311 Customer Service request Analysis

Abstract

This document is prepared to show the analysis of 311 customer Service request calls from New York city. This reveals the understanding of the pattern of the data and visualize the major types of complaints using various data wrangling techniques.

Below are the analysis tasks to be performed:

1) Import the Customer service requests dataset and understand it
2) Performa basic data exploratory analysis
3) Find major types of complaints

Introduction

Customer service is a process through which customers are assisted and supported for the services they have agreed to. In this process the customer complaints are collected, responded and closed within speculated time to satisfy the customers.

Objectives

The objective of this document is to understand the customer service request complaints from Now York city area, the kind of major complaints that has been occurred and the response time for the complaints.

Importing the libraries and dataset:

The libraries that was imported to work on the dataset are:

- Pandas – for analyzing, exploring and manipulating data
- Numpy – for supporting large, multi-dimensional arrays and matrices using functions
- Matplotlib.pyplot – for data visualization and graphical plotting
- Seaborn – for data visualization of statistical graphs

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
## Understand the dataset

# Import the dataset

dataset=pd.read_csv('D:\Simplilearn\Assessments\Data Science with Python\Customer Service Requests-311\Dataset/311_csr.csv',low_m

#Visualize the dataset

dataset.head()
```

	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	Bri High Segn
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	NaN	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	NaN	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	NaN	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	NaN	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	NaN	NaN	

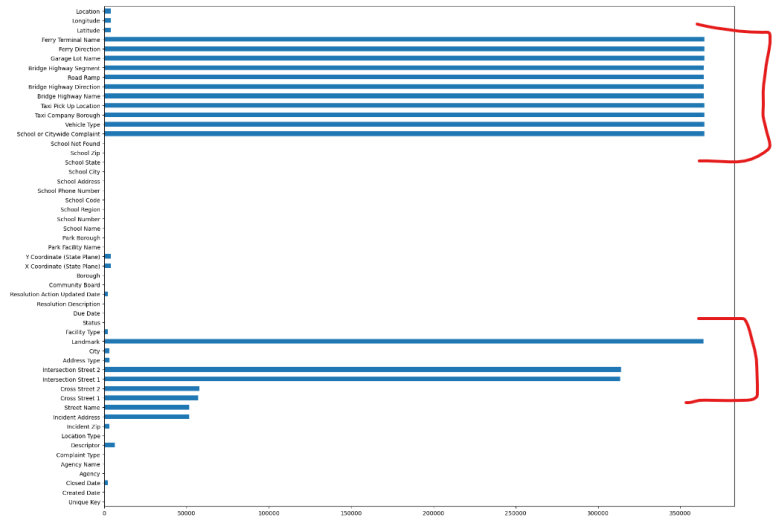
The dataset seems to have many rows and columns for the service requests from New York city. Hence, I checked the shape of the dataset and found that the dataset has 364558 rows and 53 columns.

The display also reveals many columns having null values. Hence, I used df.isna().sum() to check the number of null values in all the columns. I found lots of columns having null values with big numbers.

```
In [7]: dataset.isna().sum()

Out[7]: 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
        School Address                      0
        School City                         0
        School State                        0
        School Zip                          1
        School Not Found                    0
        School or Citywide Complaint   364558
        Vehicle Type                   364558
        Taxi Company Borough           364558
        Taxi Pick Up Location          364558
        Bridge Highway Name            364261
        Bridge Highway Direction       364261
        Road Ramp                      364296
```

I used frequency plot to see the null values using Bar graph

```
# Draw a frequency plot to show null values in each column of the dataframe

df_null=dataset.isna().sum()
plt.figure(figsize=(20,16))
df_null.plot(kind="barh")
plt.show()
```



I used missing value percentage formula to check the highest percentage columns of null values and found many columns having more than 80% of null values.

```
In [11]: ##Check the missing value percentage of all rows in a dataframe, Since in the previous steps we have found large number of null v

         missing_percentage=(dataset.isna().sum(axis=0)/dataset.shape[0])*100
         missing_percentage
```

```
Out[11]: Unique Key                        0.000000
         Created Date                      0.000000
         Closed Date                       0.000000
         Agency                            0.000000
         Agency Name                       0.000000
         Complaint Type                    0.000000
         Descriptor                        1.793598
         Location Type                     0.035894
         Incident Zip                      0.186373
         Incident Address                 14.270923
         Street Name                      14.270923
         Cross Street 1                   15.277337
         Cross Street 2                   15.314059
         Intersection Street 1            86.021200
         Intersection Street 2            86.055437
         Address Type                      0.256504
         City                              0.186097
         Landmark                         99.896459
         Facility Type                     0.004970
         Status                            0.000000
         Due Date                          0.000276
         Resolution Description            0.000000
         Resolution Action Updated Date    0.010768
         Community Board                   0.000000
         Borough                           0.000000
         X Coordinate (State Plane)        0.471317
         Y Coordinate (State Plane)        0.471317
         Park Facility Name                0.000000
         Park Borough                      0.000000
         School Name                       0.000000
         School Number                     0.000000
         School Region                     0.000276
         School Code                       0.000276
         School Phone Number               0.000000
         School Address                    0.000000
         School City                       0.000000
         School State                      0.000000
         School Zip                        0.000276
         School Not Found                  0.000000
         School or Citywide Complaint    100.000000
         Vehicle Type                    100.000000
         Taxi Company Borough            100.000000
```

Since the average missing percentage of null value is <30% which is the industry standard allowed for the null values in a dataset to create a good machine learning model. I removed the columns having more than 80% of null values.

```
In [12]: missing_percentage.mean()
```

```
Out[12]: 27.087349816083353
```

```
In [13]: ## Since the percentage of missing vlaues in the dataset is less than 30% of the industry practice allowed for Null values
         ## Hence we can drop the columns with null values having more than 80% from the dataset

         remove_cols=pd.DataFrame(dataset.columns.to_list()).set_index(0)
         remove_cols=remove_cols[dataset.isna().sum()/dataset.shape[0]*100 < 80].reset_index()
         remove_cols
```

Out[13]:

	0
0	Unique Key
1	Created Date
2	Closed Date
3	Agency
4	Agency Name
5	Complaint Type
6	Descriptor
7	Location Type
8	Incident Zip
9	Incident Address
10	Street Name
11	Cross Street 1
12	Cross Street 2
13	Address Type
14	City
15	Facility Type
16	Status
17	Due Date
18	Resolution Description
19	Resolution Action Updated Date
20	Community Board
21	Borough

I have also noticed many of the School related columns along with Park facility name having unspecified words in it cells.

```
In [15]: ## The School related columns contains most of entries as unspecified
         dataset['School Name'].value_counts()

Out[15]: Unspecified                    362176
         Alley Pond Park - Nature Center       1
         Name: School Name, dtype: int64

In [17]: dataset['School Code'].value_counts()

Out[17]: Unspecified    362176
         Name: School Code, dtype: int64

In [18]: ## Lets remove all school related columns as most of the entries are Unspecified
         dataset=dataset.drop(dataset.filter(regex='School').columns,axis=1)
         dataset
```

Out[18]:

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	Incident Zip	Incident Address	...	Resolution Action Updated Date	Community Board	
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	...	01/01/2016 12:55:15 AM	12 MANHATTAN	M
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	...	01/01/2016 01:26:57 AM	01 QUEENS	
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	...	01/01/2016 04:51:03 AM	07 BRONX	
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	...	01/01/2016 07:43:13 AM	10 BRONX	
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	...	01/01/2016 03:24:42 AM	04 QUEENS	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
364553	29609918	01/01/2015 12:04:44 AM	01/01/2015 10:22:31 AM	NYPD	New York City Police Department	Illegal Parking	Blocked Hydrant	Street/Sidewalk	11421.0	84-25 85 ROAD	...	01/01/2015 10:22:31 AM	09 QUEENS	
364554	29608392	01/01/2015 12:04:28 AM	01/01/2015 02:25:02 AM	NYPD	New York City Police Department	Noise - Vehicle	Car/Truck Horn	Street/Sidewalk	10468.0	2555 SEDGWICK AVENUE	...	01/01/2015 02:25:02 AM	07 BRONX	
364555	29607589	01/01/2015 12:01:30 AM	01/01/2015 12:20:33 AM	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Music/Party	Street/Sidewalk	10031.0	508 WEST 139 STREET	...	01/01/2015 12:20:33 AM	09 MANHATTAN	M

```
In [23]: ## Park Facility Name also has unspecified entries, check the column
         dataset['Park Facility Name'].value_counts()

Out[23]: Unspecified                    362176
         Alley Pond Park - Nature Center       1
         Name: Park Facility Name, dtype: int64

In [24]: dataset=dataset.drop(['Park Facility Name'],axis=1)
         dataset.head()
```

I have also noticed similar entries in Community Board, Borough and Park Borough columns. Hence, I decided to drop the columns Park Borough and Community Board from the dataset.

```
In [19]: ## Checking similar entries in Borough, Park Borough and community Board
         dataset['Borough'].value_counts()

Out[19]: BROOKLYN         118851
         QUEENS           100754
         MANHATTAN         77439
         BRONX             49164
         STATEN ISLAND     15334
         Unspecified         635
         Name: Borough, dtype: int64

In [20]: dataset['Park Borough'].value_counts()

Out[20]: BROOKLYN         118851
         QUEENS           100754
         MANHATTAN         77439
         BRONX             49164
         STATEN ISLAND     15334
         Unspecified         635
         Name: Park Borough, dtype: int64

In [21]: dataset['Community Board'].value_counts()

Out[21]: 12 MANHATTAN                14134
         01 BROOKLYN                 12802
         05 QUEENS                   11821
         01 QUEENS                   11637
         09 QUEENS                   10027
                                     ...
         26 BRONX                       11
         80 QUEENS                      10
         56 BROOKLYN                     9
         Unspecified QUEENS              2
         Unspecified STATEN ISLAND       2
         Name: Community Board, Length: 75, dtype: int64

In [22]: ## Since Community board, Borough and Park Borough has similar entries, let's remove Community Board and Park Borough columns
         dataset=dataset.drop(['Community Board'], axis=1)
         dataset=dataset.drop(['Park Borough'], axis=1)
         dataset.head()

Out[22]:
```

Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	Incident Zip	Incident Address	...	Due Date	Resolution Description	Resoluti Acti Updat Da
												The Police	

The columns of dates i.e. Created Date, Closed Date and Due date were in object dtype, due to which it was difficult to calculate the response time in secs. Hence, I had to convert the respective dates' columns in datetime dtype.

```
In [25]: dataset.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 362177 entries, 0 to 364557
Data columns (total 26 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   Unique Key                      362177 non-null  int64
 1   Created Date                    362177 non-null  object
 2   Closed Date                     362177 non-null  object
 3   Agency                          362177 non-null  object
 4   Agency Name                     362177 non-null  object
 5   Complaint Type                  362177 non-null  object
 6   Descriptor                      355681 non-null  object
 7   Location Type                   362047 non-null  object
 8   Incident Zip                    361502 non-null  float64
 9   Incident Address                310491 non-null  object
 10  Street Name                     310491 non-null  object
 11  Cross Street 1                  306846 non-null  object
 12  Cross Street 2                  306713 non-null  object
 13  Address Type                    361248 non-null  object
 14  City                            361503 non-null  object
 15  Facility Type                   362159 non-null  object
 16  Status                          362177 non-null  object
 17  Due Date                        362176 non-null  object
 18  Resolution Description          362177 non-null  object
 19  Resolution Action Updated Date  362138 non-null  object
 20  Borough                         362177 non-null  object
 21  X Coordinate (State Plane)      360470 non-null  float64
 22  Y Coordinate (State Plane)      360470 non-null  float64
 23  Latitude                        360470 non-null  float64
 24  Longitude                       360470 non-null  float64
 25  Location                        360470 non-null  object
dtypes: float64(5), int64(1), object(20)
memory usage: 74.6+ MB

In [26]: ## Created and Closed date are in object type. Hence, need to change it to datetime type

         dataset["Created Date"]=pd.to_datetime(dataset["Created Date"])
         dataset["Closed Date"]=pd.to_datetime(dataset["Closed Date"])
         dataset['Due Date']=pd.to_datetime(dataset['Due Date'])
         dataset.head()
```

I then calculated the response time of all service requests and converted the Elapsed time into secs using total_seconds() function.

```
In [28]: ## Calculate the time elapsed in closed and created date for Response and closure

dataset["Elapsed_Time"]=dataset['Closed Date'] - dataset['Created Date']
Elapsed_Time=[]
for x in dataset["Closed Date"]-dataset["Created Date"]:
    close=x.total_seconds()
    Elapsed_Time.append(close)

dataset["Elapsed_Time"]=Elapsed_Time
```

```
In [29]: ## Print the column of Elapsed_Time from the dataset to check if it is converted into secs
dataset.head()
```

Out[29]:

pe	Incident Zip	Incident Address	...	Due Date	Resolution Description	Resolution Action Updated Date	Borough	X Coordinate (State Plane)	Y Coordinate (State Plane)	Latitude	Longitude	Location	Elapsed_Time
alk	10034.0	71 VERMILYEA AVENUE	...	2016-01-01 07:59:45	The Police Department responded and upon arriv...	01/01/2016 12:55:15 AM	MANHATTAN	1005409.0	254678.0	40.865682	-73.923501	(40.86568153633767, -73.92350095571744)	3330.0
alk	11105.0	27-07 23 AVENUE	...	2016-01-01 07:59:44	The Police Department responded to the complai...	01/01/2016 01:26:57 AM	QUEENS	1007766.0	221986.0	40.775945	-73.915094	(40.775945312321085, -73.91509393898605)	5233.0
alk	10458.0	2897 VALENTINE AVENUE	...	2016-01-01 07:59:29	The Police Department responded and upon arriv...	01/01/2016 04:51:03 AM	BRONX	1015081.0	256380.0	40.870325	-73.888525	(40.870324522111424, -73.88852464418646)	17494.0
alk	10461.0	2940 BAISLEY AVENUE	...	2016-01-01 07:57:46	The Police Department responded to the complai...	01/01/2016 07:43:13 AM	BRONX	1031740.0	243899.0	40.835994	-73.828379	(40.83599404683083, -73.82837939584206)	27927.0
alk	11373.0	87-14 57 ROAD	...	2016-01-01 07:56:58	The Police Department responded and upon arriv...	01/01/2016 03:24:42 AM	QUEENS	1019123.0	206375.0	40.733060	-73.874170	(40.733059618956815, -73.87416975810375)	12464.0

I viewed the descriptive statistics of the response time and found that the average response time is 151113 secs i.e 251 minutes i.e 4 hours per service requests.

```
In [30]: ## View the descriptive statistics of newly created column i.e. Elapsed_Time
dataset_new=dataset
pd.options.display.float_format = "{:.2f}".format
dataset_new["Elapsed_Time"].describe()
```

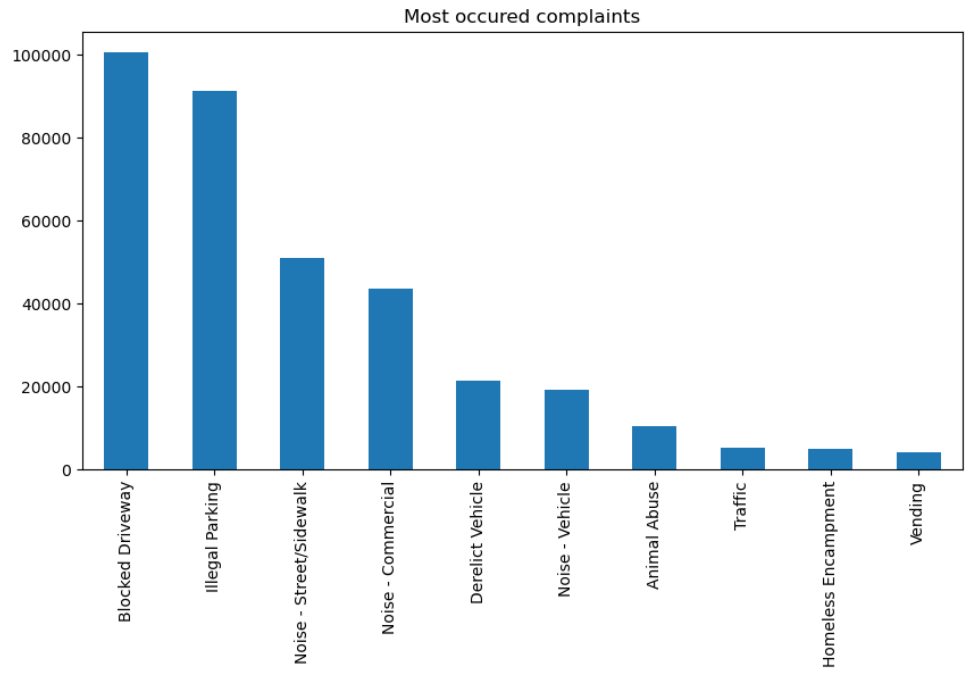
```
Out[30]: count    362177.00
         mean      15113.30
         std       21102.55
         min          61.00
         25%        4533.00
         50%        9616.00
         75%       18878.00
         max     2134342.00
         Name: Elapsed_Time, dtype: float64
```

To visualize the complaints based on cities, I found that the column City has 674 null values which I removed

```
In [32]: dataset_new['Complaint Type'].isna().sum(),dataset_new['City'].isna().sum()
Out[32]: (0, 674)

In [33]: ## City column has 674 null values
         ## Remove null values from City column
         dataset_new.dropna(subset=['City'], inplace=True)

In [34]: dataset_new['City'].isna().sum()
Out[34]: 0

In [35]: dataset_new['City'].unique()
Out[35]: 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)
```

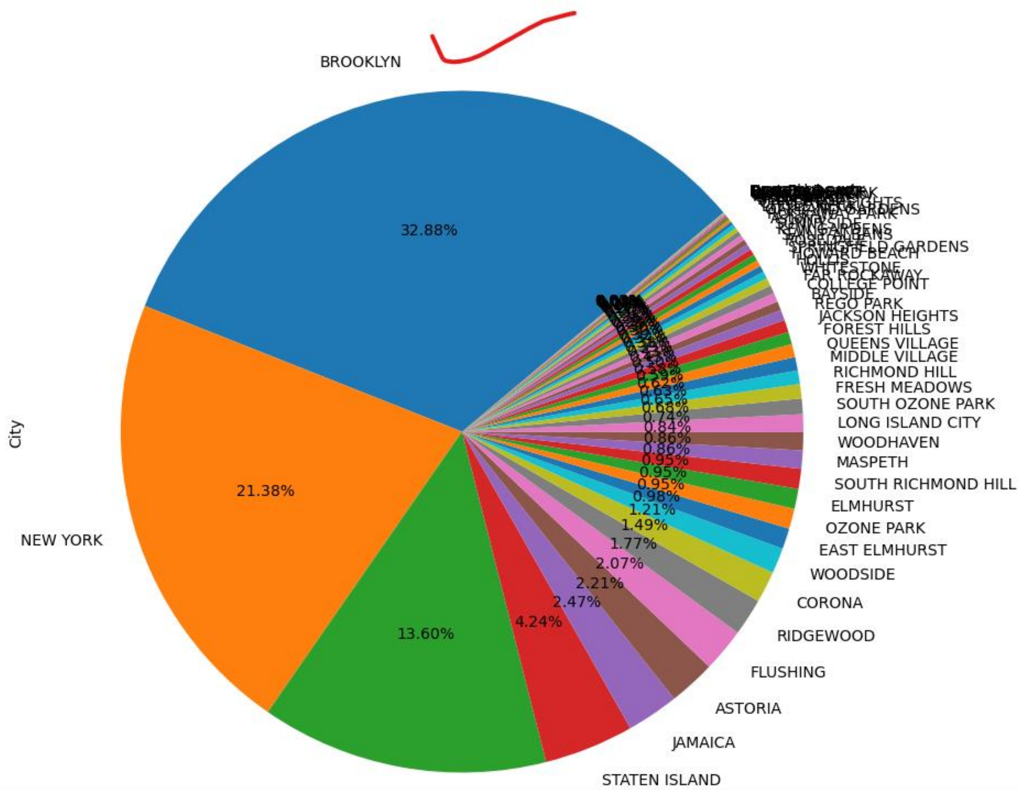
I visualized the most occurred complaints using bar chart and found the top 10 complaints are:

1) Blocked Driveway
2) Illegal Parking
3) Noise – Street/Sidewalk
4) Noise – Commercial
5) Derelict Vehicle
6) Noise- Vehicle
7) Animal Abuse
8) Traffic
9) Homeless Encampment
10) Vending



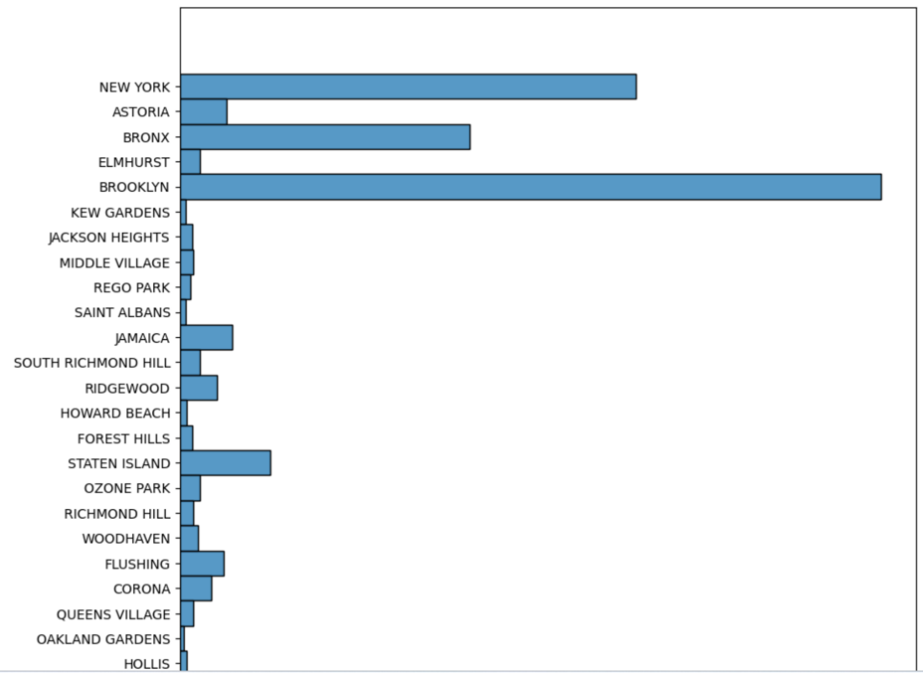
Also, I visualize the complaints based on cities using Pie chart and found Brooklyn has major complaints out of other cities.

```
In [37]: ## Lets see the complaints based on cities through Pie chart

         dataset_new['City'].value_counts().plot(kind='pie',autopct='%1.2f%%',startangle=40,figsize=(10,20))

Out[37]: <AxesSubplot:ylabel='City'>
```



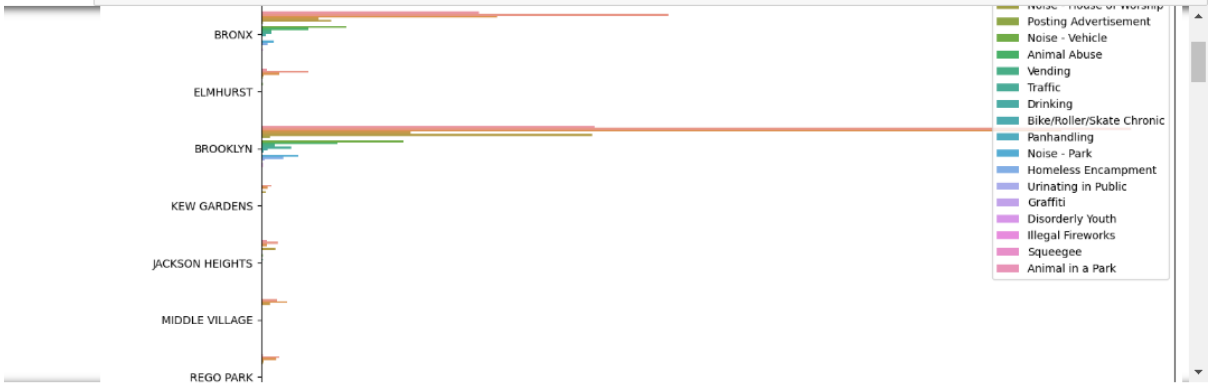
Frequency plot for city-wise complaints

```
In [37]: ## Frequency plot for City-wise complaints
         plt.figure(figsize=(10,20))
         sns.histplot(data=dataset_new,y='City')

Out[37]: <AxesSubplot:xlabel='Count', ylabel='City'>
```

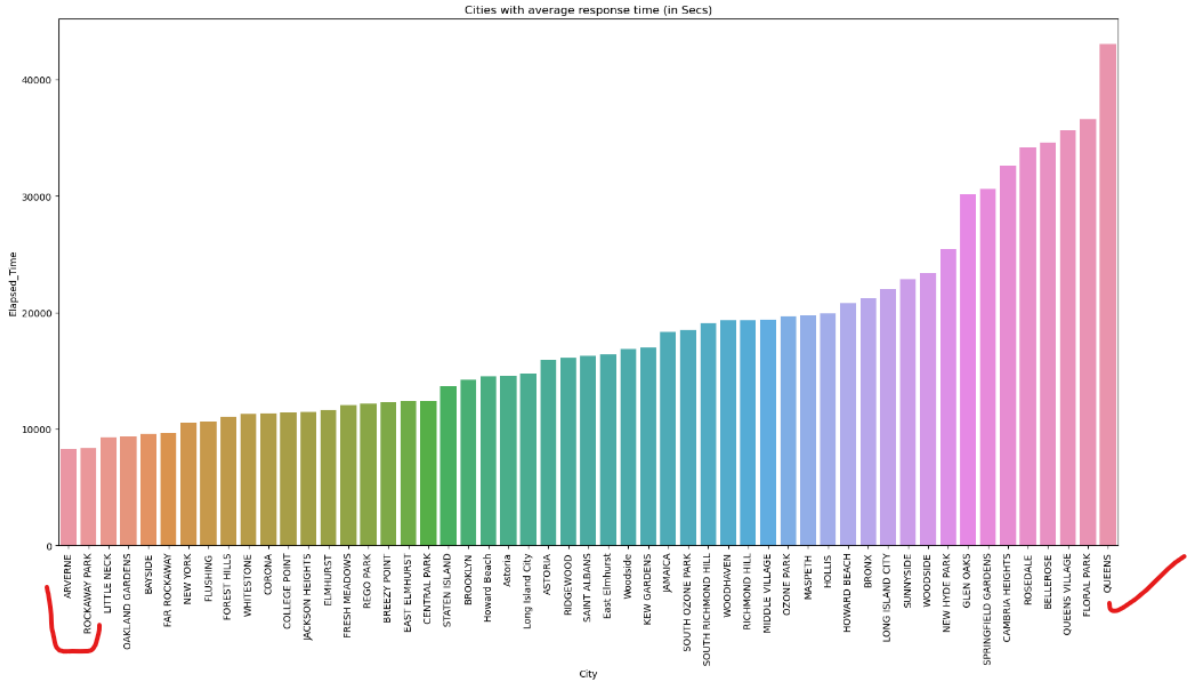


Now, I visualize the major types of complaints based on each city and found that New York has majorly complaints for Noise – Street/Sidewalk whereas Bronx and Brooklyn has major complaints on Illegal Parking and Blocked Driveway. Also, the complaints on Squeegee and Animal in a Park are the minor ones for each city.

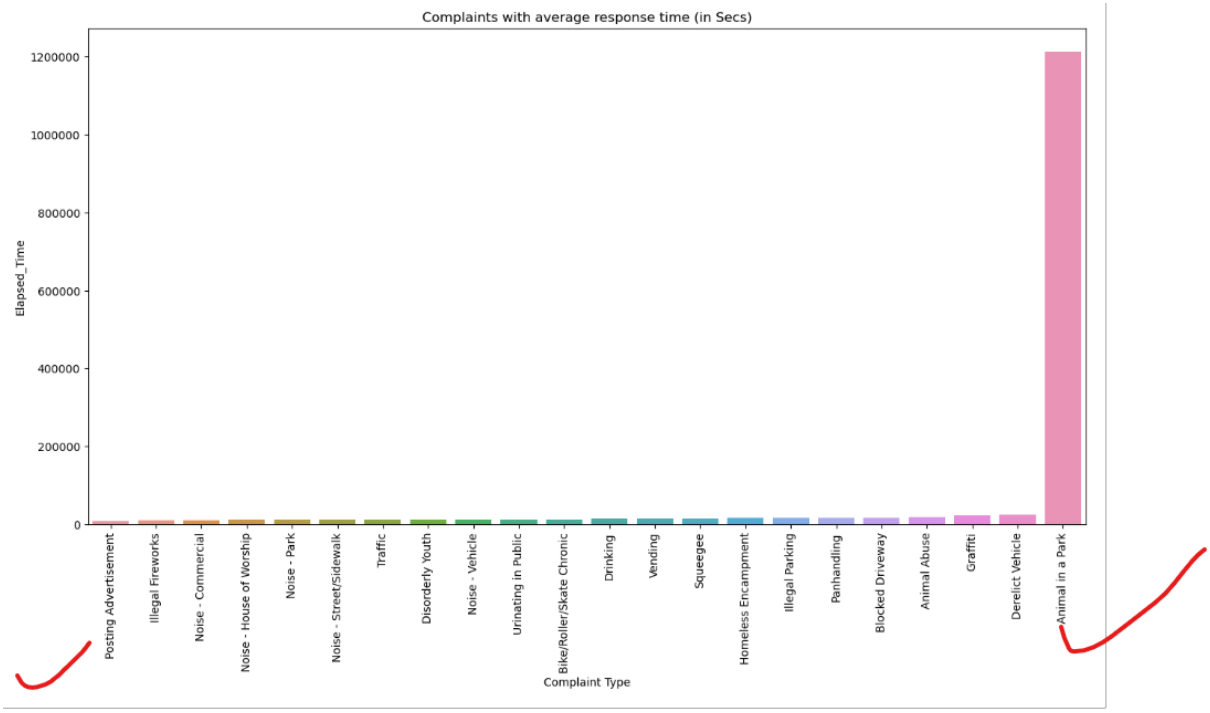
```
In [38]: ## Major types of complaints in each city

         plt.figure(figsize=(15,50))
         sns.countplot(y='City', hue='Complaint Type', data=dataset_new)
```



The cities with average response time was visualized and found that “Arverne” and “Rockaway Park” has fastest response time while Queens is the slowest.



Similarly, the complaints with average response time was visualized and found that “Posting Advertisement” has fastest response while Animal in a Park has the slowest response time.



With this chart we may see that the complaints have been almost responded in a similar pattern except for Animal in a Park.

Finally, a separate dataset was created for the complaints based on each city with its count.

```
In [74]: df_cce=dataset_new[['City','Complaint Type','Elapsed_Time']].copy()
         cce=pd.DataFrame({'Count':df_cce.groupby(['City','Complaint Type']).size()})
         cce.head(60)
```

Out[74]:

City	Complaint Type	Count
ARVERNE	Animal Abuse	46
	Blocked Driveway	50
	Derelict Vehicle	32
	Disorderly Youth	2
	Drinking	1
	Graffiti	1
	Homeless Encampment	4
	Illegal Parking	62
	Noise - Commercial	2
	Noise - House of Worship	14
	Noise - Park	2
	Noise - Street/Sidewalk	29
	Noise - Vehicle	10
	Panhandling	1
	Traffic	1
	Urinating in Public	1
	Vending	1
ASTORIA	Animal Abuse	170
	Bike/Roller/Skate Chronic	16
	Blocked Driveway	3436
	Derelict Vehicle	426
	Disorderly Youth	5
	Drinking	43
	Graffiti	4
	Homeless Encampment	22

Conclusion:

The analysis revealed that Brooklyn has major complaints out of all the cities. While Blocked Driveway and Illegal parking are the major complaints from each Cities. Simultaneously, the response time of each complaint on an average has been calculated to 4.19 hours.

The fastest response time is from Arverne and Rockaway while slowest is from Queens. Also, the complaints of “Posting advertisement” was responded faster while “Animal in a Park” was responded slowest.