Customer Service Requests Analysis

DESCRIPTION

Background of Problem Statement :

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.

Problem Objective :

Perform a service request data analysis of New York City 311 calls. You will focus on the data wrangling techniques to understand the pattern in the data and also visualize the major complaint types.

Analysis Tasks to be performed:

(Perform a service request data analysis of New York City 311 calls)

1) Import a 311 NYC service request.
2) Read or convert the columns 'Created Date' and Closed Date' to datetime datatype and create a new column 'Request_Closing_Time' as the time elapsed between request creation and request closing. (Hint: Explore the package/module datetime)
3) Provide major insights/patterns that you can offer in a visual format (graphs or tables); at least 4 major conclusions that you can come up with after generic data mining.
4) Order the complaint types based on the average 'Request_Closing_Time', grouping them for different locations.
5) Perform a statistical test for the following:
   Please note: For the below statements you need to state the Null and Alternate and then provide a statistical test to accept or reject the Null Hypothesis along with the corresponding 'p-value'.

   - Whether the average response time across complaint types is similar or not (overall)
   - Are the type of complaint or service requested and location related?

Dataset Description :

- Unique Key (Plain text) - Unique identifier for the complaints
- Created Date (Date and Time) - The date and time on which the complaint is raised
- Closed Date (Date and Time) - The date and time on which the complaint is closed
- Agency (Plain text) - Agency code
- Agency Name (Plain text) - Name of the agency
- Complaint Type (Plain text) - Type of the complaint
- Descriptor (Plain text) - Complaint type label (Heating - Heat, Traffic Signal Condition - Controller)
- Location Type (Plain text) - Type of the location (Residential, Restaurant, Bakery, etc)
- Incident Zip (Plain text) - Zip code for the location
- Incident Address (Plain text) - Address of the location
- Street Name (Plain text) - Name of the street
- Cross Street 1 (Plain text) - Detail of cross street
- Cross Street 2 (Plain text) - Detail of another cross street
- Intersection Street 1 (Plain text) - Detail of intersection street if any
- Intersection Street 2 (Plain text) - Detail of another intersection street if any
- Address Type (Plain text) - Categorical (Address or Intersection)
- City (Plain text) - City for the location
- Landmark (Plain text) - Empty field
- Facility Type (Plain text) - N/A
- Status (Plain text) - Categorical (Closed or Pending)
- Due Date (Date and Time) - Date and time for the pending complaints
- Resolution Action Updated Date (Date and Time) - Date and time when the resolution was provided
- Community Board (Plain text) - Categorical field (specifies the community board with its code)
- Borough (Plain text) - Categorical field (specifies the community board)
- X Coordinate (State Plane) (Number)
- Y Coordinate (State Plane) (Number)
- Park Facility Name (Plain text) - Unspecified
- Park Borough (Plain text) - Categorical (Unspecified, Queens, Brooklyn etc)
- School Name (Plain text) - Unspecified
- School Number (Plain text) - Unspecified
- School Region (Plain text) - Unspecified
- School Code (Plain text) - Unspecified
- School Phone Number (Plain text) - Unspecified
- School Address (Plain text) - Unspecified
- School City (Plain text) - Unspecified
- School State (Plain text) - Unspecified
- School Zip (Plain text) - Unspecified
- School Not Found (Plain text) - Empty Field
- School or Citywide Complaint (Plain text) - Empty Field
- Vehicle Type (Plain text) - Empty Field
- Taxi Company Borough (Plain text) - Empty Field
- Taxi Pick Up Location (Plain text) - Empty Field
- Bridge Highway Name (Plain text) - Empty Field
- Bridge Highway Direction (Plain text) - Empty Field
- Road Ramp (Plain text) - Empty Field
- Bridge Highway Segment (Plain text) - Empty Field
- Garage Lot Name (Plain text) - Empty Field
- Ferry Direction (Plain text) - Empty Field
- Ferry Terminal Name (Plain text) - Empty Field
- Latitude (Number) - Latitude of the location
- Longitude (Number) - Longitude of the location
- Location (Coordinates) - Coordinates (Latitude, Longitude)

In [1]:
```
# Import necessary packages
import pandas as pd
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
```

In [2]:
```
# Using set_option function to display the needed no. of rows and columns
pd.set_option('display.max_columns',30)
pd.set_option('display.max_rows',800)
#To ignore warnings
warnings.simplefilter('ignore')
```

Task 1

- Import a 311 NYC service request

In [3]:
```
# Load the dataset using pandas
df = pd.read_csv('311_Service_Requests_from_2010_to_Present.csv',low_memory=False)
```

In [4]:
```
# Head method to view first 3 rows of the dataset
df.head(3)
```

Out[4]:

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	Incident Zip	Incident Address	Street Name	S
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	VERMILYEA AVENUE	ACA ST
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	23 AVENUE	ST
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	VALENTINE AVENUE	EAS ST

3 rows × 53 columns

In [5]:
```
# Understand the dataset
df.shape
```

Out[5]: (300698, 53)

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300698 entries, 0 to 300697
Data columns (total 53 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   Unique Key                      300698 non-null  int64
 1   Created Date                    300698 non-null  object
 2   Closed Date                     298534 non-null  object
 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  float64
 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  object
 21  Resolution Description          300698 non-null  object
 22  Resolution Action Updated Date  298511 non-null  object
 23  Community Board                 300698 non-null  object
 24  Borough                         300698 non-null  object
 25  X Coordinate (State Plane)      297158 non-null  float64
 26  Y Coordinate (State Plane)      297158 non-null  float64
 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  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             243 non-null     object
 44  Bridge Highway Direction        243 non-null     object
 45  Road Ramp                       213 non-null     object
 46  Bridge Highway Segment          213 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                        297158 non-null  float64
 51  Longitude                       297158 non-null  float64
 52  Location                        297158 non-null  object
dtypes: float64(10), int64(1), object(42)
memory usage: 121.6+ MB
```

In [7]:
```
# Changing the column names for easy access
df.columns = df.columns.str.replace(" ","_").str.lower()
```

In [8]:
```
# Create new dataset with necessary columns
nyc_dataset = df[['unique_key','created_date','closed_date','agency','agency_name','complaint_type','de
scriptor','location_type',
'incident_zip','incident_address','street_name','cross_street_1','cross_street_2','address_type','cit
y','status','due_date',
'resolution_description','resolution_action_updated_date','community_board','borough','x_coordinate_(s
tate_plane)',
'y_coordinate_(state_plane)','park_borough','latitude','longitude','location']]
```

In [9]:
```
# Check for duplicates and NA values
nyc_dataset.duplicated().sum()
```

Out[9]: 0

In [10]:
```
nyc_dataset.isna().sum()
```

Out[10]:
```
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
incident_address                  44410
street_name                       44410
cross_street_1                    49279
cross_street_2                    49779
address_type                       2815
city                               2614
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_borough                          0
latitude                           3540
longitude                          3540
location                           3540
dtype: int64
```

In [11]:
```
# Drop NA values in necessary columns
nyc_dataset.dropna(subset=['city','latitude'],inplace=True)
```

In [12]:
```
# Write the cleaned dataset back
nyc_dataset.to_csv('Nyc_cleaned.csv',index=False)
```

Task 2

- Read or convert the columns 'Created Date' and Closed Date' to datetime datatype
- create a new column 'Request_Closing_Time' as the time elapsed between request creation and request closing.

In [13]:
```
# Load the cleaned dataset
nyc = pd.read_csv('Nyc_cleaned.csv',parse_dates=['created_date','closed_date','resolution_action_update
d_date'])
nyc.head(3)
```

Out[13]:

	unique_key	created_date	closed_date	agency	agency_name	complaint_type	descriptor	location_type	incident_zip	incident_addre
0	32310363	2015-12-31 23:59:45	2016-01-01 00:55:00	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Music/Party	Street/Sidewalk	10034.0	71 VERMILYI AVENI
1	32309934	2015-12-31 23:59:44	2016-01-01 01:26:00	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewalk	11105.0	27-07 AVENI
2	32309159	2015-12-31 23:59:29	2016-01-01 04:51:00	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewalk	10458.0	2897 VALENTII AVENI

In [14]:
```
# Calculating the response time in hrs and in minutes
nyc['request_closing_time_hrs'] = nyc['closed_date'].values - nyc['created_date'].values
nyc['request_closing_time_mins'] = nyc['request_closing_time_hrs']/np.timedelta64(1,'m')
```

In [15]:
```
nyc.head()
```

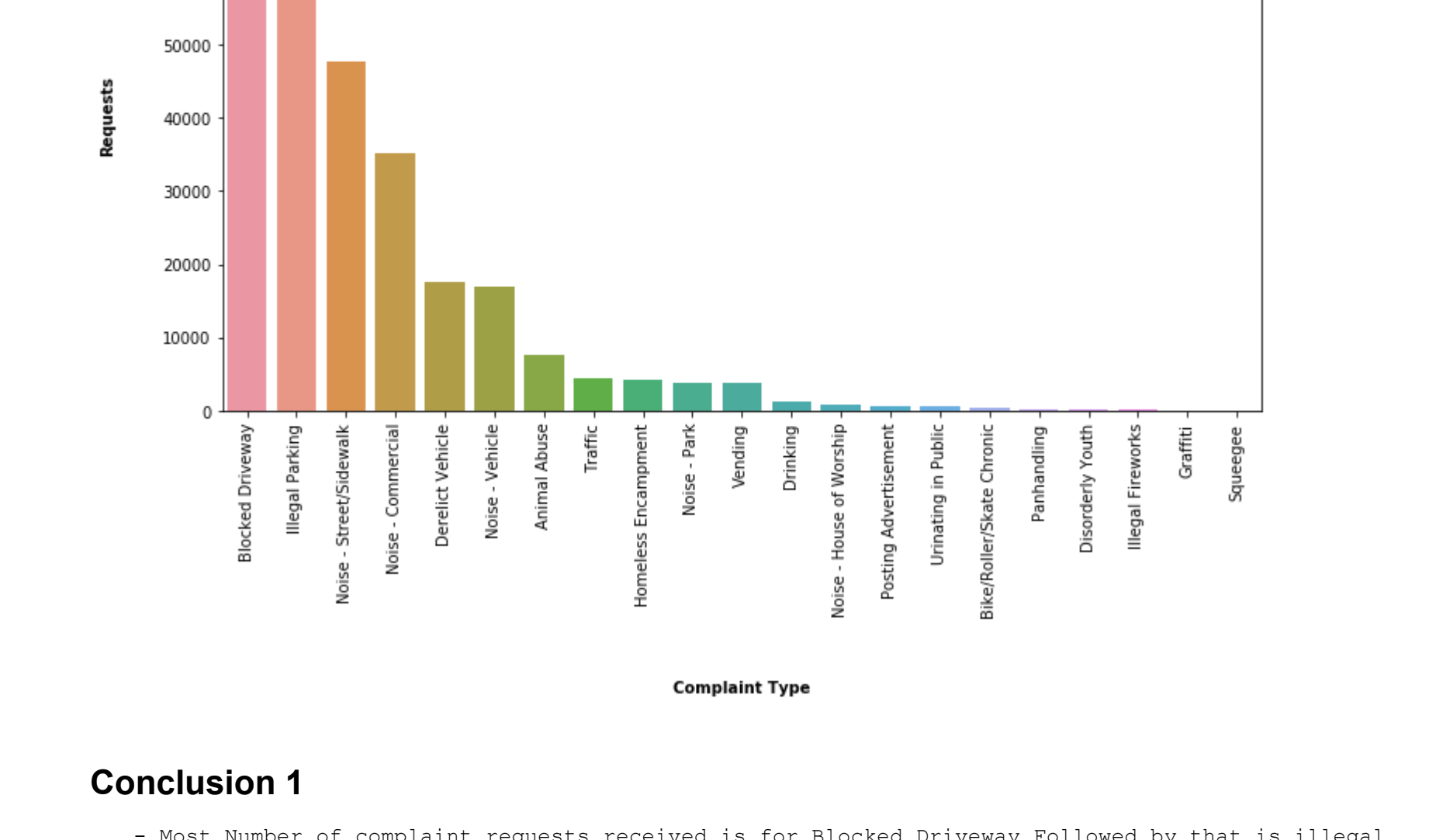
Out[15]:

	unique_key	created_date	closed_date	agency	agency_name	complaint_type	descriptor	location_type	incident_zip	incident_addre
0	32310363	2015-12-31 23:59:45	2016-01-01 00:55:00	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Music/Party	Street/Sidewalk	10034.0	71 VERMILY AVENI
1	32309934	2015-12-31 23:59:44	2016-01-01 01:26:00	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewalk	11105.0	27-07 AVENI
2	32309159	2015-12-31 23:59:29	2016-01-01 04:51:00	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewalk	10458.0	2897 VALENTI AVENI
3	32305098	2015-12-31 23:57:46	2016-01-01 07:43:00	NYPD	New York City Police Department	Illegal Parking	Commercial Overnight Parking	Street/Sidewalk	10461.0	2940 BAISL AVENI
4	32306529	2015-12-31 23:56:58	2016-01-01 03:24:00	NYPD	New York City Police Department	Illegal Parking	Blocked Sidewalk	Street/Sidewalk	11373.0	87-14 57 RO

Task 3

- Visualization
- Atleast 4 main conclusions

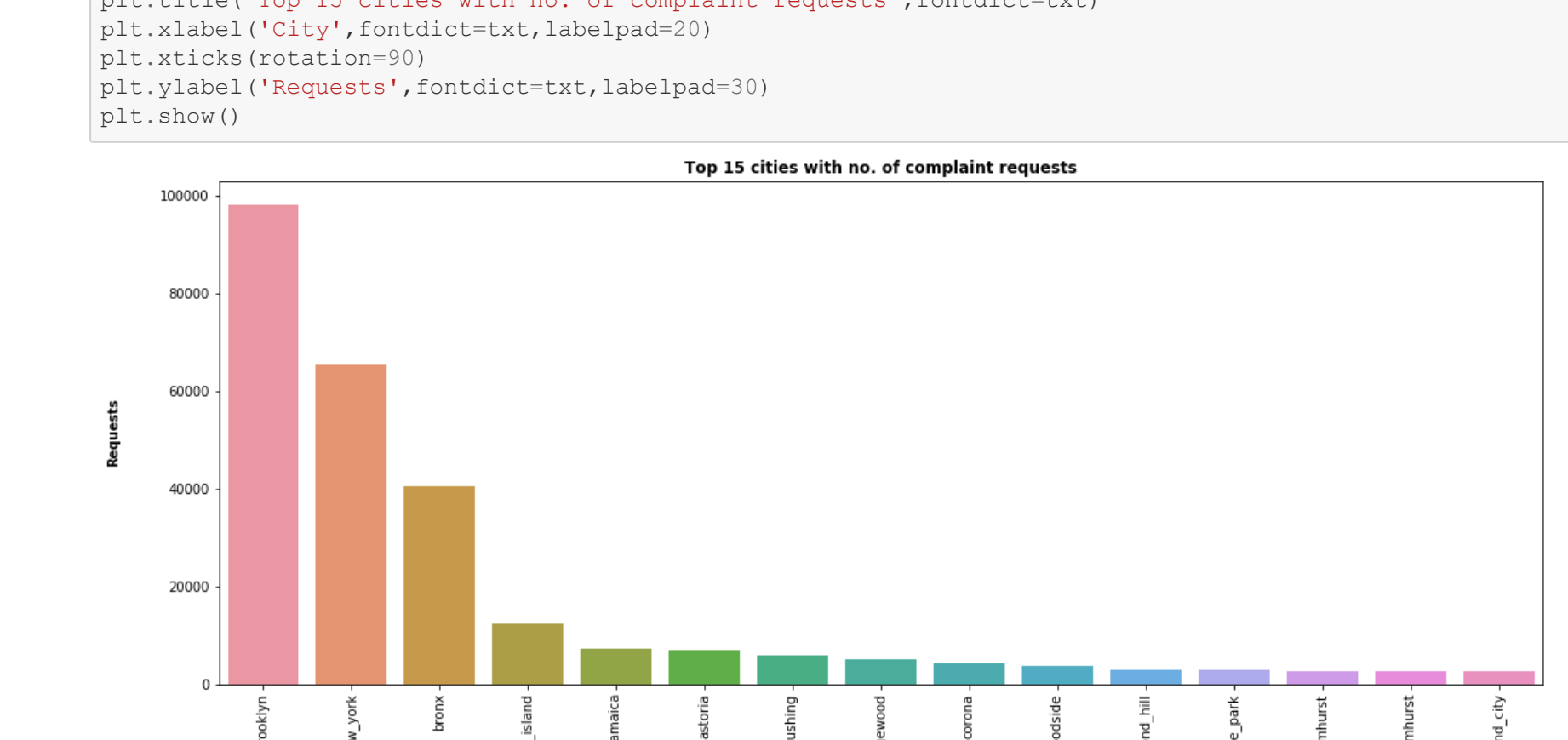
In [16]:
```
# Visualizing complaint types based on the no. of requests
nyc['city'] = nyc['city'].str.lower().str.replace(" ","_")
plt.figure(figsize=(12,7))
sns.countplot(x='complaint_type',data=nyc,order=nyc['complaint_type'].value_counts().index)
txt={'weight':'bold'}
plt.title("Complaint Types and No. of requests per complaint_type",fontdict=txt)
plt.xlabel('Complaint Type',fontdict=txt,labelpad=20)
plt.ylabel('Requests',fontdict=txt,labelpad=20)
plt.xticks(rotation=90)
plt.show()
```



Conclusion 1

- Most Number of complaint requests received is for Blocked Driveway Followed by that is illegal parking.

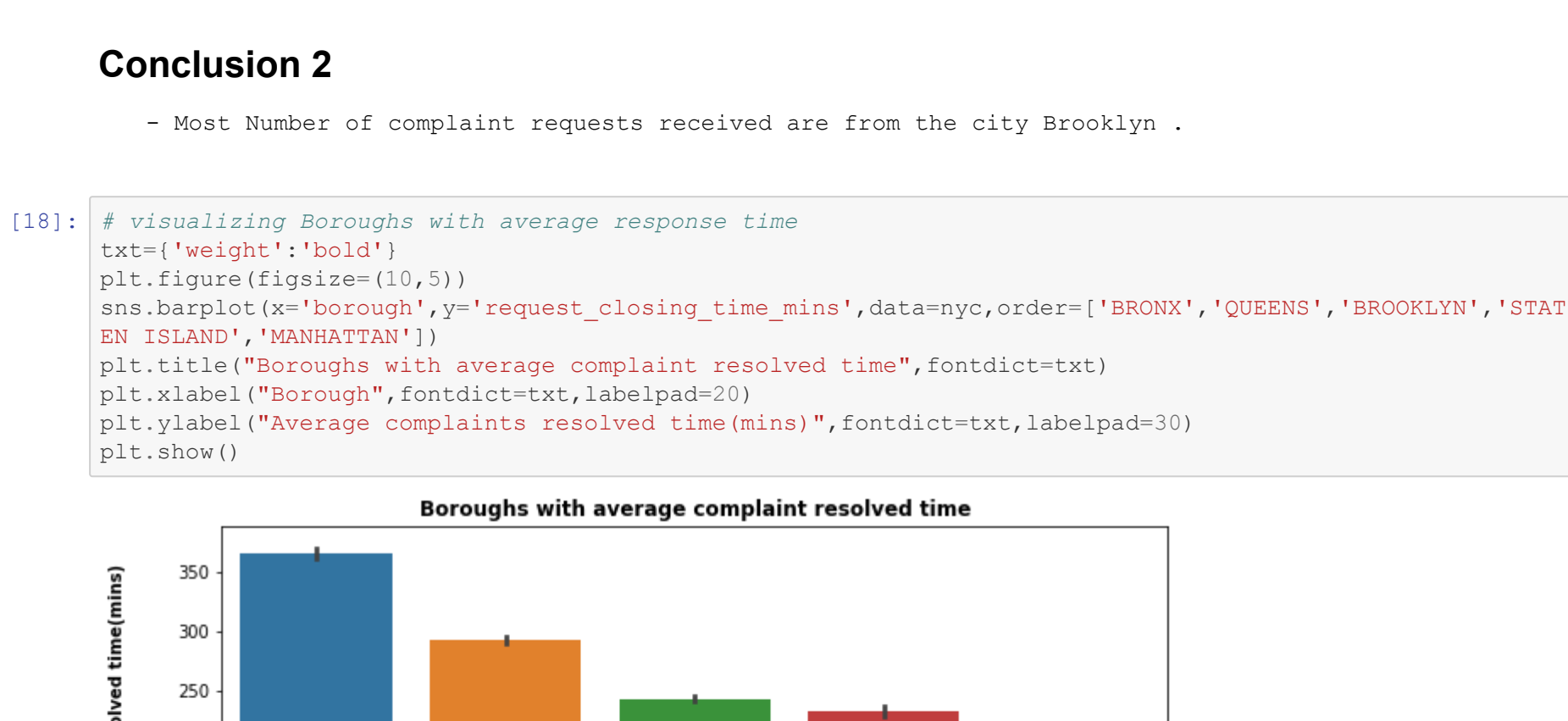
In [17]:
```
# Visualizing no. of requests from top 15 cities
city_top15 = nyc['city'].value_counts().head(15)
top15 = city_top15.index
txt={'weight':'bold'}
plt.figure(figsize=(13,7))
sns.countplot(x='city',data=nyc[nyc.city.isin(top15)],order=top15)
plt.title("Top 15 cities with no. of complaint requests",fontdict=txt)
plt.xlabel('City',fontdict=txt,labelpad=20)
plt.xticks(rotation=90)
plt.ylabel('Requests',fontdict=txt,labelpad=20)
plt.show()
```



Conclusion 2

- Most Number of complaint requests received are from the city brooklyn .

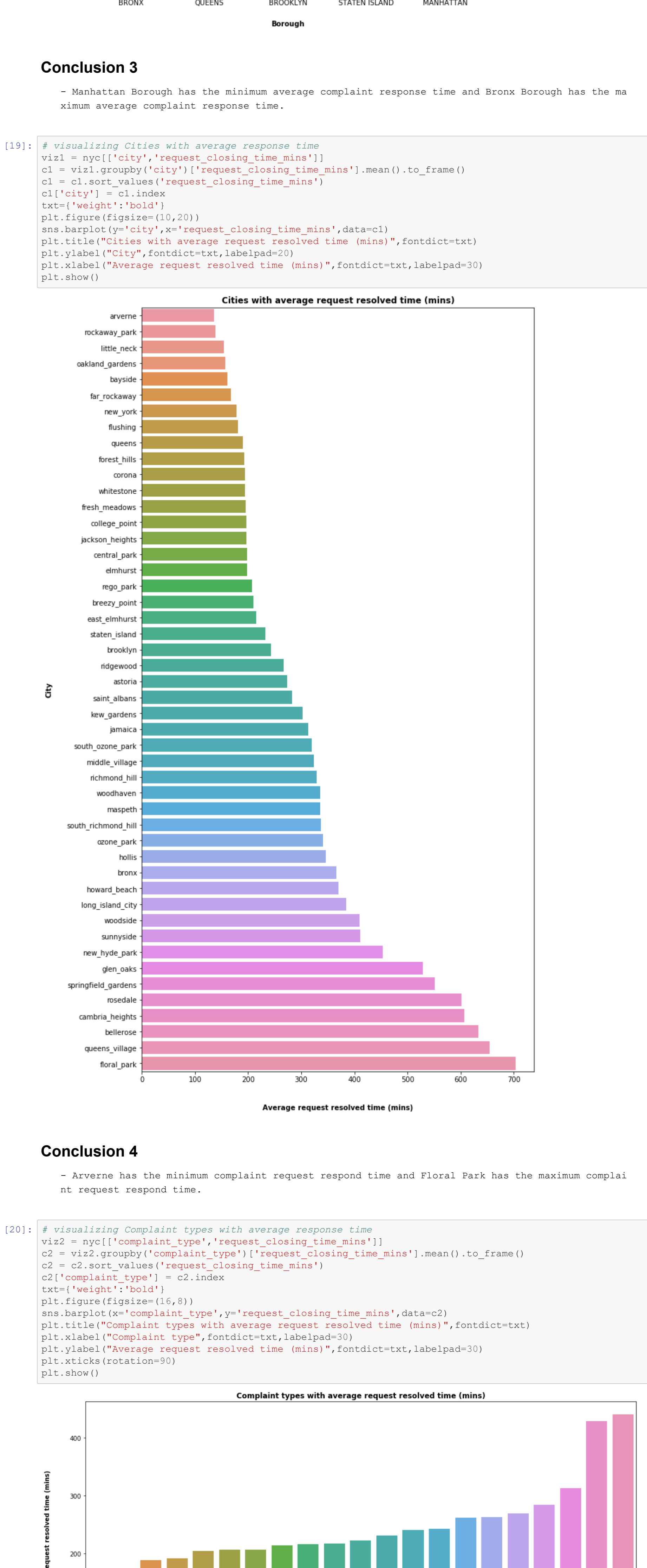
In [18]:
```
# Visualizing Boroughs with average response time
txt={'weight':'bold'}
plt.figure(figsize=(10,5))
sns.barplot(x="borough",y='request_closing_time_mins',data=nyc,order=['BRONX','QUEENS','BROOKLYN','STAT
EN ISLAND','MANHATTAN'])
plt.title("Boroughs with average complaint resolved time",fontdict=txt)
plt.xlabel("Borough",fontdict=txt,labelpad=20)
plt.ylabel("Average complaints resolved time(mins)",fontdict=txt,labelpad=30)
plt.show()
```



Conclusion 3

- Manhattan Borough has the minimum average complaint response time and Bronx Borough has the maximum average complaint response time.

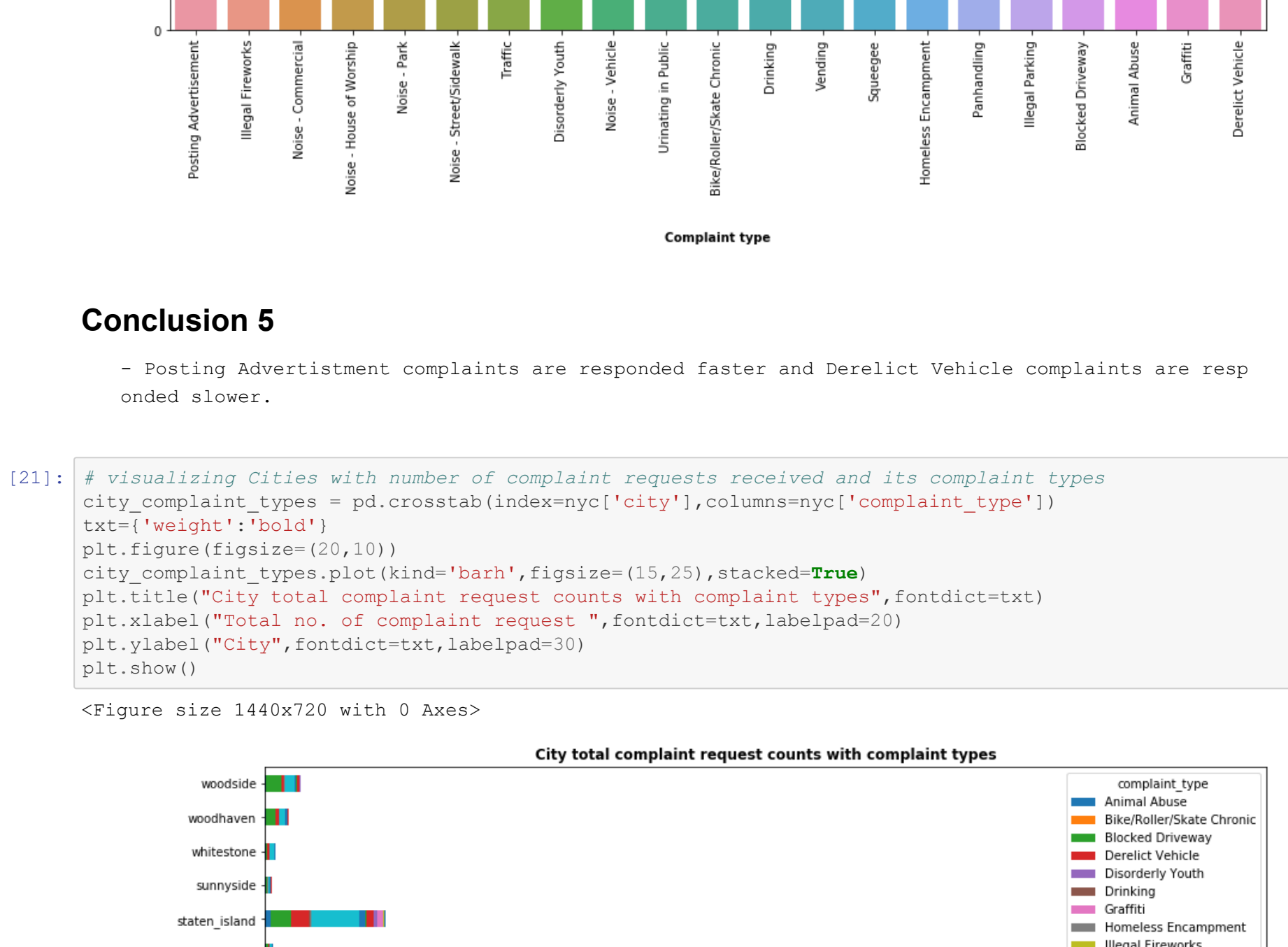
In [19]:
```
# Visualizing Cities with average response time
viz1 = nyc[['city','request_closing_time_mins']]
c1 = viz1.groupby('city')['request_closing_time_mins'].mean().to_frame()
c1 = c1.sort_values('request_closing_time_mins')
c1['city'] = c1.index
txt={'weight':'bold'}
plt.figure(figsize=(10,20))
sns.barplot(y='city',x='request_closing_time_mins',data=c1)
plt.title("Cities with average request resolved time (mins)",fontdict=txt)
plt.xlabel("Average request resolved time (mins)",fontdict=txt)
plt.ylabel("City",fontdict=txt,labelpad=30)
plt.show()
```



Conclusion 4

- Arverne has the minimum complaint request respond time and Floral Park has the maximum complaint request respond time.

In [20]:
```
# Visualizing Complaint types with average response time
viz2 = nyc[['complaint_type','request_closing_time_mins']]
c2 = viz2.groupby('complaint_type')['request_closing_time_mins'].mean().to_frame()
c2 = c2.sort_values('request_closing_time_mins')
c2['complaint_type'] = c2.index
txt={'weight':'bold'}
plt.figure(figsize=(10,8))
sns.barplot(x='complaint_type',y='request_closing_time_mins',data=c2)
plt.title("Complaint types with average request resolved time (mins)",fontdict=txt)
plt.xlabel("Complaint type",fontdict=txt,labelpad=30)
plt.ylabel("Average request resolved time (mins)",fontdict=txt,labelpad=30)
plt.xticks(rotation=90)
plt.show()
```

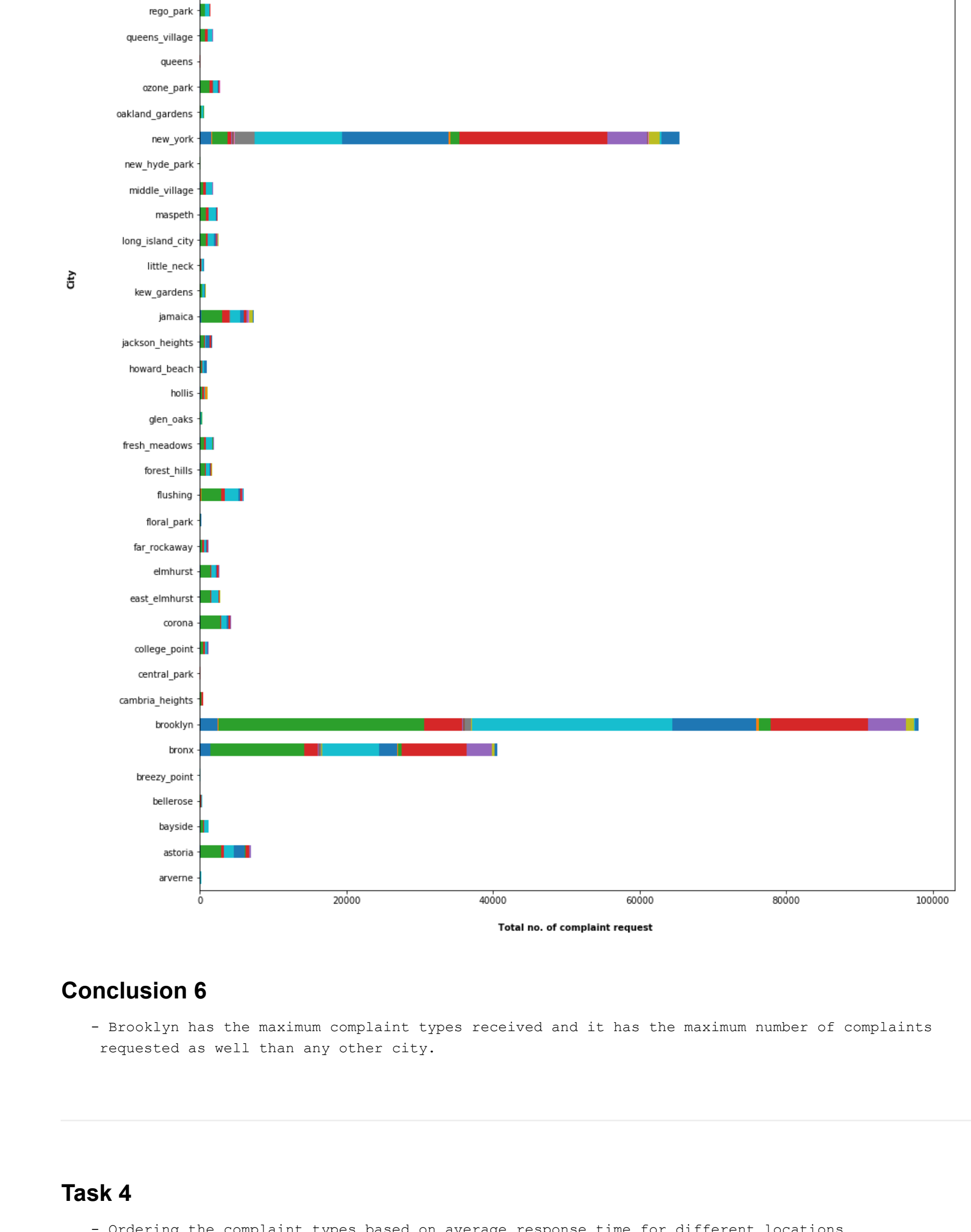


Conclusion 5

- Posting Advertisment complaints are responded faster and Derelict Vehicle complaints are responded slower.

In [21]:
```
# visualizing Cities with number of complaint requests received and its complaint types
city_complaint_types = pd.crosstab(index=nyc['city'],columns=nyc['complaint_type'])
txt={'weight':'bold'}
plt.figure(figsize=(20,10))
city_complaint_types.plot(kind='barh',figsize=(15,25),stacked=True)
plt.title("City total complaint request counts with complaint types",fontdict=txt)
plt.xlabel("Total no. of complaint request ",fontdict=txt,labelpad=20)
plt.ylabel("City",fontdict=txt,labelpad=30)
plt.show()
```

<Figure size 1440x720 with 0 Axes>



Conclusion 6

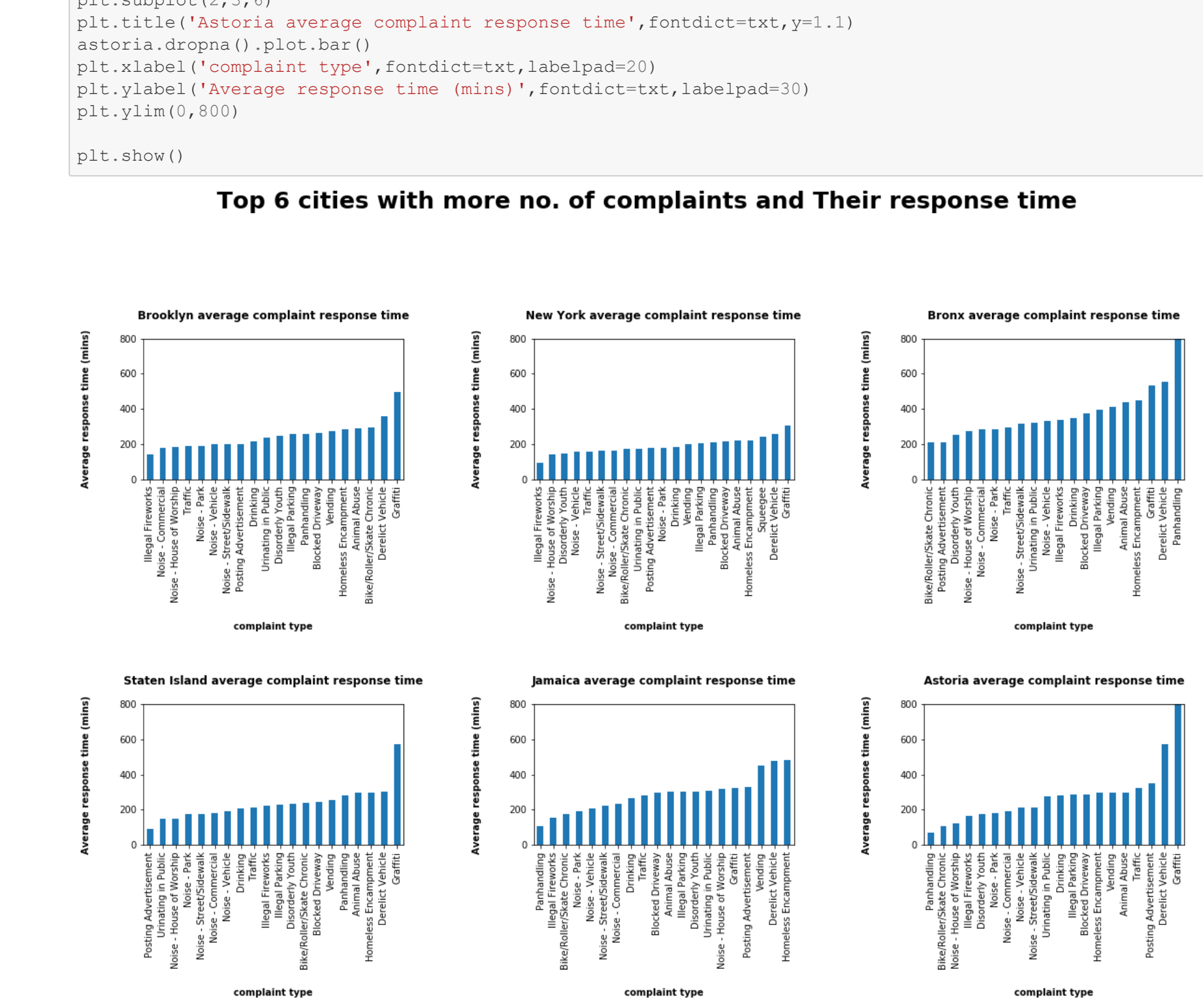
- Brooklyn has the maximum complaint requests received and it has the maximum number of complaints requested as well than any other city.

Task 4

- Ordering the complaint types based on average response time for different locations

In [22]:
```
# Grouping complaints by cities and finding mean response time
# Sorting the mean response time of different complaint types for each city
city_complainttype_group = nyc.groupby(['city','complaint_type'])['request_closing_time_mins'].mean().un
stack(level=1)
city_complainttype_group = city_complainttype_group.T
col = city_complainttype_group.columns
for i in col:
    exec("{} = city_complainttype_group['{}'].sort_values()".format(i,i))
```

In [23]:
```
# Visualizing the top 6 cities with the mean response time sorted for different complaint types
plt.subplots_adjust(hspace=1.6,wspace=0.5)
plt.suptitle("Top 6 cities with more no. of complaints and Their response time",fontweight="bold",fonts
ize=25,y=1.1)
txt={'weight':'bold'}
plt.subplot(2,3,1)
plt.title('Brooklyn average complaint response time',fontdict=txt,y=1.1)
brooklyn.dropna().plot.bar()
plt.xlabel('complaint type',fontdict=txt,labelpad=20)
plt.ylabel('Average response time (mins)',fontdict=txt,labelpad=30)
plt.ylim(0,800)

plt.subplot(2,3,2)
plt.title('New York average complaint response time',fontdict=txt,y=1.1)
new_york.dropna().plot.bar()
plt.xlabel('complaint type',fontdict=txt,labelpad=20)
plt.ylabel('Average response time (mins)',fontdict=txt,labelpad=30)
plt.ylim(0,800)

plt.subplot(2,3,3)
plt.title('Bronx average complaint response time',fontdict=txt,y=1.1)
bronx.dropna().plot.bar()
plt.xlabel('complaint type',fontdict=txt,labelpad=20)
plt.ylabel('Average response time (mins)',fontdict=txt,labelpad=30)
plt.ylim(0,800)

plt.subplot(2,3,4)
plt.title('Staten Island average complaint response time',fontdict=txt,y=1.1)
staten_island.dropna().plot.bar()
plt.xlabel('complaint type',fontdict=txt,labelpad=20)
plt.ylabel('Average response time (mins)',fontdict=txt,labelpad=30)
plt.ylim(0,800)

plt.subplot(2,3,5)
plt.title('Jamaica average complaint response time',fontdict=txt,y=1.1)
jamaica.dropna().plot.bar()
plt.xlabel('complaint type',fontdict=txt,labelpad=20)
plt.ylabel('Average response time (mins)',fontdict=txt,labelpad=30)
plt.ylim(0,800)

plt.subplot(2,3,6)
plt.title('Astoria average complaint response time',fontdict=txt,y=1.1)
astoria.dropna().plot.bar()
plt.xlabel('complaint type',fontdict=txt,labelpad=20)
plt.ylabel('Average response time (mins)',fontdict=txt,labelpad=30)
plt.ylim(0,800)

plt.show()
```



In [ ]:

Task 5

Statistical Test

```
- Whether the average response time across complaint types is similar or not (overall)
- Are the type of complaint or service requested and location related?
```

In [24]:
```
# Dropping NA values for statistical testing
nyc.dropna(subset=['request_closing_time_hrs','request_closing_time_mins'],inplace=True)
```

F-Test

Testing at Confidence level(95%) => alpha value = 0.05

*** Null Hypothesis : H0 : There is no significant difference in average response time across different complaint types**

*** Alternate Hypothesis : H1 : There is a significant difference in average response time across different complaint types**

In [32]:
```
# Storing mean response time for various complaint types
complaints = nyc['complaint_type'].value_counts().index
for i in range(len(complaints)):
    exec("sample{} = nyc.loc[(nyc['complaint_type'] == '{}') , 'request_closing_time_mins']".format(i+1
,complaints[i]))
```

In [33]:
```
# Performing F-statics
fscore,pvalue = stats.f_oneway(sample1,sample2,sample3,sample4,sample5,sample6,sample7,sample8,sample9,
sample10,sample11,sample12,sample13,sample14,sample15,sample16,sample17,sample18,sample19,sample20,samp
le21)
print("score : {:.2f} , pvalue : {:.2f}".format(fscore,pvalue))
```

```
score : 407.78 , pvalue : 0.00
```

Here , pvalue (0.00) < alpha value(0.05)

We reject our Null Hypothesis

```
- There is a significant difference in average response time across different complaint types
(i.e) the average response time across different complaint types is not similar (overall)
```

Chi-Square Test of Independence

Testing at Confidence level(95%) => alpha value = 0.05

*** Null Hypothesis : H0 : There is no significant relation between type of complaint and location**

*** Alternate Hypothesis : H1 : There is some significant relation between type of complaint and location**

In [30]:
```
# Performing Chi-square test of independence
location_complaint_type = pd.crosstab(nyc['complaint_type'],nyc['location'])
```

In [31]:
```
cscore,pval,df,et = stats.chi2_contingency(location_complaint_type)
print("score : {:.2f} , pvalue : {:.2f}".format(cscore,pval))
```

```
score : 4160248.36 , pvalue : 0.00
```

Here , pvalue (0.00) < alpha value(0.05)

We reject our Null Hypothesis

```
- There is some significant relation between type of complaint and location (i.e) The type
  of complaint or service requested and the location are related
```

In [ ]: