In [17]:

```
import pandas as pd
import io
import requests
import lxml
import numpy as np
import folium
from sklearn.cluster import KMeans
from pandas.io.json import json_normalize # tranform JSON file into a panda
import matplotlib.cm as cm
import matplotlib.colors as colors
url="https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M"
dfs = pd.read_html(url)
```

In [18]:

```
print(len(dfs))
df = dfs[0]
df.info()
```

```
3
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180 entries, 0 to 179
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   Postal Code    180 non-null    object
 1   Borough        180 non-null    object
 2   Neighbourhood  180 non-null    object
dtypes: object(3)
memory usage: 4.3+ KB
```

- The dataframe will consist of three columns: PostalCode, Borough, and Neighborhood
- Only process the cells that have an assigned borough. Ignore cells with a borough that is Not assigned.
- More than one neighborhood can exist in one postal code area. For example, in the table on the Wikipedia page, you will notice that M5A is listed twice and has two neighborhoods: Harbourfront and Regent Park. These two rows will be combined into one row with the neighborhoods separated with a comma as shown in row 11 in the above table.
- If a cell has a borough but a Not assigned neighborhood, then the neighborhood will be the same as the borough.
- Clean your Notebook and add Markdown cells to explain your work and any assumptions you are making.
- In the last cell of your notebook, use the .shape method to print the number of rows of your dataframe.

In [19]:

```
# Dropping row within Borough that's not assigned
df['Borough'] =df['Borough'][~df.Borough.str.contains("Not assigned")]
```

In [20]:

```
# Group by postal code and combined neighbourhood if sharing same postal c
df = df.groupby(['Postal Code', 'Borough'], sort=False).agg(', '.join)
df.reset_index(inplace=True)
```

In [21]:
```
# If neightbourhood not assigned, then assigned it with borough
df['Neighbourhood'] = np.where(df['Neighbourhood'] == 'Not assigned', df['B
```

In [22]:
```
df
```

Out[22]:

	Postal Code	Borough	Neighbourhood
0	M3A	North York	Parkwoods
1	M4A	North York	Victoria Village
2	M5A	Downtown Toronto	Regent Park, Harbourfront
3	M6A	North York	Lawrence Manor, Lawrence Heights
4	M7A	Downtown Toronto	Queen's Park, Ontario Provincial Government
...	...	...	...
98	M8X	Etobicoke	The Kingsway, Montgomery Road, Old Mill North
99	M4Y	Downtown Toronto	Church and Wellesley
100	M7Y	East Toronto	Business reply mail Processing Centre, South C...
101	M8Y	Etobicoke	Old Mill South, King's Mill Park, Sunnylea, Hu...
102	M8Z	Etobicoke	Mimico NW, The Queensway West, South of Bloor,...

103 rows × 3 columns

Import in the geospatial coordinate

In [23]:
```
df2 = pd.read_csv('Geospatial_Coordinates.csv')
```

In [24]:
```
# join both table on postal
df = df.join(df2.set_index('Postal Code'), on='Postal Code')
```

Since we only want specific location which is Toronto downtown, We will only extract those within the coordinate

In [25]:
```
df = df.loc[df['Borough']== 'Downtown Toronto']
```

In [26]:

```
df.head()
```

Out[26]:

	Postal Code	Borough	Neighbourhood	Latitude	Longitude
2	M5A	Downtown Toronto	Regent Park, Harbourfront	43.654260	-79.360636
4	M7A	Downtown Toronto	Queen's Park, Ontario Provincial Government	43.662301	-79.389494
9	M5B	Downtown Toronto	Garden District, Ryerson	43.657162	-79.378937
15	M5C	Downtown Toronto	St. James Town	43.651494	-79.375418
20	M5E	Downtown Toronto	Berczy Park	43.644771	-79.373306

In [27]:

```
latitude =  43.654260
longitude = -79.360636
```

In [28]:

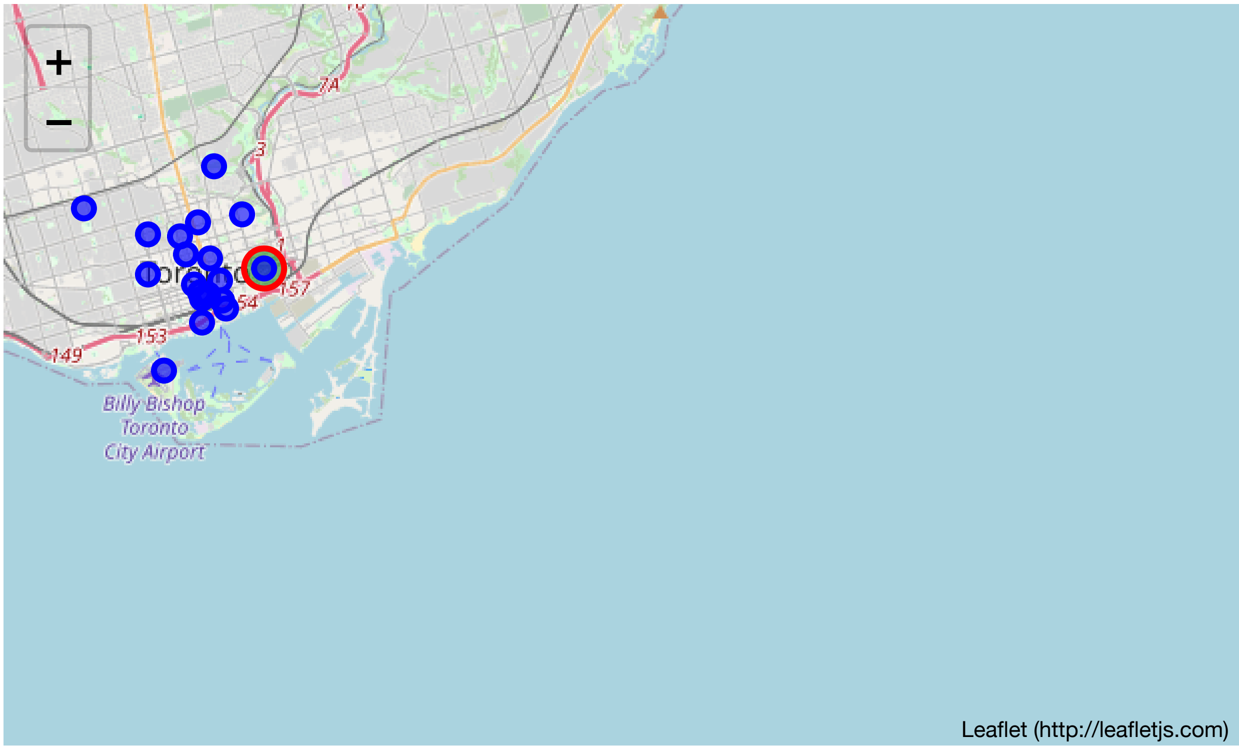
```
venues_map = folium.Map(location=[latitude, longitude], zoom_start=13) # ge

# add a red circle marker to represent the Downtown Toronto
folium.CircleMarker(
    [latitude, longitude],
    radius=10,
    color='red',
    popup='Downtown Toronto',
    fill = True,
    fill_color = 'green',
    fill_opacity = 0.5
).add_to(venues_map)

# add the Borough as blue circle markers
for lat, lng, label in zip(df['Latitude'], df['Longitude'], df['Borough']):
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        color='blue',
        popup=label,
        fill = True,
        fill_color='blue',
        fill_opacity=0.6
    ).add_to(venues_map)

# # display map
venues_map
```

Out[28]:



Clustering the Neighborhood

In [29]:

```
# df.head()
```

In [30]:
```
# Toronto_cluser = df.transpose()
# Toronto_cluser.columns = ['Group-{}'.format(i) for i in range(0,len(Toron
# Toronto_cluser
```

Seperation into 3 different cluster

In [31]:
```
# set number of clusters
kclusters = 3

manhattan_grouped_clustering = df.drop(['Postal Code', 'Borough','Neighbour

# run k-means clustering
kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(manhattan_grouped

# check cluster labels generated for each row in the dataframe
kmeans.labels_

# Insert k cluster as column into df

df.insert(0, 'cluster label', kmeans.labels_)
```

Mapping the cluster

In [32]:

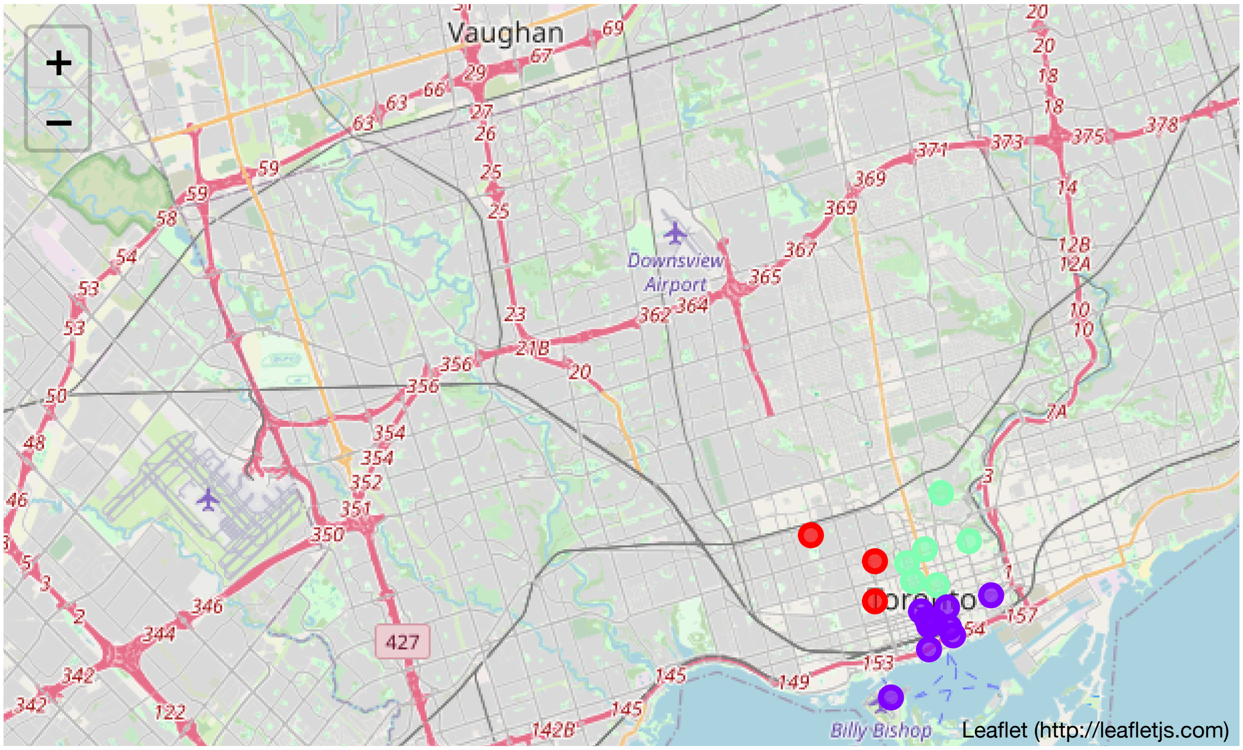
```
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=11)

# set color scheme for the clusters
x = np.arange(kclusters)
ys = [i + x + (i*x)**2 for i in range(kclusters)]
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(df['Latitude'], df['Longitude'], df['Neig
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)

map_clusters
```

Out[32]:



San Jose Food and Interesting venues

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: