Applied Data Science with Python

Certification Project

Author: Charusmita Shah

Customer Service Request Analysis

Objectives:

- To assess the data and prepare a fresh dataset for training and prediction
- To plot a bar graph to identify the relationship between two variables
- To visualize the major types of complaints in each city

Problem Statement:
You've been asked to analyze data on service request (311) calls from New York City. You've also been asked to utilize data wrangling techniques to understand the patterns in the data and visualize the major types of complaints. Note: Download the 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.

Domain: General

Analysis to be done: Analyze data of service request calls from New York City and visualize major types of complaints.

Content: Dataset: : 311-service-requests-nyc.csv
Fields in the data:

Unique Key - The unique identification number
Created Date - The date when the request was created
Closed Date - The date when the request was closed
Agency - The agency that handled the case
Agency Name - The full name of the agency that handled the case
Complaint Type - The type of complaint received
Descriptor - The description of the complaint
Location Type - The type of location where the incident occurred
Incident Zip - The zip code of the location
Incident Address - The location at which the incident occurred
Street Name - The name of the street
Cross Street 1 - The cross of the street 1
Cross Street 2 - The cross of street 2
Intersection Street 1 - The first point of intersection of both streets
Intersection Street 2 The second point of intersection of both streets
Address Type - The type of the address

City - The city where the incident occurred
Landmark - The landmark near the incident that occurred
Facility Type - The type of the facility Status The status of the complaint
Status - The status of the complaint
Due Date - The due date of the complaint
Resolution Description - The resolution provided by the police department
Resolution Action Updated Date - The date at which the resolution was provided
Community Board - The location of the community board
Borough - The town, or area inside a large town, that has some form of local govt.
X Coordinate (State Plane) - The X coordinate of the plane
Y Coordinate (State Plane) - The Y coordinate of the plane
Park Facility Name - The name of the park facility
Park Borough - The park town, or area inside a large town, that has some form of local government
School Name - The name of the school (optional)
School Number - Number of the school (optional)
School Region - Region of the school (optional)
School Code - Code of the school (optional)
School Phone Number - Contact information of the school (optional)
School Address - Address of the school (optional)
School City - City at which the school is located (optional)
School State - State in which the school is located (optional)
School Zip - Zip code of the school (optional)
School Not Found - Valid if the school is not found (optional)
School or Citywide Complaint - Contains the complaint of the school (optional)
Vehicle Type - Type of vehicle used (optional)
Taxi Company Borough - Information on the taxi company (optional)
Taxi Pick Up Location - Pick up location of the taxi (optional)
Bridge Highway Name - Name of the highway bridge (optional)
Bridge Highway Direction - Direction of the highway bridge (optional)
Road Ramp - Information on the road ramp (optional)
Bridge Highway Segment - Segment of the bridge (optional)
Garage Lot Name - Name of the garage (optional)
Ferry Direction - Ferry direction information (optional)
Ferry Terminal Name - Name of the ferry terminal (optional)
Latitude - Latitude value
Longitude - Longitude value
Location - Location information

Tasks to Perform (detailed *with Solutions*):

1. Understand the dataset:

1.1 Import the dataset

Ans . We read the csv file with the read_csv function from pandas from the specific folder.

```
df = pd.read_csv("Customer_Service_Requests_Analysis_Dataset/
311_Service_Requests_from_2010_to_Present.csv")
```

1.2 Visualize the dataset

Ans. We can visualize what the data looks like with head() which shows the first 5 rows with total number of columns output in the bottom to give us an idea of how wide the data frame is.

```
df.head()
```

```
df.head()
```

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	Incident Zip	Incident Address	...	Bridge Highway Name
0	32310363	12/31/2015 11:59:45 PM	01/01/2016 12:55:15 AM	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Music/Party	Street/Sidewalk	10034.0	71 VERMILYEA AVENUE	...	NaN
1	32309934	12/31/2015 11:59:44 PM	01/01/2016 01:26:57 AM	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewalk	11105.0	27-07 23 AVENUE	...	NaN
2	32309159	12/31/2015 11:59:29 PM	01/01/2016 04:51:03 AM	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewalk	10458.0	2897 VALENTINE AVENUE	...	NaN
3	32305098	12/31/2015 11:57:46 PM	01/01/2016 07:43:13 AM	NYPD	New York City Police Department	Illegal Parking	Commercial Overnight Parking	Street/Sidewalk	10461.0	2940 BAISLEY AVENUE	...	NaN
4	32306529	12/31/2015 11:56:58 PM	01/01/2016 03:24:42 AM	NYPD	New York City Police Department	Illegal Parking	Blocked Sidewalk	Street/Sidewalk	11373.0	87-14 57 ROAD	...	NaN

5 rows × 53 columns

1.3 Print the columns of the DataFrame

Ans. `df.columns`

```
[7]: df.columns
```

```
[7]: Index(['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'],
      dtype='object')
```

1.4 Identify the shape of the dataset

Ans. `df.shape`

```
[8]: df.shape
```

```
[8]: (364558, 53)
```

1.5 Identify the variables with null values

Ans. `df.isnull().sum()`

```
[9]: df.isnull().sum()
```

```
[9]: 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
```

2. Perform basic data exploratory analysis:

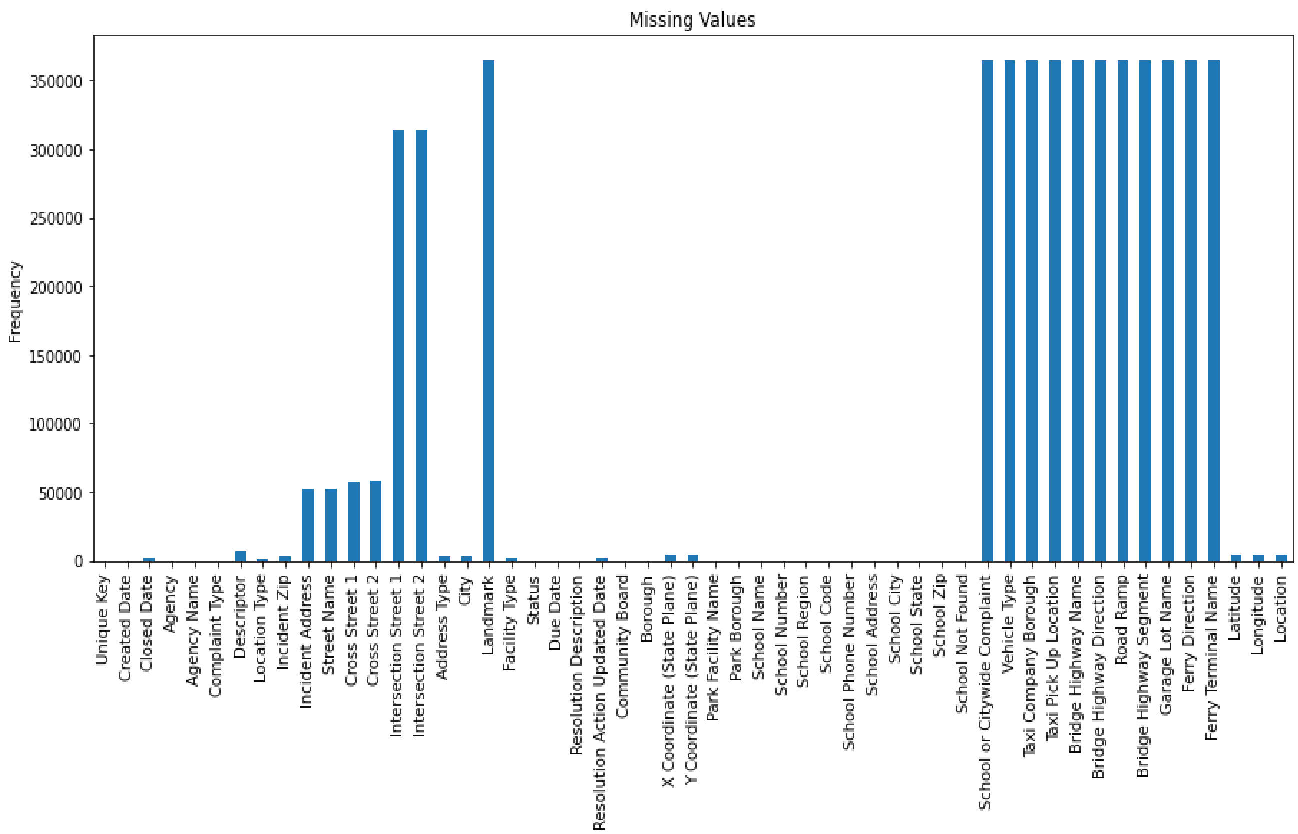
2.1 Draw a frequency plot to show the number of null values in each column of the DataFrame

Ans .

```
df.isnull().sum().plot(kind=”bar”, figsize=(15,6),
title=”Missing values”, ylabel=”Frequency”)
```

```
[9]: df.isnull().sum().plot(kind="bar", figsize=(15,6), title="Missing Values", ylabel="Frequency")
```

```
[9]: <AxesSubplot:title={'center':'Missing Values'}, ylabel='Frequency'>
```



2.2 Missing value treatment

2.2.1 Remove the records whose Closed Date values are null

Ans. We check by first dropping the null records and then checking the sum of null Closed Date records which should be zero, hence confirming our code to be correct .

```
df.dropna(subset=[“Closed Date”], inplace=True)
```

```
[ ]: # 2.2 Missing value treatment
     # 2.2.1 Remove the records whose Closed Datevalues are null
```

```
[11]: df.dropna(subset=["Closed Date"], inplace=True)
```

```
[12]: df.isnull().sum()["Closed Date"]
```

```
[12]: 0
```

Before moving ahead , let us drop columns which are not required for analysis which will lead to cleaner data and quicker and more efficient processing.

```
df =  df[['Unique Key', 'Created Date', 'Closed Date', 'Agency', 'Agency Name','Complaint Type', 'Descriptor', 'Location Type', 'Incident Zip', 'City', 'Status', 'Due Date', 'Resolution Description','Borough', 'Latitude', 'Longitude', 'Location']]
```

```
# Before moving ahead , let us drop columns which are not required for analysis which
# will lead to cleaner data and quicker and more efficient processing.
df =  df[['Unique Key', 'Created Date', 'Closed Date', 'Agency', 'Agency Name','Complaint Type', 'Descriptor', 'Location Type',
          'Incident Zip', 'City', 'Status', 'Due Date', 'Resolution Description','Borough', 'Latitude', 'Longitude', 'Location']]
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 362177 entries, 0 to 364557
Data columns (total 17 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   Unique Key              362177 non-null  int64
 1   Created Date            362177 non-null  object
 2   Closed Date             362177 non-null  object
 3   Agency                  362177 non-null  object
 4   Agency Name             362177 non-null  object
 5   Complaint Type          362177 non-null  object
 6   Descriptor              355681 non-null  object
 7   Location Type           362047 non-null  object
 8   Incident Zip            361502 non-null  float64
 9   City                    361503 non-null  object
 10  Status                  362177 non-null  object
 11  Due Date                362176 non-null  object
 12  Resolution Description  362177 non-null  object
 13  Borough                 362177 non-null  object
 14  Latitude                360470 non-null  float64
 15  Longitude               360470 non-null  float64
 16  Location                360470 non-null  object
dtypes: float64(3), int64(1), object(13)
memory usage: 49.7+ MB
```

2.3 Analyze the date column,and remove entries that have an incorrect timeline

2.3.1 Calculate the time elapsed in closed and creation date

Ans .

```
df['Created Date'] = pd.to_datetime(df['Created Date'])
df['Closed Date'] = pd.to_datetime(df['Closed Date'])
df['Elapsed Time'] = df['Closed Date'] - df['Created Date']
```

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

```
[20]: df['Elapsed Time'] = df['Closed Date'] - df['Created Date']
```

```
[21]: df.loc[0:10, ["Unique Key", "Created Date", "Closed Date", "Elapsed Time"]]
```

[21]:

	Unique Key	Created Date	Closed Date	Elapsed Time
0	32310363	2015-12-31 23:59:45	2016-01-01 00:55:15	0 days 00:55:30
1	32309934	2015-12-31 23:59:44	2016-01-01 01:26:57	0 days 01:27:13
2	32309159	2015-12-31 23:59:29	2016-01-01 04:51:03	0 days 04:51:34
3	32305098	2015-12-31 23:57:46	2016-01-01 07:43:13	0 days 07:45:27
4	32306529	2015-12-31 23:56:58	2016-01-01 03:24:42	0 days 03:27:44
5	32306554	2015-12-31 23:56:30	2016-01-01 01:50:11	0 days 01:53:41
6	32306559	2015-12-31 23:55:32	2016-01-01 01:53:54	0 days 01:58:22
7	32307009	2015-12-31 23:54:05	2016-01-01 01:42:54	0 days 01:48:49
8	32308581	2015-12-31 23:53:58	2016-01-01 08:27:32	0 days 08:33:34
9	32308391	2015-12-31 23:53:58	2016-01-01 01:17:40	0 days 01:23:42
10	32305071	2015-12-31 23:52:58	2016-01-01 07:41:38	0 days 07:48:40

2.3.2 Convert the calculated date to seconds to get a better representation

Ans.

```
df['Elapsed Time'] = df['Elapsed Time']
.dt.total_seconds().astype(int)
```

```
[25]: df['Elapsed Time'] = df['Elapsed Time'].dt.total_seconds().astype(int)
```

```
[26]: df.loc[0:10, ["Unique Key", "Created Date", "Closed Date", "Elapsed Time"]]
```

[26]:

	Unique Key	Created Date	Closed Date	Elapsed Time
0	32310363	2015-12-31 23:59:45	2016-01-01 00:55:15	3330
1	32309934	2015-12-31 23:59:44	2016-01-01 01:26:57	5233
2	32309159	2015-12-31 23:59:29	2016-01-01 04:51:03	17494
3	32305098	2015-12-31 23:57:46	2016-01-01 07:43:13	27927
4	32306529	2015-12-31 23:56:58	2016-01-01 03:24:42	12464
5	32306554	2015-12-31 23:56:30	2016-01-01 01:50:11	6821
6	32306559	2015-12-31 23:55:32	2016-01-01 01:53:54	7102
7	32307009	2015-12-31 23:54:05	2016-01-01 01:42:54	6529
8	32308581	2015-12-31 23:53:58	2016-01-01 08:27:32	30814
9	32308391	2015-12-31 23:53:58	2016-01-01 01:17:40	5022
10	32305071	2015-12-31 23:52:58	2016-01-01 07:41:38	28120

2.3.3 View the descriptive statistics for the newly created column

Ans. Here we can see the Elapsed time column info using `df.info()`

```
[28]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 362177 entries, 0 to 364557
Data columns (total 18 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   Unique Key              362177 non-null  int64
 1   Created Date            362177 non-null  datetime64[ns]
 2   Closed Date             362177 non-null  datetime64[ns]
 3   Agency                  362177 non-null  object
 4   Agency Name             362177 non-null  object
 5   Complaint Type          362177 non-null  object
 6   Descriptor              355681 non-null  object
 7   Location Type           362047 non-null  object
 8   Incident Zip            361502 non-null  float64
 9   City                    361503 non-null  object
 10  Status                  362177 non-null  object
 11  Due Date                362176 non-null  object
 12  Resolution Description  362177 non-null  object
 13  Borough                 362177 non-null  object
 14  Latitude                360470 non-null  float64
 15  Longitude               360470 non-null  float64
 16  Location                360470 non-null  object
 17  Elapsed Time            362177 non-null  int64
dtypes: datetime64[ns](2), float64(3), int64(2), object(11)
memory usage: 62.5+ MB
```

2.3.4 Check the number of null values in the Complaint_Type and City Columns

Ans. df.isnull().sum()[["City", "Complaint Type"]]

```
[ ]: # 2.3.4 Check the number of null values in the Complaint_Typeand Citycolumns

[12]: df.isnull().sum()[["City", "Complaint Type"]]

[12]: City              674
      Complaint Type      0
      dtype: int64

[ ]:
```

2.3.5 Impute the NA value with Unknown City

Ans. df["City"].fillna(value="Unknown City", inplace=True)

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

df.isnull().sum()["City"]

0

df.loc[df['City'] == "Unknown City"]
```

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	Incident Zip	City	Status
33	32306700	2015-12-31 23:18:10	2016-01-02 01:04:03	NYPD	New York City Police Department	Illegal Parking	Double Parked Blocking Traffic	Street/Sidewalk	NaN	Unknown City	Closed
283	32309451	2015-12-31 17:40:16	2016-01-01 10:59:11	NYPD	New York City Police Department	Illegal Parking	Blocked Hydrant	Street/Sidewalk	NaN	Unknown City	Closed

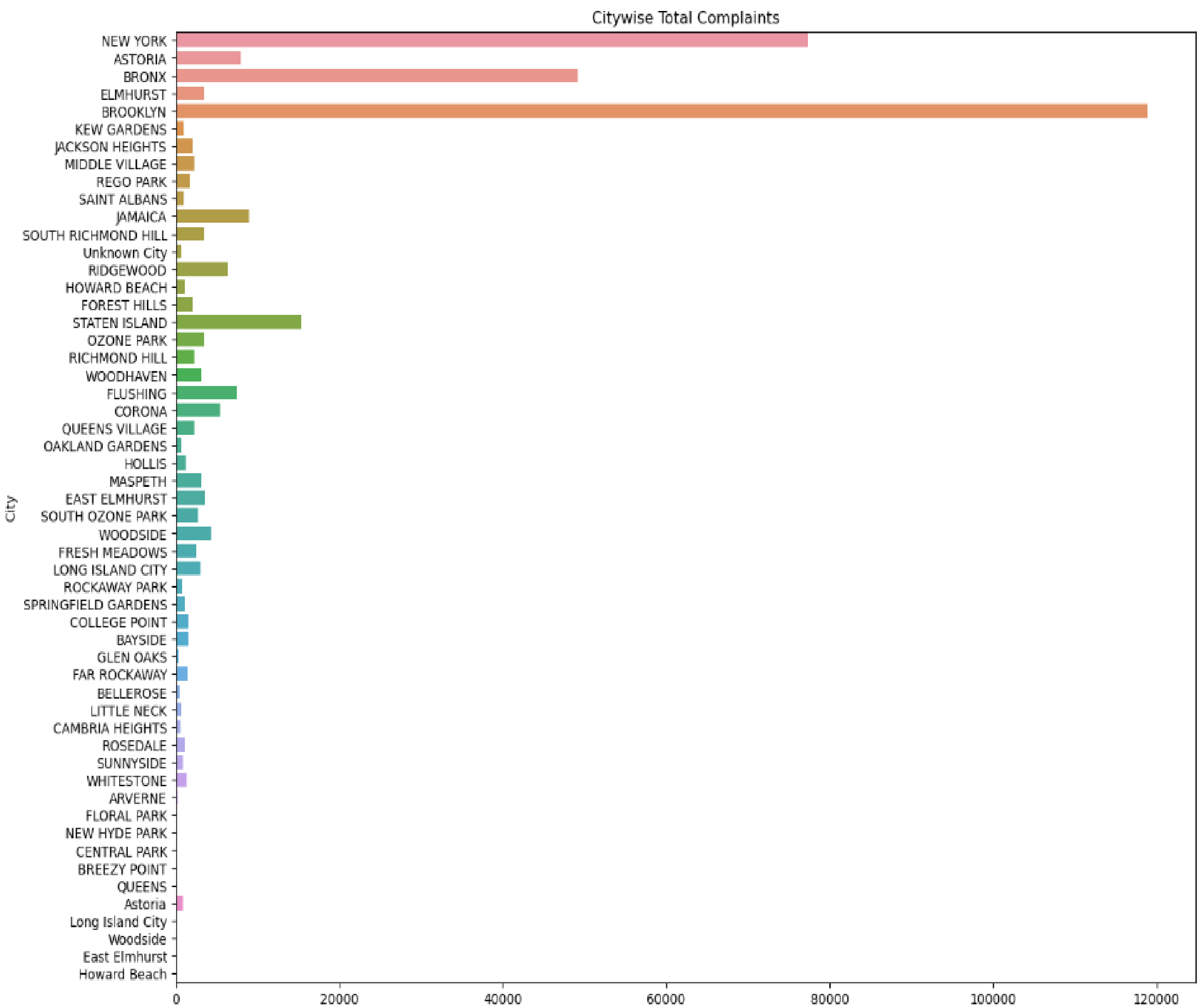
PTO

2.3.6 Draw a frequency plot for the complaints in each city

Ans.

```
plt.figure(figsize=(15,12))
sns.countplot(data=df, y='City')
plt.title("Citywise Total Complaints")
plt.show()
```

```
plt.figure(figsize=(15,12))
sns.countplot(data=df, y='City')
plt.title("Citywise Total Complaints")
plt.show()
```



2.3.7 Create a scatter and hexbin plot of the concentration of complaints across Brooklyn

Ans. First make a dataframe with brooklyn city or borough and then perform next operations.

```
df_Brooklyn = df[df['Borough'] == 'BROOKLYN']
```

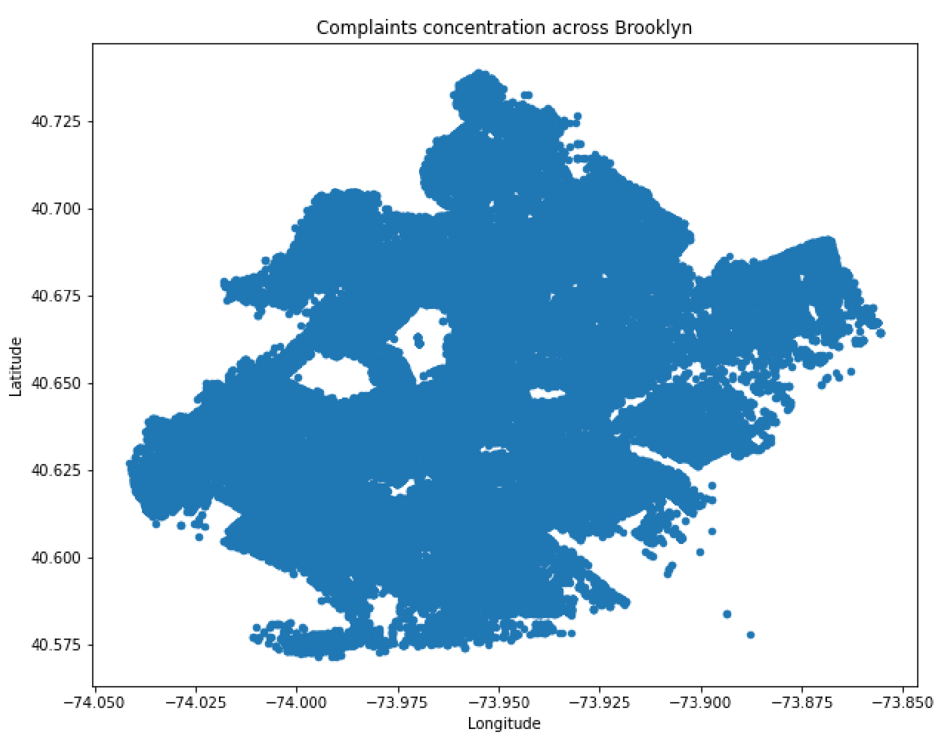
```
df_Brooklyn = df[df['Borough'] == 'BROOKLYN']
```

```
df_Brooklyn.head()
```

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	Incident Zip	City	Status	Due Date	Resolution Description	Borough
5	32306554	2015-12-31 23:56:30	2016-01-01 01:50:11	NYPD	New York City Police Department	Illegal Parking	Posted Parking Sign Violation	Street/Sidewalk	11215.0	BROOKLYN	Closed	01/01/2016 07:56:30 AM	The Police Department responded and upon arriv...	BROOKLYN
9	32308391	2015-12-31 23:53:58	2016-01-01 01:17:40	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewalk	11219.0	BROOKLYN	Closed	01/01/2016 07:53:58 AM	The Police Department responded and upon arriv...	BROOKLYN

```
df_Brooklyn[['Latitude', 'Longitude']].plot(kind='scatter',
x='Longitude', y='Latitude', title='Complaints concentration
across Brooklyn', figsize=(10,8))
```

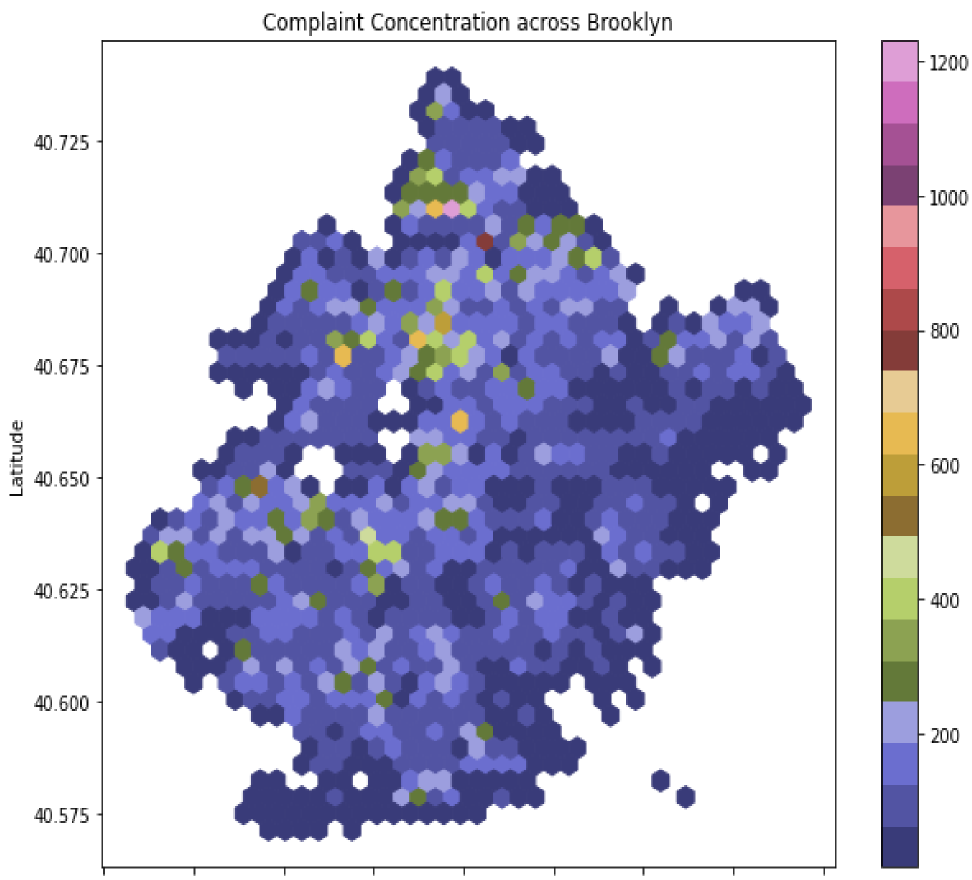
```
<AxesSubplot:title={'center':'Complaints concentration across Brooklyn'}, xlabel='Longitude', ylabel='Latitude'>
```



Now for hexbin plot:

```
df_Brooklyn[['Latitude','Longitude']].plot(kind='hexbin',x='Long
itude',y='Latitude', title="Complaint Concentration across
Brooklyn", figsize=(10,8), gridsize=40,
colormap='tab20b',mincnt=1)
```

```
<AxesSubplot:title={'center':'Complaint Concentration across Brooklyn'}, xlabel='Longitude', ylabel='Latitude'>
```



So as we see in the plot , more purple has less complaints, and pinker have more.

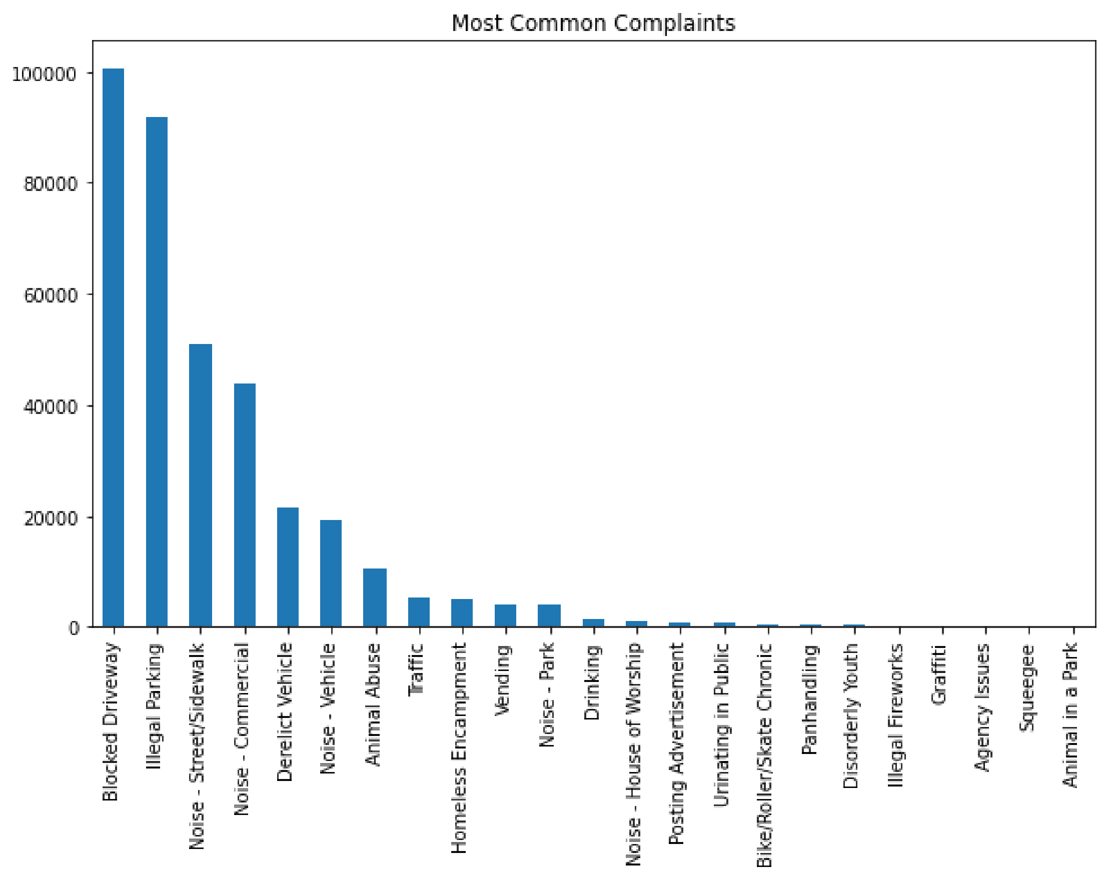
3. Find major types of complaints:

3.1 Plot a bar graph to show the types of complaints

Ans.

```
df['Complaint
Type'].value_counts().plot(kind="bar",figsize=(10,6),title="Most
Common Complaints")
```

```
: <AxesSubplot:title={'center':'Most Common Complaints'}>
```



3.2 Check the frequency of various types of complaints for New York City

Ans. First we get the data frame with only New York City data

```
df_NewYork = df[df['City'] == 'NEW YORK']
```

And then we calculate the frequency of various complaint types.

```
df_NewYork['Complaint
Type'].value_counts().sort_values(ascending=False)
```

```
df_NewYork['Complaint Type'].value_counts().sort_values(ascending=False)
```

```
Noise - Street/Sidewalk      22245
Noise - Commercial           18686
Illegal Parking              14549
Noise - Vehicle               6294
Homeless Encampment           3060
Blocked Driveway              2705
Vending                       2638
Animal Abuse                  1941
Traffic                       1769
Noise - Park                  1243
Derelict Vehicle               695
Drinking                       321
Urinating in Public            264
Bike/Roller/Skate Chronic      254
Noise - House of Worship       222
Panhandling                    206
Disorderly Youth                81
Posting Advertisement           49
Illegal Fireworks               38
Graffiti                        25
Squeegee                         4
Name: Complaint Type, dtype: int64
```

3.3 Find the top 10 complaint types

Ans.

```
df['Complaint
Type'].value_counts().sort_values(ascending=False)[:10]
```

```
Blocked Driveway           100624
Illegal Parking             91716
Noise - Street/Sidewalk     51139
Noise - Commercial          43751
Derelict Vehicle            21518
Noise - Vehicle             19301
Animal Abuse                10530
Traffic                      5196
Homeless Encampment          4879
Vending                      4185
Name: Complaint Type, dtype: int64
```

3.4 Display the various types of complaints in each city

Ans. `df.groupby(['City','Complaint Type']).size()`

```
df.groupby(['City','Complaint Type']).size()
```

```
City      Complaint Type
ARVERNE   Animal Abuse                46
          Blocked Driveway            50
          Derelict Vehicle            32
          Disorderly Youth             2
          Drinking                     1
                                    ...
Woodside  Blocked Driveway            27
          Derelict Vehicle             8
          Illegal Parking            124
          Noise - Commercial           2
          Noise - Street/Sidewalk      5
Length: 792, dtype: int64
```

3.5 Create a DataFrame, df_new, which contains cities as columns and complaint types in rows

Ans.

```
df_new  = df.groupby(['City','Complaint Type']).size().unstack()
df_new
```

```
df_new  = df.groupby(['City','Complaint Type']).size().unstack()
df_new
```

Complaint Type	Agency Issues	Animal Abuse	Animal in a Park	Bike/Roller/Skate Chronic	Blocked Driveway	Derelict Vehicle	Disorderly Youth	Drinking	Graffiti	Homeless Encampment	...
City											
ARVERNE	NaN	46.0	NaN	NaN	50.0	32.0	2.0	1.0	1.0	4.0	...
ASTORIA	NaN	170.0	NaN	16.0	3436.0	426.0	5.0	43.0	4.0	32.0	...
Astoria	NaN	NaN	NaN	NaN	159.0	14.0	NaN	NaN	NaN	NaN	...
BAYSIDE	NaN	53.0	NaN	NaN	514.0	231.0	2.0	1.0	3.0	2.0	...
BELLEROSE	NaN	15.0	NaN	1.0	138.0	120.0	2.0	1.0	NaN	1.0	...
BREEZY POINT	NaN	2.0	NaN	NaN	3.0	3.0	NaN	1.0	NaN	NaN	...
BRONX	NaN	1971.0	NaN	22.0	17062.0	2402.0	66.0	206.0	15.0	275.0	...
BROOKLYN	NaN	3191.0	NaN	124.0	36445.0	6257.0	79.0	291.0	60.0	948.0	...
CAMBRIA HEIGHTS	NaN	15.0	NaN	NaN	177.0	148.0	NaN	NaN	NaN	6.0	...

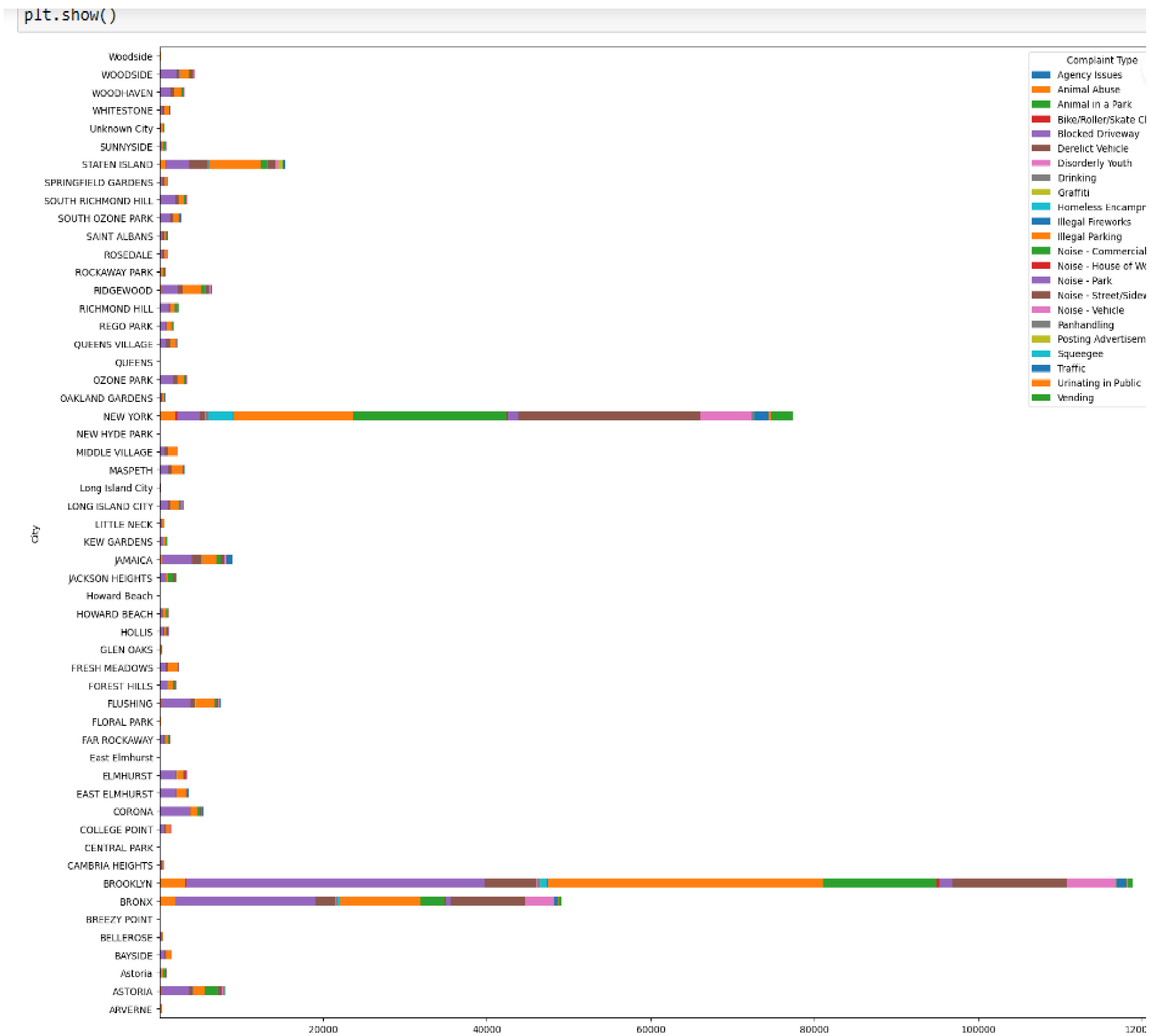
4. Visualize the major types of complaints in each city

4.1Draw another chart that shows the types of complaints in each city in a single chart, where different colors show the different types of complaints

Ans.

```
df_new.plot(kind="barh", stacked=True, figsize=(20,18))
plt.show()
```

Graph for all cities below *(tough to see)*



4.2 Sort the complaint types based on the average Request_Closing_Time grouping them for different locations

Ans.

#Getting the request_closing_time in minutes

```
df['Request_Closing_Time']= (df['Closed
Date'].values-df['Created Date'].values)/np.timedelta64(1,'m')
```

#And then computing

```
data_request_closing = df.groupby(['City','Complaint
Type'])['Request_Closing_Time'].mean()
data_request_closing.unstack().fillna(0).head()
```

```
[71]: data_request_closing = df.groupby(['City','Complaint Type'])['Request_Closing_Time'].mean()
      data_request_closing.unstack().fillna(0).head()
```

[71]:

Complaint Type City	Agency Issues	Animal Abuse	Animal in a Park	Bike/Roller/Skate Chronic	Blocked Driveway	Derelict Vehicle	Disorderly Youth	Drinking	Graffiti	Homeless Encampment
ARVERNE	0.0	139.986594	0.0	0.000000	138.647333	189.900000	215.475000	14.316667	91.800000	109.020833
ASTORIA	0.0	286.781373	0.0	111.979167	275.265575	540.776643	161.896667	258.637984	845.704167	295.071875
Astoria	0.0	0.000000	0.0	0.000000	274.881027	417.207143	0.000000	0.000000	0.000000	0.000000
BAYSIDE	0.0	186.725786	0.0	0.000000	155.765759	204.360462	155.641667	114.133333	273.022222	172.375000
BELLEROSE	0.0	532.820000	0.0	293.600000	501.905435	920.781389	110.525000	235.083333	0.000000	2348.833333

5 rows × 23 columns

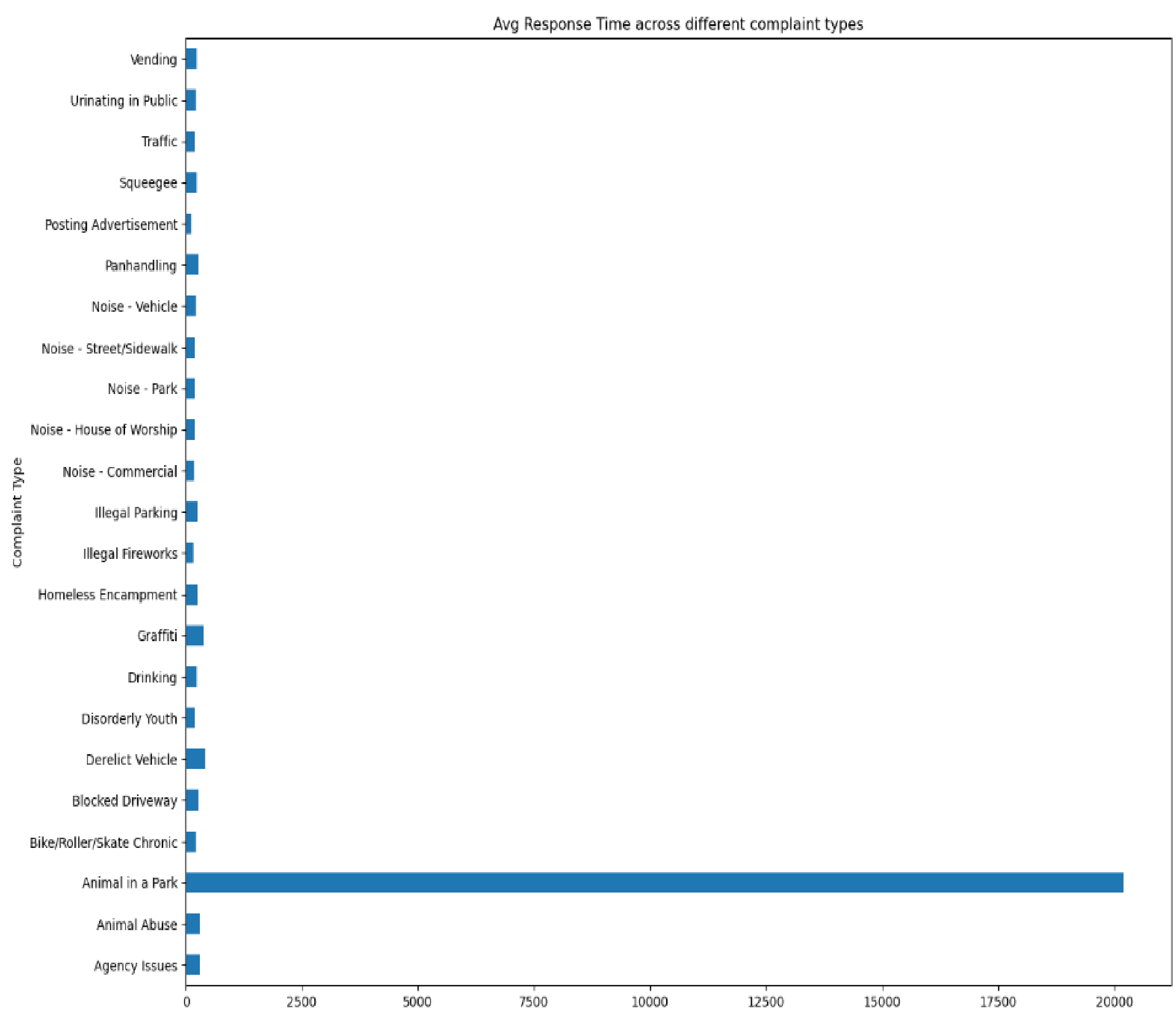
5. See whether the average response time across different complaint types is similar (overall)

5.1 Visualize the average ofRequest_Closing_Time

Ans.

```
plt.figure(figsize=(15,12))
plt.title("Avg Response Time across different complaint types")
df.groupby('Complaint
Type')['Request_Closing_Time'].mean().plot(kind="barh")
```

Out[122]: <AxesSubplot:title={'center':'Avg Response Time across different complaint types'}, ylabel='Complaint Type'>



6. Identify the significant variables by performing statistical analysis using p-values

Ans.

Chi Square Test

Null Hypothesis: There is no significant relation between complaint type and city.

Alternate Hypothesis: There is some significant relation between complaint type and city.

```
location_complaint_type = pd.crosstab(df['Complaint Type'],
df['City'])
cscore,pval,df,et = st.chi2_contingency(location_complaint_type)
print("score : {:f} , pvalue : {:f}".format(cscore,pval))
```

```
[98]: cscore,pval,df,et = st.chi2_contingency(location_complaint_type)
      print("score : {:.2f} , pvalue : {:.2f}".format(cscore,pval))

      score : 145971.80 , pvalue : 0.00

•[84]: # As pvalue is less than 0.05, we reject our Null Hypothesis
       # There is significant relation between complaint type and city

[ ]:
```

8.Present your observations.

1) Time taken for solving different complaint types are different.
2) Complaint Types are dependent on the City.
3) Maximum number of complaints from cities are in order as follows : ['BROOKLYN', 'NEW YORK', 'BRONX', 'STATEN ISLAND', 'JAMAICA']
4) Complaints tend to get closed between 150 to 300 hours.