In [144…
```
#importing necessary files

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import io
```

In [145…
```
#setting the directory
%cd "C:\Users\Public\Python\Project2\"
```

```
C:\Users\Public\Python\Project2
```

In [146…
```
#importing the data
servicerequest = pd.read_csv('311_Service_Requests_from_2010_to_Present.csv')
```

```
C:\Users\Linnet Ann Verghese\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3165: DtypeWarning: Columns (48,49) have mixed types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

In [147…
```
#checking the data types of all the columns
servicerequest.dtypes
```

Out[147…
```
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
```

In [148…

```
#displaying 1st 5 rows
servicerequest.head()
```

Out[148…

| | Unique Key | Created Date | Closed Date | Agency | Agency Name | Complaint Type | Descriptor | Location Type | Incid |
|---|---|---|---|---|---|---|---|---|---|
| **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 | 100 |
| **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 | 111 |
| **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 | 104 |
| **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 | 104 |
| **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 | 113 |

5 rows × 53 columns

In [149…

```
#checking the total no:of rows and columns
servicerequest.shape
```

Out[149… (300698, 53)

DATA PREPROCESSING

In [150…

```
#Finding & deleting empty columns
```

```
#From data description the following columns are empty.

servicerequest.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 [151…
```
#Deleting the empty columns
emptycolumns = ['Landmark','School Not Found','School or Citywide Complaint','Vehicle T
                'Taxi Pick Up Location','Bridge Highway Name','Bridge Highway Direction
                'Bridge Highway Segment','Garage Lot Name','Ferry Direction','Ferry Ter
servicerequest=servicerequest.drop(emptycolumns,axis=1)
```

In [152…
```
#From data description the following columns are unspecified.
servicerequest['Park Facility Name'].value_counts()  #ensuring there is no data
```

Out[152…
```
Unspecified                        300697
Alley Pond Park - Nature Center         1
Name: Park Facility Name, dtype: int64
```

In [153…
```
#Deleting the unspecified columns
unspecifiedcolumns =['Park Facility Name','School Name','School Number','School Region'
                     'School Address','School City','School State','School Zip']
servicerequest=servicerequest.drop(unspecifiedcolumns,axis=1)
```

In [154…
```
#Deleting duplicate columns
#Deleting location as it has the Latitude ,Longitude which are already seperate columns
servicerequest=servicerequest.drop(['Location'],axis=1)
```

In [155…
```
#checking for single valued numerical columns
servicerequest.describe()
```

Out[155…

| | Unique Key | Incident Zip | X Coordinate (State Plane) | Y Coordinate (State Plane) | Latitude | Longitude |
|---|---|---|---|---|---|---|
| **count** | 3.006980e+05 | 298083.000000 | 2.971580e+05 | 297158.000000 | 297158.000000 | 297158.000000 |
| **mean** | 3.130054e+07 | 10848.888645 | 1.004854e+06 | 203754.534416 | 40.725885 | -73.925630 |
| **std** | 5.738547e+05 | 583.182081 | 2.175338e+04 | 29880.183529 | 0.082012 | 0.078454 |
| **min** | 3.027948e+07 | 83.000000 | 9.133570e+05 | 121219.000000 | 40.499135 | -74.254937 |
| **25%** | 3.080118e+07 | 10310.000000 | 9.919752e+05 | 183343.000000 | 40.669796 | -73.972142 |
| **50%** | 3.130436e+07 | 11208.000000 | 1.003158e+06 | 201110.500000 | 40.718661 | -73.931781 |
| **75%** | 3.178446e+07 | 11238.000000 | 1.018372e+06 | 224125.250000 | 40.781840 | -73.876805 |
| **max** | 3.231065e+07 | 11697.000000 | 1.067173e+06 | 271876.000000 | 40.912869 | -73.700760 |

In [156…
```
objectcols = servicerequest.select_dtypes(include=['object'])
objectcols.Agency.value_counts()
#Agency is a single valued column, so we can drop it
servicerequest = servicerequest.drop(['Agency'],axis=1)
```

In [157…
```
objectcols['Agency Name'].value_counts() #New York City Police Department & NYPD are sa
```

Out[157…
```
New York City Police Department    300690
Internal Affairs Bureau                 6
```

```
NYPD                                    2
Name: Agency Name, dtype: int64
```

In [158…
```
objectcols['Facility Type'].value_counts()
#Facility Type is a single valued column ,so we can drop it
servicerequest = servicerequest.drop(['Facility Type'],axis=1)
```

In [159…
```
objectcols.isnull().sum().sort_values(ascending=False)
```

Out[159…
```
Intersection Street 2             257336
Intersection Street 1             256840
Cross Street 2                     49779
Cross Street 1                     49279
Incident Address                   44410
Street Name                        44410
Descriptor                          5914
Address Type                        2815
City                                2614
Resolution Action Updated Date      2187
Facility Type                       2171
Closed Date                         2164
Location Type                        131
Due Date                               3
Community Board                        0
Borough                                0
Resolution Description                 0
Created Date                           0
Status                                 0
Complaint Type                         0
Agency Name                            0
Agency                                 0
Park Borough                           0
dtype: int64
```

In [160…
```
#total values= 300698
servicerequest['Intersection Street 2'].isnull().value_counts()
```

Out[160…
```
True     257336
False     43362
Name: Intersection Street 2, dtype: int64
```

In [161…
```
257336/300698 *100
#since more than 70% columns has missing data we can fill the missing column NA
```

Out[161…
```
85.57955157666495
```

In [162…
```
servicerequest['Intersection Street 1'].isnull().value_counts()
```

Out[162…
```
True     256840
False     43858
Name: Intersection Street 1, dtype: int64
```

In [163…
```
256840/300698 *100
```

Out[163…
```
85.41460202595295
```

In [164…
```
#since more than 70% columns has missing data we can fill the missing column NA
```

In [165…
```
missingcols = ['Intersection Street 1','Intersection Street 2']
for col in missingcols :
    servicerequest[col]=servicerequest[col].fillna('NotAvailable')
```

In [167…
```
#removing the duplicate agency name
servicerequest['Agency Name']= servicerequest['Agency Name'].replace(['NYPD'],'New York
```

In [168…
```
#verifying the data
servicerequest['Agency Name'].value_counts()
```

Out[168…
```
New York City Police Department    300692
Internal Affairs Bureau                 6
Name: Agency Name, dtype: int64
```

In [169…
```
#changing Created Date,Closed Date,Due Date and Resolution Action Updated Date to datet

servicerequest['Created Date']=pd.to_datetime(servicerequest['Created Date'])
servicerequest['Closed Date']=pd.to_datetime(servicerequest['Closed Date'])
servicerequest['Due Date']=pd.to_datetime(servicerequest['Due Date'])
servicerequest['Resolution Action Updated Date']=pd.to_datetime(servicerequest['Resolut
```

In [170…
```
#Since both these columns have null values & their values cant be less than the created
#we can fill it with created date

servicerequest['Due Date']=servicerequest['Due Date'].fillna(servicerequest['Created Da
servicerequest['Resolution Action Updated Date']=servicerequest['Resolution Action Upda
(servicerequest['Created Date'])
```

Out[170…
```
0        2015-12-31 23:59:45
1        2015-12-31 23:59:44
2        2015-12-31 23:59:29
3        2015-12-31 23:57:46
4        2015-12-31 23:56:58
                 ...
300693   2015-03-29 00:33:41
300694   2015-03-29 00:33:28
300695   2015-03-29 00:33:03
300696   2015-03-29 00:33:02
300697   2015-03-29 00:33:01
Name: Created Date, Length: 300698, dtype: datetime64[ns]
```

In [171…
```
servicerequest['Due Date'].isnull().value_counts()#verifying that Due date has no more
```

Out[171…
```
False    300698
Name: Due Date, dtype: int64
```

In [172…
```
servicerequest['Resolution Action Updated Date'].isnull().value_counts()
#verifying that Resolution action update date has no more null values
```

Out[172…
```
False    300698
Name: Resolution Action Updated Date, dtype: int64
```

In [173…
```
#since the rest of the columns have less than 70% of missing data, we can fill it with
```

In [174…
```
#running a loop to find the most frequent value and fill it in for each col.
for col in servicerequest :
    servicerequest[col] = servicerequest[col].fillna(servicerequest[col].value_counts()
```

In [175…
```
servicerequest.isnull().sum().sort_values(ascending=False)
```

Out[175…
```
Unique Key                        0
Address Type                      0
Latitude                          0
Park Borough                      0
Y Coordinate (State Plane)        0
X Coordinate (State Plane)        0
Borough                           0
Community Board                   0
Resolution Action Updated Date    0
Resolution Description            0
Due Date                          0
Status                            0
City                              0
Intersection Street 2             0
Created Date                      0
Intersection Street 1             0
Cross Street 2                    0
Cross Street 1                    0
Street Name                       0
Incident Address                  0
Incident Zip                      0
Location Type                     0
Descriptor                        0
Complaint Type                    0
Agency Name                       0
Closed Date                       0
Longitude                         0
dtype: int64
```

In [177…
```
#coverting the data into upper case to make it look uniform
servicerequest.apply(lambda x: x.astype(str).str.upper())
```

Out[177…

| | Unique Key | Created Date | Closed Date | Agency Name | Complaint Type | Descriptor | Location T |
|---|---|---|---|---|---|---|---|
| **0** | 32310363 | 2015-12-31 23:59:45 | 2016-01-01 00:55:00 | NEW YORK CITY POLICE DEPARTMENT | NOISE - STREET/SIDEWALK | LOUD MUSIC/PARTY | STREET/SIDEW |
| **1** | 32309934 | 2015-12-31 23:59:44 | 2016-01-01 01:26:00 | NEW YORK CITY POLICE DEPARTMENT | BLOCKED DRIVEWAY | NO ACCESS | STREET/SIDEW |

| | Unique Key | Created Date | Closed Date | Agency Name | Complaint Type | Descriptor | Location T |
|---|---|---|---|---|---|---|---|
| **2** | 32309159 | 2015-12-31 23:59:29 | 2016-01-01 04:51:00 | NEW YORK CITY POLICE DEPARTMENT | BLOCKED DRIVEWAY | NO ACCESS | STREET/SIDEW |
| **3** | 32305098 | 2015-12-31 23:57:46 | 2016-01-01 07:43:00 | NEW YORK CITY POLICE DEPARTMENT | ILLEGAL PARKING | COMMERCIAL OVERNIGHT PARKING | STREET/SIDEW |
| **4** | 32306529 | 2015-12-31 23:56:58 | 2016-01-01 03:24:00 | NEW YORK CITY POLICE DEPARTMENT | ILLEGAL PARKING | BLOCKED SIDEWALK | STREET/SIDEW |
| **...** | ... | ... | ... | ... | ... | ... | |
| **300693** | 30281872 | 2015-03-29 00:33:41 | 2015-11-08 07:34:00 | NEW YORK CITY POLICE DEPARTMENT | NOISE - COMMERCIAL | LOUD MUSIC/PARTY | CLUB/BAR/RESTAURA |
| **300694** | 30281230 | 2015-03-29 00:33:28 | 2015-03-29 02:33:59 | NEW YORK CITY POLICE DEPARTMENT | BLOCKED DRIVEWAY | PARTIAL ACCESS | STREET/SIDEW |
| **300695** | 30283424 | 2015-03-29 00:33:03 | 2015-03-29 03:40:20 | NEW YORK CITY POLICE DEPARTMENT | NOISE - COMMERCIAL | LOUD MUSIC/PARTY | CLUB/BAR/RESTAURA |
| **300696** | 30280004 | 2015-03-29 00:33:02 | 2015-03-29 04:38:35 | NEW YORK CITY POLICE DEPARTMENT | NOISE - COMMERCIAL | LOUD MUSIC/PARTY | CLUB/BAR/RESTAURA |
| **300697** | 30281825 | 2015-03-29 00:33:01 | 2015-03-29 04:41:50 | NEW YORK CITY POLICE DEPARTMENT | NOISE - COMMERCIAL | LOUD MUSIC/PARTY | STORE/COMMERC |

300698 rows × 27 columns

In [178…

```
#dropping the unique key as its not required
servicerequest= servicerequest.drop(['Unique Key'],axis=1)
```

In [181…
```
servicerequest.columns
```

Out[181…
```
Index(['Created Date', 'Closed Date', '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', 'Status', 'Due Date', 'Resolution Description',
       'Resolution Action Updated Date', 'Community Board', 'Borough',
       'X Coordinate (State Plane)', 'Y Coordinate (State Plane)',
       'Park Borough', 'Latitude', 'Longitude'],
      dtype='object')
```

In [182…
```
#identifying categorical columns
categoricalcols= servicerequest[['Agency Name','Complaint Type','Descriptor','Location
                                 'Community Board','Borough','Park Borough']]
```

In [183…
```
#identifying the numerical columns which are supposed to be object columns and change t
cols = ['Incident Zip','X Coordinate (State Plane)','Y Coordinate (State Plane)','Latit
servicerequest = servicerequest.astype({'Incident Zip': np.object, 'X Coordinate (State
                                        'Y Coordinate (State Plane)': np.object,'Latitu
```

```
<ipython-input-183-871d5c661d81>:3: DeprecationWarning: `np.object` is a deprecated alia
s for the builtin `object`. To silence this warning, use `object` by itself. Doing this
will not modify any behavior and is safe.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/relea
se/1.20.0-notes.html#deprecations
  servicerequest = servicerequest.astype({'Incident Zip': np.object, 'X Coordinate (Stat
e Plane)': np.object,
<ipython-input-183-871d5c661d81>:4: DeprecationWarning: `np.object` is a deprecated alia
s for the builtin `object`. To silence this warning, use `object` by itself. Doing this
will not modify any behavior and is safe.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/relea
se/1.20.0-notes.html#deprecations
  'Y Coordinate (State Plane)': np.object,'Latitude':np.object,'Longitude':np.object})
```

In [184…
```
servicerequest.dtypes #ensuring the datatype is correct
```

Out[184…
```
Created Date                      datetime64[ns]
Closed Date                       datetime64[ns]
Agency Name                              object
Complaint Type                           object
Descriptor                               object
Location Type                            object
Incident Zip                             object
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
Status                                   object
Due Date                          datetime64[ns]
Resolution Description                   object
Resolution Action Updated Date           object
Community Board                          object
```

```
Borough                           object
X Coordinate (State Plane)        object
Y Coordinate (State Plane)        object
Park Borough                      object
Latitude                          object
Longitude                         object
dtype: object
```

In [186…
```
#Converting the time elapsed into minutes
#getting the requests that are in closed status.
closedservicerequests =servicerequest[servicerequest['Status']=='Closed']
```

In [187…
```
closedservicerequests.columns
```

Out[187…
```
Index(['Created Date', 'Closed Date', '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', 'Status', 'Due Date', 'Resolution Description',
       'Resolution Action Updated Date', 'Community Board', 'Borough',
       'X Coordinate (State Plane)', 'Y Coordinate (State Plane)',
       'Park Borough', 'Latitude', 'Longitude'],
      dtype='object')
```

In [188…
```
#1)create a new column ‘Request_Closing_Time’ as the time elapsed between request creat
# Created a new column and set it to 0. iterows is to iterate thru the rows and for eac
#closed date and created date and convert the value into minutes.
closedservicerequests['Request_Closing_Time']=0
for index,req in closedservicerequests.iterrows() :
    Request_Closing_Time = int(((req['Closed Date']-req['Created Date']).total_seconds(
    closedservicerequests.at[index,'Request_Closing_Time'] = Request_Closing_Time
closedservicerequests
```

```
<ipython-input-188-93c1312f794b>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  closedservicerequests['Request_Closing_Time']=0
```

Out[188…

| | Created Date | Closed Date | Agency Name | Complaint Type | Descriptor | Location Type | Incident Zip | Inci Adc |
|---|---|---|---|---|---|---|---|---|
| **0** | 2015-12-31 23:59:45 | 2016-01-01 00:55:00 | New York City Police Department | Noise - Street/Sidewalk | Loud Music/Party | Street/Sidewalk | 10034.0 | VERMI AVE |
| **1** | 2015-12-31 23:59:44 | 2016-01-01 01:26:00 | New York City Police Department | Blocked Driveway | No Access | Street/Sidewalk | 11105.0 | 27-( AVE |

| | Created Date | Closed Date | Agency Name | Complaint Type | Descriptor | Location Type | Incident Zip | Inci Add |
|---|---|---|---|---|---|---|---|---|
| **2** | 2015-12-31 23:59:29 | 2016-01-01 04:51:00 | New York City Police Department | Blocked Driveway | No Access | Street/Sidewalk | 10458.0 | VALEN AVE |
| **3** | 2015-12-31 23:57:46 | 2016-01-01 07:43:00 | New York City Police Department | Illegal Parking | Commercial Overnight Parking | Street/Sidewalk | 10461.0 | BAI AVE |
| **4** | 2015-12-31 23:56:58 | 2016-01-01 03:24:00 | New York City Police Department | Illegal Parking | Blocked Sidewalk | Street/Sidewalk | 11373.0 | 87-1 R |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **300692** | 2015-03-29 00:34:32 | 2015-03-29 01:13:01 | New York City Police Department | Noise - Commercial | Loud Music/Party | Store/Commercial | 10002.0 | 81 HE ST |
| **300694** | 2015-03-29 00:33:28 | 2015-03-29 02:33:59 | New York City Police Department | Blocked Driveway | Partial Access | Street/Sidewalk | 11418.0 | 100-1 AVE |
| **300695** | 2015-03-29 00:33:03 | 2015-03-29 03:40:20 | New York City Police Department | Noise - Commercial | Loud Music/Party | Club/Bar/Restaurant | 11206.0 | THR AVE |
| **300696** | 2015-03-29 00:33:02 | 2015-03-29 04:38:35 | New York City Police Department | Noise - Commercial | Loud Music/Party | Club/Bar/Restaurant | 10461.0 | 3151 TREM AVE |
| **300697** | 2015-03-29 00:33:01 | 2015-03-29 04:41:50 | New York City Police Department | Noise - Commercial | Loud Music/Party | Store/Commercial | 10036.0 | 251 V 48 ST |

298471 rows × 27 columns

In [189…

```
#2)Provide major insights/patterns that you can offer in a visual format (graphs or tab
#at least 4 major conclusions that you can come up with after generic data mining.
```

In [190…
```
# a)  Descriptor sorted based on the Request closing time.
closedservicerequests['Request_Closing_Time'].groupby(closedservicerequests['Descriptor
```

Out[190…
```
Descriptor
Animal Waste                       20210.000000
Police Report Not Requested          623.000000
With License Plate                   441.499119
Police Report Requested              379.111111
Detached Trailer                     330.368764
Chained                              320.813084
Neglected                            318.687037
Other (complaint details)            316.649212
Language Access Complaint            315.333333
Overnight Commercial Storage         302.281481
Underage - Licensed Est              299.774074
Tortured                             298.723854
No Shelter                           296.719895
Partial Access                       290.889667
No Access                            281.769047
Blocked Sidewalk                     278.556667
Posted Parking Sign Violation        278.025015
Commercial Overnight Parking         270.892651
Double Parked Blocking Vehicle       268.799382
Unlicensed                           265.305650
Blocked Hydrant                      257.171952
Double Parked Blocking Traffic       246.689851
Chronic Stoplight Violation          240.800000
In Car                               240.199203
Chronic Speeding                     238.160448
Unauthorized Bus Layover             234.102239
Truck Route Violation                232.669299
Car/Truck Music                      224.102343
In Prohibited Area                   218.876421
In Public                            215.684267
Playing in Unsuitable Place          215.371429
Building                             208.366667
Loud Music/Party                     205.246373
Engine Idling                        202.080421
Nuisance/Truant                      200.000000
Drag Racing                          199.188571
Car/Truck Horn                       196.991695
Loud Talking                         196.642463
Congestion/Gridlock                  191.106275
Banging/Pounding                     182.964234
After Hours - Licensed Est           180.909091
Loud Television                      155.989247
Vehicle                              109.052811
Name: Request_Closing_Time, dtype: float64
```

In [191…
```
#b)Complaint Type sorted based on the Request closing time.
closedservicerequests['Request_Closing_Time'].groupby(closedservicerequests['Complaint
```

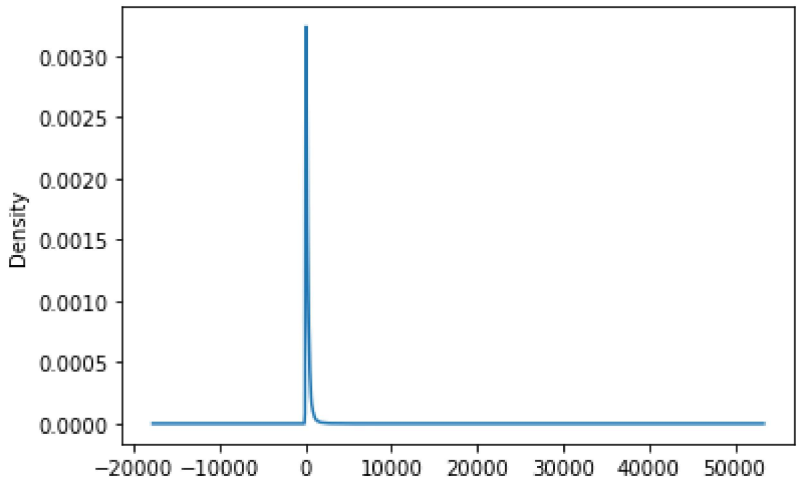
Out[191…
```
Complaint Type
Animal in a Park    20210.0
Name: Request_Closing_Time, dtype: float64
```

In [192…
```
#Complaint Type "Animal in a Park"  took an average of 20210 minutes.
```

In [193…
```
#c)Density Graph to show the'Request_Closing_Time'
```

```
closedservicerequests['Request_Closing_Time'].plot(kind='density')
```

Out[193… <AxesSubplot:ylabel='Density'>



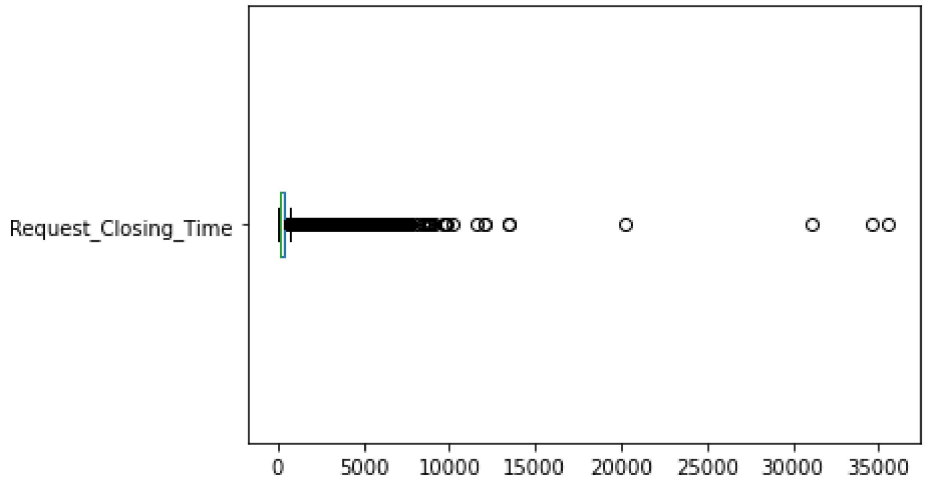

In [194…
```
#d)Location Type sorted based on the Request closing time.
closedservicerequests['Request_Closing_Time'].groupby(closedservicerequests['Location T
```

Out[194…
```
Location Type
Subway Station                   141.970588
Club/Bar/Restaurant              185.769954
House of Worship                 191.523193
Store/Commercial                 197.783035
Park/Playground                  206.836876
Highway                          223.074766
Bridge                           229.000000
Roadway Tunnel                   266.085714
Street/Sidewalk                  268.193365
Residential Building             288.775330
House and Store                  300.462366
Residential Building/House       309.203222
Parking Lot                      319.863248
Commercial                       320.193548
Vacant Lot                       448.103896
Park                           20210.000000
Name: Request_Closing_Time, dtype: float64
```

In [195…
```
#Request_Closing_Time of The Location Type 'Subway Station' was the fastest
```

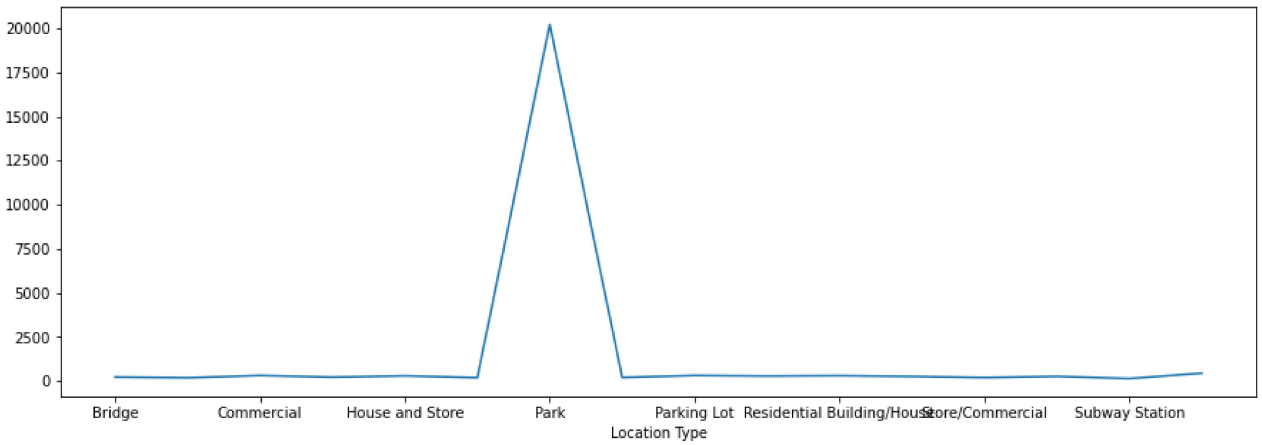
In [196…
```
#e)Box graph showing closed service request based on the Request Closing time
closedservicerequests['Request_Closing_Time'].plot(kind='box',vert=False)
```

Out[196… <AxesSubplot:>




In [197…
```
#f)plotting a graph showing that the most duration to close the request is taken for lo
fig,axs =plt.subplots(figsize=(15,5))
closedservicerequests['Request_Closing_Time'].groupby(closedservicerequests['Location T
```

Out[197… <AxesSubplot:xlabel='Location Type'>




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

In [198…
```
closedservicerequests['Request_Closing_Time'].groupby([closedservicerequests['Complaint
                                                       closedservicerequests['Location Type']]).me
```

Out[198…
```
Complaint Type            Location Type
Animal in a Park          Park                   20210.000000
Derelict Vehicle          Roadway Tunnel          1077.600000
Graffiti                  Street/Sidewalk          721.960000
Derelict Vehicle          Highway                  491.307692
Urinating in Public       Club/Bar/Restaurant      474.904762
                                                     ...
Posting Advertisement     Street/Sidewalk          108.839655
Noise - House of Worship  Street/Sidewalk           82.000000
Panhandling               Park/Playground           72.666667
Urinating in Public       Subway Station            68.916667
```

```
Noise - Commercial        Street/Sidewalk          62.666667
Name: Request_Closing_Time, Length: 72, dtype: float64
```

# 4)Whether the average response time across complaint types is similar or not (overall)

In [199…
```
#Null hypothesis - There is No significant difference in average response time across c
#Alt hypothesis - TThere is significant difference in average response time across comp
```

In [200…
```
closedservicerequests['Complaint Type'].value_counts() #list of all the complaint types
```

Out[200…
```
Blocked Driveway            76793
Illegal Parking             74515
Noise - Street/Sidewalk     48068
Noise - Commercial          35245
Derelict Vehicle            17585
Noise - Vehicle             17032
Animal Abuse                 7766
Traffic                      4493
Homeless Encampment          4410
Noise - Park                 4021
Vending                      3793
Drinking                     1275
Noise - House of Worship      929
Posting Advertisement         647
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 [201…
```
#Since response time is a numerical field and complaint type is a categorical multinomi
#Seggregating the data based on the Complaint Type.

Blocked_Driveway = closedservicerequests[closedservicerequests['Complaint Type']=='Bloc
Illegal_Parking = closedservicerequests[closedservicerequests['Complaint Type']=='Illeg
Noise_Street = closedservicerequests[closedservicerequests['Complaint Type']=='Noise -
Noise_Commercial = closedservicerequests[closedservicerequests['Complaint Type']=='Nois
Derelict_Vehicle = closedservicerequests[closedservicerequests['Complaint Type']=='Dere
Noise_Vehicle = closedservicerequests[closedservicerequests['Complaint Type']=='Noise -
Animal_Abuse = closedservicerequests[closedservicerequests['Complaint Type']=='Animal A
Traffic = closedservicerequests[closedservicerequests['Complaint Type']=='Traffic']
Homeless = closedservicerequests[closedservicerequests['Complaint Type']=='Homeless Enc
Noise_Park = closedservicerequests[closedservicerequests['Complaint Type']=='Noise - Pa
Vending = closedservicerequests[closedservicerequests['Complaint Type']=='Vending']
Drinking = closedservicerequests[closedservicerequests['Complaint Type']=='Drinking']
Noise_prayer = closedservicerequests[closedservicerequests['Complaint Type']=='Noise -
Posting_Advertisement = closedservicerequests[closedservicerequests['Complaint Type']==
Urinating_Public   = closedservicerequests[closedservicerequests['Complaint Type']=='Ur
Bike_Chronic = closedservicerequests[closedservicerequests['Complaint Type']=='Bike/Rol
Manhandling = closedservicerequests[closedservicerequests['Complaint Type']=='Panhandli
Disorderly_Youth = closedservicerequests[closedservicerequests['Complaint Type']=='Diso
```

```
Illegal_Fireworks = closedservicerequests[closedservicerequests['Complaint Type']=='Ill
Graffiti = closedservicerequests[closedservicerequests['Complaint Type']=='Graffiti']
Agency_Issues = closedservicerequests[closedservicerequests['Complaint Type']=='Agency
Squeegee = closedservicerequests[closedservicerequests['Complaint Type']=='Squeegee']
Animal_Park = closedservicerequests[closedservicerequests['Complaint Type']=='Animal in
```

In [202…

```
from scipy.stats import f_oneway #importing the appropraite library to conduct the ANOV
```

In [203…

```
#Input for the test is the request closing time for all the different Complaint types.
f_oneway(Blocked_Driveway.Request_Closing_Time,Illegal_Parking.Request_Closing_Time,
         Noise_Street.Request_Closing_Time,Noise_Commercial.Request_Closing_Time,Dereli
         Noise_Vehicle.Request_Closing_Time,Animal_Abuse.Request_Closing_Time,Traffic.R
         Homeless.Request_Closing_Time,Noise_Park.Request_Closing_Time,Vending.Request_
         Drinking.Request_Closing_Time,Noise_prayer.Request_Closing_Time,Posting_Advert
         Urinating_Public.Request_Closing_Time, Bike_Chronic.Request_Closing_Time,Manha
          Disorderly_Youth.Request_Closing_Time,Illegal_Fireworks.Request_Closing_Time
         Agency_Issues.Request_Closing_Time,Squeegee.Request_Closing_Time,Animal_Park.R
```

Out[203… F_onewayResult(statistic=513.9606866025157, pvalue=0.0)

In [204…

```
#Since p-value 0.0<0.05, reject Null .
#So the conclusion in this case is that there is significant difference in average resp
# The average response time across complaint types is not similar.
0.0<0.05
```

Out[204… True

# 5)Are the type of complaint or service requested and location related?

In [205…

```
#Since both the columns are categorical columns, we can use CHi-SQUARE test  of INDEPEN
#Null hypothesis - There is no association between the Complaint type and the location.
#Alt hypothesis - There is association between the Complaint type and the location.
```

In [206…

```
from scipy.stats import chi2_contingency #importing Chi-square test
```

In [207…

```
#The input for this test is cross tabulation of both the columns.
chi2_contingency(pd.crosstab(closedservicerequests['Complaint Type'],closedservicereque
```

Out[207…

```
(1327778.1127065355,
 0.0,
 330,
 array([[4.02049110e-05, 3.46305001e-01, 1.24635224e-03, 4.30192548e-03,
         1.86952836e-03, 1.86349763e-02, 2.01024555e-05, 9.55067662e-02,
         2.35198730e-03, 4.56325740e-03, 1.39772373e-01, 7.03585943e-04,
         4.05727860e-01, 4.97672471e+00, 6.83483488e-04, 1.54788907e-03],
        [5.20385565e-02, 4.48234106e+02, 1.61319525e+00, 5.56812555e+00,
         2.41979288e+00, 2.41198709e+01, 2.60192783e-02, 1.23617591e+02,
         3.04425556e+00, 5.90637616e+00, 1.80912042e+02, 9.10674739e-01,
         5.25147093e+02, 6.44154068e+03, 8.84655461e-01, 2.00348443e+00],
```

```
[6.70081850e-06, 5.77175002e-02, 2.07725374e-04, 7.16987580e-04,
 3.11588060e-04, 3.10582938e-03, 3.35040925e-06, 1.59177944e-02,
 3.91997883e-04, 7.60542900e-04, 2.32953955e-02, 1.17264324e-04,
 6.76213099e-02, 8.29454118e-01, 1.13913915e-04, 2.57981512e-04],
[2.84114705e-03, 2.44722201e+01, 8.80755584e-02, 3.04002734e-01,
 1.32113338e-01, 1.31687166e+00, 1.42057352e-03, 6.74914481e+00,
 1.66207102e-01, 3.22470190e-01, 9.87724771e+00, 4.97200733e-02,
 2.86714354e+01, 3.51688546e+02, 4.82994998e-02, 1.09384161e-01],
[5.14575955e-01, 4.43229999e+03, 1.59518546e+01, 5.50596272e+01,
 2.39277819e+01, 2.38505955e+02, 2.57287978e-01, 1.22237518e+03,
 3.01026934e+01, 5.84043709e+01, 1.78892331e+03, 9.00507922e+00,
 5.19284325e+03, 6.36962701e+04, 8.74779124e+00, 1.98111743e+01],
[1.17833893e-01, 1.01496224e+03, 3.65285070e+00, 1.26082266e+01,
 5.47927604e+00, 5.46160096e+01, 5.89169467e-02, 2.79914414e+02,
 6.89328276e+00, 1.33741469e+01, 4.09649530e+02, 2.06209313e+00,
 1.18912074e+03, 1.45859507e+04, 2.00317619e+00, 4.53660490e+00],
[1.91643409e-03, 1.65072051e+01, 5.94094569e-02, 2.05058448e-01,
 8.91141853e-02, 8.88267202e-01, 9.58217046e-04, 4.55248919e+00,
 1.12111394e-01, 2.17515269e-01, 6.66248312e+00, 3.35375966e-02,
 1.93396946e+01, 2.37223878e+02, 3.25793796e-02, 7.37827126e-02],
[8.54354359e-03, 7.35898127e+01, 2.64849851e-01, 9.14159165e-01,
 3.97274777e-01, 3.95993246e+00, 4.27177180e-03, 2.02951878e+01,
 4.99797300e-01, 9.69692198e-01, 2.97016293e+01, 1.49512013e-01,
 8.62171702e+01, 1.05755400e+03, 1.45240241e-01, 3.28926428e-01],
[7.57192491e-04, 6.52207752e+00, 2.34729672e-02, 8.10195965e-02,
 3.52094508e-02, 3.50958720e-01, 3.78596246e-04, 1.79871076e+00,
 4.42957607e-02, 8.59413477e-02, 2.63237970e+00, 1.32508686e-02,
 7.64120802e+00, 9.37283153e+01, 1.28722723e-02, 2.91519109e-02],
[2.95506096e-02, 2.54534176e+02, 9.16068898e-01, 3.16191523e+00,
 1.37410335e+00, 1.36967076e+01, 1.47753048e-02, 7.01974731e+01,
 1.72871066e+00, 3.35399419e+00, 1.02732694e+02, 5.17135668e-01,
 2.98209977e+02, 3.65789266e+03, 5.02360363e-01, 1.13769847e+00],
[1.12573751e-03, 9.69654003e+00, 3.48978628e-02, 1.20453913e-01,
 5.23467942e-02, 5.21779335e-01, 5.62868754e-04, 2.67418945e+00,
 6.58556443e-02, 1.27771207e-01, 3.91362645e+00, 1.97004064e-02,
 1.13603801e+01, 1.39348292e+02, 1.91375377e-02, 4.33408941e-02],
[4.99311491e-01, 4.30081953e+03, 1.54786562e+01, 5.34263295e+01,
 2.32179843e+01, 2.31430876e+02, 2.49655745e-01, 1.18611445e+03,
 2.92097222e+01, 5.66718542e+01, 1.73585640e+03, 8.73795109e+00,
 5.03880191e+03, 6.18067736e+04, 8.48829535e+00, 1.92234924e+01],
[2.36170348e-01, 2.03425329e+03, 7.32128079e+00, 2.52702273e+01,
 1.09819212e+01, 1.09464956e+02, 1.18085174e-01, 5.61022662e+02,
 1.38159654e+01, 2.68053345e+01, 8.21046216e+02, 4.13298109e+00,
 2.38331307e+03, 2.92341104e+04, 4.01489592e+00, 9.09255841e+00],
[6.22506039e-03, 5.36195577e+01, 1.92976872e-01, 6.66081462e-01,
 2.89465308e-01, 2.88531549e+00, 3.11253020e-03, 1.47876310e+01,
 3.64166033e-01, 7.06544354e-01, 2.16414224e+01, 1.08938557e-01,
 6.28201969e+01, 7.70562875e+02, 1.05826027e-01, 2.39664825e-01],
[2.69439912e-02, 2.32082068e+02, 8.35263727e-01, 2.88300706e+00,
 1.25289559e+00, 1.24885399e+01, 1.34719956e-02, 6.40054511e+01,
 1.57622349e+00, 3.05814300e+00, 9.36707854e+01, 4.71519846e-01,
 2.71905287e+02, 3.33523501e+03, 4.58047851e-01, 1.03734366e+00],
[3.22094944e-01, 2.77436480e+03, 9.98494326e+00, 3.44641590e+01,
 1.49774149e+01, 1.49291006e+02, 1.61047472e-01, 7.65136539e+02,
 1.88425542e+01, 3.65577761e+01, 1.11976307e+03, 5.63666152e+00,
 3.25042113e+03, 3.98702005e+04, 5.47561405e+00, 1.24006553e+01],
[1.14128341e-01, 9.83044463e+02, 3.53797856e+00, 1.22117325e+01,
 5.30696785e+00, 5.28984860e+01, 5.70641704e-02, 2.71111874e+02,
 6.67650794e+00, 1.29535667e+01, 3.96767177e+02, 1.99724596e+00,
 1.15172615e+03, 1.41272625e+04, 1.94018179e+00, 4.39394112e+00],
[2.04374964e-03, 1.76038376e+01, 6.33562390e-02, 2.18681212e-01,
 9.50343584e-02, 9.47277960e-01, 1.02187482e-03, 4.85492728e+00,
 1.19559354e-01, 2.31965585e-01, 7.10509564e+00, 3.57656188e-02,
 2.06244995e+01, 2.52983506e+02, 3.47437439e-02, 7.86843613e-02],
[4.33542957e-03, 3.73432226e+01, 1.34398317e-01, 4.63890964e-01,
```

```
       2.01597475e-01, 2.00947161e+00, 2.16771479e-03, 1.02988130e+01,
       2.53622630e-01, 4.92071257e-01, 1.50721209e+01, 7.58700175e-02,
       4.37509875e+01, 5.36656814e+02, 7.37023027e-02, 1.66914039e-01],
      [2.68032740e-05, 2.30870001e-01, 8.30901495e-04, 2.86795032e-03,
       1.24635224e-03, 1.24233175e-02, 1.34016370e-05, 6.36711774e-02,
       1.56799153e-03, 3.04217160e-03, 9.31815821e-02, 4.69057295e-04,
       2.70485240e-01, 3.31781647e+00, 4.55655658e-04, 1.03192605e-03],
      [3.01067775e-02, 2.59324728e+02, 9.33310104e-01, 3.22142520e+00,
       1.39996516e+00, 1.39544914e+01, 1.50533888e-02, 7.15186501e+01,
       1.76124649e+00, 3.41711925e+00, 1.04666212e+02, 5.26868607e-01,
       3.03822546e+02, 3.72673735e+03, 5.11815218e-01, 1.15911094e+00],
      [3.96688455e-03, 3.41687601e+01, 1.22973421e-01, 4.24456647e-01,
       1.84460132e-01, 1.83865099e+00, 1.98344228e-03, 9.42333426e+00,
       2.32062746e-01, 4.50241397e-01, 1.37908742e+01, 6.94204797e-02,
       4.00318155e+01, 4.91036838e+02, 6.74370374e-02, 1.52725055e-01],
      [2.54162046e-02, 2.18922478e+02, 7.87902342e-01, 2.71953389e+00,
       1.18185351e+00, 1.17804108e+01, 1.27081023e-02, 6.03761940e+01,
       1.48684797e+00, 2.88473922e+00, 8.83594353e+01, 4.44783580e-01,
       2.56487629e+02, 3.14611947e+03, 4.32075478e-01, 9.78523877e-01]]))
```

In [208…

```
# Since p-value =0.0<0.5, Reject Null
#SO the conclusion is that there is association between the Complaint type and the loca
#The Complaint type and the location type are related.
```