311 data wrangling

June 22, 2023

[1]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

[2]:
```python
df=pd.read_csv("311_Service_Requests_from_2010_to_Present.csv")
```

[3]:
```python
df.head(5)
```

[3]:
```
   Unique Key            Created Date             Closed Date Agency  \
0    32310363  12/31/2015 11:59:45 PM  01/01/2016 12:55:15 AM   NYPD
1    32309934  12/31/2015 11:59:44 PM  01/01/2016 01:26:57 AM   NYPD
2    32309159  12/31/2015 11:59:29 PM  01/01/2016 04:51:03 AM   NYPD
3    32305098  12/31/2015 11:57:46 PM  01/01/2016 07:43:13 AM   NYPD
4    32306529  12/31/2015 11:56:58 PM  01/01/2016 03:24:42 AM   NYPD

                       Agency Name           Complaint Type  \
0  New York City Police Department  Noise - Street/Sidewalk
1  New York City Police Department         Blocked Driveway
2  New York City Police Department         Blocked Driveway
3  New York City Police Department           Illegal Parking
4  New York City Police Department           Illegal Parking

                    Descriptor    Location Type  Incident Zip  \
0             Loud Music/Party  Street/Sidewalk       10034.0
1                    No Access  Street/Sidewalk       11105.0
2                    No Access  Street/Sidewalk       10458.0
3  Commercial Overnight Parking  Street/Sidewalk       10461.0
4             Blocked Sidewalk  Street/Sidewalk       11373.0

        Incident Address  … Bridge Highway Name Bridge Highway Direction  \
0    71 VERMILYEA AVENUE  …                 NaN                      NaN
1        27-07 23 AVENUE  …                 NaN                      NaN
2  2897 VALENTINE AVENUE  …                 NaN                      NaN
3    2940 BAISLEY AVENUE  …                 NaN                      NaN
4          87-14 57 ROAD  …                 NaN                      NaN
```

```
  Road Ramp Bridge Highway Segment Garage Lot Name Ferry Direction  \
0       NaN                    NaN             NaN             NaN
1       NaN                    NaN             NaN             NaN
2       NaN                    NaN             NaN             NaN
3       NaN                    NaN             NaN             NaN
4       NaN                    NaN             NaN             NaN

  Ferry Terminal Name   Latitude  Longitude  \
0                 NaN  40.865682 -73.923501
1                 NaN  40.775945 -73.915094
2                 NaN  40.870325 -73.888525
3                 NaN  40.835994 -73.828379
4                 NaN  40.733060 -73.874170

                                   Location
0   (40.86568153633767, -73.92350095571744)
1  (40.775945312321085, -73.91509393898605)
2  (40.870324522111424, -73.88852464418646)
3   (40.83599404683083, -73.82837939584206)
4  (40.733059618956815, -73.87416975810375)

[5 rows x 53 columns]
```

[4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48673 entries, 0 to 48672
Data columns (total 53 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Unique Key                      48673 non-null  int64
 1   Created Date                    48673 non-null  object
 2   Closed Date                     48409 non-null  object
 3   Agency                          48673 non-null  object
 4   Agency Name                     48673 non-null  object
 5   Complaint Type                  48673 non-null  object
 6   Descriptor                      47990 non-null  object
 7   Location Type                   48673 non-null  object
 8   Incident Zip                    48367 non-null  float64
 9   Incident Address                43011 non-null  object
 10  Street Name                     43011 non-null  object
 11  Cross Street 1                  42283 non-null  object
 12  Cross Street 2                  42235 non-null  object
 13  Intersection Street 1           5602 non-null   object
 14  Intersection Street 2           5548 non-null   object
 15  Address Type                    48324 non-null  object
 16  City                            48366 non-null  object
 17  Landmark                        29 non-null     object
 18  Facility Type                   48414 non-null  object
 19  Status                          48673 non-null  object
 20  Due Date                        48673 non-null  object
 21  Resolution Description          48672 non-null  object
 22  Resolution Action Updated Date  48413 non-null  object
 23  Community Board                 48672 non-null  object
 24  Borough                         48672 non-null  object
 25  X Coordinate (State Plane)      48264 non-null  float64
 26  Y Coordinate (State Plane)      48264 non-null  float64
 27  Park Facility Name              48672 non-null  object
 28  Park Borough                    48672 non-null  object
 29  School Name                     48672 non-null  object
 30  School Number                   48672 non-null  object
 31  School Region                   48672 non-null  object
 32  School Code                     48673 non-null  object
 33  School Phone Number             48673 non-null  object
 34  School Address                  48673 non-null  object
 35  School City                     48672 non-null  object
 36  School State                    48672 non-null  object
 37  School Zip                      48672 non-null  object
 38  School Not Found                48671 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             44 non-null     object
 44  Bridge Highway Direction        44 non-null     object
 45  Road Ramp                       35 non-null     object
 46  Bridge Highway Segment          35 non-null     object
 47  Garage Lot Name                 0 non-null      float64
 48  Ferry Direction                 0 non-null      float64
 49  Ferry Terminal Name             0 non-null      float64
 50  Latitude                        48263 non-null  float64
 51  Longitude                       48263 non-null  float64
 52  Location                        48263 non-null  object
dtypes: float64(12), int64(1), object(40)
memory usage: 19.7+ MB
```

[5]:

```python
df.shape  #shape of database
```

[5]:

```
(48673, 53)
```

[6]:

```python
null_counts = df.isnull().sum()   # null values
```

[7]:

```python
null_counts
```

```
[7]: Unique Key                            0
     Created Date                          0
     Closed Date                         264
     Agency                                0
     Agency Name                           0
     Complaint Type                        0
     Descriptor                          683
     Location Type                         0
     Incident Zip                        306
     Incident Address                   5662
     Street Name                        5662
     Cross Street 1                     6390
     Cross Street 2                     6438
     Intersection Street 1             43071
     Intersection Street 2             43125
     Address Type                        349
     City                                307
     Landmark                          48644
     Facility Type                       259
     Status                                0
     Due Date                              0
     Resolution Description                1
     Resolution Action Updated Date      260
     Community Board                       1
     Borough                               1
     X Coordinate (State Plane)          409
     Y Coordinate (State Plane)          409
     Park Facility Name                    1
     Park Borough                          1
     School Name                           1
     School Number                         1
     School Region                         1
     School Code                           0
     School Phone Number                   0
     School Address                        0
     School City                           1
     School State                          1
     School Zip                            1
     School Not Found                      2
     School or Citywide Complaint      48673
     Vehicle Type                      48673
     Taxi Company Borough              48673
     Taxi Pick Up Location             48673
     Bridge Highway Name               48629
     Bridge Highway Direction          48629
     Road Ramp                         48638
     Bridge Highway Segment            48638
```

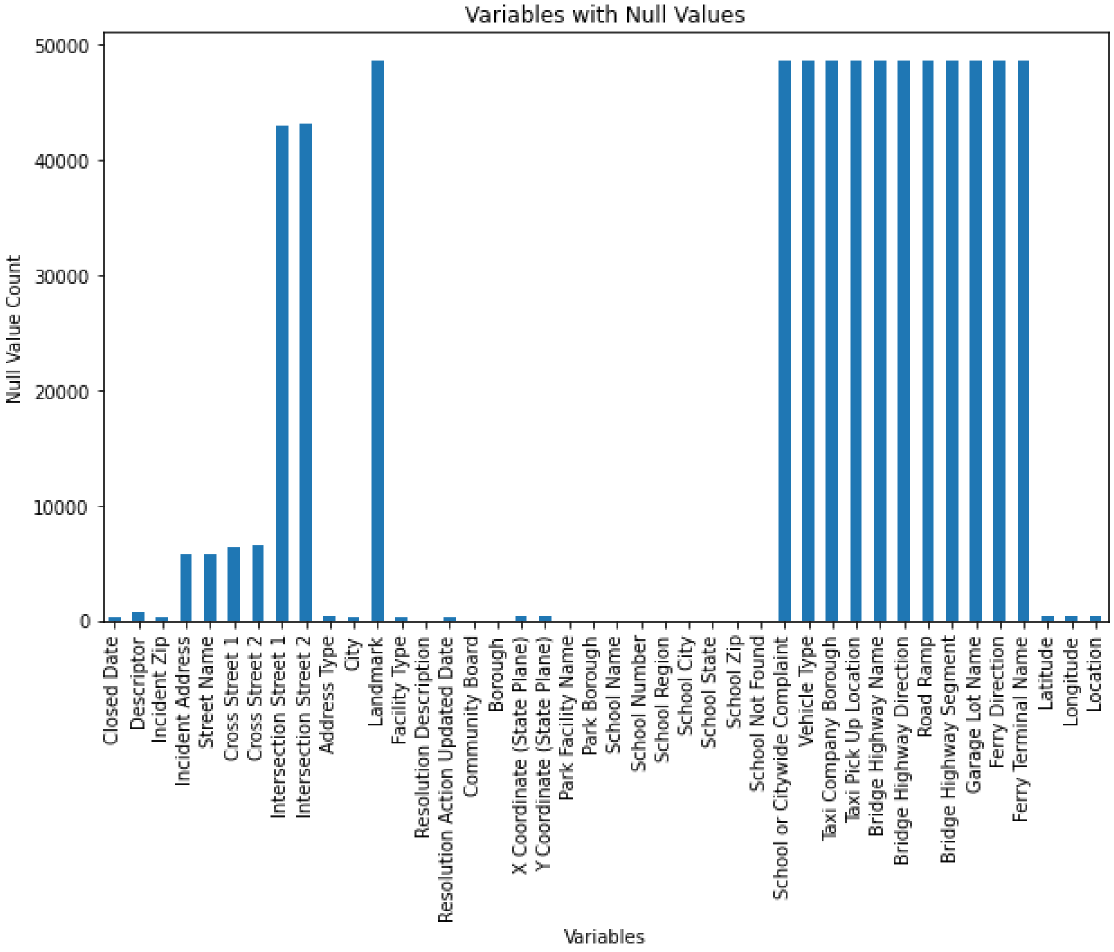
```
Garage Lot Name                 48673
Ferry Direction                 48673
Ferry Terminal Name             48673
Latitude                          410
Longitude                         410
Location                          410
dtype: int64
```

[28]:

```python
null_vars = null_counts[null_counts > 0]  # plot a graph of null values to visualize
```

[29]:

```python
plt.figure(figsize=(10, 6))
null_vars.plot(kind='bar')
plt.title('Variables with Null Values')
plt.xlabel('Variables')
plt.ylabel('Null Value Count')
plt.show()
```



[30]:
```
#dropping unneccessary columns
columns_to_drop = ["Intersection Street 1","Intersection Street␣
 ↪2","Landmark","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"]
df = df.drop(columns=columns_to_drop)
```

[41]:
```
df.shape
```

[41]:
```
(48673, 40)
```

[31]:
```
unique_values = df['Created Date'].unique()
unique_values
```

[31]:
```
array(['2015-12-31T23:59:45.000000000', '2015-12-31T23:59:44.000000000',
       '2015-12-31T23:59:29.000000000', …,
       '2015-11-14T11:00:28.000000000', '2015-11-14T10:59:17.000000000',
       '2015-11-14T10:59:06.000000000'], dtype='datetime64[ns]')
```

[32]:
```
unique_values1 = df['Closed Date'].unique()
unique_values1
```

[32]:
```
array(['2016-01-01T00:55:15.000000000', '2016-01-01T01:26:57.000000000',
       '2016-01-01T04:51:03.000000000', …,
       '2015-11-14T11:42:22.000000000', '2015-11-14T12:51:31.000000000',
       '2015-11-14T12:14:52.000000000'], dtype='datetime64[ns]')
```

[52]:
```
#  convert to pd.date format
df['Created Date'] = pd.to_datetime(df['Created Date'])
```

[53]:
```
#convert to  pd.date format
df['Closed Date'] = pd.to_datetime(df['Closed Date'])
```

[54]:
```
# Calculate the response time for each row
df['Request_Closing_Time'] = df['Closed Date'] - df['Created Date']
```

[55]:
```
# Print the response time for each row
print(df['Request_Closing_Time'])
```

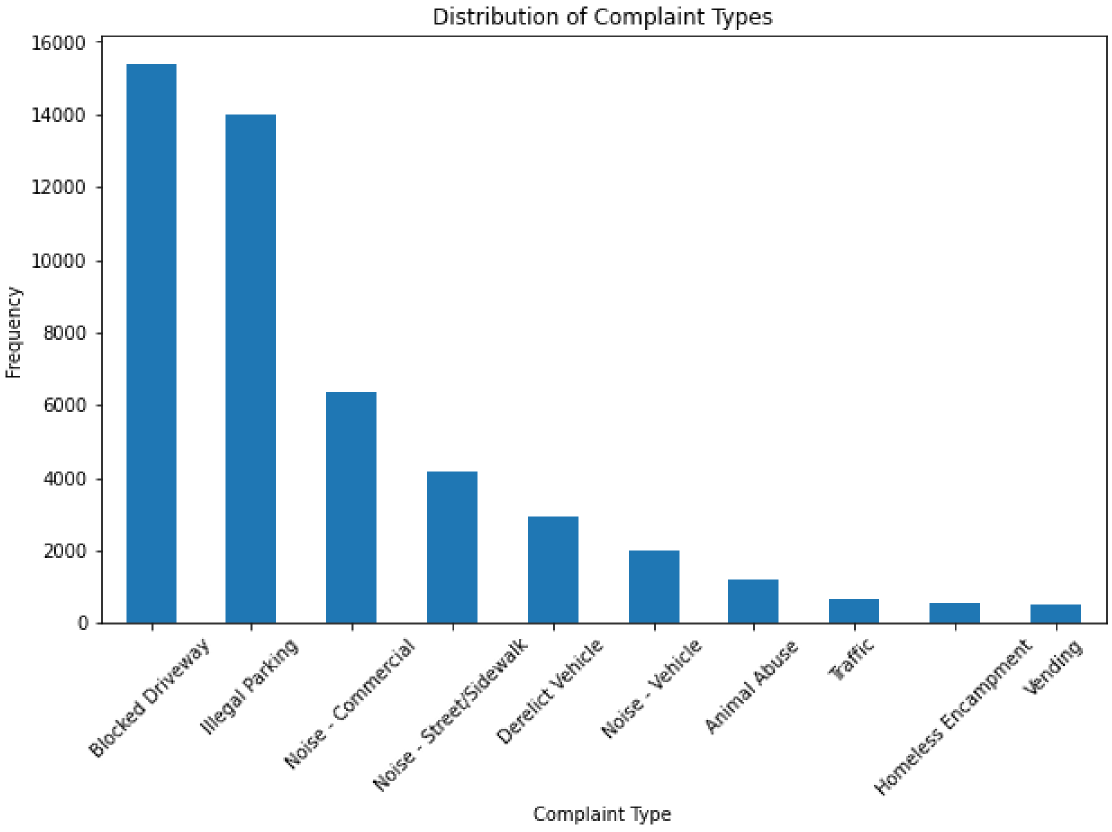
```
0       0 days 00:55:30
1       0 days 01:27:13
2       0 days 04:51:34
3       0 days 07:45:27
4       0 days 03:27:44
              …
48668   0 days 03:06:02
```

```
48669   0 days 02:54:49
48670   0 days 00:41:54
48671   0 days 01:52:14
48672   0 days 01:15:46
Name: Request_Closing_Time, Length: 48673, dtype: timedelta64[ns]
```

Based on generic data mining of the service request data, 1. Distribution of Complaint Types:

[56]:

```
complaint_types = df['Complaint Type'].value_counts().head(10)  # Top 10␣
 ↪complaint types

# Plotting the bar chart
plt.figure(figsize=(10, 6))
complaint_types.plot(kind='bar')
plt.xlabel('Complaint Type')
plt.ylabel('Frequency')
plt.title('Distribution of Complaint Types')
plt.xticks(rotation=45)
plt.show()
```



[57]:

```
#These are the top issues reported and their frequencies
```

[58]:

```
average_response_time = df['Request_Closing_Time'].dt.total_seconds() / (60 *␣
 ↪60) #request closing time in hours
print("Average Response Time:", average_response_time)
```

```
Average Response Time: 0        0.925000
1         1.453611
2         4.859444
3         7.757500
4         3.462222
            …
48668     3.100556
48669     2.913611
48670     0.698333
48671     1.870556
48672     1.262778
Name: Request_Closing_Time, Length: 48673, dtype: float64
```

[59]:

```
# Group by complaint type and calculate the average request closing time
average_response_time = df.groupby('Complaint Type')['Request_Closing_Time']
```

[63]:

```
df['Complaint Type'] = df['Complaint Type'].fillna('Unknown')
```

[64]:

```
# Group by complaint type and calculate the average request closing time
average_closing_time = df.groupby('Complaint Type')['Request_Closing_Time']

# Print the average closing time for each complaint type
print(average_closing_time)
```

```
<pandas.core.groupby.generic.SeriesGroupBy object at 0x7f050e8090d0>
```

[66]:

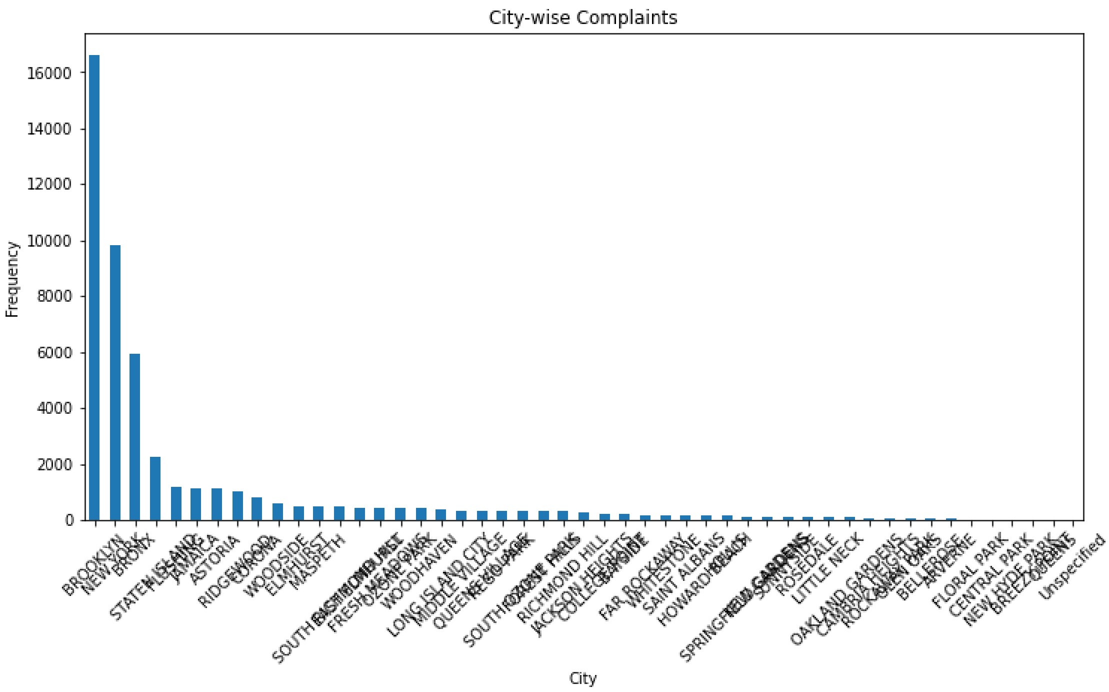
```
# Print the complaint types with longer and shorter response times
longer_response_types = average_closing_time.tail(5)  # Example: Print top 5␣
 ↪complaint types with longer response times
shorter_response_types = average_closing_time.head(5)  # Example: Print top 5␣
 ↪complaint types with shorter response times

print("Complaint types with longer response times:")
print(longer_response_types)

print("\nComplaint types with shorter response times:")
print(shorter_response_types)
```

```
Complaint types with longer response times:
23527    0 days 00:19:01
25910    0 days 02:08:33
27205    0 days 03:29:16
28024    0 days 09:15:50
```

```
33619   0 days 01:53:46
                 …
48668   0 days 03:06:02
48669   0 days 02:54:49
48670   0 days 00:41:54
48671   0 days 01:52:14
48672   0 days 01:15:46
Name: Request_Closing_Time, Length: 99, dtype: timedelta64[ns]

Complaint types with shorter response times:
0       0 days 00:55:30
1       0 days 01:27:13
2       0 days 04:51:34
3       0 days 07:45:27
4       0 days 03:27:44
                 …
23527   0 days 00:19:01
25745   0 days 06:00:15
34227   0 days 12:06:43
37949   0 days 01:29:20
45424   1 days 05:00:42
Name: Request_Closing_Time, Length: 99, dtype: timedelta64[ns]
```

[67]:

```
#Visualize the number of complaints reported in different locations (e.g.,␣
 ↪boroughs, neighborhoods) using a bar chart or map.
#Frequency Plot for City-wise Complaints
plt.figure(figsize=(12, 6))
df['City'].value_counts().plot(kind='bar')
plt.title('City-wise Complaints')
plt.xlabel('City')
plt.ylabel('Frequency')
plt.xticks(rotation=45)
plt.show()
```



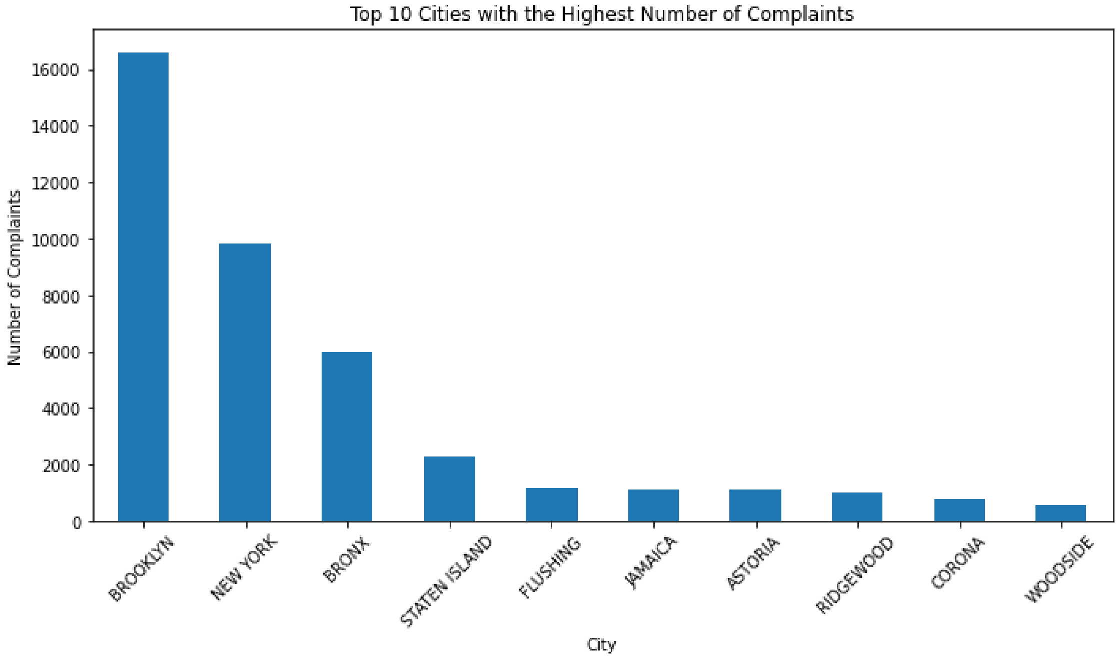
[68]:

```python
# Top 10 complaints types
top_10_complaints = df['Complaint Type'].value_counts().head(10)
print(top_10_complaints)
```

```
Blocked Driveway           15396
Illegal Parking            14012
Noise - Commercial          6374
Noise - Street/Sidewalk     4172
Derelict Vehicle            2963
Noise - Vehicle             1984
Animal Abuse                1210
Traffic                      673
Homeless Encampment          545
Vending                      498
Name: Complaint Type, dtype: int64
```

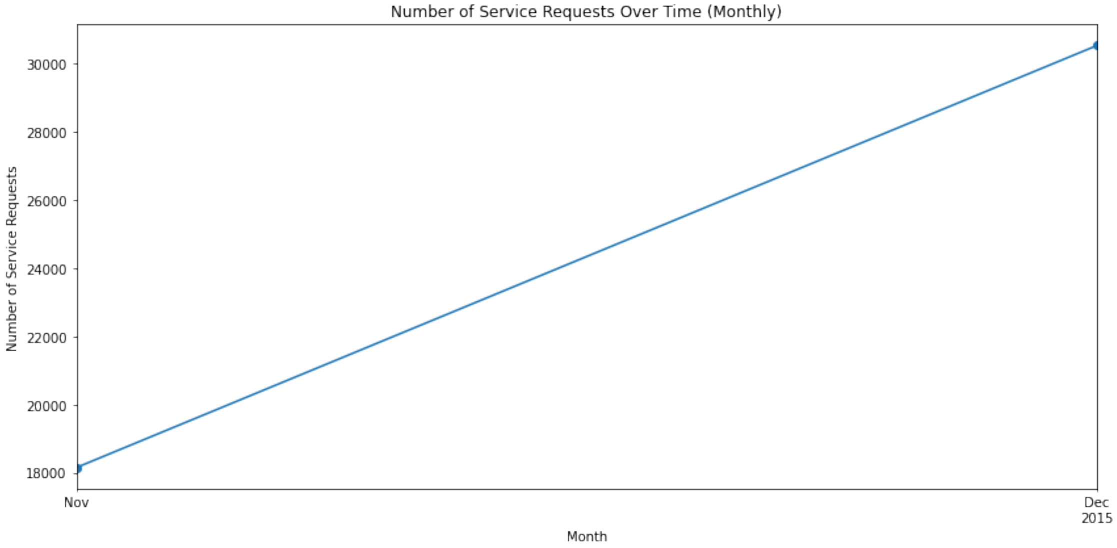
[69]:

```python
complaints_by_city = df['City'].value_counts()
plt.figure(figsize=(10, 6))
complaints_by_city.head(10).plot(kind='bar')
plt.xlabel('City')
plt.ylabel('Number of Complaints')
plt.title('Top 10 Cities with the Highest Number of Complaints')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



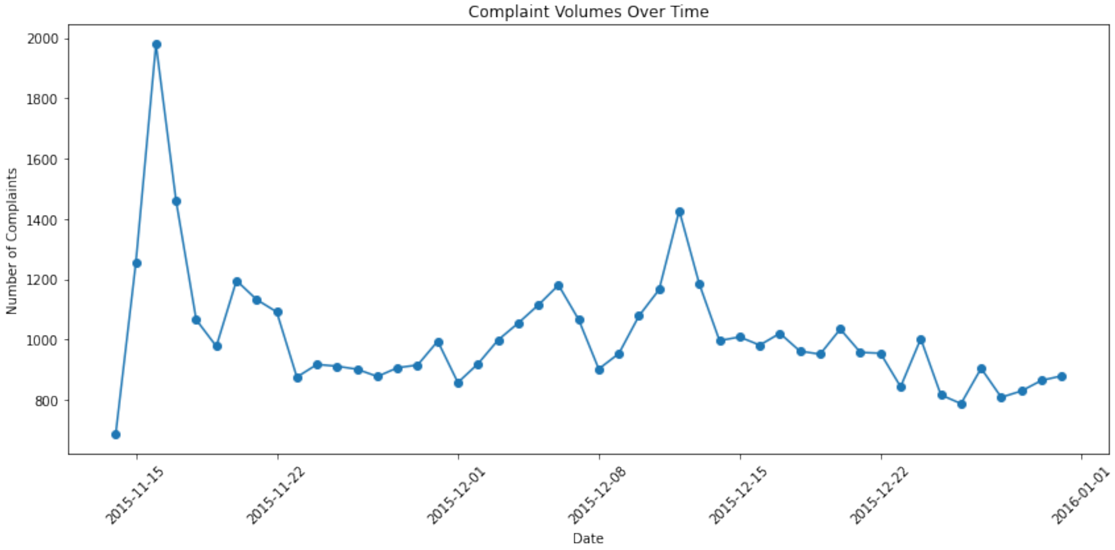
[70]:

```
#Plot the number of service requests over time (e.g., monthly or yearly) using␣
↪a line chart or bar chart.

df['Created Date'] = pd.to_datetime(df['Created Date'])

# Group by month and count the number of requests in each month
requests_by_month = df.groupby(df['Created Date'].dt.to_period('M')).size()

# Plotting the line chart to visualize the number of service requests over time
plt.figure(figsize=(12, 6))
requests_by_month.plot(kind='line', marker='o')
plt.xlabel('Month')
plt.ylabel('Number of Service Requests')
plt.title('Number of Service Requests Over Time (Monthly)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[72]:
```
#To identify any notable trends in complaint volumes, such as increasing or␣
↪decreasing patterns

# Group by date and count the number of complaints on each date
complaints_by_date = df.groupby(df['Created Date'].dt.date).size()
```

[73]:
```
# Plotting the line chart to visualize the complaint volumes over time
plt.figure(figsize=(12, 6))
complaints_by_date.plot(kind='line', marker='o')
plt.xlabel('Date')
plt.ylabel('Number of Complaints')
plt.title('Complaint Volumes Over Time')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[74]:

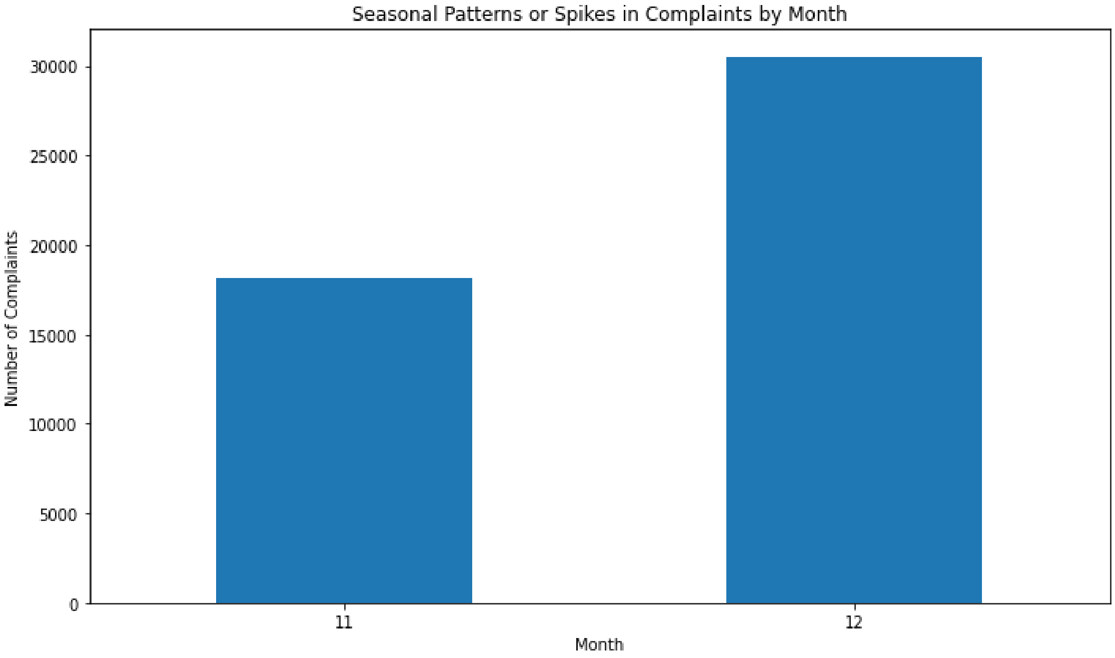
```
#To analyze seasonal patterns or spikes in specific periods

# Extract the month from the 'Created Date' column
df['Month'] = df['Created Date'].dt.month
```

[75]:

```
# Count the number of complaints in each month
complaints_by_month = df.groupby('Month').size()
```

[76]:

```
# Plotting the bar chart to visualize the seasonal patterns or spikes

plt.figure(figsize=(10, 6))
complaints_by_month.plot(kind='bar')
plt.xlabel('Month')
plt.ylabel('Number of Complaints')
plt.title('Seasonal Patterns or Spikes in Complaints by Month')
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```


Seasonal Patterns or Spikes in Complaints by Month
Number of Complaints
0
5000
10000
15000
20000
25000
30000
11
12
Month

[ ]: