Introduction & Problem Statement

During my recent JOB change i needed to decide between Toronto or New york as my work area. So, i analyzed the population of each city along with "things to do" activities in both the places.

TORONTO

QUEENS

Observation:

```
Queens is a New York City borough on Long Island
 across the East River from Manhattan. It has a p
opulation of about 2.5million people, average age
in 30s which is very much suited me.

On the other hand, Toronto, the capital of the pr
ovince of Ontario, is a major Canadian city along
Lake Ontario’s northwestern shore. It's a dynamic
metropolis with a core of soaring skyscrapers, al
l dwarfed by the iconic, free-standing CN Tower.
 Toronto also has many green spaces, from the ord
erly oval of Queen’s Park to 400-acre High Park a
nd its trails, sports facilities and zoo. It also
has population of about 3 million and average age
group in 40s.

Now, looking at this i wanted to find set of simi
larities and dissimilarities between the two citi
es using the help data already available online w
ithin in the First 5 Mile radius.
```

In [ ]:

Data gathering and Analysis via visualization

Sources of Data and Methods to extract them

The Wikipedia page https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M contains a list of postal codes Toronto region of Canada. we will use web scraping techniques to extract the postal code, borough and neighborhood information via Beautifulsoup and panda packages. then we will get the geographical coordinates for each neighborhood.

For New york (Queens) - we will New_york_datasets available in the cognitive labs, we will get coordinates using Geocode .

Later we will visualize the data on the map and plot using folium and matplotlib python packages.

Importing Libraries

In [6]:

```
import numpy as np # library to handle data in a vectorized manner

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

# import pgeocode
import json # library to handle JSON files
from bs4 import BeautifulSoup
#!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

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
from matplotlib import pyplot as plt

# import k-means from clustering stage
from sklearn.cluster import KMeans
from urllib.request import urlopen
#!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
import seaborn as sns

print('Libraries imported.')
```

```
Libraries imported.
```

1. preparing new york dataframe from sourced json file

In [19]:

```
get_ipython().system("wget -q -O 'newyork_data.json' https://cocl.us/new_york_
dataset")
print('Data downloaded!')

with open('newyork_data.json') as json_data:
    newyork_data = json.load(json_data)

neighborhoods_data = newyork_data['features']
# define the dataframe columns
column_names = ['Borough', 'Neighborhood', 'Latitude', 'Longitude']

# instantiate the dataframe
neighborhoods = pd.DataFrame(columns=column_names)

for data in neighborhoods_data:
    borough = neighborhood_name = data['properties']['borough']
    neighborhood_name = data['properties']['name']

    neighborhood_latlon = data['geometry']['coordinates']
    neighborhood_lat = neighborhood_latlon[1]
    neighborhood_lon = neighborhood_latlon[0]

    neighborhoods = neighborhoods.append({'Borough': borough,
                                          'Neighborhood': neighborhood_name,
                                          'Latitude': neighborhood_lat,
                                          'Longitude': neighborhood_lon}, igno
re_index=True)

neighborhoods.head()
```

```
Data downloaded!
```

Out[19]:

	Borough	Neighborhood	Latitude	Longitude
0	Bronx	Wakefield	40.894705	-73.847201
1	Bronx	Co-op City	40.874294	-73.829939
2	Bronx	Eastchester	40.887556	-73.827806
3	Bronx	Fieldston	40.895437	-73.905643
4	Bronx	Riverdale	40.890834	-73.912585

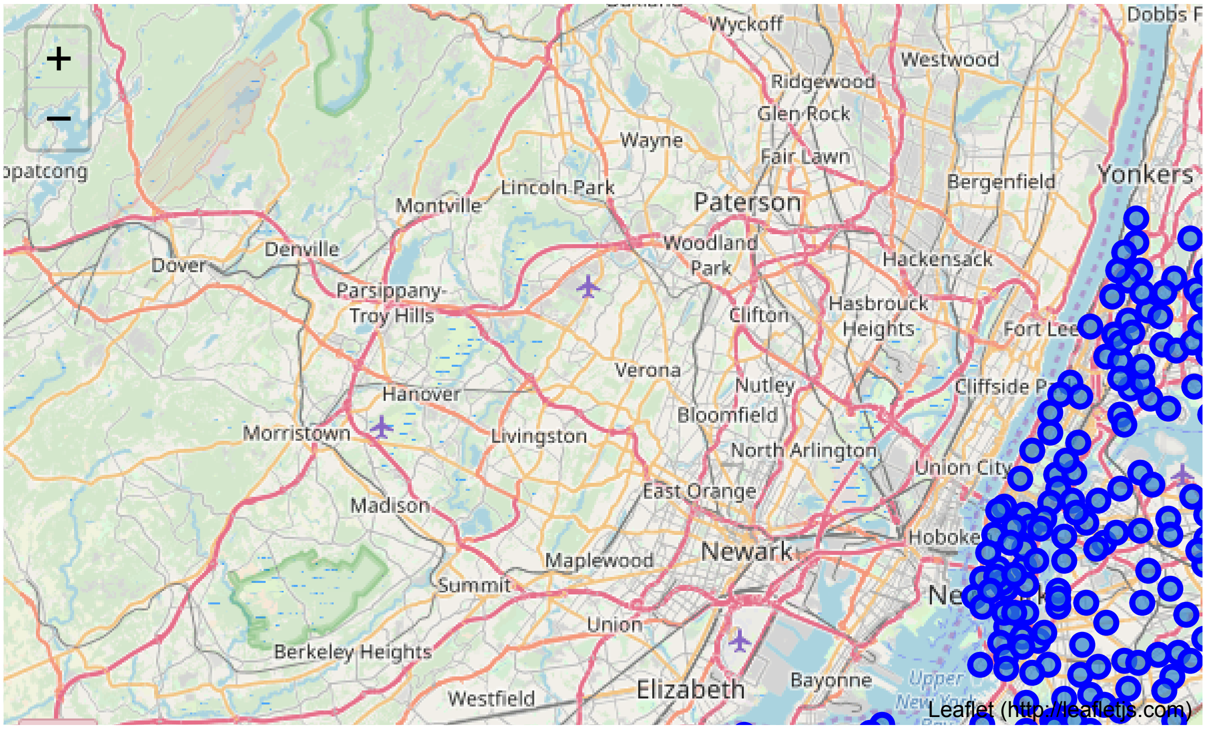
2. Create a NY map highlighting all venues spots

In [20]:

```
address = 'New York City, NY'

geolocator = Nominatim(user_agent="ny_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of New York City are {}, {}.'.format(latitud
e, longitude))

# create map of New York using latitude and longitude values
map_newyork = folium.Map(location=[latitude, longitude], zoom_start=10)

# add markers to map
for lat, lng, borough, neighborhood in zip(neighborhoods['Latitude'], neighbor
hoods['Longitude'], neighborhoods['Borough'], 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_newyork)

map_newyork
```

```
The geograpical coordinate of New York City are 40.7127281, -74.0060152.
```

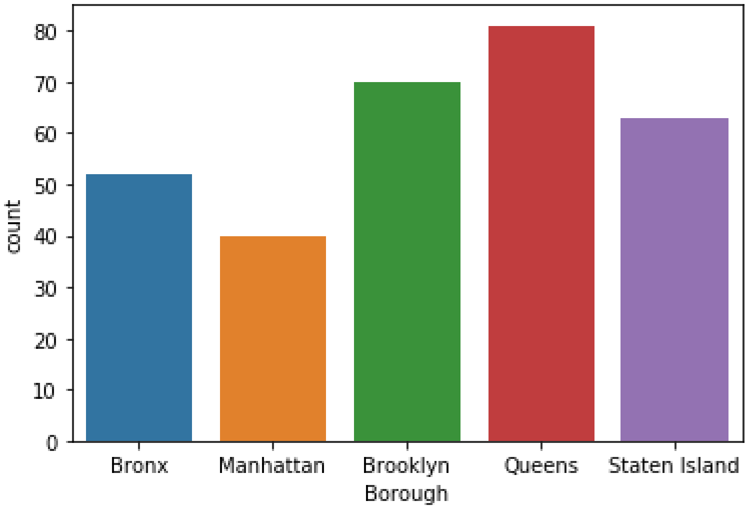
Out[20]:



3. Since Queens has the maximum neighborhoods, i will use Queens for analysis purpose.

```
In [14]: sns.countplot(x="Borough",data=neighborhoods)
         plt.show
```

```
Out[14]: <function matplotlib.pyplot.show(*args, **kw)>
```



Now, Let's import , wrangle , visualize and Analyze data for Toronto .....

4. Fetch "List of postal Codes, Borough and neighborhood" information from the [Link (https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M)](https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M) and clean the data to create a dataframe with coordinates

```
In [8]: # address = 'Queens, NY'
        def findnth(string, substring, n):
            parts = string.split(substring, n + 1)
            if len(parts) <= n + 1:
                return -1
            return len(string) - len(parts[-1]) - len(substring)
```

In [14]:

```
url = "https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M"
html = urlopen(url)
body= BeautifulSoup(html,'lxml')

table= body.find_all("table")
df = pd.read_html(str(table))[0]
# print("Before Removing Not Assigned:",df.shape)

lst=[]
for idx,ele in df.iterrows():
    for x in ele.values:
        if x.find("Not assigned")==-1:
            lst.append(x)
        else:
            continue

lst1=[]
lst2=[]
lst3=[]
print(len(lst))
df=pd.DataFrame()
for idx in range(len(lst)):
    lst1.append(lst[idx][:3])
    pos1=findnth(lst[idx],"(",0)
    pos2=findnth(lst[idx],"(",1)
    if pos1!=-1:
        lst2.append(lst[idx][pos1+1:pos2])
        lst3.append(lst[idx][3:pos1])
    else:
        lst2.append(lst[idx][3:])
        lst3.append("Not assigned")
    lst3


# print(lst3)
df.insert(0,"zip",lst1)
df.insert(1,"borough",lst3)
df.insert(2,"Neighborhood",lst2)
# lst1
df.head(5)

#Fetches Coordinates from csv file
df_data = pd.read_csv("Geospatial_Coordinates.csv")

#Join the two dataframes
df_merged=df.set_index("zip").join(df_data.set_index("Postal Code")).reset_ind
ex()
df_merged.head()
```

```
103
```

Out[14]:

	zip	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	Not assigned	Queen's Park / Ontario Provincial Government	43.662301	-79.389494

In [21]:

```
# !pip install lxml
# geopy bs4
# !pip install bs4
# !pip install geopy
```

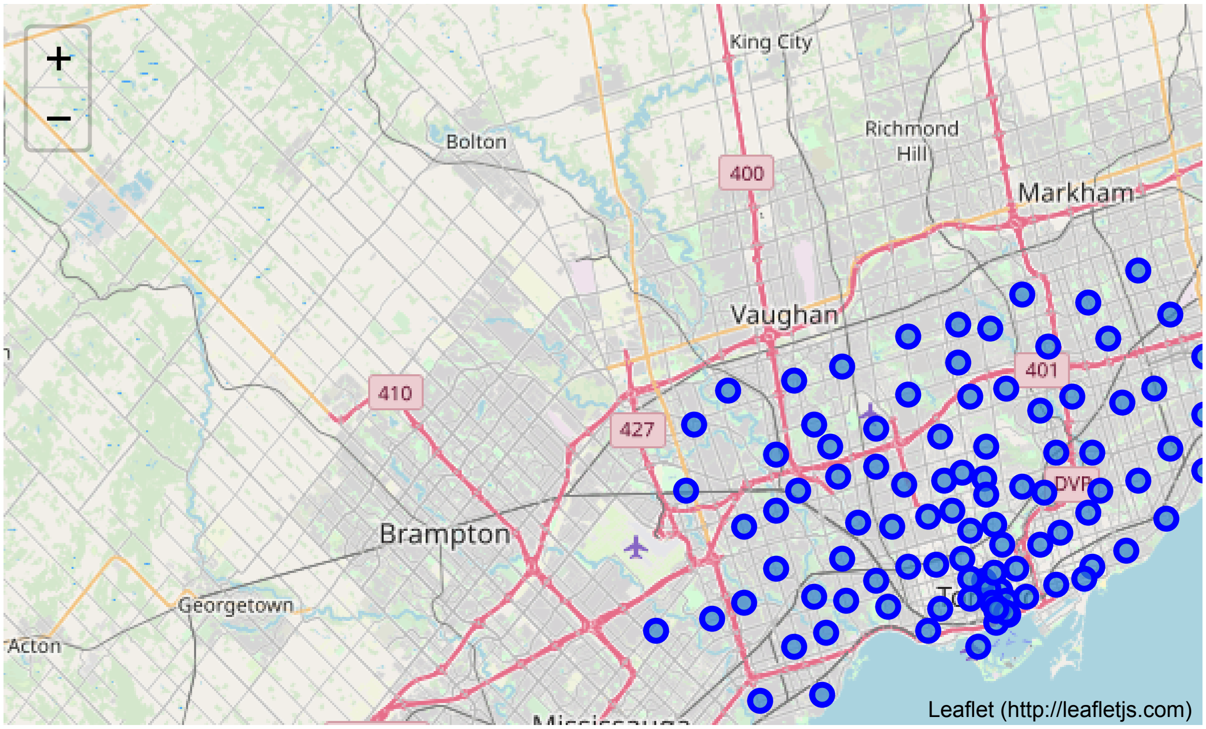
5. Let's visualize the Toronto Neighborhoods on the map....

In [17]:

```
address = 'Toronto Ontario, CA'

geolocator = Nominatim(user_agent="ny_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of Toronto Ontario, CA are {}, {}.'.format(l
atitude, longitude))

# 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_merged['Latitude'], df_merged['L
ongitude'], df_merged['borough'], df_merged['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_toronto)

map_toronto
```

```
The geograpical coordinate of Toronto Ontario, CA are 43.653963, -79.387207.
```

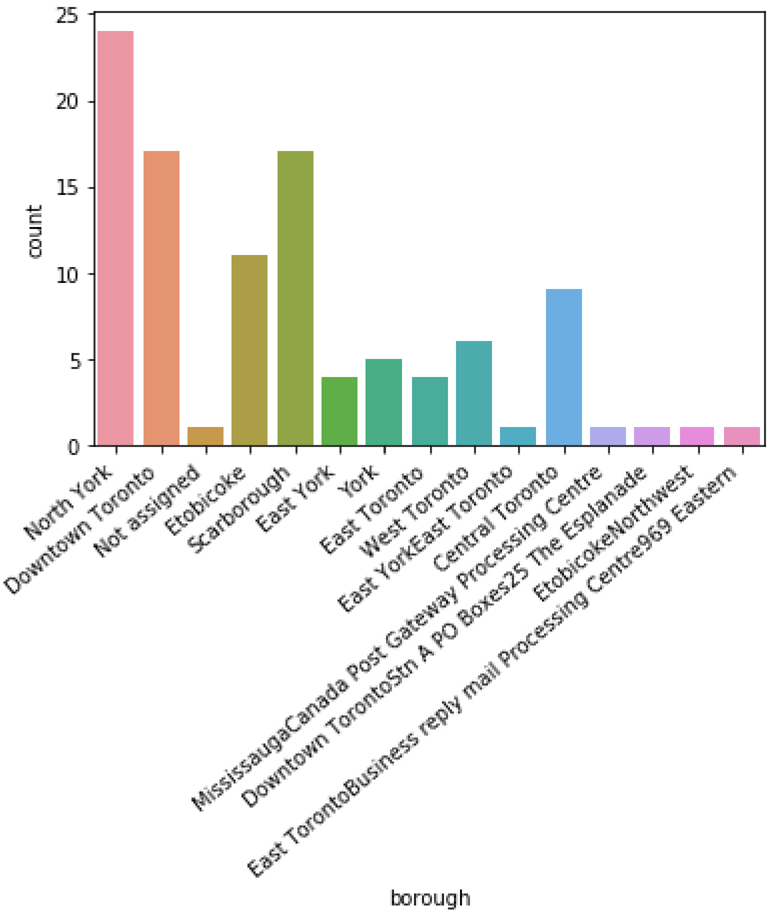
Out[17]:



6. Data Visualization of Borough/County of Toronto Region...

In [15]:
```
ax=sns.countplot(x="borough",data=df_merged)
ax.set_xticklabels(ax.get_xticklabels(), rotation=40, ha="right")
plt.show
```

Out[15]:
```
<function matplotlib.pyplot.show(*args, **kw)>
```



In [ ]:

In [91]:
```
# !pip install folium
# !pip install pgeocode
```

In [3]:
```
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 pgeocode
import json # library to handle JSON files
from bs4 import BeautifulSoup
#!conda install -c conda-forge geopy --yes # uncomment this line if you have
n't completed the Foursquare API lab
from geopy.geocoders import Nominatim # convert an address into latitude and l
ongitude values

import requests # library to handle requests
from pandas.io.json import json_normalize # tranform JSON file into a pandas d
ataframe

# 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
from urllib.request import urlopen
#!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.')
```

```
Libraries imported.
```

preparing new york dataframe from sourced json file

In [4]:

```
!wget -q -O 'newyork_data.json' https://cocl.us/new_york_dataset
print('Data downloaded!')

with open('newyork_data.json') as json_data:
    newyork_data = json.load(json_data)

neighborhoods_data = newyork_data['features']
# define the dataframe columns
column_names = ['Borough', 'Neighborhood', 'Latitude', 'Longitude']

# instantiate the dataframe
neighborhoods = pd.DataFrame(columns=column_names)

for data in neighborhoods_data:
    borough = neighborhood_name = data['properties']['borough']
    neighborhood_name = data['properties']['name']

    neighborhood_latlon = data['geometry']['coordinates']
    neighborhood_lat = neighborhood_latlon[1]
    neighborhood_lon = neighborhood_latlon[0]

    neighborhoods = neighborhoods.append({'Borough': borough,
                                          'Neighborhood': neighborhood_name,
                                          'Latitude': neighborhood_lat,
                                          'Longitude': neighborhood_lon}, igno
re_index=True)

neighborhoods.head()
```

```
Data downloaded!
```

Out[4]:

	Borough	Neighborhood	Latitude	Longitude
0	Bronx	Wakefield	40.894705	-73.847201
1	Bronx	Co-op City	40.874294	-73.829939
2	Bronx	Eastchester	40.887556	-73.827806
3	Bronx	Fieldston	40.895437	-73.905643
4	Bronx	Riverdale	40.890834	-73.912585

Create a NY map highlighting all venues spots

In [5]:

```python
address = 'New York City, NY'

geolocator = Nominatim(user_agent="ny_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of New York City are {}, {}.'.format(latitude, longitude))

# create map of New York using latitude and longitude values
map_newyork = folium.Map(location=[latitude, longitude], zoom_start=10)

# add markers to map
for lat, lng, borough, neighborhood in zip(neighborhoods['Latitude'], neighborhoods['Longitude'], neighborhoods['Borough'], 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_newyork)

# map_newyork
```

```
The geograpical coordinate of New York City are 40.7127281, -74.0060152.
```

Lets create another dataframe only for "Queens" Borough and review its venue

In [6]:

```python
Queens_data = neighborhoods[neighborhoods['Borough'] == 'Queens'].reset_index(drop=True)
Queens_data.head()
```

Out[6]:

	Borough	Neighborhood	Latitude	Longitude
0	Queens	Astoria	40.768509	-73.915654
1	Queens	Woodside	40.746349	-73.901842
2	Queens	Jackson Heights	40.751981	-73.882821
3	Queens	Elmhurst	40.744049	-73.881656
4	Queens	Howard Beach	40.654225	-73.838138

In [7]:

```
CLIENT_ID = 'XBBTYWTYMYK0XHBDXYTUQ34PQ1HUBBE0LPMF3Z05W4PX0XCD' # your Foursquare ID
CLIENT_SECRET = '3NGRPBZAY2BFXJP0Y3OK22UIT4LPD0UTIM4DS5FOWPIWGO1F' # your Foursquare Secret
VERSION = '20180605' # Foursquare API version

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

Explore Neighborhood in Queens...

In [8]:
```
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={}'.format(
            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 [9]:

```
LIMIT = 100 # limit of number of venues returned by Foursquare API
radius = 8000 # define radius of 5 miles


# \\ # create URL
url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secre
t={}&v={}&ll={},{}&radius={}&limit={}'.format(
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    latitude,
    longitude,
    radius,
    LIMIT)
url
```

Out[9]:

```
'https://api.foursquare.com/v2/venues/explore?&client_id=XBBTYWTYMYK0XHBDXYTU
Q34PQ1HUBBE0LPMF3Z05W4PX0XCD&client_secret=3NGRPBZAY2BFXJP0Y3OK22UIT4LPD0UTIM
4DS5FOWPIWGO1F&v=20180605&ll=40.7127281,-74.0060152&radius=8000&limit=100'
```

In [10]:

```
results=requests.get(url).json()
# results
```

In [11]:

```
# type your answer here

Queens_venues = getNearbyVenues(names=Queens_data['Neighborhood'],
                                   latitudes=Queens_data['Latitude'],
                                   longitudes=Queens_data['Longitude']
                                  )



# Queens_venues.head()
```

Out[11]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Astoria	40.768509	-73.915654	Favela Grill	40.767348	-73.917897	Brazilian Restaurant
1	Astoria	40.768509	-73.915654	Orange Blossom	40.769856	-73.917012	Gourmet Shop
2	Astoria	40.768509	-73.915654	Titan Foods Inc.	40.769198	-73.919253	Gourmet Shop
3	Astoria	40.768509	-73.915654	CrossFit Queens	40.769404	-73.918977	Gym
4	Astoria	40.768509	-73.915654	Simply Fit Astoria	40.769114	-73.912403	Gym

In [12]:
```
print(Queens_venues.shape)
Queens_venues.head()
# Queens_venues["Venue Category"].unique()
# Queens_venues.to_csv("C:\Queens_venues.csv")
```

```
(2129, 7)
```

Out[12]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Astoria	40.768509	-73.915654	Favela Grill	40.767348	-73.917897	Brazilian Restaurant
1	Astoria	40.768509	-73.915654	Orange Blossom	40.769856	-73.917012	Gourmet Shop
2	Astoria	40.768509	-73.915654	Titan Foods Inc.	40.769198	-73.919253	Gourmet Shop
3	Astoria	40.768509	-73.915654	CrossFit Queens	40.769404	-73.918977	Gym
4	Astoria	40.768509	-73.915654	Simply Fit Astoria	40.769114	-73.912403	Gym

Analyze Each Neighborhood

In [13]:
```
# # one hot encoding
Queens_onehot = pd.get_dummies(Queens_venues[['Venue Category']], prefix="", p
refix_sep="")

# add neighborhood column back to dataframe
Queens_onehot['Neighbourhood'] = Queens_venues['Neighborhood']

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

Queens_onehot.head()
```

Out[13]:

	Neighbourhood	Accessories Store	Afghan Restaurant	American Restaurant	Arepa Restaurant	Argentinian Restaurant	Art Gallery	A Museu
0	Astoria	0	0	0	0	0	0	
1	Astoria	0	0	0	0	0	0	
2	Astoria	0	0	0	0	0	0	
3	Astoria	0	0	0	0	0	0	
4	Astoria	0	0	0	0	0	0	

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

In [14]:

```
Queens_grouped = Queens_onehot.groupby('Neighbourhood').mean().reset_index()
Queens_grouped.head()
```

Out[14]:

	Neighbourhood	Accessories Store	Afghan Restaurant	American Restaurant	Arepa Restaurant	Argentinian Restaurant	Art Gallery	A Museu
0	Arverne	0.000000	0.0	0.000000	0.0	0.0	0.0	0
1	Astoria	0.000000	0.0	0.010000	0.0	0.0	0.0	0
2	Astoria Heights	0.000000	0.0	0.000000	0.0	0.0	0.0	0
3	Auburndale	0.000000	0.0	0.052632	0.0	0.0	0.0	0
4	Bay Terrace	0.027027	0.0	0.054054	0.0	0.0	0.0	0

FEtch top 10 venues for each Neighborhood and sort them

In [15]:

```
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 [16]:

```
num_top_venues = 10

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

# create columns according to number of top venues
columns = ['Neighbourhood']
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['Neighbourhood'] = Queens_grouped['Neighbourhood']

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

neighborhoods_venues_sorted.head()
```

Out[16]:

	Neighbourhood	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 M Comr Ve
0	Arverne	Surf Spot	Metro Station	Sandwich Place	Playground	Wine Shop	Bed & Breakfast	P P
1	Astoria	Bar	Hookah Bar	Greek Restaurant	Middle Eastern Restaurant	Seafood Restaurant	Mediterranean Restaurant	In Restau
2	Astoria Heights	Burger Joint	Bakery	Plaza	Playground	Hostel	Pizza Place	Sta
3	Auburndale	Italian Restaurant	Pet Store	Fast Food Restaurant	Bar	Furniture / Home Store	Gymnastics Gym	T Ga S
4	Bay Terrace	Clothing Store	Women's Store	Lingerie Store	Mobile Phone Shop	Shoe Store	Kids Store	Do S

4.Cluster Neighborhood

In [17]:

```
k_clusters=5

Queens_top_5_clustering=Queens_grouped.drop("Neighbourhood",axis=1)

kmeans=KMeans(n_clusters=k_clusters,random_state=0).fit(Queens_top_5_clusterin
g)
neighborhoods_venues_sorted.insert(0,"cluster labels",kmeans.labels_)
kmeans.labels_[0:10]
```

Out[17]:

```
array([4, 4, 4, 4, 4, 4, 0, 4, 4, 4], dtype=int32)
```

In [18]:

```
Queens_merged = Queens_data.set_index("Neighborhood").join(neighborhoods_venue
s_sorted.set_index("Neighbourhood"))
Queens_merged.reset_index(inplace=True)
Queens_merged.head()
```

Out[18]:

	Neighborhood	Borough	Latitude	Longitude	cluster labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4 C
0	Astoria	Queens	40.768509	-73.915654	4	Bar	Hookah Bar	Greek Restaurant	Re
1	Woodside	Queens	40.746349	-73.901842	4	Grocery Store	Pizza Place	Filipino Restaurant	A Re
2	Jackson Heights	Queens	40.751981	-73.882821	4	Latin American Restaurant	South American Restaurant	Peruvian Restaurant	
3	Elmhurst	Queens	40.744049	-73.881656	4	Thai Restaurant	Mexican Restaurant	South American Restaurant	Re
4	Howard Beach	Queens	40.654225	-73.838138	4	Pharmacy	Italian Restaurant	Bank	

In [ ]:

Find top 5 neighborhood with maximum number of venues, Sort them and find our their priorities

In [19]:

```
Queens_df_grp=Queens_venues.groupby("Neighborhood").count()
Queens_df_grp=Queens_df_grp["Venue"]
Queens_df_grp=pd.DataFrame(Queens_df_grp)
Queens_df_grp.rename(columns={"Venue":"counts"},inplace=True)
Queens_df_grp.sort_values(by="counts",ascending=False,inplace=True)
Queens_df_grp.head()
top_5_df=Queens_df_grp.reset_index()
top_5_ven=top_5_df["Neighborhood"][:5].tolist()
print(top_5_ven,type(top_5_ven))

# Queens_top_5_venues=Queens_merged[Queens_merged.isin({"Neighborhood":top_5_v
en})["Neighborhood"]].reset_index(drop=True)
Queens_top_5_venues=Queens_merged
```

```
['Sunnyside Gardens', 'Astoria', 'Jackson Heights', 'Woodside', 'Bayside'] <c
lass 'list'>
```

For top 5 Neighborhood, what are the three utmost priorities and least 3 priorities

In [20]:

```
Queens_top_5_venues.columns[:4]
queens_columns=list(Queens_top_5_venues.columns[:8])+list(Queens_top_5_venues.
columns[-3:])
Queens_top_5_venues=Queens_top_5_venues[queens_columns]
Queens_top_5_venues.head()
```

Out[20]:

	Neighborhood	Borough	Latitude	Longitude	cluster labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	8 C
0	Astoria	Queens	40.768509	-73.915654	4	Bar	Hookah Bar	Greek Restaurant	
1	Woodside	Queens	40.746349	-73.901842	4	Grocery Store	Pizza Place	Filipino Restaurant	A Re
2	Jackson Heights	Queens	40.751981	-73.882821	4	Latin American Restaurant	South American Restaurant	Peruvian Restaurant	Pł
3	Elmhurst	Queens	40.744049	-73.881656	4	Thai Restaurant	Mexican Restaurant	South American Restaurant	
4	Howard Beach	Queens	40.654225	-73.838138	4	Pharmacy	Italian Restaurant	Bank	Sa

Now Let's visualize this on the Queens map. First find out the coordinates for Queens using geocode

In [21]:

```
address = 'Queens, NY'

geolocator = Nominatim(user_agent="ny_explorer")
Q_location = geolocator.geocode(address)
Q_latitude = Q_location.latitude
Q_longitude = Q_location.longitude
print('The geograpical coordinate of Queens are {}, {}.'.format(Q_latitude, Q_
longitude))
```

```
The geograpical coordinate of Queens are 40.7498243, -73.7976337.
```

In [22]:

```
# create map
map_clusters = folium.Map(location=[Q_latitude, Q_longitude], zoom_start=11)

# set color scheme for the clusters
rainbow = ['#006eff', '#eb00bc', '#80ff80', '#ff6060', '#ffff00']

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster,top in zip(Queens_top_5_venues['Latitude'], Queens_
top_5_venues['Longitude'], Queens_top_5_venues['Neighborhood'], Queens_top_5_v
enues['cluster labels'],
                                 Queens_top_5_venues["1st Most Common Venue"
]):
    label = folium.Popup("Area= "+str(poi) + ", Top cat="+str(top)+', Cluster
 ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=False,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)

# map_clusters
```

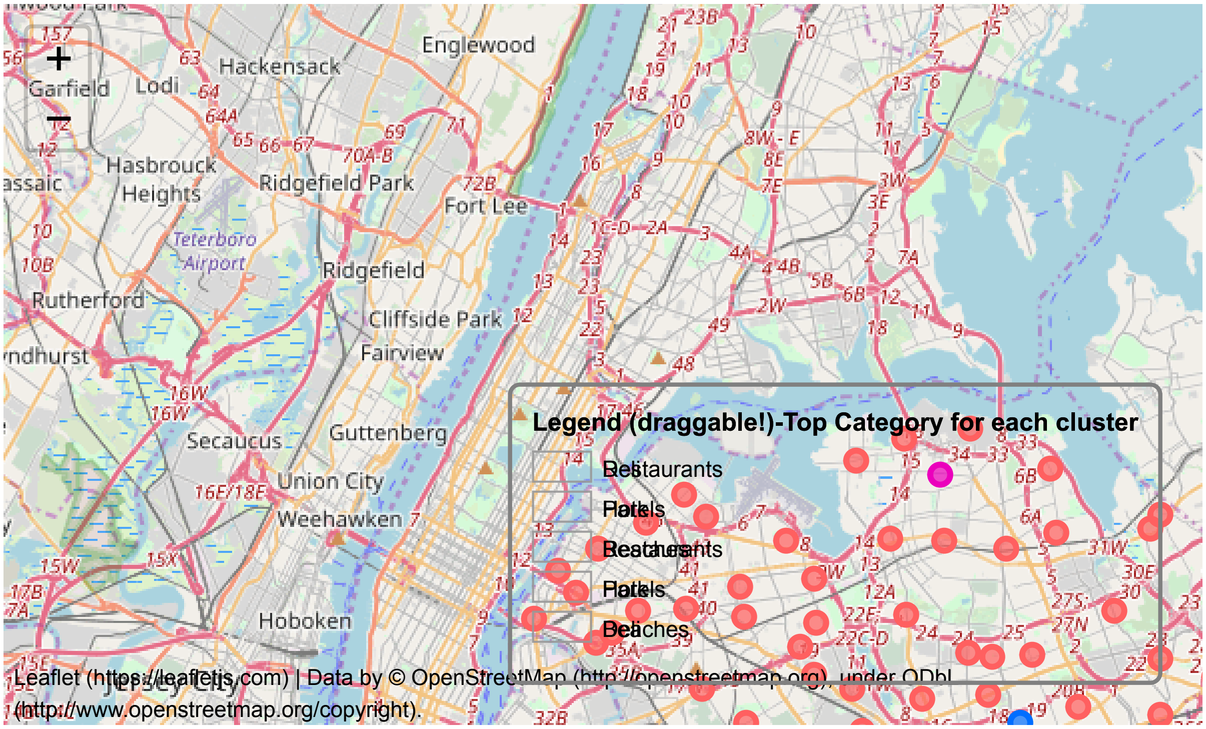
In [ ]:

In [24]:

```
from branca.element import Template, MacroElement

template = """
{% macro html(this, kwargs) %}

<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>jQuery UI Draggable - Default functionality</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.12.1/themes/base/jquery-ui.css">

  <script src="https://code.jquery.com/jquery-1.12.4.js"></script>
  <script src="https://code.jquery.com/ui/1.12.1/jquery-ui.js"></script>

  <script>
  $( function() {
    $( "#maplegend" ).draggable({
                    start: function (event, ui) {
                        $(this).css({
                            right: "auto",
                            top: "auto",
                            bottom: "auto"
                        });
                    }
                });
});

  </script>
</head>
<body>


<div id='maplegend' class='maplegend'
    style='position: absolute; z-index:9999; border:2px solid grey; background-color:rgba(255, 255, 255, 0.8);
     border-radius:6px; padding: 10px; font-size:14px; right: 20px; bottom: 20px;'>

<div class='legend-title'>Legend (draggable!)-Top Category for each cluster</div>
<div class='legend-scale'>
  <ul class='legend-labels'>
    <li><span style='background:Blue;opacity:0.7;'></span>Deli</li>
    <li><span style='background:Yellow;opacity:0.7;'></span>Park</li>
    <li><span style='background:Red;opacity:0.7;'></span>Restaurants</li>
    <li><span style='background:Pink;opacity:0.7;'></span>Hotels</li>
    <li><span style='background:green;opacity:0.7;'></span>Beaches</li>

  </ul>
</div>
</div>

</body>
```

```
</html>

<style type='text/css'>
  .maplegend .legend-title {
    text-align: left;
    margin-bottom: 5px;
    font-weight: bold;
    font-size: 90%;
    }
  .maplegend .legend-scale ul {
    margin: 0;
    margin-bottom: 5px;
    padding: 0;
    float: left;
    list-style: none;
    }
  .maplegend .legend-scale ul li {
    font-size: 80%;
    list-style: none;
    margin-left: 0;
    line-height: 18px;
    margin-bottom: 2px;
    }
  .maplegend ul.legend-labels li span {
    display: block;
    float: left;
    height: 16px;
    width: 30px;
    margin-right: 5px;
    margin-left: 0;
    border: 1px solid #999;
    }
  .maplegend .legend-source {
    font-size: 80%;
    color: #777;
    clear: both;
    }
  .maplegend a {
    color: #777;
    }
</style>
{% endmacro %}"""

macro = MacroElement()
macro._template = Template(template)

map_clusters.get_root().add_child(macro)

map_clusters
```

Out[24]:



It's very clear from the map that topmost category in the Queens Borough is "Restaurants" and then "Hotels"

In [ ]:

```

```

Now let's perform the similar analysis of the data of "Toronto" region and find out the similarities and dissimilarities..

In [25]:
```
import types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

client_d13087807b0347c2a99293bfd1b8c452 = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='Ue7rjZqwsLHj-JYHzK5e2mT4v8VWC2cRn2vVFLolNghX',
    ibm_auth_endpoint="https://iam.ng.bluemix.net/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3-api.us-geo.objectstorage.service.networklayer.co
m')

body = client_d13087807b0347c2a99293bfd1b8c452.get_object(Bucket='datasciencep
rojectlast-donotdelete-pr-fwh3jzvekzw4xg',Key='Geospatial_Coordinates.csv')['B
ody']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__,
body )

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



df_data.head()
```

Out[25]:

	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

In [26]:
```
# address = 'Queens, NY'
def findnth(string, substring, n):
    parts = string.split(substring, n + 1)
    if len(parts) <= n + 1:
        return -1
    return len(string) - len(parts[-1]) - len(substring)
```

In [27]:

```
url = "https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M"
html = urlopen(url)
body= BeautifulSoup(html,'lxml')

table= body.find_all("table")
df = pd.read_html(str(table))[0]
# print("Before Removing Not Assigned:",df.shape)

lst=[]
for idx,ele in df.iterrows():
    for x in ele.values:
        if x.find("Not assigned")==-1:
            lst.append(x)
        else:
            continue

lst1=[]
lst2=[]
lst3=[]
print(len(lst))
df=pd.DataFrame()
for idx in range(len(lst)):
    lst1.append(lst[idx][:3])
    pos1=findnth(lst[idx],"(",0)
    pos2=findnth(lst[idx],"(",1)
    if pos1!=-1:
        lst2.append(lst[idx][pos1+1:pos2])
        lst3.append(lst[idx][3:pos1])
    else:
        lst2.append(lst[idx][3:])
        lst3.append("Not assigned")
    lst3


# print(lst3)
df.insert(0,"zip",lst1)
df.insert(1,"borough",lst2)
df.insert(2,"Neighborhood",lst3)
# lst1
df.head(5)
```

```
103
```

Out[27]:

	zip	borough	Neighborhood
0	M3A	Parkwoods	North York
1	M4A	Victoria Village	North York
2	M5A	Regent Park / Harbourfront	Downtown Toronto
3	M6A	Lawrence Manor / Lawrence Heights	North York
4	M7A	Queen's Park / Ontario Provincial Government	Not assigned

In [28]:

```
df_merged=df.set_index("zip").join(df_data.set_index("Postal Code")).reset_index()
df_merged.head()
```

Out[28]:

	zip	borough	Neighborhood	Latitude	Longitude
0	M3A	Parkwoods	North York	43.753259	-79.329656
1	M4A	Victoria Village	North York	43.725882	-79.315572
2	M5A	Regent Park / Harbourfront	Downtown Toronto	43.654260	-79.360636
3	M6A	Lawrence Manor / Lawrence Heights	North York	43.718518	-79.464763
4	M7A	Queen's Park / Ontario Provincial Government	Not assigned	43.662301	-79.389494

In [29]:

```
LIMIT = 100 # limit of number of venues returned by Foursquare API
radius = 8000 # define radius of 5 miles
latitude=43.6
longitude=-79.3
# \\ # create URL
url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    latitude,
    longitude,
    radius,
    LIMIT)
url
results=requests.get(url).json()
toronto_venues = getNearbyVenues(names=df_merged['Neighborhood'],
                                   latitudes=df_merged['Latitude'],
                                   longitudes=df_merged['Longitude']
                                  )


toronto_venues.head()
```

Out[29]:

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

Analyze Each Neighborhood

In [30]:

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

# add neighborhood column back to dataframe
toronto_onehot['Neighbourhood'] = 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()
toronto_grouped = toronto_onehot.groupby('Neighbourhood').mean().reset_index()
toronto_grouped.head()
```

Out[30]:

	Neighbourhood	Accessories Store	Afghan Restaurant	Airport	Airport Food Court	Airport Gate	Airport Lounge	Airport Service	-
0	Central Toronto	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	(
1	Downtown Toronto	0.0	0.000843	0.000843	0.000843	0.000843	0.001686	0.001686	(
2	Downtown TorontoStn A PO Boxes25 The Esplanade	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	(
3	East Toronto	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	(
4	East TorontoBusiness reply mail Processing Cen...	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	(

FEtch top 10 venues for each Neighborhood and sort them

In [81]:

```
num_top_venues = 10

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

# create columns according to number of top venues
columns = ['Neighbourhood']
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['Neighbourhood'] = toronto_grouped['Neighbourhood'
]

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

neighborhoods_venues_sorted.head()
```

Out[81]:

	Neighbourhood	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
0	Central Toronto	Coffee Shop	Sandwich Place	Park	Café	Sushi Restaurant	Restaurant	Pizza Place
1	Downtown Toronto	Coffee Shop	Café	Restaurant	Hotel	Italian Restaurant	Japanese Restaurant	Bakery
2	Downtown TorontoStn A PO Boxes25 The Esplanade	Coffee Shop	Café	Restaurant	Seafood Restaurant	Cocktail Bar	Hotel	Japanese Restaurant
3	East Toronto	Greek Restaurant	Coffee Shop	Italian Restaurant	Café	Brewery	Ice Cream Shop	Park
4	East TorontoBusiness reply mail Processing Cen...	Yoga Studio	Auto Workshop	Park	Pizza Place	Restaurant	Butcher	Burrito Place

In [82]:

```
toronto_grouped.head()
df_merged.shape
# neighborhoods_venues_sorted.head()
```

Out[82]:

```
(103, 5)
```

4.Cluster Neighborhood

In [85]:

```
k_clusters=5

toronto_top_5_clustering=toronto_grouped.drop("Neighbourhood",axis=1)
kmeans=KMeans(n_clusters=k_clusters,random_state=0).fit(toronto_top_5_clusteri
ng)
print(neighborhoods_venues_sorted.shape)
neighborhoods_venues_sorted.insert(0,"cluster labels",kmeans.labels_)
# print(kmeans.labels_,len(toronto_top_5_clustering))


# In[18]:


toronto_merged = df_merged.set_index("Neighborhood").join(neighborhoods_venues
_sorted.set_index("Neighbourhood"))
toronto_merged.reset_index(inplace=True)
toronto_merged.head()
```

```
(15, 11)
```

Out[85]:

	index	zip	borough	Latitude	Longitude	cluster labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Mo Comm Ven
0	Central Toronto	M4N	Lawrence Park	43.728020	-79.388790	0	Coffee Shop	Sandwich Place	Park	Ca
1	Central Toronto	M5N	Roselawn	43.711695	-79.416936	0	Coffee Shop	Sandwich Place	Park	Ca
2	Central Toronto	M4P	Davisville North	43.712751	-79.390197	0	Coffee Shop	Sandwich Place	Park	Ca
3	Central Toronto	M5P	Forest Hill North & West	43.696948	-79.411307	0	Coffee Shop	Sandwich Place	Park	Ca
4	Central Toronto	M4R	North Toronto West	43.715383	-79.405678	0	Coffee Shop	Sandwich Place	Park	Ca

Find top 5 neighborhood with maximum number of venues, Sort them and find our their priorities

In [86]:
```
toronto_df_grp=toronto_venues.groupby("Neighborhood").count()
toronto_df_grp=toronto_df_grp["Venue"]
toronto_df_grp=pd.DataFrame(toronto_df_grp)
toronto_df_grp.rename(columns={"Venue":"counts"},inplace=True)
toronto_df_grp.sort_values(by="counts",ascending=False,inplace=True)
toronto_df_grp.head()
top_5_df=toronto_df_grp.reset_index()
top_5_ven=top_5_df["Neighborhood"][:5].tolist()
print(top_5_ven,type(top_5_ven))

# toronto_top_5_venues=toronto_merged[toronto_merged.isin({"Neighborhood":top_
5_ven})["Neighborhood"]].reset_index(drop=True)
toronto_top_5_venues=toronto_merged


# <h3> For top 5 Neighborhood, what are the three utmost priorities and least
 3 priorities

# In[20]:


toronto_top_5_venues.columns[:4]
toronto_columns=list(toronto_top_5_venues.columns[:9])+list(toronto_top_5_venu
es.columns[-3:])
toronto_top_5_venues=toronto_top_5_venues[toronto_columns]
toronto_top_5_venues.head()
```

```
['Downtown Toronto', 'North York', 'West Toronto', 'Central Toronto', 'East T
oronto'] <class 'list'>
```

Out[86]:

	index	zip	borough	Latitude	Longitude	cluster labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	8th Mc Comm Ven
0	Central Toronto	M4N	Lawrence Park	43.728020	-79.388790	0	Coffee Shop	Sandwich Place	Park	Dess Sh
1	Central Toronto	M5N	Roselawn	43.711695	-79.416936	0	Coffee Shop	Sandwich Place	Park	Dess Sh
2	Central Toronto	M4P	Davisville North	43.712751	-79.390197	0	Coffee Shop	Sandwich Place	Park	Dess Sh
3	Central Toronto	M5P	Forest Hill North & West	43.696948	-79.411307	0	Coffee Shop	Sandwich Place	Park	Dess Sh
4	Central Toronto	M4R	North Toronto West	43.715383	-79.405678	0	Coffee Shop	Sandwich Place	Park	Dess Sh

Now Let's visualize this on the Toronto map. First find out the coordinates for Toronto using geocode

In [87]:

```
address = 'toronto, CA'

geolocator = Nominatim(user_agent="ny_explorer")
T_location = geolocator.geocode(address)
T_latitude = T_location.latitude
T_longitude = T_location.longitude
print('The geographical coordinate of toronto are {}, {}.'.format(T_latitude, T
_longitude))
```

```
The geograpical coordinate of toronto are 43.653963, -79.387207.
```

In [89]:

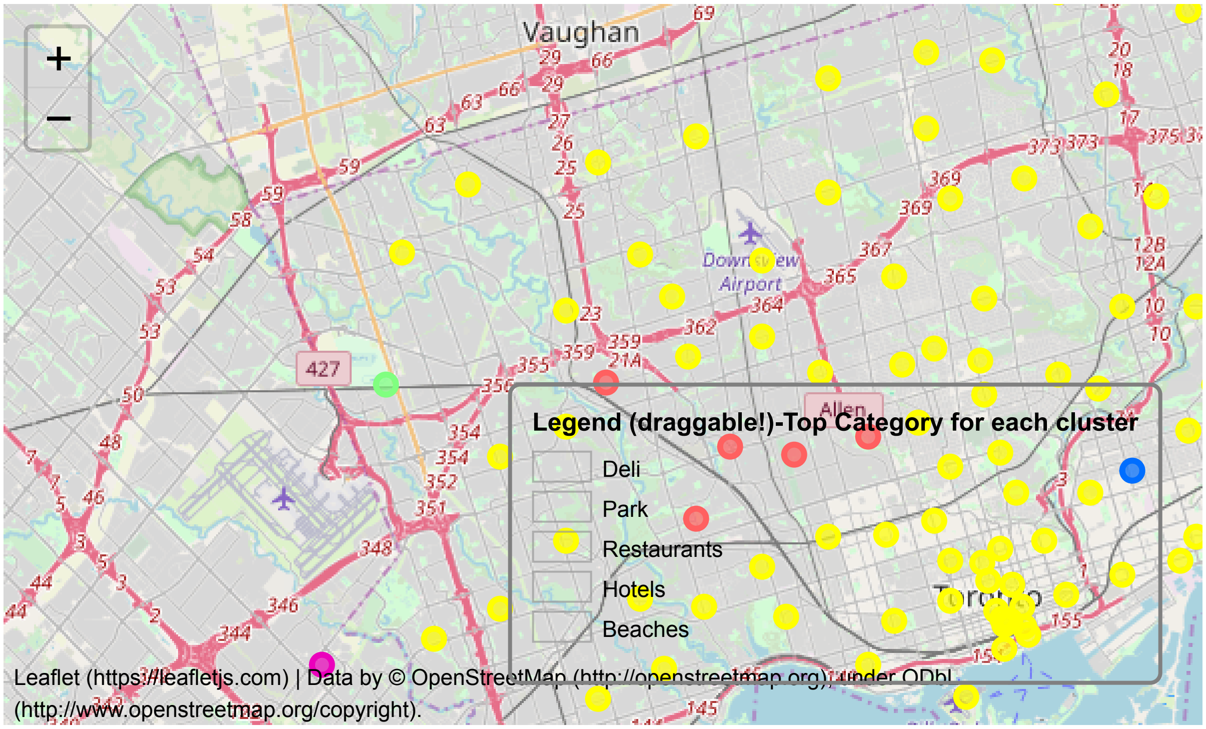
```
# create map
map_clusters_t = folium.Map(location=[T_latitude, T_longitude], zoom_start=11)

# set color scheme for the clusters
rainbow = ['#006eff', '#eb00bc', '#80ff80', '#ff6060', '#ffff00']

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster,top in zip(toronto_top_5_venues['Latitude'], toront
o_top_5_venues['Longitude'], toronto_top_5_venues['borough'], toronto_top_5_ve
nues['cluster labels'],
                                 toronto_top_5_venues["1st Most Common Venue"
]):
    label = folium.Popup("Area= "+str(poi) + ", Top cat="+str(top)+', Cluster
 ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=False,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters_t)

# map_clusters_t
```

In [90]:

```
template = """
{% macro html(this, kwargs) %}

<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>jQuery UI Draggable - Default functionality</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.12.1/themes/base/jquery-
ui.css">

  <script src="https://code.jquery.com/jquery-1.12.4.js"></script>
  <script src="https://code.jquery.com/ui/1.12.1/jquery-ui.js"></script>

  <script>
  $( function() {
    $( "#maplegend" ).draggable({
                    start: function (event, ui) {
                        $(this).css({
                            right: "auto",
                            top: "auto",
                            bottom: "auto"
                        });
                    }
                });
});

  </script>
</head>
<body>


<div id='maplegend' class='maplegend'
    style='position: absolute; z-index:9999; border:2px solid grey; background
-color:rgba(255, 255, 255, 0.8);
     border-radius:6px; padding: 10px; font-size:14px; right: 20px; bottom: 20
px;'>

<div class='legend-title'>Legend (draggable!)-Top Category for each cluster</d
iv>
<div class='legend-scale'>
  <ul class='legend-labels'>
    <li><span style='background:Blue;opacity:0.7;'></span>Deli</li>
    <li><span style='background:Red;opacity:0.7;'></span>Park</li>
    <li><span style='background:Yellow;opacity:0.7;'></span>Restaurants</li>
    <li><span style='background:Pink;opacity:0.7;'></span>Hotels</li>
    <li><span style='background:green;opacity:0.7;'></span>Beaches</li>

  </ul>
</div>
</div>

</body>
</html>
```

```
<style type='text/css'>
  .maplegend .legend-title {
    text-align: left;
    margin-bottom: 5px;
    font-weight: bold;
    font-size: 90%;
    }
  .maplegend .legend-scale ul {
    margin: 0;
    margin-bottom: 5px;
    padding: 0;
    float: left;
    list-style: none;
    }
  .maplegend .legend-scale ul li {
    font-size: 80%;
    list-style: none;
    margin-left: 0;
    line-height: 18px;
    margin-bottom: 2px;
    }
  .maplegend ul.legend-labels li span {
    display: block;
    float: left;
    height: 16px;
    width: 30px;
    margin-right: 5px;
    margin-left: 0;
    border: 1px solid #999;
    }
  .maplegend .legend-source {
    font-size: 80%;
    color: #777;
    clear: both;
    }
  .maplegend a {
    color: #777;
    }
</style>
{% endmacro %}"""

macro = MacroElement()
macro._template = Template(template)

map_clusters_t.get_root().add_child(macro)

map_clusters_t
```

Out[90]:


+
−
Vaughan
Downsview Airport
Allen
427
Toronto
Legend (draggable!)-Top Category for each cluster
Deli
Park
Restaurants
Hotels
Beaches
Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

Conclusion

In terms versatilities of things to do and living a lifestyle with the same age group, Queen has less population than toronto and more number of good restaurants.

Queen has a age group of younger generation which suites my scenario.

In [ ]: