Customer Service Requests Analysis

Background of Problem Statement:

NYC 311's mission is to provide the public with quick and easy access to all New York City government services and information while offering the best customer service. Each day, NYC311 receives thousands of requests related to several hundred types of non-emergency services, including noise complaints, plumbing issues, and illegally parked cars. These requests are received by NYC311 and forwarded to the relevant agencies such as the police, buildings, or transportation. The agency responds to the request, addresses it, and then closes it.

Problem Objective:

Perform a service request data analysis of New York City 311 calls. You will focus on the data wrangling techniques to understand the pattern in the data and also visualize the major complaint types.
Domain: Customer Service

Analysis Tasks to be performed:

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

1. Import a 311 NYC service request.
2. Read or convert the columns 'Created Date' and Closed Date' to datetime datatype and create a new column 'Request_Closing_Time' as the time elapsed between request creation and request closing. (Hint: Explore the package/module datetime)
3. Provide major insights/patterns that you can offer in a visual format (graphs or tables); at least 4 major conclusions that you can come up with after generic data mining.
4. Order the complaint types based on the average 'Request_Closing_Time', grouping them for different locations.
5. Perform a statistical test for the following:

Please note: For the below statements you need to state the Null and Alternate and then provide a statistical test to accept or reject the Null Hypothesis along with the corresponding 'p-value'.

- Whether the average response time across complaint types is similar or not (overall)
- Are the type of complaint or service requested and location related?

Solution for this problem statement

1) ### import libraries

```
#for linear algebra
import numpy as np
# for data processing session
import pandas as pd
#for data visualization
import matplotlib.pyplot as plt
from matplotlib import style
#seaborn is also data visualization library built on top of the matplotlib
import seaborn as sns
```

Customer Service Requests Analysis

#now we use matplotlib inline which is used for the output of plotting commands is displayed inline within frontends like jupitar notebook.
#Mainly used for inline plotting
%matplotlib inline

```
### import libraries
#for linear algebra
import numpy as np
# for data processing session
import pandas as pd
#for data visualization
import matplotlib.pyplot as plt
from matplotlib import style
#seaborn is also data visualization library built on top of the matplotlib
import seaborn as sns
#now we use matplotlib inline which is used for the output of plotting commands is displayed inline within frontends like
#Mainly used for inline plotting
%matplotlib inline
```

2) #Now import the csv file
Service_Request_csv = pd.read_csv("311_Service_Requests_from_2010_to_Present.csv")

```
#Now import the csv file
Service_Request_csv = pd.read_csv("311_Service_Requests_from_2010_to_Present.csv")
```

```
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2718: DtypeWarning: Columns (48,49) have mixed type
  interactivity=interactivity, compiler=compiler, result=result)
```

3) #now showing the first five rows of the service request csv
Service_Request_csv.head()

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	Incident Zip	Incident Address	Stree Nar
0	32310363	12/31/2015 11:59:45 PM	01-01-16 0:55	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Music/Party	Street/Sidewalk	10034.0	71 VERMILYEA AVENUE	VERMILYE AVENU
1	32309934	12/31/2015 11:59:44 PM	01-01-16 1:26	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewalk	11105.0	27-07 23 AVENUE	23 AVENU
2	32309159	12/31/2015 11:59:29 PM	01-01-16 4:51	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewalk	10458.0	2897 VALENTINE AVENUE	VALENTIN AVENU
3	32305098	12/31/2015 11:57:46 PM	01-01-16 7:43	NYPD	New York City Police Department	Illegal Parking	Commercial Overnight Parking	Street/Sidewalk	10461.0	2940 BAISLEY AVENUE	BAISLE AVENU

4) #now showing the last five rows of the service request csv
Service_Request_csv.tail()

Customer Service Requests Analysis

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	Incident Zip	Incident Address
300693	30281872	03/29/2015 12:33:41 AM	NaN	NYPD	New York City Police Department	Noise - Commercial	Loud Music/Party	Club/Bar/Restaurant	NaN	CRESCENT AVENUE
300694	30281230	03/29/2015 12:33:28 AM	03/29/2015 02:33:59 AM	NYPD	New York City Police Department	Blocked Driveway	Partial Access	Street/Sidewalk	11418.0	100-17 87 AVENUE
300695	30283424	03/29/2015 12:33:03 AM	03/29/2015 03:40:20 AM	NYPD	New York City Police Department	Noise - Commercial	Loud Music/Party	Club/Bar/Restaurant	11206.0	162 THROOP AVENUE
300696	30280004	03/29/2015 12:33:02 AM	03/29/2015 04:38:35 AM	NYPD	New York City Police Department	Noise - Commercial	Loud Music/Party	Club/Bar/Restaurant	10461.0	3151 EAST TREMONT AVENUE
		03/29/2015	03/29/2015		New York					

5) #how many rows & columns are present in this service request dataset
Service_Request_csv.shape

```
#how many rows & columns are present in this service request dataset
Service_Request_csv.shape
```

```
(300698, 53)
```

6) #how many columns are present inside the service request dataset
print(Service_Request_csv.columns.to_list())

['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', 'Intersection Street 1', 'Intersection Street 2', 'Address Type', 'City', 'Landmark', 'Facility Type', 'Status', 'Due Date', 'Resolution Description', 'Resolution Action Updated Date', 'Community Board', 'Borough', 'X Coordinate (State Plane)', 'Y Coordinate (State Plane)', 'Park Facility Name', 'Park Borough', 'School Name', 'School Number', 'School Region', 'School Code', 'School Phone Number', 'School Address', 'School City', 'School State', 'School Zip', 'School Not Found', 'School or Citywide Complaint', 'Vehicle Type', 'Taxi Company Borough', 'Taxi Pick Up Location', 'Bridge Highway Name', 'Bridge Highway Direction', 'Road Ramp', 'Bridge Highway Segment', 'Garage Lot Name', 'Ferry Direction', 'Ferry Terminal Name', 'Latitude', 'Longitude', 'Location']

```
#how many columns are present inside the service request dataset
print(Service_Request_csv.columns.to_list())
```

```
['Unique Key', 'Created Date', 'Closed Date', 'Agency', 'Agency Name', 'Complaint Type', 'Descriptor', 'Location Type', 'Ir
```

7) #how many unique columns are present inside service request dataset for the c olumn name "Complaint Type"
#The unique() function is used to find the unique elements of an array
Service_Request_csv["Complaint Type"].unique()

Customer Service Requests Analysis

```
#how many unique columns are present inside service request dataset for the column name "Complaint Type"
#The unique() function is used to find the unique elements of an array
Service_Request_csv["Complaint Type"].unique()
```

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

8) #how many unique columns are present inside service request dataset for the column name "Descriptor"

```
#The unique() function is used to find the unique elements of an array
Service_Request_csv["Descriptor"].unique()
```

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


```
#how many unique columns are present inside service request dataset for the column name "Descriptor"
#The unique() function is used to find the unique elements of an array
Service_Request_csv["Descriptor"].unique()
```



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


9) #how many missing values are present or not inside this dataset

Customer Service Requests Analysis

```
#using isNa function
Service_Request_csv.isna().any()
```

```
Unique Key                      False
Created Date                    False
Closed Date                      True
Agency                          False
Agency Name                     False
Complaint Type                  False
Descriptor                       True
Location Type                    True
Incident Zip                     True
Incident Address                 True
Street Name                      True
Cross Street 1                   True
Cross Street 2                   True
Intersection Street 1            True
Intersection Street 2            True
Address Type                     True
City                             True
Landmark                         True
Facility Type                    True
Status                          False
Due Date                         True
Resolution Description          False
Resolution Action Updated Date   True
Community Board                 False
Borough                         False
X Coordinate (State Plane)       True
Y Coordinate (State Plane)       True
Park Facility Name              False
Park Borough                    False
School Name                     False
School Number                   False
School Region                    True
School Code                      True
School Phone Number             False
School Address                  False
School City                     False
School State                    False
School Zip                       True
School Not Found                False
School or Citywide Complaint     True
Vehicle Type                     True
Taxi Company Borough             True
Taxi Pick Up Location            True
Bridge Highway Name              True
Bridge Highway Direction         True
Road Ramp                        True
Bridge Highway Segment           True
```

```
Garage Lot Name                 True
Ferry Direction                 True
Ferry Terminal Name             True
Latitude                        True
Longitude                       True
Location                        True
dtype: bool
```

10) #total missing values are present inside this dataset
Service_Request_csv.isna().sum()

```
Unique Key                      0
Created Date                    0
Closed Date                  2164
Agency                          0
Agency Name                     0
Complaint Type                  0
Descriptor                   5914
Location Type                 131
Incident Zip                 2615
Incident Address            44410
Street Name                 44410
Cross Street 1              49279
Cross Street 2              49779
Intersection Street 1      256840
Intersection Street 2      257336
Address Type                 2815
City                         2614
Landmark                   300349
Facility Type                2171
Status                          0
Due Date                        3
Resolution Description          0
Resolution Action Updated Date  2187
Community Board                 0
Borough                         0
X Coordinate (State Plane)   3540
Y Coordinate (State Plane)   3540
Park Facility Name              0
Park Borough                    0
School Name                     0
School Number                   0
School Region                   1
School Code                     1
School Phone Number             0
School Address                  0
School City                     0
School State                    0
School Zip                      1
School Not Found                0
```

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

```
11) #information of the Service_Request_csv dataframe
Service_Request_csv.info()
```

```
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
```

12) #check the datatype of the dataset
Service_Request_csv.dtypes

```
Unique Key                      int64
Created Date                   object
Closed Date                    object
Agency                         object
Agency Name                    object
Complaint Type                 object
Descriptor                     object
Location Type                  object
Incident Zip                  float64
Incident Address               object
Street Name                    object
Cross Street 1                 object
Cross Street 2                 object
Intersection Street 1          object
Intersection Street 2          object
Address Type                   object
City                           object
Landmark                       object
```

```
Facility Type                    object
Status                           object
Due Date                         object
Resolution Description           object
Resolution Action Updated Date   object
Community Board                  object
Borough                          object
X Coordinate (State Plane)       float64
Y Coordinate (State Plane)       float64
Park Facility Name               object
Park Borough                     object
School Name                      object
School Number                    object
School Region                    object
School Code                      object
School Phone Number              object
School Address                   object
School City                      object
School State                     object
School Zip                       object
School Not Found                 object
School or Citywide Complaint     float64
Vehicle Type                     float64
Taxi Company Borough             float64
Taxi Pick Up Location            float64
Bridge Highway Name              object
Bridge Highway Direction         object
Road Ramp                        object
Bridge Highway Segment           object
Garage Lot Name                  float64
Ferry Direction                  object
Ferry Terminal Name              object
Latitude                         float64
Longitude                        float64
Location                         object
dtype: object
```

13) #computes and displays summary statistics for a Service_Request_csv datafra
me

```
Service_Request_csv.describe()
```

Customer Service Requests Analysis

	Unique Key	Incident Zip	X Coordinate (State Plane)	Y Coordinate (State Plane)	School or Citywide Complaint	Vehicle Type	Taxi Company Borough	Taxi Pick Up Location	Garage Lot Name	Latitude
count	3.006980e+05	298083.000000	2.971580e+05	297158.000000	0.0	0.0	0.0	0.0	0.0	297158.000000
mean	3.130054e+07	10848.888645	1.004854e+06	203754.534416	NaN	NaN	NaN	NaN	NaN	40.725885
std	5.738547e+05	583.182081	2.175338e+04	29880.183529	NaN	NaN	NaN	NaN	NaN	0.082012
min	3.027948e+07	83.000000	9.133570e+05	121219.000000	NaN	NaN	NaN	NaN	NaN	40.499135
25%	3.080118e+07	10310.000000	9.919752e+05	183343.000000	NaN	NaN	NaN	NaN	NaN	40.669796
50%	3.130436e+07	11208.000000	1.003158e+06	201110.500000	NaN	NaN	NaN	NaN	NaN	40.718661
75%	3.178446e+07	11238.000000	1.018372e+06	224125.250000	NaN	NaN	NaN	NaN	NaN	40.781840
max	3.231065e+07	11697.000000	1.067173e+06	271876.000000	NaN	NaN	NaN	NaN	NaN	40.912869

14) #create a custom dataframe with Complaint Type & City

```
customDataObject={'count':Service_Request_csv.groupby(['Complaint Type','City']).size()}
ComplaintTypeCity = pd.DataFrame(customDataObject).reset_index()
```

15) #show the custom complaint type city

```
ComplaintTypeCity
```

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

764 rows × 3 columns

Customer Service Requests Analysis

16) #just show first five rows from the dataset

```
ComplaintTypeCity.head()
```

	Complaint Type	City	count
0	Animal Abuse	ARVERNE	38
1	Animal Abuse	ASTORIA	125
2	Animal Abuse	BAYSIDE	37
3	Animal Abuse	BELLEROSE	7
4	Animal Abuse	BREEZY POINT	2

17) #get the individual column size for the column name Borough, Complaint Type & Descriptor

```
#get an array of column name
getColumnArray=["Borough","Complaint Type","Descriptor"]
Service_Request_csv.groupby(getColumnArray).size()
```

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

Python Dates A date in Python is not a data type of its own, but we can import a module named datetime to work with dates as date objects.

```
import datetime
```

18) #Create a dataframe with parsed date

```
#If True and parse_dates is enabled, pandas will attempt to infer the format of the datetime strings in the columns.
Service_Request_csv_withParsedDate=pd.read_csv(
   "311_Service_Requests_from_2010_to_Present.csv",
   parse_dates=["Created Date","Closed Date"])
```

Customer Service Requests Analysis

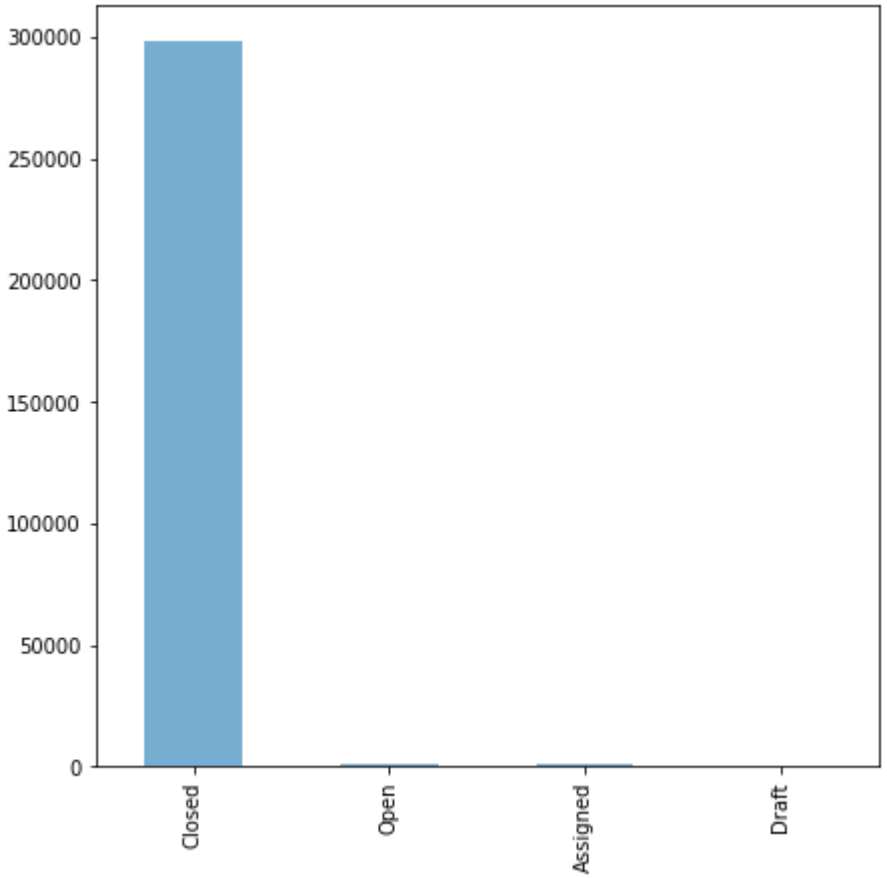
```
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2718: DtypeWarning: Columns (48,49) have mixed type
  interactivity=interactivity, compiler=compiler, result=result)
```

19) #calculate the Request Closing Time

```
Service_Request_csv_withParsedDate["Request_Closing_Time"] = Service_Request_csv_withParsedDate["Closed Date"] - Service_Request_csv_withParsedDate["Created Date"]
```

20) #Visualize the status of the ticket

```
Service_Request_csv_withParsedDate["Status"].value_counts().plot(kind='bar',alpha=0.6,figsize=(7,7))
plt.show()
```



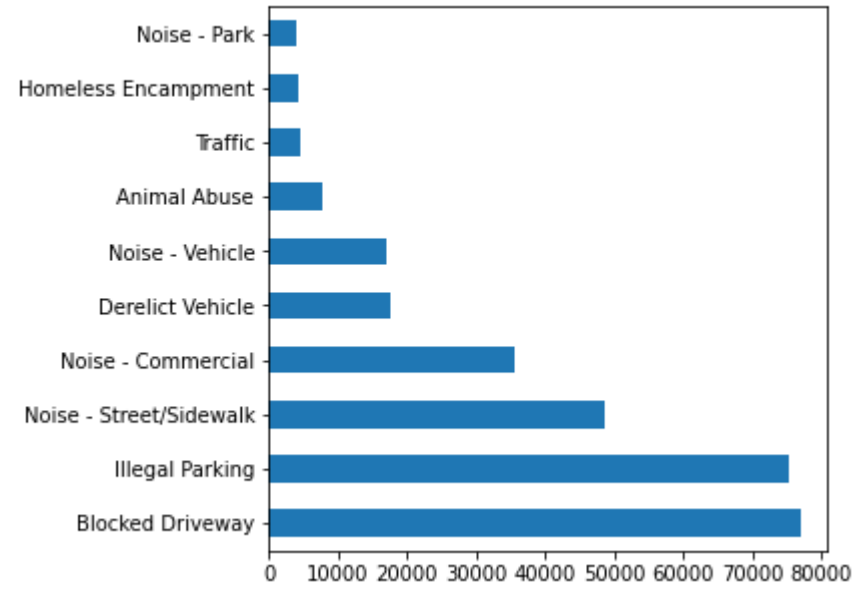
Matplotlib is a library in Python and it is numerical – mathematical extension for NumPy library.The figure module provides the top-level Artist, the Figure, which contains all the plot elements.This module is used to control the default spacing of the subplots and top level container for all plot elements.

21) #Complaint type Breakdown with bar plot to figure out majority of complaint t ypes and top 10 complaints

#Matplotlib is a library in Python and it is numerical – mathematical extension for NumPy lib rary.

```
Service_Request_csv["Complaint Type"].value_counts().head(10).plot(kind='barh',figsize=(5 ,5))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7055fc3e50>
```



22) #column wise groupby & size

```
Service_Request_csv.groupby(["Borough","Complaint Type","Descriptor"]).size()
```

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

23) #calculate the major complaint type

```
#subset: It's an array which limits the dropping process to passed rows/columns through list.
majorComplaints = Service_Request_csv.dropna(subset=["Complaint Type"])
majorComplaints = majorComplaints.groupby("Complaint Type")
#Pandas sort_values() function sorts a data frame in Ascending or Descending order of passe
d Column.
sortedComplaintType = majorComplaints.size().sort_values(ascending=False)
#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. ... level
sortedComplaintType = sortedComplaintType.to_frame('count').reset_index()
```

24) #how many complaints are there, that list i have find out

```
sortedComplaintType.head()
```

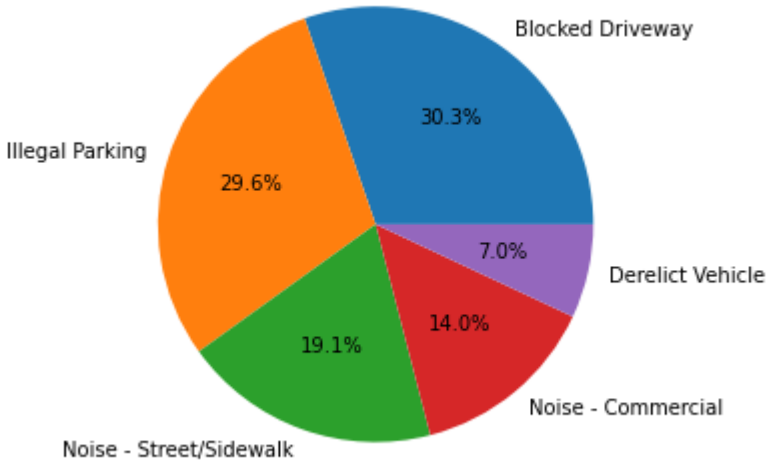
	Complaint Type	count
0	Blocked Driveway	77044
1	Illegal Parking	75361
2	Noise - Street/Sidewalk	48612
3	Noise - Commercial	35577
4	Derelict Vehicle	17718

25) #create a pie chart of this complaint type

```
sortedComplaintType = sortedComplaintType.head()
plt.figure(figsize=(5,5))

#We use autopct to display the percent value using Python string formatting.
#For example, autopct='%1.1f%%' means that for each pie wedge, the format string is '1.1f%'
.
plt.pie(sortedComplaintType['count'], labels=sortedComplaintType['Complaint Type'], autop
ct="%1.1f%%")
plt.show()
```

Customer Service Requests Analysis



26) #group dataset by complaint type to display plot against city

```
grouped_by_complaint_type = Service_Request_csv.groupby('Complaint Type')
```

27) #groupeddata with Blocked Driverway column type
#get how many groups are present on Blocked Driveway

```
grp_data = grouped_by_complaint_type.get_group("Blocked Driveway")
```

28) #get all column list

```
grp_data.columns.to_list()
```

```
['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',
 'Intersection Street 1',
 'Intersection Street 2',
 'Address Type',
 'City',
 'Landmark',
 'Facility Type',
 'Status',
 'Due Date',
 'Resolution Description',
 'Resolution Action Updated Date',
 'Community Board',
 'Borough',
```

```
 'X Coordinate (State Plane)',
 'Y Coordinate (State Plane)',
 'Park Facility Name',
 'Park Borough',
 'School Name',
 'School Number',
 'School Region',
 'School Code',
 'School Phone Number',
 'School Address',
 'School City',
 'School State',
 'School Zip',
 'School Not Found',
 'School or Citywide Complaint',
 'Vehicle Type',
 'Taxi Company Borough',
 'Taxi Pick Up Location',
 'Bridge Highway Name',
 'Bridge Highway Direction',
 'Road Ramp',
 'Bridge Highway Segment',
 'Garage Lot Name',
 'Ferry Direction',
 'Ferry Terminal Name',
 'Latitude',
 'Longitude',
       'Location']
```

29) #how many rows & columns are present for this Blocked Driveway dataset
grp_data.shape

```
(77044, 53)
```

30) #to get the null values for this dataset
Service_Request_csv.isnull().sum()

```
Unique Key                      0
Created Date                    0
Closed Date                  2164
Agency                          0
Agency Name                     0
Complaint Type                  0
Descriptor                   5914
Location Type                 131
Incident Zip                 2615
Incident Address            44410
Street Name                 44410
Cross Street 1              49279
Cross Street 2              49779
Intersection Street 1      256840
Intersection Street 2      257336
Address Type                 2815
City                         2614
Landmark                   300349
```

Customer Service Requests Analysis

```
Facility Type                         2171
Status                                   0
Due Date                                 3
Resolution Description                   0
Resolution Action Updated Date        2187
Community Board                          0
Borough                                  0
X Coordinate (State Plane)            3540
Y Coordinate (State Plane)            3540
Park Facility Name                       0
Park Borough                             0
School Name                              0
School Number                            0
School Region                            1
School Code                              1
School Phone Number                      0
School Address                           0
School City                              0
School State                             0
School Zip                               1
School Not Found                         0
School or Citywide Complaint        300698
Vehicle Type                        300698
Taxi Company Borough                300698
Taxi Pick Up Location               300698
Bridge Highway Name                 300455
Bridge Highway Direction            300455
Road Ramp                           300485
Bridge Highway Segment              300485
Garage Lot Name                     300698
Ferry Direction                     300697
Ferry Terminal Name                 300696
Latitude                              3540
Longitude                             3540
Location                              3540
dtype: int64
```

31) #drop blank values for City column

```
Service_Request_csv["City"].dropna(inplace=True)
```

32) #check shape after dropping null values

```
Service_Request_csv["City"].shape
```

```
(300698,)
```

33) #count of null values in grouped city column data

```
grp_data["City"].isnull().sum()
```

```
283
```

Customer Service Requests Analysis

34) #fix those nan values with "Unknown city" value instead
#The fillna() function is used to fill NA/NaN values using the specified method.

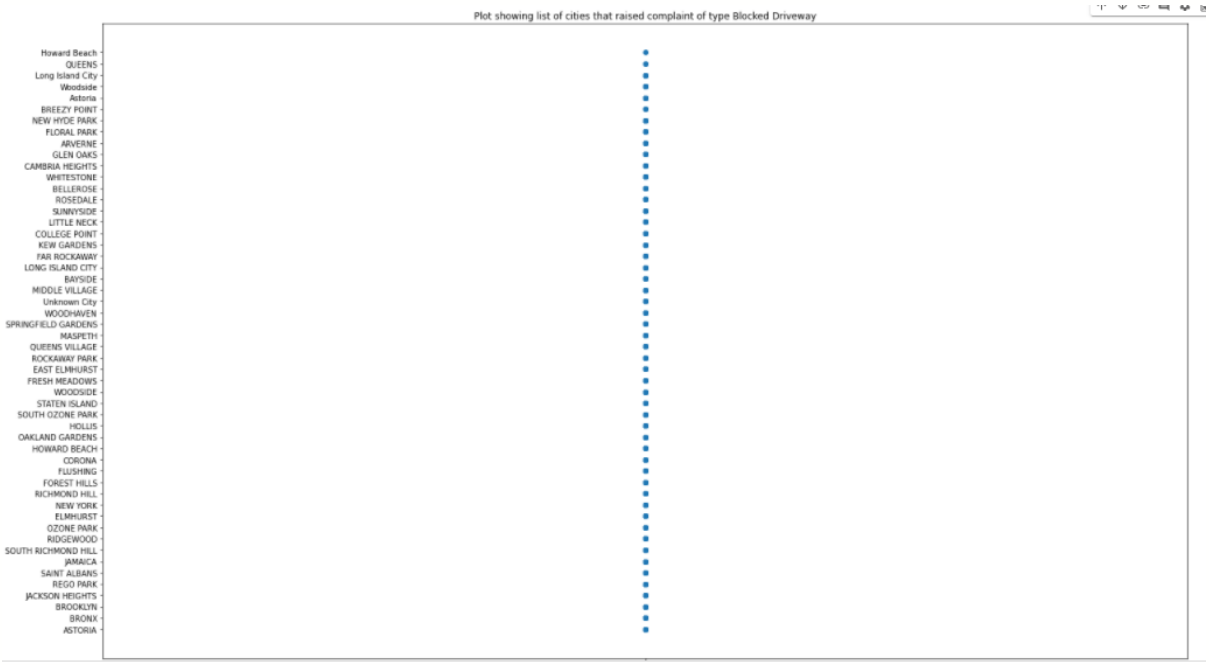
```
grp_data["City"].fillna("Unknown City", inplace=True)
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/series.py:4536: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vie
  downcast=downcast,
```

35) #Scatter plot displaying all the cities that raised complaint of type 'Blocked Dr iveway'

```
plt.figure(figsize=(25,15))
plt.scatter(grp_data["Complaint Type"], grp_data["City"])
plt.title("Plot showing list of cities that raised complaint of type Blocked Driveway")
plt.show()
```



36) #fix Location type those NAN with "unknown Location" value instead

```
Service_Request_csv["Location Type"].fillna("Unknown Loc",inplace=True)
```

37) #how many values are present for the column name Location Type

```
Service_Request_csv["Location Type"].values
```

```
array(['Street/Sidewalk', 'Street/Sidewalk', 'Street/Sidewalk', ...,
       'Club/Bar/Restaurant', 'Club/Bar/Restaurant',
'Store/Commercial'],
      dtype=object)
```

38) #Find top 10 major complaint types and their counts

```
grouped_by_complaint_type["Complaint Type"].value_counts().nlargest(10)
```

Customer Service Requests Analysis

```
Complaint Type           Complaint Type
Blocked Driveway         Blocked Driveway          77044
Illegal Parking          Illegal Parking           75361
Noise - Street/Sidewalk  Noise - Street/Sidewalk   48612
Noise - Commercial       Noise - Commercial        35577
Derelict Vehicle         Derelict Vehicle          17718
Noise - Vehicle          Noise - Vehicle           17083
Animal Abuse             Animal Abuse               7778
Traffic                  Traffic                    4498
Homeless Encampment      Homeless Encampment        4416
Noise - Park             Noise - Park               4042
Name: Complaint Type, dtype: int64
```

39) #fix Location type those NAN with "unknown Location" value instead

```
Service_Request_csv["Location Type"].fillna("Unknown Loc", inplace=True)
```

```
Service_Request_csv['Location Type'].values
```

```
array(['Street/Sidewalk', 'Street/Sidewalk', 'Street/Sidewalk', ...,
       'Club/Bar/Restaurant', 'Club/Bar/Restaurant', 'Store/Commercial'],
      dtype=object)
```

40) #count of null values in grouped location type column data

```
grp_data['Location Type'].isnull().sum()
```

37

Customer Service Requests Analysis