



The Battle of Neighborhoods Vancouver

RICHARD BALBUENA

“Most of the world will make decisions by either guessing or using their gut. They will be either lucky or wrong.”

SUHAIL DOSHI, CHIEF EXECUTIVE OFFICER, MIXPANEL.

Introduction

Vancouver is a coastal seaport city in western Canada, located in the Lower Mainland region of British Columbia. As the most populous city in the province, the 2016 census recorded 631,486 people in the city, up from 603,502 in 2011. The Greater Vancouver area had a population of 2,463,431 in 2016, making it the third-largest metropolitan area in Canada. Vancouver has the highest population density in Canada, with over 5,400 people per square kilometre, which makes it the fifth-most densely populated city with over 250,000 residents in North America, behind New York City, Guadalajara, San Francisco, and Mexico City. Vancouver is one of the most ethnically and linguistically diverse cities in Canada: 52% of its residents are not native English speakers, 48.9% are native speakers of neither English nor French, and 50.6% of residents belong to visible minority groups. Vancouver is consistently named as one of the top five worldwide cities for livability and quality of life, and the Economist Intelligence Unit acknowledged it as the first city ranked among the top ten of the world's most well-living cities for ten consecutive years. In 2011, the city planned to become the greenest city in the world by 2020. Vancouverism is the city's urban planning design philosophy.

Background

Safety is the top concern when moving to a new area. If you don't feel safe in your own home, you you're not going to be able to enjoy living there.

Key Questions

- A. What is the most common crime in Vancouver?
- B. Which of the boroughs have the highest crime rates?
- C. Which of the boroughs have the lowest crime rates?
- D. Identify the top 10 venues in each neighborhood

Business Problem

This project aims to select the safest borough in Vancouver, BC Canada based on the total crime, explore the neighborhoods of that borough to find the 10 most common venues in each neighborhood and finally cluster the neighborhoods using K-Mean clustering.

Interest and stakeholders

Expats who are considering to relocate to Vancouver and Real States companies will be interested to identify the safest borough in Vancouver and explore its neighborhoods and common venues around each neighborhood

Data section

The data acquisition for this project is a combination of data from two sources:

- The first data source of the project uses Vancouver crime data. This dataset contains the Vancouver Police Department's crime records from 2013 to 2019. It can be found at: <https://vancouver.ca/police/organization/planning-research-audit/stats-crime-rate.html>
- Foursquare API as its prime data gathering source as it has a database of more than 105 million places, especially their places API which provides the ability to perform location search, location sharing and details about a business. Photos, tips and reviews jolted by Foursquare users can also be used in many productive ways to add value to the results.

Methodology

- HTTP requests would be made to this Foursquare API server using postal codes of the Vancouver city neighborhoods to pull the location information (Latitude and Longitude).
- Foursquare API search feature would be enabled to collect the nearby places of the neighborhoods.
- Due to http request limitations the number of places per neighborhood parameter would reasonably be set to 100 and the radius parameter would be set to 700.

Methodology

- Folium and Python visualization library would be used to visualize the neighborhoods cluster distribution of Vancouver city over an interactive leaflet map.
- Extensive comparative analysis of two randomly picked neighborhoods would be carried out to derive the desirable insights from the outcomes using python's scientific libraries Pandas, NumPy and Scikit-learn.
- Unsupervised machine learning algorithm K-mean clustering would be applied to form the clusters of different categories of places residing in and around the neighborhoods. These clusters from each of those two chosen neighborhoods would be analyzed individually collectively and comparatively to derive the conclusions.

Methodology

- Initial data exploration and summary.

	TYPE	YEAR	MONTH	DAY	HOUR	MINUTE	HUNDRED_BLOCK	NEIGHBOURHOOD	X	Y
0	Break and Enter Commercial	2012	12	14	8	52	NaN	Oakridge	491285.000000	5.453433e+06
1	Break and Enter Commercial	2019	3	7	2	6	10XX SITKA SQ	Fairview	490612.964805	5.457110e+06
2	Break and Enter Commercial	2019	8	27	4	12	10XX ALBERNI ST	West End	491007.779775	5.459174e+06
3	Break and Enter Commercial	2014	8	8	5	13	10XX ALBERNI ST	West End	491015.943352	5.459166e+06
4	Break and Enter Commercial	2005	11	14	3	9	10XX ALBERNI ST	West End	491021.385727	5.459161e+06

Python packages and Dependencies

2. Load Data into a DataFrame

Hidden Cell personal data. I used IBM boto3 to load data. Code used:

```
import types from botocore.client import Config import ibm_boto3
```

```
def iter(self): return 0 Personnel credentials
```

add missing iter method, so pandas accepts body as file-like object

```
if not hasattr(body, "iter"): body.iter = types.MethodType( iter, body )
```

If you are reading an Excel file into a pandas DataFrame, replace read_csv by read_excel in the next statement.

```
crime_records = pd.read_csv(body) crime_records.head()
```

```
In [ ]: # The code was removed by Watson Studio for sharing.
```

2. Load Data into a DataFrame

Hidden Cell personal data. I used IBM boto3 to load data. Code used:

```
import types from botocore.client import Config import ibm_boto3
```

```
def iter(self): return 0 Personel credentials
```

```
#add missing iter method, so pandas accepts body as file-like object if not hasattr(body, "iter"): body.iter = types.MethodType( iter, body )
```

```
#If you are reading an Excel file into a pandas DataFrame, replace read_csv by read_excel in the next statement. crime_records = pd.read_csv(body) crime_records.head()
```



```
# hidden_cell
import types
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
```

Load Data into a DataFrame

```
crime_records = pd.read_csv(body)
crime_records.head()
```

: [2]:

	TYPE	YEAR	MONTH	DAY	HOUR	MINUTE	HUNDRED_BLOCK	NEIGHBOURHOOD	X	Y
0	Break and Enter Commercial	2012	12	14	8	52	NaN	Oakridge	491285.000000	5.453433e+06
1	Break and Enter Commercial	2019	3	7	2	6	10XX SITKA SQ	Fairview	490612.964805	5.457110e+06
2	Break and Enter Commercial	2019	8	27	4	12	10XX ALBERNI ST	West End	491007.779775	5.459174e+06
3	Break and Enter Commercial	2014	8	8	5	13	10XX ALBERNI ST	West End	491015.943352	5.459166e+06
4	Break and Enter Commercial	2005	11	14	3	9	10XX ALBERNI ST	West End	491021.385727	5.459161e+06

3. Initial data exploration and summary

3.0 Missing data detection

```
# Reusable function for detecting missing data
def missing_value_describe(data):
    # check missing values in training data
    missing_value_stats = (data.isnull().sum() / len(data)*100)
    missing_value_col_count = sum(missing_value_stats > 0)
    missing_value_stats = missing_value_stats.sort_values(ascending=False)[:missing_value_col_count]
    print("Number of columns with missing values:", missing_value_col_count)
    if missing_value_col_count != 0:
        # print out column names with missing value percentage
        print("\nMissing percentage (descending):")
        print(missing_value_stats)
    else:
        print("No missing data!!!")
missing_value_describe(crime_records)
```

Number of columns with missing values: 4

Missing percentage (descending):

NEIGHBOURHOOD 10.409943

Y 0.019230

X 0.019230

HUNDRED_BLOCK 0.002083

dtype: float64

From the above missing data examination, we have 4 columns with missing data: they are NEIGHBOURHOOD (10.42% missing), MINUTE(10.02% missing), HOUR(10.02% missing), and HUNDRED_BLOCK(0.002% missing).

The missing percentage of MINUTE and HOUR columns are the same we are going to disregard the missing information. However, the HUNDRED_BLOCK, which is the physical address, has ~10% lower missing value than the NEIGHBOURHOOD column. The missing data is due to Statistics Canada definition of neighbourhoods within municipalities. Neighbourhoods within the City of Vancouver are based on the census tract (CT) concept within census metropolitan area (CMA). The missing NEIGHBOURHOOD columns might be caused by unlabeled neighbourhoods in the Statistics Canada dataset or they just aren't labeled. I am thinking about create a dictionary for each street-neighbourhood pair to predict the missing neighbourhood.

3. Initial data exploration and summary

3.0 Missing data detection

```
# Reusable function for detecting missing data
def missing_value_describe(data):
    # check missing values in training data
    missing_value_stats = (data.isnull().sum() / len(data)*100)
    missing_value_col_count = sum(missing_value_stats > 0)
    missing_value_stats = missing_value_stats.sort_values(ascending=False)[:missing_value_col_count]
    print("Number of columns with missing values:", missing_value_col_count)
    if missing_value_col_count != 0:
        # print out column names with missing value percentage
        print("\nMissing percentage (descending):")
        print(missing_value_stats)
    else:
        print("No missing data!!!")
missing_value_describe(crime_records)
```

Number of columns with missing values: 4

Missing percentage (descending):

NEIGHBOURHOOD 10.409943

Y 0.019230

X 0.019230

HUNDRED_BLOCK 0.002083

dtype: float64

From the above missing data examination, we have 4 columns with missing data: they are NEIGHBOURHOOD (10.42% missing), MINUTE(10.02% missing), HOUR(10.02% missing), and HUNDRED_BLOCK(0.002% missing).

The missing percentage of MINUTE and HOUR columns are the same we are going to disregard the missing information. However, the HUNDRED_BLOCK, which is the physical address, has ~10% lower missing value than the NEIGHBOURHOOD column. The missing data is due to Statistics Canada definition of neighbourhoods within municipalities. Neighbourhoods within the City of Vancouver are based on the census tract (CT) concept within census metropolitan area (CMA). The missing NEIGHBOURHOOD columns might be caused by unlabeled neighbourhoods in the Statistics Canada dataset or they just aren't labeled. I am thinking about create a dictionary for each street-neighbourhood pair to predict the missing neighbourhood.

3.1 Dimension of dataset

```
print("the dimension:", crime_records.shape)
```

```
the dimension: (624038, 10)
```

We have over 600K rows and 10 columns in our dataset.

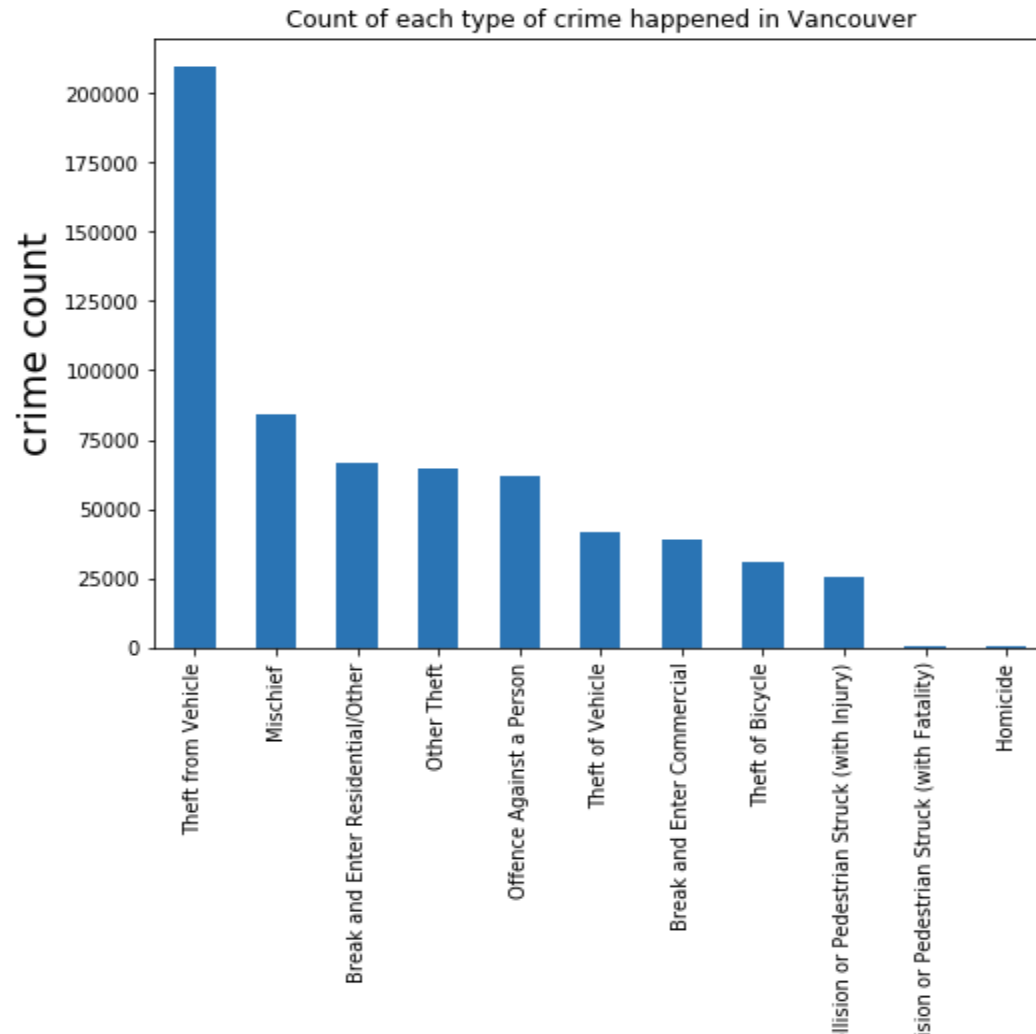
4. Distribution of each type of crime:

```
crime_records['TYPE'].value_counts()
```

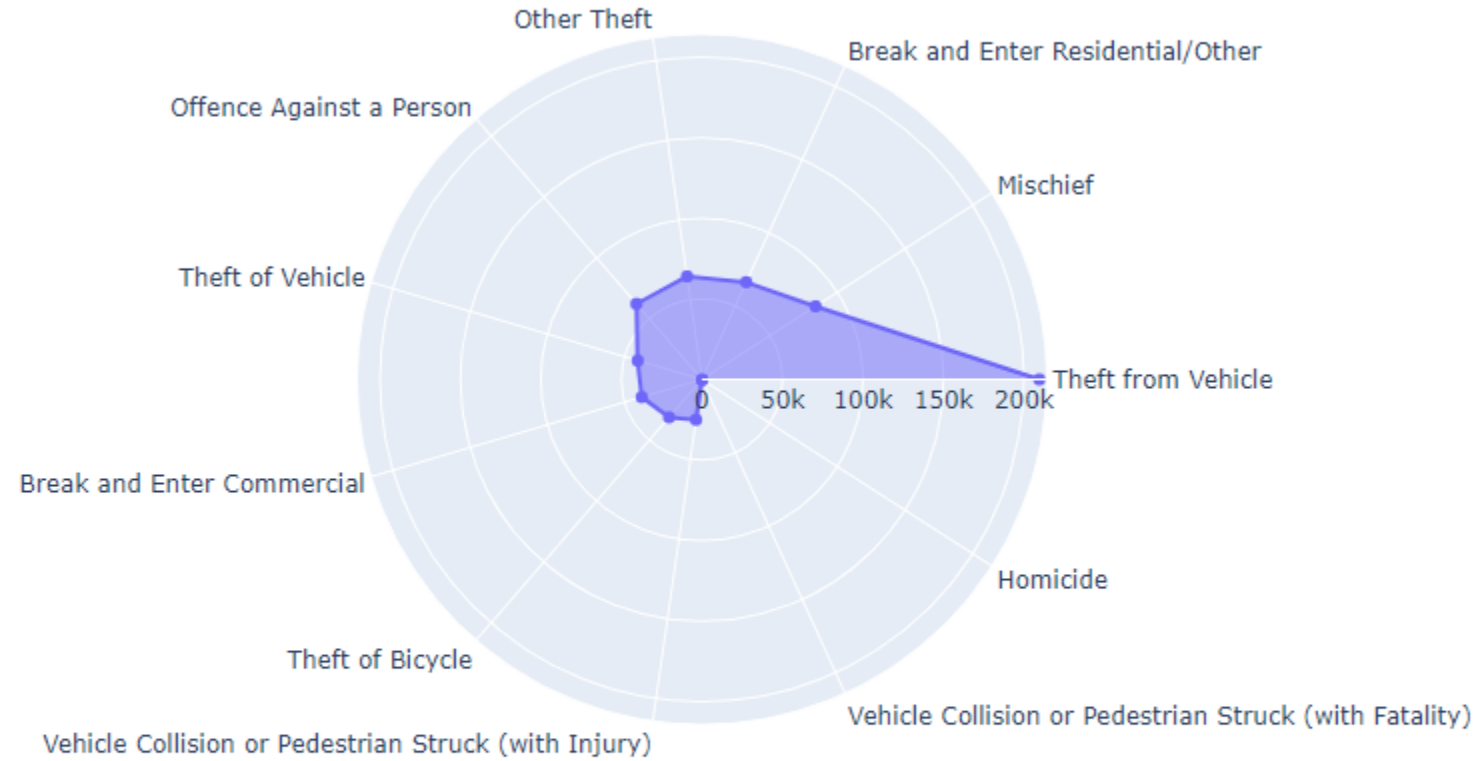
```
Out[5]: Theft from Vehicle      209609
        Mischief                83970
        Break and Enter Residential/Other  66378
        Other Theft             64611
        Offence Against a Person  62078
        Theft of Vehicle         41528
        Break and Enter Commercial  38916
        Theft of Bicycle         31112
        Vehicle Collision or Pedestrian Struck (with Injury)  25294
        Vehicle Collision or Pedestrian Struck (with Fatality)  290
        Homicide                 252
        Name: TYPE, dtype: int64
```

```
# crime type distribution
nameplot = crime_records['TYPE'].value_counts().plot.bar(title='Count of each type of crime happened in Vancouver', figsize=(8,6))
nameplot.set_xlabel('category',size=20)
nameplot.set_ylabel('crime count',size=20)
```

Out[6]: Text(0, 0.5, 'crime count')



Vancouver Crime Record Radar Chart by Crime Type



It appears that the most common type of crime is associated with vehicle in Vancouver: Theft from Vehicle, Theft of Bicycle and Vehicle Collision. Since 2003, there are 209609 crime categorized as "Theft from Vehicle".

Has the crime decreased over the years in Vancouver?

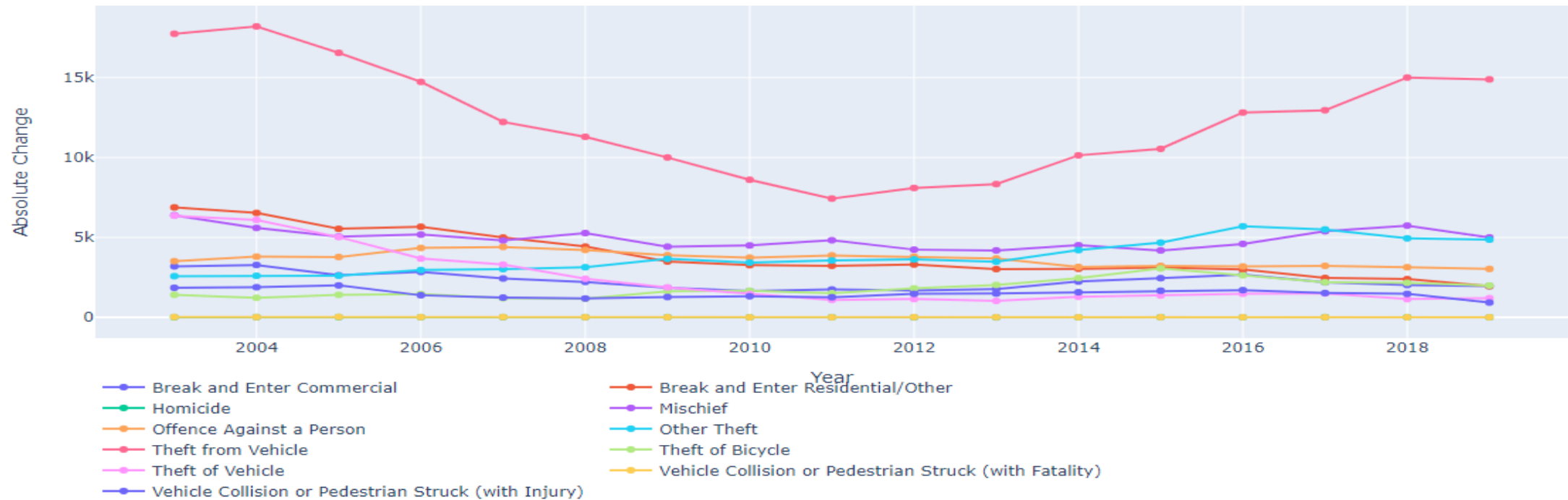
```
# gather yearly count of crime in Vancouver
for current_type in crime_types:
    current_crime = crime_records[crime_records["TYPE"]==current_type]
    current_crime_counts = current_crime["YEAR"].value_counts(sort=False)
    current_crime_index = current_crime_counts.index.tolist()
    current_crime_index, current_crime_counts = zip(*sorted(zip(current_crime_index, current_crime_counts)))
    crime_count_by_year[current_type] = current_crime_counts
crime_count_by_year
```

t[110]:

	year	Break and Enter Commercial	Break and Enter Residential/Other	Homicide	Mischief	Offence Against a Person	Other Theft	Theft from Vehicle	Theft of Bicycle	Theft of Vehicle	Vehicle Collision or Pedestrian Struck (with Fatality)	Vehicle Collision or Pedestrian Struck (with Injury)
0	2003	3197	6883	18	6391	3518	2582	17744	1418	6361	25	1856
1	2004	3283	6538	22	5601	3802	2605	18204	1230	6102	22	1892
2	2005	2639	5542	22	5062	3779	2611	16554	1416	5031	33	2003
3	2006	2844	5674	17	5184	4345	2966	14734	1467	3682	20	1388
4	2007	2436	4996	19	4810	4413	3024	12226	1203	3305	24	1239
5	2008	2224	4432	18	5276	4225	3142	11298	1176	2420	17	1186
6	2009	1858	3497	18	4429	3893	3662	10007	1641	1882	14	1278

Let's visualize the change of crime over the years across all the types of crime

Crimes Over the Years in Vancouver by Type



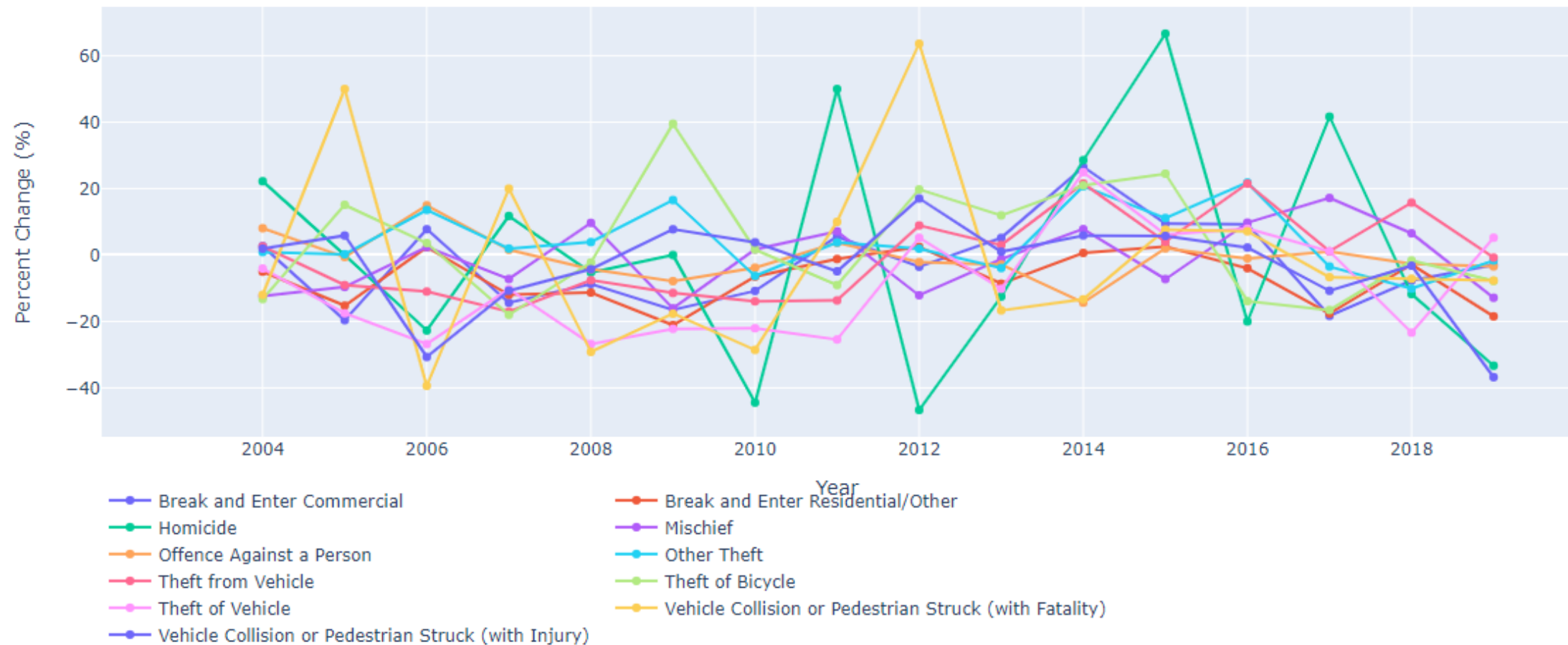
Percent Change of Crimes Over the Years

```
: # calculate percent change of crime for each type of crime
crime_count_by_year_percent_change = crime_count_by_year[crime_count_by_year.columns[1:]].pct_change() * 100
crime_count_by_year_percent_change["year"] = year_labels
crime_count_by_year_percent_change
```

Out[11]:

	Break and Enter Commercial	Break and Enter Residential/Other	Homicide	Mischief	Offence Against a Person	Other Theft	Theft from Vehicle	Theft of Bicycle	Theft of Vehicle	Vehicle Collision or Pedestrian Struck (with Fatality)	Vehicle Collision or Pedestrian Struck (with Injury)	year
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2003
1	2.690022	-5.012349	22.222222	-12.361133	8.072769	0.890782	2.592426	-13.258110	-4.071687	-12.000000	1.939655	2004
2	-19.616205	-15.234017	0.000000	-9.623282	-0.604945	0.230326	-9.063942	15.121951	-17.551622	50.000000	5.866808	2005
3	7.768094	2.381812	-22.727273	2.410115	14.977507	13.596323	-10.994322	3.601695	-26.813755	-39.393939	-30.703944	2006
4	-14.345992	-11.949242	11.764706	-7.214506	1.565017	1.955496	-17.021854	-17.995910	-10.239001	20.000000	-10.734870	2007
5	-8.702791	-11.289031	-5.263158	9.688150	-4.260140	3.902116	-7.590381	-2.244389	-26.777610	-29.166667	-4.277643	2008
6	-16.456835	-21.096570	0.000000	-16.053829	-7.857988	16.549968	-11.426801	39.540816	-22.231405	-17.647059	7.757167	2009

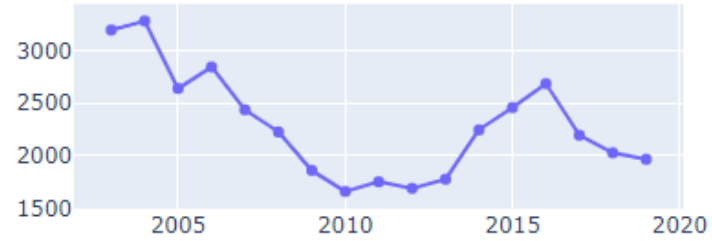
Percent Change of Crimes Over the Years in Vancouver by Type



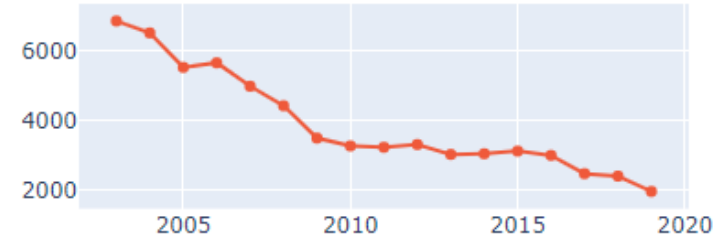
Using the percentage equals 0 as your reference, you will observe that the crimes of "Vehicle Collision or Pedestrian Struck (with Fatality)", "Theft of Bicycle", and "Homicide" has relative more positive percentage changes through the years

Crimes in Vancouver Over the Years

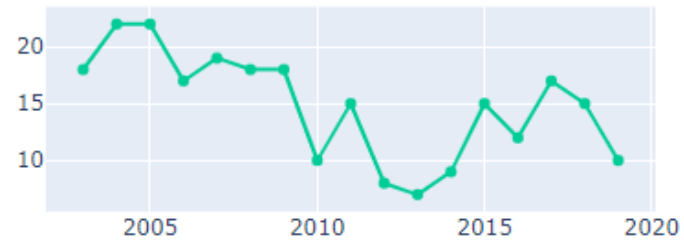
1. Break and Enter Commercial



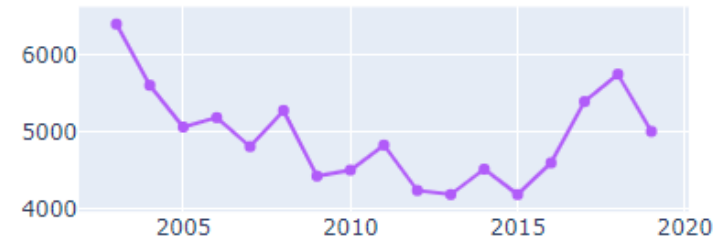
2. Break and Enter Residential/Other



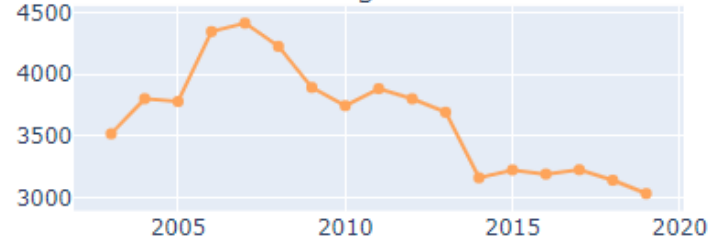
3. Homicide



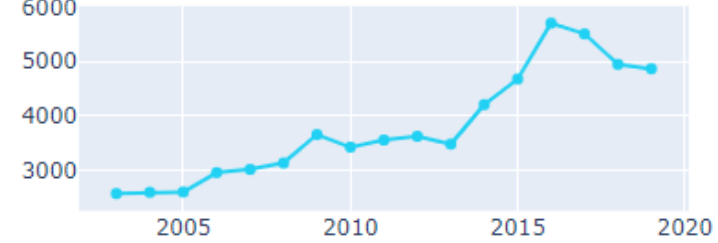
4. Mischief



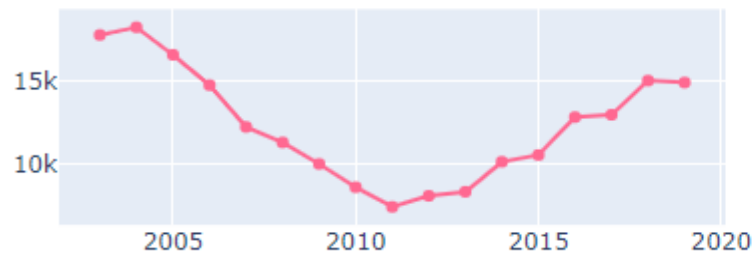
5. Offence Against a Person



6. Other Theft



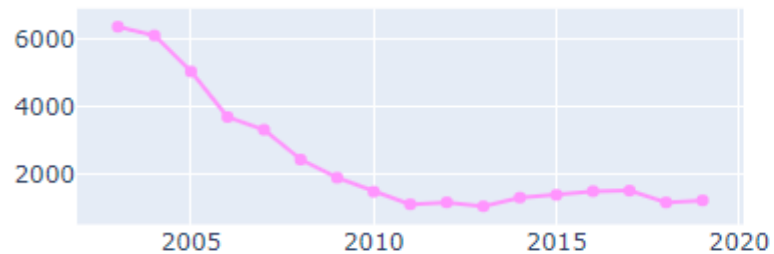
7. Theft from Vehicle



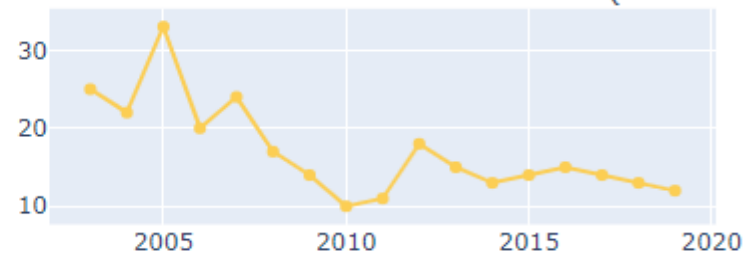
8. Theft of Bicycle



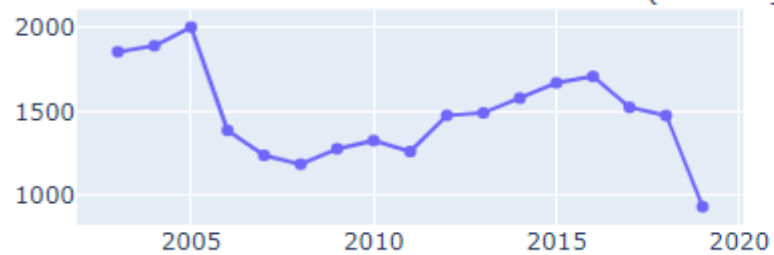
9. Theft of Vehicle



10. Vehicle Collision or Pedestrian Struck (with Fatality)



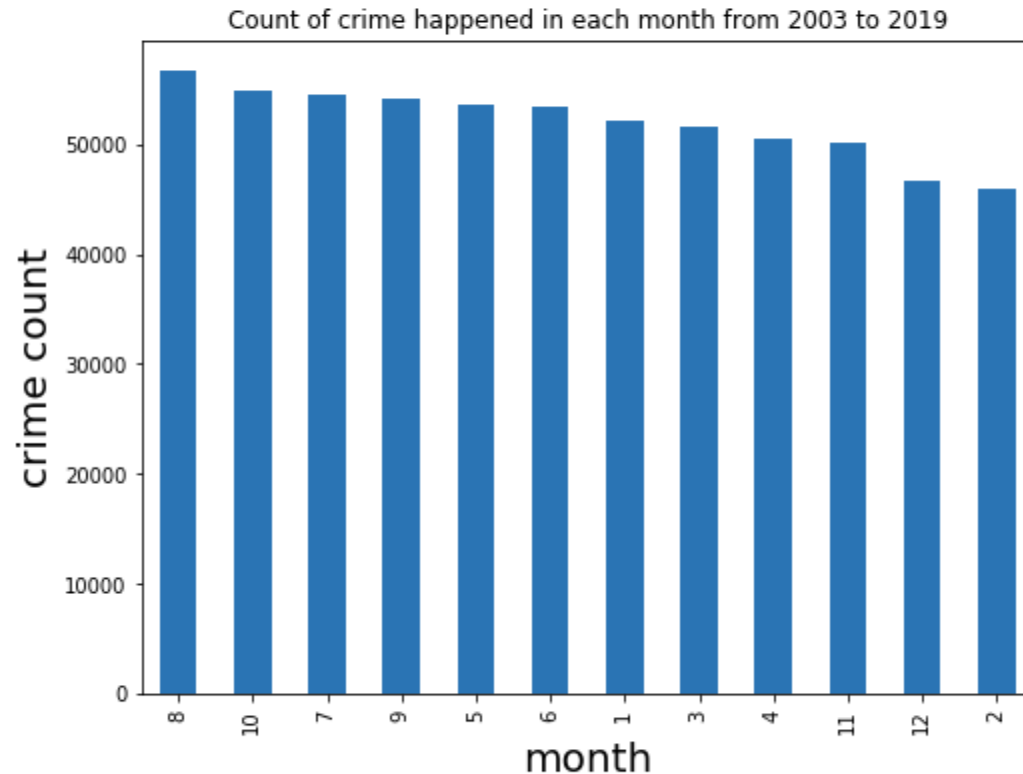
11. Vehicle Collision or Pedestrian Struck (with Injury)



6.0 What months have the most of the crimes recorded?

```
In [14]: nameplot = crime_records["MONTH"].value_counts().plot.bar(title='Count of crime happened in each month from 2003 to 2019', figsize=(8,6))
nameplot.set_xlabel('month',size=20)
nameplot.set_ylabel('crime count',size=20)
```

Out[14]: Text(0, 0.5, 'crime count')



Overall, "Theft from Vehicle" remains to be the main type of crime happened in Vancouver in 2018 through the months. The 2nd and the 3rd most common crimes are "Mischief" and "Offence Against a Person" in 2018.

It appears that the months of summer time has more crime recorded than other months.

7. Crime investigation based on location data

We have 4 columns describe the locations of the given crime records. They are: HUNDRED_BLOCK, NEIGHBOURHOOD, X, and Y. Let's explore these columns now.

7.0 How many unique blocks do we have?

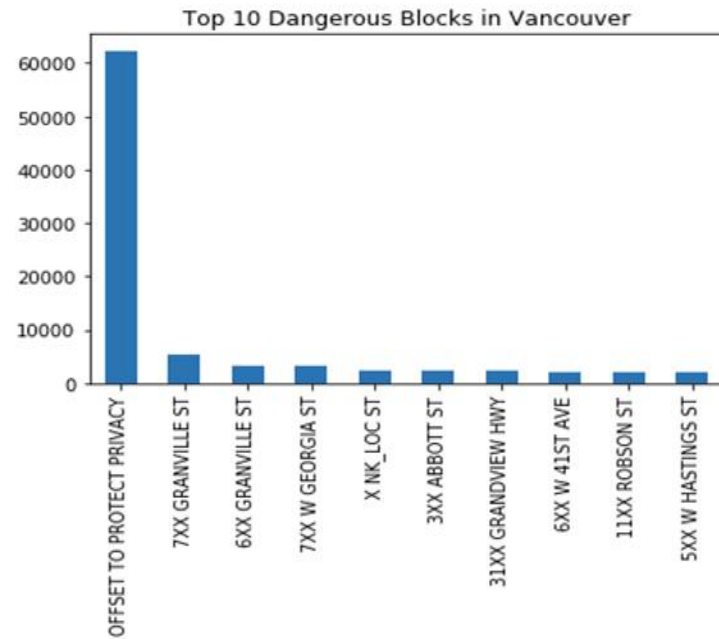
```
[17]: len(crime_records["HUNDRED_BLOCK"].unique())
```

```
Out[17]: 22181
```

That's a lot of missing blocks. Let explore it.

```
: crime_records["HUNDRED_BLOCK"].value_counts().plot.bar(  
    title='Top 10 Dangerous Blocks in Vancouver')  
nameplot.set_xlabel('block name',size=20)  
nameplot.set_ylabel('count',size=20)
```

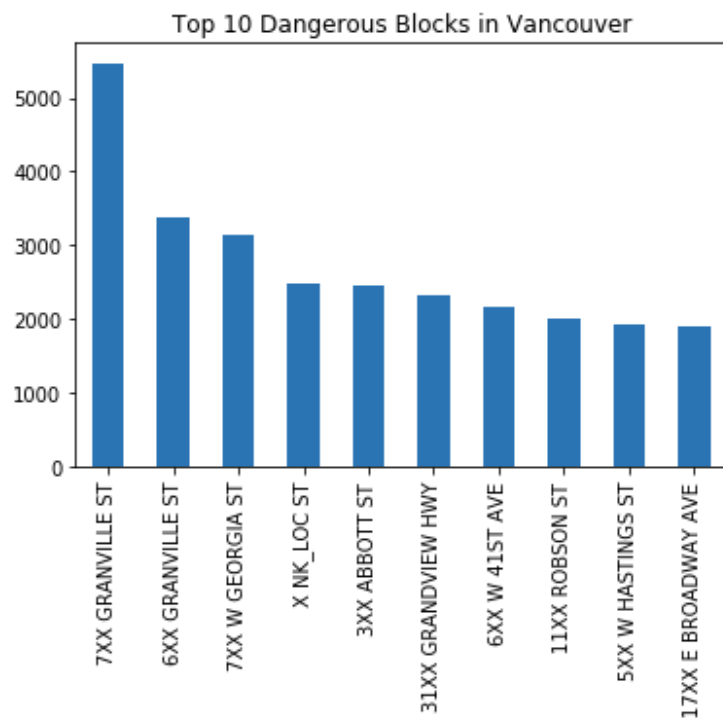
```
Out[18]: Text(26.200000000000003, 0.5, 'count')
```




To protect privacy, police labeled about 10% of the data to be "OFFSET TO PROTECT PRIVACY". Let's graph the top 10 crime-populated block without the "OFFSET TO PROTECT PRIVACY" rows

```
In [19]: crime_records["HUNDRED_BLOCK"].value_counts()[1:11].plot.bar(
        title='Top 10 Dangerous Blocks in Vancouver')
nameplot.set_xlabel('block name',size=20)
nameplot.set_ylabel('count',size=20)
```


Out[19]: Text(26.200000000000003, 0.5, 'count')



Granville street seems to be very dangerous from the above visualization. Well, 60K of 609K rows of data are offsetted to protect privacy. We won't be able to use those data because the HUNDRED_BLOCK, NEIGHBOURHOOD data, X, and Y data will be inaccurate to describe the crime happened in Vancouver from 2003 to 2019. Let's create a new dataframe with usable data that's not labeled as "OFFSET TO PROTECT PRIVACY"

```
:  crime_records_without_offset = crime_records[crime_records["HUNDRED_BLOCK"] != "OFFSET TO PROTECT PRIVACY"]
crime_records_without_offset.shape
```

Out[20]: (561708, 10)

```
:  crime_records_without_offset.head()
```

Out[21]:

	TYPE	YEAR	MONTH	DAY	HOUR	MINUTE	HUNDRED_BLOCK	NEIGHBOURHOOD	X	Y
0	Break and Enter Commercial	2012	12	14	8	52	NaN	Oakridge	491285.000000	5.453433e+06
1	Break and Enter Commercial	2019	3	7	2	6	10XX SITKA SQ	Fairview	490612.964805	5.457110e+06
2	Break and Enter Commercial	2019	8	27	4	12	10XX ALBERNI ST	West End	491007.779775	5.459174e+06
3	Break and Enter Commercial	2014	8	8	5	13	10XX ALBERNI ST	West End	491015.943352	5.459166e+06
4	Break and Enter Commercial	2005	11	14	3	9	10XX ALBERNI ST	West End	491021.385727	5.459161e+06

7.1 Classification model to predict the missing NEIGHBOURHOOD column's data.

We are not interested in block number anymore. Let's remove the number by stripping the first block before the first white space in the HUNDRED_BLOCK data

```
2]: # drop missing row in crime record where only 0.002% of HUNDRED_BLOCK row are missing
crime_records_without_offset = crime_records_without_offset[pd.notnull(crime_records_without_offset["HUNDRED_BLOCK"])]
```

```
3]: block_names = crime_records_without_offset["HUNDRED_BLOCK"]
block_names.isnull().sum() # make sure we don't have missing row for HUNDRED_BLOCK now
```

Out[23]: 0

If the street number's first character is a number and the street's name has more than 2 word, I remove the street number from it. Reason:

1. The street name's beginning can a letter which is not a street number. I don't want the location information lost through this preprocessing process.
2. The street name can be 2 words only such as Hasting Street. I don't want the first word to be lost and let the dataset only contains the "street" part.

```
1]: def remove_street_number(street_name):
    # if the beginning is a digit and the street name contains more than 2 words
    if street_name[0].isdigit() and len(street_name.split(" ")) > 2:
        _, _, tail = street_name.partition(" ")
        return tail
    return street_name

# apply the above function to remove the street number in each row
crime_records_without_offset["HUNDRED_BLOCK"] = block_names.apply(remove_street_number)
```

```
[25]: crime_records_without_offset["HUNDRED_BLOCK"].head()
```

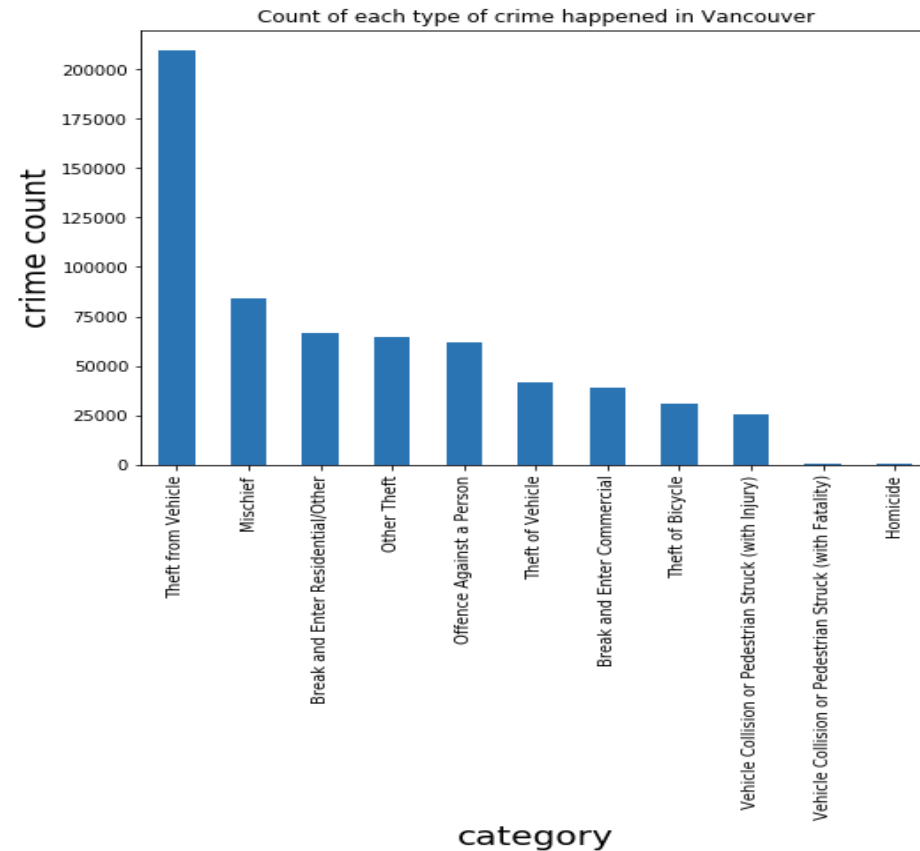
```
Out[25]: 1      SITKA SQ
        2      ALBERNI ST
        3      ALBERNI ST
        4      ALBERNI ST
        5      ALBERNI ST
        Name: HUNDRED_BLOCK, dtype: object
```

We have successfully removed the block number. Now, let's associate the block name to the neighbourhood data to create a dictionary to check whether the missing neighbourhood data's street has association with existing block already.

```
[26]: street_with_missing_neighbourhood = crime_records_without_offset[crime_records_without_offset["NEIGHBOURHOOD"].isnull()]["HUNDRED_BLOCK"].value_counts()
      street_with_missing_neighbourhood
```

```
Out[26]: X NK_LOC ST                2488
        GRANVILLE STREET BRDG      4
        BURRARD STREET BRDG         3
        SW MARINE DR / TAMATH CRES   2
        BLOCK BOUNDARY RD           2
        GRANVILLE ST BRIDGE        2
        CAMBIE ST BRIDGE             2
        CAMBIE STREET BRDG           2
        X NK_LOC ST "SQUAMISH"       2
        E HASTINGS ST / CASSIAR ST   2
```

Distribution of each type of crime



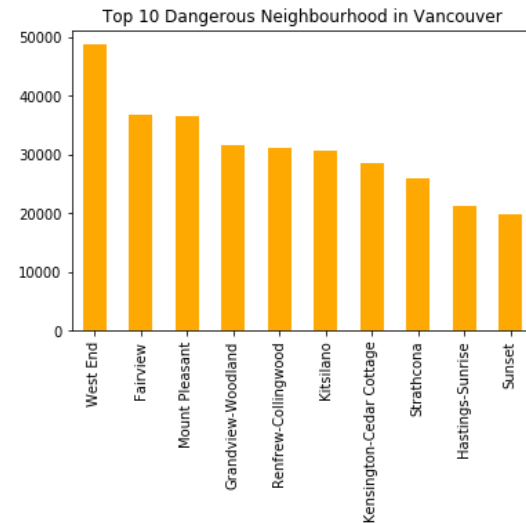
It appears that the most common type of crime is associated with vehicle in Vancouver: Theft from Vehicle, Theft of Bicycle and Vehicle Collision. Since 2003, there are 209609 crime categorized as "Theft from Vehicle".

Top 10 most dangerous neighbourhood in Vancouver

8. Top 10 most dangerous neighbourhood in Vancouver

```
[38]: crime_records["NEIGHBOURHOOD"].value_counts()[1:11].plot.bar(
      title='Top 10 Dangerous Neighbourhood in Vancouver', color="orange")
      nameplot.set_xlabel('neighbourhood name',size=20)
      nameplot.set_ylabel('count',size=20)
```

Out[38]: Text(26.200000000000003, 0.5, 'count')



It turns out the most crime populated neighbourhood is West End. Let's use a treemap to visualize crimes in West End

Crimes in West End

```
In [39]: # crimes in west end
west_end_crimes = crime_records_without_offset[crime_records_without_offset["NEIGHBOURHOOD"]=="West End"]
values = west_end_crimes["TYPE"].value_counts()
labels = values.index.tolist()
labels
```

```
Out[39]: ['Theft from Vehicle',
          'Other Theft',
          'Mischief',
          'Break and Enter Residential/Other',
          'Theft of Bicycle',
          'Break and Enter Commercial',
          'Theft of Vehicle',
          'Vehicle Collision or Pedestrian Struck (with Injury)',
          'Vehicle Collision or Pedestrian Struck (with Fatality)']
```

```
In [40]: values
```

```
Out[40]: Theft from Vehicle      20401
          Other Theft            7256
          Mischief               6418
          Break and Enter Residential/Other  3649
          Theft of Bicycle       3449
          Break and Enter Commercial  3304
          Theft of Vehicle       2881
          Vehicle Collision or Pedestrian Struck (with Injury)  1347
          Vehicle Collision or Pedestrian Struck (with Fatality)  8
          Name: TYPE, dtype: int64
```

8 of 1328 collision resulted fatality. The "or" really makes the data unclear and they didn't state the total death count. Let's combine the last two element of the list together and reduce the string length for better visualization.

```
In [41]: labels.pop()
labels[-1] = "Veh. Collis/ Ped Struck"
labels
```

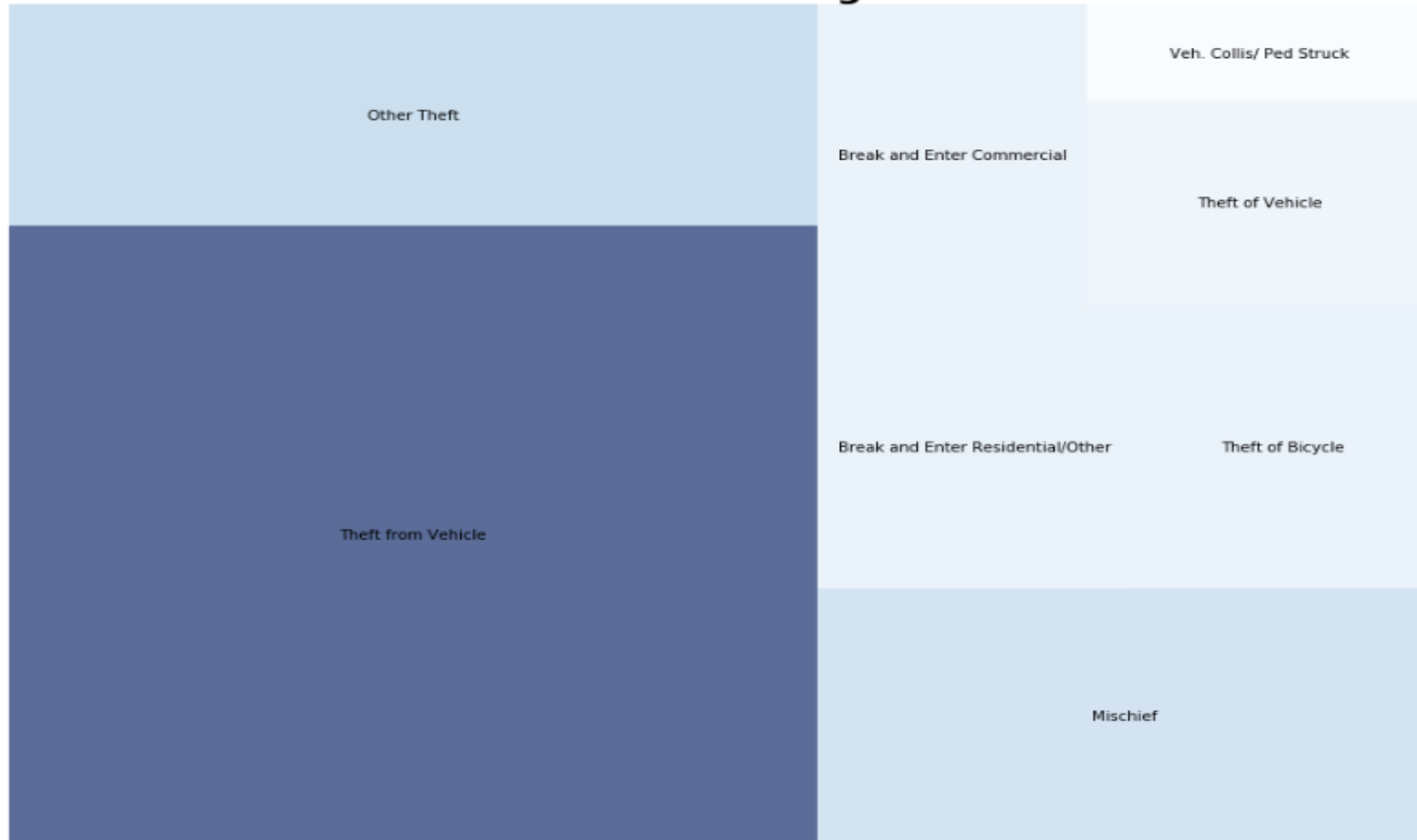
```
Out[41]: ['Theft from Vehicle',
          'Other Theft',
          'Mischief',
          'Break and Enter Residential/Other',
          'Theft of Bicycle',
          'Break and Enter Commercial',
          'Theft of Vehicle',
          'Veh. Collis/ Ped Struck']
```

```
In [42]: values[-2] += values[-1]
values = values[:len(values)-1]
values
```

```
Out[42]: Theft from Vehicle      20401
          Other Theft            7256
          Mischief               6418
          Break and Enter Residential/Other  3649
          Theft of Bicycle       3449
          Break and Enter Commercial  3304
          Theft of Vehicle       2881
          Vehicle Collision or Pedestrian Struck (with Injury)  1355
          Name: TYPE, dtype: int64
```

Requirement already satisfied: squarify in /opt/conda/envs/Python36/lib/python3.6/site-packages (0.4.3)

Crimes in West End Neighbourhood



Let's take a look the x and y values of the crime happened in West End neighbourhood

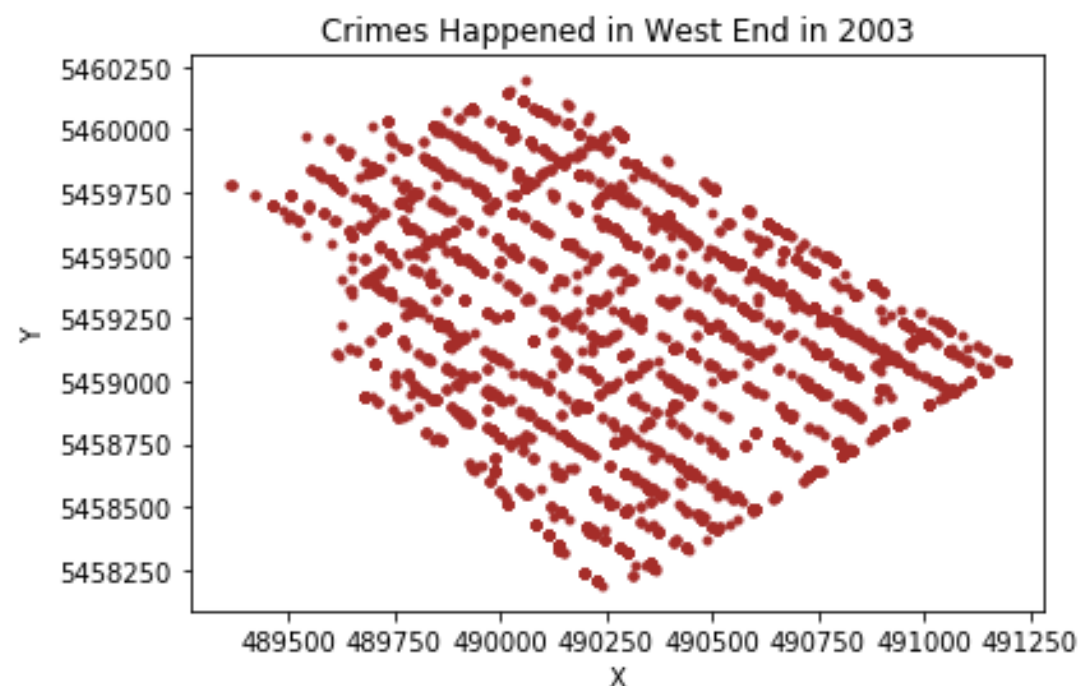
Let's take a look the x and y values of the crime happened in West End neighbourhood

```
In [145]: west_end_X = crime_records_without_offset[crime_records_without_offset["NEIGHBOURHOOD"]=="West End"]["X"]  
west_end_Y = crime_records_without_offset[crime_records_without_offset["NEIGHBOURHOOD"]=="West End"]["Y"]  
plt.scatter(west_end_X,west_end_Y, color="red", marker=".")  
plt.xlabel("X")  
plt.ylabel("Y")  
plt.title('Crimes Happened in West End from 2003 to 2019')  
plt.show()
```

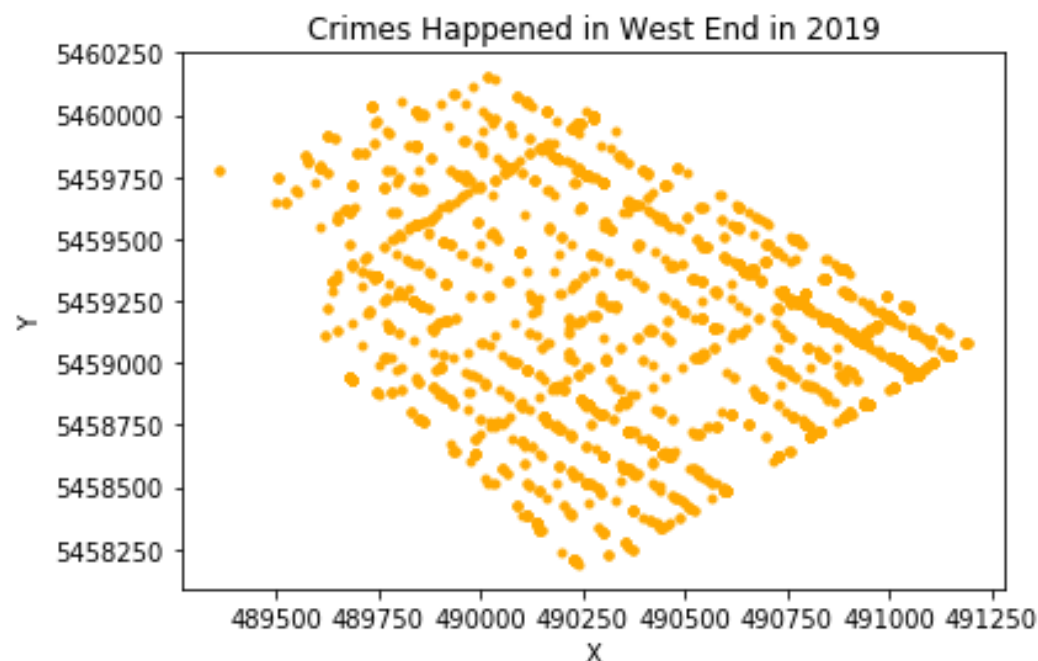


The scatter plot is actually the shape of the West End neighbourhood. Crimes happened on almost every street of West End from 2003 to 2019.

```
In [146]: cimes_2003 = crime_records_without_offset[crime_records_without_offset["YEAR"]==2003]
west_end_X_2003 = cimes_2003[cimes_2003["NEIGHBOURHOOD"]=="West End"]["X"]
west_end_Y_2003 = cimes_2003[cimes_2003["NEIGHBOURHOOD"]=="West End"]["Y"]
plt.scatter(west_end_X_2003,west_end_Y_2003, color="brown", marker=".")
plt.xlabel("X")
plt.ylabel("Y")
plt.title('Crimes Happened in West End in 2003')
plt.show()
```



```
In [147]: cimes_2019 = crime_records_without_offset[crime_records_without_offset["YEAR"]==2019]
west_end_X_2019 = cimes_2019[cimes_2019["NEIGHBOURHOOD"]=="West End"]["X"]
west_end_Y_2019 = cimes_2019[cimes_2019["NEIGHBOURHOOD"]=="West End"]["Y"]
plt.scatter(west_end_X_2019,west_end_Y_2019, color="orange", marker=".")
plt.xlabel("X")
plt.ylabel("Y")
plt.title('Crimes Happened in West End in 2019')
plt.show()
```



Comparing the X, and Y of records from West End, the above plots show the crimes scattered through the neighbourhood and depicted almost the full street map of the West End neighbourhood.

9. Vancouver Crimes Heatmap with Folium

Function to convert UTM to Longitude and Latitude system.

```
In [148]: import math

def utmToLatLong(utmEasting, utmNorthing, utmZone):
    eastingOffset = 500000.0
    northingOffset = 10000000.0
    k0 = 0.9996
    equatorialRadius = 6378137.0
    eccSquared = 0.006694380023
    eccPrimeSquared = eccSquared / (1 - eccSquared)
    e1 = (1 - math.sqrt(1 - eccSquared)) / (1 + math.sqrt(1 - eccSquared));
    rad2deg = 180.0/math.pi

    # Casts input from string to floats or ints
    # Removes 500,000 metre offset for Longitude
    xUTM = float(utmEasting) - eastingOffset
    yUTM = float(utmNorthing)
    zoneNumber = int(utmZone)

    # Finds the origin longitude for the zone
    lonOrigin = (zoneNumber - 1) * 6 - 180 + 3 # +3 puts in zone centre

    M = yUTM / k0 #This finds the meridional arc
    mu = M / (equatorialRadius * (1- eccSquared / 4 - 3 * eccSquared * eccSquared / 64 -5 * eccSquared * e
ccSquared * eccSquared /256))
```

```

# Calculates the footprint latitude
phi1Rad = mu + (3 * e1 / 2 - 27 * e1 * e1 * e1 / 32) * math.sin(2*mu) + ( 21 * e1 * e1 / 16 - 55 * e1 *
e1 * e1 * e1 / 32) * math.sin( 4 * mu) + (151 * e1 * e1 * e1 / 96) * math.sin(6 * mu)
phi1 = phi1Rad * rad2deg

# Variables for conversion equations
N1 = equatorialRadius / math.sqrt( 1 - eccSquared * math.sin(phi1Rad) * math.sin(phi1Rad))
T1 = math.tan(phi1Rad) * math.tan(phi1Rad)
C1 = eccPrimeSquared * math.cos(phi1Rad) * math.cos(phi1Rad)
R1 = equatorialRadius * (1 - eccSquared) / math.pow(1 - eccSquared * math.sin(phi1Rad) * math.sin(phi1
Rad), 1.5)
D = xUTM / (N1 * k0)

# Calculate latitude, in decimal degrees
lat = phi1Rad - ( N1 * math.tan(phi1Rad) / R1) * (D * D / 2 - (5 + 3 * T1 + 10 * C1 - 4 * C1 * C1 - 9
* eccPrimeSquared) * D * D * D * D / 24 + (61 + 90 * T1 + 298 * C1 + 45 * T1 * T1 - 252 * eccPrimeSquared
- 3 * C1 * C1) * D * D * D * D * D * D / 720)
lat = lat * rad2deg

# Calculate longitude, in decimal degrees
lon = (D - (1 + 2 * T1 + C1) * D * D * D / 6 + (5 - 2 * C1 + 28 * T1 - 3 * C1 * C1 + 8 * eccPrimeSquar
ed + 24 * T1 * T1) * D * D * D * D * D * D / 120) / math.cos(phi1Rad)
lon = lonOrigin + lon * rad2deg

return lon, lat

```

In [149]: `crime_records_without_offset = crime_records_without_offset.reset_index(drop=True)`

In [150]: `crimes_vancouver_X = crime_records_without_offset["X"]
crimes_vancouver_Y = crime_records_without_offset["Y"]
longitude = []
latitude = []
for i in range(len(crimes_vancouver_X)):
 lon, lat = utmToLatLong(crimes_vancouver_X[i], crimes_vancouver_Y[i], 10)
 longitude.append(lon)
 latitude.append(lat)`


```
In [149]: crime_records_without_offset = crime_records_without_offset.reset_index(drop=True)
```

```
In [150]: crimes_vancouver_X = crime_records_without_offset["X"]
crimes_vancouver_Y = crime_records_without_offset["Y"]
longitude = []
latitude = []
for i in range(len(crimes_vancouver_X)):
    lon, lat = utmToLatLong(crimes_vancouver_X[i], crimes_vancouver_Y[i], 10)
    longitude.append(lon)
    latitude.append(lat)
```

```
In [151]: crime_records_without_offset["LONGITUDE"] = longitude
crime_records_without_offset["LATITUDE"] = latitude
crime_records_without_offset.head()
```

Out[151]:

	TYPE	YEAR	MONTH	DAY	HOUR	MINUTE	HUNDRED_BLOCK	NEIGHBOURHOOD	X	Y
0	Break and Enter Commercial	2019	3	7	2	6	SITKA SQ	Fairview	490612.964805	5.457110e+0
1	Break and Enter Commercial	2019	8	27	4	12	ALBERNI ST	West End	491007.779775	5.459174e+0
2	Break and Enter Commercial	2014	8	8	5	13	ALBERNI ST	West End	491015.943352	5.459166e+0
3	Break and Enter Commercial	2005	11	14	3	9	ALBERNI ST	West End	491021.385727	5.459161e+0
4	Break and Enter Commercial	2006	5	21	4	50	ALBERNI ST	West End	491021.385727	5.459161e+0

We have the latitude and longitude value now! let's find out where should the neighbourhood be for the record with "S / L 300 BLK E HASTINGS ST" HUNDRED_BLOCK value.

```
In [152]: missing_neighbourhood = crime_records_without_offset[crime_records_without_offset["HUNDRED_BLOCK"]=="S / L  
300 BLK E HASTINGS ST"]  
missing_neighbourhood_Lon = missing_neighbourhood["LONGTITUDE"]  
missing_neighbourhood_Lat = missing_neighbourhood["LATITUDE"]  
lat_lon_pair = list(zip(missing_neighbourhood_Lat, missing_neighbourhood_Lon))  
print("Latitude and Longitude of the missing neighbourhood row")  
lat_lon_pair
```

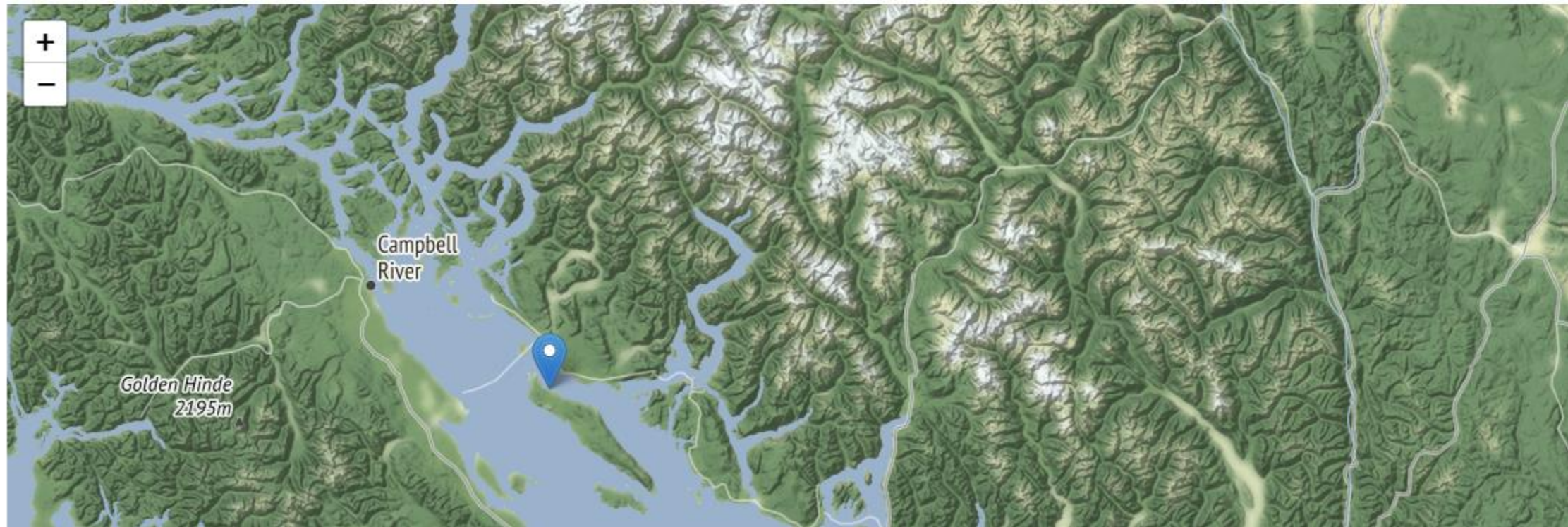
Latitude and Longitude of the missing neighbourhood row

```
Out[152]: [(49.75531925010842, -124.54975724974244)]
```


```
In [53]: print("Where is the missing neighbourhood row located?")
missing_neighbourhood_vis = folium.Map(location= [49.4985, -123.8159],
                                           tiles = "Stamen Terrain",
                                           zoom_start = 9)
popups = ["Missing Neighbourhood Record"]
MarkerCluster(lat_lon_pair, popups=popups).add_to(missing_neighbourhood_vis)
missing_neighbourhood_vis
```

Where is the missing neighbourhood row located?


Out[53]:



It turns out that's far away from the city of Vancouver. Let's consider it as an outlier and remove it from the dataset.

```
In [54]:  # not include the row in our analysis
crime_records_without_offset = crime_records_without_offset[crime_records_without_offset["HUNDRED_BLOCK"]!="S / L 300 BLK E HASTINGS ST"]
crime_records_without_offset = crime_records_without_offset.reset_index(drop=True) # reset index
missing_neighbourhood_val_count() # checking missing neighbourhood value
```


```
Out[54]: CAMBIE ST BRIDGE                2
         SE MARINE DR / 8198 FRASER ST    1
         RENFREW ST / MCGILL ST          1
         .....
         .
```


```
In [55]:  missing_value_describe(crime_records_without_offset)
```

Number of columns with missing values: 5

Missing percentage (descending):

LATITUDE	0.021459
LONGITUDE	0.021459
Y	0.021459
X	0.021459
NEIGHBOURHOOD	0.017883
dtype:	float64

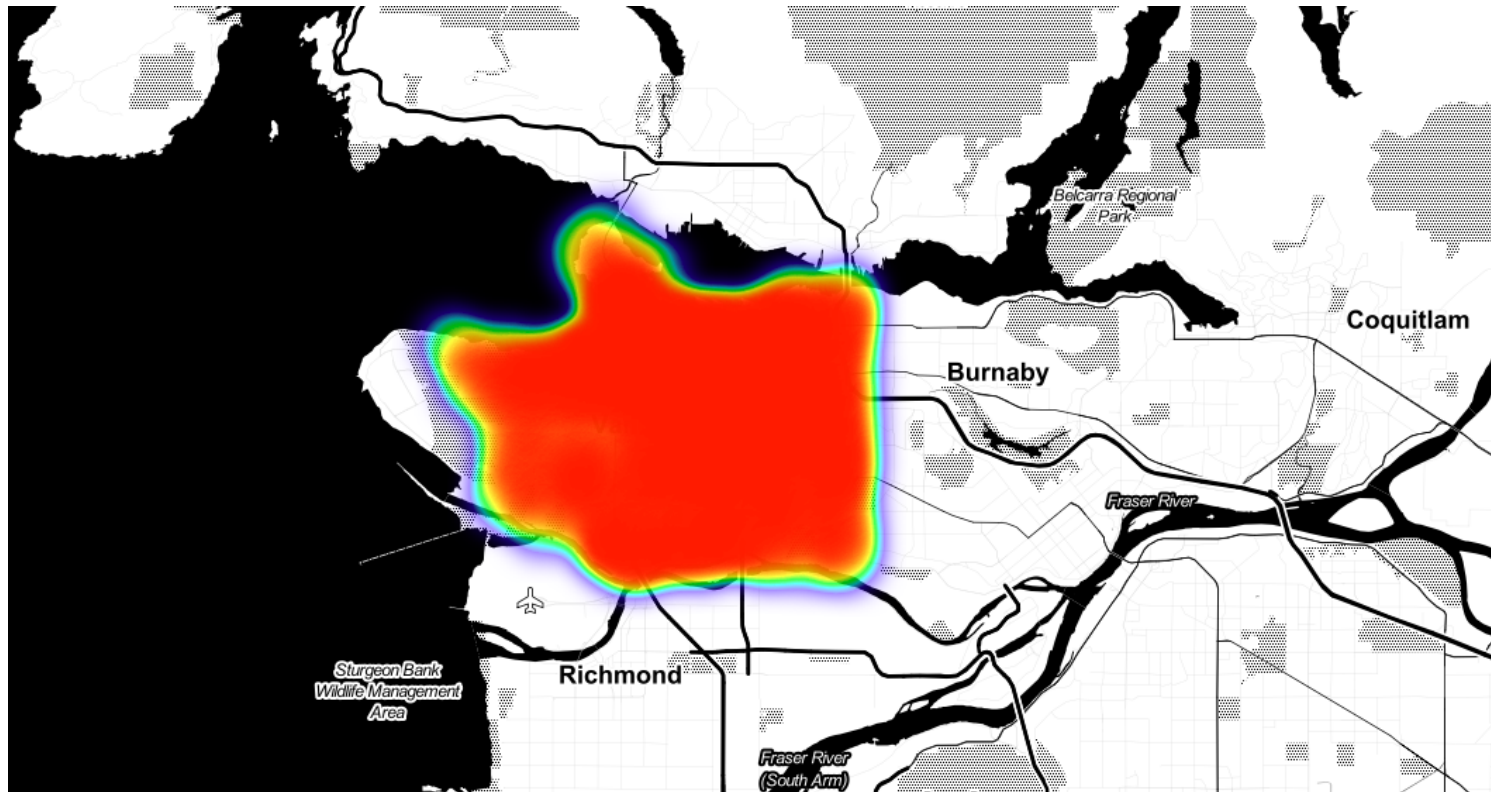
```
In [56]:  crime_records_without_offset = crime_records_without_offset.dropna()
```

```
In [57]:  missing_value_describe(crime_records_without_offset)
```


Number of columns with missing values: 0
No missing data!!!


We don't have row with missing neighbourhood value now!

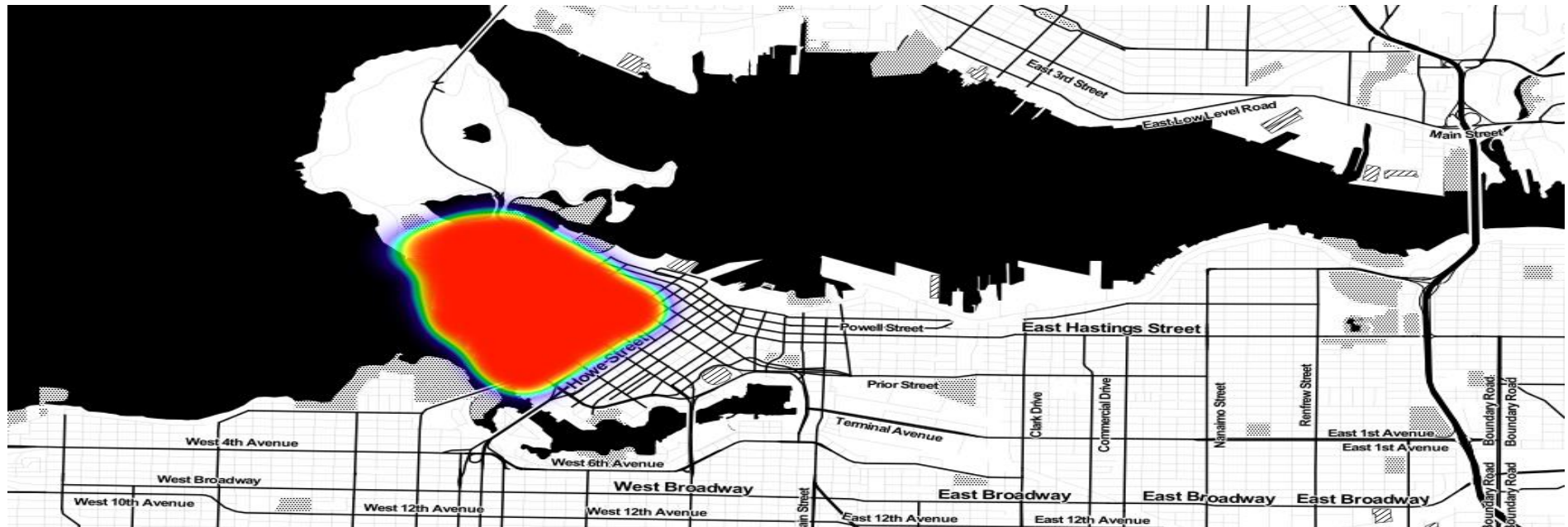
Crime Heatmap of 2019



It's very crime populated. Let's zoom in to see to the crimes in West End (2019).

```
In [60]:  # return a list of latitude and longitude data based on specified year and crime type
def get_record_locations_by_year_and_neighbourhood(year, neighbourhood):
    crimes = crime_records_without_offset[crime_records_without_offset["YEAR"]==year]
    lat = crimes[crimes["NEIGHBOURHOOD"]==neighbourhood]["LATITUDE"]
    lon = crimes[crimes["NEIGHBOURHOOD"]==neighbourhood]["LONGITUDE"]
    return list(zip(lat, lon))
```

```
In [61]:  latlon_west_end_2019 = get_record_locations_by_year_and_neighbourhood(2019, "West End")
crime_heatmap = folium.Map(location= [49.28416, -123.13150],
                                tiles = "Stamen Toner",
                                zoom_start = 15)
HeatMap(latlon_west_end_2019).add_to(crime_heatmap)
crime_heatmap
```



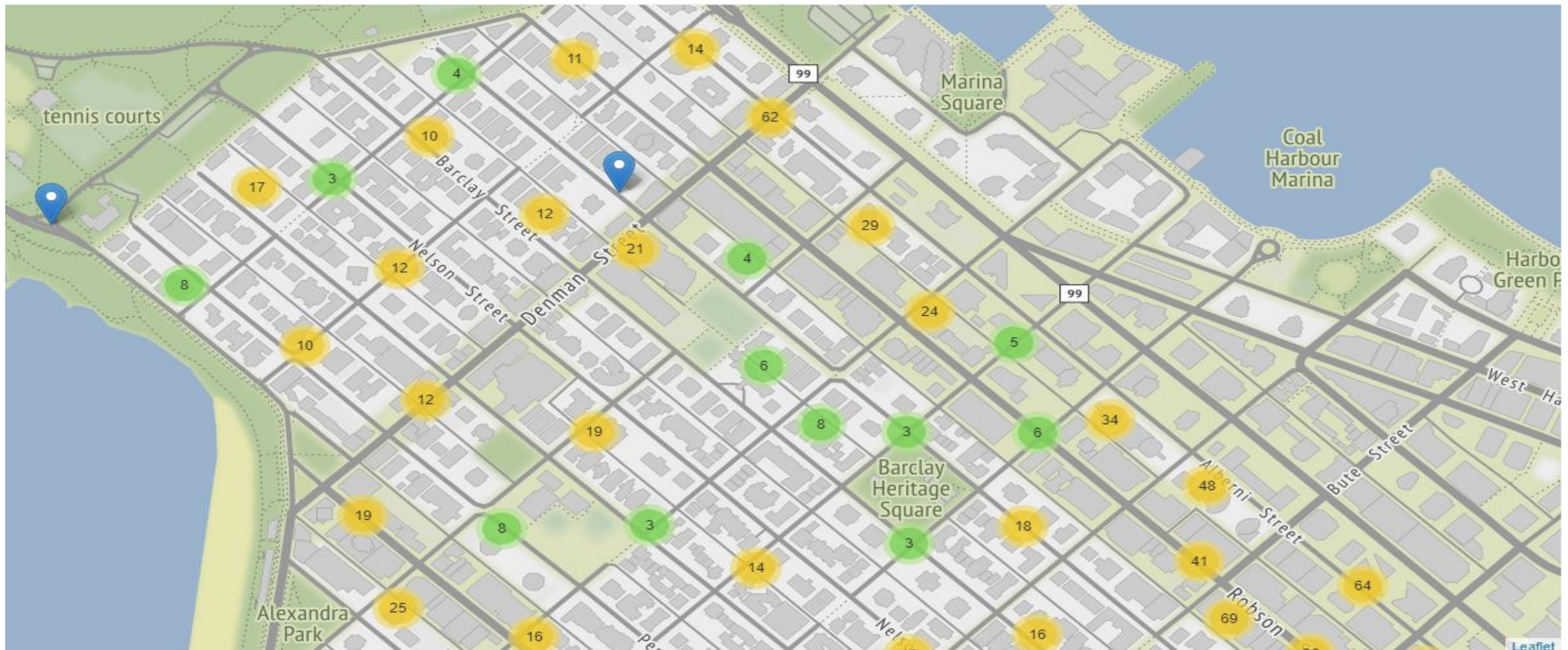
Let's take look at the "Theft from Vehicle" crimes in West End (2019)

```
In [63]: print("West End 'Theft from Vehicle' Crimes Heatmap in 2019")
latlon_west_end_2019 = get_record_locations_by_year_type_neighbourhood(2019, "Theft from Vehicle", "West End")
west_end_TFC = folium.Map(location= [49.28416, -123.13150],
                             tiles = "Stamen Terrain",
                             zoom_start = 15)
popups = ["Theft from Vehicle" for _ in range(len(latlon_west_end_2019))] # Popups texts are simple numbers.
MarkerCluster(latlon_west_end_2019, popups=popups).add_to(west_end_TFC)
west_end_TFC
```

West End 'Theft from Vehicle' Crimes Heatmap in 2019



West End “Theft from Vehicle”

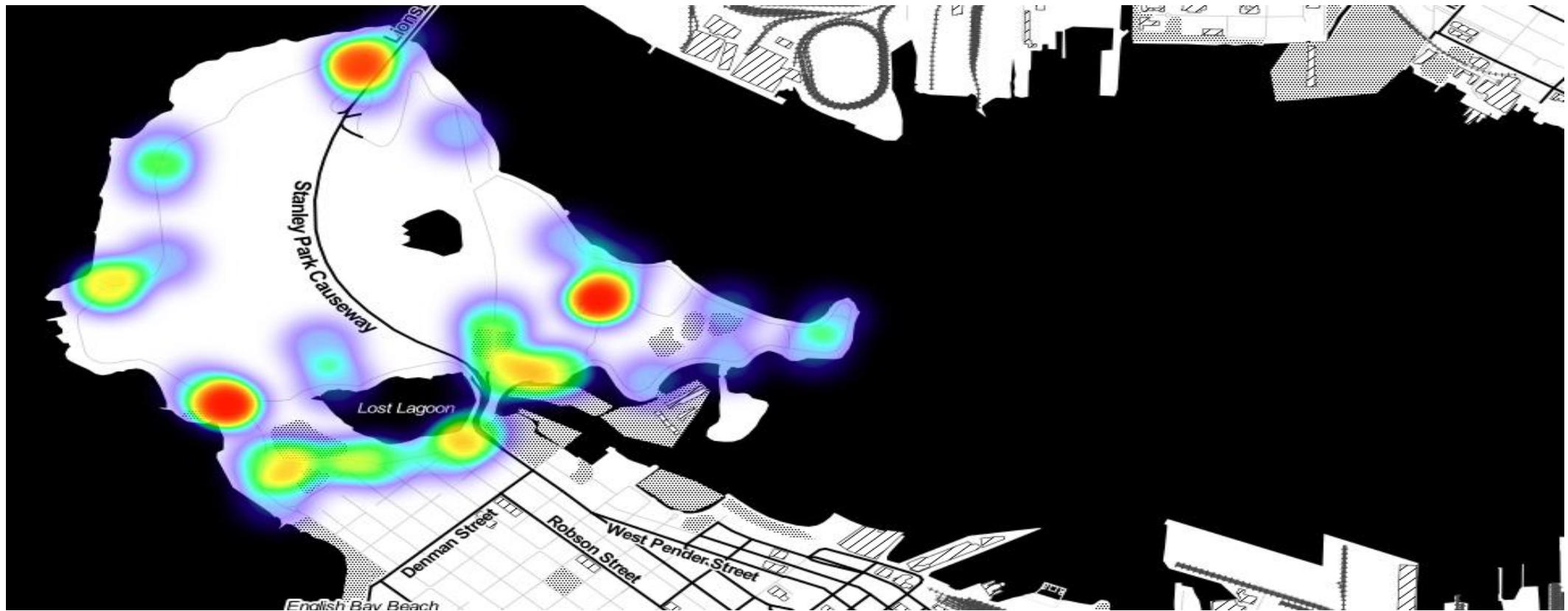


Crimes happened in Stanley Park in 2019 so far

```
In [64]: ▶ print("Crimes happened in Stanley Park in 2019 so far")
latlon_stanely_park_2019 = get_record_locations_by_year_and_neighbourhood(2019, "Stanley Park")
crime_heatmap = folium.Map(location=[49.30330, -123.14635],
                             tiles = "Stamen Toner",
                             zoom_start = 15)
HeatMap(latlon_stanely_park_2019).add_to(crime_heatmap)
crime_heatmap
```

Crimes happened in Stanley Park in 2019 so far

Crimes Happened in Stanley Park



Crimes Happened in Stanley Park



Crimes Happened in Stanley Park




Vancouver crime data comments

Through this exploratory analysis of the Vancouver crime data:

1. I found that the most common crime in Vancouver is related to vehicle such "Theft of Vehicle", "Theft of Bicycle" and "Vehicle Collision". Definitely, watch out for your bikes, motorcycle, and cars.
2. West End neighbourhood remain the most crime populated area from 2003 to 2019. The location record of crimes committed in West End can be used to draw a clear street map of West End.
3. The occurrences of crimes are quite evenly distributed if crimes are counted by day or counted by month. However, crime occurs more in after work hours and midnight in Vancouver.

11. Exploring venues in the top 10 most dangerous neighbourhood in Vancouver using Foursquare API

11.0 Cluster

In [67]:  crime_records_without_offset.head()

Out[67]:

	TYPE	YEAR	MONTH	DAY	HOUR	MINUTE	HUNDRED_BLOCK	NEIGHBOURHOOD		X	Y	LONGTITUDE	LATITUDE
0	Break and Enter Commercial	2019	3	7	2	6	SITKA SQ	Fairview	490612.964805	5.457110e+06	-123.129029	49.266678	
1	Break and Enter Commercial	2019	8	27	4	12	ALBERNI ST	West End	491007.779775	5.459174e+06	-123.123649	49.285255	
2	Break and Enter Commercial	2014	8	8	5	13	ALBERNI ST	West End	491015.943352	5.459166e+06	-123.123536	49.285181	
3	Break and Enter Commercial	2005	11	14	3	9	ALBERNI ST	West End	491021.385727	5.459161e+06	-123.123461	49.285132	
4	Break and Enter Commercial	2006	5	21	4	50	ALBERNI ST	West End	491021.385727	5.459161e+06	-123.123461	49.285132	

11.1 Define Foursquare Credentials and Version

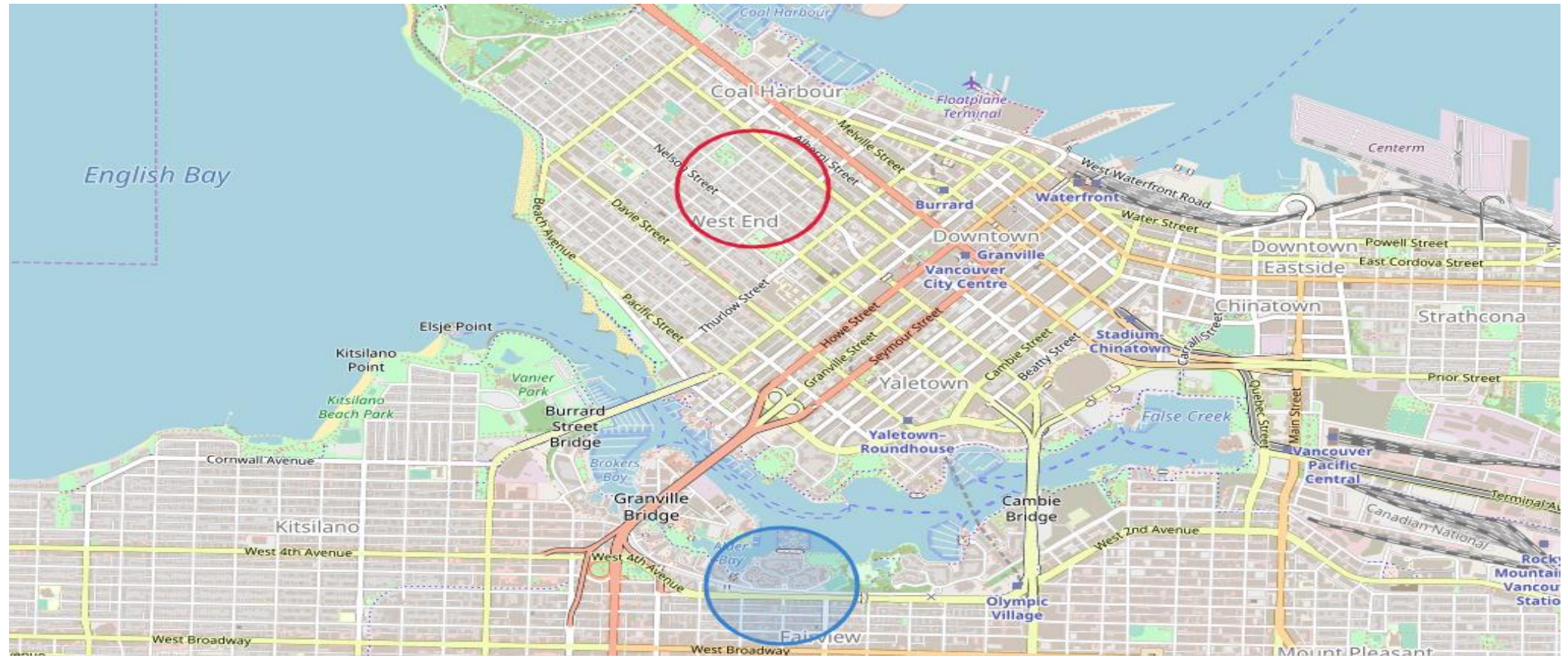
```
In [68]: ▶ # create map of Vancouver using latitude and longitude values
map_Vancouver = folium.Map(location=[49.30330, -123.14635], zoom_start=13)

folium.CircleMarker(
    radius=50,
    location=[49.285646, -123.13062],
    popup='West End',
    color='crimson',
    fill=False,
).add_to(map_Vancouver)

folium.CircleMarker(
    location=[49.2666, -123.12902],
    radius=50,
    popup='Faireview',
    color='#3186cc',
    fill=True,
    fill_color='#3186cc'
).add_to(map_Vancouver)

map_Vancouver
```


West End and Fairview



11.2 Exploring Fairview

In order to define an instance of the geocoder, we need to define a user_agent. We will name our agent foursquare_agent, as shown below.

```
In [70]: address = 'Vancouver, Canada'

geolocator = Nominatim(user_agent="foursquare_agent")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print(latitude, longitude)

49.2608724 -123.1139529
```

```
In [71]: latitude = 49.2608724
longitude = -123.1139529
```

Search for a specific venue category

```
In [72]: search_query = 'Restaurant'
radius = 500
print(search_query + ' .... OK!')

Restaurant .... OK!
```

Define the corresponding URL

```
In [73]: url = 'https://api.foursquare.com/v2/venues/search?client_id={}&client_secret={}&ll={},{}&v={} &query={}&radius={}&limit={}'.format(CLIENT_ID, CLIENT_SECRET, lat, lon, VERSION, QUERY, RADIUS, LIMIT)
```

```
Out[73]: 'https://api.foursquare.com/v2/venues/search?client_id=JN4A3W4535BG44IBDVB02GNZAJFZRHVIDCPX3V1MXXXDGI2&client_secret=CE0FJU0L3UDLQPCTN3EIRUSPXMWSREGHV20GQWAJ04XDM4C0&ll=49.2608724,-123.1139529&v=20180604&query=Restaurant&radius=500&limit=30'
```

Send the GET Request and examine the results

```
In [74]: results = requests.get(url).json()
          results
```

[illegible]

Get relevant part of JSON and transform it into a pandas dataframe


```
In [75]: # assign relevant part of JSON to venues
venues = results['response']['venues']

# transform venues into a dataframe
dataframe = json_normalize(venues)
dataframe.head()
```

Out[75]:

	categories	hasPerk	id	location.address	location.cc	location.city	location.country	location.crossStreet	location.distance	location.formattedAddress
0	[[{'id': '4bf58dd8d48988d145941735', 'name': 'C...'}]]	False	4aa7fa5ef964a5206d4e20e3	201-555 W. 12th Ave.	CA	Vancouver	Canada	at Cambie St.	106	[201-555 W. 12th Ave. (at Cambie St.), Vancouv...
1	[[{'id': '4bf58dd8d48988d145941735', 'name': 'C...'}]]	False	4ab04bfaf964a520ed6620e3	532 W Broadway	CA	Vancouver	Canada	btwn Cambie & Ash	280	[532 W Broadway (btwn Cambie & Ash), Vancouver...
2	[]	False	542d0404498eabf6092127b4	NaN	CA	Vancouver	Canada	NaN	39	[Vancouver BC, Canada]

Define information of interest and filter dataframe

```
In [76]:  # keep only columns that include venue name, and anything that is associated with location
filtered_columns = ['name', 'categories'] + [col for col in dataframe.columns if col.startswith('location.')] + ['id']
dataframe_filtered = dataframe.loc[:, filtered_columns]

# function that extracts the category of the venue
def get_category_type(row):
    try:
        categories_list = row['categories']
    except:
        categories_list = row['venue.categories']

    if len(categories_list) == 0:
        return None
    else:
        return categories_list[0]['name']

# filter the category for each row
dataframe_filtered['categories'] = dataframe_filtered.apply(get_category_type, axis=1)

# clean column names by keeping only last term
dataframe_filtered.columns = [column.split('.')[0] for column in dataframe_filtered.columns]

dataframe_filtered
```

Let's visualize the Restaurants that are nearby

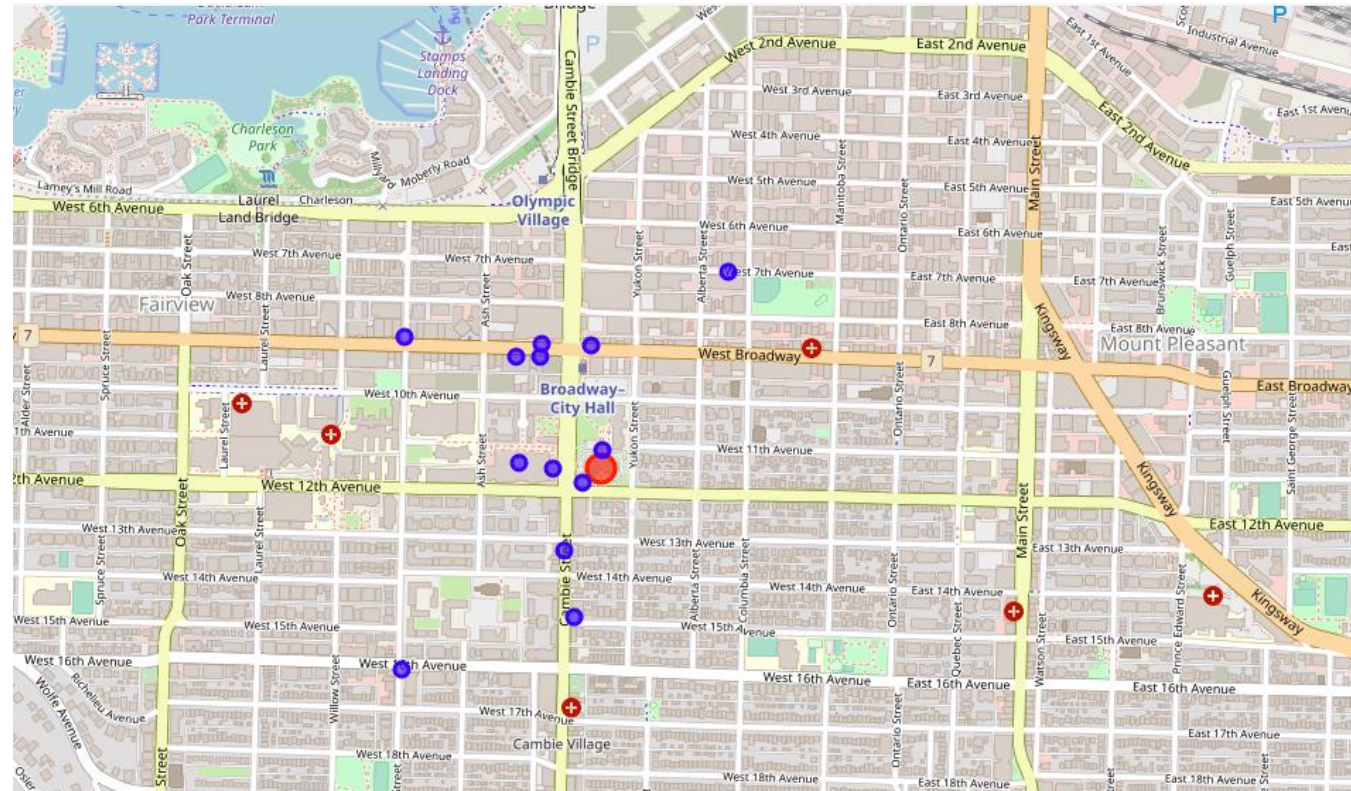
```
In [77]: dataframe_filtered.name
```

```
Out[77]: 0    Kirin Seafood Restaurant 玉麒麟海鮮酒家 (Kirin Seafoo...
          1    Peaceful Restaurant 和平饭店
          2    SBC Restaurant
          3    FigMint Restaurant & Lounge
          4    Curry Point Restaurant
          5    New Sun Restaurant
          6    Kanpachi Japanese Restaurant
          7    Old Ginger Szechuan Restaurant
          8    Shanghai Lu Restaurant
          9    Toko Restaurant
         10    Yolks Breakfast
         11    Chef Lekker Catering
         12    Shiro
          Name: name, dtype: object
```

Exploring venues

	categories	hasPerk	id	location.address	location.cc	location.city	location.country	location.crossStreet	location.distance	location.formattedAddress
0	[[{"id": "4bf58dd8d48988d145941735", "name": "C..."}]]	False	4aa7fa5ef964a5206d4e20e3	201-555 W. 12th Ave.	CA	Vancouver	Canada	at Cambie St.	106	[201-555 W. 12th Ave. (at Cambie St.), Vancouv...
1	[[{"id": "4bf58dd8d48988d145941735", "name": "C..."}]]	False	4ab04bfaf964a520ed6620e3	532 W Broadway	CA	Vancouver	Canada	btwn Cambie & Ash	280	[532 W Broadway (btwn Cambie & Ash), Vancouver...
2	[]	False	542d0404498eabf6092127b4	NaN	CA	Vancouver	Canada	NaN	39	[Vancouver BC, Canada]
3	[[{"id": "4bf58dd8d48988d121941735", "name": "L..."}]]	False	4aa7ebd3f964a520034e20e3	500 West 12th	CA	Vancouver	Canada	@ Cambie Street	51	[500 West 12th (@ Cambie Street), Vancouver BC...
4	[[{"id": "4bf58dd8d48988d10f941735", "name": "L..."}]]	False	4f496e68e4b05ebee07f7c18	NaN	CA	NaN	Canada	NaN	180	[Canada]

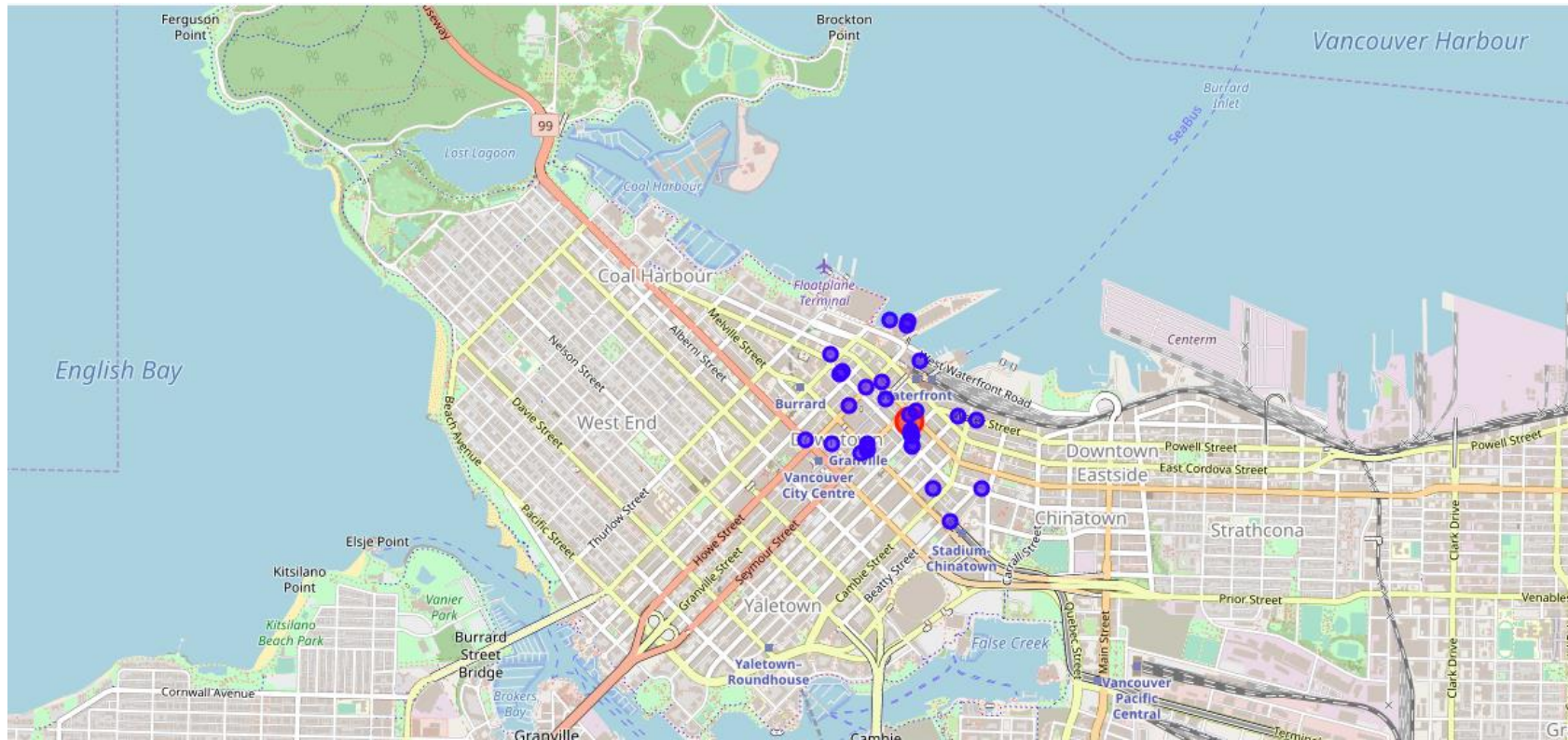
Fairview Top Venues



West End Top Venues

	name	categories	address	cc	city	country	crossStreet	distance	formattedAddress	labeledLatLngs	lat	lng	neighborhood	postalCode	state	id
0	Yagger's Downtown Restaurant & Sports Bar	Sports Bar	433 West Pender	CA	Vancouver	Canada	Homer	109	[433 West Pender (Homer), Vancouver BC V6B 1V2...	[{"label": "display", "lat": 49.28316092121001...	49.283161	-123.112503	NaN	V6B 1V2	BC	4aa82b93f964a520c84f20e3
1	Hawthorn Restaurant	Lounge	801 W Georgia St	CA	Vancouver	Canada	at Howe St	499	[801 W Georgia St (at Howe St), Vancouver BC V...	[{"label": "display", "lat": 49.28336228267866...	49.283362	-123.119462	NaN	V6P 1C7	BC	4d2cce46ae3a8cfa4067bf70
2	Peaceful Restaurant	Chinese Restaurant	602 Seymour St	CA	Vancouver	Canada	at Dunsmuir St	240	[602 Seymour St (at Dunsmuir St), Vancouver BC...	[{"label": "display", "lat": 49.28292966132952...	49.282930	-123.115428	NaN	NaN	BC	576c4083498e4989bae8764a
3	Shanghai Xiao Long Bao Restaurant 上海九鼎	Chinese Restaurant	469 Richards St	CA	Vancouver	Canada	at W Pender St	58	[469 Richards St (at W Pender St), Vancouver B...	[{"label": "display", "lat": 49.28361009495606...	49.283610	-123.112657	NaN	V6B 6H6	BC	50022eace4b0fbf9e9a7d77e

West End Venues Cluster



Discussion

The aim of this project is to help people who want to relocate to the safest borough in Vancouver, expats can choose the neighborhoods to which they want to relocate based on the most common venues in it. But most important is to choose a neighborhood with a relative high safety index like Hasting-Sunrise and Sunset.

Through this exploratory analysis of the Vancouver crime data:

I found that the most common crime in Vancouver is related to vehicle such "Theft of Vehicle", "Theft of Bicycle" and "Vehicle Collision". Definitely, watch out for your bikes, motorcycle, and cars.

West End neighbourhood remain the most crime populated area from 2003 to 2019. The location record of crimes committed in West End can be used to draw a clear street map of West End.

The occurrences of crimes are quite evenly distributed if crimes are counted by day or counted by month. However, crime occurs more in after work hours and midnight in Vancouver.

Conclusion

This project helps a person get a better understanding of the neighborhoods with respect to the most common venues in that neighborhood. It is always helpful to make use of technology to stay one step ahead for example in finding out more about places before moving into a neighborhood.

We have just taken safety as a primary concern to shortlist the safest neighborhood of Vancouver. The future of this project includes taking others factors such as cost of living in the areas into consideration to shortlist the borough, such as filtering areas based on a predefined budget. It will be interesting to includes demographic data to evaluate the population of the different neighborhoods.