Customer Service Requests Analysis

```
# Imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
from scipy.stats import f_oneway
from scipy.stats import chi2_contingency
import warnings
warnings.filterwarnings('ignore')

# Import dataset
customer_service_df =
pd.read_csv('311_Service_Requests_from_2010_to_Present.csv',
dtype='str')

# Get first 5 records
customer_service_df.head(5)
```

```
  Unique Key            Created Date    Closed Date Agency  \
0   32310363  12/31/2015 11:59:45 PM  01-01-16 0:55   NYPD
1   32309934  12/31/2015 11:59:44 PM  01-01-16 1:26   NYPD
2   32309159  12/31/2015 11:59:29 PM  01-01-16 4:51   NYPD
3   32305098  12/31/2015 11:57:46 PM  01-01-16 7:43   NYPD
4   32306529  12/31/2015 11:56:58 PM  01-01-16 3:24   NYPD

                      Agency Name           Complaint Type  \
0  New York City Police Department  Noise - Street/Sidewalk
1  New York City Police Department         Blocked Driveway
2  New York City Police Department         Blocked Driveway
3  New York City Police Department          Illegal Parking
4  New York City Police Department          Illegal Parking

                    Descriptor    Location Type Incident Zip  \
0              Loud Music/Party  Street/Sidewalk        10034
1                    No Access  Street/Sidewalk        11105
2                    No Access  Street/Sidewalk        10458
3  Commercial Overnight Parking  Street/Sidewalk        10461
4              Blocked Sidewalk  Street/Sidewalk        11373

        Incident Address  ... Bridge Highway Name Bridge Highway
Direction  \
0    71 VERMILYEA AVENUE  ...                 NaN
NaN
1        27-07 23 AVENUE  ...                 NaN
NaN
2  2897 VALENTINE AVENUE  ...                 NaN
```

```
NaN
3    2940 BAISLEY AVENUE  ...                 NaN
NaN
4          87-14 57 ROAD  ...                 NaN
NaN

  Road Ramp Bridge Highway Segment Garage Lot Name Ferry Direction  \
0       NaN                     NaN              NaN            NaN
1       NaN                     NaN              NaN            NaN
2       NaN                     NaN              NaN            NaN
3       NaN                     NaN              NaN            NaN
4       NaN                     NaN              NaN            NaN

  Ferry Terminal Name     Latitude     Longitude  \
0                 NaN  40.86568154  -73.92350096
1                 NaN  40.77594531  -73.91509394
2                 NaN  40.87032452  -73.88852464
3                 NaN  40.83599405   -73.8283794
4                 NaN  40.73305962  -73.87416976

                                Location
0   (40.86568153633767, -73.92350095571744)
1  (40.775945312321085, -73.91509393898605)
2  (40.870324522111424, -73.88852464418646)
3   (40.83599404683083, -73.82837939584206)
4  (40.733059618956815, -73.87416975810375)

[5 rows x 53 columns]
```

1.1. Identify the shape of the dataset

```
customer_service_df.shape
```

```
(300698, 53)
```

```
# dataset colums
customer_service_df.columns
```

```
Index(['Unique Key', 'Created Date', 'Closed Date', 'Agency', 'Agency
Name',
       'Complaint Type', 'Descriptor', 'Location Type', 'Incident
Zip',
       'Incident Address', 'Street Name', 'Cross Street 1', 'Cross
Street 2',
       'Intersection Street 1', 'Intersection Street 2', 'Address
Type',
       'City', 'Landmark', 'Facility Type', 'Status', 'Due Date',
       'Resolution Description', 'Resolution Action Updated Date',
       'Community Board', 'Borough', 'X Coordinate (State Plane)',
       'Y Coordinate (State Plane)', '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', 'Latitude', 'Longitude', 'Location'],
      dtype='object')
```

1.2. Identify variables with null values

```
na_cols = customer_service_df.isna().sum()
na_cols[na_cols > 0]
```

```
Closed Date                          2164
Descriptor                           5914
Location Type                         131
Incident Zip                         2615
Incident Address                    44410
Street Name                         44410
Cross Street 1                      49279
Cross Street 2                      49779
Intersection Street 1              256840
Intersection Street 2              257336
Address Type                         2815
City                                 2614
Landmark                           300349
Facility Type                        2171
Due Date                                3
Resolution Action Updated Date       2187
X Coordinate (State Plane)           3540
Y Coordinate (State Plane)           3540
School Region                           1
School Code                             1
School Zip                              1
School or Citywide Complaint       300698
Vehicle Type                       300698
Taxi Company Borough               300698
Taxi Pick Up Location              300698
Bridge Highway Name                300455
Bridge Highway Direction           300455
Road Ramp                          300485
Bridge Highway Segment             300485
Garage Lot Name                    300698
Ferry Direction                    300697
Ferry Terminal Name                300696
Latitude                             3540
Longitude                            3540
```

```
Location                            3540
dtype: int64
```

2.1. Utilize missing value treatment

2.2. Analyze the date column and remove the entries if it has an incorrect timeline

```
# Convert datatype to datetime for columns
customer_service_df['Created Date'] =
pd.to_datetime(customer_service_df['Created Date'])
customer_service_df['Closed Date'] =
pd.to_datetime(customer_service_df['Closed Date'])
customer_service_df['Due Date'] =
pd.to_datetime(customer_service_df['Due Date'])
customer_service_df['Resolution Action Updated Date'] =
pd.to_datetime(customer_service_df['Resolution Action Updated Date'])

# comvert longitude and latitude to numeric
customer_service_df['Longitude'] =
pd.to_numeric(customer_service_df['Longitude'])
customer_service_df['Latitude'] =
pd.to_numeric(customer_service_df['Latitude'])

# Drop columns with all NaN values
customer_service_df.dropna(axis=1, how='all', inplace=True)

# Updated columns
customer_service_df.columns
```

```
Index(['Unique Key', 'Created Date', 'Closed Date', 'Agency', 'Agency
Name',
       'Complaint Type', 'Descriptor', 'Location Type', 'Incident
Zip',
       'Incident Address', 'Street Name', 'Cross Street 1', 'Cross
Street 2',
       'Intersection Street 1', 'Intersection Street 2', 'Address
Type',
       'City', 'Landmark', 'Facility Type', 'Status', 'Due Date',
       'Resolution Description', 'Resolution Action Updated Date',
       'Community Board', 'Borough', 'X Coordinate (State Plane)',
       'Y Coordinate (State Plane)', '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', 'Bridge Highway Name',
       'Bridge Highway Direction', 'Road Ramp', 'Bridge Highway
Segment',
       'Ferry Direction', 'Ferry Terminal Name', 'Latitude',
'Longitude',
       'Location'],
      dtype='object')
```

```
# Add new column Request_Closing_Time in hours
customer_service_df['Request_Closing_Time'] =
customer_service_df['Closed Date'] - customer_service_df['Created
Date']
customer_service_df['Request_Closing_Time'] =
customer_service_df['Request_Closing_Time'].apply(lambda x:
x.total_seconds()/3600)

customer_service_df['Request_Closing_Time'][:5]
```

```
0    0.920833
1    1.437778
2    4.858611
3    7.753889
4    3.450556
Name: Request_Closing_Time, dtype: float64
```

```
customer_service_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300698 entries, 0 to 300697
Data columns (total 49 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   Unique Key                      300698 non-null  object
 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  object
 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  datetime64[ns]
 21  Resolution Description          300698 non-null  object
 22  Resolution Action Updated Date  298511 non-null  datetime64[ns]
 23  Community Board                 300698 non-null  object
 24  Borough                         300698 non-null  object
 25  X Coordinate (State Plane)      297158 non-null  object
 26  Y Coordinate (State Plane)      297158 non-null  object
```

```
 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  Bridge Highway Name             243 non-null     object
 40  Bridge Highway Direction        243 non-null     object
 41  Road Ramp                       213 non-null     object
 42  Bridge Highway Segment          213 non-null     object
 43  Ferry Direction                 1 non-null       object
 44  Ferry Terminal Name             2 non-null       object
 45  Latitude                        297158 non-null  float64
 46  Longitude                       297158 non-null  float64
 47  Location                        297158 non-null  object
 48  Request_Closing_Time            298534 non-null  float64
dtypes: datetime64[ns](4), float64(3), object(42)
memory usage: 112.4+ MB
```

We have lots of columns, but we are mostly focusing on created date, closing date, due date, request closing time, location, complaint type, city, angency, longitude, latitude so that we can get some insight from it.

```
cs_df = customer_service_df[
    ['Created Date', 'Closed Date', 'Due Date',
'Request_Closing_Time', 'Complaint Type',
     'Location Type', 'City', 'Agency', 'Latitude', 'Longitude']]
cs_df.head()
```

```
         Created Date         Closed Date            Due Date  \
0 2015-12-31 23:59:45 2016-01-01 00:55:00 2016-01-01 07:59:00
1 2015-12-31 23:59:44 2016-01-01 01:26:00 2016-01-01 07:59:00
2 2015-12-31 23:59:29 2016-01-01 04:51:00 2016-01-01 07:59:00
3 2015-12-31 23:57:46 2016-01-01 07:43:00 2016-01-01 07:57:00
4 2015-12-31 23:56:58 2016-01-01 03:24:00 2016-01-01 07:56:00

   Request_Closing_Time           Complaint Type    Location Type
City  \
0              0.920833  Noise - Street/Sidewalk  Street/Sidewalk  NEW
YORK
1              1.437778         Blocked Driveway  Street/Sidewalk
ASTORIA
2              4.858611         Blocked Driveway  Street/Sidewalk
BRONX
```

```
3              7.753889          Illegal Parking  Street/Sidewalk
BRONX
4              3.450556          Illegal Parking  Street/Sidewalk
ELMHURST

  Agency   Latitude  Longitude
0   NYPD  40.865682 -73.923501
1   NYPD  40.775945 -73.915094
2   NYPD  40.870325 -73.888525
3   NYPD  40.835994 -73.828379
4   NYPD  40.733060 -73.874170
```

```
# check for NaN
cs_df.isna().sum()
```

```
Created Date              0
Closed Date            2164
Due Date                  3
Request_Closing_Time   2164
Complaint Type            0
Location Type           131
City                   2614
Agency                    0
Latitude               3540
Longitude              3540
dtype: int64
```

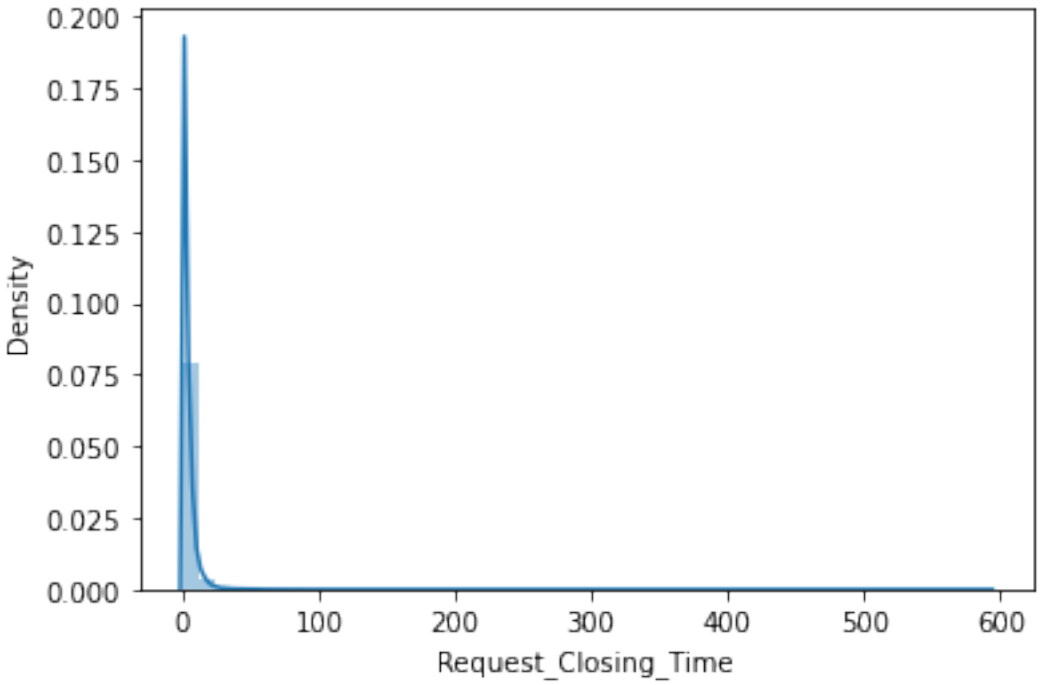
We have 300698 records in total. So we can easily drop the records with NaN values.

```
cs_df.dropna(inplace=True)
cs_df.shape
```

```
(296939, 10)
```

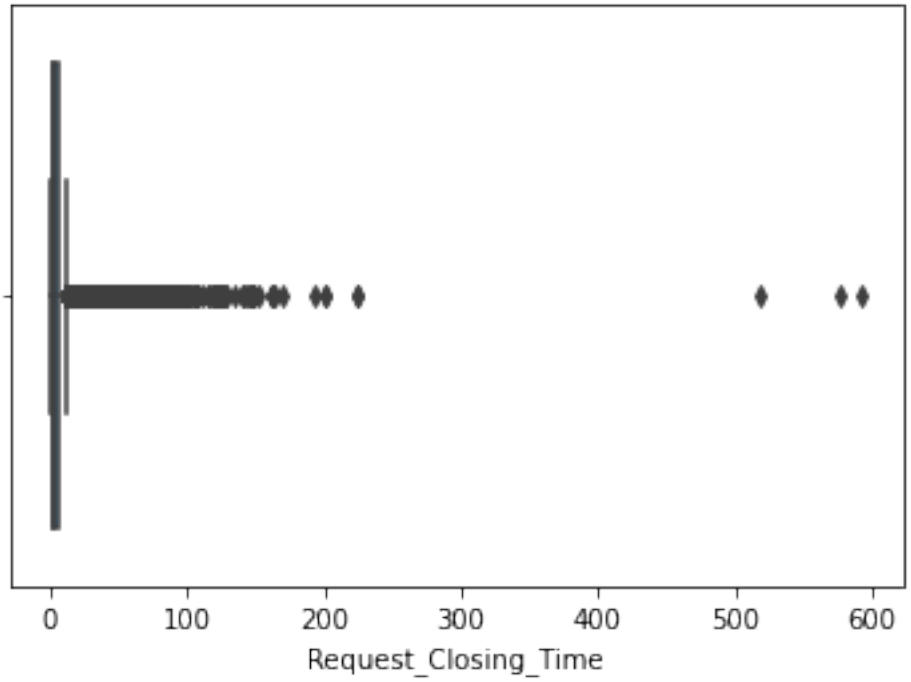
We can now start analysing with different plots and get rid of outliers.

```
# Closing time distribution
sns.distplot(cs_df['Request_Closing_Time'])
plt.show()
```



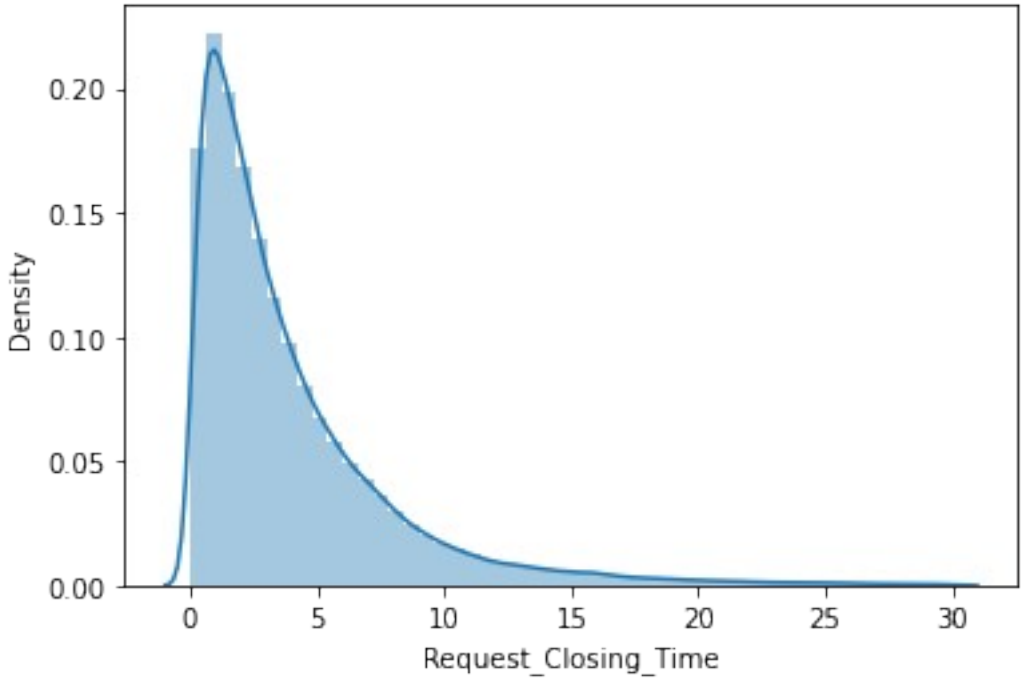
Most of the requests get completed within 0-20 hours, but there is few outliers. We can check in box-plot.

```
sns.boxplot(cs_df['Request_Closing_Time'])
plt.show()
```




```
# Check for outliers
np.percentile(cs_df['Request_Closing_Time'], 99)
```

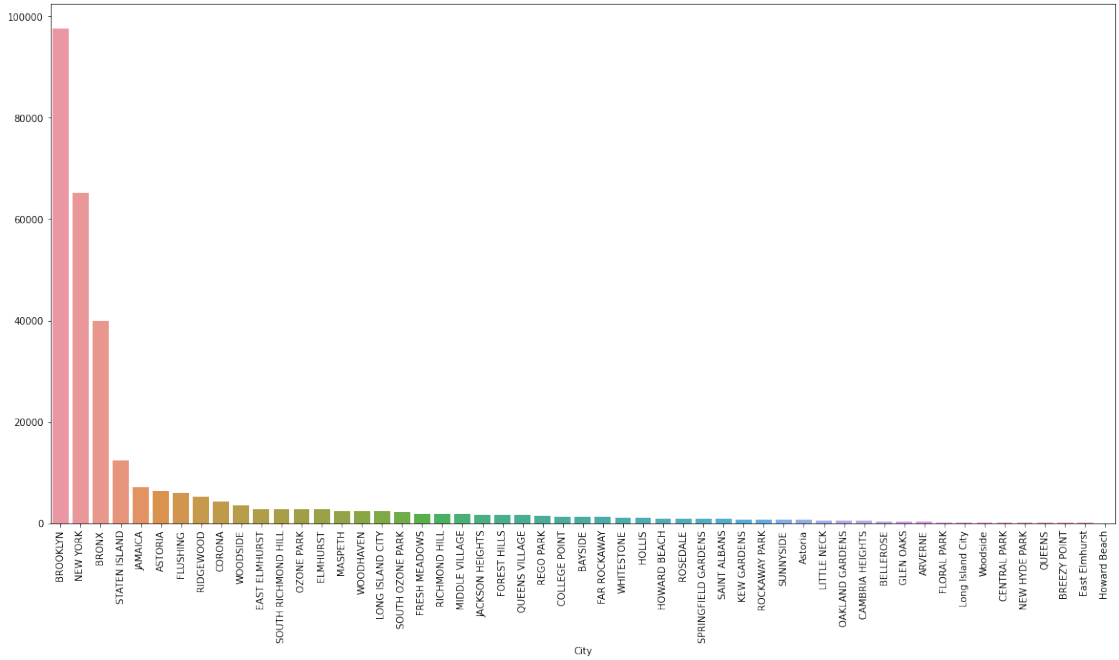
```
26.136777777777752

# % of extreme values
cs_df[cs_df['Request_Closing_Time'] <= 30].shape[0] / cs_df.shape[0]

0.9925944385883969

#We can remove the outliers.
cs_df = cs_df[cs_df['Request_Closing_Time'] < 30]

# check distribution again
sns.distplot(cs_df['Request_Closing_Time'])
plt.show()
```



Most of the cases get resolved within 0-15 hours and few of them take more time.

2.3. Draw a frequency plot for city-wise complaints

```
city_complaints =
cs_df.groupby('City').size().sort_values(ascending=False)
plt.figure(figsize=(20,10))
sns.barplot(city_complaints.index, city_complaints)
plt.xticks(rotation=90)
plt.show()
```



Brooklyn, New York and Bronk cities have much higer complaints.

2.4. Draw scatter and hexbin plots for complaint concentration across Brooklyn

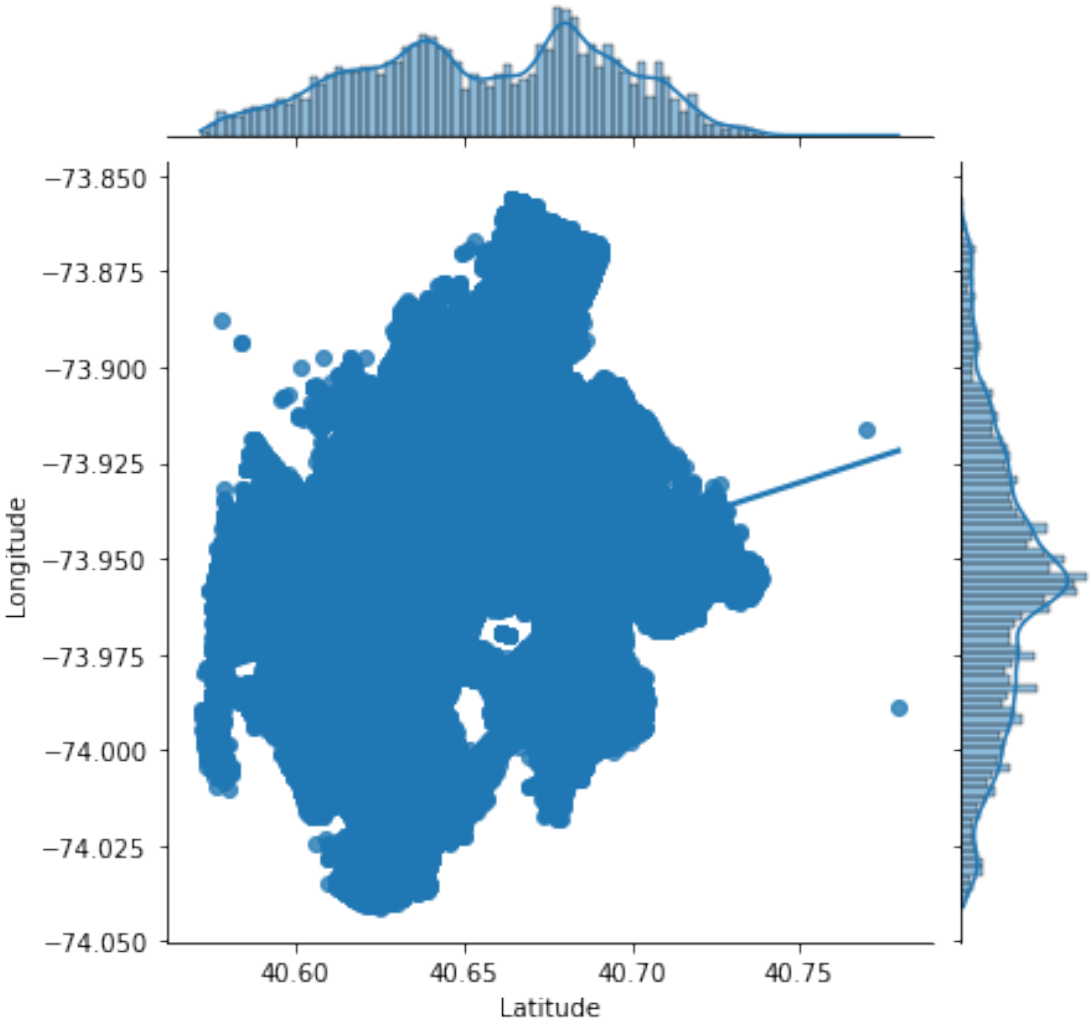
```
cs_df_brooklyn = cs_df[cs_df['City'] == 'BROOKLYN']
```

```
cs_df_brooklyn.shape
```

```
(97666, 9)
```

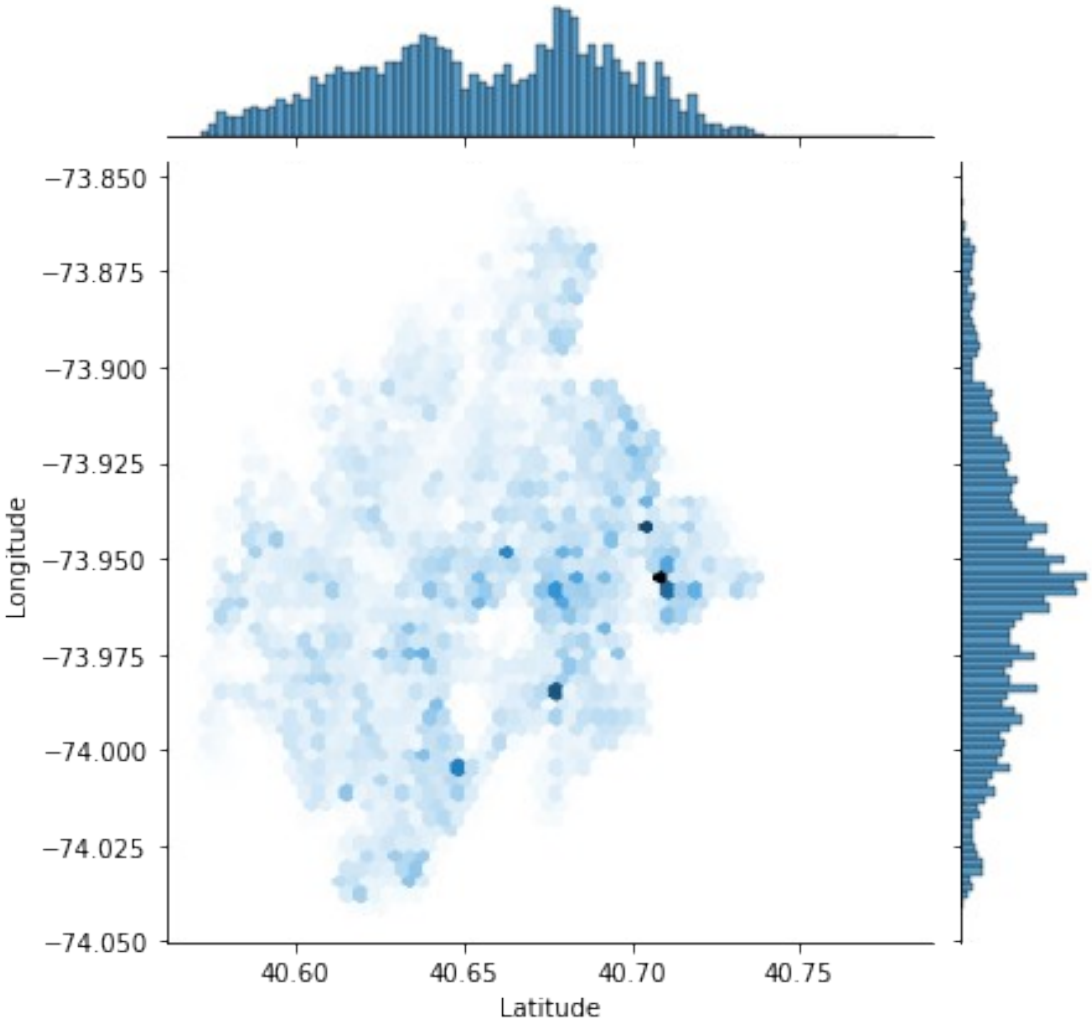
```
sns.jointplot('Latitude', 'Longitude', data=cs_df_brooklyn,
kind='reg')
```

```
<seaborn.axisgrid.JointGrid at 0x7f3b45c5f750>
```



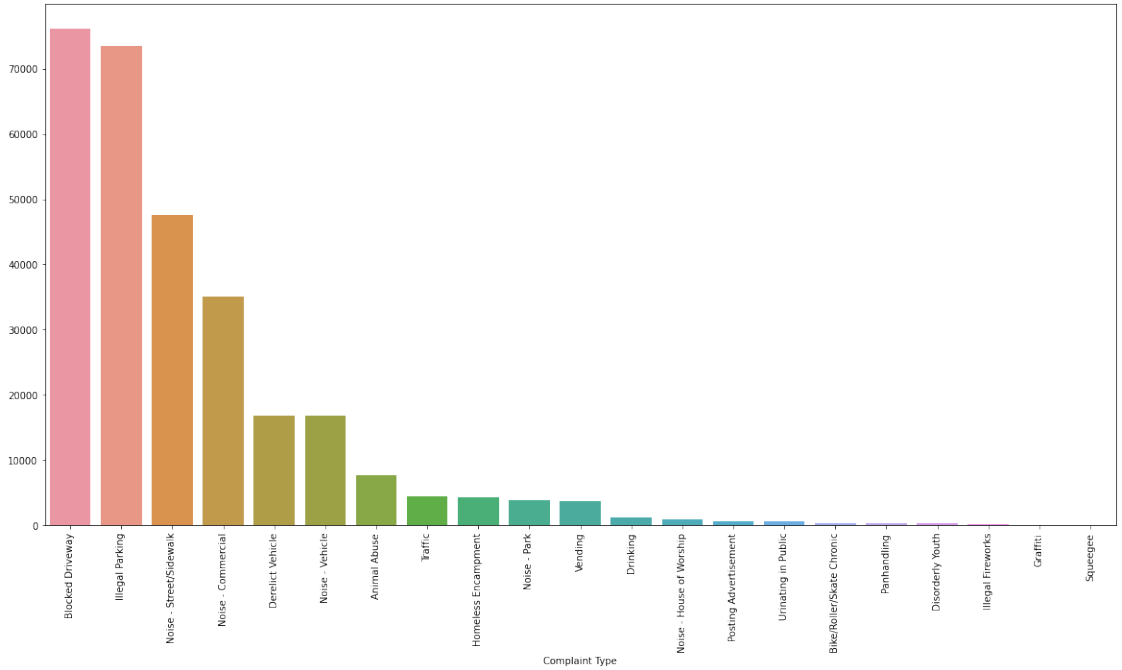

```
sns.jointplot('Latitude', 'Longitude', data=cs_df_brooklyn,
kind='hex')
```

```
<seaborn.axisgrid.JointGrid at 0x7f3b441f4c50>
```



3.1. Plot a bar graph of count vs. complaint types

```
complaints_counts = cs_df.groupby('Complaint
Type').size().sort_values(ascending=False)
plt.figure(figsize=(20,10))
sns.barplot(complaints_counts.index, complaints_counts)
plt.xticks(rotation=90)
plt.show()
```



In the city, most of the cases are for blocked driveway, illegal parking and noise. As NYC is a large busy city. We can expect such insights.

3.2. Find the top 10 types of complaints

```
complaints_counts[:10]
```

```
Complaint Type
Blocked Driveway           76155
Illegal Parking            73528
Noise - Street/Sidewalk    47543
Noise - Commercial         35027
Derelict Vehicle           16897
Noise - Vehicle            16792
Animal Abuse                7663
Traffic                     4440
Homeless Encampment         4335
Noise - Park                3917
dtype: int64
```

3.3. Display the types of complaints in each city in a separate dataset

```
city_complaints = cs_df[['City', 'Complaint Type']].groupby('City')
['Complaint Type'].value_counts()

city_complaints
```

```
City      Complaint Type
ARVERNE   Illegal Parking             58
          Animal Abuse                38
          Blocked Driveway            35
          Noise - Street/Sidewalk     29
```

```
          Derelict Vehicle            27
                                     ...
Woodside  Illegal Parking            100
          Blocked Driveway            11
          Noise - Street/Sidewalk      5
          Derelict Vehicle             2
          Noise - Commercial           2
Name: Complaint Type, Length: 760, dtype: int64
```

```
# Agency
cs_df['Agency'].nunique()
```

```
1
```

We have only NYPD operating on all these areas. As it has a single value. We can get rid of it.

```
cs_df.drop('Agency', axis=1, inplace=True)
```

4. Visualize the major types of complaints in each city

```
complaintsTypes = cs_df['Complaint Type'].values.tolist()
len(complaintsTypes)
```

```
294739
```

```
# Unique complaints
len(set(complaintsTypes))
```

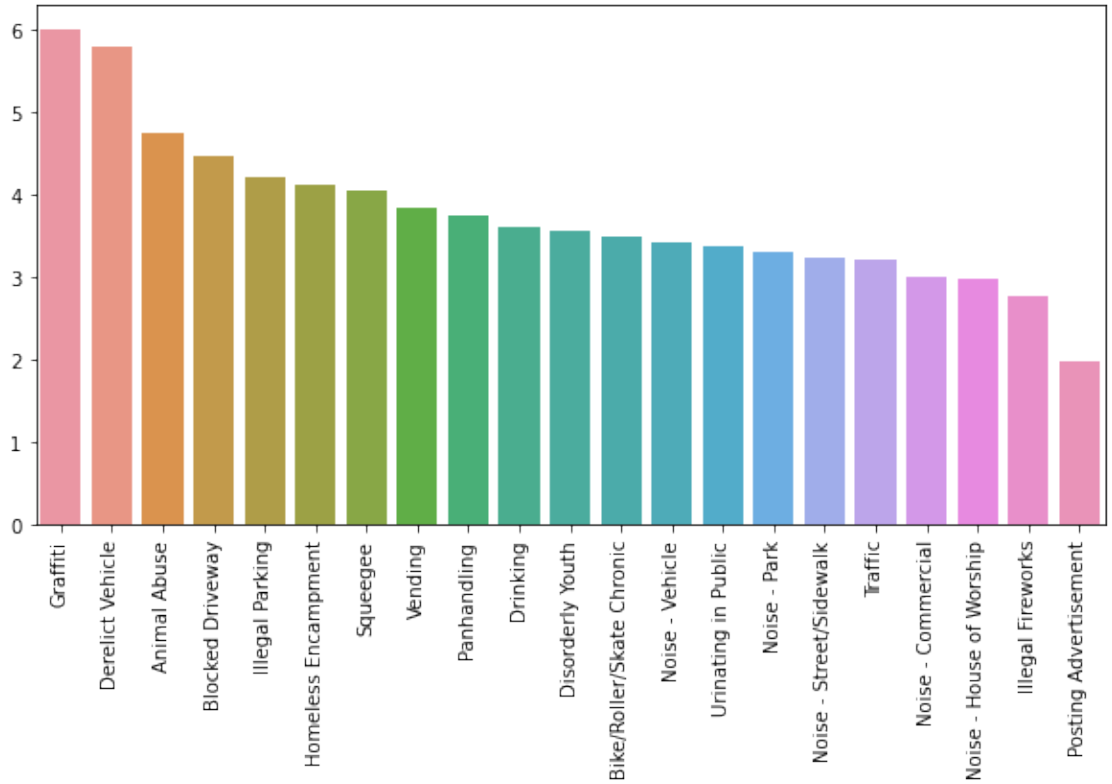
```
21
```

```
complaints = ' '.join(complaintsTypes) + ' '

wordcloud = WordCloud(width = 800, height = 800,
                background_color ='white',
                min_font_size = 10).generate(complaints)

plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```

5. Check if the average response time across various types of complaints

Average response time can refer to the request closing time.

```
closing_time_per_type = cs_df.groupby('Complaint Type').mean()
['Request_Closing_Time'].sort_values(ascending=False)
plt.figure(figsize=(10,5))
sns.barplot(closing_time_per_type.index.values,
closing_time_per_type.values)
plt.xticks(rotation=90)
plt.show()
```



There is not much difference among the closing times for all these complaint types. But we ca say, Derelict Vehicle and Graffiti take more than to resolved than others.

6. Identify significant variables by performing a statistical analysis using p-values and chi-square values (Optional)

We can pick few columns against the complaint type to see whether thet are significant or not with chi-sqaure test.

```
features = ['Request_Closing_Time', 'Location Type', 'City']
target = 'Complaint Type'

for feature in features:
    cs_cat_df_cross_tab = pd.crosstab(index=cs_df[target],
columns=cs_df[feature])
    print('P-value in chi-square test for {} : {}'.format(feature,
chi2_contingency(cs_cat_df_cross_tab)[1]))
```

```
P-value in chi-square test for Request_Closing_Time : 1.0
P-value in chi-square test for Location Type : 0.0
P-value in chi-square test for City : 0.0
```

So we can say, Location Type and City are highly correlated with Complaint Type, where are Request_Closing_Time is not as it has p-value more than 0.5.