The Battle of Neighborhood

Segmenting and Clustering Neighborhoods of Toronto and Waterloo Ontario by Jahangir Nasimi

Table of Contents

Introduction

This project aims at creating a tool that can be used to explore facilities in and around neighbourhoods in a way that allows residents to make informed decisions and select the most optimal location that best caters to their needs.

An example of such need could be smaller cities and towns in various provinces of Canada. Canada has had a long and successful immigration policy that allows regulating the country's population. From its onset in 1870, the influx of immigrants has been tailored to grow the population, settle the land, and provide labour and financial capital for the economy. However, in a democratic country, such as Canada, both the newcomers and the existing residents have a right to move around and select a location with the housing prices, good schools, medical facilities and job opportunities that suit their needs and financial means.

This project would be highly beneficial for those who seek a systematic, algorithm-based approach for comparison and selection of the desired parameters, e.g. access to public or catholic schools, supermarkets, shopping malls, movie theaters, hospitals, or religious communities. This selection can be made using analytical tools rather than basic internet searches, reliance on real estate agencies or word of mouth that could be biased or misleading, whether intentionally or not. Studies such as [1] show that people need to think critically about claims and compare their options using a systematic concept because simply trusting the sources of information may not be a sufficient basis for making the best choice: • Competing interests can result in misleading claims. • Personal experiences or anecdotes alone are an unreliable basis for most claims. • Opinions of acquittances, real-estate agents or newspaper/social media publications are not solely a reliable basis for making informed-decisions. • Endorsements by community leaders or other respectable individuals do not guarantee that comparisons have been fair.

Objective: The main objective of this project is to propose an approach for selection of an optimal community in a new city for someone looking to relocate. The individual using this proposed tool would be able to compile, sort and filter parameters of interest, for example:

1. A list of houses/residencies that can be sorted in terms of housing prices in an ascending or descending order, or
2. A list of restaurants that cater particular type of cuisine that can be sorted in terms of location, prices, rating and reviews.

The problem for the Canadian province of Ontario is that most immigrants to the province make their home in the GTA, specifically 77 per cent, according to an August report from the Conference Board of Canada [2]. However, as their populations age and young adults move away, small cities and towns across Canada are increasingly looking to immigration as a way to rejuvenate their workforce and expand their tax base. But many struggle to find reliable data and make informed decision to move to smaller rural areas due to lack of data and unavailability of unbiased selection tools. The city of Waterloo, Ontario will be used in this project to demonstrate the proposed approach as opposed to Toronto. Specific Objectives: Specifically, this project looks at a community of Russian-speaking expatriates who are looking to re-settle in the Greater Toronto Area (GTA) or in other smaller neighbouring communities in the province of Ontario. This project will propose a tool that can be used to perform searches and comparisons between various neighbourhoods in order to determine a community with specific desired parameters, e.g., access to grocery stores and marketplaces where Russian-produced goods and food supplies are available.

Additional Parameters: Additional parameters could include ease of access to children's daycares and after-school activities where Russian language is taught. Another important factor could be availability of various medical facilities (e.g., Russian-speaking family doctors, dentists, phycologists and other medical specialists) for Russian-speaking patients and access to nursing homes where services in Russian language are available for seniors. Restaurants offering traditional Russian cuisine is another important factor that may sway a customer's decision to relocate to a particular community.

Data

The data used in this project was acquired mostly from Wikipedia.com and Toronto.ca websites. In order to be used in the proposed approach it was formatted and restructured into a csv. file. The files used in the project can be found on github repository as well as referenced in [3]. The project data was transferred to a local drive and compiled to dataframes as can be seen from the project code.

Based on the 2011 Canadian census data, most of Russian-native speakers in Toronto (GTA) reside in the geographical area of York. Per [4], there are more than 35,000 people in this area who list Russian as their mother tongue.

Next, Foursquare approach was used for data segmentation and clustering as follows:

Foursquare API: This project uses Four-square API as its prime data gathering source as it has a database of millions of places, especially their places API which provides the ability to perform location search, location sharing and details about a business. Important to point out that 100% data capture accuracy cannot be achieved due to inherent data source omissions and inaccuracies.

Before we get the data and start exploring it, let's download all the dependencies that we will need.

In [1]:
```
import numpy as np # library to handle data in a vectorized manner

import pandas as pd # library for data analsysis
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

import json # library to handle JSON files

# !conda install -c conda-forge geopy --yes # uncomment this line if you haven't completed the Foursquare API lab
from geopy.geocoders import Nominatim # convert an address into latitude and longitude values
# !conda install -c conda-forge wikipedia  --yes
print('Wikipedia installed')
import wikipedia as wp
import requests # library to handle requests
from pandas.io.json import json_normalize # tranform JSON file into a pandas dataframe

# Matplotlib and associated plotting modules
import matplotlib.cm as cm
import matplotlib.colors as colors

# import k-means from clustering stage
from sklearn.cluster import KMeans

#!conda install -c conda-forge folium=0.5.0 --yes # uncomment this line if you haven't completed the Foursquare API lab
import folium # map rendering library

print('Libraries imported.')
```

```
Wikipedia installed
Libraries imported.
```

1. Download and Explore Dataset for GTA

Type *Markdown* and LaTeX: α^2

Load and explore the data

In [2]:
```
#Get the html source
html = wp.page("List_of_postal_codes_of_Canada:_M").html().encode("UTF-8")
df = pd.read_html(html)[0]
#save dataset
df.to_csv('neighborhoods_postalcodes.csv',index=False)
```

Type *Markdown* and LaTeX: α^2

Tranform the data into a *pandas* dataframe and eliminate records with not assigned Borough

In [3]:
```
#read dataset
neighborhoods = pd.read_csv('neighborhoods_postalcodes.csv',sep=",",skiprows=1)

#eliminate records with not assigned Borough
neighborhoods=neighborhoods.replace({'Neighbourhood':['Not assigned']},"Queen's Park")
neighborhoods = neighborhoods[~neighborhoods["Borough"].isin(['Not assigned'])]
```

In [4]:
```
#Change Column names
neighborhoods=neighborhoods.rename(columns={'Postal code':'PostalCode'})
neighborhoods.head()
```

Out[4]:

	PostalCode	Borough	Neighborhood
2	M3A	North York	Parkwoods
3	M4A	North York	Victoria Village
4	M5A	Downtown Toronto	Regent Park / Harbourfront
5	M6A	North York	Lawrence Manor / Lawrence Heights
6	M7A	Downtown Toronto	Queen's Park / Ontario Provincial Government

Quickly examine the resulting dataframe.

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

Out[5]:

	PostalCode	Borough	Neighborhood
2	M3A	North York	Parkwoods
3	M4A	North York	Victoria Village
4	M5A	Downtown Toronto	Regent Park / Harbourfront
5	M6A	North York	Lawrence Manor / Lawrence Heights
6	M7A	Downtown Toronto	Queen's Park / Ontario Provincial Government

And make sure that the dataset has all 10 boroughs and 103 neighborhoods.

In [6]:
```
print('The dataframe has {} boroughs and {} neighborhoods.'.format(
        len(neighborhoods['Borough'].unique()),
        neighborhoods.shape[0]
    )
)
```

```
The dataframe has 10 boroughs and 103 neighborhoods.
```

Use geopy library to get the latitude and longitude values of Toronto.

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

In [7]:
```
address = "Toronto, ON"

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

```
The geograpical coordinate of Toronto city are 43.6534817, -79.3839347.
```

Get Latitude and the longitude coordinates of each neighborhood

In [8]:
```
neighborhoods_geo_coor = pd.read_csv("./Geospatial_Coordinates.csv")
neighborhoods_geo_coor.head()
```

Out[8]:

	Postal Code	Latitude	Longitude
0	M1B	43.806686	-79.194353
1	M1C	43.784535	-79.160497
2	M1E	43.763573	-79.188711
3	M1G	43.770992	-79.216917
4	M1H	43.773136	-79.239476

Left Join 2 dataframes "neighborhoods" and "neighborhoods_geo_coor" into one dataframe to get Latitude, Longitude for each neighborhoods.

In [9]:
```
df_neighborhoods = pd.merge(neighborhoods, neighborhoods_geo_coor, how='left', left_on = 'PostalCode', right_on = 'Postal Code')
# remove the "Postal Code" column
df_neighborhoods.drop("Postal Code", axis=1, inplace=True)
df_neighborhoods.head()
```

Out[9]:

	PostalCode	Borough	Neighborhood	Latitude	Longitude
0	M3A	North York	Parkwoods	43.753259	-79.329656
1	M4A	North York	Victoria Village	43.725882	-79.315572
2	M5A	Downtown Toronto	Regent Park / Harbourfront	43.654260	-79.360636
3	M6A	North York	Lawrence Manor / Lawrence Heights	43.718518	-79.464763
4	M7A	Downtown Toronto	Queen's Park / Ontario Provincial Government	43.662301	-79.389494

Methodology

The approach in this project is based on the methodology discussed in Week 3 lab and is as follows:

- First, neighbourhood postal codes were converted into the equivalent latitude and longitude values.
- Next, Foursquare API was used to explore neighborhoods in both cities, Toronto and Waterloo. The data for Toronto was limited only to those corresponding to the York region.
- Next, a function to determine common venue categories in each neighbourhood was created and executed.
- Next, the neighbourhoods were clustered according to the venue features.
- Subsequently, K-means clustering algorithm was used to complete the analysis. Folium library was utilized to visualize the emerging clusters for

neighborhoods in Toronto and Waterloo.

These steps are described in more detail below:

Work Flow: Using credentials of Foursquare API, features of near-by places of the neighborhoods would be mined. Due to http request limitations, the number of places per neighborhood parameter would be reasonably set to 50, the radius parameter would be set to 500 for Toronto, and to 2000 for the city of Waterloo.

Clustering Approach: To compare the similarities of the two cities, it was decided to explore neighborhoods, segment them, and group them into clusters to find similar neighborhoods in a big city like Waterloo and Toronto. To be able to do that, it is necessary to cluster data which is a form of unsupervised machine learning: k-means clustering algorithm.

Libraries Used to Develop the Project:

- Pandas: For creating and manipulating dataframes.
- Folium: Python visualization library would be used to visualize the neighborhoods cluster distribution of using interactive leaflet map.
- Scikit Learn: For importing k-means clustering.
- JSON: Library to handle JSON files.
- XML: To separate data from presentation and XML stores data in plain text format.
- Geocoder: To retrieve Location Data.
- Beautiful Soup and Requests: To scrap and library to handle http requests.
- Matplotlib: Python Plotting Module.

Results

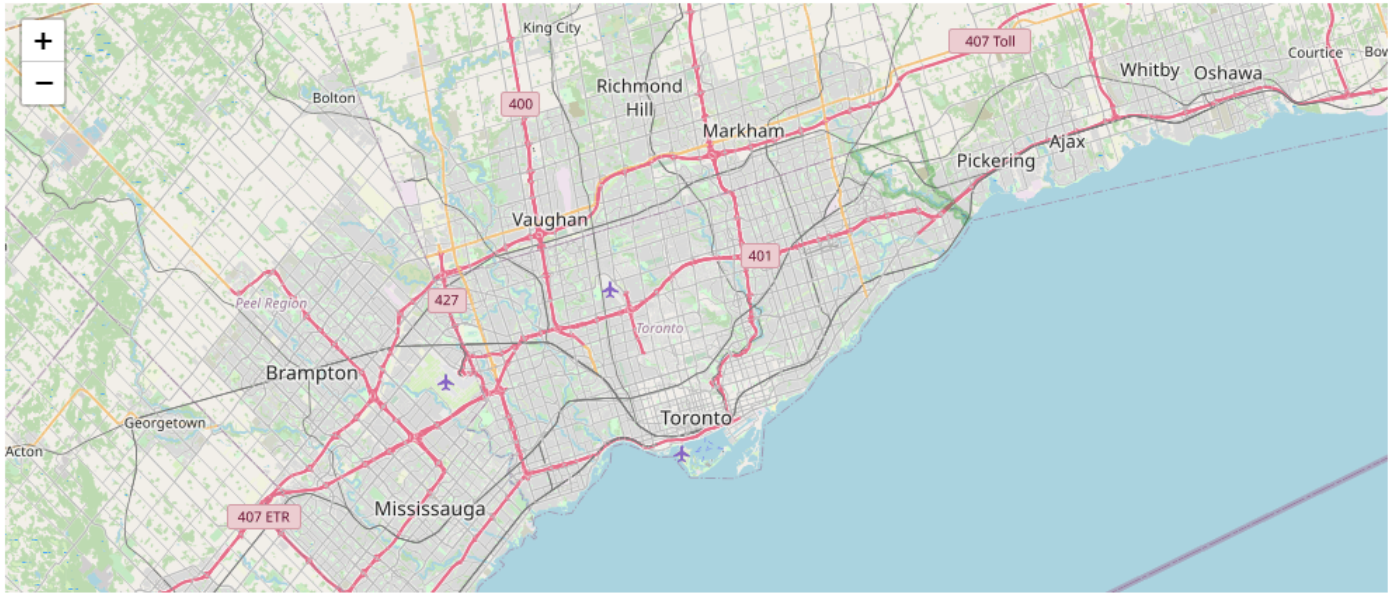
[Top](#toc)

Using the proposed methodology, it was determined that North York area in Toronto and Waterloo-Central area in the city of Waterloo both showed the highest number of neighbourhoods in a cluster. Shown below are the code and the geo-spacial graphs for the project data. Tables for each cluster with the venue categories are also shown here.

Create a map of the whole GTA

In [10]:
```
# create map of Toronto using latitude and longitude values
map_neighborhoods = folium.Map(location=[latitude, longitude], zoom_start=10)
map_neighborhoods
```

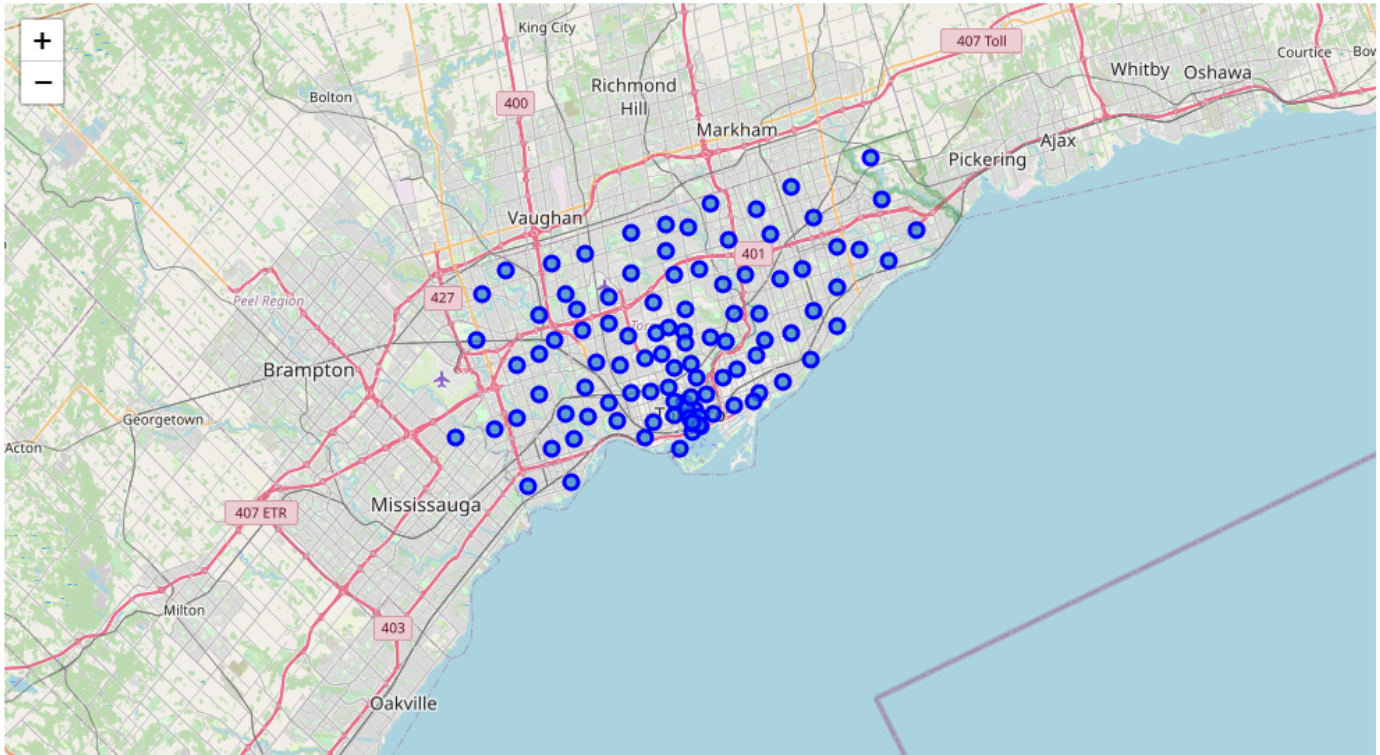
Out[10]:



Add markers to GTA map.

In [11]:
```
for lat, lng, borough, neighborhood in zip(
        df_neighborhoods['Latitude'],
        df_neighborhoods['Longitude'],
        df_neighborhoods['Borough'],
        df_neighborhoods['Neighborhood']):
    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,
        parse_html=False).add_to(map_neighborhoods)


map_neighborhoods
```

Out[11]:



In [12]:
```
map_neighborhoods.save(outfile= "Toront_map_neighborhoods.html")
```

Create a data frame with boroughs that contain the word "York".

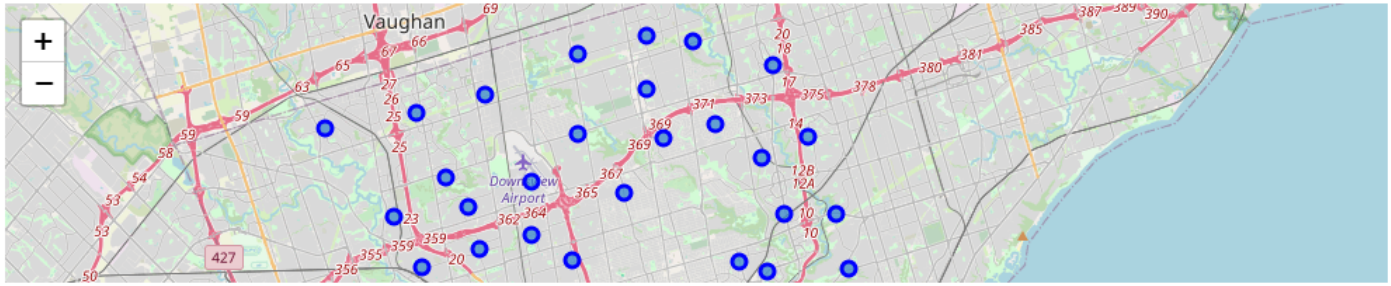
In [13]:
```
df_neighborhoods_to = df_neighborhoods[df_neighborhoods['Borough'].str.contains("York")].reset_index(drop=True)
#df_neighborhoods_to = df_neighborhoods
df_neighborhoods_to.head(11)
```

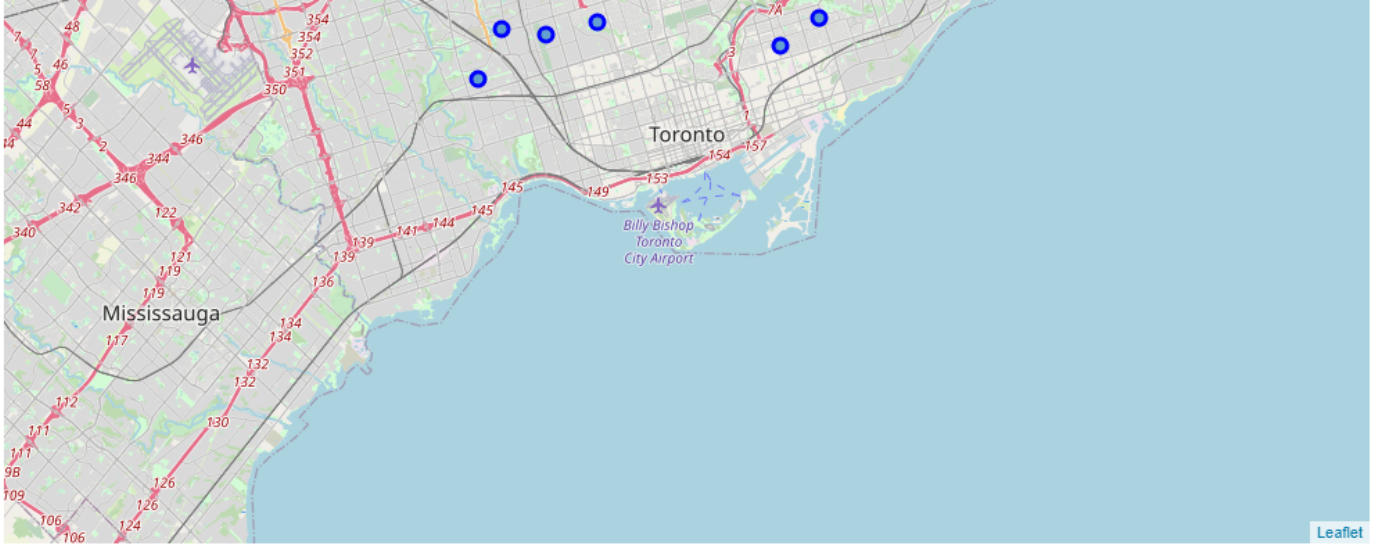
Out[13]:

	PostalCode	Borough	Neighborhood	Latitude	Longitude
0	M3A	North York	Parkwoods	43.753259	-79.329656
1	M4A	North York	Victoria Village	43.725882	-79.315572
2	M6A	North York	Lawrence Manor / Lawrence Heights	43.718518	-79.464763
3	M3B	North York	Don Mills	43.745906	-79.352188
4	M4B	East York	Parkview Hill / Woodbine Gardens	43.706397	-79.309937
5	M6B	North York	Glencairn	43.709577	-79.445073
6	M3C	North York	Don Mills	43.725900	-79.340923
7	M4C	East York	Woodbine Heights	43.695344	-79.318389
8	M6C	York	Humewood-Cedarvale	43.693781	-79.428191
9	M6E	York	Caledonia-Fairbanks	43.689026	-79.453512
10	M4G	East York	Leaside	43.709060	-79.363452

In [14]:
```
map_neighborhoods_to = folium.Map(location=[latitude, longitude], zoom_start=11)
for lat, lng, borough, neighborhood in zip(
        df_neighborhoods_to['Latitude'],
        df_neighborhoods_to['Longitude'],
        df_neighborhoods_to['Borough'],
        df_neighborhoods_to['Neighborhood']):
    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,
        parse_html=False).add_to(map_neighborhoods_to)

map_neighborhoods_to
```

Out[14]:





Next, we are going to start utilizing the Foursquare API to explore the neighborhoods and segment them.

Define Foursquare Credentials and Version

In [15]:
```
CLIENT_ID = 'ChangeMe' # your Foursquare ID
CLIENT_SECRET = 'ChangeMe' # your Foursquare Secret
VERSION = '20180605' # Foursquare API version

print('Your credentails:')
print('CLIENT_ID: ' + CLIENT_ID)
print('CLIENT_SECRET:' + CLIENT_SECRET)
```

```
Your credentails:
CLIENT_ID: 41BUH4RPNJI2JDHGAUFKIMORK55CUOQHBVQWMLS2RKMARK13
CLIENT_SECRET:CQUM4PUTMBRRRNUPUXW2SJ25MAMGCN0TGWRIWU334Z43NTLE
```

Let's explore the first neighborhood in our dataframe.

Get the neighborhood's name.

In [16]:
```
neighborhood_name = df_neighborhoods_to.loc[0, 'Neighborhood']
print(f"The first neighborhood's name is '{neighborhood_name}'.")

neighborhood_latitude = df_neighborhoods_to.loc[0, 'Latitude']
neighborhood_longitude = df_neighborhoods_to.loc[0, 'Longitude']

print('Latitude and longitude values of {} are {}, {}.'.format(neighborhood_name,
                                                               neighborhood_latitude,
                                                               neighborhood_longitude))
```

```
The first neighborhood's name is 'Parkwoods'.
Latitude and longitude values of Parkwoods are 43.7532586, -79.3296565.
```

Get the neighborhood's latitude and longitude values.

In [17]:
```
neighborhood_latitude = df_neighborhoods_to.loc[0, 'Latitude'] # neighborhood latitude value
neighborhood_longitude = df_neighborhoods_to.loc[0, 'Longitude'] # neighborhood longitude value
radius = 500
```

```
neighborhood_name = df_neighborhoods_to.loc[0, 'Neighborhood'] # neighborhood name

print('Latitude and longitude values of {} are {}, {}.'.format(neighborhood_name,
                                                               neighborhood_latitude,
                                                               neighborhood_longitude))
```

```
Latitude and longitude values of Parkwoods are 43.7532586, -79.3296565.
```

Now, let's get the top 100 venues that are in Parkwoods within a radius of 500 meters.

First, let's create the GET request URL. Name your URL **url**.

In [18]:
```
# limit of number of venues returned by Foursquare API
LIMIT = 50
# define radius
radius = 500
url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    neighborhood_latitude,
    neighborhood_longitude,
    radius,
    LIMIT)

# get the result to a json file
results = requests.get(url).json()
```

Extract the category of the venue

From the Foursquare lab in the previous module, we know that all the information is in the *items* key. Before we proceed, let's borrow the **get_category_type** function from the Foursquare lab.

In [19]:
```
# 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']
```

Now we are ready to clean the json and structure it into a *pandas* dataframe.

In [20]:
```
venues = results['response']['groups'][0]['items']

nearby_venues = json_normalize(venues) # flatten JSON

# filter columns
filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat', 'venue.location.lng']
nearby_venues =nearby_venues.loc[:, filtered_columns]

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

# clean columns
nearby_venues.columns = [col.split(".")[-1] for col in nearby_venues.columns]
```

```
nearby_venues.head()
```

Out[20]:

	name	categories	lat	lng
0	Brookbanks Park	Park	43.751976	-79.332140
1	Variety Store	Food & Drink Shop	43.751974	-79.333114

And how many venues were returned by Foursquare?

In [21]:
```
print('{} venues were returned by Foursquare.'.format(nearby_venues.shape[0]))
```

```
2 venues were returned by Foursquare.
```

2. Explore Neighborhoods in North York

Let's create a function to repeat the same process to all the neighborhoods in North York

In [22]:
```
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={}&limit={}'.forma
            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)
```

Now write the code to run the above function on each neighborhood and create a new dataframe called *neighborhoods_to_venues*.

In [23]:
```
neighborhoods_to_venues = getNearbyVenues(names=df_neighborhoods_to['Neighborhood'],
                                          latitudes=df_neighborhoods_to['Latitude'],
                                          longitudes=df_neighborhoods_to['Longitude']
                                         )
```

Let's check the size of the resulting dataframe

In [24]:
```
print(neighborhoods_to_venues.shape)
neighborhoods_to_venues.head()
```

```
(322, 7)
```

Out[24]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Parkwoods	43.753259	-79.329656	Brookbanks Park	43.751976	-79.332140	Park
1	Parkwoods	43.753259	-79.329656	Variety Store	43.751974	-79.333114	Food & Drink Shop
2	Victoria Village	43.725882	-79.315572	Victoria Village Arena	43.723481	-79.315635	Hockey Arena
3	Victoria Village	43.725882	-79.315572	Tim Hortons	43.725517	-79.313103	Coffee Shop
4	Victoria Village	43.725882	-79.315572	Portugril	43.725819	-79.312785	Portuguese Restaurant

Let's check how many venues were returned for each neighborhood

In [25]:
```
neighborhoods_to_venues.groupby('Neighborhood').count()
```

Out[25]:

Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Bathurst Manor / Wilson Heights / Downsview North	19	19	19	19	19	19
Bayview Village	4	4	4	4	4	4
Bedford Park / Lawrence Manor East	24	24	24	24	24	24
Caledonia-Fairbanks	4	4	4	4	4	4
Del Ray / Mount Dennis / Keelsdale and Silverthorn	4	4	4	4	4	4
Don Mills	27	27	27	27	27	27
Downsview	15	15	15	15	15	15
East Toronto	3	3	3	3	3	3
Fairview / Henry Farm / Oriole	50	50	50	50	50	50
Glencairn	5	5	5	5	5	5
Hillcrest Village	5	5	5	5	5	5
Humber Summit	1	1	1	1	1	1
Humberlea / Emery	2	2	2	2	2	2
Humewood-Cedarvale	4	4	4	4	4	4
Lawrence Manor / Lawrence Heights	17	17	17	17	17	17
Leaside	32	32	32	32	32	32
North Park / Maple Leaf Park / Upwood Park	4	4	4	4	4	4
Northwood Park / York University	5	5	5	5	5	5
Parkview Hill / Woodbine Gardens	11	11	11	11	11	11

Parkwoods	2	2	2	2	2	2
Runnymede / The Junction North	3	3	3	3	3	3
Thorncliffe Park	20	20	20	20	20	20
Victoria Village	6	6	6	6	6	6
Weston	1	1	1	1	1	1
Willowdale	39	39	39	39	39	39
Woodbine Heights	10	10	10	10	10	10
York Mills West	5	5	5	5	5	5

Let's find out how many unique categories can be curated from all the returned venues

In [26]:
```
print('There are {} uniques categories.'.format(len(neighborhoods_to_venues['Venue Category'].unique())))
```

```
There are 118 uniques categories.
```

3. Analyze Each Neighborhood

In [27]:
```
# one hot encoding
neighborhoods_to_onehot = pd.get_dummies(neighborhoods_to_venues[['Venue Category']], prefix="", prefix_sep="")

# add neighborhood column back to dataframe
neighborhoods_to_onehot['Neighborhood'] = neighborhoods_to_venues['Neighborhood']

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

neighborhoods_to_onehot.head()
```

Out[27]:

	Neighborhood	Accessories Store	Airport	American Restaurant	Arts & Crafts Store	Asian Restaurant	Athletics & Sports	Bagel Shop	Bakery	Bank	Bar	Baseball Field	Basketball Court	Beer Store	Bike Shop	Boutique	Bre
0	Parkwoods	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	Parkwoods	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	Victoria Village	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	Victoria Village	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	Victoria Village	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

And let's examine the new dataframe size.

In [28]:
```
neighborhoods_to_onehot.shape
```

Out[28]:
```
(322, 119)
```

Next, let's group rows by neighborhood and by taking the mean of the frequency of occurrence of each category

In [29]:
```
neighborhoods_to_grouped = neighborhoods_to_onehot.groupby('Neighborhood').mean().reset_index()
neighborhoods_to_grouped.head()
```

Out[29]:

	Arts

	Neighborhood	Accessories Store	Airport	American Restaurant	& Crafts Store	Asian Restaurant	Athletics & Sports	Bagel Shop	Bakery	Bank	Bar	Baseball Field	Basketball Court	Beer Store	Bike Shop	Boutique
0	Bathurst Manor / Wilson Heights / Downsview North	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.105263	0.0	0.0	0.0	0.0	0.0	0.0
1	Bayview Village	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.250000	0.0	0.0	0.0	0.0	0.0	0.0
2	Bedford Park / Lawrence Manor East	0.0	0.0	0.041667	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0
3	Caledonia-Fairbanks	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0
4	Del Ray / Mount Dennis / Keelsdale and Silvert...	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0

Let's confirm the new size

In [30]:
```
neighborhoods_to_grouped.shape
```

Out[30]:
```
(27, 119)
```

Let's print each neighborhood along with the top 10 most common venues and put that into a *pandas* dataframe

In [31]:
```
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]

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'] = neighborhoods_to_grouped['Neighborhood']

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

neighborhoods_venues_sorted.head()
```

Out[31]:

	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	Bathurst Manor / Wilson Heights / Downsview North	Coffee Shop	Bank	Supermarket	Middle Eastern Restaurant	Pharmacy	Pizza Place	Deli / Bodega	Bridal Shop	Restaurant	Sandwich Place

1	Bayview Village	Japanese Restaurant	Chinese Restaurant	Café	Bank	Dog Run	Construction & Landscaping	Convenience Store	Cosmetics Shop	Curling Ice	Dance Studio
2	Bedford Park / Lawrence Manor East	Coffee Shop	Italian Restaurant	Restaurant	Sushi Restaurant	Sandwich Place	Café	Liquor Store	Butcher	Pharmacy	Pizza Place
3	Caledonia-Fairbanks	Park	Spa	Women's Store	Frozen Yogurt Shop	Furniture / Home Store	Comfort Food Restaurant	Concert Hall	Construction & Landscaping	Convenience Store	Cosmetics Shop
4	Del Ray / Mount Dennis / Keelsdale and Silvert...	Fast Food Restaurant	Coffee Shop	Sandwich Place	Diner	Comfort Food Restaurant	Concert Hall	Construction & Landscaping	Convenience Store	Cosmetics Shop	Curling Ice

4. Cluster Neighborhoods

Run *k*-means to cluster the neighborhood into 3 clusters.

In [33]:
```
# set number of clusters
kclusters = 3

neighborhoods_to_grouped_clustering = neighborhoods_to_grouped.drop('Neighborhood', 1)

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

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

Out[33]:
```
array([0, 0, 0, 2, 0, 0, 0, 0, 0, 0])
```

Let's create a new dataframe that includes the cluster as well as the top 10 venues for each neighborhood.

In [34]:
```
# add clustering labels
neighborhoods_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)

neighborhoods_to_merged = df_neighborhoods_to

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

neighborhoods_to_merged.head(20) # check the last columns!
```

Out[34]:

	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	7t Cc
0	M3A	North York	Parkwoods	43.753259	-79.329656	2.0	Park	Food & Drink Shop	Yoga Studio	Discount Store	Concert Hall	Construction & Landscaping	Conve
1	M4A	North York	Victoria Village	43.725882	-79.315572	0.0	Pizza Place	Coffee Shop	French Restaurant	Intersection	Portuguese Restaurant	Hockey Arena	Yoga
2	M6A	North York	Lawrence Manor / Lawrence Heights	43.718518	-79.464763	0.0	Clothing Store	Furniture / Home Store	Accessories Store	Event Space	Shoe Store	Miscellaneous Shop	S Good
3	M3B	North York	Don Mills	43.745906	-79.352188	0.0	Beer Store	Restaurant	Coffee Shop	Asian Restaurant	Japanese Restaurant	Gym	C Res
4	M4B	East York	Parkview Hill / Woodbine Gardens	43.706397	-79.309937	0.0	Pizza Place	Pet Store	Gym / Fitness Center	Pharmacy	Intersection	Fast Food Restaurant	

5	M6B	North York	Glencairn	43.709577	-79.445073	0.0	Japanese Restaurant	Pizza Place	Pub	Metro Station	Park	Yoga Studio	Res
6	M3C	North York	Don Mills	43.725900	-79.340923	0.0	Beer Store	Restaurant	Coffee Shop	Asian Restaurant	Japanese Restaurant	Gym	Res
7	M4C	East York	Woodbine Heights	43.695344	-79.318389	0.0	Pharmacy	Beer Store	Cosmetics Shop	Curling Ice	Dance Studio	Diner	Skati
8	M6C	York	Humewood-Cedarvale	43.693781	-79.428191	0.0	Dog Run	Field	Trail	Hockey Arena	Deli / Bodega	Diner	Di Res
9	M6E	York	Caledonia-Fairbanks	43.689026	-79.453512	2.0	Park	Spa	Women's Store	Frozen Yogurt Shop	Furniture / Home Store	Comfort Food Restaurant	Conc
10	M4G	East York	Leaside	43.709060	-79.363452	0.0	Sporting Goods Shop	Coffee Shop	Furniture / Home Store	Burger Joint	Bank	Sports Bar	
11	M2H	North York	Hillcrest Village	43.803762	-79.363452	0.0	Golf Course	Athletics & Sports	Pool	Mediterranean Restaurant	Dog Run	French Restaurant	
12	M3H	North York	Bathurst Manor / Wilson Heights / Downsview North	43.754328	-79.442259	0.0	Coffee Shop	Bank	Supermarket	Middle Eastern Restaurant	Pharmacy	Pizza Place	
13	M4H	East York	Thorncliffe Park	43.705369	-79.349372	0.0	Indian Restaurant	Yoga Studio	Sandwich Place	Gas Station	Fast Food Restaurant	Grocery Store	
14	M2J	North York	Fairview / Henry Farm / Oriole	43.778517	-79.346556	0.0	Clothing Store	Coffee Shop	Fast Food Restaurant	Bank	Tea Room	Restaurant	
15	M3J	North York	Northwood Park / York University	43.767980	-79.487262	0.0	Coffee Shop	Miscellaneous Shop	Caribbean Restaurant	Bar	Massage Studio	Discount Store	Cons Lands
16	M4J	East York	East Toronto	43.685347	-79.338106	0.0	Park	Convenience Store	Coffee Shop	Yoga Studio	Discount Store	Concert Hall	Cons Lands
17	M2K	North York	Bayview Village	43.786947	-79.385975	0.0	Japanese Restaurant	Chinese Restaurant	Café	Bank	Dog Run	Construction & Landscaping	Conve
18	M3K	North York	Downsview	43.737473	-79.464763	0.0	Park	Grocery Store	Baseball Field	Bank	Shopping Mall	Business Service	Liqu
19	M2L	North York	York Mills / Silver Hills	43.757490	-79.374714	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

In [35]:
```
print(neighborhoods_to_merged.dtypes)
```

```
PostalCode                object
Borough                   object
Neighborhood              object
Latitude                 float64
Longitude                float64
Cluster Labels           float64
1st Most Common Venue     object
2nd Most Common Venue     object
3rd Most Common Venue     object
4th Most Common Venue     object
5th Most Common Venue     object
6th Most Common Venue     object
7th Most Common Venue     object
8th Most Common Venue     object
9th Most Common Venue     object
10th Most Common Venue    object
dtype: object
```

In [36]:
```
neighborhoods_to_merged = neighborhoods_to_merged[~neighborhoods_to_merged["Cluster Labels"].isin(['NaN'])]
neighborhoods_to_merged['Cluster Labels'] = neighborhoods_to_merged['Cluster Labels'].astype(np.int64)
print(neighborhoods_to_merged.dtypes)
```

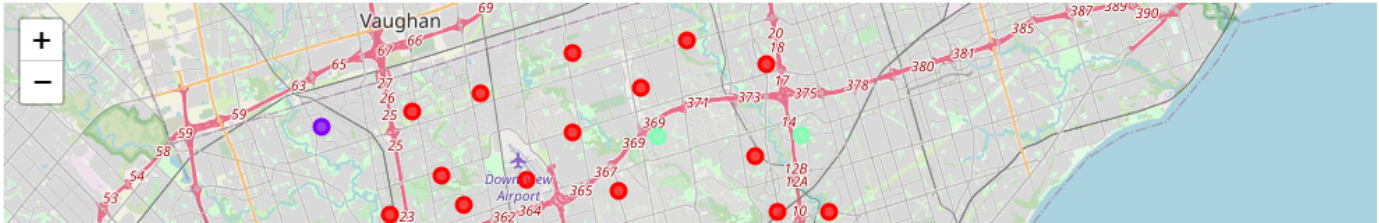
```
PostalCode                object
Borough                   object
Neighborhood              object
Latitude                 float64
Longitude                float64
Cluster Labels             int64
1st Most Common Venue     object
2nd Most Common Venue     object
3rd Most Common Venue     object
4th Most Common Venue     object
5th Most Common Venue     object
6th Most Common Venue     object
7th Most Common Venue     object
8th Most Common Venue     object
9th Most Common Venue     object
10th Most Common Venue    object
dtype: object
```

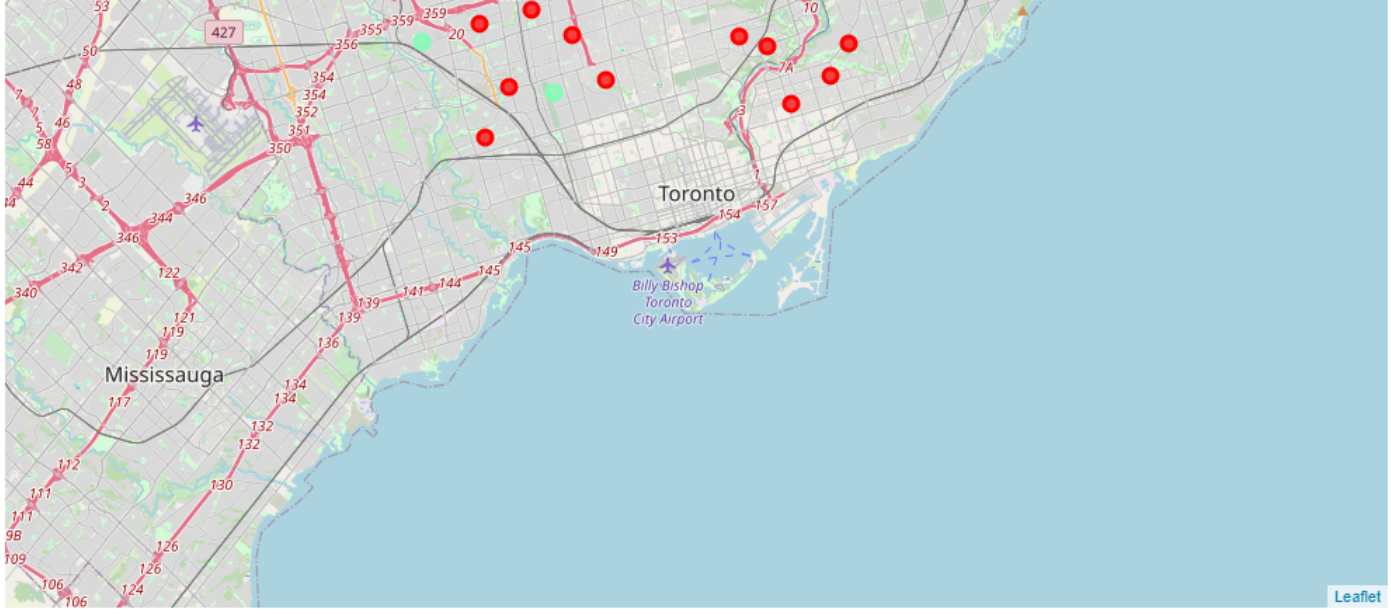
In [ ]:
```
```

Finally, let's visualize the resulting clusters

In [37]:
```
# create map
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(
        neighborhoods_to_merged['Latitude'],
        neighborhoods_to_merged['Longitude'],
        neighborhoods_to_merged['Neighborhood'],
        neighborhoods_to_merged['Cluster Labels']):
    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)

map_clusters
```

Out[37]:





5. Examine Clusters

Lets examine each cluster and determine the discriminating venue categories that distinguish each cluster. Based on the defining categories, we can then assign a name to each cluster.

Cluster 1

In [38]:
```
neighborhoods_to_merged.loc[neighborhoods_to_merged['Cluster Labels'] == 0, neighborhoods_to_merged.columns[[1] + list(range(5, ne
```

Out[38]:

	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
1	North York	0	Pizza Place	Coffee Shop	French Restaurant	Intersection	Portuguese Restaurant	Hockey Arena	Yoga Studio	Dance Studio	Department Store	Deli / Bodega
2	North York	0	Clothing Store	Furniture / Home Store	Accessories Store	Event Space	Shoe Store	Miscellaneous Shop	Sporting Goods Shop	Boutique	Coffee Shop	Women's Store
3	North York	0	Beer Store	Restaurant	Coffee Shop	Asian Restaurant	Japanese Restaurant	Gym	Chinese Restaurant	Discount Store	Sandwich Place	Café
4	East York	0	Pizza Place	Pet Store	Gym / Fitness Center	Pharmacy	Intersection	Fast Food Restaurant	Bank	Breakfast Spot	Gastropub	Athletics & Sports
5	North York	0	Japanese Restaurant	Pizza Place	Pub	Metro Station	Park	Yoga Studio	Comfort Food Restaurant	Concert Hall	Construction & Landscaping	Convenience Store
6	North York	0	Beer Store	Restaurant	Coffee Shop	Asian Restaurant	Japanese Restaurant	Gym	Chinese Restaurant	Discount Store	Sandwich Place	Café
7	East York	0	Pharmacy	Beer Store	Cosmetics Shop	Curling Ice	Dance Studio	Diner	Skating Rink	Park	Athletics & Sports	Video Store
8	York	0	Dog Run	Field	Trail	Hockey Arena	Deli / Bodega	Diner	Dim Sum Restaurant	Dessert Shop	Department Store	Yoga Studio

10	East York	0	Sporting Goods Shop	Coffee Shop	Furniture / Home Store	Burger Joint	Bank	Sports Bar	Dessert Shop	Brewery	Restaurant	Breakfast Spot
11	North York	0	Golf Course	Athletics & Sports	Pool	Mediterranean Restaurant	Dog Run	French Restaurant	Fried Chicken Joint	Concert Hall	Construction & Landscaping	Furniture / Home Store
12	North York	0	Coffee Shop	Bank	Supermarket	Middle Eastern Restaurant	Pharmacy	Pizza Place	Deli / Bodega	Bridal Shop	Restaurant	Sandwich Place
13	East York	0	Indian Restaurant	Yoga Studio	Sandwich Place	Gas Station	Fast Food Restaurant	Grocery Store	Gym	Discount Store	Housing Development	Liquor Store
14	North York	0	Clothing Store	Coffee Shop	Fast Food Restaurant	Bank	Tea Room	Restaurant	Movie Theater	Liquor Store	Burger Joint	Pharmacy
15	North York	0	Coffee Shop	Miscellaneous Shop	Caribbean Restaurant	Bar	Massage Studio	Discount Store	Construction & Landscaping	Convenience Store	Cosmetics Shop	Curling Ice
16	East York	0	Park	Convenience Store	Coffee Shop	Yoga Studio	Discount Store	Concert Hall	Construction & Landscaping	Cosmetics Shop	Curling Ice	Dance Studio
17	North York	0	Japanese Restaurant	Chinese Restaurant	Café	Bank	Dog Run	Construction & Landscaping	Convenience Store	Cosmetics Shop	Curling Ice	Dance Studio
18	North York	0	Park	Grocery Store	Baseball Field	Bank	Shopping Mall	Business Service	Liquor Store	Hotel	Gym / Fitness Center	Discount Store
20	North York	0	Park	Grocery Store	Baseball Field	Bank	Shopping Mall	Business Service	Liquor Store	Hotel	Gym / Fitness Center	Discount Store
21	North York	0	Park	Construction & Landscaping	Bakery	Basketball Court	Dog Run	Concert Hall	Convenience Store	Cosmetics Shop	Curling Ice	Dance Studio
24	North York	0	Park	Grocery Store	Baseball Field	Bank	Shopping Mall	Business Service	Liquor Store	Hotel	Gym / Fitness Center	Discount Store
25	North York	0	Coffee Shop	Italian Restaurant	Restaurant	Sushi Restaurant	Sandwich Place	Café	Liquor Store	Butcher	Pharmacy	Pizza Place
26	York	0	Fast Food Restaurant	Coffee Shop	Sandwich Place	Diner	Comfort Food Restaurant	Concert Hall	Construction & Landscaping	Convenience Store	Cosmetics Shop	Curling Ice
27	North York	0	Paper / Office Supplies Store	Baseball Field	Yoga Studio	Discount Store	Concert Hall	Construction & Landscaping	Convenience Store	Cosmetics Shop	Curling Ice	Dance Studio
28	North York	0	Ramen Restaurant	Pizza Place	Coffee Shop	Café	Sandwich Place	Restaurant	Bubble Tea Shop	Plaza	Pharmacy	Pet Store
29	North York	0	Park	Grocery Store	Baseball Field	Bank	Shopping Mall	Business Service	Liquor Store	Hotel	Gym / Fitness Center	Discount Store
30	York	0	Caribbean Restaurant	Bus Line	Brewery	Yoga Studio	Discount Store	Construction & Landscaping	Convenience Store	Cosmetics Shop	Curling Ice	Dance Studio
33	North York	0	Ramen Restaurant	Pizza Place	Coffee Shop	Café	Sandwich Place	Restaurant	Bubble Tea Shop	Plaza	Pharmacy	Pet Store

Cluster 2

In [39]:
```
neighborhoods_to_merged.loc[neighborhoods_to_merged['Cluster Labels'] == 1, neighborhoods_to_merged.columns[[1] + list(range(5, ne
```

Out[39]:

	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

22	North York	1	Pizza Place	Discount Store	Comfort Food Restaurant	Concert Hall	Construction & Landscaping	Convenience Store	Cosmetics Shop	Curling Ice	Dance Studio	Deli / Bodega

Cluster 3

In [40]:
```
neighborhoods_to_merged.loc[neighborhoods_to_merged['Cluster Labels'] == 2, neighborhoods_to_merged.columns[[1] + list(range(5, ne
```

Out[40]:

	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
0	North York	2	Park	Food & Drink Shop	Yoga Studio	Discount Store	Concert Hall	Construction & Landscaping	Convenience Store	Cosmetics Shop	Curling Ice	Dance Studio
9	York	2	Park	Spa	Women's Store	Frozen Yogurt Shop	Furniture / Home Store	Comfort Food Restaurant	Concert Hall	Construction & Landscaping	Convenience Store	Cosmetics Shop
31	York	2	Park	Yoga Studio	Discount Store	Concert Hall	Construction & Landscaping	Convenience Store	Cosmetics Shop	Curling Ice	Dance Studio	Deli / Bodega
32	North York	2	Park	Convenience Store	Bank	Bar	Dog Run	Concert Hall	Construction & Landscaping	Cosmetics Shop	Curling Ice	Dance Studio

Segmenting and Clustering Neighborhoods in Waterloo

1. Download and Explore Dataset

Load and explore the data

Tranform the data into a *pandas* dataframe and eliminate Coulumn with no name

In [41]:
```
# read Waterloo dataset
wneighborhoods = pd.read_csv('waterloo_Geospatial_Coordinates.csv',sep=",")
```

In [42]:
```
# check the data
wneighborhoods.head()
```

Out[42]:

	Postal code	Borough	Neighborhood	Unnamed: 3	Latitude	Longitude
0	N2L 5Y9	Waterloo South	Beechwood	NaN	43.458418	-80.557473
1	N2T 2Y2	Waterloo West	Clair Hills	NaN	43.454848	-80.579832
2	N2K 3H7	Waterloo East	Colonial Acres	NaN	43.507148	-80.520306
3	N2L 5Y9	Waterloo West	COLUMBIA FOREST	NaN	43.458418	-80.557473
4	N2V 2S2	Waterloo North	Conservation Meadows	NaN	43.490019	-80.58534

Drop Column with no name and examine the resulting dataframe.

In [43]:
```
wneighborhoods.drop(wneighborhoods.columns[[3]], axis=1, inplace=True)
wneighborhoods.head(18)
```

Out[43]:

	Postal code	Borough	Neighborhood	Latitude	Longitude
0	N2L 5Y9	Waterloo South	Beechwood	43.458418	-80.557473
1	N2T 2Y2	Waterloo West	Clair Hills	43.454848	-80.579832
2	N2K 3H7	Waterloo East	Colonial Acres	43.507148	-80.520306
3	N2L 5Y9	Waterloo West	COLUMBIA FOREST	43.458418	-80.557473
4	N2V 2S2	Waterloo North	Conservation Meadows	43.490019	-80.58534
5	N2K 3N8	Waterloo East	KIWANIS PARK	43.501733	-80.479472
6	N2T 2T6	Waterloo	LAURELWOOD	43.466481	-80.582579
7	N2K 2X1	Waterloo East	LEXINGTON	43.502022	-80.500654
8	N2J 3W2	Waterloo	LINCOLN HEIGHTS	43.479862	-80.503212
9	N2K 1V9	Waterloo	LINCOLN VILLAGE	43.490705	-80.501044
10	N2L 2N3	Waterloo	MAPLE HILLS	43.450887	-80.546273
11	N2V 1Y6	Waterloo North	NORTH LAKESHORE	43.499129	-80.563996
12	N2K 3X2	Waterloo	UNIVERSITY DOWNS	43.489621	-80.487291
13	N2T 1S8	Waterloo	UPPER BEECHWOOD	43.462144	-80.569948
14	N2J 2Z4	Waterloo	UPTOWN WATERLOO	43.483673	-80.526867
15	N2V 0B3	Waterloo West	VISTA HILLS	43.451008	-80.591636
16	N2L 2G8	Waterloo	WESTMOUNT	43.459421	-80.529979
17	N2T 1Y5	Waterloo West	WESTVALE	43.445358	-80.557746

Convert Longitude from object to float64

In [44]:
```
wneighborhoods["Longitude"] = wneighborhoods["Longitude"].astype(str).astype(float)
print(wneighborhoods.dtypes)
```

```
Postal code      object
Borough          object
Neighborhood     object
Latitude        float64
Longitude       float64
dtype: object
```

And make sure that the dataset has all 5 boroughs and 18 neighborhoods.

In [45]:
```
print('The dataframe has {} boroughs and {} neighborhoods.'.format(
        len(wneighborhoods['Borough'].unique()),
        wneighborhoods.shape[0]
    )
)
```

```
The dataframe has 5 boroughs and 18 neighborhoods.
```

Use geopy library to get the latitude and longitude values of Waterloo.

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

In [46]:
```
address = "Waterloo, ON"

geolocator = Nominatim(user_agent="Waterloo_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
```

```
print('The geograpical coordinate of Waterloo city are {}, {}.'.format(latitude, longitude))
```

```
The geograpical coordinate of Waterloo city are 43.466874, -80.524635.
```

Check Latitude and the longitude coordinates of neighborhood

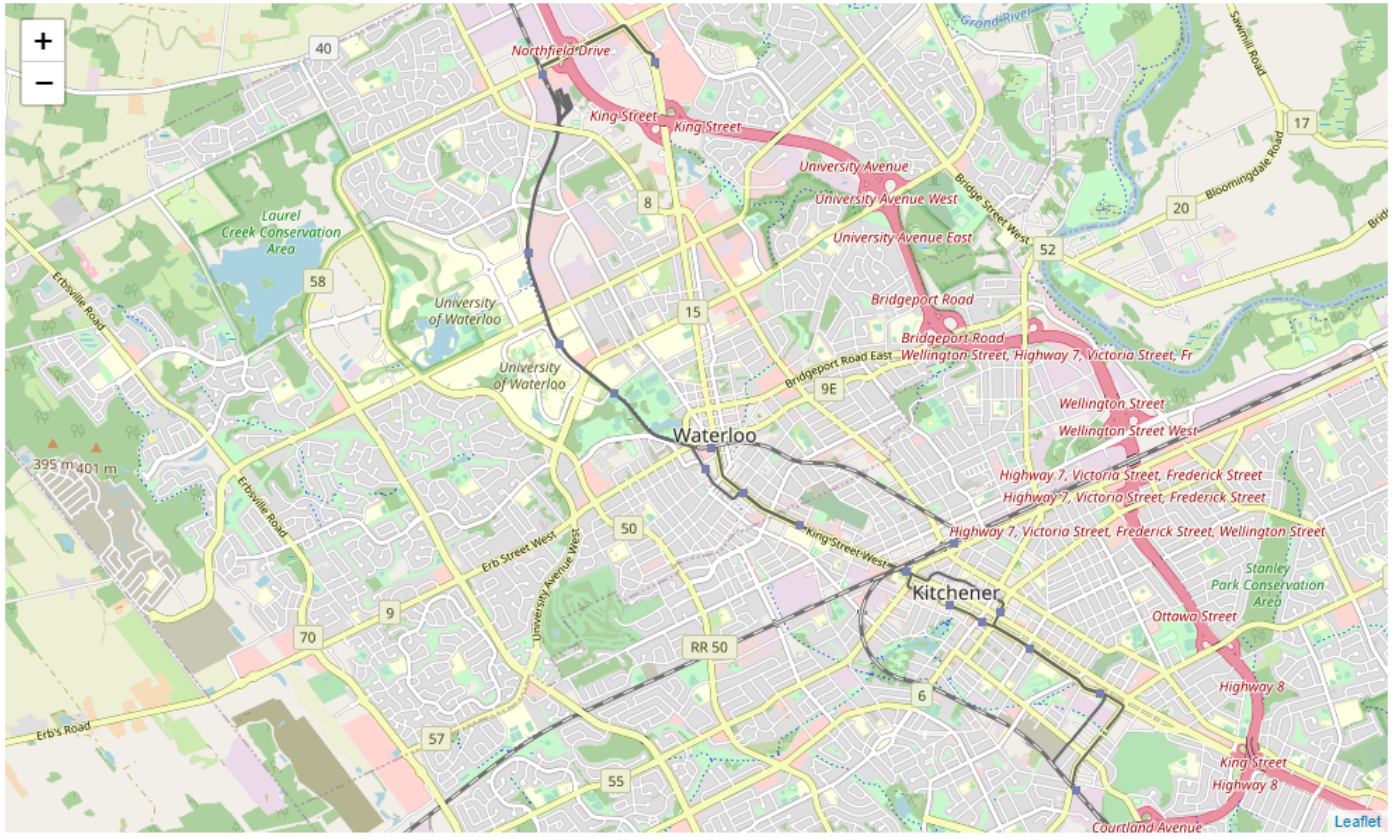
In [47]:
```
wneighborhoods.head()
```

Out[47]:

	Postal code	Borough	Neighborhood	Latitude	Longitude
0	N2L 5Y9	Waterloo South	Beechwood	43.458418	-80.557473
1	N2T 2Y2	Waterloo West	Clair Hills	43.454848	-80.579832
2	N2K 3H7	Waterloo East	Colonial Acres	43.507148	-80.520306
3	N2L 5Y9	Waterloo West	COLUMBIA FOREST	43.458418	-80.557473
4	N2V 2S2	Waterloo North	Conservation Meadows	43.490019	-80.585340

In [48]:
```
# create map of Waterloo using latitude and longitude values
map_neighborhoods = folium.Map(location=[latitude, longitude], zoom_start=13)
map_neighborhoods
```

Out[48]:

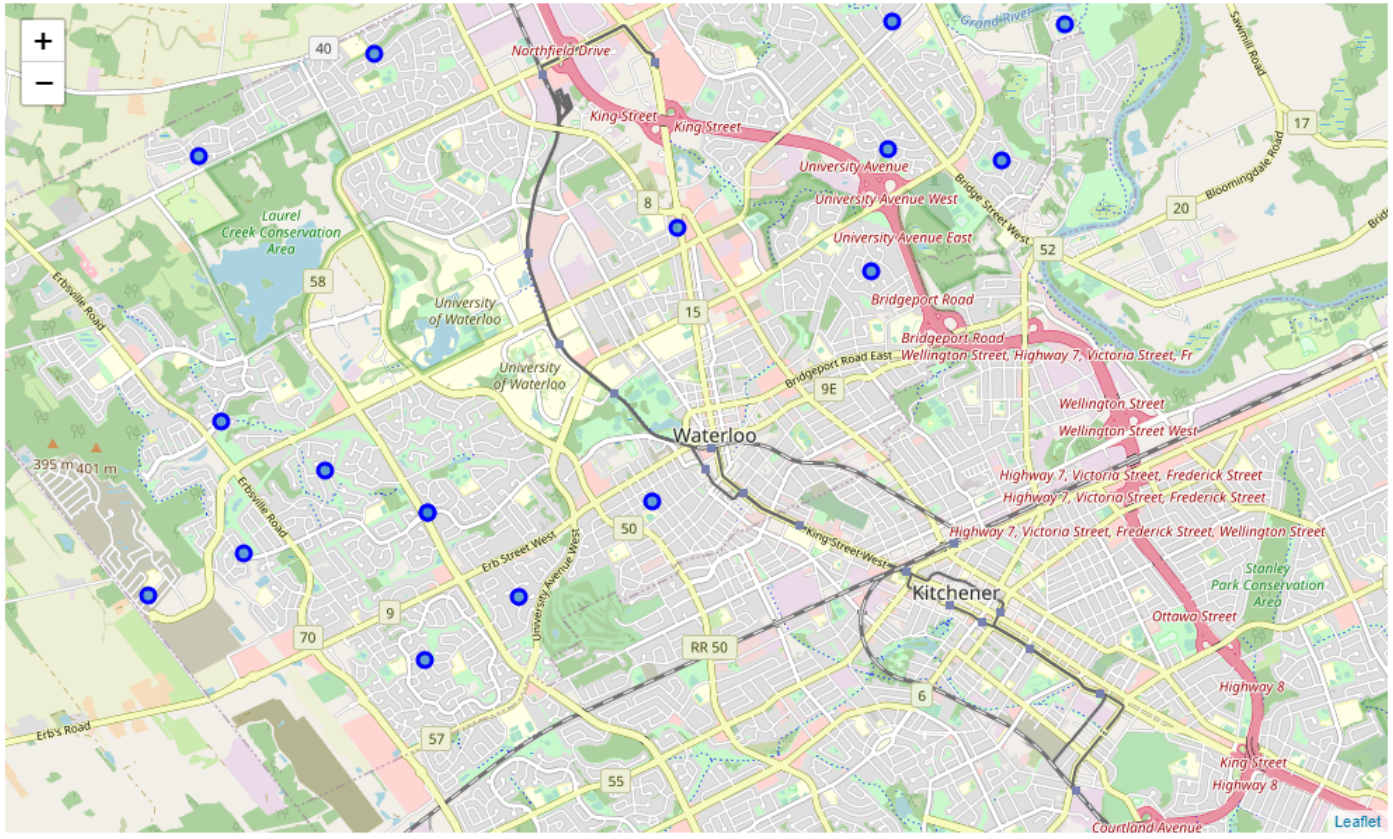


Add markers to Waterloo map.

In [49]:
```
# Cretae a new data fraim
df_neighborhoods = wneighborhoods
```

```
for lat, lng, borough, neighborhood in zip(
        df_neighborhoods['Latitude'],
        df_neighborhoods['Longitude'],
        df_neighborhoods['Borough'],
        df_neighborhoods['Neighborhood']):
    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,
        parse_html=False).add_to(map_neighborhoods)

map_neighborhoods
```

Out[49]:



Create a data frame with boroughs that contain the word "Waterloo".

In [50]:
```
df_neighborhoods_to = df_neighborhoods[df_neighborhoods['Borough'].str.contains("Waterloo")].reset_index(drop=True)
df_neighborhoods_to.head()
```

Out[50]:

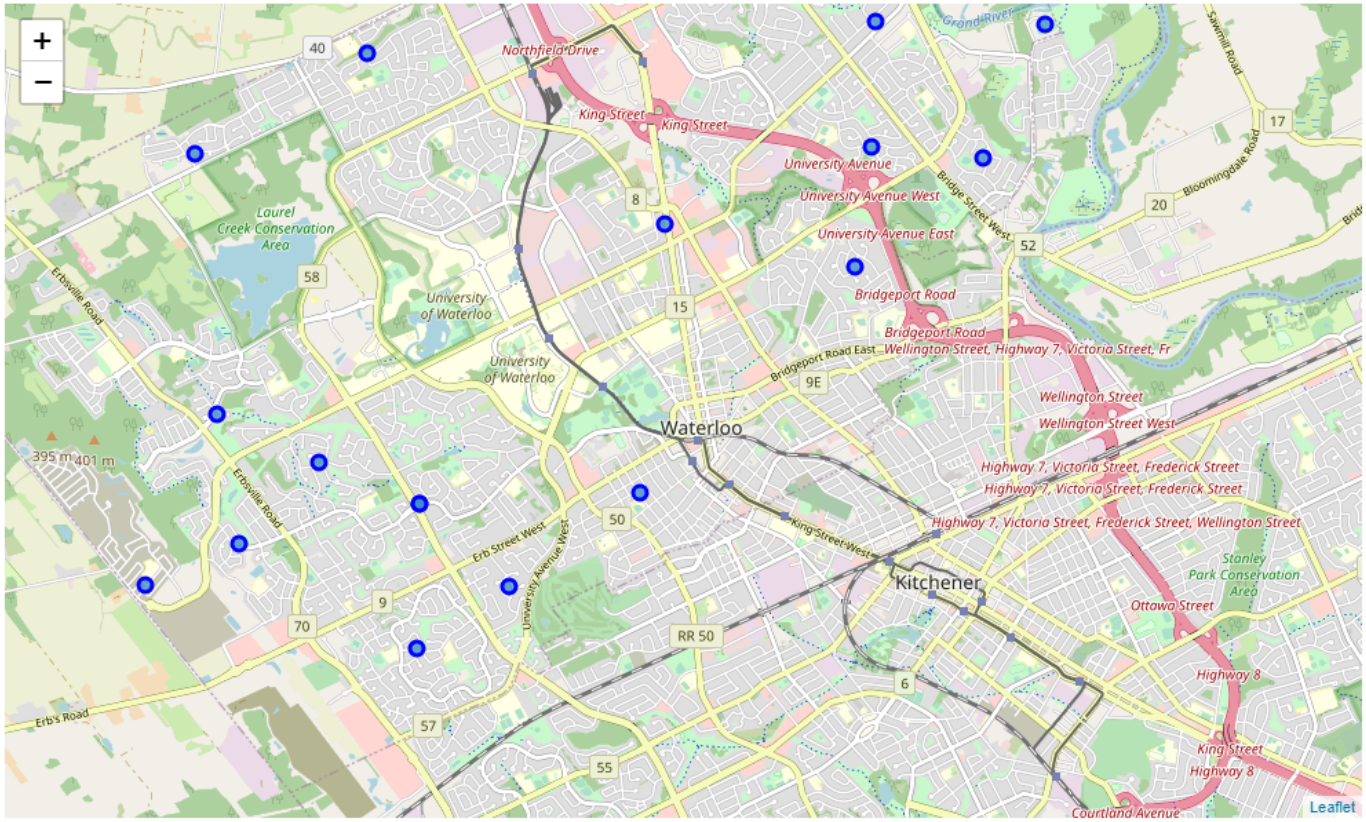
	Postal code	Borough	Neighborhood	Latitude	Longitude
0	N2L 5Y9	Waterloo South	Beechwood	43.458418	-80.557473
1	N2T 2Y2	Waterloo West	Clair Hills	43.454848	-80.579832

2	N2K 3H7	Waterloo East	Colonial Acres	43.507148	-80.520306
3	N2L 5Y9	Waterloo West	COLUMBIA FOREST	43.458418	-80.557473
4	N2V 2S2	Waterloo North	Conservation Meadows	43.490019	-80.585340

create Neighborhood map

In [51]:
```
map_neighborhoods_to = folium.Map(location=[latitude, longitude], zoom_start=13)
for lat, lng, borough, neighborhood in zip(
        df_neighborhoods_to['Latitude'],
        df_neighborhoods_to['Longitude'],
        df_neighborhoods_to['Borough'],
        df_neighborhoods_to['Neighborhood']):
    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,
        parse_html=False).add_to(map_neighborhoods_to)

map_neighborhoods_to
```

Out[51]:



Next, we are going to start utilizing the Foursquare API to explore the neighborhoods and segment them.

Let's explore the first neighborhood in our dataframe.

Get the neighborhood's name.

In [52]:
```
neighborhood_name = df_neighborhoods_to.loc[0, 'Neighborhood']
print(f"The first neighborhood's name is '{neighborhood_name}'.")

neighborhood_latitude = df_neighborhoods_to.loc[0, 'Latitude']
neighborhood_longitude = df_neighborhoods_to.loc[0, 'Longitude']

print('Latitude and longitude values of {} are {}, {}.'.format(neighborhood_name,
                                                               neighborhood_latitude,
                                                               neighborhood_longitude))
```

```
The first neighborhood's name is 'Beechwood'.
Latitude and longitude values of Beechwood are 43.458418, -80.557473.
```

In [53]:
```
neighborhood_latitude = df_neighborhoods_to.loc[0, 'Latitude'] # neighborhood latitude value
neighborhood_longitude = df_neighborhoods_to.loc[0, 'Longitude'] # neighborhood longitude value
radius = 500
neighborhood_name = df_neighborhoods_to.loc[0, 'Neighborhood'] # neighborhood name

print('Latitude and longitude values of {} are {}, {}.'.format(neighborhood_name,
                                                               neighborhood_latitude,
                                                               neighborhood_longitude))
```

```
Latitude and longitude values of Beechwood are 43.458418, -80.557473.
```

Now, let's get the top 50 venues that are in Beechwood within a radius of 2000 meters.

First, let's create the GET request URL. Name your URL **url**.

In [54]:
```
# limit of number of venues returned by Foursquare API
LIMIT = 50
# define radius
radius = 2000
url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    neighborhood_latitude,
    neighborhood_longitude,
    radius,
    LIMIT)

# get the result to a json file
results = requests.get(url).json()
```

Extract the category of the venue

From the Foursquare lab in the previous module, we know that all the information is in the *items* key. Before we proceed, let's borrow the **get_category_type** function from the Foursquare lab.

In [55]:
```
# 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']
```

Now we are ready to clean the json and structure it into a *pandas* dataframe.

In [56]:
```
venues = results['response']['groups'][0]['items']

nearby_venues = json_normalize(venues) # flatten JSON

# filter columns
filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat', 'venue.location.lng']
nearby_venues =nearby_venues.loc[:, filtered_columns]

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

# clean columns
nearby_venues.columns = [col.split(".")[-1] for col in nearby_venues.columns]

nearby_venues.head()
```

Out[56]:

	name	categories	lat	lng
0	Zehrs	Supermarket	43.453083	-80.555858
1	GoodLife Fitness	Gym	43.452983	-80.555471
2	Churchill Arms	Gastropub	43.454589	-80.546438
3	Starbucks	Coffee Shop	43.467966	-80.568014
4	Dutchies Fresh Market	Grocery Store	43.444885	-80.570749

And how many venues were returned by Foursquare?

In [57]:
```
print('{} venues were returned by Foursquare.'.format(nearby_venues.shape[0]))
```

```
50 venues were returned by Foursquare.
```

2. Explore Neighborhoods in Waterloo

Let's create a function to repeat the same process to all the neighborhoods in Waterloo

In [58]:
```
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={}&limit={}'.forma
            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)
```

Now write the code to run the above function on each neighborhood and create a new dataframe called *neighborhoods_to_venues*.

In [59]:
```
neighborhoods_to_venues = getNearbyVenues(names=df_neighborhoods_to['Neighborhood'],
                                          latitudes=df_neighborhoods_to['Latitude'],
                                          longitudes=df_neighborhoods_to['Longitude']
                                         )
```

Let's check the size of the resulting dataframe

In [60]:
```
print(neighborhoods_to_venues.shape)
neighborhoods_to_venues.head()
```

```
(76, 7)
```

Out[60]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Beechwood	43.458418	-80.557473	Regency Park	43.460033	-80.560102	Park
1	Beechwood	43.458418	-80.557473	Turtle pond toys	43.460430	-80.561360	Toy / Game Store
2	Beechwood	43.458418	-80.557473	Clair Lake Park	43.461903	-80.558080	Park
3	Beechwood	43.458418	-80.557473	Sargent Munchell Marketing Headquarters	43.457600	-80.562800	Brewery
4	Beechwood	43.458418	-80.557473	Centennial Baseball Field 2	43.461258	-80.552789	Baseball Field

Let's check how many venues were returned for each neighborhood

In [61]:
```
neighborhoods_to_venues.groupby('Neighborhood').count()
```

Out[61]:

Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Beechwood	5	5	5	5	5	5
COLUMBIA FOREST	5	5	5	5	5	5
Clair Hills	2	2	2	2	2	2
Colonial Acres	7	7	7	7	7	7

Conservation Meadows	2	2	2	2	2	2
KIWANIS PARK	5	5	5	5	5	5
LAURELWOOD	1	1	1	1	1	1
LEXINGTON	4	4	4	4	4	4
LINCOLN HEIGHTS	4	4	4	4	4	4
LINCOLN VILLAGE	2	2	2	2	2	2
MAPLE HILLS	3	3	3	3	3	3
NORTH LAKESHORE	4	4	4	4	4	4
UNIVERSITY DOWNS	8	8	8	8	8	8
UPPER BEECHWOOD	2	2	2	2	2	2
UPTOWN WATERLOO	16	16	16	16	16	16
VISTA HILLS	1	1	1	1	1	1
WESTMOUNT	3	3	3	3	3	3
WESTVALE	2	2	2	2	2	2

Let's find out how many unique categories can be curated from all the returned venues

In [62]:
```
print('There are {} uniques categories.'.format(len(neighborhoods_to_venues['Venue Category'].unique())))
```

```
There are 50 uniques categories.
```

3. Analyze Each Neighborhood in Waterloo

In [63]:
```
# one hot encoding
neighborhoods_to_onehot = pd.get_dummies(neighborhoods_to_venues[['Venue Category']], prefix="", prefix_sep="")

# add neighborhood column back to dataframe
neighborhoods_to_onehot['Neighborhood'] = neighborhoods_to_venues['Neighborhood']

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

neighborhoods_to_onehot.head()
```

Out[63]:

	Neighborhood	American Restaurant	Baseball Field	Bath House	Brewery	Burger Joint	Café	Chinese Restaurant	Clothing Store	Coffee Shop	College Classroom	Construction & Landscaping	Convenience Store	Dog Run	Dry Cleaner	R
0	Beechwood	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	Beechwood	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	Beechwood	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	Beechwood	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
4	Beechwood	0	1	0	0	0	0	0	0	0	0	0	0	0	0	

And let's examine the new dataframe size.

In [64]:
```
neighborhoods_to_onehot.shape
```

Out[64]: (76, 51)

Next, let's group rows by neighborhood and by taking the mean of the frequency of occurrence of each category

In [65]:
```
neighborhoods_to_grouped = neighborhoods_to_onehot.groupby('Neighborhood').mean().reset_index()
neighborhoods_to_grouped.head()
```

Out[65]:

	Neighborhood	American Restaurant	Baseball Field	Bath House	Brewery	Burger Joint	Café	Chinese Restaurant	Clothing Store	Coffee Shop	College Classroom	Construction & Landscaping	Convenience Store	Dog Run	Dry Cleaner	R
0	Beechwood	0.0	0.2	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	COLUMBIA FOREST	0.0	0.2	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	Clair Hills	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	Colonial Acres	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	Conservation Meadows	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

Let's confirm the new size

In [66]:
```
neighborhoods_to_grouped.shape
```

Out[66]:
```
(18, 51)
```

Let's print each neighborhood along with the top 10 most common venues and put that into a *pandas* dataframe

In [67]:
```
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]

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'] = neighborhoods_to_grouped['Neighborhood']

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

neighborhoods_venues_sorted.head(18)
```

Out[67]:

	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	Beechwood	Park	Baseball Field	Toy / Game Store	Brewery	Wings Joint	Dry Cleaner	Home Service	Grocery Store	Gastropub	Gas Station

1	COLUMBIA FOREST	Park	Baseball Field	Toy / Game Store	Brewery	Wings Joint	Dry Cleaner	Home Service	Grocery Store	Gastropub	Gas Station
2	Clair Hills	Playground	Food Truck	Wings Joint	Home Service	Grocery Store	Gastropub	Gas Station	Furniture / Home Store	Food & Drink Shop	Fast Food Restaurant
3	Colonial Acres	Furniture / Home Store	Food & Drink Shop	Indian Restaurant	Martial Arts Dojo	Pharmacy	Wings Joint	Home Service	Grocery Store	Gastropub	Gas Station
4	Conservation Meadows	Trailer Park	Home Service	Wings Joint	Convenience Store	Grocery Store	Gastropub	Gas Station	Furniture / Home Store	Food Truck	Food & Drink Shop
5	KIWANIS PARK	Playground	River	Dog Run	Pool	Home Service	Burger Joint	Baseball Field	Grocery Store	Gastropub	Gas Station
6	LAURELWOOD	Pub	Wings Joint	Convenience Store	Home Service	Grocery Store	Gastropub	Gas Station	Furniture / Home Store	Food Truck	Food & Drink Shop
7	LEXINGTON	Home Service	Bath House	Nature Preserve	Construction & Landscaping	Wings Joint	Dry Cleaner	Grocery Store	Gastropub	Gas Station	Furniture / Home Store
8	LINCOLN HEIGHTS	Pizza Place	Grocery Store	Shop & Service	Dry Cleaner	Wings Joint	Home Service	Gastropub	Gas Station	Furniture / Home Store	Food Truck
9	LINCOLN VILLAGE	Gas Station	Park	Wings Joint	Dog Run	Home Service	Grocery Store	Gastropub	Furniture / Home Store	Food Truck	Food & Drink Shop
10	MAPLE HILLS	Sushi Restaurant	Gastropub	Nail Salon	Wings Joint	Dog Run	Home Service	Grocery Store	Gas Station	Furniture / Home Store	Food Truck
11	NORTH LAKESHORE	Park	College Classroom	Fast Food Restaurant	Wings Joint	Dog Run	Home Service	Grocery Store	Gastropub	Gas Station	Furniture / Home Store
12	UNIVERSITY DOWNS	Convenience Store	Grocery Store	Mediterranean Restaurant	Sandwich Place	Chinese Restaurant	Coffee Shop	Pet Store	Pizza Place	Dry Cleaner	Gastropub
13	UPPER BEECHWOOD	Photography Studio	Pool	Wings Joint	Dog Run	Home Service	Grocery Store	Gastropub	Gas Station	Furniture / Home Store	Food Truck
14	UPTOWN WATERLOO	Restaurant	Wings Joint	Record Shop	Café	Clothing Store	Coffee Shop	Fast Food Restaurant	Warehouse Store	Korean Restaurant	Pizza Place
15	VISTA HILLS	Burger Joint	Wings Joint	Dog Run	Hotel	Home Service	Grocery Store	Gastropub	Gas Station	Furniture / Home Store	Food Truck
16	WESTMOUNT	Hotel	Gastropub	Nightclub	Dog Run	Home Service	Grocery Store	Gas Station	Furniture / Home Store	Food Truck	Food & Drink Shop
17	WESTVALE	Park	Wings Joint	Dog Run	Home Service	Grocery Store	Gastropub	Gas Station	Furniture / Home Store	Food Truck	Food & Drink Shop

4. Cluster Neighborhoods in Waterloo

In [68]:
```
# set number of clusters
kclusters = 3

neighborhoods_to_grouped_clustering = neighborhoods_to_grouped.drop('Neighborhood', 1)

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

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

Out[68]:
```
array([0, 0, 2, 2, 2, 2, 1, 2, 2, 0])
```

Let's create a new dataframe that includes the cluster as well as the top 10 venues for each neighborhood.

In [69]:
```
# add clustering labels
neighborhoods_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)

neighborhoods_to_merged = df_neighborhoods_to

# merge toronto_grouped with toronto_data to add latitude/longitude for each neighborhood
```

```
neighborhoods_to_merged = neighborhoods_to_merged.join(neighborhoods_venues_sorted.set_index('Neighborhood'), on='Neighborhood')

neighborhoods_to_merged.head(21) # check the last columns!
```

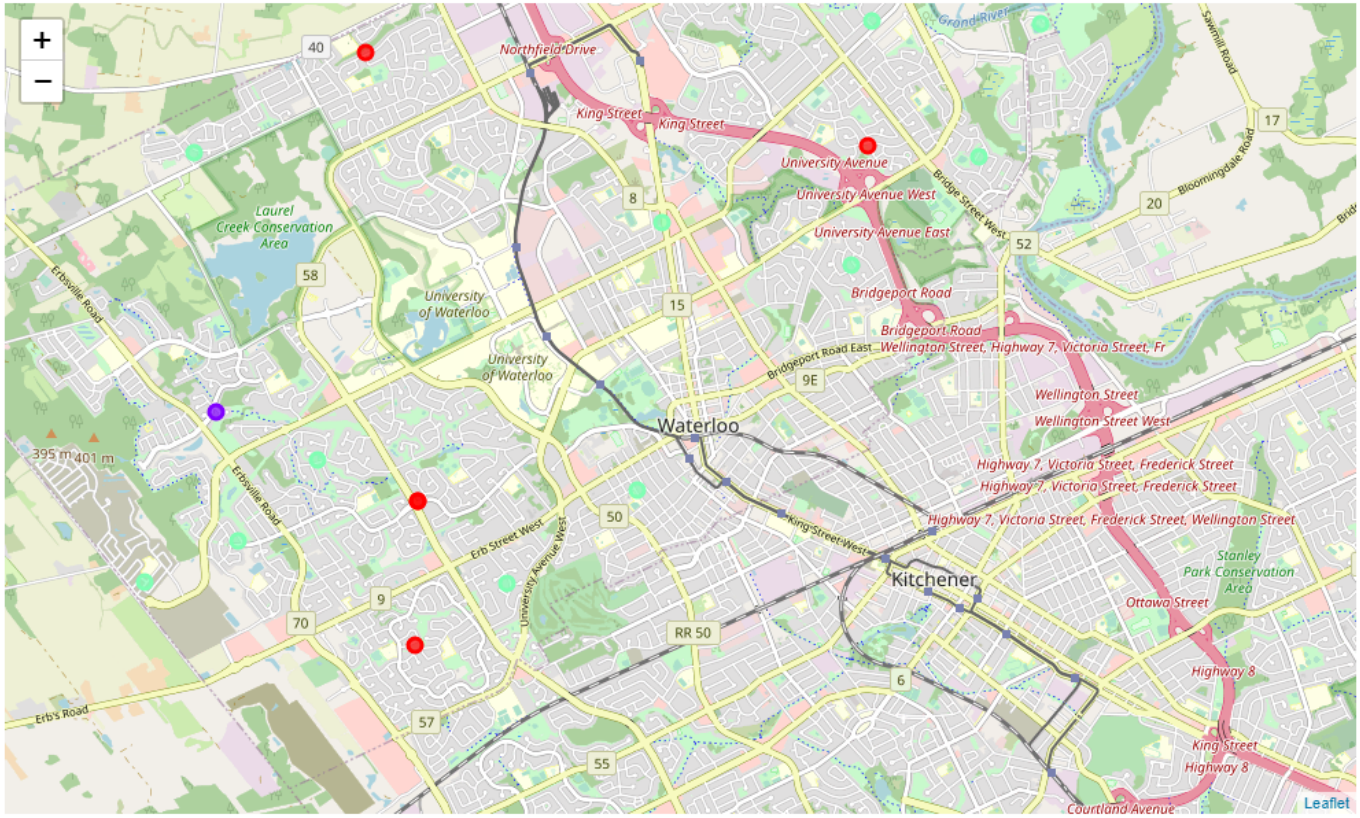
Out[69]:

	Postal code	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 Most Common Venue	8 C
0	N2L 5Y9	Waterloo South	Beechwood	43.458418	-80.557473	0	Park	Baseball Field	Toy / Game Store	Brewery	Wings Joint	Dry Cleaner	Home Service	
1	N2T 2Y2	Waterloo West	Clair Hills	43.454848	-80.579832	2	Playground	Food Truck	Wings Joint	Home Service	Grocery Store	Gastropub	Gas Station	Fu
2	N2K 3H7	Waterloo East	Colonial Acres	43.507148	-80.520306	2	Furniture / Home Store	Food & Drink Shop	Indian Restaurant	Martial Arts Dojo	Pharmacy	Wings Joint	Home Service	
3	N2L 5Y9	Waterloo West	COLUMBIA FOREST	43.458418	-80.557473	0	Park	Baseball Field	Toy / Game Store	Brewery	Wings Joint	Dry Cleaner	Home Service	
4	N2V 2S2	Waterloo North	Conservation Meadows	43.490019	-80.585340	2	Trailer Park	Home Service	Wings Joint	Convenience Store	Grocery Store	Gastropub	Gas Station	Fu
5	N2K 3N8	Waterloo East	KIWANIS PARK	43.501733	-80.479472	2	Playground	River	Dog Run	Pool	Home Service	Burger Joint	Baseball Field	
6	N2T 2T6	Waterloo	LAURELWOOD	43.466481	-80.582579	1	Pub	Wings Joint	Convenience Store	Home Service	Grocery Store	Gastropub	Gas Station	Fu
7	N2K 2X1	Waterloo East	LEXINGTON	43.502022	-80.500654	2	Home Service	Bath House	Nature Preserve	Construction & Landscaping	Wings Joint	Dry Cleaner	Grocery Store	Ga
8	N2J 3W2	Waterloo	LINCOLN HEIGHTS	43.479862	-80.503212	2	Pizza Place	Grocery Store	Shop & Service	Dry Cleaner	Wings Joint	Home Service	Gastropub	
9	N2K 1V9	Waterloo	LINCOLN VILLAGE	43.490705	-80.501044	0	Gas Station	Park	Wings Joint	Dog Run	Home Service	Grocery Store	Gastropub	Fu
10	N2L 2N3	Waterloo	MAPLE HILLS	43.450887	-80.546273	2	Sushi Restaurant	Gastropub	Nail Salon	Wings Joint	Dog Run	Home Service	Grocery Store	
11	N2V 1Y6	Waterloo North	NORTH LAKESHORE	43.499129	-80.563996	0	Park	College Classroom	Fast Food Restaurant	Wings Joint	Dog Run	Home Service	Grocery Store	Ga
12	N2K 3X2	Waterloo	UNIVERSITY DOWNS	43.489621	-80.487291	2	Convenience Store	Grocery Store	Mediterranean Restaurant	Sandwich Place	Chinese Restaurant	Coffee Shop	Pet Store	
13	N2T 1S8	Waterloo	UPPER BEECHWOOD	43.462144	-80.569948	2	Photography Studio	Pool	Wings Joint	Dog Run	Home Service	Grocery Store	Gastropub	
14	N2J 2Z4	Waterloo	UPTOWN WATERLOO	43.483673	-80.526867	2	Restaurant	Wings Joint	Record Shop	Café	Clothing Store	Coffee Shop	Fast Food Restaurant	Wa
15	N2V 0B3	Waterloo West	VISTA HILLS	43.451008	-80.591636	2	Burger Joint	Wings Joint	Dog Run	Hotel	Home Service	Grocery Store	Gastropub	
16	N2L 2G8	Waterloo	WESTMOUNT	43.459421	-80.529979	2	Hotel	Gastropub	Nightclub	Dog Run	Home Service	Grocery Store	Gas Station	Fu
17	N2T 1Y5	Waterloo West	WESTVALE	43.445358	-80.557746	0	Park	Wings Joint	Dog Run	Home Service	Grocery Store	Gastropub	Gas Station	Fu

Finally, let's visualize the resulting clusters

In [71]:
```
# create map
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=13)

# 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(
        neighborhoods_to_merged['Latitude'],
        neighborhoods_to_merged['Longitude'],
        neighborhoods_to_merged['Neighborhood'],
        neighborhoods_to_merged['Cluster Labels']):
    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)

map_clusters
```

Out[71]:



5. Examine Clusters

Lets examine each cluster and determine the discriminating venue categories that distinguish each cluster. Based on the defining categories, we can then assign a name to each cluster.

Cluster 1

In [72]:
```
neighborhoods_to_merged.loc[neighborhoods_to_merged['Cluster Labels'] == 0, neighborhoods_to_merged.columns[[1] + list(range(5, ne
```

Out[72]:

	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
0	Waterloo South	0	Park	Baseball Field	Toy / Game Store	Brewery	Wings Joint	Dry Cleaner	Home Service	Grocery Store	Gastropub	Gas Station
3	Waterloo West	0	Park	Baseball Field	Toy / Game Store	Brewery	Wings Joint	Dry Cleaner	Home Service	Grocery Store	Gastropub	Gas Station
9	Waterloo	0	Gas Station	Park	Wings Joint	Dog Run	Home Service	Grocery Store	Gastropub	Furniture / Home Store	Food Truck	Food & Drink Shop
11	Waterloo North	0	Park	College Classroom	Fast Food Restaurant	Wings Joint	Dog Run	Home Service	Grocery Store	Gastropub	Gas Station	Furniture / Home Store
17	Waterloo West	0	Park	Wings Joint	Dog Run	Home Service	Grocery Store	Gastropub	Gas Station	Furniture / Home Store	Food Truck	Food & Drink Shop

Cluster 2

In [73]:
```
neighborhoods_to_merged.loc[neighborhoods_to_merged['Cluster Labels'] == 1, neighborhoods_to_merged.columns[[1] + list(range(5, ne
```

Out[73]:

	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
6	Waterloo	1	Pub	Wings Joint	Convenience Store	Home Service	Grocery Store	Gastropub	Gas Station	Furniture / Home Store	Food Truck	Food & Drink Shop

Cluster 3

In [74]:
```
neighborhoods_to_merged.loc[neighborhoods_to_merged['Cluster Labels'] == 2, neighborhoods_to_merged.columns[[1] + list(range(5, ne
```

Out[74]:

	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
1	Waterloo West	2	Playground	Food Truck	Wings Joint	Home Service	Grocery Store	Gastropub	Gas Station	Furniture / Home Store	Food & Drink Shop	Fast Food Restaurant
2	Waterloo East	2	Furniture / Home Store	Food & Drink Shop	Indian Restaurant	Martial Arts Dojo	Pharmacy	Wings Joint	Home Service	Grocery Store	Gastropub	Gas Station
4	Waterloo North	2	Trailer Park	Home Service	Wings Joint	Convenience Store	Grocery Store	Gastropub	Gas Station	Furniture / Home Store	Food Truck	Food & Drink Shop
5	Waterloo East	2	Playground	River	Dog Run	Pool	Home Service	Burger Joint	Baseball Field	Grocery Store	Gastropub	Gas Station
7	Waterloo East	2	Home Service	Bath House	Nature Preserve	Construction & Landscaping	Wings Joint	Dry Cleaner	Grocery Store	Gastropub	Gas Station	Furniture / Home Store
8	Waterloo	2	Pizza Place	Grocery Store	Shop & Service	Dry Cleaner	Wings Joint	Home Service	Gastropub	Gas Station	Furniture / Home Store	Food Truck

10	Waterloo	2	Sushi Restaurant	Gastropub	Nail Salon	Wings Joint	Dog Run	Home Service	Grocery Store	Gas Station	Furniture / Home Store	Food Truck
12	Waterloo	2	Convenience Store	Grocery Store	Mediterranean Restaurant	Sandwich Place	Chinese Restaurant	Coffee Shop	Pet Store	Pizza Place	Dry Cleaner	Gastropub
13	Waterloo	2	Photography Studio	Pool	Wings Joint	Dog Run	Home Service	Grocery Store	Gastropub	Gas Station	Furniture / Home Store	Food Truck
14	Waterloo	2	Restaurant	Wings Joint	Record Shop	Café	Clothing Store	Coffee Shop	Fast Food Restaurant	Warehouse Store	Korean Restaurant	Pizza Place
15	Waterloo West	2	Burger Joint	Wings Joint	Dog Run	Hotel	Home Service	Grocery Store	Gastropub	Gas Station	Furniture / Home Store	Food Truck
16	Waterloo	2	Hotel	Gastropub	Nightclub	Dog Run	Home Service	Grocery Store	Gas Station	Furniture / Home Store	Food Truck	Food & Drink Shop

Discussion

[Top](#toc)

Based on the cluster data for the two cities of interest, the classification for each cluster was done with calculation of venue categories (i.e., the most common category). Analysis of each cluster does not produce conclusive one-to-one results when Foursquare - Most Common Venue data is used. Thus, the following assumption had to be made for each cluster:

- Cluster 1: North York: Mix (i.e., more than one (1) category)
- Cluster 2: North York: Pizza Place
- Cluster 3: North York: Park
- Cluster 1: Waterloo: Park
- Cluster 2: Waterloo: Pub
- Cluster 3: Waterloo: Mix (i.e., more than one (1) category)

Next, a gap in determination of specific districts and establishing correlations between common venues was identified. A systematic, quantitative approach of venues recorded in Foursquare was needed in order to advance onto the next stage. This was further exacerbated by the fact that not all cities have similar common venues despite obvious similarities. Therefore, an additional step of applying a suitable extraction and special integration method was performed.

Conclusion

[Top](#toc)

The work performed in this Capstone was focused on design and application of a new systematic tool that can be used to perform quantitative analysis of data between different cities. Two large Canadian cities in the province of Ontario, specifically Toronto and Waterloo, were selected to demonstrate the proposed approach. In this project, the specific parameters of interest were the availability of Russian language-based services, such as restaurants offering authentic Russian-style cuisine and service. The results indicate that the chosen methodology was successful in identification of neighbourhoods with similar demographic composition and venue categories. It showed high promise for those who seek a systematic, algorithm-based approach for comparison and selection of the desired parameters and can be applied to any of the desired features, e.g., access to public or catholic schools, supermarkets, shopping malls, movie theaters, hospitals, or religious communities.

Future studies can focus on establishing relations withing the acquired data, as well as additional exploration of tools and methods for further optimization of the results.

References

[Top](#toc)

[1] Jeffrey K. Aronson, “Key concepts for making informed choices”, A Nature Research Journal, 12 August 2019

[2] https://www.conferenceboard.ca/e-library/abstract.aspx?did=10342&AspxAutoDetectCookieSupport=1

[3] https://github.com/jnasimi/Coursera_Capstone

[4] https://www.toronto.com/news-story/5588878-russian-in-toronto-10-neighbourhoods-where-you-re-likely-to-hear-it/

In [ ]: