Customer Service Requests Analysis

NYC 311's mission is to provide the public with quick and easy access to all New York City government services and information while offering the best customer service. Each day, NYC311 receives thousands of requests related to several hundred types of non-emergency services, including noise complaints, plumbing issues, and illegally parked cars. These requests are received by NYC311 and forwarded to the relevant agencies such as the police, buildings, or transportation. The agency responds to the request, addresses it, and then closes it.

In [1]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
import chart_studio.plotly as py
import plotly.express as px
import cufflinks as cf
from plotly.offline import download_plotlyjs,init_notebook_mode,plot,iplot
import  plotly.graph_objects as go
```

In [2]:
```
init_notebook_mode(connected=True)
cf.go_offline()
```

In [3]:
```
df = pd.read_csv(r"C:\Users\DELL\Desktop\Python\311_Service.csv")
```

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

Out[4]:

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	Incident Zip	Incident Address	...	Bridge Highway Name	I
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	
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	
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	
3	32305098	12/31/2015 11:57:46 PM	01-01-16 7:43	NYPD	New York City Police Department	Illegal Parking	Commercial Overnight Parking	Street/Sidewalk	10461.0	2940 BAISLEY AVENUE	...	NaN	
4	32306529	12/31/2015 11:56:58 PM	01-01-16 3:24	NYPD	New York City Police Department	Illegal Parking	Blocked Sidewalk	Street/Sidewalk	11373.0	87-14 57 ROAD	...	NaN	

5 rows × 53 columns

In [5]:
```
df.shape
```

Out[5]:
```
(300698, 53)
```

In [6]:
```
df.columns
```

Out[6]:
```
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')
```

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300698 entries, 0 to 300697
Data columns (total 53 columns):
Unique Key                          300698 non-null int64
Created Date                        300698 non-null object
Closed Date                         298534 non-null object
Agency                              300698 non-null object
Agency Name                         300698 non-null object
Complaint Type                      300698 non-null object
Descriptor                          294784 non-null object
Location Type                       300567 non-null object
Incident Zip                        298083 non-null float64
Incident Address                    256288 non-null object
Street Name                         256288 non-null object
Cross Street 1                      251419 non-null object
Cross Street 2                      250919 non-null object
Intersection Street 1               43858 non-null object
Intersection Street 2               43362 non-null object
Address Type                        297883 non-null object
City                                298084 non-null object
Landmark                            349 non-null object
Facility Type                       298527 non-null object
Status                              300698 non-null object
Due Date                            300695 non-null object
Resolution Description              300698 non-null object
Resolution Action Updated Date      298511 non-null object
Community Board                     300698 non-null object
Borough                             300698 non-null object
X Coordinate (State Plane)          297158 non-null float64
Y Coordinate (State Plane)          297158 non-null float64
Park Facility Name                  300698 non-null object
Park Borough                        300698 non-null object
School Name                         300698 non-null object
School Number                       300698 non-null object
School Region                       300697 non-null object
School Code                         300697 non-null object
School Phone Number                 300698 non-null object
School Address                      300698 non-null object
School City                         300698 non-null object
School State                        300698 non-null object
School Zip                          300697 non-null object
School Not Found                    300698 non-null object
School or Citywide Complaint        0 non-null float64
```

```
Vehicle Type                    0 non-null float64
Taxi Company Borough            0 non-null float64
Taxi Pick Up Location           0 non-null float64
Bridge Highway Name             243 non-null object
Bridge Highway Direction        243 non-null object
Road Ramp                       213 non-null object
Bridge Highway Segment          213 non-null object
Garage Lot Name                 0 non-null float64
Ferry Direction                 1 non-null object
Ferry Terminal Name             2 non-null object
Latitude                        297158 non-null float64
Longitude                       297158 non-null float64
Location                        297158 non-null object
dtypes: float64(10), int64(1), object(42)
memory usage: 73.4+ MB
```

Lets check for missing values

In [8]:
```
missing_df = df.isnull().sum(axis=0).reset_index()
```

In [9]:
```
missing_df.columns = ['column','count']
```

In [10]:
```
missing_df = missing_df[missing_df['count']>0]
```

In [11]:
```
missing_df =missing_df.sort_values('count',ascending=False)
```

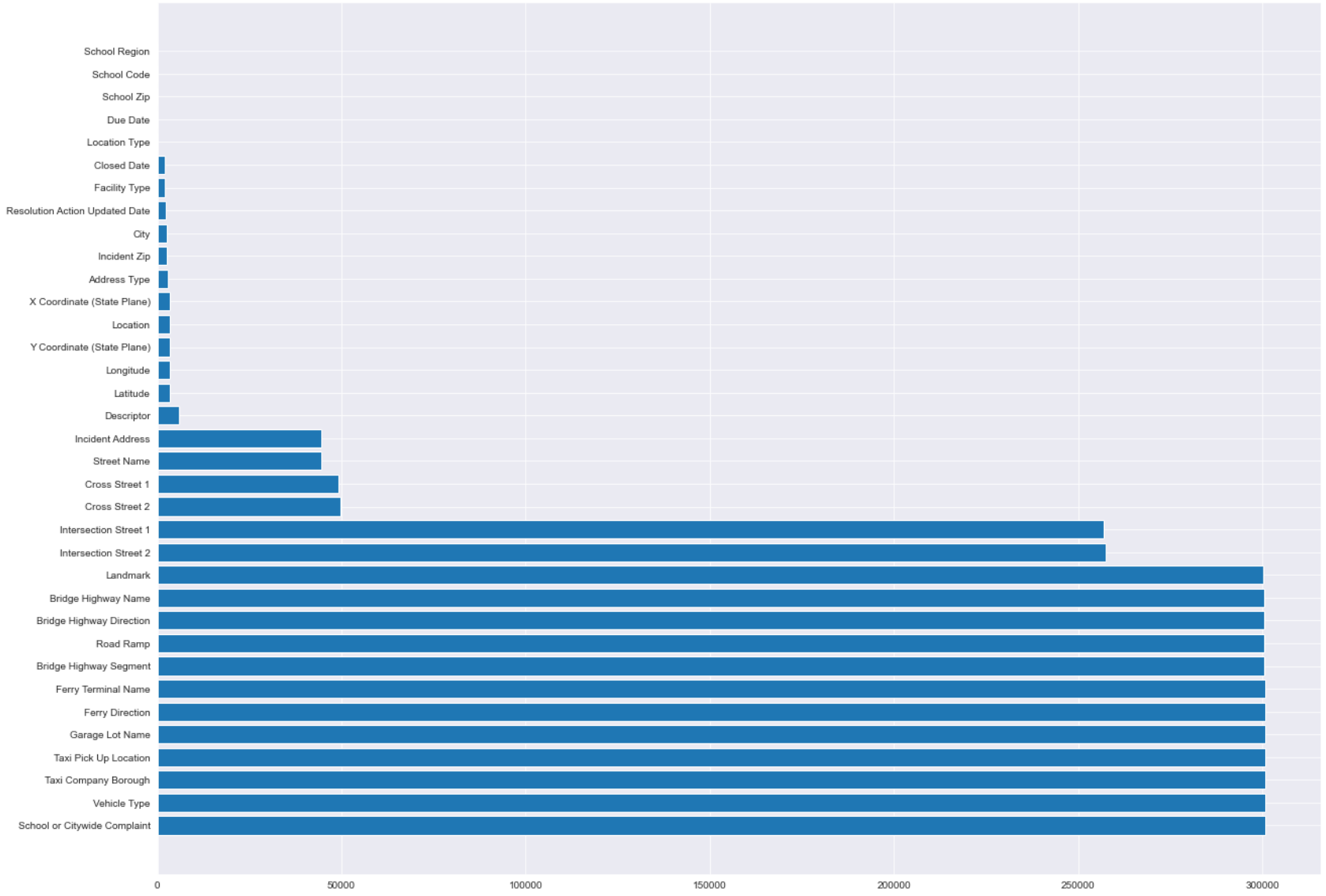
In [12]:
```
missing_df
```

Out[12]:

	column	count
39	School or Citywide Complaint	300698
40	Vehicle Type	300698
41	Taxi Company Borough	300698
42	Taxi Pick Up Location	300698
47	Garage Lot Name	300698
48	Ferry Direction	300697
49	Ferry Terminal Name	300696
46	Bridge Highway Segment	300485
45	Road Ramp	300485
44	Bridge Highway Direction	300455
43	Bridge Highway Name	300455
17	Landmark	300349
14	Intersection Street 2	257336
13	Intersection Street 1	256840
12	Cross Street 2	49779
11	Cross Street 1	49279
10	Street Name	44410
9	Incident Address	44410
6	Descriptor	5914
50	Latitude	3540
51	Longitude	3540
26	Y Coordinate (State Plane)	3540
52	Location	3540
25	X Coordinate (State Plane)	3540
15	Address Type	2815
8	Incident Zip	2615

	column	count
16	City	2614
22	Resolution Action Updated Date	2187
18	Facility Type	2171
2	Closed Date	2164
7	Location Type	131
20	Due Date	3
37	School Zip	1
32	School Code	1
31	School Region	1

In [13]:
```
sns.set_style("darkgrid")
fig = plt.figure(figsize=(16,12))
ax = fig.add_axes([0,0,1,1])
ax.barh(missing_df['column'],missing_df['count'])
plt.show()
```



In [14]:
```
#From the above graph looks like after the column intersection street 1 have high null values.. So we will dr
op those colums
```

In [15]:
```
na_columns = missing_df.head(18)
```

In [16]:
```
na_columns
```

Out[16]:

	column	count
39	School or Citywide Complaint	300698
40	Vehicle Type	300698
41	Taxi Company Borough	300698
42	Taxi Pick Up Location	300698
47	Garage Lot Name	300698
48	Ferry Direction	300697
49	Ferry Terminal Name	300696
46	Bridge Highway Segment	300485
45	Road Ramp	300485
44	Bridge Highway Direction	300455
43	Bridge Highway Name	300455
17	Landmark	300349
14	Intersection Street 2	257336
13	Intersection Street 1	256840
12	Cross Street 2	49779
11	Cross Street 1	49279
10	Street Name	44410
9	Incident Address	44410

In [17]:
```
# Lets drop these columns
df =df.drop(na_columns['column'],axis=1)
```

In [18]:
```
df.isna().sum()
```

Out[18]:
```
Unique Key                        0
Created Date                      0
Closed Date                    2164
Agency                            0
Agency Name                       0
Complaint Type                    0
Descriptor                     5914
Location Type                   131
Incident Zip                   2615
Address Type                   2815
City                           2614
Facility Type                  2171
Status                            0
Due Date                          3
Resolution Description            0
Resolution Action Updated Date 2187
Community Board                   0
Borough                           0
X Coordinate (State Plane)     3540
Y Coordinate (State Plane)     3540
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                       3540
Longitude                      3540
Location                       3540
dtype: int64
```

Lets check on others null values in column

Closed Date - Null values might be because of the service request is not in the state closed or resolved.

Lets validate the same

In [19]:
```
df[df['Status'] != 'Closed']['Closed Date'].isna().sum()
```

Out[19]: 2164

In [20]:
```
# We could figure out that closed date null values are because of they are not closed or resolved
```

In [21]:
```
df[df['Status'] != 'Closed']['Resolution Action Updated Date'].isna().sum()
```

Out[21]: 2185

We will impute the null values in closed day,resolution action updated date as today.. This might be logically incorrect but we dont want to remove those records so for now will impute the today date

In [22]:
```
from datetime import date
```

In [23]:
```
today = date.today()
```

In [24]:
```
df['Closed Date'] = df['Closed Date'].fillna(today)
```

In [25]:
```
df['Resolution Action Updated Date'] = df['Resolution Action Updated Date'].fillna(today)
```

In [26]:
```
## Lets remove the rows with null values in other columns
```

In [27]:
```
data_df = df.dropna(axis=0)
```

In [28]:
```
data_df.isna().sum()
```

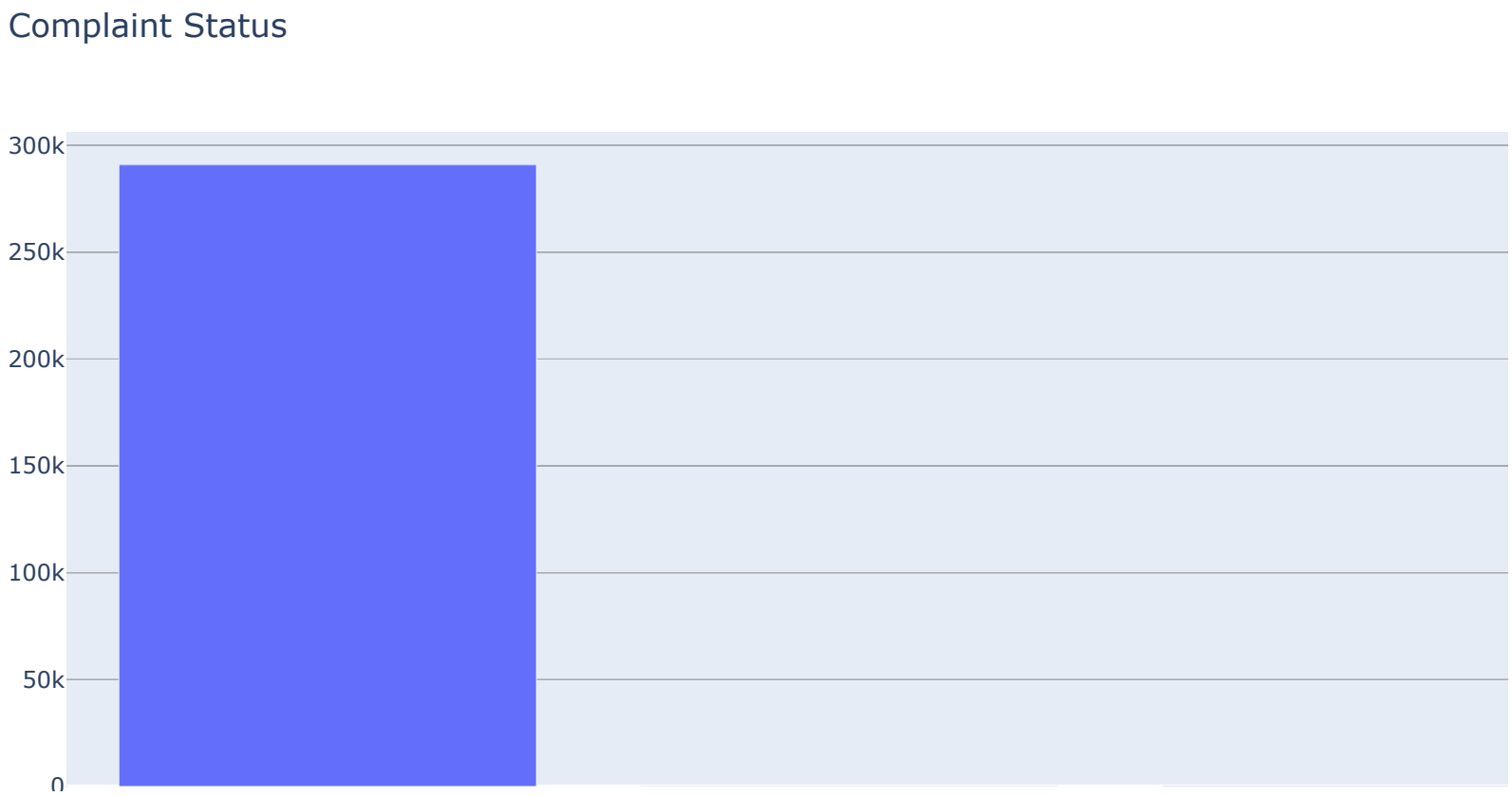
Out[28]:
```
Unique Key                        0
Created Date                      0
Closed Date                       0
Agency                            0
Agency Name                       0
Complaint Type                    0
Descriptor                        0
Location Type                     0
Incident Zip                      0
Address Type                      0
City                              0
Facility Type                     0
Status                            0
Due Date                          0
Resolution Description            0
Resolution Action Updated Date    0
Community Board                   0
Borough                           0
X Coordinate (State Plane)        0
Y Coordinate (State Plane)        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
Location                          0
dtype: int64
```

In [29]:
```
# Lets explore the data
```

In [119]:
```
status = data_df['Status'].value_counts().head(10).reset_index()
```

In [121]:
```
status.columns = ['Status','count']
```

In [122]:
```
data = [go.Bar(
   x = status['Status'],
   y = status['count']
)]
layout = go.Layout(
   {"title":"Complaint Status"})

fig = go.Figure(data=data, layout = layout)
iplot(fig)
```



In [31]:
```
data_df['Status'].value_counts().head(1)[0] / df.shape[0]
```

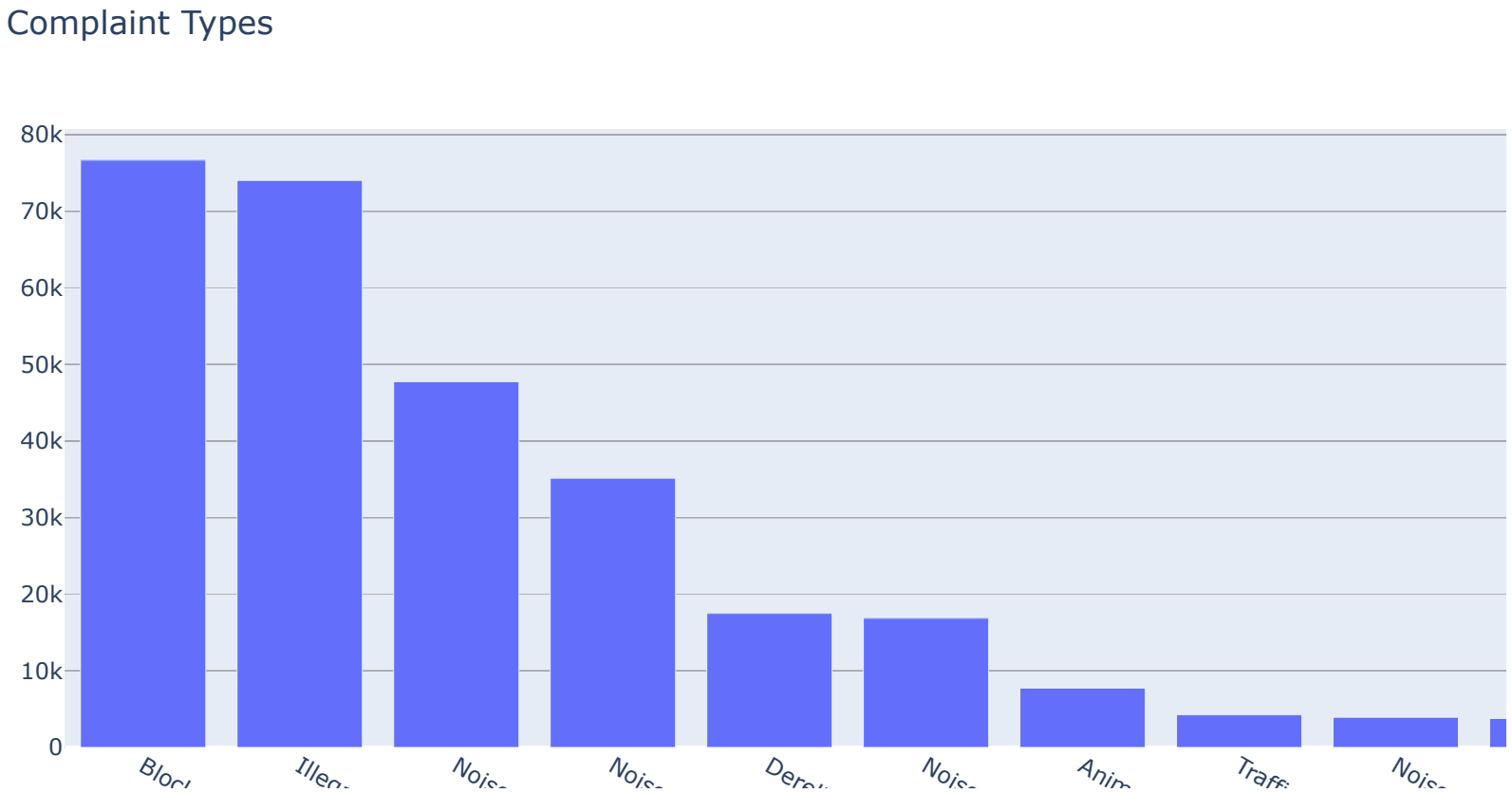
Out[31]:
```
0.9673592774145489
```

Looks like 96 % of the request are closed. Lets see the complaint types

In [116]:
```
count = data_df['Complaint Type'].value_counts().head(10).reset_index()
```

In [117]:
```
count.columns = ['Type','count']
```

In [118]:
```
data = [go.Bar(
   x = count['Type'],
   y = count['count']
)]
layout = go.Layout(
   {"title":"Complaint Types"})

fig = go.Figure(data=data, layout = layout)
iplot(fig)
```



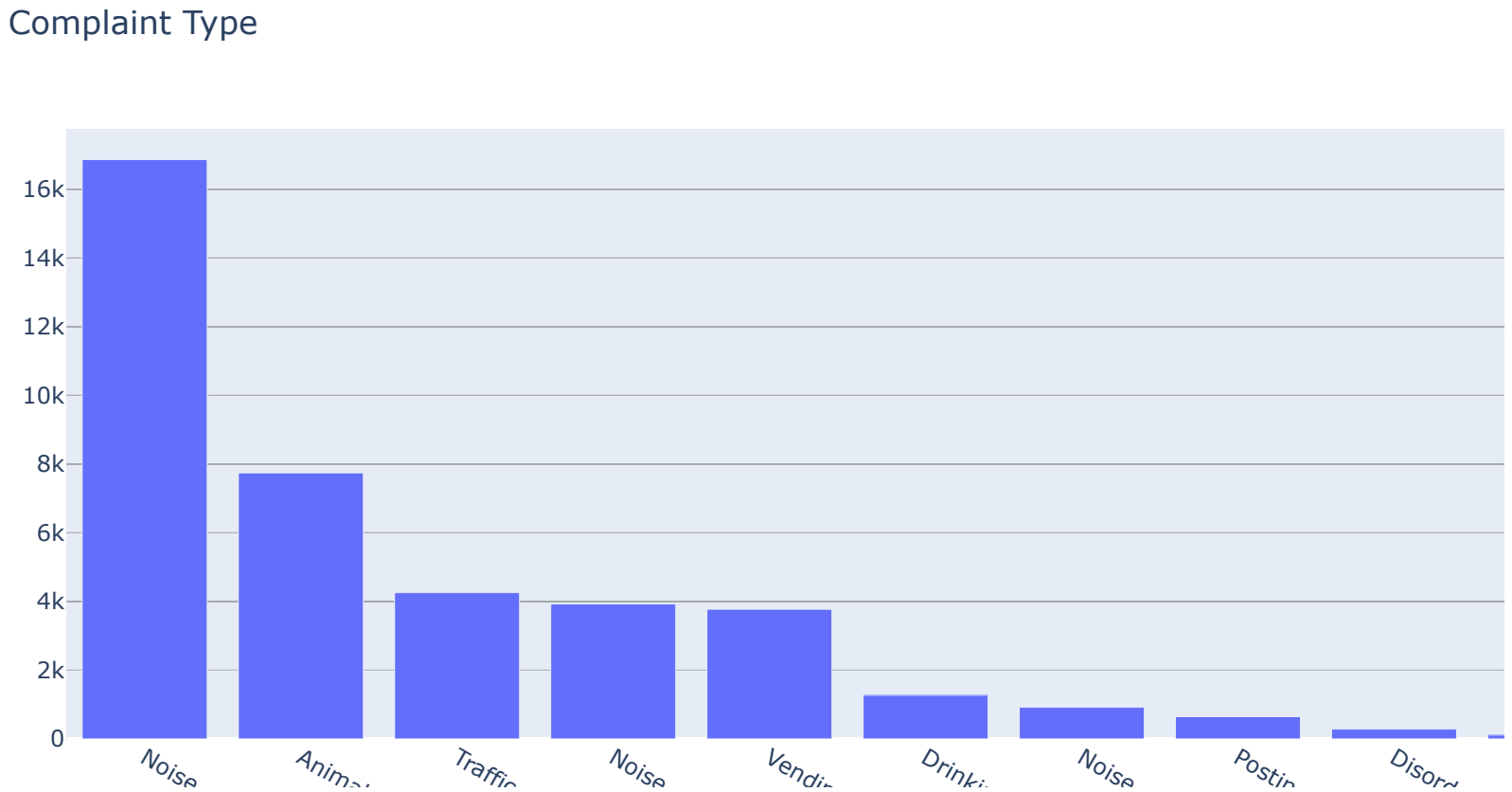
We could see that Blocked Driveway and illegal parking complaint types plays the major role more than 50 % and noise related complaints plays about 33 %

Lets check for least complaint types

In [108]:
```
tail = data_df['Complaint Type'].value_counts().tail(10).reset_index()
```

In [109]:
```
tail.columns = ['Type','count']
```

In [110]:
```
data = [go.Bar(
   x = tail['Type'],
   y = tail['count']
)]
layout = go.Layout(
   {"title":"Complaint Type"})

fig = go.Figure(data=data, layout = layout)
iplot(fig)
```

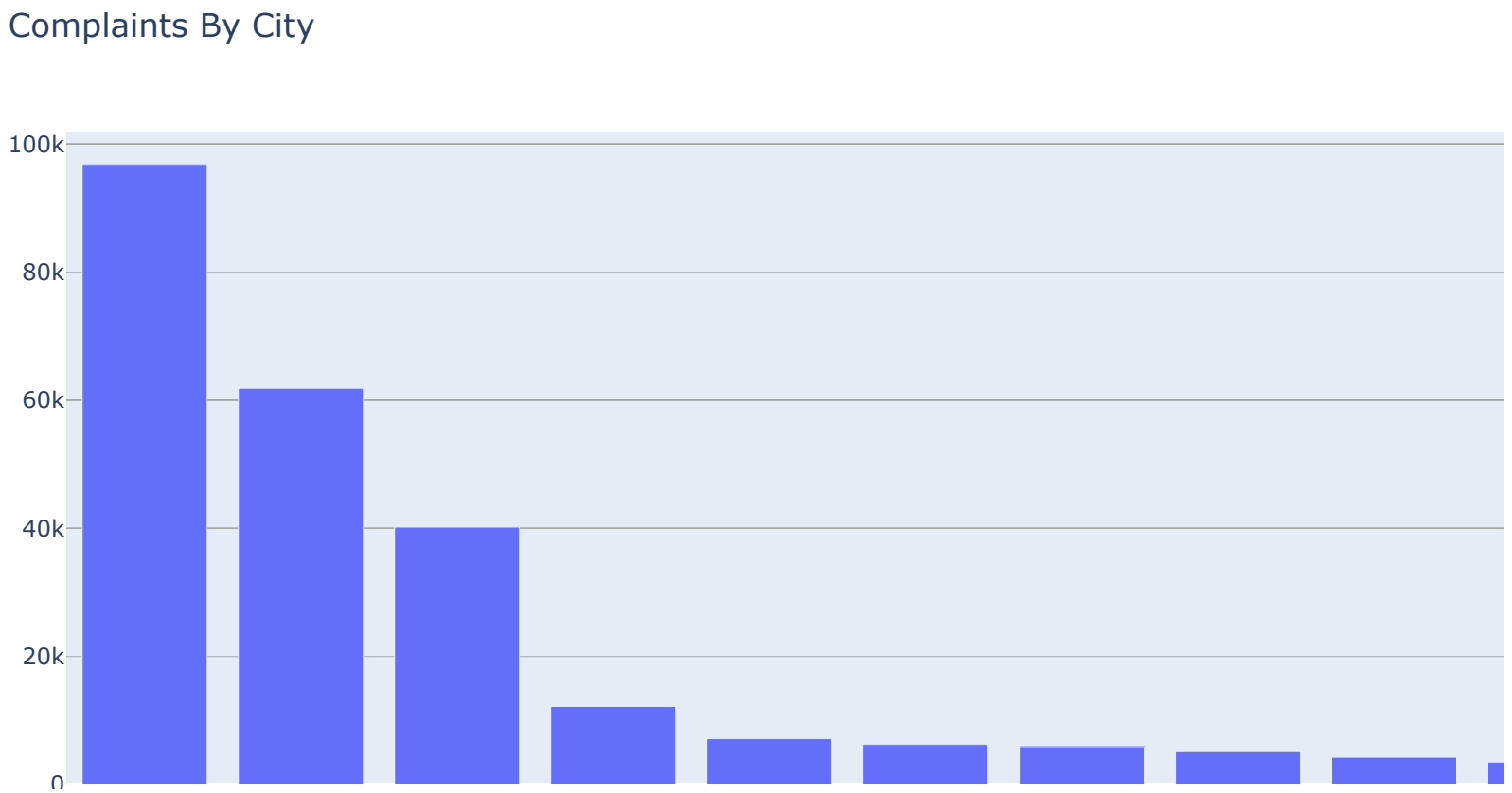


Volume of request by city wise

In [111]:
```
city = data_df['City'].value_counts().head(10).reset_index()
```

In [112]:
```
city.columns = ['city','count']
```

In [114]:
```
data = [go.Bar(
   x = city['city'],
   y = city['count']
)]
layout = go.Layout(
   {"title":"Complaints By City"})

fig = go.Figure(data=data, layout = layout)
iplot(fig)
```



In [98]:
```
data_df['City'].value_counts().head(1)[0] / data_df.shape[0]
```

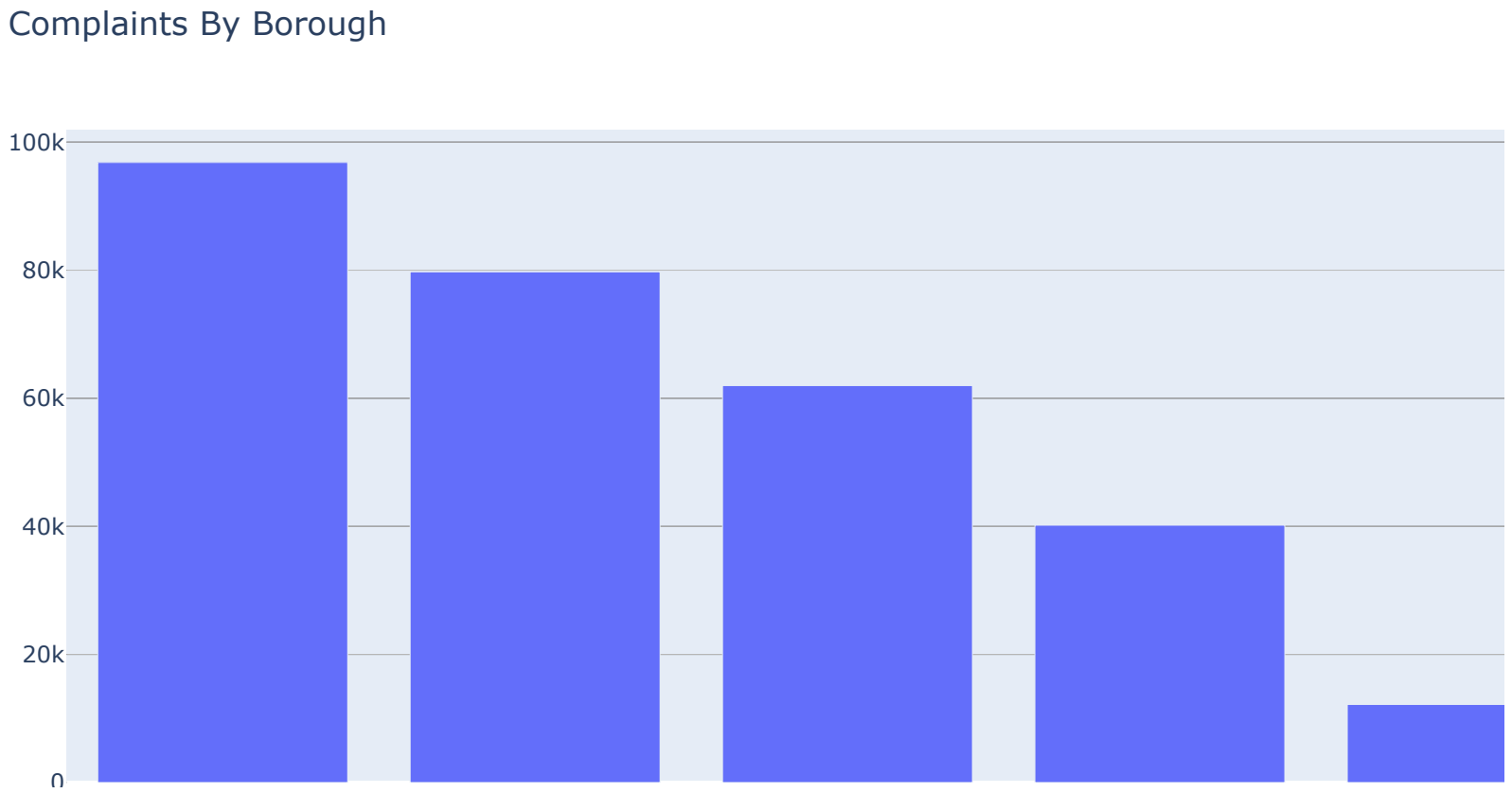
Out[98]:
```
0.3328256975980694
```

The city BROOKLYN has the most no of request / complaints.

In [99]:
```
Borough = data_df['Borough'].value_counts().head(10).reset_index()
```

In [100]:
```
Borough.columns = ['Borough','count']
```

In [115]:
```
data = [go.Bar(
   x = Borough['Borough'],
   y = Borough['count']
)]
layout = go.Layout(
   {"title":"Complaints By Borough"})

fig = go.Figure(data=data, layout = layout)

iplot(fig)
```



In [42]:
```
# Lets check for agency
```

In [43]:
```
data_df['Agency Name'].value_counts().reset_index()
```

Out[43]:

	index	Agency Name
0	New York City Police Department	290891

New York City Police Department is the only agency where the complaint request where assigned

Lets explore on when the request created,monthly trend and resolving time

In [36]:
```
import datetime
```

In [37]:
```
data_df['Created Date'] = pd.to_datetime(data_df['Created Date'])
```

In [38]:
```
data_df['Closed Date'] = pd.to_datetime(data_df['Closed Date'])
```

In [39]:
```
data_df['Due Date'] = pd.to_datetime(data_df['Due Date'])
```

In [40]:
```
data_df['Resolution Action Updated Date'] = pd.to_datetime(data_df['Resolution Action Updated Date'])
```

In [41]:
```
data_df['Created_month'] = data_df['Created Date'].dt.month
data_df['Created_year'] = data_df['Created Date'].dt.year
data_df['Created_date'] =data_df['Created Date'].dt.day
data_df['Created_hour'] = data_df['Created Date'].dt.hour
data_df['Created_quarter'] = data_df['Created Date'].dt.quarter
data_df['Created_day_of_week'] = data_df['Created Date'].dt.dayofweek
```

Lets explore resolving time of request.. We will only take the request that are closed/resolved

In [42]:
```
hist_data = data_df[data_df['Status']=='Closed']
```

In [43]:
```
hist_data.shape
```

Out[43]:
```
(290883, 41)
```

In [44]:
```
hist_data['resolved_time'] = hist_data['Closed Date'] - hist_data['Created Date']
```

In [45]:
```
## Lets convert resolved time in hours
```

In [46]:
```
hist_data['resolved_time'] = hist_data['resolved_time'].apply(lambda x: int(x.total_seconds() / 3600))
```

In [47]:
```
hist_data['resolved_time'].describe()
```

Out[47]:
```
count    290883.000000
mean          3.819144
std           6.064767
min           0.000000
25%           1.000000
50%           2.000000
75%           5.000000
max         592.000000
Name: resolved_time, dtype: float64
```

In [48]:
```
month  = hist_data.groupby(['Created_month']).agg({"resolved_time":"mean","Unique Key":"count"}).reset_index
()
```

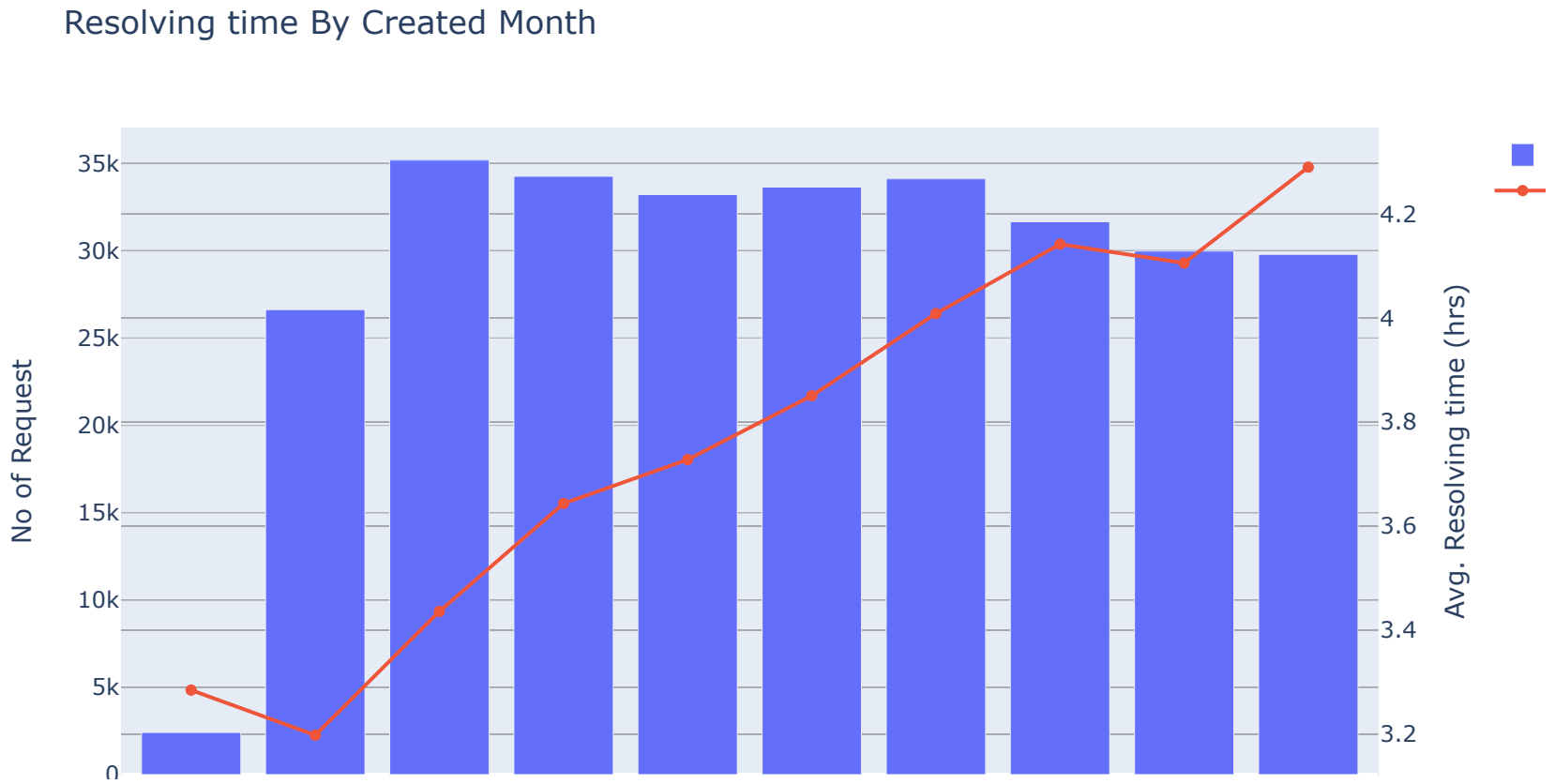
In [49]:
```
month.columns = ['created_month','resolved_time','count']
```

In [50]:
```
month
```

Out[50]:

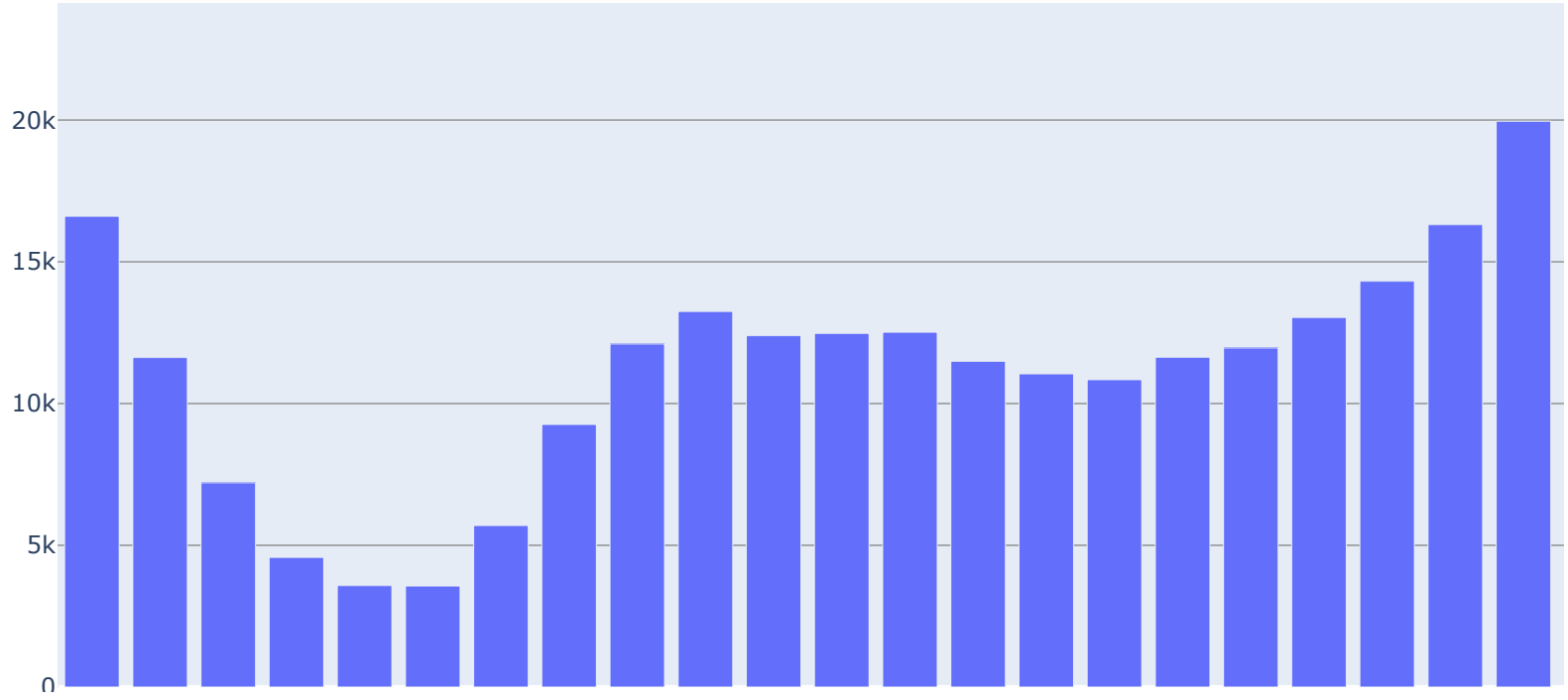
	created_month	resolved_time	count
0	3	3.284765	2409
1	4	3.198114	26621
2	5	3.436074	35189
3	6	3.643416	34264
4	7	3.727842	33216
5	8	3.850578	33643
6	9	4.008530	34116
7	10	4.141923	31658
8	11	4.105674	29979
9	12	4.290016	29788

In [58]:
```
from plotly.subplots import make_subplots

# Create figure with secondary y-axis
fig = make_subplots(specs=[[{"secondary_y": True}]])

# Add traces
fig.add_trace(
    go.Bar(x=month['created_month'], y=month['count'], name="No of Request"),
    secondary_y=False,
)

fig.add_trace(
    go.Line(x=month['created_month'], y=month['resolved_time'], name="Avg. Resolving time"),
    secondary_y=True,
)

# Add figure title
fig.update_layout(
    title_text="Resolving time By Created Month"
)

# Set x-axis title
fig.update_xaxes(title_text="Month")

# Set y-axes titles
fig.update_yaxes(title_text="No of Request", secondary_y=False)
fig.update_yaxes(title_text="Avg. Resolving time (hrs)", secondary_y=True)

fig.show()
```



In [63]:
```
hour = data_df['Created_hour'].value_counts().sort_index().reset_index()
hour.columns = ['Created_hour','count']
```

In [64]:
```
data = [go.Bar(
   x = hour['Created_hour'],
   y = hour['count']
)]
fig = go.Figure(data=data)
iplot(fig)
```



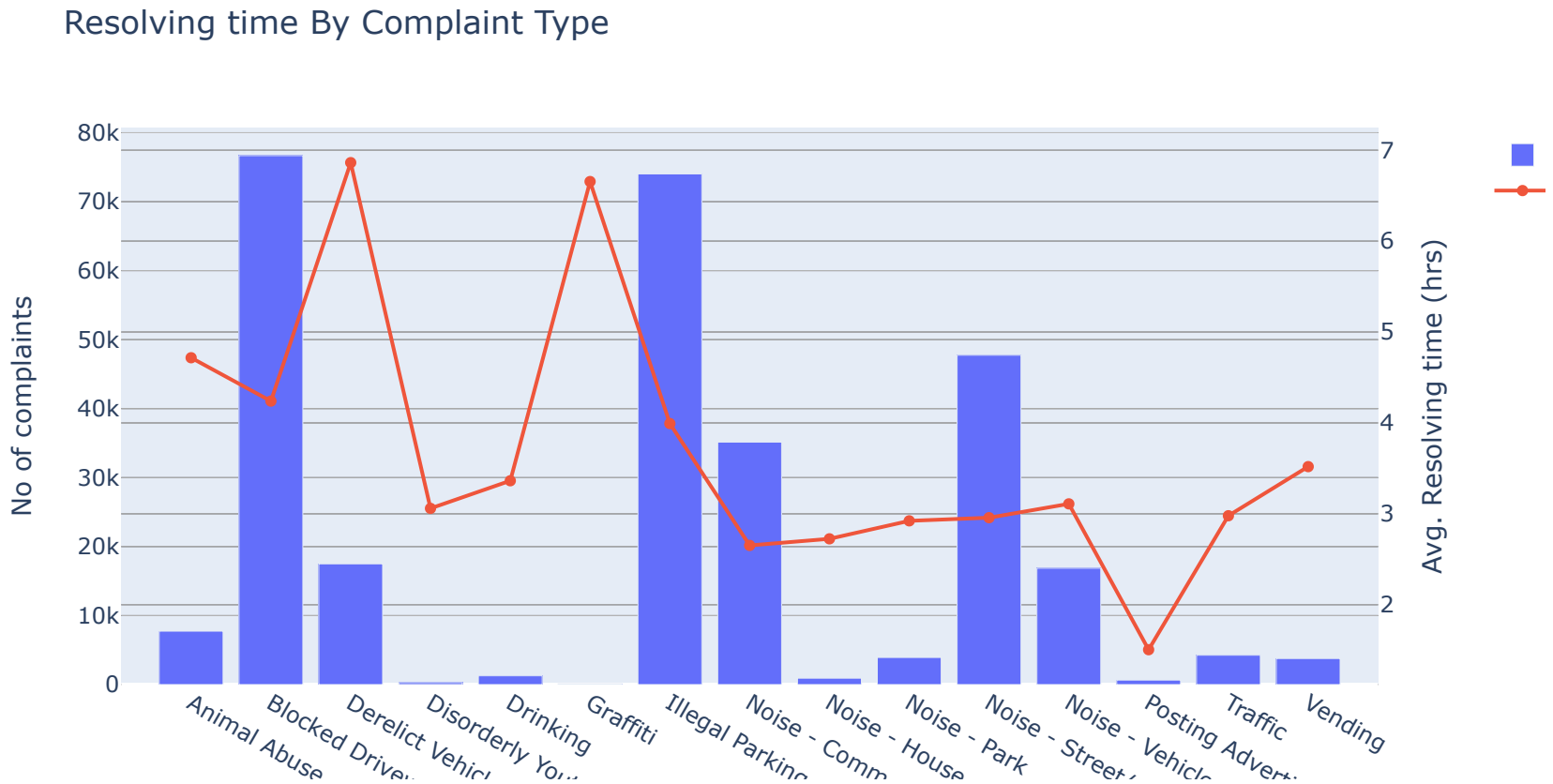
In [62]:
```
## This shows that after 8 PM the number of complaints are more
```

In [76]:
```
day = hist_data.groupby(['Created_day_of_week']).agg({"resolved_time":"mean","Unique Key":"count"}).reset_ind
ex()
day.columns = ['Created day','resolved_time','count']
```

In [80]:
```
fig = make_subplots(specs=[[{"secondary_y": True}]])

# Add traces
fig.add_trace(
    go.Bar(x=day['Created day'], y=day['count'], name="No of Complaints"),
    secondary_y=False,
)

fig.add_trace(
    go.Line(x=day['Created day'], y=day['resolved_time'], name="Avg. Resolving time"),
    secondary_y=True,
)

# Add figure title
fig.update_layout(
    title_text="Resolving time By Created day in week"
)

# Set x-axis title
fig.update_xaxes(title_text="Complaint Type")

# Set y-axes titles
fig.update_yaxes(title_text="No of complaints", secondary_y=False)
fig.update_yaxes(title_text="Avg. Resolving time (hrs)", secondary_y=True)

fig.show()
```

In [61]:
```
resolved_time = hist_data.groupby(['Complaint Type']).agg({"resolved_time":"mean","Unique Key":"count"}).reset_index()
```

In [65]:
```
fig = make_subplots(specs=[[{"secondary_y": True}]])

# Add traces
fig.add_trace(
    go.Bar(x=resolved_time['Complaint Type'], y=resolved_time['Unique Key'], name="No of Complaints"),
    secondary_y=False,
)

fig.add_trace(
    go.Line(x=resolved_time['Complaint Type'], y=resolved_time['resolved_time'], name="Avg. Resolving time"),
    secondary_y=True,
)

# Add figure title
fig.update_layout(
    title_text="Resolving time By Complaint Type"
)

# Set x-axis title
fig.update_xaxes(title_text="Complaint Type")

# Set y-axes titles
fig.update_yaxes(title_text="No of complaints", secondary_y=False)
fig.update_yaxes(title_text="Avg. Resolving time (hrs)", secondary_y=True)

fig.show()
```

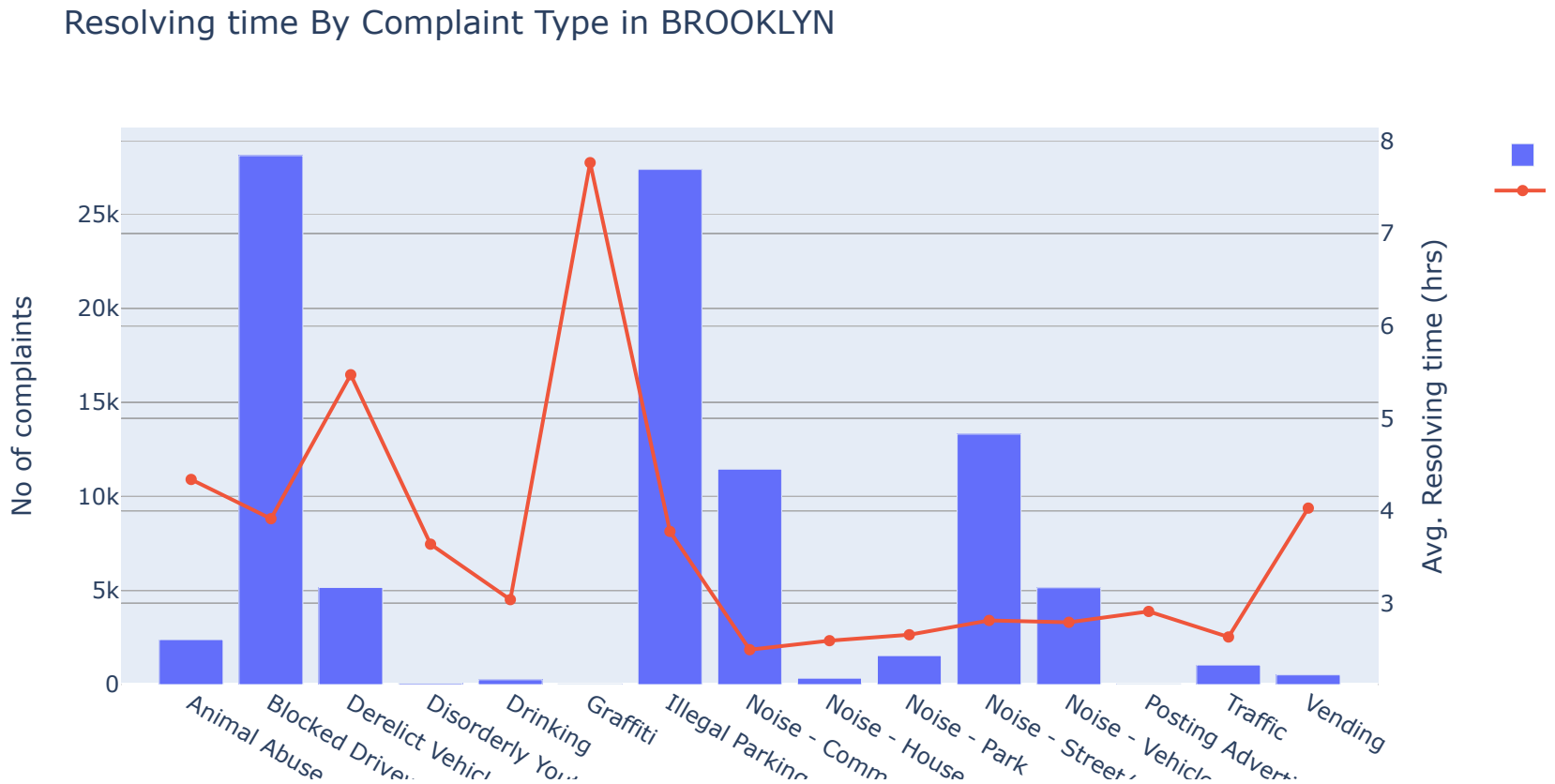


We could find that Blocked Driveway has very less resolving time compared to other complaint types. Lets explore the Brookyln city which has most no of request

In [67]:
```
brooklyn = hist_data[hist_data['City'] =='BROOKLYN']
```

In [70]:
```
Borough = brooklyn.groupby(['Complaint Type']).agg({"resolved_time":"mean","Unique Key":"count"}).reset_index
()
```

In [75]:
```
fig = make_subplots(specs=[[{"secondary_y": True}]])

# Add traces
fig.add_trace(
    go.Bar(x=Borough['Complaint Type'], y=Borough['Unique Key'], name="No of Complaints"),
    secondary_y=False,
)

fig.add_trace(
    go.Line(x=Borough['Complaint Type'], y=Borough['resolved_time'], name="Avg. Resolving time"),
    secondary_y=True,
)

# Add figure title
fig.update_layout(
    title_text="Resolving time By Complaint Type in BROOKLYN "
)

# Set x-axis title
fig.update_xaxes(title_text="Complaint Type")

# Set y-axes titles
fig.update_yaxes(title_text="No of complaints", secondary_y=False)
fig.update_yaxes(title_text="Avg. Resolving time (hrs)", secondary_y=True)

fig.show()
```



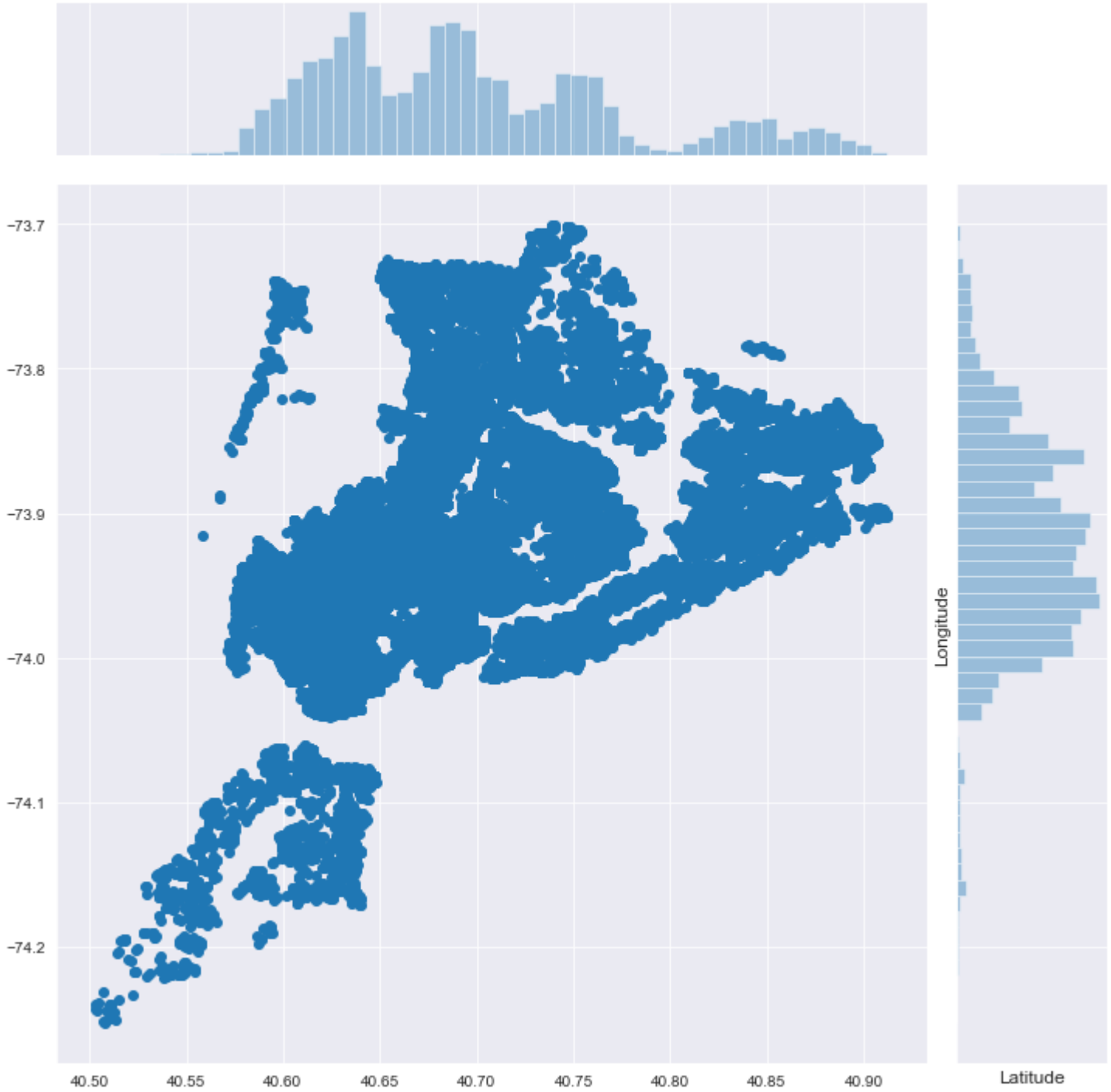
In [66]:
```
drive = data_df[(data_df['Complaint Type']=='Blocked Driveway')|(data_df['Complaint Type']=='Illegal Parking'
) & (data_df['City']== 'BROOKLYN')]
```

In [82]:
```
drive.shape
```

Out[82]: (104064, 41)

In [83]:
```
plt.figure(figsize=(12,12))
sns.jointplot(x=drive.Latitude.values, y=drive.Longitude.values, size=10,)
plt.ylabel('Longitude', fontsize=12)
plt.xlabel('Latitude', fontsize=12)
plt.show()
```

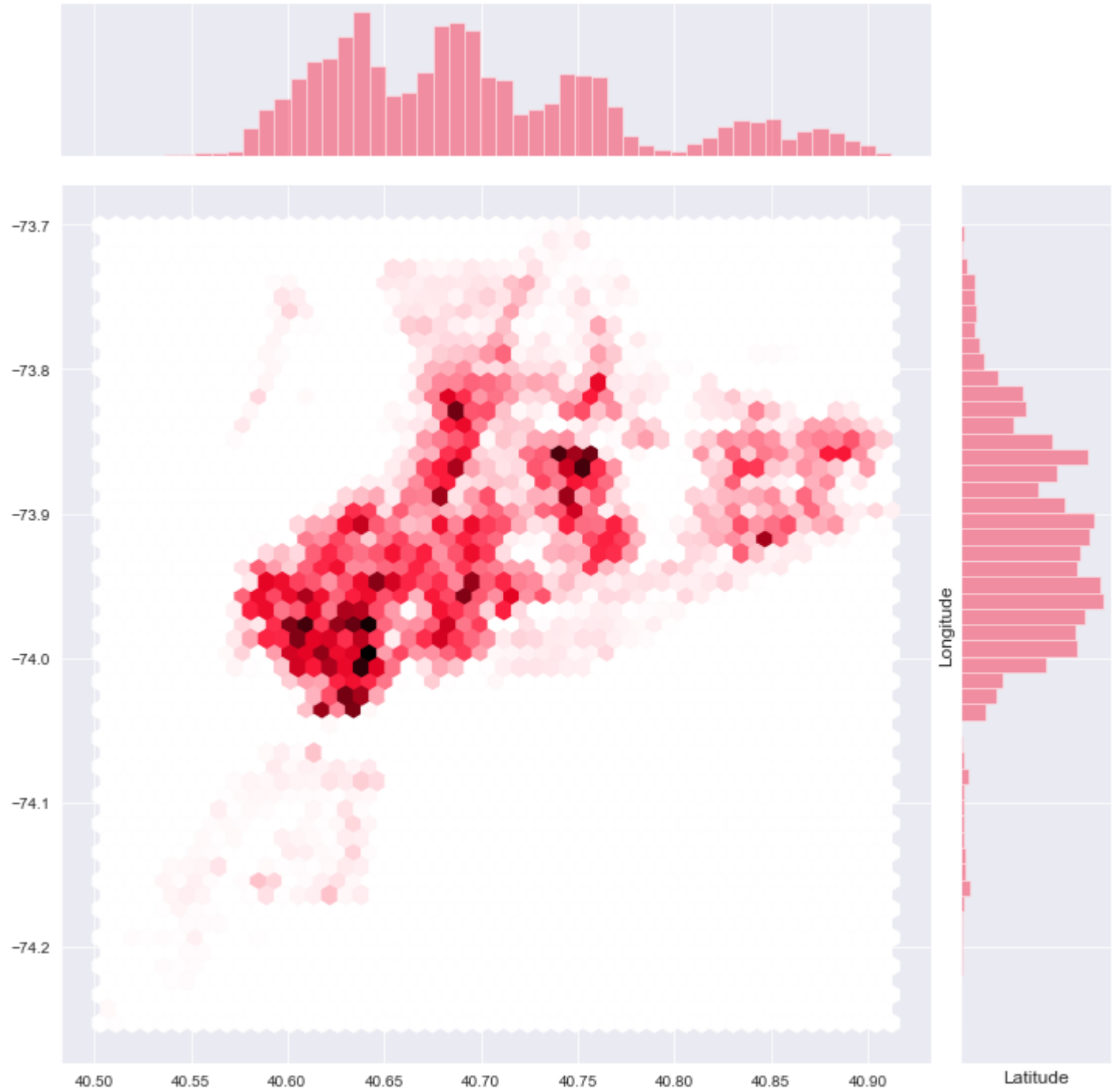
```
<Figure size 864x864 with 0 Axes>
```


−73.7
−73.8
−73.9
−74.0
−74.1
−74.2
Longitude
40.50
40.55
40.60
40.65
40.70
40.75
40.80
40.85
40.90
Latitude

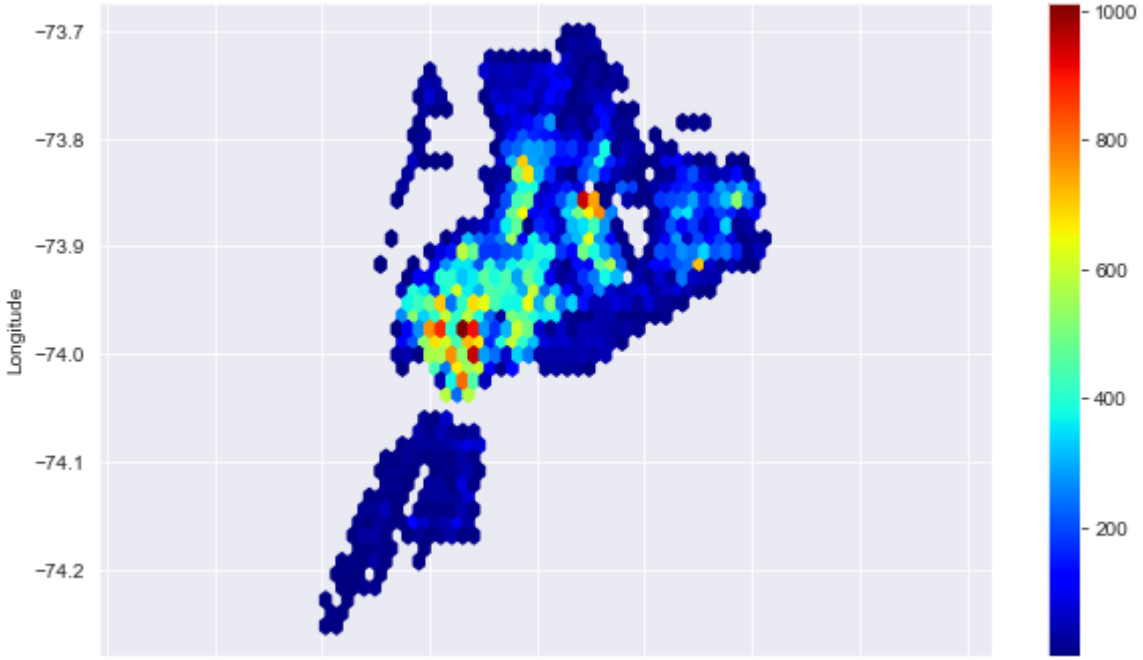
In [84]:
```
#### We couldn't find much information from this. Lets try hex..
### (Scatter plot in ploty - taking too much time and my bad lap goes on hanging.. Stay tuned will find and p
ut)
```

In [89]:
```
plt.figure(figsize=(12,12))
sns.jointplot(x=drive.Latitude.values, y=drive.Longitude.values,kind="hex", color="#fc0328", size=10)
plt.ylabel('Longitude', fontsize=12)
plt.xlabel('Latitude', fontsize=12)
plt.show()
```

```
<Figure size 864x864 with 0 Axes>
```


−73.7
−73.8
−73.9
−74.0
−74.1
−74.2
40.50
40.55
40.60
40.65
40.70
40.75
40.80
40.85
40.90
Longitude
Latitude

In [90]:
```
drive.plot(kind='hexbin',x='Latitude',y='Longitude',gridsize=40,colormap='jet',mincnt=1,figsize=(10,6)).axis(
'equal')
plt.show()
```



This is better . We can see the most complaints from central NYC

Conclusion

From the exploration we conclude that nearly 50 % of complaints are blocked driveway and illegal parking These two complaints might also inter connected because of illegal parking there can be a chance of blocked driveway in the city.The complaints after 8 PM for blocked driveway and illegal parking are about nearly 57 %.

Brookyln has the most (70 %) no of blocked driveway and illegal praking complaints. 2/3 of blocked driveway and illegal parking compaints where from Brookyln

We could see that resolving time for blocked driveway illegal parking are lesser (around 4 hrs) compared to others complaints..But this might differ for each city..

The other 33 % of complaints where based on the noise related issues.

In [ ]:

In [ ]: