In [1]:

```
#importing required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
# importing the CSV file into Python environment
raw_data = pd.read_csv(r'D:\education\simplilearn\data science\projects\Project-1 311 NYC c
```

In [3]:

```
# checking the top five observations of the data imported
raw_data.head()
```

Out[3]:

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Typ
0	32310363	12/31/2015 11:59:45 PM	01-01-2016 00:55	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Music/Party	Street/Sidewal
1	32309934	12/31/2015 11:59:44 PM	01-01-2016 01:26	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewal
2	32309159	12/31/2015 11:59:29 PM	01-01-2016 04:51	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewal
3	32305098	12/31/2015 11:57:46 PM	01-01-2016 07:43	NYPD	New York City Police Department	Illegal Parking	Commercial Overnight Parking	Street/Sidewal
4	32306529	12/31/2015 11:56:58 PM	01-01-2016 03:24	NYPD	New York City Police Department	Illegal Parking	Blocked Sidewalk	Street/Sidewal

5 rows × 53 columns

In [4]:

```
# checking the type of data
type(raw_data)
```

Out[4]:

```
pandas.core.frame.DataFrame
```

Identifing the shape of the raw dataset

In [5]:

```
raw_data.shape
```

Out[5]:

```
(364558, 53)
```

Identifing the variables with null values

In [6]:

```
raw_data.isnull().sum()
```

```
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
```

Creating a separate DataFrame from the raw_data with required columns as per problem statement

As per the problem statement, required columns are :

- Unique Key
- Created Date
- Closed Date
- Complaint Type
- City
- Latitude
- Longitude

Thus, creating a separate data frame with the selected columns:

In [7]:

```
# Creating a new data frame with above mentioned columns
data = raw_data[['Unique Key','Created Date','Closed Date', 'Complaint Type', 'City', 'Lati
```

In [8]:

```
# Checking the top five observations of the new dataframe created:
data.head()
```

Out[8]:

	Unique Key	Created Date	Closed Date	Complaint Type	City	Latitude	Longitude
0	32310363	12/31/2015 11:59:45 PM	01-01-2016 00:55	Noise - Street/Sidewalk	NEW YORK	40.865682	-73.923501
1	32309934	12/31/2015 11:59:44 PM	01-01-2016 01:26	Blocked Driveway	ASTORIA	40.775945	-73.915094
2	32309159	12/31/2015 11:59:29 PM	01-01-2016 04:51	Blocked Driveway	BRONX	40.870325	-73.888525
3	32305098	12/31/2015 11:57:46 PM	01-01-2016 07:43	Illegal Parking	BRONX	40.835994	-73.828379
4	32306529	12/31/2015 11:56:58 PM	01-01-2016 03:24	Illegal Parking	ELMHURST	40.733060	-73.874170

1.1 Identifying the shape of the data set

In [9]:

```
data.shape
```

Out[9]:

```
(364558, 7)
```

1.2 Identifying the variables with Null values

In [10]:

```
data.isna().sum()
```

Out[10]:

```
Unique Key           0
Created Date         0
Closed Date       2381
Complaint Type       0
City              2997
Latitude          4030
Longitude         4030
dtype: int64
```

It shows that in the data selected, Closed Date column, Due date column and city column have Null values in the rows

Basic data exploratory analysis

In [11]:

```
# checking the column data types
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 364558 entries, 0 to 364557
Data columns (total 7 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   Complaint Type  364558 non-null  object 
 4   City            361561 non-null  object 
 5   Latitude        360528 non-null  float64
 6   Longitude       360528 non-null  float64
dtypes: float64(2), int64(1), object(4)
memory usage: 19.5+ MB
```

Here, it can be seen that date columns (Created Date & Closed Date) are in object type which is incorrect format. They are to be in datetime format. Thus, they are to be coverted into correct format.

Basic data exploratory analysis

2.1 Analyzing the date column

In [12]:

```
#Correcting the datetime to the correct datetime format
data['Created Date'] = pd.to_datetime(data['Created Date'])
data['Closed Date'] = pd.to_datetime(data['Closed Date'])
```

In [13]:

```
# checking whether date columns are converted into correct format or not
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 364558 entries, 0 to 364557
Data columns (total 7 columns):
 #   Column          Non-Null Count   Dtype         
---  ------          --------------   -----         
 0   Unique Key      364558 non-null  int64         
 1   Created Date    364558 non-null  datetime64[ns]
 2   Closed Date     362177 non-null  datetime64[ns]
 3   Complaint Type  364558 non-null  object        
 4   City            361561 non-null  object        
 5   Latitude        360528 non-null  float64       
 6   Longitude       360528 non-null  float64       
dtypes: datetime64[ns](2), float64(2), int64(1), object(2)
memory usage: 19.5+ MB
```

It shows that date columns are converted into correct format and all other columns are in correct format

Basic data exploratory analysis

2.1 Missing value treatment: Checking missing values in the columns and removing the observations with Null values

In [14]:

```
#checking for any Null values
data.isna().sum()
```

Out[14]:

```
Unique Key           0
Created Date         0
Closed Date       2381
Complaint Type       0
City              2997
Latitude          4030
Longitude         4030
dtype: int64
```

In [15]:

```
#dropping the observations with Null values in the columns
data = data.dropna(subset = ['Closed Date','City','Latitude','Longitude'])
```

In [16]:

```
#checking the new shape of the final cleaned data
data.shape
```

Out[16]:

```
(360429, 7)
```

In [17]:

```
# checking for any Null values
data.isnull().sum()
```

Out[17]:

```
Unique Key        0
Created Date      0
Closed Date       0
Complaint Type    0
City              0
Latitude          0
Longitude         0
dtype: int64
```

Here, it is found that all the Null values are removed from the data and dates are also in correct format. This finally cleaned data can be saved into an excel file using to_excel() command.

Basic data exploratory analysis

2.3 Drawing a frequency plot for city-wise complaints

In [18]:

```
#grouping the data city wise for plotting their complaint_count frequencies
```

In [19]:

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

```
SUNNYSIDE                944
Astoria                  905
ROCKAWAY PARK            829
OAKLAND GARDENS          715
LITTLE NECK              712
CAMBRIA HEIGHTS          617
BELLEROSE                487
GLEN OAKS                361
ARVERNE                  258
FLORAL PARK              196
Long Island City         170
Woodside                 166
NEW HYDE PARK            129
CENTRAL PARK             110

QUEENS                    36
BREEZY POINT              31
East Elmhurst             30
Howard Beach               1
Name: City, dtype: int64
```

From the above output, it is found that there are few city names which are entered twice with upper case and lower case. Their names to be corrected as python considers them as different objects as python is case sensitive.

In [20]:

```
# correcting the names of the cities which are in lower case
data['City'] = data['City'].replace(['Howard Beach','East Elmhurst','Woodside','Long Island
```

In [21]:

```
#checking for the change
pd.DataFrame(data['City'].value_counts())
```

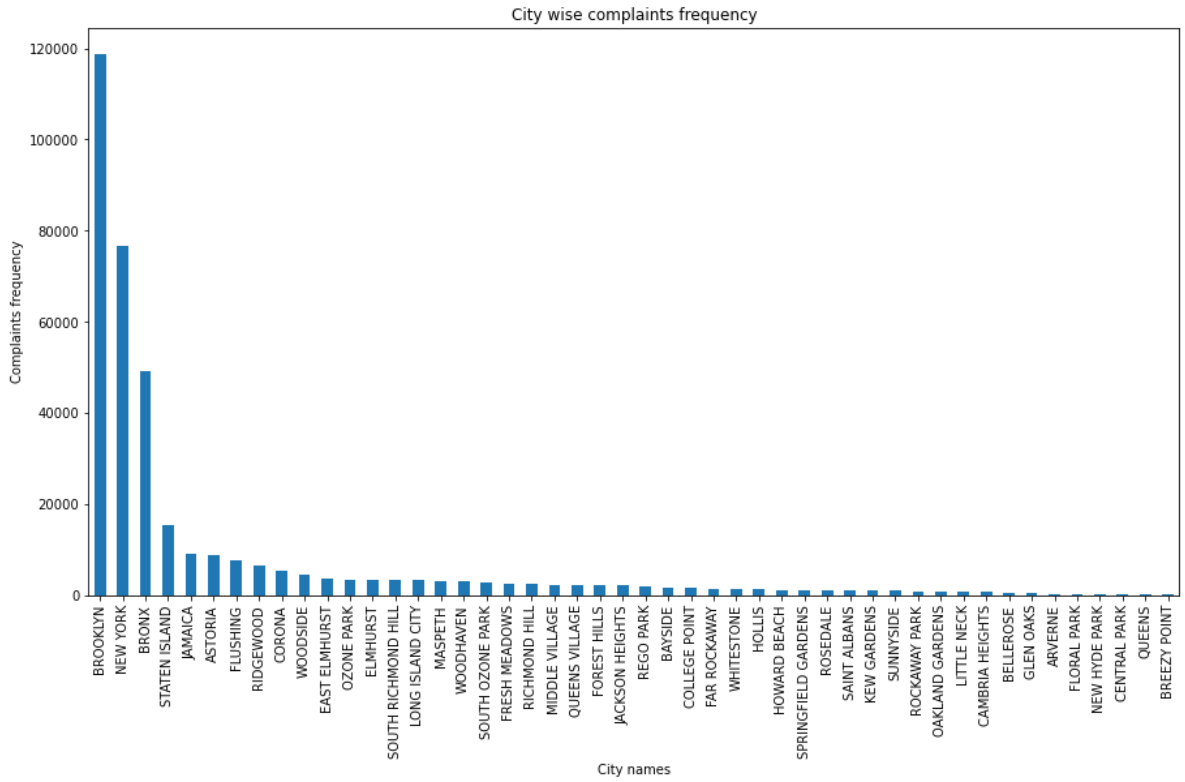
Out[21]:

	City
BROOKLYN	118632
NEW YORK	76634
BRONX	49048
STATEN ISLAND	15326
JAMAICA	8920
ASTORIA	8879
FLUSHING	7481
RIDGEWOOD	6388
CORONA	5382
WOODSIDE	4520
EAST ELMHURST	3587
OZONE PARK	3446
ELMHURST	3438
SOUTH RICHMOND HILL	3430
LONG ISLAND CITY	3189
MASPETH	3116
WOODHAVEN	3102
SOUTH OZONE PARK	2668
FRESH MEADOWS	2449
RICHMOND HILL	2333
MIDDLE VILLAGE	2290
QUEENS VILLAGE	2251
FOREST HILLS	2120
JACKSON HEIGHTS	2105
REGO PARK	1805
BAYSIDE	1548
COLLEGE POINT	1544
FAR ROCKAWAY	1396
WHITESTONE	1367
HOLLIS	1231
HOWARD BEACH	1144
SPRINGFIELD GARDENS	1094
ROSEDALE	1086

	City
SAINT ALBANS	1047
KEW GARDENS	1008
SUNNYSIDE	944
ROCKAWAY PARK	829
OAKLAND GARDENS	715
LITTLE NECK	712
CAMBRIA HEIGHTS	617
BELLEROSE	487
GLEN OAKS	361
ARVERNE	258
FLORAL PARK	196
NEW HYDE PARK	129
CENTRAL PARK	110
QUEENS	36
BREEZY POINT	31

In [22]:

```
# plotting the bar graph uning pandas library
plt.figure(figsize = (15,8))
data['City'].value_counts().plot(kind = 'bar', title= 'City wise complaints frequency')
plt.ylabel('Complaints frequency')
plt.xlabel('City names')
plt.show()
```



Basic data exploratory analysis

2.4 scatter and hexbin plots for complaint concentration across Brooklyn

In [23]:

```
brooklyn_data = data[data['City'] == 'BROOKLYN']
```

In [24]:

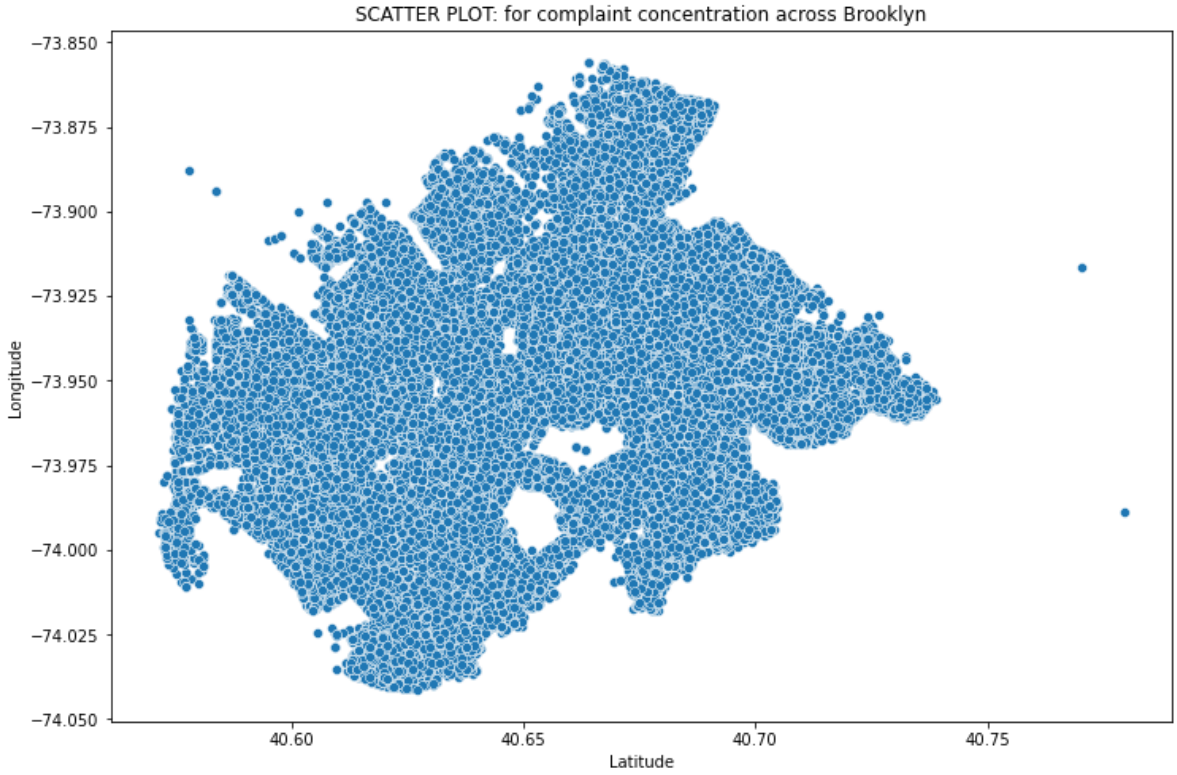
```
brooklyn_data.head()
```

Out[24]:

	Unique Key	Created Date	Closed Date	Complaint Type	City	Latitude	Longitude
5	32306554	2015-12-31 23:56:30	2016-01-01 01:50:00	Illegal Parking	BROOKLYN	40.660823	-73.992568
9	32308391	2015-12-31 23:53:58	2016-01-01 01:17:00	Blocked Driveway	BROOKLYN	40.623793	-73.999539
13	32305074	2015-12-31 23:47:58	2016-01-01 08:18:00	Illegal Parking	BROOKLYN	40.687511	-73.874505
17	32310273	2015-12-31 23:44:52	2016-01-01 00:36:00	Noise - Commercial	BROOKLYN	40.679154	-73.983430
18	32306617	2015-12-31 23:40:59	2016-01-01 02:37:00	Noise - Commercial	BROOKLYN	40.616550	-73.930202

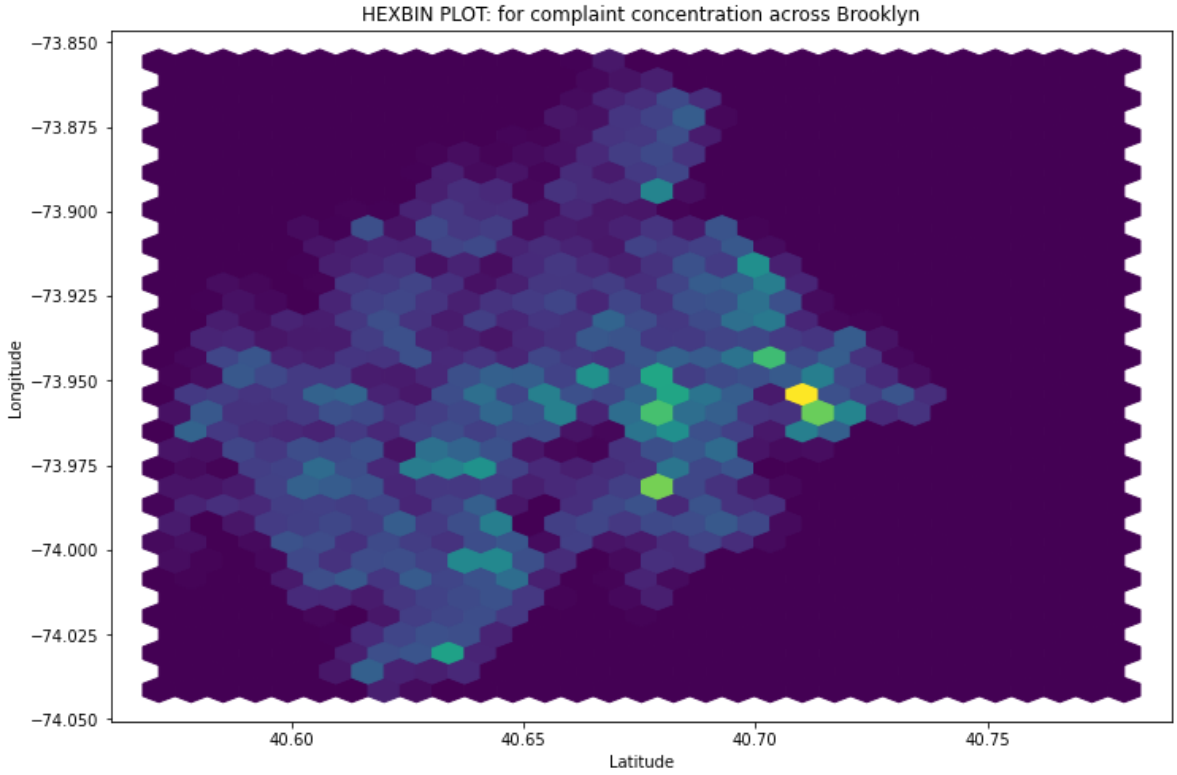
In [25]:

```
x = brooklyn_data['Latitude']
y = brooklyn_data['Longitude']
plt.figure(figsize = (12,8))
sns.scatterplot(x,y)
plt.title('SCATTER PLOT: for complaint concentration across Brooklyn')
plt.show()
```



In [26]:

```
plt.figure(figsize = (12,8))
plt.hexbin(x,y, gridsize=30)
plt.xlabel('Latitude')
plt.ylabel('Longitude')
plt.title('HEXBIN PLOT: for complaint concentration across Brooklyn')
plt.show()
```



In [ ]:

Finding major type of complaints

3.1 Plotting a bar graph of count vs. complaint types

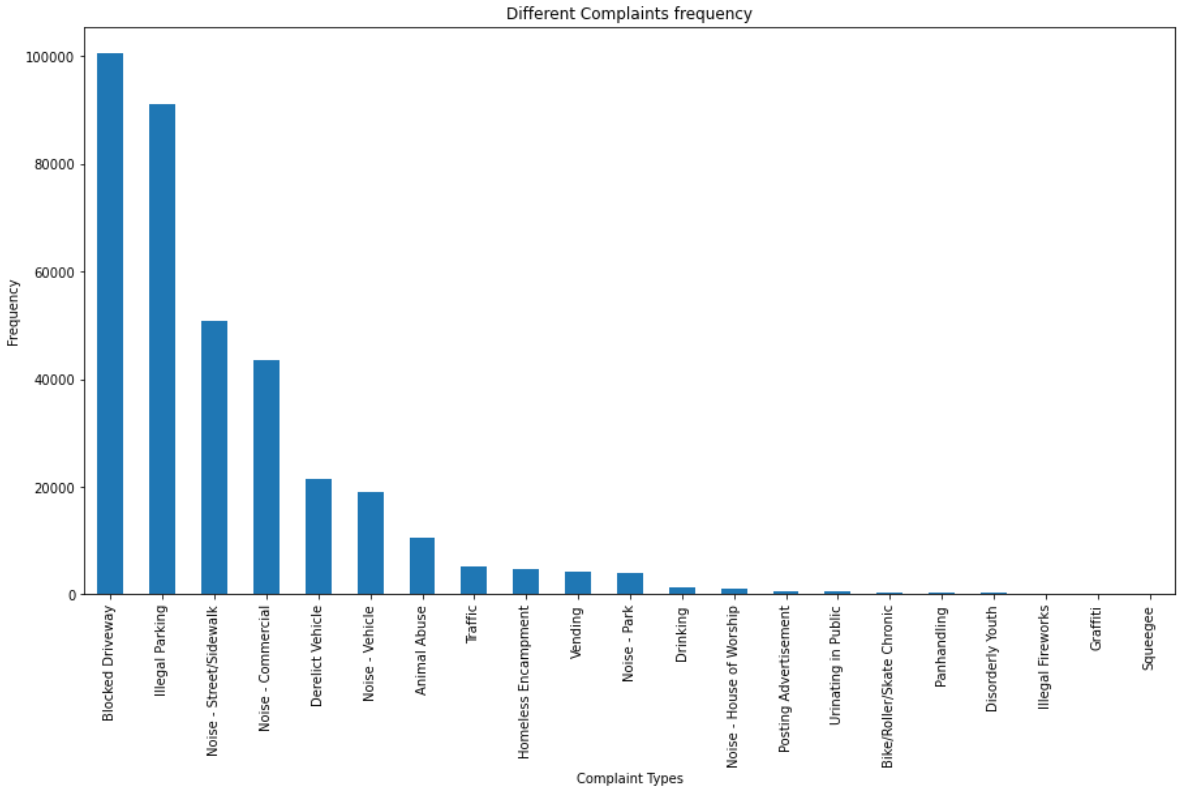
In [27]:

```
print(data['Complaint Type'].value_counts())
```

```
Blocked Driveway             100492
Illegal Parking               91092
Noise - Street/Sidewalk       50800
Noise - Commercial            43629
Derelict Vehicle              21427
Noise - Vehicle               19125
Animal Abuse                  10503
Traffic                        5167
Homeless Encampment            4829
Vending                        4164
Noise - Park                   3995
Drinking                       1400
Noise - House of Worship       1061
Posting Advertisement           679
Urinating in Public             641
Bike/Roller/Skate Chronic       463
Panhandling                     320
Disorderly Youth                314
Illegal Fireworks               167
Graffiti                        157
Squeegee                          4
Name: Complaint Type, dtype: int64
```

In [28]:

```
# plotting graph for count Vs complaint types
plt.figure(figsize = (15,8))
data['Complaint Type'].value_counts().plot(kind = 'bar', title = 'Different Complaints freq
plt.xlabel('Complaint Types')
plt.ylabel('Frequency')
plt.show()
```



Finding major type of complaints

3.2 Top 10 types of complaints

In [29]:

```
#top 10 types of complaints
major_complaint_types = pd.DataFrame(data['Complaint Type'].value_counts().head(10))
major_complaint_types
```

Out[29]:

	Complaint Type
Blocked Driveway	100492
Illegal Parking	91092
Noise - Street/Sidewalk	50800
Noise - Commercial	43629
Derelict Vehicle	21427
Noise - Vehicle	19125
Animal Abuse	10503
Traffic	5167
Homeless Encampment	4829
Vending	4164

Displaying type of complaints in each city in a separate data set and Visualizing the major type of complaints in each city

In [30]:

```
# Creating a list of City names
city_list = data['City'].unique()
```

In [31]:

```
print(city_list)
```

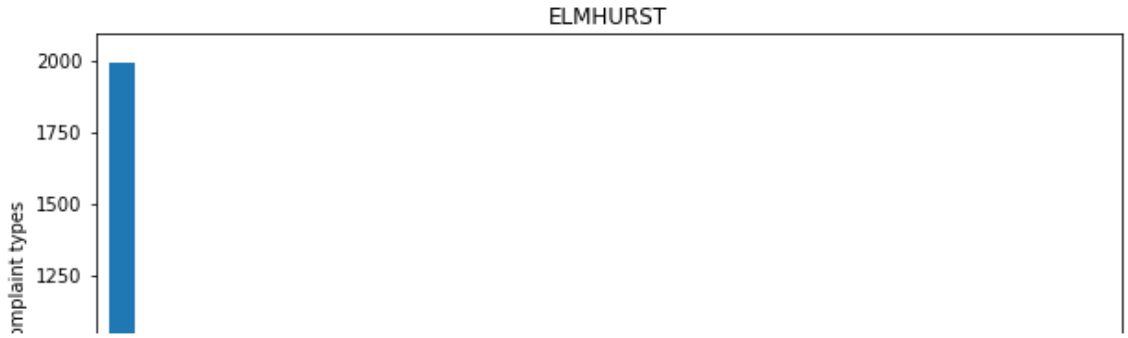
```
['NEW YORK' 'ASTORIA' 'BRONX' 'ELMHURST' 'BROOKLYN' 'KEW GARDENS'
 'JACKSON HEIGHTS' 'MIDDLE VILLAGE' 'REGO PARK' 'SAINT ALBANS' 'JAMAICA'
 'SOUTH RICHMOND HILL' 'RIDGEWOOD' 'HOWARD BEACH' 'FOREST HILLS'
 'STATEN ISLAND' 'OZONE PARK' 'RICHMOND HILL' 'WOODHAVEN' 'FLUSHING'
 'CORONA' 'QUEENS VILLAGE' 'OAKLAND GARDENS' 'HOLLIS' 'MASPETH'
 'EAST ELMHURST' 'SOUTH OZONE PARK' 'WOODSIDE' 'FRESH MEADOWS'
 'LONG ISLAND CITY' 'ROCKAWAY PARK' 'SPRINGFIELD GARDENS' 'COLLEGE POINT'
 'BAYSIDE' 'GLEN OAKS' 'FAR ROCKAWAY' 'BELLEROSE' 'LITTLE NECK'
 'CAMBRIA HEIGHTS' 'ROSEDALE' 'SUNNYSIDE' 'WHITESTONE' 'ARVERNE'
 'FLORAL PARK' 'NEW HYDE PARK' 'CENTRAL PARK' 'BREEZY POINT' 'QUEENS']
```

In [32]:

```
# Crating a dict to store each city data in separate data set with city as key and its data
city_data = {}
```

In [33]:

```
# creating a for loop to store data into city_data dict and visualize each city complaints
for x in city_list:
  print('Type of complaints and their count in the city:', x)
  city_data[x] = data[(data['City'])== x ]['Complaint Type'].value_counts()
  print(city_data[x])
  plt.figure(figsize = (10,6))
  data[(data['City'])== x ]['Complaint Type'].value_counts().plot(kind = 'bar')
  plt.xlabel('Complaint Types')
  plt.ylabel('frequency of complaint types')
  plt.title(x)
  plt.show()
  print('\n')
```



In [ ]:

5.the average response time across various types of complaints

In [34]:

```
# fetching different type of complaints and making list of it
complaint_types = data['Complaint Type'].unique()
```

In [35]:

```
complaint_types
```

Out[35]:

```
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', 'Squeegee'],
      dtype=object)
```

In [36]:

```
#Creating a dict to store avg response time of each type of complaint
avg_response_time = {}
```

In [37]:

```
for x in complaint_types:
  avg_response_time[x] = (data[(data['Complaint Type'] == x)]['Closed Date'] - data[(data['
```

In [38]:

```
avg_response_time
```

Out[38]:

```
{'Noise - Street/Sidewalk': Timedelta('0 days 03:23:38.634311023'),
 'Blocked Driveway': Timedelta('0 days 04:30:17.743133781'),
 'Illegal Parking': Timedelta('0 days 04:19:44.270638475'),
 'Derelict Vehicle': Timedelta('0 days 07:01:20.382228030'),
 'Noise - Commercial': Timedelta('0 days 03:04:01.751105915'),
 'Noise - House of Worship': Timedelta('0 days 03:09:58.348727615'),
 'Posting Advertisement': Timedelta('0 days 02:01:25.103092783'),
 'Noise - Vehicle': Timedelta('0 days 03:29:46.473882352'),
 'Animal Abuse': Timedelta('0 days 05:00:51.121679520'),
 'Vending': Timedelta('0 days 03:59:14.567243035'),
 'Traffic': Timedelta('0 days 03:25:15.018579446'),
 'Drinking': Timedelta('0 days 03:50:03.062857142'),
 'Bike/Roller/Skate Chronic': Timedelta('0 days 03:35:41.935205183'),
 'Panhandling': Timedelta('0 days 04:23:42.390625'),
 'Noise - Park': Timedelta('0 days 03:23:22.648060075'),
 'Homeless Encampment': Timedelta('0 days 04:17:55.948643611'),
 'Urinating in Public': Timedelta('0 days 03:35:58.723868954'),
 'Graffiti': Timedelta('0 days 06:27:55.515923566'),
 'Disorderly Youth': Timedelta('0 days 03:26:35.308917197'),
 'Illegal Fireworks': Timedelta('0 days 02:48:41.113772455'),
 'Squeegee': Timedelta('0 days 04:02:44.250000')}
```

In [39]:

```
df_avg_response_time = pd.DataFrame(avg_response_time, index = ['Avg response time'])
```

In [40]:

```
df_avg_response_time
```

Out[40]:

	Noise - Street/Sidewalk	Blocked Driveway	Illegal Parking	Derelict Vehicle	
Avg response time	0 days 03:23:38.634311023	0 days 04:30:17.743133781	0 days 04:19:44.270638475	0 days 07:01:20.382228030	03:04:

1 rows × 21 columns

In [41]:

```
df_avg_response_time.shape
```

Out[41]:

```
(1, 21)
```

In [42]:

```
#transposing the df_avg_response_time dataframe
df_avg_response_time = df_avg_response_time.T
```

In [43]:

```
df_avg_response_time
```

Out[43]:

	Avg response time
Noise - Street/Sidewalk	0 days 03:23:38.634311023
Blocked Driveway	0 days 04:30:17.743133781
Illegal Parking	0 days 04:19:44.270638475
Derelict Vehicle	0 days 07:01:20.382228030
Noise - Commercial	0 days 03:04:01.751105915
Noise - House of Worship	0 days 03:09:58.348727615
Posting Advertisement	0 days 02:01:25.103092783
Noise - Vehicle	0 days 03:29:46.473882352
Animal Abuse	0 days 05:00:51.121679520
Vending	0 days 03:59:14.567243035
Traffic	0 days 03:25:15.018579446
Drinking	0 days 03:50:03.062857142
Bike/Roller/Skate Chronic	0 days 03:35:41.935205183
Panhandling	0 days 04:23:42.390625
Noise - Park	0 days 03:23:22.648060075
Homeless Encampment	0 days 04:17:55.948643611
Urinating in Public	0 days 03:35:58.723868954
Graffiti	0 days 06:27:55.515923566
Disorderly Youth	0 days 03:26:35.308917197
Illegal Fireworks	0 days 02:48:41.113772455
Squeegee	0 days 04:02:44.250000

In [ ]:

```

```