Customer Service request Analysis - Project Report

June 26, 2021

1 Customer Service request Analysis - Project Report

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
import datetime
from scipy import stats
%matplotlib inline
```

1.1 Task 1

1.1.1 Importing the NYC311 Data

[2]:
```
customerSR = pd.read_csv("311_Service_Requests_from_2010_to_Present.csv",␣
 ↪low_memory=False)
```

[3]:
```
customerSR.head()
```

[3]:
```
   Unique Key             Created Date    Closed Date Agency  \
0    32310363  12/31/2015 11:59:45 PM  01-01-16 0:55   NYPD
1    32309934  12/31/2015 11:59:44 PM  01-01-16 1:26   NYPD
2    32309159  12/31/2015 11:59:29 PM  01-01-16 4:51   NYPD
3    32305098  12/31/2015 11:57:46 PM  01-01-16 7:43   NYPD
4    32306529  12/31/2015 11:56:58 PM  01-01-16 3:24   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]
```

Checking Shape of the dataframe

[4]:
```
customerSR.shape
```

[4]:
```
(300698, 53)
```

Checking datatypes and null values of each column

[5]:
```
customerSR.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 300698 entries, 0 to 300697
Data columns (total 53 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   Unique Key                      300698 non-null  int64  
 1   Created Date                    300698 non-null  object 
 2   Closed Date                     298534 non-null  object 
 3   Agency                          300698 non-null  object 
 4   Agency Name                     300698 non-null  object 
 5   Complaint Type                  300698 non-null  object 
 6   Descriptor                      294784 non-null  object 
 7   Location Type                   300567 non-null  object 
 8   Incident Zip                    298083 non-null  float64
 9   Incident Address                256288 non-null  object 
 10  Street Name                     256288 non-null  object 
 11  Cross Street 1                  251419 non-null  object 
 12  Cross Street 2                  250919 non-null  object 
 13  Intersection Street 1           43858 non-null   object 
 14  Intersection Street 2           43362 non-null   object 
 15  Address Type                    297883 non-null  object 
 16  City                            298084 non-null  object 
 17  Landmark                        349 non-null     object 
 18  Facility Type                   298527 non-null  object 
 19  Status                          300698 non-null  object 
 20  Due Date                        300695 non-null  object 
 21  Resolution Description          300698 non-null  object 
 22  Resolution Action Updated Date  298511 non-null  object 
 23  Community Board                 300698 non-null  object 
 24  Borough                         300698 non-null  object 
 25  X Coordinate (State Plane)      297158 non-null  float64
 26  Y Coordinate (State Plane)      297158 non-null  float64
 27  Park Facility Name              300698 non-null  object 
 28  Park Borough                    300698 non-null  object 
 29  School Name                     300698 non-null  object 
 30  School Number                   300698 non-null  object 
 31  School Region                   300697 non-null  object 
 32  School Code                     300697 non-null  object 
 33  School Phone Number             300698 non-null  object 
 34  School Address                  300698 non-null  object 
 35  School City                     300698 non-null  object 
 36  School State                    300698 non-null  object 
 37  School Zip                      300697 non-null  object 
 38  School Not Found                300698 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             243 non-null     object 
```

```
 44  Bridge Highway Direction        243 non-null     object
 45  Road Ramp                       213 non-null     object
 46  Bridge Highway Segment          213 non-null     object
 47  Garage Lot Name                 0 non-null       float64
 48  Ferry Direction                 1 non-null       object
 49  Ferry Terminal Name             2 non-null       object
 50  Latitude                        297158 non-null  float64
 51  Longitude                       297158 non-null  float64
 52  Location                        297158 non-null  object
dtypes: float64(10), int64(1), object(42)
memory usage: 121.6+ MB
```

Checking Missing Values

[6]:
```
missing = customerSR.isnull().sum()  # findig out missing values
missing = missing.sort_values() # sorting the missing values
missing = missing[missing > 0] # columns with 0 missing values are removed
```

Converting it into percent so that we can get the idea of what percent of missing values are present in the columns

[7]:
```
missing_percent = (missing/customerSR.shape[0])*100
missing_percent
```

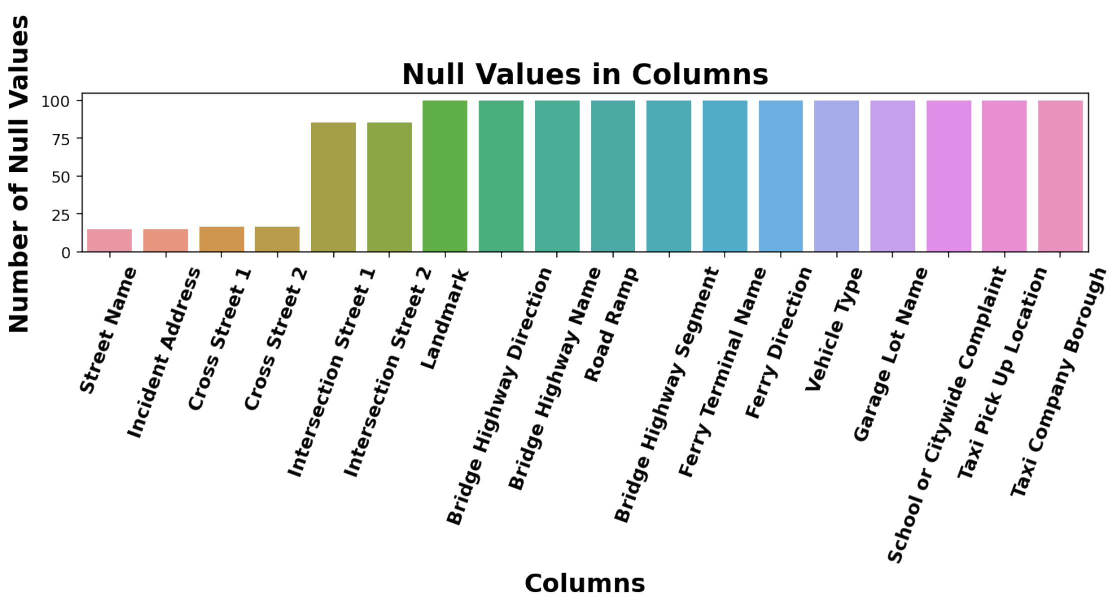
[7]:
```
School Zip                         0.000333
School Region                      0.000333
School Code                        0.000333
Due Date                           0.000998
Location Type                      0.043565
Closed Date                        0.719659
Facility Type                      0.721987
Resolution Action Updated Date     0.727308
City                               0.869311
Incident Zip                       0.869643
Address Type                       0.936155
Latitude                           1.177261
Y Coordinate (State Plane)         1.177261
X Coordinate (State Plane)         1.177261
Longitude                          1.177261
Location                           1.177261
Descriptor                         1.966757
Street Name                       14.768971
Incident Address                  14.768971
Cross Street 1                    16.388203
Cross Street 2                    16.554483
Intersection Street 1             85.414602
Intersection Street 2             85.579552
```

```
Landmark                          99.883937
Bridge Highway Direction          99.919188
Bridge Highway Name               99.919188
Road Ramp                         99.929165
Bridge Highway Segment            99.929165
Ferry Terminal Name               99.999335
Ferry Direction                   99.999667
Vehicle Type                     100.000000
Garage Lot Name                  100.000000
School or Citywide Complaint     100.000000
Taxi Pick Up Location            100.000000
Taxi Company Borough             100.000000
dtype: float64
```

since there are missing values ranging from 0% to 100%, I am putting filter for the missing values greater than 2%

[8]:
```
missing_percent = missing_percent[missing_percent>2]
```

[9]:
```
# plotting the missing values as bar graph
plt.figure(figsize=(10,5),dpi=140)
sns.barplot(x= missing_percent.index, y = missing_percent.values)
plt.xticks(rotation=70,fontsize=12)
plt.title('Null Values in Columns', fontsize=18, fontweight="bold")
plt.ylabel('Number of Null Values', fontsize=16, fontweight="bold")
plt.xlabel('Columns', fontsize=16, fontweight="bold")
plt.xticks(fontsize=12, fontweight="bold")
plt.tight_layout()
```



I see that there are columns that have more then 85% missing values, therefore, dropping them

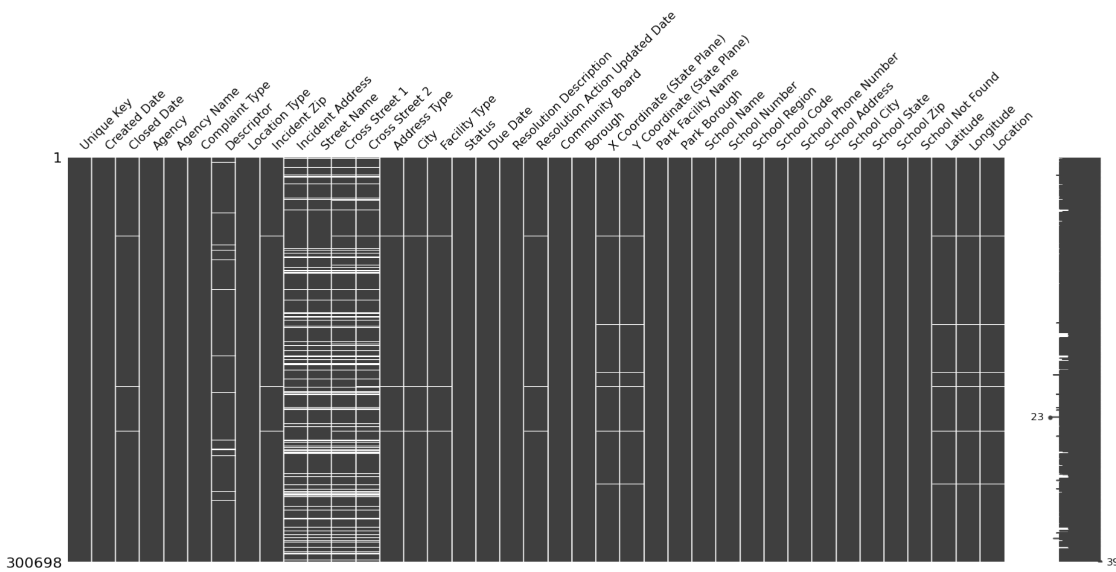
[10]:
```python
cols_to_drop = missing_percent[missing_percent>85].index
customerSR = customerSR.drop(cols_to_drop.to_list(), axis=1)
print(f'{len(cols_to_drop)} columns have been dropped. as they were having more␣
↪than 85% missing values.')
```

```
14 columns have been dropped. as they were having more than 85% missing values.
```

Visualizing the missing numbers in remaining columns

[11]:
```python
plt.figure(figsize=(15,9), dpi=150)
msno.matrix(customerSR)
plt.show()
```

```
<Figure size 2250x1350 with 0 Axes>
```



1.1.2 From the above matrix of missing values, we get the following observations:

- From the above matrix, 4 colums have lot of missing values across the same rows.
- Also, the missing values across these 4 colums almost lies in same row.
- Imputation may not be appropriate, we may have lot of dummy/guessed data

Columns having lesser than 85% missing values (which are not removed)

[12]:
```
less_85_missing = (missing_percent[~(missing_percent>85)]).round(2)
heading = pd.Series({'Columns':'Percent Missing Values'})
heading.append(less_85_missing)
```

[12]:
```
Columns                Percent Missing Values
Street Name                             14.77
Incident Address                        14.77
Cross Street 1                          16.39
Cross Street 2                          16.55
dtype: object
```

1.2 Task 2

1.2.1 Converting the columns 'Created Date' and Closed Date' to datetime, and creating new column 'Request_Closing_Time'

Checking if there are any null values in date related columns

[13]:
```
dates = customerSR[['Created Date','Closed Date','Due Date','Resolution Action␣
↪Updated Date']].isnull().sum()
```

Converting in Percent Format, for better understanding

[14]:
```
(dates/customerSR.shape[0])*100
```

[14]:
```
Created Date                      0.000000
Closed Date                       0.719659
Due Date                          0.000998
Resolution Action Updated Date    0.727308
dtype: float64
```

Since missing dates are less then 1%, I am dropping them, becuase imputation will still be a guess

[15]:
```
customerSR.dropna( how='any', subset=['Created Date','Closed Date','Due␣
↪Date','Resolution Action Updated Date'], inplace=True)
```

All the missing dates have been removed

[16]:
```
customerSR[['Created Date','Closed Date','Due Date','Resolution Action Updated␣
↪Date']].isnull().sum()
```

[16]:
```
Created Date                      0
Closed Date                       0
Due Date                          0
Resolution Action Updated Date    0
```

```
dtype: int64
```

Checing the data type of the dates column

[17]:
```
dates = customerSR[['Created Date','Closed Date','Due Date','Resolution Action␣
↪Updated Date']]
dates.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 298495 entries, 0 to 300697
Data columns (total 4 columns):
 #   Column                         Non-Null Count   Dtype
---  ------                         --------------   -----
 0   Created Date                   298495 non-null  object
 1   Closed Date                    298495 non-null  object
 2   Due Date                       298495 non-null  object
 3   Resolution Action Updated Date 298495 non-null  object
dtypes: object(4)
memory usage: 11.4+ MB
```

Converting the date-related columns to 'datetime' object

[18]:
```
# Below is the loop-way of converting to datetime, but it took longer time. so␣
↪commenting

# dates_index = ['Created Date','Closed Date','Due Date','Resolution Action␣
↪Updated Date']
# for index in dates_index:
#     customerSR[index] = pd.to_datetime(customerSR[index])
```

[19]:
```
dates_index = ['Created Date','Closed Date','Due Date','Resolution Action␣
↪Updated Date']
customerSR[dates_index] = customerSR[dates_index].apply(pd.to_datetime)
```

All the date related columns have been converted to datetime object type

[20]:
```
customerSR[['Created Date','Closed Date','Due Date','Resolution Action Updated␣
↪Date']].head()
```

[20]:
```
         Created Date         Closed Date            Due Date  \
0 2015-12-31 23:59:45 2016-01-01 00:55:00 2016-01-01 07:59:00
1 2015-12-31 23:59:44 2016-01-01 01:26:00 2016-01-01 07:59:00
2 2015-12-31 23:59:29 2016-01-01 04:51:00 2016-01-01 07:59:00
3 2015-12-31 23:57:46 2016-01-01 07:43:00 2016-01-01 07:57:00
4 2015-12-31 23:56:58 2016-01-01 03:24:00 2016-01-01 07:56:00
```

```
  Resolution Action Updated Date
0            2016-01-01 00:55:00
1            2016-01-01 01:26:00
2            2016-01-01 04:51:00
3            2016-01-01 07:43:00
4            2016-01-01 03:24:00
```

Creating 'Request_Closing_Time' as the time difference between creation and closing of request

[21]:
```
customerSR['Request_Closing_Time'] = customerSR['Closed Date'] -␣
↪customerSR['Created Date']
customerSR[['Created Date','Closed Date','Due Date','Resolution Action Updated␣
↪Date','Request_Closing_Time']].head()
```

[21]:
```
         Created Date         Closed Date            Due Date  \
0 2015-12-31 23:59:45 2016-01-01 00:55:00 2016-01-01 07:59:00
1 2015-12-31 23:59:44 2016-01-01 01:26:00 2016-01-01 07:59:00
2 2015-12-31 23:59:29 2016-01-01 04:51:00 2016-01-01 07:59:00
3 2015-12-31 23:57:46 2016-01-01 07:43:00 2016-01-01 07:57:00
4 2015-12-31 23:56:58 2016-01-01 03:24:00 2016-01-01 07:56:00

  Resolution Action Updated Date Request_Closing_Time
0            2016-01-01 00:55:00      0 days 00:55:15
1            2016-01-01 01:26:00      0 days 01:26:16
2            2016-01-01 04:51:00      0 days 04:51:31
3            2016-01-01 07:43:00      0 days 07:45:14
4            2016-01-01 03:24:00      0 days 03:27:02
```

[22]:
```
customerSR.columns
```

[22]:
```
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',
       'Address Type', 'City', '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', 'Latitude', 'Longitude', 'Location',
       'Request_Closing_Time'],
      dtype='object')
```

'Request_Closing_Time' is in days, hours:mins:sec format, converting it into total no. of seconds

[23]:
```
customerSR['Request_Closing_Time_tot_hrs'] = customerSR['Request_Closing_Time']/
↪np.timedelta64(1, 'h')
customerSR['Request_Closing_Time'] = customerSR['Request_Closing_Time']/np.
↪timedelta64(1, 's')
customerSR[['Created Date','Closed Date','Due Date','Resolution Action Updated␣
↪Date','Request_Closing_Time']].head()
```

[23]:
```
          Created Date         Closed Date            Due Date  \
0 2015-12-31 23:59:45 2016-01-01 00:55:00 2016-01-01 07:59:00
1 2015-12-31 23:59:44 2016-01-01 01:26:00 2016-01-01 07:59:00
2 2015-12-31 23:59:29 2016-01-01 04:51:00 2016-01-01 07:59:00
3 2015-12-31 23:57:46 2016-01-01 07:43:00 2016-01-01 07:57:00
4 2015-12-31 23:56:58 2016-01-01 03:24:00 2016-01-01 07:56:00

  Resolution Action Updated Date  Request_Closing_Time
0            2016-01-01 00:55:00                3315.0
1            2016-01-01 01:26:00                5176.0
2            2016-01-01 04:51:00               17491.0
3            2016-01-01 07:43:00               27914.0
4            2016-01-01 03:24:00               12422.0
```

1.3 Task 3:

1.3.1 Analysis - Major Insights

[24]:
```
customerSR.columns
```

[24]:
```
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',
       'Address Type', 'City', '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', 'Latitude', 'Longitude', 'Location',
       'Request_Closing_Time', 'Request_Closing_Time_tot_hrs'],
      dtype='object')
```

[25]:
```
customerSR['Agency'].unique()
```

[25]:
```
array(['NYPD'], dtype=object)
```

[26]:
```
(customerSR['Agency Name'].value_counts()/customerSR.shape[0])*100
```

```
[26]: New York City Police Department    99.99799
Internal Affairs Bureau             0.00201
Name: Agency Name, dtype: float64
```

- Agency is NYPD for all records
- Agency Names are 'New York City Police Department', and 'Internal Affairs Bureau'
- 99.99% records are for 'New York City Police Department'
- Of 300698 records, only 6 are for 'Internal Affairs Bureau'

Checking 'Compaint Types' and respective number of compaints/requests

```
[27]: customerSR['Complaint Type'].isnull().any() # Checking if there is any null␣
      ↪value in 'complaint types'
```

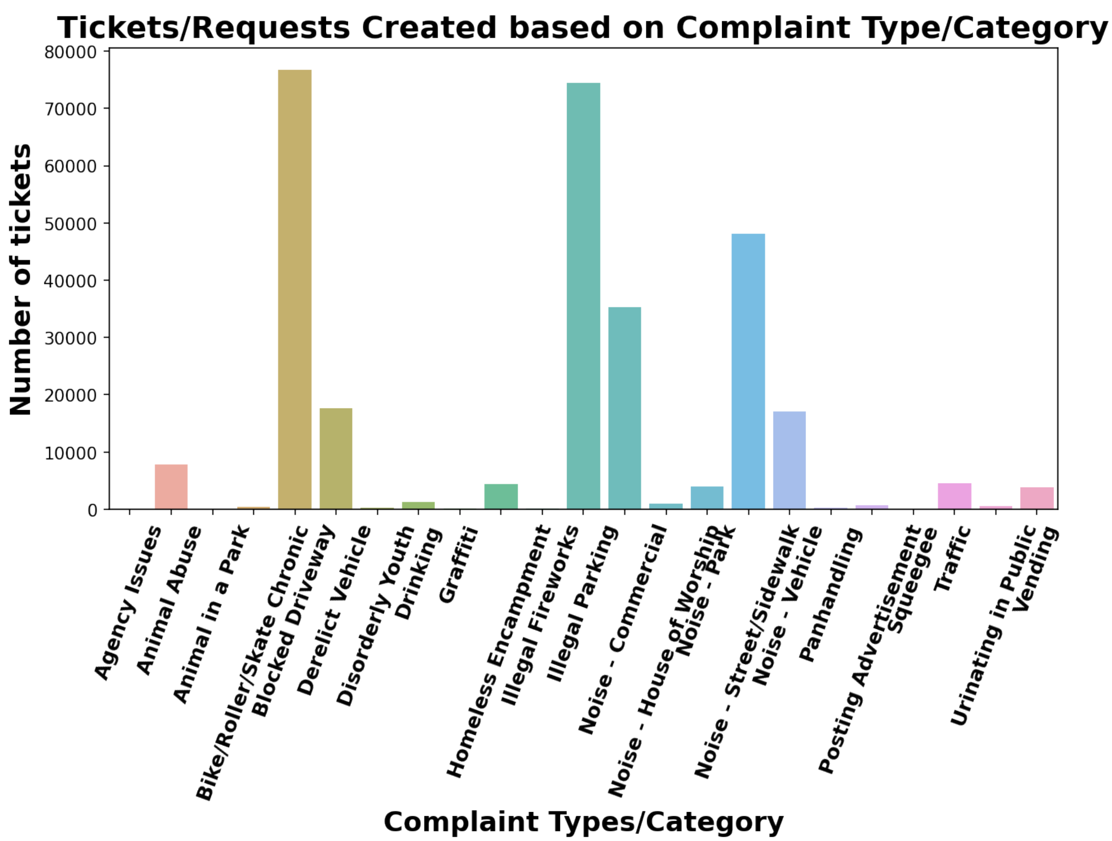
```
[27]: False
```

```
[28]: complaint_types = customerSR.groupby('Complaint Type').agg('count')['Unique␣
      ↪Key']
complaint_types = pd.DataFrame(complaint_types)
complaint_types
```

```
[28]:                            Unique Key
Complaint Type
Agency Issues                        6
Animal Abuse                      7768
Animal in a Park                     1
Bike/Roller/Skate Chronic          424
Blocked Driveway                 76804
Derelict Vehicle                 17586
Disorderly Youth                   286
Drinking                          1275
Graffiti                           113
Homeless Encampment               4414
Illegal Fireworks                  168
Illegal Parking                  74521
Noise - Commercial               35245
Noise - House of Worship           929
Noise - Park                      4021
Noise - Street/Sidewalk          48068
Noise - Vehicle                  17032
Panhandling                        305
Posting Advertisement              647
Squeegee                             4
Traffic                           4493
Urinating in Public                592
Vending                           3793
```

Visualizing with the help of Bar-Graph

[29]:
```
plt.figure(figsize=(10,5),dpi = 150)
sns.barplot(x = complaint_types['Unique Key'].index,
            y = complaint_types['Unique Key'].values,
            alpha = 0.8)
plt.title('Tickets/Requests Created based on Complaint Type/Category', fontsize=18, fontweight="bold")
plt.ylabel('Number of tickets', fontsize=16, fontweight="bold")
plt.xlabel('Complaint Types/Category', fontsize=16, fontweight="bold")
plt.xticks(rotation=70,fontsize=12,fontweight="bold")
plt.show()
```



Same can also be achieved by value_counts() function, which gives Complaint Types in decending order by default

[30]:
```
complaint_types = customerSR['Complaint Type'].value_counts()
complaint_types = pd.DataFrame(complaint_types)
complaint_types
```

[30]:
```
                              Complaint Type
Blocked Driveway                       76804
Illegal Parking                        74521
Noise - Street/Sidewalk                48068
Noise - Commercial                     35245
Derelict Vehicle                       17586
Noise - Vehicle                        17032
Animal Abuse                            7768
Traffic                                 4493
Homeless Encampment                     4414
Noise - Park                            4021
Vending                                 3793
Drinking                                1275
Noise - House of Worship                 929
Posting Advertisement                    647
Urinating in Public                      592
Bike/Roller/Skate Chronic                424
Panhandling                              305
Disorderly Youth                         286
Illegal Fireworks                        168
Graffiti                                 113
Agency Issues                              6
Squeegee                                   4
Animal in a Park                           1
```

Let us visualize it with percent and cumulative percent, to get better idea

[31]:
```
complaint_types['Percent of Complaints'] = ((complaint_types['Complaint Type']/
↪sum(complaint_types['Complaint Type']))*100).round(2)
complaint_types['Cumulative Percent'] = complaint_types['Percent of␣
↪Complaints'].cumsum()
complaint_types
```

[31]:
```
                              Complaint Type  Percent of Complaints  \
Blocked Driveway                       76804                  25.73
Illegal Parking                        74521                  24.97
Noise - Street/Sidewalk                48068                  16.10
Noise - Commercial                     35245                  11.81
Derelict Vehicle                       17586                   5.89
Noise - Vehicle                        17032                   5.71
Animal Abuse                            7768                   2.60
Traffic                                 4493                   1.51
Homeless Encampment                     4414                   1.48
Noise - Park                            4021                   1.35
Vending                                 3793                   1.27
Drinking                                1275                   0.43
Noise - House of Worship                 929                   0.31
```

```
Posting Advertisement                  647                   0.22
Urinating in Public                    592                   0.20
Bike/Roller/Skate Chronic              424                   0.14
Panhandling                            305                   0.10
Disorderly Youth                       286                   0.10
Illegal Fireworks                      168                   0.06
Graffiti                               113                   0.04
Agency Issues                            6                   0.00
Squeegee                                 4                   0.00
Animal in a Park                         1                   0.00

                           Cumulative Percent
Blocked Driveway                        25.73
Illegal Parking                         50.70
Noise - Street/Sidewalk                 66.80
Noise - Commercial                      78.61
Derelict Vehicle                        84.50
Noise - Vehicle                         90.21
Animal Abuse                            92.81
Traffic                                 94.32
Homeless Encampment                     95.80
Noise - Park                            97.15
Vending                                 98.42
Drinking                                98.85
Noise - House of Worship                99.16
Posting Advertisement                   99.38
Urinating in Public                     99.58
Bike/Roller/Skate Chronic               99.72
Panhandling                             99.82
Disorderly Youth                        99.92
Illegal Fireworks                       99.98
Graffiti                               100.02
Agency Issues                          100.02
Squeegee                               100.02
Animal in a Park                       100.02
```

[32]:
```
# total number of compaint types
len(complaint_types)
```

[32]: 23

[33]:
```
# complaint type corresponding to the minimum number of complains
complaint_types[complaint_types['Complaint Type'] == complaint_types['Complaint␣
↪Type'].min()].index[0]
```

[33]: 'Animal in a Park'

[34]:
```python
# complaint type corresponding to the maximum number of complains
complaint_types[complaint_types['Complaint Type'] == complaint_types['Complaint␣
↪Type'].max()].index[0]
```

[34]:
```
'Blocked Driveway'
```

[35]:
```python
# top 5 compalint types
complaint_types.head().index.to_list()
```

[35]:
```
['Blocked Driveway',
 'Illegal Parking',
 'Noise - Street/Sidewalk',
 'Noise - Commercial',
 'Derelict Vehicle']
```

[36]:
```python
# Requests contributing top 95% complaint types
complaint_types[complaint_types['Cumulative Percent']>=95].index.to_list()
```

[36]:
```
['Homeless Encampment',
 'Noise - Park',
 'Vending',
 'Drinking',
 'Noise - House of Worship',
 'Posting Advertisement',
 'Urinating in Public',
 'Bike/Roller/Skate Chronic',
 'Panhandling',
 'Disorderly Youth',
 'Illegal Fireworks',
 'Graffiti',
 'Agency Issues',
 'Squeegee',
 'Animal in a Park']
```

[37]:
```python
# Number of complaint types comprising of 95% of total complains
len(complaint_types[complaint_types['Cumulative Percent']>=95].index)
```

[37]:
```
15
```

[38]:
```python
# lower 5 compalint types
complaint_types.tail().index.to_list()[::-1]
```

[38]:
```
['Animal in a Park',
 'Squeegee',
 'Agency Issues',
 'Graffiti',
 'Illegal Fireworks']
```

Analysis - Complaint type and Number of requests

- There are total of 23 compaint types/categories
- Of these 23 Categories, 15 complaint types are contributing to 95% of all the requests. There 15 complaint types should be focussed upon, such that lesser requests are created for these types
- Maximum compaint type is for 'Blocked Driveway'. This should be checked if permanent solution can be arranged, to reduce the count of such tickets/requests
- Minimum complaint type is for 'Animal in a Park'.
- Top 5 complaint categories are as follows:
  - 'Blocked Driveway'
  - 'Illegal Parking'
  - 'Noise - Street/Sidewalk'
  - 'Noise - Commercial'
  - 'Derelict Vehicle'
- Lowest 5 complaint categories are as follows:
  - 'Animal in a Park'
  - 'Squeegee'
  - 'Agency Issues'
  - 'Graffiti'
  - 'Illegal Fireworks'

Exploring Dates - Created, Closed and Time Elapsed between these

[39]:
```
# Checking if there is any missing value
customerSR[['Created Date','Closed Date','Request_Closing_Time']].isnull().any()
```

[39]:
```
Created Date            False
Closed Date             False
Request_Closing_Time    False
dtype: bool
```

[40]:
```
customerSR[['Created Date','Closed Date','Request_Closing_Time']].head()
```

[40]:

	Created Date	Closed Date	Request_Closing_Time
0	2015-12-31 23:59:45	2016-01-01 00:55:00	3315.0
1	2015-12-31 23:59:44	2016-01-01 01:26:00	5176.0
2	2015-12-31 23:59:29	2016-01-01 04:51:00	17491.0
3	2015-12-31 23:57:46	2016-01-01 07:43:00	27914.0
4	2015-12-31 23:56:58	2016-01-01 03:24:00	12422.0

Extracting Month Name and number(1-12), and year

[41]:
```
# To get number of respective month, 1-12
customerSR['Created_Date_Month_num'] = customerSR['Created Date'].dt.month

#To get number of respective year
customerSR['Created_Date_Year'] = customerSR['Created Date'].dt.year
```

```
# To get name of the respective month, Jan to Dec
customerSR['Created_Date_Month'] = customerSR['Created Date'].dt.month_name()
```

[42]:
```
customerSR[['Created␣
↪Date','Created_Date_Month','Created_Date_Year','Created_Date_Month_num']].
↪head()
```

[42]:
```
         Created Date Created_Date_Month  Created_Date_Year  \
0 2015-12-31 23:59:45           December               2015
1 2015-12-31 23:59:44           December               2015
2 2015-12-31 23:59:29           December               2015
3 2015-12-31 23:57:46           December               2015
4 2015-12-31 23:56:58           December               2015

   Created_Date_Month_num
0                      12
1                      12
2                      12
3                      12
4                      12
```

[43]:
```
customerSR['Created_Date_Month'].unique().tolist()
```

[43]:
```
['December',
 'November',
 'October',
 'September',
 'August',
 'July',
 'June',
 'May',
 'April',
 'March']
```

Let us see Requests/tickets created on monthly basis

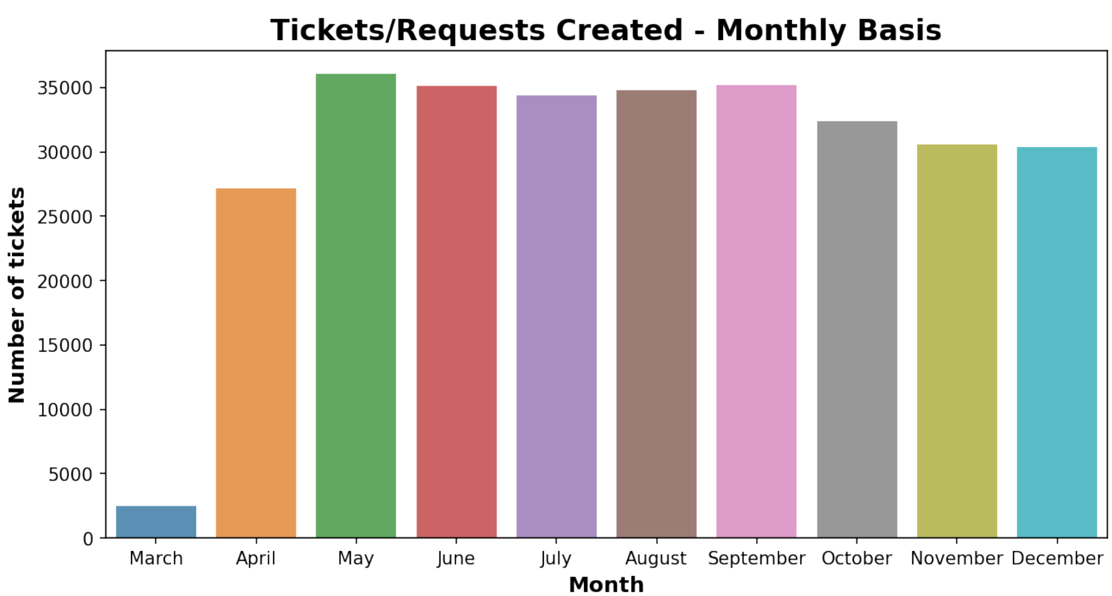
[44]:
```
month_wise_ticket = customerSR.
↪groupby(['Created_Date_Month_num','Created_Date_Month']).count()['Unique␣
↪Key']
month_wise_ticket = pd.DataFrame(month_wise_ticket.reset_index(level=0,␣
↪drop=True))
month_wise_ticket
```

[44]:
```
                    Unique Key
Created_Date_Month
March                     2457
```

```
April                     27168
May                       36069
June                      35142
July                      34378
August                    34773
September                 35176
October                   32398
November                  30594
December                  30340
```

Visualizing in the form of Bar Graph

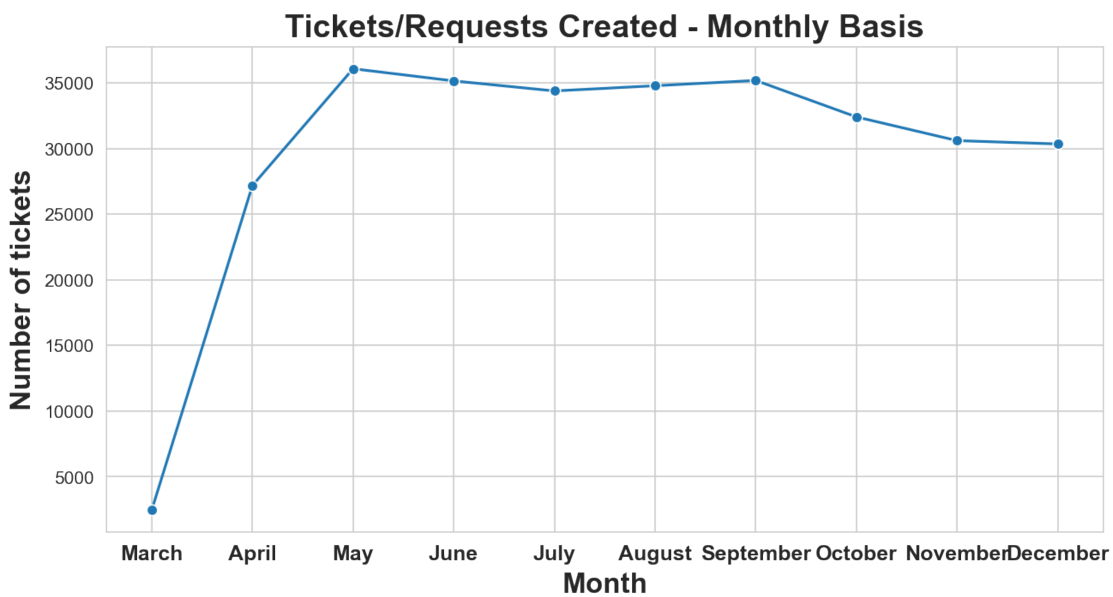
[45]:
```
plt.figure(figsize=(10,5), dpi=150)
sns.barplot(x = month_wise_ticket['Unique Key'].index,
            y = month_wise_ticket['Unique Key'].values,
            alpha = 0.8)
plt.title('Tickets/Requests Created - Monthly Basis',␣
 ↪fontsize=16,fontweight="bold")
plt.ylabel('Number of tickets', fontsize=12,fontweight="bold")
plt.xlabel('Month', fontsize=12,fontweight="bold")
plt.show()
```



Also, let us plot the trend chart, monthwise, to see how the patten of request creation looks like

[46]:
```
plt.figure(figsize=(10,5), dpi=150)
sns.set_style("whitegrid")
```

```
sns.lineplot(x = month_wise_ticket['Unique Key'].index,
            y = month_wise_ticket['Unique Key'].values, marker='o')
plt.title('Tickets/Requests Created - Monthly Basis', fontsize=18,␣
 ↪fontweight="bold")
plt.ylabel('Number of tickets', fontsize=16, fontweight="bold")
plt.xlabel('Month', fontsize=16, fontweight="bold")
plt.xticks(fontsize=12, fontweight="bold")
plt.show()
```



[47]:
```
month_wise_ticket = pd.DataFrame(month_wise_ticket).sort_values(by = 'Unique␣
 ↪Key', ascending=False)
month_wise_ticket['Percent of Requests'] = ((month_wise_ticket['Unique Key']/
 ↪sum(month_wise_ticket['Unique Key']))*100).round(2)
month_wise_ticket['Cumulative Percentage'] = month_wise_ticket['Percent of␣
 ↪Requests'].cumsum()
month_wise_ticket
```

[47]:

	Unique Key	Percent of Requests	Cumulative Percentage
Created_Date_Month			
May	36069	12.08	12.08
September	35176	11.78	23.86
June	35142	11.77	35.63
August	34773	11.65	47.28
July	34378	11.52	58.80
October	32398	10.85	69.65
November	30594	10.25	79.90
December	30340	10.16	90.06

```
April                 27168              9.10               99.16
March                  2457              0.82               99.98
```

[48]:
```
# month having maximum requests/complains
month_wise_ticket[month_wise_ticket['Unique Key'] == month_wise_ticket['Unique␣
 ↪Key'].max()].index[0]
```

[48]: 'May'

[49]:
```
# month having Minimum requests/complains
month_wise_ticket[month_wise_ticket['Unique Key'] == month_wise_ticket['Unique␣
 ↪Key'].min()].index[0]
```

[49]: 'March'

[50]:
```
# 75% of the tickets/requests created are in the following months
month_wise_ticket[month_wise_ticket['Cumulative Percentage']<=75].index.
 ↪to_list()
```

[50]: ['May', 'September', 'June', 'August', 'July', 'October']

[51]:
```
customerSR['Created_Date_Year'].value_counts()
```

[51]:
```
2015    298495
Name: Created_Date_Year, dtype: int64
```

1.3.2 Analysis - Number of requests 'Created' on Monthly Basis

- Requests are created on the following months:
  - March
  - April
  - May
  - June
  - July
  - August
  - September
  - October
  - November
  - December
- There were no requests/tickets created in the month of 'January' and 'February', based on the given data
- As we observe from trendchart, we see the tickets/requests were minimum in the month of 'March'. It rose up to attain highest value in the month of 'May'.
- From month 'May' till 'September', the ticket count was almost consistant.
- From month 'September' to 'December', we observe slow decrease in the count of requests/ticket created.

- Maximum tickets/requests created was recorded in the month of 'May'
- Minimum tickets/requests created was recorded in the month of 'March'
- 75% of the tickets/requests created are in the following months:
  - 'May', 'September', 'June', 'August', 'July', 'October'
- We have data for year 2015 only. Therefore, yearwise granuality analysis couldn't be achieved.

Let us see Requests/tickets 'Closed' on monthly basis

[52]:
```
customerSR[['Unique Key','Closed Date']].isnull().any()
```

[52]:
```
Unique Key     False
Closed Date    False
dtype: bool
```

[53]:
```
customerSR[['Unique Key','Closed Date']].head()
```

[53]:
```
   Unique Key         Closed Date
0    32310363 2016-01-01 00:55:00
1    32309934 2016-01-01 01:26:00
2    32309159 2016-01-01 04:51:00
3    32305098 2016-01-01 07:43:00
4    32306529 2016-01-01 03:24:00
```

Extracting Month Name and number(1-12), and year

[54]:
```
# To get number of respective month, 1-12
customerSR['Closed_Date_Month_num'] = customerSR['Closed Date'].dt.month

#To get number of respective year
customerSR['Closed_Date_Year'] = customerSR['Closed Date'].dt.year

# To get name of the respective month, Jan to Dec
customerSR['Closed_Date_Month'] = customerSR['Closed Date'].dt.month_name()
```

[55]:
```
customerSR[['Unique Key','Closed␣
↪Date','Closed_Date_Month_num','Closed_Date_Year','Closed_Date_Month']].head()
```

[55]:
```
   Unique Key         Closed Date  Closed_Date_Month_num  Closed_Date_Year  \
0    32310363 2016-01-01 00:55:00                      1              2016
1    32309934 2016-01-01 01:26:00                      1              2016
2    32309159 2016-01-01 04:51:00                      1              2016
3    32305098 2016-01-01 07:43:00                      1              2016
4    32306529 2016-01-01 03:24:00                      1              2016

  Closed_Date_Month
0           January
1           January
```

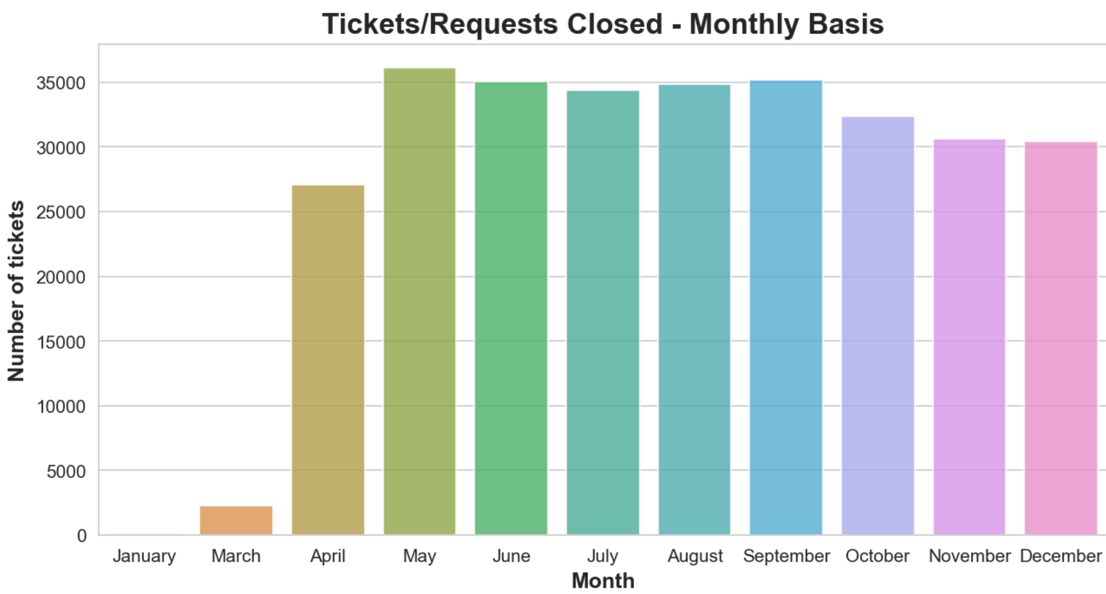
```
2           January
3           January
4           January
```

[56]:
```
month_wise_ticket_closed = customerSR.
 ↪groupby(['Closed_Date_Month_num','Closed_Date_Month']).count()['Unique Key']
month_wise_ticket_closed = pd.DataFrame(month_wise_ticket_closed.
 ↪reset_index(level=0, drop=True))
month_wise_ticket_closed
```

[56]:

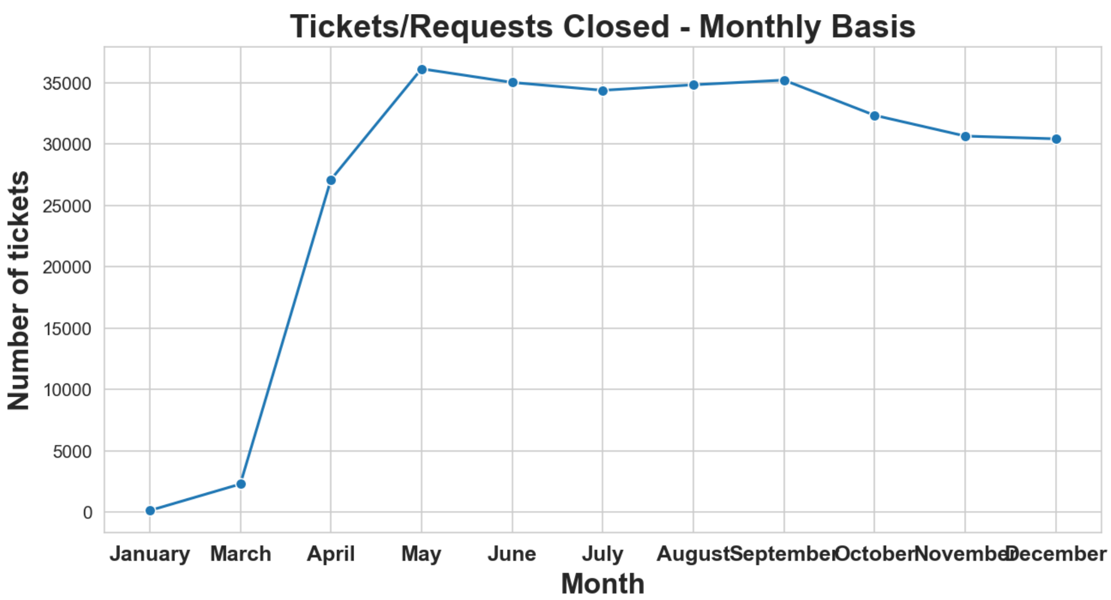
Closed_Date_Month	Unique Key
January	126
March	2286
April	27101
May	36132
June	35021
July	34378
August	34827
September	35210
October	32342
November	30650
December	30422

[57]:
```
plt.figure(figsize=(10,5), dpi=150)
sns.barplot(x = month_wise_ticket_closed['Unique Key'].index,
            y = month_wise_ticket_closed['Unique Key'].values,
            alpha = 0.8)
plt.title('Tickets/Requests Closed - Monthly Basis',␣
 ↪fontsize=16,fontweight="bold")
plt.ylabel('Number of tickets', fontsize=12,fontweight="bold")
plt.xlabel('Month', fontsize=12,fontweight="bold")
plt.show()
```



Also, let us plot the trend chart, monthwise, to see how the patten of request closure looks like

[58]:
```
plt.figure(figsize=(10,5),dpi=150)
sns.set_style("whitegrid")
sns.lineplot(x = month_wise_ticket_closed['Unique Key'].index,
             y = month_wise_ticket_closed['Unique Key'].values, marker='o')
plt.title('Tickets/Requests Closed - Monthly Basis', fontsize=18,␣
 ↪fontweight="bold")
plt.ylabel('Number of tickets', fontsize=16, fontweight="bold")
plt.xlabel('Month', fontsize=16, fontweight="bold")
plt.xticks(fontsize=12, fontweight="bold")
plt.show()
```



[59]:
```
month_wise_ticket_closed = pd.DataFrame(month_wise_ticket_closed).
 ↪sort_values(by = 'Unique Key', ascending=False)
month_wise_ticket_closed['Percent of Requests'] =␣
 ↪((month_wise_ticket_closed['Unique Key']/
 ↪sum(month_wise_ticket_closed['Unique Key']))*100).round(2)
month_wise_ticket_closed['Cumulative Percentage'] =␣
 ↪month_wise_ticket_closed['Percent of Requests'].cumsum()
month_wise_ticket_closed
```

[59]:

	Unique Key	Percent of Requests	Cumulative Percentage
Closed_Date_Month			
May	36132	12.10	12.10
September	35210	11.80	23.90
June	35021	11.73	35.63
August	34827	11.67	47.30
July	34378	11.52	58.82
October	32342	10.84	69.66
November	30650	10.27	79.93
December	30422	10.19	90.12
April	27101	9.08	99.20
March	2286	0.77	99.97
January	126	0.04	100.01

[60]:
```
# month having maximum requests/complains closed
month_wise_ticket_closed[month_wise_ticket_closed['Unique Key'] ==␣
 ↪month_wise_ticket_closed['Unique Key'].max()].index[0]
```

[60]: 'May'

```
[61]: # month having minimum requests/complains closed
month_wise_ticket_closed[month_wise_ticket_closed['Unique Key'] ==␣
↪month_wise_ticket_closed['Unique Key'].min()].index[0]
```

[61]: 'January'

```
[62]: # top 75% requests were closed in the following month
month_wise_ticket_closed[month_wise_ticket_closed['Cumulative Percentage'] <=␣
↪75].index.to_list()
```

[62]: ['May', 'September', 'June', 'August', 'July', 'October']

1.3.3 Analysis - Number of requests 'Created' on Monthly Basis

- Requests were closed in the following months:
  - January
  - March
  - April
  - May
  - June
  - July
  - August
  - September
  - October
  - November
  - December
- No ticket/requests were closed in the month of February, as per the given data.
- Maximum requests were closed in the month of 'May'
- Minimum requests were closed in the month of 'January'
- From the trend chart, we observe that lowest number of requests were closed in the month of 'January'.
- Then, gradual increase in the ticket/requests closure rose upto highest mark in the month of 'May'.
- From month 'May' to 'September', the closure count was observed as almost constant.
- Very slowly, after 'September', the count of ticket/requeest closure dropped till month 'December'.
- Top 75% requests were closed in the month of 'May', 'September', 'June', 'August', 'July', 'October'.

Analysing the time elapsed between requests created and closed

```
[63]: # Distribution of closing time
customerSR[['Request_Closing_Time']].dtypes
```

```
dist_of_closing_time = customerSR['Request_Closing_Time'].describe().
↪apply(lambda x: format(x, 'f'))
dist_of_closing_time = pd.DataFrame(dist_of_closing_time)
dist_of_closing_time
```

[63]:

```
      Request_Closing_Time
count       298495.000000
mean         15532.086604
std          21923.264767
min             60.000000
25%           4593.000000
50%           9775.000000
75%          19260.000000
max        2134342.000000
```

defined a function to change number of seconds to time, days-hours-minutes-seconds format

[64]:

```
def seconds_to_time(sec):
    res = datetime.timedelta(seconds = sec)
    return res
```

Since the no. of seconds are not conveying any information about 'Request_Closing_Time', adding more column to understand time better

[65]:

```
dist_of_closing_time['Time_in_Days']  = list(map(lambda x: seconds_to_time(x),
                                                                                ␣
↪dist_of_closing_time['Request_Closing_Time'].astype(float)))
dist_of_closing_time['Time_in_Hours'] = dist_of_closing_time['Time_in_Days']/np.
↪timedelta64(1, 'h')
dist_of_closing_time['Time_in_Hours'] = np.
↪round(dist_of_closing_time['Time_in_Hours'],2)
```

[66]:

```
dist_of_closing_time
```

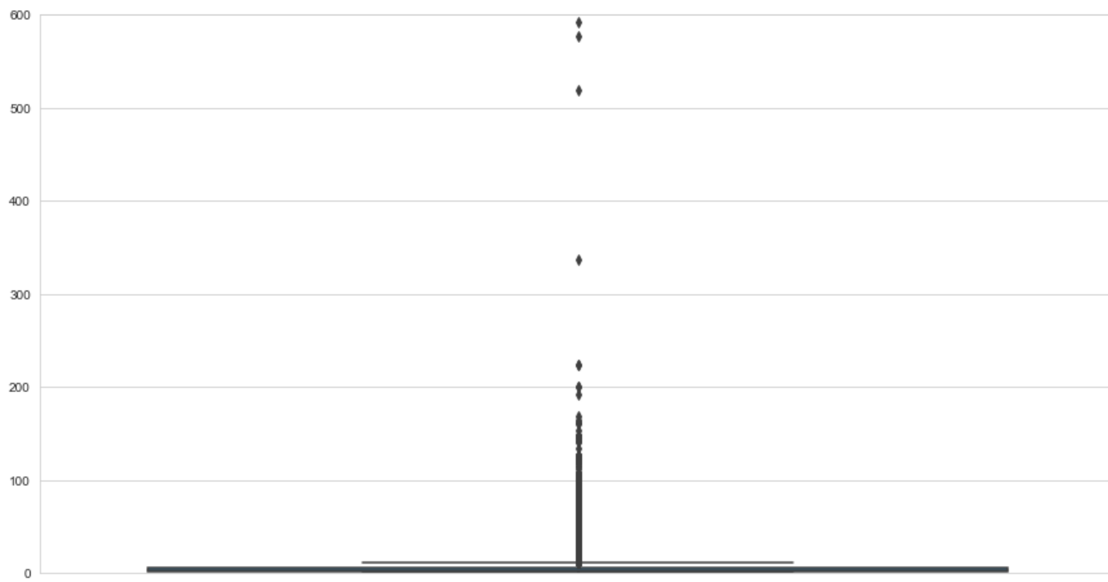
[66]:

```
      Request_Closing_Time            Time_in_Days  Time_in_Hours
count       298495.000000          3 days 10:54:55          82.92
mean         15532.086604 0 days 04:18:52.086604           4.31
std          21923.264767 0 days 06:05:23.264767           6.09
min             60.000000          0 days 00:01:00           0.02
25%           4593.000000          0 days 01:16:33           1.28
50%           9775.000000          0 days 02:42:55           2.72
75%          19260.000000          0 days 05:21:00           5.35
max        2134342.000000         24 days 16:52:22         592.87
```

- So the mean time to serve any request is 04 hours, with standard deviation of 06 hours
- The mean is almost double of the median, indicating there must be outliars

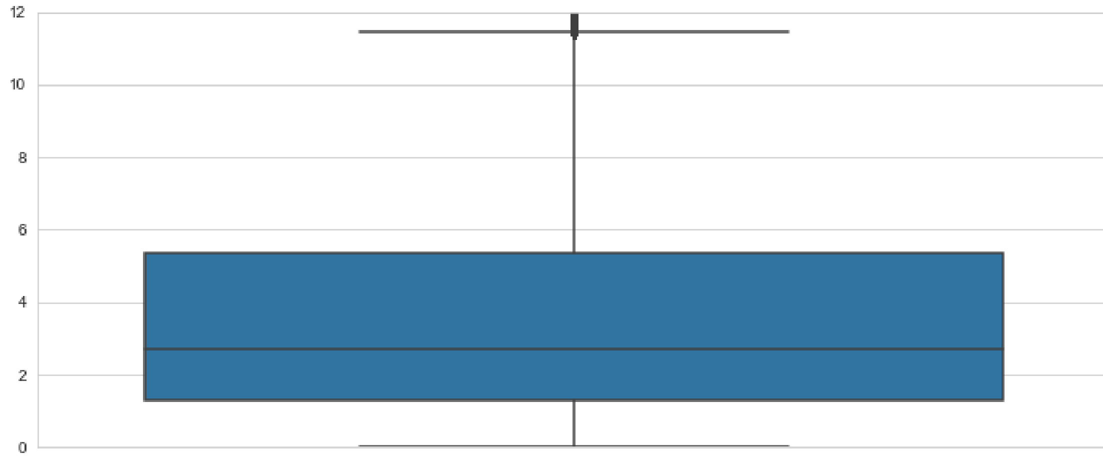
Plotting boxplot to see if there are any outliars

[67]:
```
plt.figure(figsize=(15,8))
#sns.set_style("whitegrid")
sns.boxplot(y=customerSR['Request_Closing_Time_tot_hrs'].values)
plt.ylim(0, 600)
plt.show()
```



In the above figure, the boxplot is barely visible. Though we see lot of dots, indicating outliars, however, the boplot is not clear. Let us try to decrease the ylim to lower value

[68]:
```
plt.figure(figsize=(12,5))
#sns.set_style("whitegrid")
sns.boxplot(y=customerSR['Request_Closing_Time_tot_hrs'].values)
plt.ylim(0, 12)
plt.show()
```



Boxplot is now clear. There are lot of outliars in our data. Let us use z-scores to figure th count of outliars in closing time.

[69]:
```
customerSR['Request_Closing_Time'].describe().apply(lambda x: format(x, 'f'))
```

[69]:
```
count     298495.000000
mean       15532.086604
std        21923.264767
min           60.000000
25%         4593.000000
50%         9775.000000
75%        19260.000000
max      2134342.000000
Name: Request_Closing_Time, dtype: object
```

Calculating the number of outliars in the column

[70]:
```
z = np.abs(stats.zscore(customerSR['Request_Closing_Time']))
threshold = 2.7
rownums = np.where(z > threshold)
rownums[0]
```

[70]:
```
array([    24,     33,    937, …, 297683, 298029, 298149], dtype=int64)
```

[71]:
```
len(rownums[0])
```

[71]:
```
5087
```

- considering 2.7 as the threshold (2.7 std deviations from mean), we have around 5087 values as outliars

[72]:
```
Q1 = customerSR['Request_Closing_Time'].quantile(0.25)
Q3 = customerSR['Request_Closing_Time'].quantile(0.75)
IQR = Q3 - Q1
print(seconds_to_time(IQR))
```

```
4:04:27
```

- because we have lot of outliars in our data,
- considering median and IQR, time taken to close any request on an average is 02 hours 42 minutes
- with 04 hours 4 minutes of deviation (IQR)
- some request exceptionally take longer times, maximum exceeding upto 24 days

[73]:
```
customerSR['Request_Closing_Days_Time'] = customerSR['Request_Closing_Time']*np.
↪timedelta64(1, 's')
```

[74]:
```
customerSR['Request_Closing_Days_Time'].head()
```

[74]:
```
0   0 days 00:55:15
1   0 days 01:26:16
2   0 days 04:51:31
3   0 days 07:45:14
4   0 days 03:27:02
Name: Request_Closing_Days_Time, dtype: timedelta64[ns]
```

Creating few columns, extracting days, hours, minutes for better analysis

[75]:
```
# closing_time['Closing_time_days'] = closing_time['Request_Closing_Time'].dt.
↪days
customerSR['Closing_time_hours'] = customerSR['Request_Closing_Days_Time'].dt.
↪components.hours
customerSR['Closing_time_mins'] = customerSR['Request_Closing_Days_Time'].dt.
↪components.minutes
customerSR['Nearest Days'] = customerSR['Request_Closing_Days_Time'].dt.
↪round('d')
customerSR['Closing_days'] = customerSR['Nearest Days'].dt.days
```

[76]:
```
customerSR[['Request_Closing_Time','Closing_days','Closing_time_hours','Closing_time_mins']].
↪head()
```

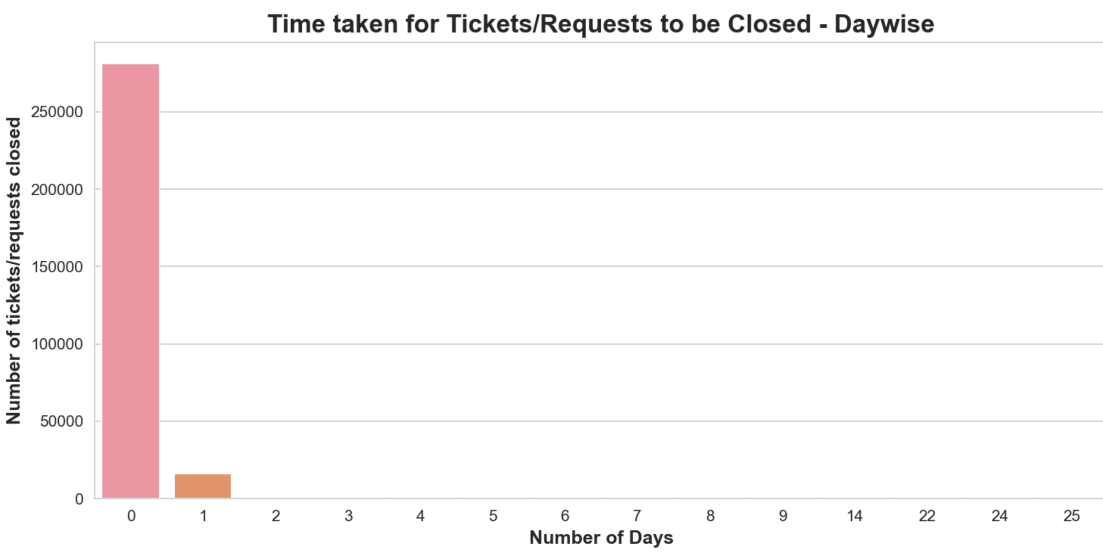
[76]:

	Request_Closing_Time	Closing_days	Closing_time_hours	Closing_time_mins
0	3315.0	0	0	55
1	5176.0	0	1	26
2	17491.0	0	4	51
3	27914.0	0	7	45
4	12422.0	0	3	27

[77]:
```
tickets_closed_days = customerSR['Closing_days'].value_counts()
tickets_closed_days
```

[77]:
```
0     281050
1      16031
2       1074
3        216
4         58
5         38
6         14
7          5
8          3
9          2
14         1
22         1
24         1
25         1
Name: Closing_days, dtype: int64
```

[78]:
```
plt.figure(figsize=(10,5),dpi=150)
#tickets_closed_days.plot.bar()
sns.barplot(x = tickets_closed_days.index,
           y = tickets_closed_days.values)
plt.title('Time taken for Tickets/Requests to be Closed - Daywise',␣
 ↪fontsize=16,fontweight="bold")
plt.ylabel('Number of tickets/requests closed', fontsize=12,fontweight="bold")
plt.xlabel('Number of Days', fontsize=12,fontweight="bold")
plt.xticks(rotation=0)
plt.tight_layout()
```



[79]:
```
# number of tickets/requests which took longer time to close
sum(tickets_closed_days[tickets_closed_days.index >= 14])
```

[79]: 4

[80]:
```
# Unique Keys of the requests which took exceptionally longer time to close, >␣
↪14 days
customerSR[customerSR['Closing_days']>=14][['Unique Key','Complaint Type']]
```

[80]:

	Unique Key	Complaint Type
21268	32167187	Animal Abuse
23664	32154771	Illegal Parking
244488	30684975	Noise - Street/Sidewalk
283132	30427220	Animal in a Park

[81]:
```
(tickets_closed_days/sum(tickets_closed_days))*100
```

[81]:
```
0     94.155681
1      5.370609
2      0.359805
3      0.072363
4      0.019431
5      0.012731
6      0.004690
7      0.001675
8      0.001005
9      0.000670
14     0.000335
22     0.000335
24     0.000335
25     0.000335
Name: Closing_days, dtype: float64
```

1.3.4 Analysis - Time elapsed to resolve the requests/tickets.

- Maximum number of tickets are closed within a day.
- 99.5% tickets are closed between 0-1 days.
- Remaining exceptional 0.5% tickets are closed between 2 days to 25 days.
- There are 4 requests which took more than 14 days to resolve.
- Unique Keys of the requests which took exceptionally longer time to close, > 14 days, along with the 'Complaint Type', or complaint category.
  - 21268: 'Animal Abuse'
  - 23664:'Illegal Parking'
  - 244488:'Noise - Street/Sidewalk'
  - 283132: 'Animal in a Park'

Understanding Citywise Closing time of the requests

[82]:
```
customerSR[['Unique Key','Request_Closing_Time','Location Type','City']].head()
```

[82]:
```
   Unique Key  Request_Closing_Time    Location Type      City
0    32310363                3315.0  Street/Sidewalk  NEW YORK
1    32309934                5176.0  Street/Sidewalk   ASTORIA
2    32309159               17491.0  Street/Sidewalk     BRONX
3    32305098               27914.0  Street/Sidewalk     BRONX
4    32306529               12422.0  Street/Sidewalk  ELMHURST
```

[83]:
```
customerSR[['Unique Key','Request_Closing_Time','Location Type','City']].
 ↪isnull().sum()
```

[83]:
```
Unique Key                0
Request_Closing_Time      0
Location Type            91
City                    506
dtype: int64
```

imputing missing cities and Location Type with mode (most occuring categories)

[84]:
```
mode = customerSR.filter(['Location Type','City']).mode()
customerSR[['Location Type','City']]=customerSR[['Location Type','City']].
 ↪fillna(mode.iloc[0])
```

[85]:
```
customerSR[['Unique Key','Request_Closing_Time','Location Type','City']].
 ↪isnull().sum()
```

[85]:
```
Unique Key              0
Request_Closing_Time    0
Location Type           0
City                    0
dtype: int64
```

[86]:
```
customerSR['City'].unique()
```

[86]:
```
array(['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',
'Astoria', 'Long Island City', 'Woodside', 'East Elmhurst',
'Howard Beach'], dtype=object)
```

[87]:
```
citywise_avg_closing_time = customerSR.groupby('City').
 ↪agg('mean')['Request_Closing_Time'].sort_values(ascending=True)
```

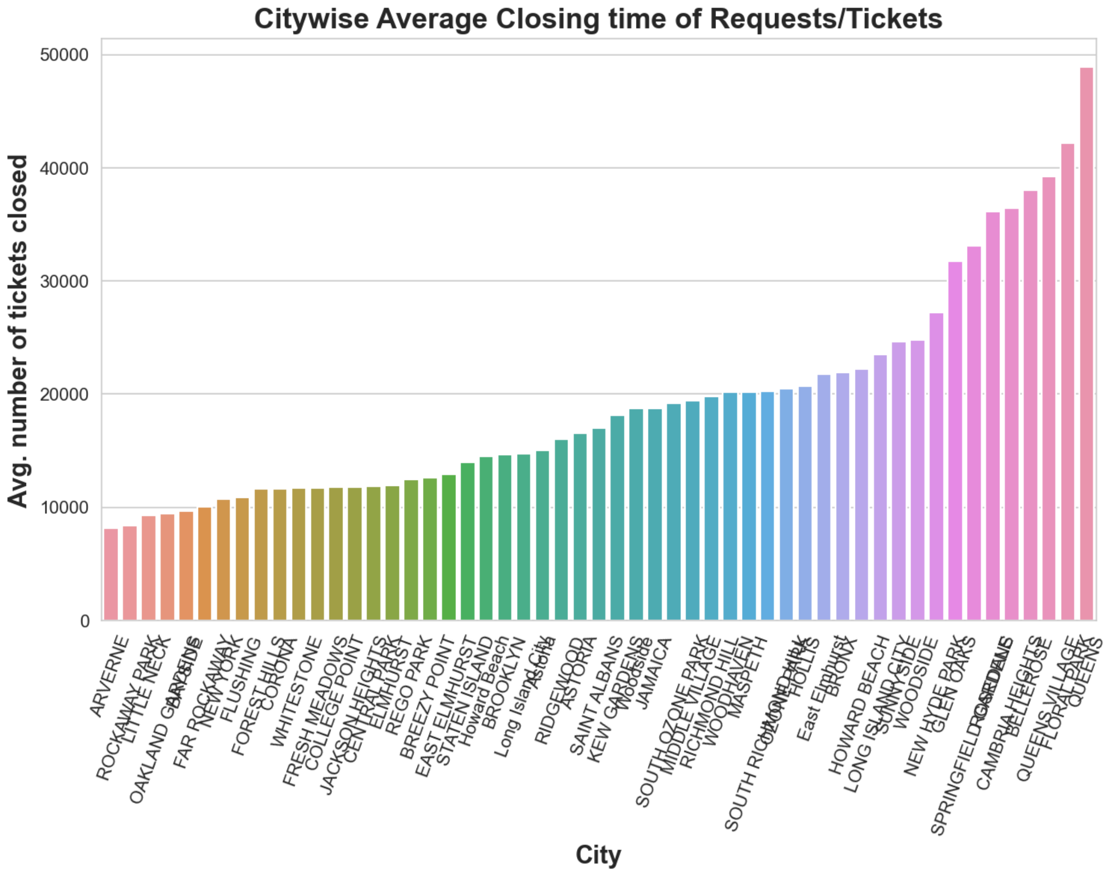
[88]:
```
len(citywise_avg_closing_time)
```

[88]:
```
53
```

[89]:
```
citywise_avg_closing_time.head()
```

[89]:
```
City
ARVERNE             8153.736364
ROCKAWAY PARK       8348.024161
LITTLE NECK         9279.618962
OAKLAND GARDENS     9471.188748
BAYSIDE             9645.599509
Name: Request_Closing_Time, dtype: float64
```

[90]:
```
plt.figure(figsize=(10,6),dpi=150)
sns.barplot(x = citywise_avg_closing_time.index, y = citywise_avg_closing_time.
 ↪values)
plt.title('Citywise Average Closing time of Requests/Tickets',␣
 ↪fontsize=16,fontweight="bold")
plt.ylabel('Avg. number of tickets closed', fontsize=14,fontweight="bold")
plt.xlabel('City', fontsize=14,fontweight="bold")
plt.xticks(rotation=70)
plt.show()
```



Converting number of seconds to days, hh:mm:ss to get more insight

[91]:
```
citywise_avg_closing_time = citywise_avg_closing_time.apply(lambda x:␣
↪seconds_to_time(x))
```

[92]:
```
citywise_avg_closing_time
```

[92]:
```
City
ARVERNE                  0 days 02:15:53.736364
ROCKAWAY PARK            0 days 02:19:08.024161
LITTLE NECK              0 days 02:34:39.618962
OAKLAND GARDENS          0 days 02:37:51.188748
BAYSIDE                  0 days 02:40:45.599509
FAR ROCKAWAY             0 days 02:47:23.986429
NEW YORK                 0 days 02:58:21.843718
FLUSHING                 0 days 03:01:04.909548
FOREST HILLS             0 days 03:13:26.941943
CORONA                   0 days 03:13:42.617373
WHITESTONE               0 days 03:14:41.330601
```

```
FRESH MEADOWS          0 days 03:15:33.269900
COLLEGE POINT          0 days 03:16:25.070492
JACKSON HEIGHTS        0 days 03:16:25.197867
CENTRAL PARK           0 days 03:17:39.515464
ELMHURST               0 days 03:18:38.449102
REGO PARK              0 days 03:27:39.940108
BREEZY POINT           0 days 03:29:47.366667
EAST ELMHURST          0 days 03:34:39.582510
STATEN ISLAND          0 days 03:52:48.963032
Howard Beach                  0 days 04:01:45
BROOKLYN               0 days 04:04:25.881500
Long Island City       0 days 04:06:02.731343
Astoria                0 days 04:11:04.578212
RIDGEWOOD              0 days 04:26:30.456800
ASTORIA                0 days 04:35:56.086730
SAINT ALBANS           0 days 04:43:15.125899
KEW GARDENS            0 days 05:02:34.713359
Woodside                      0 days 05:12:05
JAMAICA                0 days 05:12:46.395607
SOUTH OZONE PARK       0 days 05:19:40.719742
MIDDLE VILLAGE         0 days 05:23:05.854958
RICHMOND HILL          0 days 05:29:39.516824
WOODHAVEN              0 days 05:35:48.759545
MASPETH                0 days 05:35:59.148314
SOUTH RICHMOND HILL    0 days 05:37:02.952055
OZONE PARK             0 days 05:40:51.822142
HOLLIS                 0 days 05:45:36.609684
East Elmhurst          0 days 06:02:52.071429
BRONX                  0 days 06:05:47.025237
HOWARD BEACH           0 days 06:09:39.137487
LONG ISLAND CITY       0 days 06:32:16.464887
SUNNYSIDE              0 days 06:51:07.219917
WOODSIDE               0 days 06:53:41.781259
NEW HYDE PARK          0 days 07:33:21.938776
GLEN OAKS              0 days 08:48:56.633987
SPRINGFIELD GARDENS    0 days 09:11:34.403628
ROSEDALE               0 days 10:01:52.053145
CAMBRIA HEIGHTS        0 days 10:07:25.593291
BELLEROSE              0 days 10:33:23.194667
QUEENS VILLAGE         0 days 10:54:24.676406
FLORAL PARK            0 days 11:43:10.276316
QUEENS                 0 days 13:35:35.187500
Name: Request_Closing_Time, dtype: timedelta64[ns]
```

[93]:
```
# since the average time to resolve the requests is lesser then 1 day,␣
↪converting it into hours
citywise_avg_closing_time = citywise_avg_closing_time/np.timedelta64(1, 'h')
```

```
citywise_avg_closing_time
```

[93]:
```
City
ARVERNE                 2.264927
ROCKAWAY PARK           2.318896
LITTLE NECK             2.577672
OAKLAND GARDENS         2.630886
BAYSIDE                 2.679333
FAR ROCKAWAY            2.789996
NEW YORK                2.972734
FLUSHING                3.018030
FOREST HILLS            3.224151
CORONA                  3.228505
WHITESTONE              3.244814
FRESH MEADOWS           3.259242
COLLEGE POINT           3.273631
JACKSON HEIGHTS         3.273666
CENTRAL PARK            3.294310
ELMHURST                3.310680
REGO PARK               3.461094
BREEZY POINT            3.496491
EAST ELMHURST           3.577662
STATEN ISLAND           3.880268
Howard Beach            4.029167
BROOKLYN                4.073856
Long Island City        4.100759
Astoria                 4.184605
RIDGEWOOD               4.441794
ASTORIA                 4.598913
SAINT ALBANS            4.720868
KEW GARDENS             5.042976
Woodside                5.201389
JAMAICA                 5.212888
SOUTH OZONE PARK        5.327978
MIDDLE VILLAGE          5.384960
RICHMOND HILL           5.494310
WOODHAVEN               5.596878
MASPETH                 5.599763
SOUTH RICHMOND HILL     5.617487
OZONE PARK              5.681062
HOLLIS                  5.760169
East Elmhurst           6.047798
BRONX                   6.096396
HOWARD BEACH            6.160872
LONG ISLAND CITY        6.537907
SUNNYSIDE               6.852006
WOODSIDE                6.894939
```

```
NEW HYDE PARK           7.556094
GLEN OAKS               8.815732
SPRINGFIELD GARDENS     9.192890
ROSEDALE               10.031126
CAMBRIA HEIGHTS        10.123776
BELLEROSE              10.556443
QUEENS VILLAGE         10.906855
FLORAL PARK            11.719521
QUEENS                 13.593108
Name: Request_Closing_Time, dtype: float64
```

[94]:
```
# top 5 cities with minimum average time taken to resolve requests
citywise_avg_closing_time.head(5).index.to_list()
```

[94]:
```
['ARVERNE', 'ROCKAWAY PARK', 'LITTLE NECK', 'OAKLAND GARDENS', 'BAYSIDE']
```

[95]:
```
# top 5 cities with maximum average time taken to resolve requests
citywise_avg_closing_time.tail(5).index.to_list()
```

[95]:
```
['CAMBRIA HEIGHTS', 'BELLEROSE', 'QUEENS VILLAGE', 'FLORAL PARK', 'QUEENS']
```

[96]:
```
# Range of the Average time to resolve the request
round(citywise_avg_closing_time.max() - citywise_avg_closing_time.min(),2)
```

[96]:
```
11.33
```

- We have 53 total cities and their average time to resolve the requests -top 5 cities with minimum average time taken to resolve requests 'ARVERNE', 'ROCKAWAY PARK', 'LITTLE NECK', 'OAKLAND GARDENS', 'BAYSIDE'
- top 5 cities with maximum average time taken to resolve requests 'CAMBRIA HEIGHTS', 'BELLEROSE', 'QUEENS VILLAGE', 'FLORAL PARK', 'QUEENS'
- minimum average time taken to resolve the requests is 2.26 hours, and its happens in the city 'ARVERNE'
- maximum average time taken to resolve the requests is 13.59 hours, and its happens in the city 'Queens'

1.4 Task 4

1.4.1 Order the complaint types based on the average 'Request_Closing_Time', grouping them for different locations.

[97]:
```
location_based_complaints = customerSR.groupby('Location Type').
 ↪agg('count')['Unique Key'].sort_values()
location_based_complaints
```

```
[97]: Location Type
Park                              1
Bridge                            2
Subway Station                   34
Roadway Tunnel                   35
Commercial                       62
Vacant Lot                       77
House and Store                  93
Parking Lot                     117
Highway                         214
Residential Building            227
House of Worship                927
Park/Playground                4751
Residential Building/House     6953
Club/Bar/Restaurant           17227
Store/Commercial              20183
Street/Sidewalk              247592
Name: Unique Key, dtype: int64
```

Converting it into percentage to get more insight

```
[98]: percent_location_based_complaints = round((location_based_complaints/
       ↪sum(location_based_complaints))*100,2)
      percent_location_based_complaints
```

```
[98]: Location Type
Park                           0.00
Bridge                         0.00
Subway Station                 0.01
Roadway Tunnel                 0.01
Commercial                     0.02
Vacant Lot                     0.03
House and Store                0.03
Parking Lot                    0.04
Highway                        0.07
Residential Building           0.08
House of Worship               0.31
Park/Playground                1.59
Residential Building/House     2.33
Club/Bar/Restaurant            5.77
Store/Commercial               6.76
Street/Sidewalk               82.95
Name: Unique Key, dtype: float64
```

- 83% of the compaints are for location type Street/Sidewalk

Let us club the complains lesser than 0.5% as 'Others'

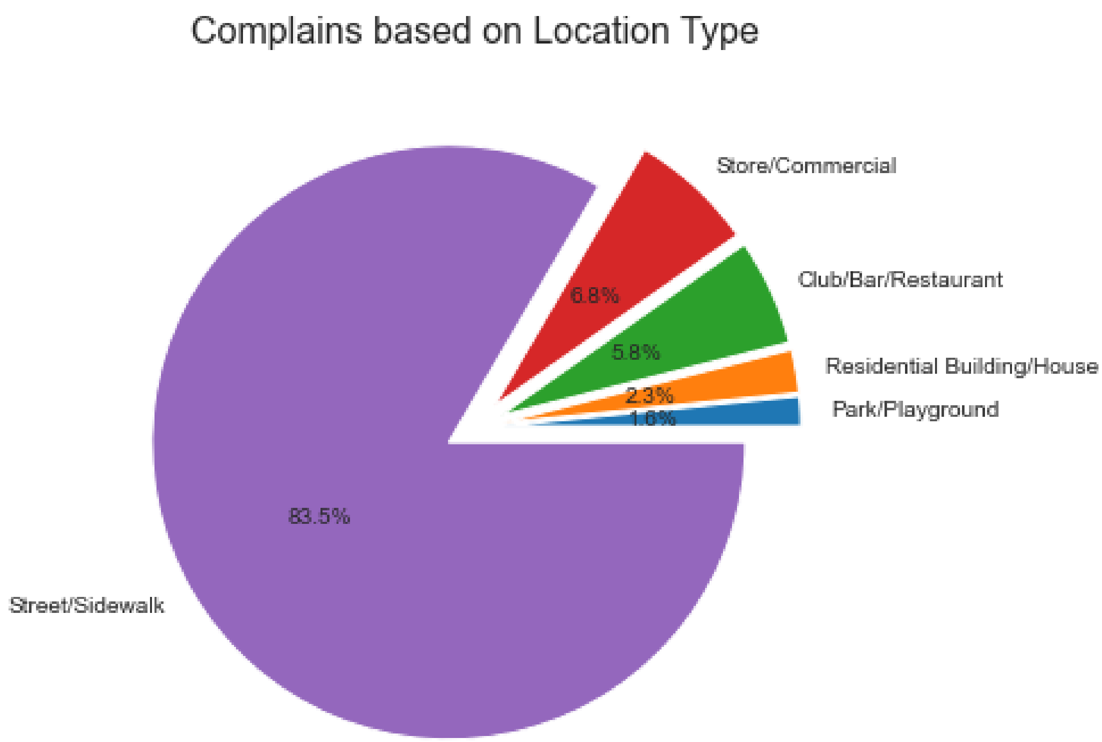
[99]:
```
percent_location_based_complaints_new =␣
↪percent_location_based_complaints[percent_location_based_complaints>0.5]
percent_location_based_complaints_new['Others'] =␣
↪sum(percent_location_based_complaints[percent_location_based_complaints<=0.
↪5])
percent_location_based_complaints_new = percent_location_based_complaints_new.
↪sort_values()
percent_location_based_complaints_new
```

[99]:
```
Location Type
Others                          0.60
Park/Playground                 1.59
Residential Building/House      2.33
Club/Bar/Restaurant             5.77
Store/Commercial                6.76
Street/Sidewalk                82.95
Name: Unique Key, dtype: float64
```

Since 'Others' category is having value less than 1%, dropping it

[100]:
```
percent_location_based_complaints_new =␣
↪percent_location_based_complaints_new[percent_location_based_complaints_new>1]
```

[101]:
```
pie, ax = plt.subplots(figsize=[12,6])
labels = percent_location_based_complaints_new.index
plt.pie(x=percent_location_based_complaints_new, autopct="%.1f%%",explode=[0.
↪1]*len(percent_location_based_complaints_new), labels=labels, pctdistance=0.
↪5)
plt.title("Complains based on Location Type", fontsize=16);
plt.show()
```



[102]:
```
# top 3 'Location Types' from where the complains are maximum
percent_location_based_complaints_new.tail(3).index.to_list()[::-1]
```

[102]:
```
['Street/Sidewalk', 'Store/Commercial', 'Club/Bar/Restaurant']
```

1.5 Task 4

1.5.1 Statistical Tests

1.5.2 Task 4.1

1.5.3 Whether the average response time across complaint types is similar or not (overall)

[103]:
```
customerSR.columns
```

[103]:
```
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',
       'Address Type', 'City', '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', 'Latitude', 'Longitude', 'Location',
         'Request_Closing_Time', 'Request_Closing_Time_tot_hrs',
         'Created_Date_Month_num', 'Created_Date_Year', 'Created_Date_Month',
         'Closed_Date_Month_num', 'Closed_Date_Year', 'Closed_Date_Month',
         'Request_Closing_Days_Time', 'Closing_time_hours', 'Closing_time_mins',
         'Nearest Days', 'Closing_days'],
        dtype='object')
```

[104]:
```
customerSR[['Unique Key','Complaint Type','Request_Closing_Time']].head()
```

[104]:

	Unique Key	Complaint Type	Request_Closing_Time
0	32310363	Noise - Street/Sidewalk	3315.0
1	32309934	Blocked Driveway	5176.0
2	32309159	Blocked Driveway	17491.0
3	32305098	Illegal Parking	27914.0
4	32306529	Illegal Parking	12422.0

[105]:
```
# checking if there are any null values
customerSR[['Unique Key','Complaint Type','Request_Closing_Time']].isnull().
 ↪sum()
```

[105]:
```
Unique Key              0
Complaint Type          0
Request_Closing_Time    0
dtype: int64
```

[106]:
```
complaint_types_list = list(customerSR['Complaint Type'].unique())
print(f'There are {len(complaint_types_list)} unique complaint types')
```

```
There are 23 unique complaint types
```

1.5.4 Whether the average response time across complaint types is similar or not (overall)

- Null Hypothesis: Average response time of all the compaint types are similar (equal)
- Alternate Hypothesis: At least one of the average response time of the compaint types is different

*** In order to perform ANOVA, I would need Complaint Types and the respective data points (Request Closed Time)**

*** So, I am creating dictionary, such that Complaint Type category is key and data points are values**

[107]:
```
compliant_type_datapoints = {}
for complaint in complaint_types_list:
    compliant_type_datapoints[complaint]=list(customerSR[customerSR['Complaint␣
↪Type']==complaint]['Request_Closing_Time'])
```

[108]:
```
compliant_type_datapoints.keys()
```

[108]:
```
dict_keys(['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', 'Agency Issues', 'Squeegee', 'Animal in a Park'])
```

[109]:
```
stats.f_oneway(
compliant_type_datapoints['Noise - Street/Sidewalk'],
compliant_type_datapoints['Blocked Driveway'],
compliant_type_datapoints['Illegal Parking'],
compliant_type_datapoints['Derelict Vehicle'],
compliant_type_datapoints['Noise - Commercial'],
compliant_type_datapoints['Noise - House of Worship'],
compliant_type_datapoints['Posting Advertisement'],
compliant_type_datapoints['Noise - Vehicle'],
compliant_type_datapoints['Animal Abuse'],
compliant_type_datapoints['Vending'],
compliant_type_datapoints['Traffic'],
compliant_type_datapoints['Drinking'],
compliant_type_datapoints['Bike/Roller/Skate Chronic'],
compliant_type_datapoints['Panhandling'],
compliant_type_datapoints['Noise - Park'],
compliant_type_datapoints['Homeless Encampment'],
compliant_type_datapoints['Urinating in Public'],
compliant_type_datapoints['Graffiti'],
compliant_type_datapoints['Disorderly Youth'],
compliant_type_datapoints['Illegal Fireworks'],
compliant_type_datapoints['Agency Issues'],
compliant_type_datapoints['Squeegee'],
compliant_type_datapoints['Animal in a Park']
)
```

[109]:
```
F_onewayResult(statistic=514.1217802705925, pvalue=0.0)
```

- Since p value is low, 0, in our case, we reject Null Hypotheses
- This implies that
- At least one of the average response time of the compaint types is different

This is also evident from the below code:

[110]:
```
customerSR.groupby('Complaint Type').agg('mean')['Request_Closing_Time'].
↪apply(lambda x: seconds_to_time(x))
```

[110]:
```
Complaint Type
Agency Issues                0 days 05:15:37.166667
Animal Abuse                 0 days 05:12:47.677781
Animal in a Park                   14 days 00:50:05
Bike/Roller/Skate Chronic    0 days 03:45:59.264151
Blocked Driveway             0 days 04:44:27.346727
Derelict Vehicle             0 days 07:21:52.617878
Disorderly Youth             0 days 03:33:30.902098
Drinking                     0 days 03:51:42.598431
Graffiti                     0 days 07:09:04.504425
Homeless Encampment          0 days 04:22:01.862936
Illegal Fireworks            0 days 02:45:40.101190
Illegal Parking              0 days 04:30:03.624670
Noise - Commercial           0 days 03:08:49.749014
Noise - House of Worship     0 days 03:11:35.874058
Noise - Park                 0 days 03:24:32.394429
Noise - Street/Sidewalk      0 days 03:26:43.428934
Noise - Vehicle              0 days 03:35:20.434124
Panhandling                  0 days 04:22:21.963934
Posting Advertisement        0 days 01:58:33.582689
Squeegee                     0 days 04:02:44.250000
Traffic                      0 days 03:27:00.082573
Urinating in Public          0 days 03:37:35.991554
Vending                      0 days 04:00:51.161877
Name: Request_Closing_Time, dtype: timedelta64[ns]
```

- From the above series, we see that 'Animal in a Park' complaint type has an average request closing time
- of 14 days, compared to other compalint types, which is having average of lesser than a day

Are the type of complaint or service requested and location related?

- Null Hypothesis: There is no relationship between the 'Complaint Type' and 'Location Type'
- Alternate Hypothesis: There is a relationship between the 'Complaint Type' and 'Location Type'

[111]:
```
customerSR[['Complaint Type','Location Type']].dtypes
```

[111]:
```
Complaint Type    object
Location Type     object
dtype: object
```

'Complaint Type' and 'Location Type' are boject types. We will need to use Chi Square test to check if there is any relationship between them

[112]:
```
# Checking if there are any null values
customerSR[['Complaint Type','Location Type']].isnull().any()
```

[112]:
```
Complaint Type    False
Location Type     False
dtype: bool
```

[113]:
```
customerSR[['Complaint Type','Location Type']].head()
```

[113]:
```
            Complaint Type    Location Type
0  Noise - Street/Sidewalk  Street/Sidewalk
1         Blocked Driveway  Street/Sidewalk
2         Blocked Driveway  Street/Sidewalk
3          Illegal Parking  Street/Sidewalk
4          Illegal Parking  Street/Sidewalk
```

Creating a cross-table between 'Complaint Types' and 'Location Type'

[114]:
```
table = customerSR.pivot_table(index = 'Complaint Type', columns='Location␣
↪Type', aggfunc='count')['Unique Key']
table
```

[114]:
```
Location Type              Bridge  Club/Bar/Restaurant  Commercial  Highway  \
Complaint Type
Agency Issues                 NaN                  NaN         NaN      NaN
Animal Abuse                  NaN                  NaN        62.0      NaN
Animal in a Park              NaN                  NaN         NaN      NaN
Bike/Roller/Skate Chronic     NaN                  NaN         NaN      NaN
Blocked Driveway              NaN                  NaN         NaN      NaN
Derelict Vehicle              NaN                  NaN         NaN     13.0
Disorderly Youth              NaN                  NaN         NaN      NaN
Drinking                      NaN                365.0         NaN      NaN
Graffiti                      NaN                  NaN         NaN      NaN
Homeless Encampment           2.0                  NaN         NaN     15.0
Illegal Fireworks             NaN                  NaN         NaN      NaN
Illegal Parking               NaN                  NaN         NaN      NaN
Noise - Commercial            NaN              16841.0         NaN      NaN
Noise - House of Worship      NaN                  NaN         NaN      NaN
Noise - Park                  NaN                  NaN         NaN      NaN
Noise - Street/Sidewalk       NaN                  NaN         NaN      NaN
Noise - Vehicle               NaN                  NaN         NaN      NaN
Panhandling                   NaN                  NaN         NaN      NaN
Posting Advertisement         NaN                  NaN         NaN      NaN
Squeegee                      NaN                  NaN         NaN      NaN
Traffic                       NaN                  NaN         NaN    186.0
```

```
Urinating in Public           NaN                21.0         NaN      NaN
Vending                       NaN                 NaN         NaN      NaN

Location Type              House and Store  House of Worship  Park  \
Complaint Type
Agency Issues                          NaN               NaN   NaN
Animal Abuse                          93.0               NaN   NaN
Animal in a Park                       NaN               NaN   1.0
Bike/Roller/Skate Chronic              NaN               NaN   NaN
Blocked Driveway                       NaN               NaN   NaN
Derelict Vehicle                       NaN               NaN   NaN
Disorderly Youth                       NaN               NaN   NaN
Drinking                               NaN               NaN   NaN
Graffiti                               NaN               NaN   NaN
Homeless Encampment                    NaN               NaN   NaN
Illegal Fireworks                      NaN               NaN   NaN
Illegal Parking                        NaN               NaN   NaN
Noise - Commercial                     NaN               NaN   NaN
Noise - House of Worship               NaN             927.0   NaN
Noise - Park                           NaN               NaN   NaN
Noise - Street/Sidewalk                NaN               NaN   NaN
Noise - Vehicle                        NaN               NaN   NaN
Panhandling                            NaN               NaN   NaN
Posting Advertisement                  NaN               NaN   NaN
Squeegee                               NaN               NaN   NaN
Traffic                                NaN               NaN   NaN
Urinating in Public                    NaN               NaN   NaN
Vending                                NaN               NaN   NaN

Location Type              Park/Playground  Parking Lot  Residential Building  \
Complaint Type
Agency Issues                          NaN          NaN                   NaN
Animal Abuse                         122.0        110.0                 227.0
Animal in a Park                       NaN          NaN                   NaN
Bike/Roller/Skate Chronic              NaN          NaN                   NaN
Blocked Driveway                       NaN          NaN                   NaN
Derelict Vehicle                       NaN          NaN                   NaN
Disorderly Youth                       NaN          NaN                   NaN
Drinking                              98.0          NaN                   NaN
Graffiti                               NaN          NaN                   NaN
Homeless Encampment                  353.0          NaN                   NaN
Illegal Fireworks                      8.0          NaN                   NaN
Illegal Parking                        NaN          NaN                   NaN
Noise - Commercial                     NaN          NaN                   NaN
Noise - House of Worship               NaN          NaN                   NaN
Noise - Park                        4021.0          NaN                   NaN
Noise - Street/Sidewalk                NaN          NaN                   NaN
```

Noise - Vehicle	NaN	NaN	NaN
Panhandling	6.0	NaN	NaN
Posting Advertisement	NaN	7.0	NaN
Squeegee	NaN	NaN	NaN
Traffic	NaN	NaN	NaN
Urinating in Public	38.0	NaN	NaN
Vending	105.0	NaN	NaN

Location Type	Residential Building/House	Roadway Tunnel \
Complaint Type		
Agency Issues	NaN	NaN
Animal Abuse	5081.0	NaN
Animal in a Park	NaN	NaN
Bike/Roller/Skate Chronic	25.0	NaN
Blocked Driveway	NaN	NaN
Derelict Vehicle	NaN	5.0
Disorderly Youth	77.0	NaN
Drinking	289.0	NaN
Graffiti	56.0	NaN
Homeless Encampment	983.0	1.0
Illegal Fireworks	33.0	NaN
Illegal Parking	NaN	NaN
Noise - Commercial	NaN	NaN
Noise - House of Worship	NaN	NaN
Noise - Park	NaN	NaN
Noise - Street/Sidewalk	NaN	NaN
Noise - Vehicle	NaN	NaN
Panhandling	16.0	NaN
Posting Advertisement	54.0	NaN
Squeegee	NaN	NaN
Traffic	NaN	29.0
Urinating in Public	138.0	NaN
Vending	201.0	NaN

Location Type	Store/Commercial	Street/Sidewalk	Subway Station \
Complaint Type			
Agency Issues	NaN	6.0	NaN
Animal Abuse	521.0	1530.0	22.0
Animal in a Park	NaN	NaN	NaN
Bike/Roller/Skate Chronic	53.0	346.0	NaN
Blocked Driveway	NaN	76804.0	NaN
Derelict Vehicle	NaN	17491.0	NaN
Disorderly Youth	8.0	201.0	NaN
Drinking	90.0	433.0	NaN
Graffiti	32.0	25.0	NaN
Homeless Encampment	512.0	2548.0	NaN
Illegal Fireworks	2.0	125.0	NaN

```
Illegal Parking                                NaN          74521.0             NaN
Noise - Commercial                         18401.0              3.0             NaN
Noise - House of Worship                       NaN              2.0             NaN
Noise - Park                                   NaN              NaN             NaN
Noise - Street/Sidewalk                        NaN          48068.0             NaN
Noise - Vehicle                                NaN          17032.0             NaN
Panhandling                                   60.0            223.0             NaN
Posting Advertisement                          6.0            580.0             NaN
Squeegee                                       NaN              4.0             NaN
Traffic                                        NaN           4278.0             NaN
Urinating in Public                           66.0            317.0            12.0
Vending                                      432.0           3055.0             NaN

Location Type              Vacant Lot
Complaint Type
Agency Issues                     NaN
Animal Abuse                      NaN
Animal in a Park                  NaN
Bike/Roller/Skate Chronic         NaN
Blocked Driveway                  NaN
Derelict Vehicle                 77.0
Disorderly Youth                  NaN
Drinking                          NaN
Graffiti                          NaN
Homeless Encampment               NaN
Illegal Fireworks                 NaN
Illegal Parking                   NaN
Noise - Commercial                NaN
Noise - House of Worship          NaN
Noise - Park                      NaN
Noise - Street/Sidewalk           NaN
Noise - Vehicle                   NaN
Panhandling                       NaN
Posting Advertisement             NaN
Squeegee                          NaN
Traffic                           NaN
Urinating in Public               NaN
Vending                           NaN
```

[115]:
```
table = table.fillna(0)
table = table.astype(int)
table = table.loc[:, (table != 0).any(axis=0)]
table
```

[115]:
```
Location Type              Bridge  Club/Bar/Restaurant  Commercial  Highway  \
Complaint Type
Agency Issues                   0                    0           0        0
```

Animal Abuse	0	0	62	0
Animal in a Park	0	0	0	0
Bike/Roller/Skate Chronic	0	0	0	0
Blocked Driveway	0	0	0	0
Derelict Vehicle	0	0	0	13
Disorderly Youth	0	0	0	0
Drinking	0	365	0	0
Graffiti	0	0	0	0
Homeless Encampment	2	0	0	15
Illegal Fireworks	0	0	0	0
Illegal Parking	0	0	0	0
Noise - Commercial	0	16841	0	0
Noise - House of Worship	0	0	0	0
Noise - Park	0	0	0	0
Noise - Street/Sidewalk	0	0	0	0
Noise - Vehicle	0	0	0	0
Panhandling	0	0	0	0
Posting Advertisement	0	0	0	0
Squeegee	0	0	0	0
Traffic	0	0	0	186
Urinating in Public	0	21	0	0
Vending	0	0	0	0

Location Type	House and Store	House of Worship	Park \
Complaint Type			
Agency Issues	0	0	0
Animal Abuse	93	0	0
Animal in a Park	0	0	1
Bike/Roller/Skate Chronic	0	0	0
Blocked Driveway	0	0	0
Derelict Vehicle	0	0	0
Disorderly Youth	0	0	0
Drinking	0	0	0
Graffiti	0	0	0
Homeless Encampment	0	0	0
Illegal Fireworks	0	0	0
Illegal Parking	0	0	0
Noise - Commercial	0	0	0
Noise - House of Worship	0	927	0
Noise - Park	0	0	0
Noise - Street/Sidewalk	0	0	0
Noise - Vehicle	0	0	0
Panhandling	0	0	0
Posting Advertisement	0	0	0
Squeegee	0	0	0
Traffic	0	0	0
Urinating in Public	0	0	0

Vending	0	0	0

Location Type	Park/Playground	Parking Lot	Residential Building \
Complaint Type			
Agency Issues	0	0	0
Animal Abuse	122	110	227
Animal in a Park	0	0	0
Bike/Roller/Skate Chronic	0	0	0
Blocked Driveway	0	0	0
Derelict Vehicle	0	0	0
Disorderly Youth	0	0	0
Drinking	98	0	0
Graffiti	0	0	0
Homeless Encampment	353	0	0
Illegal Fireworks	8	0	0
Illegal Parking	0	0	0
Noise - Commercial	0	0	0
Noise - House of Worship	0	0	0
Noise - Park	4021	0	0
Noise - Street/Sidewalk	0	0	0
Noise - Vehicle	0	0	0
Panhandling	6	0	0
Posting Advertisement	0	7	0
Squeegee	0	0	0
Traffic	0	0	0
Urinating in Public	38	0	0
Vending	105	0	0

Location Type	Residential Building/House	Roadway Tunnel \
Complaint Type		
Agency Issues	0	0
Animal Abuse	5081	0
Animal in a Park	0	0
Bike/Roller/Skate Chronic	25	0
Blocked Driveway	0	0
Derelict Vehicle	0	5
Disorderly Youth	77	0
Drinking	289	0
Graffiti	56	0
Homeless Encampment	983	1
Illegal Fireworks	33	0
Illegal Parking	0	0
Noise - Commercial	0	0
Noise - House of Worship	0	0
Noise - Park	0	0
Noise - Street/Sidewalk	0	0
Noise - Vehicle	0	0

Panhandling	16	0
Posting Advertisement	54	0
Squeegee	0	0
Traffic	0	29
Urinating in Public	138	0
Vending	201	0

Location Type	Store/Commercial	Street/Sidewalk	Subway Station \
Complaint Type			
Agency Issues	0	6	0
Animal Abuse	521	1530	22
Animal in a Park	0	0	0
Bike/Roller/Skate Chronic	53	346	0
Blocked Driveway	0	76804	0
Derelict Vehicle	0	17491	0
Disorderly Youth	8	201	0
Drinking	90	433	0
Graffiti	32	25	0
Homeless Encampment	512	2548	0
Illegal Fireworks	2	125	0
Illegal Parking	0	74521	0
Noise - Commercial	18401	3	0
Noise - House of Worship	0	2	0
Noise - Park	0	0	0
Noise - Street/Sidewalk	0	48068	0
Noise - Vehicle	0	17032	0
Panhandling	60	223	0
Posting Advertisement	6	580	0
Squeegee	0	4	0
Traffic	0	4278	0
Urinating in Public	66	317	12
Vending	432	3055	0

Location Type	Vacant Lot
Complaint Type	
Agency Issues	0
Animal Abuse	0
Animal in a Park	0
Bike/Roller/Skate Chronic	0
Blocked Driveway	0
Derelict Vehicle	77
Disorderly Youth	0
Drinking	0
Graffiti	0
Homeless Encampment	0
Illegal Fireworks	0
Illegal Parking	0

```
Noise - Commercial                    0
Noise - House of Worship              0
Noise - Park                          0
Noise - Street/Sidewalk               0
Noise - Vehicle                       0
Panhandling                           0
Posting Advertisement                 0
Squeegee                              0
Traffic                               0
Urinating in Public                   0
Vending                               0
```

[116]:
```
chi_calculated, p_val, degree_f, expected_mat = stats.chi2_contingency(table)
```

[117]:
```
print(p_val)
```

```
0.0
```

Since the p-value is low, 0 in our case, lower than 0.05, we reject Null Hypothesis

There is a relationship between the 'Complaint Type' and 'Location Type'

1.5.5 ========================== End of Document ==========================