Table of Contents

Table of Figures

1 Data Understanding

Data understanding is the process of studying, exploring, and giving sense to data to obtain valuable insights. It is a basic process in data analysis and data science. Data understanding helps to figure out whether there are problems with data, find patterns in data, and decide what other analysis may be useful (IBM, 2021).

The dataset has records of 311 service requests made by customers in New York City since 2010 AD. These requests show different complaints reported to city agencies and officials like the police, sanitation, and public works. The goal is to study and show trends in complaints and find the patterns in the service requests. Dataset has more than 50 columns and thousands of rows where each rows shows a different complaint or service request. The key data or important data in this dataset includes the type of complaint, which agency or officials is responsible, the borough, dates and times for when the request was made and closed, and location details. It also includes information about school, parks, vehicles, and the landmarks but many of these details are not useful for a complaint analysis. The data has different types like text for complaint types and boroughs, dates for when requests were created and closed, and numerical or coordinates. There are missing and inconsistent entries in some columns knowing these details. Understanding these is very important for cleaning and preparing the dataset for analysis.

Table summarizing the columns:

Table 1 Table summarizing the columns of the dataset

s.no	Column Name	Description (Inferred)	Data Type
1	Unique Key	Unique identifier for each complaint	int64
2	Created Date	Date and time when the complaint was filed	object
3	Closed Date	Date and time when the complaint was resolved	object
4	Agency	Code for the agency handling the complaint	object
5	Agency Name	Full name of the agency	object
6	Complaint Type	Type/category of the complaint	object
7	Descriptor	Detailed description within the complaint type	object
8	Location Type	Location classification (e.g., Street, Residential)	object
9	Incident Zip	ZIP code of the incident location	float64
10	Incident Address	Address where the issue occurred	object
11	Street Name	Street name of the incident	object

12	Cross Street 1	First nearby cross street	object
13	Cross Street 2	Second nearby cross street	object
14	Intersection Street 1	First street in an intersection	object
15	Intersection Street 2	Second street in an intersection	object
16	Address Type	Type of address (residential, commercial, etc.)	object
17	City	City name	object
18	Landmark	Landmark associated with the location	object
19	Facility Type	Type of facility involved	object
20	Status	Current status of the request	object
21	Due Date	Expected resolution date	object
22	Resolution Description	Description of how the issue was resolved	object
23	Resolution Action Updated Date	Last date resolution action was updated	object
24	Community Board	Community board code	object
25	Borough	Borough name (e.g., Bronx, Manhattan)	object

26	X Coordinate (State Plane)	X coordinate in NYC's spatial reference system	float64
27	Y Coordinate (State Plane)	Y coordinate in NYC's spatial reference system	float64
28	Park Facility Name	Name of park (if applicable)	object
29	Park Borough	Borough where the park is located	object
30	School Name	Name of school involved (if applicable)	object
31	School Number	ID or number of the school	object
32	School Region	Region of the school	object
33	School Code	Code for the school	object
34	School Phone Number	Contact number of the school	object
35	School Address	Physical address of the school	object
36	School City	City where the school is located	object
37	School State	State where the school is located	object
38	School Zip	ZIP code of the school	object
39	School Not Found	Indicator if school was not found	object
40	School or Citywide Complaint	Indicator for school/citywide complaint	float64

41	Vehicle Type	Type of vehicle involved (if any)	float64
42	Taxi Company Borough	Borough where taxi company is based	float64
43	Taxi Pick Up Location	Pickup location for taxis	float64
44	Bridge Highway Name	Bridge or highway name involved	object
45	Bridge Highway Direction	Direction on the bridge/highway	object
46	Road Ramp	Road ramp information	object
47	Bridge Highway Segment	Segment of bridge or highway	object
48	Garage Lot Name	Name of garage lot	float64
49	Ferry Direction	Direction of ferry (if applicable)	object
50	Ferry Terminal Name	Name of ferry terminal	object
51	Latitude	Geographic latitude of the incident	float64
52	Longitude	Geographic longitude of the incident	float64
53	Location	Tuple of (Latitude, Longitude)	object

2 Data Preparation

Data Preparation is the process of cleaning, organizing and transforming raw data into a format which is ready to use for analysis. It includes fixing errors, handling missing values and structuring data correctly so it can be used effectively for reporting and analysis (AWS, 2025).

2.1 Importing the dataset

First, I have imported Pandas and NumPy, then created a variable df which store and read the csv file and used head( ) function to verify if the file is loaded or not and it prints the first five rows of the dataset. The dataset was successfully imported using pandas.read_csv(), which allows us to analyse over 300,000 records of NYC 311 service requests. This dataset includes columns such as complaint types, location, agency, timestamps, and resolution details.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
# Load the CSV file
df = pd.read_csv("Customer Service_Requests_from_2010_to_Present.csv", low_memory=False)
df.head(3)
```

	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	Bridge Highway Segment	Garag L Nan
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	NaN	Na
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	NaN	Na
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	NaN	Na

3 rows × 53 columns

Figure 1 Importing dataset

2.2 Providing insights on the information and details that the provided dataset carries.

The dataset carries detailed information about the 311 service requests made by New York City Customers which includes the following:

- Time Related details like when did the request created and closed.
- Location details like Zip codes, city, boroughs, and coordinates.
- Complaint Specifics which include type, descriptor, and resolution.
-
- Agency Involvement showing which city department managed or handled the complaint.

```
[36]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300698 entries, 0 to 300697
Data columns (total 54 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   Unique Key                      300698 non-null  int64
 1   Created Date                    300698 non-null  datetime64[ns]
 2   Closed Date                     298534 non-null  datetime64[ns]
 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
```

Figure 2 Insights of the dataset

2.3 Converting 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.

A new column Request_Closing_Time was created by the difference of Created Date and Closed Date which calculates the duration between when a complaint was filed and when it was resolved. Also, it was converted into second so that kurtosis and skewness can be calculated without any error.

[28]:
```
# Convert "Created Date" and "Closed Date" to datetime
df['Created Date'] = pd.to_datetime(df['Created Date'],errors='coerce')
df['Closed Date'] = pd.to_datetime(df['Closed Date'],errors='coerce')

# Calculate time difference and converting request closing time into seconds
df['Request_Closing_Time'] = (df['Closed Date'] - df['Created Date']).dt.total_seconds() / 3600

df[['Created Date','Closed Date','Request_Closing_Time']].head()
```

[28]:

	Created Date	Closed Date	Request_Closing_Time
0	2015-12-31 23:59:45	2016-01-01 00:55:00	0.920833
1	2015-12-31 23:59:44	2016-01-01 01:26:00	1.437778
2	2015-12-31 23:59:29	2016-01-01 04:51:00	4.858611
3	2015-12-31 23:57:46	2016-01-01 07:43:00	7.753889
4	2015-12-31 23:56:58	2016-01-01 03:24:00	3.450556

Figure 3 Creating new column Request_Closing_Time

2.4 Writing a python program to drop irrelevant Columns.

There are irrelevant columns to be dropped that was provided in the question. This code drops 39 irrelevant columns like Agency Name, Street Name, School Name etc. we only take meaningful columns like Complaint Type, Agency, Borough, Status etc. The clean dataset now has fewer column which makes analysis easier and faster.

```
[21]: #define a list of column names that we do not need for analysis
irrelevant_columns = [
    'Agency Name','Incident Address','Street Name','Cross Street 1','Cross Street 2',
    'Intersection Street 1', 'Intersection Street 2','Address Type','Park Facility Name',
    'Park Borough','School Name', 'School Number','School Region','School Code',
    'School Phone Number','School Address','School City','School State','School Zip',
    'School Not Found','School or Citywide Complaint','Vehicle Type', 'Taxi Company Borough',
    'Taxi Pick Up Location','Bridge Highway Name','Bridge Highway Direction','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'
]

df_cleaned = df.drop(columns=irrelevant_columns, errors='ignore') #removing them from dataset
print(df_cleaned.columns.tolist())

['Unique Key', 'Created Date', 'Closed Date', 'Agency', 'Complaint Type', 'Descriptor', 'Location Type', 'Incident Zip', 'City', 'Status', 'Resolution Descriptio
n', 'Borough', 'Latitude', 'Longitude', 'Request_Closing_Time']
```

Figure 4 Removing Irrelevant Column

2.5 Writing a python program to see the unique values from all the columns in the dataframe.

This code makes a dictionary with the unique values for each column in the dataset stored in df_cleaned after removing all NaN values. For example:

- Complaint type has 15 unique types like noise, illegal parking etc.
- Agency has 1 value mostly NYPD.
- City has 53 unique names.
- Borough includes 5 areas: Bronx, Brooklyn, Manhattan, Queens, and Staten Island

  And so on.

This helps us see the variety and range of values in each feature for further analysis.

```python
#getting unique values from all the columns
unique_info = []

for column in df_cleaned.columns:
    unique = df_cleaned[column].nunique()
    dtype = str(df_cleaned[column].dtype)

    if unique < 10 and df_cleaned[column].dtype == 'object':
        sample_values = ', '.join(str(x) for x in df_cleaned[column].unique()[:5])
        if unique > 5:
            sample_values += ", ..."
    else:
        if df_cleaned[column].dtype == 'object':
            sample_values = f"Too many to display ({unique} unique values)"
        else:
            sample_values = f"Range: {df_cleaned[column].min()} to {df_cleaned[column].max()}"

    unique_info.append({
        'Column': column,
        'Data Type': dtype,
        'Unique Values': unique,
        'Sample Values': sample_values
    })

unique_info_df = pd.DataFrame(unique_info)
(unique_info_df)
```

	Column	Data Type	Unique Values	Sample Values
0	Unique Key	int64	291107	Range: 30279480 to 32310649
1	Created Date	datetime64[ns]	251970	Range: 2015-03-29 00:33:01 to 2015-12-31 23:59:45
2	Closed Date	datetime64[ns]	231991	Range: 2015-03-29 00:57:23 to 2016-01-03 16:22:00
3	Agency	object	1	NYPD
4	Complaint Type	object	15	Too many to display (15 unique values)
5	Descriptor	object	41	Too many to display (41 unique values)
6	Location Type	object	14	Too many to display (14 unique values)
7	Incident Zip	float64	200	Range: 83.0 to 11697.0
8	City	object	53	Too many to display (53 unique values)
9	Status	object	1	Closed
10	Resolution Description	object	12	Too many to display (12 unique values)
11	Borough	object	5	MANHATTAN, QUEENS, BRONX, BROOKLYN, STATEN ISLAND
12	Latitude	float64	123013	Range: 40.49913462 to 40.9128688
13	Longitude	float64	123112	Range: -74.25493722 to -73.70076037
14	Request_Closing_Time	float64	47134	Range: 0.016666666666666666 to 592.8727777777777

Figure 5 Unique values from all the columns

2.6 Writing a python program to remove the NaN missing values from updated dataframe.

All rows with a NaN (Not a Number) values in any column are deleted to keep the dataset complete, clean and meaningful which prevents mistakes and errors during analysis and maintains overall good data quality.

```
df_cleaned = df_cleaned.dropna() #removes any rows in the dataset that contain missing (NaN) values in any column
After_cleaned_NaN_count=df_cleaned.isna().sum()
After_cleaned_NaN_count
```

```
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 7 Removing NaN Values from all columns

3 Data Analysis

Data analysis is a process of examining, cleaning, transforming, and applying meaning to data to find useful information, make conclusions and make decisions. It focuses on using different tools and techniques to spot patterns, relationships, and trends in data that contain meaningful insights (Eldridge, 2025).

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

This following summary statistics shows:

- Sum:

This adds all the values in the column for data like closing time it shows the total hours spent on all closed requests.

- Mean:

This shows the average value for each column. For example, in request_closing_hours it tells us how many hours it usually takes to close a service request. High average means slower response time. The mean latitude (40.72) and longitude (-73.92) suggest the data is likely based around a location in New York.

- Standard Deviation:

This measures how much the values differ from the mean. A large standard deviation in request_closing_hours means some requests takes much longer or shorter than average to close.

- Skewness:

Skewness shows if the data is balanced or not. A positive skew means most requests are closed quickly but a few take a long time. A negative skew means the exact opposite of positive skew. The skewness is negative for Incident Zip (-2.55), suggesting the distribution of ZIP codes is skewed to the left which means more incidents happen in lower numbered ZIP codes.

- Kurtosis:

It indicates how peaked the data is. High kurtosis means the most values are close to the average, but a few are extreme. Low kurtosis means the value are more equally spread out. A high kurtosis value for Request_Closing_Time(14.29) suggests that most closing times are around the mean, but there are some extreme values too.

The very high kurtosis of Request_Closing_Hours suggests that, while most requests have closing hours around the average (4.31 hours), there are some outliers with much higher closing times.

```
summary_stats = df_cleaned.describe().T[['mean', 'std']] #shows mean and standard deviation from df_cleaned dataset
summary_stats['sum'] = df_cleaned.select_dtypes(include=np.number).sum() #create a new column sum and prints total for each numeric column
summary_stats['skewness'] = df_cleaned.select_dtypes(include=np.number).skew() #measures asymmetry of the data distribution
summary_stats['kurtosis'] = df_cleaned.select_dtypes(include=np.number).kurtosis() #measure of the tailedness of a distribution

summary_stats
```

]:

	mean	std	sum	skewness	kurtosis
Unique Key	31301576.242739	575377.738707	9.112108e+12	0.016898	-1.176593
Created Date	2015-08-14 11:25:43.378747648	NaN	NaN	NaN	NaN
Closed Date	2015-08-14 15:44:15.511413248	NaN	NaN	NaN	NaN
Incident Zip	10857.977349	580.280774	3.160833e+09	-2.553956	37.827777
Latitude	40.725681	0.082411	1.185553e+07	0.123114	-0.734818
Longitude	-73.925035	0.078654	-2.152010e+07	-0.312739	1.455600
Request_Closing_Time	4.308926	6.062641	1.254358e+06	14.299525	849.777081
Request_Closing_Hours	4.308926	6.062641	1.254358e+06	14.299525	849.777081

Figure 8 Summary Statistics

3.2 Writing a Python program to calculate and show correlation of all variables

This code creates a correlation matrix which shows how strongly numeric variables are related to each other. This matrix helps identify hidden patterns between features.

The value ranges from -1 to 1:

- 1 means perfect positive correlation when one goes up other goes up.
- -1 means perfect negative correlation when one goes up other goes down.
- 0 means no correlation.

Below are the main insights:

a) The request_closing_hours has a positive correlation with request_ closing_time which means that as the closing hour increases the time too close requests also increases.Both these fields have low correlations with other variables, showing that the time taken to close a request is independent of location (ZIP, latitude, longitude) and unique identifier.

b) Latitude and Longitude have a positive correlation (0.369), which is typical as they both represent geographical locations, there is a small correlation between Latitude and Incident Zip -0.499 showing that geography influences the ZIP codes of incidents.

```
[32]: correlation_matrix = df_cleaned.corr(numeric_only=True) #corealtion of all variables
      correlation_matrix
```

[32]:

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

Figure 9 Correlation between variables

In [9]:
```
# Plot correlation heatmap
sns.heatmap(correlation_matrix, cmap="YlGnBu", annot=True)

# Display heatmap
plt.show()
```

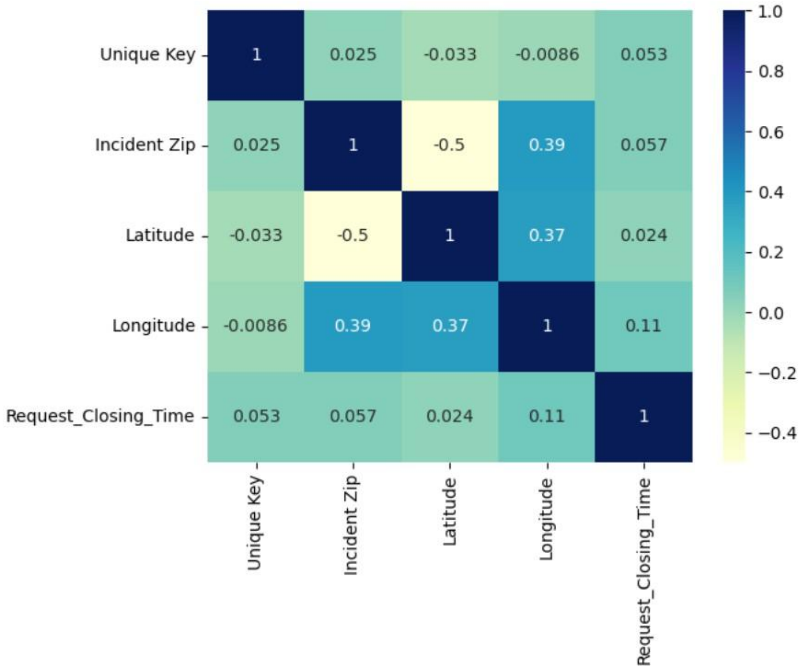

Figure 10 Heatmap of Correlation

4 Data Exploration

Data Exploration is initial investigation of a data to find patterns, trends and anomalies. It provides analysts with an opportunity to know the characteristics of data and their relationship, before a analysis is conducted where they get to learn about the structure of the dataset and insights from it (GeeksforGeeks, 2024).

4.1 Providing four major insights through visualization that I come up after data mining

Insight 1: Complaints Over Time

In 2015 AD, complaints were low with fewer number less than 5000 in March but in the month of May there was a sudden increase in complaints around 35,000 which is the highest number of complaints throughout the year. After the sudden increasement of the complaints, the number of complaints suddenly decrease month by month. In November 2015, complaints had dropped by about 3000-4000 which shows official were very concerned about the complaints.

```
#complaints over time
df['Created Month'] = df['Created Date'].dt.to_period('M').astype(str)
complaints_over_time = df.groupby('Created Month').size()
plt.figure(figsize=(14,6))
complaints_over_time.plot()
plt.title('Complaints Over Time')
plt.ylabel('Number of Complaints')
plt.xlabel('Month')
plt.xticks(rotation=45)
plt.show()
```

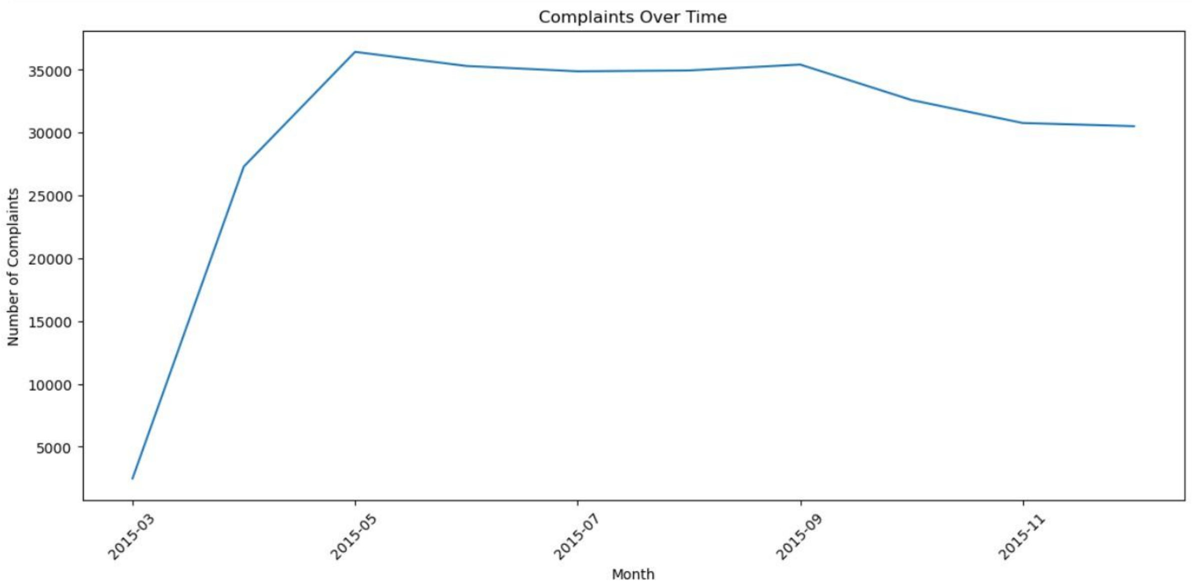

Figure 11 Distribution of Request Closing Time

Insight 2: Complaint Types and their Frequency

1) **Blocked Driveway:** The most frequent complaint is Blocked Driveway with 70,000 complaints which is the highest number.
2) **Illegal Parking:** Illegal Parking is the second highest number of complaints with over 50,000 complaints.
3) **Noise-Related Complaints:** Complaints related to noise are with categories like Noise - Streets/Sidewalk and Noise Commercial having thousands of complaints each.
4) **Vehicle Issues:** Derelict Vehicle and Noise Vehicle are also common complaints, but they are less frequent than noise or parking related complaints.
5) **Less Frequent Complaints:** There are fewer complaints about issues like Traffic, Animal Abuse, and Vending, with each having around 5,000-10,000 complaints.
6) **Rare Complaints:** The few common complaints are Drinking, Posting Advertisement, Disorderly Youth, and Graffiti, each with under 2,000 complaints.

In [11]:
```
# Bar chart for the complaint types and their frequency
complaint_count = df_cleaned['Complaint Type'].value_counts()

plt.figure(figsize=(10, 6))
complaint_count.plot(kind='bar', color='lightcoral')
plt.title('Frequency of Different Complaint Types')
plt.xlabel('Complaint Type')
plt.ylabel('Frequency')
plt.xticks(rotation=90)
plt.grid(True)
plt.show()
```

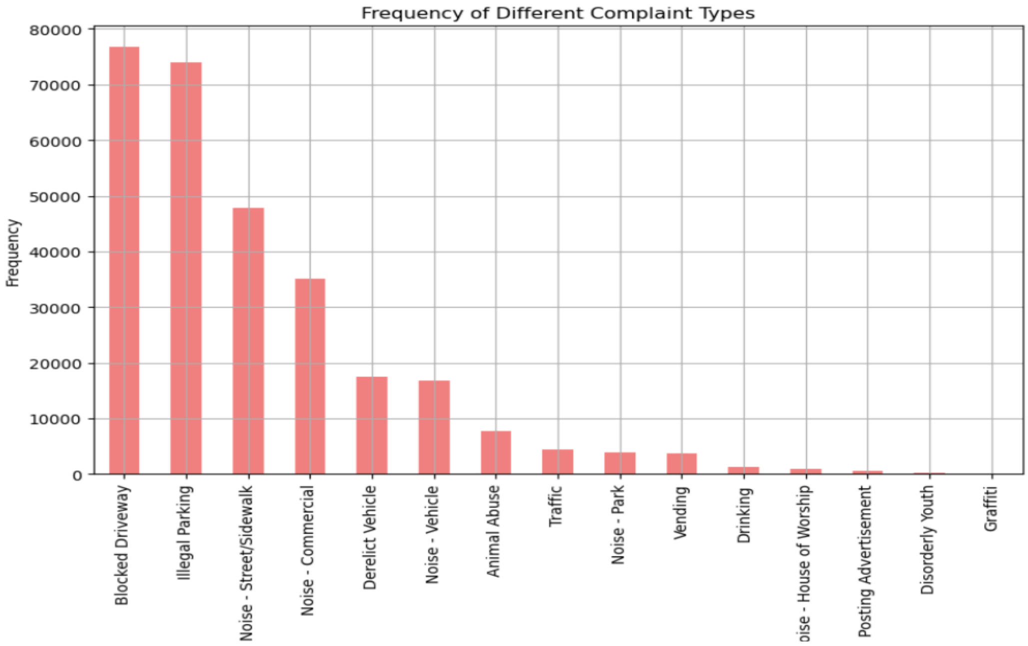

Figure 12 Frequency of different complaint types

Insight 3: Request Closing Time by Borough

The graph sows that the Brooklyn has the highest number of complaints with nearly 90,000 complaints and the second highest number of complaints is in Queens with just under 80,000 complaints. Manhattan and Bronx have around 50,000 and 30,000 complaints respectively while Staten Island has the fewest number of complaints among all. Thus, pattern shows that the larger or more populated Boroughs have more complaints.

In [12]:
```
plt.figure(figsize=(10, 6))
borough_counts = df_cleaned['Borough'].value_counts()
borough_counts.plot(kind='bar', color='lightcoral')
plt.title('Requests by Borough', fontsize=16)
plt.xlabel('Borough', fontsize=12)
plt.ylabel('Count of Complaints', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

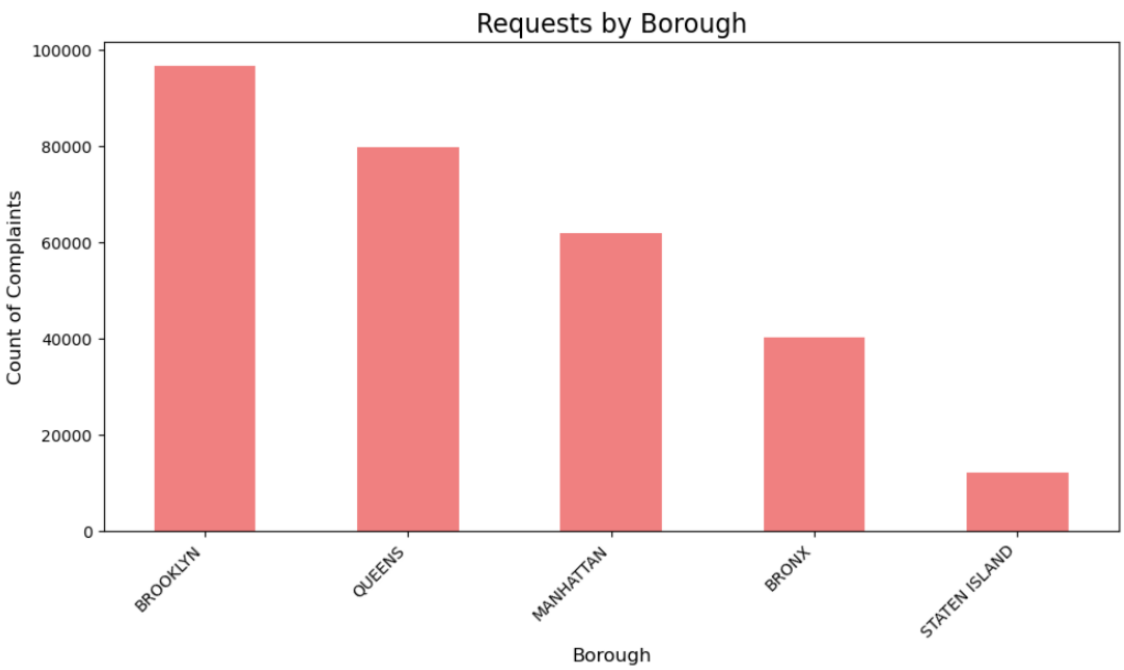

Figure 13 Request by Borough

Insight 4: Complaint Types and Request Closing Time

1) **Shortest Closing Times:**

   Posting Advertisement has the shortest average closing time with just over 1 second. Other complaints type like Noise Commercial, Noise House of Worship, Noise Park and Noise Vehicle also have shortest closing time with average of 2 to 3 seconds.

2) **Longest Closing Times:**

   Decrepit Vehicle has the longest closing time which is more than 7 seconds. Graffiti, Animal Abuse and Blocked Driveway also have long closing time than others between 5 to 6 seconds.

Complaint types related to Noise and Traffic have shorter closing times whereas Vehicle related complaints and public disturbance complaints like graffiti and animal abuse have higher closing time compared to others which shows they are more complex to solve and requires more time to solve.

```
In [13]: # Bar chart showing complaint types categorized by average request closing time
avg_closing_time_by_complaint = df_cleaned.groupby('Complaint Type')['Request_Closing_Time'].mean()

plt.figure(figsize=(10, 6))
avg_closing_time_by_complaint.sort_values().plot(kind='bar', color='lightgreen')
plt.title('Average Request Closing Time by Complaint Type')
plt.xlabel('Complaint Type')
plt.ylabel('Average Closing Time (seconds)')
plt.xticks(rotation=90)
plt.grid(True)
plt.show()
```

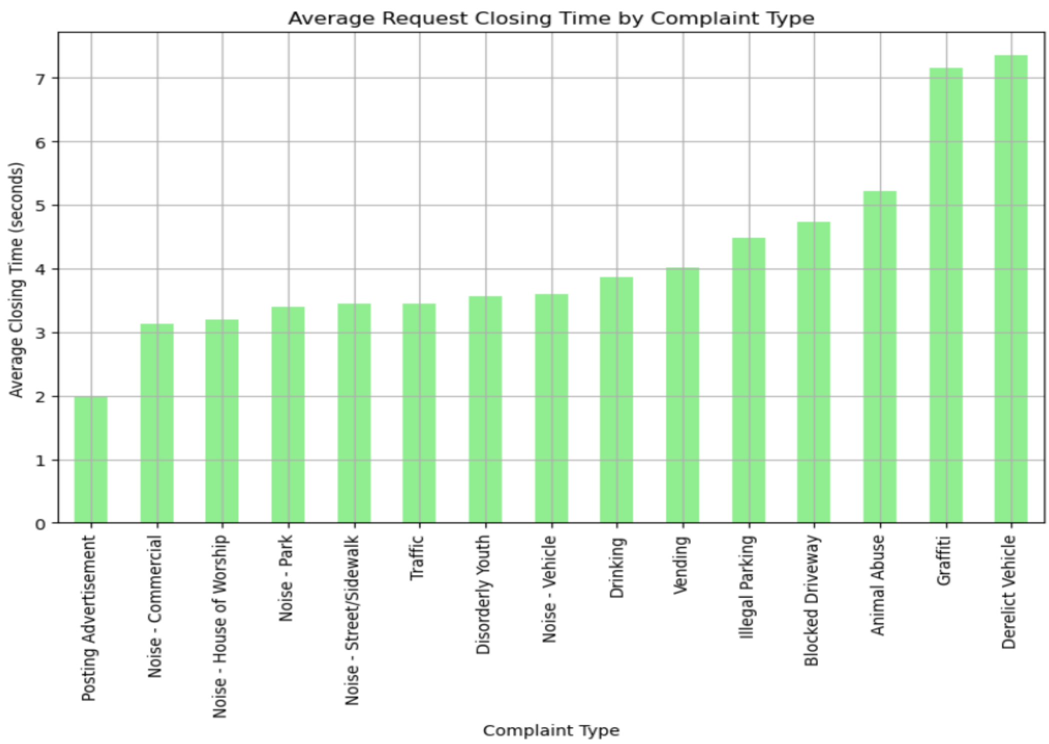

Figure 14 Average Request Closing Time by Complaint Type

4.2 Arranging the complaint types according to their average 'Request_Closing_Time', categorized by various locations.

This chart shows the average request closing time for different complaint types across five boroughs.

1) Bronx has the highest average request closing time for Blocked Driveway and Animal Abuse complaints.
2) Brooklyn has similar closing time across most complaints, but Vending complaints takes longer to close as compared to others.
3) Manhattan has a lower request closing time compared to other boroughs for most complaint types.
4) Queens has similar closing times to Brooklyn with Blocked Driveway and Disorderly Youth complaints are taking longer to close.
5) Staten Island has the shortest closing time overall with a drop in the average closing time for Disorderly Youth and Traffic complaints.
6) Traffic and Vending complaints have shorter closing times across most of the boroughs.

The variation in response times and closing times shows that some boroughs handle complaints more quickly than others depending on the type of complaint.

```
# Grouping by Borough and Complaint Type to analyze
avg_closing_time_by_borough_complaint = df_cleaned.groupby(['Borough', 'Complaint Type'])['Request_Closing_Time'].mean().unstack()

plt.figure(figsize=(12, 8))
avg_closing_time_by_borough_complaint.plot(kind='bar', stacked=True, figsize=(10, 6))
plt.title('Average Request Closing Time by Borough and Complaint Type')
plt.xlabel('Borough')
plt.ylabel('Average Closing Time (seconds)')
plt.xticks(rotation=45)
plt.legend(title='Complaint Type', bbox_to_anchor=(1, 1))
plt.show()
```

<Figure size 1200x800 with 0 Axes>

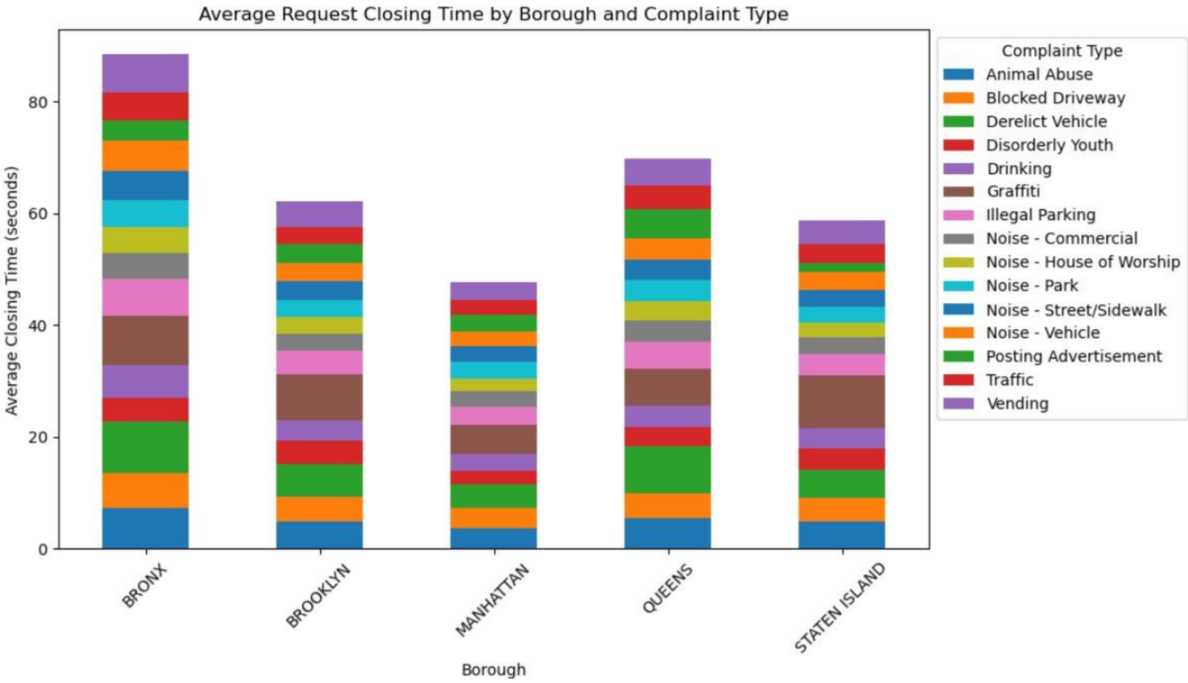

Figure 15 Average Request Closing Time by Borough and Complaint Type

5 Statistical Testing

Null Hypothesis (H0)

The null hypothesis (H0) is a statement that says there is no change, effect, or difference. It represents the current belief or status quo that we accept unless there is strong evidence against it.

Alternate Hypothesis (H1)

The alternate hypothesis (H1) is a statement that goes against the null hypothesis. It represents what we want to prove or find evidence for in the test. It challenges the current belief (the null hypothesis), and we try to gather enough proof to reject the null and support the alternate hypothesis instead.

P-Value

The p-value is a number between 0 and 1 that shows how strong the evidence is against the null hypothesis. If this number is small (usually less than 0.05), it means the data we observed doesn't fit well with the idea that the null hypothesis is true.

In simple terms, the p-value tells us how likely it is to get the results we did if the null hypothesis was correct. A small p-value means it's unlikely that the results happened just by chance.

Figure 16 Null Hypothesis, Alternate Hypothesis And P-Value

Statistical testing refers to the process of conducting mathematical analysis to check and make decisions based on whether an observed pattern in data is real or simply a result of chance (Wikipedia, 2025). It is important due to the following reasons:

1) Makes sure findings are based on evidence, not coincidence.
2) Prevents recognizing patterns that do not exist.
3) Determines meaningful differences between groups
4) Displays other factors that could affect results.
5) Tests if there is a significant correlation between variables.

5.1 Test 1: Whether the average response time across complaint types is similar or not.

We will perform an ANOVA test to check if there are significant differences in average response times between different complaint types.

Null Hypothesis (H0): The average response times across all complaint types are the same.

Alternate Hypothesis (H1): At least one complaint type has a different average response time.

```
In [20]: from scipy import stats

         # Group data by Complaint Type and calculate the Request Closing Time
         complaint_group = df_cleaned.groupby('Complaint Type')['Request_Closing_Time']

         # Perform ANOVA test
         f_stat, p_value = stats.f_oneway(*[group.dropna() for name, group in complaint_group])


         print(f"ANOVA test result: F-statistic = {f_stat}, p-value = {p_value}")

         # Interpret results
         if p_value < 0.05:
             print("Reject the null hypothesis: There is a significant difference in average response time across complaint types.")
         else:
             print("Fail to reject the null hypothesis: There is no significant difference in average response time across complaint types.")
```

```
ANOVA test result: F-statistic = 578.9120337398356, p-value = 0.0
Reject the null hypothesis: There is a significant difference in average response time across complaint types.
```

Figure 17 Evidence of Test 1

The p-value is less than 0.05, we reject the null hypothesis (H0) and conclude that there are statistically significant differences in the average response times across complaint types.

5.2 Test 2: Whether the type of complaint or service requested and location are related.

We will use the Chi-Square Test of Independence to check whether the type of complaint is related to the location (borough).

Null Hypothesis (H0): Complaint type and location are independent.

Alternate Hypothesis (H1): Complaint type and location are related.

```
In [19]: # Create a contingency table for Complaint Type vs Borough
contingency_table = pd.crosstab(df_cleaned['Complaint Type'], df_cleaned['Borough'])

# Perform Chi-Square test
chi2_stat, p_value, dof, expected = stats.chi2_contingency(contingency_table)

print(f"Chi-Square test result: Chi2-statistic = {chi2_stat}, p-value = {p_value}")

# Interpret results
if p_value < 0.05:
    print("Reject the null hypothesis: There is a relationship between complaint type and location (borough).")
else:
    print("Fail to reject the null hypothesis: There is no relationship between complaint type and location (borough).")

Chi-Square test result: Chi2-statistic = 73264.62164334783, p-value = 0.0
Reject the null hypothesis: There is a relationship between complaint type and location (borough).
```

Figure 18 Evidence of Test 2

The p-value is less than 0.05, we reject the null hypothesis (H0) and conclude that complaint type and location are related.

6 References

AWS, 2025. *What is Data Preparation?.* [Online] Available at: https://aws.amazon.com/what-is/data-preparation/
[Accessed 14 May 2025].

Eldridge, S., 2025. *data analysis.* [Online] Available at: https://www.britannica.com/science/data-analysis
[Accessed 14 May 2025].

GeeksforGeeks, 2024. *What is Data Exploration and its process?.* [Online] Available at: https://www.geeksforgeeks.org/what-is-data-exploration-and-its-process/
[Accessed 14 May 2025].

IBM, 2021. *Data Understanding Overview.* [Online] Available at: https://www.ibm.com/docs/en/spss-modeler/saas?topic=understanding-data-overview
[Accessed 14 May 2025].

Wikipedia, 2025. *Statistical hypothesis test.* [Online] Available at: https://en.wikipedia.org/wiki/Statistical_hypothesis_test
[Accessed 14 May 2025].