```
In [1]: # importing libraries

import pandas as pd
import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

Import dataset

```
In [2]: df = pd.read_csv('311_dataset.csv',low_memory=False)
```

```
In [3]: df.head()
```

Out[3]:

| | Unique Key | Created Date | Closed Date | Agency | Agency Name | Complaint Type | Descriptor | Location Type | Incident Zip | Incident Address | ... | Bridge Highway Name | Bri High Direct |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 32310363 | 12/31/2015 11:59:45 PM | 01-01-16 0:55 | NYPD | New York City Police Department | Noise - Street/Sidewalk | Loud Music/Party | Street/Sidewalk | 10034.0 | 71 VERMILYEA AVENUE | ... | NaN | N |
| 1 | 32309934 | 12/31/2015 11:59:44 PM | 01-01-16 1:26 | NYPD | New York City Police Department | Blocked Driveway | No Access | Street/Sidewalk | 11105.0 | 27-07 23 AVENUE | ... | NaN | N |
| 2 | 32309159 | 12/31/2015 11:59:29 PM | 01-01-16 4:51 | NYPD | New York City Police Department | Blocked Driveway | No Access | Street/Sidewalk | 10458.0 | 2897 VALENTINE AVENUE | ... | NaN | N |
| 3 | 32305098 | 12/31/2015 11:57:46 PM | 01-01-16 7:43 | NYPD | New York City Police Department | Illegal Parking | Commercial Overnight Parking | Street/Sidewalk | 10461.0 | 2940 BAISLEY AVENUE | ... | NaN | N |
| 4 | 32306529 | 12/31/2015 11:56:58 PM | 01-01-16 3:24 | NYPD | New York City Police Department | Illegal Parking | Blocked Sidewalk | Street/Sidewalk | 11373.0 | 87-14 57 ROAD | ... | NaN | N |

5 rows × 53 columns

```
In [4]: # shape of the dataframe
df.shape

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

```
In [5]: # columns in df
df.columns

Out[5]: Index(['Unique Key', 'Created Date', 'Closed Date', 'Agency', 'Agency Name',
       'Complaint Type', 'Descriptor', 'Location Type', 'Incident Zip',
       'Incident Address', 'Street Name', 'Cross Street 1', 'Cross Street 2',
       'Intersection Street 1', 'Intersection Street 2', 'Address Type',
       'City', 'Landmark', 'Facility Type', 'Status', 'Due Date',
       'Resolution Description', 'Resolution Action Updated Date',
       'Community Board', 'Borough', 'X Coordinate (State Plane)',
       'Y Coordinate (State Plane)', 'Park Facility Name', 'Park Borough',
       'School Name', 'School Number', 'School Region', 'School Code',
       'School Phone Number', 'School Address', 'School City', 'School State',
       'School Zip', 'School Not Found', 'School or Citywide Complaint',
       'Vehicle Type', 'Taxi Company Borough', 'Taxi Pick Up Location',
       'Bridge Highway Name', 'Bridge Highway Direction', 'Road Ramp',
       'Bridge Highway Segment', 'Garage Lot Name', 'Ferry Direction',
       'Ferry Terminal Name', 'Latitude', 'Longitude', 'Location'],
      dtype='object')
```

```
In [6]: # converting the columns into lower case and replacing the space with '_'.

df.columns = df.columns.str.lower().str.replace(' ','_')
```

```
In [7]: # checking for NaN or null values in df
df.isna().sum()

Out[7]: unique_key                             0
created_date                           0
closed_date                         2164
agency                                 0
agency_name                            0
complaint_type                         0
descriptor                          5914
location_type                        131
incident_zip                        2615
incident_address                   44410
street_name                        44410
cross_street_1                     49279
cross_street_2                     49779
intersection_street_1             256840
intersection_street_2             257336
address_type                        2815
city                                2614
landmark                          300349
facility_type                       2171
status                                 0
due_date                               3
resolution_description                 0
resolution_action_updated_date      2187
community_board                        0
borough                                0
x_coordinate_(state_plane)          3540
y_coordinate_(state_plane)          3540
park_facility_name                     0
park_borough                           0
school_name                            0
school_number                          0
school_region                          1
school_code                            1
school_phone_number                    0
school_address                         0
school_city                            0
school_state                           0
school_zip                             1
school_not_found                       0
school_or_citywide_complaint      300698
vehicle_type                      300698
taxi_company_borough              300698
taxi_pick_up_location             300698
bridge_highway_name               300455
bridge_highway_direction          300455
road_ramp                         300485
bridge_highway_segment            300485
garage_lot_name                   300698
ferry_direction                   300697
ferry_terminal_name               300696
latitude                            3540
longitude                           3540
location                            3540
dtype: int64
```

```
In [8]: df.nunique()

Out[8]: unique_key                        300698
created_date                      259493
closed_date                       237165
agency                                 1
agency_name                            3
complaint_type                        24
descriptor                            45
location_type                         18
incident_zip                         201
incident_address                  107652
street_name                         7320
cross_street_1                      5982
cross_street_2                      5823
intersection_street_1               4413
intersection_street_2               4172
address_type                           5
city                                  53
landmark                             116
facility_type                          1
status                                 4
due_date                          259851
resolution_description                18
resolution_action_updated_date    237895
community_board                       75
borough                                6
x_coordinate_(state_plane)         63226
y_coordinate_(state_plane)         73694
park_facility_name                     2
park_borough                           6
school_name                            2
school_number                          2
school_region                          1
school_code                            1
school_phone_number                    2
school_address                         2
school_city                            2
school_state                           2
school_zip                             1
school_not_found                       1
school_or_citywide_complaint           0
vehicle_type                           0
taxi_company_borough                   0
taxi_pick_up_location                  0
bridge_highway_name                   29
bridge_highway_direction              34
road_ramp                              2
bridge_highway_segment               160
garage_lot_name                        0
ferry_direction                        1
ferry_terminal_name                    2
latitude                          125122
longitude                         125216
location                          126048
dtype: int64
```

**- Read or convert the columns 'Created Date' and Closed Date' to datetime datatype**

- create a new column 'Request_Closing_Time' as the time elapsed between request creation and request closing

```
In [9]: df['closed_date'] = pd.to_datetime(df['closed_date'])
```

```
In [10]: df['created_date'] = pd.to_datetime(df['created_date'])
```

```
In [11]: requested_closing_time = df['closed_date'] - df['created_date']
```

```
In [12]: df['requested_closing_time'] = requested_closing_time
```

```
In [13]: # converting the date difference into minutes

requested_closing_time_min = df['requested_closing_time']/np.timedelta64(1,'m')
```

```
In [14]: df['requested_closing_time_min'] = requested_closing_time_min
```

```
In [15]: df.head()
```

Out[15]:

| | unique_key | created_date | closed_date | agency | agency_name | complaint_type | descriptor | location_type | incident_zip | incident_address |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 32310363 | 2015-12-31 23:59:45 | 2016-01-01 00:55:00 | NYPD | New York City Police Department | Noise - Street/Sidewalk | Loud Music/Party | Street/Sidewalk | 10034.0 | 71 VERMILYEA AVENUE |
| 1 | 32309934 | 2015-12-31 23:59:44 | 2016-01-01 01:26:00 | NYPD | New York City Police Department | Blocked Driveway | No Access | Street/Sidewalk | 11105.0 | 27-07 23 AVENUE |
| 2 | 32309159 | 2015-12-31 23:59:29 | 2016-01-01 04:51:00 | NYPD | New York City Police Department | Blocked Driveway | No Access | Street/Sidewalk | 10458.0 | 2897 VALENTINE AVENUE |
| 3 | 32305098 | 2015-12-31 23:57:46 | 2016-01-01 07:43:00 | NYPD | New York City Police Department | Illegal Parking | Commercial Overnight Parking | Street/Sidewalk | 10461.0 | 2940 BAISLEY AVENUE |
| 4 | 32306529 | 2015-12-31 23:56:58 | 2016-01-01 03:24:00 | NYPD | New York City Police Department | Illegal Parking | Blocked Sidewalk | Street/Sidewalk | 11373.0 | 87-14 57 ROAD |

5 rows × 55 columns

```
In [16]: # creating new dataframe

new_df = df[['unique_key','created_date','closed_date','agency', 'agency_name','complaint_type', 'descriptor
             'address_type','city','facility_type', 'status', 'due_date','borough', 'location',
        'requested_closing_time', 'requested_closing_time_min']]
```

```
In [17]: new_df.head()
```

Out[17]:

| | unique_key | created_date | closed_date | agency | agency_name | complaint_type | descriptor | location_type | address_type | city | facil |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 32310363 | 2015-12-31 23:59:45 | 2016-01-01 00:55:00 | NYPD | New York City Police Department | Noise - Street/Sidewalk | Loud Music/Party | Street/Sidewalk | ADDRESS | NEW YORK | |
| 1 | 32309934 | 2015-12-31 23:59:44 | 2016-01-01 01:26:00 | NYPD | New York City Police Department | Blocked Driveway | No Access | Street/Sidewalk | ADDRESS | ASTORIA | |
| 2 | 32309159 | 2015-12-31 23:59:29 | 2016-01-01 04:51:00 | NYPD | New York City Police Department | Blocked Driveway | No Access | Street/Sidewalk | ADDRESS | BRONX | |
| 3 | 32305098 | 2015-12-31 23:57:46 | 2016-01-01 07:43:00 | NYPD | New York City Police Department | Illegal Parking | Commercial Overnight Parking | Street/Sidewalk | ADDRESS | BRONX | |
| 4 | 32306529 | 2015-12-31 23:56:58 | 2016-01-01 03:24:00 | NYPD | New York City Police Department | Illegal Parking | Blocked Sidewalk | Street/Sidewalk | ADDRESS | ELMHURST | |

Converting the series city and borough into lower case to avoid redudancies

```
In [18]: new_df['city'] = new_df['city'].str.lower()
pd.options.mode.chained_assignment = None

C:\Users\Naveen\anaconda3\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

```
In [19]: new_df['borough'] = new_df['borough'].str.lower()
pd.options.mode.chained_assignment = None
```

```
In [20]: new_df.head()
new_df.shape

Out[20]: (300698, 17)
```

```
In [21]: new_df.duplicated().sum()

Out[21]: 0
```

```
In [22]: new_df.isnull().sum()

Out[22]: unique_key                       0
created_date                     0
closed_date                   2164
agency                           0
agency_name                      0
complaint_type                   0
descriptor                    5914
location_type                  131
address_type                  2815
city                          2614
facility_type                 2171
status                           0
due_date                         3
borough                          0
location                      3540
requested_closing_time        2164
requested_closing_time_min    2164
dtype: int64
```

```
In [23]: pd.set_option('mode.chained_assignment', None)
new_df.dropna(inplace=True)
```

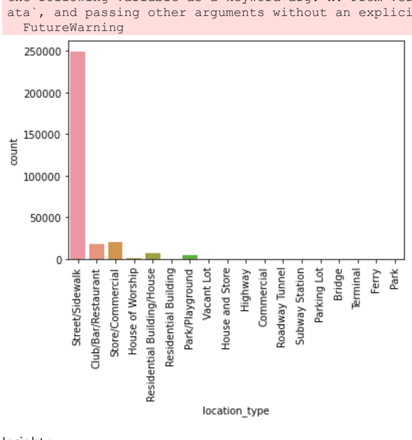
```
In [24]: new_df.isnull().sum()

Out[24]: unique_key                    0
created_date                  0
closed_date                   0
agency                        0
agency_name                   0
complaint_type                0
descriptor                    0
location_type                 0
address_type                  0
city                          0
facility_type                 0
status                        0
due_date                      0
borough                       0
location                      0
requested_closing_time        0
requested_closing_time_min    0
dtype: int64
```

```
In [25]: new_df.dtypes

Out[25]: unique_key                              int64
created_date                   datetime64[ns]
closed_date                    datetime64[ns]
agency                                 object
agency_name                            object
complaint_type                         object
descriptor                             object
location_type                          object
address_type                           object
city                                   object
facility_type                          object
status                                 object
due_date                               object
borough                                object
location                               object
requested_closing_time        timedelta64[ns]
requested_closing_time_min            float64
dtype: object
```

### Provide major insights/patterns that you can offer in a visual format; at least 4 major conclusions that you can come up with after generic data mining.

### 1)

```
In [26]: df['location_type'].value_counts()
plot=sns.countplot(df['location_type'])
plot.set_xticklabels(plot.get_xticklabels(),rotation=90)
plt.show()

C:\Users\Naveen\AppData\Roaming\Python\Python37\site-packages\seaborn\_decorators.py:43: FutureWarning: Pass
the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `d
ata`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

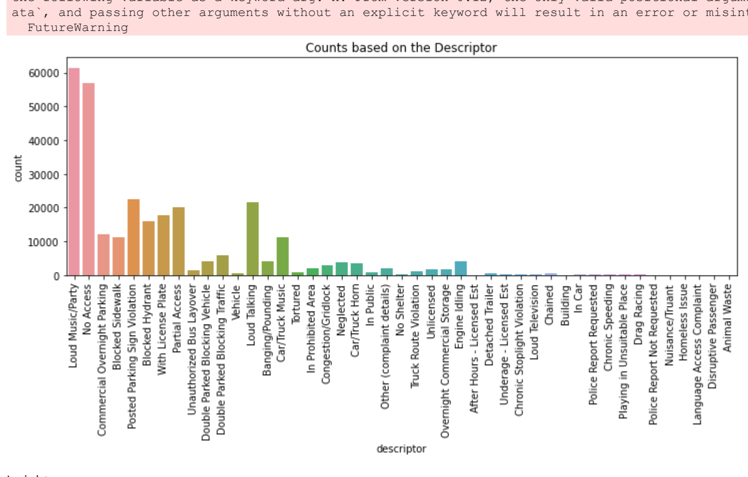



Insight :

```
- Street/ SideWalk is the most used Location Type by New York people. From the plot, we can see it
out-counts most other
types
```

### 2)

```
In [27]: plt.figure(figsize=(12,4))
plot2=sns.countplot(df['descriptor'])
plt.title('Counts based on the Descriptor')
plot2.set_xticklabels(plot2.get_xticklabels(),rotation=90)
plt.show()

C:\Users\Naveen\AppData\Roaming\Python\Python37\site-packages\seaborn\_decorators.py:43: FutureWarning: Pass
the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `d
ata`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

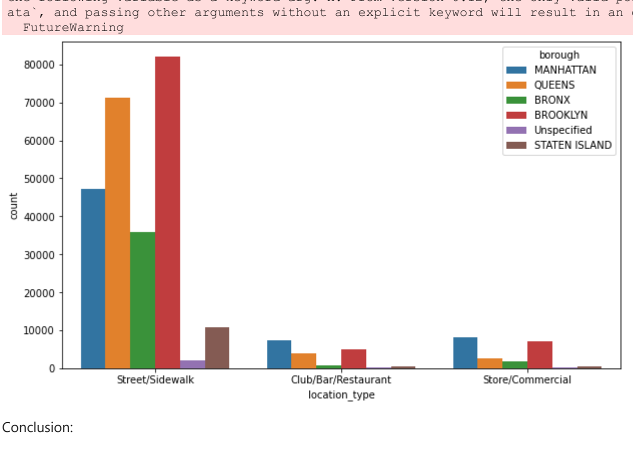



Insights:

```
- "Descriptor" variable plotted here is based on the Complaint type
- From the above, it is seen that Loud Music/Party is highest registered count by the people against
others.
- It comes under the "Noise" Complaint type.
```

### 3)

```
In [28]: plt.figure(figsize=(10,6))
sns.countplot(df.loc[df['location_type'].isin(['Street/Sidewalk','Store/Commercial','Club/Bar/Restaurant'])]
              ['location_type'],data=df,hue='borough')
plt.show()

C:\Users\Naveen\AppData\Roaming\Python\Python37\site-packages\seaborn\_decorators.py:43: FutureWarning: Pass
the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `d
ata`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

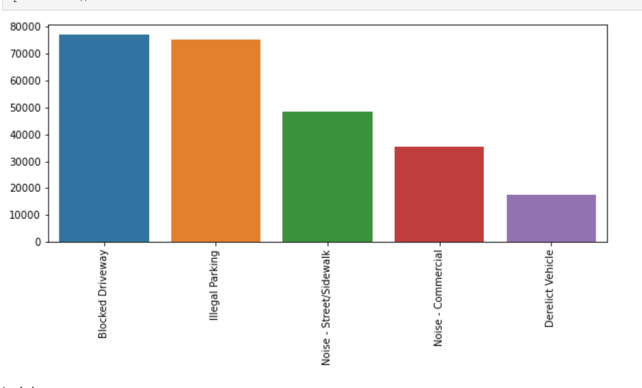



Conclusion:

```
- The plot is used to measure of usage count of location type's in the Cities of Newyork
- From above, we can infer that, Street/SideWalks of BROOKLYN is most used by the people of Newyork
- Footfall in Sidewalks is higher than anyother compared in overall Borough.
```

### 4)

```
In [29]: plt.figure(figsize=(10,4))
series=df['complaint_type'].value_counts()
series.nlargest().index
plot4=sns.barplot(x=series.nlargest().index,y=series.nlargest().values)
plot4.set_xticklabels(plot4.get_xticklabels(),rotation =90)
plt.show()
```




Insights :

```
- The above graph is used to find out which complaint type has been registered high in count.
- As visualised, since Driveway Block is has high count, it is the problem faced most, by the people
around.
```

### Order the complaint types based 'Request_Closing_Time', grouping them for different locations.

```
In [30]: df.pivot_table(index = 'complaint_type',
                      columns='city',
                      values= 'requested_closing_time_min',
                      aggfunc= lambda x : x.mean(),
                       fill_value='-')
```

Out[30]:

| city | ARVERNE | ASTORIA | Astoria | BAYSIDE | BELLEROSE | BREEZY POINT | BRONX | BROOKLYN | CAMBRIA HEIGHTS | CENTRAL PARK | ... | SAINT ALBANS | SO OZ P |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **complaint_type** | | | | | | | | | | | | | |
| **Animal Abuse** | 129.218 | 300.01 | - | 196.471 | 763.481 | 156.808 | 440.13 | 289.95 | 681.35 | - | ... | 442.305 | 20 |
| **Animal in a Park** | - | - | - | - | - | - | - | - | - | - | ... | - | |
| **Bike/Roller/Skate Chronic** | - | 104.359 | - | - | 294 | - | 207.538 | 300.282 | - | - | ... | - | 2 |
| **Blocked Driveway** | 151.558 | 288.967 | 294.878 | 153.784 | 605.965 | 79.6611 | 375.706 | 264.647 | 461.759 | - | ... | 281.207 | 265 |
| **Derelict Vehicle** | 178.093 | 581.355 | 374.169 | 201.587 | 1030.09 | 428.711 | 553.654 | 356.849 | 966.647 | - | ... | 354.03 | 623 |
| **Disorderly Youth** | 215.475 | 174.156 | - | 178.383 | 111 | - | 254.304 | 249.031 | - | - | ... | 105.55 | 160 |
| **Drinking** | 14.3167 | 283.345 | - | 114 | 235.083 | 158 | 347.615 | 212.434 | - | - | ... | 57.6833 | 292 |
| **Graffiti** | 92 | 845.704 | - | 273.094 | - | - | 533.943 | 494.592 | - | - | ... | - | |
| **Homeless Encampment** | 108.888 | 295.055 | - | 172.533 | 2348 | - | 446.514 | 281.689 | 1367.37 | - | ... | 454.242 | 110 |
| **Illegal Fireworks** | - | 166.408 | - | - | 400 | - | 336.492 | 140.375 | 91.8333 | - | ... | - | 3 |
| **Illegal Parking** | 138.982 | 290.003 | 282.677 | 153.76 | 492.188 | 236.533 | 394.851 | 256.334 | 674.606 | 275.658 | ... | 251.058 | 28 |
| **Noise - Commercial** | 137.233 | 187.982 | 212.513 | 134.02 | 404.446 | 152.371 | 281.841 | 179.166 | 228.601 | - | ... | 186.015 | 2 |
| **Noise - House of Worship** | 93.7318 | 121.333 | - | 212.092 | 131.817 | - | 273.573 | 184.192 | 158.458 | - | ... | 278.75 | 129 |
| **Noise - Park** | 77 | 179.707 | - | 196.354 | 84.85 | - | 281.912 | 189.032 | - | - | ... | 49.2167 | 342 |
| **Noise - Street/Sidewalk** | 119.544 | 207.07 | 222.794 | 91.86 | 544.094 | 60 | 313.579 | 197.76 | 276.483 | 196.016 | ... | 201.971 | 219 |
| **Noise - Vehicle** | 111.593 | 210.551 | - | 102.505 | 154.99 | 79.4667 | 333.648 | 197.007 | 415.026 | - | ... | 207.515 | 227 |
| **Panhandling** | 62 | 69 | - | - | 449 | - | 852.824 | 258.989 | - | - | ... | - | |
| **Posting Advertisement** | - | 352.2 | - | - | 135.6 | - | 207.72 | 201.657 | - | - | ... | - | 77.1 |
| **Squeegee** | - | - | - | - | - | - | - | - | - | - | ... | - | |
| **Traffic** | - | 324.611 | - | 91.6167 | 345.531 | - | 295.421 | 186.776 | 512.803 | - | ... | 237.906 | 213 |
| **Urinating in Public** | 41.5167 | 277.543 | - | - | 452.483 | - | 323.406 | 233.959 | - | - | ... | 390.8 | 1 |
| **Vending** | 29 | 296.125 | - | 112.675 | - | - | 409.542 | 271.686 | - | - | ... | 143.308 | 211 |

22 rows × 53 columns

## Provide a statistical test to accept or reject the Null Hypothesis along with the corresponding 'p-value'.

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

- $H_0$: Overall ComplaintType and Average Response Time doesn't have any significant difference
- $H_a$: Atleast one of the ComplaintType's and Avg. Response Time has significant difference

```
In [31]: import scipy.stats as stats
```

```
In [32]: top = new_df['complaint_type'].value_counts().index
top

Out[32]: Index(['Blocked Driveway', 'Illegal Parking', 'Noise - Street/Sidewalk',
       'Noise - Commercial', 'Derelict Vehicle', 'Noise - Vehicle',
       'Animal Abuse', 'Traffic', 'Noise - Park', 'Vending', 'Drinking',
       'Noise - House of Worship', 'Posting Advertisement', 'Disorderly Youth',
       'Graffiti'],
      dtype='object')
```

```
In [33]: from statsmodels.formula.api import ols
import statsmodels.api as sm
```

```
In [34]: sample = new_df[new_df['complaint_type'].isin(top)]
mod = ols('requested_closing_time_min ~ complaint_type', data = sample).fit()
sm.stats.anova_lm(mod)
```

Out[34]:

| | df | sum_sq | mean_sq | F | PR(>F) |
|---|---|---|---|---|---|
| **complaint_type** | 14.0 | 1.043265e+09 | 7.451890e+07 | 578.768616 | 0.0 |
| **Residual** | 290869.0 | 3.745061e+10 | 1.287542e+05 | NaN | NaN |

### F-stat

Conclusion:

```
- From the F-stat, the p_value < 0.05
- Hence, we reject null hypothesis
```

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

Hypothesis:

- $H_0$: The complaint or service requested and location are not related/ independent
- $H_0$: They are related/ dependent

```
In [35]: topall = df['complaint_type'].value_counts().index
sample = df[df['complaint_type'].isin(topall)]
ct = pd.crosstab(sample['complaint_type'], sample.location)
ct
```

Out[35]:

| location | (40.49913462101514, -74.24348482977875) | (40.49967332981336, -74.2379063249761) | (40.49994886080869, -74.23740031497493) | (40.49999700116009, -74.23801175120917) | (40.50002168207532, -74.23802262609722) | (40.500049 -74.2380 |
|---|---|---|---|---|---|---|
| **complaint_type** | | | | | | |
| **Animal Abuse** | 0 | 0 | 0 | 0 | 0 | |
| **Bike/Roller/Skate Chronic** | 0 | 0 | 0 | 0 | 0 | |
| **Blocked Driveway** | 0 | 0 | 0 | 0 | 0 | |
| **Derelict Vehicle** | 0 | 1 | 0 | 0 | 0 | |
| **Disorderly Youth** | 0 | 0 | 0 | 0 | 0 | |
| **Drinking** | 0 | 0 | 0 | 0 | 0 | |
| **Graffiti** | 0 | 0 | 0 | 0 | 0 | |
| **Homeless Encampment** | 0 | 0 | 0 | 0 | 0 | |
| **Illegal Fireworks** | 0 | 0 | 0 | 0 | 0 | |
| **Illegal Parking** | 1 | 0 | 1 | 1 | 1 | |
| **Noise - Commercial** | 0 | 0 | 0 | 0 | 0 | |
| **Noise - House of Worship** | 0 | 0 | 0 | 0 | 0 | |
| **Noise - Park** | 0 | 0 | 0 | 0 | 0 | |
| **Noise - Street/Sidewalk** | 0 | 0 | 0 | 0 | 0 | |
| **Noise - Vehicle** | 0 | 0 | 0 | 0 | 0 | |
| **Panhandling** | 0 | 0 | 0 | 0 | 0 | |
| **Posting Advertisement** | 0 | 0 | 0 | 0 | 0 | |
| **Squeegee** | 0 | 0 | 0 | 0 | 0 | |
| **Traffic** | 0 | 0 | 0 | 0 | 0 | |
| **Urinating in Public** | 0 | 0 | 0 | 0 | 0 | |
| **Vending** | 0 | 0 | 0 | 0 | 0 | |

21 rows × 126048 columns

```
In [36]: chisquare , p , df, et = stats.chi2_contingency(ct)
```

```
In [37]: print('Results : statistic={:.4f}, pvalue={}'.format(chisquare,p))

Results : statistic=4161473.2250, pvalue=0.0
```

Conclusion:

- Here, p_value < 0.05
- Thus " we reject null hypothesis"

```
In [ ]:
```