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
```

1998-2021: Canada Crime Index

Investigate what is the Canadian Crime Index

In [ ]:
```
df_index=pd.read_csv("Crime Index Canada.csv")
# Deleted the following columns as we do not need them.
df_index.drop(['UOM', 'UOM_ID', 'DGUID',
       'SCALAR_FACTOR', 'SCALAR_ID', 'VECTOR', 'COORDINATE', 'STATUS',
       'SYMBOL', 'TERMINATED', 'DECIMALS'], axis=1, inplace =True)

# extract the 'GEO' name and number in separate columns
df_index[['GEO_NAME', 'GEO_NUMBER']] = df_index['GEO'].str.extract(r'([^\[\]]+) \[(\d+

# drop the original 'GEO' column
df_index.drop('GEO', axis=1, inplace=True)
df_index.drop('GEO_NUMBER', axis=1, inplace=True)

# drop rows where 'GEO_NAME' is null
df_index.dropna(subset=['GEO_NAME'], inplace=True)

# print the updated dataframe
df_index
```

Out[ ]:

	REF_DATE	Statistics	VALUE	GEO_NAME
10	1998	Crime severity index	76.38	Newfoundland and Labrador
11	1998	Violent crime severity index	58.43	Newfoundland and Labrador
12	1998	Non-violent crime severity index	83.28	Newfoundland and Labrador
13	1998	Youth crime severity index	109.72	Newfoundland and Labrador
14	1998	Youth violent crime severity index	55.73	Newfoundland and Labrador
...	...	...	...	...
19231	2021	Percent change in weighted clearance rate	-8.15	Nunavut
19232	2021	Violent weighted clearance rate	74.56	Nunavut
19233	2021	Percent change in violent weighted clearance rate	-0.48	Nunavut
19234	2021	Non-violent weighted clearance rate	46.66	Nunavut
19235	2021	Percent change in non-violent weighted clearan...	-12.62	Nunavut

18409 rows × 4 columns

In [ ]:
```
# convert 'REF_DATE' column to datetime
df_index['REF_DATE'] = pd.to_datetime(df_index['REF_DATE'], format='%Y')
```

In [ ]:
```
df_index.columns
```

Out[ ]:
```
Index(['REF_DATE', 'Statistics', 'VALUE', 'GEO_NAME'], dtype='object')
```

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 18409 entries, 10 to 19235
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   REF_DATE    18409 non-null  datetime64[ns]
 1   Statistics  18409 non-null  object
 2   VALUE       15249 non-null  float64
 3   GEO_NAME    18409 non-null  object
dtypes: datetime64[ns](1), float64(1), object(2)
memory usage: 719.1+ KB
```

In [ ]:
```
df_index['Statistics'].value_counts()
```

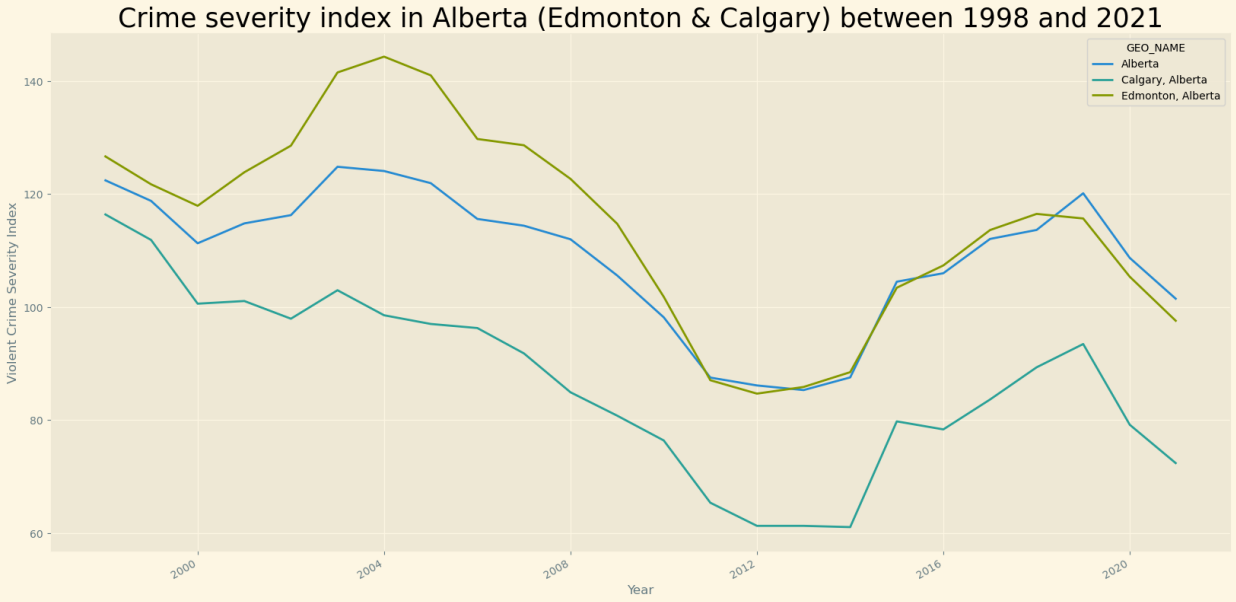
Out[ ]:
```
Crime severity index                                          1071
Weighted clearance rate                                       1071
Violent crime severity index                                  1071
Non-violent weighted clearance rate                           1071
Violent weighted clearance rate                               1071
Percent change in non-violent weighted clearance rate         1071
Youth non-violent crime severity index                        1071
Youth violent crime severity index                            1071
Youth crime severity index                                    1071
Non-violent crime severity index                              1071
Percent change in crime severity index                        1031
Percent change in violent crime severity index                1031
Percent change in non-violent crime severity index            1031
Percent change in weighted clearance rate                     1031
Percent change in violent weighted clearance rate             1031
Percent change in youth crime severity index                   848
Percent change in youth violent crime severity index           848
Percent change in youth non-violent crime severity index       848
Name: Statistics, dtype: int64
```

In [ ]:
```
# drop any rows with missing values
df_index.dropna(inplace=True)
```

We are only interested in Alberta- lets look at the Crime Index for the two Major Cities, Edmonton & Calgary, compared with the overall Alberta index.

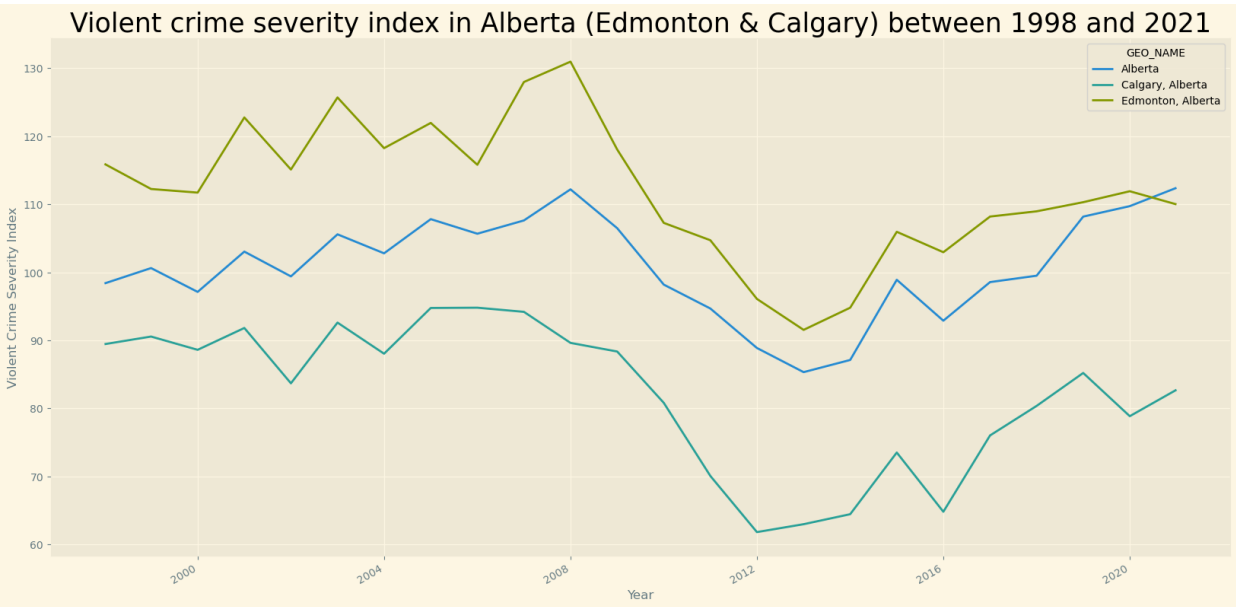
In [ ]:
```
""" Funciton to plot line diagrams over time for Arguments, data, Index Type and the C
def Plot_Crime_Index(data, statistic, *Cities):
    # filter the dataframe to include only Edmonton, Calgary, and Alberta
    data = data[data['GEO_NAME'].isin(Cities)]

    # filter the dataframe to include only 'Violent crime severity index' statistics
    data = data[data['Statistics'] == statistic]

    # filter the dataframe to include only years between 1998 and 2021
    data = data[(data['REF_DATE'] >= '1998-01-01') & (data['REF_DATE'] <= '2021-01-30'

    # pivot the dataframe to have 'GEO_NAME' as columns and 'REF_DATE' as index
    data_pivot = data.pivot(index='REF_DATE', columns='GEO_NAME', values='VALUE')

    # create the line plot with three separate lines
    ax = data_pivot.plot(kind='line', figsize=(20,10), x_compat=True)

    # set the plot title and axes labels
    ax.set_title(f'{statistic} in Alberta (Edmonton & Calgary) between 1998 and 2021',
    ax.set_xlabel('Year')
    ax.set_ylabel('Violent Crime Severity Index')

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

In [ ]:
```
# Over all Crime Severity Index.
cities=['Edmonton, Alberta', 'Calgary, Alberta', 'Alberta']
Plot_Crime_Index(df_index, 'Crime severity index', *cities)
```



> Looking at the Over Crime Severity Index, we see that Edmonton's index (after 2010) was the same as the overall Alberta Index. However, Calgary's index was much lower for all the years.

In [ ]:
```
# Violent Crime Severity Index
Plot_Crime_Index(df_index, 'Violent crime severity index', *cities)
```



> Digging deeper and looking at the Violent Crime Index, well, its a whole different story here. Edmonton's Voilent Crime Index has been much higher than the overall Alberta and Calgary's Violent Crime Index. The Violent Crime Index includes crimes such as Assault, Homicide, Robbery and Sexual Assault.

> So we need to look at why is the Severe Crime Index so much higher for Edmonton?

2009-2019: Edmonton Crime data by Neighborhood

Data: Edmonton Neighborhood-CrimeData-Weather from 2009 to 2019

In [ ]:
```
df=pd.read_csv('neighborhooddata-crimedata-weatherdata.csv', encoding_errors='ignore')
df.rename(columns={'Neighbourhood Name': 'NGH_Name', 'Neighbourhood Number':'NGH_Numbe
#df.drop(['Month-Year.1', 'Month-Year'], axis=1, inplace =True)
df.drop(['Descriptive Name'], axis=1, inplace =True)
df.drop(['Month-Year.1'], axis=1, inplace =True)
df.drop(['Occurrence Reported Quarter'], axis=1, inplace =True)
df.rename(columns={'Occurrence Violation Type Group': 'Violation_Type', 'Occurrence Re
df.rename(columns={'Occurrence Reported Quarter': 'QRT', 'Number of Occurrences':'Sum_

df['Month-Year']=pd.to_datetime(df['Month-Year'], format='%b-%y')
df['DT_Year']=df['Month-Year'].dt.year
df['DT_Month']=df['Month-Year'].dt.month_name()
df.drop(['Reported_Year'], axis=1, inplace =True)
df.drop(['Reported_Month'], axis=1, inplace =True)
df.drop(['Month-Year'], 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. Min. (C)'], axis=1, inplace =True)
df.drop(['Air Temp. Avg. Max. (C)'], axis=1, inplace =True)
print('Shape: ',df.shape)
print('Total Null Values : ', df.isnull().sum().sum())
print('Columns: ',df.columns)
print()
print('INFO: ', df.info())
print()


df.head(2)
```

```
Shape:  (112121, 9)
Total Null Values :  0
Columns:  Index(['NGH_Name', 'NGH_Number', 'Latitude', 'Longitude', 'Violation_Type',
       'Sum_Occurrences', 'DT_Year', 'DT_Month', 'AVG_Temp'],
      dtype='object')

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 112121 entries, 0 to 112120
Data columns (total 9 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   NGH_Name         112121 non-null  object
 1   NGH_Number       112121 non-null  int64
 2   Latitude         112121 non-null  float64
 3   Longitude        112121 non-null  float64
 4   Violation_Type   112121 non-null  object
 5   Sum_Occurrences  112121 non-null  int64
 6   DT_Year          112121 non-null  int64
 7   DT_Month         112121 non-null  object
 8   AVG_Temp         112121 non-null  float64
dtypes: float64(3), int64(3), object(3)
memory usage: 7.7+ MB
INFO:  None
```

Out[ ]:

	NGH_Name	NGH_Number	Latitude	Longitude	Violation_Type	Sum_Occurrences	DT_Year	DT_
0	AMBLESIDE	5505	53.430642	-113.600677	Break and Enter	1	2015	
1	ANTHONY HENDAY SOUTH	4014	53.432209	-113.547667	Break and Enter	1	2015	

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

Out[ ]:
```
Theft From Vehicle    28993
Break and Enter       24381
Assault               20887
Theft Of Vehicle      20661
Robbery                7651
Sexual Assaults        5766
Theft Over $5000       3549
Homicide                233
Name: Violation_Type, dtype: int64
```

> Filter Data to only Violent Crimes=Assault, Homicide, Sexual Assault, Robbery

In [ ]:
```
# Filter the data to the years 2009-2019
df_filtered = df.loc[(df['DT_Year'] >= 2009) & (df['DT_Year'] <= 2019)]

# Filter the data to only include the desired violation types
violation_types = ['Assault', 'Homicide', 'Sexual Assaults', 'Robbery']
df_filtered = df_filtered.loc[df_filtered['Violation_Type'].isin(violation_types)]
df_filtered
```

Out[ ]:

	NGH_Name	NGH_Number	Latitude	Longitude	Violation_Type	Sum_Occurrences	DT_Ye
32595	ALLENDALE	5010	53.502277	-113.504821	Assault	1	20
32596	ANTHONY HENDAY RAMPART	4023	53.627497	-113.576375	Assault	1	20
32597	ASPEN GARDENS	5020	53.477535	-113.545664	Assault	1	20
32598	ATHLONE	3010	53.588026	-113.550982	Assault	1	20
32599	BALWIN	2020	53.587951	-113.455075	Assault	1	20
...	...	...	...	...	...	...	
112116	DOWNTOWN	1090	53.539767	-113.499421	Assault	35	20
112117	DOWNTOWN	1090	53.539767	-113.499421	Assault	44	20
112118	DOWNTOWN	1090	53.539767	-113.499421	Assault	37	20
112119	DOWNTOWN	1090	53.539767	-113.499421	Assault	33	20
112120	DOWNTOWN	1090	53.539767	-113.499421	Assault	36	20

34537 rows × 9 columns

> What is the Total of Violent Crimes in Edmonton, between 2009 and 2019?

In [ ]:
```
# What are the most common types of crimes between 2009 & 2019
most_common_crimes=pd.DataFrame({'Count': df_filtered['Violation_Type'].value_counts()
most_common_crimes
```

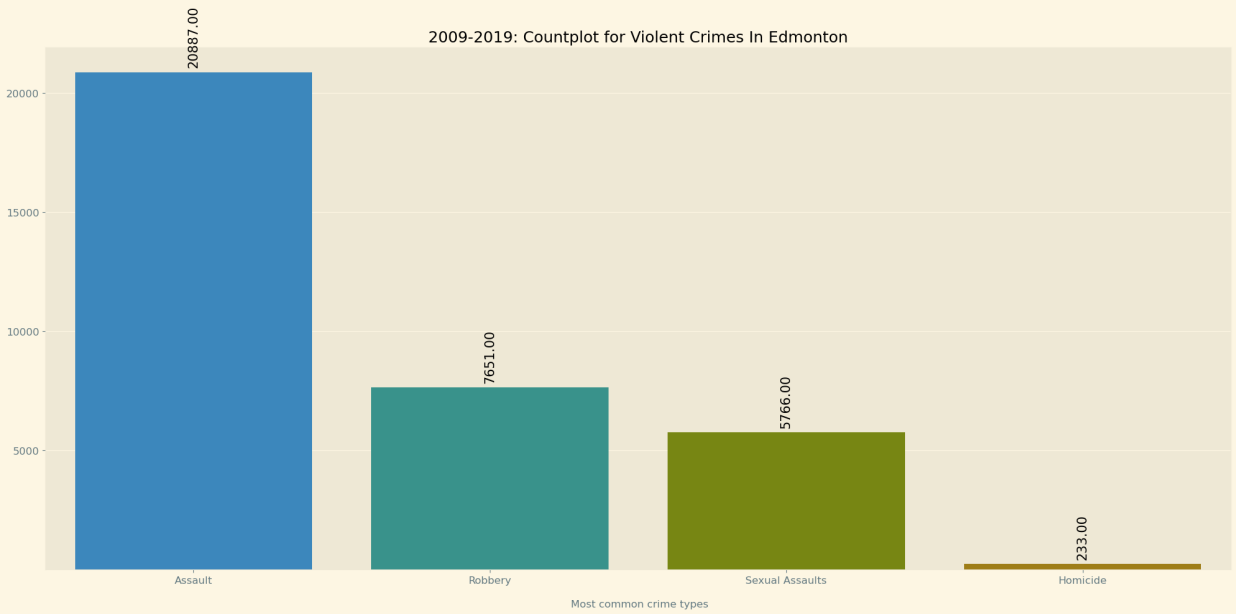
Out[ ]:

	Count
Assault	20887
Robbery	7651
Sexual Assaults	5766
Homicide	233

> Countplot of the Total Violent Crimes committed in Edmonton, between 2009 and 2019.

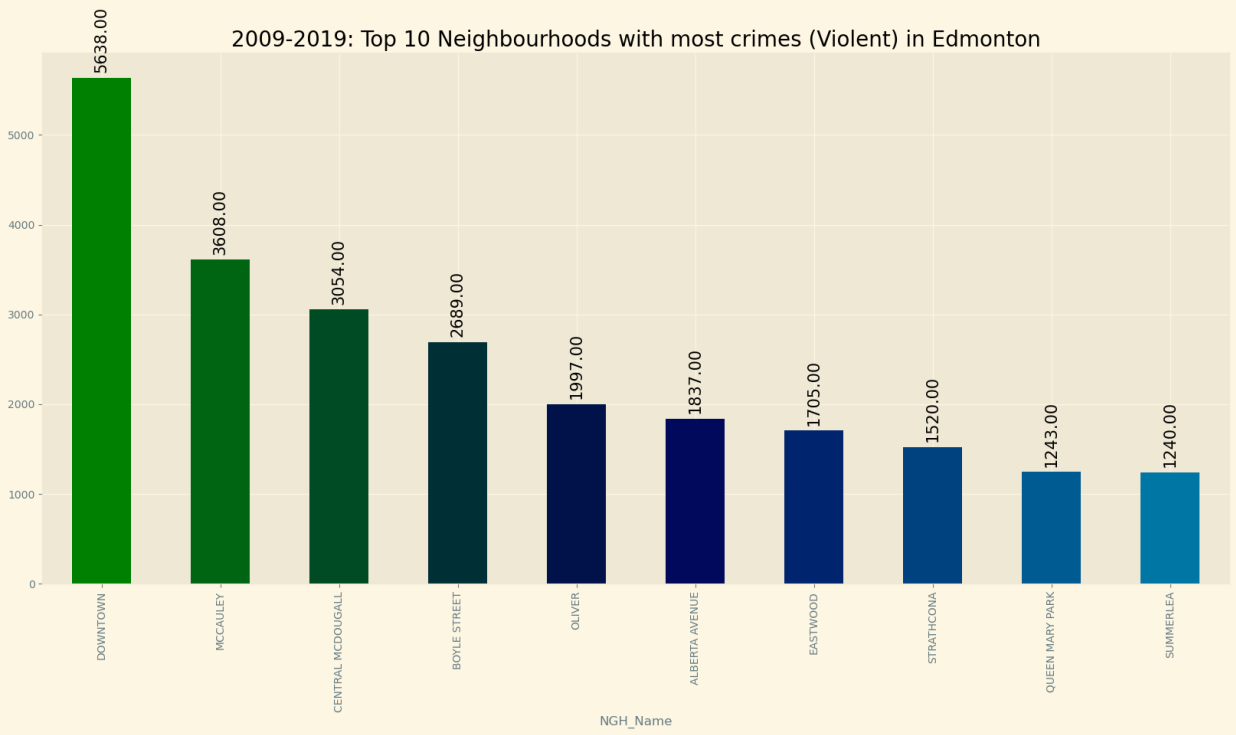
In [ ]:
```
plt.figure(figsize = (20,10))
splot=sns.barplot(x = most_common_crimes.index, y = 'Count', data = most_common_crimes
# add annotations
for p in splot.patches:
    splot.annotate(format(p.get_height(), '.2f'), (p.get_x() + p.get_width() / 2., p.g
                   ha='center', va='bottom', fontsize=15, color='black', rotation=90,
                   textcoords='offset points')
plt.yticks((np.arange(5000, most_common_crimes['Count'].max(), 5000)))
```

```
plt.ylabel(None)
plt.tick_params(labelsize = 12)
plt.xlabel('\n Most common crime types', fontsize = 12)
plt.title('2009-2019: Countplot for Violent Crimes In Edmonton', fontsize = 18)
plt.tight_layout()
```



> Looking at the Top 10 Neighborhoods with the highest Violent Crimes in Edmonton, between 2009 and 2019.

In [ ]:
```
color = plt.cm.ocean(np.linspace(0, 1, 15))
splot=df_filtered.groupby(['NGH_Name'])['Sum_Occurrences'].sum().sort_values(ascending
for p in splot.patches:
    splot.annotate(format(p.get_height(), '.2f'), (p.get_x() + p.get_width() / 2., p.g
                   ha='center', va='bottom', fontsize=15, color='black', rotation=90,
                   textcoords='offset points')
plt.title('2009-2019: Top 10 Neighbourhoods with most crimes (Violent) in Edmonton',fo
plt.xticks(rotation = 90)
plt.show()
```



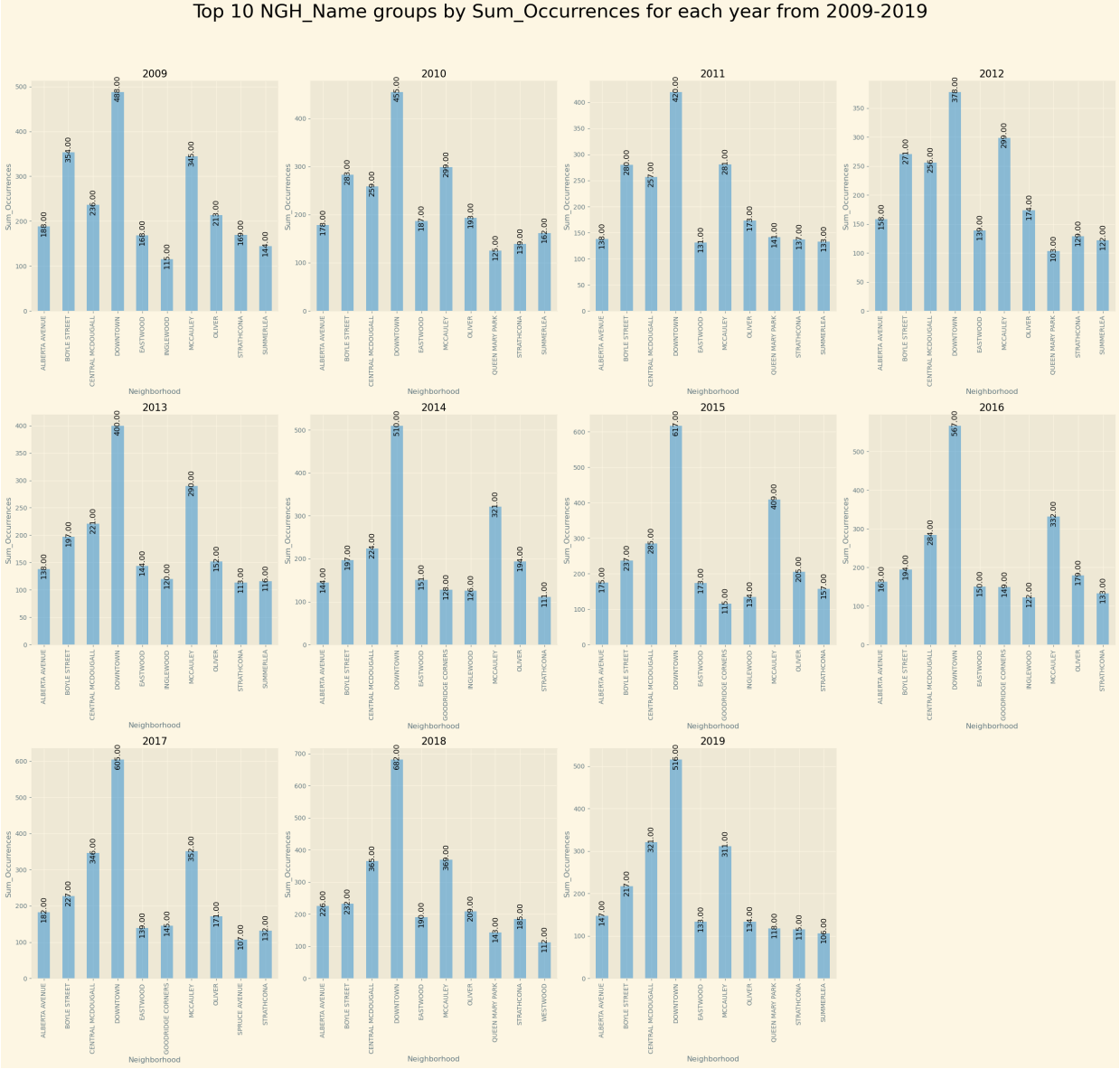
In [ ]:
```
top_10=['DOWNTOWN', 'MCCUULEY', 'CENTRAL MCDOUGALL', 'BOYLE STREET',
        'OLIVER', 'ALBERTA AVENUE', 'EASTWOOD', 'STRATHCONA', 'QUEEN MARY PARK', 'SUM
# calculate total sum of occurrences across all neighborhoods
total_occurrences = df['Sum_Occurrences'].sum()

# filter dataframe to only include top 6 neighborhoods
top_10_occurrences = df[df['NGH_Name'].isin(top_10)]['Sum_Occurrences'].sum()

# calculate percentage of total crime held by top 6 neighborhoods
percentage_top_6 = (top_10_occurrences / total_occurrences) * 100

# print the result
print("The top 10 neighborhoods hold {:.2f}% of the overall crime".format(percentage_t
```

```
The top 10 neighborhoods hold 17.91% of the overall crime
```

> Looking at the Top 10 Neighborhoods for every year between 2009 and 2019. Do the same cities show up every year?

In [ ]:
```
# Define the violation types of interest
#violation_types = ['Assault', 'Homicide', 'Sexual Assaults', 'Robbery']

# Filter the data to the years 2009-2019 and the violation types of interest
#df = df[(df['DT_Year'] >= 2009) & (df['DT_Year'] <= 2019) & (df['Violation_Type'].isi

# Group the data by year and NGH_Name and compute the Sum_Occurrences
grouped = df_filtered.groupby(['DT_Year', 'NGH_Name'])['Sum_Occurrences'].sum().reset_

# Create a list of years to loop over
years = range(2009, 2020)

# Create a 3x4 grid of subplots
fig, axes = plt.subplots(nrows=3, ncols=4, figsize=(25,25))
```

```
# Flatten the axes array for easier indexing
axes = axes.flatten()

# Iterate over the years and plot the top 10 NGH_Name groups for each year
for i, year in enumerate(years):
    # Filter the data to the current year
    year_data = grouped[grouped['DT_Year'] == year]

    # Sort the groups by Sum_Occurrences and select the top 10
    top_groups = year_data.groupby('NGH_Name')['Sum_Occurrences'].sum().sort_values(as

    # Filter the data to only include the top 10 groups
    year_data = year_data[year_data['NGH_Name'].isin(top_groups.index)]

    # Pivot the data to create a table with NGH_Name as rows and year as columns
    pivot_table = year_data.pivot(index='NGH_Name', columns='DT_Year', values='Sum_Occ

    # Create a bar plot of the pivot table
    ax = axes[i]
    pivot_table.plot(kind='bar', ax=ax, stacked=True, alpha=0.5, legend=None)
    ax.set_xlabel('Neighborhood')
    ax.set_ylabel('Sum_Occurrences')
    ax.set_title(str(year))
    # add annotations
    for p in ax.patches:
        ax.annotate(format(p.get_height(), '.2f'), (p.get_x() + p.get_width() / 2., p.
                    ha='center', va='center', fontsize=12, color='black', rotation=90,
                    textcoords='offset points')

# Hide any unused subplots
for i in range(len(years), len(axes)):
    fig.delaxes(axes[i])

# Add a main title to the figure
fig.suptitle('Top 10 NGH_Name groups by Sum_Occurrences for each year from 2009-2019',

# Adjust the layout and spacing of the subplots
fig.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```



> Its seems that there are some Neighborhoods that keep showing up in the Top10 for every year between 2009 and 2019.

Top 6 Neighborhoods that consistently show up are:

- Downtown
- Alberta Avenue
- Boyle Street
- Central McDougall
- McCaulley
- Oliver

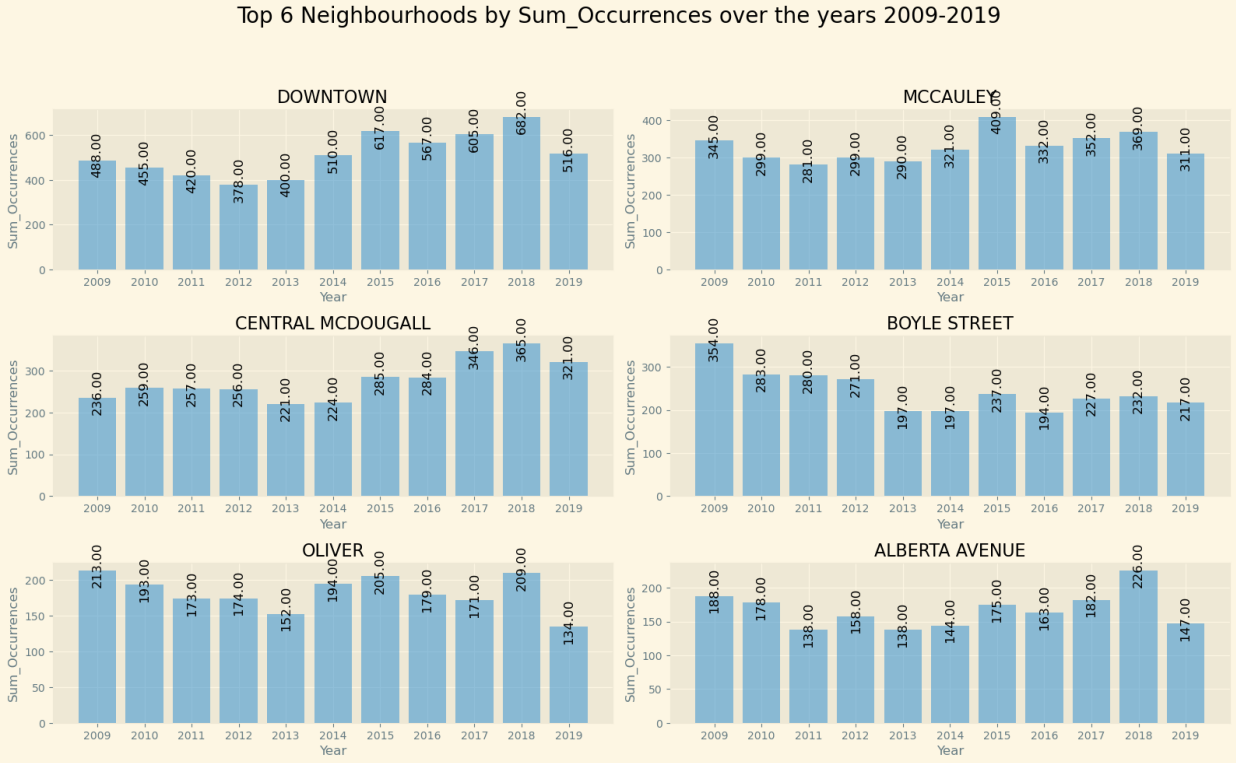
The next 5 Neighborhoods come next, that intermitantly show up in the top 10:

- Eastwood
- Queen Mary Park
- Inglewood
- Strathcona
- Summerlea

> Lets view the Top 6 a little differently.

In [ ]:
```
# Define the violation types of interest
#violation_types = ['Assault', 'Homicide', 'Sexual Assaults', 'Robbery']

# Filter the data to the years 2009-2019 and the violation types of interest
#df = df[(df['DT_Year'] >= 2009) & (df['DT_Year'] <= 2019) & (df['Violation_Type'].isi
# Aggregate the Sum_Occurrences values by NGH_Name and year
grouped = df_filtered.groupby(['NGH_Name', df_filtered['DT_Year']])['Sum_Occurrences']
#grouped = df.groupby(['NGH_Name', df['DT_Year']])['Sum_Occurrences'].sum().sort_value

# Sort the groups by Sum_Occurrences and select the top 15
top_groups = grouped.groupby('NGH_Name')['Sum_Occurrences'].sum().sort_values(ascendir

# Filter the grouped data to only include the top 15 groups
grouped = grouped[grouped['NGH_Name'].isin(top_groups.index)]

# Pivot the grouped data to create a table with NGH_Name as rows and year as columns
pivot_table = grouped.pivot(index='NGH_Name', columns='DT_Year', values='Sum_Occurrenc

# Create a 4x4 grid of subplots
fig, axes = plt.subplots(nrows=5, ncols=2, figsize=(16, 16))

# Flatten the axes array for easier indexing
axes = axes.flatten()

# Iterate over the top 15 groups and plot each group in a separate subplot
for i, group in enumerate(top_groups.index):
    data = pivot_table.loc[group].values
    ax = axes[i]
    ax.bar(range(11), data, align='center', alpha=0.5)
    ax.set_xticks(range(11))
    ax.set_xticklabels(pivot_table.columns)
    ax.set_xlabel('Year')
    ax.set_ylabel('Sum_Occurrences')
    ax.set_title(group)
    # add annotations
    for p in ax.patches:
        ax.annotate(format(p.get_height(), '.2f'), (p.get_x() + p.get_width() / 2., p.
                    ha='center', va='center', fontsize=12, color='black', rotation=90,
                    textcoords='offset points')

# Hide any unused subplots
for i in range(len(top_groups.index), len(axes)):
    fig.delaxes(axes[i])
```

```
# Add a main title to the figure
fig.suptitle('Top 6 Neighbourhoods by Sum_Occurrences over the years 2009-2019', fonts

# Adjust the layout and spacing of the subplots
fig.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```



> Looking at the breakdown of Violent Crimes committed within the Top6, which are the most popular neighborhoods every year, between 2009 ad 2019.

In [ ]:

```
# Define the violation types of interest
#violation_types = ['Assault', 'Homicide', 'Sexual Assaults', 'Robbery']

# Filter the data to the years 2009-2019 and the violation types of interest
#df = df[(df['DT_Year'] >= 2009) & (df['DT_Year'] <= 2019) & (df['Violation_Type'].isi

# Group the data by Neighborhood and Violation_Type and compute the Sum_Occurrences
grouped = df_filtered.groupby(['NGH_Name', 'Violation_Type'])['Sum_Occurrences'].sum()

# Sort the groups by Sum_Occurrences and select the top 8 neighborhoods
top_groups = grouped.groupby('NGH_Name')['Sum_Occurrences'].sum().sort_values(ascendir

# Filter the data to only include the top 8 neighborhoods
grouped = grouped[grouped['NGH_Name'].isin(top_groups.index)]

# Create a 4x4 grid of bar plots, where each plot shows the breakdown of Violation_Typ
fig, axes = plt.subplots(nrows=5, ncols=2, figsize=(25, 25))

# Flatten the axes array for easier indexing
axes = axes.flatten()

# Iterate over the top 8 neighborhoods and create a bar plot for each one
```

```
for i, group in enumerate(top_groups.index):
    # Filter the data to the current neighborhood
    group_data = grouped[grouped['NGH_Name'] == group]

    # Pivot the data to create a table with Violation_Type as rows and Neighborhood as
    pivot_table = group_data.pivot(index='Violation_Type', columns='NGH_Name', values=

    # Create a bar plot of the pivot table
    ax = axes[i]
    pivot_table.plot(kind='bar', ax=ax, alpha=0.5)
    ax.set_xlabel('Violation_Type')
    ax.set_ylabel('Sum_Occurrences')
    ax.set_title(group)

    # add annotations
    for p in ax.patches:
        ax.annotate(format(p.get_height(), '.2f'), (p.get_x() + p.get_width() / 2., p.
                    ha='center', va='bottom', fontsize=15, color='black', rotation=90,
                    textcoords='offset points')

# Hide any unused subplots
for i in range(len(top_groups.index), len(axes)):
    fig.delaxes(axes[i])

# Add a main title to the figure
fig.suptitle('Top 6 Neighborhoods by Sum_Occurrences with a breakdown of Violation_Typ

# Adjust the layout and spacing of the subplots
fig.tight_layout(rect=[0, 0.03, 1, 0.95])

plt.show()
```

> Pivot df_filtered

Before proceeding with our analysis of severe crime trends in Edmonton neighborhoods, we first used Pandas to pivot our filtered dataset. This involved creating a new dataframe, 'pivoted_df_filtered,' using the Pandas pivot_table function. The pivot_table function allowed us to aggregate the crime data by neighborhood, location, average temperature, year, and month, while also breaking it down by Violation Type. Specifically, we set the index to include 'NGH_Name,' 'NGH_Number,' 'Latitude,' 'Longitude,' 'AVG_Temp,' 'DT_Year,' and 'DT_Month,' and set the columns to 'Violation_Type.' The values were set to 'Sum_Occurrences,' and we used 'aggfunc=sum' to aggregate the values by summing them.

In [ ]:
```
# Create a pivot table that shows the total number of occurrences for each violation t
pivoted_df_filtered= pd.pivot_table(df_filtered, index=['NGH_Name', 'NGH_Number', 'Lat
pivoted_df_filtered = pivoted_df_filtered.reset_index()
pivoted_df_filtered
```

Out[ ]:

Violation_Type	NGH_Name	NGH_Number	Latitude	Longitude	AVG_Temp	DT_Year	DT_Month
0	ABBOTTSFIELD	2010	53.574143	-113.388758	-20.763214	2019	February
1	ABBOTTSFIELD	2010	53.574143	-113.388758	-17.671935	2009	December
2	ABBOTTSFIELD	2010	53.574143	-113.388758	-16.388929	2014	February
3	ABBOTTSFIELD	2010	53.574143	-113.388758	-15.673710	2012	December
4	ABBOTTSFIELD	2010	53.574143	-113.388758	-14.465968	2010	December
...	...	...	...	...	...	...	...
23884	YOUNGSTOWN INDUSTRIAL	4660	53.552476	-113.610588	15.504839	2011	August
23885	YOUNGSTOWN INDUSTRIAL	4660	53.552476	-113.610588	16.247581	2009	July
23886	YOUNGSTOWN INDUSTRIAL	4660	53.552476	-113.610588	16.505484	2014	August
23887	YOUNGSTOWN INDUSTRIAL	4660	53.552476	-113.610588	17.485806	2018	July
23888	YOUNGSTOWN INDUSTRIAL	4660	53.552476	-113.610588	18.354516	2015	July

23889 rows × 11 columns

In [ ]:
```
print('2009-2019: Average of Assaults: ', pivoted_df_filtered['Assault'].mean())
print('2009-2019: Average of Homicide: ', pivoted_df_filtered['Homicide'].mean())
print('2009-2019: Average of Robbery: ', pivoted_df_filtered['Robbery'].mean())
print('2009-2019: Average of Sexual Assaults: ', pivoted_df_filtered['Sexual Assaults'
```

```
2009-2019: Average of Assaults:  2.5423835237975636
2009-2019: Average of Homicide:  0.009837163548076521
2009-2019: Average of Robbery:  0.5177278245217464
2009-2019: Average of Sexual Assaults:  0.30980786135878435
```

> Looking at the Top 6 most popular Neighborhoods, lets see how they fair by month, between 2009 and 2019.

In [ ]:
```
top_ngh=['DOWNTOWN', 'MCCAULEY', 'CENTRAL MCDOUGALL', 'BOYLE STREET', 'OLIVER', 'ALBER
```

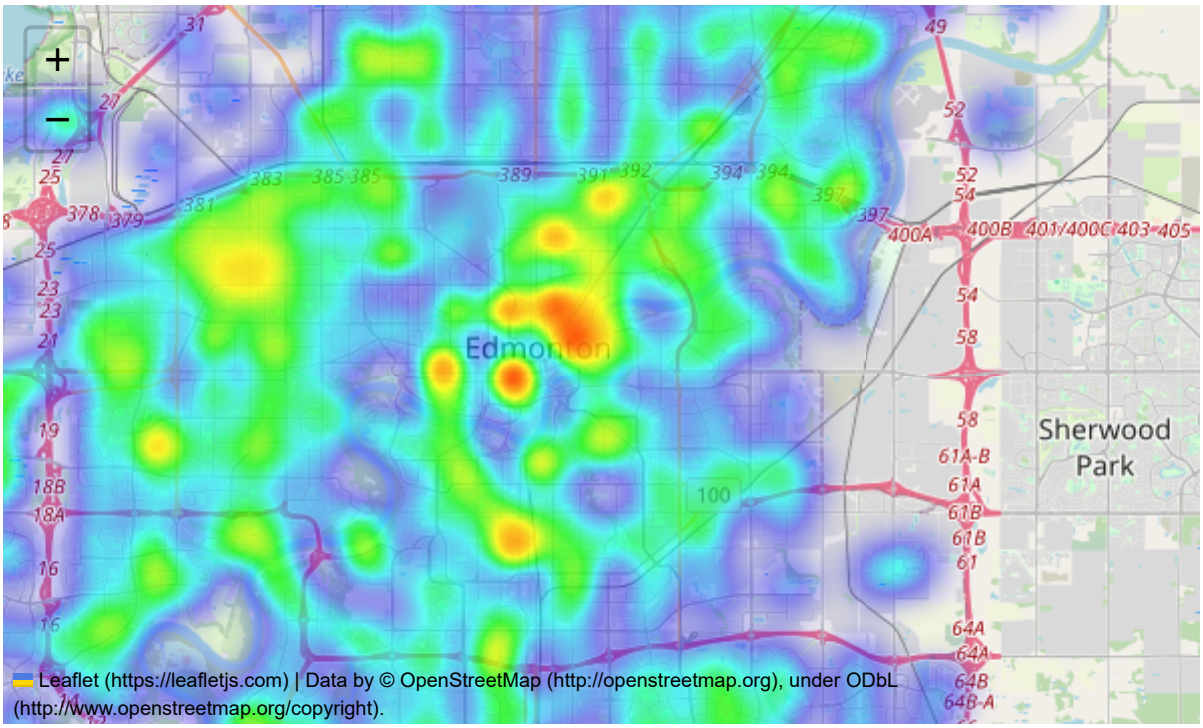
In [ ]:
```
top_10=['DOWNTOWN', 'MCCUULEY', 'CENTRAL MCDOUGALL', 'BOYLE STREET',
        'OLIVER', 'ALBERTA AVENUE', 'EASTWOOD', 'STRATHCONA', 'QUEEN MARY PARK', 'SUN
```

Visualizing the Violent Crimes in Edmonton with Folium and Geopandas

Foluim Heat Map

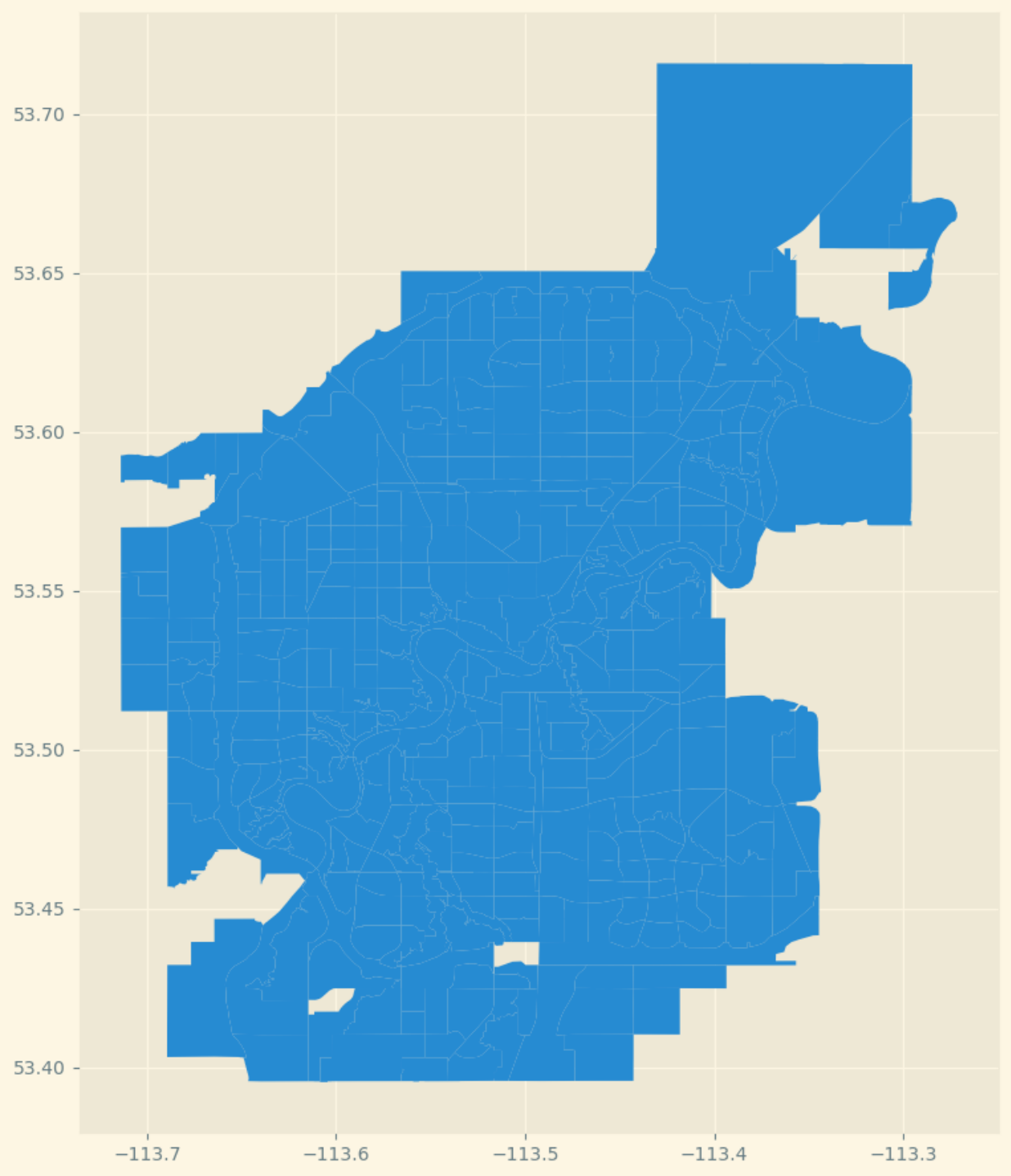
In [ ]:
```
# Folium
location_of_most_common_crimes = df_filtered[df_filtered['Violation_Type'].isin(most_c

my_map=folium.Map(location = [53.4,-113.07], #Initiate map on Boston city
                  zoom_start = 11,
                  min_zoom = 11
)

#HeatMap(data=location_of_most_common_crimes.sample(10000), radius=16).add_to(my_map)
HeatMap(data=location_of_most_common_crimes, radius=16).add_to(my_map)


my_map
```

Out[ ]:



Geopandas

In [ ]:
```
# 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()
edmonton_shp.plot()
plt.tight_layout()
```

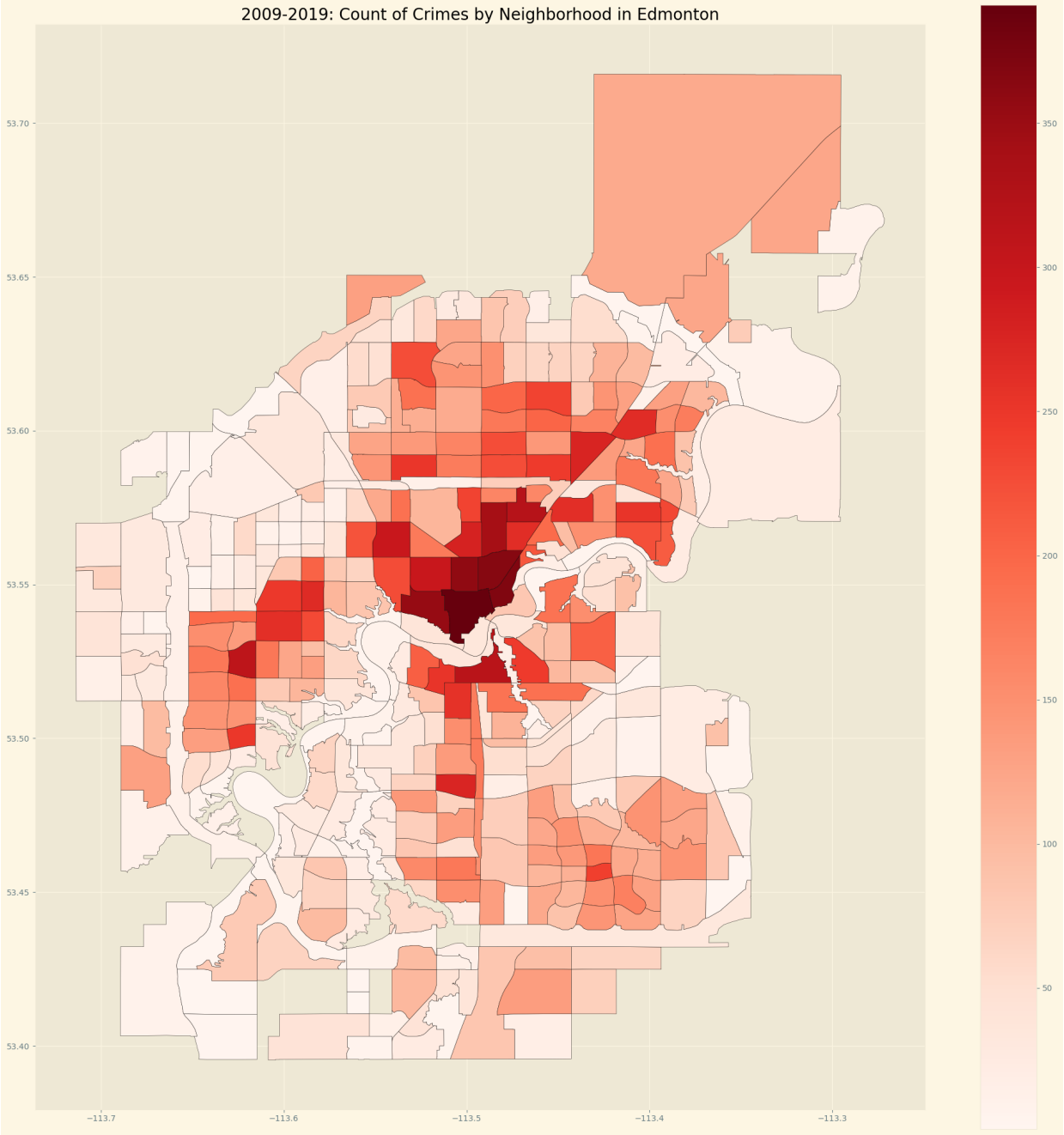


In [ ]:
```
Neighborhoods=pd.DataFrame({'Count':df_filtered['NGH_Name'].value_counts().sort_values
Neighborhoods.head(10)
```

Out[ ]:

	Count
DOWNTOWN	391
BOYLE STREET	371
MCCAULEY	366
CENTRAL MCDOUGALL	359
OLIVER	354
ALBERTA AVENUE	340
EASTWOOD	320
STRATHCONA	316
SUMMERLEA	311
INGLEWOOD	298

In [ ]:
```
edmonton_shp['Number_Crimes'] = edmonton_shp['NGH_Name'].map(Neighborhoods['Count']) #
# This takes the number of times each individual NGH_Name shows up and creates a colum
ax = edmonton_shp.plot(column = edmonton_shp['Number_Crimes'], cmap = 'Reds', legend =
#add_label()
plt.title('2009-2019: Count of Crimes by Neighborhood in Edmonton', fontsize = 20)
plt.tight_layout()
```



Edmonton Police Services Statistics

In [ ]:
```
EPS=pd.read_csv('EPS_Personnel_Statistics.csv')
EPS
```

Out[ ]:

	Year	Date	Police Strength	Police/Civilian Strength	Police Officers	Male Police Officers	Female Police Officers	% Female Officers	Civilian/Other Personnel
0	2000	06/15/2000 12:00:00 AM	174.6	235.4	1176	1034	142	12.1	386
1	2001	06/15/2001 12:00:00 AM	168.7	220.2	1152	1012	140	12.2	331
2	2002	06/15/2002 12:00:00 AM	164.3	215.8	1142	1008	134	11.7	332
3	2003	06/15/2003 12:00:00 AM	174.3	224.5	1225	1066	159	13.0	335
4	2004	06/15/2004 12:00:00 AM	173.9	229.2	1253	1065	188	15.0	358
5	2005	06/15/2005 12:00:00 AM	180.9	235.0	1334	1125	209	15.7	359
6	2006	05/15/2006 12:00:00 AM	178.7	237.5	1356	1133	223	16.4	395
7	2007	05/15/2007 12:00:00 AM	175.0	235.4	1364	1142	222	16.3	423
8	2008	05/15/2008 12:00:00 AM	168.6	237.4	1345	1107	238	17.7	505
9	2009	05/15/2009 12:00:00 AM	178.1	247.9	1457	1180	277	19.0	520
10	2010	05/15/2010 12:00:00 AM	195.9	268.9	1628	1320	308	18.9	571
11	2011	05/15/2011 12:00:00 AM	190.2	264.2	1607	1305	302	18.8	589
12	2012	05/15/2012 12:00:00 AM	185.4	262.8	1603	1314	289	18.0	618
13	2013	05/15/2013 12:00:00 AM	183.6	263.7	1639	1334	305	18.6	641
14	2014	05/15/2014 12:00:00 AM	179.2	263.4	1655	1343	312	18.9	697

	Year	Date	Police Strength	Police/Civilian Strength	Police Officers	Male Police Officers	Female Police Officers	% Female Officers	Civilian/Other Personnel
15	2015	05/15/2015 12:00:00 AM	176.3	256.0	1665	1361	304	18.3	699
16	2016	05/15/2016 12:00:00 AM	180.3	266.4	1739	1411	328	18.9	777
17	2017	05/15/2017 12:00:00 AM	180.6	269.8	1775	1440	335	18.9	827
18	2018	05/15/2018 12:00:00 AM	187.3	274.1	1882	1496	386	20.5	812

In [ ]:
```
EPS.columns
```

Out[ ]:
```
Index(['Year', 'Date', 'Police Strength', 'Police/Civilian Strength',
       'Police Officers', 'Male Police Officers', 'Female Police Officers',
       '% Female Officers', 'Civilian/Other Personnel'],
      dtype='object')
```

In [ ]:
```
ax = EPS.plot.bar(x='Year', y='Police Officers')
for index, row in EPS.iterrows():
    ax.annotate(row['Police Officers'], xy=(index, row['Police Officers']), ha='center
plt.title('2000-2018: EPS Personnel Statistics.')
plt.show()
```