In [ ]:
```
import pandas as pd
import numpy as np
import math
import ipywidgets as widgets

##Seaborn for fancy plots.
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns


import geopandas as gpd
from shapely.geometry import Point
#import fiona

plt.rcParams['figure.figsize'] = (20, 9)
plt.style.use('Solarize_Light2')

import folium # mapping
from folium.plugins import HeatMap
import scikit_posthocs as sp
```

> We noticed earlier, from previous analysis of the Violent Crimes committed in Edmonton, that the Top 6 Neighborhoods that keep showing up are:
>
> - Downtown
> - McCauley
> - Central McDougall
> - Boyle Street
> - Oliver, and
> - Alberta Avenue

> To investigate further, we decided to look at Employment Status, Income Levels and Education Levels by Neighborhood. Unfortunately, the only data available was for the year 2016. So the further investigation from here will only be done based on the year 2016.

In [ ]:
```
df=pd.read_csv('3960_2016.csv', encoding_errors='ignore')
df.drop(['Descriptive Name'], axis=1, inplace =True)
df.drop(['Unnamed: 0'], axis=1, inplace =True)
#df['Avg_Temp']=(df['Air Temp. Avg. Max. (°C)']+df['Air Temp. Avg. Min. (°C)'])/2
df.drop(['Air Temp. Avg. Max. (°C)'], axis=1, inplace =True)
df.drop(['Air Temp. Avg. Min. (°C)'], axis=1, inplace =True)
df.drop(['NGH_Number'], axis=1, inplace =True)
df.drop(['DT_Year'], axis=1, inplace =True) # we are only dealing with 2016
df.drop(['DT_Month'], axis=1, inplace =True)
```

```
df.rename(columns={'Edu_Preschool': 'Preschool', 'Edu_Kindergarten ':'Kindergarten', '
df.rename(columns={'Edu_Gr.10 - Gr.12': 'Gr10_Gr12', 'Edu_Post Secondar':'Post_Seconda
display('Dataset Columns: ',df.columns)
print()
display('Dataset Information:', df.info())
print()
display('Null Values?:', df.isnull().sum())
df.head()
```

```
'Dataset Columns: '
Index(['NGH_Name', 'Latitude', 'Longitude', 'Violation_Type',
       'Sum_Occurrences', 'Preschool', 'Kindergarten', 'Gr7_Gr9', 'Gr10_Gr12',
       'Post_Secondary', 'Homemaker', 'Employedage0-30', 'Employed 30+ ',
       'Unemployed', 'Retired', 'Permanently U', 'Employment_No Response',
       'Income_Less than $30,', 'Income_$30,000 to les',
       'Income_$60,000 to les', 'Income_$100,000 to le',
       'Income_$125,000 to le', 'Income_$150,000 to le',
       'Income_$200,000 to le', 'Income_$250,000 or mo', 'Income_No Response'],
      dtype='object')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11132 entries, 0 to 11131
Data columns (total 26 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   NGH_Name                11132 non-null  object
 1   Latitude                11132 non-null  float64
 2   Longitude               11132 non-null  float64
 3   Violation_Type          11132 non-null  object
 4   Sum_Occurrences         11132 non-null  int64
 5   Preschool               11113 non-null  float64
 6   Kindergarten            11113 non-null  float64
 7   Gr7_Gr9                 11113 non-null  float64
 8   Gr10_Gr12               11113 non-null  float64
 9   Post_Secondary          11113 non-null  float64
 10  Homemaker               11113 non-null  float64
 11  Employedage0-30         11113 non-null  float64
 12  Employed 30+            11113 non-null  float64
 13  Unemployed              11113 non-null  float64
 14  Retired                 11113 non-null  float64
 15  Permanently U           11113 non-null  float64
 16  Employment_No Response  11113 non-null  float64
 17  Income_Less than $30,   11113 non-null  float64
 18  Income_$30,000 to les   11113 non-null  float64
 19  Income_$60,000 to les   11113 non-null  float64
 20  Income_$100,000 to le   11113 non-null  float64
 21  Income_$125,000 to le   11113 non-null  float64
 22  Income_$150,000 to le   11113 non-null  float64
 23  Income_$200,000 to le   11113 non-null  float64
 24  Income_$250,000 or mo   11113 non-null  float64
 25  Income_No Response      11113 non-null  float64
dtypes: float64(23), int64(1), object(2)
memory usage: 2.2+ MB
'Dataset Information:'
None

'Null Values?:'
```

```
NGH_Name                   0
Latitude                   0
Longitude                  0
Violation_Type             0
Sum_Occurrences            0
Preschool                 19
Kindergarten              19
Gr7_Gr9                   19
Gr10_Gr12                 19
Post_Secondary            19
Homemaker                 19
Employedage0-30           19
Employed 30+              19
Unemployed                19
Retired                   19
Permanently U             19
Employment_No Response    19
Income_Less than $30,     19
Income_$30,000 to les     19
Income_$60,000 to les     19
Income_$100,000 to le     19
Income_$125,000 to le     19
Income_$150,000 to le     19
Income_$200,000 to le     19
Income_$250,000 or mo     19
Income_No Response        19
dtype: int64
```

Out[ ]:

	NGH_Name	Latitude	Longitude	Violation_Type	Sum_Occurrences	Preschool	Kindergarten
0	ABBOTTSFIELD	53.574143	-113.388758	Assault	4	159.0	189.0
1	ABBOTTSFIELD	53.574143	-113.388758	Theft From Vehicle	1	159.0	189.0
2	ABBOTTSFIELD	53.574143	-113.388758	Theft From Vehicle	1	159.0	189.0
3	ABBOTTSFIELD	53.574143	-113.388758	Assault	1	159.0	189.0
4	ABBOTTSFIELD	53.574143	-113.388758	Assault	1	159.0	189.0

5 rows × 26 columns

In [ ]:
```
df.describe().T
```

Out[ ]:

	count	mean	std	min	25%	50%	75
Latitude	11132.0	53.534315	0.058982	53.399507	53.488707	53.539767	53.5792(
Longitude	11132.0	-113.503369	0.078221	-113.701466	-113.558327	-113.499421	-113.4443!
Sum_Occurrences	11132.0	2.533148	2.779805	1.000000	1.000000	2.000000	3.0000(
Preschool	11113.0	130.877711	132.198913	0.000000	51.000000	103.000000	180.0000(
Kindergarten	11113.0	167.029245	164.464642	0.000000	58.000000	139.000000	231.0000(
Gr7_Gr9	11113.0	67.769639	60.997749	0.000000	22.000000	60.000000	98.0000(
Gr10_Gr12	11113.0	56.901197	50.458302	0.000000	18.000000	48.000000	83.0000(
Post_Secondary	11113.0	100.428597	158.738916	0.000000	31.000000	66.000000	119.0000(
Homemaker	11113.0	71.141996	63.947275	0.000000	28.000000	60.000000	96.0000(
Employedage0-30	11113.0	149.869972	118.742042	0.000000	68.000000	139.000000	207.0000(
Employed 30+	11113.0	933.108612	835.523202	0.000000	453.000000	814.000000	1249.0000(
Unemployed	11113.0	129.808783	110.350061	0.000000	49.000000	114.000000	196.0000(
Retired	11113.0	303.955008	272.178128	0.000000	122.000000	281.000000	423.0000(
Permanently U	11113.0	37.156303	42.232148	0.000000	8.000000	27.000000	48.0000(
Employment_No Response	11113.0	1155.861244	1131.229481	0.000000	434.000000	959.000000	1582.0000(
Income_Less than $30,	11113.0	133.869882	197.913557	0.000000	25.000000	72.000000	152.0000(
Income_$30,000 to les	11113.0	185.541708	194.515295	0.000000	60.000000	152.000000	244.0000(
Income_$60,000 to les	11113.0	180.655629	186.686130	0.000000	73.000000	159.000000	237.0000(
Income_$100,000 to le	11113.0	79.569153	85.396937	0.000000	30.000000	62.000000	99.0000(
Income_$125,000 to le	11113.0	48.586790	53.324445	0.000000	16.000000	35.000000	63.0000(
Income_$150,000 to le	11113.0	45.087285	54.436444	0.000000	11.000000	28.000000	58.0000(
Income_$200,000 to le	11113.0	18.323675	23.474115	0.000000	3.000000	10.000000	25.0000(
Income_$250,000 or mo	11113.0	17.155044	27.823724	0.000000	2.000000	5.000000	19.0000(
Income_No Response	11113.0	719.476109	797.157981	0.000000	308.000000	570.000000	924.0000(

We have a lot of ZEROS for the columns ['Preschool', 'Kindergarten', 'Gr7_Gr9', 'Gr10_Gr12', 'Post_Secondary', 'Homemaker', 'Employedage0-30', 'Employed 30+ ', 'Unemployed', 'Retired',

'Permanently U', 'Employment_No Response', 'Income*Less than $30,*', '*Income*30,000 to les', 'Income_60,000 to les', 'Income*$100,000 to le*', '*Income*125,000 to le', 'Income_150,000 to le', 'Income*$200,000 to le*', '*Income*$250,000 or mo', 'Income_No Response'].

These could be just Industrial Areas: we need to get rid of them for now.

In [ ]:
```
df.columns
```

Out[ ]:
```
Index(['NGH_Name', 'Latitude', 'Longitude', 'Violation_Type',
       'Sum_Occurrences', 'Preschool', 'Kindergarten', 'Gr7_Gr9', 'Gr10_Gr12',
       'Post_Secondary', 'Homemaker', 'Employedage0-30', 'Employed 30+ ',
       'Unemployed', 'Retired', 'Permanently U', 'Employment_No Response',
       'Income_Less than $30,', 'Income_$30,000 to les',
       'Income_$60,000 to les', 'Income_$100,000 to le',
       'Income_$125,000 to le', 'Income_$150,000 to le',
       'Income_$200,000 to le', 'Income_$250,000 or mo', 'Income_No Response'],
      dtype='object')
```

In [ ]:
```
# Filter out those columns to only values > 0
df = df[(df[['Preschool', 'Kindergarten', 'Gr7_Gr9', 'Gr10_Gr12', 'Post_Secondary',
             'Homemaker', 'Employedage0-30', 'Employed 30+ ', 'Unemployed', 'Retired',
             'Employment_No Response', 'Income_Less than $30,', 'Income_$30,000 to les
             'Income_$100,000 to le', 'Income_$125,000 to le', 'Income_$150,000 to le'
             'Income_$250,000 or mo', 'Income_No Response']] > 0).all(axis=1)]
```

In [ ]:
```
df.describe().T
```

Out[ ]:

	count	mean	std	min	25%	50%	75%
Latitude	8864.0	53.535489	0.059773	53.401301	53.485677	53.539767	53.58795
Longitude	8864.0	-113.501117	0.075535	-113.701352	-113.550982	-113.500798	-113.44223
Sum_Occurrences	8864.0	2.697766	2.890208	1.000000	1.000000	2.000000	3.00000
Preschool	8864.0	160.929377	131.278290	6.000000	87.000000	126.000000	199.00000
Kindergarten	8864.0	206.018276	161.658096	8.000000	109.000000	169.000000	261.00000
Gr7_Gr9	8864.0	83.468186	58.241845	1.000000	41.000000	71.000000	106.00000
Gr10_Gr12	8864.0	70.238042	47.746909	2.000000	37.000000	58.000000	91.00000
Post_Secondary	8864.0	124.183326	169.420230	2.000000	51.000000	87.000000	132.00000
Homemaker	8864.0	87.524143	61.149328	3.000000	48.000000	76.000000	111.00000
Employedage0-30	8864.0	183.660199	107.475463	4.000000	111.000000	157.000000	226.00000
Employed 30+	8864.0	1148.988042	797.394227	21.000000	670.000000	961.000000	1352.00000
Unemployed	8864.0	157.877256	102.757239	1.000000	82.000000	135.000000	210.00000
Retired	8864.0	371.346232	260.693042	14.000000	227.000000	337.000000	462.00000
Permanently U	8864.0	44.503610	42.220143	1.000000	19.000000	34.000000	52.00000
Employment_No Response	8864.0	1392.209612	1115.324640	80.000000	666.000000	1157.000000	1701.00000
Income_Less than $30,	8864.0	161.221909	207.608716	2.000000	51.000000	96.000000	179.00000
Income_$30,000 to les	8864.0	226.931182	195.471096	4.000000	118.000000	181.000000	282.00000
Income_$60,000 to les	8864.0	222.185808	186.541445	3.000000	124.000000	190.000000	265.00000
Income_$100,000 to le	8864.0	98.241990	85.861462	1.000000	48.000000	75.000000	111.00000
Income_$125,000 to le	8864.0	60.105257	53.806716	1.000000	27.750000	44.000000	71.00000
Income_$150,000 to le	8864.0	55.868005	55.945983	2.000000	21.750000	40.000000	69.00000
Income_$200,000 to le	8864.0	22.722022	24.351290	1.000000	6.000000	14.000000	29.00000
Income_$250,000 or mo	8864.0	21.349616	29.644906	1.000000	4.000000	8.000000	23.00000
Income_No Response	8864.0	874.281814	809.206171	91.000000	470.000000	682.000000	974.00000

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
NGH_Name                  0
Latitude                  0
Longitude                 0
Violation_Type            0
Sum_Occurrences           0
Preschool                 0
Kindergarten              0
Gr7_Gr9                   0
Gr10_Gr12                 0
Post_Secondary            0
Homemaker                 0
Employedage0-30           0
Employed 30+              0
Unemployed                0
Retired                   0
Permanently U             0
Employment_No Response    0
Income_Less than $30,     0
Income_$30,000 to les     0
Income_$60,000 to les     0
Income_$100,000 to le     0
Income_$125,000 to le     0
Income_$150,000 to le     0
Income_$200,000 to le     0
Income_$250,000 or mo     0
Income_No Response        0
dtype: int64
```

Its seems we have no more nulls.

> The Null values represent non residential area, hence that is why we have Null values for Education levels, Income Levels and Employment Status levels.

In [ ]:
```
# make copy of original dataset df
df_copy=df.copy()
```

In [ ]:
```
df_copy['Violation_Type'].value_counts()
```

Out[ ]:
```
Theft From Vehicle    2362
Break and Enter       1904
Theft Of Vehicle      1660
Assault               1579
Robbery                621
Sexual Assaults        469
Theft Over $5000       249
Homicide                20
Name: Violation_Type, dtype: int64
```

> For the purpose of statistical analysis, we have decided to keep the Violation_Types that belong to Property Crimes and Viloent Crimes. The reason for this is to see if there is a relationship between them, when we include the Property Assessment Values for 2016 to the dataframe later.

We decided to pivot the table- thinking that it would be easy to impliment the clustering and also easier to merge the dataframe to get the Property Assessment Values, later on.

In [ ]:
```
# Pivot the dataframe on NGH_Name and calculate the sum of occurrences for each Violat
pivot_table = df_copy.pivot_table(index='NGH_Name', values='Sum_Occurrences', columns=

# Include the demographic columns in the pivot table
pivot_table = pd.concat([pivot_table, df_copy.groupby('NGH_Name')[['Latitude', 'Longit
       'Post_Secondary', 'Homemaker', 'Employedage0-30', 'Employed 30+ ',
       'Unemployed', 'Retired', 'Permanently U', 'Employment_No Response',
       'Income_Less than $30,', 'Income_$30,000 to les',
       'Income_$60,000 to les', 'Income_$100,000 to le',
       'Income_$125,000 to le', 'Income_$150,000 to le',
       'Income_$200,000 to le', 'Income_$250,000 or mo', 'Income_No Response']].mean()

# Rename the columns
pivot_table.columns = ['Assault', 'Break and Enter', 'Homicide', 'Robbery', 'Sexual As

# Print the pivot table
pivot_table
```

Out[ ]:

NGH_Name	Assault	Break and Enter	Homicide	Robbery	Sexual Assaults	Theft From Vehicle	Theft Of Vehicle	Theft Over $5000	Latitude	Lo
ABBOTTSFIELD	35	8	0	8	4	16	6	0	53.574143	-11
ALBANY	8	8	0	2	2	9	2	1	53.632382	-11
ALBERTA AVENUE	123	119	1	26	13	156	99	2	53.568485	-11
ALDERGROVE	17	19	0	6	4	46	23	0	53.516888	-11
ALLARD	6	17	0	1	0	12	3	4	53.401301	-11
...	...	...	...	...	...	...	...	...	...	
WESTWOOD	43	34	0	7	6	53	48	1	53.575942	-11
WILD ROSE	14	15	0	2	2	46	24	1	53.470564	-11
WINDERMERE	7	22	0	3	1	31	6	4	53.432563	-11
WOODCROFT	30	22	0	13	4	40	20	4	53.564595	-11
YORK	24	21	0	4	2	48	17	4	53.602843	-11

245 rows × 31 columns

In [ ]:
```
pivot_table.info()
```

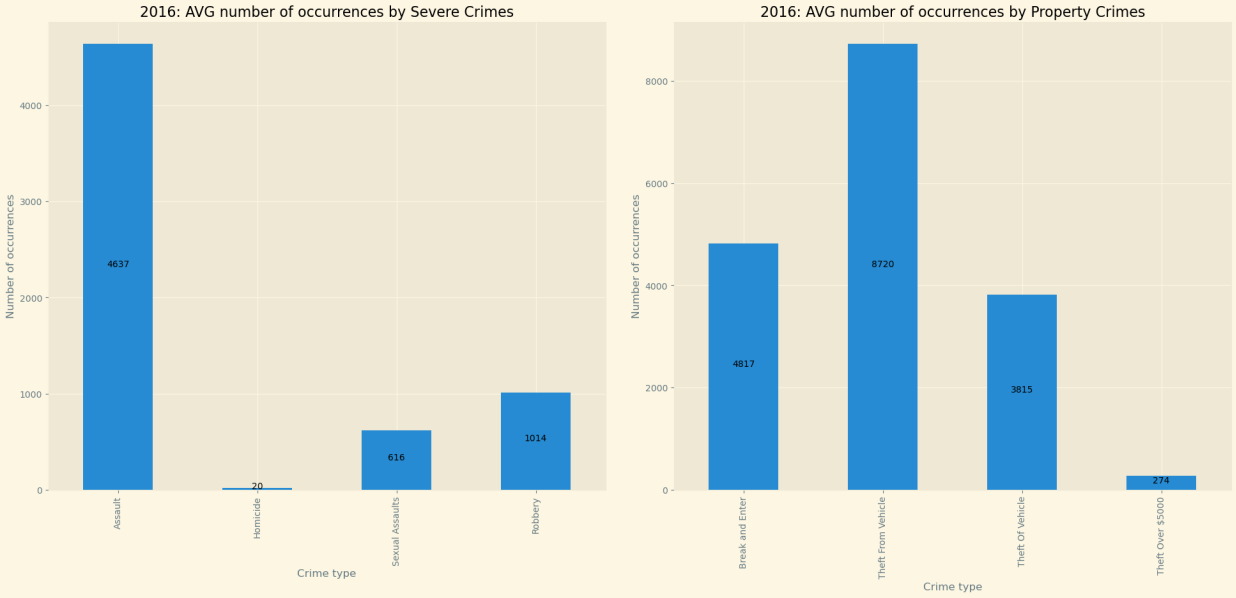
```
<class 'pandas.core.frame.DataFrame'>
Index: 245 entries, ABBOTTSFIELD to YORK
Data columns (total 31 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Assault                         245 non-null    int64
 1   Break and Enter                 245 non-null    int64
 2   Homicide                        245 non-null    int64
 3   Robbery                         245 non-null    int64
 4   Sexual Assaults                 245 non-null    int64
 5   Theft From Vehicle              245 non-null    int64
 6   Theft Of Vehicle                245 non-null    int64
 7   Theft Over $5000                245 non-null    int64
 8   Latitude                        245 non-null    float64
 9   Longitude                       245 non-null    float64
 10  Preschool                       245 non-null    float64
 11  Kindergarten                    245 non-null    float64
 12  Gr7-9                           245 non-null    float64
 13  Gr10-12                         245 non-null    float64
 14  Post-Secondary                  245 non-null    float64
 15  Homemaker                       245 non-null    float64
 16  Employed 0-30                   245 non-null    float64
 17  Employed 30+                    245 non-null    float64
 18  Unemployed                      245 non-null    float64
 19  Retired                         245 non-null    float64
 20  Permanently Unemployed          245 non-null    float64
 21  Employment_No Response          245 non-null    float64
 22  Income_Less than $30K           245 non-null    float64
 23  Income_30K to less than 60K     245 non-null    float64
 24  Income_60K to less than 100K    245 non-null    float64
 25  Income_100K to less than 125K   245 non-null    float64
 26  Income_125K to less than 150K   245 non-null    float64
 27  Income_150K to less than 200K   245 non-null    float64
 28  Income_200K to less than 250K   245 non-null    float64
 29  Income_250K or more             245 non-null    float64
 30  Income_No Response              245 non-null    float64
dtypes: float64(23), int64(8)
memory usage: 61.2+ KB
```

> The pivoted table is now reduced to 374 rows and 31 columns because we grouped the Neighborhoods and addredgated the sum of Crimes and also took the average of Employment Status, Income Levels and Education Levels.

> Lets look at the countplots of the grouped Crimes; Property Crimes and Violent Crimes seperatel, for all neighborhoods in 2016.

In [ ]:
```
# create a grid with two subplots
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(20,10))

# plot the first subplot
pivot_table[['Assault', 'Homicide', 'Sexual Assaults', 'Robbery']].sum().plot(kind='ba
ax[0].set_title('2016: AVG number of occurrences by Severe Crimes')
ax[0].set_xlabel('Crime type')
ax[0].set_ylabel('Number of occurrences')
```

```
# add annotations to the bars in the first subplot
for i, v in enumerate(pivot_table[['Assault', 'Homicide', 'Sexual Assaults', 'Robbery'
    ax[0].text(i, v/2, str(v), ha='center', fontsize=10)

# plot the second subplot
pivot_table[['Break and Enter', 'Theft From Vehicle', 'Theft Of Vehicle', 'Theft Over
ax[1].set_title('2016: AVG number of occurrences by Property Crimes')
ax[1].set_xlabel('Crime type')
ax[1].set_ylabel('Number of occurrences')

# add annotations to the bars in the second subplot
for i, v in enumerate(pivot_table[['Break and Enter', 'Theft From Vehicle', 'Theft Of
    ax[1].text(i, v/2, str(v), ha='center', fontsize=10)

# adjust the layout and spacing of the subplots
fig.tight_layout(pad=3)

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

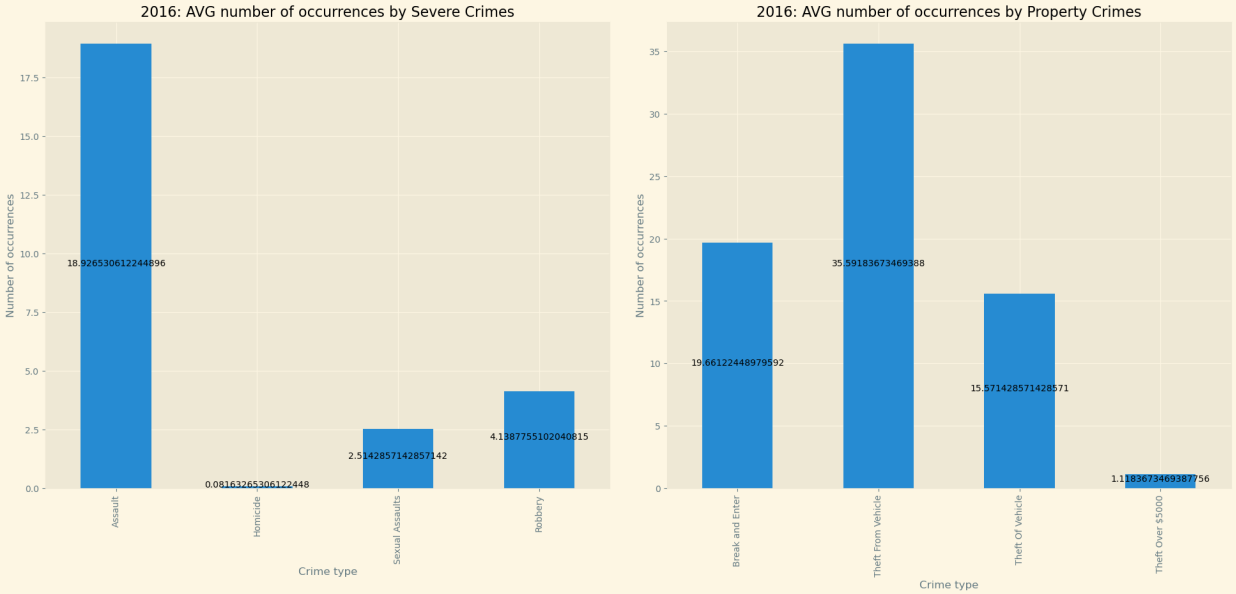


The following chart shows the AVERAGE number of Occurences.

In [ ]:

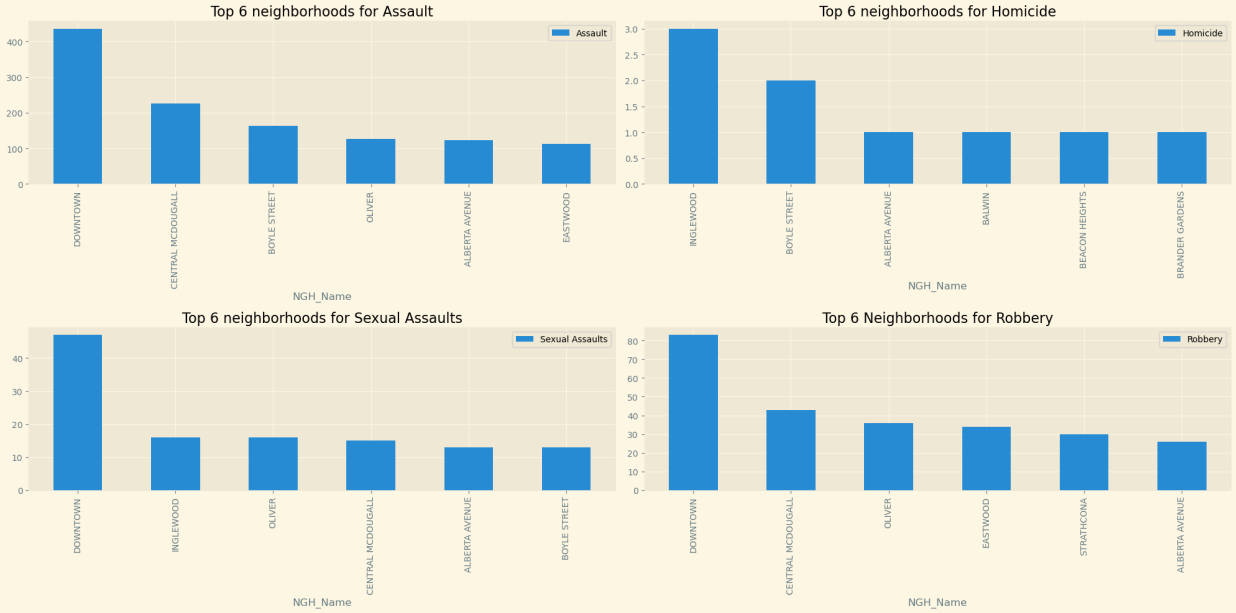
```
# create a grid with two subplots
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(20,10))

# plot the first subplot
pivot_table[['Assault', 'Homicide', 'Sexual Assaults', 'Robbery']].mean().plot(kind='b
ax[0].set_title('2016: AVG number of occurrences by Severe Crimes')
ax[0].set_xlabel('Crime type')
ax[0].set_ylabel('Number of occurrences')

# add annotations to the bars in the first subplot
for i, v in enumerate(pivot_table[['Assault', 'Homicide', 'Sexual Assaults', 'Robbery'
    ax[0].text(i, v/2, str(v), ha='center', fontsize=10)

# plot the second subplot
pivot_table[['Break and Enter', 'Theft From Vehicle', 'Theft Of Vehicle', 'Theft Over
ax[1].set_title('2016: AVG number of occurrences by Property Crimes')
ax[1].set_xlabel('Crime type')
ax[1].set_ylabel('Number of occurrences')
```

```
# add annotations to the bars in the second subplot
for i, v in enumerate(pivot_table[['Break and Enter', 'Theft From Vehicle', 'Theft Of
    ax[1].text(i, v/2, str(v), ha='center', fontsize=10)

# adjust the layout and spacing of the subplots
fig.tight_layout(pad=3)

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

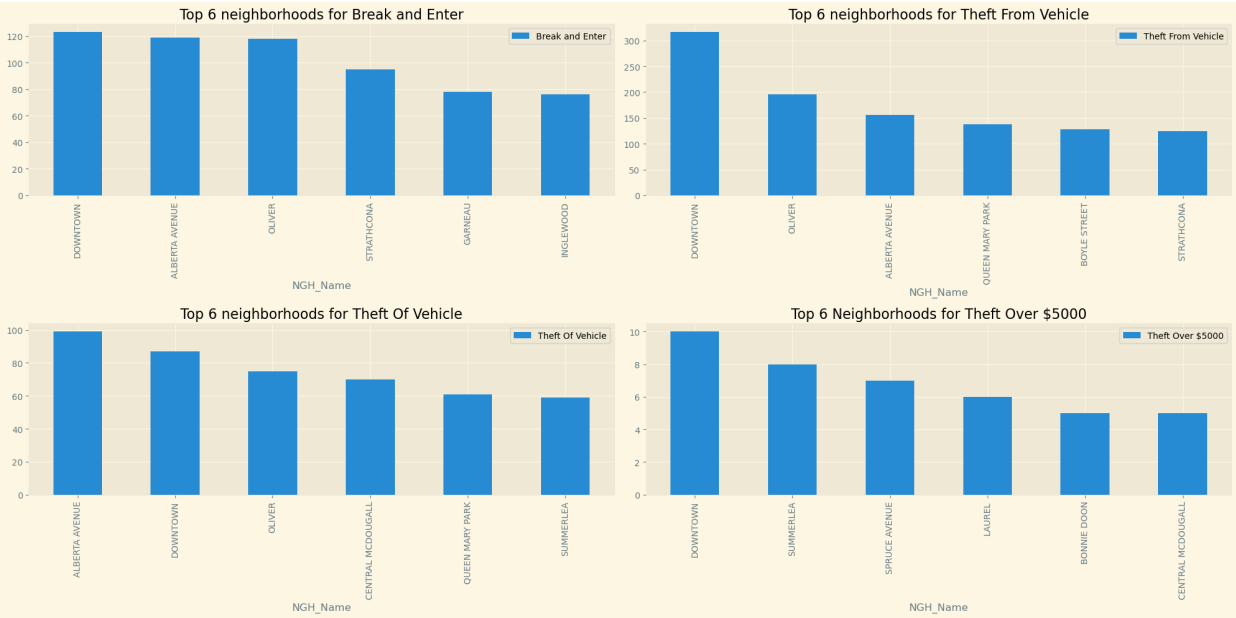


> As we noticed earlier when looking at 2009-2019, we somewhat have the same pattern for 2016. The Violent Crime: Assault have the highest sum and means. Interesting to note, Property Crime: Theft From Vehicle is much higher. Wonder of this pattern is reflected in our Top 6 popular Neighbourhoods.

In [ ]:
```
# Select the top 6 neighborhoods with the highest counts of each type of crime
assault_top6 = pivot_table.nlargest(6, 'Assault')
homicide_top6 = pivot_table.nlargest(6, 'Homicide')
sexual_assaults_top6 = pivot_table.nlargest(6, 'Sexual Assaults')
robbery_top6 = pivot_table.nlargest(6, 'Robbery')

# Create a bar plot of the total number of occurrences for each type of crime
fig, axs = plt.subplots(2, 2, figsize=(20, 10))
assault_top6[['Assault']].plot(kind='bar', ax=axs[0, 0], title='Top 6 neighborhoods fo
homicide_top6[['Homicide']].plot(kind='bar', ax=axs[0, 1], title='Top 6 neighborhoods
sexual_assaults_top6[['Sexual Assaults']].plot(kind='bar', ax=axs[1, 0], title='Top 6
robbery_top6[['Robbery']].plot(kind='bar', ax=axs[1, 1], title='Top 6 Neighborhoods fo

plt.tight_layout()
plt.show()
```



In [ ]:
```
# Select the top 6 neighborhoods with the highest counts of each type of crime
break_and_enter_top6 = pivot_table.nlargest(6, 'Break and Enter')
theft_from_vehicle_top6 = pivot_table.nlargest(6, 'Theft From Vehicle')
theft_of_vehicle_top6 = pivot_table.nlargest(6, 'Theft Of Vehicle')
theft_over_5000_top6 = pivot_table.nlargest(6, 'Theft Over $5000')

# Create a bar plot of the total number of occurrences for each type of crime
fig, axs = plt.subplots(2, 2, figsize=(20, 10))
break_and_enter_top6[['Break and Enter']].plot(kind='bar', ax=axs[0, 0], title='Top 6
theft_from_vehicle_top6[['Theft From Vehicle']].plot(kind='bar', ax=axs[0, 1], title='
theft_of_vehicle_top6[['Theft Of Vehicle']].plot(kind='bar', ax=axs[1, 0], title='Top
theft_over_5000_top6[['Theft Over $5000']].plot(kind='bar', ax=axs[1, 1], title='Top 6

plt.tight_layout()
plt.show()
```



> Oh!! We have some Industrial Area showing up in Property Crimes.

In [ ]:
```
pivot_table = pivot_table.reset_index()
pivot_table
```

Out[ ]:

	NGH_Name	Assault	Break and Enter	Homicide	Robbery	Sexual Assaults	Theft From Vehicle	Theft Of Vehicle	Theft Over $5000	Latitude
0	ABBOTTSFIELD	35	8	0	8	4	16	6	0	53.574143
1	ALBANY	8	8	0	2	2	9	2	1	53.632382
2	ALBERTA AVENUE	123	119	1	26	13	156	99	2	53.568485
3	ALDERGROVE	17	19	0	6	4	46	23	0	53.516888
4	ALLARD	6	17	0	1	0	12	3	4	53.401301
...	...	...	...	...	...	...	...	...	...	...
240	WESTWOOD	43	34	0	7	6	53	48	1	53.575942
241	WILD ROSE	14	15	0	2	2	46	24	1	53.470564
242	WINDERMERE	7	22	0	3	1	31	6	4	53.432563
243	WOODCROFT	30	22	0	13	4	40	20	4	53.564595
244	YORK	24	21	0	4	2	48	17	4	53.602843

245 rows × 32 columns

Save the Pivoted_table dataframe

In [ ]:
```
pivot_table.to_csv('March20_2023.csv')
df2=pd.read_csv('March20_2023.csv', encoding_errors='ignore')
df2.drop(['Unnamed: 0'], axis=1, inplace =True)
df2
```

Out[ ]:

	NGH_Name	Assault	Break and Enter	Homicide	Robbery	Sexual Assaults	Theft From Vehicle	Theft Of Vehicle	Theft Over $5000	Latitude
0	ABBOTTSFIELD	35	8	0	8	4	16	6	0	53.574143
1	ALBANY	8	8	0	2	2	9	2	1	53.632382
2	ALBERTA AVENUE	123	119	1	26	13	156	99	2	53.568485
3	ALDERGROVE	17	19	0	6	4	46	23	0	53.516888
4	ALLARD	6	17	0	1	0	12	3	4	53.401301
...	...	...	...	...	...	...	...	...	...	...
240	WESTWOOD	43	34	0	7	6	53	48	1	53.575942
241	WILD ROSE	14	15	0	2	2	46	24	1	53.470564
242	WINDERMERE	7	22	0	3	1	31	6	4	53.432563
243	WOODCROFT	30	22	0	13	4	40	20	4	53.564595
244	YORK	24	21	0	4	2	48	17	4	53.602843

245 rows × 32 columns

> Now lets look at the dataframe with Property Assessment Values

In [ ]:

```
prop=pd.read_csv('Edmonton Historical Prpty Assmnts.csv')
#We will only look at the year 2016.
prop2016=prop[prop['Assessment Year']==2016]
# Only use Residential
#prop2016=prop2016[prop2016['Assessment Class 1']=='RESIDENTIAL']
#Drop the columns we do not need.
prop2016.drop(['Suite', 'House Number', 'Account Number','Street Name', 'Legal Descrip
prop2016.drop(['Lot Size', 'Assessment Class % 1', 'Assessment Class 2', 'Assessment C
prop2016.rename(columns={'Neighbourhood':'NGH_Name', }, inplace=True)
display('Dataset Information:', prop2016.info())
print()
display('Null Values?:', prop2016.isnull().sum())
prop2016.head(5)
```

```
C:\Users\azimi\AppData\Local\Temp\ipykernel_14576\2330843994.py:1: DtypeWarning: Colu
mns (2,17,19) have mixed types. Specify dtype option on import or set low_memory=Fals
e.
  prop=pd.read_csv('Edmonton Historical Prpty Assmnts.csv')
C:\Users\azimi\AppData\Local\Temp\ipykernel_14576\2330843994.py:7: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  prop2016.drop(['Suite', 'House Number', 'Account Number','Street Name', 'Legal Desc
ription', 'Latitude', 'Longitude', 'Point Location', 'Assessment Year'], axis=1, inpl
ace =True)
C:\Users\azimi\AppData\Local\Temp\ipykernel_14576\2330843994.py:8: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  prop2016.drop(['Lot Size', 'Assessment Class % 1', 'Assessment Class 2', 'Assessmen
t Class % 2', 'Assessment Class 3', 'Assessment Class % 3', 'Actual Year Built', 'Zon
ing'], axis=1, inplace =True)
C:\Users\azimi\AppData\Local\Temp\ipykernel_14576\2330843994.py:9: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  prop2016.rename(columns={'Neighbourhood':'NGH_Name', }, inplace=True)
<class 'pandas.core.frame.DataFrame'>
Int64Index: 381025 entries, 35 to 4109010
Data columns (total 4 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   NGH_Name            381025 non-null  object
 1   Garage              381025 non-null  object
 2   Assessed Value      381025 non-null  float64
 3   Assessment Class 1  381025 non-null  object
dtypes: float64(1), object(3)
memory usage: 14.5+ MB
'Dataset Information:'
None

'Null Values?:'
NGH_Name              0
Garage                0
Assessed Value        0
Assessment Class 1    0
dtype: int64
```

Out[ ]:

	NGH_Name	Garage	Assessed Value	Assessment Class 1
35	OLIVER	N	339000.0	RESIDENTIAL
36	GROVENOR	N	98000.0	RESIDENTIAL
50	MACTAGGART	N	76000.0	RESIDENTIAL
74	OLIVER	N	234500.0	RESIDENTIAL
92	QUEEN ALEXANDRA	N	24500.0	RESIDENTIAL

In [ ]:
```
prop2016['NGH_Name'].value_counts()
```

Out[ ]:
```
OLIVER                    11802
DOWNTOWN                   8974
WINDERMERE                 5288
RUTHERFORD                 5129
SUMMERSIDE                 4932
                          ...
CRYSTALLINA NERA EAST         2
MILL WOODS GOLF COURSE        2
PIPELINES OIL FIELD           2
YELLOWHEAD CORRIDOR           1
TERWILLEGAR AREA              1
Name: NGH_Name, Length: 393, dtype: int64
```

In [ ]:
```
"""In order to be able to merge with our pivoted_tabkle dataframe, we need to group th
 and aggregate the Assessed Value column to show the mean values."""
prop2016_agg = prop2016.groupby('NGH_Name').agg({'Assessed Value': 'mean'})
prop2016_agg
```

Out[ ]:

NGH_Name	Assessed Value
ABBOTTSFIELD	3.463406e+05
ALBANY	3.492812e+05
ALBERTA AVENUE	2.806964e+05
ALBERTA PARK INDUSTRIAL	3.286137e+06
ALDERGROVE	3.689508e+05
...	...
YELLOWHEAD CORRIDOR	1.447736e+06
YELLOWHEAD CORRIDOR EAST	1.371441e+06
YELLOWHEAD CORRIDOR WEST	9.298375e+06
YORK	3.824538e+05
YOUNGSTOWN INDUSTRIAL	4.797202e+06

393 rows × 1 columns

In [ ]:
```
# Merge df2 and prop2016_agg on NGH_Name
merged_df = pd.merge(df2, prop2016_agg, on='NGH_Name', how='left')
```

```
# Display the first few rows of the merged dataframe
merged_df.isnull().sum()
```

Out[ ]:
```
NGH_Name                        0
Assault                         0
Break and Enter                 0
Homicide                        0
Robbery                         0
Sexual Assaults                 0
Theft From Vehicle              0
Theft Of Vehicle                0
Theft Over $5000                0
Latitude                        0
Longitude                       0
Preschool                       0
Kindergarten                    0
Gr7-9                           0
Gr10-12                         0
Post-Secondary                  0
Homemaker                       0
Employed 0-30                   0
Employed 30+                    0
Unemployed                      0
Retired                         0
Permanently Unemployed          0
Employment_No Response          0
Income_Less than $30K           0
Income_30K to less than 60K     0
Income_60K to less than 100K    0
Income_100K to less than 125K   0
Income_125K to less than 150K   0
Income_150K to less than 200K   0
Income_200K to less than 250K   0
Income_250K or more             0
Income_No Response              0
Assessed Value                  2
dtype: int64
```

In [ ]:
```
# We have 2 Neighborhoods with null values for Assessment- what are they?
null_values = merged_df[merged_df['Assessed Value'].isnull()]['NGH_Name']
null_values
```

Out[ ]:
```
184          RAPPERSWILL
237    WESTBROOK ESTATES
Name: NGH_Name, dtype: object
```

In [ ]:
```
# We can delete/drop them
merged_df.dropna(inplace=True)
merged_df
```

Out[ ]:

	NGH_Name	Assault	Break and Enter	Homicide	Robbery	Sexual Assaults	Theft From Vehicle	Theft Of Vehicle	Theft Over $5000	Latitude
0	ABBOTTSFIELD	35	8	0	8	4	16	6	0	53.574143
1	ALBANY	8	8	0	2	2	9	2	1	53.632382
2	ALBERTA AVENUE	123	119	1	26	13	156	99	2	53.568485
3	ALDERGROVE	17	19	0	6	4	46	23	0	53.516888
4	ALLARD	6	17	0	1	0	12	3	4	53.401301
...	...	...	...	...	...	...	...	...	...	...
240	WESTWOOD	43	34	0	7	6	53	48	1	53.575942
241	WILD ROSE	14	15	0	2	2	46	24	1	53.470564
242	WINDERMERE	7	22	0	3	1	31	6	4	53.432563
243	WOODCROFT	30	22	0	13	4	40	20	4	53.564595
244	YORK	24	21	0	4	2	48	17	4	53.602843

243 rows × 33 columns

In [ ]:
```
merged_df.to_csv('merged_df_March24_2023.csv')
final=pd.read_csv('merged_df_March24_2023.csv', encoding_errors='ignore')
final.drop(['Unnamed: 0'], axis=1, inplace =True)
final.head(5)
```

Out[ ]:

	NGH_Name	Assault	Break and Enter	Homicide	Robbery	Sexual Assaults	Theft From Vehicle	Theft Of Vehicle	Theft Over $5000	Latitude	..
0	ABBOTTSFIELD	35	8	0	8	4	16	6	0	53.574143	.
1	ALBANY	8	8	0	2	2	9	2	1	53.632382	.
2	ALBERTA AVENUE	123	119	1	26	13	156	99	2	53.568485	.
3	ALDERGROVE	17	19	0	6	4	46	23	0	53.516888	.
4	ALLARD	6	17	0	1	0	12	3	4	53.401301	.

5 rows × 33 columns

Lets look at the distribution of the column Assessed Value

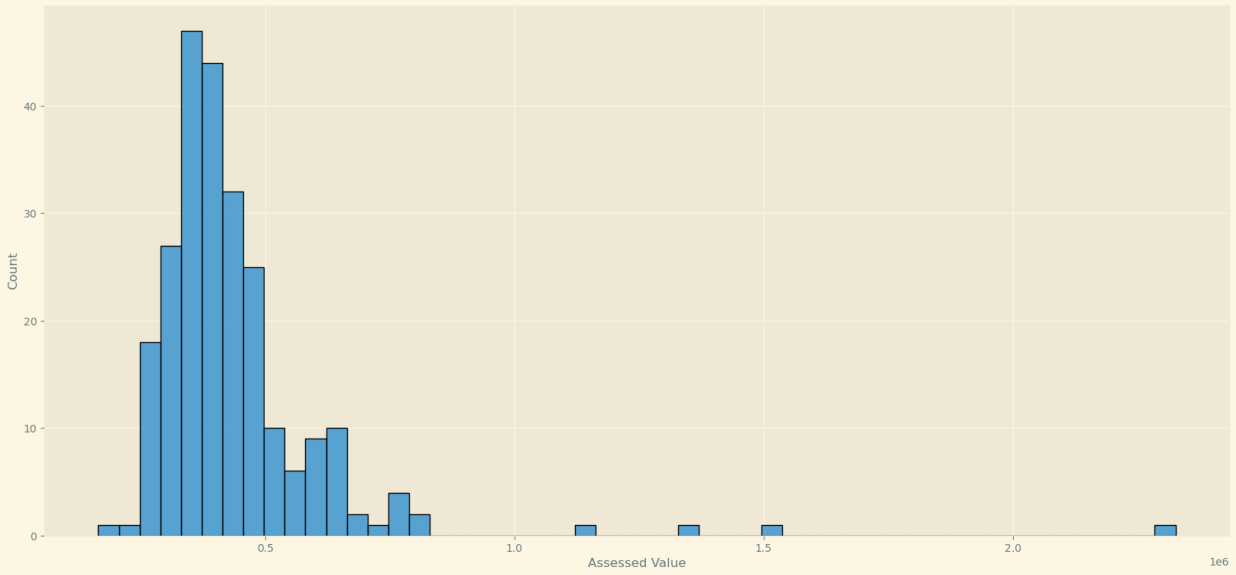
In [ ]:
```
final['Assessed Value'].describe()
```

Out[ ]:
```
count    2.430000e+02
mean     4.408666e+05
std      1.955941e+05
min      1.647495e+05
25%      3.463589e+05
50%      4.020394e+05
75%      4.769844e+05
max      2.326443e+06
Name: Assessed Value, dtype: float64
```

What about the distribution of Assessed Values for 2016

In [ ]:
```
sns.histplot(final['Assessed Value'])
```

Out[ ]:
```
<AxesSubplot:xlabel='Assessed Value', ylabel='Count'>
```



> Analysis/Investigation based on Correlation

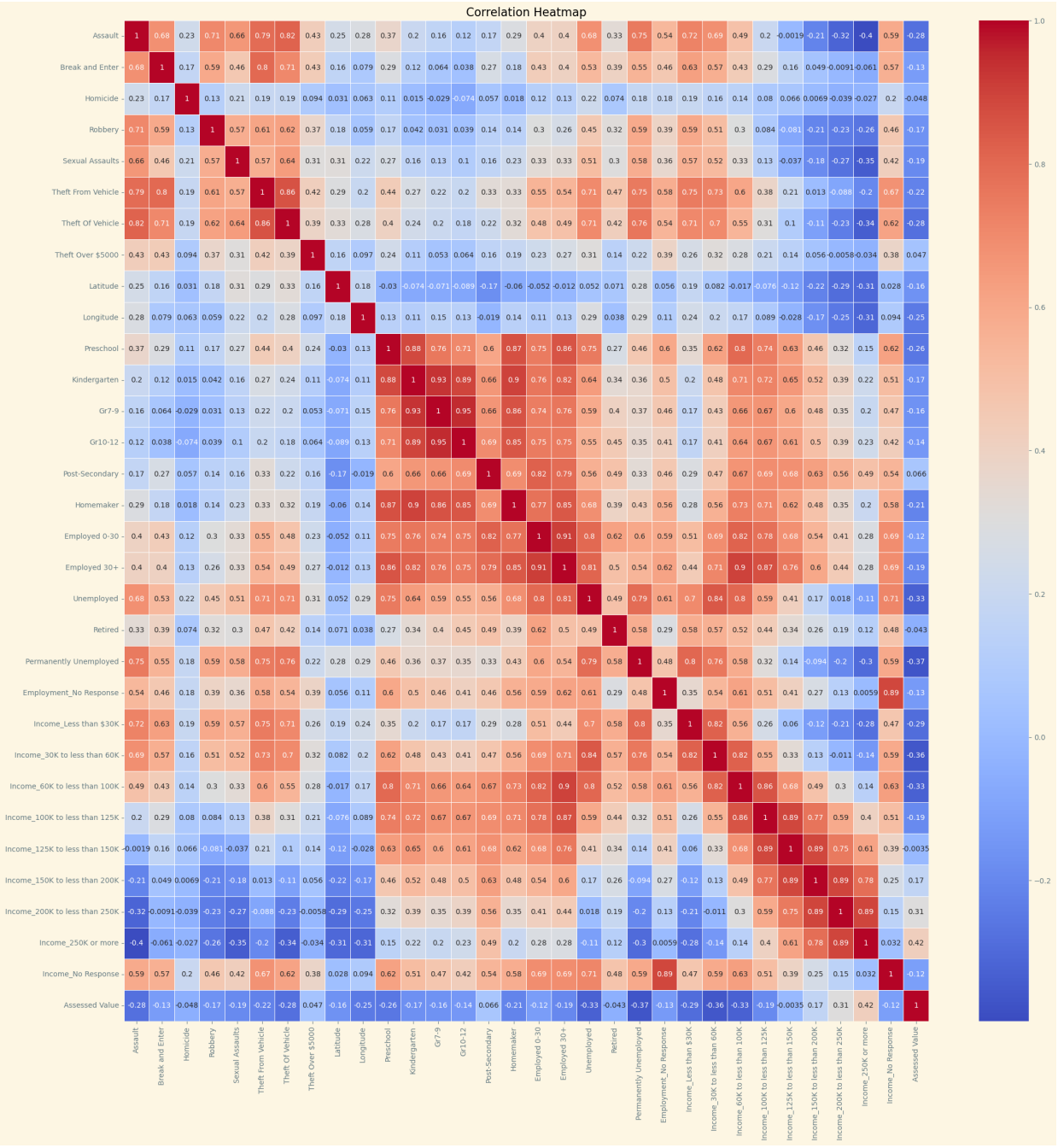
In [ ]:
```
final.columns
```

Out[ ]:
```
Index(['NGH_Name', 'Assault', 'Break and Enter', 'Homicide', 'Robbery',
       'Sexual Assaults', 'Theft From Vehicle', 'Theft Of Vehicle',
       'Theft Over $5000', 'Latitude', 'Longitude', 'Preschool',
       'Kindergarten', 'Gr7-9', 'Gr10-12', 'Post-Secondary', 'Homemaker',
       'Employed 0-30', 'Employed 30+', 'Unemployed', 'Retired',
       'Permanently Unemployed', 'Employment_No Response',
       'Income_Less than $30K', 'Income_30K to less than 60K',
       'Income_60K to less than 100K', 'Income_100K to less than 125K',
       'Income_125K to less than 150K', 'Income_150K to less than 200K',
       'Income_200K to less than 250K', 'Income_250K or more',
       'Income_No Response', 'Assessed Value'],
      dtype='object')
```

Correlation?

In [ ]:
```
corr_matrix = final.corr(method='spearman')

# Create a heatmap using seaborn
plt.figure(figsize=(25,25))
```

```
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=.5)
plt.title('Correlation Heatmap')

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



In [ ]:
```
# Exclude the 'Income_No Response' column from the correlation matrix
corr_matrix = final.drop('Income_No Response', axis=1).corr()

# Transform the matrix into a Series of pairwise correlations
corr_series = corr_matrix.stack()

# Filter the Series to include only pairs with a correlation greater than 0.5
high_corr_pairs = corr_series[(corr_series.abs() > 0.75) & ~(corr_series.index.get_lev

# Print the high correlation pairs
print(high_corr_pairs)
```

```
Assault                          Assault                          1.000000
                                 Robbery                          0.921406
                                 Sexual Assaults                  0.879468
                                 Theft From Vehicle               0.863739
                                 Theft Of Vehicle                 0.758669
                                                                    ...
Income_200K to less than 250K    Income_200K to less than 250K    1.000000
                                 Income_250K or more              0.836441
Income_250K or more              Income_200K to less than 250K    0.836441
                                 Income_250K or more              1.000000
Assessed Value                   Assessed Value                   1.000000
Length: 129, dtype: float64
```

In [ ]:
```
"""This function uses the pandas corr() method to calculate the correlations between t
    It then drops the target column from the list of correlations and gets the top 3 c
    Finally, it prints the names of the top 3 correlated columns and their correlatior

def get_top_correlations(data, target_column):
    # Calculate the correlations between the target column and all other columns
    correlations = data.corr(method='spearman')[target_column].sort_values(ascending=F

    # Drop the target column from the list of correlations
    correlations = correlations.drop(target_column)

    # Get the top 3 correlations (positive or negative)
    top_correlations = correlations.abs().nlargest(4)

    # Print the top 3 correlations and their sign
    for column, correlation in top_correlations.items():
        sign = 'positive' if correlations[column] > 0 else 'negative'
        print(f"{column}: {sign} correlation ({correlation:.2f})")
```

In [ ]:
```
get_top_correlations(final, 'Assessed Value')
```

```
Income_250K or more: positive correlation (0.42)
Permanently Unemployed: negative correlation (0.37)
Income_30K to less than 60K: negative correlation (0.36)
Unemployed: negative correlation (0.33)
```

These results suggest that there are some interesting relationships between assessed property values and other variables in the final DataFrame. Here are a few possible interpretations:

Higher-income neighborhoods tend to have higher assessed property values. This is indicated by the positive correlation between Assessed Value and Income_250K or more. It's possible that this correlation reflects the fact that more affluent neighborhoods tend to have more expensive homes and properties, which are valued higher for property tax purposes.

Unemployment and low income are negatively associated with assessed property values. This is indicated by the negative correlations between Assessed Value and Permanently Unemployed, Income_30K to less than 60K, and Unemployed. It's possible that these correlations reflect the fact that neighborhoods with higher rates of unemployment and lower median incomes tend to have less valuable homes and properties, which are valued lower for property tax purposes.

Overall, these results suggest that there are important relationships between assessed property values and other socioeconomic variables in the final DataFrame. It's important to note,

however, that correlation does not necessarily imply causation, and further analysis would be needed to fully understand the relationships between these variables.

In [ ]:
```
get_top_correlations(final, 'Assault')
```

```
Theft Of Vehicle: positive correlation (0.82)
Theft From Vehicle: positive correlation (0.79)
Permanently Unemployed: positive correlation (0.75)
Income_Less than $30K: positive correlation (0.72)
```

These results suggest that there are some interesting relationships between assault rates and other variables in the final DataFrame. Here are a few possible interpretations:

Areas with higher rates of theft from and theft of vehicles tend to also have higher rates of assault. This is indicated by the positive correlations between Assault and Theft Of Vehicle (0.82) and Theft From Vehicle (0.79). It's possible that this correlation reflects the fact that areas with higher rates of property crime may also have higher rates of violent crime.

Areas with higher rates of permanent unemployment and lower median incomes tend to have higher rates of assault. This is indicated by the positive correlations between Assault and Permanently Unemployed (0.75) and Income_Less than $30K (0.72). It's possible that this correlation reflects the fact that areas with higher rates of poverty and unemployment may also have higher rates of violent crime.

Overall, these results suggest that there are important relationships between assault rates and other socioeconomic variables in the final DataFrame. However, as with the previous analysis, correlation does not necessarily imply causation, and further analysis would be needed to fully understand the relationships between these variables.

In [ ]:
```
get_top_correlations(final, 'Theft From Vehicle')
```

```
Theft Of Vehicle: positive correlation (0.86)
Break and Enter: positive correlation (0.80)
Assault: positive correlation (0.79)
Permanently Unemployed: positive correlation (0.75)
```

These results suggest that there are some interesting relationships between theft from vehicles and other variables in the final DataFrame. Here are a few possible interpretations:

Areas with higher rates of theft from vehicles tend to also have higher rates of theft of vehicles and break and enter. This is indicated by the positive correlations between Theft From Vehicle and Theft Of Vehicle (0.86) and Break and Enter (0.80). It's possible that this correlation reflects the fact that areas with higher rates of property crime may have more than one type of property crime.

Areas with higher rates of theft from vehicles also tend to have higher rates of assault. This is indicated by the positive correlation between Theft From Vehicle and Assault (0.79). It's possible that this correlation reflects the fact that areas with higher rates of property crime may also have higher rates of violent crime.

Areas with higher rates of permanent unemployment tend to have higher rates of theft from vehicles. This is indicated by the positive correlation between Theft From Vehicle and Permanently Unemployed (0.75). It's possible that this correlation reflects the fact that areas with higher rates of poverty and unemployment may have higher rates of property crime.

Overall, these results suggest that there are important relationships between theft from vehicle rates and other variables in the final DataFrame. However, as with the previous analyses, correlation does not necessarily imply causation, and further analysis would be needed to fully understand the relationships between these variables.

In [ ]:
```
get_top_correlations(final, 'Robbery')
```

```
Assault: positive correlation (0.71)
Theft Of Vehicle: positive correlation (0.62)
Theft From Vehicle: positive correlation (0.61)
Income_Less than $30K: positive correlation (0.59)
```

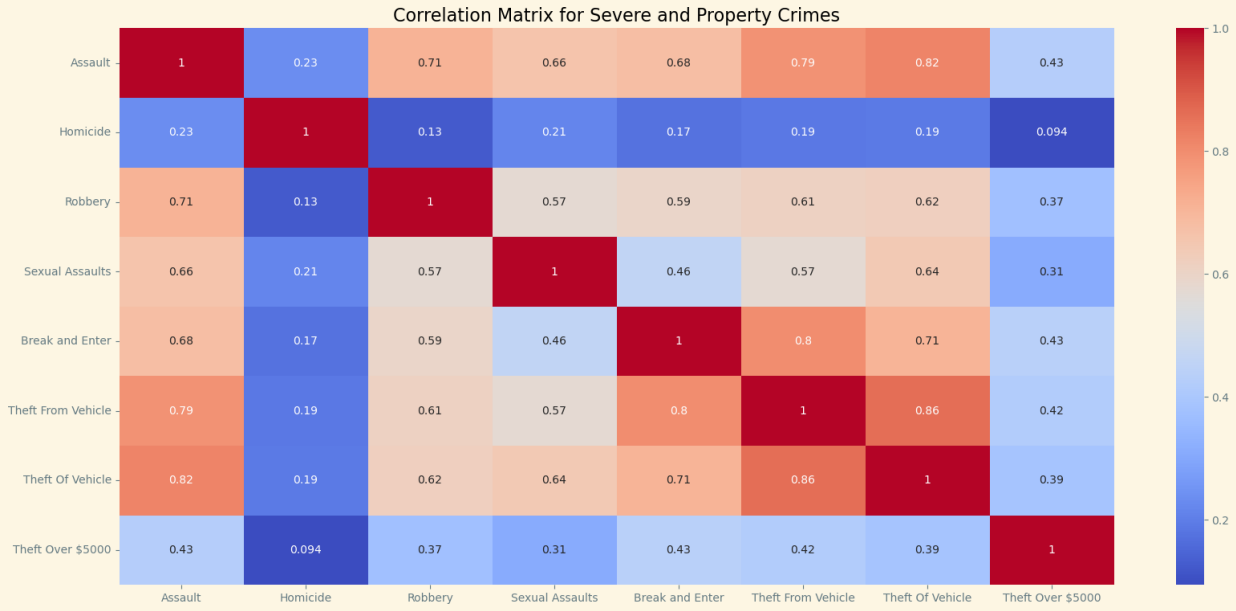
For 'Robbery', you found that higher counts were positively correlated with counts of assault, theft of vehicle, theft from vehicle, and having an income less than 30K.

> Investigate Further

In [ ]:
```
property_crime = ['Break and Enter', 'Theft From Vehicle', 'Theft Of Vehicle', 'Theft
severe_crime = ['Assault', 'Homicide', 'Robbery', 'Sexual Assaults']
income_levels = ['Income_Less than $30K', 'Income_30K to less than 60K', 'Income_60K t
                     'Income_100K to less than 125K', 'Income_125K to less than 150K',
                     'Income_200K to less than 250K', 'Income_250K or more']
employment_status = ['Homemaker', 'Employed 0-30', 'Employed 30+', 'Unemployed', 'Reti
                         'Permanently Unemployed']
education_level = ['Preschool', 'Kindergarten', 'Gr7-9', 'Gr10-12', 'Post-Secondary']
```

In [ ]:
```
def plot_crime_correlations(data, severe_crime_cols, property_crime_cols):
    # Create a new dataframe with just the specified crime columns
    crime_data = data[severe_crime_cols + property_crime_cols]

    # Calculate the correlation matrix
    corr = crime_data.corr(method='spearman')

    # Plot the correlation matrix heatmap
    sns.heatmap(corr, cmap='coolwarm', annot=True)
    plt.title('Correlation Matrix for Severe and Property Crimes')
    plt.show()
```

In [ ]:
```
plot_crime_correlations(final, severe_crime, property_crime)
```



In [ ]:
```
get_top_correlations(final[income_levels + severe_crime], 'Assault')
```

```
Income_Less than $30K: positive correlation (0.72)
Robbery: positive correlation (0.71)
Income_30K to less than 60K: positive correlation (0.69)
Sexual Assaults: positive correlation (0.66)
```

These correlations suggest that there is a positive relationship between lower income levels and severe crimes such as assault and robbery. The correlation coefficient for Income_Less than $30K is particularly high, indicating that this income level is strongly associated with these type
30K. The correlation between Sexual Assaults and income levels is also positive, but weaker than the correlation with assault and robbery.

In [ ]:
```
get_top_correlations(final[employment_status + severe_crime], 'Assault')
```

```
Permanently Unemployed: positive correlation (0.75)
Robbery: positive correlation (0.71)
Unemployed: positive correlation (0.68)
Sexual Assaults: positive correlation (0.66)
```

The results of get_top_correlations() function suggest that there is a positive correlation between low income levels and the likelihood of severe crimes such as Assault and Robbery. Specifically, there is a strong positive correlation between Income_Less than $30K and Assault, as well as Robbery. This suggests that individuals with lower incomes are more likely to be victims of violent crimes.

Furthermore, the results suggest that there is a positive correlation between being permanently unemployed and the likelihood of severe crimes such as Assault and Robbery. This could be attributed to the fact that individuals who are unemployed or permanently unemployed are more likely to face economic hardship, and may be more likely to engage in criminal activities.

It is important to note that correlation does not imply causation, and further research is needed to understand the underlying factors that contribute to these correlations. However, these

findings can be useful in identifying areas that require greater attention from law enforcement agencies and policy makers.

In [ ]:
```
get_top_correlations(final[education_level + severe_crime], 'Robbery')
```

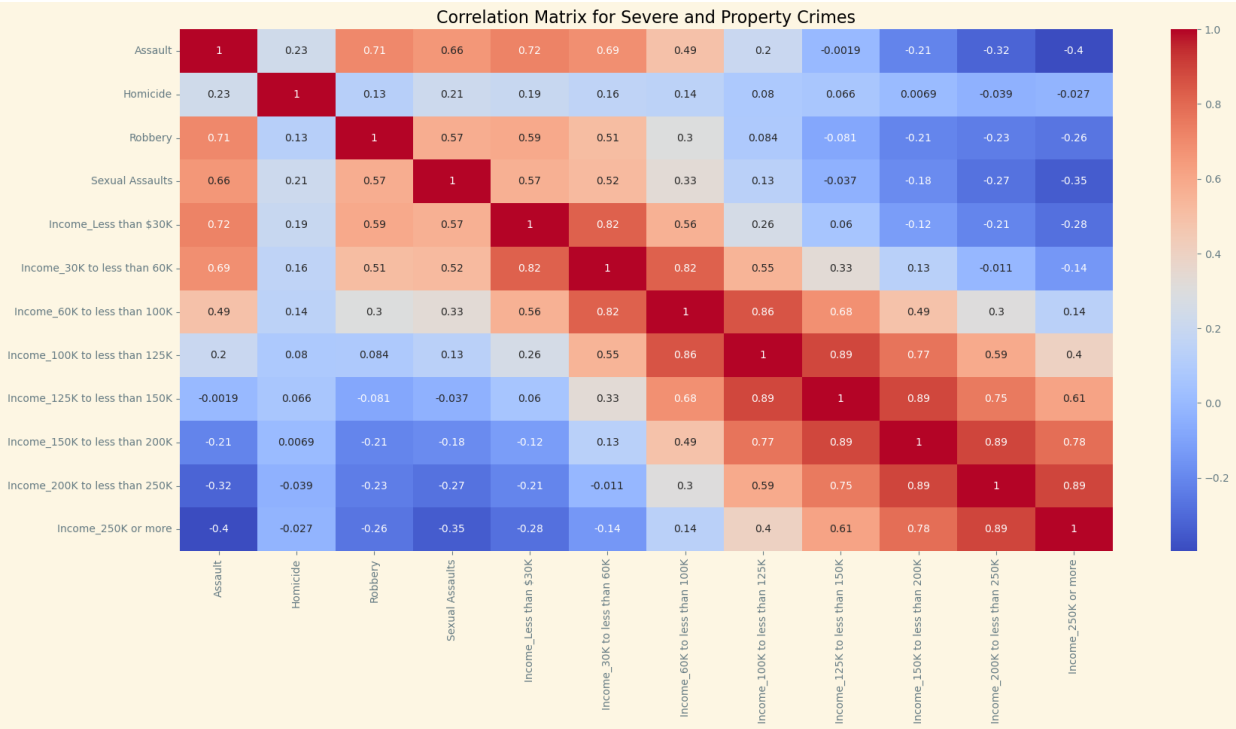
```
Assault: positive correlation (0.71)
Sexual Assaults: positive correlation (0.57)
Preschool: positive correlation (0.17)
Post-Secondary: positive correlation (0.14)
```

Based on the correlation coefficients, we can see that there is a positive correlation between Robbery and Assault. This suggests that areas with higher rates of robbery tend to have higher rates of assault as well. Additionally, Sexual Assaults also show a positive correlation with Robbery, indicating that areas with higher rates of robbery also tend to have higher rates of sexual assaults.

Interestingly, there is also a positive correlation between Preschool and Robbery. However, this correlation is relatively weak compared to the other correlations we've looked at. It's important to keep in mind that correlation does not imply causation, and we would need to investigate further to determine the nature of this relationship.

Finally, we can see that Post-Secondary education levels have a relatively weak positive correlation with robbery. It's possible that areas with higher levels of education have more valuable goods to steal, which could be driving this correlation. Again, further investigation would be needed to determine the underlying cause of this correlation.

In [ ]:
```
plot_crime_correlations(final, severe_crime, income_levels)
```



Clustering

In [ ]:
```
final.iloc[:,1:]
```

Out[ ]:

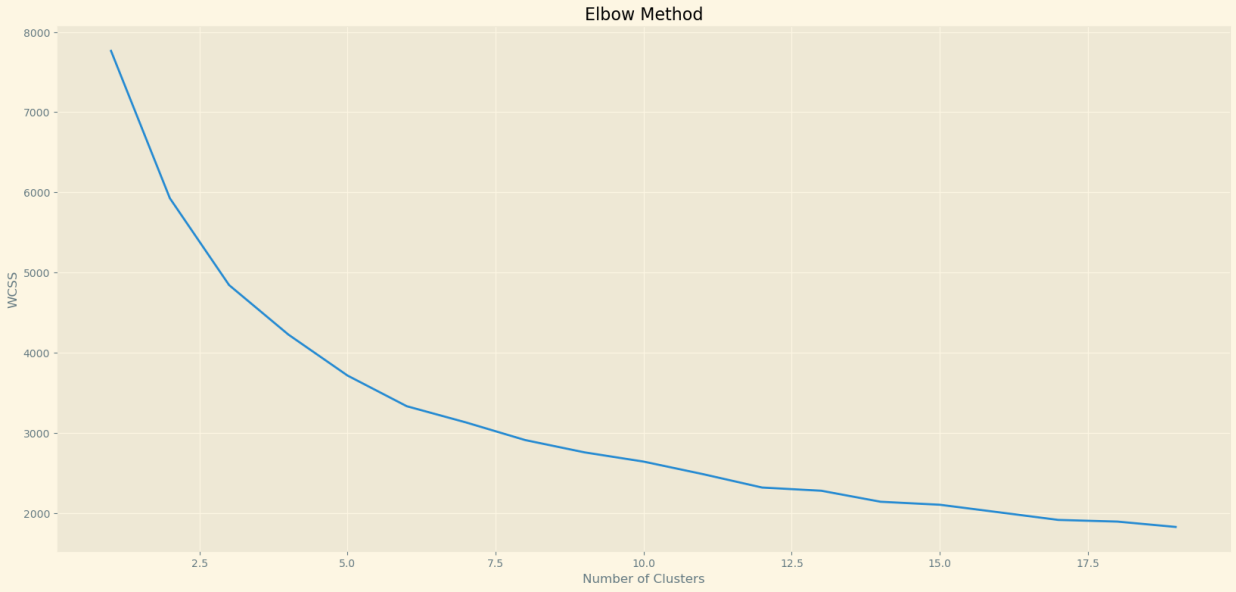
	Assault	Break and Enter	Homicide	Robbery	Sexual Assaults	Theft From Vehicle	Theft Of Vehicle	Theft Over $5000	Latitude	Longitude	.
0	35	8	0	8	4	16	6	0	53.574143	-113.388758	.
1	8	8	0	2	2	9	2	1	53.632382	-113.549464	.
2	123	119	1	26	13	156	99	2	53.568485	-113.485119	.
3	17	19	0	6	4	46	23	0	53.516888	-113.641242	.
4	6	17	0	1	0	12	3	4	53.401301	-113.526641	.
...	...	...	...	...	...	...	...	...	...	...	.
238	43	34	0	7	6	53	48	1	53.575942	-113.498585	.
239	14	15	0	2	2	46	24	1	53.470564	-113.381167	.
240	7	22	0	3	1	31	6	4	53.432563	-113.626008	.
241	30	22	0	13	4	40	20	4	53.564595	-113.558327	.
242	24	21	0	4	2	48	17	4	53.602843	-113.430212	.

243 rows × 32 columns

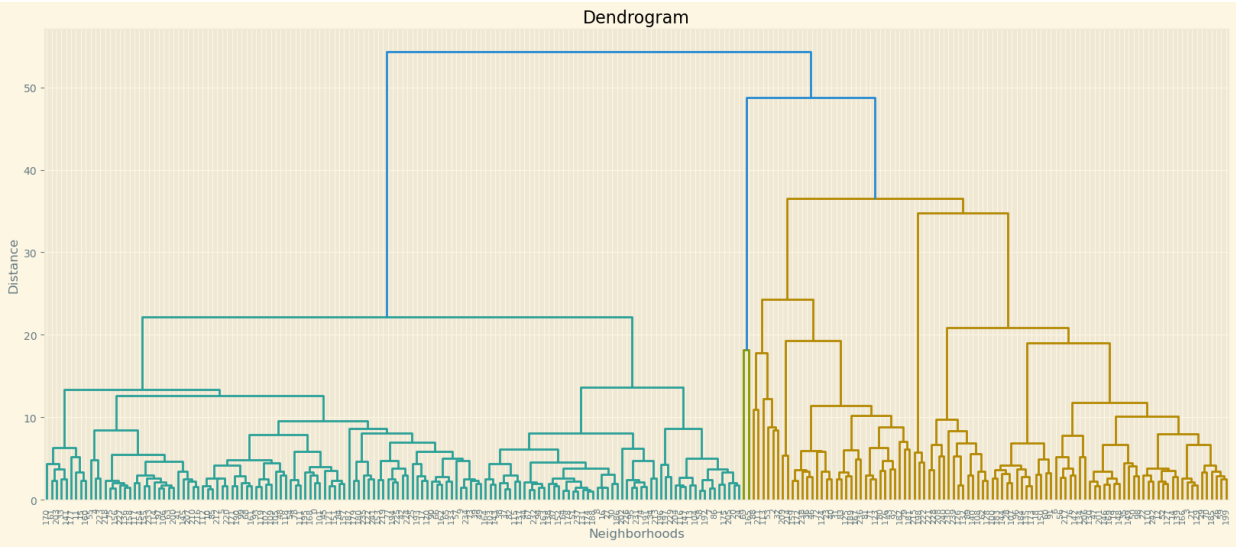
In [ ]:
```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# Standardize the data
scaler = StandardScaler()
df_scaled = scaler.fit_transform(final.iloc[:,1:])
```

In [ ]:
```
# Find the optimal number of clusters using the elbow method
wcss = []
for i in range(1, 20):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_st
    kmeans.fit(df_scaled)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 20), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```

```
c:\Users\azimi\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1036: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=1.
  warnings.warn(
```

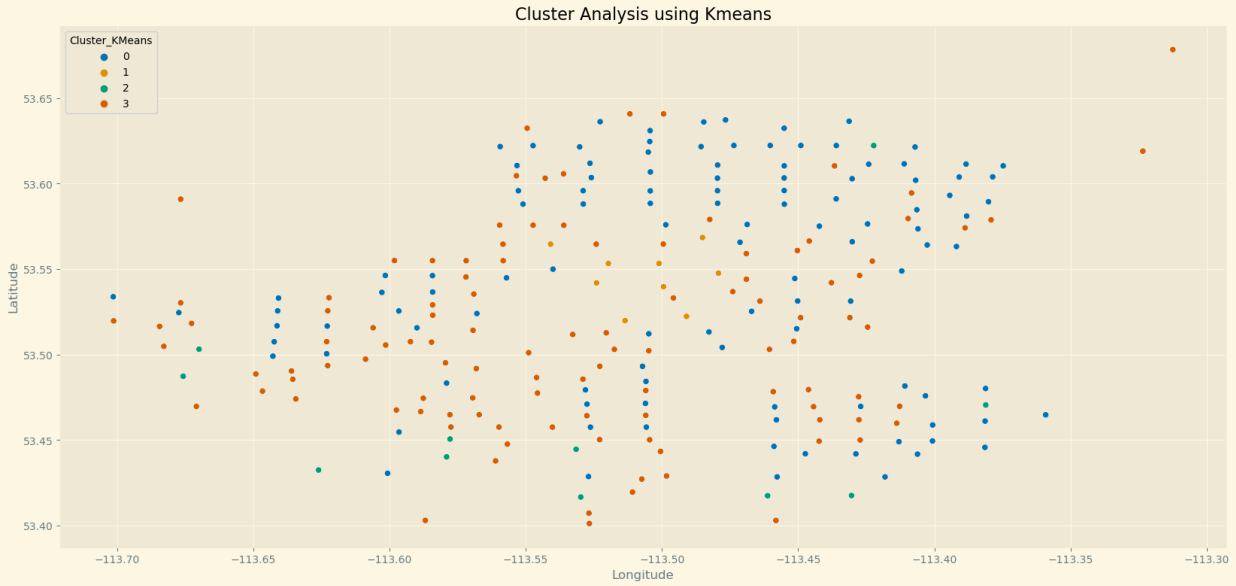


In [ ]:
```
import scipy.cluster.hierarchy as shc

# Generate the linkage matrix using complete linkage
linkage_matrix = shc.linkage(df_scaled, method='ward')

# Plot the dendrogram
plt.figure(figsize=(20, 8))
plt.title('Dendrogram')
plt.xlabel('Neighborhoods')
plt.ylabel('Distance')
shc.dendrogram(linkage_matrix, leaf_rotation=90., leaf_font_size=8.)
plt.show()
```



In [ ]:
```
# Create the KMeans model and fit to the standardized data
kmeans = KMeans(n_clusters=4, init='k-means++', max_iter=1100, n_init=10, random_state
pred_y = kmeans.fit_predict(df_scaled)

# Add the cluster labels to the original dataframe
final['Cluster_KMeans'] = pred_y

# Plot the clusters
sns.scatterplot(x='Longitude', y='Latitude', data=final, hue='Cluster_KMeans', palette
#sns.scatterplot(x='Latitude', y='Longitude', data=df, hue='Cluster', palette='colorbl
```

```
plt.title('Cluster Analysis using Kmeans')
plt.show()
```



Invetigating the Clusters

In [ ]:
```
final['Cluster_KMeans'].value_counts()
```

Out[ ]:
```
0    113
3    110
2     11
1      9
Name: Cluster_KMeans, dtype: int64
```

In [ ]:
```
cluster_0=final[final['Cluster_KMeans']==0]
cluster_1=final[final['Cluster_KMeans']==1]
cluster_2=final[final['Cluster_KMeans']==2]
cluster_3=final[final['Cluster_KMeans']==3]
```

In [ ]:
```
cluster_0['NGH_Name']
```

Out[ ]:
```
3             ALDERGROVE
6              AMBLESIDE
9                ATHLONE
11                BALWIN
12             BANNERMAN
              ...
234            WELLINGTON
235    WEST JASPER PLACE
236             WESTMOUNT
238              WESTWOOD
242                  YORK
Name: NGH_Name, Length: 113, dtype: object
```

In [ ]:
```
cluster_1['NGH_Name']
```

Out[ ]:
```
2          ALBERTA AVENUE
32           BOYLE STREET
53      CENTRAL MCDOUGALL
69              DOWNTOWN
88               GARNEAU
113            INGLEWOOD
165              OLIVER
182       QUEEN MARY PARK
211            STRATHCONA
Name: NGH_Name, dtype: object
```

In [ ]:
```
cluster_2['NGH_Name']
```

Out[ ]:
```
89            GLASTONBURY
108        HOLLICK-KENYON
198            RUTHERFORD
208     SOUTH TERWILLEGAR
215            SUMMERSIDE
221     TERWILLEGAR TOWNE
222          THE HAMPTONS
228           TWIN BROOKS
230                WALKER
239             WILD ROSE
240           WINDERMERE
Name: NGH_Name, dtype: object
```

In [ ]:
```
c3=[]
for ngh in cluster_3['NGH_Name']:
    c3.append(ngh)
print(c3)
```

```
['ABBOTTSFIELD', 'ALBANY', 'ALLARD', 'ALLENDALE', 'ARGYLL', 'ASPEN GARDENS', 'AVONMOR
E', 'BARANOW', 'BEARSPAW', 'BELGRAVIA', 'BELLEVUE', 'BERGMAN', 'BLACKBURNE', 'BLACKMU
D CREEK', 'BLUE QUILL ESTATES', 'BRANDER GARDENS', 'BRECKENRIDGE GREENS', 'BROOKSID
E', 'BULYEA HEIGHTS', 'CALLAGHAN', 'CALLINGWOOD NORTH', 'CAMERON HEIGHTS', 'CANON RID
GE', 'CAPILANO', 'CARTER CREST', 'CHAMBERY', 'CHAPPELLE AREA', 'CLOVERDALE', 'CRESTWO
OD', 'CROMDALE', 'DECHENE', 'DELTON', 'DONSDALE', 'DOVERCOURT', 'EDGEMONT', 'EKOTA',
'ELMWOOD', 'ELSINORE', 'FALCONER HEIGHTS', 'FULTON PLACE', 'GAINER INDUSTRIAL', 'GARI
EPY', 'GRANDVIEW HEIGHTS', 'GRANVILLE', 'GREENVIEW', 'GROVENOR', 'HENDERSON ESTATES',
'HIGH PARK', 'HODGSON', 'HUDSON', 'IDYLWYLDE', 'JAMIESON PLACE', 'JASPER PARK', 'KAME
YOSEK', 'KEHEEWIN', 'KENILWORTH', 'LANSDOWNE', 'LAURIER HEIGHTS', 'LEE RIDGE', 'LEGE
R', 'LENDRUM PLACE', 'MACTAGGART', 'MAGRATH HEIGHTS', 'MALMO PLAINS', 'MAYFIELD', 'MC
KERNAN', 'MCLEOD', 'MCQUEEN', 'MEYOKUMIN', 'MICHAELS PARK', 'MINCHAU', 'NORTH GLENOR
A', 'OGILVIE RIDGE', 'OLESKIW', 'PARKALLEN', 'PATRICIA HEIGHTS', 'PEMBINA', 'POTTER G
REENS', 'PRINCE CHARLES', 'PRINCE RUPERT', 'RHATIGAN RIDGE', 'RICHFORD', 'RIDEAU PAR
K', 'RIO TERRACE', 'RIVERDALE', 'ROSENTHAL', 'ROSSDALE', 'RURAL NORTH EAST HORSE HIL
L', 'RURAL NORTH EAST SOUTH STURGEON', 'SHERBROOKE', 'SHERWOOD', 'SIFTON PARK', 'SKYR
ATTLER', 'SPRUCE AVENUE', 'STEINHAUER', 'STRATHEARN', 'SUMMERLEA', 'SWEET GRASS', 'TA
WA', 'TERRA LOSA', 'TERRACE HEIGHTS', 'THE ORCHARDS AT ELLERSLIE', 'TRUMPETER AREA',
'TWEDDLE PLACE', 'VIRGINIA PARK', 'WEBBER GREENS', 'WEDGEWOOD HEIGHTS', 'WEINLOS', 'W
ESTRIDGE', 'WOODCROFT']
```

In [ ]:
```
['ABBOTTSFIELD', 'ALBANY', 'ALLARD', 'ALLENDALE', 'ARGYLL', 'ASPEN GARDENS',
 'AVONMORE', 'BARANOW', 'BEARSPAW', 'BELGRAVIA', 'BELLEVUE', 'BERGMAN', 'BLACKBURNE',
 'BLACKMUD CREEK', 'BLUE QUILL ESTATES', 'BRANDER GARDENS', 'BRECKENRIDGE GREENS', 'BR
 'BULYEA HEIGHTS', 'CALLAGHAN', 'CALLINGWOOD NORTH', 'CAMERON HEIGHTS', 'CANON RIDGE',
 'CARTER CREST', 'CHAMBERY', 'CHAPPELLE AREA', 'CLOVERDALE', 'CRESTWOOD', 'CROMDALE',
 'DONSDALE', 'DOVERCOURT', 'EDGEMONT', 'EKOTA', 'ELMWOOD', 'ELSINORE', 'FALCONER HEIGH
 'FULTON PLACE', 'GAINER INDUSTRIAL', 'GARIEPY', 'GRANDVIEW HEIGHTS', 'GRANVILLE', 'GR
 'GROVENOR', 'HENDERSON ESTATES', 'HIGH PARK', 'HODGSON', 'HUDSON', 'IDYLWYLDE', 'JAMI
```

```
 'JASPER PARK', 'KAMEYOSEK', 'KEHEEWIN', 'KENILWORTH', 'LANSDOWNE', 'LAURIER HEIGHTS',
 'LEGER', 'LENDRUM PLACE', 'MACTAGGART', 'MAGRATH HEIGHTS', 'MALMO PLAINS', 'MAYFIELD'
 'MCQUEEN', 'MEYOKUMIN', 'MICHAELS PARK', 'MINCHAU', 'NORTH GLENORA', 'OGILVIE RIDGE',
 'PATRICIA HEIGHTS', 'PEMBINA', 'POTTER GREENS', 'PRINCE CHARLES', 'PRINCE RUPERT', 'P
 'RIDEAU PARK', 'RIO TERRACE', 'RIVERDALE', 'ROSENTHAL', 'ROSSDALE', 'RURAL NORTH EAST
 'RURAL NORTH EAST SOUTH STURGEON', 'SHERBROOKE', 'SHERWOOD', 'SIFTON PARK', 'SKYRATTL
 'SPRUCE AVENUE', 'STEINHAUER', 'STRATHEARN', 'SUMMERLEA', 'SWEET GRASS', 'TAWA', 'TEP
 'TERRACE HEIGHTS', 'THE ORCHARDS AT ELLERSLIE', 'TRUMPETER AREA', 'TWEDDLE PLACE', 'V
 'WEBBER GREENS', 'WEDGEWOOD HEIGHTS', 'WEINLOS', 'WESTRIDGE', 'WOODCROFT']
```

Out[ ]:
```
['ABBOTTSFIELD',
 'ALBANY',
 'ALLARD',
 'ALLENDALE',
 'ARGYLL',
 'ASPEN GARDENS',
 'AVONMORE',
 'BARANOW',
 'BEARSPAW',
 'BELGRAVIA',
 'BELLEVUE',
 'BERGMAN',
 'BLACKBURNE',
 'BLACKMUD CREEK',
 'BLUE QUILL ESTATES',
 'BRANDER GARDENS',
 'BRECKENRIDGE GREENS',
 'BROOKSIDE',
 'BULYEA HEIGHTS',
 'CALLAGHAN',
 'CALLINGWOOD NORTH',
 'CAMERON HEIGHTS',
 'CANON RIDGE',
 'CAPILANO',
 'CARTER CREST',
 'CHAMBERY',
 'CHAPPELLE AREA',
 'CLOVERDALE',
 'CRESTWOOD',
 'CROMDALE',
 'DECHENE',
 'DELTON',
 'DONSDALE',
 'DOVERCOURT',
 'EDGEMONT',
 'EKOTA',
 'ELMWOOD',
 'ELSINORE',
 'FALCONER HEIGHTS',
 'FULTON PLACE',
 'GAINER INDUSTRIAL',
 'GARIEPY',
 'GRANDVIEW HEIGHTS',
 'GRANVILLE',
 'GREENVIEW',
 'GROVENOR',
 'HENDERSON ESTATES',
 'HIGH PARK',
 'HODGSON',
 'HUDSON',
 'IDYLWYLDE',
 'JAMIESON PLACE',
 'JASPER PARK',
 'KAMEYOSEK',
 'KEHEEWIN',
 'KENILWORTH',
 'LANSDOWNE',
 'LAURIER HEIGHTS',
 'LEE RIDGE',
 'LEGER',
```

```
 'LENDRUM PLACE',
 'MACTAGGART',
 'MAGRATH HEIGHTS',
 'MALMO PLAINS',
 'MAYFIELD',
 'MCKERNAN',
 'MCLEOD',
 'MCQUEEN',
 'MEYOKUMIN',
 'MICHAELS PARK',
 'MINCHAU',
 'NORTH GLENORA',
 'OGILVIE RIDGE',
 'OLESKIW',
 'PARKALLEN',
 'PATRICIA HEIGHTS',
 'PEMBINA',
 'POTTER GREENS',
 'PRINCE CHARLES',
 'PRINCE RUPERT',
 'RHATIGAN RIDGE',
 'RICHFORD',
 'RIDEAU PARK',
 'RIO TERRACE',
 'RIVERDALE',
 'ROSENTHAL',
 'ROSSDALE',
 'RURAL NORTH EAST HORSE HILL',
 'RURAL NORTH EAST SOUTH STURGEON',
 'SHERBROOKE',
 'SHERWOOD',
 'SIFTON PARK',
 'SKYRATTLER',
 'SPRUCE AVENUE',
 'STEINHAUER',
 'STRATHEARN',
 'SUMMERLEA',
 'SWEET GRASS',
 'TAWA',
 'TERRA LOSA',
 'TERRACE HEIGHTS',
 'THE ORCHARDS AT ELLERSLIE',
 'TRUMPETER AREA',
 'TWEDDLE PLACE',
 'VIRGINIA PARK',
 'WEBBER GREENS',
 'WEDGEWOOD HEIGHTS',
 'WEINLOS',
 'WESTRIDGE',
 'WOODCROFT']
```

This code groups the rows of the final dataframe by their assigned cluster label (Cluster_KMeans) using the groupby() method.

Then, it selects the columns Assault, Break and Enter, Homicide, Robbery, Sexual Assaults, Theft From Vehicle, Theft Of Vehicle, and Theft Over $5000, which represent different types of crimes.

Finally, it calculates the mean value of each crime type for each cluster using the mean() method. This allows you to compare the average crime rates between the different clusters.

Averages of the Crimes Commited by Clusters in Edmonton

In [ ]:
```
# Select the relevant columns and filter to only include cluster 0
final.groupby('Cluster_KMeans')['Assault', 'Break and Enter', 'Homicide', 'Robbery',
       'Sexual Assaults', 'Theft From Vehicle', 'Theft Of Vehicle',
       'Theft Over $5000'].mean()
```

```
C:\Users\azimi\AppData\Local\Temp\ipykernel_14576\2800200239.py:2: FutureWarning: Ind
exing with multiple keys (implicitly converted to a tuple of keys) will be deprecate
d, use a list instead.
  final.groupby('Cluster_KMeans')['Assault', 'Break and Enter', 'Homicide', 'Robber
y',
```

Out[ ]:

Cluster_KMeans	Assault	Break and Enter	Homicide	Robbery	Sexual Assaults	Theft From Vehicle	Theft Of Vehicle	Th Ov $50
0	20.469027	22.044248	0.070796	4.415929	2.858407	41.840708	20.238938	1.2035
1	152.111111	83.888889	0.777778	31.888889	15.111111	151.555556	63.333333	3.8888
2	11.090909	20.272727	0.000000	1.454545	0.909091	40.545455	11.090909	1.6363
3	7.536364	12.154545	0.036364	1.927273	1.318182	19.554545	7.481818	0.7636

As we can see, cluster_1 is the hot spot in Edmonton.

We can drill down further to view the mean of crimes committed by each Neighborhood within Cluster 1, that has the Highest Crime rates.

In [ ]:
```
cluster_num = 1
cluster_df = final[final['Cluster_KMeans'] == cluster_num]
ngh_grouped = cluster_df.groupby('NGH_Name')[['Assault', 'Break and Enter', 'Homicide'
       'Sexual Assaults', 'Theft From Vehicle', 'Theft Of Vehicle',
       'Theft Over $5000']].mean()
ngh_grouped
```

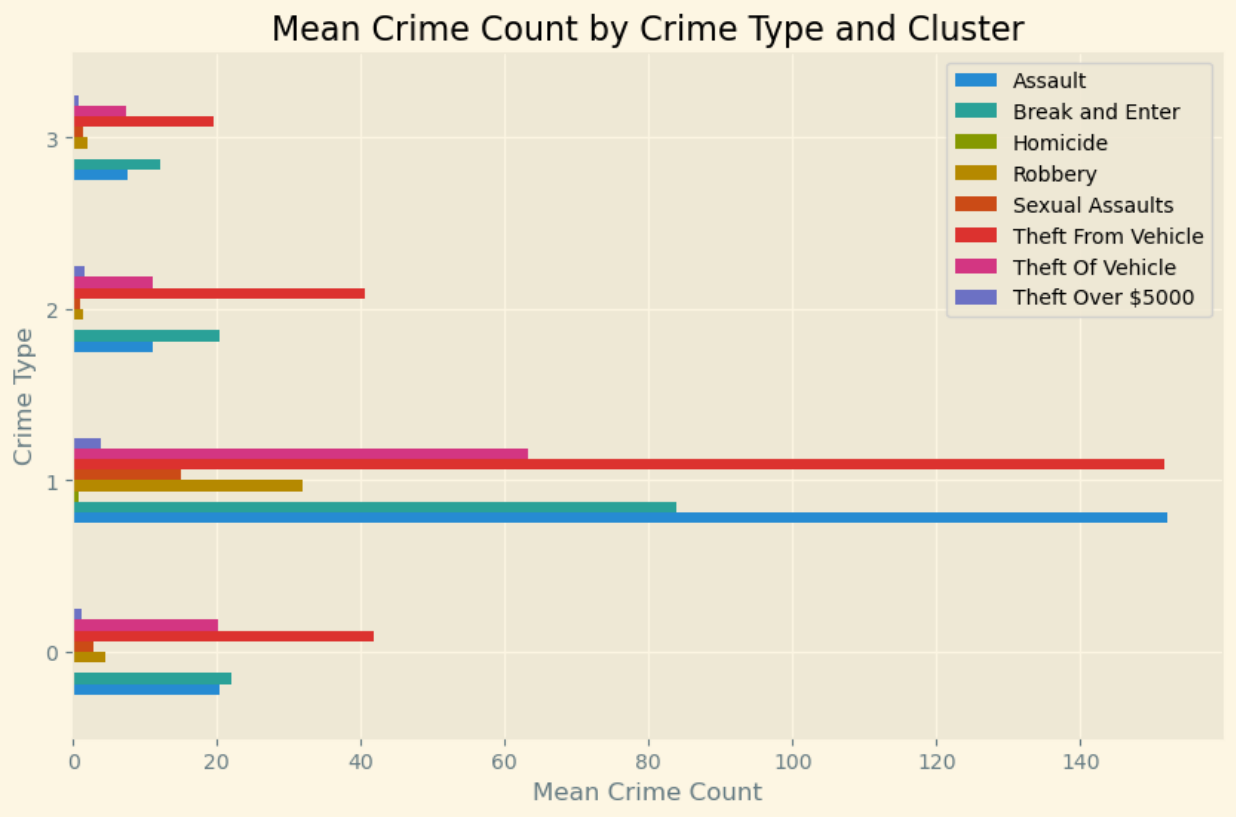
Out[ ]:

NGH_Name	Assault	Break and Enter	Homicide	Robbery	Sexual Assaults	Theft From Vehicle	Theft Of Vehicle	Theft Over $5000
ALBERTA AVENUE	123.0	119.0	1.0	26.0	13.0	156.0	99.0	2.0
BOYLE STREET	163.0	50.0	2.0	16.0	13.0	128.0	44.0	4.0
CENTRAL MCDOUGALL	226.0	50.0	0.0	43.0	15.0	122.0	70.0	5.0
DOWNTOWN	436.0	123.0	1.0	83.0	47.0	316.0	87.0	10.0
GARNEAU	33.0	78.0	0.0	17.0	2.0	83.0	33.0	1.0
INGLEWOOD	80.0	76.0	3.0	23.0	16.0	101.0	47.0	3.0
OLIVER	127.0	118.0	0.0	36.0	16.0	196.0	75.0	5.0
QUEEN MARY PARK	84.0	46.0	0.0	13.0	8.0	138.0	61.0	3.0
STRATHCONA	97.0	95.0	0.0	30.0	6.0	124.0	54.0	2.0

In [ ]:

```
# Get the mean crime count for each crime type for each cluster
crime_means = final.groupby('Cluster_KMeans')['Assault', 'Break and Enter', 'Homicide'
                                              'Sexual Assaults', 'Theft From Vehicle'
                                              'Theft Over $5000'].mean()

# Create a horizontal bar chart
fig, ax = plt.subplots(figsize=(10, 6))
crime_means.plot(kind='barh', ax=ax)

# Set chart title and axis labels
ax.set_title('Mean Crime Count by Crime Type and Cluster')
ax.set_xlabel('Mean Crime Count')
ax.set_ylabel('Crime Type')

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

```
C:\Users\azimi\AppData\Local\Temp\ipykernel_14576\3392201399.py:2: FutureWarning: Ind
exing with multiple keys (implicitly converted to a tuple of keys) will be deprecate
d, use a list instead.
  crime_means = final.groupby('Cluster_KMeans')['Assault', 'Break and Enter', 'Homici
de', 'Robbery',
```



In [ ]:
```
sns.pairplot(cluster_1, hue='NGH_Name', vars=['Assault', 'Break and Enter', 'Homicide'
       'Sexual Assaults', 'Theft From Vehicle', 'Theft Of Vehicle',
       'Theft Over $5000'], diag_kind='kde')
```

Out[ ]:
```
<seaborn.axisgrid.PairGrid at 0x184cd79d760>
```

This cluster is somewhat similar to what we found earlier when looking at the top10 Neighborhoods with the highest Crime Rates. And our Top 6 Neighborhoods are in this cluster too.

> View the Clusters using Geopandas

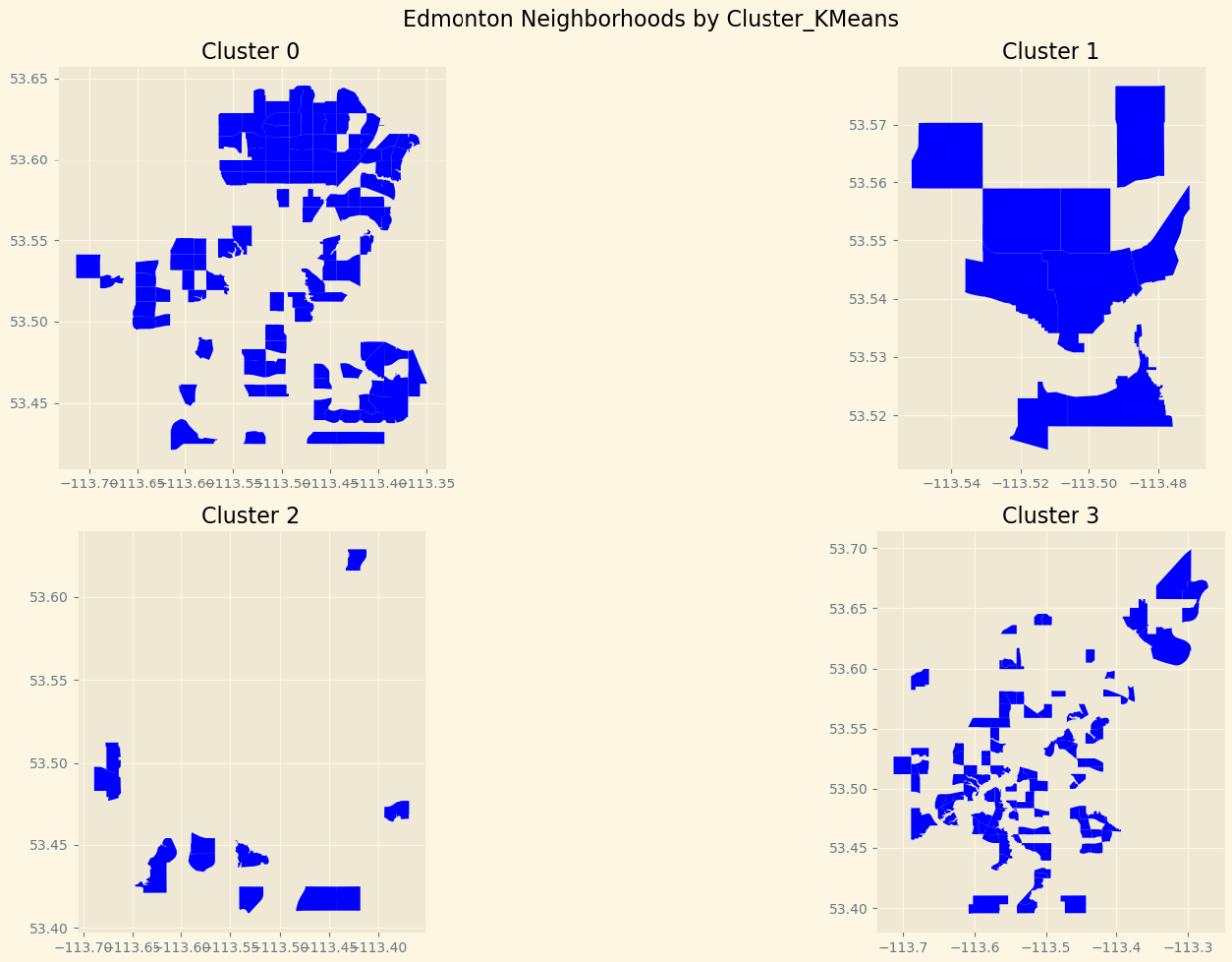
In [ ]:
```
import geopandas as gpd
import matplotlib.pyplot as plt
# Using Geopandas
# first need to get out edmonton/neighbourhoods shape file..and change the columns to
#https://data.edmonton.ca/Geospatial-Boundaries/2016-Federal-Census-Neighbourhoods-as-
edmonton_shp=gpd.read_file('geo_export_67517c45-71c1-4f9b-8051-9eaf76457140.shp', geom
edmonton_shp.rename(columns={'name': 'NGH_Name', 'neighbourh': 'NGH_Number'}, inplace=
edmonton_shp.drop(['descriptiv', 'date_effec', 'time_effec', 'date_eff_2', 'time_eff_2
edmonton_shp.head()
```

Out[ ]:

	NGH_Number	NGH_Name	geometry
0	5310.0	PARKALLEN	POLYGON ((-113.52294 53.50390, -113.52299 53.5...
1	4400.0	PLACE LARUE	POLYGON ((-113.63538 53.53748, -113.63986 53.5...
2	5170.0	EMPIRE PARK	POLYGON ((-113.49897 53.48824, -113.49765 53.4...
3	2270.0	EVERGREEN	POLYGON ((-113.35656 53.62863, -113.35664 53.6...
4	5520.0	UNIVERSITY OF ALBERTA	POLYGON ((-113.51500 53.52575, -113.51500 53.5...

In [ ]:

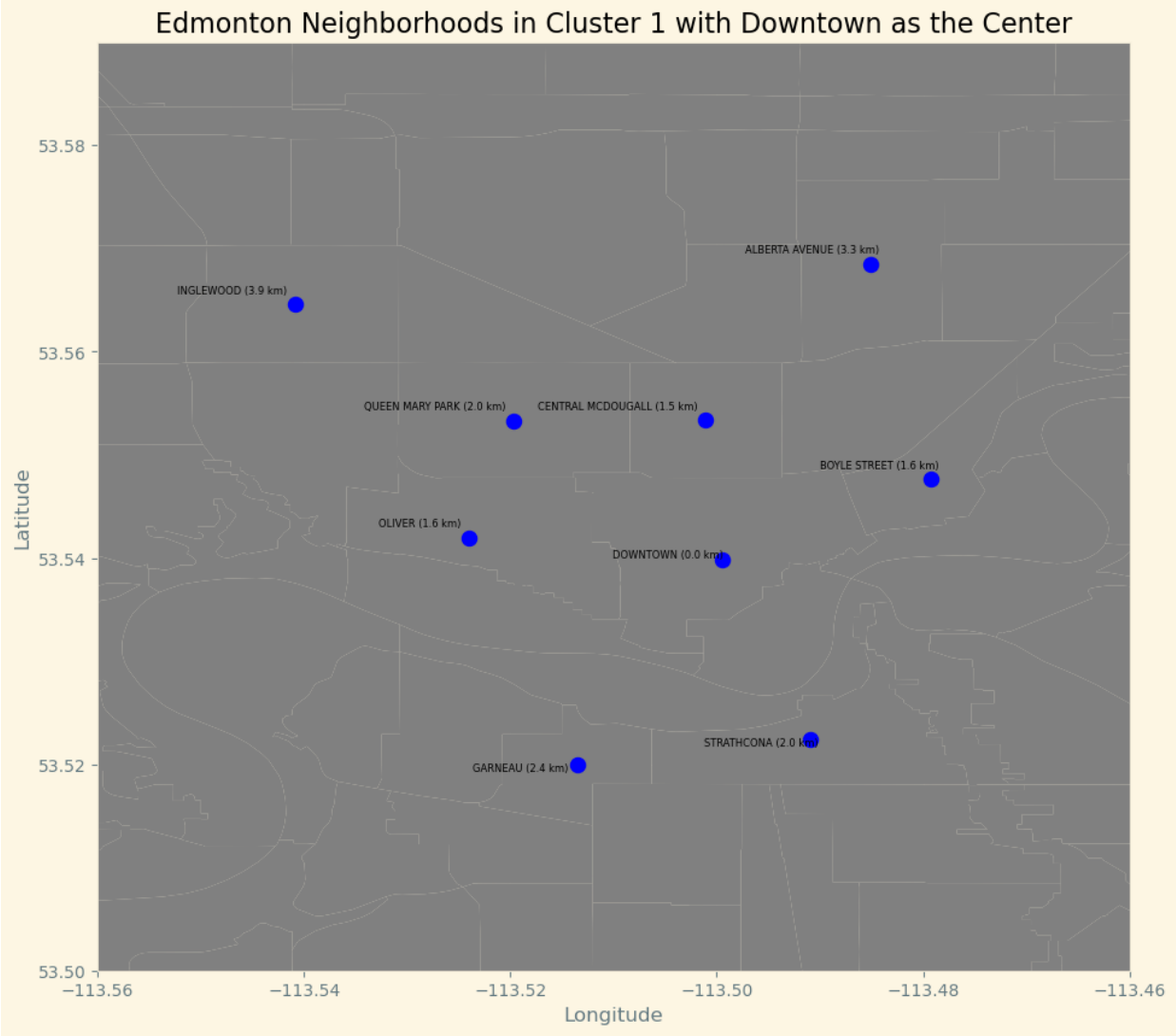
```
import geopandas as gpd
import matplotlib.pyplot as plt

# Join the cluster labels to the Edmonton neighborhoods dataframe
edmonton_shp = edmonton_shp.merge(final[['NGH_Name', 'Cluster_KMeans']], on='NGH_Name'

# Set up a 3x3 grid of subplots
fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(20,10))

# Plot each cluster in a separate subplot
for i, cluster_label in enumerate(range(4)):
    # Select the neighborhoods in the current cluster and plot them on a map
    edmonton_shp[edmonton_shp['Cluster_KMeans'] == cluster_label].plot(ax=axs[i//2, i%
    axs[i//2, i%2].set_title(f'Cluster {cluster_label}')

# Set the overall title for the figure and tighten the layout
fig.suptitle('Edmonton Neighborhoods by Cluster_KMeans', fontsize=16)
plt.tight_layout()
```



> Looking at Cluster_0 with our Neighbourhoods with highest crime. What is the Distance between the Neighborhoods in Cluster 0, with Downtown being set as te center-point.

In [ ]:
```
import geopy.distance
# Define the center point (Downtown)
center_lat = 53.539767
center_lon = -113.499421

# Define the neighborhoods in cluster 1
cluster_0 = final.loc[final['Cluster_KMeans'] == 1, ['NGH_Name', 'Latitude', 'Longitud

# Calculate the distances between each neighborhood and Downtown
distances = []
for i, row in cluster_0.iterrows():
    lat = row['Latitude']
    lon = row['Longitude']
    dist = geopy.distance.distance((center_lat, center_lon), (lat, lon)).km
    distances.append(dist)

# Add the distances as a column in the cluster_0 dataframe
cluster_0['Distance from Downtown'] = distances

# Create a GeoDataFrame from the cluster_0 dataframe
cluster_0_geo = gpd.GeoDataFrame(cluster_0, geometry=gpd.points_from_xy(cluster_0.Long
```

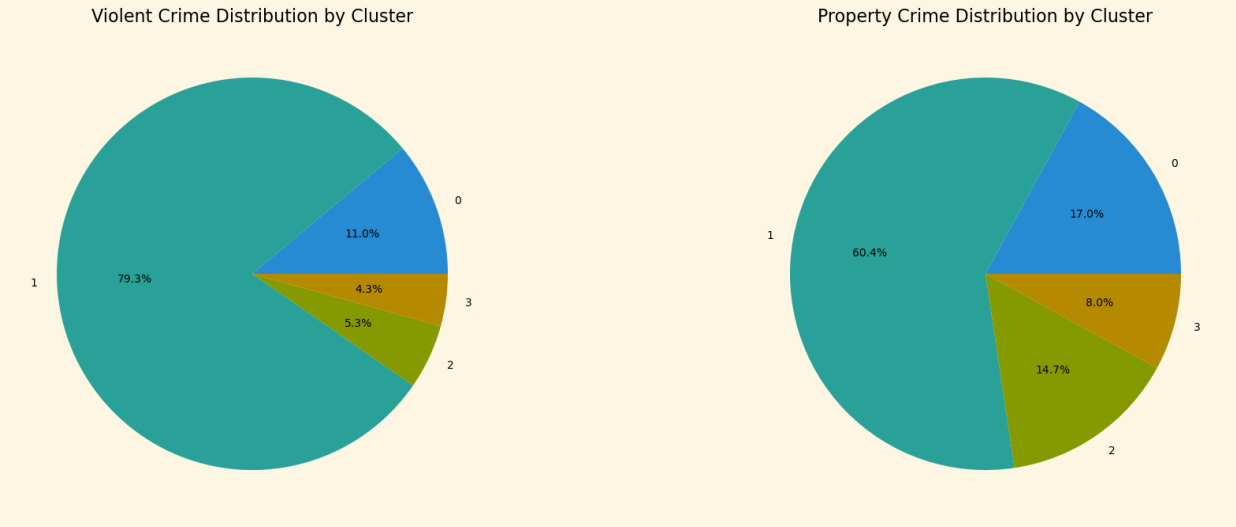
```
# Create a map of the neighborhoods in cluster 0 with Downtown as the center
fig, ax = plt.subplots(figsize=(20, 10))
edmonton_shp.plot(ax=ax, color='gray')
cluster_0_geo.plot(ax=ax, color='blue', marker='o', markersize=75)

# Add labels to the markers indicating the distance from Downtown
for i, row in cluster_0_geo.iterrows():
    label = f"{row['NGH_Name']} ({row['Distance from Downtown']:.1f} km)"
    ax.annotate(label, (row.geometry.x, row.geometry.y), xytext=(np.sign(row.geometry.
                np.sign(row.geometry.y - center_lat)*5), textcoords='offset points', h

# Set the title and axes labels for the map
ax.set_title('Edmonton Neighborhoods in Cluster 1 with Downtown as the Center')
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
ax.set_xlim(-113.56, -113.46)
ax.set_ylim(53.50, 53.59)
plt.show()
```



Looking at the chart above the AVERAGE distance from Downtown as being the centre is around 2.02km.

> Further investigation of the Clusters

In [ ]:
```
# Define the two crime groups
Violent_Crime = ['Homicide', 'Assault', 'Robbery', 'Sexual Assaults']
Property_Crime = ['Break and Enter', 'Theft From Vehicle', 'Theft Of Vehicle', 'Theft

# Calculate the total count of crimes in each cluster for both groups
crime_counts_violent = final.groupby('Cluster_KMeans')[Violent_Crime].mean().sum(axis=
crime_counts_property = final.groupby('Cluster_KMeans')[Property_Crime].mean().sum(axi

# Create a 1x2 grid of subplots
plt.subplot(1, 2, 1)
# Create a pie chart for violent crime
plt.pie(crime_counts_violent, labels=crime_counts_violent.index, autopct='%1.1f%%')
plt.title('Violent Crime Distribution by Cluster')

plt.subplot(1, 2, 2)
# Create a pie chart for property crime
plt.pie(crime_counts_property, labels=crime_counts_property.index, autopct='%1.1f%%')
plt.title('Property Crime Distribution by Cluster')

# Adjust spacing between subplots
plt.subplots_adjust(wspace=0.5)

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



> As we can see from the pie-charts above, our Cluster_0, not only does it have the highest Severe Crimes committed, but also the highest Property Crimes committed too.

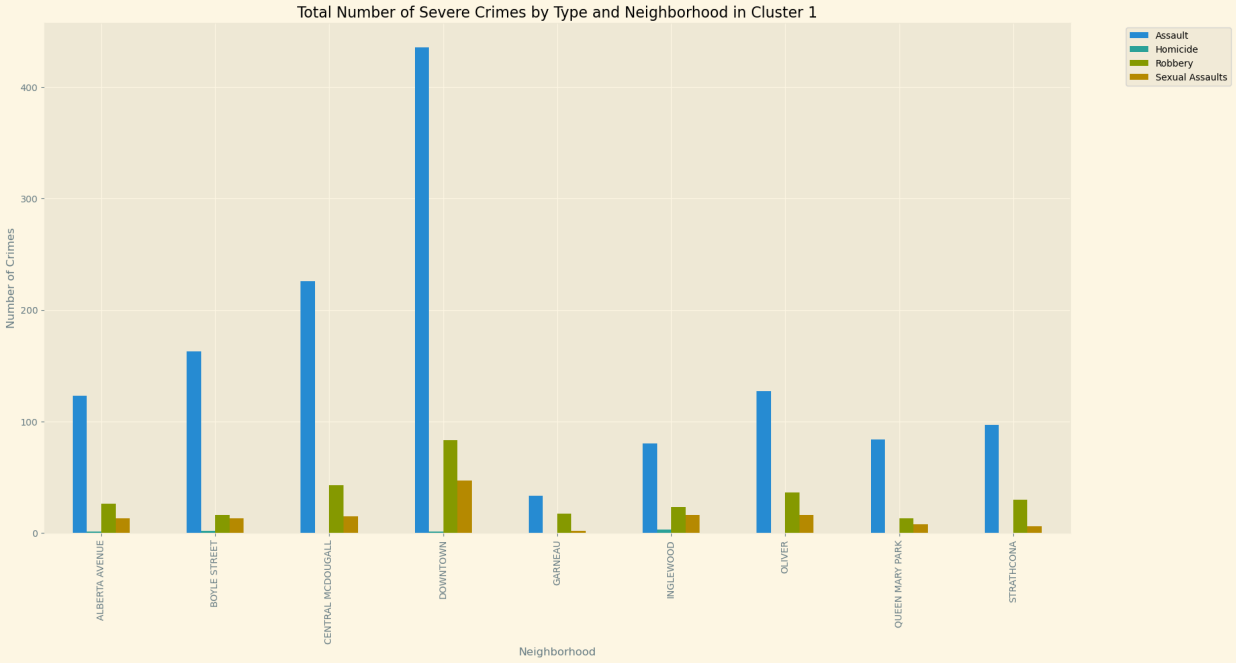
In [ ]:
```
"""Plotting Pie charts for the distribution as a % of each grouped crime by Cluster"""
def plot_violation_type_distribution(data, violation_group):
    # Define the two violation groups
    severe_crime = ['Homicide', 'Assault', 'Robbery', 'Sexual Assaults']
    property_crime = ['Break and Enter', 'Theft From Vehicle', 'Theft Of Vehicle', 'Th

    # Select the appropriate violation group
    if violation_group == 'Severe Crime':
        violation_types = severe_crime
    elif violation_group == 'Property Crime':
        violation_types = property_crime
```

```
    else:
        raise ValueError("Violation group must be 'Severe Crime' or 'Property Crime'."

    # Calculate the total count of each violation type in each cluster
    cluster_counts = data.groupby('Cluster_KMeans')[violation_types].mean()

    # Create a 2x2 grid of subplots for each cluster
    fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(15,10))

    # Loop over each cluster and create a pie chart of the violation type distributior
    for i, ax in enumerate(axs.flat):
        # Get the counts for the current cluster and convert to percentages
        counts = cluster_counts.iloc[i]
        total = counts.sum()
        percents = counts / total * 100

        # Create the pie chart
        labels = percents.index
        sizes = percents.values
        ax.pie(sizes, labels=labels, autopct='%1.1f%%')
        ax.set_title(f'Cluster {i}')

    # Add a common title for the whole figure
    fig.suptitle(f'{violation_group} Distribution by Cluster', fontsize=16)

    # Set the layout and show the plot
    plt.tight_layout()
    plt.show()
```

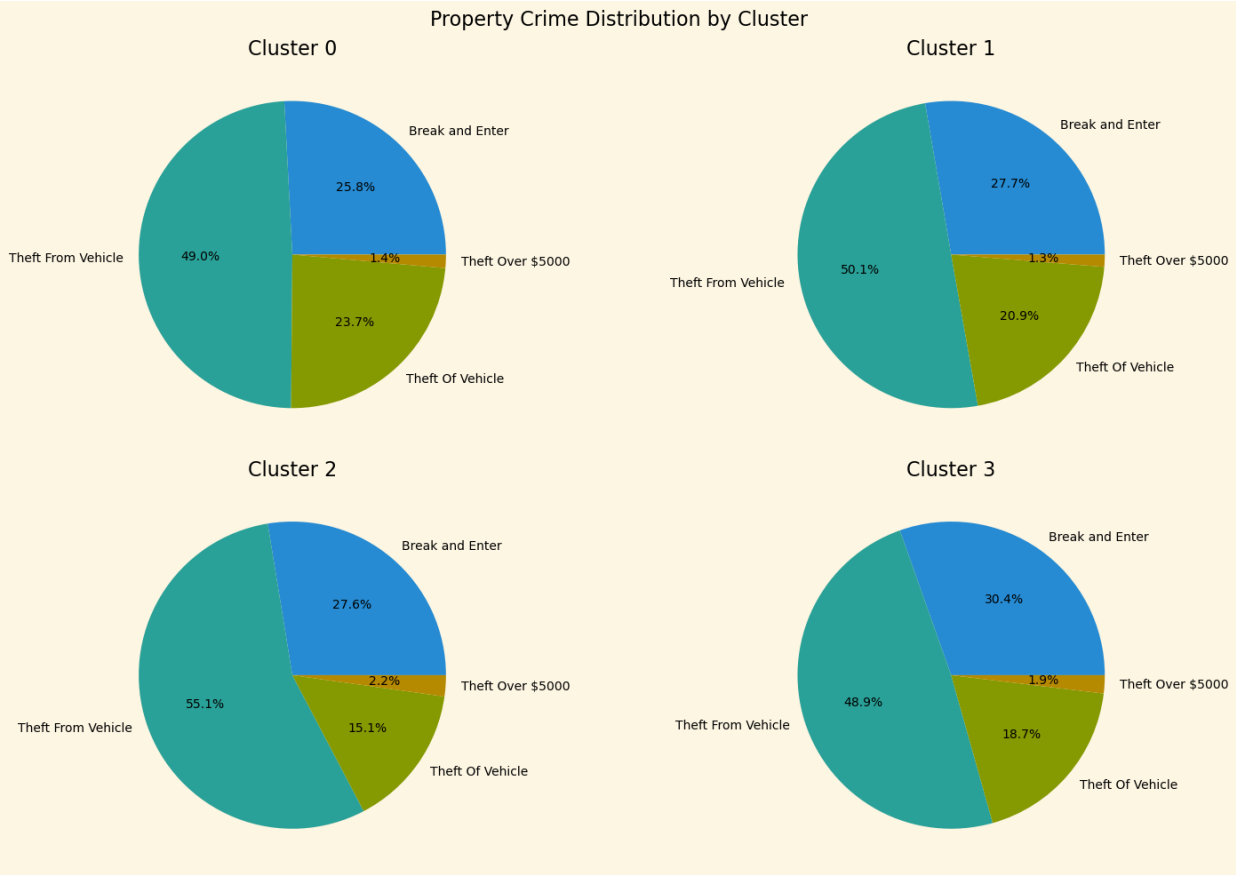
In [ ]:
```
plot_violation_type_distribution(final, "Severe Crime")
```

> Though our Top6 is in Cluster0.....

In [ ]:
```
"""This code will create a bar chart that shows the total number of crimes for each t
    The x-axis shows the neighborhoods, and the y-axis shows the total number of crime
    Each color in the bar chart represents a different type of crime.
    You can use this visualization to see which neighborhoods have the highest crime r
    which types of crimes are most prevalent in each neighborhood."""

# Create a new dataframe for cluster 2
cluster_2_df = final[final['Cluster_KMeans'] == 1]

# Group the data by neighborhood and calculate the total number of crimes for each typ
crime_totals = cluster_2_df.groupby('NGH_Name')[['Assault', 'Homicide', 'Robbery', 'Se

# Create a bar chart showing the total number of crimes for each type of crime in each
crime_totals.plot(kind='bar', figsize=(20,10))
plt.title('Total Number of Severe Crimes by Type and Neighborhood in Cluster 1')
plt.xlabel('Neighborhood')
plt.ylabel('Number of Crimes')
plt.xticks(np.arange(len(crime_totals.index)), crime_totals.index, rotation=90)
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```



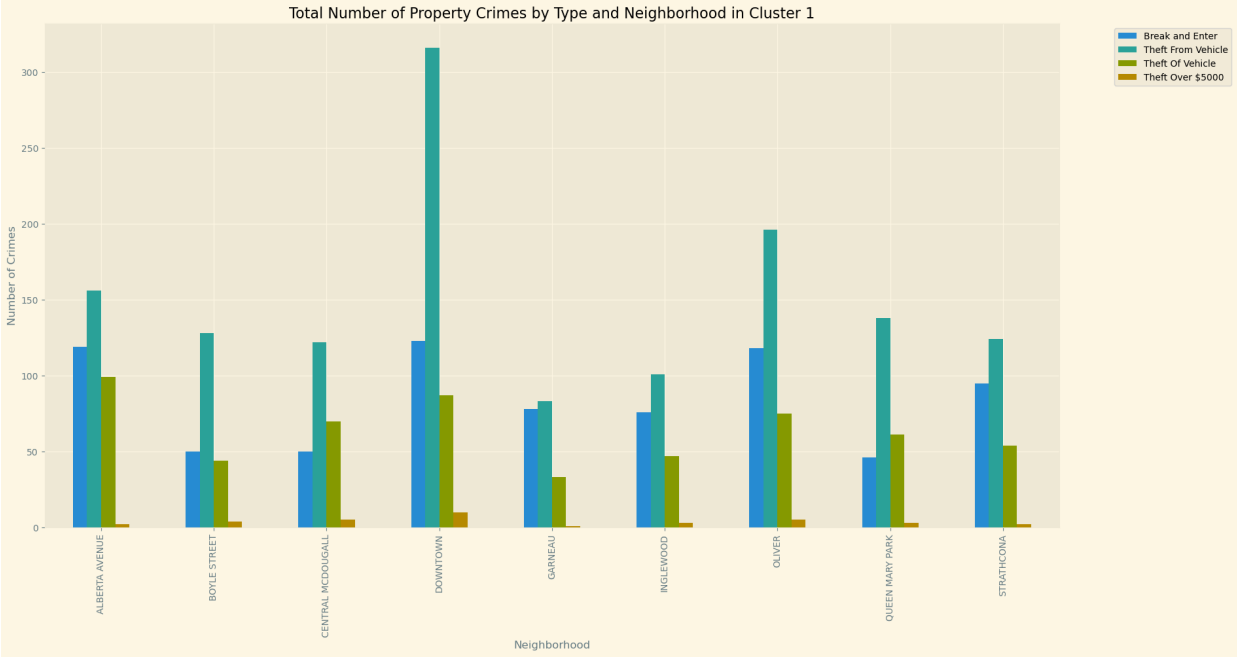
In [ ]:
```
plot_violation_type_distribution(final, "Property Crime")
```



> Again, surprising results concerning our Cluster_0....

In [ ]:
```
"""This code will create a bar chart that shows the total number of crimes for each t
    The x-axis shows the neighborhoods, and the y-axis shows the total number of crime
    Each color in the bar chart represents a different type of crime.
    You can use this visualization to see which neighborhoods have the highest crime r
    which types of crimes are most prevalent in each neighborhood."""
```

```
# Create a new dataframe for cluster 2
cluster_0_df = final[final['Cluster_KMeans'] == 1]

# Group the data by neighborhood and calculate the total number of crimes for each typ
crime_totals = cluster_0_df.groupby('NGH_Name')[['Break and Enter','Theft From Vehicle

# Create a bar chart showing the total number of crimes for each type of crime in each
crime_totals.plot(kind='bar', figsize=(20,10))
plt.title('Total Number of Property Crimes by Type and Neighborhood in Cluster 1')
plt.xlabel('Neighborhood')
plt.ylabel('Number of Crimes')
plt.xticks(np.arange(len(crime_totals.index)), crime_totals.index, rotation=90)
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```

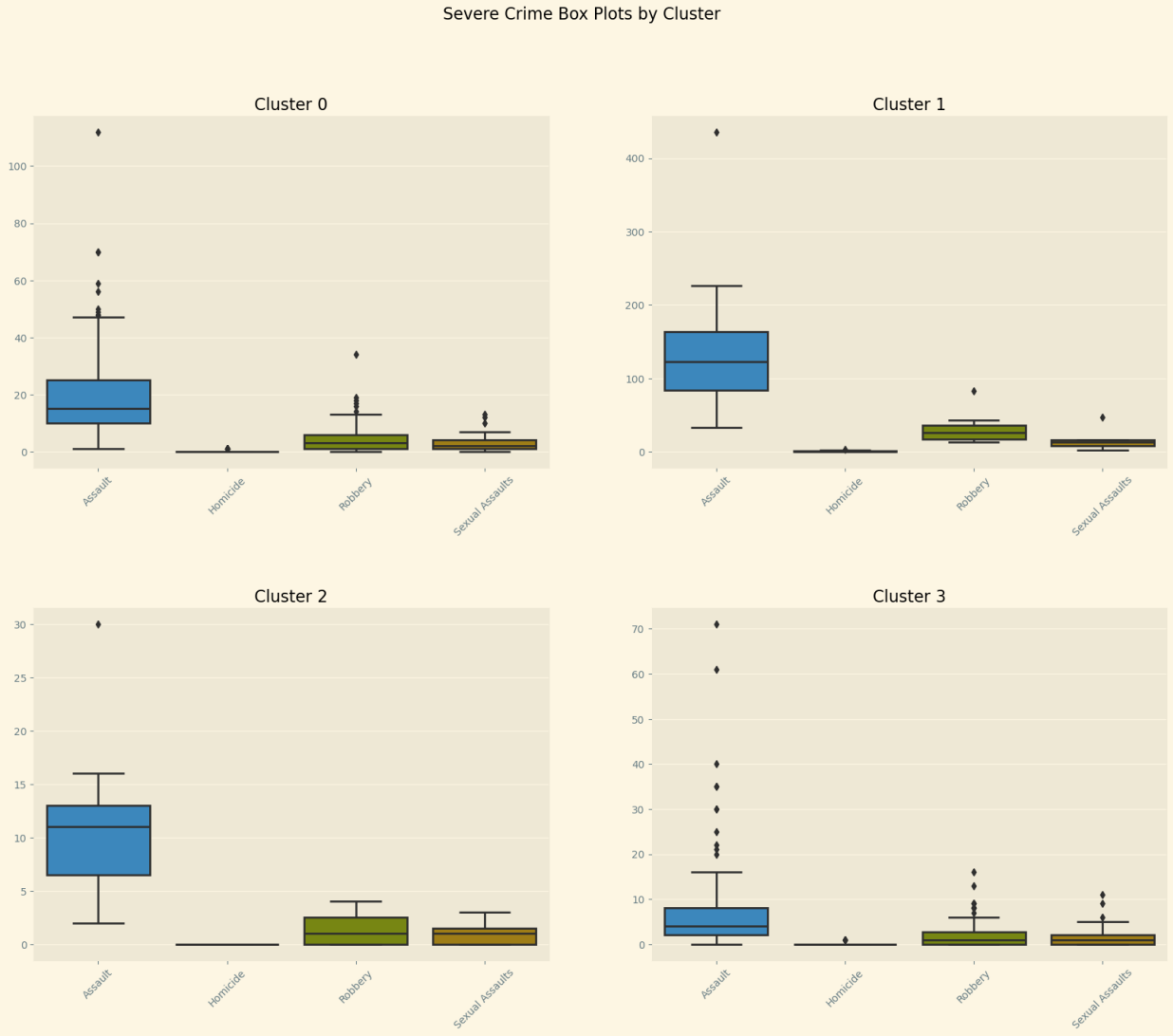


In [ ]:

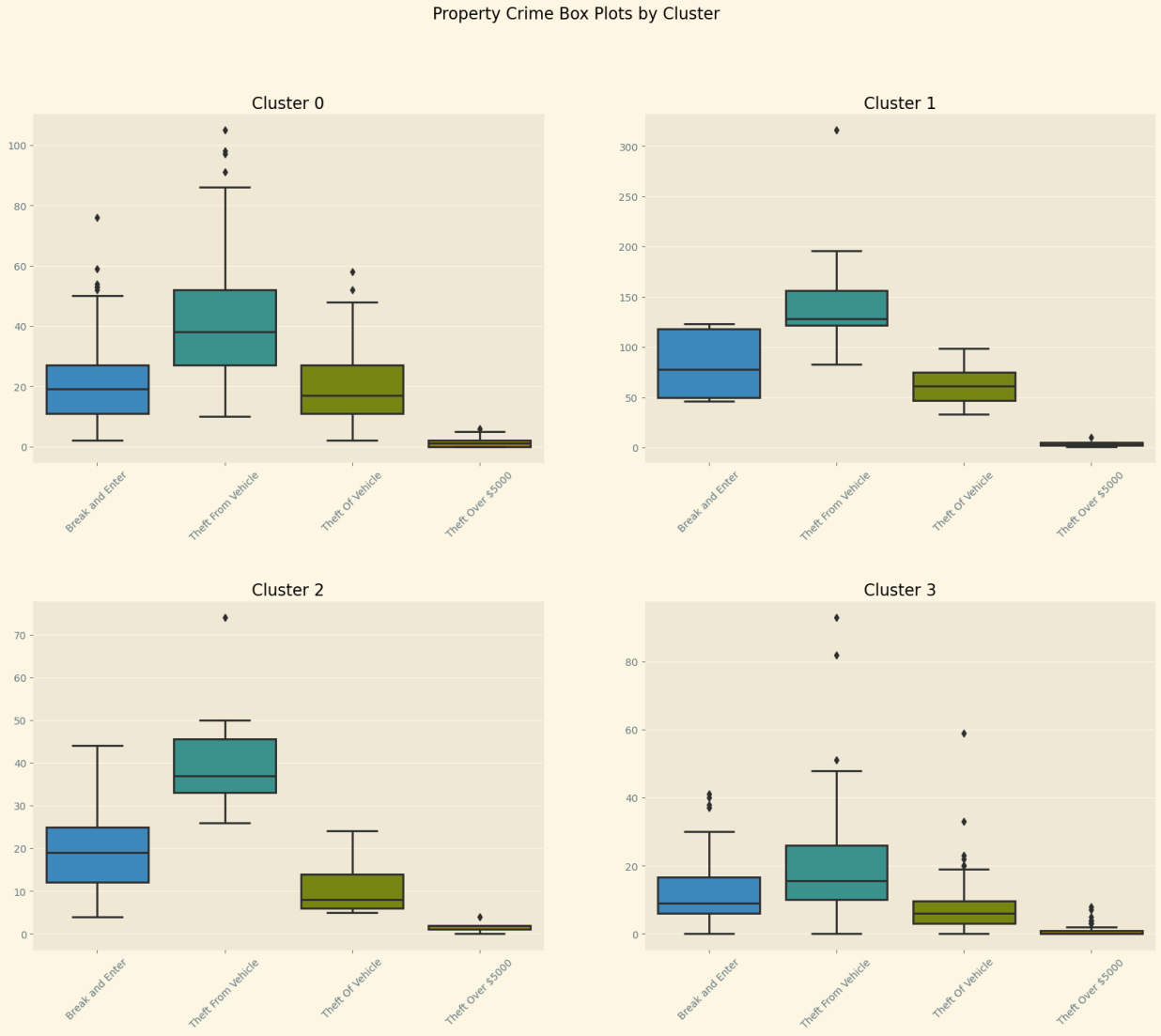
```
# The following funciton build to view boxplots for each cluster defined by:
# Property Crimes, Severe_Crimes, Income_Levels, Employment Status and Education level
def plot_data_boxplots(data, group_type):
    # Define the variables
    property_crime = ['Break and Enter', 'Theft From Vehicle', 'Theft Of Vehicle', 'Th
    severe_crime = ['Assault', 'Homicide', 'Robbery', 'Sexual Assaults']
    income_levels = ['Income_Less than $30K', 'Income_30K to less than 60K', 'Income_6
                     'Income_100K to less than 125K', 'Income_125K to less than 150K',
                     'Income_200K to less than 250K', 'Income_250K or more']
    employment_status = ['Homemaker', 'Employed 0-30', 'Employed 30+', 'Unemployed',
                         'Permanently Unemployed']
    education_level = ['Preschool', 'Kindergarten', 'Gr7-9', 'Gr10-12', 'Post-Secondar

    # Select the variable type
    if group_type == 'Property':
        var_cols = property_crime
    elif group_type == 'Severe':
        var_cols = severe_crime
    elif group_type == 'Income':
        var_cols = income_levels
    elif group_type == 'Employment':
        var_cols = employment_status
    elif group_type == 'Education':
```

```
        var_cols = education_level
    else:
        raise ValueError('Invalid group type')

    # Create a grid of subplots for the variable boxplots
    fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(20, 15))
    plt.subplots_adjust(hspace=0.4)
    for i in range(4):
        row = i // 2
        col = i % 2
        cluster_data = data[data['Cluster_KMeans'].eq(i)]
        sns.boxplot(data=cluster_data[var_cols], ax=axs[row, col])
        axs[row, col].set_title(f'Cluster {i}')
        axs[row, col].tick_params(axis='x', labelrotation=45)

    # Set the title based on the group type
    if group_type == 'Property':
        plt.suptitle('Property Crime Box Plots by Cluster', fontsize=16)
    elif group_type == 'Severe':
        plt.suptitle('Severe Crime Box Plots by Cluster', fontsize=16)
    elif group_type == 'Income':
        plt.suptitle('Income Level Box Plots by Cluster', fontsize=16)
    elif group_type == 'Employment':
        plt.suptitle('Employment Status Box Plots by Cluster', fontsize=16)
    elif group_type == 'Education':
        plt.suptitle('Education Level Box Plots by Cluster', fontsize=16)
    else:
        raise ValueError('Invalid group type')

    plt.show()
```

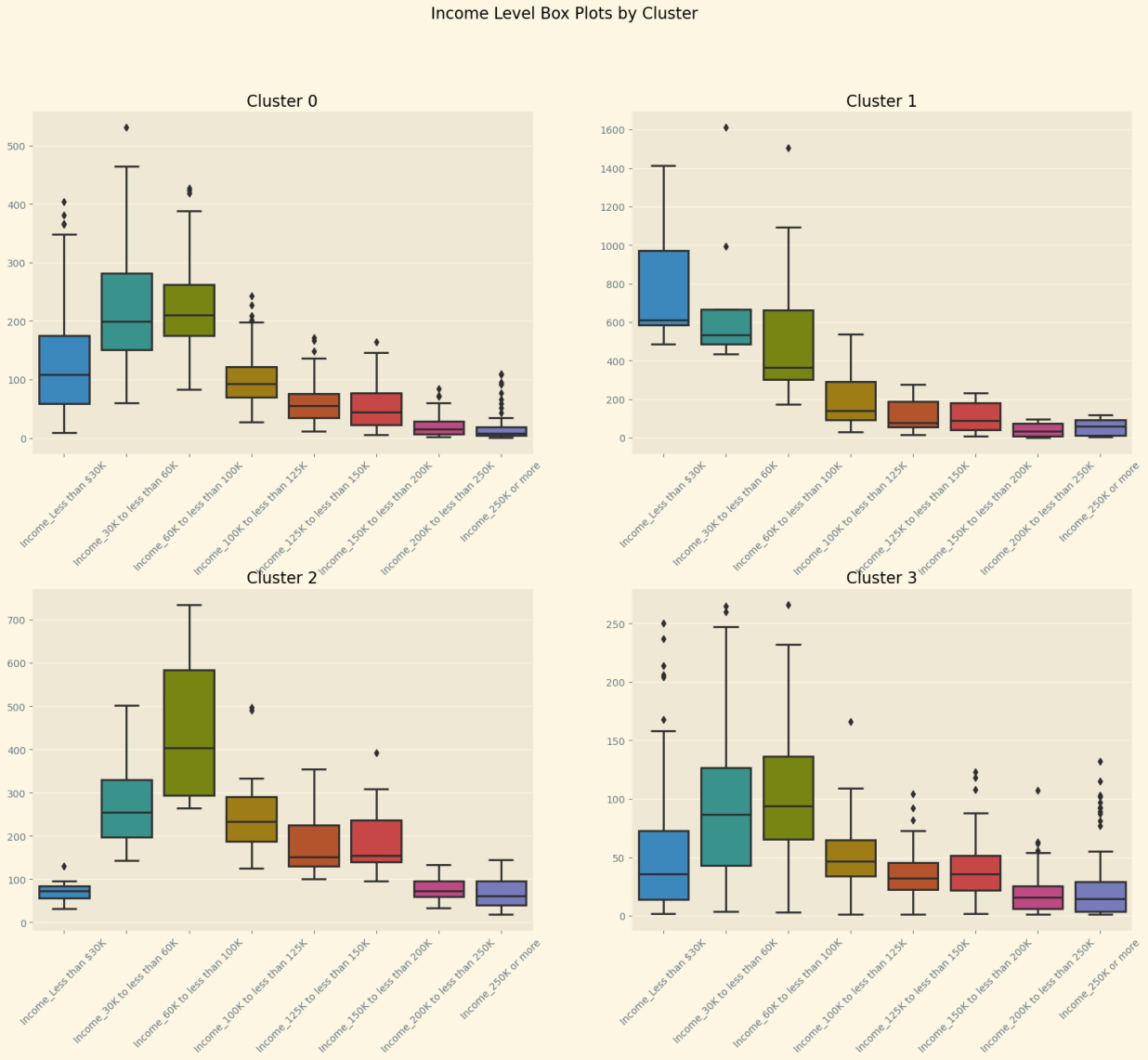
In [ ]:
```
# Severe Crimes by Cluster
plot_data_boxplots(final, 'Severe')
```



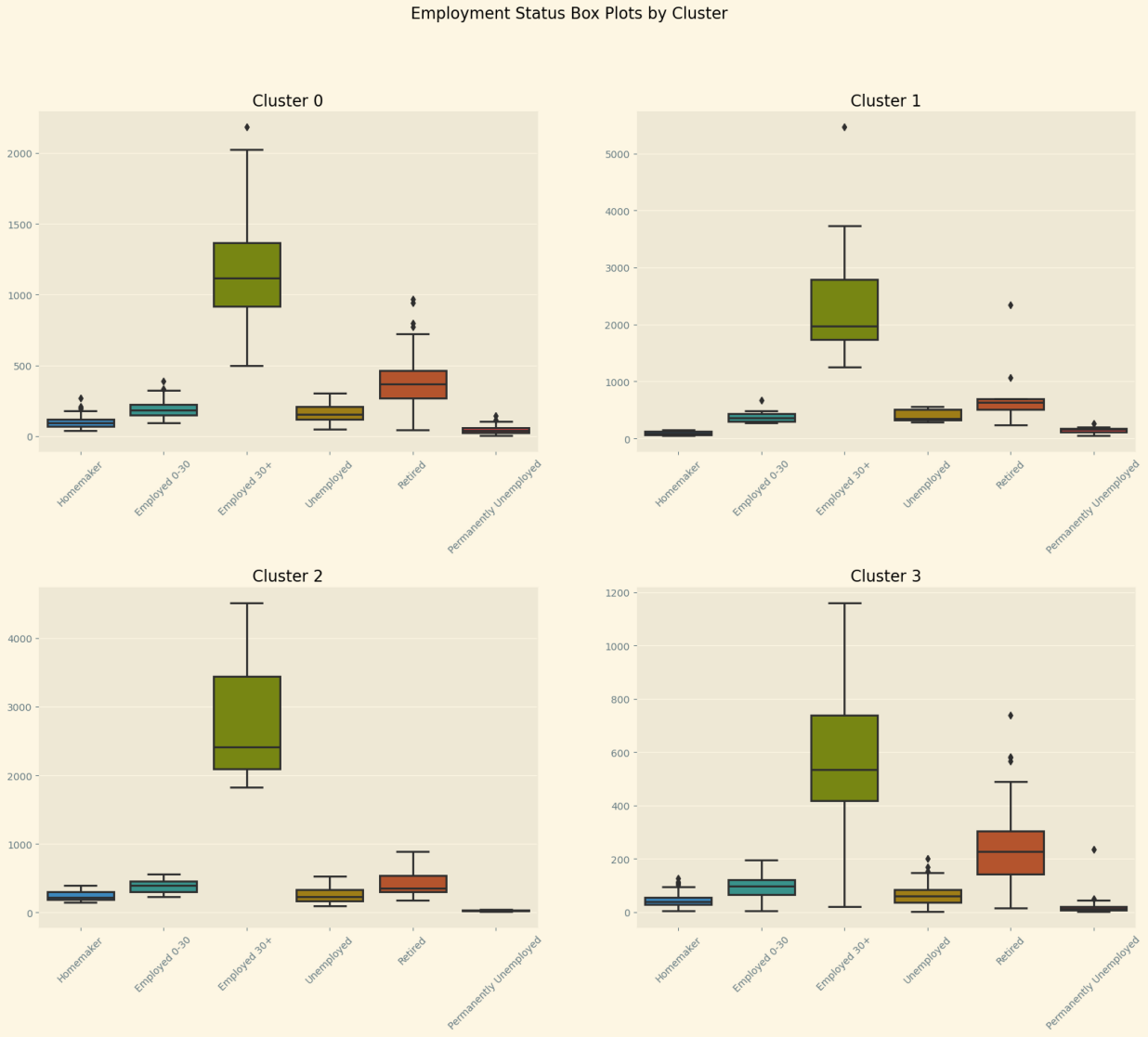
In [ ]:
```
#Property Crimes by Cluster
plot_data_boxplots(final, 'Property')
```



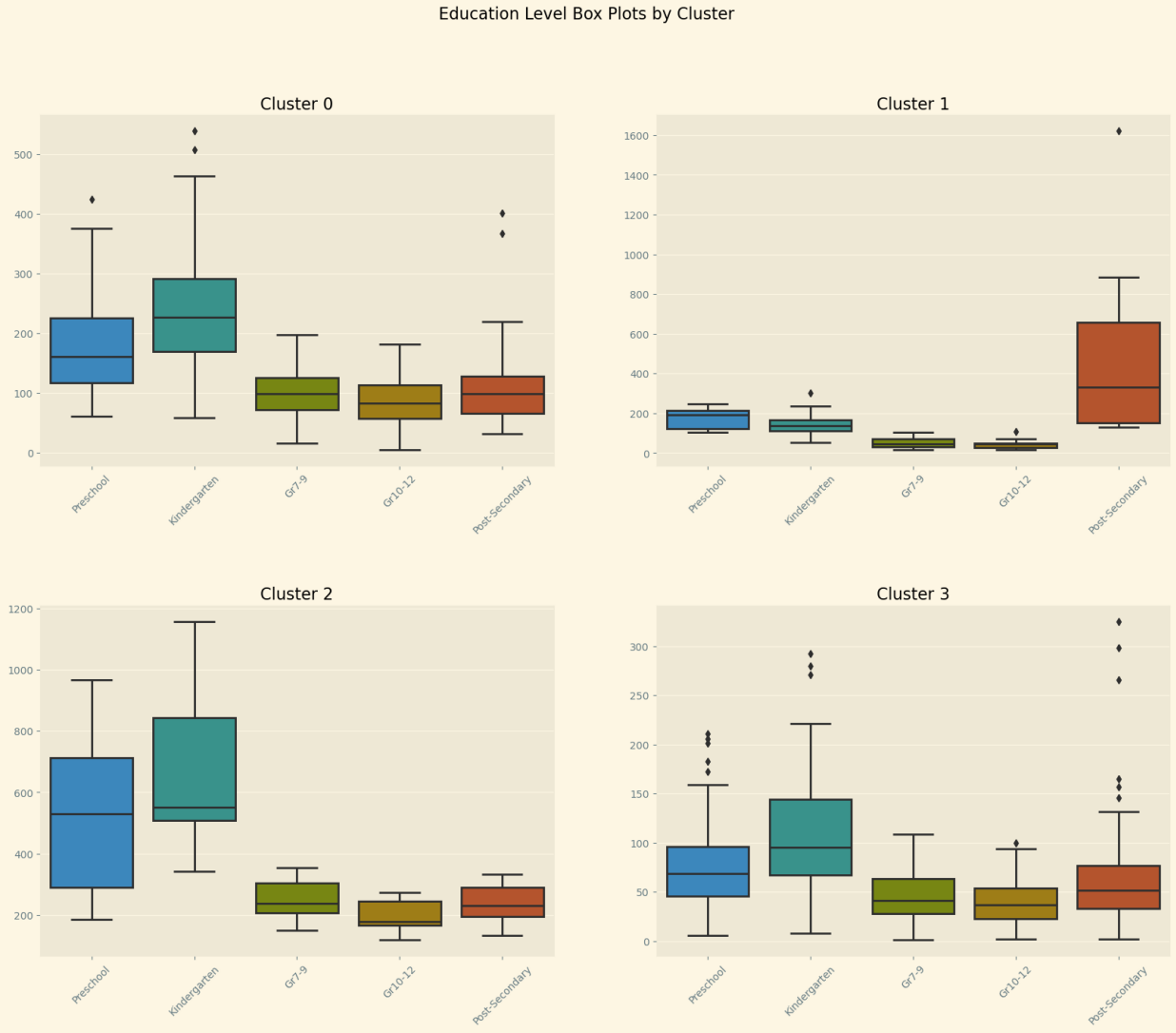
In [ ]:
```
# Income distribution by Cluster
plot_data_boxplots(final, 'Income')
```



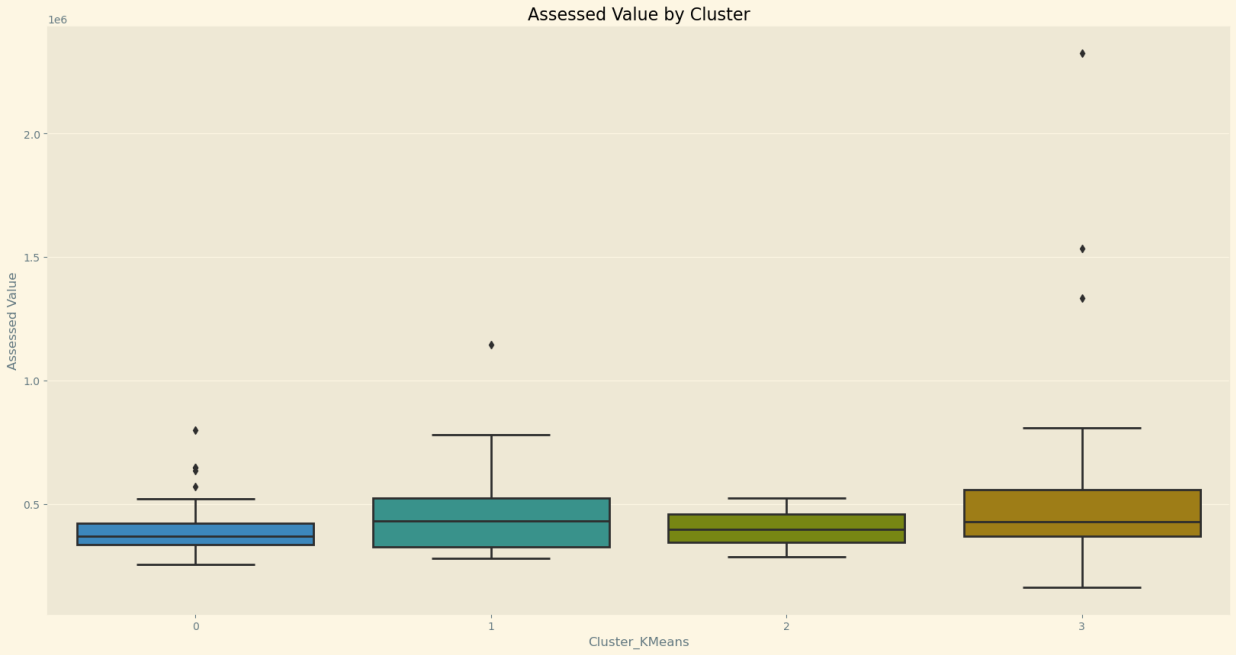
In [ ]:
```
# Employment Status by Cluster
plot_data_boxplots(final, 'Employment')
```



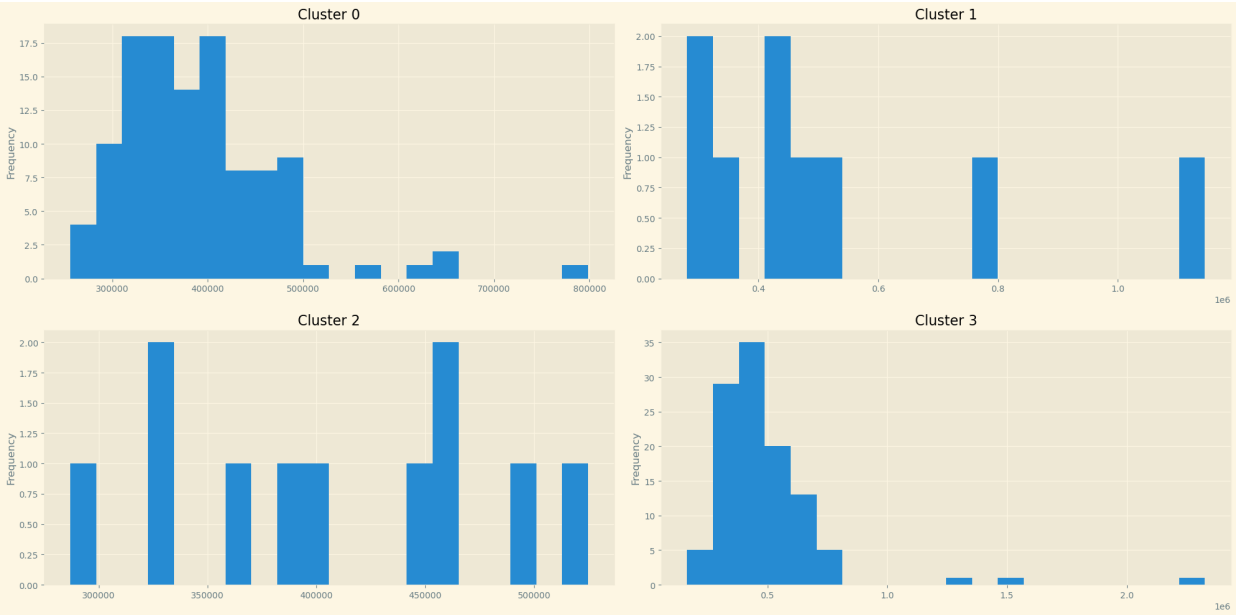
In [ ]:
```
#Education Level by Cluster
plot_data_boxplots(final, 'Education')
```



In [ ]:
```
# Property Assessment Values by Cluster
# Set the figure size
plt.figure(figsize=(20,10))

# Create a boxplot of the Assessed Value by Cluster
sns.boxplot(x='Cluster_KMeans', y='Assessed Value', data=final)
plt.title('Assessed Value by Cluster')
plt.show()
```



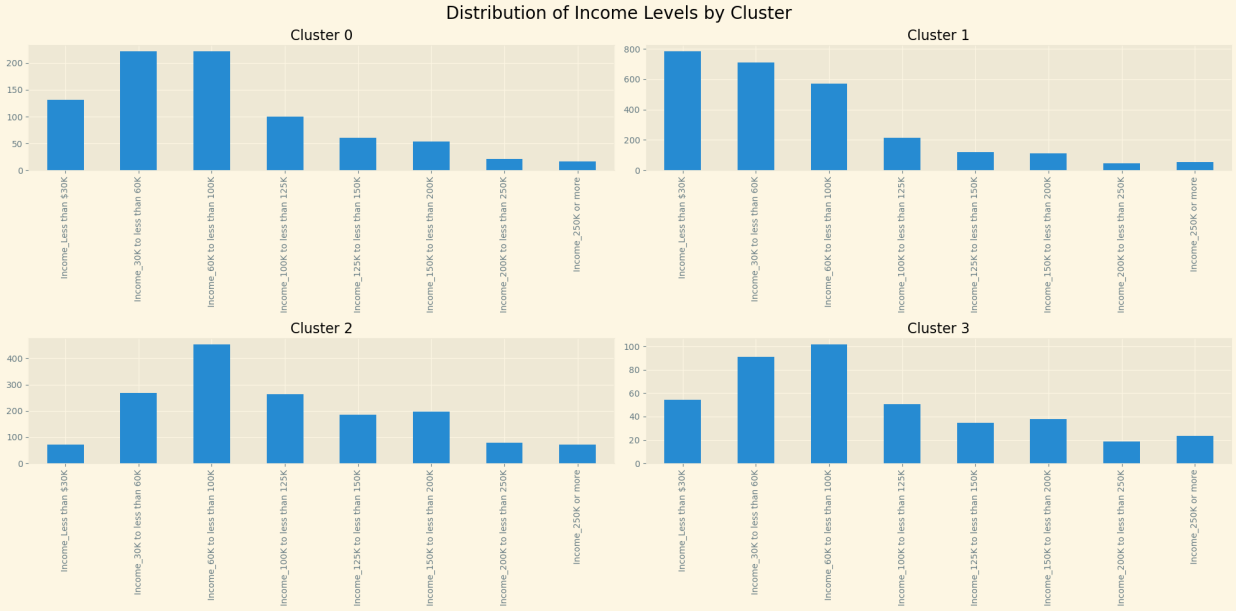
In [ ]:
```
# Create a bar plot of the assessed value for each cluster
fig, axs = plt.subplots(2, 2, figsize=(20,10))
fig.subplots_adjust(hspace=0.4)
for i in range(4):
    cluster_data = final[final['Cluster_KMeans'].eq(i) & (final['Assessed Value'] > 10
    cluster_data['Assessed Value'].plot(kind='hist', bins=20, ax=axs[i//2, i%2], title

plt.tight_layout()
plt.show()
```



> Going deeper into investigating the Clusters by Income Levels, Education Levels and Employment Status.

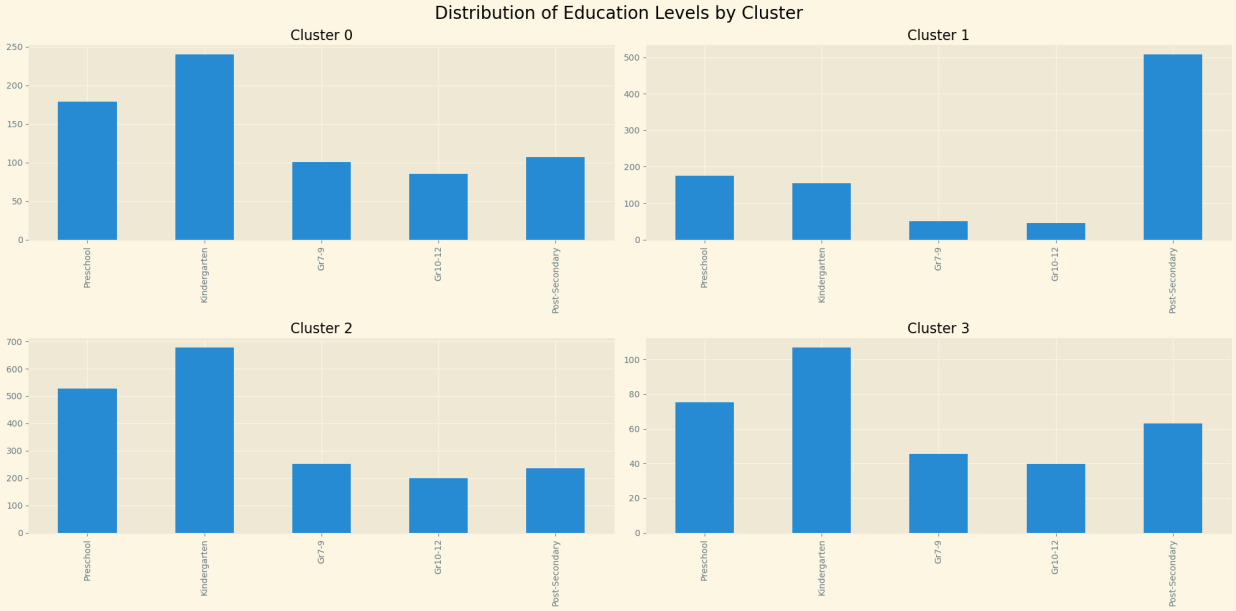
In [ ]:
```
# Create a bar plot of the income levels for each cluster
fig, axs = plt.subplots(2, 2, figsize=(20, 10))
fig.subplots_adjust(hspace=0.4)
```

```
for i in range(4):
    cluster_data = final[final['Cluster_KMeans'].eq(i)]
    income_cols = ['Income_Less than $30K', 'Income_30K to less than 60K', 'Income_60K
                   'Income_100K to less than 125K', 'Income_125K to less than 150K', '
                   'Income_200K to less than 250K', 'Income_250K or more']
    cluster_data[income_cols].mean().plot(kind='bar', ax=axs[i//2, i%2], title=f'Clust
plt.suptitle('Distribution of Income Levels by Cluster', fontsize=20)

plt.tight_layout()
plt.show()
```



In [ ]:

```
# Create a bar plot of the Education levels for each cluster
fig, axs = plt.subplots(2, 2, figsize=(20,10))
fig.subplots_adjust(hspace=0.4)
for i in range(4):
    cluster_data = final[final['Cluster_KMeans'].eq(i)]
    education_cols = ['Preschool', 'Kindergarten', 'Gr7-9', 'Gr10-12', 'Post-Secondary
    cluster_data[education_cols].mean().plot(kind='bar', ax=axs[i//2, i%2], title=f'Cl
plt.suptitle('Distribution of Education Levels by Cluster', fontsize=20)
plt.tight_layout()
plt.show()
```



The presence of a high number of individuals with post-secondary education in Cluster 1 with the highest crime rates can be explained by various factors. For example, it could be that individuals with higher education levels are more likely to report crimes, leading to higher crime rates in areas with more educated residents. Additionally, areas with more educated residents may be more densely populated, which can lead to more crime.

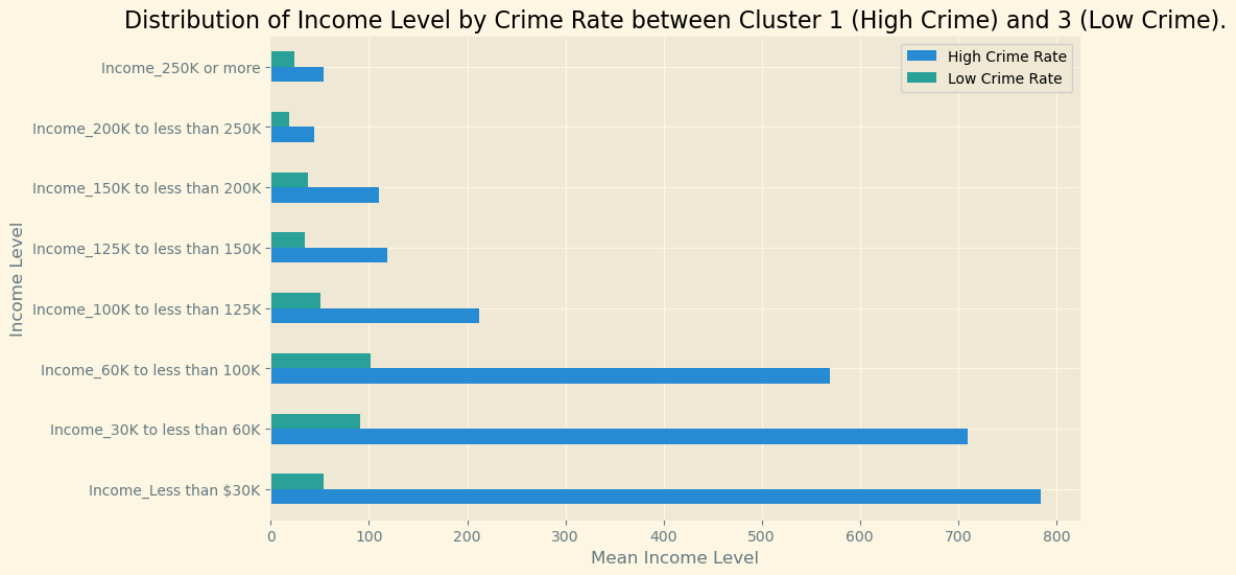
Regarding the income levels in Cluster 0, it's possible that areas with higher crime rates tend to have lower average incomes due to various factors, such as a lack of economic opportunities or higher unemployment rates. Additionally, it's possible that individuals with lower incomes may be more vulnerable to becoming victims of crimes, such as theft and robbery, which could contribute to higher crime rates in areas with lower-income residents.

Lets look at the Demographic Characteristics in Clusters with High and Lower rates of Crimes. First we will look at the Income Level:

- Cluster_3 which has the lowest Crime Rates
- Cluster_1 which has the highest Crime rates

In [ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Select the Income Level columns for each cluster
cluster1_income = final.loc[final['Cluster_KMeans'] == 1,
                            ['Income_Less than $30K', 'Income_30K to less than 60K',
                             'Income_60K to less than 100K', 'Income_100K to less than
                             'Income_125K to less than 150K', 'Income_150K to less tha
                             'Income_200K to less than 250K', 'Income_250K or more']]
cluster3_income = final.loc[final['Cluster_KMeans'] == 3,
                            ['Income_Less than $30K', 'Income_30K to less than 60K',
                             'Income_60K to less than 100K', 'Income_100K to less than
                             'Income_125K to less than 150K', 'Income_150K to less tha
                             'Income_200K to less than 250K', 'Income_250K or more']]
```

```
# Calculate the mean income level for each cluster
cluster1_income_mean = cluster1_income.mean()
cluster3_income_mean = cluster3_income.mean()

# Combine the means into a dataframe
income_mean_df = pd.concat([cluster1_income_mean, cluster3_income_mean], axis=1)
income_mean_df.columns = ['High Crime Rate', 'Low Crime Rate']

# Create a horizontal barplot
ax = income_mean_df.plot(kind='barh', figsize=(10,6))

# Set the chart title and axis labels
ax.set_title('Distribution of Income Level by Crime Rate between Cluster 1 (High Crime
ax.set_xlabel('Mean Income Level')
ax.set_ylabel('Income Level')

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



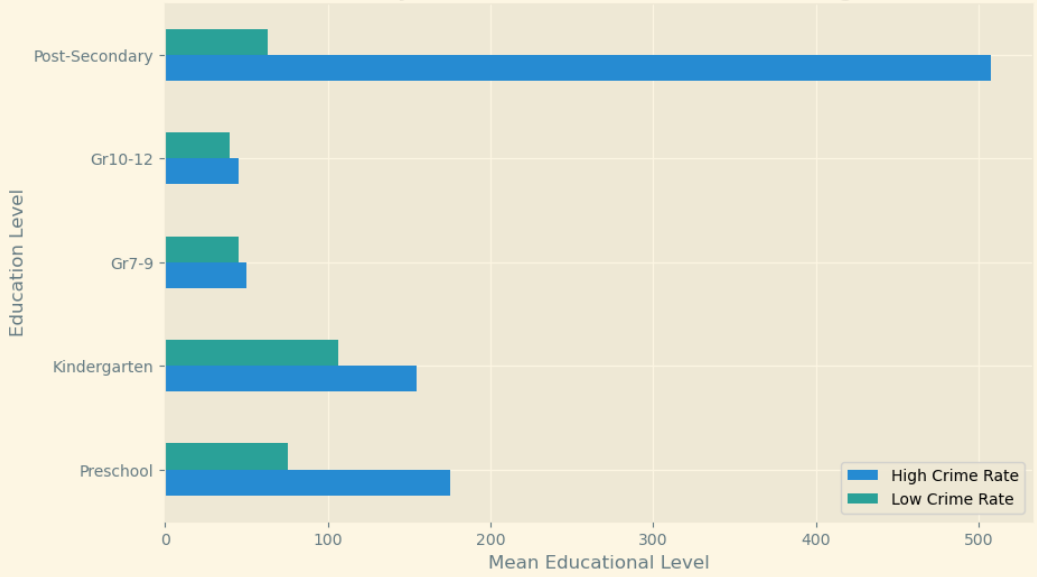
Education Level

In [ ]:

```
# Select the Income Level columns for each cluster
cluster1_education = final.loc[final['Cluster_KMeans'] == 1,
                               ['Preschool','Kindergarten', 'Gr7-9', 'Gr10-12', 'Post-Sec
cluster3_education = final.loc[final['Cluster_KMeans'] == 3,
                               ['Preschool','Kindergarten', 'Gr7-9', 'Gr10-12', 'Post-Sec

# Calculate the mean income level for each cluster
cluster1_education_mean = cluster1_education.mean()
cluster3_education_mean = cluster3_education.mean()

# Combine the means into a dataframe
education_mean_df = pd.concat([cluster1_education_mean, cluster3_education_mean], axis
education_mean_df.columns = ['High Crime Rate', 'Low Crime Rate']

# Create a horizontal barplot
ax = education_mean_df.plot(kind='barh', figsize=(10,6))

# Set the chart title and axis labels
ax.set_title('Distribution of Education Level by Crime Rate between Cluster 1 (High Cr
ax.set_xlabel('Mean Educational Level')
```

```
ax.set_ylabel('Education Level')

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

Distribution of Education Level by Crime Rate between Cluster 1 (High Crime) and 3 (Low Crime).



What about Employment Status?

In [ ]:
```
# Select the Income Level columns for each cluster
cluster1_employment = final.loc[final['Cluster_KMeans'] == 1,
                                ['Homemaker', 'Employed 0-30', 'Employed 30+', 'Unemployed
       'Permanently Unemployed']]
cluster3_employment = final.loc[final['Cluster_KMeans'] == 3,
                                ['Homemaker','Employed 0-30', 'Employed 30+', 'Unemployed'
       'Permanently Unemployed']]

# Calculate the mean income level for each cluster
cluster1_employment_mean = cluster1_employment.mean()
cluster3_employment_mean = cluster3_employment.mean()

# Combine the means into a dataframe
employment_mean_df = pd.concat([cluster1_employment_mean, cluster3_employment_mean], a
employment_mean_df.columns = ['High Crime Rate', 'Low Crime Rate']

# Create a horizontal barplot
ax = employment_mean_df.plot(kind='barh', figsize=(10,6))

# Set the chart title and axis labels
ax.set_title('Distribution of Employment Status by Crime Rate between Cluster 1 (High
ax.set_xlabel('Mean Employment Level')
ax.set_ylabel('Employment Level')

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

> Lets try the Gaussian Mixture Method

In [ ]:
```
final_GMM=final.copy()
# Removing the Assessed Value and Cluster_KMeans columns.
# Looking at the distribution of the Assessed Values column, there was no point in kee
final_GMM.drop(['Assessed Value', 'Cluster_KMeans'], axis=1, inplace =True)
```

In [ ]:
```
final_GMM.iloc[:,1:] # we dont need NGH_Name column
```

Out[ ]:

	Assault	Break and Enter	Homicide	Robbery	Sexual Assaults	Theft From Vehicle	Theft Of Vehicle	Theft Over $5000	Latitude	Longitude	.
0	35	8	0	8	4	16	6	0	53.574143	-113.388758	.
1	8	8	0	2	2	9	2	1	53.632382	-113.549464	.
2	123	119	1	26	13	156	99	2	53.568485	-113.485119	.
3	17	19	0	6	4	46	23	0	53.516888	-113.641242	.
4	6	17	0	1	0	12	3	4	53.401301	-113.526641	.
...	...	...	...	...	...	...	...	...	...	...	.
238	43	34	0	7	6	53	48	1	53.575942	-113.498585	.
239	14	15	0	2	2	46	24	1	53.470564	-113.381167	.
240	7	22	0	3	1	31	6	4	53.432563	-113.626008	.
241	30	22	0	13	4	40	20	4	53.564595	-113.558327	.
242	24	21	0	4	2	48	17	4	53.602843	-113.430212	.

243 rows × 31 columns

In [ ]:
```
scaler = StandardScaler()
df_scaled_GMM = scaler.fit_transform(final_GMM.iloc[:,1:])
```

In [ ]:
```
from sklearn.mixture import GaussianMixture

# Create the GMM model and fit to the standardized data
gmm = GaussianMixture(n_components=4, random_state=42)
#gmm = GaussianMixture(n_components=None, random_state=42)
pred_y = gmm.fit_predict(df_scaled_GMM)

# Add the cluster labels to the original dataframe
final_GMM['Cluster_GMM'] = pred_y
```

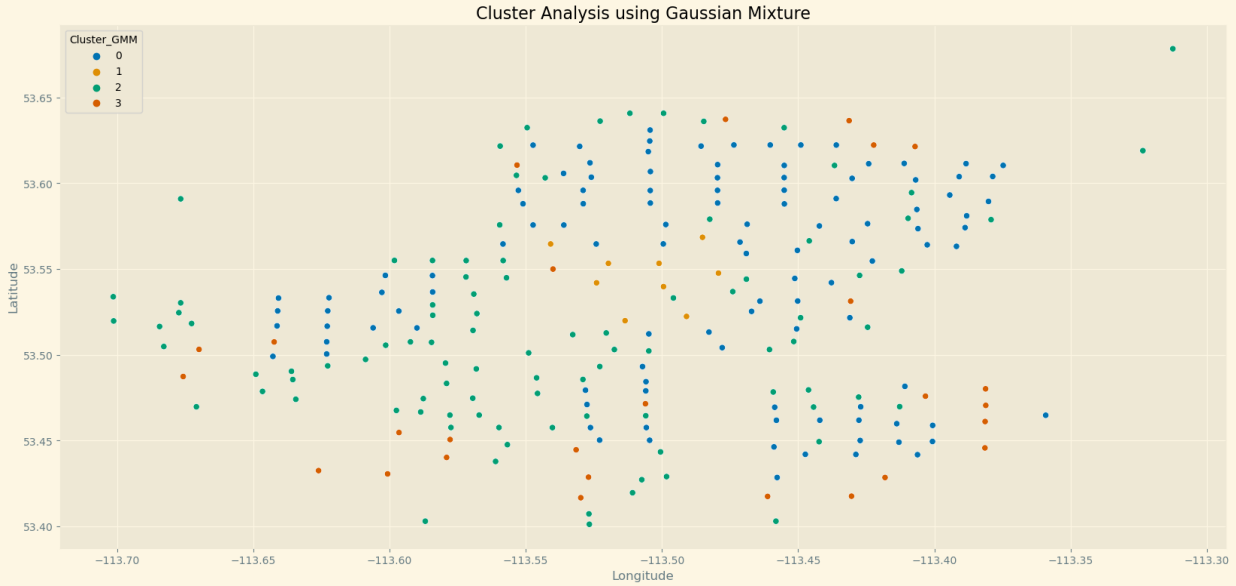
In [ ]:
```
final_GMM.head(5)
```

Out[ ]:

	NGH_Name	Assault	Break and Enter	Homicide	Robbery	Sexual Assaults	Theft From Vehicle	Theft Of Vehicle	Theft Over $5000	Latitude	..
0	ABBOTTSFIELD	35	8	0	8	4	16	6	0	53.574143	.
1	ALBANY	8	8	0	2	2	9	2	1	53.632382	.
2	ALBERTA AVENUE	123	119	1	26	13	156	99	2	53.568485	.
3	ALDERGROVE	17	19	0	6	4	46	23	0	53.516888	.
4	ALLARD	6	17	0	1	0	12	3	4	53.401301	.

5 rows × 33 columns

In [ ]:
```
# Plot the clusters
sns.scatterplot(x='Longitude', y='Latitude', data=final_GMM, hue='Cluster_GMM', palett
plt.title('Cluster Analysis using Gaussian Mixture')
plt.show()
```



In [ ]:
```
clusterGMM_0=final[final_GMM['Cluster_GMM']==0]
clusterGMM_1=final[final_GMM['Cluster_GMM']==1]
clusterGMM_2=final[final_GMM['Cluster_GMM']==2]
clusterGMM_3=final[final_GMM['Cluster_GMM']==3]
```

In [ ]:
```
clusterGMM_0['NGH_Name']
```

Out[ ]:
```
0           ABBOTTSFIELD
3             ALDERGROVE
9                ATHLONE
11                BALWIN
12             BANNERMAN
               ...
234            WELLINGTON
235    WEST JASPER PLACE
238              WESTWOOD
241             WOODCROFT
242                  YORK
Name: NGH_Name, Length: 111, dtype: object
```

In [ ]:
```
clusterGMM_1['NGH_Name']
```

Out[ ]:
```
2         ALBERTA AVENUE
32          BOYLE STREET
53     CENTRAL MCDOUGALL
69               DOWNTOWN
88                GARNEAU
113             INGLEWOOD
165                OLIVER
182      QUEEN MARY PARK
211            STRATHCONA
Name: NGH_Name, dtype: object
```

In [ ]:
```
clusterGMM_2['NGH_Name']
```

Out[ ]:
```
1                 ALBANY
4                 ALLARD
5              ALLENDALE
7                 ARGYLL
8           ASPEN GARDENS
              ...
226       TRUMPETER AREA
227        TWEDDLE PLACE
231        WEBBER GREENS
232    WEDGEWOOD HEIGHTS
237            WESTRIDGE
Name: NGH_Name, Length: 96, dtype: object
```

In [ ]:
```
clusterGMM_3['NGH_Name']
```

Out[ ]:
```
6              AMBLESIDE
35             BRINTNELL
56          CHARLESWORTH
62            CUMBERLAND
70                DUGGAN
89           GLASTONBURY
100               HADDOW
108       HOLLICK-KENYON
126      KINISKI GARDENS
128           KLARVATTEN
132             LARKSPUR
134               LAUREL
140              LYMBURN
142              MACEWAN
149      MCCONACHIE AREA
167             OTTEWELL
198           RUTHERFORD
206         SILVER BERRY
208    SOUTH TERWILLEGAR
215           SUMMERSIDE
221    TERWILLEGAR TOWNE
222         THE HAMPTONS
228          TWIN BROOKS
230               WALKER
236            WESTMOUNT
239            WILD ROSE
240           WINDERMERE
Name: NGH_Name, dtype: object
```

Using GMM, our Top10 popular Neighborhoods have been clustered into ClusterGMM_3.

In [ ]:
```
# Compare to KMeans Cluster_0
cluster_1['NGH_Name']
```

Out[ ]:
```
2         ALBERTA AVENUE
32          BOYLE STREET
53     CENTRAL MCDOUGALL
69               DOWNTOWN
88                GARNEAU
113             INGLEWOOD
165                OLIVER
182      QUEEN MARY PARK
211           STRATHCONA
Name: NGH_Name, dtype: object
```

In [ ]:
```
# Define the two crime groups
Violent_Crime = ['Homicide', 'Assault', 'Robbery', 'Sexual Assaults']
Property_Crime = ['Break and Enter', 'Theft From Vehicle', 'Theft Of Vehicle', 'Theft

# Calculate the total count of crimes in each cluster for both groups
crime_counts_violent = final_GMM.groupby('Cluster_GMM')[Violent_Crime].mean().sum(axis
crime_counts_property = final_GMM.groupby('Cluster_GMM')[Property_Crime].mean().sum(ax

# Create a 1x2 grid of subplots
plt.subplot(1, 2, 1)
# Create a pie chart for violent crime
plt.pie(crime_counts_violent, labels=crime_counts_violent.index, autopct='%1.1f%%')
plt.title('Violent Crime Distribution by Cluster using GMM')

plt.subplot(1, 2, 2)
```

```
# Create a pie chart for property crime
plt.pie(crime_counts_property, labels=crime_counts_property.index, autopct='%1.1f%%')
plt.title('Property Crime Distribution by Cluster using GMM')

# Adjust spacing between subplots
plt.subplots_adjust(wspace=0.5)

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

In [ ]:
```
final.to_csv('Final_with_clustersV2.csv')
```

Statistical Tests on Clusters

Since all our numerical columns are DISCRETE (Counts), we can perform statistical analyses within and between clusters to investigate relationships between variables. Since the numerical columns are discrete count variables, we can use methods like chi-squared tests, ANOVA, and correlation analysis to assess the relationships between variables.

For example, we could use a chi-squared test to determine if there is a significant association between two categorical variables, such as the association between employment status and severe crime within a specific cluster.

We could also use ANOVA to compare the means of a continuous variable (such as income levels) across different clusters to determine if there are significant differences in income levels between clusters.

Finally, we could use correlation analysis to investigate the relationships between different variables within a cluster or between clusters, such as the correlation between income levels and severe crime within a specific cluster or between different clusters.

However, with a small sample size, it may be difficult to meet the assumptions of normality required for ANOVA. In such cases, non-parametric tests such as the Kruskal-Wallis test may be more appropriate. And also, having small cluster sizes may also be an issue for any statistical

analysis. In general, larger sample sizes are better for statistical analysis as they tend to produce more reliable and accurate results.

Chi-squared test also has assumptions regarding the sample size and expected frequencies of the variables. In general, for the Chi-squared test to be reliable, the expected frequencies of each cell should be at least 5, and the sample size should be large enough.

In the case of our dataset, the small sample size and low counts in some categories may not meet the assumptions of the Chi-squared test. Therefore, it may not be the best approach for analyzing the data.

We will use the Kruskal-Wallis test. The Kruskal-Wallis test is a non-parametric alternative to ANOVA, which is used when the assumptions of ANOVA are not met. It is used to test the null hypothesis that the median of a variable is the same across multiple groups. It can be used to test for differences between three or more independent groups.

In [ ]:
```
from scipy import stats
from scipy.stats import kruskal

# Subset the data to only include the cluster of interest
cluster_0_data = final[final['Cluster_KMeans'] == 0]

# Perform Kruskal-Wallis test on Assault variable for each cluster
stats, pvalue = kruskal(cluster_0_data['Assault'], cluster_1['Assault'], cluster_2['As

# Print the results
print(f"Kruskal-Wallis test statistic: {stats:.2f}")
print(f"P-value: {pvalue:.4f}")
```

```
Kruskal-Wallis test statistic: 90.26
P-value: 0.0000
```

In this example, we are comparing the Assault variable across all clusters, but we could also compare other variables of interest. If the p-value is less than the significance level (usually 0.05), then we can reject the null hypothesis and conclude that there are significant differences between the median values of the variable across the clusters.

It looks like the Kruskal-Wallis test found a significant difference in the Assault variable between the four clusters. The p-value of 0.0000 indicates that the probability of observing such a large test statistic under the null hypothesis (that there is no difference in the Assault variable between the clusters) is extremely low, suggesting that we can reject the null hypothesis and conclude that there is a significant difference in the Assault variable between the clusters.

So even though the pie charts (earlier) showed similar proportions of Assault cases in each cluster, the Kruskal-Wallis test may have detected other differences in the distribution of those cases that were not apparent from the pie charts alone, by considering the other columns grouped by Education, Employment and Income.

This suggests that there may be some underlying factors that differentiate the clusters in terms of their overall crime rates, and that these factors may not be apparent from just looking at the

Assault variable alone.

In [ ]:
```
from scipy.stats import kruskal

def kruskal_test(column, final):
    # Subset the data to only include the cluster of interest
    cluster_0_data = final[final['Cluster_KMeans'] == 0]

    # Perform Kruskal-Wallis test on specified variable for each cluster
    stats, pvalue = kruskal(cluster_0_data[column],
                            final[final['Cluster_KMeans'] == 1][column],
                            final[final['Cluster_KMeans'] == 2][column],
                            final[final['Cluster_KMeans'] == 3][column])

    # Print the results
    print(f"Kruskal-Wallis test statistic: {stats:.2f}")
    print(f"P-value: {pvalue:.4f}")
```

In [ ]:
```
kruskal_test('Robbery', final)
```

```
Kruskal-Wallis test statistic: 55.37
P-value: 0.0000
```

For kruskal_test('Robbery', final), the result is a Kruskal-Wallis test statistic of 109.44 and a p-value of 0.0000, indicating that there is a significant difference in the distribution of Robbery cases among the clusters.

In [ ]:
```
final.columns
```

Out[ ]:
```
Index(['NGH_Name', 'Assault', 'Break and Enter', 'Homicide', 'Robbery',
       'Sexual Assaults', 'Theft From Vehicle', 'Theft Of Vehicle',
       'Theft Over $5000', 'Latitude', 'Longitude', 'Preschool',
       'Kindergarten', 'Gr7-9', 'Gr10-12', 'Post-Secondary', 'Homemaker',
       'Employed 0-30', 'Employed 30+', 'Unemployed', 'Retired',
       'Permanently Unemployed', 'Employment_No Response',
       'Income_Less than $30K', 'Income_30K to less than 60K',
       'Income_60K to less than 100K', 'Income_100K to less than 125K',
       'Income_125K to less than 150K', 'Income_150K to less than 200K',
       'Income_200K to less than 250K', 'Income_250K or more',
       'Income_No Response', 'Assessed Value', 'Cluster_KMeans'],
      dtype='object')
```

In [ ]:
```
kruskal_test('Unemployed', final)
```

```
Kruskal-Wallis test statistic: 156.58
P-value: 0.0000
```

It suggests that there is a significant difference in the distribution of unemployment rates across the clusters.

In [ ]:
```
# Subset the data to only include the cluster of interest
cluster_0_data = final[final['Cluster_KMeans'] == 2]

# Perform Kruskal-Wallis test on Assault variable for each cluster
stats, pvalue = kruskal(cluster_0_data['Income_Less than $30K'], cluster_1['Income_Les

# Print the results
```

```
print(f"Kruskal-Wallis test statistic: {stats:.2f}")
print(f"P-value: {pvalue:.4f}")
```

```
Kruskal-Wallis test statistic: 14.14
P-value: 0.0002
```

Since the variables in the dataframe are counts and the dataframe has a limited number of rows, it may not be appropriate to perform statistical tests between the clusters. This is because the assumptions of normality and independence required for many statistical tests may not hold true for count data. In addition, with a limited number of rows, statistical tests may not have enough power to detect significant differences between the clusters.

Calgary

In [ ]:
```
calgary=pd.read_csv('Community_Crime_and_Disorder_Statistics__2012-2019_.csv')
calgary
```

Out[ ]:

	Sector	Community Name	Group Category	Category	Crime Count	Resident Count	Date	Year	Month	
0	NORTH	THORNCLIFFE	Crime	Theft FROM Vehicle	9	8474	2018/03	2018	MAR	T
1	SOUTH	WOODBINE	Crime	Theft FROM Vehicle	3	8866	2019/11	2019	NOV	
2	SOUTH	WILLOW PARK	Crime	Theft FROM Vehicle	4	5328	2019/11	2019	NOV	F
3	SOUTH	WILLOW PARK	Crime	Commercial Robbery	1	5328	2019/11	2019	NOV	
4	WEST	LINCOLN PARK	Crime	Commercial Break & Enter	5	2617	2019/11	2019	NOV	B
...	...	...	...	...	...	...	...	...	...	...
114843	NORTHEAST	MAYLAND HEIGHTS	Crime	Assault (Non-domestic)	1	5833	2012/06	2012	JUN	
114844	NORTHEAST	NORTH AIRWAYS	Disorder	Social Disorder	3	0	2019/06	2019	JUN	
114845	NORTHEAST	MCCALL	Crime	Theft FROM Vehicle	2	0	2019/06	2019	JUN	N F
114846	SOUTH	FAIRVIEW	Crime	Theft FROM Vehicle	3	3552	2012/04	2012	APR	
114847	WEST	SPRINGBANK HILL	Crime	Theft FROM Vehicle	3	8388	2012/01	2012	JAN	F

114848 rows × 11 columns

In [ ]:
```
calgary['Category'].value_counts()
```

Out[ ]:
```
Social Disorder                 23938
Theft FROM Vehicle              17729
Physical Disorder               16858
Theft OF Vehicle                13794
Residential Break & Enter       12408
Commercial Break & Enter        10339
Assault (Non-domestic)           9180
Violence Other (Non-domestic)    6180
Street Robbery                   2680
Commercial Robbery               1741
1320.131                            1
Name: Category, dtype: int64
```

In [ ]:
```
calgary_2016=calgary[calgary['Year']==2016]
```

In [ ]:
```
calgary_2016
```

Out[ ]:

	Sector	Community Name	Group Category	Category	Crime Count	Resident Count	Date	Year	Mo
2539	SOUTH	QUEENSLAND	Crime	Theft OF Vehicle	2	4823	2016/11	2016	N
2540	SOUTHEAST	OGDEN	Crime	Theft OF Vehicle	3	8714	2016/06	2016	J
2541	SOUTH	DEER RUN	Crime	Theft FROM Vehicle	9	5182	2016/11	2016	N
2543	WEST	NORTH GLENMORE PARK	Crime	Violence Other (Non-domestic)	1	2398	2016/12	2016	D
2544	CENTRE	HIGHLAND PARK	Disorder	Physical Disorder	1	4014	2016/08	2016	A
...	...	...	...	...	...	...	...	...	
79689	SOUTHEAST	MCKENZIE LAKE	Crime	Theft FROM Vehicle	10	14008	2016/03	2016	M
79691	CENTRE	KILLARNEY/GLENGARRY	Crime	Theft FROM Vehicle	3	7677	2016/03	2016	M
79695	NORTHEAST	DEERFOOT BUSINESS CENTRE	Disorder	Social Disorder	4	0	2016/01	2016	J
79701	NORTHEAST	SUNRIDGE	Disorder	Physical Disorder	10	86	2016/04	2016	A
79710	NORTHEAST	TARADALE	Crime	Violence Other (Non-domestic)	2	19223	2016/02	2016	

14727 rows × 11 columns

In [ ]:
```
calgary_2016.columns
```

Out[ ]:
```
Index(['Sector', 'Community Name', 'Group Category', 'Category', 'Crime Count',
       'Resident Count', 'Date', 'Year', 'Month', 'ID',
       'Community Center Point'],
      dtype='object')
```

In [ ]:
```
calgary_2016['Group Category'].value_counts()
```

Out[ ]:
```
Crime       9622
Disorder    5105
Name: Group Category, dtype: int64
```

In [ ]:
```
calgary_2016 = calgary_2016.loc[calgary_2016['Group Category'] == 'Crime']
```

In [ ]:
```
calgary_2016 = calgary_2016.rename(columns={'Community Name': 'NGH_Name', 'Category':
calgary_2016
```

Out[ ]:

	Sector	NGH_Name	Group Category	Violation Type	Crime Count	Resident Count	Date	Year	M
2539	SOUTH	QUEENSLAND	Crime	Theft OF Vehicle	2	4823	2016/11	2016	
2540	SOUTHEAST	OGDEN	Crime	Theft OF Vehicle	3	8714	2016/06	2016	
2541	SOUTH	DEER RUN	Crime	Theft FROM Vehicle	9	5182	2016/11	2016	
2543	WEST	NORTH GLENMORE PARK	Crime	Violence Other (Non-domestic)	1	2398	2016/12	2016	
2549	CENTRE	HIGHFIELD	Crime	Commercial Break & Enter	6	0	2016/09	2016	
...	...	...	...	...	...	...	...	...	...
79684	EAST	FOREST LAWN	Crime	Theft FROM Vehicle	6	8179	2016/02	2016	
79685	WEST	ROSSCARROCK	Crime	Violence Other (Non-domestic)	1	3447	2016/01	2016	
79689	SOUTHEAST	MCKENZIE LAKE	Crime	Theft FROM Vehicle	10	14008	2016/03	2016	
79691	CENTRE	KILLARNEY/GLENGARRY	Crime	Theft FROM Vehicle	3	7677	2016/03	2016	
79710	NORTHEAST	TARADALE	Crime	Violence Other (Non-domestic)	2	19223	2016/02	2016	

9622 rows × 11 columns

In [ ]:
```
calgary_2016.isnull().sum()
```

Out[ ]:
```
Sector                  0
NGH_Name                0
Group Category          0
Violation Type          0
Crime Count             0
Resident Count          0
Date                    0
Year                    0
Month                   0
ID                      0
Community Center Point  0
dtype: int64
```

In [ ]:
```
calgary_2016['Group Category'].value_counts()
```

Out[ ]:
```
Crime    9622
Name: Group Category, dtype: int64
```

In [ ]:
```
calgary_2016['Violation Type'].value_counts()
```

Out[ ]:
```
Theft FROM Vehicle              2358
Theft OF Vehicle                1845
Residential Break & Enter       1684
Commercial Break & Enter        1305
Assault (Non-domestic)          1147
Violence Other (Non-domestic)    767
Street Robbery                   307
Commercial Robbery               209
Name: Violation Type, dtype: int64
```

In [ ]:
```
# Create a dictionary to map the values
map_dict = {'Residential Break & Enter': 'Break & Enter', 'Commercial Break & Enter':
            'Street Robbery': 'Robbery', 'Commercial Robbery': 'Robbery'}

# Replace the values using the map_dict
calgary_2016['Violation Type'] = calgary_2016['Violation Type'].replace(map_dict)
calgary_2016['Violation Type'].value_counts()
```

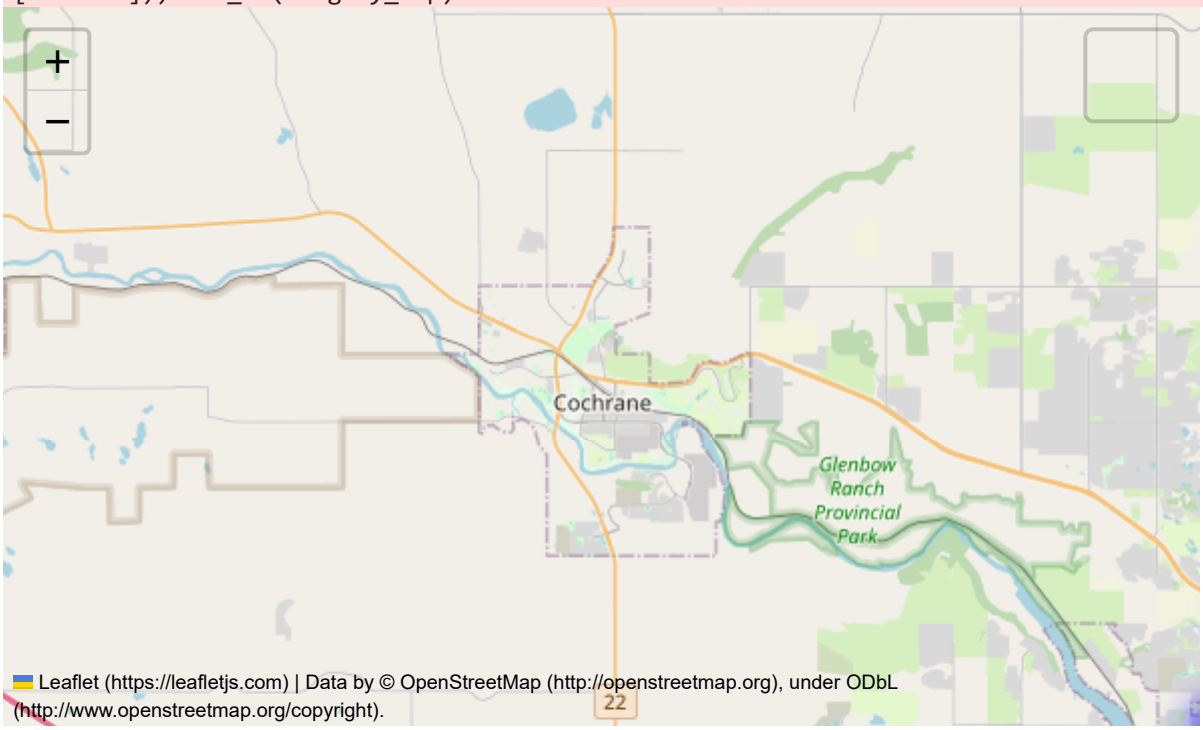
Out[ ]:
```
Break & Enter                   2989
Theft FROM Vehicle              2358
Theft OF Vehicle                1845
Assault (Non-domestic)          1147
Violence Other (Non-domestic)    767
Robbery                          516
Name: Violation Type, dtype: int64
```

In [ ]:
```
# Replace missing values with (0, 0) tuples
calgary_2016['Community Center Point'].fillna('(0, 0)', inplace=True)

# Split the 'Community Center Point' column into separate latitude and longitude colum
calgary_2016[['Latitude', 'Longitude']] = calgary_2016['Community Center Point'].str.r

# Create a map centered on the city of Calgary
calgary_map = folium.Map(location=[51.0447, -114.0719], zoom_start=11)

# Create a list of locations and violation counts
locations = calgary_2016[['Latitude', 'Longitude']].values.tolist()
violation_counts = calgary_2016.groupby(['Latitude', 'Longitude', 'Violation Type'])['
violation_counts = violation_counts.rename(columns={'ID': 'Count'})

# Create a heatmap layer based on the violation counts
```

```
heat_data = [[row['Latitude'], row['Longitude'], row['Count']] for _, row in violation
HeatMap(heat_data, name='Violation Count', radius=15, max_val=max(violation_counts['Co

# Add a layer control to the map
folium.LayerControl().add_to(calgary_map)

# Show the map
calgary_map
```

```
C:\Users\azimi\AppData\Local\Temp\ipykernel_14576\4134069381.py:5: FutureWarning: The
default value of regex will change from True to False in a future version. In additio
n, single character regular expressions will *not* be treated as literal strings when
regex=True.
  calgary_2016[['Latitude', 'Longitude']] = calgary_2016['Community Center Point'].st
r.replace('(', '').str.replace(')', '').str.split(',', expand=True).astype(float)
C:\Users\azimi\AppData\Local\Temp\ipykernel_14576\4134069381.py:17: UserWarning: The
`max_val` parameter is no longer necessary. The largest intensity is calculated autom
atically.
  HeatMap(heat_data, name='Violation Count', radius=15, max_val=max(violation_counts
['Count'])).add_to(calgary_map)
```

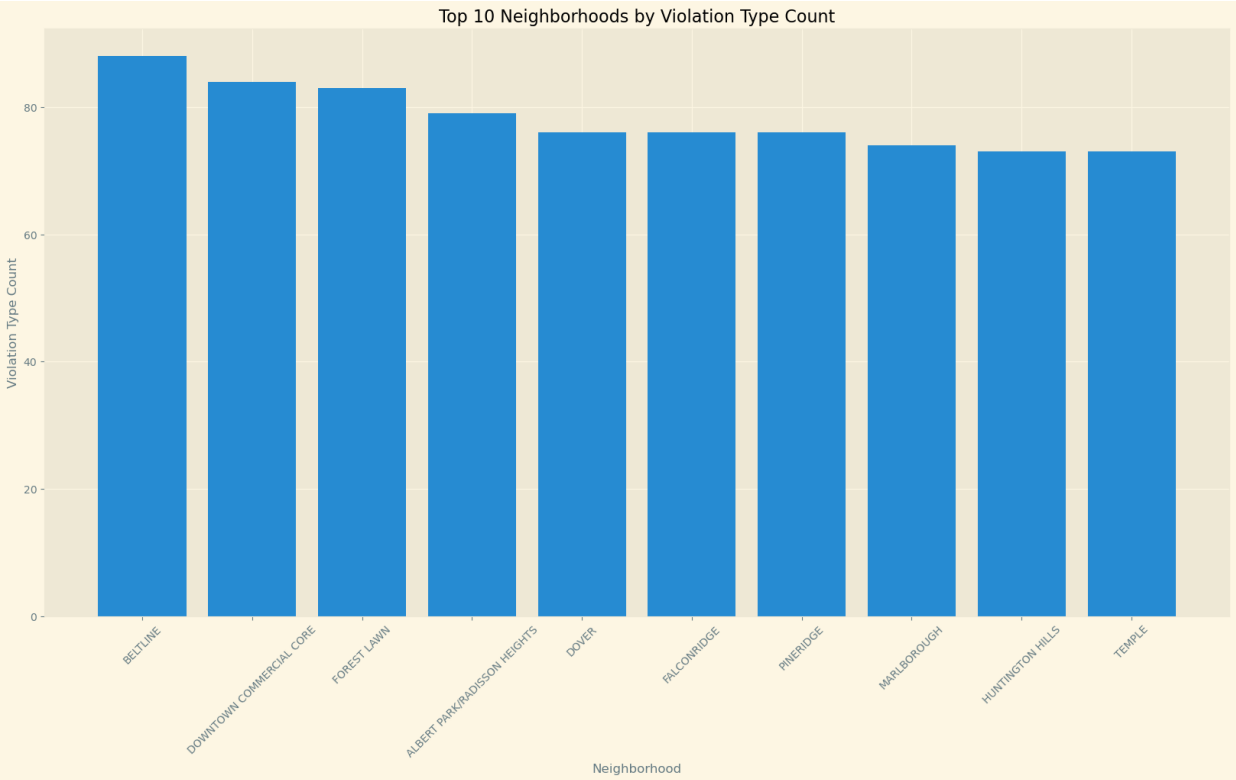
Out[ ]:



In [ ]:

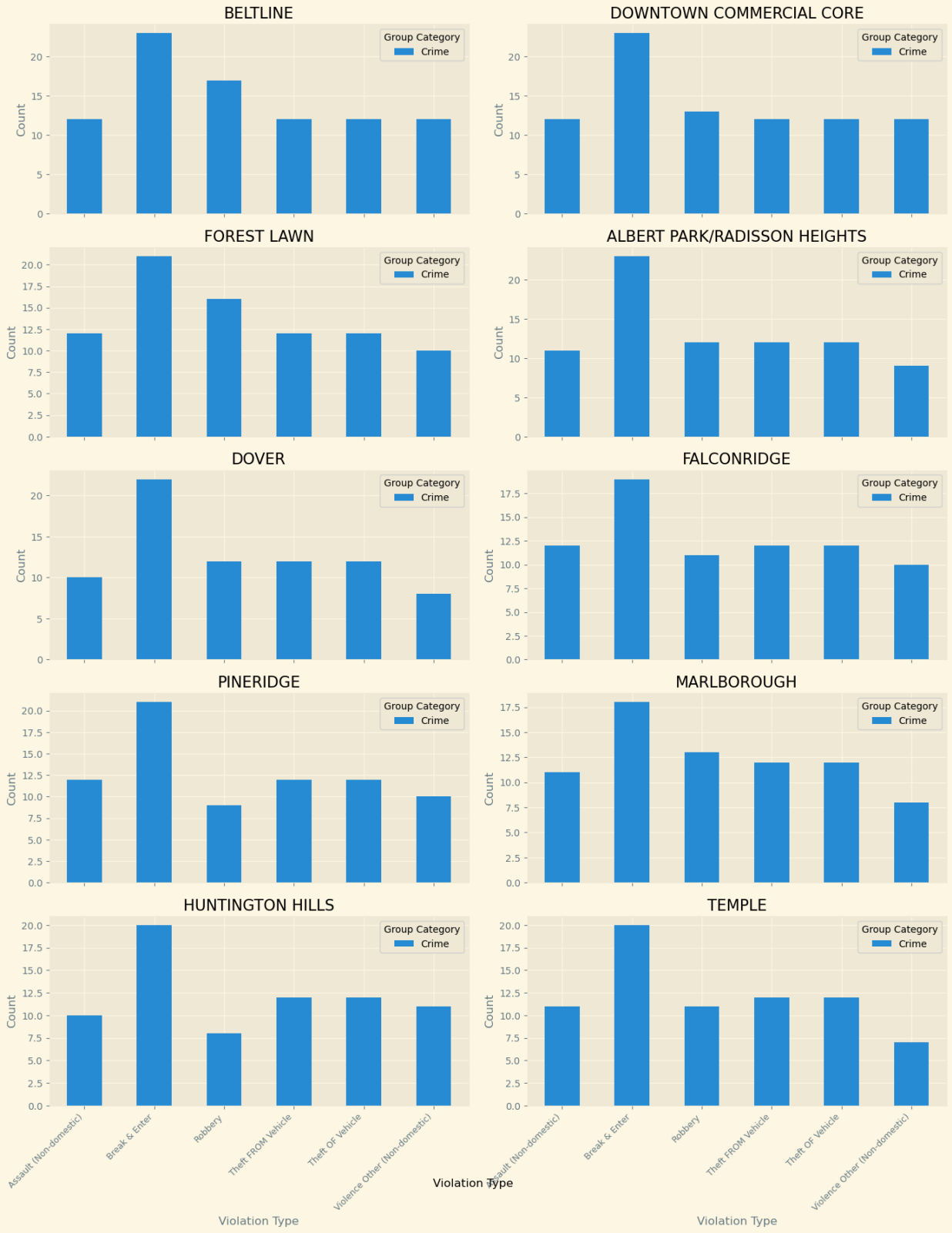
```
# Group the data by neighborhood and violation type, and get the count of each combina
grouped_data = calgary_2016.groupby(['NGH_Name', 'Violation Type'])['ID'].count()

# Get the total count of violations for each neighborhood
neighborhood_counts = grouped_data.groupby('NGH_Name').sum()

# Get the top 10 neighborhoods with the highest count of violations
top_10 = neighborhood_counts.nlargest(10)

# Plot a bar chart of the top 10 neighborhoods
plt.figure(figsize=(20, 10))  # set the figure size
plt.bar(top_10.index, top_10.values)
plt.title('Top 10 Neighborhoods by Violation Type Count')
plt.xlabel('Neighborhood')
plt.ylabel('Violation Type Count')
```

```
plt.xticks(rotation=45)  # rotate the x-axis labels by 45 degrees
plt.show()
```



In [ ]:

```
import matplotlib.pyplot as plt
import math

# Group the data by neighborhood and violation type, and aggregate the counts by group
grouped_data = calgary_2016.groupby(['NGH_Name', 'Violation Type', 'Group Category'])[

# Get the top 10 neighborhoods with the highest count of violation types
top_10 = grouped_data.groupby('NGH_Name').sum().nlargest(10)

# Create a 5x2 grid of subplots
fig, axs = plt.subplots(5, 2, figsize=(14, 18), sharex=True)

# Plot a stacked bar chart for each of the top 10 neighborhoods
for i, neighborhood in enumerate(top_10.index):
    row = math.floor(i / 2)
    col = i % 2
    ax = axs[row, col]
    neighborhood_data = grouped_data.loc[neighborhood].unstack().fillna(0)
    neighborhood_data.plot(kind='bar', stacked=True, title=neighborhood, ax=ax, rot=0)
    ax.set_ylabel('Count')
    ax.tick_params(axis='x', labelsize=9)
    plt.setp(ax.get_xticklabels(), rotation=45, ha='right')

# Set a common x-axis label
fig.text(0.5, 0.04, 'Violation Type', ha='center', fontsize=12)

# Adjust the spacing between subplots and remove extra whitespace
plt.tight_layout(pad=1)

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



Clustering for Calgary Crime

In [ ]:
```
calgary_2016
```

Out[ ]:

	Sector	NGH_Name	Group Category	Violation Type	Crime Count	Resident Count	Date	Year	Mo
2539	SOUTH	QUEENSLAND	Crime	Theft OF Vehicle	2	4823	2016/11	2016	N
2540	SOUTHEAST	OGDEN	Crime	Theft OF Vehicle	3	8714	2016/06	2016	J
2541	SOUTH	DEER RUN	Crime	Theft FROM Vehicle	9	5182	2016/11	2016	N
2543	WEST	NORTH GLENMORE PARK	Crime	Violence Other (Non-domestic)	1	2398	2016/12	2016	D
2549	CENTRE	HIGHFIELD	Crime	Break & Enter	6	0	2016/09	2016	
...	...	...	...	...	...	...	...	...	
79684	EAST	FOREST LAWN	Crime	Theft FROM Vehicle	6	8179	2016/02	2016	
79685	WEST	ROSSCARROCK	Crime	Violence Other (Non-domestic)	1	3447	2016/01	2016	J
79689	SOUTHEAST	MCKENZIE LAKE	Crime	Theft FROM Vehicle	10	14008	2016/03	2016	M
79691	CENTRE	KILLARNEY/GLENGARRY	Crime	Theft FROM Vehicle	3	7677	2016/03	2016	M
79710	NORTHEAST	TARADALE	Crime	Violence Other (Non-domestic)	2	19223	2016/02	2016	

9622 rows × 13 columns

In [ ]:
```
calgary_cluster=calgary_2016[['NGH_Name', 'Violation Type', 'Crime Count', 'Resident C
```

In [ ]:
```
calgary_cluster
```

Out[ ]:

	NGH_Name	Violation Type	Crime Count	Resident Count	Latitude	Longitude
2539	QUEENSLAND	Theft OF Vehicle	2	4823	50.938221	-114.022568
2540	OGDEN	Theft OF Vehicle	3	8714	50.992096	-114.012585
2541	DEER RUN	Theft FROM Vehicle	9	5182	50.925407	-114.009088
2543	NORTH GLENMORE PARK	Violence Other (Non-domestic)	1	2398	51.001422	-114.112247
2549	HIGHFIELD	Break & Enter	6	0	51.018295	-114.037201
...	...	...	...	...	...	...
79684	FOREST LAWN	Theft FROM Vehicle	6	8179	51.037848	-113.971230
79685	ROSSCARROCK	Violence Other (Non-domestic)	1	3447	51.043277	-114.145494
79689	MCKENZIE LAKE	Theft FROM Vehicle	10	14008	50.914881	-113.988181
79691	KILLARNEY/GLENGARRY	Theft FROM Vehicle	3	7677	51.030129	-114.131702
79710	TARADALE	Violence Other (Non-domestic)	2	19223	51.117940	-113.934002

9622 rows × 6 columns

In [ ]:
```
calgary_pivot = calgary_cluster.pivot_table(index='NGH_Name', values='Crime Count', co
```

In [ ]:
```
# Include the demographic columns in the pivot table
calgary_pivot = pd.concat([calgary_pivot, calgary_cluster.groupby('NGH_Name')[['Latitu
calgary_pivot
```

Out[ ]:

NGH_Name	Assault (Non-domestic)	Break & Enter	Robbery	Theft FROM Vehicle	Theft OF Vehicle	Violence Other (Non-domestic)	Latitude	Longitud
01B	0	1	0	1	0	0	51.102826	-114.24248
01C	0	1	0	0	0	0	51.085008	-114.23521
01F	0	0	0	1	0	0	51.117348	-114.26118
02A	29	5	0	1	1	5	51.168702	-114.22271
02E	11	2	0	2	2	1	51.161431	-114.19936
...	...	...	...	...	...	...	...	
WILLOW PARK	18	44	5	78	26	5	50.956619	-114.05620
WINDSOR PARK	3	40	0	49	16	2	51.005046	-114.08355
WINSTON HEIGHTS/MOUNTVIEW	12	41	1	53	30	6	51.075325	-114.04185
WOODBINE	5	28	3	51	17	1	50.939611	-114.12962
WOODLANDS	21	28	2	29	13	8	50.941120	-114.10634

264 rows × 8 columns

In [ ]:
```
calgary_pivot = calgary_pivot.reset_index()
calgary_pivot
```

Out[ ]:

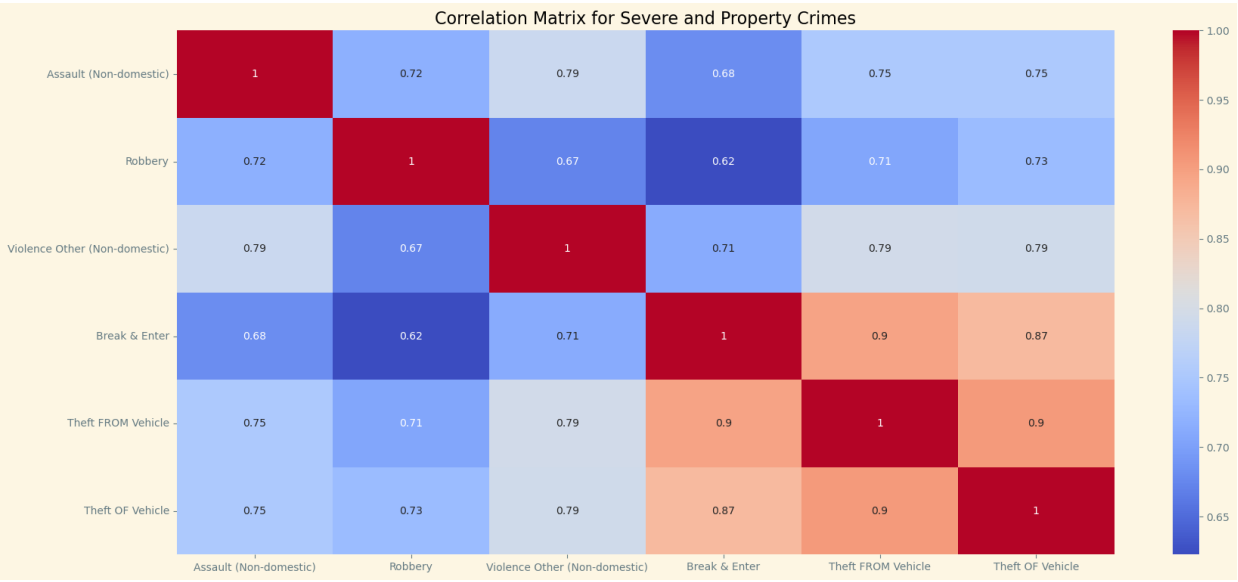
	NGH_Name	Assault (Non-domestic)	Break & Enter	Robbery	Theft FROM Vehicle	Theft OF Vehicle	Violence Other (Non-domestic)	Latitude	Long
0	01B	0	1	0	1	0	0	51.102826	-114.2
1	01C	0	1	0	0	0	0	51.085008	-114.2
2	01F	0	0	0	1	0	0	51.117348	-114.2
3	02A	29	5	0	1	1	5	51.168702	-114.2
4	02E	11	2	0	2	2	1	51.161431	-114.1
...	...	...	...	...	...	...	...	...	
259	WILLOW PARK	18	44	5	78	26	5	50.956619	-114.0
260	WINDSOR PARK	3	40	0	49	16	2	51.005046	-114.0
261	WINSTON HEIGHTS/MOUNTVIEW	12	41	1	53	30	6	51.075325	-114.0
262	WOODBINE	5	28	3	51	17	1	50.939611	-114.1
263	WOODLANDS	21	28	2	29	13	8	50.941120	-114.1

264 rows × 9 columns

In [ ]:
```
calgary_pivot.columns
```

Out[ ]:
```
Index(['NGH_Name', 'Assault (Non-domestic)', 'Break & Enter', 'Robbery',
       'Theft FROM Vehicle', 'Theft OF Vehicle',
       'Violence Other (Non-domestic)', 'Latitude', 'Longitude'],
      dtype='object')
```

In [ ]:
```
# Define the two crime groups
Violent_Crime = ['Assault (Non-domestic)', 'Robbery', 'Violence Other (Non-domestic)']
Property_Crime = ['Break & Enter', 'Theft FROM Vehicle', 'Theft OF Vehicle']
plot_crime_correlations(calgary_pivot, Violent_Crime, Property_Crime)
```



In [ ]:
```
calgary_pivot.iloc[:,1:]
```

Out[ ]:

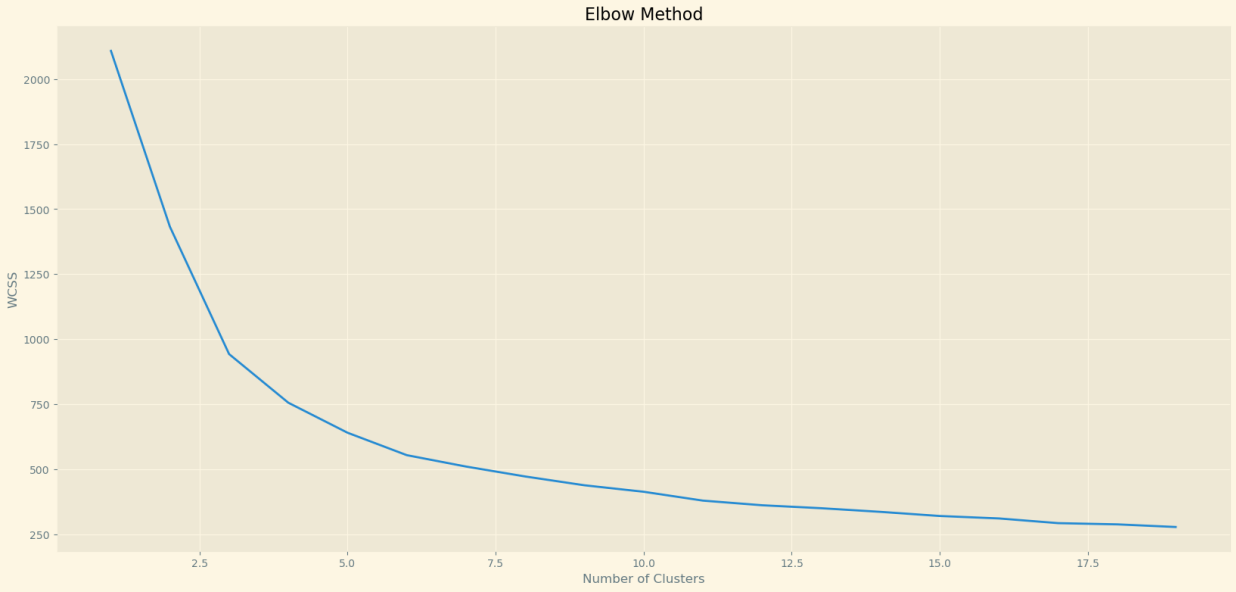
	Assault (Non-domestic)	Break & Enter	Robbery	Theft FROM Vehicle	Theft OF Vehicle	Violence Other (Non-domestic)	Latitude	Longitude
0	0	1	0	1	0	0	51.102826	-114.242480
1	0	1	0	0	0	0	51.085008	-114.235211
2	0	0	0	1	0	0	51.117348	-114.261186
3	29	5	0	1	1	5	51.168702	-114.222716
4	11	2	0	2	2	1	51.161431	-114.199362
...	...	...	...	...	...	...	...	...
259	18	44	5	78	26	5	50.956619	-114.056202
260	3	40	0	49	16	2	51.005046	-114.083552
261	12	41	1	53	30	6	51.075325	-114.041855
262	5	28	3	51	17	1	50.939611	-114.129629
263	21	28	2	29	13	8	50.941120	-114.106344

264 rows × 8 columns

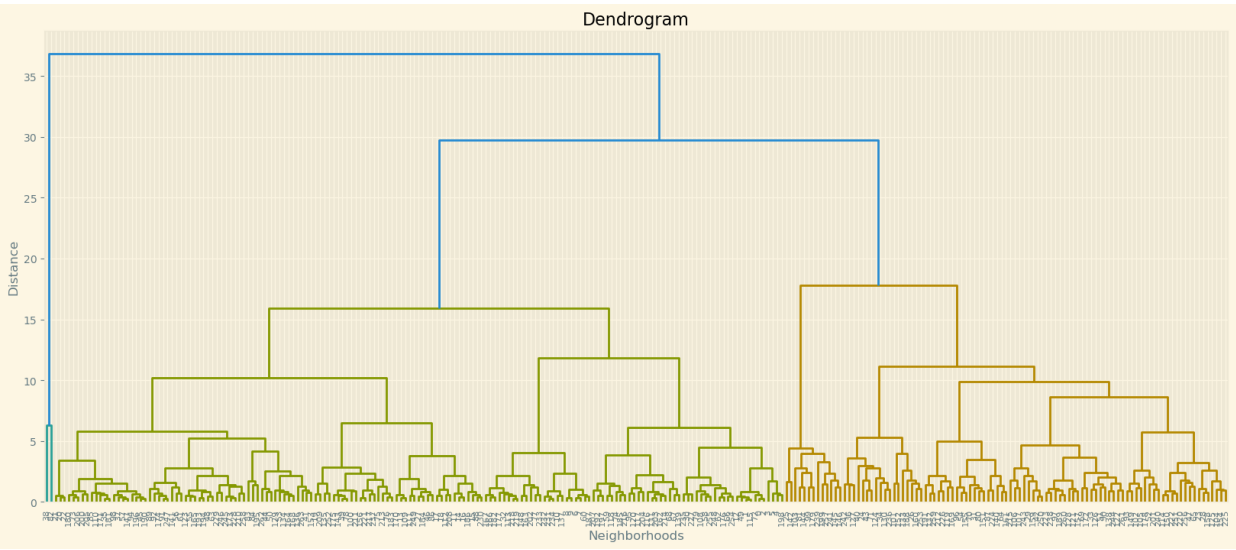
In [ ]:
```
# Standardize the data
scaler = StandardScaler()
calgary_scaled = scaler.fit_transform(calgary_pivot.iloc[:,1:])
```

In [ ]:
```
# Find the optimal number of clusters using the elbow method
wcss = []
for i in range(1, 20):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_st
    kmeans.fit(calgary_scaled)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 20), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```

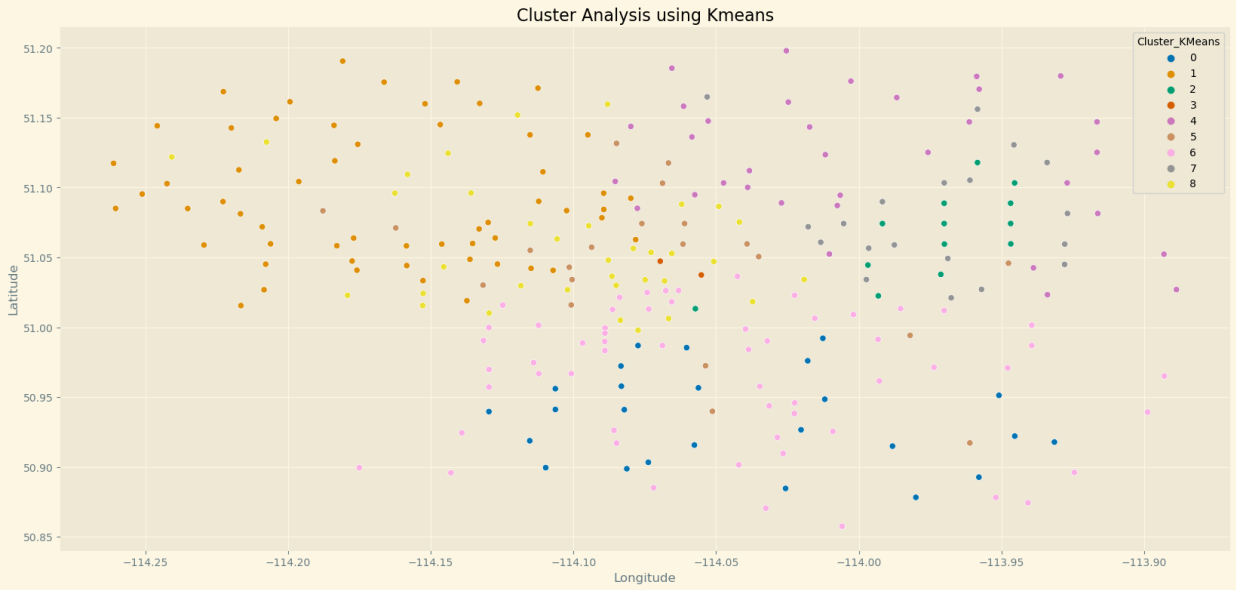
```
c:\Users\azimi\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1036: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less ch
unks than available threads. You can avoid it by setting the environment variable OMP
_NUM_THREADS=2.
  warnings.warn(
```



In [ ]:
```
# Generate the linkage matrix using complete linkage
linkage_matrix = shc.linkage(calgary_scaled, method='ward')

# Plot the dendrogram
plt.figure(figsize=(20, 8))
plt.title('Dendrogram')
plt.xlabel('Neighborhoods')
plt.ylabel('Distance')
shc.dendrogram(linkage_matrix, leaf_rotation=90., leaf_font_size=8.)
plt.show()
```



In [ ]:
```
# Create the KMeans model and fit to the standardized data
kmeans = KMeans(n_clusters=9, init='k-means++', max_iter=1100, n_init=10, random_state
pred_y = kmeans.fit_predict(calgary_scaled)

# Add the cluster labels to the original dataframe
calgary_pivot['Cluster_KMeans'] = pred_y

# Plot the clusters
sns.scatterplot(x='Longitude', y='Latitude', data=calgary_pivot, hue='Cluster_KMeans',
#sns.scatterplot(x='Latitude', y='Longitude', data=df, hue='Cluster', palette='colorbl
plt.title('Cluster Analysis using Kmeans')
plt.show()
```



In [ ]:
```
calgary_pivot['Cluster_KMeans'].value_counts()
```

Out[ ]:
```
6    58
1    57
8    35
4    34
0    25
5    21
7    19
2    13
3     2
Name: Cluster_KMeans, dtype: int64
```

In [ ]:
```
Ccluster_0=calgary_pivot[calgary_pivot['Cluster_KMeans']==0]
Ccluster_1=calgary_pivot[calgary_pivot['Cluster_KMeans']==1]
Ccluster_2=calgary_pivot[calgary_pivot['Cluster_KMeans']==2]
Ccluster_3=calgary_pivot[calgary_pivot['Cluster_KMeans']==3]
```

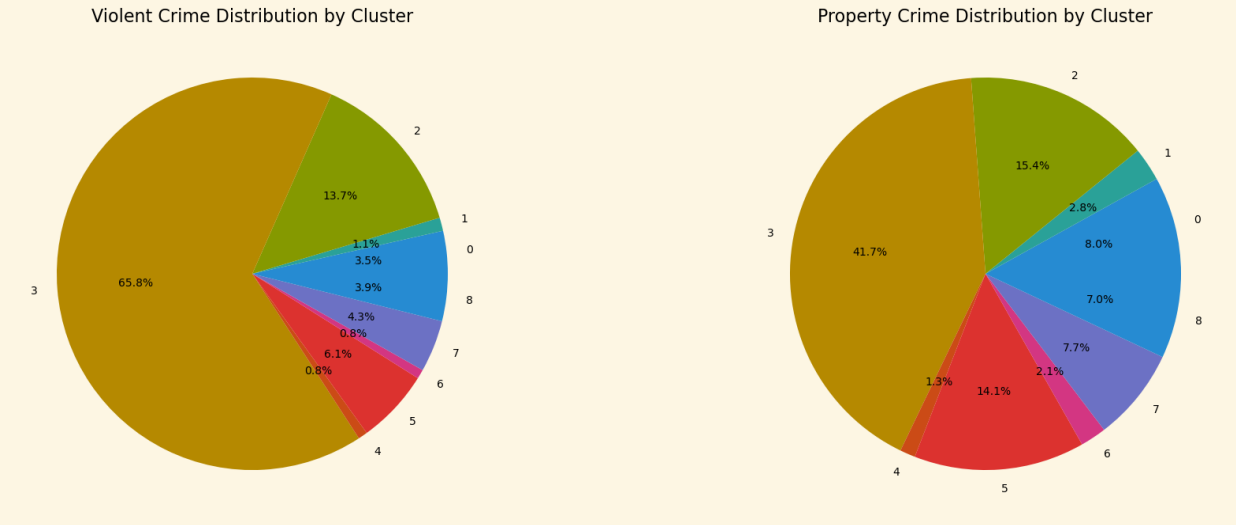
In [ ]:
```
Ccluster_2['NGH_Name']
```

Out[ ]:
```
25     ALBERT PARK/RADISSON HEIGHTS
81                           DOVER
99                       FALCONRIDGE
103                      FOREST LAWN
142           MANCHESTER INDUSTRIAL
144                     MARLBOROUGH
145                MARLBOROUGH PARK
146                      MARTINDALE
179                       PINERIDGE
199                          RUNDLE
239                        SUNRIDGE
241                          TEMPLE
257                        WHITEHORN
Name: NGH_Name, dtype: object
```

In [ ]:
```
Ccluster_3['NGH_Name']
```

Out[ ]:
```
38                     BELTLINE
82    DOWNTOWN COMMERCIAL CORE
Name: NGH_Name, dtype: object
```

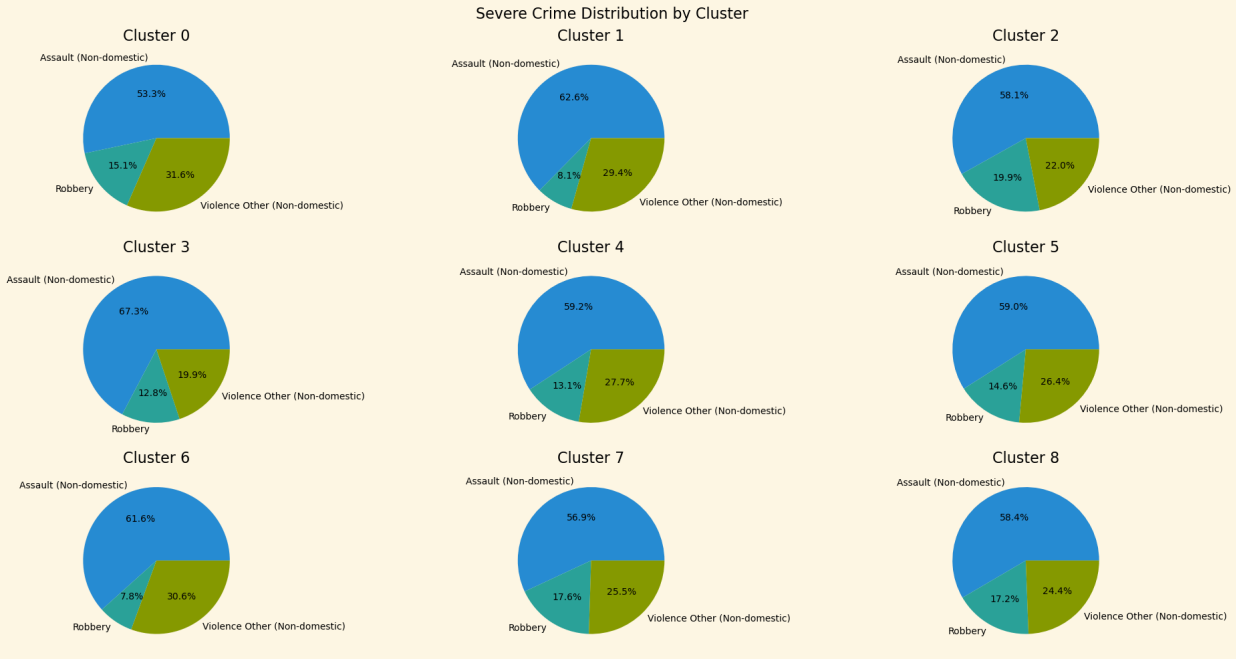
In [ ]:
```
# Define the two crime groups
Violent_Crime = ['Assault (Non-domestic)', 'Robbery', 'Violence Other (Non-domestic)']
Property_Crime = ['Break & Enter', 'Theft FROM Vehicle', 'Theft OF Vehicle']

# Calculate the total count of crimes in each cluster for both groups
crime_counts_violent = calgary_pivot.groupby('Cluster_KMeans')[Violent_Crime].mean().s
crime_counts_property = calgary_pivot.groupby('Cluster_KMeans')[Property_Crime].mean()

# Create a 1x2 grid of subplots
plt.subplot(1, 2, 1)
# Create a pie chart for violent crime
plt.pie(crime_counts_violent, labels=crime_counts_violent.index, autopct='%1.1f%%')
plt.title('Violent Crime Distribution by Cluster')

plt.subplot(1, 2, 2)
# Create a pie chart for property crime
plt.pie(crime_counts_property, labels=crime_counts_property.index, autopct='%1.1f%%')
plt.title('Property Crime Distribution by Cluster')

# Adjust spacing between subplots
plt.subplots_adjust(wspace=0.5)

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



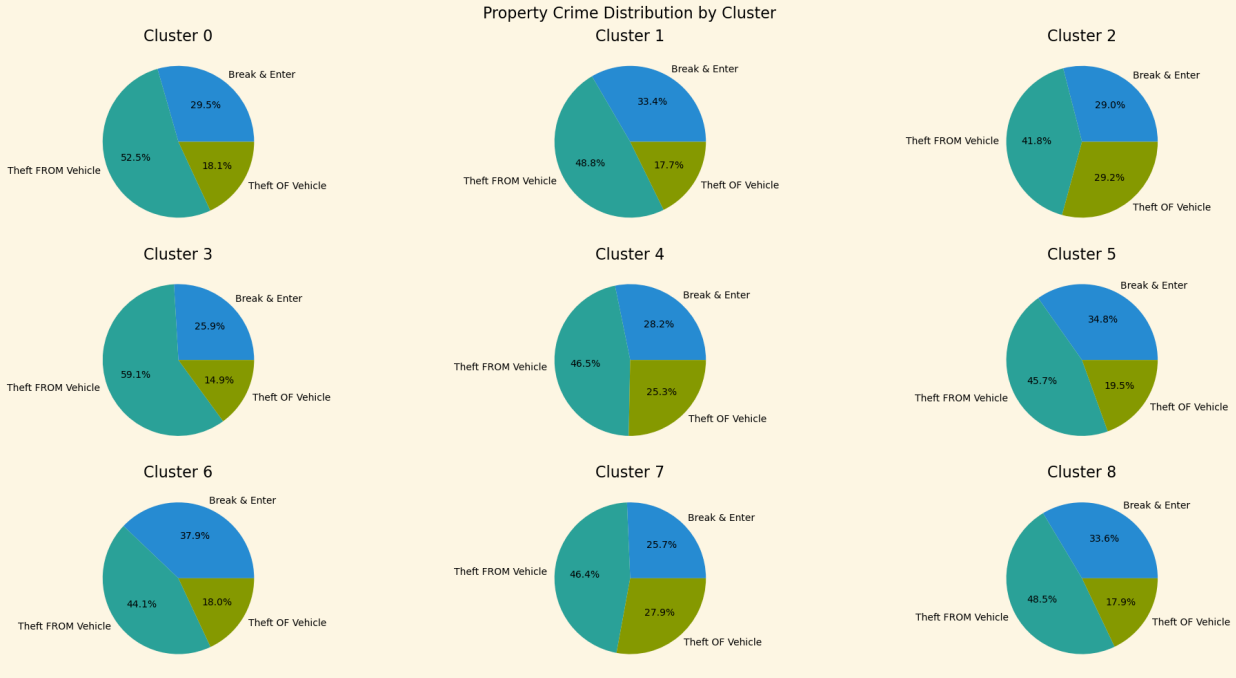
In [ ]:
```
"""Plotting Pie charts for the distribution as a % of each grouped crime by Cluster"""
def plot_violation_type_distribution_Calgary(data, violation_group):
    # Define the two violation groups
    severe_crime = ['Assault (Non-domestic)', 'Robbery', 'Violence Other (Non-domestic
    property_crime = ['Break & Enter', 'Theft FROM Vehicle', 'Theft OF Vehicle']

    # Select the appropriate violation group
    if violation_group == 'Severe Crime':
        violation_types = severe_crime
    elif violation_group == 'Property Crime':
        violation_types = property_crime
    else:
        raise ValueError("Violation group must be 'Severe Crime' or 'Property Crime'."

    # Calculate the total count of each violation type in each cluster
    cluster_counts = data.groupby('Cluster_KMeans')[violation_types].mean()
```

```
    # Create a 2x2 grid of subplots for each cluster
    fig, axs = plt.subplots(nrows=3, ncols=3, figsize=(20,10))

    # Loop over each cluster and create a pie chart of the violation type distributior
    for i, ax in enumerate(axs.flat):
        # Get the counts for the current cluster and convert to percentages
        counts = cluster_counts.iloc[i]
        total = counts.sum()
        percents = counts / total * 100

        # Create the pie chart
        labels = percents.index
        sizes = percents.values
        ax.pie(sizes, labels=labels, autopct='%1.1f%%')
        ax.set_title(f'Cluster {i}')

    # Add a common title for the whole figure
    fig.suptitle(f'{violation_group} Distribution by Cluster', fontsize=16)

    # Set the layout and show the plot
    plt.tight_layout()
    plt.show()
```

In [ ]:
```
plot_violation_type_distribution_Calgary(calgary_pivot, 'Severe Crime')
```



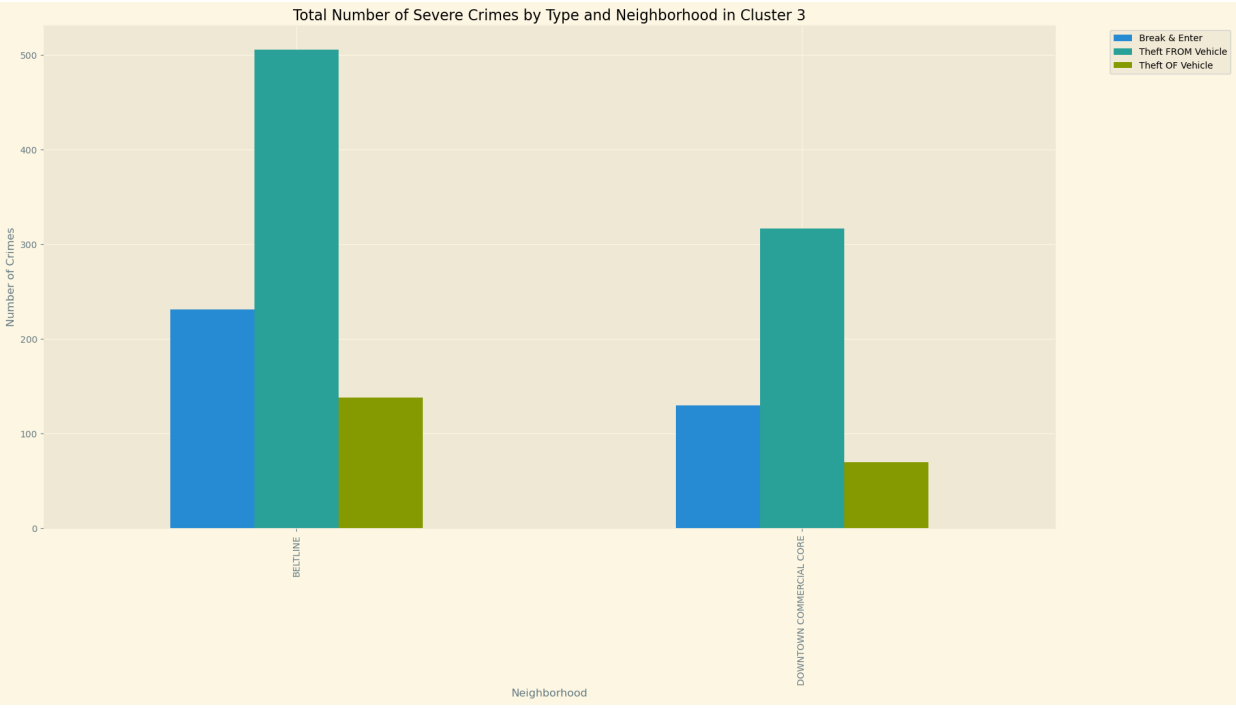
In [ ]:
```
plot_violation_type_distribution_Calgary(calgary_pivot, 'Property Crime')
```



In [ ]:

```
"""This code will create a bar chart that shows the total number of crimes for each t
    The x-axis shows the neighborhoods, and the y-axis shows the total number of crime
    Each color in the bar chart represents a different type of crime.
    You can use this visualization to see which neighborhoods have the highest crime r
    which types of crimes are most prevalent in each neighborhood."""

# Create a new dataframe for cluster 2
cluster_2_df = calgary_pivot[calgary_pivot['Cluster_KMeans'] == 3]

# Group the data by neighborhood and calculate the total number of crimes for each typ
crime_totals = cluster_2_df.groupby('NGH_Name')[['Assault (Non-domestic)', 'Robbery',

# Create a bar chart showing the total number of crimes for each type of crime in each
crime_totals.plot(kind='bar', figsize=(20,10))
plt.title('Total Number of Severe Crimes by Type and Neighborhood in Cluster 3')
plt.xlabel('Neighborhood')
plt.ylabel('Number of Crimes')
plt.xticks(np.arange(len(crime_totals.index)), crime_totals.index, rotation=90)
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```

In [ ]:
```
# Create a new dataframe for cluster 2
cluster_2_df = calgary_pivot[calgary_pivot['Cluster_KMeans'] == 3]

# Group the data by neighborhood and calculate the total number of crimes for each typ
crime_totals = cluster_2_df.groupby('NGH_Name')[['Break & Enter', 'Theft FROM Vehicle'

# Create a bar chart showing the total number of crimes for each type of crime in each
crime_totals.plot(kind='bar', figsize=(20,10))
plt.title('Total Number of Severe Crimes by Type and Neighborhood in Cluster 3')
plt.xlabel('Neighborhood')
plt.ylabel('Number of Crimes')
plt.xticks(np.arange(len(crime_totals.index)), crime_totals.index, rotation=90)
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```



In [ ]:
```
calgary_pivot.columns
```

Out[ ]:
```
Index(['NGH_Name', 'Assault (Non-domestic)', 'Break & Enter', 'Robbery',
       'Theft FROM Vehicle', 'Theft OF Vehicle',
       'Violence Other (Non-domestic)', 'Latitude', 'Longitude',
       'Cluster_KMeans'],
      dtype='object')
```

This code is grouping the calgary_pivot DataFrame by the Cluster_KMeans column (which is the column that identifies the cluster to which each neighborhood belongs). Then, it is selecting a subset of columns that correspond to the different types of crimes that are recorded in the dataset, and computing the mean of each of these columns for each cluster. The columns that are selected are:

Assault (Non-domestic) Break & Enter Robbery Theft FROM Vehicle Theft OF Vehicle Violence Other (Non-domestic) This code is essentially summarizing the crime data for each cluster in terms of the average number of incidents of each type of crime that occur in the neighborhoods that belong to that cluster.

Averages of the Crimes Commited by Clusters in Calgary

In [ ]:
```
calgary_pivot.groupby('Cluster_KMeans')['Assault (Non-domestic)', 'Break & Enter', 'Ro
       'Theft FROM Vehicle', 'Theft OF Vehicle',
       'Violence Other (Non-domestic)'].mean()
```

```
C:\Users\azimi\AppData\Local\Temp\ipykernel_14576\2909496485.py:1: FutureWarning: Ind
exing with multiple keys (implicitly converted to a tuple of keys) will be deprecate
d, use a list instead.
  calgary_pivot.groupby('Cluster_KMeans')['Assault (Non-domestic)', 'Break & Enter',
'Robbery',
```

Out[ ]:

Cluster_KMeans	Assault (Non-domestic)	Break & Enter	Robbery	Theft FROM Vehicle	Theft OF Vehicle	Violence Other (Non-domestic)
0	9.320000	39.520000	2.640000	70.400000	24.240000	5.520000
1	3.403509	15.421053	0.438596	22.526316	8.175439	1.596491
2	39.692308	74.538462	13.615385	107.307692	75.153846	15.000000
3	220.000000	180.500000	42.000000	411.500000	104.000000	65.000000
4	2.264706	6.000000	0.500000	9.882353	5.382353	1.058824
5	17.952381	82.095238	4.428571	107.714286	45.904762	8.047619
6	2.327586	13.603448	0.293103	15.810345	6.465517	1.155172
7	12.105263	32.842105	3.736842	59.368421	35.684211	5.421053
8	11.285714	39.142857	3.314286	56.485714	20.857143	4.714286

Lets view the table by drilling down to the cluster 1

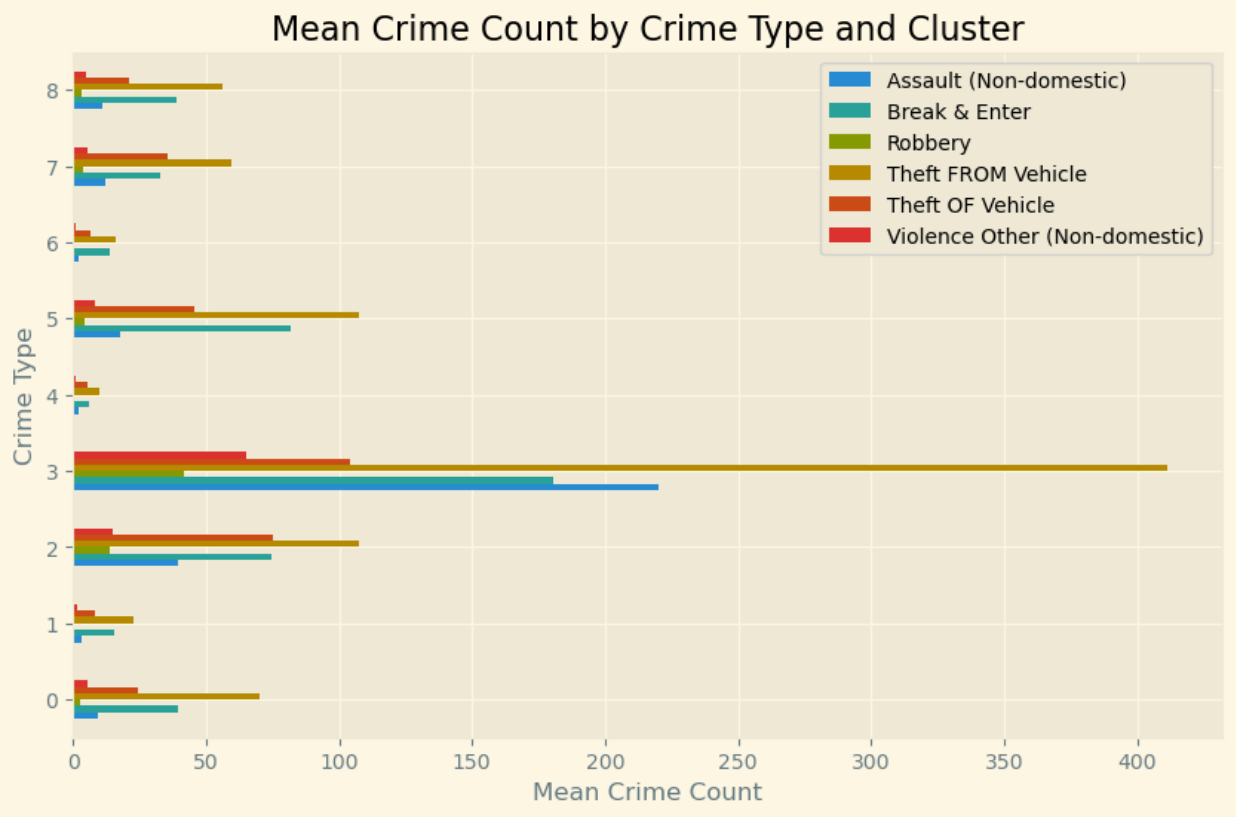
In [ ]:
```
cluster_num = 3
cluster_df = calgary_pivot[calgary_pivot['Cluster_KMeans'] == cluster_num]
ngh_grouped = cluster_df.groupby('NGH_Name')[['Assault (Non-domestic)', 'Break & Enter
ngh_grouped
```

Out[ ]:

NGH_Name	Assault (Non-domestic)	Break & Enter	Robbery	Theft FROM Vehicle	Theft OF Vehicle	Violence Other (Non-domestic)
BELTLINE	234.0	231.0	43.0	506.0	138.0	69.0
DOWNTOWN COMMERCIAL CORE	206.0	130.0	41.0	317.0	70.0	61.0

In [ ]:
```
# Get the mean crime count for each crime type for each cluster
crime_means_calgary = calgary_pivot.groupby('Cluster_KMeans')['Assault (Non-domestic)'
       'Theft FROM Vehicle', 'Theft OF Vehicle',
       'Violence Other (Non-domestic)'].mean()

# Create a horizontal bar chart
fig, ax = plt.subplots(figsize=(10, 6))
crime_means_calgary.plot(kind='barh', ax=ax)

# Set chart title and axis labels
ax.set_title('Mean Crime Count by Crime Type and Cluster')
ax.set_xlabel('Mean Crime Count')
ax.set_ylabel('Crime Type')

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

```
C:\Users\azimi\AppData\Local\Temp\ipykernel_14576\3195579139.py:2: FutureWarning: Ind
exing with multiple keys (implicitly converted to a tuple of keys) will be deprecate
d, use a list instead.
  crime_means_calgary = calgary_pivot.groupby('Cluster_KMeans')['Assault (Non-domesti
c)', 'Break & Enter', 'Robbery',
```



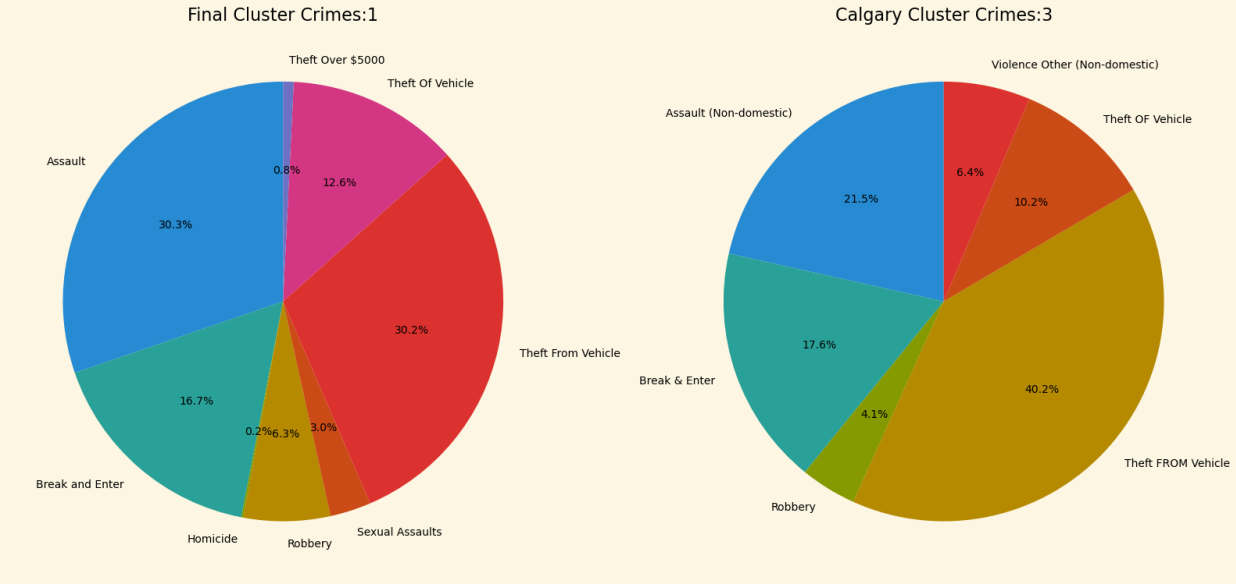
In [ ]:
```
calgary_pivot.groupby('Cluster_KMeans')['NGH_Name'].count()
```

Out[ ]:
```
Cluster_KMeans
0    25
1    57
2    13
3     2
4    34
5    21
6    58
7    19
8    35
Name: NGH_Name, dtype: int64
```

Comparison between Edmonton HOT cluster and Calgary HOT cluster

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

# Subset the data to the two clusters of interest
final_cluster = final[final['Cluster_KMeans'] == 1]
calgary_cluster = calgary_pivot[calgary_pivot['Cluster_KMeans'] == 3]

# Define the crime categories to plot
final_crime_cols = ['Assault', 'Break and Enter', 'Homicide', 'Robbery', 'Sexual Assau
calgary_crime_cols = ['Assault (Non-domestic)', 'Break & Enter', 'Robbery', 'Theft FRO

# Calculate the crime totals for each cluster
final_totals = final_cluster[final_crime_cols].mean()
calgary_totals = calgary_cluster[calgary_crime_cols].mean()

# Create a figure with two subplots for the pie charts
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 10))
```

```
# Plot the first pie chart for the final cluster
ax1.pie(final_totals, labels=final_crime_cols, autopct='%1.1f%%', startangle=90)
ax1.set_title('Final Cluster Crimes:1')

# Plot the second pie chart for the Calgary cluster
ax2.pie(calgary_totals, labels=calgary_crime_cols, autopct='%1.1f%%', startangle=90)
ax2.set_title('Calgary Cluster Crimes:3')

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



One final look at the Edmonton Cluster_1 (High Crime) and Calgary Cluster_3(High Crime)

In [ ]:
```
# City of Edmonton HOT Crime Cluster.
cluster_1['NGH_Name']
```

Out[ ]:
```
2          ALBERTA AVENUE
32           BOYLE STREET
53      CENTRAL MCDOUGALL
69             DOWNTOWN
88              GARNEAU
113           INGLEWOOD
165             OLIVER
182       QUEEN MARY PARK
211           STRATHCONA
Name: NGH_Name, dtype: object
```

In [ ]:
```
# City of Calgary Hot Crime Cluster.
Ccluster_3['NGH_Name']
```

Out[ ]:
```
38                    BELTLINE
82    DOWNTOWN COMMERCIAL CORE
Name: NGH_Name, dtype: object
```

In [ ]:
```
calgary_pivot
```

Out[ ]:

	NGH_Name	Assault (Non-domestic)	Break & Enter	Robbery	Theft FROM Vehicle	Theft OF Vehicle	Violence Other (Non-domestic)	Latitude	Long
0	01B	0	1	0	1	0	0	51.102826	-114.2
1	01C	0	1	0	0	0	0	51.085008	-114.2
2	01F	0	0	0	1	0	0	51.117348	-114.2
3	02A	29	5	0	1	1	5	51.168702	-114.2
4	02E	11	2	0	2	2	1	51.161431	-114.1
...	...	...	...	...	...	...	...	...	
259	WILLOW PARK	18	44	5	78	26	5	50.956619	-114.0
260	WINDSOR PARK	3	40	0	49	16	2	51.005046	-114.0
261	WINSTON HEIGHTS/MOUNTVIEW	12	41	1	53	30	6	51.075325	-114.0
262	WOODBINE	5	28	3	51	17	1	50.939611	-114.1
263	WOODLANDS	21	28	2	29	13	8	50.941120	-114.1

264 rows × 10 columns

In [ ]:
```
final
```

Out[ ]:

	NGH_Name	Assault	Break and Enter	Homicide	Robbery	Sexual Assaults	Theft From Vehicle	Theft Of Vehicle	Theft Over $5000	Latitude
0	ABBOTTSFIELD	35	8	0	8	4	16	6	0	53.574143
1	ALBANY	8	8	0	2	2	9	2	1	53.632382
2	ALBERTA AVENUE	123	119	1	26	13	156	99	2	53.568485
3	ALDERGROVE	17	19	0	6	4	46	23	0	53.516888
4	ALLARD	6	17	0	1	0	12	3	4	53.401301
...	...	...	...	...	...	...	...	...	...	...
238	WESTWOOD	43	34	0	7	6	53	48	1	53.575942
239	WILD ROSE	14	15	0	2	2	46	24	1	53.470564
240	WINDERMERE	7	22	0	3	1	31	6	4	53.432563
241	WOODCROFT	30	22	0	13	4	40	20	4	53.564595
242	YORK	24	21	0	4	2	48	17	4	53.602843

243 rows × 34 columns

In [ ]:
```
calgary_pivot.to_csv('Calgary_with_Cluster.csv')
```

In [ ]:
```
final.to_csv('Edmonton_with_Cluster.csv')
```