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

#Import and suppress warnings
import warnings
warnings.filterwarnings('ignore')


#Making a list of missing value types
missing_values = ["n/a", "na", "--", "...", "NaN"]

#Read the data set by indexing City as 1st Coloumn
df = pd.read_csv("311_Service_Requests_from_2010_to_Present.csv", na_values = missing_values)
df.head(20)

#display(df) - display entire df
```

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	Incident Zip	Incide Addre
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	VERMILYE AVENU
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 AVENU
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	28 VALENTIN AVENU
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	29 BAISLE AVENU
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 ROA
5	32306554	12/31/2015 11:56:30 PM	01/01/2016 01:50:11 AM	NYPD	New York City Police Department	Illegal Parking	Posted Parking Sign Violation	Street/Sidewalk	11215.0	260 STREE
6	32306559	12/31/2015 11:55:32 PM	01/01/2016 01:53:54 AM	NYPD	New York City Police Department	Illegal Parking	Blocked Hydrant	Street/Sidewalk	10032.0	524 WES 169 STREE
7	32307009	12/31/2015 11:54:05 PM	01/01/2016 01:42:54 AM	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewalk	10457.0	501 EAS 171 STREE
8	32308581	12/31/2015 11:53:58 PM	01/01/2016 08:27:32 AM	NYPD	New York City Police Department	Illegal Parking	Posted Parking Sign Violation	Street/Sidewalk	11415.0	83- LEFFER BOULEVAR
9	32308391	12/31/2015 11:53:58 PM	01/01/2016 01:17:40 AM	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewalk	11219.0	1408 STREE
10	32305071	12/31/2015 11:52:58 PM	01/01/2016 07:41:38 AM	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewalk	11372.0	34-06 STREE

12/31/2015 01/01/2016 New York Blocked 17

```
#To find any duplicated or variables/columns which has missing values.
vars_with_na = [var for var in df.columns if df[var].isnull().sum() > 0]
print (vars_with_na)
```

```
['Closed Date', 'Descriptor', 'Location Type', 'Incident Zip', 'Incident Address', 'Street Name', 'Cross Street 1', 'Cross Street 2', '
```

12 32305074 11:47:58 08:18:47 NYPD City Police Illegal Parking Parking Street/Sidewalk 11208.0 38 C

```
#get df size
print('\nDataFrame Size : ', df.size)

#get df shape
print('\nDataFrame Shape : ', df.shape)

#get df dimension
print('\nDataFrame Dimension : ', df.ndim)

print('\nDataFrame rows : ', df.shape[0])
print('\nDataFrame columns : ', df.shape[1])
```

```
DataFrame Size :  19321574

DataFrame Shape :  (364558, 53)

DataFrame Dimension :  2

DataFrame rows :  364558

DataFrame columns :  53
```

PM AM Department

```
#To remove columns with missing value(s):
df.dropna(axis="columns")   # or axis=1

#Show which df entries are NA.
df.isna()

#Look at your missing data
missing_data = df.isnull()
missing_data.head()

# Checking the missing values
df.isnull().sum()
```

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

```
Garage Lot Name                 364558
Ferry Direction                 364557
Ferry Terminal Name             364556
Latitude                          4030
Longitude                         4030
Location                          4030
dtype: int64
```

```
#Using a for loop in Python to figure out the number of missing values in each column
for column in missing_data.columns.values.tolist():
    print(column)
    print(missing_data[column].value_counts())
    print("")
```

```
Unique Key
False    364558
Name: Unique Key, dtype: int64

Created Date
False    364558
Name: Created Date, dtype: int64

Closed Date
False    362177
True       2381
Name: Closed Date, dtype: int64

Agency
False    364558
Name: Agency, dtype: int64

Agency Name
False    364558
Name: Agency Name, dtype: int64

Complaint Type
False    364558
Name: Complaint Type, dtype: int64

Descriptor
False    358057
True       6501
Name: Descriptor, dtype: int64

Location Type
False    364425
```

```
True        133
Name: Location Type, dtype: int64

Incident Zip
False    361560
True       2998
Name: Incident Zip, dtype: int64

Incident Address
False    312859
True      51699
Name: Incident Address, dtype: int64

Street Name
False    312859
True      51699
Name: Street Name, dtype: int64

Cross Street 1
False    307370
True      57188
Name: Cross Street 1, dtype: int64

Cross Street 2
False    306753
True      57805
```

```
#Perform basic data exploratory analysis to utilise missing value treatment
#Analyze the date column and remove the entries if it has an incorrect timeline
#Delete Summary Rows and Columns in the Dataset.
#Delete Extra Rows like blank rows, page numbers, etc.

df_cols_rmv = ['Unique Key', 'Incident Address', 'Street Name', 'Cross Street 1', 'Cross Street 2',
               'Intersection Street 1', 'Intersection Street 2', 'Landmark', 'Facility Type',
             'Due Date', 'Resolution Description','Community Board', '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', 'Location', 'Address Type', 'Agency',
             'Resolution Action Updated Date', 'Descriptor', 'Location Type']

# Remove the columns added to the df_cols_rmv list from df dataframe
df.drop(df_cols_rmv, axis = 1, inplace = True)
```

```
# reset index, because we droped two rows
df.reset_index(drop=True, inplace=True)

# changing cols with rename()
df.columns = df.columns.str.replace('Complaint Type', 'Complaint')

#Observe your new data - There's no more missing values
df.head()
```

	Created Date	Closed Date	Agency Name	Complaint	Incident Zip	City	Status	Borough	Latitude	Longitude
0	12/31/2015 11:59:45 PM	01/01/2016 12:55:15 AM	New York City Police Department	Noise - Street/Sidewalk	10034.0	NEW YORK	Closed	MANHATTAN	40.865682	-73.923501
1	12/31/2015 11:59:44 PM	01/01/2016 01:26:57 AM	New York City Police Department	Blocked Driveway	11105.0	ASTORIA	Closed	QUEENS	40.775945	-73.915094
2	12/31/2015 11:59:29 PM	01/01/2016 04:51:03 AM	New York City Police Department	Blocked Driveway	10458.0	BRONX	Closed	BRONX	40.870325	-73.888525
3	12/31/2015 11:57:46 PM	01/01/2016 07:43:13 AM	New York City Police Department	Illegal Parking	10461.0	BRONX	Closed	BRONX	40.835994	-73.828379

```
# Checking the missing values after dropping columns that have missing values in the dataset.
df.isnull().sum()
```

```
Created Date       0
Closed Date     2381
Agency Name        0
Complaint          0
Incident Zip    2998
City            2997
Status             0
Borough            0
Latitude        4030
Longitude       4030
dtype: int64
```

```
#Using a for loop in Python to figure out the number of missing values in each column
for column in missing_data.columns.values.tolist():
    print(column)
```

```
        print(missing_data[column].value_counts())
    #Replacing missing data by frequency (mode)
        print("")
```

```
Unique Key
False    364558
Name: Unique Key, dtype: int64

Created Date
False    364558
Name: Created Date, dtype: int64

Closed Date
False    362177
True       2381
Name: Closed Date, dtype: int64

Agency
False    364558
Name: Agency, dtype: int64

Agency Name
False    364558
Name: Agency Name, dtype: int64

Complaint Type
False    364558
Name: Complaint Type, dtype: int64

Descriptor
False    358057
True       6501
Name: Descriptor, dtype: int64

Location Type
False    364425
True        133
Name: Location Type, dtype: int64

Incident Zip
False    361560
True       2998
Name: Incident Zip, dtype: int64

Incident Address
False    312859
True      51699
```

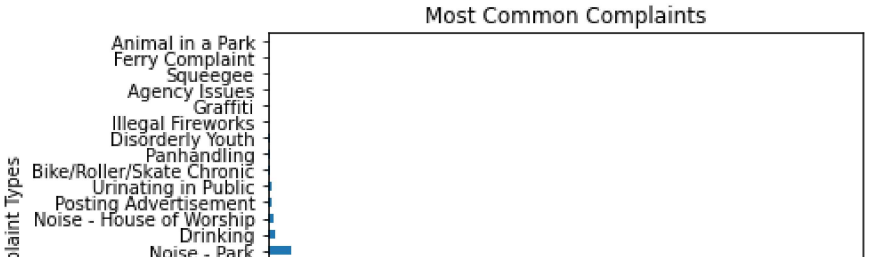
```
Name: Incident Address, dtype: int64

Street Name
False    312859
True      51699
Name: Street Name, dtype: int64

Cross Street 1
False    307370
True      57188
Name: Cross Street 1, dtype: int64

Cross Street 2
False    306753
True      57805
```

```
#3. Draw a frequency plot for city-wise complaints using Histogram bar chart
# Let's calculate the percentage of each Complaint category.
df.Complaint.value_counts(normalize=True)
df.Complaint.describe()
# Display complaint types
df['Complaint'].value_counts()

# Count complain types by city.
df.groupby(['Borough', 'Complaint']).size()
```

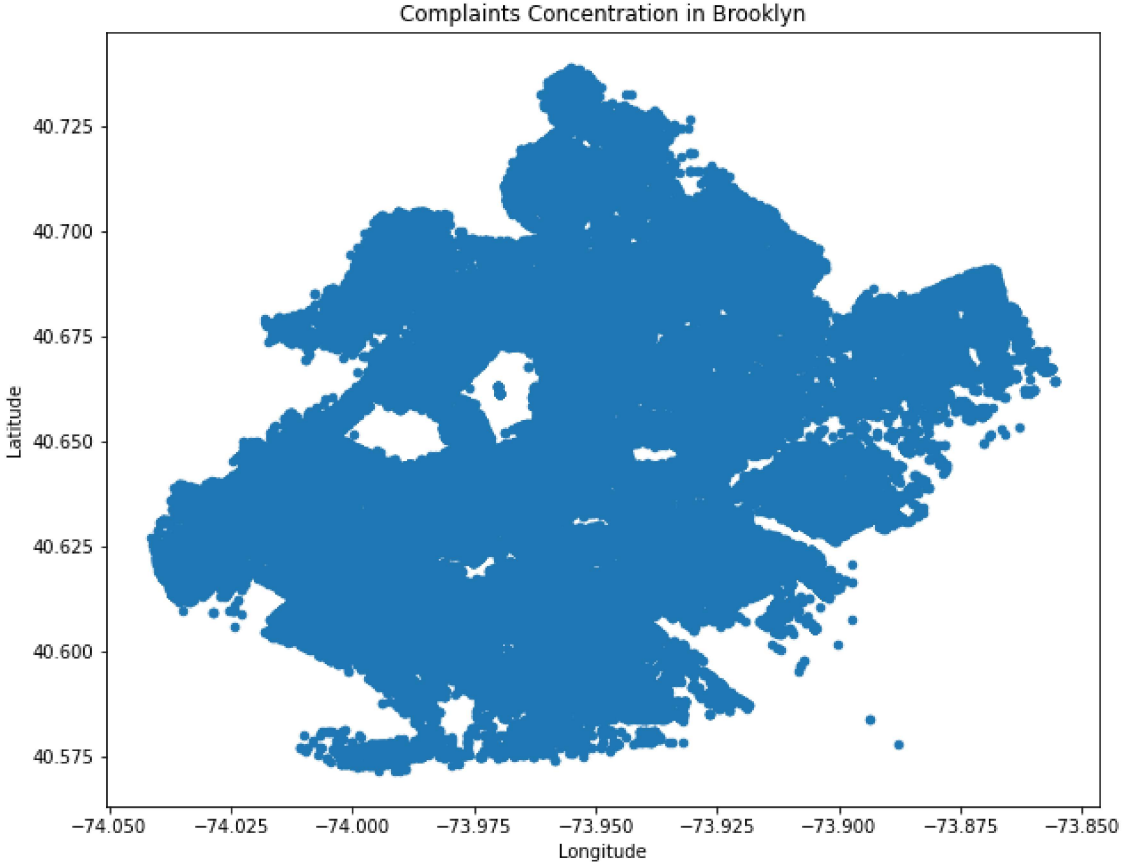
```
Borough      Complaint
BRONX        Animal Abuse                 1971
             Bike/Roller/Skate Chronic      22
             Blocked Driveway            17063
             Derelict Vehicle             2403
             Disorderly Youth               66
                                           ...
Unspecified  Noise - Vehicle                58
             Panhandling                     3
             Posting Advertisement           1
             Traffic                         1
             Vending                         7
Length: 119, dtype: int64
```

```
#plot the bar graph of percentage Complaint
df.Complaint.value_counts(normalize=True).plot.barh()

#df.Complaint.value_counts(normalize=True).plot.pie()

#set x and y labels and plot title
```

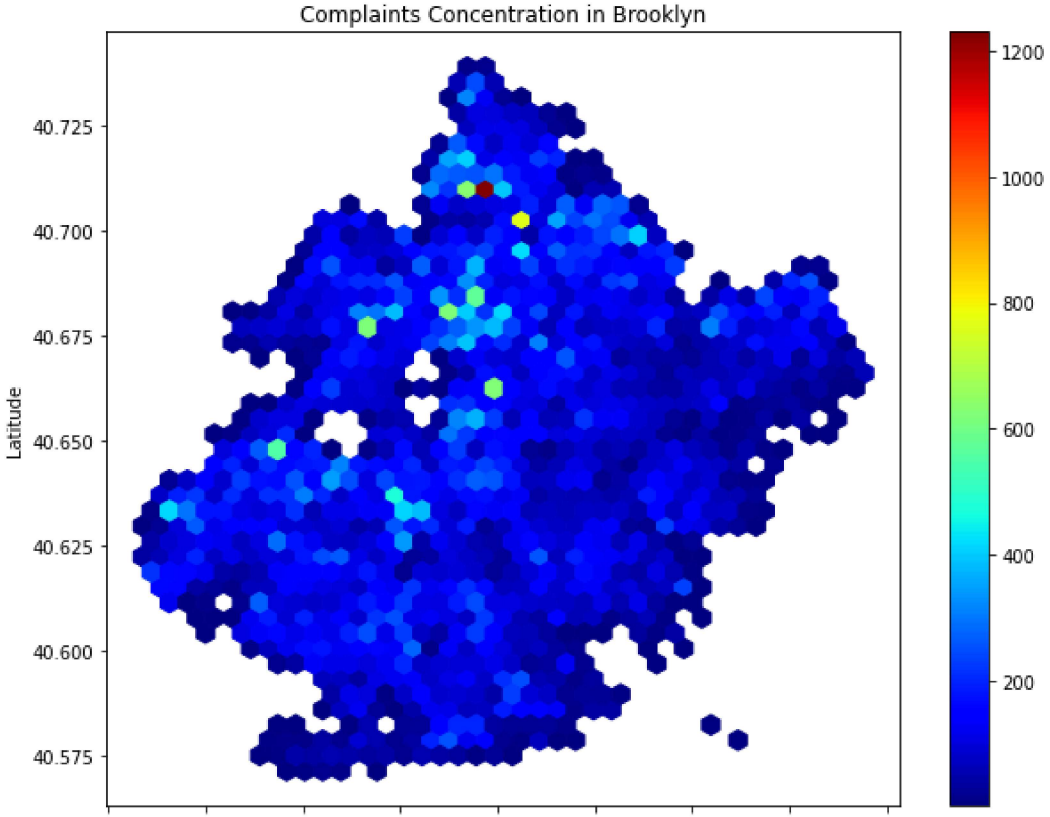
```
plt.xlabel("% Complaint")
plt.ylabel ("Complaint Types")
plt.title ("Most Common Complaints")

plt.show()

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

#Get the top 10 complaint types
#df['Complaint Type'].value_counts().nlargest(10)
```




```
#Filtering Borough = Brooklyn
df_Brooklyn = df.loc[df['Borough'] == 'BROOKLYN']

#Draw scatter plots for complaint concentration across Brooklyn
df_Brooklyn[['Longitude', 'Latitude']].plot(kind = 'scatter', x='Longitude', y='Latitude', title = 'Complaints Concentration in Brooklyn', fi
```



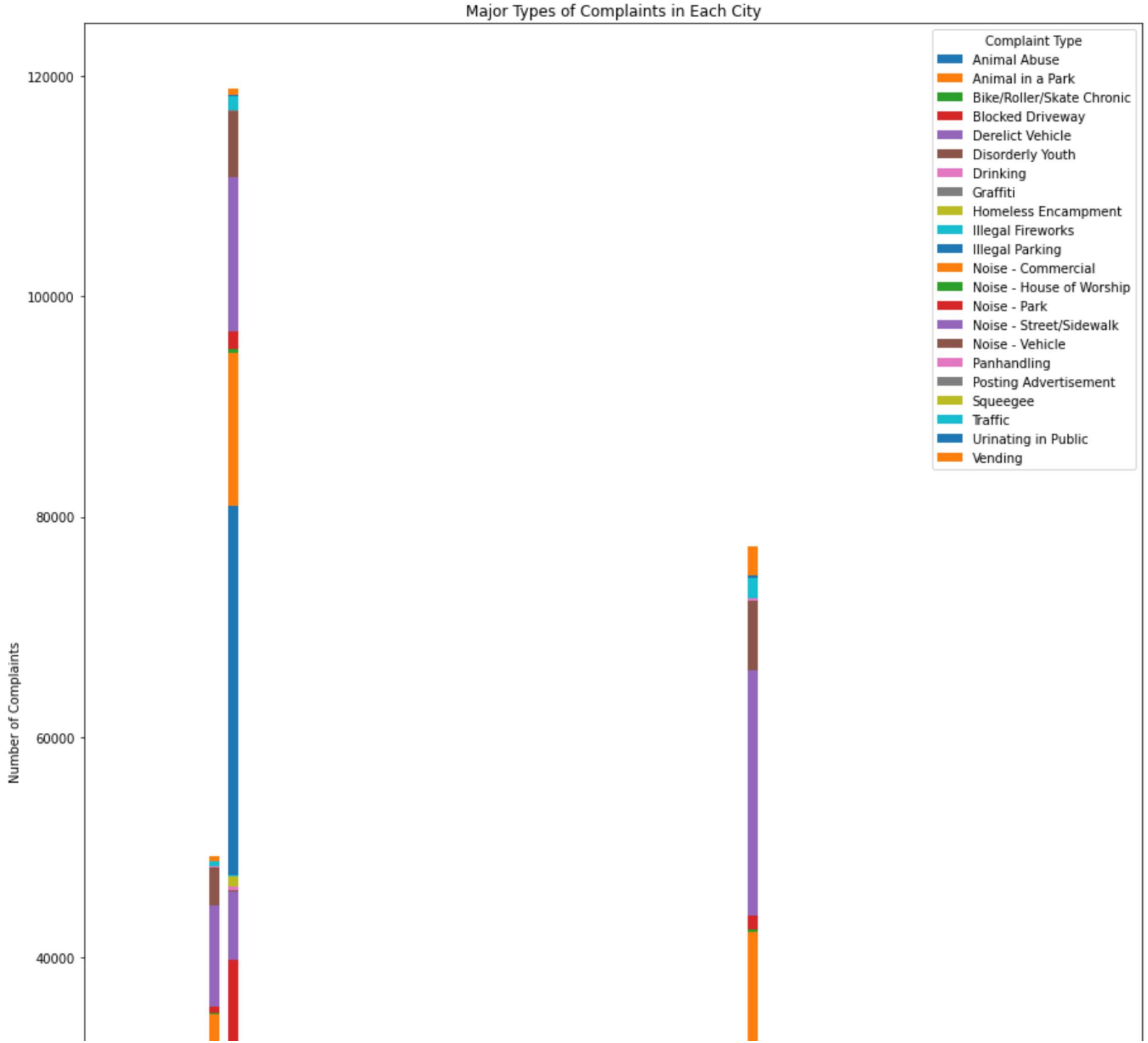
```
#Draw hexbin plot for complaint concentration across Brooklyn
df_Brooklyn[['Longitude', 'Latitude']].plot(kind = 'hexbin', x='Longitude', y='Latitude', gridsize=40,
    colormap = 'jet', mincnt=1, title = 'Complaints Concentration in Brooklyn', figsize = (10, 8));
```




```
#3. Load separate dataset into a pandas dataframe
#Making a list of missing value types
missing_values = ["n/a", "na", "--", "...", "NaN"]
df = pd.read_csv("311_Service_Requests_from_2010_to_Present.csv", na_values = missing_values)


# Group the complaints by city and complaint type
#grouped = df.groupby(['City', 'Complaint Type']).size()
grouped = df.groupby(['City', 'Complaint Type']).size().reset_index(name='Count')

# Print the new dataframe
print(grouped)
```

```
            City           Complaint Type  Count
0     ARVERNE              Animal Abuse     46
1     ARVERNE          Blocked Driveway     50
2     ARVERNE          Derelict Vehicle     32
3     ARVERNE          Disorderly Youth      2
4     ARVERNE                  Drinking      1
..        ...                       ...    ...
772  Woodside          Blocked Driveway     27
773  Woodside          Derelict Vehicle      8
774  Woodside           Illegal Parking    124
775  Woodside       Noise - Commercial      2
776  Woodside  Noise - Street/Sidewalk      5

[777 rows x 3 columns]
```

```
#4. Visualize the major types of complaints in each city
# Pivot the data to make the complaint types the columns
pivoted = grouped.pivot(index='City', columns='Complaint Type', values='Count')

# Plot a stacked bar chart for each city
pivoted.plot(kind='bar', stacked=True, figsize=(15,20))

# Add a title and labels for the axes
plt.title('Major Types of Complaints in Each City')
plt.xlabel('City')
plt.ylabel('Number of Complaints')

# Show the plot
plt.show()
```


Major Types of Complaints in Each City
Number of Complaints
120000
100000
80000
60000
40000
Complaint Type
Animal Abuse
Animal in a Park
Bike/Roller/Skate Chronic
Blocked Driveway
Derelict Vehicle
Disorderly Youth
Drinking
Graffiti
Homeless Encampment
Illegal Fireworks
Illegal Parking
Noise - Commercial
Noise - House of Worship
Noise - Park
Noise - Street/Sidewalk
Noise - Vehicle
Panhandling
Posting Advertisement
Squeegee
Traffic
Urinating in Public
Vending

```
#5. Analyse Response Time by changing Created & Closed date columns to YYYY-MM-DD format
df['Created Date'] = pd.to_datetime(df['Created Date'])
df['Closed Date'] = pd.to_datetime(df['Closed Date'])
```

```
#Check whether the date are in the correct order
df.loc[df['Created Date']>=df['Closed Date']].shape
```

```
(0, 53)
```

```
# Drop rows where 'Closed Date' is empty
df = df[df['Closed Date'].notnull()]
```

```
#Create a new column 'Resolution Time' in terms of days, as the time difference between 'Created Date' & 'Closed date'.
df['Resolution Time'] = (df['Closed Date'] - df['Created Date']).dt.days
df.head()
```

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	Incident Zip	Incident Address	...	Br Hig Direc
0	32310363	2015-12-31 23:59:45	2016-01-01 00:55:15	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Music/Party	Street/Sidewalk	10034.0	71 VERMILYEA AVENUE	...	
1	32309934	2015-12-31 23:59:44	2016-01-01 01:26:57	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewalk	11105.0	27-07 23 AVENUE	...	
2	32309159	2015-12-31 23:59:29	2016-01-01 04:51:03	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewalk	10458.0	2897 VALENTINE AVENUE	...	
3	32305098	2015-12-31 23:57:46	2016-01-01 07:43:13	NYPD	New York City Police Department	Illegal Parking	Commercial Overnight Parking	Street/Sidewalk	10461.0	2940 BAISLEY AVENUE	...	
4	32306529	2015-12-31 23:56:58	2016-01-01 03:24:42	NYPD	New York City Police Department	Illegal Parking	Blocked Sidewalk	Street/Sidewalk	11373.0	87-14 57 ROAD	...	

5 rows × 54 columns

```
#Create a new column 'Resolution Time' in terms of seconds, as the time difference between 'Created Date' & 'Closed date'.
df['Resolution Time'] = (df['Closed Date'] - df['Created Date']).dt.total_seconds()
df.head()
```

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	Incident Zip	Incident Address	...	Br Hig Direc
0	32310363	2015-12-31 23:59:45	2016-01-01 00:55:15	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Music/Party	Street/Sidewalk	10034.0	71 VERMILYEA AVENUE	...	
1	32309934	2015-12-31 23:59:44	2016-01-01 01:26:57	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewalk	11105.0	27-07 23 AVENUE	...	
2	32309159	2015-12-31 23:59:29	2016-01-01 04:51:03	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewalk	10458.0	2897 VALENTINE AVENUE	...	
3	32305098	2015-12-31 23:57:46	2016-01-01 07:43:13	NYPD	New York City Police Department	Illegal Parking	Commercial Overnight Parking	Street/Sidewalk	10461.0	2940 BAISLEY AVENUE	...	
4	32306529	2015-12-31 23:56:58	2016-01-01 03:24:42	NYPD	New York City Police Department	Illegal Parking	Blocked Sidewalk	Street/Sidewalk	11373.0	87-14 57 ROAD	...	

5 rows × 54 columns

```
# Get the response resolution time according to complaint type
df.groupby('Complaint Type')['Resolution Time'].mean().sort_values()
```

```
Complaint Type
Agency Issues                0.000000
Squeegee                     0.000000
Posting Advertisement        0.001473
Disorderly Youth             0.003175
Illegal Fireworks            0.005814
Noise - Commercial           0.005943
Noise - Park                 0.006359
Urinating in Public          0.007800
Noise - Vehicle              0.007875
Vending                      0.008363
Bike/Roller/Skate Chronic    0.008421
```

```
Noise - House of Worship      0.008427
Noise - Street/Sidewalk       0.008917
Drinking                      0.009972
Homeless Encampment           0.010248
Traffic                       0.010970
Blocked Driveway              0.011786
Illegal Parking               0.012964
Animal Abuse                  0.020513
Panhandling                   0.024615
Graffiti                      0.044586
Derelict Vehicle              0.065155
Animal in a Park             14.000000
Name: Resolution Time, dtype: float64
```

```
# Get the response resolution time according to City & Complaint type
df.groupby(['City','Complaint Type'])['Resolution Time'].mean()
```

```
City      Complaint Type
ARVERNE   Animal Abuse               8399.195652
          Blocked Driveway           8318.840000
          Derelict Vehicle          11394.000000
          Disorderly Youth          12928.500000
          Drinking                    859.000000
                                        ...
Woodside  Blocked Driveway          15566.185185
          Derelict Vehicle          19994.500000
          Illegal Parking           17293.459677
          Noise - Commercial         8619.000000
          Noise - Street/Sidewalk   12285.600000
Name: Resolution Time, Length: 777, dtype: float64
```

```
#Focus on Bike/Roller/Skate Chronic Resolution time
df[df['Complaint Type'] == 'Bike/Roller/Skate Chronic']
df.loc[:,['Resolution Time']]
```

	Resolution Time
0	3330.0
1	5233.0
2	17494.0
3	27927.0
4	12464.0
...	...

```
#6. Using crosstab and Chi square test to check if the complaints and city are related
pd.crosstab(df["City"],df["Complaint Type"])
```

Complaint Type City	Animal Abuse	Animal in a Park	Bike/Roller/Skate Chronic	Blocked Driveway	Derelict Vehicle	Disorderly Youth	Drinking	Graffiti	Homeless Encampment	Illega Firework
ARVERNE	46	0	0	50	32	2	1	1	4	
ASTORIA	170	0	16	3436	426	5	43	4	32	
Astoria	0	0	0	159	14	0	0	0	0	
BAYSIDE	53	0	0	514	231	2	1	3	2	
BELLEROSE	15	0	1	138	120	2	1	0	1	
BREEZY POINT	2	0	0	3	3	0	1	0	0	
BRONX	1971	0	22	17062	2402	66	206	15	275	2
BROOKLYN	3191	0	124	36445	6257	79	291	60	948	6
CAMBRIA HEIGHTS	15	0	0	177	148	0	0	0	6	
CENTRAL PARK	0	0	0	0	0	0	0	0	0	
COLLEGE POINT	35	0	0	597	223	1	1	2	3	
CORONA	104	0	0	3597	72	6	34	4	26	
EAST ELMHURST	85	0	1	1925	136	1	9	3	2	
ELMHURST	59	0	2	1992	94	2	13	1	34	
East Elmhurst	0	0	0	0	2	0	0	0	0	
FAR ROCKAWAY	111	0	0	383	215	1	4	0	16	
FLORAL PARK	7	0	0	33	74	1	1	0	0	
FLUSHING	191	0	3	3640	532	2	47	6	26	
FOREST	78	0	6	873	71	1	1	3	18	

HILLS

```
#import required libraries
from scipy.stats import chi2_contingency
```

```
#contingency table
table = pd.crosstab(df["City"],df["Complaint Type"])

# Get chi-square value , p-value, degrees of freedom, expected frequencies using the function chi2_contingency
stat, p, dof, expected = chi2_contingency(table)

# select significance value
alpha = 0.05

# Determine whether to reject or keep your null hypothesis
print('significance=%.3f, p=%.3f' % (alpha, p))
if p <= alpha:
    print('Variables are associated (reject H0)')
else:
    print('Variables are not associated(fail to reject H0)')
```

```
 significance=0.050, p=0.000
 Variables are associated (reject H0)
```

```
# Perform a chi-square test for independence
#The chi-square value measures the degree of association between the two categorical variables, and the p-value measures the statistical sigr

chi2, p_value, dof, expected = stats.chi2_contingency(table)
# Print the results of the chi-square test
print('Chi-square value:', chi2)
print('P-value:', p_value)
print('Degrees of freedom:', dof)
print('Expected frequencies:', expected)
```

```
 Chi-square value: 141343.23155024176
 P-value: 0.0
 Degrees of freedom: 1092
 Expected frequencies: [[7.54353629e+00 7.16453252e-04 3.38882388e-01 ... 3.72125819e+00
   4.59246535e-01 2.99764041e+00]
  [2.32742851e+02 2.21049341e-02 1.04556338e+01 ... 1.14813028e+02
   1.41692628e+01 9.24870444e+01]
  [2.63586886e+01 2.50343704e-03 1.18412572e+00 ... 1.30028520e+01
   1.60470314e+00 1.04743806e+01]
  ...
```