Data science Project write up

Nicolas Mkhabele

Customer Service Requests Analysis

First I used the pandas library to import the csv file containing the data to the notebook.

```
[2] import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
```

```
[3] df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/311_Service_Requests_from_2010_to_Present.csv')
```

```
<ipython-input-3-bd77e6c0f894>:1: DtypeWarning: Columns (48,49) have mixed types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/311_Service_Requests_from_2010_to_Present.csv')
```

I then used the head function to have a look at the data so I can have a good idea of what I'm working with.

```
[4] df.head()
```

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	Incident Zip	Incident Address	...	Bridge Highway Name
0	32310363	12/31/2015 11:59:45 PM	01/01/2016 12:55:15 AM	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Music/Party	Street/Sidewalk	10034.0	71 VERMILYEA AVENUE	...	NaN
1	32309934	12/31/2015 11:59:44 PM	01/01/2016 01:26:57 AM	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewalk	11105.0	27-07 23 AVENUE	...	NaN
2	32309159	12/31/2015 11:59:29 PM	01/01/2016 04:51:03 AM	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewalk	10458.0	2897 VALENTINE AVENUE	...	NaN
3	32305098	12/31/2015 11:57:46 PM	01/01/2016 07:43:13 AM	NYPD	New York City Police Department	Illegal Parking	Commercial Overnight Parking	Street/Sidewalk	10461.0	2940 BAISLEY AVENUE	...	NaN
4	32306529	12/31/2015 11:56:58 PM	01/01/2016 03:24:42 AM	NYPD	New York City Police Department	Illegal Parking	Blocked Sidewalk	Street/Sidewalk	11373.0	87-14 57 ROAD	...	NaN

The next thing to do is check the shape of the data using the following command:

```
[5] df.shape

(364558, 53)
```

Ad can be seen, we have 364558 rows and 53 columns. Its time to clean the dataset and remove all missing values and duplicates, for that we use the isnull function to check for any missing values first.

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

```
Unique Key                          0
Created Date                        0
Closed Date                      2381
Agency                              0
Agency Name                         0
Complaint Type                      0
Descriptor                       6501
Location Type                     133
Incident Zip                     2998
Incident Address                51699
Street Name                     51699
Cross Street 1                  57188
Cross Street 2                  57805
Intersection Street 1          313438
Intersection Street 2          314046
Address Type                     3252
City                             2997
Landmark                       364183
Facility Type                    2389
Status                              0
Due Date                            3
Resolution Description              0
Resolution Action Updated Date   2402
Community Board                     0
Borough                             0
X Coordinate (State Plane)       4030
Y Coordinate (State Plane)       4030
Park Facility Name                  0
Park Borough                        0
School Name                         0
School Number                       0
School Region                       1
School Code                         1
School Phone Number                 0
```

We had a lot of missing values so first we use feature selection to get rid of all the unnecessary columns for our analysis by using the drop function as follows:

```
[7] df = df.drop(columns=['School or Citywide Complaint','Vehicle Type','Taxi Company Borough', 'Taxi Pick Up Location', 'Bridge Highway Name', 'Bridge Highway Direction','Road Ramp'])
```

```
[8] df = df.drop(columns=['Bridge Highway Segment', 'Garage Lot Name', 'Ferry Direction', 'Ferry Terminal Name', 'Descriptor', 'X Coordinate (State Plane)', 'Y Coordinate (State Plane)', 'Location'])
```

```
[9] df = df.drop(columns=['Landmark', 'Intersection Street 2', 'Intersection Street 1', 'Cross Street 2', 'Cross Street 1', 'Street Name', 'Incident Address'])
```

Once we have removed the unnecessary columns. We remove rows with missing values and check our data once again.

```
[10] df.isna().sum()
```

```
Unique Key                          0
Created Date                        0
Closed Date                      2381
Agency                              0
Agency Name                         0
Complaint Type                      0
Location Type                     133
Incident Zip                     2998
Address Type                     3252
City                             2997
Facility Type                    2389
Status                              0
Due Date                            3
Resolution Description              0
Resolution Action Updated Date   2402
Community Board                     0
Borough                             0
Park Facility Name                  0
Park Borough                        0
School Name                         0
School Number                       0
School Region                       1
School Code                         1
School Phone Number                 0
School Address                      0
School City                         0
School State                        0
School Zip                          1
School Not Found                    0
Latitude                         4030
Longitude                        4030
dtype: int64
```

```
[11] df = df.drop(columns=['Location Type', 'Incident Zip', 'Address Type', 'Facility Type', 'Resolution Action Updated Date'])
```

```
[12] df = df.dropna()
```

Once we have removed all missing values and checked our data we remove the possible duplicates in the data before checking the shape of the dataframe .

```
df.isna().sum()
```

```
Unique Key                0
Created Date              0
Closed Date               0
Agency                    0
Agency Name               0
Complaint Type            0
City                      0
Status                    0
Due Date                  0
Resolution Description    0
Community Board           0
Borough                   0
Park Facility Name        0
Park Borough              0
School Name               0
School Number             0
School Region             0
School Code               0
School Phone Number       0
School Address            0
School City               0
School State              0
School Zip                0
School Not Found          0
Latitude                  0
Longitude                 0
dtype: int64
```

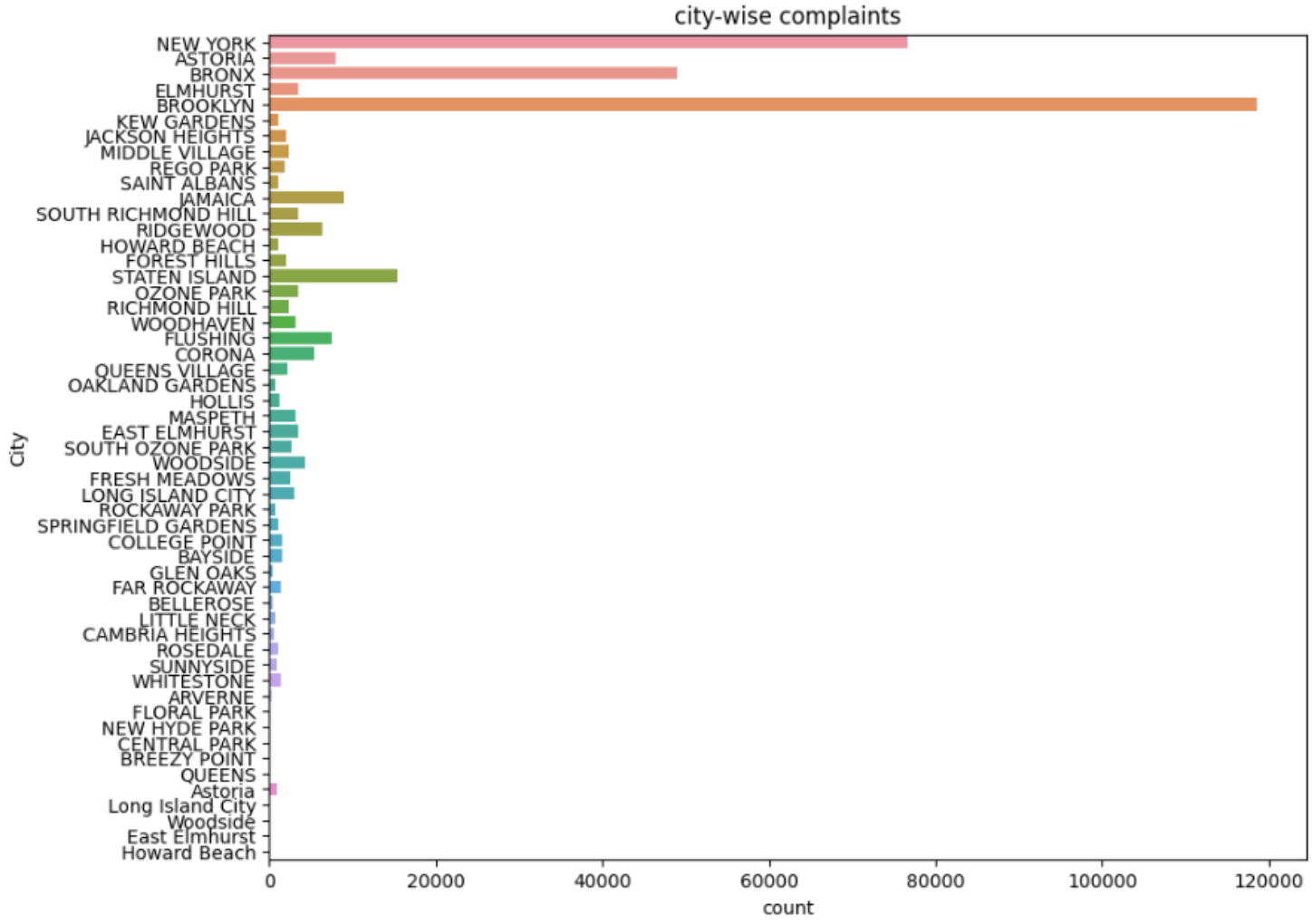
```
[14] df = df.drop_duplicates()
```

```
[15] df.shape
```

```
(360428, 26)
```

Now that our data in clean. We use exploratory data analysis. The following code is used to display the city-wise complaints.

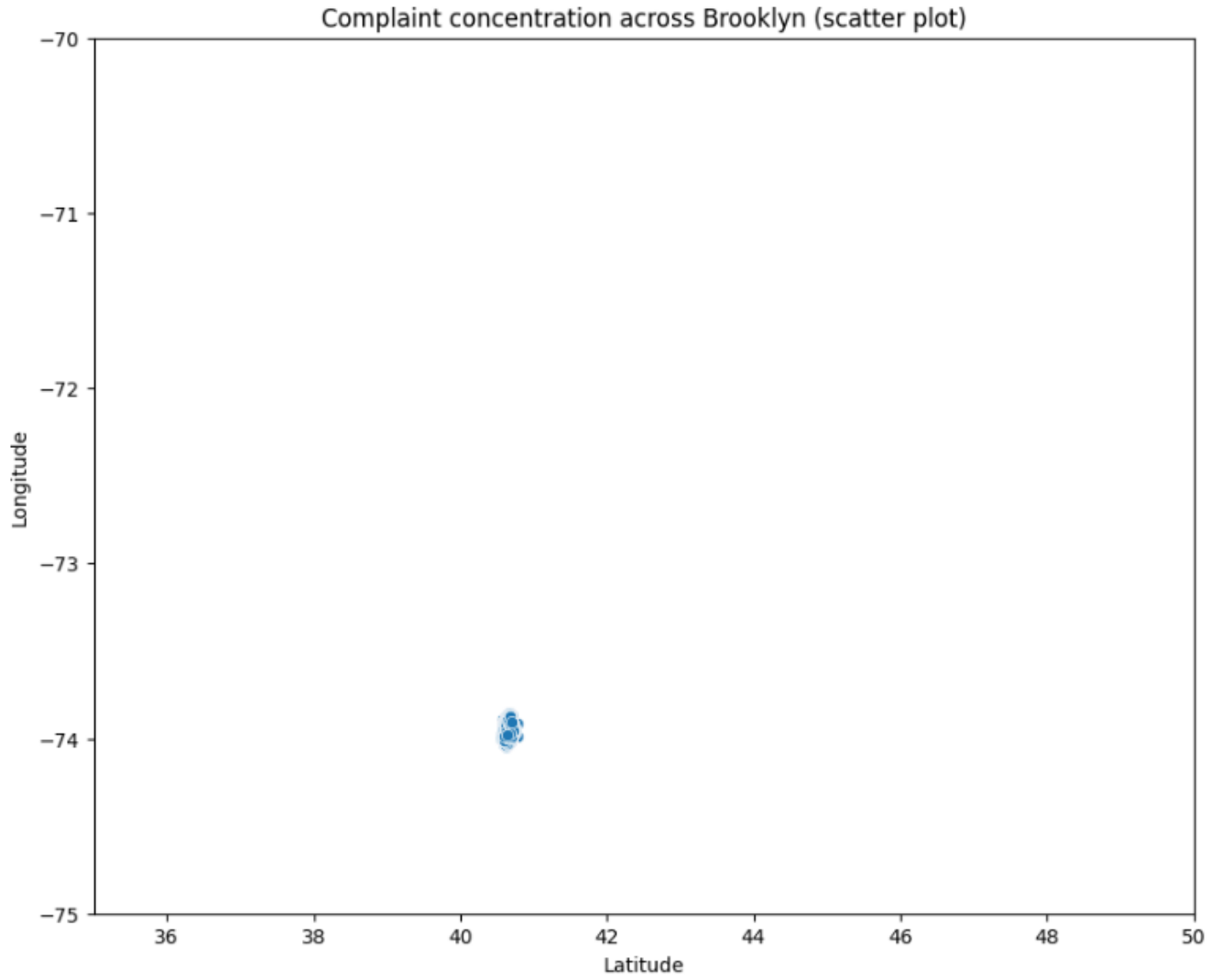
```
plt.figure(figsize=(10,8))
plt.title('city-wise complaints')
sns.countplot(y='City', data=df)
plt.show()
```



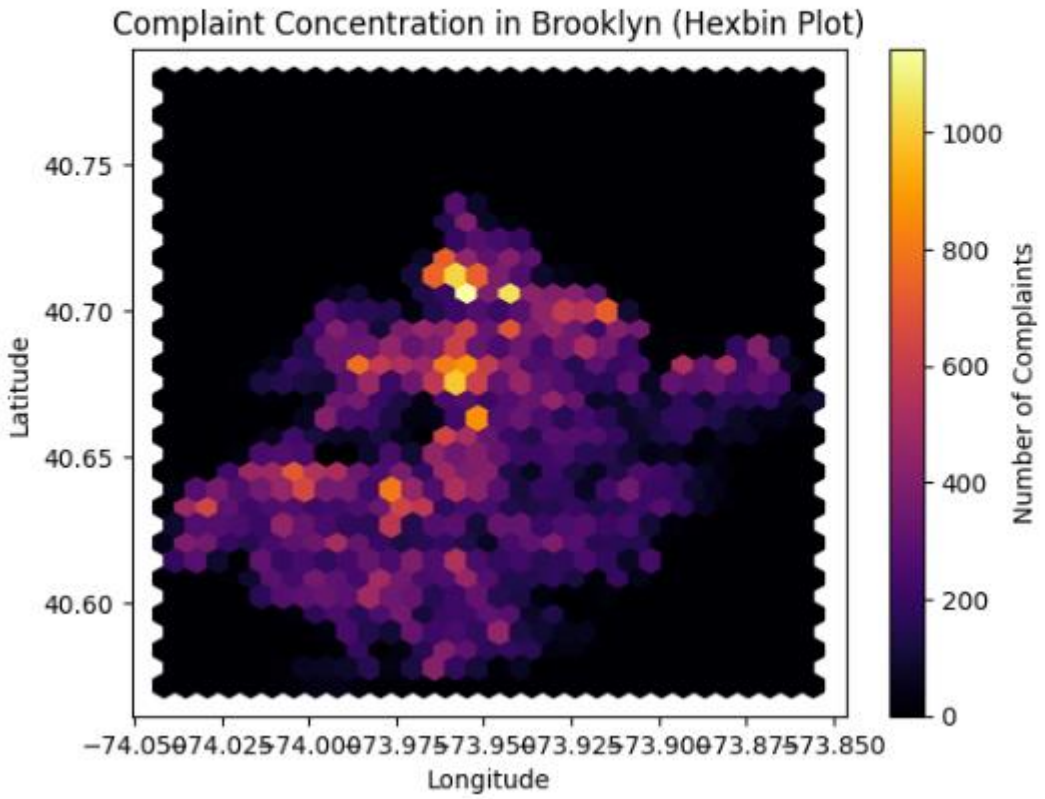
We also display the Brooklyn scatter plot and hexbin plot and get the following results:

```
[43] brooklyn_data = df[df['City'] == 'BROOKLYN']
```

```
[51] plt.figure(figsize=(10,8))
     plt.title("Complaint concentration across Brooklyn (scatter plot)")
     plt.ylim(-75,-70)
     plt.xlim(35, 50)
     sns.scatterplot(x=brooklyn_data['Latitude'], y=brooklyn_data['Longitude'],data=brooklyn_data)
     plt.show()
```

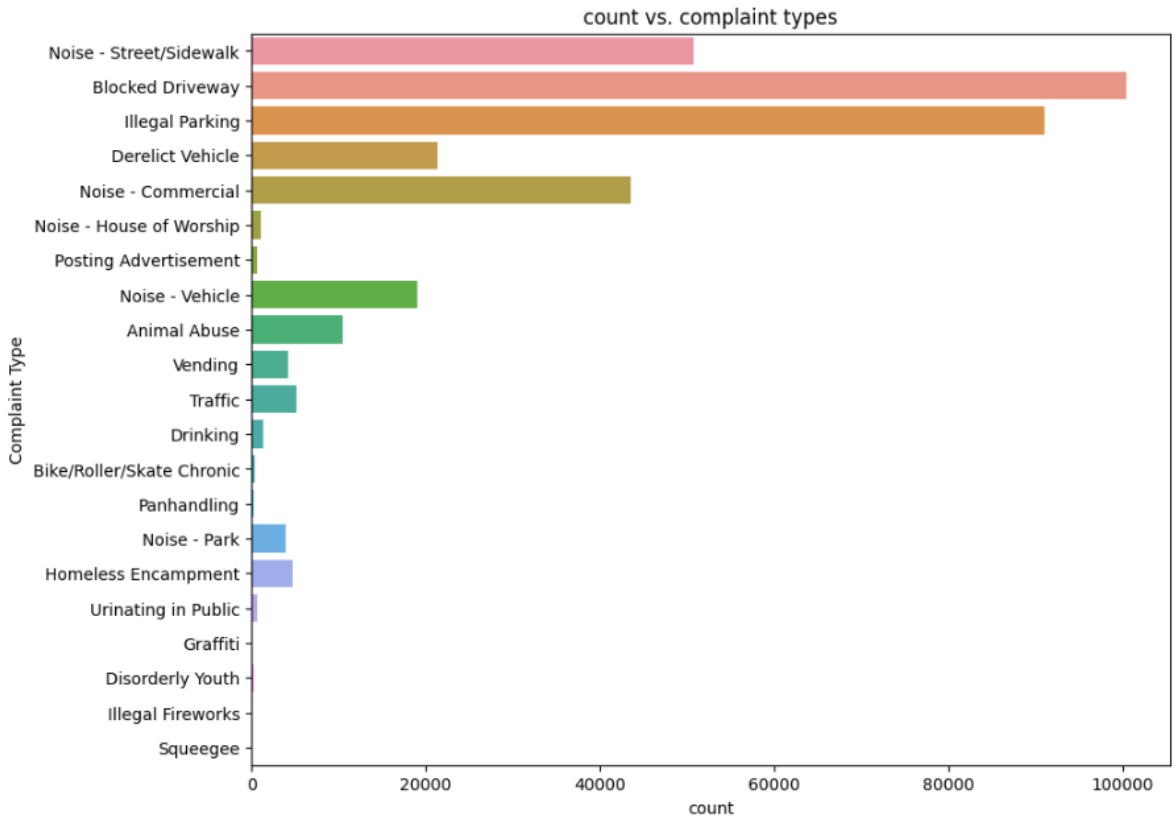


```
plt.hexbin(brooklyn_data['Longitude'], brooklyn_data['Latitude'], gridsize=30, cmap='inferno')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.title('Complaint Concentration in Brooklyn (Hexbin Plot)')
plt.colorbar(label='Number of Complaints')
plt.show()
```



To understand our data more, we Plot a bar graph of count vs. complaint types using the following code:

```
plt.figure(figsize=(10,8))
plt.title("count vs. complaint types")
sns.countplot(y='Complaint Type', data=df)
plt.show()
```

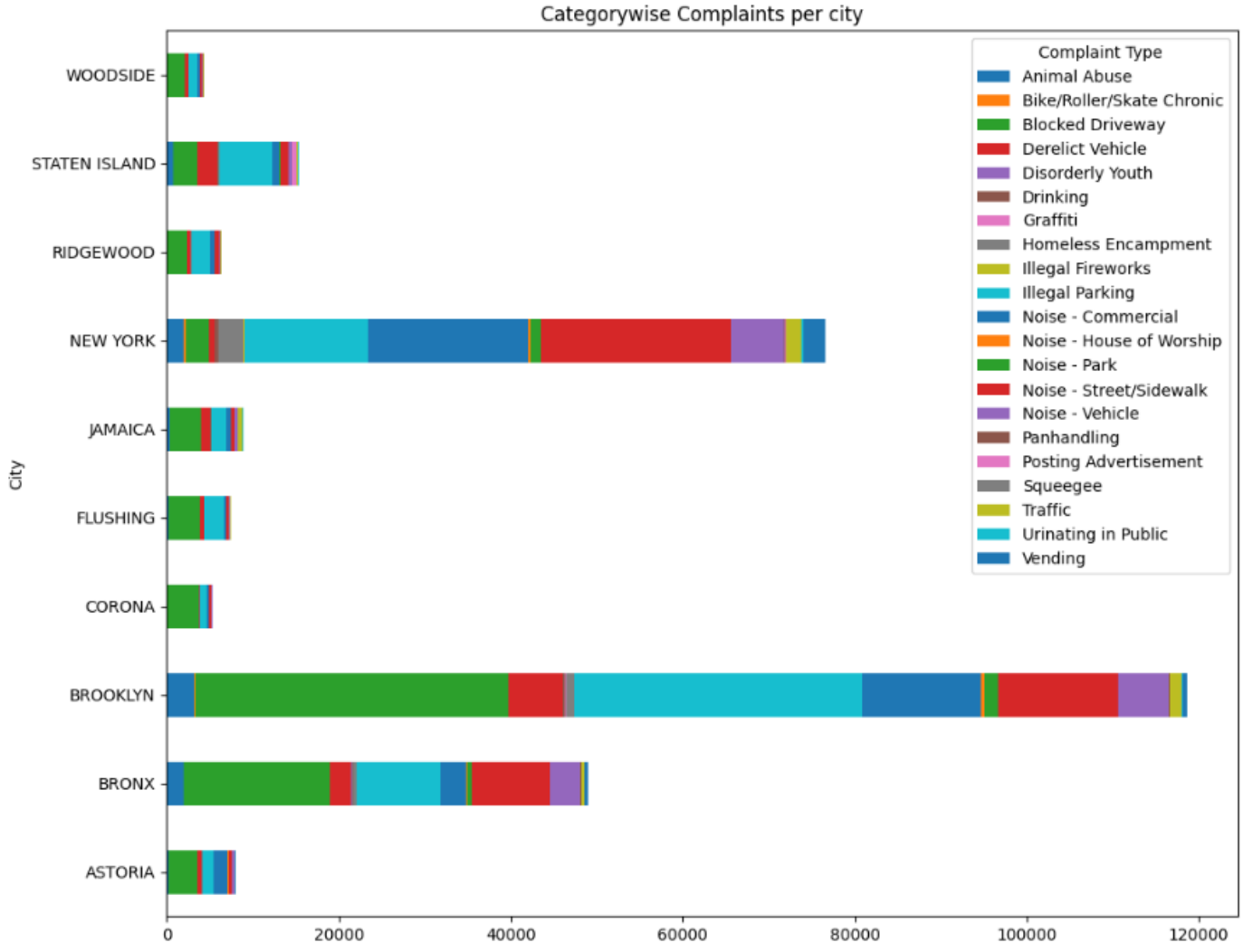


Find major types of complaints:

1. Plot a bar graph of count vs. complaint types
2. Find the top 10 types of complaints
3. Display the types of complaints in each city in a separate dataset

```
[54] top10cities = df['City'].value_counts().head(10).index.to_list()
     dstop5 = df[df.City.isin(top10cities)]
     #citywise complaint counts(typewise)
     df1 = pd.crosstab(dstop5['City'],dstop5['Complaint Type'])
```

```
[55] ##citywise complaint counts(typewise)
     df1.plot(kind='barh',stacked=True,figsize=(12,10))
     plt.title('Categorywise Complaints per city')
     plt.show()
```



Lastly we check if the average response time across various types of complaints and get the following results:

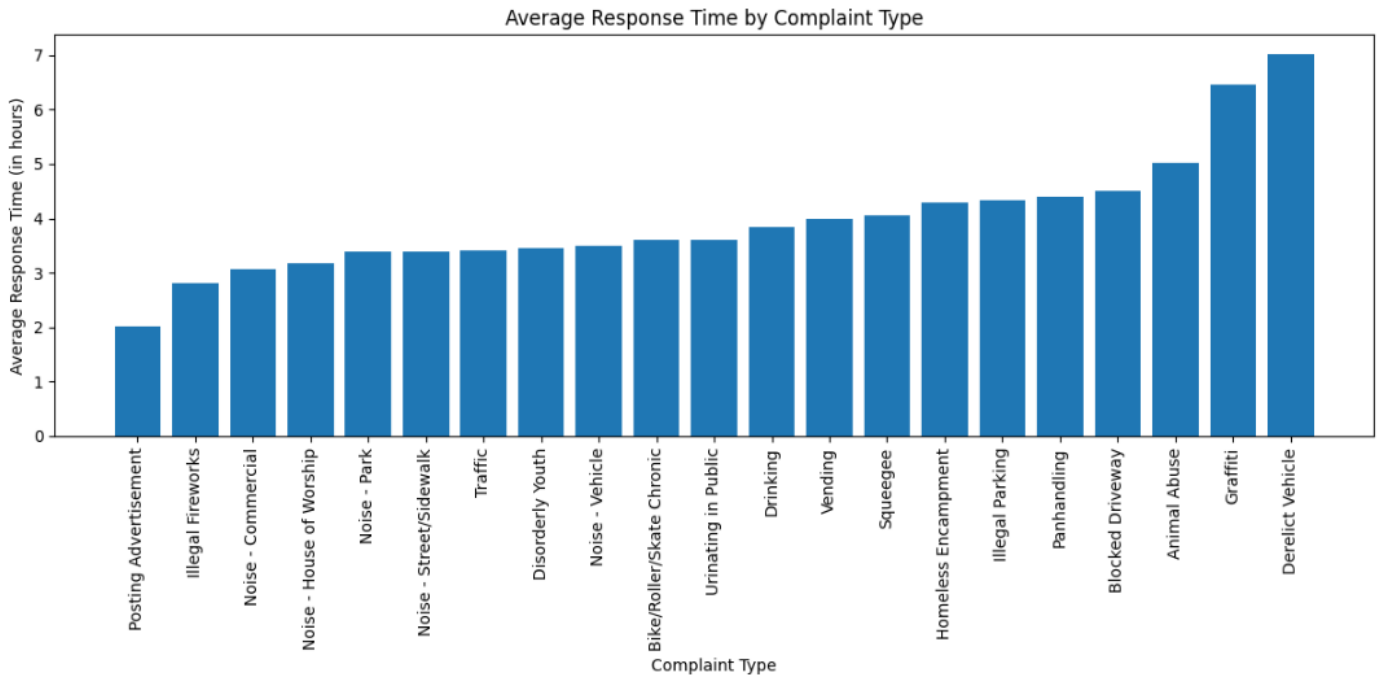
Check if the average response time across various types of complaints

```
df['Closed Date'] = pd.to_datetime(df['Closed Date'])
df['Created Date'] = pd.to_datetime(df['Created Date'])
```

```
[ ] df['Response time'] = df['Closed Date'] - df['Created Date']
    df['Response Time'] = df['Response time'].dt.total_seconds() / 3600
```

```
[ ] filtered_data = df[['Complaint Type', 'Response Time']]
    average_response_time = filtered_data.groupby('Complaint Type')['Response Time'].mean().reset_index()
    sorted_data = average_response_time.sort_values('Response Time')
```

```
[ ] plt.figure(figsize=(12, 6))
    plt.bar(sorted_data['Complaint Type'], sorted_data['Response Time'])
    plt.xlabel('Complaint Type')
    plt.ylabel('Average Response Time (in hours)')
    plt.xticks(rotation=90)
    plt.title('Average Response Time by Complaint Type')
    plt.tight_layout()
    plt.show()
```



The rest of the code and plots are in the python notebook file attached.