Customer Service Requests Analysis

311 Service Requests from 2010 to Present

DESCRIPTION: You have been asked to perform data analysis of service request (311) calls from New York City. You have also been asked to utilize data wrangling techniques to understand the pattern in the data and visualize the major types of complaints

INITIALLY IMPORT ALL LIBRARY

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


import warnings
warnings.filterwarnings('ignore')
```

IMPORT THE DATA SET BY USING PANDAS LIBRARY

```
calls = pd.read_csv('311_Service_Requests_from_2010_to_Present.csv')
```

1.UNDERSTAND THE DATASET:

** calls.head(): This will display the first few rows of the dataset**

** calls. shape: This will give you the number of rows and columns in the dataset**

** calls.info (): This will display information about each column, such as its data type and the number of non-null values**

** calls.describe(): This will give you a summary of statistics for each numeric column in the dataset**

>>> calls.head(): This will display the first few rows of the dataset <<<

```
calls.head(2)
```

```
   Unique Key            Created Date             Closed Date 
Agency  \
0    32310363  12/31/2015 11:59:45 PM  01/01/2016 12:55:15 AM   NYPD
```

```
1    32309934  12/31/2015 11:59:44 PM  01/01/2016 01:26:57 AM   NYPD


                       Agency Name           Complaint Type 
Descriptor  \
0  New York City Police Department  Noise - Street/Sidewalk  Loud 
Music/Party
1  New York City Police Department         Blocked Driveway         No 
Access

     Location Type  Incident Zip      Incident Address  ...  \
0  Street/Sidewalk       10034.0  71 VERMILYEA AVENUE  ...
1  Street/Sidewalk       11105.0      27-07 23 AVENUE  ...

  Bridge Highway Name Bridge Highway Direction Road Ramp  \
0                 NaN                      NaN       NaN
1                 NaN                      NaN       NaN

  Bridge Highway Segment Garage Lot Name Ferry Direction Ferry 
Terminal Name  \
0                    NaN             NaN             NaN 
NaN
1                    NaN             NaN             NaN 
NaN

    Latitude  Longitude                                  Location
0  40.865682 -73.923501   (40.86568153633767, -73.92350095571744)
1  40.775945 -73.915094  (40.775945312321085, -73.91509393898605)

[2 rows x 53 columns]
```

>>> calls.shape: This will give us the number of rows and columns in the dataset <<<

```
calls.shape
```

```
(364558, 53)
```

>>> calls.info():This will display information about each column,such as its data type and the number of non-null values<<<

```
calls.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 364558 entries, 0 to 364557
Data columns (total 53 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   Unique Key                      364558 non-null  int64  
 1   Created Date                    364558 non-null  object 
 2   Closed Date                     362177 non-null  object 
 3   Agency                          364558 non-null  object 
 4   Agency Name                     364558 non-null  object 
 5   Complaint Type                  364558 non-null  object 
 6   Descriptor                      358057 non-null  object 
 7   Location Type                   364425 non-null  object 
 8   Incident Zip                    361560 non-null  float64
 9   Incident Address                312859 non-null  object 
 10  Street Name                     312859 non-null  object 
 11  Cross Street 1                  307370 non-null  object 
 12  Cross Street 2                  306753 non-null  object 
 13  Intersection Street 1           51120 non-null   object 
 14  Intersection Street 2           50512 non-null   object 
 15  Address Type                    361306 non-null  object 
 16  City                            361561 non-null  object 
 17  Landmark                        375 non-null     object 
 18  Facility Type                   362169 non-null  object 
 19  Status                          364558 non-null  object 
 20  Due Date                        364555 non-null  object 
 21  Resolution Description          364558 non-null  object 
 22  Resolution Action Updated Date  362156 non-null  object 
 23  Community Board                 364558 non-null  object 
 24  Borough                         364558 non-null  object 
 25  X Coordinate (State Plane)      360528 non-null  float64
 26  Y Coordinate (State Plane)      360528 non-null  float64
 27  Park Facility Name              364558 non-null  object 
 28  Park Borough                    364558 non-null  object 
 29  School Name                     364558 non-null  object 
 30  School Number                   364558 non-null  object 
 31  School Region                   364557 non-null  object 
 32  School Code                     364557 non-null  object 
 33  School Phone Number             364558 non-null  object 
 34  School Address                  364558 non-null  object 
 35  School City                     364558 non-null  object 
 36  School State                    364558 non-null  object 
 37  School Zip                      364557 non-null  object 
 38  School Not Found                364558 non-null  object 
 39  School or Citywide Complaint    0 non-null       float64
 40  Vehicle Type                    0 non-null       float64
 41  Taxi Company Borough            0 non-null       float64
 42  Taxi Pick Up Location           0 non-null       float64
 43  Bridge Highway Name             297 non-null     object 
 44  Bridge Highway Direction        297 non-null     object 
 45  Road Ramp                       262 non-null     object
 46  Bridge Highway Segment          262 non-null     object
 47  Garage Lot Name                 0 non-null       float64
 48  Ferry Direction                 1 non-null       object
 49  Ferry Terminal Name             2 non-null       object
 50  Latitude                        360528 non-null  float64
 51  Longitude                       360528 non-null  float64
 52  Location                        360528 non-null  object
dtypes: float64(10), int64(1), object(42)
memory usage: 147.4+ MB
```

>>> calls.describe(): This will give us a summary of statistics for each numeric column in the dataset <<<

```
calls.describe()

         Unique Key   Incident Zip  X Coordinate (State Plane)  \
count  3.645580e+05  361560.000000                3.605280e+05
mean   3.106595e+07   10858.496659                1.005043e+06
std    7.331531e+05     578.263114                2.196362e+04
min    2.960737e+07      83.000000                9.133570e+05
25%    3.049938e+07   10314.000000                9.919460e+05
50%    3.108795e+07   11209.000000                1.003470e+06
75%    3.167433e+07   11238.000000                1.019134e+06
max    3.231065e+07   11697.000000                1.067186e+06

       Y Coordinate (State Plane)  School or Citywide Complaint
Vehicle Type  \
count               360528.000000                           0.0
0.0
mean                203425.305782                           NaN
NaN
std                  29842.192857                           NaN
NaN
min                 121185.000000                           NaN
NaN
25%                 182945.000000                           NaN
NaN
50%                 201023.000000                           NaN
NaN
75%                 222790.000000                           NaN
NaN
max                 271876.000000                           NaN
NaN

       Taxi Company Borough  Taxi Pick Up Location  Garage Lot Name  \
count                   0.0                    0.0              0.0
```

```
mean                    NaN                     NaN                NaN
std                     NaN                     NaN                NaN
min                     NaN                     NaN                NaN
25%                     NaN                     NaN                NaN
50%                     NaN                     NaN                NaN
75%                     NaN                     NaN                NaN
max                     NaN                     NaN                NaN

            Latitude      Longitude
count  360528.000000  360528.000000
mean       40.724980     -73.924946
std         0.081907       0.079213
min        40.499040     -74.254937
25%        40.668742     -73.972253
50%        40.718406     -73.930643
75%        40.778166     -73.874098
max        40.912869     -73.700715
```

1.1 UNDERSTANDING THE SHAPE OF THE DATASET

** To identify the shape of a dataset, we can use the. shape attribute in pandas **

** This will output a tuple containing the number of rows and columns in the dataset in the format (rows, columns) **

** My output is (364558, 53)**

```
calls.shape
```

```
(364558, 53)
```

1.2 IDENTIFY VARIABLES WITH NULL VALUES

** To identify variables with null values in a dataset, you can use the isnull() function in pandas **

** This function returns a Boolean Data Frame showing which values are missing (True) and which values are present (False) **

** There are two type we can identify**

** 1.calls.isnull().sum()**

** 2.calls.isna().sum()**

** Total null values from data set also we can find by using this attribute **

** calls.isna().sum().sum()**

```
null_counts=calls.isnull().sum()
```

```
print("Variables with null values:\n\n",null_counts[null_counts > 0])
```

```
Variables with null values:

 Closed Date                         2381
Descriptor                          6501
Location Type                        133
Incident Zip                        2998
Incident Address                   51699
Street Name                        51699
Cross Street 1                     57188
Cross Street 2                     57805
Intersection Street 1             313438
Intersection Street 2             314046
Address Type                        3252
City                                2997
Landmark                          364183
Facility Type                       2389
Due Date                               3
Resolution Action Updated Date      2402
X Coordinate (State Plane)          4030
Y Coordinate (State Plane)          4030
School Region                          1
School Code                            1
School Zip                             1
School or Citywide Complaint      364558
Vehicle Type                      364558
Taxi Company Borough              364558
Taxi Pick Up Location             364558
Bridge Highway Name               364261
Bridge Highway Direction          364261
Road Ramp                         364296
Bridge Highway Segment            364296
Garage Lot Name                   364558
Ferry Direction                   364557
Ferry Terminal Name               364556
Latitude                            4030
Longitude                           4030
Location                            4030
dtype: int64
```

```
calls.isna().sum()
```

```
Unique Key                             0
Created Date                           0
Closed Date                         2381
Agency                                 0
Agency Name                            0
Complaint Type                         0
Descriptor                          6501
Location Type                        133
Incident Zip                        2998
Incident Address                   51699
Street Name                        51699
Cross Street 1                     57188
Cross Street 2                     57805
Intersection Street 1             313438
Intersection Street 2             314046
Address Type                        3252
City                                2997
Landmark                          364183
Facility Type                       2389
Status                                 0
Due Date                               3
Resolution Description                 0
Resolution Action Updated Date      2402
Community Board                        0
Borough                                0
X Coordinate (State Plane)          4030
Y Coordinate (State Plane)          4030
Park Facility Name                     0
Park Borough                           0
School Name                            0
School Number                          0
School Region                          1
School Code                            1
School Phone Number                    0
School Address                         0
School City                            0
School State                           0
School Zip                             1
School Not Found                       0
School or Citywide Complaint      364558
Vehicle Type                      364558
Taxi Company Borough              364558
Taxi Pick Up Location             364558
Bridge Highway Name               364261
Bridge Highway Direction          364261
Road Ramp                         364296
Bridge Highway Segment            364296
Garage Lot Name                   364558
```

```
Ferry Direction                   364557
Ferry Terminal Name               364556
Latitude                            4030
Longitude                           4030
Location                            4030
dtype: int64
```

```
calls.isna().sum().sum()
```

```
5262284
```

2. PERFORM BASIC DATA EXPLORATORY ANALYSIS

2.1 UTILIZE MISSING VALUE TREATMENT

** Total 14 columns are having almost Null values (Unnecessary Column)**

** we should remove those columns before treating null value using dropna or fillna**

** Because if use dropna all rows are deleting**

** Example calls['Garage Lot Name'] has 364558 null values if we use dropna pandas attribute all row were deleting **

** Eventually we could not able to Imputing missing values because unnecessarily added with our dataset **

Step- 1

>>> So below some self-study technique I used

***>>> Cut-off values should occur or must be remove the column**

>>> Total row * 90% = result

>>> 364558 * 90% = 328102.2 data must be occur in columns or otherwise we will remove the column to get better result

>>> Example 'Taxi Company Borough' has na values 300698 absolutly looks useless column

```
calls.drop(['Intersection Street 1'],axis=1,inplace=True)
calls.drop(['Intersection Street 2'],axis=1,inplace=True)
calls.drop(['Landmark'],axis=1,inplace=True)
calls.drop(['School or Citywide Complaint'],axis=1,inplace=True)
calls.drop(['Vehicle Type'],axis=1,inplace=True)
calls.drop(['Taxi Company Borough'],axis=1,inplace=True)
calls.drop(['Taxi Pick Up Location'],axis=1,inplace=True)
calls.drop(['Bridge Highway Name'],axis=1,inplace=True)
calls.drop(['Bridge Highway Direction'],axis=1,inplace=True)
calls.drop(['Road Ramp'],axis=1,inplace=True)
calls.drop(['Bridge Highway Segment'],axis=1,inplace=True)
calls.drop(['Garage Lot Name'],axis=1,inplace=True)
calls.drop(['Ferry Direction'],axis=1,inplace=True)
calls.drop(['Ferry Terminal Name'],axis=1,inplace=True)
```

```
# deleted some useless columns eariler it was 53 columns now only 39
useful columns
calls.shape
```

```
(364558, 39)
```

```
calls.isna().sum()
```

```
Unique Key                            0
Created Date                          0
Closed Date                        2381
Agency                                0
Agency Name                           0
Complaint Type                        0
Descriptor                         6501
Location Type                       133
Incident Zip                       2998
Incident Address                  51699
Street Name                       51699
Cross Street 1                    57188
Cross Street 2                    57805
Address Type                       3252
City                               2997
Facility Type                      2389
Status                                0
Due Date                              3
Resolution Description                0
Resolution Action Updated Date     2402
Community Board                       0
Borough                               0
X Coordinate (State Plane)         4030
Y Coordinate (State Plane)         4030
Park Facility Name                    0
Park Borough                          0
School Name                           0
School Number                         0
School Region                         1
School Code                           1
School Phone Number                   0
School Address                        0
School City                           0
School State                          0
School Zip                            1
School Not Found                      0
Latitude                           4030
Longitude                          4030
Location                           4030
dtype: int64
```

focusing Na values for filling values

>>> I devided my Data set into two types one Numerical columns and another one is Categorical columns

>>> for Categorical Na values I am using filna by mode function (attribute from pandas)

>>> for Numerical Na values I am using filna by Interpolate function (attribute from pandas)

```
# this helps to understand how many columns are numerical and other
statistics details
calls.describe()
```

```
         Unique Key   Incident Zip  X Coordinate (State Plane)  \
count  3.645580e+05  361560.000000                3.605280e+05   
mean   3.106595e+07   10858.496659                1.005043e+06   
std    7.331531e+05     578.263114                2.196362e+04   
min    2.960737e+07      83.000000                9.133570e+05   
25%    3.049938e+07   10314.000000                9.919460e+05   
50%    3.108795e+07   11209.000000                1.003470e+06   
75%    3.167433e+07   11238.000000                1.019134e+06   
max    3.231065e+07   11697.000000                1.067186e+06   

       Y Coordinate (State Plane)       Latitude      Longitude  
count               360528.000000  360528.000000  360528.000000  
mean                203425.305782      40.724980     -73.924946  
std                  29842.192857       0.081907       0.079213  
min                 121185.000000      40.499040     -74.254937  
25%                 182945.000000      40.668742     -73.972253  
50%                 201023.000000      40.718406     -73.930643  
75%                 222790.000000      40.778166     -73.874098  
max                 271876.000000      40.912869     -73.700715  
```

```
# Seperating Na values by (Numerical and Categerical) by using
describe () function
# apart from statistical columns I considerd time columns and
categorical columns

numerical_Na_Values = calls[['Incident Zip','X Coordinate (State
Plane)','Y Coordinate (State Plane)','Latitude','Longitude']]

categorical_Na_Values = calls[['Closed Date','Descriptor','Location
Type','Incident Address', 'Street Name', 'Cross Street 1',
                               'Cross Street 2','Address Type',
'City', 'Facility Type','Due Date','Resolution Description',
                               'School Region', 'School
Code','Location']]
```

```
numerical_Na_Values.head(3)
```

```
   Incident Zip  X Coordinate (State Plane)  Y Coordinate (State
Plane)  \
0       10034.0                   1005409.0
254678.0
1       11105.0                   1007766.0
221986.0
2       10458.0                   1015081.0
256380.0

    Latitude  Longitude
0  40.865682 -73.923501
1  40.775945 -73.915094
2  40.870325 -73.888525
```

```
categorical_Na_Values.head(1)
```

```
              Closed Date        Descriptor    Location Type  \
0  01/01/2016 12:55:15 AM  Loud Music/Party  Street/Sidewalk

      Incident Address       Street Name  Cross Street 1   Cross
Street 2  \
0  71 VERMILYEA AVENUE  VERMILYEA AVENUE  ACADEMY STREET  WEST 204
STREET

  Address Type      City Facility Type                Due Date  \
0      ADDRESS  NEW YORK      Precinct  01/01/2016 07:59:45 AM

                              Resolution Description School Region  \
0  The Police Department responded and upon arriv...   Unspecified
```

```
   School Code                                 Location
0  Unspecified  (40.86568153633767, -73.92350095571744)
```

Closed date has Na values It is really import column to find the Average Response Time if using Interpolate function for numerical could not able get accurate result so that in Closed Date Na value removed from the data set

```
calls = calls.dropna(subset=['Closed Date'])

calls.head(1)
```

```
   Unique Key            Created Date             Closed Date
Agency  \
0    32310363  12/31/2015 11:59:45 PM  01/01/2016 12:55:15 AM   NYPD


                       Agency Name           Complaint Type
Descriptor  \
0  New York City Police Department  Noise - Street/Sidewalk  Loud
Music/Party

     Location Type  Incident Zip     Incident Address  ...   School
Code  \
0  Street/Sidewalk       10034.0  71 VERMILYEA AVENUE  ...
Unspecified

  School Phone Number School Address  School City School State
School Zip  \
0         Unspecified    Unspecified  Unspecified  Unspecified
Unspecified

  School Not Found   Latitude  Longitude  \
0                N  40.865682 -73.923501

                                  Location
0  (40.86568153633767, -73.92350095571744)

[1 rows x 39 columns]
```

```
# Categorical columns are filling by mode function

calls['Descriptor']=
calls['Descriptor'].fillna(calls['Descriptor'].mode()[0])
calls['Location Type']= calls['Location Type'].fillna(calls['Location
Type'].mode()[0])
calls['Street Name']= calls['Street Name'].fillna(calls['Street
Name'].mode()[0])
calls['Incident Address']= calls['Incident
Address'].fillna(calls['Incident Address'].mode()[0])
calls['Cross Street 1']= calls['Cross Street 1'].fillna(calls['Cross
Street 1'].mode()[0])
calls['Cross Street 2']= calls['Cross Street 2'].fillna(calls['Cross
Street 2'].mode()[0])
calls['Address Type']= calls['Address Type'].fillna(calls['Address
Type'].mode()[0])
calls['City']= calls['City'].fillna(calls['City'].mode()[0])
calls['Facility Type']= calls['Facility Type'].fillna(calls['Facility
Type'].mode()[0])
calls['Due Date']= calls['Due Date'].fillna(calls['Due Date'].mode()
[0])
calls['Resolution Action Updated Date']= calls['Resolution Action
Updated Date'].fillna(calls['Resolution Action Updated Date'].mode()
[0])
calls['Resolution Description']= calls['Resolution
Description'].fillna(calls['Resolution Description'].mode()[0])
calls['School Region']= calls['School Region'].fillna(calls['School
Region'].mode()[0])
calls['School Code']= calls['School Code'].fillna(calls['School
Code'].mode()[0])
calls['School Zip']= calls['School Zip'].fillna(calls['School
Zip'].mode()[0])
calls['Location']= calls['Location'].fillna(calls['Location'].mode()
[0])
```

```
calls.shape
```

```
(362177, 39)
```

```
calls.isna().sum()
```

```
Unique Key                            0
Created Date                          0
Closed Date                           0
Agency                                0
Agency Name                           0
Complaint Type                        0
Descriptor                            0
Location Type                         0
Incident Zip                        675
Incident Address                      0
Street Name                           0
Cross Street 1                        0
Cross Street 2                        0
```

```
Address Type                         0
City                                 0
Facility Type                        0
Status                               0
Due Date                             0
Resolution Description               0
Resolution Action Updated Date       0
Community Board                      0
Borough                              0
X Coordinate (State Plane)        1707
Y Coordinate (State Plane)        1707
Park Facility Name                   0
Park Borough                         0
School Name                          0
School Number                        0
School Region                        0
School Code                          0
School Phone Number                  0
School Address                       0
School City                          0
School State                         0
School Zip                           0
School Not Found                     0
Latitude                          1707
Longitude                         1707
Location                             0
dtype: int64
```

```
# Numerical columns are filling by interpolate with linear method
calls.interpolate(method='linear', inplace=True)
```

```
calls.shape
```

```
(362177, 39)
```

```
calls.isna().sum()
```

```
Unique Key                        0
Created Date                      0
Closed Date                       0
Agency                            0
Agency Name                       0
Complaint Type                    0
Descriptor                        0
Location Type                     0
Incident Zip                      0
Incident Address                  0
Street Name                       0
Cross Street 1                    0
Cross Street 2                    0
Address Type                      0
City                              0
Facility Type                     0
Status                            0
Due Date                          0
Resolution Description            0
Resolution Action Updated Date    0
Community Board                   0
Borough                           0
X Coordinate (State Plane)        0
Y Coordinate (State Plane)        0
Park Facility Name                0
Park Borough                      0
School Name                       0
School Number                     0
School Region                     0
School Code                       0
School Phone Number               0
School Address                    0
School City                       0
School State                      0
School Zip                        0
School Not Found                  0
Latitude                          0
Longitude                         0
Location                          0
dtype: int64
```

```
calls.head(1)
```

```
   Unique Key            Created Date             Closed Date
Agency  \
0    32310363  12/31/2015 11:59:45 PM  01/01/2016 12:55:15 AM   NYPD


                       Agency Name           Complaint Type
Descriptor  \
0  New York City Police Department  Noise - Street/Sidewalk  Loud
Music/Party

     Location Type  Incident Zip     Incident Address  ...  School
Code  \
0  Street/Sidewalk       10034.0  71 VERMILYEA AVENUE  ...
Unspecified
```

```
  School Phone Number School Address  School City School State 
School Zip  \
0         Unspecified    Unspecified  Unspecified  Unspecified 
Unspecified   

  School Not Found   Latitude  Longitude  \
0                N  40.865682 -73.923501   

                                  Location  
0  (40.86568153633767, -73.92350095571744)  

[1 rows x 39 columns]
```

2.2 Analyze the date column and remove the entries if it has an incorrect timeline

** There are two columns have date format stored values**

** 1. Created Date**

** 2. Closed Date**

```
# 1. Created Date

# step-1 Convert the date column to datetime format
calls['Created Date'] = pd.to_datetime(calls['Created Date'], 
errors='coerce')



# step - 2 Analyze the date column
print(calls['Created Date'].describe())

count                  362177
unique                 359655
top       2015-08-08 22:16:46
freq                        3
first     2015-01-01 00:00:50
last      2015-12-31 23:59:45
Name: Created Date, dtype: object



# step-3- Checking the the outlier ['string or non identical time'] by 
conditional formatting
# In that data set Created Date column start with 2015-01-01
```

```
00:00:50 ,
# if less than the start data or incorrect date format should be
delete from the dataset as outlier

incorrect_dates = calls[calls['Created Date'] < '2015-01-01 00:00:50']
```

```
# No outlier rows as per output result
incorrect_dates
```

```
Empty DataFrame
Columns: [Unique Key, Created Date, Closed Date, Agency, Agency Name,
Complaint Type, Descriptor, Location Type, Incident Zip, Incident
Address, Street Name, Cross Street 1, Cross Street 2, Address Type,
City, Facility Type, Status, Due Date, Resolution Description,
Resolution Action Updated Date, Community Board, Borough, X Coordinate
(State Plane), Y Coordinate (State Plane), Park Facility Name, Park
Borough, School Name, School Number, School Region, School Code,
School Phone Number, School Address, School City, School State, School
Zip, School Not Found, Latitude, Longitude, Location]
Index: []

[0 rows x 39 columns]
```

```
# step-3- Checking the the outlier ['string or non identical time'] by
conditional formatting
# In that data set Created Date column end with 2015-12-31 23:59:50 ,
# if greater than the start data or incorrect date format should be
delete from the dataset as outlier
incorrect_dates = calls[calls['Created Date'] >= '2015-12-31
23:59:50']
```

```
# No outlier rows as per output result

incorrect_dates
```

```
Empty DataFrame
Columns: [Unique Key, Created Date, Closed Date, Agency, Agency Name,
Complaint Type, Descriptor, Location Type, Incident Zip, Incident
Address, Street Name, Cross Street 1, Cross Street 2, Address Type,
City, Facility Type, Status, Due Date, Resolution Description,
Resolution Action Updated Date, Community Board, Borough, X Coordinate
(State Plane), Y Coordinate (State Plane), Park Facility Name, Park
Borough, School Name, School Number, School Region, School Code,
School Phone Number, School Address, School City, School State, School
Zip, School Not Found, Latitude, Longitude, Location]
```

```
Index: []

[0 rows x 39 columns]
```

```
# 2. Closed Date
# step 1 - Convert the date column to datetime format
calls['Closed Date'] = pd.to_datetime(calls['Closed Date'],
errors='coerce')
```

```
# step - 2 Analyze the date column
print(calls['Closed Date'].describe())
```

```
count                  362177
unique                 339837
top       2015-09-10 07:12:49
freq                        3
first     2015-01-01 00:20:33
last      2016-01-03 16:22:52
Name: Closed Date, dtype: object
```

```
# step-3- Checking the the outlier ['string or non identical time']
by conditional formatting
# In that data set Created Date column end with 2015-01-01 00:20:33 ,
# if less than the start data or incorrect date format should be
delete from the dataset as outlier

incorrect_dates = calls[calls['Closed Date'] < '2015-01-01 00:20:33']
```

```
# No outlier rows as per output result
incorrect_dates
```

```
Empty DataFrame
Columns: [Unique Key, Created Date, Closed Date, Agency, Agency Name,
Complaint Type, Descriptor, Location Type, Incident Zip, Incident
Address, Street Name, Cross Street 1, Cross Street 2, Address Type,
City, Facility Type, Status, Due Date, Resolution Description,
Resolution Action Updated Date, Community Board, Borough, X Coordinate
(State Plane), Y Coordinate (State Plane), Park Facility Name, Park
Borough, School Name, School Number, School Region, School Code,
School Phone Number, School Address, School City, School State, School
Zip, School Not Found, Latitude, Longitude, Location]
Index: []

[0 rows x 39 columns]
```

```
# step-3- Checking the the outlier ['string or non identical time']
by conditional formatting
# In that data set Created Date column end with 2016-01-03 16:22:53 ,
# if greater than the start data or incorrect date format should be
delete from the dataset as outlier
incorrect_dates = calls[calls['Closed Date'] >= '2016-01-03 16:22:53']
```

```
# No outlier rows as per output result
incorrect_dates
```

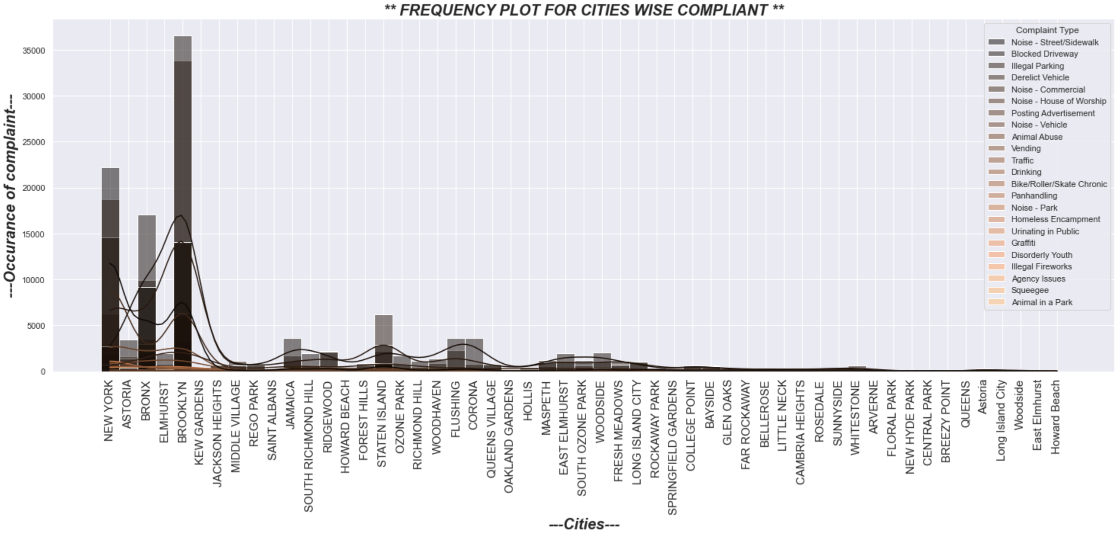
```
Empty DataFrame
Columns: [Unique Key, Created Date, Closed Date, Agency, Agency Name,
Complaint Type, Descriptor, Location Type, Incident Zip, Incident
Address, Street Name, Cross Street 1, Cross Street 2, Address Type,
City, Facility Type, Status, Due Date, Resolution Description,
Resolution Action Updated Date, Community Board, Borough, X Coordinate
(State Plane), Y Coordinate (State Plane), Park Facility Name, Park
Borough, School Name, School Number, School Region, School Code,
School Phone Number, School Address, School City, School State, School
Zip, School Not Found, Latitude, Longitude, Location]
Index: []

[0 rows x 39 columns]
```

2.1.1 Draw a frequency plot for city-wise complaints

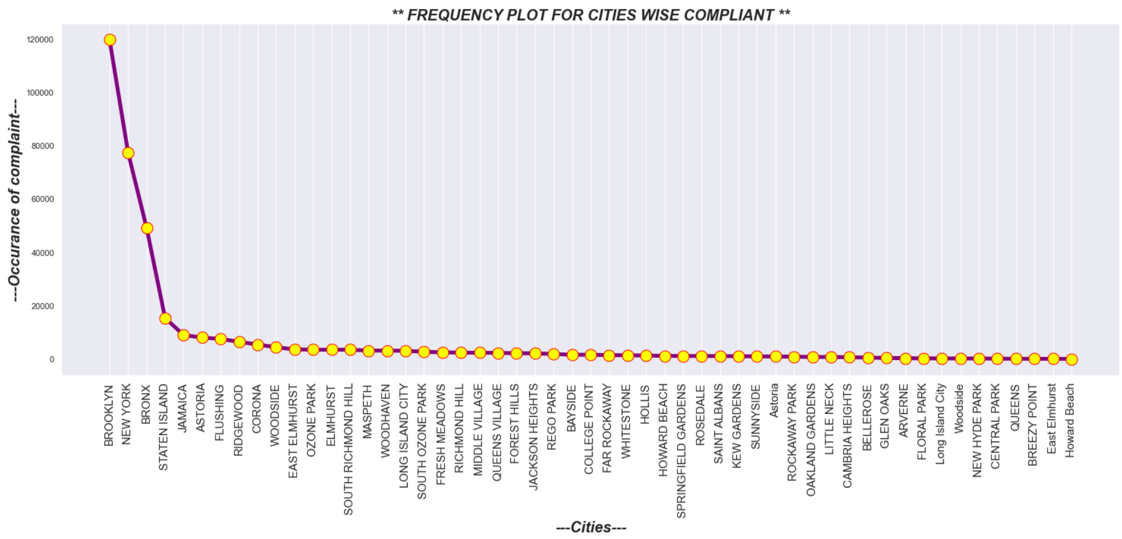
** Sloved this task by two ways**

** Histplot from seaborn and Bar Plot from Matplot.pylot **

```
# font Style for x, y labels and tile
font_style = {'family': 'Arial', 'size': 20, 'weight': 'bold',
'style': 'italic'}

# Suitable figure size by seaborn
sns.set(rc={'figure.figsize':(25,8.5)})

# Histplot can helps to make frequency plot by two categorical columns

sns.histplot(data=calls, x='City', hue='Complaint Type',
palette='copper', alpha=0.5, kde=True)

# pLot represenation by labels
plt.xlabel('---Cities---',fontdict=font_style)
plt.ylabel('---Occurance of complaint---',fontdict=font_style)
plt.title('** FREQUENCY PLOT FOR CITIES WISE COMPLIANT
**',fontdict=font_style)

# the below comment used for showing the x values by vertically
plt.xticks(rotation='vertical', ha='center',size=15)
plt.show()
```



```
# There is one more type we can create a plot the ferquency occurance
for city wise complaint

value_count= calls['City'].value_counts()

font_style = {'family': 'Arial', 'size': 20, 'weight': 'bold',
'style': 'italic'}
plt.figure(figsize=(25,8.5))
plt.plot(value_count,marker ='o',ms=15, mec='red',
mfc='yellow',c='purple',lw=5)
plt.grid(axis='y',ls='solid',color ='k',lw=0.5,alpha=0.5)
plt.xlabel('---Cities---',fontdict=font_style)
plt.ylabel('---Occurance of complaint---',fontdict=font_style)
plt.title('** FREQUENCY PLOT FOR CITIES WISE COMPLIANT
**',fontdict=font_style)
plt.xticks(rotation='vertical', ha='center',size=15)
plt.show()
```



2.2.2 Draw Scatter Plot for Complaint Concentration across Brooklyn

** Steps Involved to Draw a Scatter plot for Concetration across Brooklyn

** step1- one hot labelling for City column with new variable**

** step2- adding only Brooklyn column into original data set**

** step3 -Created a variable and used conditional Brooklyn as 1**

** now we have all columns under Brooklyn city**

** Note - scatter plot will happen basesd on numerical**

** for ploting it require two numerical columns**

** for sns it require x= numerical and y = categorical

** for hexbin will works with only Floting point values**

```
# step1- one hot labelling for City column with new variable
var1 = pd.get_dummies(calls['City'])


# checking purpose
var1.head(2)

   ARVERNE  ASTORIA  Astoria  BAYSIDE  BELLEROSE  BREEZY POINT  BRONX
\
0        0        0        0        0          0             0      0
```

```
1        0        1        0        0          0             0      0


   BROOKLYN  CAMBRIA HEIGHTS  CENTRAL PARK  ...  SAINT ALBANS  \
0         0                0             0  ...             0
1         0                0             0  ...             0

   SOUTH OZONE PARK  SOUTH RICHMOND HILL  SPRINGFIELD GARDENS  STATEN
ISLAND  \
0                 0                    0                    0
0
1                 0                    0                    0
0

   SUNNYSIDE  WHITESTONE  WOODHAVEN  WOODSIDE  Woodside
0          0           0          0         0         0
1          0           0          0         0         0

[2 rows x 53 columns]
```

```
# step2- adding only Brooklyn column into original data set

calls['BROOKLYN_STATE'] = var1['BROOKLYN']
```

```
# step3 -Created a brand new variable and used conditional as Brooklyn
as 1
# now we have all columns under Brooklyn city
select_ones_Brookyn = calls[calls['BROOKLYN_STATE'] >=1]

select_ones_Brookyn.head(3)
```

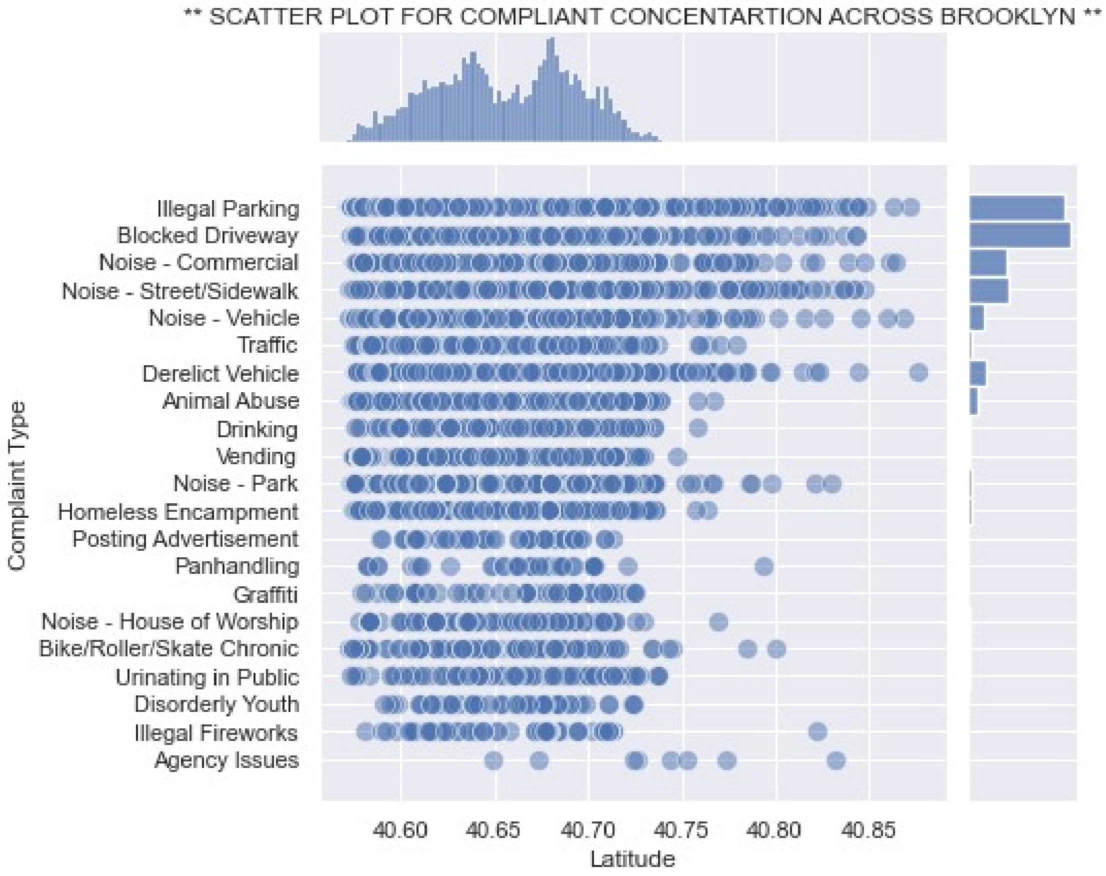
```
    Unique Key        Created Date         Closed Date Agency  \
5     32306554 2015-12-31 23:56:30 2016-01-01 01:50:11   NYPD
9     32308391 2015-12-31 23:53:58 2016-01-01 01:17:40   NYPD
13    32305074 2015-12-31 23:47:58 2016-01-01 08:18:47   NYPD

                        Agency Name    Complaint Type  \
5   New York City Police Department   Illegal Parking
9   New York City Police Department  Blocked Driveway
13  New York City Police Department   Illegal Parking

                       Descriptor    Location Type  Incident Zip  \
5   Posted Parking Sign Violation  Street/Sidewalk       11215.0
9                       No Access  Street/Sidewalk       11219.0
13  Posted Parking Sign Violation  Street/Sidewalk       11208.0
```

```
   Incident Address  ... School Phone Number School Address  School
City  \
5     260 21 STREET  ...         Unspecified    Unspecified
Unspecified
9    1408 66 STREET  ...         Unspecified    Unspecified
Unspecified
13     38 COX PLACE  ...         Unspecified    Unspecified
Unspecified

   School State   School Zip School Not Found   Latitude  Longitude  \
5   Unspecified  Unspecified                N  40.660823 -73.992568
9   Unspecified  Unspecified                N  40.623793 -73.999539
13  Unspecified  Unspecified                N  40.687511 -73.874505

                                    Location BROOKLYN_STATE
5    (40.66082272389114, -73.99256786342693)              1
9   (40.623793065806524, -73.99953890121567)              1
13   (40.68751060232221, -73.87450451131276)              1

[3 rows x 40 columns]
```

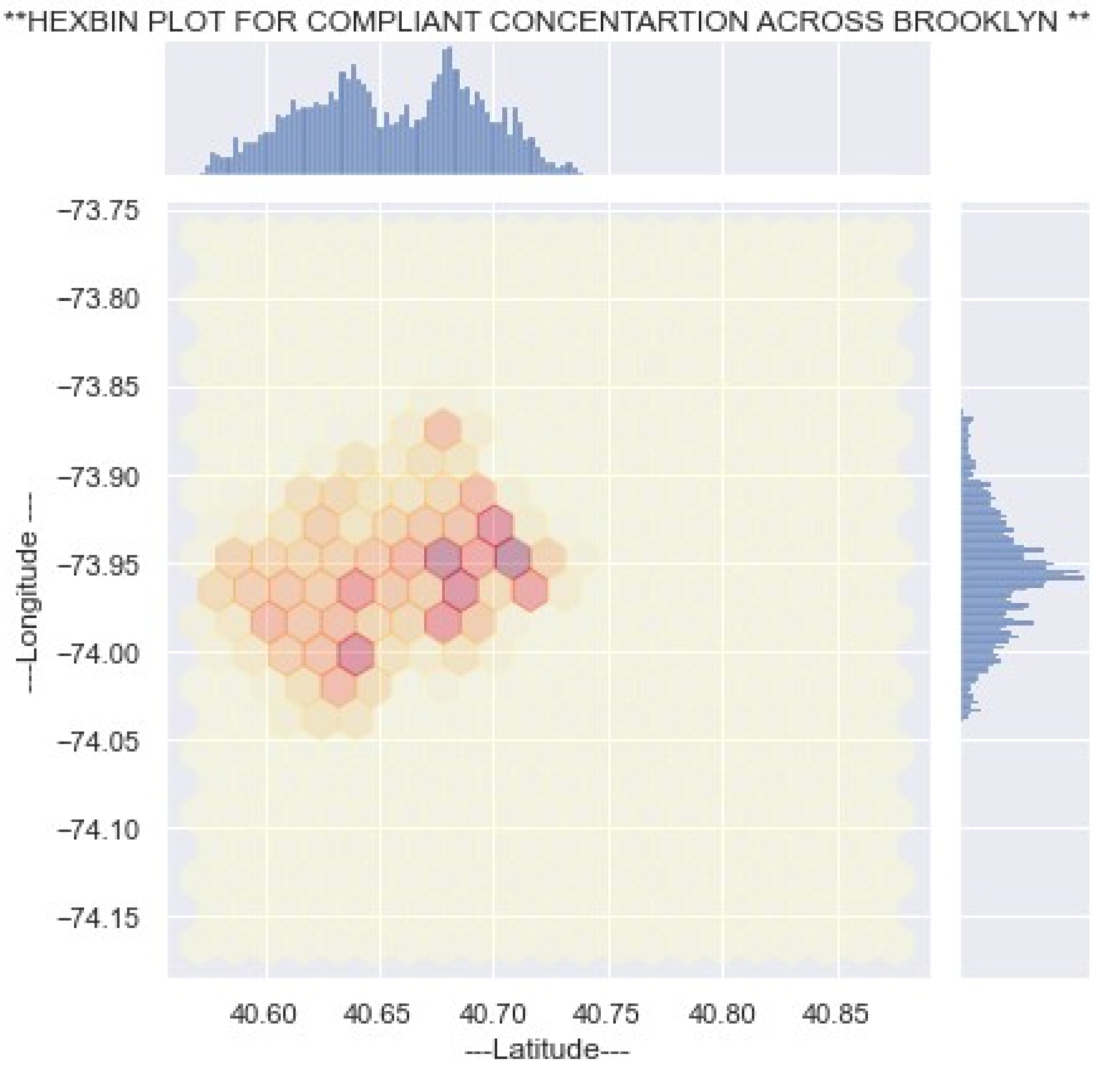
```
select_ones_Brookyn.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 119523 entries, 5 to 364546
Data columns (total 40 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   Unique Key                      119523 non-null  int64
 1   Created Date                    119523 non-null  datetime64[ns]
 2   Closed Date                     119523 non-null  datetime64[ns]
 3   Agency                          119523 non-null  object
 4   Agency Name                     119523 non-null  object
 5   Complaint Type                  119523 non-null  object
 6   Descriptor                      119523 non-null  object
 7   Location Type                   119523 non-null  object
 8   Incident Zip                    119523 non-null  float64
 9   Incident Address                119523 non-null  object
 10  Street Name                     119523 non-null  object
 11  Cross Street 1                  119523 non-null  object
 12  Cross Street 2                  119523 non-null  object
 13  Address Type                    119523 non-null  object
 14  City                            119523 non-null  object
 15  Facility Type                   119523 non-null  object
 16  Status                          119523 non-null  object
 17  Due Date                        119523 non-null  object
 18  Resolution Description          119523 non-null  object
 19  Resolution Action Updated Date  119523 non-null  object
 20  Community Board                 119523 non-null  object
 21  Borough                         119523 non-null  object
 22  X Coordinate (State Plane)      119523 non-null  float64
 23  Y Coordinate (State Plane)      119523 non-null  float64
 24  Park Facility Name              119523 non-null  object
 25  Park Borough                    119523 non-null  object
 26  School Name                     119523 non-null  object
 27  School Number                   119523 non-null  object
 28  School Region                   119523 non-null  object
 29  School Code                     119523 non-null  object
 30  School Phone Number             119523 non-null  object
 31  School Address                  119523 non-null  object
 32  School City                     119523 non-null  object
 33  School State                    119523 non-null  object
 34  School Zip                      119523 non-null  object
 35  School Not Found                119523 non-null  object
 36  Latitude                        119523 non-null  float64
 37  Longitude                       119523 non-null  float64
 38  Location                        119523 non-null  object
 39  BROOKLYN_STATE                  119523 non-null  uint8
dtypes: datetime64[ns](2), float64(5), int64(1), object(31), uint8(1)
memory usage: 36.6+ MB
```

```
# font Style for x, y labels and tile
font_style = {'family': 'Arial', 'size': 20, 'weight': 'bold',
'style': 'italic'}

plt.rcParams['figure.figsize'] = [15, 6]

# JointPlot can helps to create scatter plot by One Numerical col and
one categorical columns
sns.jointplot(x='Latitude',y ='Complaint
Type',data=select_ones_Brookyn, kind='scatter',palette
='husl',s=100,alpha=0.5)

# pLot represenation by labels
plt.xlabel('---Latitude---',size=20)
plt.ylabel('---Occurance of complaint Type---',size=20)
plt.title('** SCATTER PLOT FOR COMPLIANT CONCENTARTION ACROSS BROOKLYN
**', x=-3, y=1.2)
plt.xticks(rotation=90)
plt.show()
```



```
# Hexa bin will works only in float type value columns
sns.jointplot(x='Latitude', y ='Longitude',data=select_ones_Brookyn,
kind='hex', gridsize=20, cmap='YlOrRd',alpha=0.345)
plt.xlabel('---Latitude---',size=12)
plt.ylabel('---Longitude ---',size=12)
plt.title('**HEXBIN PLOT FOR COMPLIANT CONCENTARTION ACROSS BROOKLYN
**',y=1.2)
plt.show()
```



**HEXBIN PLOT FOR COMPLIANT CONCENTARTION ACROSS BROOKLYN **

3. Find major types of complaints**

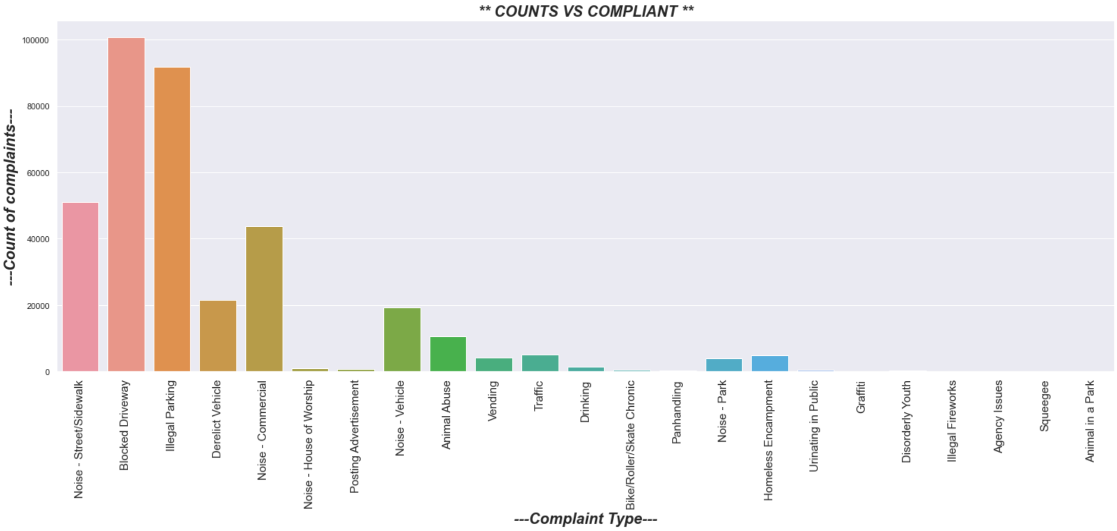
3.1 Plot Bar Graph of Count vs Compliant Types

** To plot a bar graph of count versus categorical column values **

** We can use seaborn's countplot() function **

```
# font Style for x, y labels and tile
font_style = {'family': 'Arial', 'size': 20, 'weight': 'bold',
'style': 'italic'}

# Suitable figure size by seaborn
sns.set(rc={'figure.figsize':(25,8.5)})
```

```
# countplot can helps to make frequency plot by one categorical
columns
sns.countplot(x="Complaint Type", data=calls)

# pLot represenation by labels
plt.xlabel('---Complaint Type---',fontdict=font_style)
plt.ylabel('---Count of complaints---',fontdict=font_style)
plt.title('** COUNTS VS COMPLIANT **',fontdict=font_style)

# the below comment used for showing the x values by vertically
plt.xticks(rotation='vertical', ha='center',size=15)
plt.show()
```



3.2 Find the top 10 types of complaints

** We can also use the nlargest() function to find the top n types from a categorical column in a pandas dataframe **

```
top_10_values = calls['Complaint Type'].value_counts().nlargest(10)
```

```
top_10_values
```

```
Blocked Driveway           100624
Illegal Parking             91716
Noise - Street/Sidewalk     51139
Noise - Commercial          43751
Derelict Vehicle            21518
Noise - Vehicle             19301
Animal Abuse                10530
Traffic                      5196
Homeless Encampment          4879
```

```
Vending                       4185
Name: Complaint Type, dtype: int64
```

3.3 Display the types of complaints in each city in separate dataset

** Steps Involved for this task**

** step 1- we need to create group by for city column**

** step 2- Create a dictionary of DataFrames, with one DataFrame for each category**

** step 3- Print the unique subcategories for each category**

** step 4 -filling na values by using numpy.np**

** (could not able create dataframe due to same length value error) **

** step 5 - Creating Dataframe from dictionary key value pair along with nan by pandas as new data set**

```
# step 1- we need to create group by for city column
groups = calls.groupby('City')

# step 2- Create a dictionary of DataFrames, with one DataFrame for
each category
data = {category: group['Complaint Type'].unique() for category, group
in groups}

# step 3- Print the unique subcategories for each category
for category, subcategories in data.items():
    print(f'{category}:')
    print(subcategories)
```

```
ARVERNE:
['Illegal Parking' 'Noise - Commercial' 'Animal Abuse' 'Blocked
Driveway'
 'Derelict Vehicle' 'Noise - Street/Sidewalk' 'Homeless Encampment'
 'Urinating in Public' 'Noise - Vehicle' 'Noise - House of Worship'
 'Panhandling' 'Vending' 'Disorderly Youth' 'Noise - Park' 'Drinking'
 'Graffiti' 'Traffic']
ASTORIA:
['Blocked Driveway' 'Noise - Commercial' 'Noise - Vehicle'
 'Illegal Parking' 'Noise - Street/Sidewalk' 'Bike/Roller/Skate
Chronic'
 'Animal Abuse' 'Derelict Vehicle' 'Drinking' 'Homeless Encampment'
 'Urinating in Public' 'Traffic' 'Noise - House of Worship' 'Vending'
 'Noise - Park' 'Illegal Fireworks' 'Disorderly Youth' 'Panhandling'
 'Graffiti' 'Posting Advertisement']
Astoria:
```

```
['Illegal Parking' 'Noise - Street/Sidewalk' 'Derelict Vehicle'
 'Blocked Driveway' 'Noise - Commercial']
BAYSIDE:
['Blocked Driveway' 'Derelict Vehicle' 'Illegal Parking'
 'Noise - Street/Sidewalk' 'Noise - Vehicle' 'Animal Abuse'
 'Noise - Commercial' 'Homeless Encampment' 'Graffiti' 'Noise - Park'
 'Traffic' 'Disorderly Youth' 'Noise - House of Worship' 'Vending'
 'Drinking']
BELLEROSE:
['Derelict Vehicle' 'Blocked Driveway' 'Noise - Street/Sidewalk'
 'Noise - Commercial' 'Illegal Parking' 'Animal Abuse' 'Traffic'
 'Homeless Encampment' 'Urinating in Public' 'Noise - Vehicle'
 'Bike/Roller/Skate Chronic' 'Disorderly Youth' 'Panhandling'
 'Noise - Park' 'Illegal Fireworks' 'Noise - House of Worship'
 'Posting Advertisement' 'Drinking']
BREEZY POINT:
['Noise - Street/Sidewalk' 'Blocked Driveway' 'Animal Abuse'
 'Noise - Commercial' 'Noise - Vehicle' 'Illegal Parking' 'Drinking'
 'Derelict Vehicle']
BRONX:
['Blocked Driveway' 'Illegal Parking' 'Noise - Street/Sidewalk'
 'Noise - Vehicle' 'Noise - Commercial' 'Derelict Vehicle' 'Animal
Abuse'
 'Vending' 'Traffic' 'Drinking' 'Bike/Roller/Skate Chronic' 'Noise -
Park'
 'Homeless Encampment' 'Noise - House of Worship' 'Disorderly Youth'
 'Urinating in Public' 'Panhandling' 'Graffiti' 'Illegal Fireworks'
 'Posting Advertisement']
BROOKLYN:
['Illegal Parking' 'Blocked Driveway' 'Noise - Commercial'
 'Noise - Street/Sidewalk' 'Noise - Vehicle' 'Traffic' 'Derelict
Vehicle'
 'Animal Abuse' 'Drinking' 'Vending' 'Noise - Park' 'Homeless
Encampment'
 'Posting Advertisement' 'Panhandling' 'Graffiti'
 'Noise - House of Worship' 'Bike/Roller/Skate Chronic'
 'Urinating in Public' 'Disorderly Youth' 'Illegal Fireworks'
 'Agency Issues']
CAMBRIA HEIGHTS:
['Derelict Vehicle' 'Blocked Driveway' 'Noise - Commercial'
 'Noise - Vehicle' 'Illegal Parking' 'Homeless Encampment'
 'Noise - Street/Sidewalk' 'Animal Abuse' 'Traffic' 'Illegal
Fireworks'
 'Noise - House of Worship']
CENTRAL PARK:
['Noise - Street/Sidewalk' 'Illegal Parking']
COLLEGE POINT:
['Illegal Parking' 'Blocked Driveway' 'Derelict Vehicle'
 'Noise - Street/Sidewalk' 'Noise - Vehicle' 'Noise - Commercial'
 'Traffic' 'Animal Abuse' 'Noise - Park' 'Homeless Encampment'
```

```
'Vending'
 'Graffiti' 'Disorderly Youth' 'Drinking' 'Noise - House of Worship']
CORONA:
['Blocked Driveway' 'Illegal Parking' 'Urinating in Public'
 'Noise - Commercial' 'Noise - Street/Sidewalk' 'Vending' 'Animal
Abuse'
 'Noise - Vehicle' 'Drinking' 'Derelict Vehicle'
 'Noise - House of Worship' 'Homeless Encampment' 'Traffic'
 'Disorderly Youth' 'Noise - Park' 'Posting Advertisement'
'Panhandling'
 'Graffiti']
EAST ELMHURST:
['Noise - House of Worship' 'Blocked Driveway' 'Illegal Parking'
 'Derelict Vehicle' 'Traffic' 'Noise - Commercial'
 'Noise - Street/Sidewalk' 'Noise - Vehicle' 'Animal Abuse' 'Vending'
 'Homeless Encampment' 'Urinating in Public' 'Drinking'
 'Posting Advertisement' 'Graffiti' 'Noise - Park' 'Disorderly Youth'
 'Bike/Roller/Skate Chronic']
ELMHURST:
['Illegal Parking' 'Blocked Driveway' 'Noise - Vehicle' 'Drinking'
 'Noise - Commercial' 'Noise - Street/Sidewalk' 'Traffic'
 'Homeless Encampment' 'Derelict Vehicle' 'Animal Abuse' 'Vending'
 'Noise - Park' 'Urinating in Public' 'Noise - House of Worship'
 'Panhandling' 'Bike/Roller/Skate Chronic' 'Illegal Fireworks'
 'Disorderly Youth' 'Posting Advertisement' 'Graffiti']
East Elmhurst:
['Illegal Parking' 'Derelict Vehicle']
FAR ROCKAWAY:
['Blocked Driveway' 'Illegal Parking' 'Animal Abuse' 'Noise - Vehicle'
 'Noise - Street/Sidewalk' 'Derelict Vehicle' 'Noise - Commercial'
 'Homeless Encampment' 'Noise - House of Worship' 'Noise - Park'
'Traffic'
 'Vending' 'Drinking' 'Disorderly Youth' 'Urinating in Public']
FLORAL PARK:
['Illegal Parking' 'Derelict Vehicle' 'Blocked Driveway' 'Drinking'
 'Animal Abuse' 'Noise - Commercial' 'Noise - Vehicle'
 'Noise - Street/Sidewalk' 'Disorderly Youth']
FLUSHING:
['Blocked Driveway' 'Illegal Parking' 'Derelict Vehicle' 'Traffic'
 'Vending' 'Noise - Vehicle' 'Noise - Commercial'
 'Noise - Street/Sidewalk' 'Animal Abuse' 'Drinking' 'Homeless
Encampment'
 'Noise - Park' 'Posting Advertisement' 'Graffiti' 'Illegal Fireworks'
 'Urinating in Public' 'Panhandling' 'Noise - House of Worship'
 'Disorderly Youth' 'Bike/Roller/Skate Chronic']
FOREST HILLS:
['Illegal Parking' 'Noise - Commercial' 'Blocked Driveway'
 'Noise - Vehicle' 'Animal Abuse' 'Noise - Street/Sidewalk'
 'Derelict Vehicle' 'Traffic' 'Vending' 'Bike/Roller/Skate Chronic'
 'Noise - Park' 'Panhandling' 'Homeless Encampment'
 'Posting Advertisement' 'Graffiti' 'Disorderly Youth'
 'Urinating in Public' 'Drinking' 'Illegal Fireworks'
 'Noise - House of Worship']
FRESH MEADOWS:
['Blocked Driveway' 'Illegal Parking' 'Noise - Vehicle' 'Derelict
Vehicle'
 'Animal Abuse' 'Noise - Street/Sidewalk' 'Traffic' 'Noise - Park'
 'Noise - Commercial' 'Homeless Encampment' 'Urinating in Public'
 'Drinking' 'Panhandling' 'Vending']
GLEN OAKS:
['Illegal Parking' 'Derelict Vehicle' 'Noise - Commercial'
 'Blocked Driveway' 'Noise - Park' 'Noise - Vehicle'
 'Noise - Street/Sidewalk' 'Traffic' 'Vending' 'Urinating in Public'
 'Animal Abuse']
HOLLIS:
['Blocked Driveway' 'Illegal Parking' 'Derelict Vehicle' 'Noise -
Vehicle'
 'Noise - House of Worship' 'Noise - Street/Sidewalk' 'Noise -
Commercial'
 'Animal Abuse' 'Traffic' 'Homeless Encampment' 'Drinking' 'Noise -
Park'
 'Urinating in Public' 'Disorderly Youth']
HOWARD BEACH:
['Illegal Parking' 'Blocked Driveway' 'Noise - Commercial'
 'Derelict Vehicle' 'Animal Abuse' 'Panhandling' 'Traffic'
 'Noise - Street/Sidewalk' 'Noise - Park' 'Vending' 'Noise - Vehicle'
 'Homeless Encampment' 'Disorderly Youth' 'Illegal Fireworks'
'Drinking'
 'Noise - House of Worship' 'Bike/Roller/Skate Chronic']
Howard Beach:
['Blocked Driveway']
JACKSON HEIGHTS:
['Blocked Driveway' 'Noise - House of Worship' 'Noise - Commercial'
 'Illegal Parking' 'Traffic' 'Noise - Street/Sidewalk'
 'Homeless Encampment' 'Vending' 'Noise - Park' 'Noise - Vehicle'
 'Animal Abuse' 'Drinking' 'Bike/Roller/Skate Chronic' 'Derelict
Vehicle'
 'Urinating in Public' 'Illegal Fireworks' 'Panhandling'
 'Posting Advertisement' 'Graffiti']
JAMAICA:
['Blocked Driveway' 'Illegal Parking' 'Animal Abuse' 'Derelict
Vehicle'
 'Traffic' 'Noise - Commercial' 'Homeless Encampment' 'Vending'
 'Noise - Street/Sidewalk' 'Noise - Vehicle' 'Urinating in Public'
 'Noise - Park' 'Posting Advertisement' 'Drinking'
 'Bike/Roller/Skate Chronic' 'Noise - House of Worship' 'Disorderly
Youth'
 'Illegal Fireworks' 'Panhandling' 'Graffiti']
KEW GARDENS:
['Illegal Parking' 'Animal Abuse' 'Blocked Driveway' 'Noise - Vehicle'
```

```
 'Derelict Vehicle' 'Noise - Commercial' 'Traffic'
 'Noise - Street/Sidewalk' 'Drinking' 'Vending' 'Homeless Encampment'
 'Urinating in Public' 'Noise - House of Worship' 'Posting
Advertisement']
LITTLE NECK:
['Blocked Driveway' 'Illegal Parking' 'Traffic' 'Derelict Vehicle'
 'Noise - Vehicle' 'Noise - Commercial' 'Animal Abuse'
 'Urinating in Public' 'Noise - Street/Sidewalk' 'Noise - Park'
 'Disorderly Youth' 'Drinking' 'Posting Advertisement']
LONG ISLAND CITY:
['Illegal Parking' 'Blocked Driveway' 'Noise - Commercial'
 'Noise - Street/Sidewalk' 'Derelict Vehicle' 'Vending' 'Noise - Park'
 'Noise - Vehicle' 'Traffic' 'Drinking' 'Animal Abuse' 'Graffiti'
 'Urinating in Public' 'Posting Advertisement' 'Homeless Encampment'
 'Panhandling' 'Disorderly Youth' 'Bike/Roller/Skate Chronic']
Long Island City:
['Illegal Parking' 'Noise - Commercial' 'Derelict Vehicle'
 'Blocked Driveway' 'Noise - Street/Sidewalk']
MASPETH:
['Illegal Parking' 'Blocked Driveway' 'Urinating in Public'
 'Noise - Street/Sidewalk' 'Noise - Commercial' 'Derelict Vehicle'
 'Traffic' 'Drinking' 'Animal Abuse' 'Noise - Vehicle'
 'Homeless Encampment' 'Bike/Roller/Skate Chronic' 'Noise - Park'
 'Disorderly Youth' 'Vending' 'Noise - House of Worship'
 'Illegal Fireworks' 'Graffiti']
MIDDLE VILLAGE:
['Derelict Vehicle' 'Illegal Parking' 'Blocked Driveway' 'Noise -
Vehicle'
 'Animal Abuse' 'Noise - Street/Sidewalk' 'Traffic' 'Noise -
Commercial'
 'Homeless Encampment' 'Noise - Park' 'Bike/Roller/Skate Chronic'
 'Drinking']
NEW HYDE PARK:
['Derelict Vehicle' 'Blocked Driveway' 'Illegal Parking' 'Noise -
Vehicle'
 'Animal Abuse' 'Noise - Commercial']
NEW YORK:
['Noise - Street/Sidewalk' 'Illegal Parking' 'Noise - House of
Worship'
 'Noise - Commercial' 'Blocked Driveway' 'Vending' 'Noise - Vehicle'
 'Panhandling' 'Animal Abuse' 'Noise - Park' 'Homeless Encampment'
 'Traffic' 'Derelict Vehicle' 'Drinking' 'Urinating in Public'
 'Bike/Roller/Skate Chronic' 'Graffiti' 'Posting Advertisement'
 'Disorderly Youth' 'Illegal Fireworks' 'Squeegee']
OAKLAND GARDENS:
['Blocked Driveway' 'Illegal Parking' 'Noise - Vehicle' 'Derelict
Vehicle'
 'Noise - Street/Sidewalk' 'Traffic' 'Animal Abuse' 'Noise - Park'
 'Vending' 'Drinking' 'Bike/Roller/Skate Chronic' 'Homeless
Encampment'
```

```
 'Disorderly Youth' 'Noise - Commercial']
OZONE PARK:
['Blocked Driveway' 'Illegal Parking' 'Animal Abuse' 'Derelict
Vehicle'
 'Homeless Encampment' 'Noise - Vehicle' 'Noise - Commercial'
'Traffic'
 'Noise - Street/Sidewalk' 'Drinking' 'Panhandling' 'Urinating in
Public'
 'Bike/Roller/Skate Chronic' 'Disorderly Youth' 'Noise - House of
Worship'
 'Noise - Park' 'Illegal Fireworks' 'Posting Advertisement' 'Vending']
QUEENS:
['Noise - Commercial' 'Illegal Parking' 'Noise - Street/Sidewalk'
 'Noise - House of Worship' 'Derelict Vehicle' 'Blocked Driveway'
 'Traffic' 'Homeless Encampment' 'Noise - Vehicle' 'Urinating in
Public'
 'Animal in a Park' 'Animal Abuse']
QUEENS VILLAGE:
['Animal Abuse' 'Blocked Driveway' 'Illegal Parking' 'Noise - Vehicle'
 'Derelict Vehicle' 'Noise - Commercial' 'Panhandling'
 'Homeless Encampment' 'Noise - Street/Sidewalk' 'Traffic'
 'Urinating in Public' 'Vending' 'Noise - House of Worship' 'Drinking'
 'Noise - Park' 'Illegal Fireworks' 'Graffiti' 'Posting
Advertisement']
REGO PARK:
['Blocked Driveway' 'Derelict Vehicle' 'Illegal Parking' 'Noise -
Vehicle'
 'Noise - Commercial' 'Drinking' 'Noise - Park' 'Noise -
Street/Sidewalk'
 'Traffic' 'Animal Abuse' 'Graffiti' 'Noise - House of Worship'
 'Homeless Encampment' 'Vending' 'Urinating in Public']
RICHMOND HILL:
['Blocked Driveway' 'Illegal Parking' 'Drinking' 'Noise - Vehicle'
 'Derelict Vehicle' 'Noise - Commercial' 'Animal Abuse'
 'Noise - Street/Sidewalk' 'Homeless Encampment' 'Vending'
 'Posting Advertisement' 'Traffic' 'Noise - Park' 'Illegal Fireworks'
 'Graffiti' 'Urinating in Public']
RIDGEWOOD:
['Blocked Driveway' 'Illegal Parking' 'Noise - Vehicle'
 'Noise - Commercial' 'Derelict Vehicle' 'Noise - Street/Sidewalk'
 'Animal Abuse' 'Drinking' 'Vending' 'Noise - Park' 'Traffic'
 'Bike/Roller/Skate Chronic' 'Homeless Encampment' 'Urinating in
Public'
 'Illegal Fireworks' 'Graffiti' 'Noise - House of Worship'
 'Posting Advertisement' 'Disorderly Youth']
ROCKAWAY PARK:
['Blocked Driveway' 'Animal Abuse' 'Illegal Parking'
 'Noise - Street/Sidewalk' 'Noise - Commercial' 'Vending' 'Drinking'
 'Noise - Vehicle' 'Derelict Vehicle' 'Homeless Encampment' 'Noise -
Park'
```

```
 'Traffic' 'Disorderly Youth' 'Urinating in Public']
ROSEDALE:
['Animal Abuse' 'Illegal Parking' 'Blocked Driveway' 'Derelict
Vehicle'
 'Traffic' 'Noise - Vehicle' 'Vending' 'Noise - Street/Sidewalk'
 'Drinking' 'Noise - Commercial' 'Homeless Encampment' 'Noise - Park'
 'Graffiti' 'Bike/Roller/Skate Chronic' 'Noise - House of Worship']
SAINT ALBANS:
['Blocked Driveway' 'Illegal Parking' 'Derelict Vehicle' 'Animal
Abuse'
 'Noise - Vehicle' 'Noise - Commercial' 'Noise - Street/Sidewalk'
 'Traffic' 'Vending' 'Disorderly Youth' 'Drinking' 'Homeless
Encampment'
 'Noise - Park' 'Urinating in Public' 'Noise - House of Worship']
SOUTH OZONE PARK:
['Blocked Driveway' 'Illegal Parking' 'Derelict Vehicle' 'Noise -
Vehicle'
 'Noise - Street/Sidewalk' 'Animal Abuse' 'Noise - Commercial'
'Traffic'
 'Noise - House of Worship' 'Urinating in Public' 'Drinking'
 'Homeless Encampment' 'Noise - Park' 'Vending' 'Illegal Fireworks'
 'Bike/Roller/Skate Chronic' 'Disorderly Youth' 'Posting
Advertisement'
 'Graffiti']
SOUTH RICHMOND HILL:
['Blocked Driveway' 'Illegal Parking' 'Noise - Commercial'
 'Derelict Vehicle' 'Vending' 'Noise - Street/Sidewalk' 'Drinking'
 'Noise - Vehicle' 'Traffic' 'Animal Abuse' 'Homeless Encampment'
 'Noise - House of Worship' 'Bike/Roller/Skate Chronic' 'Disorderly
Youth'
 'Illegal Fireworks' 'Noise - Park' 'Urinating in Public']
SPRINGFIELD GARDENS:
['Illegal Parking' 'Blocked Driveway' 'Animal Abuse' 'Derelict
Vehicle'
 'Traffic' 'Noise - Vehicle' 'Drinking' 'Noise - Street/Sidewalk'
 'Posting Advertisement' 'Noise - Commercial' 'Noise - House of
Worship'
 'Homeless Encampment' 'Panhandling' 'Noise - Park' 'Urinating in
Public'
 'Vending' 'Illegal Fireworks']
STATEN ISLAND:
['Posting Advertisement' 'Noise - Commercial' 'Illegal Parking'
 'Blocked Driveway' 'Bike/Roller/Skate Chronic' 'Derelict Vehicle'
 'Drinking' 'Animal Abuse' 'Noise - Vehicle' 'Homeless Encampment'
 'Noise - Street/Sidewalk' 'Disorderly Youth' 'Noise - House of
Worship'
 'Urinating in Public' 'Traffic' 'Vending' 'Panhandling' 'Noise -
Park'
 'Illegal Fireworks' 'Graffiti']
SUNNYSIDE:
```

```
['Blocked Driveway' 'Noise - Commercial' 'Noise - Street/Sidewalk'
 'Illegal Parking' 'Noise - Vehicle' 'Vending' 'Traffic'
 'Bike/Roller/Skate Chronic' 'Derelict Vehicle' 'Noise - Park'
'Drinking'
 'Animal Abuse' 'Homeless Encampment' 'Urinating in Public'
 'Posting Advertisement' 'Graffiti' 'Disorderly Youth']
WHITESTONE:
['Illegal Parking' 'Blocked Driveway' 'Derelict Vehicle'
 'Noise - Street/Sidewalk' 'Traffic' 'Noise - Commercial' 'Animal
Abuse'
 'Bike/Roller/Skate Chronic' 'Noise - Vehicle' 'Noise - Park'
 'Illegal Fireworks' 'Drinking' 'Disorderly Youth' 'Graffiti'
'Vending']
WOODHAVEN:
['Illegal Parking' 'Blocked Driveway' 'Noise - Vehicle' 'Animal Abuse'
 'Noise - Commercial' 'Derelict Vehicle' 'Noise - Street/Sidewalk'
 'Noise - House of Worship' 'Homeless Encampment' 'Noise - Park'
'Traffic'
 'Urinating in Public' 'Vending' 'Drinking' 'Bike/Roller/Skate
Chronic'
 'Panhandling']
WOODSIDE:
['Blocked Driveway' 'Illegal Parking' 'Derelict Vehicle'
 'Noise - Street/Sidewalk' 'Animal Abuse' 'Noise - Commercial'
'Vending'
 'Noise - Park' 'Noise - Vehicle' 'Drinking' 'Traffic'
 'Bike/Roller/Skate Chronic' 'Urinating in Public' 'Graffiti'
 'Homeless Encampment' 'Noise - House of Worship' 'Illegal Fireworks'
 'Disorderly Youth']
Woodside:
['Illegal Parking' 'Blocked Driveway' 'Derelict Vehicle'
 'Noise - Street/Sidewalk' 'Noise - Commercial']
```

```
# for checkin purpose
data['BROOKLYN']
```

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

```
# step 4 -filling na values by using numpy.np
#(could not able create dataframe due to same length value error)
```

```
# by using for loop and numpy.nan and

max_len = max(len(v) for v in data.values())

for k in data.keys():
    if len(data[k]) < max_len:
        data[k] = np.pad(data[k], (0, max_len - len(data[k])),
                            mode='constant', constant_values=np.nan)

#step 5 - Creating Dataframe from dictionary key value pair along with 
nan by pandas as new data set
City_wise_complaint = pd.DataFrame(data)

City_wise_complaint.head(3)
```

```
              ARVERNE             ASTORIA                  Astoria  \
0     Illegal Parking    Blocked Driveway          Illegal Parking   
1  Noise - Commercial  Noise - Commercial  Noise - Street/Sidewalk   
2        Animal Abuse     Noise - Vehicle         Derelict Vehicle   

            BAYSIDE                BELLEROSE             BREEZY POINT  
\
0  Blocked Driveway         Derelict Vehicle  Noise - Street/Sidewalk   

1  Derelict Vehicle         Blocked Driveway         Blocked Driveway   

2   Illegal Parking  Noise - Street/Sidewalk             Animal Abuse   


                     BRONX            BROOKLYN    CAMBRIA HEIGHTS  \
0         Blocked Driveway     Illegal Parking   Derelict Vehicle   
1          Illegal Parking    Blocked Driveway   Blocked Driveway   
2  Noise - Street/Sidewalk  Noise - Commercial  Noise - Commercial   

              CENTRAL PARK  ...      SAINT ALBANS  SOUTH OZONE PARK  \
0  Noise - Street/Sidewalk  ...  Blocked Driveway  Blocked Driveway   
1          Illegal Parking  ...   Illegal Parking   Illegal Parking   
2                      NaN  ...  Derelict Vehicle  Derelict Vehicle   

  SOUTH RICHMOND HILL SPRINGFIELD GARDENS          STATEN ISLAND  \
0    Blocked Driveway     Illegal Parking  Posting Advertisement   
1     Illegal Parking    Blocked Driveway     Noise - Commercial   
2  Noise - Commercial        Animal Abuse        Illegal Parking   

                 SUNNYSIDE        WHITESTONE        WOODHAVEN  \
0         Blocked Driveway   Illegal Parking  Illegal Parking   
1       Noise - Commercial  Blocked Driveway  Blocked Driveway   
2  Noise - Street/Sidewalk  Derelict Vehicle  Noise - Vehicle   

           WOODSIDE         Woodside  
```

```
0  Blocked Driveway   Illegal Parking
1   Illegal Parking  Blocked Driveway
2  Derelict Vehicle  Derelict Vehicle

[3 rows x 53 columns]
```

4.Visualize the major types of complaints in each city**

** pd.crosstab() is a pandas function that is used to create a cross-tabulation **

** Table between two or more categorical variables**

** It is a convenient way to summarize and analyze the relationships between categorical variables in a dataset**

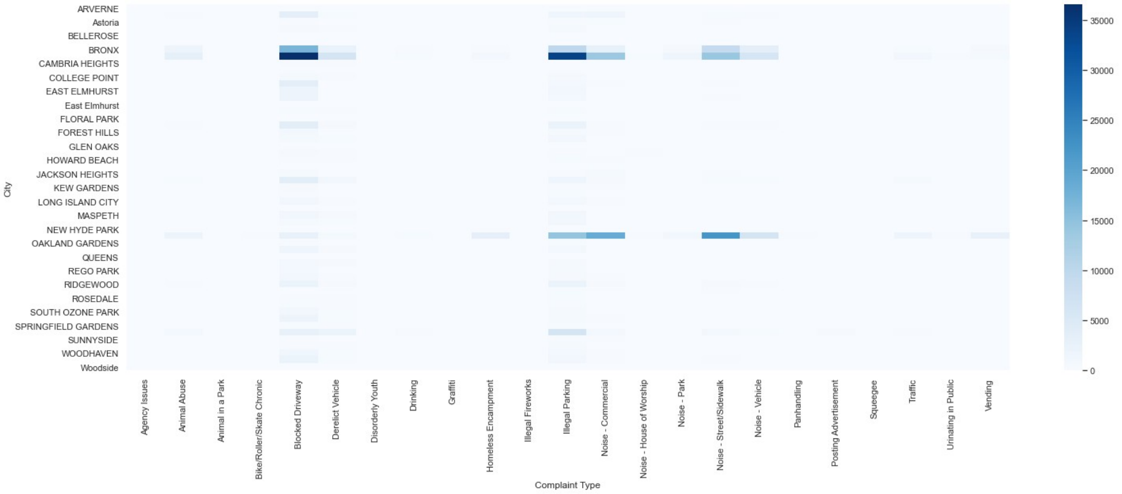
```
Heat_map_visual_Type_of_complaint = pd.crosstab(calls['City'],
calls['Complaint Type'])
```

```
calls['Complaint Type'].value_counts()
```

```
Blocked Driveway             100624
Illegal Parking               91716
Noise - Street/Sidewalk       51139
Noise - Commercial            43751
Derelict Vehicle              21518
Noise - Vehicle               19301
Animal Abuse                  10530
Traffic                        5196
Homeless Encampment            4879
Vending                        4185
Noise - Park                   4089
Drinking                       1404
Noise - House of Worship       1068
Posting Advertisement           679
Urinating in Public             641
Bike/Roller/Skate Chronic       475
Panhandling                     325
Disorderly Youth                315
Illegal Fireworks               172
Graffiti                        157
Agency Issues                     8
Squeegee                          4
Animal in a Park                  1
Name: Complaint Type, dtype: int64
```

```
sns.heatmap(Heat_map_visual_Type_of_complaint, cmap='Blues')
```

```
<AxesSubplot:xlabel='Complaint Type', ylabel='City'>
```



5.Check if the average response time across various types of Complaints

** steps involved for this task **

** step1 - Subtraction for create date column and closed Date column **

** step 2 - convert the 'Time Difference' column to seconds **

** (because without converting the out put was in days only or it was float point values)**

** step 3 - First I grouped by Complaint type column and then I used aggregate function for chekin average response**

** step 4 - pd.to_timedelta() is a pandas function**

** (for easy readable like human readable **

** it is used to convert a numeric value (in seconds, minutes, hours, or days) to a timedelta object**

** Timedelta objects represent a duration of time**

** similar to datetime objects which represent a specific point in time **

```
# step1 - Subtraction for create date column and closed Date column

calls['Time_Difference']= calls['Closed Date'] - calls['Created Date']

# for checkin purpose
calls.head(1)
```

```
   Unique Key        Created Date         Closed Date Agency  \
0    32310363 2015-12-31 23:59:45 2016-01-01 00:55:15   NYPD

                       Agency Name           Complaint Type
Descriptor  \
0  New York City Police Department  Noise - Street/Sidewalk  Loud
Music/Party

     Location Type  Incident Zip     Incident Address  ... School
Address  \
0  Street/Sidewalk       10034.0  71 VERMILYEA AVENUE  ...
Unspecified

   School City School State   School Zip School Not Found
Latitude  \
0  Unspecified  Unspecified  Unspecified                N  40.865682


   Longitude                                 Location
BROOKLYN_STATE  \
0 -73.923501  (40.86568153633767, -73.92350095571744)              0


  Time_Difference
0 0 days 00:55:30

[1 rows x 41 columns]
```

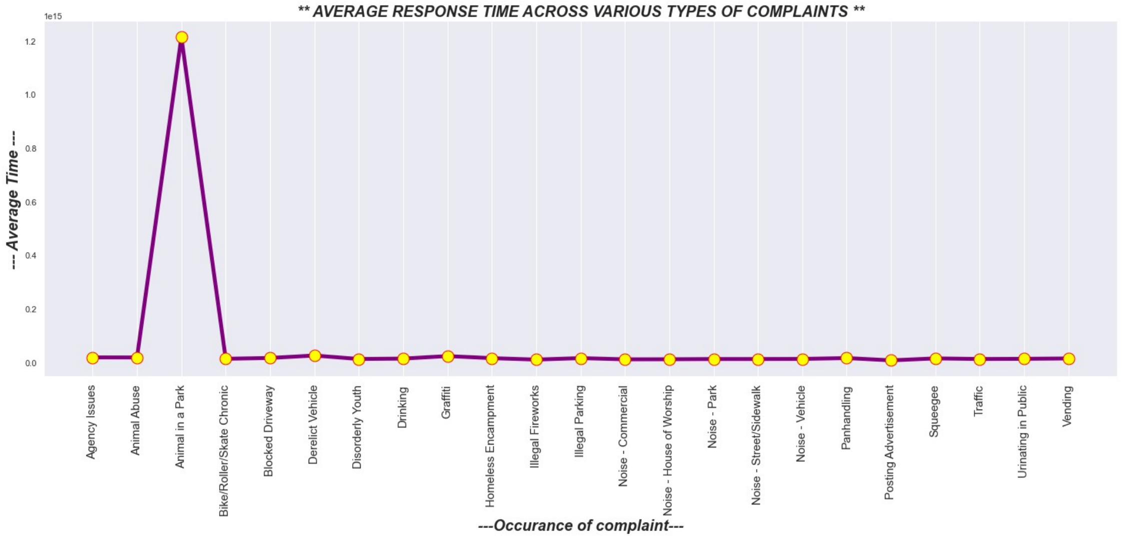
```
# step 2 - convert the 'Time Difference' column to seconds
# because without converting the out put was in days only or it was
float point values
calls['Time_Difference'] = calls['Time_Difference'].dt.total_seconds()



# step 3 - First I grouped by Complaint type column and then I used
aggregate function for chekin average response

avg_time_difference = calls.groupby('Complaint Type')
['Time_Difference'].mean()



# step 4 - pd.to_timedelta() is a pandas function
# for easy readable like human readable
# it is used to convert a numeric value (in seconds, minutes, hours,
or days) to a timedelta object.
# Timedelta objects represent a duration of time,
# similar to datetime objects which represent a specific point in
time.

avg_time_difference = pd.to_timedelta(avg_time_difference, unit='s')
```

```
avg_time_difference
```

```
Complaint Type
Agency Issues                     0 days 05:04:49.125000
Animal Abuse                   0 days 05:00:32.556030389
Animal in a Park                        14 days 00:50:34
Bike/Roller/Skate Chronic   0 days 03:38:43.688421053
Blocked Driveway            0 days 04:30:32.521515741
Derelict Vehicle            0 days 07:02:39.600102240
Disorderly Youth            0 days 03:26:03.749206349
Drinking                    0 days 03:50:21.300569801
Graffiti                    0 days 06:27:56.343949045
Homeless Encampment         0 days 04:17:31.384505022
Illegal Fireworks           0 days 02:48:33.482558140
Illegal Parking             0 days 04:20:50.435670984
Noise - Commercial          0 days 03:04:45.760531188
Noise - House of Worship    0 days 03:09:51.087078652
Noise - Park                0 days 03:23:46.055514796
Noise - Street/Sidewalk     0 days 03:23:51.295410548
Noise - Vehicle             0 days 03:29:21.800010362
Panhandling                 0 days 04:24:13.550769231
Posting Advertisement       0 days 02:01:26.256259205
Squeegee                       0 days 04:02:40.250000
Traffic                     0 days 03:25:09.120092379
Urinating in Public         0 days 03:35:59.293291732
Vending                     0 days 03:59:26.278375149
Name: Time_Difference, dtype: timedelta64[ns]
```

>>> Using Plot for Checking Average Time Response<<<

```
font_style = {'family': 'Arial', 'size': 20, 'weight': 'bold',
'style': 'italic'}
plt.figure(figsize=(25,8.5))
plt.plot(avg_time_difference,marker ='o',ms=15, mec='red',
mfc='yellow',c='purple',lw=5)
plt.grid(axis='y',ls='solid',color ='k',lw=0.5,alpha=0.5)
plt.xlabel('---Occurance of complaint---',fontdict=font_style)
plt.ylabel('--- Average Time ---',fontdict=font_style)
plt.title('** AVERAGE RESPONSE TIME ACROSS VARIOUS TYPES OF COMPLAINTS
**',fontdict=font_style)
plt.xticks(rotation='vertical', ha='center',size=15)
plt.show()
```


** AVERAGE RESPONSE TIME ACROSS VARIOUS TYPES OF COMPLAINTS **
1e15
1.2
1.0
0.8
0.6
0.4
0.2
0.0
--- Average Time ---
Agency Issues
Animal Abuse
Animal in a Park
Bike/Roller/Skate Chronic
Blocked Driveway
Derelict Vehicle
Disorderly Youth
Drinking
Graffiti
Homeless Encampment
Illegal Fireworks
Illegal Parking
Noise - Commercial
Noise - House of Worship
Noise - Park
Noise - Street/Sidewalk
Noise - Vehicle
Panhandling
Posting Advertisement
Squeegee
Traffic
Urinating in Public
Vending
---Occurance of complaint---