In [1]:
```
import pandas as pd
```

In [2]:
```
import numpy as np
```

In [3]:
```
import matplotlib.pyplot as plt
```

In [4]:
```
import seaborn as sns
```

In [5]:
```
from matplotlib import style
```

In [6]:
```
%matplotlib inline
```

In [7]:
```
cust_service = pd.read_csv('C:/Users/User/Desktop/Data Science Notes/Project/PS D
```

In [8]:
```
## Understanding the Dataset
```

In [9]:
```
cust_service.head()
```

Out[9]:

	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 12:55:15 AM	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 01:26:57 AM	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewal
2	32309159	12/31/2015 11:59:29 PM	01/01/2016 04:51:03 AM	NYPD	New York City Police Department	Blocked Driveway	No Access	Street/Sidewal
3	32305098	12/31/2015 11:57:46 PM	01/01/2016 07:43:13 AM	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 03:24:42 AM	NYPD	New York City Police Department	Illegal Parking	Blocked Sidewalk	Street/Sidewal

5 rows × 53 columns

In [10]:
```
# Shape of Dataset
cust_service.shape
```

Out[10]:
```
(364558, 53)
```

In [11]:
```
# Variables with null values
```

In [12]:
```
cust_service.isna()
```

Out[12]:

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	Incident Zip	Inc Add
0	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	
364553	False	False	False	False	False	False	False	False	False	
364554	False	False	False	False	False	False	False	False	False	
364555	False	False	False	False	False	False	False	False	False	
364556	False	False	False	False	False	False	False	False	False	
364557	False	False	False	False	False	False	False	False	False	

364558 rows × 53 columns

In [13]:
```
cust_service.isna().any()
```

Out[13]:
```
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
```

In [14]:
```
cust_service.isna().sum()
```

Out[14]:
```
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
```

In [15]:
```
cust_service.columns
```

Out[15]:
```
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')
```

In [16]:
```
## Exploratory Data Analysis
```

In [17]:
```
# Missing Value Treatment
```

Type *Markdown* and LaTeX: α^2

In [18]:
```
# Find Columns without Object values / with Number values
```

In [19]:
```
num = cust_service.select_dtypes(exclude='object')
```

In [20]:
```
num.head()
```

Out[20]:

	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	
0	32310363	10034.0	1005409.0	254678.0	NaN	NaN	NaN	NaN	NaN	40
1	32309934	11105.0	1007766.0	221986.0	NaN	NaN	NaN	NaN	NaN	40
2	32309159	10458.0	1015081.0	256380.0	NaN	NaN	NaN	NaN	NaN	40
3	32305098	10461.0	1031740.0	243899.0	NaN	NaN	NaN	NaN	NaN	40
4	32306529	11373.0	1019123.0	206375.0	NaN	NaN	NaN	NaN	NaN	40

In [21]:
```
# Find null values in 'num' dataset
```

In [22]:
```
# Analyse columns with null values
num.isna().any()
```

Out[22]:
```
Unique Key                      False
Incident Zip                     True
X Coordinate (State Plane)       True
Y Coordinate (State Plane)       True
School or Citywide Complaint     True
Vehicle Type                     True
Taxi Company Borough             True
Taxi Pick Up Location            True
Garage Lot Name                  True
Latitude                         True
Longitude                        True
dtype: bool
```

In [23]:
```
# Analyse the count of null values in all columns
num.isna().sum()
```

Out[23]:
```
Unique Key                         0
Incident Zip                    2998
X Coordinate (State Plane)      4030
Y Coordinate (State Plane)      4030
School or Citywide Complaint  364558
Vehicle Type                  364558
Taxi Company Borough          364558
Taxi Pick Up Location         364558
Garage Lot Name               364558
Latitude                        4030
Longitude                       4030
dtype: int64
```

In [24]:
```
# Find the columns those contain null values only
num.isna().any()[num.isna().any()==True]
```

Out[24]:
```
Incident Zip                    True
X Coordinate (State Plane)      True
Y Coordinate (State Plane)      True
School or Citywide Complaint    True
Vehicle Type                    True
Taxi Company Borough            True
Taxi Pick Up Location           True
Garage Lot Name                 True
Latitude                        True
Longitude                       True
dtype: bool
```

In [25]:
```
# Find the count of null values in the columns those contain null values only
num.isna().sum()[num.isna().sum()!=0]
```

Out[25]:
```
Incident Zip                      2998
X Coordinate (State Plane)        4030
Y Coordinate (State Plane)        4030
School or Citywide Complaint    364558
Vehicle Type                    364558
Taxi Company Borough            364558
Taxi Pick Up Location           364558
Garage Lot Name                 364558
Latitude                          4030
Longitude                         4030
dtype: int64
```

In [26]:
```
# Analyse the date column and remove the entries if it has incorrect timeline
```

In [27]:
```
cust_service.columns
```

Out[27]:
```
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')
```

In [28]:
```
# Identifying the columns having date
date_cols = [col for col in cust_service.columns if 'Date' in col]
```

In [29]:
```
date_cols
```

Out[29]:
```
['Created Date', 'Closed Date', 'Due Date', 'Resolution Action Updated Date']
```

In [30]:
```
cust_service['Created Date'].isna().any()
```

Out[30]:
```
False
```

In [31]:
```
cust_service['Created Date'].isna().sum()
```

Out[31]:
```
0
```

In [32]:
```
cust_service['Created Date'].head()
```

Out[32]:
```
0    12/31/2015 11:59:45 PM
1    12/31/2015 11:59:44 PM
2    12/31/2015 11:59:29 PM
3    12/31/2015 11:57:46 PM
4    12/31/2015 11:56:58 PM
Name: Created Date, dtype: object
```

In [33]:
```
cust_service.dropna(subset=['Created Date'], inplace = True)
```

In [34]:
```
cust_service['Created Date']
```

Out[34]:
```
0         12/31/2015 11:59:45 PM
1         12/31/2015 11:59:44 PM
2         12/31/2015 11:59:29 PM
3         12/31/2015 11:57:46 PM
4         12/31/2015 11:56:58 PM
                   ...
364553    01/01/2015 12:04:44 AM
364554    01/01/2015 12:04:28 AM
364555    01/01/2015 12:01:30 AM
364556    01/01/2015 12:01:29 AM
364557    01/01/2015 12:00:50 AM
Name: Created Date, Length: 364558, dtype: object
```

In [35]:
```
# Identify null values in Closed Date Column
cust_service['Closed Date'].isna().any()
```

Out[35]:
```
True
```

In [36]:
```
# Number of Null values in Closed Date column
cust_service['Closed Date'].isna().sum()
```

Out[36]:
```
2381
```

In [37]:
```
# Remove rows with null values from Closed Date Column
cust_service.dropna(subset=['Closed Date'], inplace = True)
```

In [38]:
```
# Check updated Closed Date Column
cust_service['Closed Date'].isna().any()
```

Out[38]:
```
False
```

In [39]:
```
cust_service['Closed Date'].isna().sum()
```

Out[39]:
```
0
```

In [40]:
```
date_cols
```

Out[40]:
```
['Created Date', 'Closed Date', 'Due Date', 'Resolution Action Updated Date']
```

In [41]:
```
# Identify null values in Due Date column
cust_service['Due Date'].isna().any()
```

Out[41]: True

In [42]:
```
# Find the number of null values in Due Date column
cust_service['Due Date'].isna().sum()
```

Out[42]: 1

In [43]:
```
# Remove null values from Due Date column
cust_service.dropna(subset=['Due Date'], inplace = True)
```

In [44]:
```
# Check the updated Due Date column
cust_service['Due Date'].isna().any()
```

Out[44]: False

In [45]:
```
# Identify null values in Resolution Action Updated Date column
cust_service['Resolution Action Updated Date'].isna().any()
```

Out[45]: True

In [46]:
```
# Find the number of null values in Resolution Action Updated Date column
cust_service['Resolution Action Updated Date'].isna().sum()
```

Out[46]: 38

In [47]:
```
# Remove null values in Resolution Action Updated Date column
cust_service.dropna(subset='Resolution Action Updated Date', inplace = True)
```

In [48]:
```
# Check the updated Resolution Action Updated Date column
cust_service['Resolution Action Updated Date'].isna().any()
```

Out[48]: False

In [49]:
```
cust_service.columns
```

Out[49]:
```
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')
```

In [50]:
```
cust_service['School or Citywide Complaint'].info()
```

```
<class 'pandas.core.series.Series'>
Int64Index: 362138 entries, 0 to 364557
Series name: School or Citywide Complaint
Non-Null Count  Dtype
--------------  -----
0 non-null      float64
dtypes: float64(1)
memory usage: 5.5 MB
```

In [51]:
```
cust_service['Complaint Type'].info()
```

```
<class 'pandas.core.series.Series'>
Int64Index: 362138 entries, 0 to 364557
Series name: Complaint Type
Non-Null Count   Dtype
--------------   -----
362138 non-null  object
dtypes: object(1)
memory usage: 5.5+ MB
```

In [52]:
```
## Frequency Plot for city-wise complaints
```

In [53]:
```
# Convert object values of Complaint Type in integer
```

In [54]:
```
# Index values of Dataset
cust_service.index.values
```

Out[54]:
```
array([     0,      1,      2, ..., 364555, 364556, 364557], dtype=int64)
```

In [55]:
```
# Column values of Dataset
cust_service.columns.values
```

Out[55]:
```
array(['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)
```

In [56]:
```
cust_service.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 362138 entries, 0 to 364557
Data columns (total 53 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   Unique Key                      362138 non-null  int64  
 1   Created Date                    362138 non-null  object 
 2   Closed Date                     362138 non-null  object 
 3   Agency                          362138 non-null  object 
 4   Agency Name                     362138 non-null  object 
 5   Complaint Type                  362138 non-null  object 
 6   Descriptor                      355644 non-null  object 
 7   Location Type                   362045 non-null  object 
 8   Incident Zip                    361463 non-null  float64
 9   Incident Address                310459 non-null  object 
 10  Street Name                     310459 non-null  object 
 11  Cross Street 1                  306814 non-null  object 
 12  Cross Street 2                  306681 non-null  object 
 13  Intersection Street 1           50621 non-null   object 
 14  Intersection Street 2           50497 non-null   object 
 15  Address Type                    361209 non-null  object 
 16  City                            361464 non-null  object 
 17  Landmark                        375 non-null     object 
 18  Facility Type                   362121 non-null  object 
 19  Status                          362138 non-null  object 
 20  Due Date                        362138 non-null  object 
 21  Resolution Description          362138 non-null  object 
 22  Resolution Action Updated Date  362138 non-null  object 
 23  Community Board                 362138 non-null  object 
 24  Borough                         362138 non-null  object 
 25  X Coordinate (State Plane)      360431 non-null  float64
 26  Y Coordinate (State Plane)      360431 non-null  float64
 27  Park Facility Name              362138 non-null  object 
 28  Park Borough                    362138 non-null  object 
 29  School Name                     362138 non-null  object 
 30  School Number                   362138 non-null  object 
 31  School Region                   362137 non-null  object 
 32  School Code                     362137 non-null  object 
 33  School Phone Number             362138 non-null  object 
 34  School Address                  362138 non-null  object 
 35  School City                     362138 non-null  object 
 36  School State                    362138 non-null  object 
 37  School Zip                      362137 non-null  object 
 38  School Not Found                362138 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             297 non-null     object 
 44  Bridge Highway Direction        297 non-null     object 
 45  Road Ramp                       262 non-null     object 
 46  Bridge Highway Segment          262 non-null     object 
 47  Garage Lot Name                 0 non-null       float64
 48  Ferry Direction                 0 non-null       object 
 49  Ferry Terminal Name             0 non-null       object 
```

```
 50  Latitude                        360431 non-null  float64
 51  Longitude                       360431 non-null  float64
 52  Location                        360431 non-null  object
dtypes: float64(10), int64(1), object(42)
memory usage: 149.2+ MB
```

In [57]:
```
# Data Slicing for getting data of Complaint Type & City only
cust_service.loc[:,['Complaint Type','City']]
```

Out[57]:

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

362138 rows × 2 columns

In [58]:
```
## Find Major types of Complaints
```

In [59]:
```
# Setting a variable for getting data of Complaint Type
major=cust_service.loc[:,['Complaint Type']]
```

In [60]:
```
major
```

Out[60]:

	Complaint Type
0	Noise - Street/Sidewalk
1	Blocked Driveway
2	Blocked Driveway
3	Illegal Parking
4	Illegal Parking
...	...
364553	Illegal Parking
364554	Noise - Vehicle
364555	Noise - Street/Sidewalk
364556	Blocked Driveway
364557	Blocked Driveway

362138 rows × 1 columns

In [61]:
```
# Finding number of unique values in Major
major.nunique()
```

Out[61]:
```
Complaint Type    23
dtype: int64
```

In [62]:
```
top = major.value_counts()
```

In [63]:
```
top
```

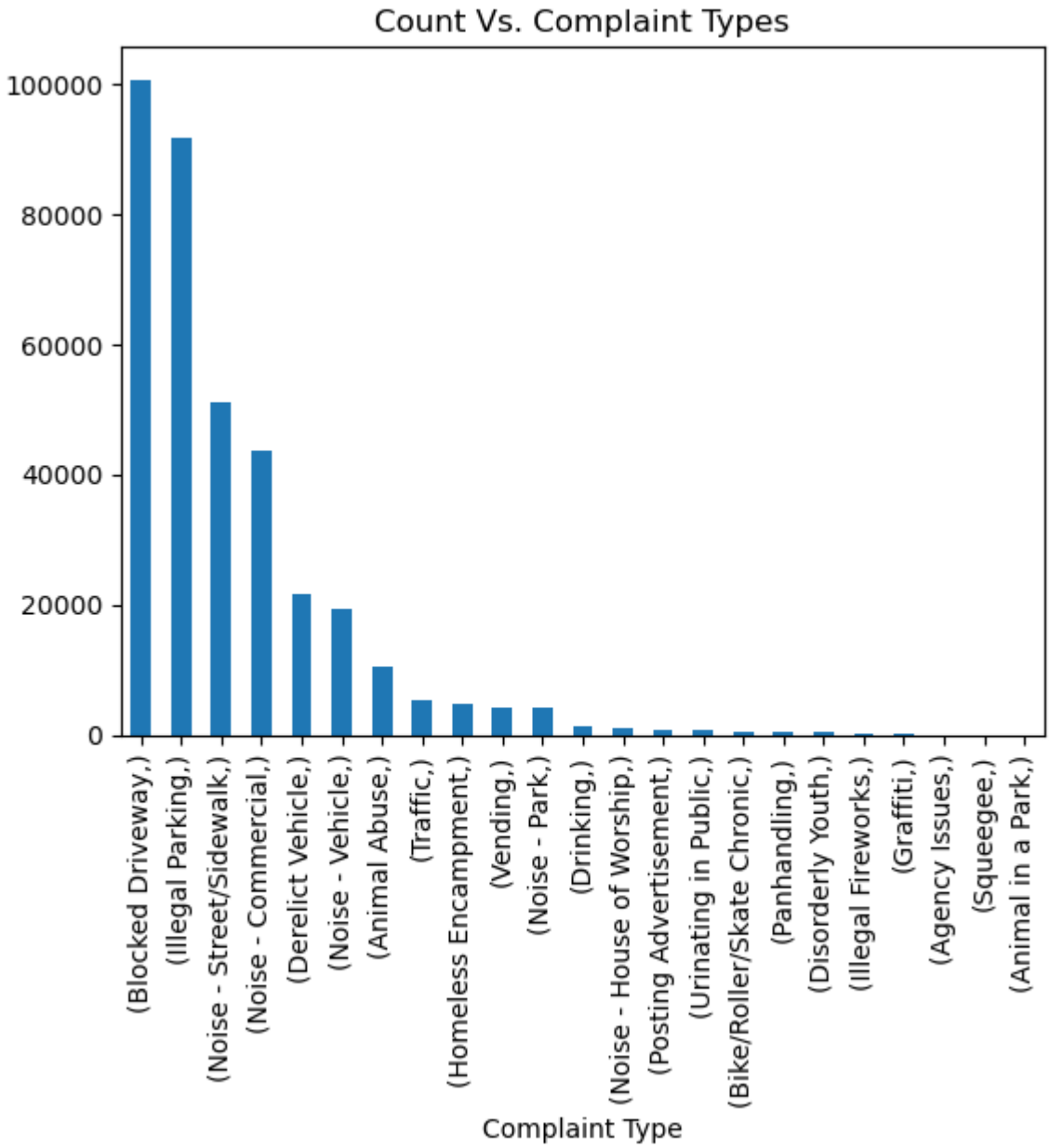
Out[63]:
```
Complaint Type
Blocked Driveway            100618
Illegal Parking              91705
Noise - Street/Sidewalk      51131
Noise - Commercial           43749
Derelict Vehicle             21516
Noise - Vehicle              19300
Animal Abuse                 10530
Traffic                       5193
Homeless Encampment           4877
Vending                       4183
Noise - Park                  4088
Drinking                      1404
Noise - House of Worship      1068
Posting Advertisement          678
Urinating in Public            641
Bike/Roller/Skate Chronic      475
Panhandling                    325
Disorderly Youth               315
Illegal Fireworks              172
Graffiti                       157
Agency Issues                    8
Squeegee                         4
Animal in a Park                 1
dtype: int64
```

In [64]:
```
# Finding Top 10 Complaints
top.head(10)
```

Out[64]:
```
Complaint Type
Blocked Driveway           100618
Illegal Parking             91705
Noise - Street/Sidewalk     51131
Noise - Commercial          43749
Derelict Vehicle            21516
Noise - Vehicle             19300
Animal Abuse                10530
Traffic                      5193
Homeless Encampment          4877
Vending                      4183
dtype: int64
```

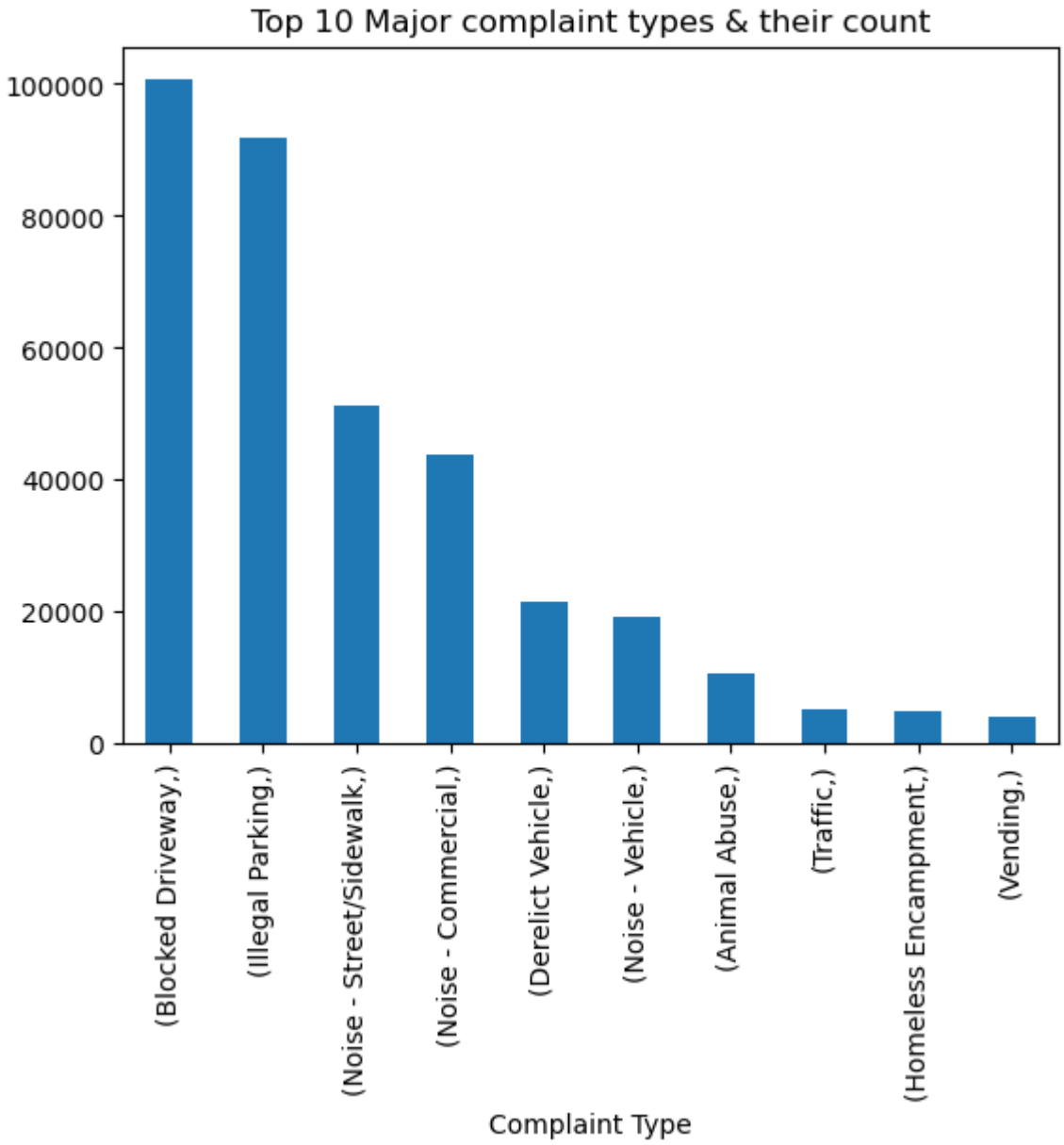
In [65]:
```
# Bar Plot for Major Complaint Types & their counts
top.plot(kind='bar',title='Count Vs. Complaint Types')
```

Out[65]:
```
<AxesSubplot:title={'center':'Count Vs. Complaint Types'}, xlabel='Complaint Ty
pe'>
```



In [66]:
```
# Bar plot for top 10 types of complaints
top.head(10).plot(kind='bar',title='Top 10 Major complaint types & their count')
```

Out[66]:
```
<AxesSubplot:title={'center':'Top 10 Major complaint types & their count'}, xla
bel='Complaint Type'>
```



In [67]:
```
## Frequency Plot for city wise complaints
```

In [68]:
```
# Set ComplaintTypeCity variable for City-wise complaints data
ComplaintTypeCity = pd.DataFrame({'Count':cust_service.groupby(['Complaint Type',
```

In [69]:
```
ComplaintTypeCity
```

Out[69]:

	Complaint Type	City	Count
0	Animal Abuse	ARVERNE	46
1	Animal Abuse	ASTORIA	170
2	Animal Abuse	BAYSIDE	53
3	Animal Abuse	BELLEROSE	15
4	Animal Abuse	BREEZY POINT	2
...	...	...	...
772	Vending	STATEN ISLAND	25
773	Vending	SUNNYSIDE	15
774	Vending	WHITESTONE	1
775	Vending	WOODHAVEN	6
776	Vending	WOODSIDE	15

777 rows × 3 columns

In [70]:
```
cust_service.groupby(['Borough','Complaint Type','Descriptor']).size()
```

Out[70]:
```
Borough      Complaint Type           Descriptor
BRONX        Animal Abuse             Chained                      166
                                      In Car                        41
                                      Neglected                    951
                                      No Shelter                   165
                                      Other (complaint details)    412
                                                                   ...
Unspecified  Noise - Street/Sidewalk  Loud Talking                   26
             Noise - Vehicle          Car/Truck Horn                 2
                                      Car/Truck Music                5
                                      Engine Idling                  2
             Vending                  In Prohibited Area             1
Length: 276, dtype: int64
```

In [ ]:
```

```

In [71]:
```
# Create a sorted Data for Complaint Type
majorcomplaints = cust_service.dropna(subset=['Complaint Type'])
majorcomplaints = cust_service.groupby('Complaint Type')
sortedcomplainttype = majorcomplaints.size().sort_values(ascending = False)
sortedcomplainttype = sortedcomplainttype.to_frame('count').reset_index()
sortedcomplainttype
```

Out[71]:

	Complaint Type	count
0	Blocked Driveway	100618
1	Illegal Parking	91705
2	Noise - Street/Sidewalk	51131
3	Noise - Commercial	43749
4	Derelict Vehicle	21516
5	Noise - Vehicle	19300
6	Animal Abuse	10530
7	Traffic	5193
8	Homeless Encampment	4877
9	Vending	4183
10	Noise - Park	4088
11	Drinking	1404
12	Noise - House of Worship	1068
13	Posting Advertisement	678
14	Urinating in Public	641
15	Bike/Roller/Skate Chronic	475
16	Panhandling	325
17	Disorderly Youth	315
18	Illegal Fireworks	172
19	Graffiti	157
20	Agency Issues	8
21	Squeegee	4
22	Animal in a Park	1

In [72]:
```
# Create a sorted Data for City
citydata = cust_service.dropna(subset=['City'])
citydata = cust_service.groupby('City')
sortedcitydata = citydata.size().sort_values(ascending = False)
sortedcitydata = sortedcitydata.to_frame('count').reset_index()
sortedcitydata
```

Out[72]:

	City	count
0	BROOKLYN	118841
1	NEW YORK	77281
2	BRONX	49163
3	STATEN ISLAND	15332
4	JAMAICA	8921
5	ASTORIA	7991
6	FLUSHING	7486
7	RIDGEWOOD	6391
8	CORONA	5382
9	WOODSIDE	4356
10	EAST ELMHURST	3557
11	OZONE PARK	3446
12	ELMHURST	3437
13	SOUTH RICHMOND HILL	3431
14	MASPETH	3117
15	WOODHAVEN	3101
16	LONG ISLAND CITY	3026
17	SOUTH OZONE PARK	2668
18	FRESH MEADOWS	2451
19	RICHMOND HILL	2333
20	MIDDLE VILLAGE	2291
21	QUEENS VILLAGE	2251
22	FOREST HILLS	2122
23	JACKSON HEIGHTS	2105
24	REGO PARK	1807
25	BAYSIDE	1550
26	COLLEGE POINT	1544
27	FAR ROCKAWAY	1397
28	WHITESTONE	1369
29	HOLLIS	1231
30	HOWARD BEACH	1144

	City	count
31	SPRINGFIELD GARDENS	1093
32	ROSEDALE	1091
33	SAINT ALBANS	1047
34	KEW GARDENS	1008
35	SUNNYSIDE	944
36	Astoria	905
37	ROCKAWAY PARK	831
38	OAKLAND GARDENS	717
39	LITTLE NECK	712
40	CAMBRIA HEIGHTS	617
41	BELLEROSE	487
42	GLEN OAKS	361
43	ARVERNE	259
44	FLORAL PARK	196
45	Long Island City	170
46	Woodside	166
47	NEW HYDE PARK	129
48	CENTRAL PARK	110
49	QUEENS	37
50	BREEZY POINT	31
51	East Elmhurst	30
52	Howard Beach	1

In [73]:
```
# Create a sorted Data for Complaint Type
citydata = cust_service.dropna(subset=['City'])
citydata = cust_service.groupby('City')
sortedcitydata = citydata.size().sort_values(ascending = False)
sortedcitydata = sortedcitydata.to_frame('count').reset_index()
sortedcitydata
```

Out[73]:

	City	count
0	BROOKLYN	118841
1	NEW YORK	77281
2	BRONX	49163
3	STATEN ISLAND	15332
4	JAMAICA	8921
5	ASTORIA	7991
6	FLUSHING	7486
7	RIDGEWOOD	6391
8	CORONA	5382
9	WOODSIDE	4356
10	EAST ELMHURST	3557
11	OZONE PARK	3446
12	ELMHURST	3437
13	SOUTH RICHMOND HILL	3431
14	MASPETH	3117
15	WOODHAVEN	3101
16	LONG ISLAND CITY	3026
17	SOUTH OZONE PARK	2668
18	FRESH MEADOWS	2451
19	RICHMOND HILL	2333
20	MIDDLE VILLAGE	2291
21	QUEENS VILLAGE	2251
22	FOREST HILLS	2122
23	JACKSON HEIGHTS	2105
24	REGO PARK	1807
25	BAYSIDE	1550
26	COLLEGE POINT	1544
27	FAR ROCKAWAY	1397
28	WHITESTONE	1369
29	HOLLIS	1231
30	HOWARD BEACH	1144

	City	count
31	SPRINGFIELD GARDENS	1093
32	ROSEDALE	1091
33	SAINT ALBANS	1047
34	KEW GARDENS	1008
35	SUNNYSIDE	944
36	Astoria	905
37	ROCKAWAY PARK	831
38	OAKLAND GARDENS	717
39	LITTLE NECK	712
40	CAMBRIA HEIGHTS	617
41	BELLEROSE	487
42	GLEN OAKS	361
43	ARVERNE	259
44	FLORAL PARK	196
45	Long Island City	170
46	Woodside	166
47	NEW HYDE PARK	129
48	CENTRAL PARK	110
49	QUEENS	37
50	BREEZY POINT	31
51	East Elmhurst	30
52	Howard Beach	1

In [74]:
```
sortedcitydata
```

Out[74]:

	City	count
0	BROOKLYN	118841
1	NEW YORK	77281
2	BRONX	49163
3	STATEN ISLAND	15332
4	JAMAICA	8921
5	ASTORIA	7991
6	FLUSHING	7486
7	RIDGEWOOD	6391
8	CORONA	5382
9	WOODSIDE	4356
10	EAST ELMHURST	3557
11	OZONE PARK	3446
12	ELMHURST	3437
13	SOUTH RICHMOND HILL	3431
14	MASPETH	3117
15	WOODHAVEN	3101
16	LONG ISLAND CITY	3026
17	SOUTH OZONE PARK	2668
18	FRESH MEADOWS	2451
19	RICHMOND HILL	2333
20	MIDDLE VILLAGE	2291
21	QUEENS VILLAGE	2251
22	FOREST HILLS	2122
23	JACKSON HEIGHTS	2105
24	REGO PARK	1807
25	BAYSIDE	1550
26	COLLEGE POINT	1544
27	FAR ROCKAWAY	1397
28	WHITESTONE	1369
29	HOLLIS	1231
30	HOWARD BEACH	1144
31	SPRINGFIELD GARDENS	1093
32	ROSEDALE	1091
33	SAINT ALBANS	1047
34	KEW GARDENS	1008

	City	count
35	SUNNYSIDE	944
36	Astoria	905
37	ROCKAWAY PARK	831
38	OAKLAND GARDENS	717
39	LITTLE NECK	712
40	CAMBRIA HEIGHTS	617
41	BELLEROSE	487
42	GLEN OAKS	361
43	ARVERNE	259
44	FLORAL PARK	196
45	Long Island City	170
46	Woodside	166
47	NEW HYDE PARK	129
48	CENTRAL PARK	110
49	QUEENS	37
50	BREEZY POINT	31
51	East Elmhurst	30
52	Howard Beach	1

In [75]:
```
citydata.head()
```

Out[75]:

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Lo
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	Stre
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	Stre
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	Stre
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	Stre
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	Stre
...	...	...	...	...	...	...	...	
112319	31537126	09/15/2015 11:16:42 PM	09/16/2015 01:08:56 AM	NYPD	New York City Police Department	Noise - Commercial	Loud Music/Party	Club/Ba
120974	31489202	09/08/2015 07:00:48 AM	09/08/2015 10:24:19 AM	NYPD	New York City Police Department	Illegal Parking	Blocked Hydrant	Stre
150271	31306636	08/13/2015 06:44:34 PM	08/14/2015 04:18:07 AM	NYPD	New York City Police Department	Illegal Parking	Posted Parking Sign Violation	Stre
198051	30979935	07/01/2015 11:55:26 PM	07/02/2015 01:19:24 AM	NYPD	New York City Police Department	Illegal Parking	Posted Parking Sign Violation	Stre
282522	30431056	04/18/2015 08:51:45 PM	04/19/2015 12:53:30 AM	NYPD	New York City Police Department	Blocked Driveway	No Access	Stre

261 rows × 53 columns

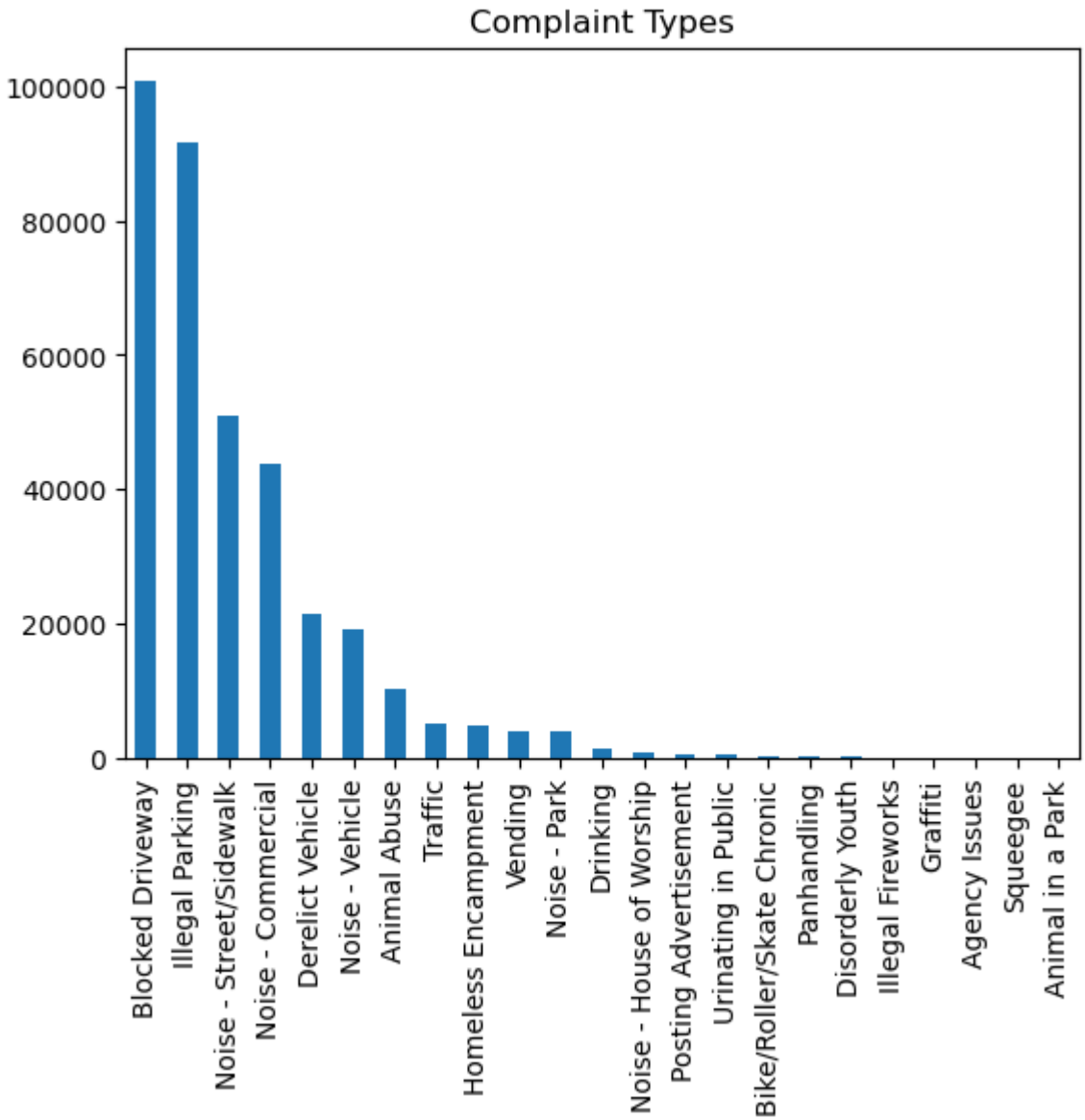
In [76]:
```
sortedcitydata['City']
```

Out[76]:
```
0                BROOKLYN
1                NEW YORK
2                   BRONX
3           STATEN ISLAND
4                 JAMAICA
5                 ASTORIA
6                FLUSHING
7               RIDGEWOOD
8                  CORONA
9                WOODSIDE
10          EAST ELMHURST
11             OZONE PARK
12               ELMHURST
13    SOUTH RICHMOND HILL
14                MASPETH
15              WOODHAVEN
16       LONG ISLAND CITY
17       SOUTH OZONE PARK
18          FRESH MEADOWS
19          RICHMOND HILL
20         MIDDLE VILLAGE
21         QUEENS VILLAGE
22           FOREST HILLS
23        JACKSON HEIGHTS
24              REGO PARK
25                BAYSIDE
26          COLLEGE POINT
27           FAR ROCKAWAY
28             WHITESTONE
29                 HOLLIS
30           HOWARD BEACH
31    SPRINGFIELD GARDENS
32              ROSEDALE
33           SAINT ALBANS
34            KEW GARDENS
35              SUNNYSIDE
36                Astoria
37          ROCKAWAY PARK
38        OAKLAND GARDENS
39             LITTLE NECK
40        CAMBRIA HEIGHTS
41              BELLEROSE
42              GLEN OAKS
43                ARVERNE
44             FLORAL PARK
45       Long Island City
46                Woodside
47          NEW HYDE PARK
48           CENTRAL PARK
49                 QUEENS
50           BREEZY POINT
51          East Elmhurst
52           Howard Beach
Name: City, dtype: object
```

In [ ]:

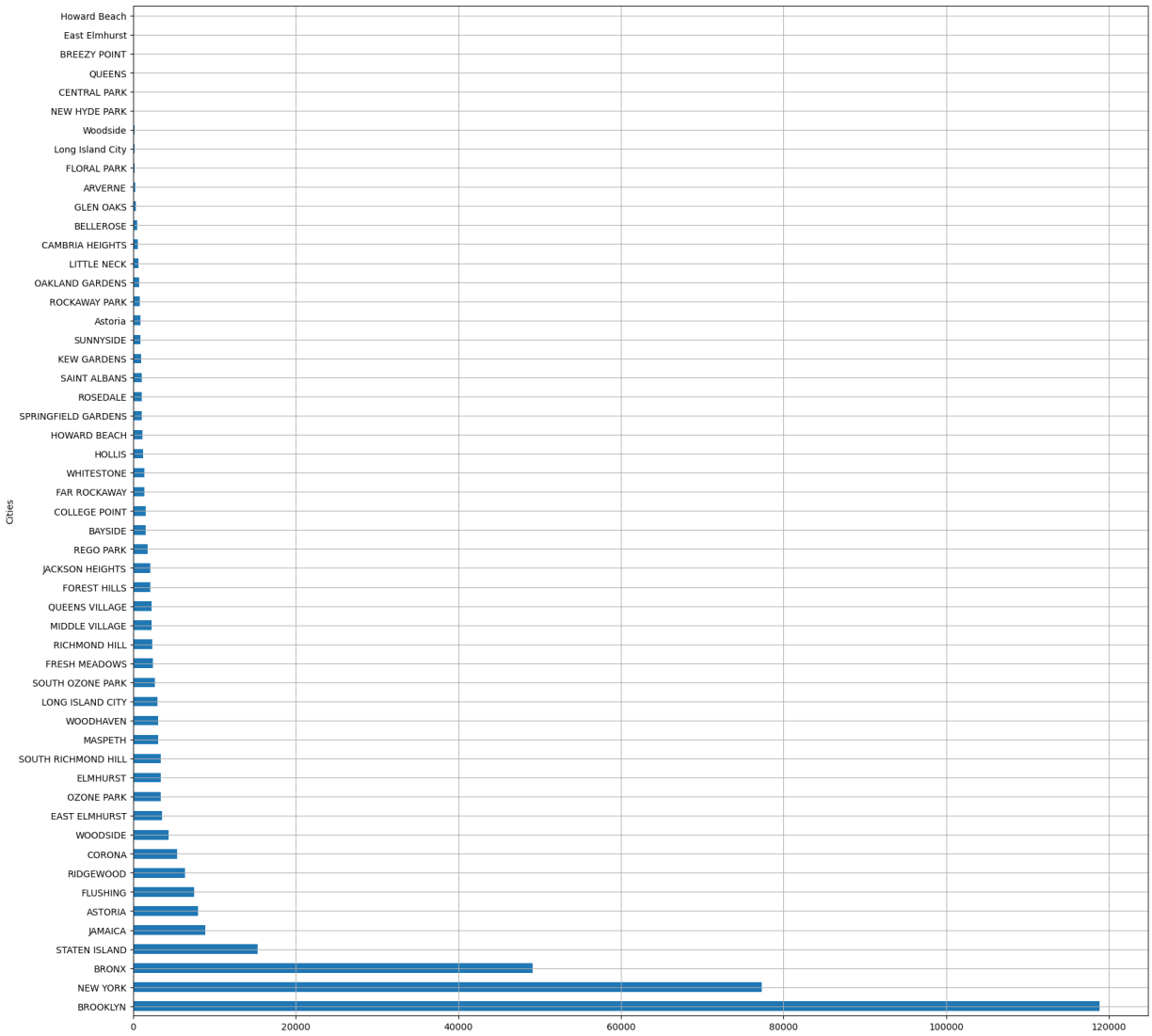
In [77]:
```
# Plot data of Complaint Type
cust_service['Complaint Type'].value_counts().plot(kind='bar',title='Complaint Ty
```

Out[77]: <AxesSubplot:title={'center':'Complaint Types'}>



In [79]:
```
cust_service['City'].value_counts().plot(kind='barh',grid=True,figsize=(20,20),yl
```

Out[79]: <AxesSubplot:ylabel='Cities'>

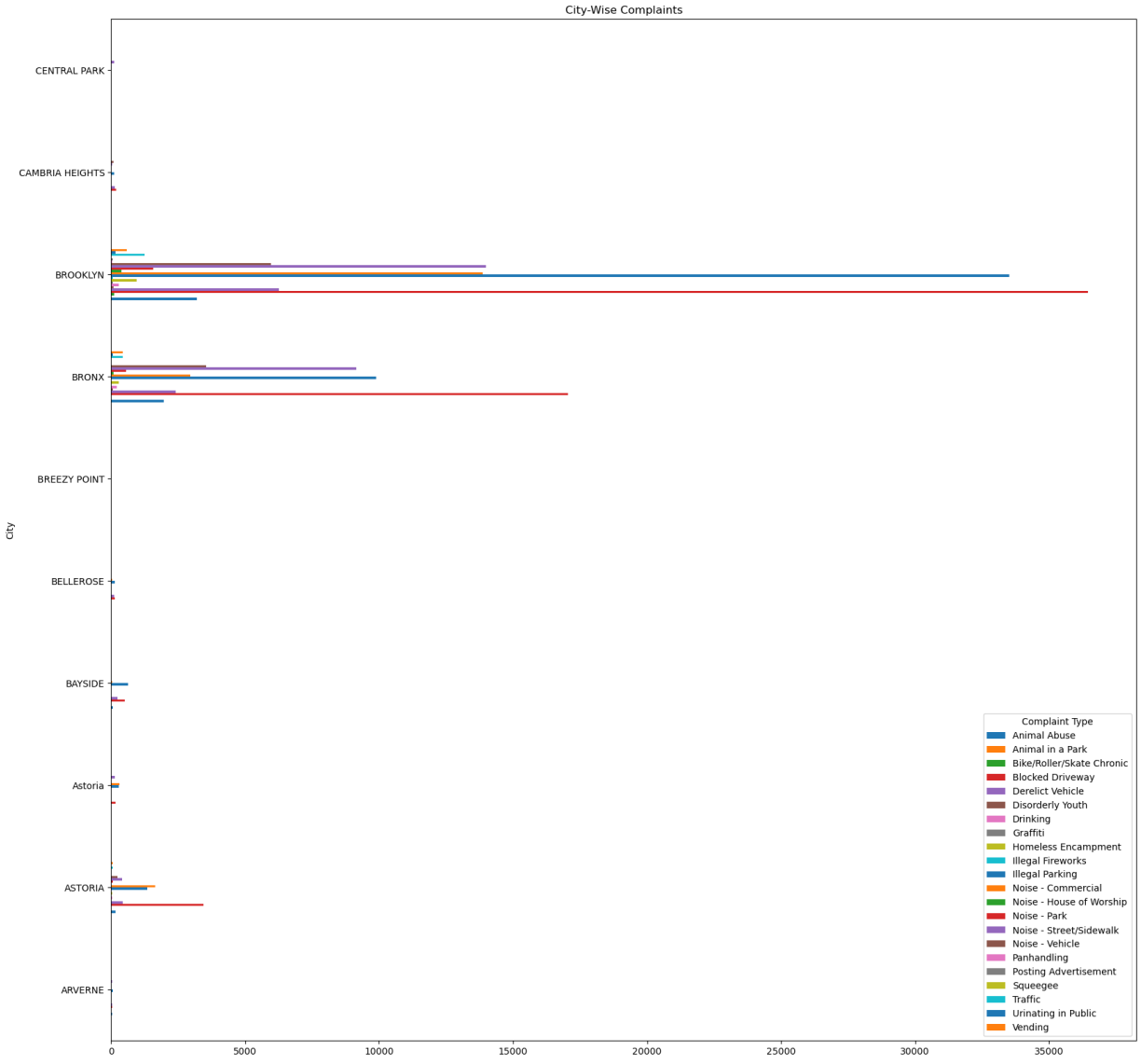


In [80]:
```
# Frequency plot for City-Wise Complaints
```

In [81]:
```
city_type=pd.crosstab(cust_service['City'],cust_service['Complaint Type'])
```

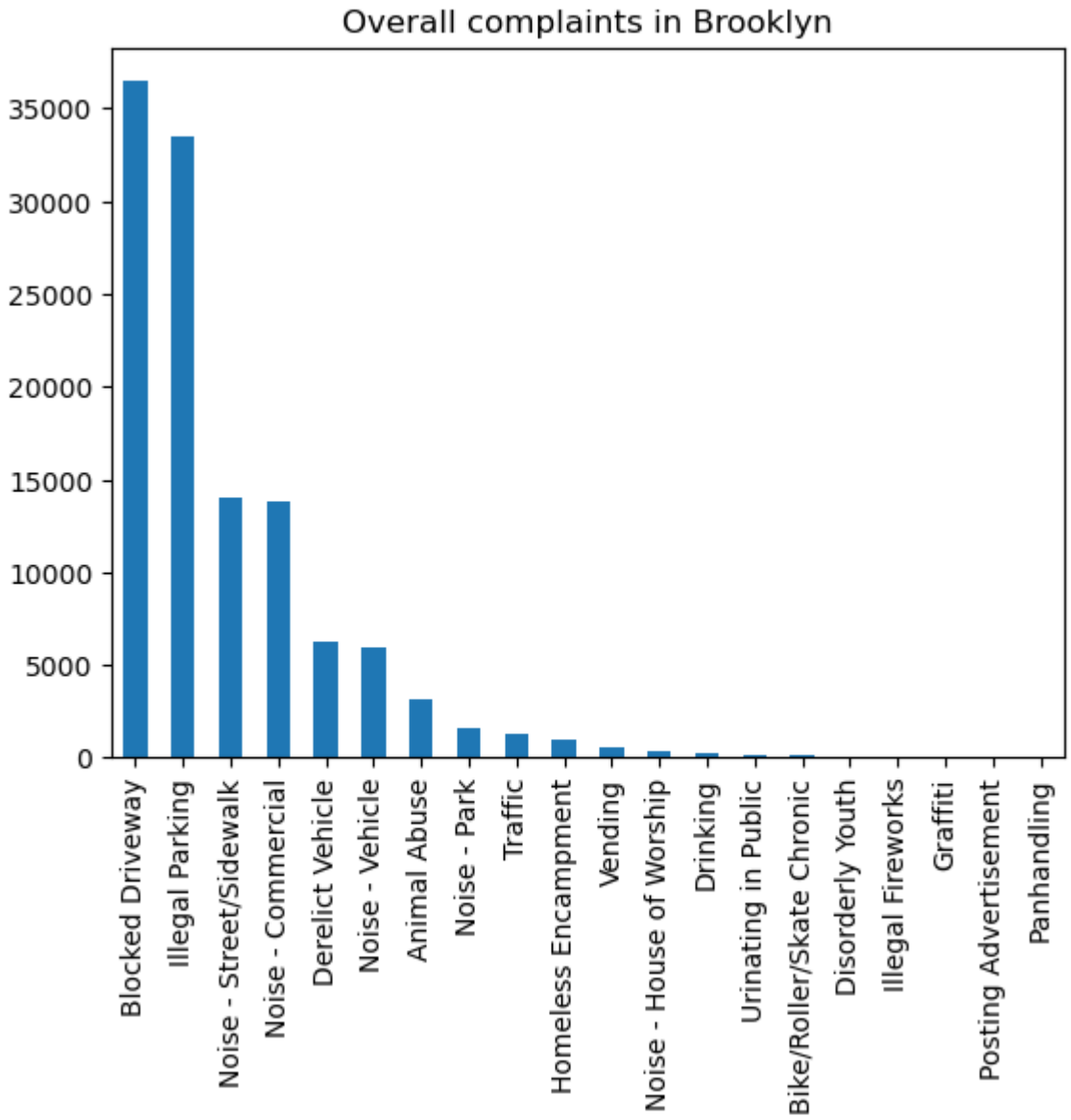
In [82]:
```
# Frequency Plot for City-Wise Complaints
city_type.head(10).plot(kind='barh',figsize=(20,20),xlabel='City',ylabel='Complai
```

Out[82]:
```
<AxesSubplot:title={'center':'City-Wise Complaints'}, ylabel='City'>
```



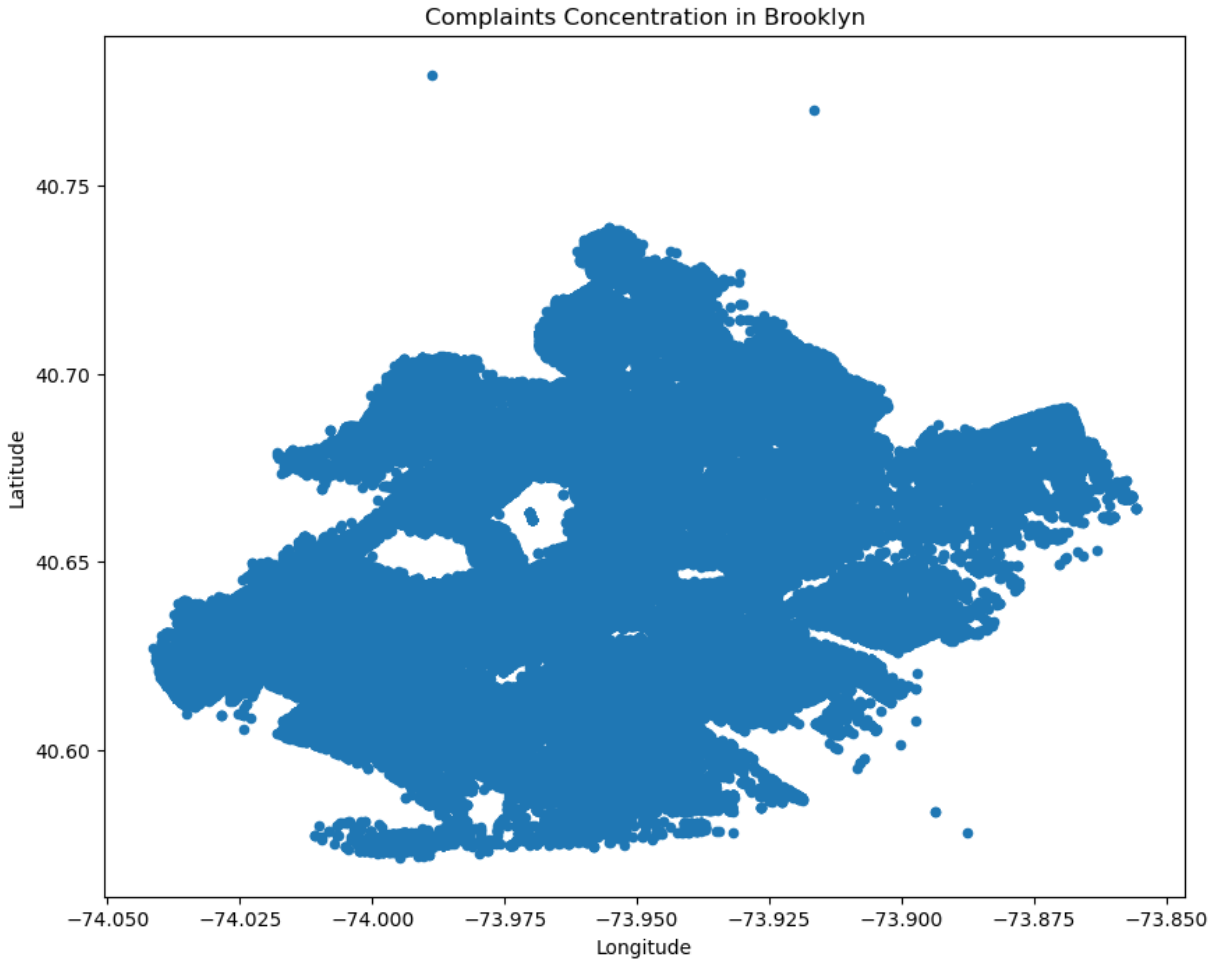
In [83]:
```
# Visualize the major types of complaints in each city
cust_service_bkl = cust_service.loc[cust_service['City']=='BROOKLYN']
cust_service_bkl['Complaint Type'].value_counts().plot(kind='bar',title='Overall
```

Out[83]: <AxesSubplot:title={'center':'Overall complaints in Brooklyn'}>



In [82]:
```
## Draw scatter and hexbin plots for complaint concentration across Brooklyn
# Scatter Plot
cust_service_bkl[['Longitude','Latitude','Complaint Type']].plot(kind='scatter',x
```

Out[82]:
```
<AxesSubplot:title={'center':'Complaints Concentration in Brooklyn'}, xlabel='L
ongitude', ylabel='Latitude'>
```



In [135]:
```
cust_service_bkl.head()
```

Out[135]:

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Ty
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/Sidew
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/Sidew
13	32305074	12/31/2015 11:47:58 PM	01/01/2016 08:18:47 AM	NYPD	New York City Police Department	Illegal Parking	Posted Parking Sign Violation	Street/Sidew
17	32310273	12/31/2015 11:44:52 PM	01/01/2016 12:36:10 AM	NYPD	New York City Police Department	Noise - Commercial	Loud Music/Party	Club/Bar/Restaur
18	32306617	12/31/2015 11:40:59 PM	01/01/2016 02:37:28 AM	NYPD	New York City Police Department	Noise - Commercial	Loud Music/Party	Club/Bar/Restaur

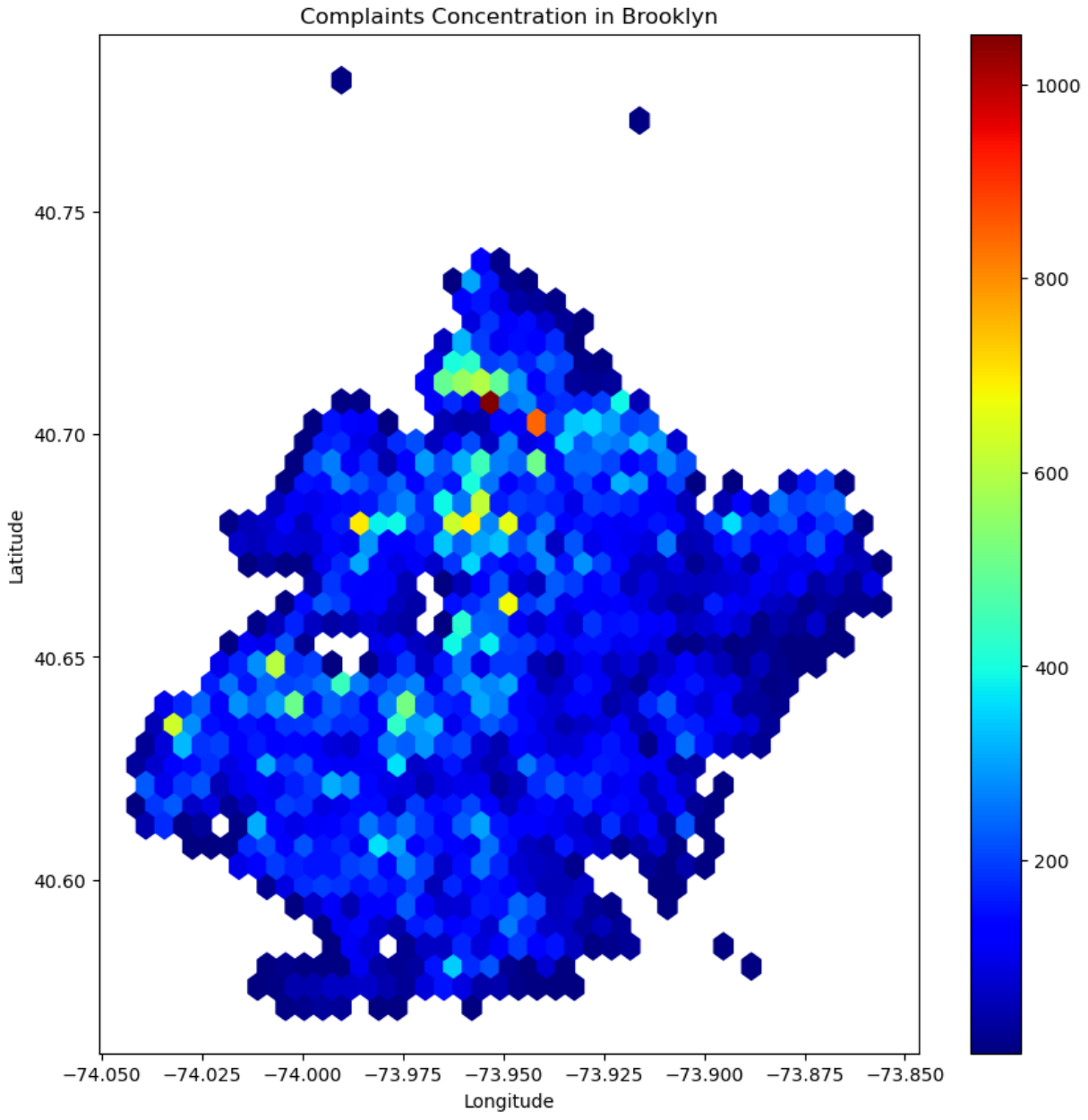
5 rows × 53 columns

In [83]:
```
sortedcitydata['City']
```

Out[83]:
```
0                BROOKLYN
1                NEW YORK
2                   BRONX
3            STATEN ISLAND
4                  JAMAICA
5                  ASTORIA
6                 FLUSHING
7                RIDGEWOOD
8                   CORONA
9                 WOODSIDE
10           EAST ELMHURST
11              OZONE PARK
12                ELMHURST
13    SOUTH RICHMOND HILL
14                 MASPETH
15               WOODHAVEN
16       LONG ISLAND CITY
17       SOUTH OZONE PARK
18          FRESH MEADOWS
19          RICHMOND HILL
20         MIDDLE VILLAGE
21         QUEENS VILLAGE
22           FOREST HILLS
23        JACKSON HEIGHTS
24                REGO PARK
25                  BAYSIDE
26           COLLEGE POINT
27            FAR ROCKAWAY
28              WHITESTONE
29                  HOLLIS
30           HOWARD BEACH
31    SPRINGFIELD GARDENS
32                ROSEDALE
33           SAINT ALBANS
34            KEW GARDENS
35              SUNNYSIDE
36                Astoria
37           ROCKAWAY PARK
38        OAKLAND GARDENS
39             LITTLE NECK
40        CAMBRIA HEIGHTS
41                BELLEROSE
42                GLEN OAKS
43                  ARVERNE
44              FLORAL PARK
45       Long Island City
46                Woodside
47           NEW HYDE PARK
48            CENTRAL PARK
49                  QUEENS
50            BREEZY POINT
51           East Elmhurst
52            Howard Beach
Name: City, dtype: object
```

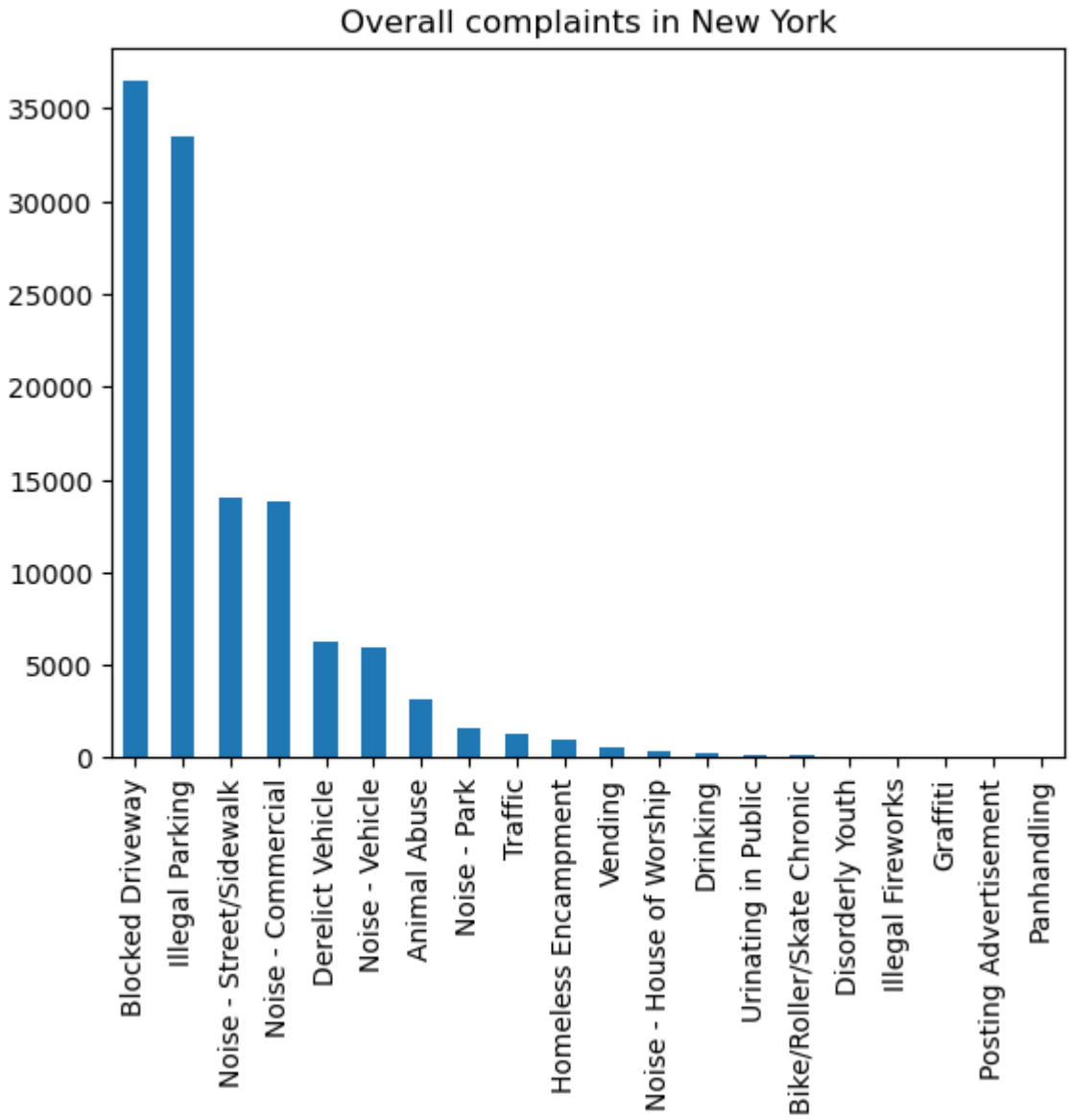
In [84]:
```
# Hexbin Plot
cust_service_bkl[['Longitude','Latitude','Complaint Type']].plot(kind='hexbin',x=
```

Out[84]:
```
<AxesSubplot:title={'center':'Complaints Concentration in Brooklyn'}, xlabel='L
ongitude', ylabel='Latitude'>
```



In [85]:
```
# Display the types of complaints in each city in a separate dataset
cust_service_ny = cust_service.loc[cust_service['City']=='NEW York']
cust_service_bkl['Complaint Type'].value_counts().plot(kind='bar',title='Overall
```

Out[85]: <AxesSubplot:title={'center':'Overall complaints in New York'}>



In [86]:
```
# Check if the average response time across various types of complaints
cust_service['Created Date']=pd.to_datetime(cust_service['Created Date'])
cust_service['Closed Date']=pd.to_datetime(cust_service['Closed Date'])
```

In [87]:
```
# Check whether data is in correct format
cust_service.loc[cust_service['Created Date']>=cust_service['Closed Date']].shape
```

Out[87]: (0, 53)

In [88]:
```
# Drop Empty Rows
cust_service = cust_service[cust_service['Closed Date'].notnull()]
cust_service.dropna(subset='Created Date',inplace=True)
cust_service.dropna(subset='Closed Date',inplace=True)
```

In [85]:
```
cust_service.loc[cust_service['Created Date']>=cust_service['Closed Date']].shape
```

Out[85]: (67408, 53)

In [90]:
```
# Calculate Resolution Time in terms of Days
cust_service['Resolution Time'] = (cust_service['Closed Date'] - cust_service['Cr
```

In [91]:
```
cust_service.head()
```

Out[91]:

	Unique Key	Created Date	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	In
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	1
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	1
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	1
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	1
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	1

5 rows × 54 columns

In [92]:
```
# Resolution according to complaint type
cust_service.groupby('Complaint Type')['Resolution Time'].mean().sort_values()
```

Out[92]:
```
Complaint Type
Agency Issues                 0.000000
Squeegee                      0.000000
Posting Advertisement         0.001475
Disorderly Youth              0.003175
Illegal Fireworks             0.005814
Noise - Commercial            0.005943
Noise - Park                  0.006360
Urinating in Public           0.007800
Noise - Vehicle               0.007876
Vending                       0.008367
Bike/Roller/Skate Chronic     0.008421
Noise - House of Worship      0.008427
Noise - Street/Sidewalk       0.008918
Drinking                      0.009972
Homeless Encampment           0.010252
Traffic                       0.010976
Blocked Driveway              0.011787
Illegal Parking               0.012965
Animal Abuse                  0.020513
Panhandling                   0.024615
Graffiti                      0.044586
Derelict Vehicle              0.065161
Animal in a Park             14.000000
Name: Resolution Time, dtype: float64
```

In [86]:
```
pip install pandoc
```

```
Collecting pandoc
  Downloading pandoc-2.3.tar.gz (33 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Collecting plumbum
  Downloading plumbum-1.8.1-py3-none-any.whl (126 kB)
     ------------------------------------ 126.7/126.7 kB 9.7 kB/s eta 0:00:00
Collecting ply
  Downloading ply-3.11-py2.py3-none-any.whl (49 kB)
     ------------------------------------ 49.6/49.6 kB 13.0 kB/s eta 0:00:00
Requirement already satisfied: pywin32 in c:\users\user\anaconda3\lib\site-pack
ages (from plumbum->pandoc) (302)
Building wheels for collected packages: pandoc
  Building wheel for pandoc (setup.py): started
  Building wheel for pandoc (setup.py): finished with status 'done'
  Created wheel for pandoc: filename=pandoc-2.3-py3-none-any.whl size=33263 sha
256=f9528e3eb4fa8d83dba96fc405cb64ce2821d2ff987c1ccb724a6dbf5abeda31
  Stored in directory: c:\users\user\appdata\local\pip\cache\wheels\69\e6\a1\1d
aa96d919c9e09a71473649b717b8da286f3f8d7719d1cfc5
Successfully built pandoc
Installing collected packages: ply, plumbum, pandoc
Successfully installed pandoc-2.3 plumbum-1.8.1 ply-3.11
Note: you may need to restart the kernel to use updated packages.
```

In [5]:
```
pip install nbconvert
```

```
Requirement already satisfied: nbconvert in c:\users\user\anaconda3\lib\site-pa
ckages (6.4.4)
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in c:\users\user\anaconda
3\lib\site-packages (from nbconvert) (0.5.13)
Requirement already satisfied: mistune<2,>=0.8.1 in c:\users\user\anaconda3\lib
\site-packages (from nbconvert) (0.8.4)
Requirement already satisfied: beautifulsoup4 in c:\users\user\anaconda3\lib\si
te-packages (from nbconvert) (4.11.1)
Requirement already satisfied: bleach in c:\users\user\anaconda3\lib\site-packa
ges (from nbconvert) (4.1.0)
Requirement already satisfied: defusedxml in c:\users\user\anaconda3\lib\site-p
ackages (from nbconvert) (0.7.1)
Requirement already satisfied: nbformat>=4.4 in c:\users\user\anaconda3\lib\sit
e-packages (from nbconvert) (5.5.0)
Requirement already satisfied: entrypoints>=0.2.2 in c:\users\user\anaconda3\li
b\site-packages (from nbconvert) (0.4)
Requirement already satisfied: traitlets>=5.0 in c:\users\user\anaconda3\lib\si
te-packages (from nbconvert) (5.1.1)
Requirement already satisfied: jupyterlab-pygments in c:\users\user\anaconda3\l
ib\site-packages (from nbconvert) (0.1.2)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\users\user\anaconda3
\lib\site-packages (from nbconvert) (1.5.0)
Requirement already satisfied: testpath in c:\users\user\anaconda3\lib\site-pac
kages (from nbconvert) (0.6.0)
Requirement already satisfied: jinja2>=2.4 in c:\users\user\anaconda3\lib\site-
packages (from nbconvert) (2.11.3)
Requirement already satisfied: jupyter-core in c:\users\user\anaconda3\lib\site
-packages (from nbconvert) (4.11.1)
Requirement already satisfied: pygments>=2.4.1 in c:\users\user\anaconda3\lib\s
ite-packages (from nbconvert) (2.11.2)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\user\anaconda3\lib
\site-packages (from jinja2>=2.4->nbconvert) (2.0.1)
Requirement already satisfied: jupyter-client>=6.1.5 in c:\users\user\anaconda3
\lib\site-packages (from nbclient<0.6.0,>=0.5.0->nbconvert) (7.3.4)
Requirement already satisfied: nest-asyncio in c:\users\user\anaconda3\lib\site
-packages (from nbclient<0.6.0,>=0.5.0->nbconvert) (1.5.5)
Requirement already satisfied: fastjsonschema in c:\users\user\anaconda3\lib\si
te-packages (from nbformat>=4.4->nbconvert) (2.16.2)
Requirement already satisfied: jsonschema>=2.6 in c:\users\user\anaconda3\lib\s
ite-packages (from nbformat>=4.4->nbconvert) (4.16.0)
Requirement already satisfied: soupsieve>1.2 in c:\users\user\anaconda3\lib\sit
e-packages (from beautifulsoup4->nbconvert) (2.3.1)
Requirement already satisfied: packaging in c:\users\user\anaconda3\lib\site-pa
ckages (from bleach->nbconvert) (21.3)
Requirement already satisfied: webencodings in c:\users\user\anaconda3\lib\site
-packages (from bleach->nbconvert) (0.5.1)
Requirement already satisfied: six>=1.9.0 in c:\users\user\anaconda3\lib\site-p
ackages (from bleach->nbconvert) (1.16.0)
Requirement already satisfied: pywin32>=1.0 in c:\users\user\anaconda3\lib\site
-packages (from jupyter-core->nbconvert) (302)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in
c:\users\user\anaconda3\lib\site-packages (from jsonschema>=2.6->nbformat>=4.4-
>nbconvert) (0.18.0)
Requirement already satisfied: attrs>=17.4.0 in c:\users\user\anaconda3\lib\sit
e-packages (from jsonschema>=2.6->nbformat>=4.4->nbconvert) (21.4.0)
```

```
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\user\anaconda
3\lib\site-packages (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconv
ert) (2.8.2)
Requirement already satisfied: pyzmq>=23.0 in c:\users\user\anaconda3\lib\site-
packages (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert) (23.2.
0)
Requirement already satisfied: tornado>=6.0 in c:\users\user\anaconda3\lib\site
-packages (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert) (6.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\user\anacon
da3\lib\site-packages (from packaging->bleach->nbconvert) (3.0.9)
Note: you may need to restart the kernel to use updated packages.
```

In [ ]: