Analysis

May 14, 2023

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os
import statsmodels.api as sm
from scipy.stats import kruskal
```

[2]:
```
#1. Understanding the dataset
```

[3]:
```
cwd = os.getcwd()
dataset_dir = os.path.join(cwd,'Dataset')
dataset_path = os.path.
 ↪join(dataset_dir,'311_Service_Requests_from_2010_to_Present.csv')
```

[4]:
```
#1.1 Import the dataset
```

[5]:
```
dataset = pd.read_csv(dataset_path, low_memory=False)
```

[6]:
```
#1.2 Visualize the dataset
```

[7]:
```
dataset.head()
```

[7]:
```
   Unique Key            Created Date              Closed Date Agency  \
0    32310363  12/31/2015 11:59:45 PM  01/01/2016 12:55:15 AM   NYPD
1    32309934  12/31/2015 11:59:44 PM  01/01/2016 01:26:57 AM   NYPD
2    32309159  12/31/2015 11:59:29 PM  01/01/2016 04:51:03 AM   NYPD
3    32305098  12/31/2015 11:57:46 PM  01/01/2016 07:43:13 AM   NYPD
4    32306529  12/31/2015 11:56:58 PM  01/01/2016 03:24:42 AM   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       10034.0
1                     No Access  Street/Sidewalk       11105.0
2                     No Access  Street/Sidewalk       10458.0
3  Commercial Overnight Parking  Street/Sidewalk       10461.0
4              Blocked Sidewalk  Street/Sidewalk       11373.0

        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  40.865682 -73.923501
1                 NaN  40.775945 -73.915094
2                 NaN  40.870325 -73.888525
3                 NaN  40.835994 -73.828379
4                 NaN  40.733060 -73.874170

                                  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]
```

[8]:
```
dataset.tail()
```

[8]:
```
        Unique Key            Created Date            Closed Date Agency  \
364553    29609918  01/01/2015 12:04:44 AM  01/01/2015 10:22:31 AM   NYPD
364554    29608392  01/01/2015 12:04:28 AM  01/01/2015 02:25:02 AM   NYPD
364555    29607589  01/01/2015 12:01:30 AM  01/01/2015 12:20:33 AM   NYPD
364556    29610889  01/01/2015 12:01:29 AM  01/01/2015 02:42:22 AM   NYPD
364557    29611816  01/01/2015 12:00:50 AM  01/01/2015 02:47:50 AM   NYPD

                           Agency Name           Complaint Type  \
364553  New York City Police Department          Illegal Parking
```

```
364554  New York City Police Department          Noise - Vehicle
364555  New York City Police Department  Noise - Street/Sidewalk
364556  New York City Police Department         Blocked Driveway
364557  New York City Police Department         Blocked Driveway

             Descriptor    Location Type  Incident Zip      Incident Address  \
364553   Blocked Hydrant  Street/Sidewalk       11421.0         84-25 85 ROAD
364554    Car/Truck Horn  Street/Sidewalk       10468.0  2555 SEDGWICK AVENUE
364555  Loud Music/Party  Street/Sidewalk       10031.0   508 WEST 139 STREET
364556         No Access  Street/Sidewalk       10466.0   931 EAST 226 STREET
364557         No Access  Street/Sidewalk       11420.0     123-19 135 STREET

        … Bridge Highway Name Bridge Highway Direction Road Ramp  \
364553  …                  NaN                       NaN       NaN
364554  …                  NaN                       NaN       NaN
364555  …                  NaN                       NaN       NaN
364556  …                  NaN                       NaN       NaN
364557  …                  NaN                       NaN       NaN

       Bridge Highway Segment Garage Lot Name Ferry Direction  \
364553                    NaN             NaN             NaN
364554                    NaN             NaN             NaN
364555                    NaN             NaN             NaN
364556                    NaN             NaN             NaN
364557                    NaN             NaN             NaN

       Ferry Terminal Name   Latitude  Longitude  \
364553                 NaN  40.695145 -73.860949
364554                 NaN  40.867830 -73.907178
364555                 NaN  40.821647 -73.950873
364556                 NaN  40.886361 -73.853290
364557                 NaN  40.674212 -73.803585

                                     Location
364553   (40.69514470265117, -73.86094888534394)
364554   (40.86782963689454, -73.90717786644662)
364555  (40.821646626438095, -73.95087342885292)
364556   (40.88636077906953, -73.85329048666742)
364557  (40.674211762243935, -73.80358548685278)

[5 rows x 53 columns]
```

[9]:
```
#1.3 Print the columns of the DataFrame
```

[10]:
```
dataset.columns
```

[10]:
```
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')
```

[11]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 364558 entries, 0 to 364557
Data columns (total 53 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   Unique Key                      364558 non-null  int64
 1   Created Date                    364558 non-null  object
 2   Closed Date                     362177 non-null  object
 3   Agency                          364558 non-null  object
 4   Agency Name                     364558 non-null  object
 5   Complaint Type                  364558 non-null  object
 6   Descriptor                      358057 non-null  object
 7   Location Type                   364425 non-null  object
 8   Incident Zip                    361560 non-null  float64
 9   Incident Address                312859 non-null  object
 10  Street Name                     312859 non-null  object
 11  Cross Street 1                  307370 non-null  object
 12  Cross Street 2                  306753 non-null  object
 13  Intersection Street 1           51120 non-null   object
 14  Intersection Street 2           50512 non-null   object
 15  Address Type                    361306 non-null  object
 16  City                            361561 non-null  object
 17  Landmark                        375 non-null     object
 18  Facility Type                   362169 non-null  object
 19  Status                          364558 non-null  object
 20  Due Date                        364555 non-null  object
 21  Resolution Description          364558 non-null  object
 22  Resolution Action Updated Date  362156 non-null  object
 23  Community Board                 364558 non-null  object
```

```
 24  Borough                          364558 non-null  object
 25  X Coordinate (State Plane)       360528 non-null  float64
 26  Y Coordinate (State Plane)       360528 non-null  float64
 27  Park Facility Name               364558 non-null  object
 28  Park Borough                     364558 non-null  object
 29  School Name                      364558 non-null  object
 30  School Number                    364558 non-null  object
 31  School Region                    364557 non-null  object
 32  School Code                      364557 non-null  object
 33  School Phone Number              364558 non-null  object
 34  School Address                   364558 non-null  object
 35  School City                      364558 non-null  object
 36  School State                     364558 non-null  object
 37  School Zip                       364557 non-null  object
 38  School Not Found                 364558 non-null  object
 39  School or Citywide Complaint     0 non-null       float64
 40  Vehicle Type                     0 non-null       float64
 41  Taxi Company Borough             0 non-null       float64
 42  Taxi Pick Up Location            0 non-null       float64
 43  Bridge Highway Name              297 non-null     object
 44  Bridge Highway Direction         297 non-null     object
 45  Road Ramp                        262 non-null     object
 46  Bridge Highway Segment           262 non-null     object
 47  Garage Lot Name                  0 non-null       float64
 48  Ferry Direction                  1 non-null       object
 49  Ferry Terminal Name              2 non-null       object
 50  Latitude                         360528 non-null  float64
 51  Longitude                        360528 non-null  float64
 52  Location                         360528 non-null  object
dtypes: float64(10), int64(1), object(42)
memory usage: 147.4+ MB
```

[12]:
```
dataset.dtypes
```

[12]:
```
Unique Key                        int64
Created Date                     object
Closed Date                      object
Agency                           object
Agency Name                      object
Complaint Type                   object
Descriptor                       object
Location Type                    object
Incident Zip                    float64
Incident Address                 object
Street Name                      object
Cross Street 1                   object
Cross Street 2                   object
```

```
Intersection Street 1              object
Intersection Street 2              object
Address Type                       object
City                               object
Landmark                           object
Facility Type                      object
Status                             object
Due Date                           object
Resolution Description             object
Resolution Action Updated Date     object
Community Board                    object
Borough                            object
X Coordinate (State Plane)        float64
Y Coordinate (State Plane)        float64
Park Facility Name                 object
Park Borough                       object
School Name                        object
School Number                      object
School Region                      object
School Code                        object
School Phone Number                object
School Address                     object
School City                        object
School State                       object
School Zip                         object
School Not Found                   object
School or Citywide Complaint      float64
Vehicle Type                      float64
Taxi Company Borough              float64
Taxi Pick Up Location             float64
Bridge Highway Name                object
Bridge Highway Direction           object
Road Ramp                          object
Bridge Highway Segment             object
Garage Lot Name                   float64
Ferry Direction                    object
Ferry Terminal Name                object
Latitude                          float64
Longitude                         float64
Location                           object
dtype: object
```

[13]:
```
#1.4 Identify shape of the dataset
```

[14]:
```
dataset.shape
```

[14]:
```
(364558, 53)
```

[15]:
```
#1.5 Identify the variables with null values
```

[16]:
```
dataset.isna().sum()
```

[16]:
```
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
```

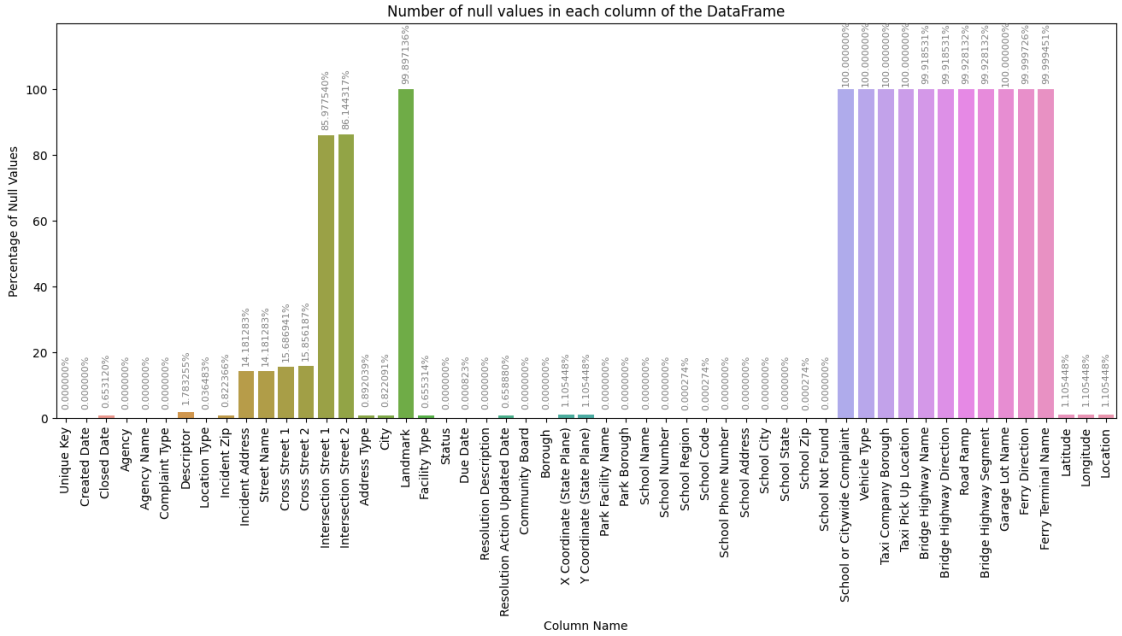
```
Bridge Highway Name                364261
Bridge Highway Direction           364261
Road Ramp                          364296
Bridge Highway Segment             364296
Garage Lot Name                    364558
Ferry Direction                    364557
Ferry Terminal Name                364556
Latitude                             4030
Longitude                            4030
Location                             4030
dtype: int64
```

[17]:
```
#2. Perform basic data exploratory analysis
```

[18]:
```
graph_dir = os.path.join(cwd,'Graph')
```

[19]:
```
#2.1  Draw a frequency plot to show the number of null values in each column of␣
 ↪the DataFrame
```

[20]:
```
null_dict = {}
row_count = len(dataset)
for column in dataset.columns:
    null_dict[column] = (dataset[column].isna().sum() / row_count) * 100

plt.figure(figsize=(16, 6))
ax = sns.barplot(x=list(null_dict.keys()), y=list(null_dict.values()))
plt.xticks(rotation=90)
ax = plt.gca()
ax.margins(None, 0.199)
plt.xlabel('Column Name')
plt.ylabel('Percentage of Null Values')
title = 'Number of null values in each column of the DataFrame'
plt.title(title)
for p in ax.patches:
    ax.annotate(f'{p.get_height():.6f}%', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()+9),
                ha='center', va='center', fontsize=8, color='gray', xytext=(0,␣
 ↪5),
                textcoords='offset points', rotation=90)
plt.savefig(os.path.join(graph_dir, title + '.png'), bbox_inches='tight')
plt.show()
```



[21]:
```
#2.2 Missing value treatment
```

[22]:
```
#2.2.1 Remove the records whose Closed Date values are null
```

[23]:
```
dataset = dataset.drop(dataset[dataset['Closed Date'].isna()==True].index)
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 362177 entries, 0 to 364557
Data columns (total 53 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  Intersection Street 1           50628 non-null   object
 14  Intersection Street 2           50504 non-null   object
```

```
 15  Address Type                    361248 non-null  object
 16  City                            361503 non-null  object
 17  Landmark                        375 non-null     object
 18  Facility Type                   362159 non-null  object
 19  Status                          362177 non-null  object
 20  Due Date                        362176 non-null  object
 21  Resolution Description          362177 non-null  object
 22  Resolution Action Updated Date  362138 non-null  object
 23  Community Board                 362177 non-null  object
 24  Borough                         362177 non-null  object
 25  X Coordinate (State Plane)      360470 non-null  float64
 26  Y Coordinate (State Plane)      360470 non-null  float64
 27  Park Facility Name              362177 non-null  object
 28  Park Borough                    362177 non-null  object
 29  School Name                     362177 non-null  object
 30  School Number                   362177 non-null  object
 31  School Region                   362176 non-null  object
 32  School Code                     362176 non-null  object
 33  School Phone Number             362177 non-null  object
 34  School Address                  362177 non-null  object
 35  School City                     362177 non-null  object
 36  School State                    362177 non-null  object
 37  School Zip                      362176 non-null  object
 38  School Not Found                362177 non-null  object
 39  School or Citywide Complaint    0 non-null       float64
 40  Vehicle Type                    0 non-null       float64
 41  Taxi Company Borough            0 non-null       float64
 42  Taxi Pick Up Location           0 non-null       float64
 43  Bridge Highway Name             297 non-null     object
 44  Bridge Highway Direction        297 non-null     object
 45  Road Ramp                       262 non-null     object
 46  Bridge Highway Segment          262 non-null     object
 47  Garage Lot Name                 0 non-null       float64
 48  Ferry Direction                 0 non-null       object
 49  Ferry Terminal Name             0 non-null       object
 50  Latitude                        360470 non-null  float64
 51  Longitude                       360470 non-null  float64
 52  Location                        360470 non-null  object
dtypes: float64(10), int64(1), object(42)
memory usage: 149.2+ MB
```

[24]:
```
#Dropping columns that are more than 50% empty
```

[25]:
```
print(len(list(dataset.columns)))

datanew = dataset
for column in list(dataset.columns) :
```

```
    if dataset[column].isna().sum()/len(dataset) > 0.75 :
        datanew.drop(column,axis=1,inplace=True)

del datanew
print(len(list(dataset.columns)),list(dataset.columns))
```

```
53
39 ['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', 'Address Type',
'City', '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', 'Latitude', 'Longitude', 'Location']
```

[26]:
```
#2.3 Analyze the date column, and remove entries that have an incorrect timeline
```

[27]:
```
#2.3.1 Calculate the time elapsed in closed and creation date
```

[28]:
```
# Convert 'Created Date' and 'Closed Date' columns to datetime format
dataset['Created Date'] = pd.to_datetime(dataset['Created Date'], format='%m/%d/
  ↪%Y %I:%M:%S %p')
dataset['Closed Date'] = pd.to_datetime(dataset['Closed Date'], format='%m/%d/
  ↪%Y %I:%M:%S %p')

# Drop entries where 'Created Date' is after 'Closed Date'
dataset = dataset[dataset['Created Date'] <= dataset['Closed Date']]

dataset['Time Elapsed'] = dataset['Closed Date'] - dataset['Created Date']
```

[29]:
```
#2.3.2 Convert the calculated date to seconds to get a better representation
```

[30]:
```
dataset['Time Elapsed'] = dataset['Time Elapsed'].dt.total_seconds()
```

[31]:
```
#2.3.3 View the descriptive statistics for the newly created column
```

[32]:
```
dataset['Time Elapsed'].describe()
```

[32]:
```
count    3.621770e+05
mean     1.511330e+04
std      2.110255e+04
min      6.100000e+01
25%      4.533000e+03
50%      9.616000e+03
75%      1.887800e+04
max      2.134342e+06
```

```
Name: Time Elapsed, dtype: float64
```

[33]:
```
#2.3.4 Check the number of null values in the Complaint_Type and City columns
```

[34]:
```
print("Number of null values in 'Complaint_Type' column:", dataset['Complaint␣
↪Type'].isnull().sum())
print("Number of null values in 'City' column:", dataset['City'].isnull().sum())
```

```
Number of null values in 'Complaint_Type' column: 0
Number of null values in 'City' column: 674
```

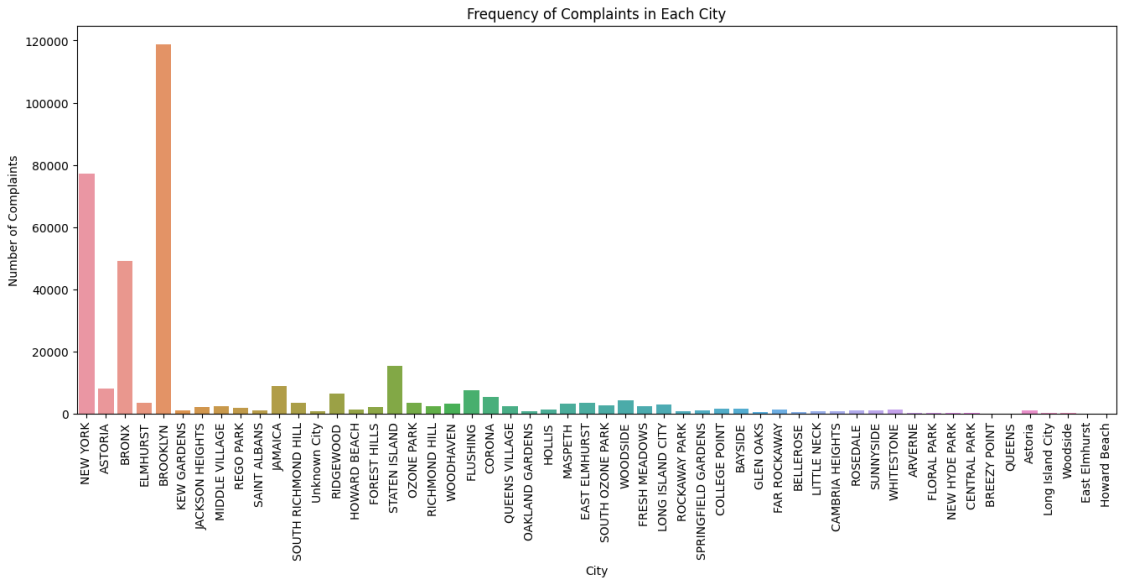
[35]:
```
#2.3.5 Impute the NA value with Unknown City
```

[36]:
```
dataset['City'].fillna(value='Unknown City', inplace=True)
print("Number of null values in 'City' column:", dataset['City'].isnull().sum())
```

```
Number of null values in 'City' column: 0
```

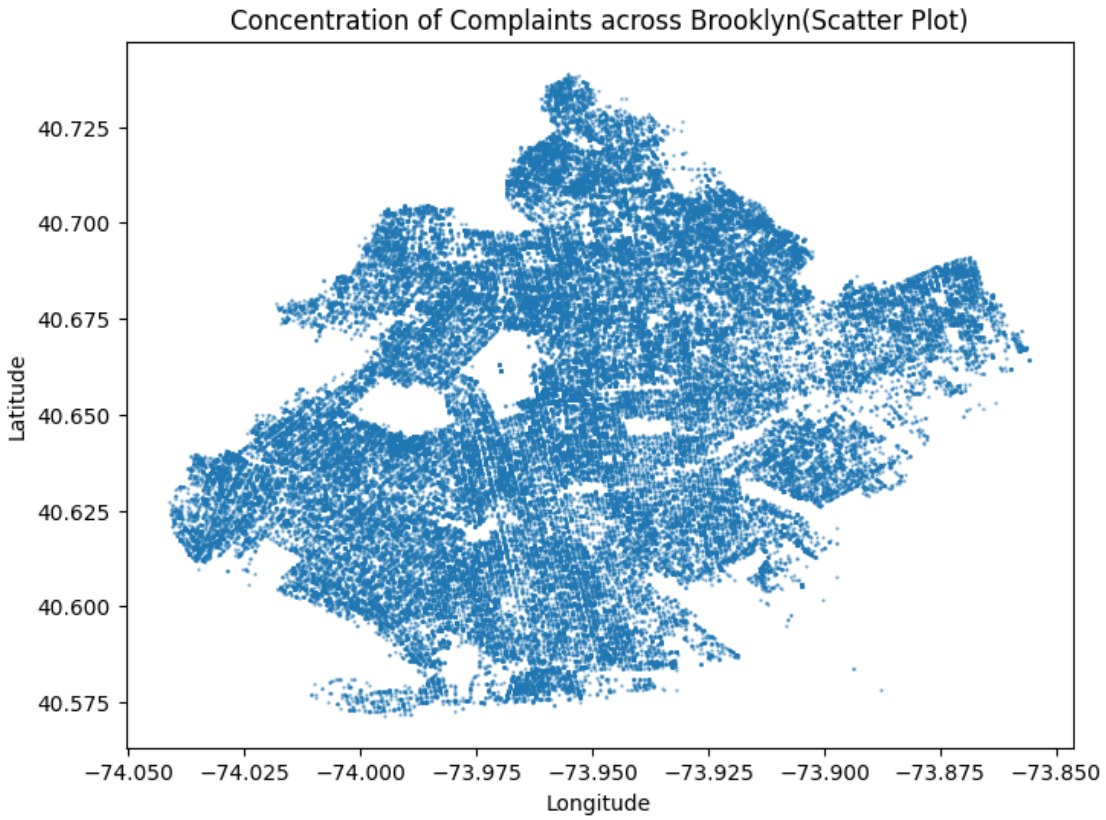
[37]:
```
#2.3.6 Draw a frequency plot for the complaints in each city
```

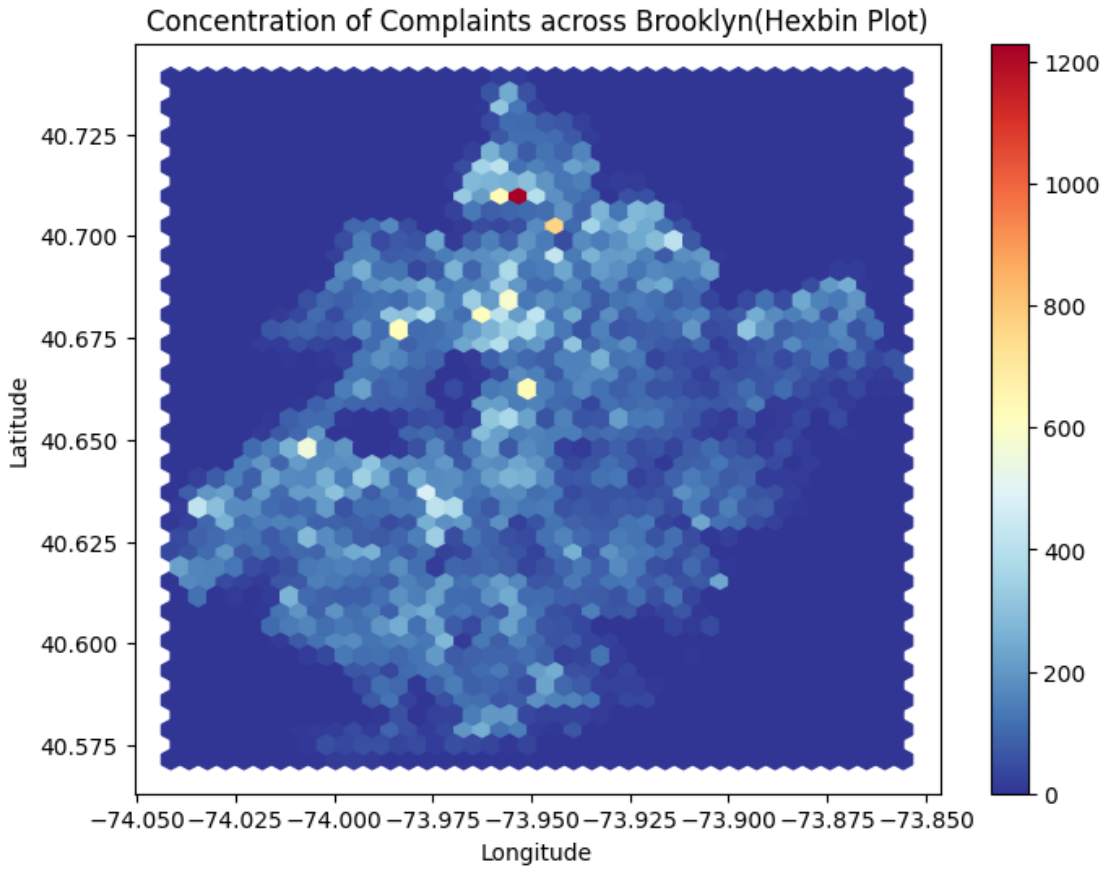
[38]:
```
plt.figure(figsize=(16, 6))
sns.countplot(x='City', data=dataset)
plt.xticks(rotation=90)
plt.xlabel('City')
plt.ylabel('Number of Complaints')
title = 'Frequency of Complaints in Each City'
plt.title(title)
plt.savefig(os.path.join(graph_dir,title+'.png'),bbox_inches='tight')
plt.show()
plt.close()
```



[39]:
```
#2.3.7 Create a scatter and hexbin plot of the concentration of complaints␣
↪across Brooklyn
```

[40]:
```
# Filter the dataset to keep only the Brooklyn borough
brooklyn_data = dataset[dataset['Borough'] == 'BROOKLYN']

# Create a scatter plot
plt.figure(figsize=(8, 6))
plt.scatter(brooklyn_data['Longitude'], brooklyn_data['Latitude'], s=0.5,␣
↪alpha=0.5)
plt.xlabel('Longitude')
plt.ylabel('Latitude')
title = 'Concentration of Complaints across Brooklyn(Scatter Plot)'
plt.title(title)
plt.savefig(os.path.join(graph_dir,title+'.png'),bbox_inches='tight')
plt.show()
plt.close()

# Create a hexbin plot
plt.figure(figsize=(8, 6))
plt.hexbin(brooklyn_data['Longitude'], brooklyn_data['Latitude'], gridsize=40,␣
↪cmap='RdYlBu_r')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
title = 'Concentration of Complaints across Brooklyn(Hexbin Plot)'
plt.title(title)
plt.colorbar()
plt.savefig(os.path.join(graph_dir,title+'.png'),bbox_inches='tight')
plt.show()
plt.close()
```


Concentration of Complaints across Brooklyn(Scatter Plot)
Latitude
40.725
40.700
40.675
40.650
40.625
40.600
40.575
−74.050
−74.025
−74.000
−73.975
−73.950
−73.925
−73.900
−73.875
−73.850
Longitude

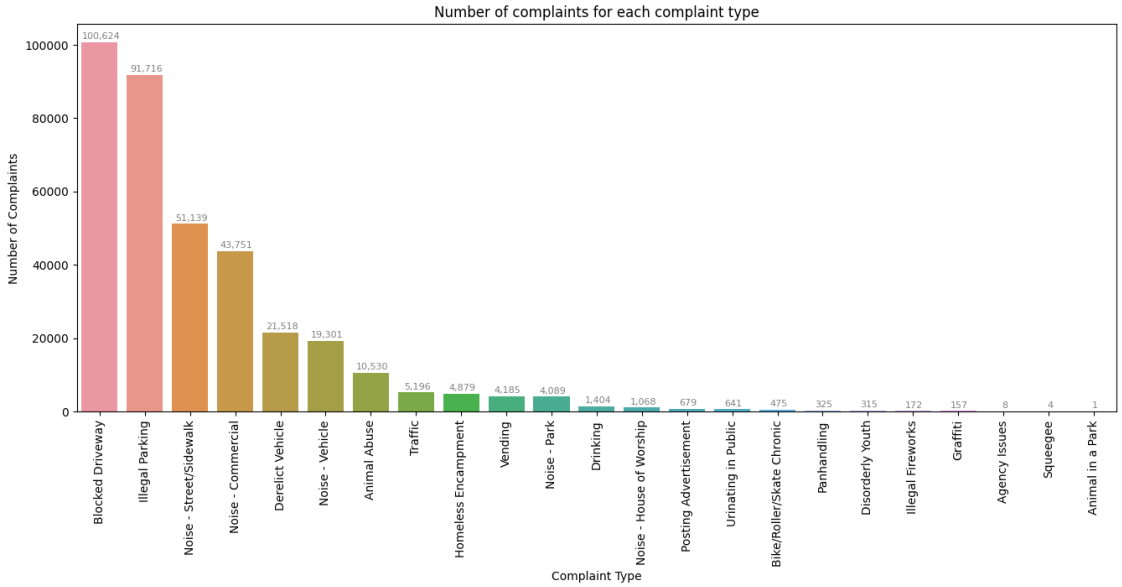


[41]:
```
#3. Find major types of complaints:
```

[42]:
```
#3.1 Plot a bar graph to show the types of complaints
```

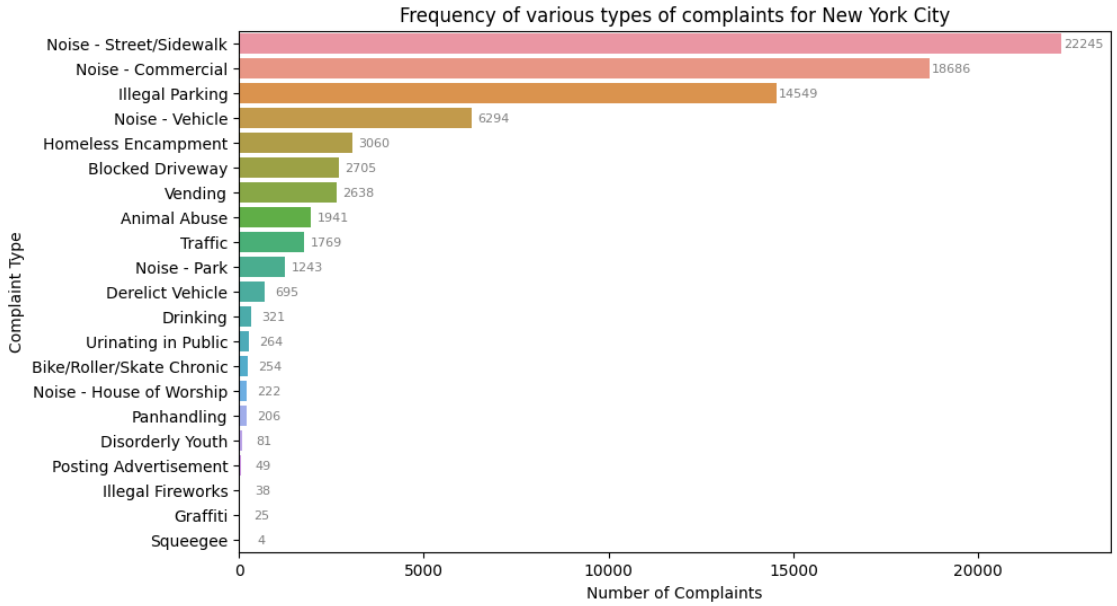
[43]:
```
complaint_counts = dataset.groupby('Complaint Type')['Unique Key'].count().
 ↪sort_values(ascending=False)
plt.figure(figsize=(16, 6))
ax = sns.barplot(x=complaint_counts.index, y=complaint_counts.values)
plt.xticks(rotation=90)
plt.xlabel('Complaint Type')
plt.ylabel('Number of Complaints')
title = 'Number of complaints for each complaint type'
plt.title(title)
for p in ax.patches:
    ax.annotate(f'{int(p.get_height()):,}', (p.get_x() + p.get_width() / 2. +0.
 ↪03, p.get_height()),
                ha='center', va='center', fontsize=8, color='gray', xytext=(0,␣
 ↪5),
                textcoords='offset points')
```

```
plt.savefig(os.path.join(graph_dir, title + '.png'), bbox_inches='tight')
plt.show()
plt.close()
```



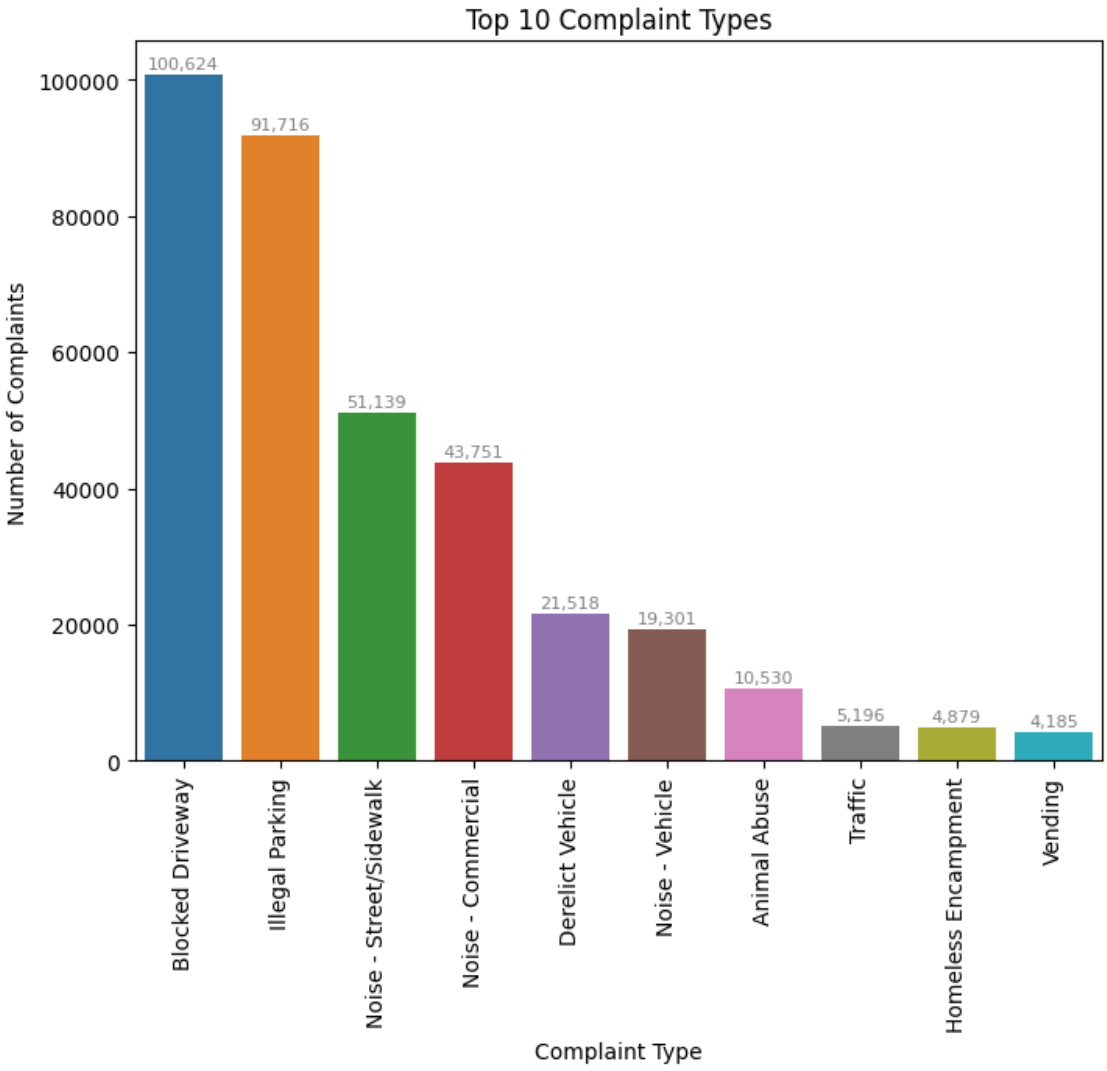
[44]:
```
#3.2 Check the frequency of various types of complaints for New York City
```

[45]:
```
nyc_data = dataset[dataset['City'] == 'NEW YORK']

plt.figure(figsize=(10, 6))
complaint_freq = nyc_data['Complaint Type'].value_counts()
ax = sns.barplot(y=complaint_freq.index, x=complaint_freq.values)
#plt.yticks(rotation=90)
ax = plt.gca()
ax.margins(0.06, None)
plt.ylabel('Complaint Type')
plt.xlabel('Number of Complaints')
title = 'Frequency of various types of complaints for New York City'
plt.title(title)
for p in ax.patches:
    ax.annotate(f'{p.get_width():.0f}', (p.get_width() + 600, p.get_y() + p.
 ↪get_height() / 2. +0.35), ha='center', va='center', fontsize=8,␣
 ↪color='gray', xytext=(0, 5), textcoords='offset points', rotation=0)
plt.savefig(os.path.join(graph_dir, title + '.png'), bbox_inches='tight')
plt.show()
```



[46]:
```
#3.3 Find the top 10 complaint types
```

[47]:
```
top_10_complaints = dataset['Complaint Type'].value_counts().nlargest(10)

plt.figure(figsize=(8, 6))
ax = sns.barplot(x=top_10_complaints.index, y=top_10_complaints.values)
plt.xticks(rotation=90)
plt.xlabel('Complaint Type')
plt.ylabel('Number of Complaints')
title = 'Top 10 Complaint Types'
plt.title(title)
for p in ax.patches:
    ax.annotate(f'{p.get_height():,.0f}', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()+6),
                ha='center', va='center', fontsize=8, color='gray', xytext=(0,␣
 ↪5),
                textcoords='offset points')
plt.savefig(os.path.join(graph_dir,title+'.png'), bbox_inches='tight')
plt.show()
plt.close()
```

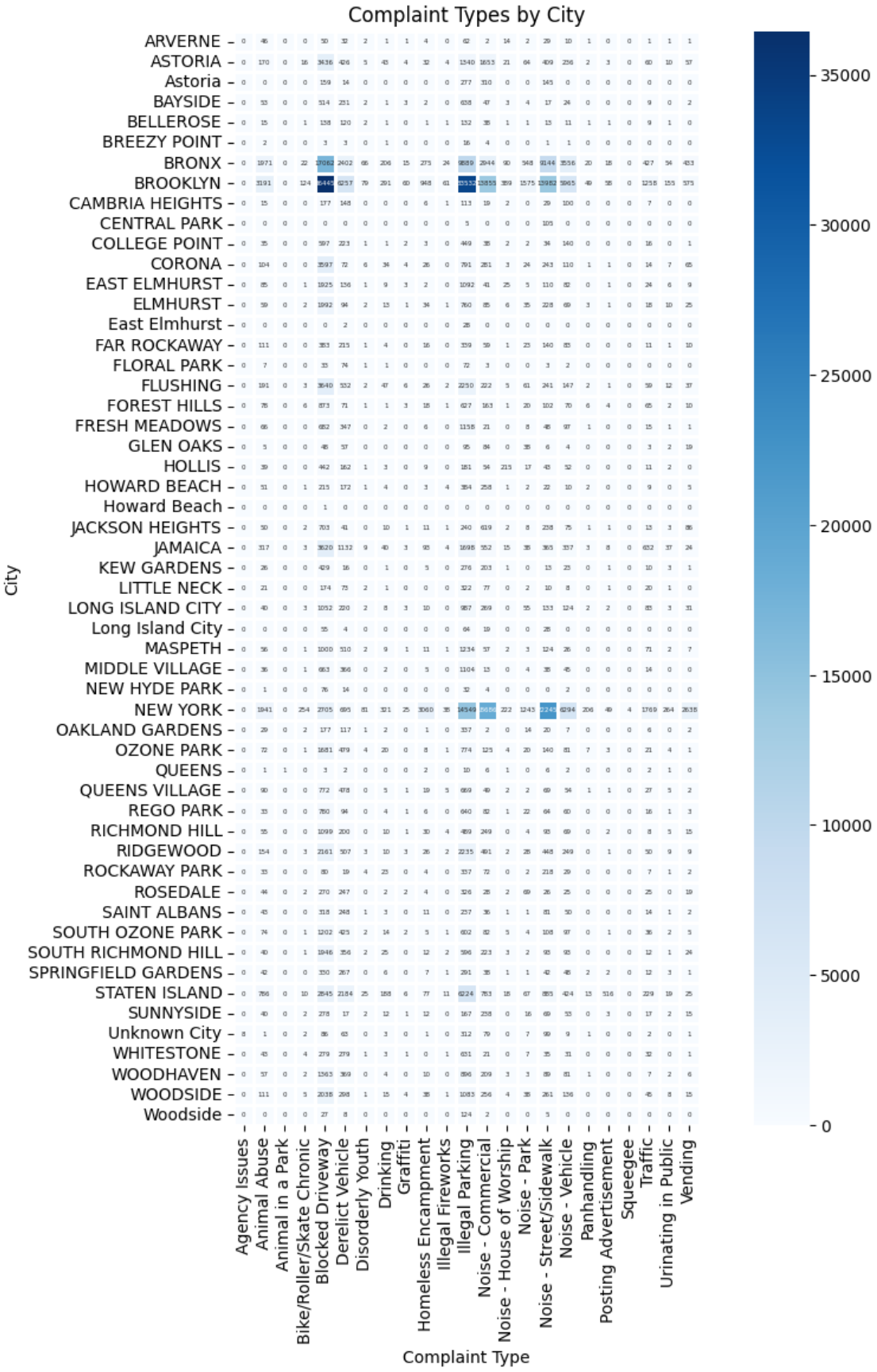


[48]:
```
#3.4 Display the various types of complaints in each city
```

[49]:
```
top_complaints_by_city = dataset.groupby(['City', 'Complaint Type'])['Unique␣
 ↪Key'].nunique().reset_index()
top_complaints_by_city = top_complaints_by_city.pivot('City', 'Complaint Type',␣
 ↪'Unique Key').fillna(0)

plt.figure(figsize=(12, 12))
sns.heatmap(top_complaints_by_city, cmap='Blues', annot=True, fmt='.0f',␣
 ↪square=True, annot_kws={"size": 4}, linewidths=1)
plt.xlabel('Complaint Type')
plt.ylabel('City')
title = 'Complaint Types by City'
```

```
plt.title(title)
plt.savefig(os.path.join(graph_dir,title+'.png'), bbox_inches='tight')
plt.show()
plt.close()
```

```
C:\Users\kanai\AppData\Local\Temp\ipykernel_19668\3667492604.py:2:
FutureWarning: In a future version of pandas all arguments of DataFrame.pivot
will be keyword-only.
  top_complaints_by_city = top_complaints_by_city.pivot('City', 'Complaint
Type', 'Unique Key').fillna(0)
```


Complaint Types by City
City
ARVERNE
ASTORIA
Astoria
BAYSIDE
BELLEROSE
BREEZY POINT
BRONX
BROOKLYN
CAMBRIA HEIGHTS
CENTRAL PARK
COLLEGE POINT
CORONA
EAST ELMHURST
ELMHURST
East Elmhurst
FAR ROCKAWAY
FLORAL PARK
FLUSHING
FOREST HILLS
FRESH MEADOWS
GLEN OAKS
HOLLIS
HOWARD BEACH
Howard Beach
JACKSON HEIGHTS
JAMAICA
KEW GARDENS
LITTLE NECK
LONG ISLAND CITY
Long Island City
MASPETH
MIDDLE VILLAGE
NEW HYDE PARK
NEW YORK
OAKLAND GARDENS
OZONE PARK
QUEENS
QUEENS VILLAGE
REGO PARK
RICHMOND HILL
RIDGEWOOD
ROCKAWAY PARK
ROSEDALE
SAINT ALBANS
SOUTH OZONE PARK
SOUTH RICHMOND HILL
SPRINGFIELD GARDENS
STATEN ISLAND
SUNNYSIDE
Unknown City
WHITESTONE
WOODHAVEN
WOODSIDE
Woodside
Agency Issues
Animal Abuse
Animal in a Park
Bike/Roller/Skate Chronic
Blocked Driveway
Derelict Vehicle
Disorderly Youth
Drinking
Graffiti
Homeless Encampment
Illegal Fireworks
Illegal Parking
Noise - Commercial
Noise - House of Worship
Noise - Park
Noise - Street/Sidewalk
Noise - Vehicle
Panhandling
Posting Advertisement
Squeegee
Traffic
Urinating in Public
Vending
Complaint Type
35000
30000
25000
20000
15000
10000
5000
0

[50]:
```
#3.5 Create a DataFrame, df_new, which contains cities as columns and complaint␣
↪types in rows
```

[51]:
```
df_new = pd.pivot_table(data=dataset, values='Unique Key', index='Complaint␣
↪Type', columns='City', aggfunc='count', fill_value=0)
df_new.info()
df_new.shape
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 23 entries, Agency Issues to Vending
Data columns (total 54 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ARVERNE              23 non-null     int64
 1   ASTORIA              23 non-null     int64
 2   Astoria              23 non-null     int64
 3   BAYSIDE              23 non-null     int64
 4   BELLEROSE            23 non-null     int64
 5   BREEZY POINT         23 non-null     int64
 6   BRONX                23 non-null     int64
 7   BROOKLYN             23 non-null     int64
 8   CAMBRIA HEIGHTS      23 non-null     int64
 9   CENTRAL PARK         23 non-null     int64
 10  COLLEGE POINT        23 non-null     int64
 11  CORONA               23 non-null     int64
 12  EAST ELMHURST        23 non-null     int64
 13  ELMHURST             23 non-null     int64
 14  East Elmhurst        23 non-null     int64
 15  FAR ROCKAWAY         23 non-null     int64
 16  FLORAL PARK          23 non-null     int64
 17  FLUSHING             23 non-null     int64
 18  FOREST HILLS         23 non-null     int64
 19  FRESH MEADOWS        23 non-null     int64
 20  GLEN OAKS            23 non-null     int64
 21  HOLLIS               23 non-null     int64
 22  HOWARD BEACH         23 non-null     int64
 23  Howard Beach         23 non-null     int64
 24  JACKSON HEIGHTS      23 non-null     int64
 25  JAMAICA              23 non-null     int64
 26  KEW GARDENS          23 non-null     int64
 27  LITTLE NECK          23 non-null     int64
 28  LONG ISLAND CITY     23 non-null     int64
 29  Long Island City     23 non-null     int64
 30  MASPETH              23 non-null     int64
 31  MIDDLE VILLAGE       23 non-null     int64
```

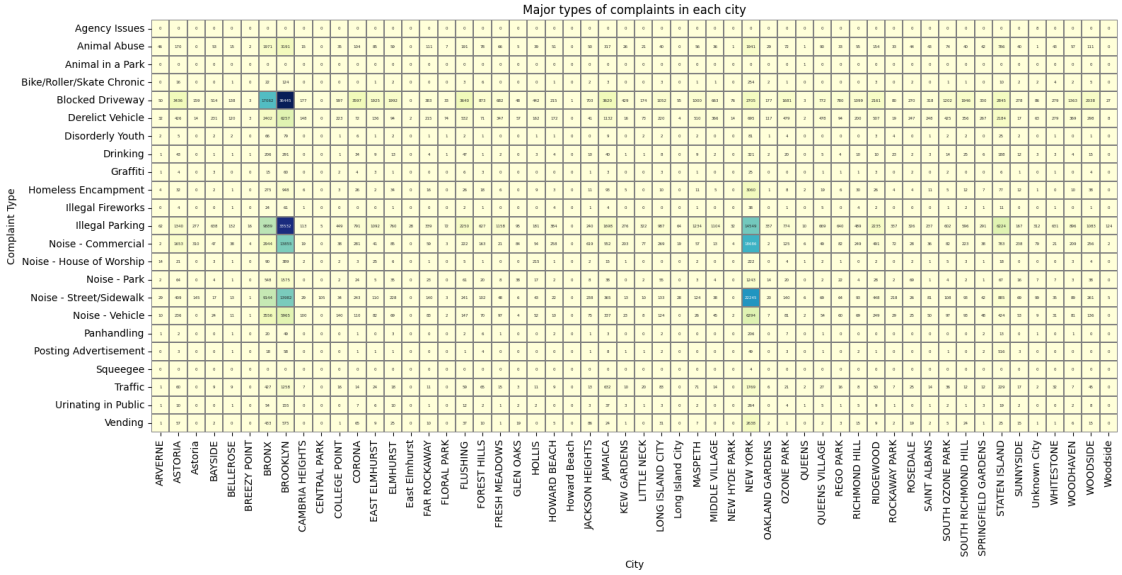
```
 32  NEW HYDE PARK         23 non-null     int64
 33  NEW YORK              23 non-null     int64
 34  OAKLAND GARDENS       23 non-null     int64
 35  OZONE PARK            23 non-null     int64
 36  QUEENS                23 non-null     int64
 37  QUEENS VILLAGE        23 non-null     int64
 38  REGO PARK             23 non-null     int64
 39  RICHMOND HILL         23 non-null     int64
 40  RIDGEWOOD             23 non-null     int64
 41  ROCKAWAY PARK         23 non-null     int64
 42  ROSEDALE              23 non-null     int64
 43  SAINT ALBANS          23 non-null     int64
 44  SOUTH OZONE PARK      23 non-null     int64
 45  SOUTH RICHMOND HILL   23 non-null     int64
 46  SPRINGFIELD GARDENS   23 non-null     int64
 47  STATEN ISLAND         23 non-null     int64
 48  SUNNYSIDE             23 non-null     int64
 49  Unknown City          23 non-null     int64
 50  WHITESTONE            23 non-null     int64
 51  WOODHAVEN             23 non-null     int64
 52  WOODSIDE              23 non-null     int64
 53  Woodside              23 non-null     int64
dtypes: int64(54)
memory usage: 9.9+ KB
```

[51]:
```
(23, 54)
```

[52]:
```
#4. Visualize the major types of complaints in each city
```

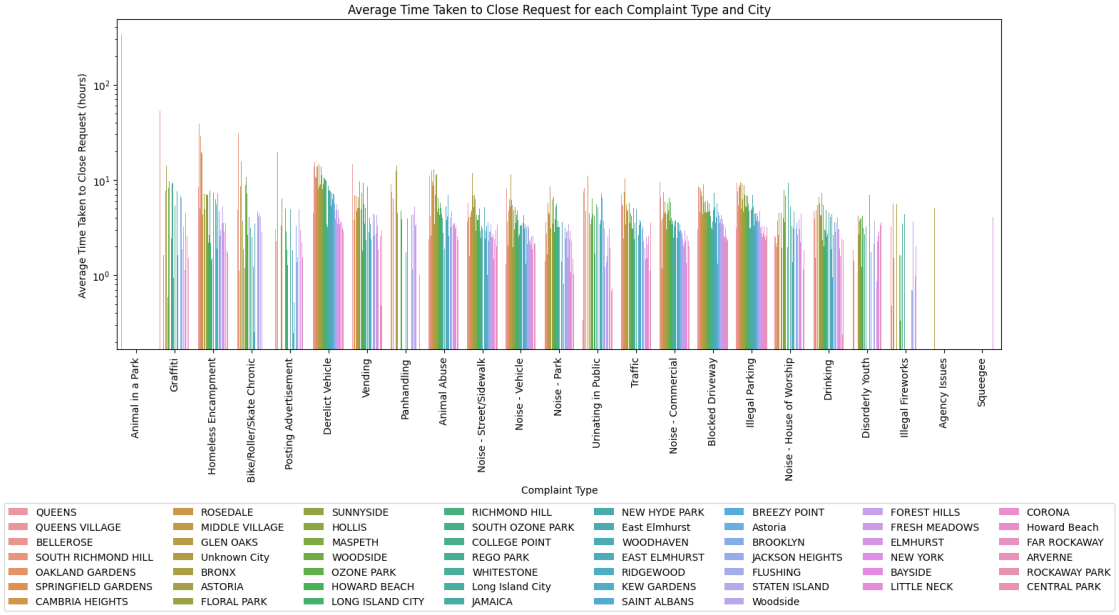
[53]:
```
#4.1 Draw another chart that shows the types of complaints in each city in a␣
 ↪single chart, where different colors show the different types of complaints
```

[54]:
```
plt.figure(figsize=(16, 10))
sns.heatmap(df_new, cmap='YlGnBu', annot=True, fmt='d', linewidths=0.2,␣
 ↪linecolor='gray', square=True, annot_kws={"size": 4}, cbar=False)
plt.xlabel('City')
plt.ylabel('Complaint Type')
title = 'Major types of complaints in each city'
plt.title(title)
plt.tight_layout()
plt.savefig(os.path.join(graph_dir, title + '.png'), bbox_inches='tight')
plt.show()
plt.close()
```



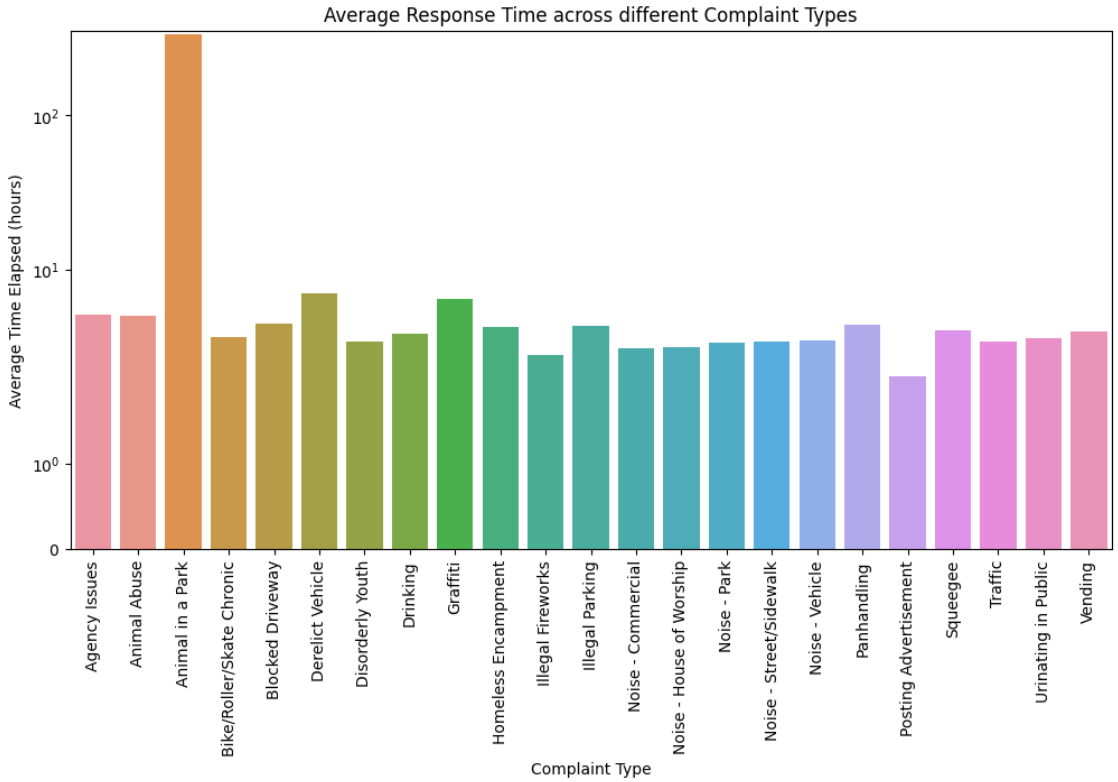
[55]:
```
#4.2 Sort the complaint types based on the average Request_Closing_Time␣
↪grouping them for different locations
```

[56]:
```
dataset['Time Elapsed (hours)'] = dataset['Time Elapsed'] / 3600

df_grouped = dataset.groupby(['Complaint Type', 'City']).agg({'Time Elapsed␣
↪(hours)': 'mean'}).reset_index()

df_sorted = df_grouped.sort_values(by='Time Elapsed (hours)', ascending=False)

# plot the data as a grouped bar chart
plt.figure(figsize=(16, 6))
ax = sns.barplot(x='Complaint Type', y='Time Elapsed (hours)', hue='City',␣
↪data=df_sorted)
plt.xticks(rotation=90)
plt.yscale('log')
plt.xlabel('Complaint Type')
plt.ylabel('Average Time Taken to Close Request (hours)')
title = 'Average Time Taken to Close Request for each Complaint Type and City'
plt.title(title)
legend = ax.legend(loc='upper center', bbox_to_anchor=(0.5, -0.45), ncol=8)
legend.get_frame().set_facecolor('white')
plt.savefig(os.path.join(graph_dir, title + '.png'), bbox_inches='tight',␣
↪bbox_extra_artists=[legend])
plt.show()
plt.close()
```



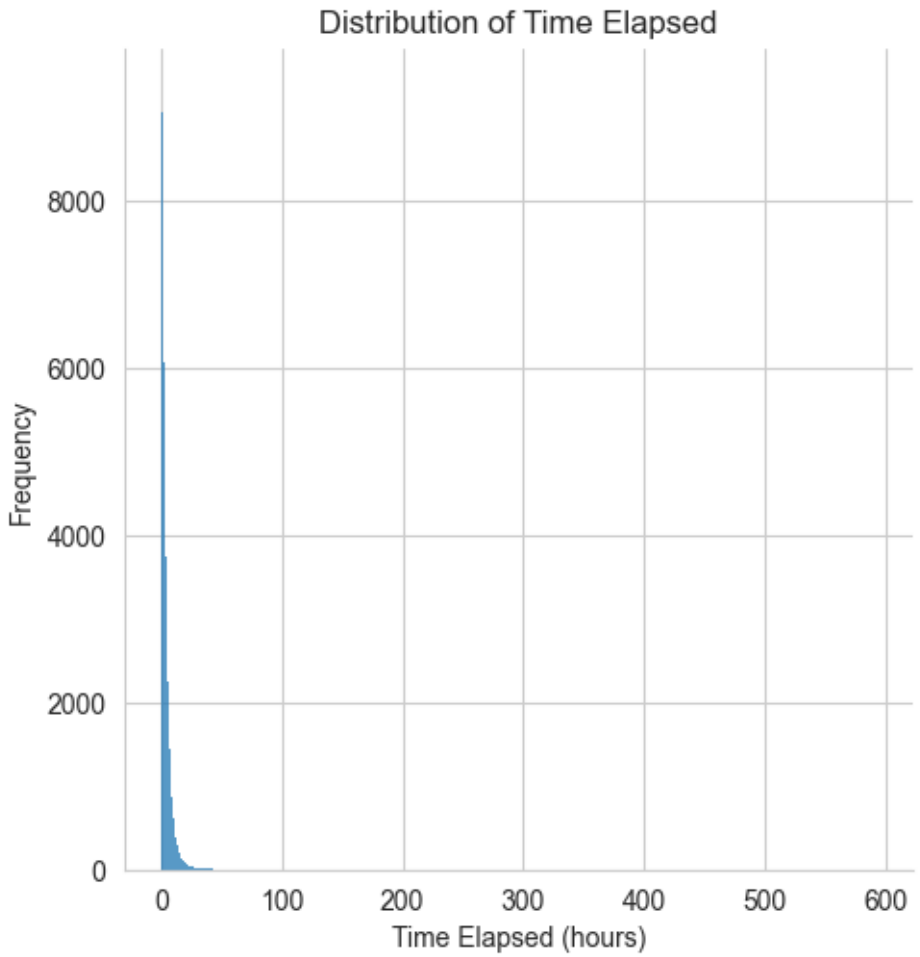
[57]:
```
#5. See whether the average response time across different complaint types is␣
↪similar (overall)
```

[58]:
```
# group data by complaint type and calculate the average response time
df_grouped = dataset.groupby(['Complaint Type']).agg({'Time Elapsed (hours)':␣
↪'mean'}).reset_index()

# plot the data as a boxplot
plt.figure(figsize=(12, 6))
ax = sns.barplot(x='Complaint Type', y='Time Elapsed (hours)', data=df_grouped)
plt.xticks(rotation=90)
plt.yscale('symlog')
plt.xlabel('Complaint Type')
plt.ylabel('Average Time Elapsed (hours)')
title = 'Average Response Time across different Complaint Types'
plt.title(title)
plt.savefig(os.path.join(graph_dir, title + '.png'), bbox_inches='tight')
plt.show()
plt.close()
```



[59]:
```
#5.1 Visualize the average of Request_Closing_Time
```

[60]:
```
sns.set_style('whitegrid')
sns.displot(dataset['Time Elapsed (hours)'], kde=False)

plt.xlabel('Time Elapsed (hours)')
plt.ylabel('Frequency')
title = 'Distribution of Time Elapsed'
plt.title(title)
plt.savefig(os.path.join(graph_dir, title + '.png'), bbox_inches='tight')
plt.show()
plt.close()
```



[61]:
```
df = dataset[['Complaint Type', 'Location Type', 'Incident Zip', 'Borough',␣
↪'Time Elapsed (hours)']].copy()

df = pd.get_dummies(df, columns=['Complaint Type', 'Location Type', 'Borough'])

df = sm.add_constant(df)

df[~np.isfinite(df)] = 1e6

model = sm.OLS(df['Time Elapsed (hours)'], df.drop(['Time Elapsed (hours)'],␣
↪axis=1)).fit()

print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     Time Elapsed (hours)   R-squared:                       0.052
```

```
Model:                            OLS   Adj. R-squared:                  0.052
Method:                 Least Squares   F-statistic:                     462.0
Date:                Sun, 14 May 2023   Prob (F-statistic):               0.00
Time:                        02:08:15   Log-Likelihood:            -1.1447e+06
No. Observations:              362177   AIC:                         2.290e+06
Df Residuals:                  362133   BIC:                         2.290e+06
Df Model:                          43
Covariance Type:            nonrobust
================================================================================
============================
                                                coef    std err          t
P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
----------------------------
const                                        10.2040      0.463     22.055
0.000       9.297      11.111
Incident Zip                             -2.082e-07   8.95e-07     -0.233
0.816   -1.96e-06    1.55e-06
Complaint Type_Agency Issues               -11.2530      2.009     -5.601
0.000     -15.191      -7.315
Complaint Type_Animal Abuse                 -5.8213      0.285    -20.450
0.000      -6.379      -5.263
Complaint Type_Animal in a Park            160.2548      2.806     57.110
0.000     154.755     165.755
Complaint Type_Bike/Roller/Skate Chronic    -6.6148      0.371    -17.834
0.000      -7.342      -5.888
Complaint Type_Blocked Driveway             -6.4426      0.275    -23.407
0.000      -6.982      -5.903
Complaint Type_Derelict Vehicle             -3.7912      0.277    -13.677
0.000      -4.334      -3.248
Complaint Type_Disorderly Youth             -7.3586      0.412    -17.866
0.000      -8.166      -6.551
Complaint Type_Drinking                     -6.8007      0.312    -21.772
0.000      -7.413      -6.188
Complaint Type_Graffiti                     -4.2628      0.517     -8.252
0.000      -5.275      -3.250
Complaint Type_Homeless Encampment          -5.8717      0.284    -20.651
0.000      -6.429      -5.314
Complaint Type_Illegal Fireworks            -7.8364      0.498    -15.735
0.000      -8.813      -6.860
Complaint Type_Illegal Parking              -6.3329      0.275    -23.015
0.000      -6.872      -5.794
Complaint Type_Noise - Commercial           -7.2483      0.296    -24.485
0.000      -7.828      -6.668
Complaint Type_Noise - House of Worship     -9.8428      3.894     -2.528
0.011     -17.475      -2.211
Complaint Type_Noise - Park                 -6.5193      0.350    -18.642
0.000      -7.205      -5.834
```

```
Complaint Type_Noise - Street/Sidewalk      -7.1474      0.276    -25.927
0.000      -7.688      -6.607
Complaint Type_Noise - Vehicle              -7.1657      0.277    -25.831
0.000      -7.709      -6.622
Complaint Type_Panhandling                  -5.7779      0.408    -14.159
0.000      -6.578      -4.978
Complaint Type_Posting Advertisement        -8.1005      0.346    -23.394
0.000      -8.779      -7.422
Complaint Type_Squeegee                     -5.7338      2.743     -2.091
0.037     -11.110      -0.358
Complaint Type_Traffic                      -7.0595      0.285    -24.757
0.000      -7.618      -6.501
Complaint Type_Urinating in Public          -6.7892      0.349    -19.470
0.000      -7.473      -6.106
Complaint Type_Vending                      -6.2805      0.286    -21.945
0.000      -6.841      -5.720
Location Type_Bridge                        -0.5779      4.094     -0.141
0.888      -8.603       7.447
Location Type_Club/Bar/Restaurant           -0.3063      0.537     -0.571
0.568      -1.358       0.746
Location Type_Commercial                    -0.5788      0.762     -0.759
0.448      -2.073       0.915
Location Type_Highway                       -0.9361      0.632     -1.481
0.139      -2.175       0.303
Location Type_House and Store               -0.4676      0.642     -0.728
0.467      -1.726       0.791
Location Type_House of Worship               2.1172      4.040      0.524
0.600      -5.801      10.035
Location Type_Park                         160.2548      2.806     57.110
0.000     154.755     165.755
Location Type_Park/Playground               -0.9370      0.563     -1.666
0.096      -2.040       0.166
Location Type_Parking Lot                   -0.3830      0.689     -0.556
0.578      -1.732       0.966
Location Type_Residential Building          -0.8369      0.584     -1.434
0.152      -1.981       0.307
Location Type_Residential Building/House    -0.0974      0.530     -0.184
0.854      -1.135       0.941
Location Type_Roadway Tunnel                 0.8573      1.016      0.844
0.399      -1.134       2.849
Location Type_Store/Commercial              -0.3257      0.535     -0.609
0.543      -1.374       0.723
Location Type_Street/Sidewalk               -0.2715      0.521     -0.521
0.602      -1.293       0.750
Location Type_Subway Station                -1.9994      1.047     -1.909
0.056      -4.052       0.053
Location Type_Vacant Lot                    -0.4573      0.785     -0.583
0.560      -1.996       1.081
```

```
Borough_BRONX                                 2.4685      0.152     16.226
0.000       2.170       2.767
Borough_BROOKLYN                              0.5025      0.151      3.336
0.001       0.207       0.798
Borough_MANHATTAN                            -0.1520      0.152     -1.002
0.316      -0.449       0.145
Borough_QUEENS                                1.0162      0.151      6.746
0.000       0.721       1.311
Borough_STATEN ISLAND                         0.0313      0.156      0.201
0.840      -0.274       0.336
Borough_Unspecified                           6.3375      0.786      8.061
0.000       4.797       7.878
==============================================================================
Omnibus:                   749365.959   Durbin-Watson:                   1.930
Prob(Omnibus):                  0.000   Jarque-Bera (JB):      20832403369.444
Skew:                          16.762   Prob(JB):                         0.00
Kurtosis:                    1177.459   Cond. No.                     1.04e+16
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The smallest eigenvalue is 6.58e-18. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

[62]:
```
#7. Perform a Kruskal-Wallis H test
```

[63]:
```
df_subset = dataset[['Complaint Type', 'Time Elapsed']]

data = {}
for comp_type in df_subset['Complaint Type'].unique():
    data[comp_type] = df_subset[df_subset['Complaint Type'] == comp_type]['Time␣
 ↪Elapsed'].values

test_stat, p_value = kruskal(*data.values())

print('Kruskal-Wallis H test:')
print(f'Test statistic: {test_stat:.4f}')
print(f'p-value: {p_value:.4f}')
```

```
Kruskal-Wallis H test:
Test statistic: 11988.2694
p-value: 0.0000
```

[64]:
```
#7.1 Fail to reject H0: All sample distributions are equal
#7.2 Reject H0: One or more sample distributions are not equal
```

[65]:
```
alpha = 0.05
if p_value > alpha:
    print("Fail to reject the null hypothesis: All sample distributions are␣
 ↪equal")
else:
    print("Reject the null hypothesis: At least one sample distribution is␣
 ↪different")
```

```
Reject the null hypothesis: At least one sample distribution is different
```

[ ]: