import all Important used Library

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.offline as py
import plotly.graph_objs as go
import matplotlib.ticker as ticker
```

Read Data in CSV format in Data Frame

```
df=pd.read_csv("311_Service_Requests_from_2010_to_Present.csv",low_mem
ory=False)
```

The info() function is used to print a concise summary of a DataFrame. This method prints information about a DataFrame including the index dtype and column dtypes, non-null values and memory usage.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300698 entries, 0 to 300697
Data columns (total 53 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   Unique Key                      300698 non-null  int64
 1   Created Date                    300698 non-null  object
 2   Closed Date                     298534 non-null  object
 3   Agency                          300698 non-null  object
 4   Agency Name                     300698 non-null  object
 5   Complaint Type                  300698 non-null  object
 6   Descriptor                      294784 non-null  object
 7   Location Type                   300567 non-null  object
 8   Incident Zip                    298083 non-null  float64
 9   Incident Address                256288 non-null  object
 10  Street Name                     256288 non-null  object
 11  Cross Street 1                  251419 non-null  object
 12  Cross Street 2                  250919 non-null  object
 13  Intersection Street 1           43858 non-null   object
 14  Intersection Street 2           43362 non-null   object
 15  Address Type                    297883 non-null  object
 16  City                            298084 non-null  object
 17  Landmark                        349 non-null     object
 18  Facility Type                   298527 non-null  object
 19  Status                          300698 non-null  object
 20  Due Date                        300695 non-null  object
 21  Resolution Description          300698 non-null  object
 22  Resolution Action Updated Date  298511 non-null  object
```

```
 23  Community Board                 300698 non-null  object
 24  Borough                         300698 non-null  object
 25  X Coordinate (State Plane)      297158 non-null  float64
 26  Y Coordinate (State Plane)      297158 non-null  float64
 27  Park Facility Name              300698 non-null  object
 28  Park Borough                    300698 non-null  object
 29  School Name                     300698 non-null  object
 30  School Number                   300698 non-null  object
 31  School Region                   300697 non-null  object
 32  School Code                     300697 non-null  object
 33  School Phone Number             300698 non-null  object
 34  School Address                  300698 non-null  object
 35  School City                     300698 non-null  object
 36  School State                    300698 non-null  object
 37  School Zip                      300697 non-null  object
 38  School Not Found                300698 non-null  object
 39  School or Citywide Complaint    0 non-null       float64
 40  Vehicle Type                    0 non-null       float64
 41  Taxi Company Borough            0 non-null       float64
 42  Taxi Pick Up Location           0 non-null       float64
 43  Bridge Highway Name             243 non-null     object
 44  Bridge Highway Direction        243 non-null     object
 45  Road Ramp                       213 non-null     object
 46  Bridge Highway Segment          213 non-null     object
 47  Garage Lot Name                 0 non-null       float64
 48  Ferry Direction                 1 non-null       object
 49  Ferry Terminal Name             2 non-null       object
 50  Latitude                        297158 non-null  float64
 51  Longitude                       297158 non-null  float64
 52  Location                        297158 non-null  object
dtypes: float64(10), int64(1), object(42)
memory usage: 121.6+ MB
```

df. head() Returns the first 5 rows of the dataframe

```
df.head()
```

```
   Unique Key            Created Date    Closed Date Agency  \
0    32310363  12/31/2015 11:59:45 PM  01-01-16 0:55   NYPD
1    32309934  12/31/2015 11:59:44 PM  01-01-16 1:26   NYPD
2    32309159  12/31/2015 11:59:29 PM  01-01-16 4:51   NYPD
3    32305098  12/31/2015 11:57:46 PM  01-01-16 7:43   NYPD
4    32306529  12/31/2015 11:56:58 PM  01-01-16 3:24   NYPD

                       Agency Name           Complaint Type  \
0  New York City Police Department  Noise - Street/Sidewalk
1  New York City Police Department         Blocked Driveway
2  New York City Police Department         Blocked Driveway
3  New York City Police Department          Illegal Parking
4  New York City Police Department          Illegal Parking
```

```
                    Descriptor    Location Type  Incident Zip  \
0              Loud Music/Party  Street/Sidewalk       10034.0
1                     No Access  Street/Sidewalk       11105.0
2                     No Access  Street/Sidewalk       10458.0
3  Commercial Overnight Parking  Street/Sidewalk       10461.0
4              Blocked Sidewalk  Street/Sidewalk       11373.0

        Incident Address  ... Bridge Highway Name Bridge Highway
Direction  \
0    71 VERMILYEA AVENUE  ...                 NaN
NaN
1        27-07 23 AVENUE  ...                 NaN
NaN
2  2897 VALENTINE AVENUE  ...                 NaN
NaN
3    2940 BAISLEY AVENUE  ...                 NaN
NaN
4          87-14 57 ROAD  ...                 NaN
NaN

  Road Ramp Bridge Highway Segment Garage Lot Name Ferry Direction  \
0       NaN                    NaN             NaN             NaN
1       NaN                    NaN             NaN             NaN
2       NaN                    NaN             NaN             NaN
3       NaN                    NaN             NaN             NaN
4       NaN                    NaN             NaN             NaN

  Ferry Terminal Name   Latitude  Longitude  \
0                 NaN  40.865682 -73.923501
1                 NaN  40.775945 -73.915094
2                 NaN  40.870325 -73.888525
3                 NaN  40.835994 -73.828379
4                 NaN  40.733060 -73.874170

                                  Location
0   (40.86568153633767, -73.92350095571744)
1  (40.775945312321085, -73.91509393898605)
2  (40.870324522111424, -73.88852464418646)
3   (40.83599404683083, -73.82837939584206)
4  (40.733059618956815, -73.87416975810375)

[5 rows x 53 columns]
```

Pandas to_datetime() method helps to convert string Date time into Python Date time object

And assign df['Created Date'] in same column

```
df['Created Date']=pd.to_datetime(df['Created Date'])
```

```
df['Closed Date']=pd.to_datetime(df['Closed Date'])
```

df.info() - check df['Closed Date'] Dtype are conver or not. its convered

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300698 entries, 0 to 300697
Data columns (total 53 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   Unique Key                      300698 non-null  int64
 1   Created Date                    300698 non-null  datetime64[ns]
 2   Closed Date                     298534 non-null  datetime64[ns]
 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: datetime64[ns](2), float64(10), int64(1), object(40)
memory usage: 121.6+ MB
```

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

```
df['Request_Closing_Time']=df['Closed Date'] - df['Created Date']
```

```
df
```

```
        Unique Key        Created Date         Closed Date Agency  \
0         32310363 2015-12-31 23:59:45 2016-01-01 00:55:00   NYPD
1         32309934 2015-12-31 23:59:44 2016-01-01 01:26:00   NYPD
2         32309159 2015-12-31 23:59:29 2016-01-01 04:51:00   NYPD
3         32305098 2015-12-31 23:57:46 2016-01-01 07:43:00   NYPD
4         32306529 2015-12-31 23:56:58 2016-01-01 03:24:00   NYPD
...            ...                 ...                 ...    ...
300693    30281872 2015-03-29 00:33:41                 NaT   NYPD
300694    30281230 2015-03-29 00:33:28 2015-03-29 02:33:59   NYPD
300695    30283424 2015-03-29 00:33:03 2015-03-29 03:40:20   NYPD
300696    30280004 2015-03-29 00:33:02 2015-03-29 04:38:35   NYPD
300697    30281825 2015-03-29 00:33:01 2015-03-29 04:41:50   NYPD

                           Agency Name           Complaint Type  \
0       New York City Police Department  Noise - Street/Sidewalk
1       New York City Police Department         Blocked Driveway
2       New York City Police Department         Blocked Driveway
3       New York City Police Department          Illegal Parking
4       New York City Police Department          Illegal Parking
...                                 ...                      ...
300693  New York City Police Department       Noise - Commercial
300694  New York City Police Department         Blocked Driveway
300695  New York City Police Department       Noise - Commercial
300696  New York City Police Department       Noise - Commercial
300697  New York City Police Department       Noise - Commercial

                         Descriptor        Location Type  Incident
```

```
Zip  \
0                  Loud Music/Party      Street/Sidewalk
10034.0
1                         No Access      Street/Sidewalk
11105.0
2                         No Access      Street/Sidewalk
10458.0
3       Commercial Overnight Parking      Street/Sidewalk
10461.0
4                  Blocked Sidewalk      Street/Sidewalk
11373.0
...                                ...                  ...           ..
.
300693             Loud Music/Party  Club/Bar/Restaurant
NaN
300694               Partial Access      Street/Sidewalk
11418.0
300695             Loud Music/Party  Club/Bar/Restaurant
11206.0
300696             Loud Music/Party  Club/Bar/Restaurant
10461.0
300697             Loud Music/Party     Store/Commercial
10036.0

               Incident Address  ... Bridge Highway Direction Road
Ramp  \
0            71 VERMILYEA AVENUE  ...                       NaN
NaN
1                27-07 23 AVENUE  ...                       NaN
NaN
2          2897 VALENTINE AVENUE  ...                       NaN
NaN
3            2940 BAISLEY AVENUE  ...                       NaN
NaN
4                  87-14 57 ROAD  ...                       NaN
NaN
...                          ...  ...                       ...       .
..
300693           CRESCENT AVENUE  ...                       NaN
NaN
300694          100-17 87 AVENUE  ...                       NaN
NaN
300695         162 THROOP AVENUE  ...                       NaN
NaN
300696  3151 EAST TREMONT AVENUE  ...                       NaN
NaN
300697        251 WEST 48 STREET  ...                       NaN
NaN

       Bridge Highway Segment Garage Lot Name Ferry Direction  \
```

```
0                       NaN             NaN             NaN
1                       NaN             NaN             NaN
2                       NaN             NaN             NaN
3                       NaN             NaN             NaN
4                       NaN             NaN             NaN
...                     ...             ...             ...
300693                  NaN             NaN             NaN
300694                  NaN             NaN             NaN
300695                  NaN             NaN             NaN
300696                  NaN             NaN             NaN
300697                  NaN             NaN             NaN

       Ferry Terminal Name   Latitude  Longitude  \
0                      NaN  40.865682 -73.923501
1                      NaN  40.775945 -73.915094
2                      NaN  40.870325 -73.888525
3                      NaN  40.835994 -73.828379
4                      NaN  40.733060 -73.874170
...                    ...        ...        ...
300693                 NaN        NaN        NaN
300694                 NaN  40.694077 -73.846087
300695                 NaN  40.699590 -73.944234
300696                 NaN  40.837708 -73.834587
300697                 NaN  40.760583 -73.985922

                                        Location Request_Closing_Time

0        (40.86568153633767, -73.92350095571744)      0 days 00:55:15

1       (40.775945312321085, -73.91509393898605)      0 days 01:26:16

2       (40.870324522111424, -73.88852464418646)      0 days 04:51:31

3        (40.83599404683083, -73.82837939584206)      0 days 07:45:14

4       (40.733059618956815, -73.87416975810375)      0 days 03:27:02

...                                          ...                  ...

300693                                       NaN                  NaT

300694    (40.69407728322387, -73.8460866160573)      0 days 02:00:31

300695   (40.69959035300927, -73.94423377144169)      0 days 03:07:17

300696    (40.8377075854206, -73.83458731019586)      0 days 04:05:33

300697   (40.76058322950115, -73.98592204392392)      0 days 04:08:49
```

```
[300698 rows x 54 columns]
```

```
df['Created Date'].value_counts()
```

```
2015-07-11 23:04:00    9
2015-11-06 23:34:00    9
2015-06-06 22:23:00    9
2015-10-09 23:56:00    8
2015-11-01 22:12:00    8
                      ..
2015-09-22 17:52:17    1
2015-09-22 17:50:43    1
2015-09-22 17:49:55    1
2015-09-22 17:49:47    1
2015-03-29 00:33:01    1
Name: Created Date, Length: 259493, dtype: int64
```

```
df['Closed Date'].value_counts()
```

```
2015-11-08 07:34:00    24
2015-10-11 07:03:00    22
2015-12-08 07:44:00    18
2015-05-10 07:01:00    18
2015-12-07 23:17:00    17
                       ..
2015-09-21 11:03:55     1
2015-09-21 08:52:27     1
2015-09-21 09:13:15     1
2015-09-21 08:26:57     1
2015-03-29 04:41:50     1
Name: Closed Date, Length: 237165, dtype: int64
```

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

```
Blocked Driveway             77044
Illegal Parking              75361
Noise - Street/Sidewalk      48612
Noise - Commercial           35577
Derelict Vehicle             17718
Noise - Vehicle              17083
Animal Abuse                  7778
Traffic                       4498
Homeless Encampment           4416
Noise - Park                  4042
Vending                       3802
Drinking                      1280
Noise - House of Worship       931
Posting Advertisement          650
Urinating in Public            592
Bike/Roller/Skate Chronic      427
```

```
Panhandling                    307
Disorderly Youth               286
Illegal Fireworks              168
Graffiti                       113
Agency Issues                    6
Squeegee                         4
Ferry Complaint                  2
Animal in a Park                 1
Name: Complaint Type, dtype: int64
```

A groupby operation involves some combination of splitting the object, applying a function, and combining the results.

This can be used to group large amounts of data and compute operations on these groups.

gropping two column Complaint Type and Unique Key and assin ther value is complaint

```
complaint=df.groupby(['Complaint Type'])['Unique Key'].count()
```

```
complaint
```

```
Complaint Type
Agency Issues                     6
Animal Abuse                   7778
Animal in a Park                  1
Bike/Roller/Skate Chronic       427
Blocked Driveway              77044
Derelict Vehicle              17718
Disorderly Youth                286
Drinking                       1280
Ferry Complaint                   2
Graffiti                        113
Homeless Encampment            4416
Illegal Fireworks               168
Illegal Parking               75361
Noise - Commercial            35577
Noise - House of Worship        931
Noise - Park                   4042
Noise - Street/Sidewalk       48612
Noise - Vehicle               17083
Panhandling                     307
Posting Advertisement           650
Squeegee                          4
Traffic                        4498
Urinating in Public             592
Vending                        3802
Name: Unique Key, dtype: int64
```

Pandas reset_index() is a method to reset index of a Data Frame. reset_index() method sets a list of integer ranging from 0 to length of data as index.

Create a new Data Frame there has two col. "Complaint Type and Unique Key" then reset there index.

```
type_comp=pd.DataFrame(complaint).reset_index()
```

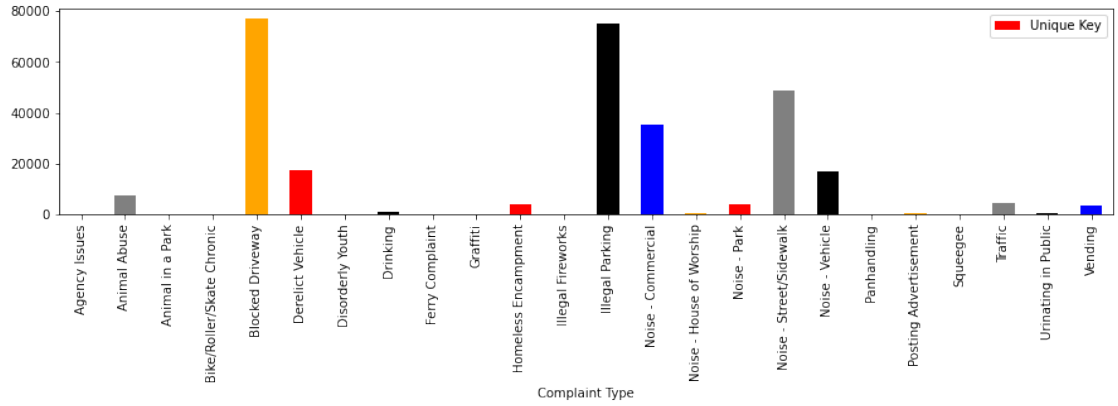
the head function are print first 5 rows . show type_comp data frame 5 rows .

```
type_comp.head()
```

```
              Complaint Type  Unique Key
0              Agency Issues           6
1               Animal Abuse        7778
2          Animal in a Park            1
3  Bike/Roller/Skate Chronic         427
4          Blocked Driveway        77044
```

Ploting type_comp data frame there value are x='Complaint Type',y='Unique Key' and result are max complaint type are Blocked Driveway and second is illegal Parking and more.

```
ec = ['red', 'gray', 'black', 'blue', 'orange']
type_comp.plot(x='Complaint Type',y='Unique
Key',kind='bar',figsize=(15,3),color = ec)
plt.show()
```



Groping two column and count there value and drop null value in Complaint Type and again grouping Complaint Type and next arranging the value in decending order and next reset there index name. get 5 value in this data frame and ploting in pie chart.

result is Blocked Driveway has 30.3% complaint and 29.6% illegal Parking and 19% Noise-Street/Sidewalk and more.

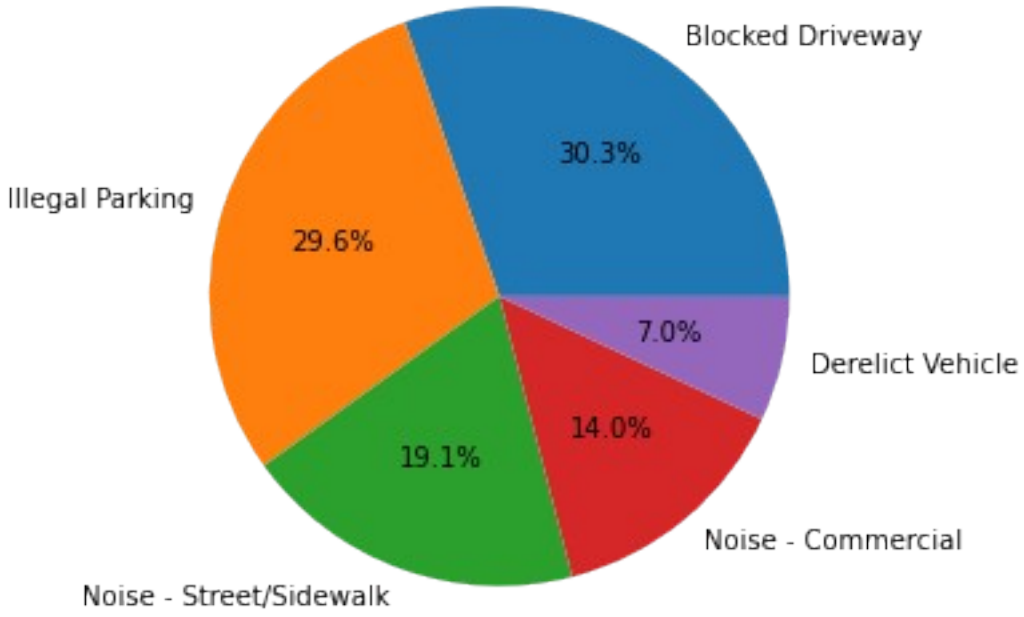
```
df.groupby(['Complaint Type'])['Unique Key'].count()
comp_dronna=df.dropna(subset=["Complaint Type"])
comp_dronna=df.groupby("Complaint Type")
sortComplaintType = comp_dronna.size().sort_values(ascending = False)
sortComplaintType = sortComplaintType.to_frame('Unique
Key').reset_index()
```

```
sortComplaintType.head(10)

sortComplaintType = sortComplaintType.head()
plt.figure(figsize=(5,5))
plt.pie(sortComplaintType['Unique
Key'],labels=sortComplaintType["Complaint Type"], autopct="%1.1f%%")
plt.show()


# labels = sortComplaintType["Complaint Type"].head()
# sizes = sortComplaintType['Unique Key'].head()
# fig1, ax1 = plt.subplots()
# ax1.pie(sizes, labels=labels, autopct='%1.1f%%',shadow=True,
startangle=90)
# ax1.axis('equal')

# plt.show()
```




```
df['City'].value_counts()
```

```
BROOKLYN               98307
NEW YORK               65994
BRONX                  40702
STATEN ISLAND          12343
JAMAICA                 7296
ASTORIA                 6330
FLUSHING                5971
```

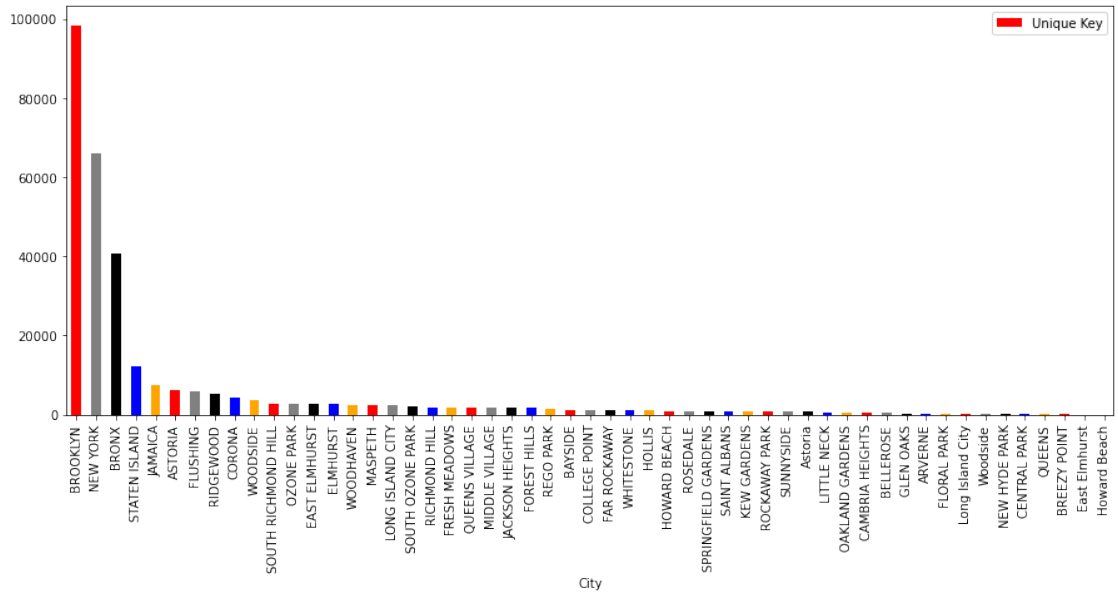
```
RIDGEWOOD               5163
CORONA                  4295
WOODSIDE                3544
SOUTH RICHMOND HILL     2774
OZONE PARK              2755
EAST ELMHURST           2734
ELMHURST                2673
WOODHAVEN               2464
MASPETH                 2462
LONG ISLAND CITY        2437
SOUTH OZONE PARK        2173
RICHMOND HILL           1904
FRESH MEADOWS           1899
QUEENS VILLAGE          1814
MIDDLE VILLAGE          1765
JACKSON HEIGHTS         1689
FOREST HILLS            1688
REGO PARK               1486
BAYSIDE                 1221
COLLEGE POINT           1220
FAR ROCKAWAY            1179
WHITESTONE              1098
HOLLIS                  1012
HOWARD BEACH             931
ROSEDALE                 922
SPRINGFIELD GARDENS      883
SAINT ALBANS             834
KEW GARDENS              771
ROCKAWAY PARK            745
SUNNYSIDE                723
Astoria                  717
LITTLE NECK              559
OAKLAND GARDENS          551
CAMBRIA HEIGHTS          477
BELLEROSE                375
GLEN OAKS                306
ARVERNE                  220
FLORAL PARK              152
Long Island City         134
Woodside                 120
NEW HYDE PARK             98
CENTRAL PARK              97
QUEENS                    32
BREEZY POINT              30
East Elmhurst             14
Howard Beach               1
Name: City, dtype: int64
```

Groping two column and count there value and drop null value in City and again grouping City and next arranging the value in decending order and next reset there index name. get 5 value in this data frame and ploting in Bar chart.

result top 5 City are Brooklyn, New York, Bronx, Staten island and jamaica has large ammount of complaint.

```
city_wise=df.groupby(['City'])['Unique Key'].count()
city_wise=df.dropna(subset=['City'])
city_wise=df.groupby('City')
sort_city = city_wise.size().sort_values(ascending = False)
sort_city = sort_city.to_frame('Unique Key').reset_index()

ec = ['red', 'gray', 'black', 'blue', 'orange']
sort_city.plot(x='City',y='Unique Key',kind='bar',figsize=(15,6),color
= ec)
plt.show()
```



PIE Chart

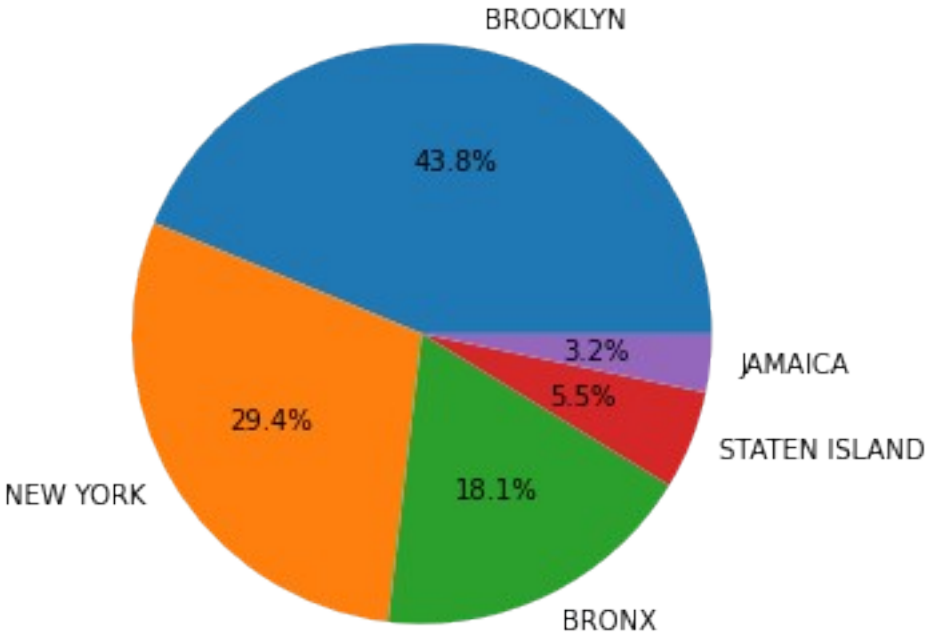
Groping two column and count there value and drop null value in City and again grouping City and next arranging the value in decending order and next reset there index name. get 5 value in this data frame and ploting in pie chart.

result Brooklynhas 43.8%, New York has 29.4%, Bronx has 18.1, Staten island has 5.5% and jamaica has 3.2% Complaint.

```
city_wise=df.groupby(['City'])['Unique Key'].count()
city_wise=df.dropna(subset=['City'])
city_wise=df.groupby('City')
sort_city = city_wise.size().sort_values(ascending = False)
```

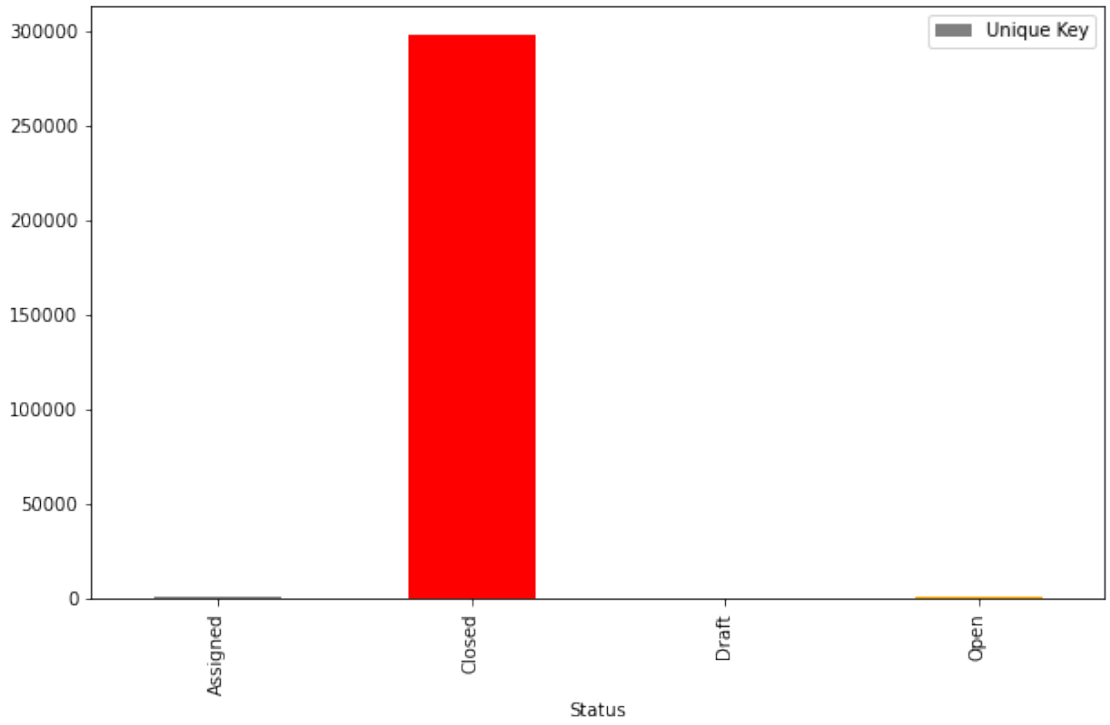
```
sort_city = sort_city.to_frame('Unique Key').reset_index()
sort_city.head(10)

sort_city = sort_city.head()
plt.figure(figsize=(5,5))
plt.pie(sort_city['Unique Key'],labels=sort_city['City'],
autopct="%1.1f%%")
plt.show()
```



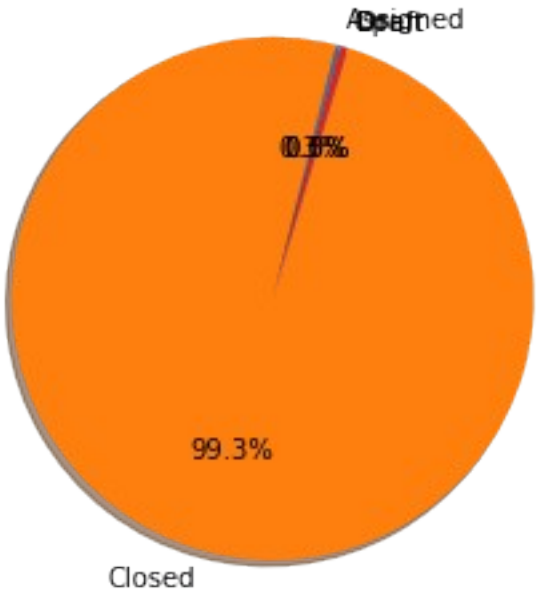
result max complaint are closed approx 300000.

```
# pd.DataFrame(df.groupby(['Status'])['Unique
Key'].count()).reset_index()
ec = ['gray','red','blue', 'orange']
comp_status=pd.DataFrame(df.groupby(['Status'])['Unique
Key'].count()).reset_index()
comp_status.plot(x='Status',y='Unique
Key',kind='bar',figsize=(10,6),color = ec)
plt.show()
```



result 99.3% Closed Complaint.

```
Statu=pd.DataFrame(df.groupby(['Status'])['Unique
Key'].count()).reset_index().head()
labels = Statu['Status']
sizes = Statu['Unique Key']
fig1, ax1 = plt.subplots()
ax1.pie(sizes, labels=labels, autopct='%1.1f%%',shadow=True,
startangle=75)
ax1.axis('equal')
plt.show()
```

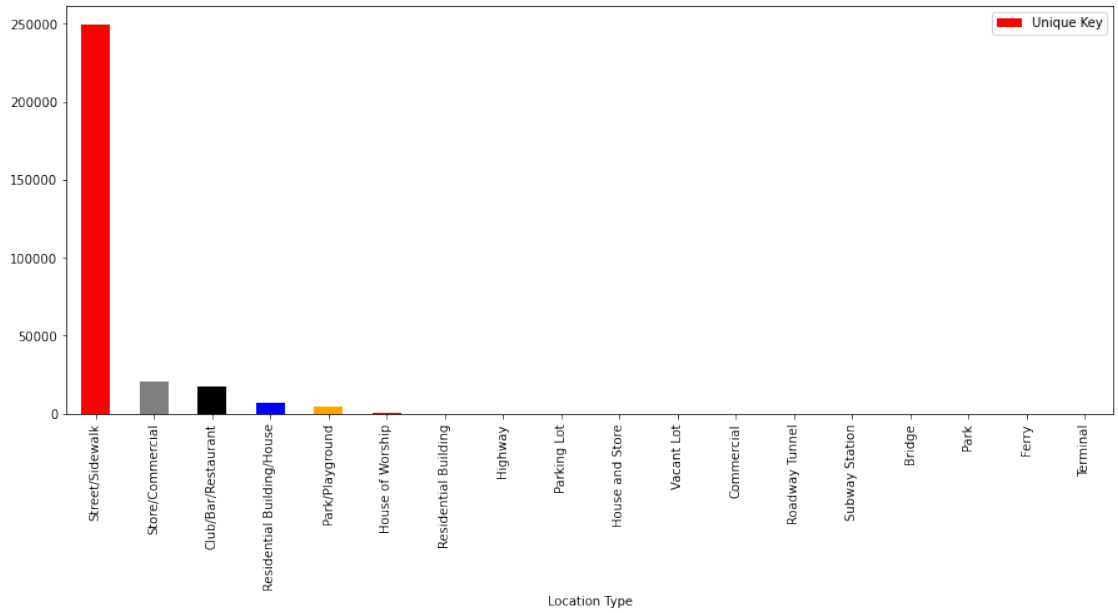


Groping two column and count there value and drop null value in Location Type and again grouping Location Type and next arranging the value in decending order and next reset there index name. get values in this data frame and ploting in Bar chart.

result Street/Sidewalk has large of complaint.

```
city_wise=df.groupby(['Location Type'])['Unique Key'].count()
city_wise=df.dropna(subset=['Location Type'])
city_wise=df.groupby('Location Type')
sort_city = city_wise.size().sort_values(ascending = False)
sort_city = sort_city.to_frame('Unique Key').reset_index()

ec = ['red', 'gray', 'black', 'blue', 'orange']
sort_city.plot(x='Location Type',y='Unique
Key',kind='bar',figsize=(15,6),color = ec)
plt.show()
```



Top 5 Complaint City

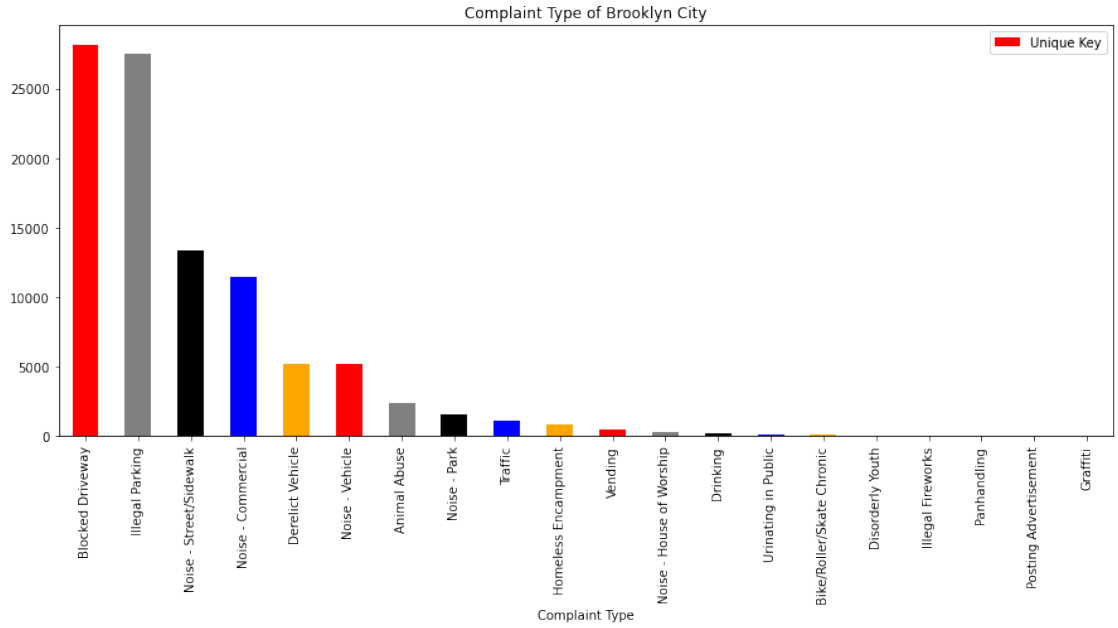
1.> Brooklyn

finding city column in Brooklyn city because this city has max complaints. Groping two column and count there value and drop null value in Complaint Type and again grouping Complaint Type and next arranging the value in decending order and next reset there index name. get values in this data frame and ploting in Bar chart.

result:- top 5 complaint 1.Blocked Driveway 2.illegal parking 3.Noise-Street/Sidewalk 4.Noise-Commericial 5. Derelict Vehicle

```
Brooklyn=df[(df['City']=='BROOKLYN')]
city_wise=Brooklyn.groupby(['Complaint Type'])['Unique Key'].count()
city_wise=Brooklyn.dropna(subset=['Complaint Type'])
city_wise=Brooklyn.groupby('Complaint Type')
sort_city = city_wise.size().sort_values(ascending = False)
sort_city = sort_city.to_frame('Unique Key').reset_index()

ec = ['red', 'gray', 'black', 'blue', 'orange']
sort_city.plot(x='Complaint Type',y='Unique Key',kind='bar',title =
'Complaint Type of Brooklyn City',figsize=(15,6),color = ec)
plt.show()
```



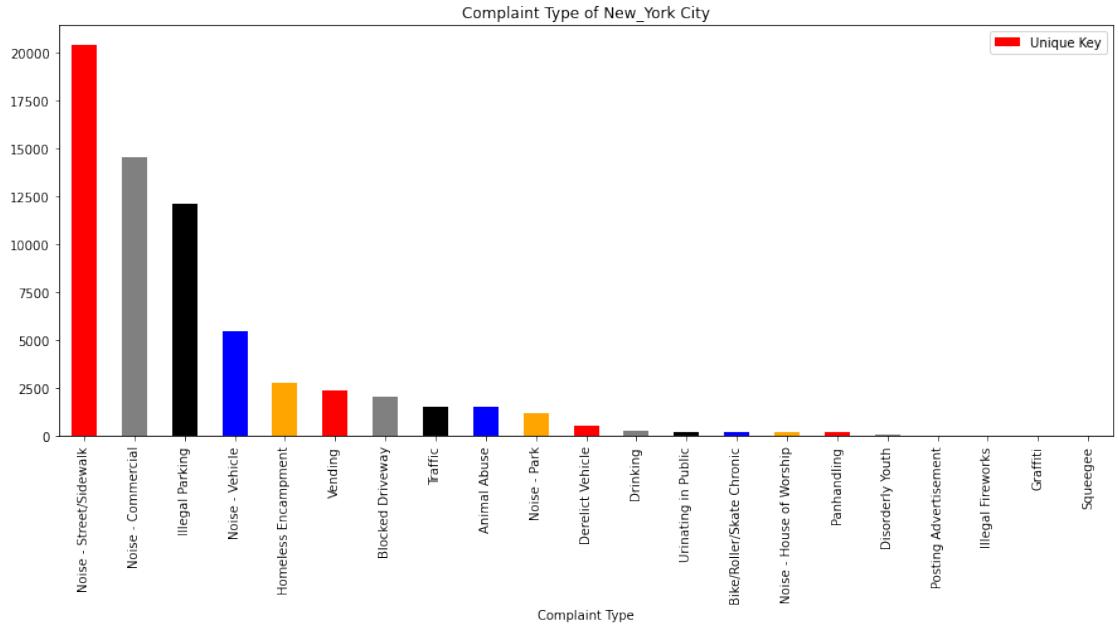
2.> New York

finding city column in New York city because this city has max complaints. Groping two column and count there value and drop null value in Complaint Type and again grouping Complaint Type and next arranging the value in decending order and next reset there index name. get values in this data frame and ploting in Bar chart.

result:- top 5 complaint 1.Noise-Street/Sidewalk 2.Noise-Commericial 3.illegal parking 4. Noise Vehicle 5.Homeless Encapment

```
New_York=df[(df['City']=='NEW YORK')]
city_wise=New_York.groupby(['Complaint Type'])['Unique Key'].count()
city_wise=New_York.dropna(subset=['Complaint Type'])
city_wise=New_York.groupby('Complaint Type')
sort_city = city_wise.size().sort_values(ascending = False)
sort_city = sort_city.to_frame('Unique Key').reset_index()

ec = ['red', 'gray', 'black', 'blue', 'orange']
sort_city.plot(x='Complaint Type',y='Unique Key',kind='bar',title =
'Complaint Type of New_York City',figsize=(15,6),color = ec)
plt.show()
```



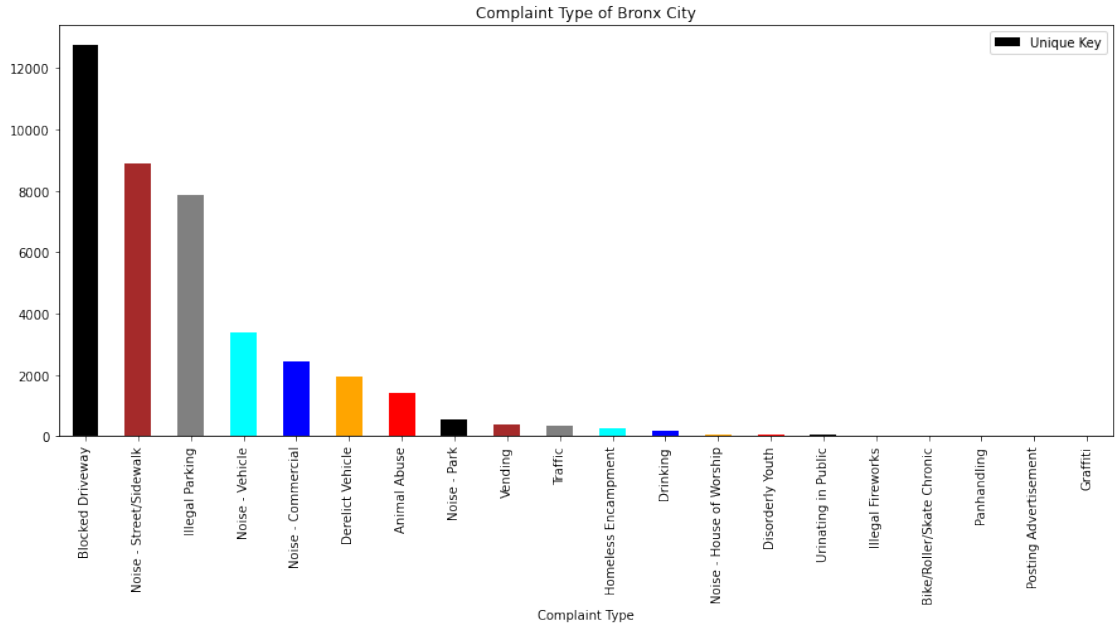
3.> Bronx

finding city column in Bronx city because this city has max complaints. Groping two column and count there value and drop null value in Complaint Type and again grouping Complaint Type and next arranging the value in decending order and next reset there index name. get values in this data frame and ploting in Bar chart.

result:- top 5 complaint 1. Blocked Driveway 2.Noise-Street/Sidewalk 3.illegal parking 4. Noise Vehicle 5.Noise Commercial

```
Bronx=df[(df['City']=='BRONX')]
city_wise=Bronx.groupby(['Complaint Type'])['Unique Key'].count()
city_wise=Bronx.dropna(subset=['Complaint Type'])
city_wise=Bronx.groupby('Complaint Type')
sort_city = city_wise.size().sort_values(ascending = False)
sort_city = sort_city.to_frame('Unique Key').reset_index()

ec = ['black','brown','gray','cyan','blue','orange','red']
sort_city.plot(x='Complaint Type',y='Unique Key',kind='bar',title =
'Complaint Type of Bronx City',figsize=(15,6),color = ec)
plt.show()
```



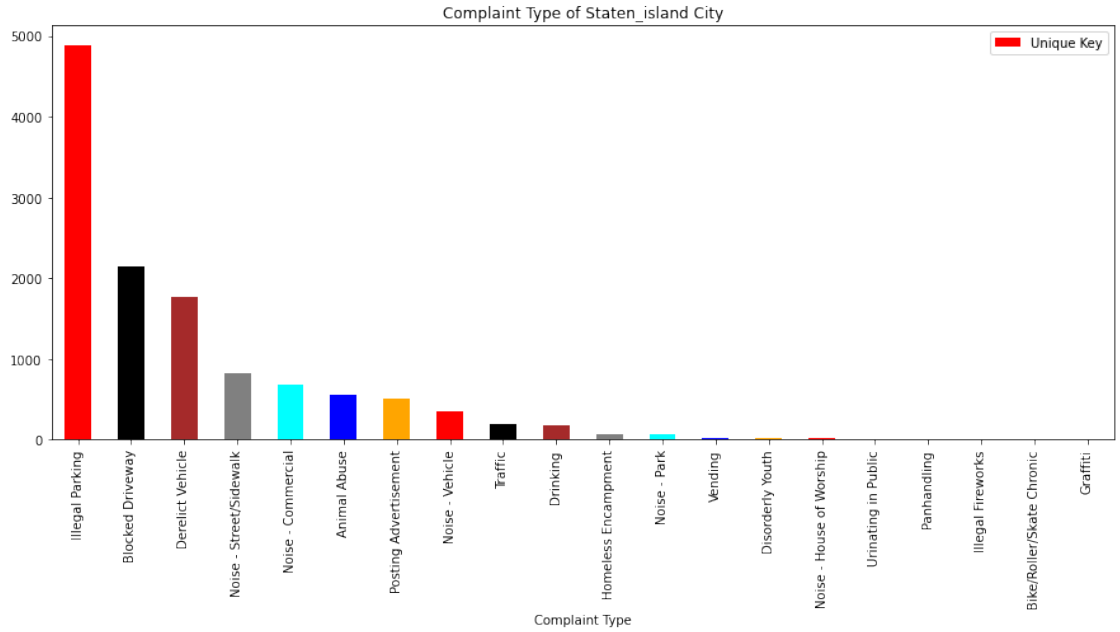
4.> Staten Island

finding city column in Staten Island city because this city has max complaints. Groping two column and count there value and drop null value in Complaint Type and again grouping Complaint Type and next arranging the value in decending order and next reset there index name. get values in this data frame and ploting in Bar chart.

result:- top 5 complaint 1.Illegal parking 2.Blocked Driveway 3. Derelict Vehicle 4. Noise-Street/Sidewalk 5.Noise Commercial

```
Staten_island=df[(df['City']=='STATEN ISLAND')]
city_wise=Staten_island.groupby(['Complaint Type'])['Unique
Key'].count()
city_wise=Staten_island.dropna(subset=['Complaint Type'])
city_wise=Staten_island.groupby('Complaint Type')
sort_city = city_wise.size().sort_values(ascending = False)
sort_city = sort_city.to_frame('Unique Key').reset_index()

ec = ['red','black','brown','gray','cyan','blue','orange']
sort_city.plot(x='Complaint Type',y='Unique Key',kind='bar',title =
'Complaint Type of Staten_island City',figsize=(15,6),color = ec)
plt.show()
```



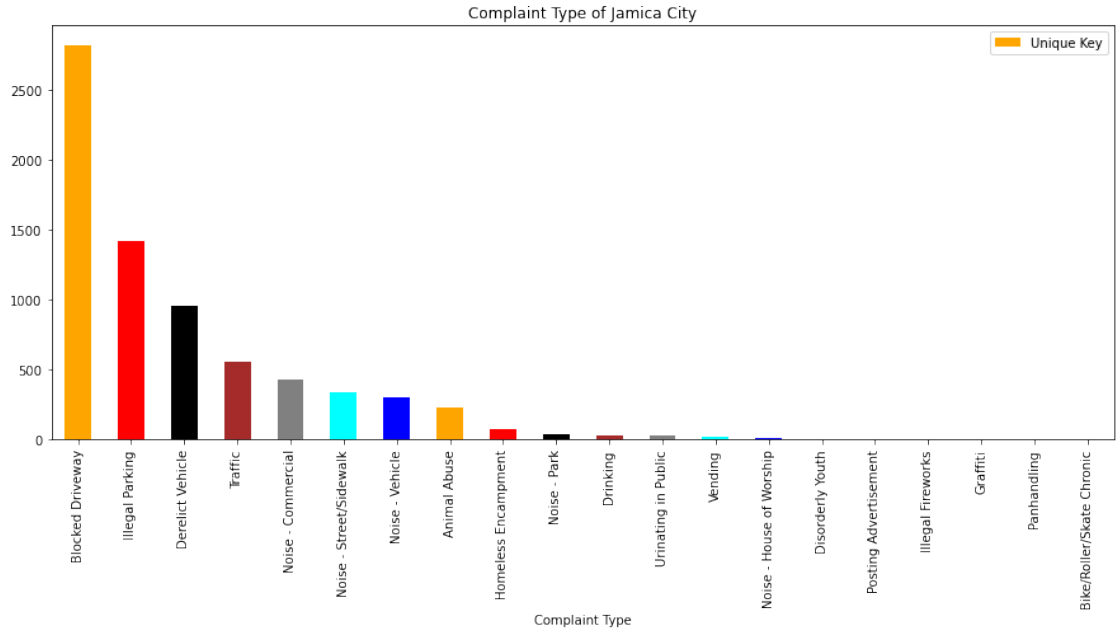
5.> Jamica

finding city column in Jamica city because this city has max complaints. Groping two column and count there value and drop null value in Complaint Type and again grouping Complaint Type and next arranging the value in decending order and next reset there index name. get values in this data frame and ploting in Bar chart.

result:- top 5 complaint 1.Blocked Driveway 2.Illegal parking 3. Derelict Vehicle 4. Traffic 5.Noise Commercial

```
Jamica=df[(df['City']=='JAMAICA')]
city_wise=Jamica.groupby(['Complaint Type'])['Unique Key'].count()
city_wise=Jamica.dropna(subset=['Complaint Type'])
city_wise=Jamica.groupby('Complaint Type')
sort_city = city_wise.size().sort_values(ascending = False)
sort_city = sort_city.to_frame('Unique Key').reset_index()

ec = ['orange','red','black','brown','gray','cyan','blue']
sort_city.plot(x='Complaint Type',y='Unique Key',kind='bar',title =
'Complaint Type of Jamica City',figsize=(15,6),color = ec)
plt.show()
```



Create Function and this function are create new column and assign Closed Date subtract Created Date values and assign in not null value and create new column week, month and year clear Request_Closing_Time column data last retun cleared data.

average time by Request_Closing_Time

```
def prepareData(df):
    df['Request_Closing_Time'] = (df['Closed Date'] - df['Created
Date']).dt.days
    RCT_Clean=df[df['Request_Closing_Time'].notnull()]
    df_cleaned = RCT_Clean[RCT_Clean['Closed Date'] >=
RCT_Clean['Created Date']]
    df_cleaned['Day of Week'] = df_cleaned['Created
Date'].dt.dayofweek
    df_cleaned['Day of Month'] = df_cleaned['Created Date'].dt.day
    df_cleaned['Month'] = df_cleaned['Created Date'].dt.month
    df_cleaned['Year'] = df_cleaned['Created Date'].dt.year
    df_cleaned=df_cleaned[df_cleaned.Borough!='Unspecified']
    return df_cleaned
```

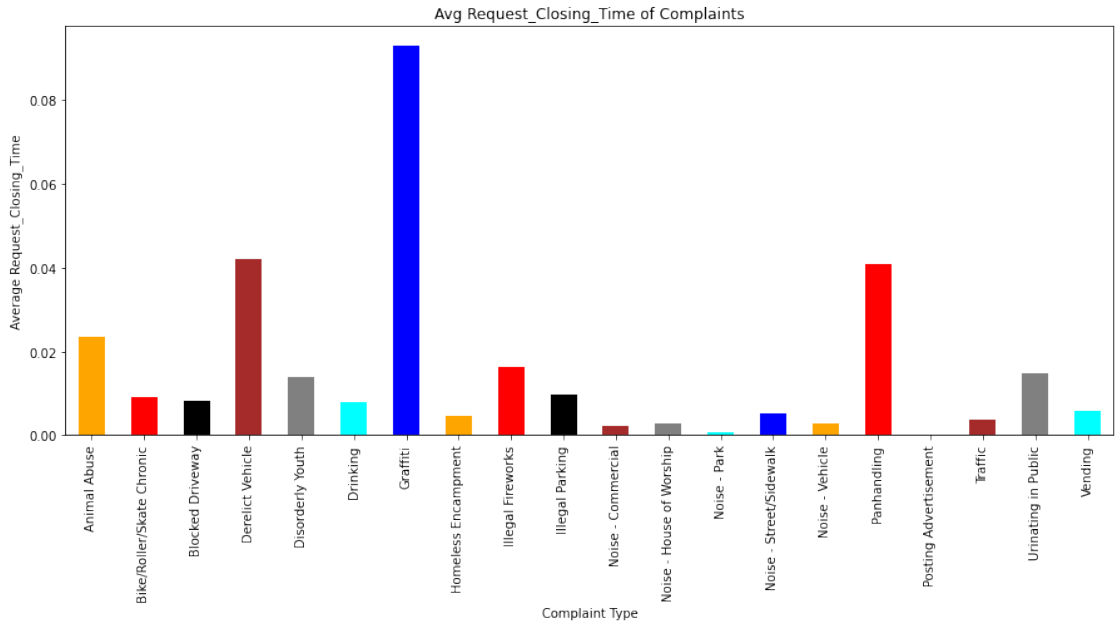
Pass the value of preparedata function name

```
df_cleaned = prepareData(df)
df_cleaned.shape
```

```
(298068, 58)
```

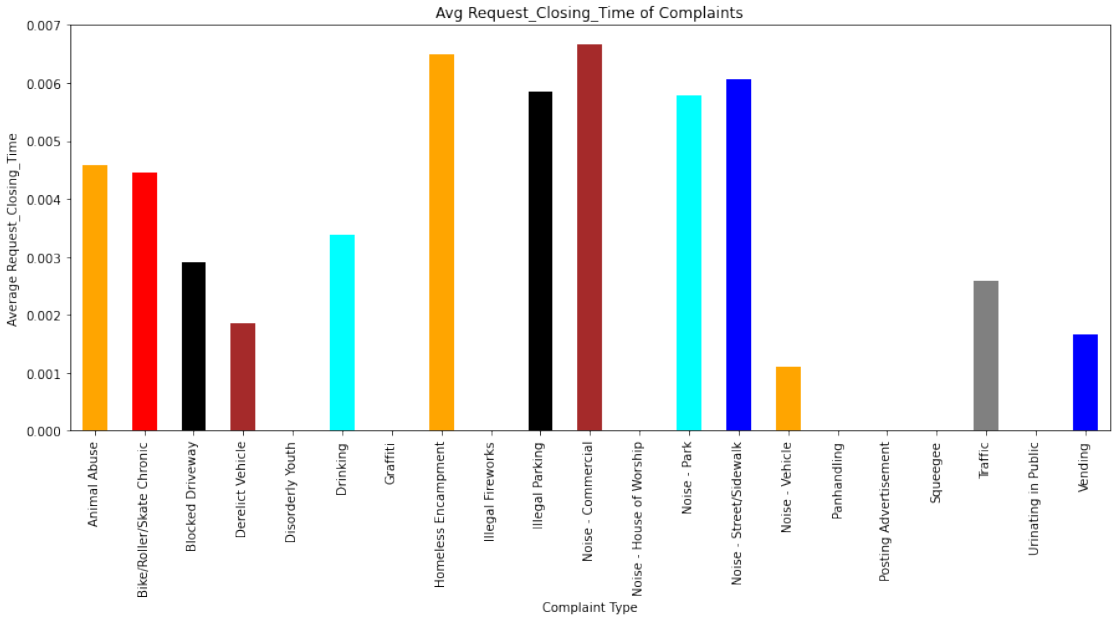
Finding Brooklyn this city in city column and grouping Complaint Type and Request_Closing_Time this mean value .and then count Complaint type then ploting figure and fig and subplot in 1,1,1 format and set label and title then last ploting x='Complaint Type',y='Request_Closing_Time' in bar graph and set color and show ploting.

```
Brooklyn=df[(df['City']=='BROOKLYN')]
var = Brooklyn.groupby('Complaint Type').Request_Closing_Time.mean()
frequent = Brooklyn['Complaint Type'].value_counts()
fig = plt.figure()
ax1 = fig.add_subplot(1,1,1)
ax1.set_xlabel('Complaint_Type')
ax1.set_ylabel('Average Request_Closing_Time')
ax1.set_title("Avg Request_Closing_Time of Complaints")
var.plot(x='Complaint
Type',y='Request_Closing_Time',kind='bar',figsize=(15,6),color = ec)
plt.show()
```



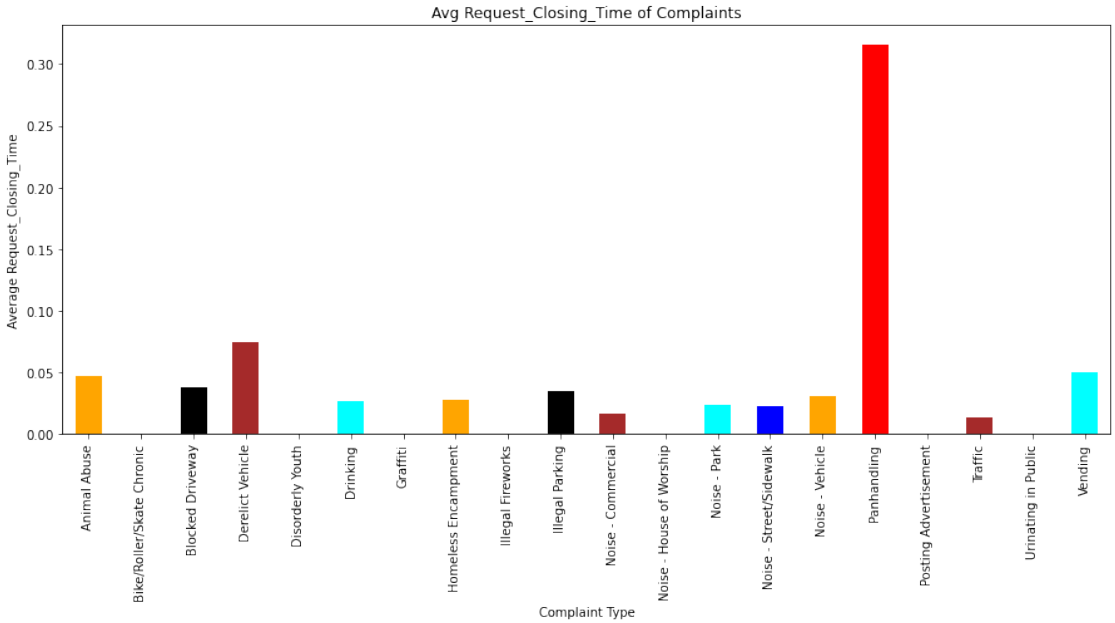
Finding New_York this city in city column and grouping Complaint Type and Request_Closing_Time this mean value .and then count Complaint type then ploting figure and fig and subplot in 1,1,1 format and set label and title then last ploting x='Complaint Type',y='Request_Closing_Time' in bar graph and set color and show ploting.

```
New_York=df[(df['City']=='NEW YORK')]
var = New_York.groupby('Complaint Type').Request_Closing_Time.mean()
frequent = New_York['Complaint Type'].value_counts()
fig = plt.figure()
ax1 = fig.add_subplot(1,1,1)
ax1.set_xlabel('Complaint_Type')
ax1.set_ylabel('Average Request_Closing_Time')
ax1.set_title("Avg Request_Closing_Time of Complaints")
var.plot(x='Complaint
Type',y='Request_Closing_Time',kind='bar',figsize=(15,6),color = ec)
plt.show()
```



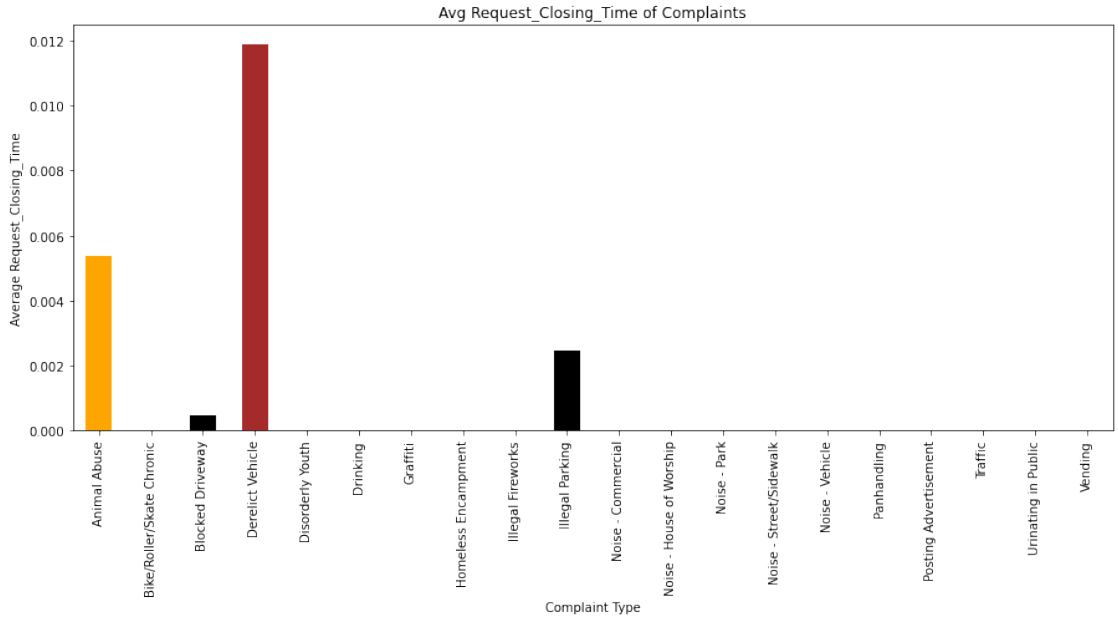
Finding Bronx this city in city column and grouping Complaint Type and Request_Closing_Time this mean value .and then count Complaint type then ploting figure and fig and subplot in 1,1,1 format and set label and title then last ploting x='Complaint Type',y='Request_Closing_Time' in bar graph and set color and show ploting.

```
Bronx=df[(df['City']=='BRONX')]
var = Bronx.groupby('Complaint Type').Request_Closing_Time.mean()
frequent = Bronx['Complaint Type'].value_counts()
fig = plt.figure()
ax1 = fig.add_subplot(1,1,1)
ax1.set_xlabel('Complaint_Type')
ax1.set_ylabel('Average Request_Closing_Time')
ax1.set_title("Avg Request_Closing_Time of Complaints")
var.plot(x='Complaint
Type',y='Request_Closing_Time',kind='bar',figsize=(15,6),color = ec)
plt.show()
```



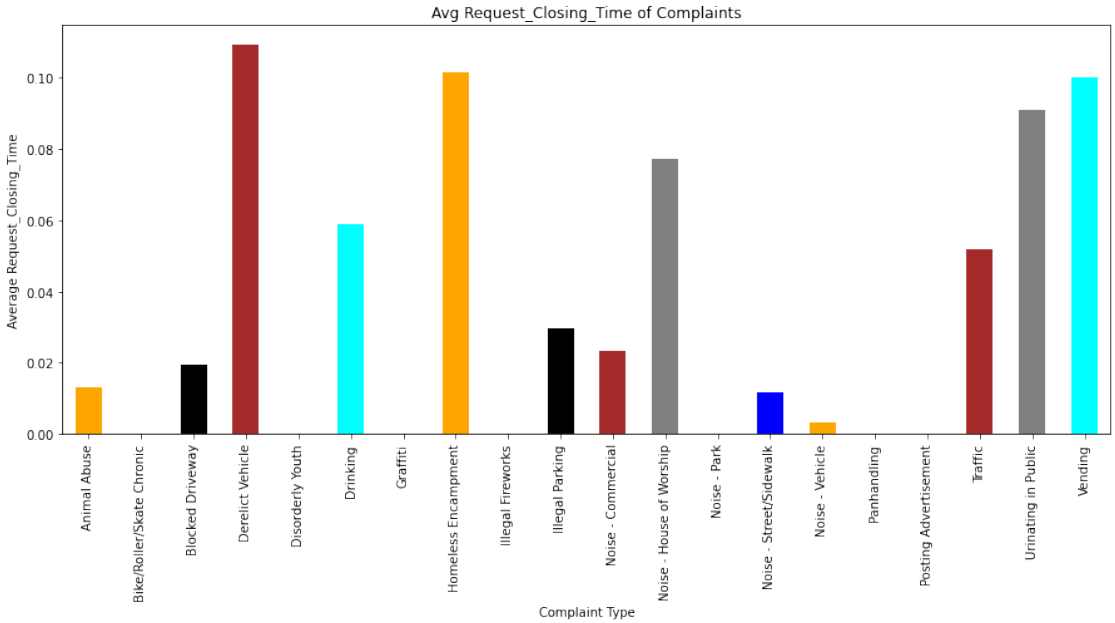
Finding Staten island this city in city column and grouping Complaint Type and Request_Closing_Time this mean value .and then count Complaint type then ploting figure and fig and subplot in 1,1,1 format and set label and title then last ploting x='Complaint Type',y='Request_Closing_Time' in bar graph and set color and show ploting.

```
Staten_island=df[(df['City']=='STATEN ISLAND')]
var = Staten_island.groupby('Complaint
Type').Request_Closing_Time.mean()
frequent = Staten_island['Complaint Type'].value_counts()
fig = plt.figure()
ax1 = fig.add_subplot(1,1,1)
ax1.set_xlabel('Complaint_Type')
ax1.set_ylabel('Average Request_Closing_Time')
ax1.set_title("Avg Request_Closing_Time of Complaints")
var.plot(x='Complaint
Type',y='Request_Closing_Time',kind='bar',figsize=(15,6),color = ec)
plt.show()
```



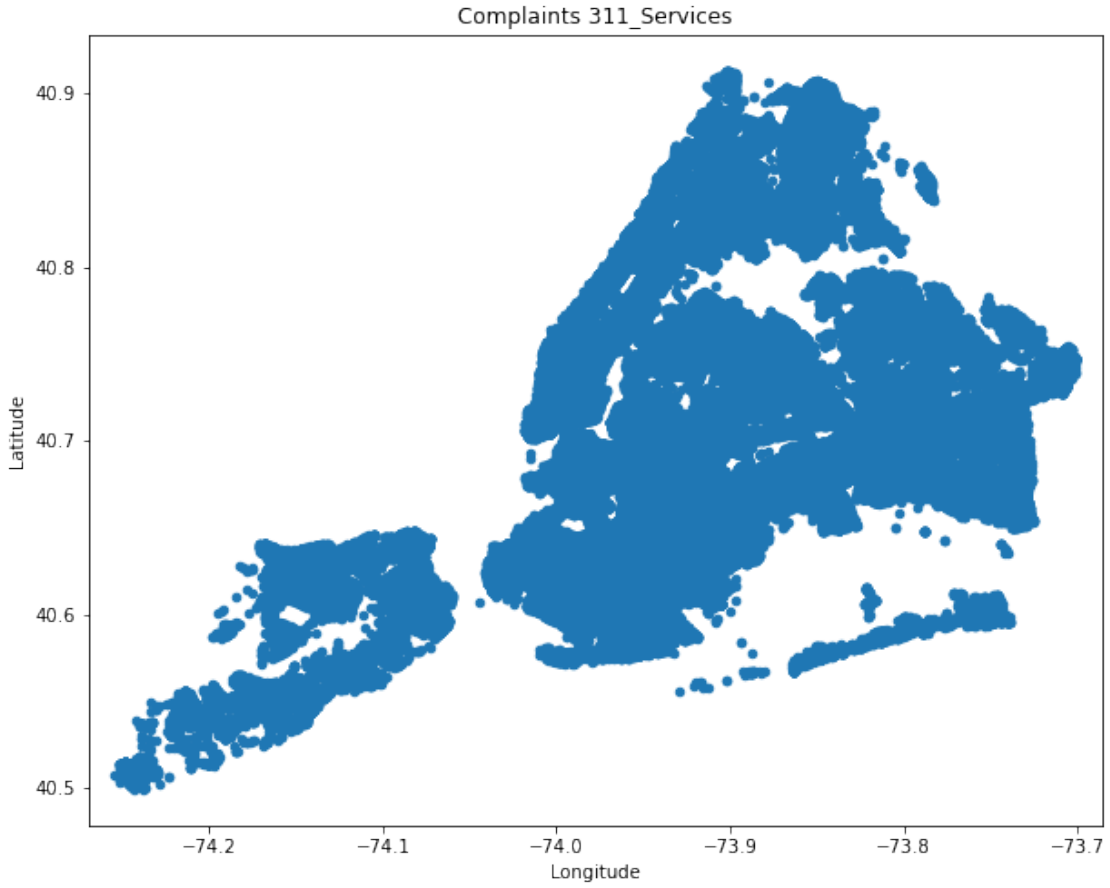
Finding Jamaica this city in city column and grouping Complaint Type and Request_Closing_Time this mean value .and then count Complaint type then ploting figure and fig and subplot in 1,1,1 format and set label and title then last ploting x='Complaint Type',y='Request_Closing_Time' in bar graph and set color and show ploting.

```
Jamaica=df[(df['City']=='JAMAICA')]
var = Jamaica.groupby('Complaint Type').Request_Closing_Time.mean()
frequent = Jamaica['Complaint Type'].value_counts()
fig = plt.figure()
ax1 = fig.add_subplot(1,1,1)
ax1.set_xlabel('Complaint_Type')
ax1.set_ylabel('Average Request_Closing_Time')
ax1.set_title("Avg Request_Closing_Time of Complaints")
var.plot(x='Complaint
Type',y='Request_Closing_Time',kind='bar',figsize=(15,6),color = ec)
plt.show()
```



get two column Longitude and Latitude and ploting in scatter graph and set figure size and set title and lebels then show.

```
df[['Longitude', 'Latitude']].plot(kind='scatter',x='Longitude',
y='Latitude',
                                   figsize=(10,8),title = 'Complaints
311_Services').axis('equal')
plt.show()
```



get two column Longitude and Latitude and ploting in hexbin graph and set figure size and set title and lebels then show.

```
df.plot(kind='hexbin', x='Longitude', y='Latitude',
gridsize=500,colormap = 'jet',mincnt=1,title = 'Air Quality issues
across NYC\n', figsize=(15,6)).axis('equal')
plt.show()
```