In [1]:
```
# to read and wrangle data
import pandas as pd
import numpy as np
import seaborn as sns
from pointpats import centrography
from datetime import datetime
from scipy import stats
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats.multicomp import pairwise_tukeyhsd
from scipy.stats import ttest_ind



# to create spatial data
import geopandas as gpd
from shapely.geometry import MultiPolygon


# for basemaps
import contextily as ctx
import leafmap.kepler as leafmap

# For spatial statistics
import esda
from esda.moran import Moran, Moran_Local
from pysal.lib import weights
from splot import esda as esdaplot

import splot
from splot.esda import moran_scatterplot, plot_moran, lisa_cluster,plot_moran_simulation
from IPython.display import display, Markdown, display_latex, display_markdown, display_html

import libpysal as lps

# Graphics
import matplotlib.pyplot as plt
import plotly.express as px
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
data = pd.read_csv("NYPD_Arrest_Data_2023.csv (1).zip")
```

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

Out[3]:

	ARREST_KEY	ARREST_DATE	PD_CD	PD_DESC	KY_CD	OFNS_DESC	LAW_CODE	LAW_CAT_CD	ARREST_BORO
0	261209118	01/01/2023	109	ASSAULT 2,1,UNCLASSIFIED	106.0	FELONY ASSAULT	PL 1200501	F	K
1	262984267	02/03/2023	515	CONTROLLED SUBSTANCE,SALE 3	117.0	DANGEROUS DRUGS	PL 2203901	F	K
2	263664549	02/15/2023	105	STRANGULATION 1ST	106.0	FELONY ASSAULT	PL 1211200	F	K
3	261345231	01/04/2023	105	STRANGULATION 1ST	106.0	FELONY ASSAULT	PL 1211200	F	M
4	263536618	02/13/2023	109	ASSAULT 2,1,UNCLASSIFIED	106.0	FELONY ASSAULT	PL 12005WX	F	K

In [4]:
```
data.shape
```

Out[4]:
```
(170095, 18)
```

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 170095 entries, 0 to 170094
Data columns (total 18 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   ARREST_KEY         170095 non-null  int64  
 1   ARREST_DATE        170095 non-null  object 
 2   PD_CD              170095 non-null  int64  
 3   PD_DESC            170095 non-null  object 
 4   KY_CD              170082 non-null  float64
 5   OFNS_DESC          170095 non-null  object 
 6   LAW_CODE           170095 non-null  object 
 7   LAW_CAT_CD         168838 non-null  object 
 8   ARREST_BORO        170095 non-null  object 
 9   ARREST_PRECINCT    170095 non-null  int64  
 10  JURISDICTION_CODE  170095 non-null  int64  
 11  AGE_GROUP          170095 non-null  object 
 12  PERP_SEX           170095 non-null  object 
 13  PERP_RACE          170095 non-null  object 
 14  X_COORD_CD         170095 non-null  int64  
 15  Y_COORD_CD         170095 non-null  int64  
 16  Latitude           170095 non-null  float64
 17  Longitude          170095 non-null  float64
dtypes: float64(3), int64(6), object(9)
memory usage: 23.4+ MB
```

In [6]:
```
data.isna().sum()
```

Out[6]:
```
ARREST_KEY              0
ARREST_DATE             0
PD_CD                   0
PD_DESC                 0
KY_CD                  13
OFNS_DESC               0
LAW_CODE                0
LAW_CAT_CD           1257
ARREST_BORO             0
ARREST_PRECINCT         0
JURISDICTION_CODE       0
AGE_GROUP               0
PERP_SEX                0
PERP_RACE               0
X_COORD_CD              0
Y_COORD_CD              0
Latitude                0
Longitude               0
dtype: int64
```

In [7]:
```
data.dropna(inplace = True)
```

In [8]:
```
data.duplicated().sum()
```

Out[8]:
```
0
```

In [9]:
```
data["AGE_GROUP"].value_counts()
```

Out[9]:
```
AGE_GROUP
25-44    96830
45-64    33151
18-24    29823
<18       6244
65+       2777
Name: count, dtype: int64
```

In [10]:
```
# Define a function to map age ranges to discrete values
def map_age_range(age_range):
    if age_range == '<18':
        return 'Teenager'
    elif age_range == '18-24':
        return 'Young Adult'
    elif age_range == '25-44':
        return 'Middle-aged Adult'
    elif age_range == '65+':
        return 'Senior'
# Apply the function to create a new column with discrete values
data['Age_Group'] = data['AGE_GROUP'].apply(map_age_range)
```

In [11]:
```
# Function to calculate midpoint of age range
def calculate_midpoint(age_range):
    if age_range == '<18':
        return 17.5  # Assuming '<18' represents ages less than 18
    elif age_range == '65+':
        return 65.0  # Assuming '65+' represents ages 65 and above
```

```
    else:
        start, end = map(int, age_range.split('-'))
        return (start + end) / 2

# Apply the function to create a new column with the midpoint values
data["New_Age"] = data['AGE_GROUP'].apply(calculate_midpoint)
```

In [12]:
```
data["New_Age"].value_counts()
```

Out[12]:
```
New_Age
34.5    96830
54.5    33151
21.0    29823
17.5     6244
65.0     2777
Name: count, dtype: int64
```

In [13]:
```
# Define a function to map age ranges to discrete values
def map_age_range(age_range):
    if age_range == '17.5':
        return 'Teenager'
    elif age_range == '21.0':
        return 'Young Adult'
    elif age_range == '34.5':
        return 'Middle-aged Adult'
    elif age_range == '54.5':
        return 'Senior Adult'
    elif age_range == "65.0":
        return "Senior"
# Apply the function to create a new column with discrete values
data['Group_Age'] = data['New_Age'].apply(map_age_range)
```

In [14]:
```
# Group by location and count crimes
crime_counts = data.groupby('Latitude').size().reset_index(name='Crime Count')
```

In [15]:
```
new_df = pd.merge(data, crime_counts, on ="Latitude" , how='left')
```

In [16]:
```
new_df.head()
```

Out[16]:

	ARREST_KEY	ARREST_DATE	PD_CD	PD_DESC	KY_CD	OFNS_DESC	LAW_CODE	LAW_CAT_CD	ARREST_BORO
0	261209118	01/01/2023	109	ASSAULT 2,1,UNCLASSIFIED	106.0	FELONY ASSAULT	PL 1200501	F	K
1	262984267	02/03/2023	515	CONTROLLED SUBSTANCE,SALE 3	117.0	DANGEROUS DRUGS	PL 2203901	F	K
2	263664549	02/15/2023	105	STRANGULATION 1ST	106.0	FELONY ASSAULT	PL 1211200	F	K
3	261345231	01/04/2023	105	STRANGULATION 1ST	106.0	FELONY ASSAULT	PL 1211200	F	M
4	263536618	02/13/2023	109	ASSAULT 2,1,UNCLASSIFIED	106.0	FELONY ASSAULT	PL 12005WX	F	K

5 rows × 22 columns

In [17]:
```
# Define a function to map age ranges to discrete values
def map_age_range(age_range):
    if age_range == 17.5:
        return 'Teenager'
    elif age_range == 21.0:
        return 'Young Adult'
    elif age_range == 34.5:
        return 'Middle-aged Adult'
    elif age_range == 54.5:
        return 'Senior Adult'
    elif age_range == 65.0:
        return "Senior"
# Apply the function to create a new column with discrete values
new_df['Group_Age'] = new_df['New_Age'].apply(map_age_range)
```

In [18]:
```
new_df.head()
```

Out[18]:

	ARREST_KEY	ARREST_DATE	PD_CD	PD_DESC	KY_CD	OFNS_DESC	LAW_CODE	LAW_CAT_CD	ARREST_BORO
0	261209118	01/01/2023	109	ASSAULT 2,1,UNCLASSIFIED	106.0	FELONY ASSAULT	PL 1200501	F	K
1	262984267	02/03/2023	515	CONTROLLED SUBSTANCE,SALE 3	117.0	DANGEROUS DRUGS	PL 2203901	F	K
2	263664549	02/15/2023	105	STRANGULATION 1ST	106.0	FELONY ASSAULT	PL 1211200	F	K
3	261345231	01/04/2023	105	STRANGULATION 1ST	106.0	FELONY ASSAULT	PL 1211200	F	M
4	263536618	02/13/2023	109	ASSAULT 2,1,UNCLASSIFIED	106.0	FELONY ASSAULT	PL 12005WX	F	K

5 rows × 22 columns

In [19]:
```
# Drop the original 'Age' column if no longer needed
new_df.drop('Age_Group', axis=1, inplace=True)
```

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

Out[20]:

	ARREST_KEY	ARREST_DATE	PD_CD	PD_DESC	KY_CD	OFNS_DESC	LAW_CODE	LAW_CAT_CD	ARREST_BORO
0	261209118	01/01/2023	109	ASSAULT 2,1,UNCLASSIFIED	106.0	FELONY ASSAULT	PL 1200501	F	K
1	262984267	02/03/2023	515	CONTROLLED SUBSTANCE,SALE 3	117.0	DANGEROUS DRUGS	PL 2203901	F	K
2	263664549	02/15/2023	105	STRANGULATION 1ST	106.0	FELONY ASSAULT	PL 1211200	F	K
3	261345231	01/04/2023	105	STRANGULATION 1ST	106.0	FELONY ASSAULT	PL 1211200	F	M
4	263536618	02/13/2023	109	ASSAULT 2,1,UNCLASSIFIED	106.0	FELONY ASSAULT	PL 12005WX	F	K

5 rows × 21 columns

In [21]:
```
# Drop the original 'Age' column if no longer needed
new_df.drop('AGE_GROUP', axis=1, inplace=True)
```

In [22]:
```
new_df.head()
```

Out[22]:

	ARREST_KEY	ARREST_DATE	PD_CD	PD_DESC	KY_CD	OFNS_DESC	LAW_CODE	LAW_CAT_CD	ARREST_BORO
0	261209118	01/01/2023	109	ASSAULT 2,1,UNCLASSIFIED	106.0	FELONY ASSAULT	PL 1200501	F	K
1	262984267	02/03/2023	515	CONTROLLED SUBSTANCE,SALE 3	117.0	DANGEROUS DRUGS	PL 2203901	F	K
2	263664549	02/15/2023	105	STRANGULATION 1ST	106.0	FELONY ASSAULT	PL 1211200	F	K
3	261345231	01/04/2023	105	STRANGULATION 1ST	106.0	FELONY ASSAULT	PL 1211200	F	M
4	263536618	02/13/2023	109	ASSAULT 2,1,UNCLASSIFIED	106.0	FELONY ASSAULT	PL 12005WX	F	K

In [23]:
```
# lets convert all date to datatime
new_df["ARREST_DATE"] = pd.to_datetime(new_df["ARREST_DATE"], format = "mixed")
```

In [24]:
```
# Extract year, month, and day from the 'date' column
new_df['YEAR'] = new_df['ARREST_DATE'].dt.year
new_df['MONTH'] = new_df['ARREST_DATE'].dt.strftime('%B')
new_df['DAY'] = new_df['ARREST_DATE'].dt.strftime('%A')
```

In [25]:
```
new_df.head()
```

Out[25]:

	ARREST_KEY	ARREST_DATE	PD_CD	PD_DESC	KY_CD	OFNS_DESC	LAW_CODE	LAW_CAT_CD	ARREST_BORO
0	261209118	2023-01-01	109	ASSAULT 2,1,UNCLASSIFIED	106.0	FELONY ASSAULT	PL 1200501	F	K
1	262984267	2023-02-03	515	CONTROLLED SUBSTANCE,SALE 3	117.0	DANGEROUS DRUGS	PL 2203901	F	K
2	263664549	2023-02-15	105	STRANGULATION 1ST	106.0	FELONY ASSAULT	PL 1211200	F	K
3	261345231	2023-01-04	105	STRANGULATION 1ST	106.0	FELONY ASSAULT	PL 1211200	F	M
4	263536618	2023-02-13	109	ASSAULT 2,1,UNCLASSIFIED	106.0	FELONY ASSAULT	PL 12005WX	F	K

5 rows × 23 columns

In [26]:
```
# Drop the original 'Age' column if no longer needed
new_df.drop('ARREST_DATE', axis=1, inplace=True)
```

In [27]:
```
data_new = new_df.copy()
```

In [28]:
```
data_new.head()
```

Out[28]:

	ARREST_KEY	PD_CD	PD_DESC	KY_CD	OFNS_DESC	LAW_CODE	LAW_CAT_CD	ARREST_BORO	ARREST_PRECIN
0	261209118	109	ASSAULT 2,1,UNCLASSIFIED	106.0	FELONY ASSAULT	PL 1200501	F	K	
1	262984267	515	CONTROLLED SUBSTANCE,SALE 3	117.0	DANGEROUS DRUGS	PL 2203901	F	K	
2	263664549	105	STRANGULATION 1ST	106.0	FELONY ASSAULT	PL 1211200	F	K	
3	261345231	105	STRANGULATION 1ST	106.0	FELONY ASSAULT	PL 1211200	F	M	
4	263536618	109	ASSAULT 2,1,UNCLASSIFIED	106.0	FELONY ASSAULT	PL 12005WX	F	K	

5 rows × 22 columns

In [29]:
```
# Define a function to map age ranges to discrete numerical values
def borough(area):
    if area == 'K':
        return "Brooklyn"
    elif area == 'B':
        return "Bronx"
    elif area == 'M':
        return "Manhattan"
    elif area == 'Q':
        return "Queens"
    elif area == 'S':
        return "Staten Island"
```

In [30]:
```
data_new['BORO'] = data_new['ARREST_BORO'].apply(borough)
```

In [31]:
```
data_new.head()
```

Out[31]:

	ARREST_KEY	PD_CD	PD_DESC	KY_CD	OFNS_DESC	LAW_CODE	LAW_CAT_CD	ARREST_BORO	ARREST_PRECIN
0	261209118	109	ASSAULT 2,1,UNCLASSIFIED	106.0	FELONY ASSAULT	PL 1200501	F	K	
1	262984267	515	CONTROLLED SUBSTANCE,SALE 3	117.0	DANGEROUS DRUGS	PL 2203901	F	K	
2	263664549	105	STRANGULATION 1ST	106.0	FELONY ASSAULT	PL 1211200	F	K	
3	261345231	105	STRANGULATION 1ST	106.0	FELONY ASSAULT	PL 1211200	F	M	
4	263536618	109	ASSAULT 2,1,UNCLASSIFIED	106.0	FELONY ASSAULT	PL 12005WX	F	K	

5 rows × 23 columns

In [32]:
```
data_new.columns
```

Out[32]:
```
Index(['ARREST_KEY', 'PD_CD', 'PD_DESC', 'KY_CD', 'OFNS_DESC', 'LAW_CODE',
       'LAW_CAT_CD', 'ARREST_BORO', 'ARREST_PRECINCT', 'JURISDICTION_CODE',
       'PERP_SEX', 'PERP_RACE', 'X_COORD_CD', 'Y_COORD_CD', 'Latitude',
       'Longitude', 'New_Age', 'Group_Age', 'Crime Count', 'YEAR', 'MONTH',
       'DAY', 'BORO'],
      dtype='object')
```

In [33]:
```
data_new["PERP_RACE"].value_counts()
```

Out[33]:
```
PERP_RACE
BLACK                            81620
WHITE HISPANIC                   42703
BLACK HISPANIC                   16301
WHITE                            16229
ASIAN / PACIFIC ISLANDER          8982
UNKNOWN                           2440
AMERICAN INDIAN/ALASKAN NATIVE     550
Name: count, dtype: int64
```

In [34]:
```
# Replacing Male/MF  with Male and Female/F  with Female
data_new["LAW_CAT_CD"].replace(["M"],value = 'Misdemeanor' , inplace=True)
data_new["LAW_CAT_CD"].replace(["F"],value = "Felony",inplace=True)
data_new["LAW_CAT_CD"].replace(["V"],value = 'Violation',inplace=True)
data_new["LAW_CAT_CD"].replace(["9", "I"],value = 'Others' , inplace=True)
```

In [35]:
```
data_new.head()
```

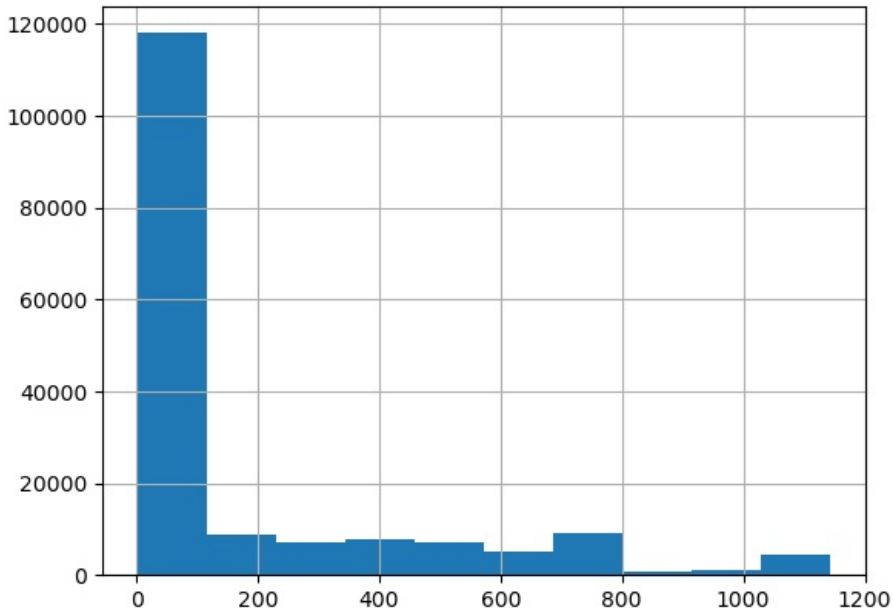
Out[35]:

	ARREST_KEY	PD_CD	PD_DESC	KY_CD	OFNS_DESC	LAW_CODE	LAW_CAT_CD	ARREST_BORO	ARREST_PRECIN
0	261209118	109	ASSAULT 2,1,UNCLASSIFIED	106.0	FELONY ASSAULT	PL 1200501	Felony	K	
1	262984267	515	CONTROLLED SUBSTANCE,SALE 3	117.0	DANGEROUS DRUGS	PL 2203901	Felony	K	
2	263664549	105	STRANGULATION 1ST	106.0	FELONY ASSAULT	PL 1211200	Felony	K	
3	261345231	105	STRANGULATION 1ST	106.0	FELONY ASSAULT	PL 1211200	Felony	M	
4	263536618	109	ASSAULT 2,1,UNCLASSIFIED	106.0	FELONY ASSAULT	PL 12005WX	Felony	K	

5 rows × 23 columns

In [36]:
```
data_new["Crime Count"].hist()
```

Out[36]:
```
<Axes: >
```



In statistics, the ANOVA (Analysis of Variance) test assumes that the data follow a normal distribution. When the data are right-skewed, meaning that there is a longer tail on the right side of the distribution, it violates the assumption of normality. This violation can affect the results of the ANOVA test.

Right-skewed data can lead to biased estimates of the means and variances, which are used in the ANOVA calculations. As a result, the ANOVA test may produce inaccurate p-values and conclusions about the statistical significance of differences between groups.

In particular, right-skewed data can inflate the Type I error rate, leading to an increased likelihood of falsely rejecting the null hypothesis (i.e., detecting differences between groups when there are none). This occurs because the right skewness can cause the mean to be pulled towards the higher end of the distribution, potentially making differences between groups appear more significant than they actually are.

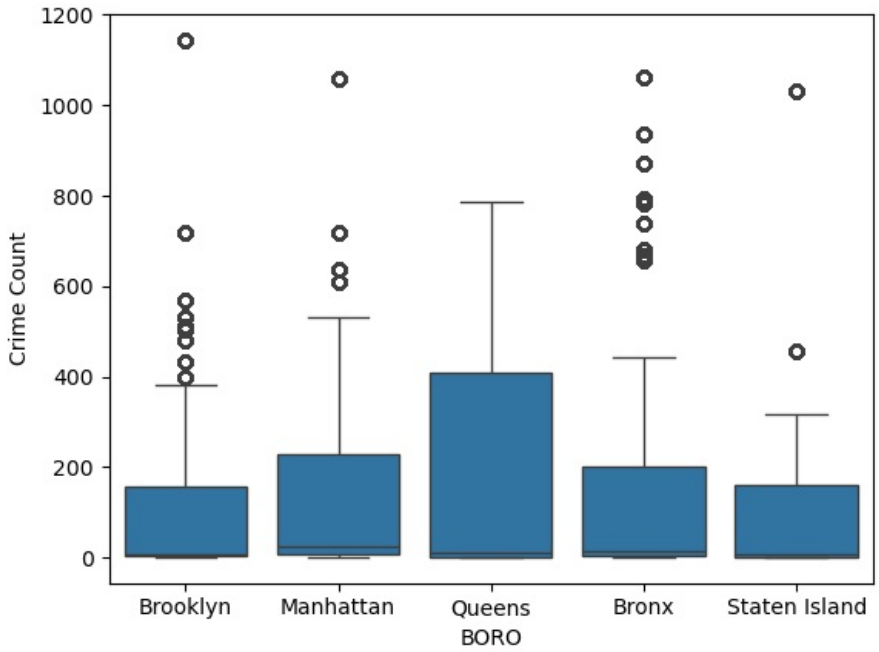
To mitigate the impact of right-skewed data on the ANOVA test, it's important to assess the distribution of the data and consider transformations or alternative statistical tests that are more robust to non-normality. Transformations such as logarithmic or square root transformations can sometimes help make the data more symmetric and closer to a normal distribution, improving the validity of the ANOVA results. Additionally, non-parametric tests like the Kruskal-Wallis test can be used as alternatives to ANOVA when the assumption of normality is violated.

In [37]:
```
df_new = data_new.copy()
```

In [38]:
```
df_new["PERP_SEX"].value_counts()
```

Out[38]:
```
PERP_SEX
M    136432
F     28903
U      3490
Name: count, dtype: int64
```

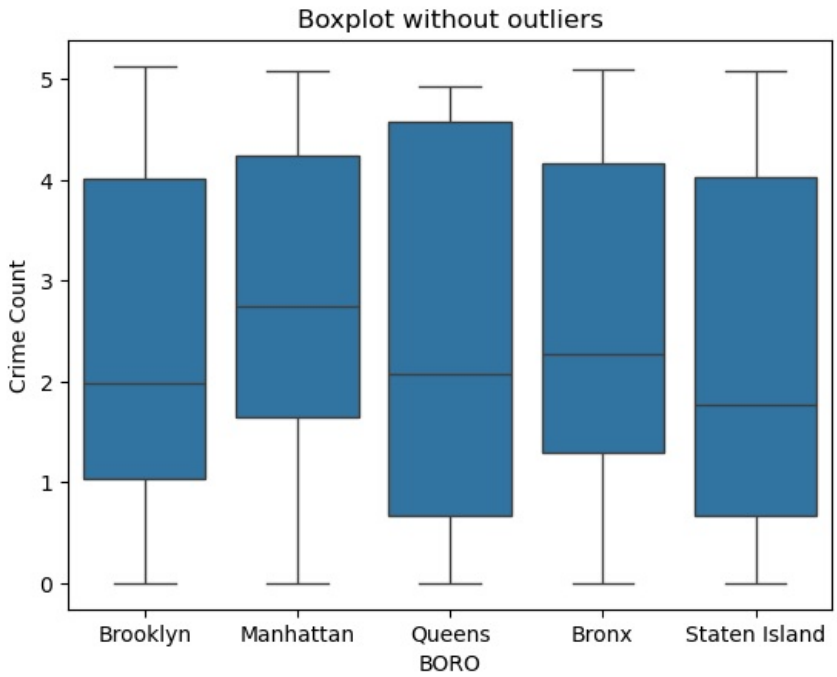
In [39]:
```
# Create boxplot using seaborn
sns.boxplot(data=df_new, x='BORO', y='Crime Count')

# Show plot
plt.show()
```



In [40]:
```
# Remove any zero or negative values (Box-Cox transformation requires strictly positive values)
positive_values = df_new[df_new['Crime Count'] > 0]['Crime Count']

# Perform the Box-Cox transformation
transformed_values, lambda_value = stats.boxcox(positive_values)

# Replace the original column with the transformed values
df_new.loc[df_new['Crime Count'] > 0, 'Crime Count'] = transformed_values

# Print the estimated lambda value
print("Lambda value:", lambda_value)
```

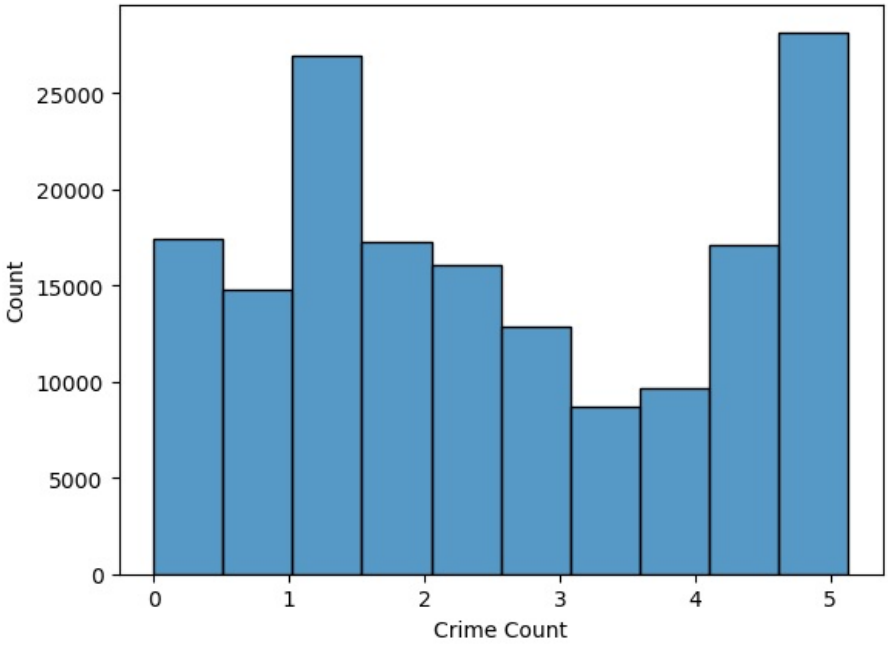
```
Lambda value: -0.09535983143313273
```

In [41]:
```
# Create a boxplot with seaborn to visualize the data without outliers
sns.boxplot(x='BORO', y='Crime Count', data=df_new)
plt.title('Boxplot without outliers')
plt.show()
```

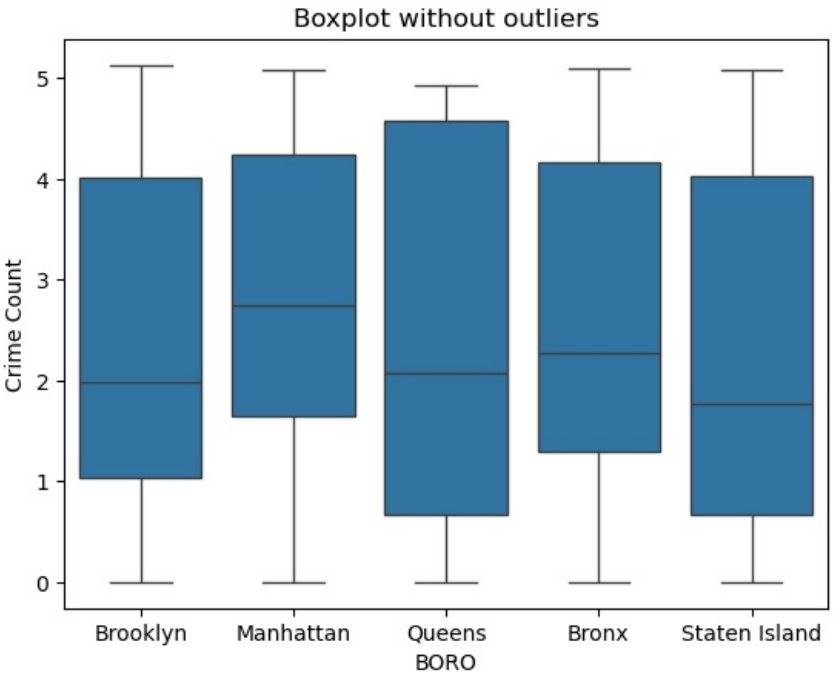


In [42]:
```
sns.histplot(data=df_new, x="Crime Count", bins=10)
```

Out[42]:
```
<Axes: xlabel='Crime Count', ylabel='Count'>
```



In [43]:
```
# Create a boxplot with seaborn to visualize the data without outliers
sns.boxplot(x='BORO', y='Crime Count', data=df_new)
plt.title('Boxplot without outliers')
plt.show()
```



In [44]:
```
df_new.rename(columns={'Crime Count': 'CRIME_COUNT'}, inplace=True)
```

In [45]:
```
# Perform one-way ANOVA using crime_per_pop_1000 and the top 5 municipalities
model = ols('CRIME_COUNT ~ BORO', data= df_new).fit()
anova_table = sm.stats.anova_lm(model, typ=2)

# Print ANOVA table
print(anova_table)

# Perform Tukey's HSD test for multiple comparisons
tukey_result = pairwise_tukeyhsd(endog=df_new['CRIME_COUNT'], groups=df_new['BORO'], alpha=0.05)

# Print the results
print(tukey_result)
```

```
                sum_sq        df           F  PR(>F)
BORO        5534.018305       4.0  515.002733     0.0
Residual  453518.453069  168820.0         NaN     NaN
     Multiple Comparison of Means - Tukey HSD, FWER=0.05
===================================================================
  group1      group2    meandiff p-adj   lower   upper  reject
-------------------------------------------------------------------
    Bronx      Brooklyn   -0.293    0.0 -0.3235 -0.2624   True
    Bronx     Manhattan   0.1547    0.0  0.1228  0.1866   True
    Bronx        Queens  -0.1885    0.0  -0.221  -0.156   True
    Bronx Staten Island  -0.3821    0.0 -0.4387 -0.3255   True
 Brooklyn     Manhattan   0.4477    0.0   0.417  0.4783   True
 Brooklyn        Queens   0.1044    0.0  0.0731  0.1358   True
 Brooklyn Staten Island  -0.0892 0.0001 -0.1451 -0.0333   True
Manhattan        Queens  -0.3432    0.0 -0.3759 -0.3106   True
Manhattan Staten Island  -0.5368    0.0 -0.5935 -0.4802   True
   Queens Staten Island  -0.1936    0.0 -0.2507 -0.1366   True
-------------------------------------------------------------------
```

- After running the ANOVA test we could observed that the number of crime in each Borough is not the same, there is a significant difference in crime occurrence between the various Borough in New York, therefore we reject the null hypothesis.

In [46]:
```
# Perform one-way ANOVA using crime_per_pop_1000 and the top 5 municipalities
model = ols('CRIME_COUNT ~ LAW_CAT_CD', data= df_new).fit()
anova_table = sm.stats.anova_lm(model, typ=2)

# Print ANOVA table
print(anova_table)

# Perform Tukey's HSD test for multiple comparisons
tukey_result = pairwise_tukeyhsd(endog=df_new['CRIME_COUNT'], groups=df_new['LAW_CAT_CD'], alpha=0.05)

# Print the results
print(tukey_result)
```

```
                  sum_sq        df         F  PR(>F)
LAW_CAT_CD    4700.924124       3.0  582.2325     0.0
Residual    454351.547251  168821.0       NaN     NaN
     Multiple Comparison of Means - Tukey HSD, FWER=0.05
===================================================================
   group1      group2   meandiff p-adj   lower   upper  reject
-------------------------------------------------------------------
     Felony Misdemeanor  -0.3131    0.0 -0.3339 -0.2923   True
     Felony      Others   0.8319    0.0  0.6662  0.9977   True
     Felony   Violation  -0.1954 0.0013 -0.3314 -0.0594   True
Misdemeanor      Others    1.145    0.0  0.9794  1.3107   True
Misdemeanor   Violation   0.1177 0.1163 -0.0182  0.2535  False
     Others   Violation  -1.0274    0.0 -1.2407 -0.8141   True
-------------------------------------------------------------------
```

- Per the description of crime base on the laws of the land in US, Crime are group into 4 categories which are Felony, Misdemeanor, Violation and Other. Per our statistical analysis almost all the categories has a statistical significant between them, meaning these categories does not occurs randomly except Misdemeanor and Violation, which has a p-values of 1.145 which is above the significant level of 0.05%.

In [47]:
```
data_new.columns
```

Out[47]:
```
Index(['ARREST_KEY', 'PD_CD', 'PD_DESC', 'KY_CD', 'OFNS_DESC', 'LAW_CODE',
       'LAW_CAT_CD', 'ARREST_BORO', 'ARREST_PRECINCT', 'JURISDICTION_CODE',
       'PERP_SEX', 'PERP_RACE', 'X_COORD_CD', 'Y_COORD_CD', 'Latitude',
       'Longitude', 'New_Age', 'Group_Age', 'Crime Count', 'YEAR', 'MONTH',
       'DAY', 'BORO'],
      dtype='object')
```

In [48]:
```
from scipy.stats import chi2_contingency
```

In [182…
```
# Create a contingency table (cross-tabulation)
contingency_table = pd.crosstab(data_new['PERP_RACE'], data_new['LAW_CAT_CD'])

# Perform chi-square test
chi2, p_value, dof, expected = chi2_contingency(contingency_table)

# Print results
# Convert the p-value to a string with real zeros
p_value_str = '{:.100f}'.format(p_value)

# Print the p-value with real zeros
print("P-value:", p_value_str)
print("Chi-square statistic:", chi2)
print("Degrees of freedom:", dof)
```

```
print("Expected frequencies table:")
print(expected)

# Check significance at alpha = 0.05
alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis. There is significant evidence of an association between PERP_RACE and LAW
else:
    print("Fail to reject the null hypothesis. There is no significant evidence of an association between PERP_I
```

```
P-value: 0.0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
Chi-square statistic: 600.9504978848743
Degrees of freedom: 18
Expected frequencies table:
[[2.40599141e+02 3.04106915e+02 2.12409300e+00 3.16985044e+00]
 [3.92920270e+03 4.96634239e+03 3.46883696e+01 5.17665393e+01]
 [3.57049125e+04 4.51294663e+04 3.15215401e+02 4.70405805e+02]
 [7.13092109e+03 9.01317605e+03 6.29542544e+01 9.39486036e+01]
 [1.06738528e+03 1.34912886e+03 9.42324893e+00 1.40626092e+01]
 [7.09942448e+03 8.97336569e+03 6.26761913e+01 9.35336413e+01]
 [1.86805548e+04 2.36114138e+04 1.64918442e+02 2.46112951e+02]]
Reject the null hypothesis. There is significant evidence of an association between PERP_RACE and LAW_CAT_CD.
```

- A chi-square test was perform between race and crime types in New York, not all expected cell frequency were greater than 5, therefore the chi-square assumption was not meet. Moreover there was strong statistical significance relationship between race and crime in New York. The p-value of 0.00 therefore the chi-square test is statistical significant and we reject the null hypothesis.So there is a strong relationship between crime and race in New York.

In [50]:
```
df_new.columns
```

Out[50]:
```
Index(['ARREST_KEY', 'PD_CD', 'PD_DESC', 'KY_CD', 'OFNS_DESC', 'LAW_CODE',
       'LAW_CAT_CD', 'ARREST_BORO', 'ARREST_PRECINCT', 'JURISDICTION_CODE',
       'PERP_SEX', 'PERP_RACE', 'X_COORD_CD', 'Y_COORD_CD', 'Latitude',
       'Longitude', 'New_Age', 'Group_Age', 'CRIME_COUNT', 'YEAR', 'MONTH',
       'DAY', 'BORO'],
      dtype='object')
```

In [51]:
```
sex_male = df_new[df_new["PERP_SEX"]== "M"]
sex_female = df_new[df_new["PERP_SEX"]== "F"]
```

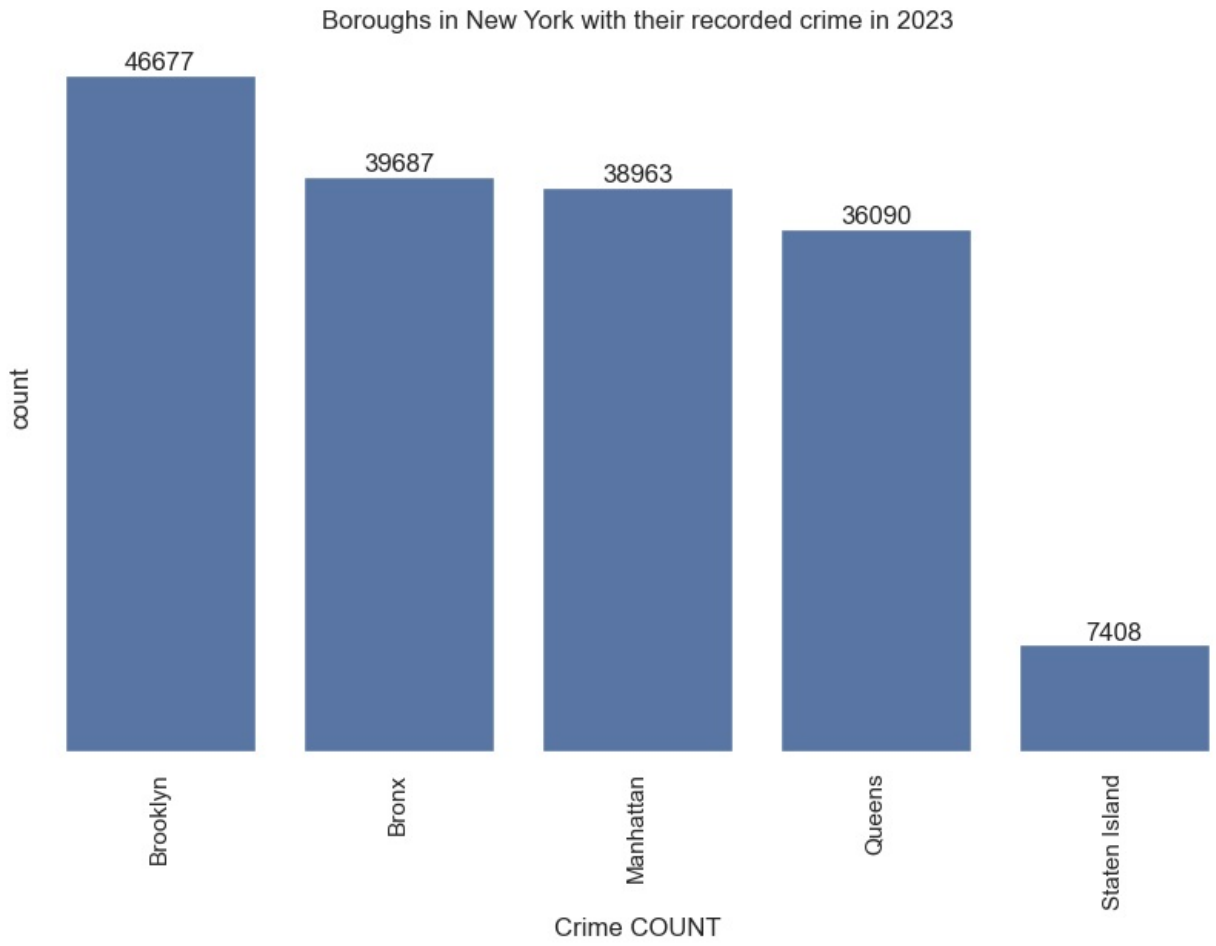
In [52]:
```
sample_m = sex_male.sample(n= 1000, random_state = 42, replace = True)
sample_f = sex_female.sample(n= 1000, random_state = 42, replace = True)
```

In [53]:
```
# Performing a two-sample t-test with a custom significance level
alpha = 0.05  # Set your desired significance level here

# Conduct the t-test
t_statistic, p_value = ttest_ind(sample_m["CRIME_COUNT"], sample_f["CRIME_COUNT"])

# Compare p-value with significance level
if p_value < alpha:
    print(f"Reject the null hypothesis. p-value: {p_value}, alpha: {alpha}")
else:
    print(f"Fail to reject the null hypothesis. p-value: {p_value}, alpha: {alpha}")
```

```
Fail to reject the null hypothesis. p-value: 0.10788298670912642, alpha: 0.05
```

- ttest statistics were perform on genders and their involvement in crime in New York in 2023. The ttest results fail to reject the null hypothesis, which indicates that there is no statistical significance different between male and female crime in New York.Both genders has the tendencies of committing various crime in New York.

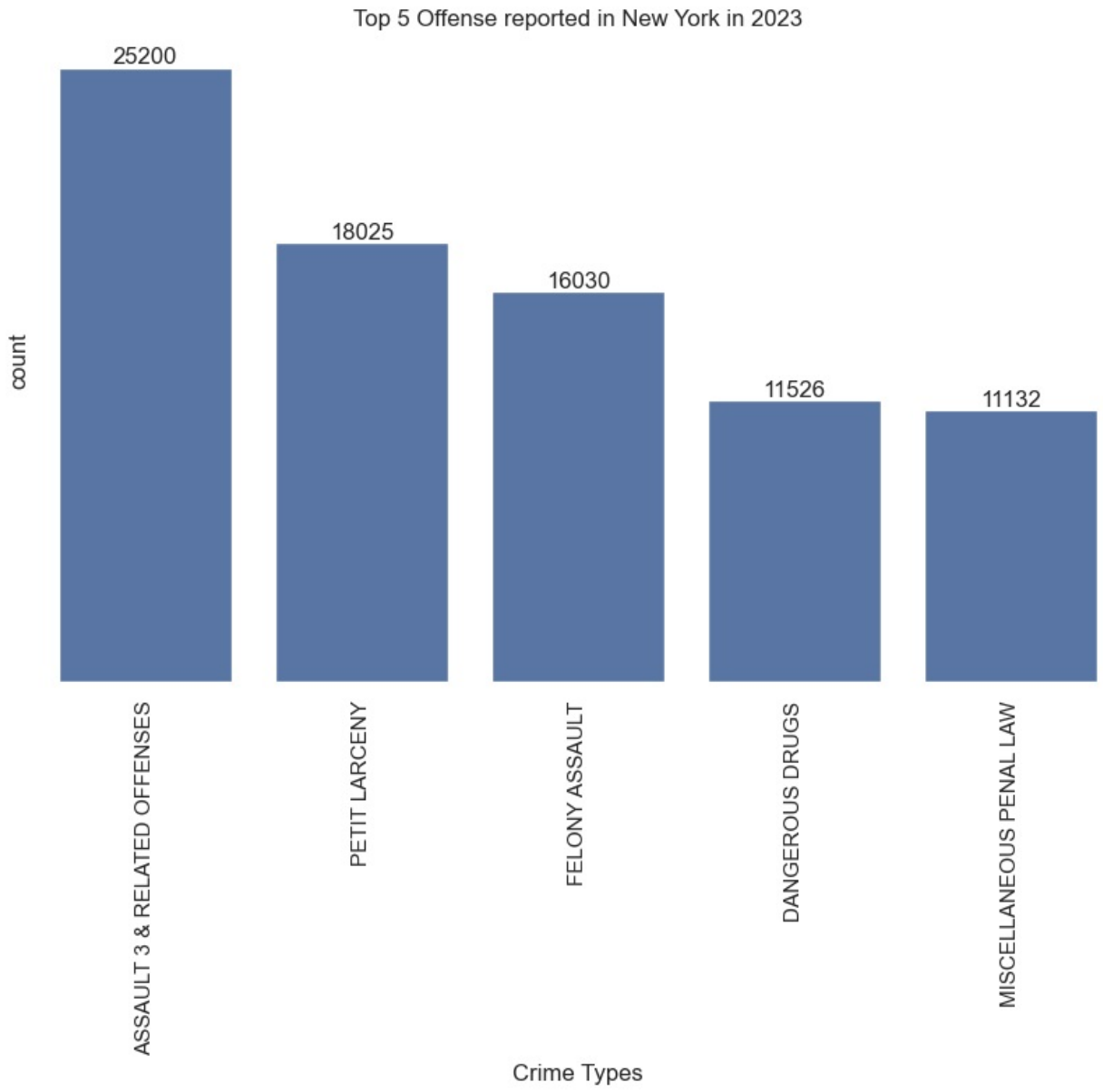
In [183…
```
plt.figure(figsize=(10, 6))
sns.set(style='darkgrid')
ax = sns.countplot(x = 'BORO',
              data = data_new,
              order = data_new['BORO'].value_counts().nlargest().index)

ax.set(xlabel='Crime COUNT', yticks=[], title='Boroughs in New York with their recorded crime in 2023', frame_o
ax.bar_label(container=ax.containers[0])
plt.xticks(rotation=90)
plt.show()
```



- Per the chart above Brooklyn has the most reported records of crime in New York followers by Bronx and Manhattan. Staten Island has the least recorded report of crime.

In [184…

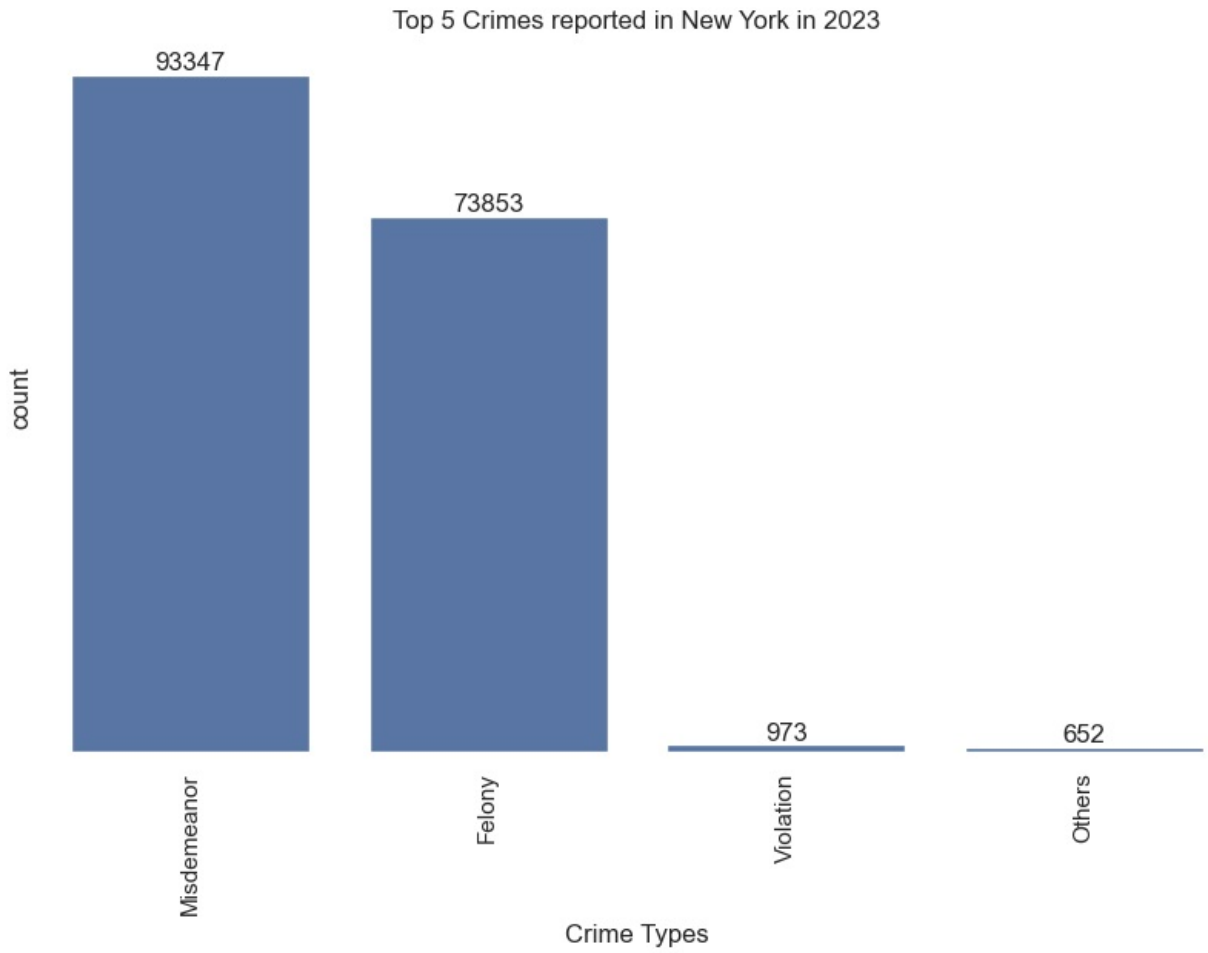
```
plt.figure(figsize=(10, 6))
sns.set(style='darkgrid')
ax = sns.countplot(x = 'OFNS_DESC',
              data = data_new,
              order = data_new['OFNS_DESC'].value_counts().nlargest().index)

ax.set(xlabel='Crime Types', yticks=[], title='Top 5 Offense reported in New York in 2023', frame_on=False) # p
ax.bar_label(container=ax.containers[0])
plt.xticks(rotation=90)
plt.show()
```



- The most reported offense in New York is Assault and it related offenses, followed by petit larceny, felony assault, dangerous drugs and miscellaneous penal law.

In [185…

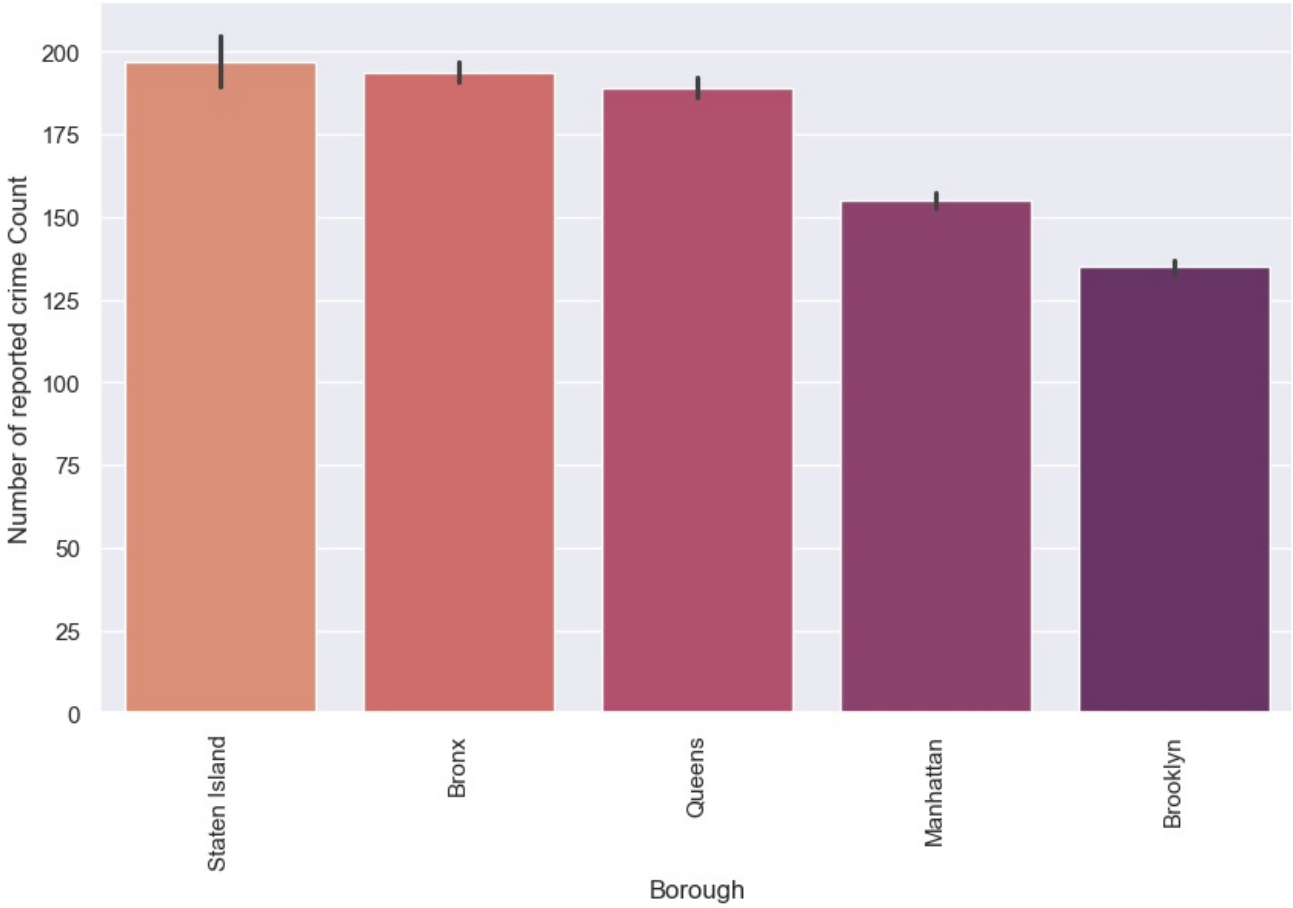
```
plt.figure(figsize=(10, 6))
sns.set(style='darkgrid')
ax = sns.countplot(x = 'LAW_CAT_CD',
              data = data_new,
              order = data_new['LAW_CAT_CD'].value_counts().nlargest().index)

ax.set(xlabel='Crime Types', yticks=[], title='Top 5 Crimes reported in New York in 2023', frame_on=False) # pr
ax.bar_label(container=ax.containers[0])
plt.xticks(rotation=90)
plt.show()
```



- The most reported crime in New York is Misdemeanor, it account for almost 60% of all crime committed in New York in 2023, followed by Felony and Violation.

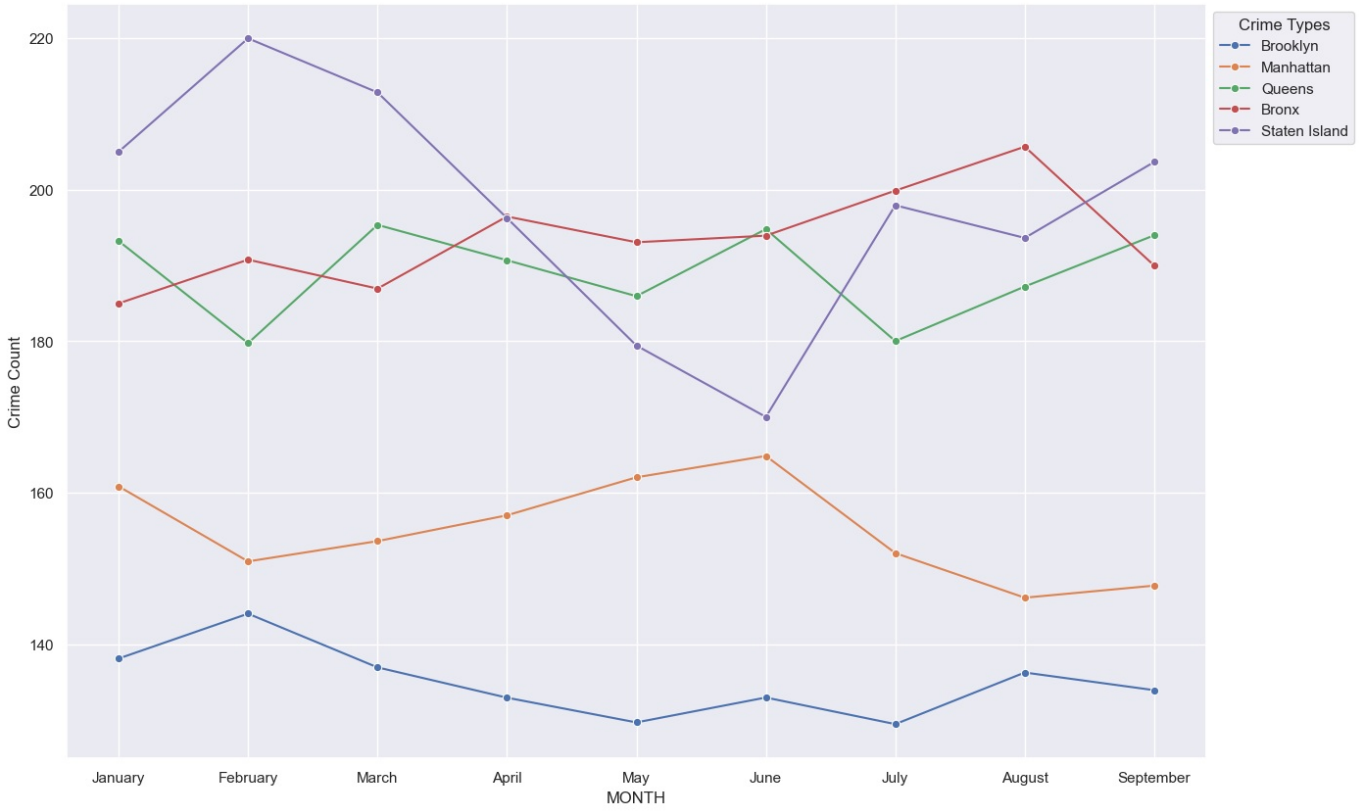
In [189…

```
plt.figure(figsize=(10, 6))
# Create a Seaborn bar plot with hue
ax =sns.barplot(x="BORO",
            y="Crime Count",
            data=data_new,
            palette ="flare",
            order=data_new.groupby("BORO")["Crime Count"].mean().head(5).sort_values(ascending=False).index)

# Set labels and title
plt.xlabel('Borough')
plt.ylabel('Number of reported crime Count')
plt.title('')

# Show the plot
plt.xticks(rotation=90)
plt.show()
```



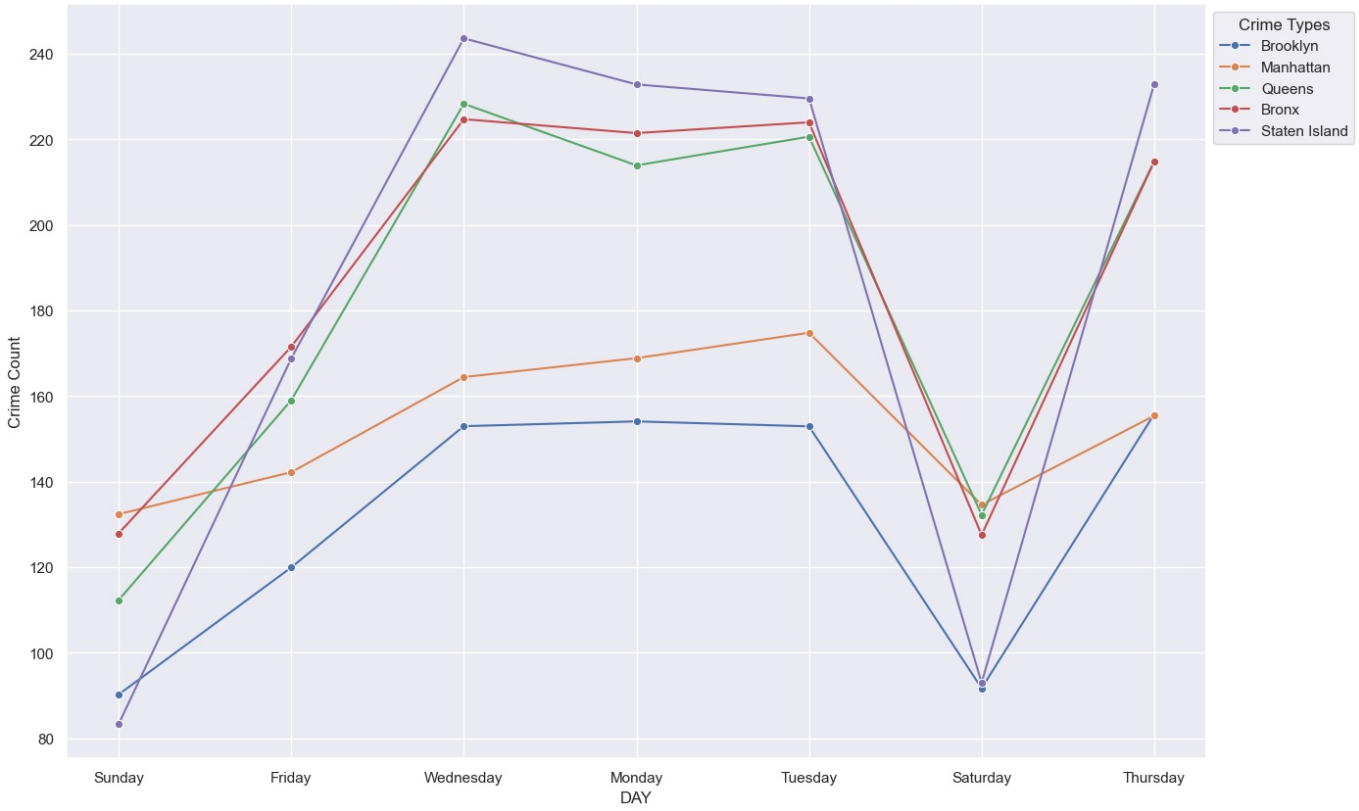
- Brooklyn has the highest reported crime cases in New York, but per average Staten Island has the highest average number of reported crime in New york. this maybe due to the following factor,Even though Brooklyn has the highest count of reported crime cases, it may also have a significantly larger population compared to Staten Island. Thus, when you calculate the average number of reported crimes per capita (or per square mile), Staten Island might have a higher average due to its smaller population and area.

In [58]:
```
#Plot a lineplot with hue and adjust legend
plt.figure(figsize=(15, 10))
sns.lineplot(x='MONTH',
             y='Crime Count',
             hue='BORO', data=data_new,
             marker='o',
             ci =None
            )

# Adjust the legend
plt.legend(title='Crime Types', bbox_to_anchor=(1, 1), loc='upper left')

# Show the plot
plt.show()
```



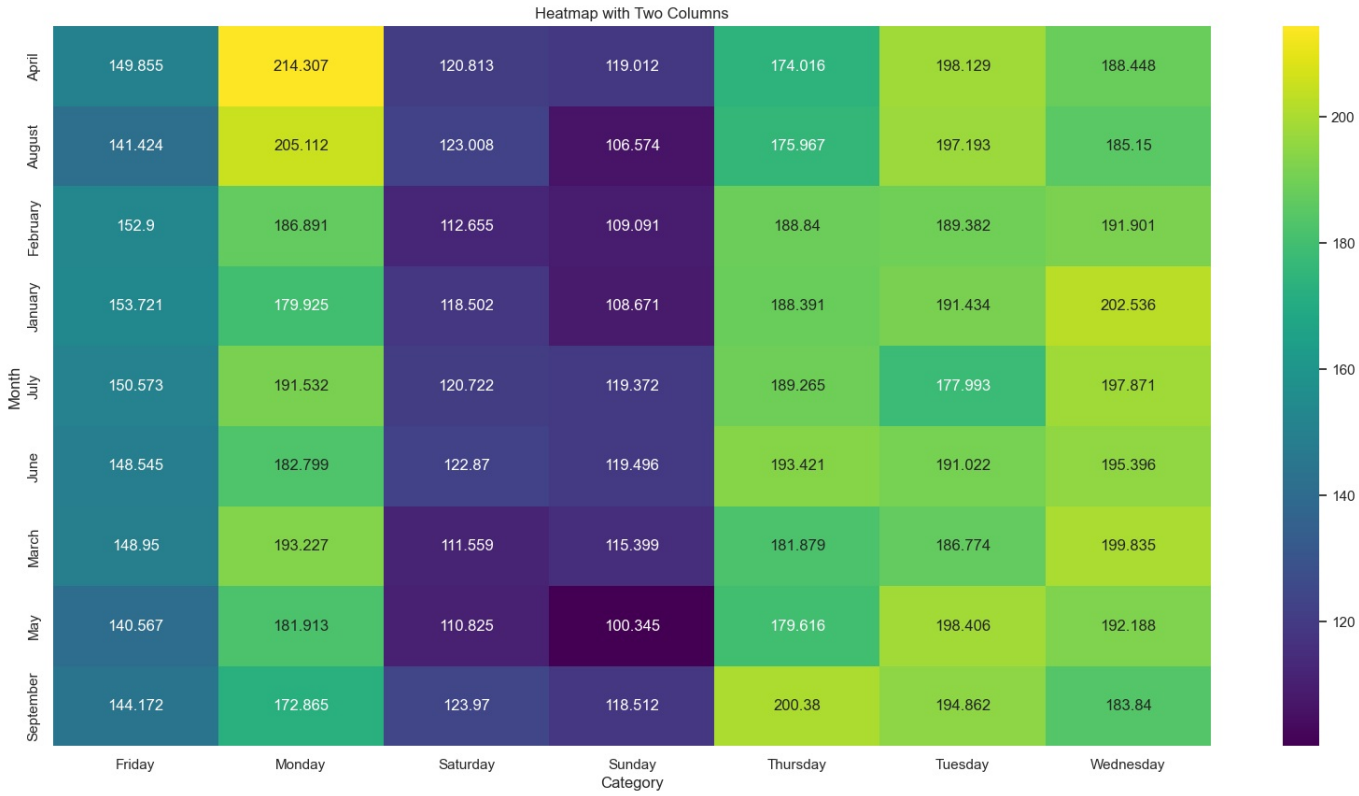
- Most of the borough in New York shows a significant increase of crime cases in June except Staten Island, which shows a drastic decline in June. In February shows a shows a peak of crime reported cases in Staten Island while other borough shows non significant changes in February. Also Staten Island and Queens shows a sign of increasing rate in September, while Bronx and Brooklyn shows a decline in the same month.

In [59]:
```
# Plot a lineplot with hue and adjust legend
plt.figure(figsize=(15, 10))
sns.lineplot(x='DAY',
             y='Crime Count',
             hue='BORO', data=data_new,
             marker='o',
             ci=None
            )

# Adjust the legend
plt.legend(title='Crime Types', bbox_to_anchor=(1, 1), loc='upper left')

# Show the plot
plt.show()
```



- From the graph above we can observed that crime occurrences decline on Sundays and start to peak on Thursdays in all the boroughs in New York in 2023.

In [60]:
```
# Pivot the DataFrame to create a matrix suitable for heatmap
heatmap_data = data_new.pivot_table(index='MONTH', columns='DAY', values='Crime Count', aggfunc='mean')

# Plot heatmap using Seaborn
plt.figure(figsize=(20, 10))
sns.heatmap(heatmap_data, annot=True, cmap='viridis', fmt='g', cbar=True)

# Customize the plot
plt.title('Heatmap with Two Columns')
plt.xlabel('Category')
plt.ylabel('Month')

# Show the plot
plt.show()
```



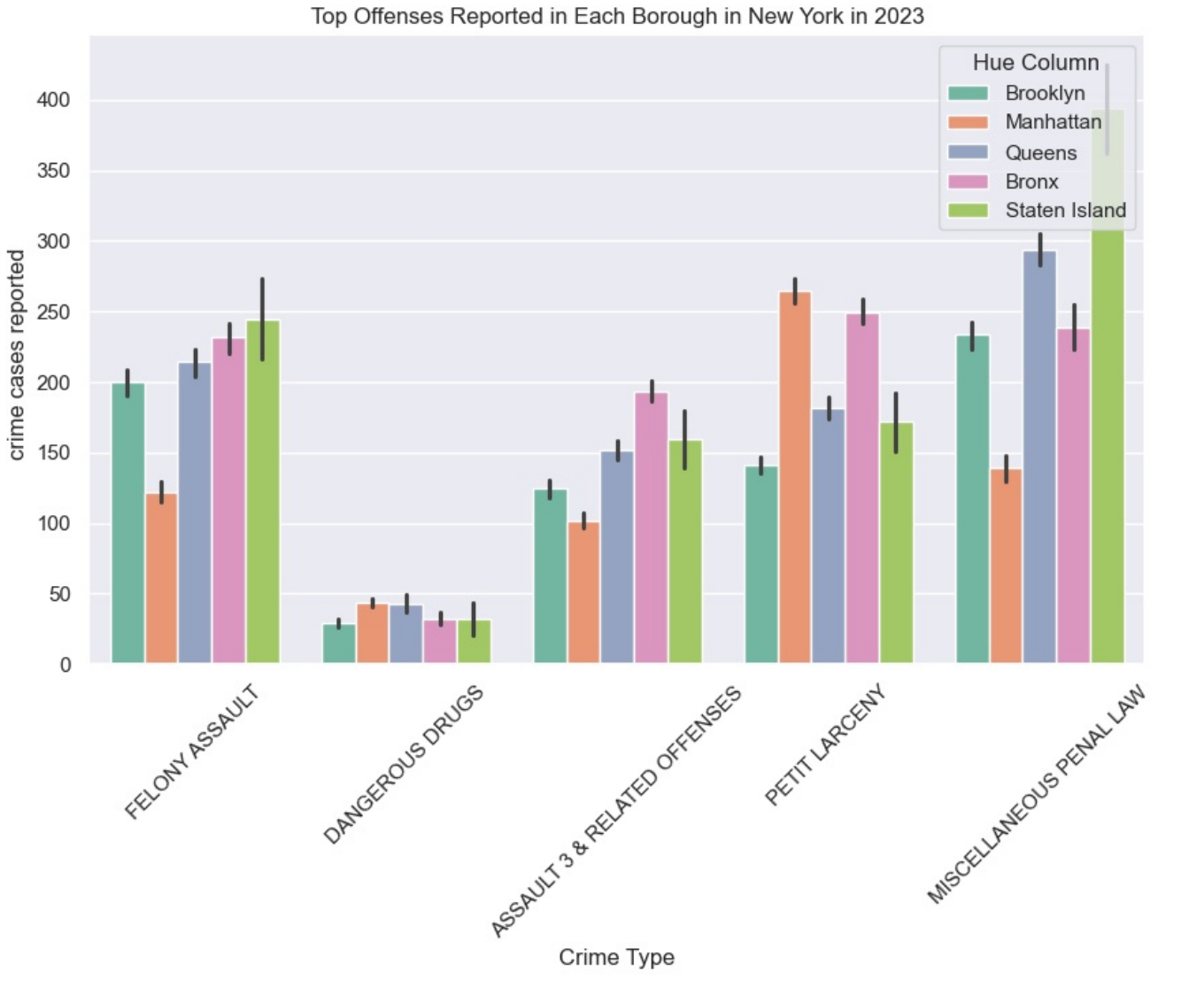
In [61]:
```
data_new.head(3)
```

Out[61]:

	ARREST_KEY	PD_CD	PD_DESC	KY_CD	OFNS_DESC	LAW_CODE	LAW_CAT_CD	ARREST_BORO	ARREST_PRECIN
0	261209118	109	ASSAULT 2,1,UNCLASSIFIED	106.0	FELONY ASSAULT	PL 1200501	Felony	K	
1	262984267	515	CONTROLLED SUBSTANCE,SALE 3	117.0	DANGEROUS DRUGS	PL 2203901	Felony	K	
2	263664549	105	STRANGULATION 1ST	106.0	FELONY ASSAULT	PL 1211200	Felony	K	

3 rows × 23 columns

In [192…
```
# Assuming 'category_column' is the categorical column to be sorted
sorted_categories = data_new['OFNS_DESC'].value_counts().index[:5]

# Filter the DataFrame to include only the top five sorted categories
df_top5 = data_new[data_new['OFNS_DESC'].isin(sorted_categories)]

# Plot barplot using Seaborn with hue
plt.figure(figsize=(10, 6))
sns.barplot(x='OFNS_DESC', y='Crime Count', hue='BORO', data=df_top5, palette='Set2')

# Customize the plot
plt.title('Top Offenses Reported in Each Borough in New York in 2023')
plt.xlabel('Crime Type')
plt.ylabel('crime cases reported ')
plt.legend(title='Hue Column',bbox_to_anchor=(1, 1), loc='upper right')
plt.xticks(rotation=45)
# Show the plot
plt.show()
```

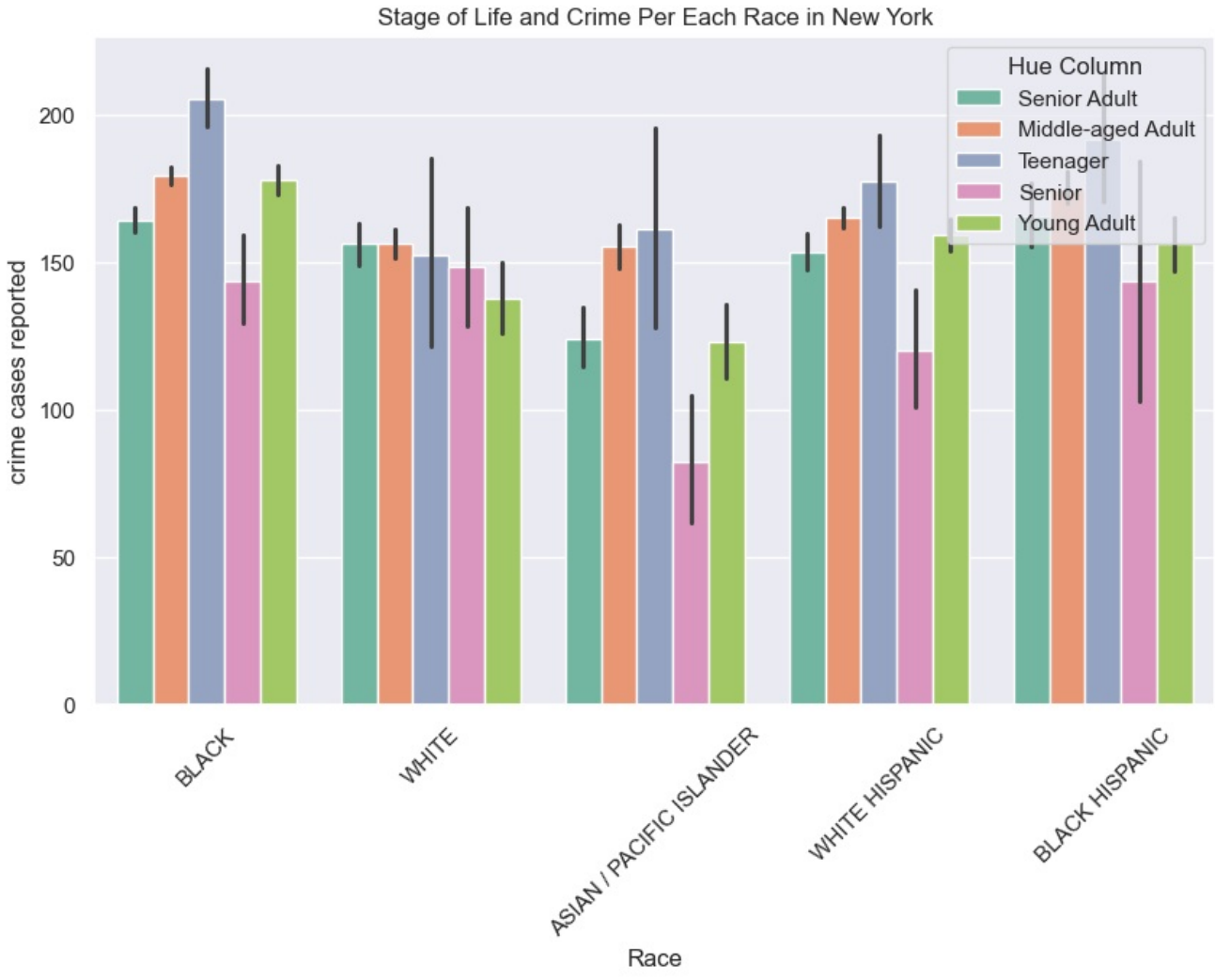


- From the above chart we can observed that Felony assault and miscellaneous are most reported offenses in Staten Island.

In [63]:
```
data_new.columns
```

Out[63]:
```
Index(['ARREST_KEY', 'PD_CD', 'PD_DESC', 'KY_CD', 'OFNS_DESC', 'LAW_CODE',
       'LAW_CAT_CD', 'ARREST_BORO', 'ARREST_PRECINCT', 'JURISDICTION_CODE',
       'PERP_SEX', 'PERP_RACE', 'X_COORD_CD', 'Y_COORD_CD', 'Latitude',
       'Longitude', 'New_Age', 'Group_Age', 'Crime Count', 'YEAR', 'MONTH',
       'DAY', 'BORO'],
      dtype='object')
```

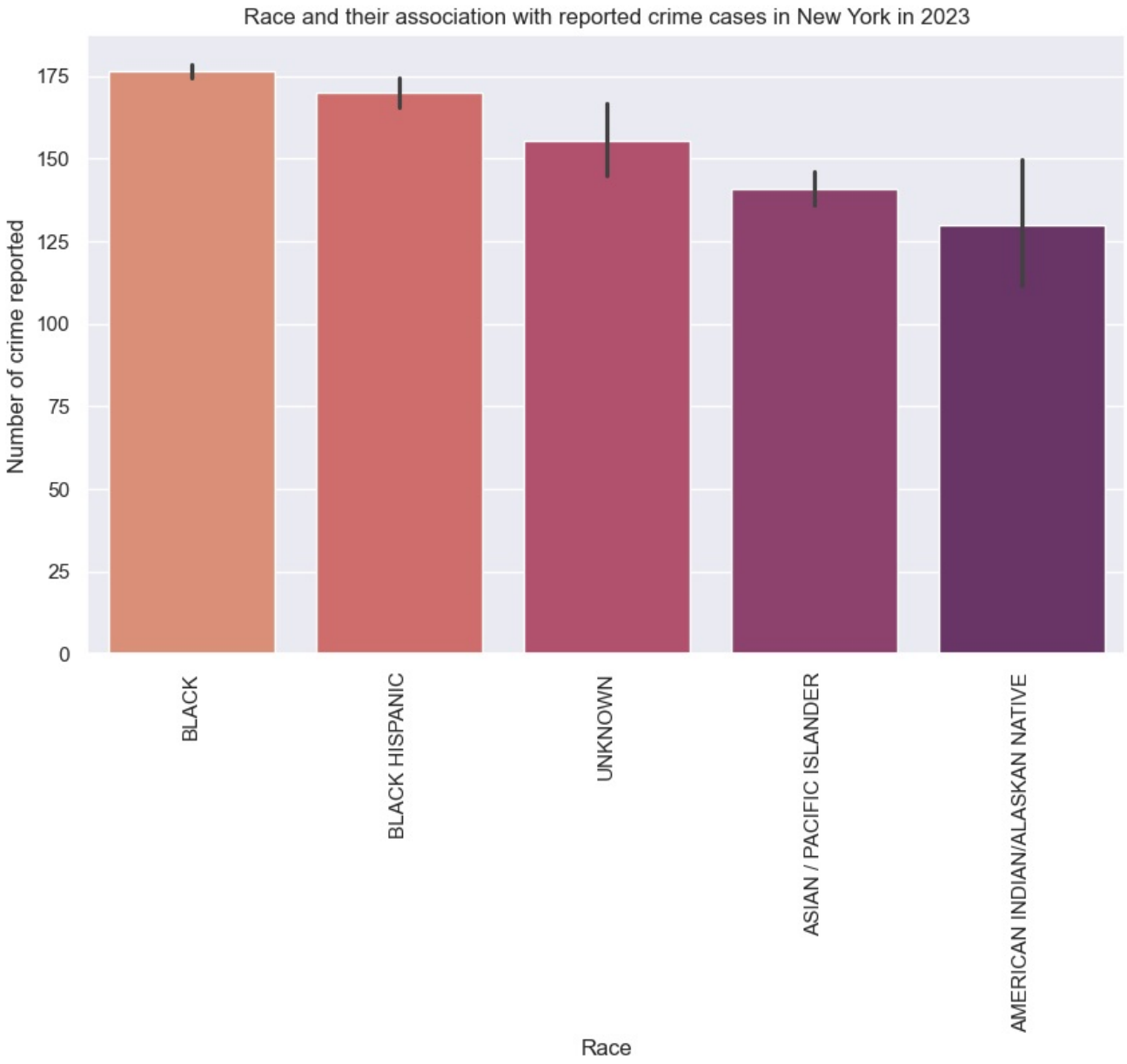
In [193…
```
# Assuming 'category_column' is the categorical column to be sorted
sorted_categories = data_new['PERP_RACE'].value_counts().index[:5]

# Filter the DataFrame to include only the top five sorted categories
df_top5 = data_new[data_new['PERP_RACE'].isin(sorted_categories)]

# Plot barplot using Seaborn with hue
plt.figure(figsize=(10, 6))
sns.barplot(x='PERP_RACE', y='Crime Count', hue='Group_Age', data=df_top5, palette='Set2')

# Customize the plot
plt.title('Stage of Life and Crime Per Each Race in New York')
plt.xlabel('Race')
plt.ylabel('crime cases reported')
plt.legend(title='Hue Column',bbox_to_anchor=(1, 1), loc='upper right')
plt.xticks(rotation=45)
# Show the plot
plt.show()
```



- From the above chart we can observed that teenagers dominates in all race concerning their association with crime except white.

In [194…
```
plt.figure(figsize=(10, 6))
# Create a Seaborn bar plot with hue
ax =sns.barplot(x="PERP_RACE",
            y="Crime Count",
            data=data_new,
            palette ="flare",
            order=data_new.groupby("PERP_RACE")["Crime Count"].mean().head(5).sort_values(ascending=False).index

# Set labels and title
plt.xlabel('Race')
plt.ylabel('Number of crime reported')
```

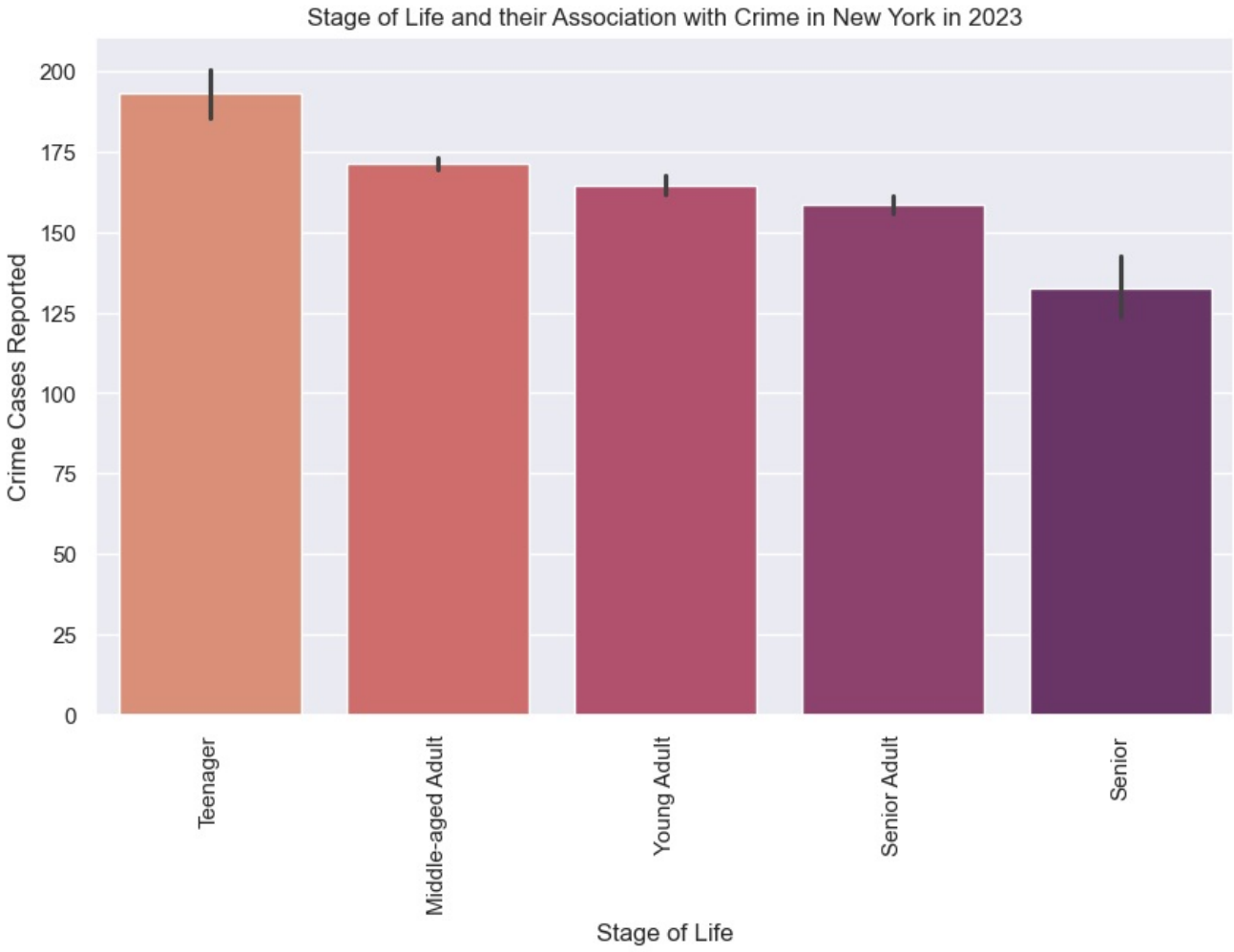
```
plt.title('Race and their association with reported crime cases in New York in 2023')

# Show the plot
plt.xticks(rotation=90)
plt.show()
```



- From the graph above we can observed that Black people dominates in crime reported cases in New York in 2023.

In [195…

```
plt.figure(figsize=(10, 6))
# Create a Seaborn bar plot with hue
ax =sns.barplot(x="Group_Age",
            y="Crime Count",
            data=data_new,
            palette ="flare",
            order=data_new.groupby("Group_Age")["Crime Count"].mean().head(5).sort_values(ascending=False).index

# Set labels and title
plt.xlabel('Stage of Life')
plt.ylabel('Crime Cases Reported')
plt.title('Stage of Life and their Association with Crime in New York in 2023')

# Show the plot
plt.xticks(rotation=90)
plt.show()
```



- • Teenagers' involvement in crime cases reported in New York in 2023, was the highest as compare to other stages of life followed by middle-age adults. This statistics is really concerning.

In [ ]:

In [ ]:

In [67]:
```
# convert pandas dataframe to geodataframe
geo_data = gpd.GeoDataFrame(data_new,
                            crs='EPSG:4326',
                            geometry=gpd.points_from_xy(data_new.Longitude, data_new.Latitude))
```

In [68]:
```
geo_data.head()
```

Out[68]:

	ARREST_KEY	PD_CD	PD_DESC	KY_CD	OFNS_DESC	LAW_CODE	LAW_CAT_CD	ARREST_BORO	ARREST_PRECIN
0	261209118	109	ASSAULT 2,1,UNCLASSIFIED	106.0	FELONY ASSAULT	PL 1200501	Felony	K	
1	262984267	515	CONTROLLED SUBSTANCE,SALE 3	117.0	DANGEROUS DRUGS	PL 2203901	Felony	K	
2	263664549	105	STRANGULATION 1ST	106.0	FELONY ASSAULT	PL 1211200	Felony	K	
3	261345231	105	STRANGULATION 1ST	106.0	FELONY ASSAULT	PL 1211200	Felony	M	
4	263536618	109	ASSAULT 2,1,UNCLASSIFIED	106.0	FELONY ASSAULT	PL 12005WX	Felony	K	

5 rows × 24 columns

In [69]:
```
# a random sample of 1500 theft cases in Vancauver wassubset for geospatial analysis
sample_geo = geo_data.sample(n=1000, random_state = 13490, replace = True)
```

In [70]:
```
sample_geo.head()
```

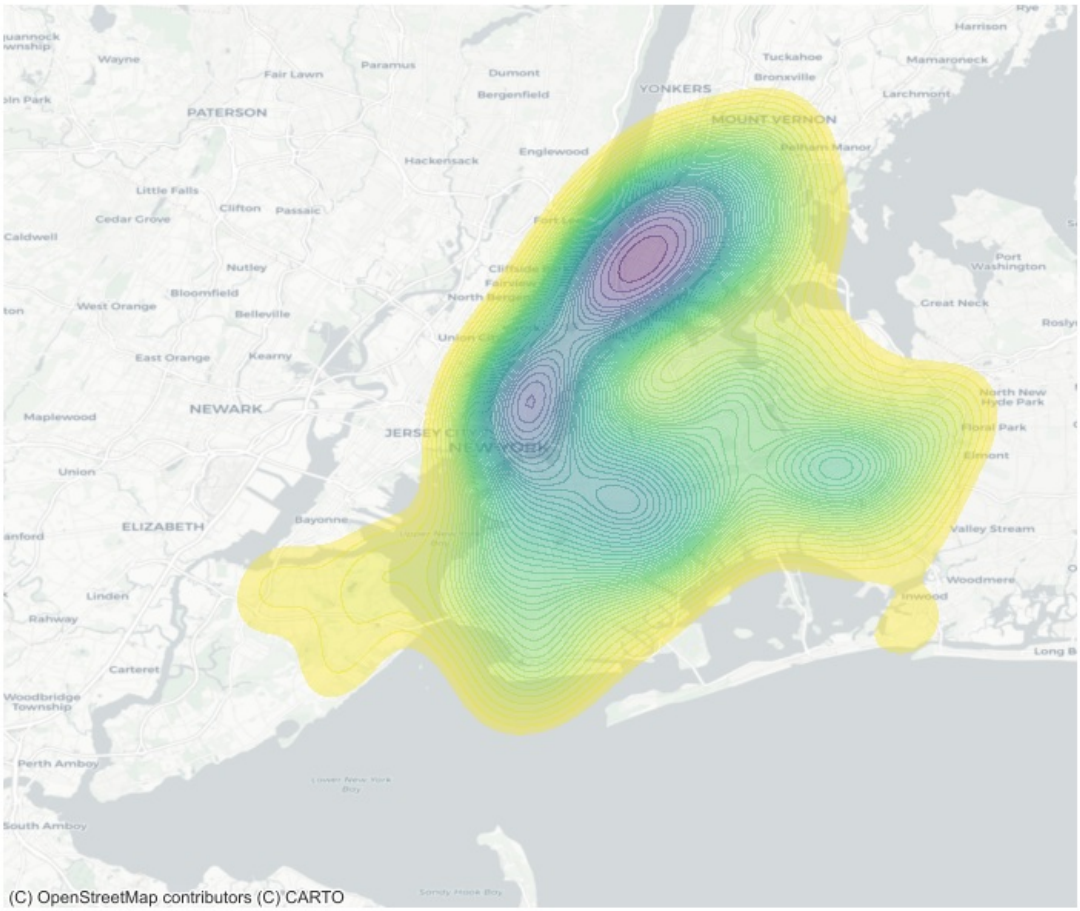
Out[70]:

	ARREST_KEY	PD_CD	PD_DESC	KY_CD	OFNS_DESC	LAW_CODE	LAW_CAT_CD	ARREST_BORO	ARRE
100644	270720256	268	CRIMINAL MIS 2 & 3	121.0	CRIMINAL MISCHIEF & RELATED OF	PL 1451000	Felony	M	
65776	263116720	339	LARCENY,PETIT FROM OPEN AREAS,	341.0	PETIT LARCENY	PL 1552500	Misdemeanor	M	
137342	262266916	502	CONTROLLED SUBSTANCE,POSSESS.	117.0	DANGEROUS DRUGS	PL 2201612	Felony	K	
72451	266701626	969	TRAFFIC,UNCLASSIFIED INFRACTIO	881.0	OTHER TRAFFIC INFRACTION	VTL051101A	Misdemeanor	B	
5564	264546689	101	ASSAULT 3	344.0	ASSAULT 3 & RELATED OFFENSES	PL 1200001	Misdemeanor	B	

5 rows × 24 columns

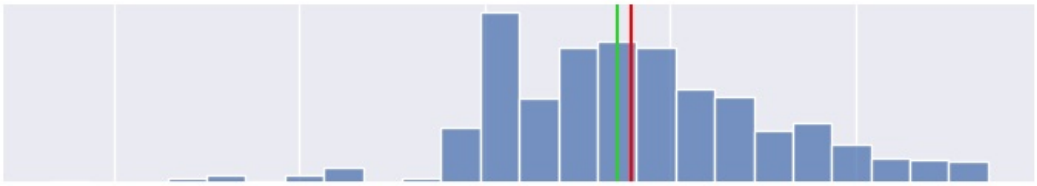
In [196…
```
mean_center = centrography.mean_center(sample_geo[["Longitude", "Latitude"]])
med_center = centrography.euclidean_median(sample_geo[["Longitude", "Latitude"]])
```

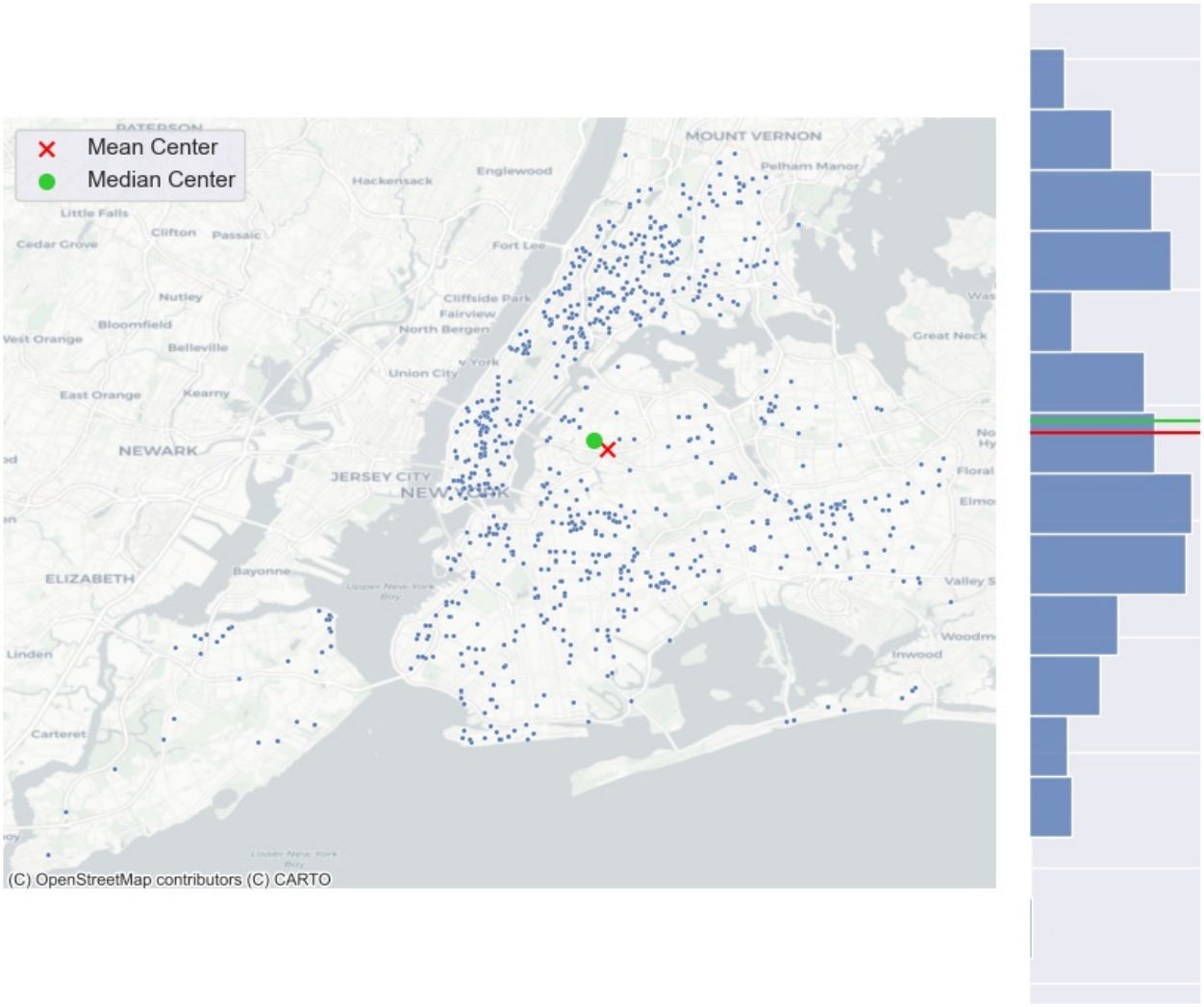
In [72]:
```
# Set up figure and axis
f, ax = plt.subplots(1, figsize=(9, 9))
# Generate and add KDE with a shading of 50 gradients
# coloured contours, 75% of transparency,
# and the reverse viridis colormap
sns.kdeplot(
    x="Longitude",
    y="Latitude",
    data=sample_geo,
    n_levels=50,
    shade=True,
    alpha=0.4,
    cmap="viridis_r",
)
# Add basemap
ctx.add_basemap(
    ax, crs=geo_data.crs,source=ctx.providers.CartoDB.Positron
)
# Remove axes
ax.set_axis_off()
```



In [197…

```
# Generate scatterplot
joint_axes = sns.jointplot(
    x="Longitude", y="Latitude", data=sample_geo, s=5, height=9
)
# Add mean point and marginal lines
joint_axes.ax_joint.scatter(
    *mean_center, color="red", marker="x", s=50, label="Mean Center"
)
joint_axes.ax_marg_x.axvline(mean_center[0], color="red")
joint_axes.ax_marg_y.axhline(mean_center[1], color="red")
# Add median point and marginal lines
joint_axes.ax_joint.scatter(
    *med_center,
    color="limegreen",
    marker="o",
    s=50,
    label="Median Center"
)
joint_axes.ax_marg_x.axvline(med_center[0], color="limegreen")
joint_axes.ax_marg_y.axhline(med_center[1], color="limegreen")
# Legend
joint_axes.ax_joint.legend()
# Add basemap
ctx.add_basemap(
    joint_axes.ax_joint, crs=geo_data.crs,source=ctx.providers.CartoDB.Positron
)
# Clean axes
joint_axes.ax_joint.set_axis_off()
# Display
plt.show()
```





- Though the kernel density shows a higher cluster of crime reported at the north, meanwhile the mean and median crime cases in the geographic area almost overlaps at the center of the map. This may suggests that despite the majority of crime are recorded in the north, the central tendency of the distribution is not heavily influenced by the northern outliers. Also there might be spatial variation in the dataset, while the northern part has high density , the distribution is relatively symmetric to balance.

In [73]:
```
coordinates = sample_geo[["Longitude", "Latitude"]].values
```

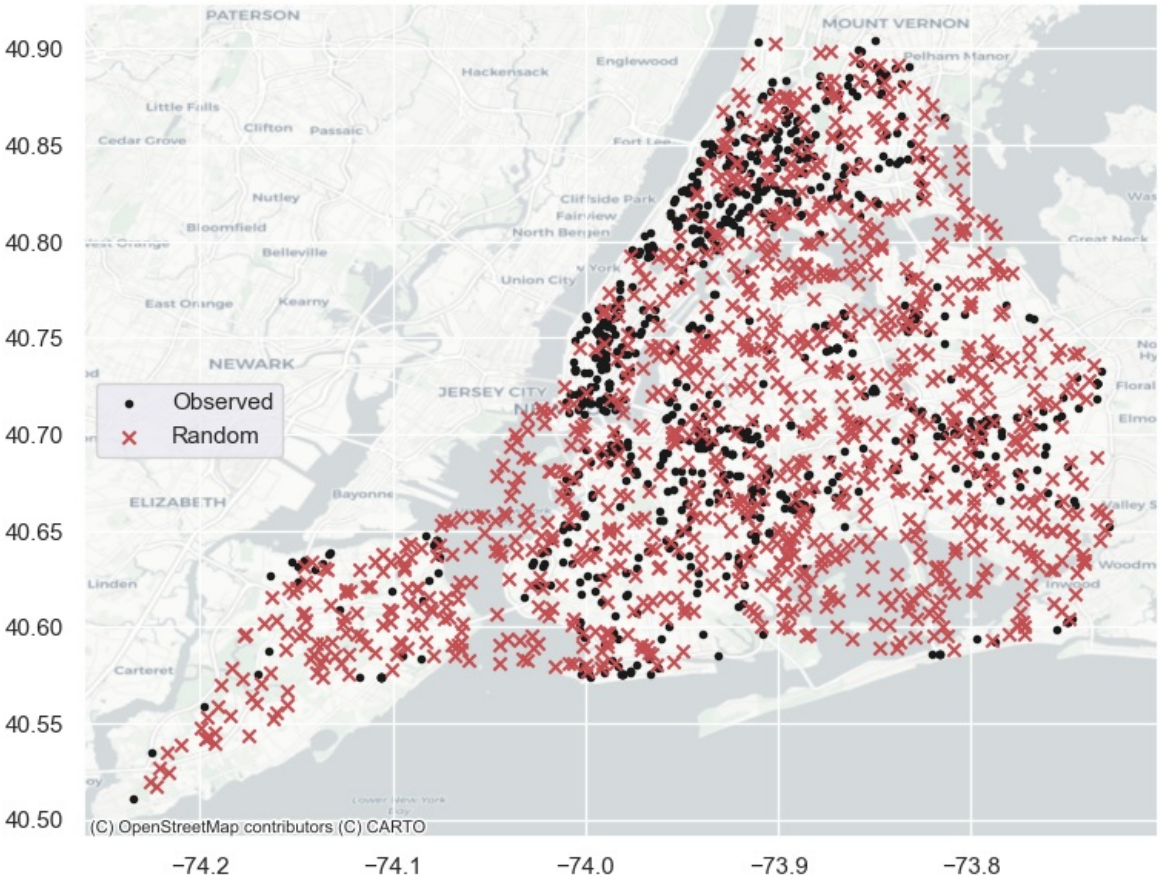
In [74]:
```
from pointpats import (
    distance_statistics,
    QStatistic,
    random,
    PointPattern,
)
```

In [75]:
```
import libpysal

alpha_shape, alpha, circs = libpysal.cg.alpha_shape_auto(
    coordinates, return_circles=True
)
```

In [76]:
```
random_pattern_ashape = random.poisson(
    alpha_shape, size=len(coordinates)
)
```

In [77]:
```
f, ax = plt.subplots(1, figsize=(9, 9))
plt.scatter(*coordinates.T, color="k", marker=".", label="Observed")
plt.scatter(
    *random_pattern_ashape.T, color="r", marker="x", label="Random"
)
ctx.add_basemap(
```

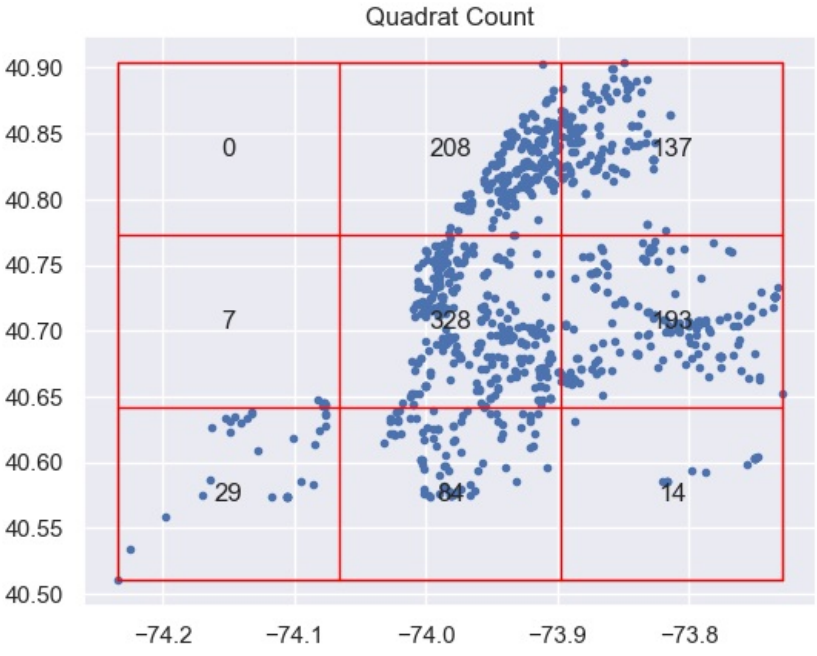
```
    ax, crs=geo_data.crs,source=ctx.providers.CartoDB.Positron
)
ax.legend(ncol=1, loc="center left")
plt.show()
```



- The map above provide a pattern derived from a known completely spatially random process,random data was generated using the Poisson point process concept to analysis a point patterns in our crime data. We can observed that, there more are clusters of crime reported 2023 at the north compare to the south of New York, These clusters are all statistical significant per our chi-square results. So we can state that Northern New York is more risky than the Southern areas.

In [78]:
```
qstat = QStatistic(coordinates)
qstat.plot()
```

Out[78]:
```
<Axes: title={'center': 'Quadrat Count'}>
```



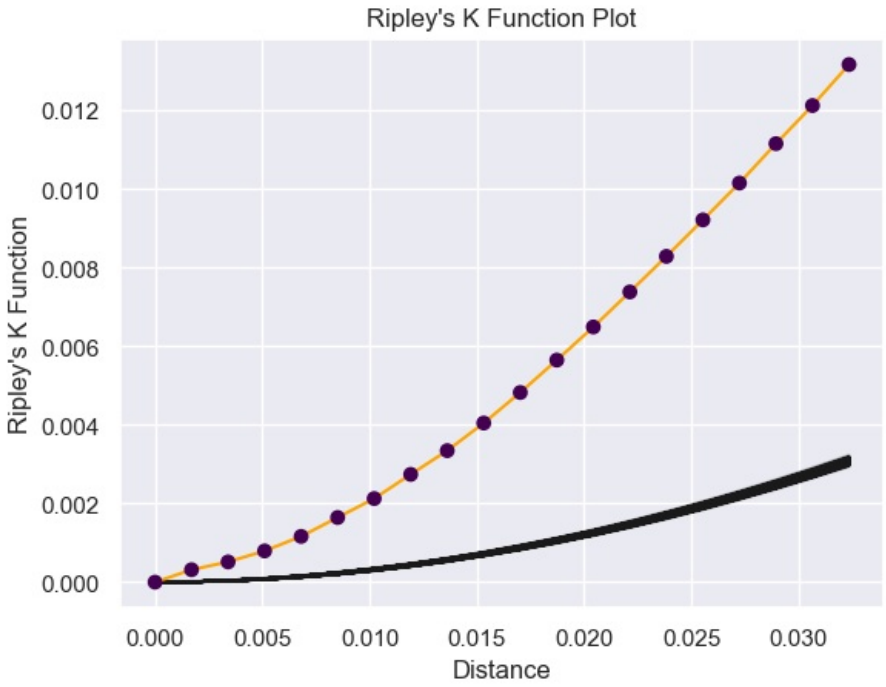
In [79]:
```
# Assuming you have calculated the chi-squared p-value
chi2_pvalue = qstat.chi2_pvalue

# Set your significance level (alpha)
alpha = 0.05
```

```
# Compare the p-value with the significance level
if chi2_pvalue < alpha:
    print(f"The chi-squared test is statistically significant at the {alpha} level. Reject the null hypothesis.
else:
    print(f"The chi-squared test is not statistically significant at the {alpha} level. Fail to reject the null
```

```
The chi-squared test is statistically significant at the 0.05 level. Reject the null hypothesis.
```

In [80]:
```
k_test = distance_statistics.k_test(sample_geo[['Longitude','Latitude']].values, keep_simulations=True)
```

In [81]:
```
# Assuming you have a point pattern called 'pp'
plt.plot(k_test.support, k_test.simulations.T, color='k', alpha=.01)
plt.plot(k_test.support, k_test.statistic, color='orange')
plt.scatter(
    k_test.support,
    k_test.statistic,
    cmap='viridis',
    c=k_test.pvalue < .05,
    zorder=4
)
plt.xlabel('Distance')
plt.ylabel('Ripley\'s K Function')
plt.title('Ripley\'s K Function Plot')
plt.show()

# Access the p-value
p_value = k_test.pvalue
print(f"P-value: {p_value}")
```




```
P-value: [0.     0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001
 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001 0.0001]
```

- With the Ripley's K plot too, we can see that the observed data is well above that of the simulated data, which confirms again that crime cases in New York is from a process that is not spatially random.

In [82]:
```
import folium
from folium.plugins import MarkerCluster
```

In [83]:
```
sample_geo.head(3)
```

Out[83]:

	ARREST_KEY	PD_CD	PD_DESC	KY_CD	OFNS_DESC	LAW_CODE	LAW_CAT_CD	ARREST_BORO	ARRE
100644	270720256	268	CRIMINAL MIS 2 & 3	121.0	CRIMINAL MISCHIEF & RELATED OF	PL 1451000	Felony	M	
65776	263116720	339	LARCENY,PETIT FROM OPEN AREAS,	341.0	PETIT LARCENY	PL 1552500	Misdemeanor	M	
137342	262266916	502	CONTROLLED SUBSTANCE,POSSESS.	117.0	DANGEROUS DRUGS	PL 2201612	Felony	K	

3 rows × 24 columns

In [177…
```
sample_geo["LAW_CAT_CD"].value_counts()
```

Out[177…
```
LAW_CAT_CD
Misdemeanor    569
Felony         418
Violation        9
Others           4
Name: count, dtype: int64
```

In [85]:
```
# Initialize Folium map centered on Vancouver
map_new = folium.Map(location=[40.712776, -74.005974], zoom_start=12)

# Create a MarkerCluster layer
marker_cluster = MarkerCluster().add_to(map_new)

# Iterate through DataFrame rows and add markers to the map
for index, row in sample_geo.iterrows():
    # Define marker color based on crime category
    if row['LAW_CAT_CD'] == 'Misdemeanor':
        marker_color = 'blue'
    elif row['LAW_CAT_CD'] == 'Felony':
        marker_color = 'red'
    elif row['LAW_CAT_CD'] == 'Violation':
        marker_color = 'green'
    else:
        marker_color = 'orange'

    # Create marker and add it to MarkerCluster layer
    folium.Marker([row['Latitude'], row['Longitude']],
                  popup=f"LAW_CAT_CD: {row['LAW_CAT_CD']}, Count: {row['Crime Count']}",
                  icon=folium.Icon(color=marker_color, icon='info-sign')).add_to(marker_cluster)

# Display the map
map_new
```

Out[85]: Make this Notebook Trusted to load map: File -> Trust Notebook

- In conclusion, the analysis of crime in New York City in 2023 reveals several key findings. Firstly, the ANOVA test indicates a significant difference in crime occurrence among the various boroughs, rejecting the null hypothesis. Additionally, while most crime categories show statistical significance, misdemeanor and violation categories do not occur randomly. A chi-square test demonstrates a strong relationship between race and crime types, although some expected cell frequencies did not meet assumptions. However, gender does not show a statistically significant difference in involvement in crime. Brooklyn has the highest count of reported crimes, but Staten Island has the highest average per capita, likely due to its smaller population. The most common offense is assault, followed by petit larceny and felony assault. Misdemeanor crimes account for the majority of reported incidents. Notably, there are fluctuations in crime rates across boroughs and months, with Staten Island showing unique patterns such as a decline in June and a peak in February. Overall, the analysis highlights the complex nature of crime distribution and its correlation with demographic factors in New York City.

- https://www.linkedin.com/in/ababio-benjamin-92385616b

In [ ]: