# Customer Service Requests Analysis.

Course-end Project 1

## DESCRIPTION – Problem Statement

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

**Note:**Download **311-service-requests-nyc.zip** file using the link given in the **Customer Service Requests Analysis** project problem statement and extract the **311\_Service\_Requests\_from\_2010\_to\_Present.csv**file

**Perform the following steps:**

1. Understand the dataset:
2. Identify the shape of the dataset
3. Identify variables with null values

2. Perform basic data exploratory analysis:

1. Utilize missing value treatment
2. Analyze the date column and remove the entries if it has an incorrect timeline
   1. Draw a frequency plot for city-wise complaints
   2. Draw scatter and hexbin plots for complaint concentration across Brooklyn

3. Find major types of complaints:

1. Plot a bar graph of count vs. complaint types
2. Find the top 10 types of complaints
3. Display the types of complaints in each city in a separate dataset

Created By – Selpa Singh

### Perform the following steps:

#### *Understand the dataset:*

In order to analyze the dataset we need to first import it, since the document format is “csv” we are going to use “read\_csv” from pandas library

*import pandas as pd*

*dataset1 = pd.read\_csv('311\_Service\_Requests\_from\_2010\_to\_Present.csv')*

Now our dataset content is imported for us to use .

Let’s see the first 5 rows of the dataset by using method head()

*dataset1.head()*

*#lookin at the 5 first rows information we can see there are several NAN values in*

*#bridge highway name/Direction/segment , road map, Garage lot name, ferry direction/terminal name*

Graphical user interface, text, application

Description automatically generated

#### *Identify the shape of the dataset*

This gives us some idea about the dataset we are dealing with, for example what columns wise information is included in this dataset, no of total columns. Now Let’s find out the total number of rows and columns in the dataset by using method ‘shape’

*Dataset1.shape*

*O/p - (364558, 53)*

#### *Identify variables with null values*

### We should check how if the dataset has any null values by applying ‘isnull’ on our dataset

dataset1.isnull

O/P:

<bound method DataFrame.isnull of 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

2 32309159 12/31/2015 11:59:29 PM 01/01/2016 04:51:03 AM NYPD

3 32305098 12/31/2015 11:57:46 PM 01/01/2016 07:43:13 AM NYPD

4 32306529 12/31/2015 11:56:58 PM 01/01/2016 03:24:42 AM NYPD

... ... ... ... ...

364553 29609918 01/01/2015 12:04:44 AM 01/01/2015 10:22:31 AM NYPD

364554 29608392 01/01/2015 12:04:28 AM 01/01/2015 02:25:02 AM NYPD

364555 29607589 01/01/2015 12:01:30 AM 01/01/2015 12:20:33 AM NYPD

364556 29610889 01/01/2015 12:01:29 AM 01/01/2015 02:42:22 AM NYPD

364557 29611816 01/01/2015 12:00:50 AM 01/01/2015 02:47:50 AM 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

... ... ...

364553 New York City Police Department Illegal Parking

364554 New York City Police Department Noise - Vehicle

364555 New York City Police Department Noise - Street/Sidewalk

364556 New York City Police Department Blocked Driveway

364557 New York City Police Department Blocked Driveway

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

... ... ... ...

364553 Blocked Hydrant Street/Sidewalk 11421.0

364554 Car/Truck Horn Street/Sidewalk 10468.0

364555 Loud Music/Party Street/Sidewalk 10031.0

364556 No Access Street/Sidewalk 10466.0

364557 No Access Street/Sidewalk 11420.0

Incident Address ... Bridge Highway Name \

0 71 VERMILYEA AVENUE ... NaN

1 27-07 23 AVENUE ... NaN

2 2897 VALENTINE AVENUE ... NaN

3 2940 BAISLEY AVENUE ... NaN

4 87-14 57 ROAD ... NaN

... ... ... ...

364553 84-25 85 ROAD ... NaN

364554 2555 SEDGWICK AVENUE ... NaN

364555 508 WEST 139 STREET ... NaN

364556 931 EAST 226 STREET ... NaN

364557 123-19 135 STREET ... NaN

Bridge Highway Direction Road Ramp Bridge Highway Segment \

0 NaN NaN NaN

1 NaN NaN NaN

2 NaN NaN NaN

3 NaN NaN NaN

4 NaN NaN NaN

... ... ... ...

364553 NaN NaN NaN

364554 NaN NaN NaN

364555 NaN NaN NaN

364556 NaN NaN NaN

364557 NaN NaN NaN

Garage Lot Name Ferry Direction Ferry Terminal Name Latitude \

0 NaN NaN NaN 40.865682

1 NaN NaN NaN 40.775945

2 NaN NaN NaN 40.870325

3 NaN NaN NaN 40.835994

4 NaN NaN NaN 40.733060

... ... ... ... ...

364553 NaN NaN NaN 40.695145

364554 NaN NaN NaN 40.867830

364555 NaN NaN NaN 40.821647

364556 NaN NaN NaN 40.886361

364557 NaN NaN NaN 40.674212

Longitude Location

0 -73.923501 (40.86568153633767, -73.92350095571744)

1 -73.915094 (40.775945312321085, -73.91509393898605)

2 -73.888525 (40.870324522111424, -73.88852464418646)

3 -73.828379 (40.83599404683083, -73.82837939584206)

4 -73.874170 (40.733059618956815, -73.87416975810375)

... ... ...

364553 -73.860949 (40.69514470265117, -73.86094888534394)

364554 -73.907178 (40.86782963689454, -73.90717786644662)

364555 -73.950873 (40.821646626438095, -73.95087342885292)

364556 -73.853290 (40.88636077906953, -73.85329048666742)

364557 -73.803585 (40.674211762243935, -73.80358548685278)

[364558 rows x 53 columns]>

Well the output has several null values and its not easy to comprehend. So lets also check the sum of all null values across rows by using ‘dataset1.isnull().sum(axis=0)’.

dataset1.isnull().sum(axis=0)

OP:

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

### Perform basic data exploratory analysis:

#### Utilize missing value treatment

Now since we have total number of rows and columns in the dataset, we will check if the dataset has any NAN values, which we can see above that have got some NAN value. In order to identify which rows /columns are more suitable for our operation we must find out number of NAN values for each column, if the total number of NAN values are less than 30% , we can drop them and continue out investigation on those columns. If total number of NAN values in any column is close to the number of rows in dataset then those columns are not useful for us, we can simply drop them to make dataset more precise.

data\_null = dataset1.isna()

#for checking all NA values we created new dataset data\_null, lets print it and see what information it contains

data\_null

#looking at the result we can confirm that isna becomes true only for

#bridge highway name/Direction/segment , road map, Garage lot name, ferry direction/terminal name

Table

Description automatically generated

*no\_of\_na\_in\_Dataset1 = dataset1.isna().sum(axis=0)*

*no\_of\_na\_in\_Dataset1*

*OP:*

*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*

We can see the several columns have NAN values close to the total number of rows. As part of missing values treatment we can remove those values or we can remove those columns entirely since its not useful for us with the amount of NAN values. Since the values are across column , we will drop them across axis 1

*#Now that we have all the missing values identified, we see that few columns have huge no of missing values*

*#those columns are not useful for our analysis, lets drop them along axis 1 i.e. column*

*dataset1.drop('Intersection Street 1',axis=1,inplace=True)*

*dataset1.drop('Intersection Street 2',axis=1,inplace=True)*

*dataset1.drop('Landmark',axis=1,inplace=True)*

*dataset1.drop('School or Citywide Complaint',axis=1,inplace=True)*

*dataset1.drop('Vehicle Type',axis=1,inplace=True)*

*dataset1.drop('Taxi Company Borough',axis=1,inplace=True)*

*dataset1.drop('Taxi Pick Up Location',axis=1,inplace=True)*

*dataset1.drop('Bridge Highway Name',axis=1,inplace=True)*

*dataset1.drop('Bridge Highway Direction',axis=1,inplace=True)*

*dataset1.drop('Road Ramp',axis=1,inplace=True)*

*dataset1.drop('Bridge Highway Segment',axis=1,inplace=True)*

*dataset1.drop('Garage Lot Name',axis=1,inplace=True)*

*dataset1.drop('Ferry Direction',axis=1,inplace=True)*

*dataset1.drop('Ferry Terminal Name',axis=1,inplace=True)*

*dataset1.info()*

*#now as a result we see that all the columns which contains mostly NA values are removed.*

*OP:*

*<class 'pandas.core.frame.DataFrame'>*

*RangeIndex: 364558 entries, 0 to 364557*

*Data columns (total 39 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 Address Type 361306 non-null object*

*14 City 361561 non-null object*

*15 Facility Type 362169 non-null object*

*16 Status 364558 non-null object*

*17 Due Date 364555 non-null object*

*18 Resolution Description 364558 non-null object*

*19 Resolution Action Updated Date 362156 non-null object*

*20 Community Board 364558 non-null object*

*21 Borough 364558 non-null object*

*22 X Coordinate (State Plane) 360528 non-null float64*

*23 Y Coordinate (State Plane) 360528 non-null float64*

*24 Park Facility Name 364558 non-null object*

*25 Park Borough 364558 non-null object*

*26 School Name 364558 non-null object*

*27 School Number 364558 non-null object*

*28 School Region 364557 non-null object*

*29 School Code 364557 non-null object*

*30 School Phone Number 364558 non-null object*

*31 School Address 364558 non-null object*

*32 School City 364558 non-null object*

*33 School State 364558 non-null object*

*34 School Zip 364557 non-null object*

*35 School Not Found 364558 non-null object*

*36 Latitude 360528 non-null float64*

*37 Longitude 360528 non-null float64*

*38 Location 360528 non-null object*

*dtypes: float64(5), int64(1), object(33)*

*memory usage: 108.5+ MB*

Let’s analyze the columns “*Created Date” and “Closed Date”*

*dataset1['Created Date']*

*OP:*

*0 12/31/2015 11:59:45 PM*

*1 12/31/2015 11:59:44 PM*

*2 12/31/2015 11:59:29 PM*

*3 12/31/2015 11:57:46 PM*

*4 12/31/2015 11:56:58 PM*

*...*

*364553 01/01/2015 12:04:44 AM*

*364554 01/01/2015 12:04:28 AM*

*364555 01/01/2015 12:01:30 AM*

*364556 01/01/2015 12:01:29 AM*

*364557 01/01/2015 12:00:50 AM*

*Name: Created Date, Length: 364558, dtype: object*

*dataset1['Closed Date']*

*OP:*

*0 01/01/2016 12:55:15 AM*

*1 01/01/2016 01:26:57 AM*

*2 01/01/2016 04:51:03 AM*

*3 01/01/2016 07:43:13 AM*

*4 01/01/2016 03:24:42 AM*

*...*

*364553 01/01/2015 10:22:31 AM*

*364554 01/01/2015 02:25:02 AM*

*364555 01/01/2015 12:20:33 AM*

*364556 01/01/2015 02:42:22 AM*

*364557 01/01/2015 02:47:50 AM*

*Name: Closed Date, Length: 364558, dtype: object*

Let’s analyze if any of the data entries are invalid. For invalid check we can perform below mentioned operations

* Check if closed Data is before created data, if so print “Some closed dates are before created date : let's drop them” if no then print “All closed dates are after created date” – and based on print lets decide to drop them or not
* We have seen above that in “closed date” column , there are 2381 NAN values, let’s drop these values

*#import numpy as np*

*if(dataset1['Closed Date'].any() < dataset1['Created Date'].any()):*

*print("Some closed dates are before created date : let's drop them")*

*else:*

*print('All closed dates are after created date')*

*OP: All closed dates are after created date*

*dataset1['Closed Date'].dropna()*

*OP: 0 01/01/2016 12:55:15 AM*

*1 01/01/2016 01:26:57 AM*

*2 01/01/2016 04:51:03 AM*

*3 01/01/2016 07:43:13 AM*

*4 01/01/2016 03:24:42 AM*

*...*

*364553 01/01/2015 10:22:31 AM*

*364554 01/01/2015 02:25:02 AM*

*364555 01/01/2015 12:20:33 AM*

*364556 01/01/2015 02:42:22 AM*

*364557 01/01/2015 02:47:50 AM*

*Name: Closed Date, Length: 362177, dtype: object*

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

##### Draw a frequency plot for city-wise complaints

##### Draw scatter and hexbin plots for complaint concentration across Brooklyn

In above Output we can see that earlier we had “2381” NAN objects and “362177” non-NAN objects in column “Closed Date”, and after treating missing values we are left with total ‘362177’ objects in ‘Closed Date’ column, so our missing values or invalid values in this column are treated now.

Let us plot the bar plot for complaints across all the cities – We can understand the pattern – which cities have move number of complaints and which cities have less complaints in comparison.

We can use seaborn library to see the countplot – this will plot a chart with number of times a complaint is made in that city.

*%matplotlib inline*

*import seaborn as sns*

*plt.figure(figsize=(20,10))*

*sns.countplot('City', data=dataset1)*

A picture containing chart

Description automatically generated

Here in this plot names of cities at the bottom are not clear, so we should increase x axis length. With the plt.figure(figsize(100,50))

We can see the highest count is for Brooklyn city.

Now we want to see top 10 complaints type

Chart, histogram

Description automatically generated

Now let us plot Hexbin plot and scatter plot for Brooklyn city since there are highest number of complaints in Brooklyn. We can understand the pattern and concentration of complaints. We have already imported seaborn for countplot. We will be using same library for hexbin plot as well. We can use lmplot from seaborn to

Graphical user interface, text, application

Description automatically generated

From the above plot we can derive a conclusion:

1. City with Highest number of complaints: Brooklyn
2. Type of Highest number of complaints: Blocked Driveway
3. There are very few or no complains about Ferry, Agency issue, Squeegee and animal in park

### Find major types of complaints:

#### Plot a bar graph of count vs. complaint types

Now let us analyze the type of complaints with plots:

To plot the count of each complaint type we will use countplot from seaborn.

*%matplotlib inline*

*import seaborn as sns*

*plt.figure(figsize=(20,10))*

*sns.countplot('Complaint Type', data=dataset1)*

Chart

Description automatically generated

#### Find the top 10 types of complaints

Let us first find out how many complaint types there are in the dataset with count of such complaints. We can use value\_counts() method from panadas, it will give us a list of all the complaint types and total number of complains of that category. Once we have this information we can read the top 10 values from this newly created dataframe.

*type\_of\_complaint = dataset1['Complaint Type'].value\_counts()*

*type\_of\_complaint*

*OP:*

*Blocked Driveway 100881*

*Illegal Parking 92679*

*Noise - Street/Sidewalk 51692*

*Noise - Commercial 44109*

*Derelict Vehicle 21661*

*Noise - Vehicle 19352*

*Animal Abuse 10541*

*Traffic 5198*

*Homeless Encampment 4879*

*Vending 4192*

*Noise - Park 4109*

*Drinking 1409*

*Noise - House of Worship 1070*

*Posting Advertisement 681*

*Urinating in Public 641*

*Bike/Roller/Skate Chronic 478*

*Panhandling 327*

*Disorderly Youth 315*

*Illegal Fireworks 172*

*Graffiti 157*

*Agency Issues 8*

*Squeegee 4*

*Ferry Complaint 2*

*Animal in a Park 1*

*Name: Complaint Type, dtype: int64*

Now lets check the shape of all\_cities to know the total count of unique cities

*type\_of\_complaint.head(10)*

*OP:*

*Blocked Driveway 100881*

*Illegal Parking 92679*

*Noise - Street/Sidewalk 51692*

*Noise - Commercial 44109*

*Derelict Vehicle 21661*

*Noise - Vehicle 19352*

*Animal Abuse 10541*

*Traffic 5198*

*Homeless Encampment 4879*

*Vending 4192*

*Name: Complaint Type, dtype: int64*

Based on result we can say that “Blocked Driveway”, Illegal Parking”, “Noise - Street/Sidewalk”, “Noise-Commercial”, “Derelict Vehicle”, “Noise-Vehicle”, “Animal Abuse” , “Traffic, “Homeless Encampment”, and “Vending are the top 10 complaint types, regardless of the cities.

#### Display the types of complaints in each city in a separate dataset

Lets create a new dataframe which contains only 2 columns of our original dataset

*dataframe = dataset1[['City', 'Complaint Type']].copy()*

*dataframe*

*OP:*

| *City* | *Complaint Type* |
| --- | --- |
| *0* | *NEW YORK* | *Noise - Street/Sidewalk* |
| *1* | *ASTORIA* | *Blocked Driveway* |
| *2* | *BRONX* | *Blocked Driveway* |
| *3* | *BRONX* | *Illegal Parking* |
| *4* | *ELMHURST* | *Illegal Parking* |
| *...* | *...* | *...* |
| *364553* | *WOODHAVEN* | *Illegal Parking* |
| *364554* | *BRONX* | *Noise - Vehicle* |
| *364555* | *NEW YORK* | *Noise - Street/Sidewalk* |
| *364556* | *BRONX* | *Blocked Driveway* |
| *364557* | *SOUTH OZONE PARK* | *Blocked Driveway* |

*364558 rows × 2 columns*

How our new dataset named- ‘dataframe’ contains list of all the cities and all the complaints. So we need to create another dataset which will have a sum of all

*final\_df = dataframe.groupby('City')*

*final\_df.ngroups*

*OP: 53*

Now we have created 53 datasets, and each dataset is based on unique element in “City” column in original dataset

Since there are 53 datasets created we will display first and last element of each dataset, also 5 of the city wise dataset for confirmation, let’s select few famous cities example, we will check only those 5 cities dataset – ‘NEW YORK’ , ‘BROOKLYN’, ‘QUEENS’, ‘LONG ISLAND CITY’, ‘CENTRAL PARK’

*final\_df.first()*

*OP:*

Graphical user interface, application

Description automatically generated

*final\_df.last()*

*OP:*

Graphical user interface

Description automatically generated with medium confidence

*df\_new\_york = final\_df.get\_group('NEW YORK')*

*df\_new\_york*

| *City* | *Complaint Type* |
| --- | --- |
| *0* | *NEW YORK* | *Noise - Street/Sidewalk* |
| *6* | *NEW YORK* | *Illegal Parking* |
| *19* | *NEW YORK* | *Noise - Street/Sidewalk* |
| *23* | *NEW YORK* | *Illegal Parking* |
| *26* | *NEW YORK* | *Noise - House of Worship* |
| *...* | *...* | *...* |
| *364542* | *NEW YORK* | *Noise - Street/Sidewalk* |
| *364543* | *NEW YORK* | *Noise - Street/Sidewalk* |
| *364547* | *NEW YORK* | *Noise - Street/Sidewalk* |
| *364552* | *NEW YORK* | *Noise - Street/Sidewalk* |
| *364555* | *NEW YORK* | *Noise - Street/Sidewalk* |

*77312 rows × 2 columns*

*df\_BROOKLYN = final\_df.get\_group('BROOKLYN')*

*df\_BROOKLYN*

| *City* | *Complaint Type* |
| --- | --- |
| *5* | *BROOKLYN* | *Illegal Parking* |
| *9* | *BROOKLYN* | *Blocked Driveway* |
| *13* | *BROOKLYN* | *Illegal Parking* |
| *17* | *BROOKLYN* | *Noise - Commercial* |
| *18* | *BROOKLYN* | *Noise - Commercial* |
| *...* | *...* | *...* |
| *364539* | *BROOKLYN* | *Blocked Driveway* |
| *364541* | *BROOKLYN* | *Blocked Driveway* |
| *364544* | *BROOKLYN* | *Noise - Commercial* |
| *364545* | *BROOKLYN* | *Blocked Driveway* |
| *364546* | *BROOKLYN* | *Blocked Driveway* |

*118862 rows × 2 columns*

*df\_QUEENS = final\_df.get\_group('QUEENS')*

*df\_QUEENS*

| *City* | *Complaint Type* |
| --- | --- |
| *26409* | *QUEENS* | *Noise - Commercial* |
| *27634* | *QUEENS* | *Illegal Parking* |
| *59425* | *QUEENS* | *Noise - Street/Sidewalk* |
| *59666* | *QUEENS* | *Noise - House of Worship* |
| *62535* | *QUEENS* | *Derelict Vehicle* |
| *98539* | *QUEENS* | *Noise - Street/Sidewalk* |
| *99755* | *QUEENS* | *Noise - Commercial* |
| *120873* | *QUEENS* | *Illegal Parking* |
| *120884* | *QUEENS* | *Blocked Driveway* |
| *125847* | *QUEENS* | *Illegal Parking* |
| *132012* | *QUEENS* | *Noise - Street/Sidewalk* |
| *132261* | *QUEENS* | *Blocked Driveway* |
| *135337* | *QUEENS* | *Noise - Commercial* |
| *147623* | *QUEENS* | *Noise - Street/Sidewalk* |
| *177755* | *QUEENS* | *Traffic* |
| *184777* | *QUEENS* | *Homeless Encampment* |
| *185726* | *QUEENS* | *Noise - Street/Sidewalk* |
| *186164* | *QUEENS* | *Illegal Parking* |
| *191695* | *QUEENS* | *Homeless Encampment* |
| *195991* | *QUEENS* | *Noise - Vehicle* |
| *197950* | *QUEENS* | *Noise - Commercial* |
| *198570* | *QUEENS* | *Urinating in Public* |
| *213695* | *QUEENS* | *Traffic* |
| *214984* | *QUEENS* | *Noise - Commercial* |
| *224916* | *QUEENS* | *Illegal Parking* |
| *226472* | *QUEENS* | *Noise - Vehicle* |
| *228376* | *QUEENS* | *Noise - Commercial* |
| *265802* | *QUEENS* | *Illegal Parking* |
| *281454* | *QUEENS* | *Noise - Street/Sidewalk* |
| *283132* | *QUEENS* | *Animal in a Park* |
| *287169* | *QUEENS* | *Illegal Parking* |
| *297676* | *QUEENS* | *Illegal Parking* |
| *301778* | *QUEENS* | *Blocked Driveway* |
| *335646* | *QUEENS* | *Illegal Parking* |
| *343060* | *QUEENS* | *Illegal Parking* |
| *348390* | *QUEENS* | *Animal Abuse* |
| *356659* | *QUEENS* | *Derelict Vehicle* |

*df\_LONG\_ISLAND\_CITY = final\_df.get\_group('LONG ISLAND CITY')*

*df\_LONG\_ISLAND\_CITY*

| *City* | *Complaint Type* |
| --- | --- |
| *177* | *LONG ISLAND CITY* | *Illegal Parking* |
| *505* | *LONG ISLAND CITY* | *Blocked Driveway* |
| *555* | *LONG ISLAND CITY* | *Blocked Driveway* |
| *626* | *LONG ISLAND CITY* | *Illegal Parking* |
| *1612* | *LONG ISLAND CITY* | *Blocked Driveway* |
| *...* | *...* | *...* |
| *363529* | *LONG ISLAND CITY* | *Illegal Parking* |
| *363636* | *LONG ISLAND CITY* | *Blocked Driveway* |
| *363638* | *LONG ISLAND CITY* | *Blocked Driveway* |
| *364236* | *LONG ISLAND CITY* | *Blocked Driveway* |
| *364432* | *LONG ISLAND CITY* | *Blocked Driveway* |

*3028 rows × 2 columns*

*df\_CENTRAL\_PARK = final\_df.get\_group('CENTRAL PARK')*

*df\_CENTRAL\_PARK*

| *City* | *Complaint Type* |
| --- | --- |
| *17122* | *CENTRAL PARK* | *Noise - Street/Sidewalk* |
| *18527* | *CENTRAL PARK* | *Noise - Street/Sidewalk* |
| *20887* | *CENTRAL PARK* | *Noise - Street/Sidewalk* |
| *23912* | *CENTRAL PARK* | *Noise - Street/Sidewalk* |
| *24915* | *CENTRAL PARK* | *Noise - Street/Sidewalk* |
| *...* | *...* | *...* |
| *334882* | *CENTRAL PARK* | *Illegal Parking* |
| *335666* | *CENTRAL PARK* | *Illegal Parking* |
| *338556* | *CENTRAL PARK* | *Noise - Street/Sidewalk* |
| *342303* | *CENTRAL PARK* | *Noise - Street/Sidewalk* |
| *362137* | *CENTRAL PARK* | *Noise - Street/Sidewalk* |

*110 rows × 2 columns*