Project 1 - Customer Service Requests Analysis

From Ms. Ankita Kale

Submitted in August 2020

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.

Domain: Customer Service

Analysis Tasks to be performed:

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

Import a 311 NYC service request.

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)

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.
Order the complaint types based on the average ‘Request_Closing_Time’, grouping them for different locations.
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?

In [1]:
```
# import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns
%matplotlib inline
%config IPCompleter.greedy=True
```

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
```

1. Import a 311 NYC service request

In [3]:
```
# read data into a DataFrame
data = pd.read_csv('311_Service_Requests_from_2010_to_Present.csv')
```

In [4]:
```
# read data into DataFrame to store original data
dataOrig = pd.read_csv('311_Service_Requests_from_2010_to_Present.csv')
```

In [5]:
```
# Show number of rows and columns
print(data.shape)
```

```
(300698, 53)
```

Total 300698 rows and 52 columns present in original dataset

In [6]:
```
data.duplicated().sum()
```

Out[6]:
```
0
```

In [7]:
```
# Dropping duplicate records
data.drop_duplicates(inplace=True)
data.shape
```

Out[7]:
```
(300698, 53)
```

After dropping duplicate records, there are total 300492 records present in the DataFrame

In [8]:
```
# Show data from dataframe
data.head()
```

Out[8]:

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type
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
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
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
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

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type
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

5 rows × 53 columns

In [9]:

```
# Show column names
#data.keys()
data.columns
```

Out[9]:

```
Index(['Unique Key', 'Created Date', 'Closed Date', 'Agency', 'Agency N
ame',
       'Complaint Type', 'Descriptor', 'Location Type', 'Incident Zip',
       'Incident Address', 'Street Name', 'Cross Street 1', 'Cross Stre
et 2',
       'Intersection Street 1', 'Intersection Street 2', 'Address Typ
e',
       '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 Boroug
h',
       '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 Complain
t',
       '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 [10]:

```
# Information  / details of dataframe, columns, rows
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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: 123.9+ MB
```

In [117]:
```
data['Complaint Type'].unique()
```

Out[117]:
```
array(['Noise - Street/Sidewalk', 'Blocked Driveway', 'Illegal Parkin
g',
       'Derelict Vehicle', 'Noise - Commercial',
       'Noise - House of Worship', 'Posting Advertisement',
       'Noise - Vehicle', 'Animal Abuse', 'Vending', 'Traffic',
       'Drinking', 'Bike/Roller/Skate Chronic', 'Panhandling',
       'Noise - Park', 'Homeless Encampment', 'Urinating in Public',
       'Graffiti', 'Disorderly Youth', 'Illegal Fireworks',
       'Agency Issues', 'Squeegee', 'Animal in a Park'], dtype=object)
```

In [12]:
```
data['Descriptor'].unique()
```

Out[12]:
```
array(['Loud Music/Party', 'No Access', 'Commercial Overnight Parking',
       'Blocked Sidewalk', 'Posted Parking Sign Violation',
```

```
       'Blocked Hydrant', 'With License Plate', 'Partial Access',
       'Unauthorized Bus Layover', 'Double Parked Blocking Vehicle',
       'Double Parked Blocking Traffic', 'Vehicle', 'Loud Talking',
       'Banging/Pounding', 'Car/Truck Music', 'Tortured',
       'In Prohibited Area', 'Congestion/Gridlock', 'Neglected',
       'Car/Truck Horn', 'In Public', 'Other (complaint details)', nan,
       'No Shelter', 'Truck Route Violation', 'Unlicensed',
       'Overnight Commercial Storage', 'Engine Idling',
       'After Hours - Licensed Est', 'Detached Trailer',
       'Underage - Licensed Est', 'Chronic Stoplight Violation',
       'Loud Television', 'Chained', 'Building', 'In Car',
       'Police Report Requested', 'Chronic Speeding',
       'Playing in Unsuitable Place', 'Drag Racing',
       'Police Report Not Requested', 'Nuisance/Truant', 'Homeless Issu
e',
       'Language Access Complaint', 'Disruptive Passenger',
       'Animal Waste'], dtype=object)
```

In [13]:

```
# Finding null values in DataFrame
data.isna().sum()
```

Out[13]:

```
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
Intersection Street 1             256840
Intersection Street 2             257336
Address Type                        2815
City                                2614

Landmark                          300349
```

```
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
School or Citywide Complaint       300698
Vehicle Type                       300698
Taxi Company Borough               300698
Taxi Pick Up Location              300698
Bridge Highway Name                300455
Bridge Highway Direction           300455
Road Ramp                          300485
Bridge Highway Segment             300485
Garage Lot Name                    300698
Ferry Direction                    300697
Ferry Terminal Name                300696
Latitude                             3540
Longitude                            3540
Location                             3540
dtype: int64
```

Missing values are available in column 'Closed Data', so we will fill those values with the mode *value because we need to use the values from this column for further processing*

In [14]:
```
complaintTypecity = pd.DataFrame({'count': data.groupby(['Complaint Typ
e','City']).size()}).reset_index()
complaintTypecity
```

Out[14]:

	Complaint Type	City	count
0	Animal Abuse	ARVERNE	38
1	Animal Abuse	ASTORIA	125
2	Animal Abuse	BAYSIDE	37
3	Animal Abuse	BELLEROSE	7
4	Animal Abuse	BREEZY POINT	2
...	...	...	...
759	Vending	STATEN ISLAND	25
760	Vending	SUNNYSIDE	15
761	Vending	WHITESTONE	1
762	Vending	WOODHAVEN	6
763	Vending	WOODSIDE	15

764 rows × 3 columns

In [15]:
```
data.loc[:,['Complaint Type','City']]
```

Out[15]:

	Complaint Type	City
0	Noise - Street/Sidewalk	NEW YORK
1	Blocked Driveway	ASTORIA
2	Blocked Driveway	BRONX
3	Illegal Parking	BRONX

	Complaint Type	City
4	Illegal Parking	ELMHURST
...	...	...
300693	Noise - Commercial	NaN
300694	Blocked Driveway	RICHMOND HILL
300695	Noise - Commercial	BROOKLYN
300696	Noise - Commercial	BRONX
300697	Noise - Commercial	NEW YORK

300698 rows × 2 columns

In [16]:
```
data.groupby(['Borough','Complaint Type','Descriptor']).size()
```

Out[16]:
```
Borough      Complaint Type          Descriptor
BRONX        Animal Abuse            Chained                        132
                                     In Car                          36
                                     Neglected                      673
                                     No Shelter                      71
                                     Other (complaint details)      311
                                                                   ...
Unspecified  Noise - Vehicle         Engine Idling                   11
             Posting Advertisement   Vehicle                          1
             Traffic                 Truck Route Violation            1
             Vending                 In Prohibited Area               2
                                     Unlicensed                       5
Length: 288, dtype: int64
```

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

In [17]:
```
# Count total number of null values present in column 'Closed Date' of
```

```
 dataframe
data['Closed Date'].isna().sum()
```

Out[17]: 2164

In [18]:
```
# find mode of Closed Date column
#mode_closed_date = data['Closed Date'].mode()
#mode_closed_date
```

In [19]:
```
data.shape
```

Out[19]: (300698, 53)

In [20]:
```
# Drop records with null values present in the column 'Closed Date' of
 dataframe
data.drop(data[data['Closed Date'].isna()].index, inplace = True)
data['Closed Date'].isna().sum()
```

Out[20]: 0

In [21]:
```
# Find number of rows and columns after dropping null values in the col
umn 'Closed Date' of dataframe
data.shape
```

Out[21]: (298534, 53)

In [22]:
```
# fill NaN values in column 'Closed Date' by mode value
# data['Closed Date'].fillna('2015-11-08 07:34:00',  inplace=True)
#data['Closed Date'].fillna('11-08-15 7:34',  inplace=True)
```

In [23]:
```
# Count total number of null values present in column 'Closed Date' of
 dataframe
data['Created Date'].isna().sum()
```

Out[23]: 0

Now, Missing values are not present in the columns 'Created Date' as well as 'Closed

Date'

In [24]:
```
# Read / Show data from column 'Created Date' from original file (Befor
e converting into datetime datatype)
data[['Created Date','Closed Date']].head()
```

Out[24]:

	Created Date	Closed Date
0	12/31/2015 11:59:45 PM	01-01-16 0:55
1	12/31/2015 11:59:44 PM	01-01-16 1:26
2	12/31/2015 11:59:29 PM	01-01-16 4:51
3	12/31/2015 11:57:46 PM	01-01-16 7:43
4	12/31/2015 11:56:58 PM	01-01-16 3:24

In [25]:
```
# Convert the columns ‘Created Date’ and Closed Date’ to datetime datat
ype
data['Created Date']=pd.to_datetime(data['Created Date'],infer_datetime
_format=True)
data['Closed Date']=pd.to_datetime(data['Closed Date'],infer_datetime_f
ormat=True)
```

In [26]:
```
# columns ‘Created Date’ and Closed Date’ after converting to datetime
 datatype
data[['Created Date','Closed Date']].head()
```

Out[26]:

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

In [27]:
```
# create a new column ‘Request_Closing_Time’ as the time elapsed betwee
n request creation and request closing.
data['Request_Closing_Time'] =data['Closed Date'] - data['Created Date'
]
print(data['Request_Closing_Time'].head(10))
```

```
0   00:55:15
1   01:26:16
2   04:51:31
3   07:45:14
4   03:27:02
5   01:53:30
6   01:57:28
7   01:47:55
8   08:33:02
9   01:23:02
Name: Request_Closing_Time, dtype: timedelta64[ns]
```

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.

Insight 1 - Finding count of each Complaint Tpye

In [28]:
```
# Finding number of complaints
data['Complaint Type'].value_counts()
```

Out[28]:
```
Blocked Driveway             76810
Illegal Parking              74532
Noise - Street/Sidewalk      48076
Noise - Commercial           35247
Derelict Vehicle             17588
Noise - Vehicle              17033
Animal Abuse                  7768
Traffic                       4496
Homeless Encampment           4416
Noise - Park                  4022
```

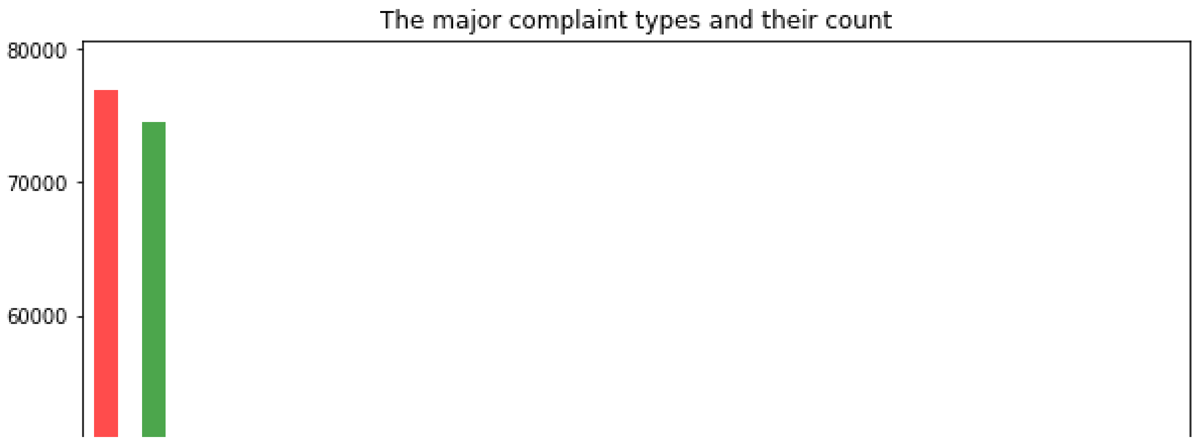
```
Vending                            3795
Drinking                           1275
Noise - House of Worship            929
Posting Advertisement               648
Urinating in Public                 592
Bike/Roller/Skate Chronic           424
Panhandling                         305
Disorderly Youth                    286
Illegal Fireworks                   168
Graffiti                            113
Agency Issues                         6
Squeegee                              4
Animal in a Park                      1
Name: Complaint Type, dtype: int64
```

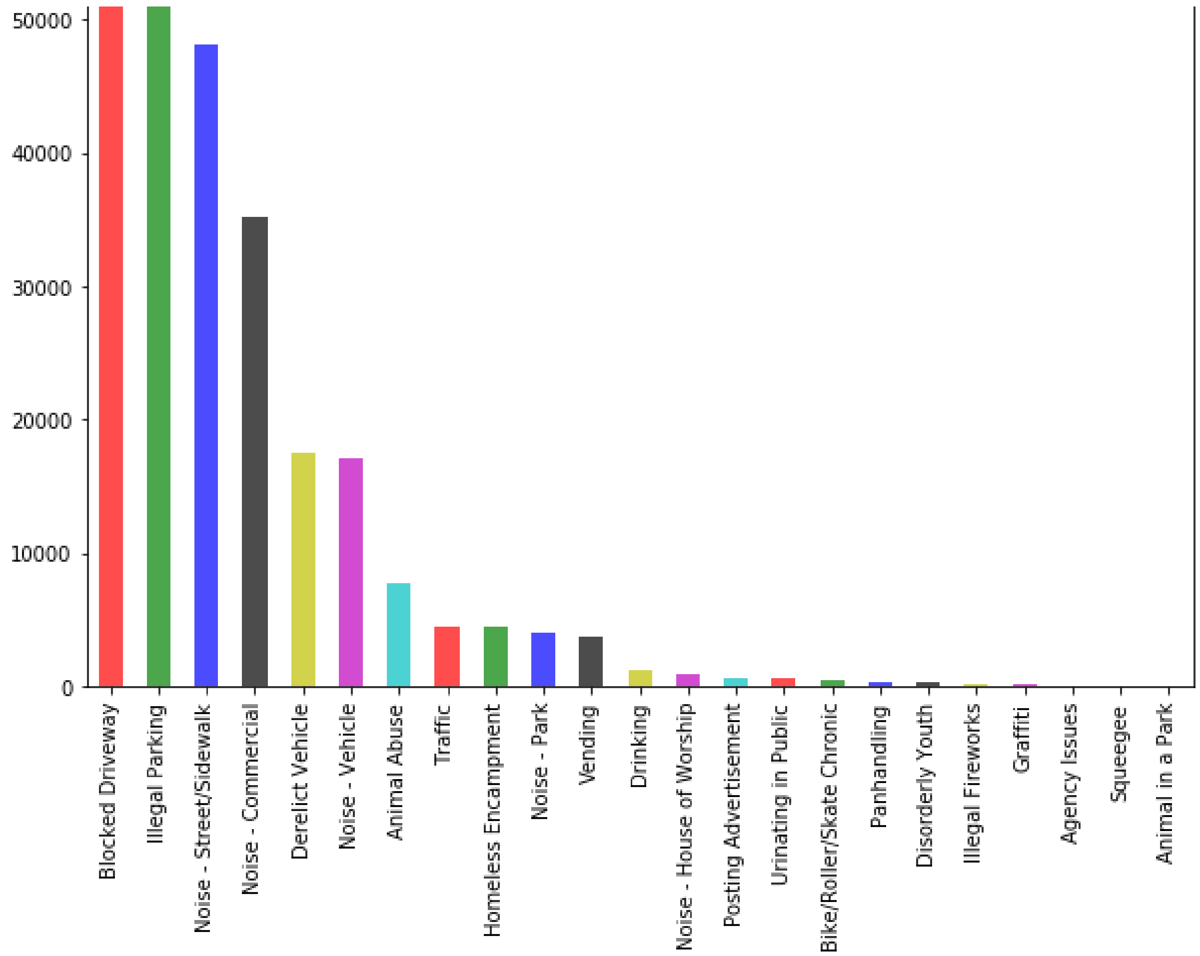
In [29]:

```python
# Plot the count of each complaint
#plt.figure(figsize=(20, 6))
#sns.countplot(x='Complaint Type', data=data, order=data['Complaint Typ
e'].value_counts().iloc[:].index)

major=data.loc[:,"Complaint Type"]
top=major.value_counts()
top.plot(kind='bar', color=list('rgbkymc'), alpha=0.7, figsize=(10,10), title='The major complaint types and their count')
```

Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x2c680e07508>



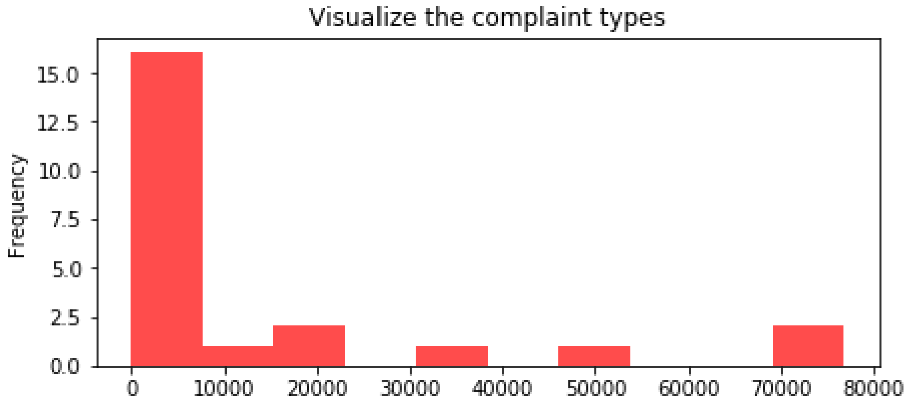


Top 5 complaints are Blocked Driveway, Illegal Parking, Noise - Street/Sidewalk, Noise - Commercial and Derelict Vehicle

In [30]:

```
#Visualize the complaint types
top.plot(kind='hist', color=list('rgbkymc'), alpha=0.7, figsize=(7,3),t
itle='Visualize the complaint types')
```

Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x2c683d1a088>



Insight 2 - Finding count of Descriptors

In [31]:
```
# Finding number of complaints
data['Descriptor'].value_counts()
```

Out[31]:
```
Loud Music/Party                  60829
No Access                         56822
Posted Parking Sign Violation     22274
Loud Talking                      21377
Partial Access                    19988
With License Plate                17588
Blocked Hydrant                   15898
Commercial Overnight Parking      11962
Car/Truck Music                   11227
Blocked Sidewalk                  10997
Double Parked Blocking Traffic     5636
Double Parked Blocking Vehicle     4208
Engine Idling                      4178
Banging/Pounding                   4110
Neglected                          3782
Car/Truck Horn                     3493
Congestion/Gridlock                2760
In Prohibited Area                 2024
Other (complaint details)          1967
Unlicensed                         1771
```

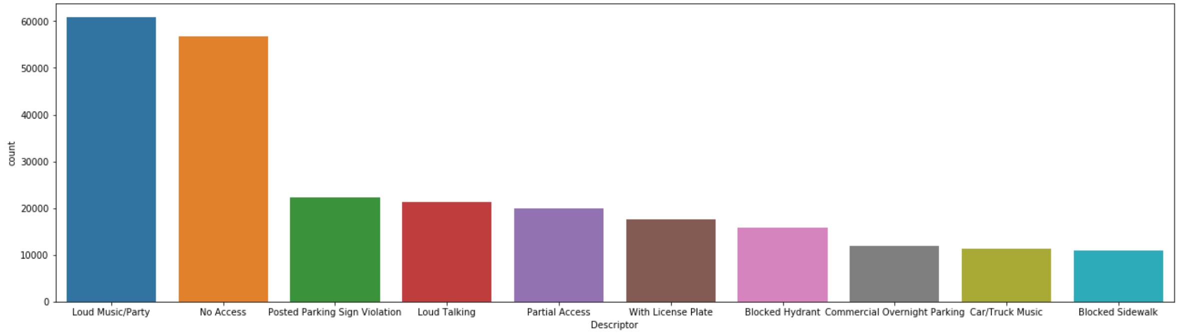
```
Overnight Commercial Storage      1756
Unauthorized Bus Layover          1340
Truck Route Violation             1013
In Public                          928
Tortured                           851
Vehicle                            588
Chained                            535
Detached Trailer                   461
No Shelter                         382
Chronic Stoplight Violation        280
Underage - Licensed Est            270
Chronic Speeding                   268
In Car                             251
Playing in Unsuitable Place        245
Drag Racing                        175
Loud Television                     93
Police Report Requested             90
After Hours - Licensed Est          77
Building                            60
Nuisance/Truant                     41
Police Report Not Requested         23
Language Access Complaint            6
Animal Waste                         1
Name: Descriptor, dtype: int64
```

In [32]:

```
# Plot the count of top 10 Descriptor
plt.figure(figsize=(22, 6))
sns.countplot(x='Descriptor', data=data, order=data['Descriptor'].value_counts().iloc[:10].index)
```

Out[32]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2c68d9bd888>
```



Top 5 complaints are Loud Music/Party, No Access, Posted Parking Sign Violation, Loud Talking and Partial Access

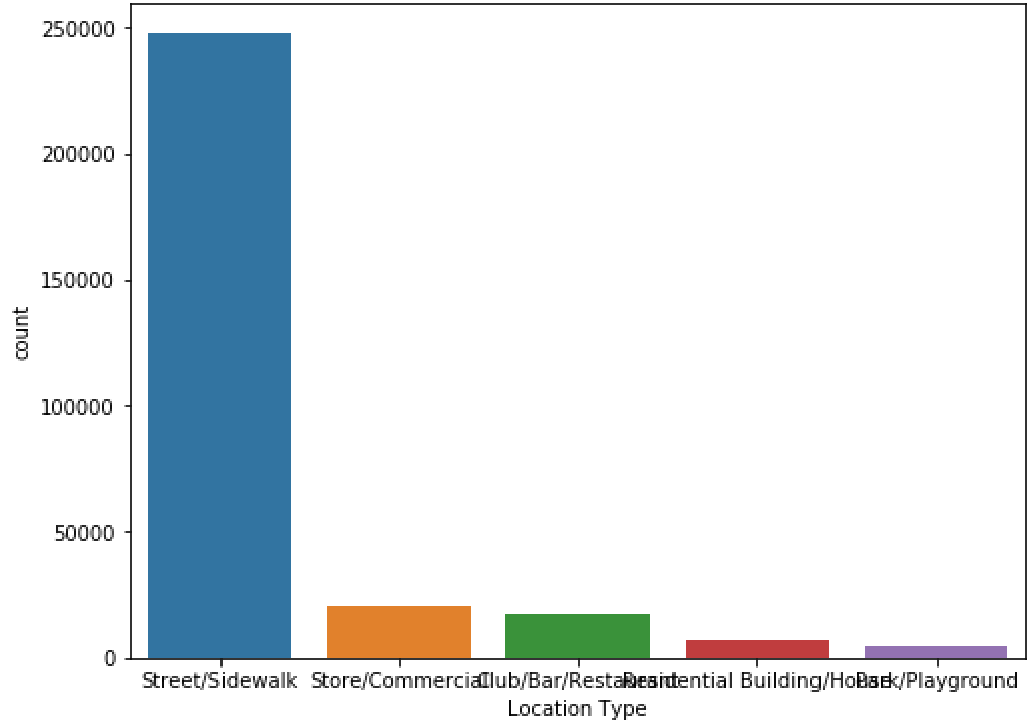
Insight 3 - Finding count of Locations and frequent location type

In [33]:
```
# Finding number of complaints
data['Location Type'].value_counts()
```

Out[33]:
```
Street/Sidewalk              247503
Store/Commercial              20183
Club/Bar/Restaurant           17227
Residential Building/House     6953
Park/Playground                4751
House of Worship                927
Residential Building            227
Highway                         214
Parking Lot                     117
House and Store                  93
Vacant Lot                       77
Commercial                       62
Roadway Tunnel                   35
Subway Station                   34
Bridge                            2
Park                              1
Name: Location Type, dtype: int64
```

In [34]:
```
# Plot the count of top 10 Descriptor
plt.figure(figsize=(8, 6))
sns.countplot(x='Location Type', data=data, order=data['Location Type']
.value_counts().iloc[:5].index)
```

Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x2c68e895bc8>



Top 5 Locations are Street/Sidewalk, Store/Commercial, Club/Bar/Restaurant, Residentia Building/House, and Park/Playground

Most frequent location type is Street/Sidewalk with 247434 complaints

Insight 4 - Plot Request Closing Time

In [35]:
```
#Insight - 1 - Categorize Request_Closing_Time as follows -
#Below 2 hours - Fast, Between 2 to 4 hours - Acceptable, Between 4 to
 6 - Slow, More than 6 hours - Very Slow
#For this, first will create new column Request_Closing_In_Hr and then
 create new column - Request_Closing_Time_Category
```

In [70]:
```
#Function to convert TimeDelta in Hour

def toHr(timeDel):
    days = timeDel.days
    hours = round(timeDel.seconds/3600, 2)
    result = (days * 24) + hours
    #print(days)
    #print(hours)
    return result
    #return round(pd.Timedelta(timeDel).seconds / 3600, 2)
```

In [37]:
```
test_days = data[data['Unique Key'] == 32122264]['Request_Closing_Time'
]
print(toHr(test_days[27704]))
print(test_days[27704])
print(test_days.dtype)
```

```
145.08
6 days 01:05:00
timedelta64[ns]
```

In [38]:
```
# Apply this function to every row of column Request_Closing_Time
data['Request_Closing_In_Hr'] = data['Request_Closing_Time'].apply(toHr
)
data['Request_Closing_In_Hr'].head()
```

Out[38]:
```
0    0.92
1    1.44
2    4.86
3    7.75
4    3.45
Name: Request_Closing_In_Hr, dtype: float64
```

In [39]:
```
import math
```

In [40]:
```
# Function to categorize hours - Less than 2 hours - Fast, Between 2 to
 4 hours - Acceptable, Between 4 to 6 - Slow,
# More than 6 hours - Very Slow

def hrToCategory(hr):
    if (math.isnan(hr)):
        return 'Unspecified'
    elif (hr < 2.0):
        return 'Fast'
    elif (4.0 > hr >= 2.0):
        return 'Acceptable'
    elif (6.0 > hr >= 4.0):
        return 'Slow'
    else:
        return 'Very Slow'

# Testing function
print(hrToCategory(1.99))

#Insight 2
#Create new column Request_Closing_Time_Category and apply function on
 column Request_Closing_In_Hr
data['Request_Closing_Time_Category'] = data['Request_Closing_In_Hr'].a
pply(hrToCategory)
data['Request_Closing_Time_Category'].head()
```

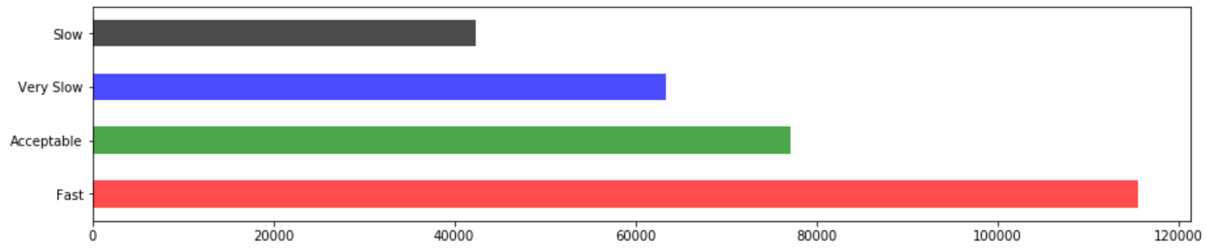
```
Fast
```

Out[40]:
```
0          Fast
1          Fast
2          Slow
3     Very Slow
4    Acceptable
Name: Request_Closing_Time_Category, dtype: object
```

In [41]:
```
data['Request_Closing_Time_Category'].value_counts()
```

Out[41]:
```
Fast           115550
Acceptable      77195
Very Slow       63388
Slow            42401
Name: Request_Closing_Time_Category, dtype: int64
```

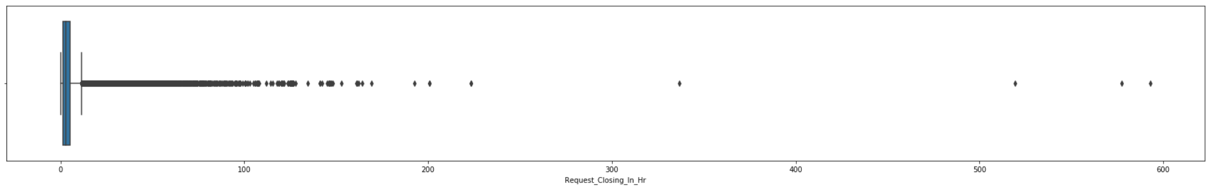
In [42]:
```
#Create Bar plot for Request_Closing_Time_Category to check frequency i
n Request_Closing_Time_Category and it prove
#Most count is in Fast category means closed less than 2 hours
data['Request_Closing_Time_Category'].value_counts().plot(kind="barh",
color=list('rgbkymc'), alpha=0.7, figsize=(15,3))
plt.show()
```



It seems that, most of the complaints resolved in less time.

In [43]:
```
fig_dims = (30, 4)
fig, ax = plt.subplots(figsize=fig_dims)
sns.boxplot(x=data["Request_Closing_In_Hr"])
```

Out[43]: <matplotlib.axes._subplots.AxesSubplot at 0x2c6959bc888>



Here we found some outliers in Request_Closing_In_Hr which I have classified into

Unspecified type

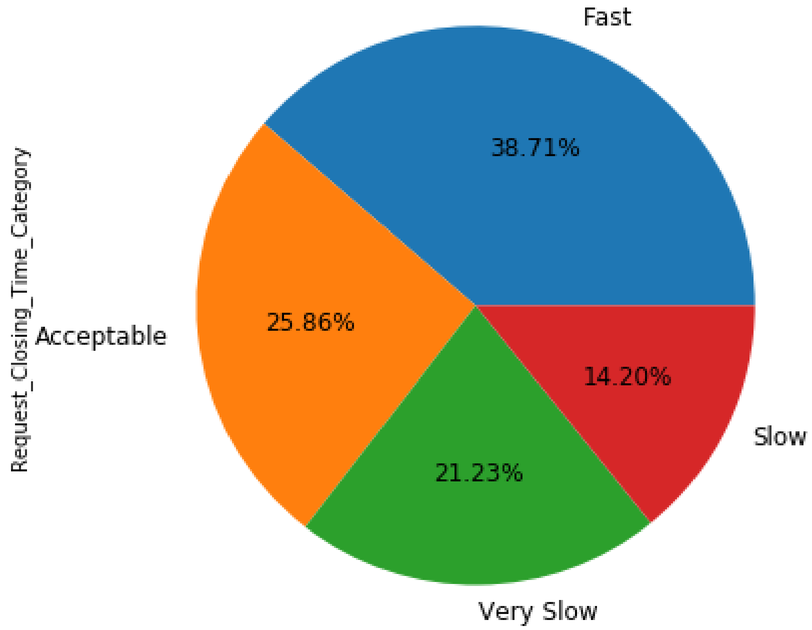
Insight 5 - Plot Pie Chart for Request Closing Time

In [44]:
```
# To align Pie Plot in the center (Horixontally)

from IPython.core.display import HTML
HTML("""
<style>
.output_png {
    display: table-cell;
    text-align: center;
    vertical-align: middle;
}
</style>
""")
```

Out[44]:

In [45]:
```
#Create Pie Chart for Request_Closing_Time_Category to check percentage
s of frequency in Request_Closing_Time_Category
data['Request_Closing_Time_Category'].value_counts().plot(kind="pie",ra
dius=0.8, figsize=(10,8), autopct='%1.2f%%', fontsize=12)
plt.show()
```



It proves that Most count is in Fast category as 41.35% from the total Request_Closing_Time_Category

Insight 6 - Plot Frequency of complaints monthwise

In [46]:
```
# Insight 1 - To check with Month have Complain creation most and least
#We will create one column with Create_Month name
#Created Series for months in text format
```

```
monthSeries = pd.Series({1: 'Jan', 2: 'Feb', 3: 'Mar', 4: 'Apr', 5: 'May', 6: 'Jun', 7: 'Jul', 8: 'Aug', 9: 'Sep', 10: 'Oct', 11: 'Nov', 12: 'Dec'})
print(monthSeries)
print(monthSeries[12])
```

```
1     Jan
2     Feb
3     Mar
4     Apr
5     May
6     Jun
7     Jul
8     Aug
9     Sep
10    Oct
11    Nov
12    Dec
dtype: object
Dec
```

In [47]:

```
import datetime as dt
import time, datetime
```

In [48]:

```
data['Created Date'].dtype

# Function to fetch month from Created Date column
def getMonth(cDate):
    a = str(cDate)
    DateTime = datetime.datetime.strptime(a, "%Y-%m-%d %H:%M:%S")
    return monthSeries[DateTime.month]

#Test function getMonth
print(data['Created Date'][0])
print(getMonth(data['Created Date'][0]))
```

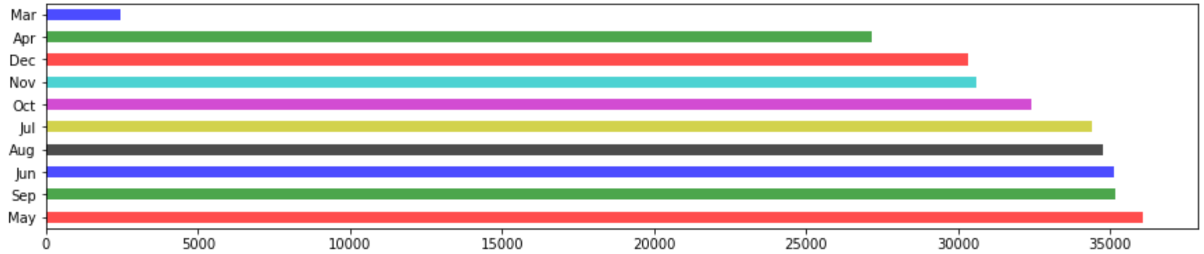
```
2015-12-31 23:59:45
Dec
```

In [49]:
```
# Created new column Created_Month and kept all text format months in t
hat column
data['Created_Month'] = data['Created Date'].apply(getMonth)
data['Created_Month']
```

Out[49]:
```
0         Dec
1         Dec
2         Dec
3         Dec
4         Dec
         ...
300692    Mar
300694    Mar
300695    Mar
300696    Mar
300697    Mar
Name: Created_Month, Length: 298534, dtype: object
```

In [50]:
```
data['Created_Month'].value_counts()

#Create Bar plot for Complain Created Month to check frequency and it p
rove Most count is in May month and least is in March a nd in January t
here is no any complain
data['Created_Month'].value_counts().plot(kind="barh", color=list('rgbk
ymc'), alpha=0.7, figsize=(15,3))
plt.show()
```



Most of the complaints get registered in the month of May

In [54]:
```
# To confirm doubt of January doesn't have any value, we used original
 dataframe and check if any entry for Jan month
dataOrig[dataOrig['Created Date'].str.startswith('01/')]
```

Out[54]:

Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	Incident Zip	Incident Address

0 rows × 53 columns

In [55]:
```
# To confirm doubt of January doesn't have any value, we used original
 dataframe and check if any entry for Jan month
dataOrig[dataOrig['Created Date'].str.startswith('02/')]
```

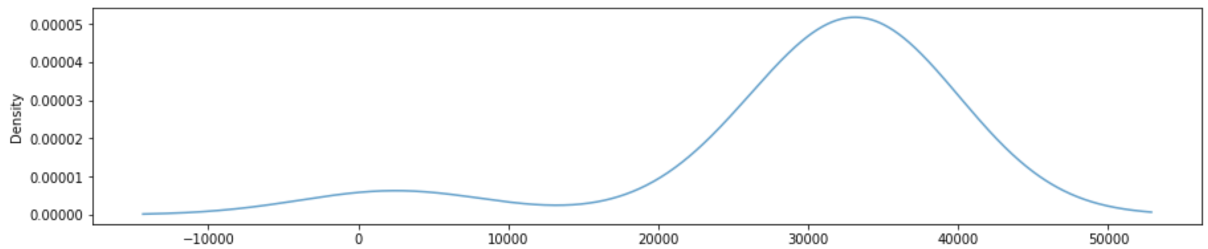
Out[55]:

Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	Incident Zip	Incident Address

0 rows × 53 columns

No complaints registered in the month of JAnuary and February

In [71]:
```
# We can see the density of complaints created month wise
data['Created_Month'].value_counts().plot(kind="density", alpha=0.7, fi
gsize=(15,3))
plt.show()
```



4. Order the complaint types based on the average ‘Request_Closing_Time’, grouping them for different locations.

In [58]:
```
#For location we can choose here City, so first check if there is missi
ng
data['City'].isnull().sum()
```

Out[58]:
```
506
```

In [59]:
```
# Fill all missing value with some default value here I used - Not Avai
lable
data['City'].fillna('Not Available', inplace=True)
data['City'].head()
```

Out[59]:
```
0     NEW YORK
1      ASTORIA
2        BRONX
3        BRONX
4     ELMHURST
Name: City, dtype: object
```

In [60]:
```
#Group them for City (location) first and Complain Type in that
df_data_grouped = data.groupby(['City', 'Complaint Type'])
```

In [61]:
```
#get average of this grouped dataframe, and get Request_Closing_Time co
lumn from there
data_mean = df_data_grouped.mean()['Request_Closing_In_Hr']
data_mean.isnull().sum()
```

Out[61]:
```
0
```

In [62]:
```
# Group by City(location) first and then Complain Type and showing aver
age of Request Closing in Hour
df_data_grouped = data.groupby(['City','Complaint Type']).agg({'Request
_Closing_In_Hr': 'mean'})
```

```
print(df_data_grouped.shape)
df_data_grouped
```

(778, 1)

Out[62]:

City	Complaint Type	Request_Closing_In_Hr
ARVERNE	**Animal Abuse**	2.153158
	Blocked Driveway	2.526000
	Derelict Vehicle	2.968889
	Disorderly Youth	3.595000
	Drinking	0.240000
...	**...**	...
Woodside	**Blocked Driveway**	6.405455
	Derelict Vehicle	4.965000
	Illegal Parking	5.219500
	Noise - Commercial	2.390000
	Noise - Street/Sidewalk	3.410000

778 rows × 1 columns

In [63]:

```
# Check if any value is NaN
df_data_grouped[df_data_grouped['Request_Closing_In_Hr'].isnull()]
```

Out[63]:

City	Complaint Type	Request_Closing_In_Hr

In [64]:

```
# drop null values from this group
df_data_grouped_withoutna = df_data_grouped.dropna()
```

In [65]:
```
# verify if new group has null values
df_data_grouped_withoutna.isnull().sum()
```

Out[65]:
```
Request_Closing_In_Hr    0
dtype: int64
```

In [66]:
```
# verify number of rows after dropping null values
print(df_data_grouped_withoutna.shape)
df_data_grouped_withoutna
```

```
(778, 1)
```

Out[66]:

		Request_Closing_In_Hr
City	**Complaint Type**	
ARVERNE	**Animal Abuse**	2.153158
	Blocked Driveway	2.526000
	Derelict Vehicle	2.968889
	Disorderly Youth	3.595000
	Drinking	0.240000
...	**...**	...
Woodside	**Blocked Driveway**	6.405455
	Derelict Vehicle	4.965000
	Illegal Parking	5.219500
	Noise - Commercial	2.390000
	Noise - Street/Sidewalk	3.410000

778 rows × 1 columns

In [67]:
```
# Sorting by column - Request_Closing_In_Hr for City on grouped
df_data_sorted = df_data_grouped_withoutna.sort_values(['City', 'Reques
t_Closing_In_Hr'])
df_data_sorted
```

Out[67]:

		Request_Closing_In_Hr
City	**Complaint Type**	
ARVERNE	**Drinking**	0.240000
	Vending	0.480000
	Urinating in Public	0.690000
	Panhandling	1.030000
	Noise - Park	1.285000
...	**...**	...
Woodside	**Noise - Commercial**	2.390000
	Noise - Street/Sidewalk	3.410000
	Derelict Vehicle	4.965000
	Illegal Parking	5.219500
	Blocked Driveway	6.405455

778 rows × 1 columns

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?

In [72]:

```
import scipy.stats as stats
```

```python
from math import sqrt
```

In [73]:
```python
##### Try ANOVA for first one

# H0 : All Complain Types and Average Response Time mean is similar

# H1 : Not similar
```

In [75]:
```python
data['Complaint Type'].value_counts()
```

Out[75]:
```
Blocked Driveway             76810
Illegal Parking              74532
Noise - Street/Sidewalk      48076
Noise - Commercial           35247
Derelict Vehicle             17588
Noise - Vehicle              17033
Animal Abuse                  7768
Traffic                       4496
Homeless Encampment           4416
Noise - Park                  4022
Vending                       3795
Drinking                      1275
Noise - House of Worship       929
Posting Advertisement          648
Urinating in Public            592
Bike/Roller/Skate Chronic      424
Panhandling                    305
Disorderly Youth               286
Illegal Fireworks              168
Graffiti                       113
Agency Issues                    6
Squeegee                         4
Animal in a Park                 1
Name: Complaint Type, dtype: int64
```

In [76]:
```python
top5_complaints_type = data['Complaint Type'].value_counts()[:5]
top5_complaints_type
```

Out[76]:

```
Blocked Driveway           76810
Illegal Parking            74532
Noise - Street/Sidewalk    48076
Noise - Commercial         35247
Derelict Vehicle           17588
Name: Complaint Type, dtype: int64
```

In [77]:
```
top5_complaints_type_names = top5_complaints_type.index
top5_complaints_type_names
```

Out[77]:
```
Index(['Blocked Driveway', 'Illegal Parking', 'Noise - Street/Sidewal
k',
       'Noise - Commercial', 'Derelict Vehicle'],
      dtype='object')
```

In [78]:
```
sample_data = data.loc[data['Complaint Type'].isin(top5_complaints_type
_names), ['Complaint Type', 'Request_Closing_In_Hr']]
sample_data.head()
```

Out[78]:

	Complaint Type	Request_Closing_In_Hr
0	Noise - Street/Sidewalk	0.92
1	Blocked Driveway	1.44
2	Blocked Driveway	4.86
3	Illegal Parking	7.75
4	Illegal Parking	3.45

In [79]:
```
sample_data.shape
```

Out[79]:
```
(252253, 2)
```

In [80]:
```
sample_data.isnull().sum()
```

Out[80]:
```
Complaint Type           0
Request_Closing_In_Hr    0
dtype: int64
```

In [83]:
```
#sample_data[~sample_data.isin(['NaN', 'NaT']).any(axis=1)] #sample_dat
a[sample_data.isnull()]

#sample_data.dropna(how='any', inplace=True)
#sample_data.isnull().sum()

# sample_data_without_null[sample_data_without_null.isnull()]
```

In [84]:
```
#sample_data.shape
```

In [85]:
```
s1 = sample_data[sample_data['Complaint Type'] == top5_complaints_type_
names[0]].Request_Closing_In_Hr
s1.head()
```

Out[85]:
```
1     1.44
2     4.86
7     1.80
9     1.38
10    7.80
Name: Request_Closing_In_Hr, dtype: float64
```

In [86]:
```
s2 = sample_data[sample_data['Complaint Type'] == top5_complaints_type_
names[1]].Request_Closing_In_Hr
s2.head()
```

Out[86]:
```
3    7.75
4    3.45
5    1.89
6    1.96
8    8.55
Name: Request_Closing_In_Hr, dtype: float64
```

In [87]:
```
s3 = sample_data[sample_data['Complaint Type'] == top5_complaints_type_
names[2]].Request_Closing_In_Hr
s3.head()
```

Out[87]:
```
0     0.92
```

```
12    2.48
19    0.78
38    0.49
54    1.50
Name: Request_Closing_In_Hr, dtype: float64
```

In [88]:
```
s4 = sample_data[sample_data['Complaint Type'] == top5_complaints_type_
names[3]].Request_Closing_In_Hr
s4.head()
```

Out[88]:
```
17    0.85
18    2.93
22    1.26
29    2.50
30    1.99
Name: Request_Closing_In_Hr, dtype: float64
```

In [89]:
```
s5 = sample_data[sample_data['Complaint Type'] == top5_complaints_type_
names[4]].Request_Closing_In_Hr
s5.head()
```

Out[89]:
```
14      10.49
151      3.95
255      1.36
256      4.13
295      0.75
Name: Request_Closing_In_Hr, dtype: float64
```

In [90]:
```
print(s1.isnull().sum())
print(s2.isnull().sum())
print(s3.isnull().sum())
print(s4.isnull().sum())
print(s5.isnull().sum())
```

```
0
0
0
0
0
```

In [91]:
```
stats.f_oneway(s1, s2, s3, s4, s5)
```

Out[91]:
```
F_onewayResult(statistic=1799.598683238952, pvalue=0.0)
```

We can see p-value is less than 0.05, so we reject the Null Hypothesisthe i.e. average response time across complaint types is not similar

In [ ]:
```
### Try ChiSquare Test for second one - # Are the type of complaint or service requested and location related?

# H0 : 2 categories - Complain Type and Location is independent means not related

# Ha : 2 categories - Complain Type and Location is dependent means related
```

In [103]:
```
top5_location = data['City'].value_counts()[:5]
top5_location
```

Out[103]:
```
BROOKLYN         98295
NEW YORK         65972
BRONX            40697
STATEN ISLAND    12338
JAMAICA           7294
Name: City, dtype: int64
```

In [104]:
```
top5_location_names = top5_location.index
top5_location_names
```

Out[104]:
```
Index(['BROOKLYN', 'NEW YORK', 'BRONX', 'STATEN ISLAND', 'JAMAICA'], dtype='object')
```

In [105]:
```
sample_data_location_c_type = data.loc[(data['Complaint Type'].isin(top5_complaints_type_names)) & (data['City'].isin(top5_location_names)), ['Complaint Type', 'City']]
sample_data_location_c_type.head()
```

Out[105]:

	Complaint Type	City
0	Noise - Street/Sidewalk	NEW YORK
2	Blocked Driveway	BRONX
3	Illegal Parking	BRONX
5	Illegal Parking	BROOKLYN
6	Illegal Parking	NEW YORK

In [106]:
```
pd.crosstab(sample_data_location_c_type['Complaint Type'], sample_data_
location_c_type['City'], margins=True)
```

Out[106]:

City	BRONX	BROOKLYN	JAMAICA	NEW YORK	STATEN ISLAND	All
Complaint Type						
Blocked Driveway	12754	28147	2817	2070	2142	47930
Derelict Vehicle	1952	5179	953	537	1766	10387
Illegal Parking	7859	27461	1421	12125	4886	53752
Noise - Commercial	2433	11458	429	14544	677	29541
Noise - Street/Sidewalk	8890	13354	339	20426	816	43825
All	33888	85599	5959	49702	10287	185435

In [101]:
```
ch2, p_value, df, exp_frq = stats.chi2_contingency(pd.crosstab(sample_d
ata_location_c_type['Complaint Type'], sample_data_location_c_type['Cit
y']))
```

In [109]:
```
print(ch2)
print(p_value)
print(df)
print(exp_frq)
```

```
40522.94060211538
```

```
0.0
16
[[ 8759.14385095 22125.0576752   1540.24251085 12846.64092539
   2658.91503761]
 [ 1898.21045649  4794.76265538   333.78883706  2784.01959716
    576.21845391]
 [ 9823.10661957 24812.56207296  1727.33393372 14407.10709413
   2981.89027961]
 [ 5398.57852078 13636.47671152   949.30740691  7917.85144121
   1638.78591959]
 [ 8008.96055222 20230.14088495  1408.32731146 11746.38094211
   2431.19030927]]
```

We can see p-value is less than 0.05, so we reject the Null Hypothesis means complain type and location are not independent.

--- Thank You ! ---