Import libraries

In [1]:
```
import pandas as pd
import numpy as np
import requests
import warnings
warnings.filterwarnings('ignore')
import geopy
from geopy.geocoders import Nominatim
import geocoder
import folium
from sklearn.cluster import KMeans
import matplotlib.cm as cm
import matplotlib.colors as colors
from IPython.display import HTML, display
```

Set the url to table location and get the content of the page in variable

In [2]:
```
url = 'https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M'
page = requests.get(url).content
```

Use the pandas read_html function to read the table. In this case the first table on the page ([0])

In [3]:
```
df_raw = pd.read_html(page,header=0)[0]
```

In [4]:
```
df_raw.head()
```

Out[4]:

	Postcode	Borough	Neighbourhood
0	M1A	Not assigned	Not assigned
1	M2A	Not assigned	Not assigned
2	M3A	North York	Parkwoods
3	M4A	North York	Victoria Village
4	M5A	Downtown Toronto	Harbourfront

Let's clean up the dataset according to specifications, filtering and replacing values

In [5]:
```
df = df_raw[df_raw['Borough'] != 'Not assigned'] #filter
df.columns = ['PostalCode','Borough','Neighborhood'] #set columns
df.loc[df['Neighborhood'] == 'Not assigned',['Neighborhood']] = df['Borough'] #replace Neigborhood with Borough if N
```

Group the dataframe and apply the string concatenation

In [6]:
```
df = df.groupby(by=['PostalCode','Borough']).agg(lambda x: ', '.join(set(x))).reset_index()
```

Finally let's show the final frame (103 records with 3 columns)

In [7]:
```
df.shape
```

Out[7]: (103, 3)

We have the dataframe setup, now let's find the longitude and latitude. I'm using the provided csv as the geocoder is malfunctioning

In [8]:
```
file = 'Geospatial_Coordinates.csv'
df_geo = pd.read_csv(file)
df_geo.rename(columns = {'Postal Code':'PostalCode'}, inplace = True)
```

We now have the location data per postalcode so now it will be joined together with the Toronto dataframe

In [9]:
```
df_toronto = pd.merge(df,df_geo, on='PostalCode',how='left')
```

This leaves with clean appended dataframe incl lat and long

In [10]:
```python
df_toronto.head()
```

Out[10]:

	PostalCode	Borough	Neighborhood	Latitude	Longitude
0	M1B	Scarborough	Rouge, Malvern	43.806686	-79.194353
1	M1C	Scarborough	Rouge Hill, Highland Creek, Port Union	43.784535	-79.160497
2	M1E	Scarborough	Morningside, Guildwood, West Hill	43.763573	-79.188711
3	M1G	Scarborough	Woburn	43.770992	-79.216917
4	M1H	Scarborough	Cedarbrae	43.773136	-79.239476

Let's start with our first map and plot the locations on the Toronto map

We'll define the start zoom location of the folium map with the address and geolocator to get the latitude and longitude

In [11]:
```python
address = 'Toronto, Ontario'

geolocator = Nominatim(user_agent="ny_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinates of Toronto are {}, {}.'.format(latitude, longitude))
```

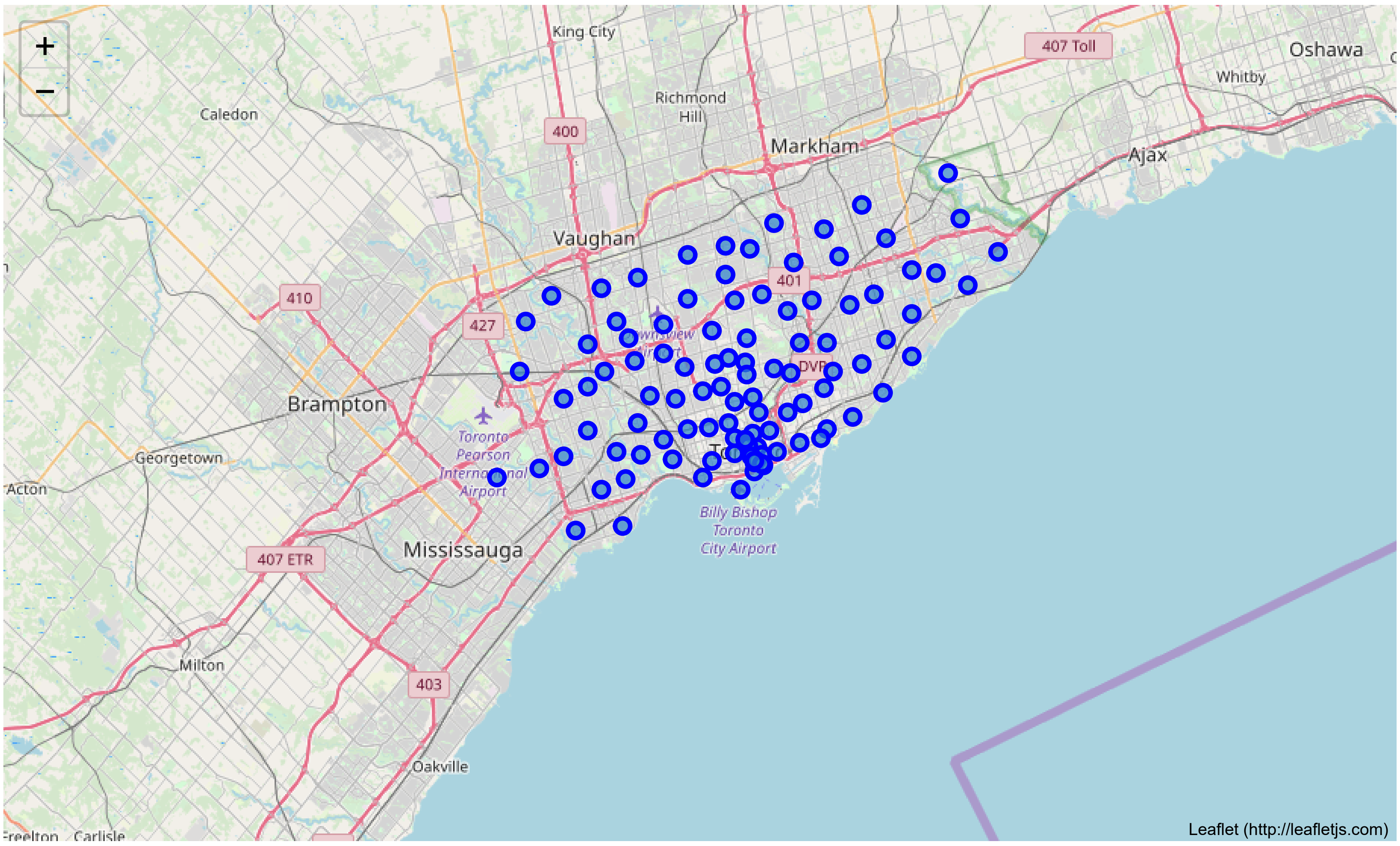
```
The geograpical coordinates of Toronto are 43.653963, -79.387207.
```

Now we plot the areas onto the Toronto map

In [12]:
```python
# Function required to show folium maps inline
```

In [22]:
```python
# create map of New York using latitude and longitude values
map_toronto = folium.Map(location=[latitude, longitude], zoom_start=10)

# add markers to map
for lat, lng, borough, neighborhood in zip(df_toronto['Latitude'], df_toronto['Longitude'], df_toronto['Borough'], d
    label = '{}, {}'.format(neighborhood, borough)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7).add_to(map_toronto)
map_toronto.save('toronto_map1.html')
display(map_toronto)
```



Now we have the initial map setup we continue with clustering. We will start with limiting the dataset as now we have 103 points on the map. We'll limit the set to only show locations for Borough's with the name Toronto in it

In [24]:
```
toronto_data = df_toronto[df_toronto.Borough.str.contains('Toronto')].reset_index(drop=True)
toronto_data.shape
```

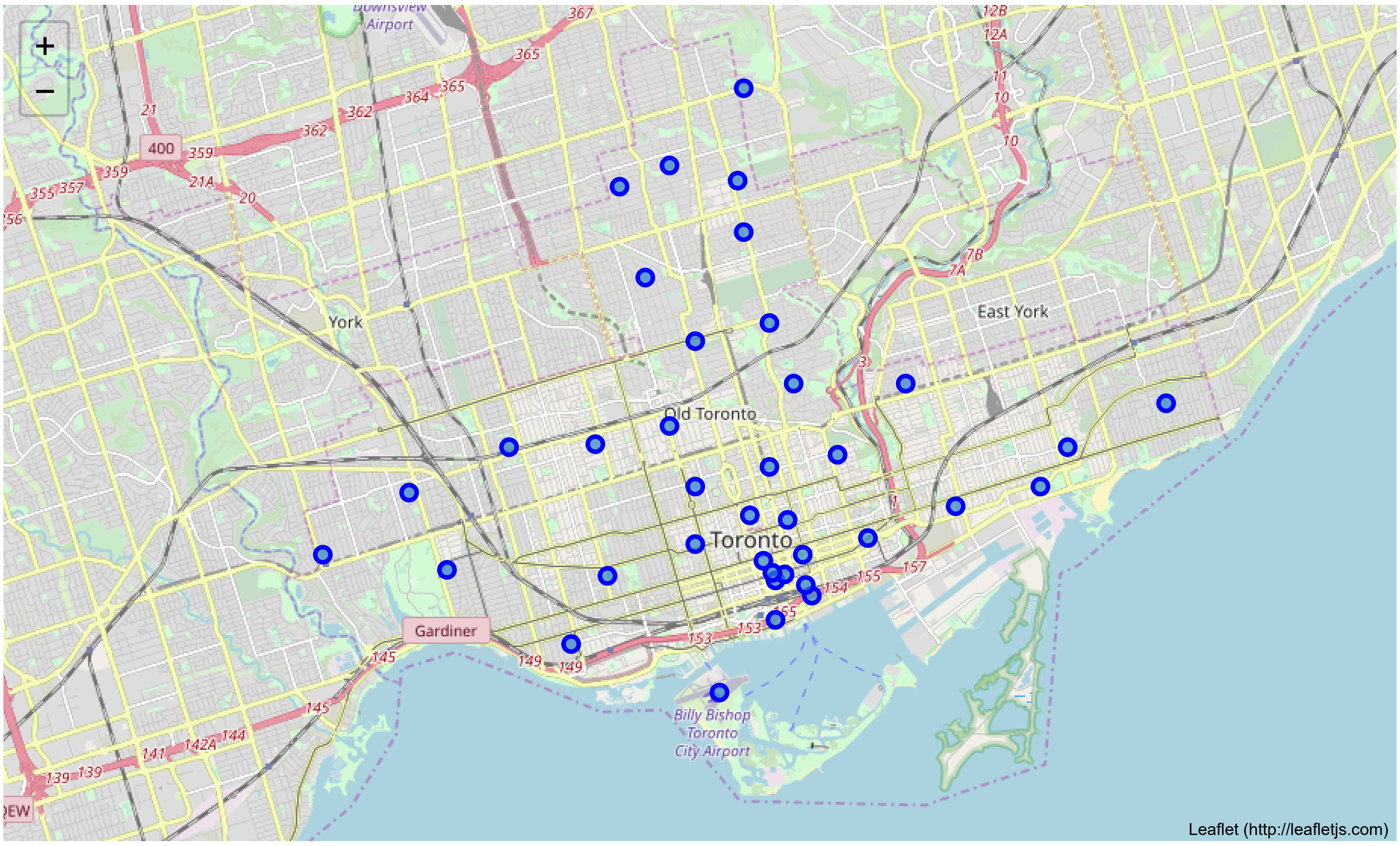
Out[24]: (38, 5)

This leaves us with 38 Boroughs we wil continue to work with. Let's map them again

In [27]:
```
# create map of New York using latitude and longitude values
map_toronto_filtered = folium.Map(location=[latitude, longitude], zoom_start=12)

# add markers to map
for lat, lng, borough, neighborhood in zip(toronto_data['Latitude'], toronto_data['Longitude'], toronto_data['Boroug
    label = '{}, {}'.format(neighborhood, borough)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7).add_to(map_toronto_filtered)

display(map_toronto_filtered)
```



Let's look at venues with foursquare to the locations we have

In [28]:
```
# @hidden cell
CLIENT_ID = '5LAMV3DNDHBER3VMUROMJNYRCJ5S35VSIB5BTJKJW2KHVG55' # your Foursquare ID
CLIENT_SECRET = '3D5FLKFOOA41T5XPDDIZTLLWTPIJWAMVQIU3AS5POKUEV1BW' # your Foursquare Secret
VERSION = '20180605' # Foursquare API version
```

In [29]:
```
# some variables we need for the below functions

LIMIT = 100 # limit of number of venues returned by Foursquare API
radius = 500 # define radius
```

Function to get venues to process all the neighborhoods in Toronto

In [30]:

```
def getNearbyVenues(names, latitudes, longitudes, radius=500):

    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        print(name)

        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&l
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT)

        # make the GET request
        results = requests.get(url).json()["response"]['groups'][0]['items']

        # return only relevant information for each nearby venue
        venues_list.append([(
            name,
            lat,
            lng,
            v['venue']['name'],
            v['venue']['location']['lat'],
            v['venue']['location']['lng'],
            v['venue']['categories'][0]['name']) for v in results])

    nearby_venues = pd.DataFrame([item for venue_list in venues_list for item in venue_list])
    nearby_venues.columns = ['Neighborhood',
                  'Neighborhood Latitude',
                  'Neighborhood Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']

    return(nearby_venues)
```

In [31]:

```
#Apply the function
toronto_venues = getNearbyVenues(names=toronto_data ['Neighborhood'],
                                   latitudes=toronto_data ['Latitude'],
                                   longitudes=toronto_data['Longitude'])
```

```
The Beaches
The Danforth West, Riverdale
The Beaches West, India Bazaar
Studio District
Lawrence Park
Davisville North
North Toronto West
Davisville
Moore Park, Summerhill East
Deer Park, Summerhill West, South Hill, Forest Hill SE, Rathnelly
Rosedale
Cabbagetown, St. James Town
Church and Wellesley
Harbourfront, Regent Park
Ryerson, Garden District
St. James Town
Berczy Park
Central Bay Street
Richmond, Adelaide, King
Union Station, Toronto Islands, Harbourfront East
Design Exchange, Toronto Dominion Centre
Commerce Court, Victoria Hotel
Roselawn
Forest Hill West, Forest Hill North
North Midtown, The Annex, Yorkville
Harbord, University of Toronto
Grange Park, Chinatown, Kensington Market
Railway Lands, Island airport, South Niagara, Bathurst Quay, Harbourfront West, CN Tower, King and Spadina
Stn A PO Boxes 25 The Esplanade
First Canadian Place, Underground city
Christie
Dovercourt Village, Dufferin
Trinity, Little Portugal
Brockton, Exhibition Place, Parkdale Village
The Junction South, High Park
Parkdale, Roncesvalles
Runnymede, Swansea
Business Reply Mail Processing Centre 969 Eastern
```

So we got 1700 records returned with 7 columns

In [32]:

```python
print(toronto_venues.shape)
toronto_venues.head()
```

(1705, 7)

Out[32]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	The Beaches	43.676357	-79.293031	Glen Manor Ravine	43.676821	-79.293942	Trail
1	The Beaches	43.676357	-79.293031	The Big Carrot Natural Food Market	43.678879	-79.297734	Health Food Store
2	The Beaches	43.676357	-79.293031	Grover Pub and Grub	43.679181	-79.297215	Pub
3	The Beaches	43.676357	-79.293031	Upper Beaches	43.680563	-79.292869	Neighborhood
4	The Danforth West, Riverdale	43.679557	-79.352188	Pantheon	43.677621	-79.351434	Greek Restaurant

We continue to analyse the neighborhoods

In [33]:

```python
# one hot encoding
toronto_onehot = pd.get_dummies(toronto_venues[['Venue Category']], prefix="", prefix_sep="")

# add neighborhood column back to dataframe
toronto_onehot['Neighborhood'] = toronto_venues['Neighborhood']

# move neighborhood column to the first column
fixed_columns = [toronto_onehot.columns[-1]] + list(toronto_onehot.columns[:-1])
toronto_onehot = toronto_onehot[fixed_columns]

toronto_onehot.head()
```

Out[33]:

	Yoga Studio	Afghan Restaurant	Airport	Airport Food Court	Airport Gate	Airport Lounge	Airport Service	Airport Terminal	American Restaurant	Antique Shop	...	Theme Restaurant	Thrift / Vintage Store	Toy / Game Store	Trail	Train Station	Veg / Rest
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	1	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	

5 rows × 234 columns

In [34]:

```python
toronto_onehot.shape
```

Out[34]: (1705, 234)

So got the categories converted to numerical values and transposed them into columns. We have 1700 records with 234 Venues categories

We group them by Neigborhood and that will leave us with 38 neighborhoods with 234 venue categories

In [35]:

```python
toronto_grouped = toronto_onehot.groupby('Neighborhood').mean().reset_index()
toronto_grouped.shape
```

Out[35]: (38, 234)

That is a lot to process so we will get the top 10 venues for each neighborhood

In [36]:

```python
def return_most_common_venues(row, num_top_venues):
    row_categories = row.iloc[1:]
    row_categories_sorted = row_categories.sort_values(ascending=False)
    return row_categories_sorted.index.values[0:num_top_venues]
```

In [37]:

```
num_top_venues = 10

indicators = ['st', 'nd', 'rd']

# create columns according to number of top venues
columns = ['Neighborhood']
for ind in np.arange(num_top_venues):
    try:
        columns.append('{}{} Most Common Venue'.format(ind+1, indicators[ind]))
    except:
        columns.append('{}th Most Common Venue'.format(ind+1))

# create a new dataframe
neighborhoods_venues_sorted = pd.DataFrame(columns=columns)
neighborhoods_venues_sorted['Neighborhood'] = toronto_grouped['Neighborhood']

for ind in np.arange(toronto_grouped.shape[0]):
    neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(toronto_grouped.iloc[ind, :], num_top_venu

neighborhoods_venues_sorted.head(3)
```

Out[37]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
0	Berczy Park	Coffee Shop	Cocktail Bar	Farmers Market	Beer Bar	Bakery	Steakhouse	Cheese Shop	Café	Seafood Restaurant	Italian Restaurant
1	Brockton, Exhibition Place, Parkdale Village	Coffee Shop	Café	Breakfast Spot	Grocery Store	Intersection	Convenience Store	Pet Store	Gym	Climbing Gym	Caribbean Restaurant
2	Business Reply Mail Processing Centre 969 Eastern	Light Rail Station	Yoga Studio	Spa	Garden Center	Garden	Fast Food Restaurant	Farmers Market	Comic Shop	Park	Recording Studio

We will now cluster the neighborhood with *k*-means into 5 clusters

In [38]:

```
# set number of clusters
kclusters = 5

toronto_grouped_clustering = toronto_grouped.drop('Neighborhood', 1)

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

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:10]
```

Out[38]:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int32)
```

In [39]:

```
kmeans_labels = kmeans.labels_
```

Let's add the clusters back to the neighborhoods and venues

In [40]:

```
# add clustering labels
neighborhoods_venues_sorted.insert(0, 'Cluster Labels', kmeans_labels)

toronto_merged = toronto_data

# merge toronto_grouped with toronto_data to add latitude/longitude for each neighborhood
toronto_merged = toronto_merged.join(neighborhoods_venues_sorted.set_index('Neighborhood'), on='Neighborhood')

toronto_merged.head()
```

Out[40]:

	PostalCode	Borough	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Mos Commo Venu
0	M4E	East Toronto	The Beaches	43.676357	-79.293031	0	Health Food Store	Trail	Pub	Dessert Shop	Falafel Restaurant	Event Space	Ethiopia Restaurai
1	M4K	East Toronto	The Danforth West, Riverdale	43.679557	-79.352188	0	Greek Restaurant	Coffee Shop	Italian Restaurant	Furniture / Home Store	Ice Cream Shop	Caribbean Restaurant	Bookstor
2	M4L	East Toronto	The Beaches West, India Bazaar	43.668999	-79.315572	0	Park	Movie Theater	Liquor Store	Board Shop	Sandwich Place	Burger Joint	Fast Foc Restaurai
3	M4M	East Toronto	Studio District	43.659526	-79.340923	0	Café	Coffee Shop	American Restaurant	Italian Restaurant	Bakery	Seafood Restaurant	Lati America Restaurai
4	M4N	Central Toronto	Lawrence Park	43.728020	-79.388790	3	Park	Jewelry Store	Swim School	Bus Line	Wings Joint	Discount Store	Falafe Restaurai

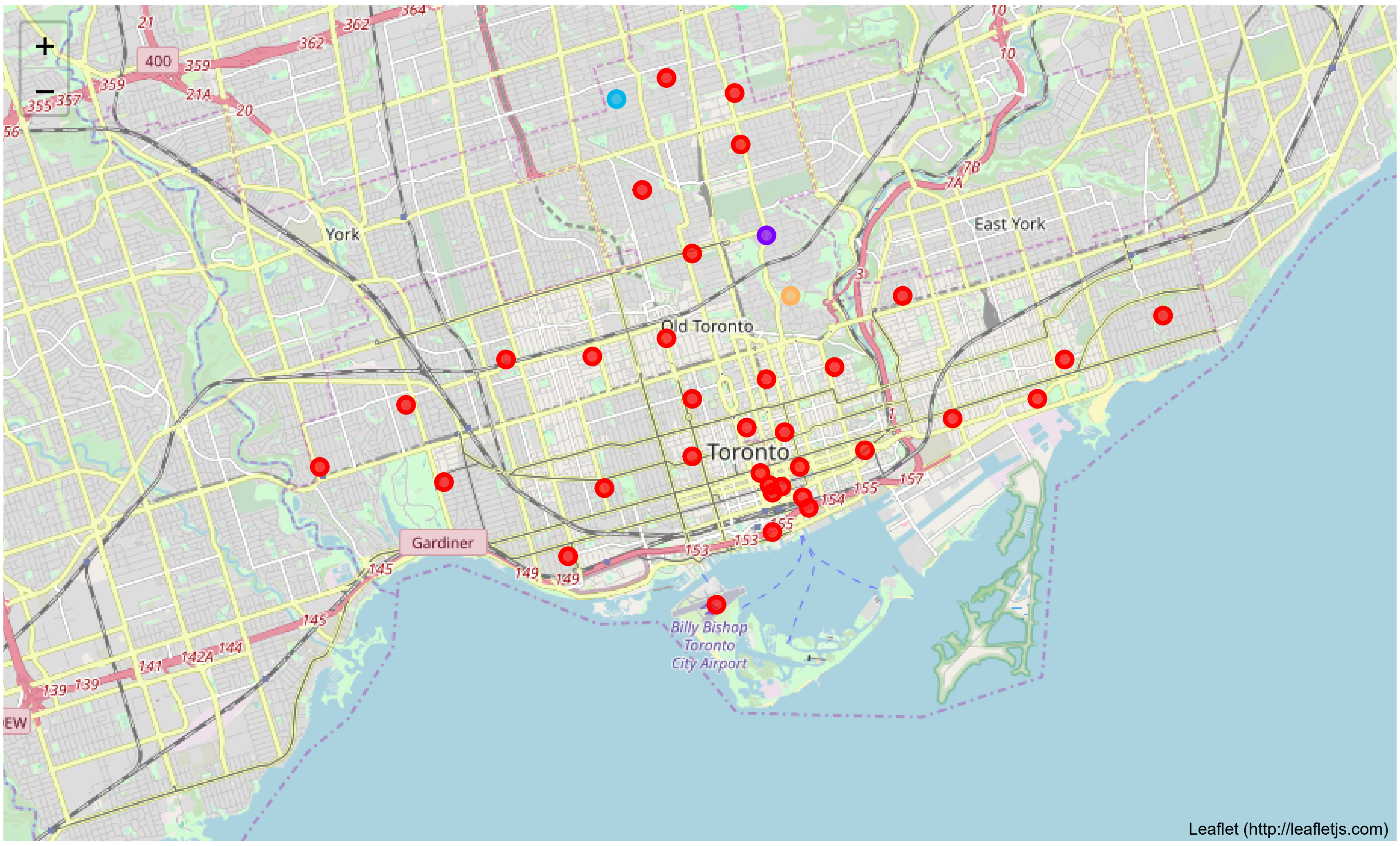
Finally, let's visualize the resulting clusters

In [43]:

```
# create map
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=12)

# 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(toronto_merged['Latitude'], toronto_merged['Longitude'], toronto_merged['Neighborh
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    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)
display(map_clusters)
```



The question then is what do the clusters represent. What is in those various clusters so we can name them better than Cluster 0-4

We will filter the toronto_merged frame into their respective variable so we analyse further

In [44]:

```
label_0 = toronto_merged.loc[toronto_merged['Cluster Labels'] == 0, toronto_merged.columns[[1] + list(range(5, toror
label_1 = toronto_merged.loc[toronto_merged['Cluster Labels'] == 1, toronto_merged.columns[[1] + list(range(5, toror
label_2 = toronto_merged.loc[toronto_merged['Cluster Labels'] == 2, toronto_merged.columns[[1] + list(range(5, toror
label_3 = toronto_merged.loc[toronto_merged['Cluster Labels'] == 3, toronto_merged.columns[[1] + list(range(5, toror
label_4 = toronto_merged.loc[toronto_merged['Cluster Labels'] == 4, toronto_merged.columns[[1] + list(range(5, toror
```

I'm not exactly sure how to find the common ground in the various clusters

In [45]:

```
venues_columns = neighborhoods_venues_sorted.columns
venues_columns = venues_columns.drop(['Cluster Labels','Neighborhood'])
```

In [46]:
```
1 label_1.head()
```

Out[46]:

	Borough	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
8	Central Toronto	1	Playground	Diner	Farmers Market	Falafel Restaurant	Event Space	Ethiopian Restaurant	Electronics Store	Eastern European Restaurant	Dumpling Restaurant	Donut Shop

In [47]:
```
1 label_4 # park
```

Out[47]:

	Borough	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
10	Downtown Toronto	4	Park	Playground	Trail	Building	Wings Joint	Diner	Event Space	Ethiopian Restaurant	Electronics Store	Eastern European Restaurant

In [ ]:
```
1
```