file containing the data is labelled for the data between 2010 to the Present time, the data as extracted below shows the data of only one year, 2015. So the data presented within this project only encompasses the data from the year of 2015, March through December.

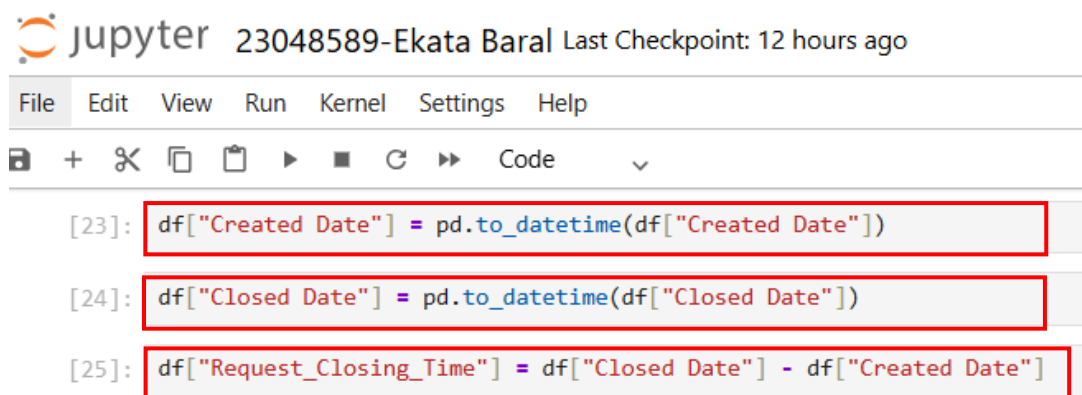
df																
	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	Incident Zip	Incident Address	...	Bridge Highway Name	Bridge Highway Direction	Road Ramp	Br Higl Segi	
	0	32310363	12/31/2015 11:59:45 PM	01-01-16 0:55	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Music/Party	Street/Sidewalk	10034.0	71 VERMILYEA AVENUE	...	NaN	NaN	NaN	
	1	32309934	12/31/2015 11:59:44 PM	01-01-16 1:26	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewalk	11105.0	27-07 23 AVENUE	...	NaN	NaN	NaN	
	2	32309159	12/31/2015 11:59:29 PM	01-01-16 4:51	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewalk	10458.0	2897 VALENTINE AVENUE	...	NaN	NaN	NaN	
	3	32305098	12/31/2015 11:57:46 PM	01-01-16 7:43	NYPD	New York City Police Department	Illegal Parking	Commercial Overnight Parking	Street/Sidewalk	10461.0	2940 BAISLEY AVENUE	...	NaN	NaN	NaN	
	4	32306529	12/31/2015 11:56:58 PM	01-01-16 3:24	NYPD	New York City Police Department	Illegal Parking	Blocked Sidewalk	Street/Sidewalk	11373.0	87-14 57 ROAD	...	NaN	NaN	NaN	
	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
10693	30281872	03/29/2015 12:33:41 AM	NaN	NYPD	New York City Police Department	Noise - Commercial	Loud Music/Party	Club/Bar/Restaurant	NaN	CRESCENT AVENUE	...	NaN	NaN	NaN		
10694	30281230	03/29/2015 12:33:28 AM	03/29/2015 02:33:59 AM	NYPD	New York City Police Department	Blocked Driveway	Partial Access	Street/Sidewalk	11418.0	100-17 87 AVENUE	...	NaN	NaN	NaN		
10695	30283424	03/29/2015 12:33:03 AM	03/29/2015 03:40:20 AM	NYPD	New York City Police Department	Noise - Commercial	Loud Music/Party	Club/Bar/Restaurant	11206.0	162 THROOP AVENUE	...	NaN	NaN	NaN		

Figure 3. Data in the data frame

The df here prints out all the data in the dataset as stored in the data frame df.

### 2.3. Convert the columns “Created Date” and “Closed Date” to datetime datatype and create a new column “Request\_Closing\_Time” as the time elapsed between request creation and request closing

The given columns “Created Date” and “Closed Date” are then assigned with the new data type using the `to_datetime` function provided by pandas in order to convert the formatting of the date to datetime format. This is to prepare the data to extract the amount of time passed between registering of the complaint until the time it was resolved.



```
jupyter 23048589-Ekata Baral Last Checkpoint: 12 hours ago
File Edit View Run Kernel Settings Help
+ ✂ 📄 📌 ▶ ■ ↺ ▶▶ Code ▼

[23]: df["Created Date"] = pd.to_datetime(df["Created Date"])

[24]: df["Closed Date"] = pd.to_datetime(df["Closed Date"])

[25]: df["Request_Closing_Time"] = df["Closed Date"] - df["Created Date"]
```

Figure 4. To change the datatype and add Request\_Closing\_Time column

jupyter 23048589-Ekata Baral Last C

File Edit View Run Kernel Settings Help

Code

[22]:

	Unique Key	Created Date	Closed Date	A
0	32310363	2015-12-31 23:59:45	2016-01-01 00:55:00	
1	32309934	2015-12-31 23:59:44	2016-01-01 01:26:00	
2	32309159	2015-12-31 23:59:29	2016-01-01 04:51:00	
3	32305098	2015-12-31 23:57:46	2016-01-01 07:43:00	
4	32306529	2015-12-31 23:56:58	2016-01-01 03:24:00	
...	...	...	...	
300693	30281872	2015-03-29 00:33:41	NaT	
300694	30281230	2015-03-29 00:33:28	2015-03-29 02:33:59	
300695	30283424	2015-03-29 00:33:03	2015-03-29 03:40:20	
300696	30280004	2015-03-29 00:33:02	2015-03-29 04:38:35	

Figure 5. Changing data type of Created Date and Closed Date

The Created Date is subtracted from the Closed Date to find the closing time of the complaint from the issued time to the time of its completion. The result is stored in “Return\_Closing\_Time” column. The column gives the time taken for the complaint to be resolved by the respective authority.



erry inal ame	Latitude	Longitude	Location	Request_Closing_Time
NaN	40.865682	-73.923501	(40.86568153633767, -73.92350095571744)	0 days 00:55:15
NaN	40.775945	-73.915094	(40.775945312321085, -73.91509393898605)	0 days 01:26:16
NaN	40.870325	-73.888525	(40.870324522111424, -73.88852464418646)	0 days 04:51:31
NaN	40.835994	-73.828379	(40.83599404683083, -73.82837939584206)	0 days 07:45:14
NaN	40.733060	-73.874170	(40.733059618956815, -73.87416975810375)	0 days 03:27:02

Figure 6. Column "Return\_Closing\_Time" is added

The data in the column shows that most complaints are solved within a day's time frame.

## 2.4. Write a python program to drop irrelevant Columns

Removing irrelevant columns and integrating relevant ones to get relevant information is an integral part of every data preparation phase. Only the necessary data required in the further stages of data analysis should remain.

The drop function is used to delete the columns from the dataset. The columns are specified inside the big brackets which contains all the irrelevant columns that needs to be deleted. The data is not changed in the dataframe permanently just yet so to permanently change the data inplace = True is used. This key word is used to change the underlying data and store the remaining data in the same dataframe. The alternative, inplace=False, only changes a copy of the data and leaves the original version intact.

```
df.drop(columns=['Agency Name','Incident Address','Street Name','Cross Street 1','Cross Street 2','Intersection Street 1','Intersection Street 2','Address','School State','School Zip','School Not Found','School or Citywide Complaint','Vehicle Type','Taxi Company Borough','Taxi Pick Up Location','Bridge Highway Segment','Road Ramp','Bridge Highway Segment','Garage Lot Name','Ferry Direction','Ferry Terminal Name','Landmark','X Coordinate (State Plane)','Y Coordinate (State Plane)','Due Date','Resolution Action Updated Date','Community Board','Facility Type','Location'], inplace=True)##inplace ensures that the drop is permanent
```

Figure 7. Dropping irrelevant columns

```
df.info()

<class 'pandas.core.frame.DataFrame'>
Index: 291107 entries, 0 to 300697
Data columns (total 15 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Unique Key                           291107 non-null int64
1   Created Date                          291107 non-null datetime64[ns]
2   Closed Date                           291107 non-null datetime64[ns]
3   Agency                                291107 non-null object
4   Complaint Type                        291107 non-null object
5   Descriptor                            291107 non-null object
6   Location Type                         291107 non-null object
7   Incident Zip                          291107 non-null float64
8   City                                  291107 non-null object
9   Status                                291107 non-null object
10  Resolution Description                 291107 non-null object
11  Borough                               291107 non-null object
12  Latitude                              291107 non-null float64
13  Longitude                             291107 non-null float64
14  Request_Closing_Time                  291107 non-null timedelta64[ns]
dtypes: datetime64[ns](2), float64(3), int64(1), object(8), timedelta64[ns](1)
memory usage: 35.5+ MB
```

*Figure 8. Irrelevant columns dropped*

The irrelevant columns are deleted leaving only the necessary columns with relevant data. From the 53 columns worth data, only 15 of the columns are left to carry out the data analysis.

## 2.5. Write a python program to remove the NaN missing values from updated dataframe.

Removing the missing values from the given data is an important step in getting the data ready, preparing it for further analysis as the missing values can create some problem in accurate data representation.

Here, all the missing values are found using the `isnull()` function. The missing columns do not give a well-rounded data so such fields are removed from the table leaving the table with only relevant data that are useful to do the further analysis. For dropping the missing values, `dropna()` function has been used along with `inplace = True`, replacing the original table with a data more equipped to facilitate better data analysis.

yter 23048589-Ekata Baral Last Checkpoint: 3 days ago

View Run Kernel Settings Help

Code

JupyterLab Python

### Write a python program to remove the NaN missing values from updated dataframe.

```
df.isnull().sum()
```

Unique Key	0
Created Date	0
Closed Date	2164
Agency	0
Complaint Type	0
Descriptor	5914
Location Type	131
Incident Zip	2615
City	2614
Status	0
Resolution Description	0
Borough	0
Latitude	3540
Longitude	3540
Request_Closing_Time	2164

dtype: int64

Figure 9. Counting all the null values in the data

```
df.dropna(inplace=True)
df
```

	Unique Key	Created Date	Closed Date	Agency	Complaint Type	Descriptor	Location Type	Incident Zip	City	Status	Resolution Description
0	32310363	2015-12-31 23:59:45	2016-01-01 00:55:00	NYPD	Noise - Street/Sidewalk	Loud Music/Party	Street/Sidewalk	10034.0	NEW YORK	Closed	The Police Department responded and upon arriv...
1	32309934	2015-12-31 23:59:44	2016-01-01 01:26:00	NYPD	Blocked Driveway	No Access	Street/Sidewalk	11105.0	ASTORIA	Closed	The Police Department responded to the complai...
2	32309159	2015-12-31 23:59:30	2016-01-01 01:51:00	NYPD	Blocked Driveway	No Access	Street/Sidewalk	10458.0	BRONX	Closed	The Police Department responded and upon

Figure 10. Dropping the missing values from the table

```
: df.isnull().sum()

: Unique Key          0
  Created Date        0
  Closed Date         0
  Agency              0
  Complaint Type      0
  Descriptor          0
  Location Type       0
  Incident Zip        0
  City               0
  Status             0
  Resolution Description 0
  Borough            0
  Latitude           0
  Longitude          0
  Request_Closing_Time 0
dtype: int64
```

Figure 11. Recounting all the null values in the data

All the null values are now removed from the dataset. The data is cleaned to prepare it for further processes.

## **2.6. Write a python program to see the unique values from all the columns in the data frame.**

In this step, all the unique values from a column are listed individually for each columns. `.unique()` function has been used for all columns to access the unique data from them. This function also gives the length and datatype of the data stored in the column which can be crucial to distinguish if the data are relevant or not. The `.unique()` function has been used for all the columns of the dataset.

### **Explanation of Insights:**

The Unique Key should have all of its values unique, so the Unique Key should contain as many unique key as there are fields in this dataset. All the other data can have repeated values with some having only one value. The Agency resolving the conflict can only be one as the complaints are mitigated by the New York City Police Department. The status of the complaint column also has a single value as the conflicts are from 2015 so they should have been resolved by now. In accessing the real time data, the columns should have more values in it.

```
df["Unique Key"].unique()

array([32310363, 32309934, 32309159, ..., 30283424, 30280004, 30281825],
      shape=(291107,))

df["Created Date"].unique()

<DatetimeArray>
['2015-12-31 23:59:45', '2015-12-31 23:59:44', '2015-12-31 23:59:29',
 '2015-12-31 23:57:46', '2015-12-31 23:56:58', '2015-12-31 23:56:30',
 '2015-12-31 23:55:32', '2015-12-31 23:54:05', '2015-12-31 23:53:58',
 '2015-12-31 23:52:58',
 ...
 '2015-03-29 00:42:48', '2015-03-29 00:37:15', '2015-03-29 00:35:28',
 '2015-03-29 00:35:23', '2015-03-29 00:35:04', '2015-03-29 00:34:32',
 '2015-03-29 00:33:28', '2015-03-29 00:33:03', '2015-03-29 00:33:02',
 '2015-03-29 00:33:01']
Length: 251970, dtype: datetime64[ns]

df["Closed Date"].unique()

<DatetimeArray>
['2016-01-01 00:55:00', '2016-01-01 01:26:00', '2016-01-01 04:51:00',
 '2016-01-01 07:43:00', '2016-01-01 03:24:00', '2016-01-01 01:50:00',
 '2016-01-01 01:53:00', '2016-01-01 01:42:00', '2016-01-01 08:27:00',
 '2016-01-01 01:17:00',
 ...
 '2015-03-29 00:57:23', '2015-03-29 02:57:41', '2015-03-29 01:02:39',
 '2015-03-29 04:14:27', '2015-03-29 08:41:24', '2015-03-29 02:52:28',
 '2015-03-29 01:13:01', '2015-03-29 02:33:59', '2015-03-29 04:38:35',
 '2015-03-29 04:41:50']
Length: 231991, dtype: datetime64[ns]

df["Agency"].unique()

array(['NYPD'], dtype=object)
```

Figure 12. Finding unique values

```
df["Complaint Type"].unique()

array(['Noise - Street/Sidewalk', 'Blocked Driveway', 'Illegal Parking',
 'Derelict Vehicle', 'Noise - Commercial',
 'Noise - House of Worship', 'Posting Advertisement',
 'Noise - Vehicle', 'Animal Abuse', 'Vending', 'Traffic',
 'Drinking', 'Noise - Park', 'Graffiti', 'Disorderly Youth'],
      dtype=object)

df["Descriptor"].unique()

array(['Loud Music/Party', 'No Access', 'Commercial Overnight Parking',
 'Blocked Sidewalk', 'Posted Parking Sign Violation',
 'Blocked Hydrant', 'With License Plate', 'Partial Access',
 'Unauthorized Bus Layover', 'Double Parked Blocking Vehicle',
 'Vehicle', 'Loud Talking', 'Banging/Pounding', 'Car/Truck Music',
 'Tortured', 'In Prohibited Area', 'Double Parked Blocking Traffic',
 'Congestion/Gridlock', 'Neglected', 'Car/Truck Horn', 'In Public',
 'Other (complaint details)', 'No Shelter', 'Truck Route Violation',
 'Unlicensed', 'Overnight Commercial Storage', 'Engine Idling',
 'After Hours - Licensed Est', 'Detached Trailer',
 'Underage - Licensed Est', 'Chronic Stoplight Violation',
 'Loud Television', 'Chained', 'Building', 'In Car',
 'Police Report Requested', 'Chronic Speeding',
 'Playing in Unsuitable Place', 'Drag Racing',
 'Police Report Not Requested', 'Nuisance/Truant'], dtype=object)

df["Location Type"].unique()

array(['Street/Sidewalk', 'Club/Bar/Restaurant', 'Store/Commercial',
 'House of Worship', 'Residential Building/House',
 'Residential Building', 'Park/Playground', 'Vacant Lot',
 'House and Store', 'Highway', 'Commercial', 'Roadway Tunnel',
 'Subway Station', 'Parking Lot'], dtype=object)

df["Incident Zip"].unique()

array([10034., 11105., 10458., 10461., 11373., 11215., 10032., 10457.,
 11415., 11219., 11372., 10453., 11208., 11379., 11374., 11412.,
 11217., 11234., 10026., 10456., 10030., 10467., 11432., 10031.,
 11419., 10024., 11201., 11216., 10462., 11385., 11414., 11213.,
 11375., 11211., 10312., 10017., 11417., 10002., 10027., 11200.]
```

Figure 13. Finding unique values (2)

```
df["City"].unique()

array(['NEW YORK', 'ASTORIA', 'BRONX', 'ELMHURST', 'BROOKLYN',
      'KEW GARDENS', 'JACKSON HEIGHTS', 'MIDDLE VILLAGE', 'REGO PARK',
      'SAINT ALBANS', 'JAMAICA', 'SOUTH RICHMOND HILL', 'RIDGEWOOD',
      'HOWARD BEACH', 'FOREST HILLS', 'STATEN ISLAND', 'OZONE PARK',
      'RICHMOND HILL', 'WOODHAVEN', 'FLUSHING', 'CORONA',
      'QUEENS VILLAGE', 'OAKLAND GARDENS', 'HOLLIS', 'MASPETH',
      'EAST ELMHURST', 'SOUTH OZONE PARK', 'WOODSIDE', 'FRESH MEADOWS',
      'LONG ISLAND CITY', 'ROCKAWAY PARK', 'SPRINGFIELD GARDENS',
      'COLLEGE POINT', 'BAYSIDE', 'GLEN OAKS', 'FAR ROCKAWAY',
      'BELLEROSE', 'LITTLE NECK', 'CAMBRIA HEIGHTS', 'ROSEDALE',
      'SUNNYSIDE', 'WHITESTONE', 'ARVERNE', 'FLORAL PARK',
      'NEW HYDE PARK', 'CENTRAL PARK', 'BREEZY POINT', 'QUEENS',
      'Astoria', 'Long Island City', 'Woodside', 'East Elmhurst',
      'Howard Beach'], dtype=object)

df["Status"].unique()

array(['Closed'], dtype=object)

df["Resolution Description"].unique()

array(['The Police Department responded and upon arrival those responsible for the condition were gone.',
      'The Police Department responded to the complaint and with the information available observed no evidence of the violation at that time.',
      'The Police Department responded to the complaint and took action to fix the condition.',
      'The Police Department issued a summons in response to the complaint.',
      'The Police Department responded to the complaint and determined that police action was not necessary.',
      'The Police Department reviewed your complaint and provided additional information below.',
      'Your request can not be processed at this time because of insufficient contact information. Please create a new Service Request on NYC.gov and provide more detailed contact information.',
      'This complaint does not fall under the Police Department's jurisdiction.',
      'The Police Department responded to the complaint and a report was prepared.',
      'The Police Department responded to the complaint but officers were unable to gain entry into the premises.',
      'The Police Department made an arrest in response to the complaint.',
      'Your complaint has been forwarded to the New York Police Department for a non-emergency response. If the police determine the vehicle is illegally parked, they will ticket the vehicle and then you may either contact a private towing company to remove the vehicle or ask your local precinct to contact 'rotation tow'. Any fees charged for towing will have to be paid by the vehicle owner. 311 will have additional information in 8 hours. Please note your service request number for future reference.'],
      dtype=object)
```

Figure 14. Finding unique values (3)

```
df["Borough"].unique()

array(['MANHATTAN', 'QUEENS', 'BRONX', 'BROOKLYN', 'STATEN ISLAND'],
      dtype=object)

df["Latitude"].unique()

array([40.86568154, 40.77594531, 40.87032452, ..., 40.77664592,
      40.70635259, 40.71605291], shape=(123013,))

df["Longitude"].unique()

array([-73.92350096, -73.91509394, -73.88852464, ..., -73.94880526,
      -73.87124456, -73.9913785 ], shape=(123112,))

df["Request_Closing_Time"].unique()

<TimedeltaArray>
['0 days 00:55:15', '0 days 01:26:16', '0 days 04:51:31', '0 days 07:45:14',
 '0 days 03:27:02', '0 days 01:53:30', '0 days 01:57:28', '0 days 01:47:55',
 '0 days 08:33:02', '0 days 01:23:02',
 ...
 '0 days 06:46:59', '0 days 07:28:23', '0 days 05:13:46', '0 days 05:19:11',
 '0 days 10:22:47', '0 days 09:46:41', '0 days 15:40:46', '0 days 04:44:52',
 '0 days 09:44:44', '0 days 15:42:26']
Length: 47134, dtype: timedelta64[ns]
```

Figure 15. Finding unique values (4)



### 3. Data Analysis

This is the major phase in the data analysis process. The relevant information are extracted from the data according to the needs and requirements of the analysis. The data are analyzed to extract the required information out of them. The data and the information extracted from them are then analyzed to find the correlation and relevancy of the data with each other, the skewness or kurtosis of the data. The information gotten out of this phase is then used to create a story about the data that will be relevant for the organization or business in the future.

#### 3.1. Write a Python program to show summary statistics of sum, mean, standard deviation, skewness, and kurtosis of the data frame.

The data is analyzed to get the relevant information out of the data. In this step all the columns with numerical data are processed to give out their sum, mean, standard deviation, skewness and kurtosis. Each of the numeric columns give out the information extracted from the sum(), mean(), std(), skew() and kurtosis() functions. These functions give out relevant information on the data from the average values of the data, how much the values deviate from each other, how skewed the data is and measures of outliers which are relevant to carry out further analysis on the data.

**Write a Python program to show summary statistics of sum, mean, standard deviation, skewness, and kurtosis of the data frame.**

```
df.describe()
```

	Unique Key	Created Date	Closed Date	Incident Zip	Latitude	Longitude	Request_Closing_Time
count	2.911070e+05	291107	291107	291107.000000	291107.000000	291107.000000	291107
mean	3.130158e+07	2015-08-14 11:25:43.378747648	2015-08-14 15:44:15.511413248	10857.977349	40.725681	-73.925035	0 days 04:18:32.132665995
min	3.027948e+07	2015-03-29 00:33:01	2015-03-29 00:57:23	83.000000	40.499135	-74.254937	0 days 00:01:00
25%	3.079934e+07	2015-06-08 15:38:00	2015-06-08 21:25:00	10314.000000	40.668926	-73.970957	0 days 01:16:30
50%	3.130675e+07	2015-08-13 22:57:41	2015-08-14 02:50:57	11209.000000	40.717782	-73.930774	0 days 02:42:38
75%	3.179091e+07	2015-10-19 15:03:16.500000	2015-10-19 20:58:35.500000	11238.000000	40.782973	-73.875788	0 days 05:20:24
max	3.231065e+07	2015-12-31 23:59:45	2016-01-03 16:22:00	11697.000000	40.912869	-73.700760	24 days 16:52:22
std	5.753777e+05	NaN	NaN	580.280774	0.082411	0.078654	0 days 06:03:45.509089128

Figure 16. Data described with sum, mean and standard deviation

**Explanation of Insight:**

The sum of all these data Unique Key, Incident Zip, latitude and longitude aren't effective since they are all either ID value or values representing the location of the place. Adding up the created date and closed date of the complaint does not give a value significant for any analysis neither does the sum of Request\_Closing\_Time column. The mean and standard deviation of the Unique Key also does not give a significant data since it only represents a unique key value for the fields.

The Incident Zip has a mean of about 10857 and a standard deviation of 580. This shows the average zip values for the complaint types, giving a deviation of 580 which is not too significant of a deviation from the central value. The central value helps to verify that the complaints are coming from the New York City area. The mean of Latitude, 40.72, and Longitude, -73.92, can give the rough estimate of the area where the complaints are coming from which seems to remain consistent with the Zip Code. The low deviation of the data, 0.0824 for Latitude and 0.0786 for Longitude signifies that the data is mostly clustered around a small area rather than being spread out throughout the city.

The mean time for the Request\_Closing\_Time is a significant data to consider since it gives us the average time taken for a complaint to be resolved. The average time taken is 4 hours and 18 minutes, this will also help us analyze the performance metric of the New York City Police Department. The deviation for this is 6 hours and 3 minutes which is significantly higher than the mean. There is a significant deviation in the complaint resolution time. A reason for such high variation is also due to the variation in complaint types as different complaint types have different time constraints to resolve them.

```
df.skew(numeric_only=True)
```

```
Unique Key      0.016898  
Incident Zip    -2.553956  
Latitude        0.123114  
Longitude       -0.312739  
dtype: float64
```

Figure 17. Skewness of all the columns with numerical data

The skew and their significance is mapped as follows:

- If the skew is  $<0.5$  the distribution is approximately normal.
- If the skew is between 0.5-0.7, the distribution is moderately skewed.
- If the skew is  $>0.7$ , then the distribution is highly skewed.

The highly and moderately skewed data can cause some problems during the data analysis phase so such skews should be mitigated using log functions and so on.

The Incident Zip column has a high negative skew value, this value needs to be mitigated to get the distribution to a more normal distribution for better data analysis based on this column. Latitude is skewed mildly in the positive direction so the distribution is almost normal. Longitude has a moderate negative skew meaning that it has a longer right tail which can cause some issue during the data analysis phase.

```
df['Request_Closing_Time'] = pd.to_timedelta(df['Request_Closing_Time'])  
  
df['Request_Closing_Time'].dt.total_seconds().skew()  
  
np.float64(0.0)
```

Figure 18. Skew for Request Closing Time column

The skew value for the Request\_Closing\_Time column is approximated to 0 so the distribution of this column is normal. This data does not need to be modified before further analysis can be carried out.

```
df.kurtosis(numeric_only=True)
```

```
Unique Key      -1.176593  
Incident Zip     37.827777  
Latitude        -0.734818  
Longitude        1.455600  
dtype: float64
```

Figure 19. Kurtosis of all the columns with numerical data

The kurtosis and their significance is mapped as follows:

- If the kurtosis is between 0-1, the distribution is mildly deviated from normal.
- If the kurtosis is between 1-3, the distribution is moderately deviated from normal.
- If the kurtosis is >3, then the distribution is highly deviated from the normal.

The high values of kurtosis is also expected to be mitigated before using them for data analysis.

Incident Zip column has a very high positive kurtosis value indicating high amount of deviation from the normal distribution, meaning that there are high number of outliers and has a high amount of same data, i.e. frequency distribution is concentrated in one area. The latitude has a mild distribution from the normal signifying a properly distributed data. The Longitude has a moderately deviated data signifying some clustered up data in certain locations and some outliers.

```
df['Request_Closing_Time'] = pd.to_timedelta(df['Request_Closing_Time'])  
df['Request_Closing_Time'].dt.total_seconds().kurtosis()  
  
np.float64(848.5545245420765)
```

*Figure 20. Kurtosis of Request Closing Time column*

The kurtosis of 848.55 signifies a significant peak of the distribution in the mean of the Request\_Closing\_Time column indicating that most complaints take around 4 hrs and 18 mins to resolve but the extremely high positive value also indicates a huge discrepancy in the time constraint. The outliers in this column are significantly skewed.

### 3.2. Write a Python program to calculate and show correlation of all variables.

Here each numerical data is tallied with each other to figure out the correlation between them. Each columns are put against each other to show the degree of correlation between them. The data below shows the significance of the correlation between the columns of the table. The positive values show a positive correlation between the values and the negative values represent a negative correlation.

```
i3]: df.select_dtypes(include='number').corr()
```

```
i3]:
```

	Unique Key	Incident Zip	Latitude	Longitude	Request_Closing_Time
Unique Key	1.000000	0.025492	-0.032613	-0.008621	0.053126
Incident Zip	0.025492	1.000000	-0.499081	0.385934	0.057182
Latitude	-0.032613	-0.499081	1.000000	0.368819	0.024497
Longitude	-0.008621	0.385934	0.368819	1.000000	0.109724
Request_Closing_Time	0.053126	0.057182	0.024497	0.109724	1.000000

Figure 21. Correlation between the variables

Here, the `.corr()` function offered by Pandas is used to find the correlation between the columns. The `.select_dtypes()` function is being used to select all the columns with a number datatype using `include = 'number'`.

#### Explanation of Insights:

The unique key is an ID, so it is not correlated to anything; it just gives a unique identifier value for each field, so the significance value of the ID mapped out with other columns shows no significant correlation between them.

The correlation of the Incident Zip code with the latitude and longitude, strong negative correlation of -0.499 and moderate positive correlation of 0.386 corresponds to how the lower value of zip code means higher value of latitude and lower value of longitude and

vice versa. This gives an insight on where exactly the place of incident is concentrated on the map.

The latitude and longitude have a moderate positive correlation with each other, signifying that when latitude increased, longitude also increased slightly. This shows that the complaints are concentrated in a slightly higher latitude also have higher longitudinal value. So the complaints from the northern side cluster around the eastern side of the city and the ones on the southern side are concentrated around the western side. Here the north, south, east and west signify the latitude and longitudinal values.

The variable 'Request\_Closing\_Time' has a weak correlation with the rest of the data but a moderate positive correlation with the longitude. This means that other aspects are not significant enough to determine the closing time but higher longitude is expected to mean a slightly higher closing time as well. Though the correlation is not huge but it is still significant enough to infer the above.

## 4. Data Exploration

Data Exploration is a fundamental part of the data analysis process where the structure of the data and the patterns in them are discovered (Coursera, 2024). This process uses data visualization and statistical methods to uncover the hidden features and characteristics of the data. The data distribution and other underlying features are discovered during this phase in order to brainstorm the ways the data can be presented and analyzed with best practices to get relevant insights on the data.

There are predominantly three types of Data Exploration (Coursera, 2024):

- **Descriptive analysis:** Here a broad trend of data is analyzed where you look at the data in a more general terms. The amount of data on each column based on the value, the range of the data and so on fall within this type. This has been used in this project before visualizing the data.
- **Visual analysis:** This includes creating various charts, plots and diagram to help visualize the data to find the underlying discrepancies in the data. Having a visual representation of the data makes it easier and convenient to figure out the underlying patterns and distribution of the data.
- **Statistical analysis:** This includes various kinds of statistical testing like Hypothesis testing and so on. It is helpful to use such forms of analysis to figure out significances and underlying patterns not obvious to the eyes. This uncovers the inconspicuous story of the data.



## 4.1. Provide four major insights through visualization after data mining.

Below are some data visualizations that can give different information on the data being analysed. This helps analyse the data with accurate data depiction and visualization of the data being used.

### 4.1.1. Frequency of Complaints

The types of complaints being registered or made are a major insight for data analysis to better plan for future service calls.

Provide four major insights through visualization that you come up after data mining.

```
complaint_counts = df['Complaint Type'].value_counts()
complaint_counts
```

Complaint Type	
Blocked Driveway	76676
Illegal Parking	74021
Noise - Street/Sidewalk	47747
Noise - Commercial	35144
Derelect Vehicle	17506
Noise - Vehicle	16868
Animal Abuse	7744
Traffic	4466
Noise - Park	3927
Vending	3773
Drinking	1270
Noise - House of Worship	920
Posting Advertisement	647
Disorderly Youth	285
Graffiti	113

```
Name: count, dtype: int64

import matplotlib.pyplot as plt

import seaborn as sns

plt.figure(figsize=(10,6))
sns.countplot(data=df, y='Complaint Type', order=df['Complaint Type'].value_counts().index)
plt.title("Frequency of Complaints")
plt.xlabel("No. of Complaints")
plt.ylabel("Complaint Type")
plt.show()
```

Figure 22. Code to find frequency of complaints

### Explanation of Code:

All the values in the Complaint Type column are counted for the number of repetitions using `.value_counts()` function. Then the number of values counted is then mapped out in a graph using Seaborn, a python library that is built off of Matplotlib (GeeksforGeeks, 2024). Using Matplotlib library gave a mashed up plot so for a clearer representation of the plot, Seaborn was used.

Seaborn still uses components of Matplotlib since it was built off of it, so `.title()`, `.xlabel()`, `.ylabel()` and `.show()` are used from the pyplot module of the Matplotlib library to give labels to the plot. The title, x and y label functions are used to label the graph and show function is used to display all figures since the figure function has been used to give the size for the graph plotted below.

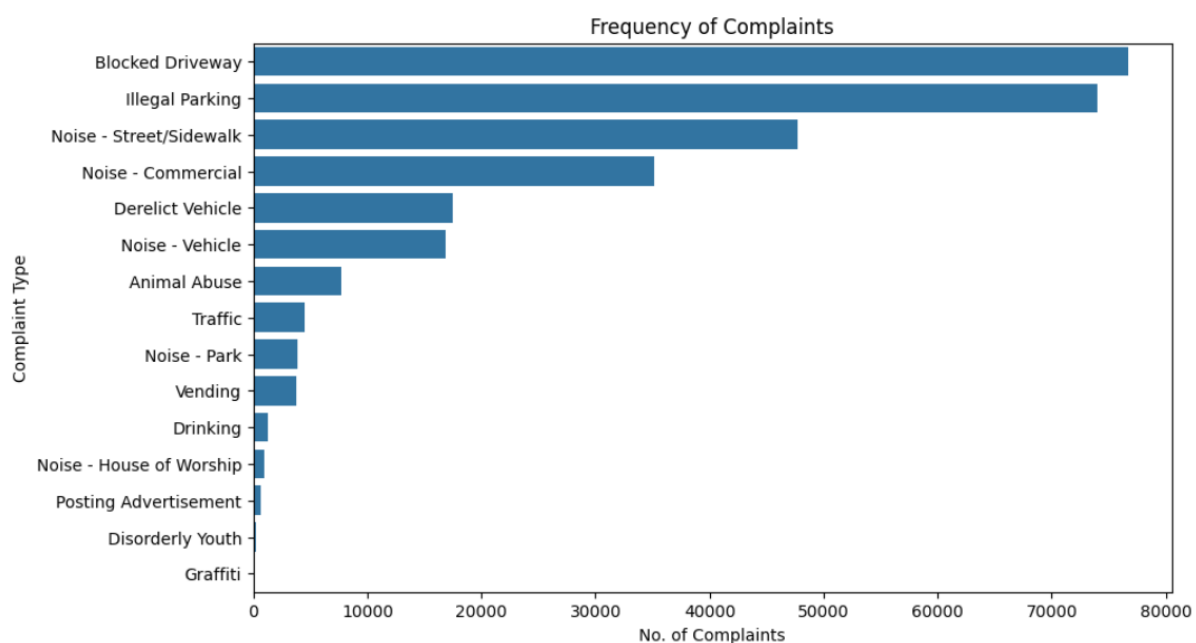


Figure 23. Graphs showing the frequency of complaints

### Explanation of Insights:

The majority of the complaints are about drive ways being blocked, followed closely by Illegal Parking. The frequency distribution of Complaint Type shows that majority of the problems are caused by vehicle users' irresponsible parking. This means that bringing

proper rules to regulate the vehicles and their users is essential to reducing such inconveniences and reducing the complaints.

Noise complaints are another frequent problem in the area, with noisy sidewalks, commercial areas and vehicles are the major issues in the area. Proper mitigating measures can be brought forth to reduce such problems before they even occur.

#### 4.1.2. Complaints by Borough

Here we are figuring out the number of complaints made based on the Borough. This gives a general view of where most complaints are made from. Based on the data below the state can determine the number of Agencies and contact people to mobilize in the area.

#### Explanation of Code:

To find the Borough with the higher complaints and those with the low complaint rates, we are mapping our data to a pie chart as given below. First, the amount of Complaints from each Borough is counted using the `.value_counts()` function as done previously. The `.plot()` function is used to specify the type of plot our Borough data is being presented in, `kind = pie` indicating the pie chart. Similar to the above graph, `.title()`, `.xlabel()` and `.ylabel()` have been used from the `pyplot` module of `Matplotlib` library.

```
: borough_counts = df['Borough'].value_counts()

borough_counts.plot(kind='pie')
plt.title('Complaints by Borough')
plt.xlabel('Borough')
plt.ylabel('Number of Complaints')

: Text(0, 0.5, 'Number of Complaints')
```

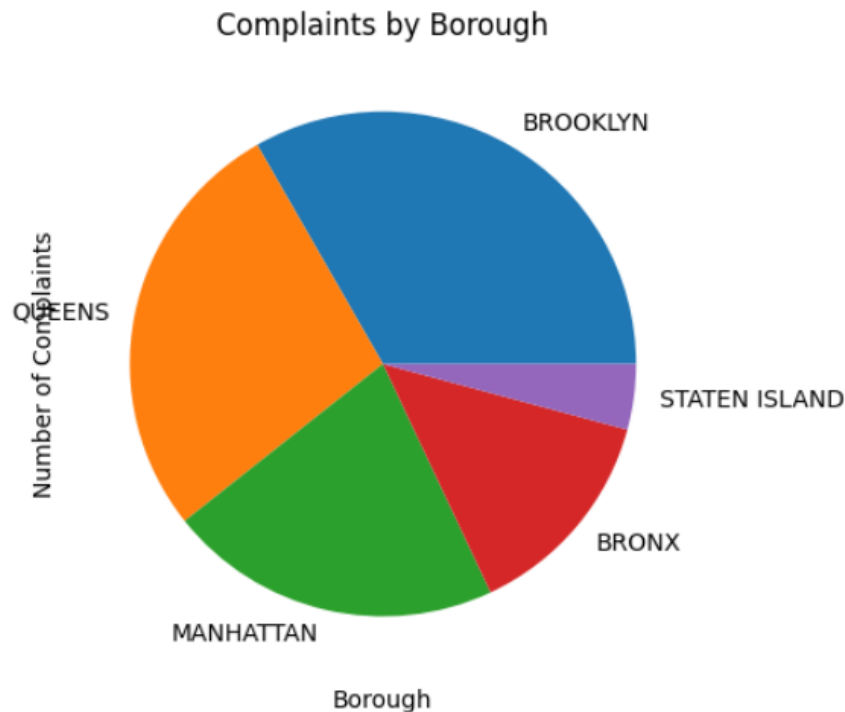


Figure 24. Complaints made based on Borough

### Explanation of Insights:

The pie chart shows where the majority of the complaint comes from. The most complaints by far comes from Brooklyn, followed by Queens, Manhattan, Bronx and Staten Island. The chart gives an insight into which area requires more attention and finding further information on these specific areas give a better idea on how to reduce the complaints from the high complaint areas.

### 4.1.3. Monthly Complaint Tracking

This kind of visualization is helpful to see the seasonal trend in the complaints and how the complaints tend to rise and decrease with the months. It is helpful to manage the city manpower and Agencies accordingly. The graph showcases the trend of complaints based on months of the year, giving insights into which months need more policing and regulations.

```
df['Created Date'] = pd.to_datetime(df['Created Date'])
monthly_complaints = df.resample('ME', on='Created Date').size()
monthly_complaints.plot(title='Monthly New York City Complaints')
plt.xlabel('Date')
plt.ylabel('Complaints')
```

*Figure 25. Code for monthly complaints tracking*

#### Explanation of Code:

Here, the format of the Created Date is changed using the `.to_datetime()` function to make it easier to extract only the months from the column. The `.resample()` function is used to sample the months data. Since the column deals with time-series data `.resample()` function helps extract the required sample data from the overall data, in this case the data is the data of months when complaints were registered. The `.size()` function gives the number of rows in each month data. The remaining code for the plotting of the graph is similar to that of the above codes.

The default graph made by the `.plot()` function when no kind of plot is specified is the line graph as given below.

```
Text(0, 0.5, 'Complaints')
```

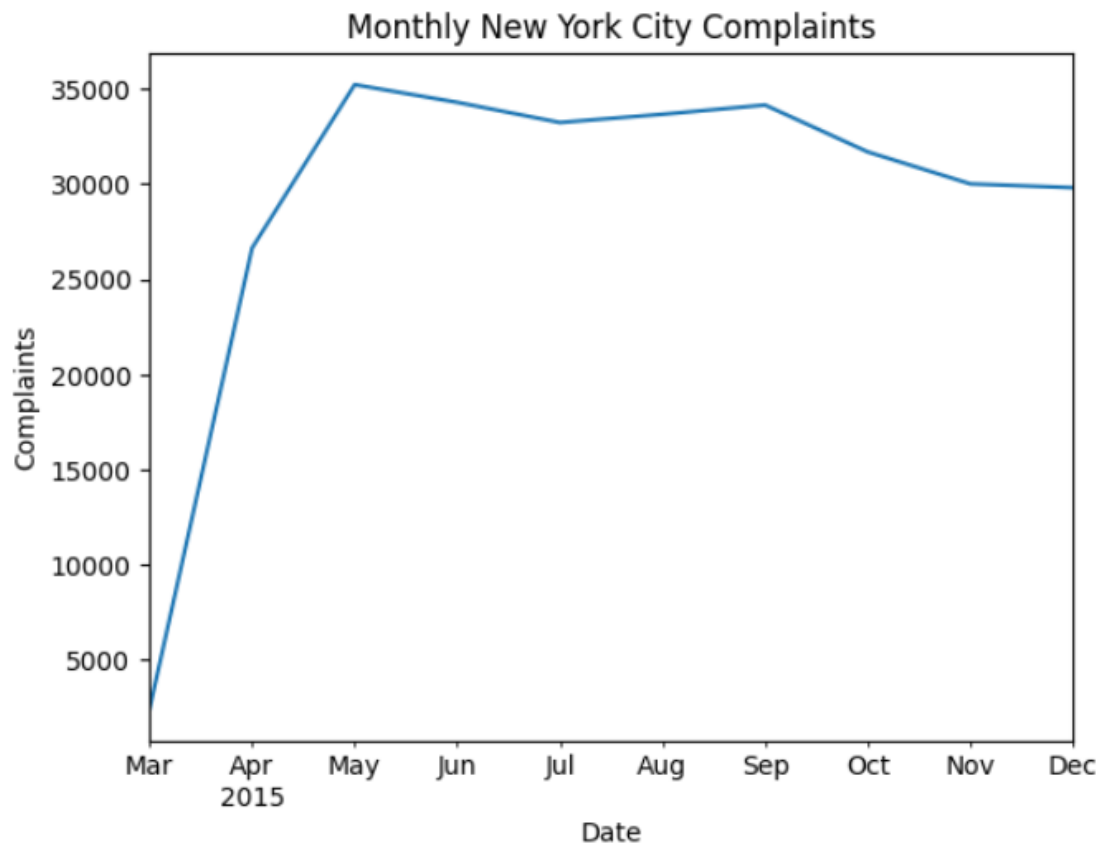


Figure 26. Graph of Complaints based on month

### Explanation of Insights:

The complaints have a steep rising trend from the month of March to April which flattens slightly from April to May and reduces from the month of May to July. This trend shows that the month of May sees the highest amount of complaints registered, after which the number of complaints drop slightly for two months before rising slightly rising for the next two. The complaints then decrease consistently until November after which the curve stabilizes though still in a downward trend but not too significantly.

#### 4.1.4. Geographical Distribution of Complaints

The geographical representation of complaints shows the relative areas from which the complaints have been made. It shows the concentration of complaint in the particular areas of the city. This maps out according to the map of the city as well as the areas with highest complaints.

#### Geographic Distribution of Complaints Using Latitude and Longitude

```
plt.figure(figsize=(12, 8))
plt.scatter(
    x=df['Longitude'],
    y=df['Latitude'],
    s=10,
    c='red',
    marker='.'
)
plt.title('Geographic Distribution of Complaints (Latitude, Longitude)')
plt.xlabel('Longitude (East)')
plt.ylabel('Latitude (North)')

Text(0, 0.5, 'Latitude (North)')
```

Figure 27. Code for the scatter plot of Geographic Distribution of Complaints

#### Explanation of Code:

The .figure() function in the code defines the size of the graph. The .scatter() function defines the graph as a scatter plot graph with x and y variables giving the column that goes in the x-axis of the graph and in the y-axis of the graph respectively. The marker indicates the symbol for marking each value on the plot, c giving the color value and s represents the size of the dots. The rest of the pyplot functions work as explained above.



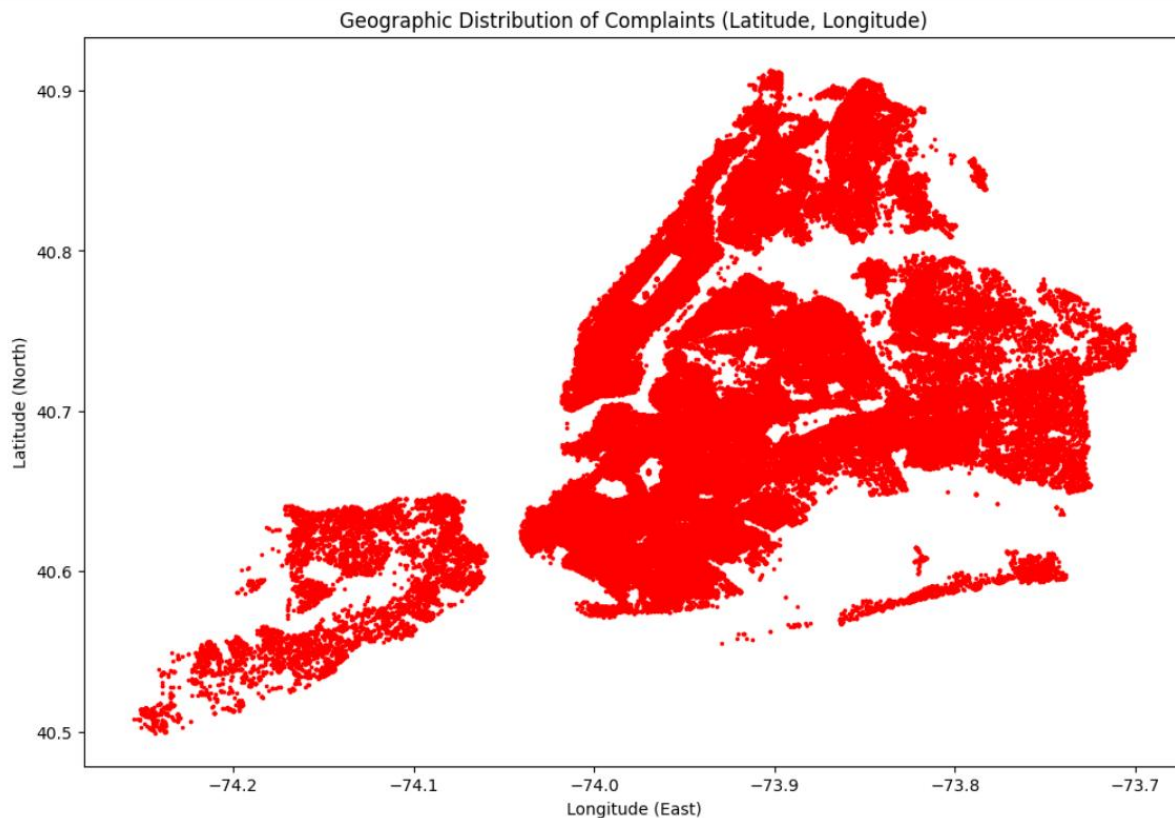


Figure 28. Scatter plot of Geographical Distribution of Complaints

### Explanation of Insights:

From the above scatter plot we can see the concentration of the complaints based on the latitude and longitude of the place. This also shows us the relative area that the city covers. Less in the south western part and more on the north eastern area. The higher amount of complaints also indicate a higher population density in the area. This also infers where these kinds of services are required the most which in this case lies in the center and eastern side of the city with higher concentration on the North.

## 4.2. Arrange the complaint types according to their average 'Request\_Closing\_Time', categorized by various locations. Illustrate it through graph as well.

The data of the Complaint Type is first grouped by Borough and closing time. The below graphs present the average closing time for each type of complaint based on the city of the complaint. This shows how quickly such matters are being taken care of, which is important to track and update the city policies and the service providers accordingly. This can be further used to find the agencies handling the response the quickest and prioritize them or observe then regarding the reasons for their success.

Arrange the complaint types according to their average 'Request\_Closing\_Time', categorized by various locations. Illustrate it through graph as well.

```
# grouping the Complaint Type based on Borough and closing time
grouped = df.groupby(['Borough', 'Complaint Type'])['Request_Closing_Time'].mean().reset_index()

# sorting the grouped data in descending order
grouped_sorted = grouped.sort_values(['Borough', 'Request_Closing_Time'], ascending=[True, False])
boroughs = grouped['Borough'].value_counts().index

figure, axes = plt.subplots(nrows=len(boroughs), figsize=(14, 20), sharex=True)

for i, borough in enumerate(boroughs):
    # Filter data for the borough
    borough_data = grouped_sorted[grouped_sorted['Borough'] == borough]

    # Create horizontal bars
    axes[i].barh(
        borough_data['Complaint Type'],
        borough_data['Request_Closing_Time'],
        color=plt.cm.tab20(i)
    )
    axes[i].set_title(f'Borough: {borough}')
    axes[i].set_xlabel('Average Closing Time (Hours)')
    axes[i].grid(axis='x', linestyle='--')

plt.suptitle('Complaint Types Based on Average Closing Time Across Boroughs')
```

Figure 29. Code for Complaint Types based on Location and Closing time

### Explanation of Code:

The given data I first grouped by the Borough and Complaint Type columns using the `.groupby()` function. A mean is taken of the `Request_Closing_Time` column based on the grouped data. The `.reset_index()` resets the index value since there is a lot of discontinuity in the index after the null data were removed.

The grouped data is then sorted using the `.sort_values` function based on the Borough and Request\_Closing\_Time column. Ascending value gives if the values are to be sorted in ascending order or descending order. Here, an array is passed in the ascending parameter since our values are sorted based on two columns. Then the data of each Borough are grouped together based on each Borough value to be mapped based on it.

The size of the graph is given using figure and axes function and the graph is plotted for each borough according to the mean time taken for each Complaint Type to be resolved.

The `.subplots()` function helps create multiple sub plots based on the grouped and sorted data, in this case the subplots are divided by Borough. Then, for each loop has been used to create the graph for each Borough by looping through each Borough value. The subplots are then graphed based on the Complaint Type and Request\_Closing\_Time values in the y-axis and x-axis respectively. `Axes[i]` give the value of the Borough indexes so a title is set for each of the borough as well as the labels for x and y axis using `.set_title()` and `.set_xlabel()` functions. The `.grid()` function creates a grid on the x-axis as specified on the code with dashes (-). This makes comparing values based on the visualization graph below much easier.

```
Text(0.5, 0.98, 'Complaint Types Based on Average Closing Time Across Boroughs')
```

Complaint Types Based on Average Closing Time Across Boroughs

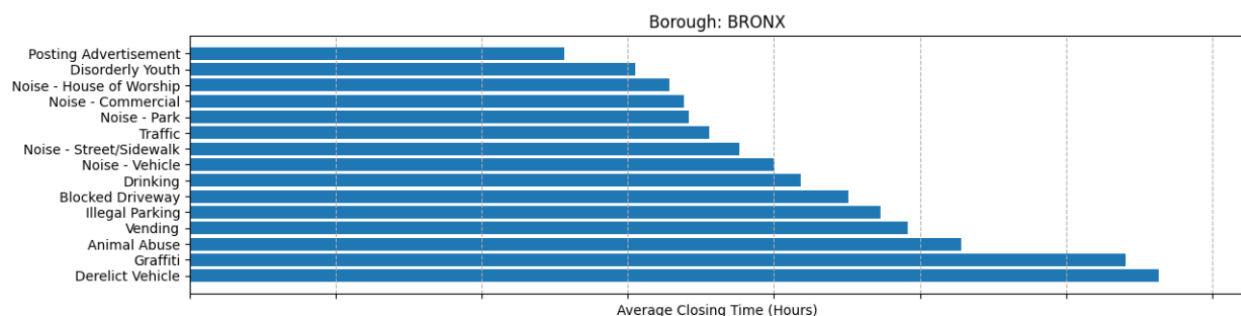


Figure 30. Complaint Type Based on Location: Bronx

### Explanation of Insights:

The Borough Bronx has the highest closing time for Derelict vehicles and the lowest for posting advertisement. So the city should mobilize more pick-up trucks for towing the vehicles. The closing time for this is also higher since towing a vehicle requires more time than just tearing out an advertisement posting. The same goes with Graffiti art, which takes more time to be painted off in comparison to other activities. The traffic is something the city can work on by providing more Police intervention wherever needed.

Mobilizing the animal care centers to be more active around the city and promptly informing them of the cases can also help reduce the time taken to resolve the animal abuse issue.

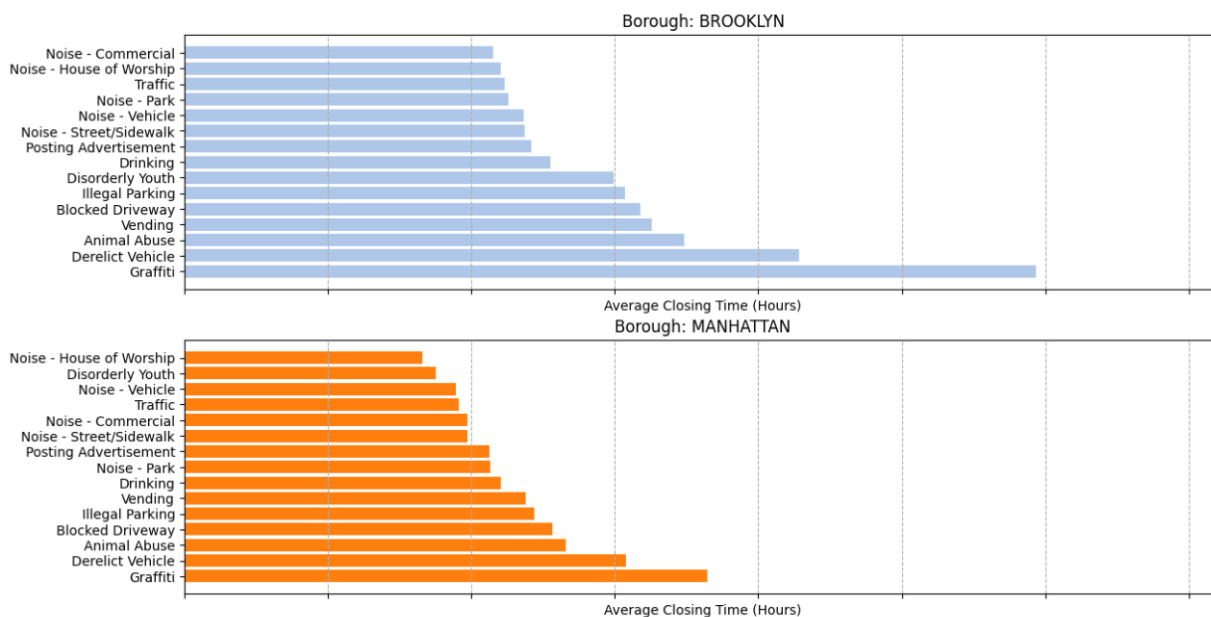


Figure 31. Complaint Type Based on Location: Brooklyn and Manhattan

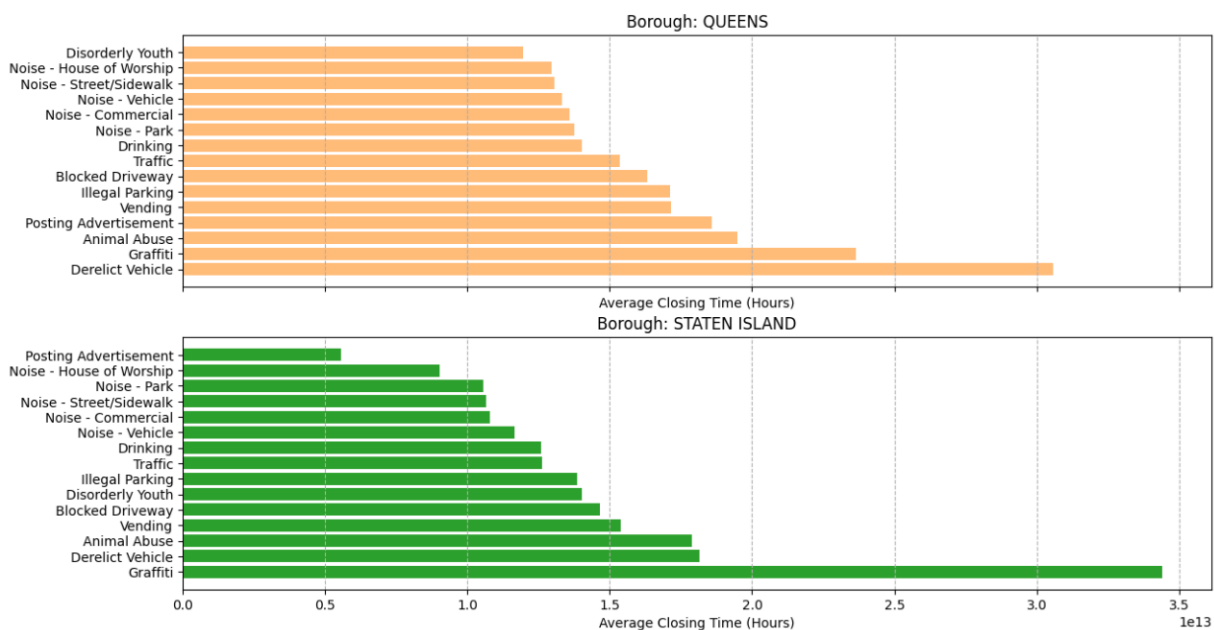


Figure 32. Complaint Type Based on Location: Queens and Staten Island

All the Borough in the New York City have the top three of their major complaints to be Graffiti, Derelict Vehicle and Animal Abuse. So based on this we can infer that most of these complaints do take longer than the handling Noise Complaints for example. So these instances can also be taken care of before they take roots. Having a regulated corner for creative expression of Graffiti art, encouraging people to do it responsibly can reduce the problem by some degree. Having more spaces for vehicle disposal should encourage people to dispose of their derelict vehicle in the proper place.

Another reason for the response time of the Graffiti being so long is also because the problem is rare compared to other problems, expending resources to reduce an uncommon problem can also take the resources away from the other matters actually needing them.

## 5. Statistical Testing

The statistical tests are used to determine the significance of the relationship between two variables. Here the statistical testing is carried out using hypothesis testing. A null hypothesis is assumed indicating no relationship between the variables and the test is carried out to determine if the data collected rejects the null hypothesis or not (Bevans, 2023). If the null hypothesis is rejected, the relationship between the columns are said to be significant. The significance is measured in our test by comparing the p value with 0.05, if the p value is greater than 0.05 the null hypothesis is accepted, meaning that the relationship between the two columns are significant.

### 5.1. Test 1

- **Whether the average response time across complaint types is similar or not.**
  - **State the Null Hypothesis (H0) and Alternate Hypothesis (H1).**
  - **Perform the statistical test and provide the p-value.**
  - **Interpret the results to accept or reject the Null Hypothesis.**

ANOVA, Analysis of Variance, test is used to compare the mean value of more than two groups. The ANOVA test is divided into two types based on the number of independent variables used for the test, one-way ANOVA test only uses a single independent variable whereas the two-way ANOVA uses two independent variables (Bevans, 2024).

Here,

H0: signifies that the response time across complaints is not similar.

H1: signifies that the response time across complaints are similar.

```

: #h0: response time across complaints is not similar
  #h1: response time across complaints is similar
  import scipy.stats as stats

  # Make sure 'Request_Closing_Time' is numeric (hours)
  if df['Request_Closing_Time'].dtype != 'float64' and df['Request_Closing_Time'].dtype != 'int64':
      df['Request_Closing_Time'] = df['Request_Closing_Time'].dt.total_seconds() / 3600

  # Pick top 5 most frequent complaint types (to keep the test manageable)
  top_complaints = df['Descriptor'].value_counts().head(5).index

  # Create a list of response times for each complaint type
  groups = [df[df['Descriptor'] == complaint]['Request_Closing_Time'] for complaint in top_complaints]

  # Perform one-way ANOVA test
  f_stat, p_value = stats.f_oneway(*groups)

  # Print results
  print(f"F-statistic: {f_stat}")
  print(f"P-value: {p_value}")

  if p_value < 0.05:
      print("Null is true. We can reject the null hypothesis.")
  else:
      print("Null is false. We can accept the null hypothesis.")

F-statistic: 770.5225609777682
P-value: 0.0
Null is true. We can reject the null hypothesis.

```

Figure 33. Test 1 Code and Result

The p value is smaller than 0.05,  $p\_value < 0.05$  so we can reject the null hypothesis. The results signify that the average closing time between each complaint is not significantly different.

The f statistics value shows how much an average value differs within different groups as compared to the variation within the same group. The f value of 770.522 is a large value signifying that the variance of the mean between the groups is highly significant in comparison to that of within the groups. High value of f signifies that the null hypothesis is not relevant.

## 5.2. Test 2

- **Whether the type of complaint or service requested and location are related.**
  - **State the Null Hypothesis (H0) and Alternate Hypothesis (H1).**
  - **Perform the statistical test and provide the p-value.**
  - **Interpret the results to accept or reject the Null Hypothesis.**

Chi-square test is another way of finding the significance of the relationship between the two variables. To check the relationship between the two variables, complaint type and borough, for the location of the complaint, two categorical data, chi-square test is used. The test is done to check whether the distribution of the two variables are independent or not.

Here,

H0, null hypothesis indicates that the complaint type and location are not related.

The alternative hypothesis, H1, gives that the columns are significantly related.



## Test 2

- Whether the type of complaint or service requested and location are related.
- State the Null Hypothesis (H0) and Alternate Hypothesis (H1).
- Perform the statistical test and provide the p-value.
- Interpret the results to accept or reject the Null Hypothesis.

```
: #h0: type of complaint or service requested and location are not related
#h1: type of complaint or service requested and location are related

contingency_table = pd.crosstab(df['Complaint Type'], df['Borough'])

complaints = df['Complaint Type'].value_counts().index
boroughs = df['Borough'].value_counts().index
filtered_table = contingency_table.loc[complaints, boroughs]

# Perform Chi-Square Test
chi2, p_value, dof, expected = stats.chi2_contingency(filtered_table)

: p_value

: np.float64(0.0)
```

Figure 34. Chi square test to check the complaint and location are related

Since the `p_value` is smaller than 0.05 ( $p\_value < 0.05$ ), the null hypothesis is rejected, so the data shows a strong relationship between location and the complaint types. Based on this insight, the city can mobilize the personnel and resources according to the needs of the particular Borough.

## 6. Conclusion

This project provided a great insight on various ways of handling data for data analysis. We learnt about various steps of Data Analysis, from Data Understanding to Data Exploration. During the course of the project there were a lot of learning opportunity from learning various methods to achieve the same result, using various libraries like Pandas for reading and importing the dataset into the jupyter notebook platform, Matplotlib for graphs like bar charts, box plots, scatter plot and so on, scipy for measuring statistical significance of the data, like chi-square test, find the p value and so on.

This project also allowed to dig deeper into the data and figure out how the data represents the lived reality of people and how such data can be analyzed to figure out valuable insights, in this case, on how the situation of Complaints can be made better. The project gave a hands-on understanding of how the data is handled and analyzed beyond charts and into words and interpretations to bring real time changes in the system.

## References

Coursera, 2024. *What Is Data Exploration? Definition, Types, Uses, and More*. [Online]  
Available at: <https://www.coursera.org/articles/data-exploration>  
[Accessed 9 May 2025].

GeeksforGeeks, 2024. *Difference Between Matplotlib VS Seaborn*. [Online]  
Available at: <https://www.geeksforgeeks.org/difference-between-matplotlib-vs-seaborn/>  
[Accessed 13 May 2025].

Luna, Z., 2021. *CRISP-DM Phase 2: Data Understanding*. [Online]  
Available at: <https://medium.com/analytics-vidhya/crisp-dm-phase-2-data-understanding-b4d627ba6b45>  
[Accessed 03 05 2025].

Pandas, 2024. *pandas documentation*. [Online]  
Available at: <https://pandas.pydata.org/docs/>  
[Accessed 13 May 2025].

US Geological Survey, 2024. *What is the State Plane Coordinate System? Can GPS provide coordinates in these values?*. [Online]  
Available at: <https://www.usgs.gov/faqs/what-state-plane-coordinate-system-can-gps-provide-coordinates-these-values>  
[Accessed 03 05 2025].