In [5]:

```
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 th
from geopy.geocoders import Nominatim # convert an address into latitude and longitude valu

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

# for webscraping import Beautiful Soup
from bs4 import BeautifulSoup

import xml

!conda install -c conda-forge folium=0.5.0 --yes # uncomment this line if you haven't compl
import folium # map rendering library

print('Libraries imported.')
```

```
usage: conda-script.py [-h] [-V] command ...
conda-script.py: error: unrecognized arguments: # uncomment this line if you
haven't completed the Foursquare API lab

Libraries imported.

usage: conda-script.py [-h] [-V] command ...
conda-script.py: error: unrecognized arguments: # uncomment this line if you
haven't completed the Foursquare API lab
```

In [6]:

```
url = requests.get('https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M').text
soup = BeautifulSoup(url,'lxml')
```

In [7]:

```
table_post = soup.find('table')
fields = table_post.find_all('td')

postcode = []
borough = []
neighbourhood = []

for i in range(0, len(fields), 3):
    postcode.append(fields[i].text.strip())
    borough.append(fields[i+1].text.strip())
    neighbourhood.append(fields[i+2].text.strip())

df_pc = pd.DataFrame(data=[postcode, borough, neighbourhood]).transpose()
df_pc.columns = ['Postcode', 'Borough', 'Neighbourhood']
df_pc.head()
```

Out[7]:

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

In [8]:

```
df_pc.head()
```

Out[8]:

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

In [9]:

```
df_pc['Borough'].replace('Not assigned', np.nan, inplace=True)
df_pc.dropna(subset=['Borough'], inplace=True)

df_pc.head()
```

Out[9]:

	Postcode	Borough	Neighbourhood
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

In [10]:

```
df_pc.head()
```

Out[10]:

	Postcode	Borough	Neighbourhood
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

In [11]:

```
df_pcn = df_pc.groupby(['Postcode', 'Borough'])['Neighbourhood'].apply(', '.join).reset_ind
df_pcn.columns = ['Postcode', 'Borough', 'Neighbourhood']
df_pcn
```

86	M7R	Mississauga	Canada Post Gateway Processing Centre
87	M7Y	East Toronto	Business reply mail Processing Centre
88	M8V	Etobicoke	New Toronto / Mimico South / Humber Bay Shores
89	M8W	Etobicoke	Alderwood / Long Branch
90	M8X	Etobicoke	The Kingsway / Montgomery Road / Old Mill North
91	M8Y	Etobicoke	Old Mill South / King's Mill Park / Sunnylea /...
92	M8Z	Etobicoke	Mimico NW / The Queensway West / South of Bloo...
93	M9A	Etobicoke	Islington Avenue
94	M9B	Etobicoke	West Deane Park / Princess Gardens / Martin Gr...
95	M9C	Etobicoke	Eringate / Bloordale Gardens / Old Burnhamthor...
96	M9L	North York	Humber Summit
97	M9M	North York	Humberlea / Emery
98	M9N	York	Weston

In [12]:

```

df_pcn['Neighbourhood'].replace('Not assigned', "Queen's Park", inplace=True)

df_pcn
```

Out[12]:

	Postcode	Borough	Neighbourhood
0	M1B	Scarborough	Malvern / Rouge
1	M1C	Scarborough	Rouge Hill / Port Union / Highland Creek
2	M1E	Scarborough	Guildwood / Morningside / West Hill
3	M1G	Scarborough	Woburn
4	M1H	Scarborough	Cedarbrae
5	M1J	Scarborough	Scarborough Village
6	M1K	Scarborough	Kennedy Park / Ionview / East Birchmount Park
7	M1L	Scarborough	Golden Mile / Clairlea / Oakridge
8	M1M	Scarborough	Cliffside / Cliffcrest / Scarborough Village West
9	M1N	Scarborough	Birch Cliff / Cliffside West

In [13]:

```
df_pcn.shape
```

Out[13]:

```
(103, 3)
```

In [14]:

```
df_geo = pd.read_csv('http://cocl.us/Geospatial_data')
df_geo.columns = ['Postcode', 'Latitude', 'Longitude']
```

In [15]:

```
df_pos = pd.merge(df_pcn, df_geo, on=['Postcode'], how='inner')

df_tor = df_pos[['Borough', 'Neighbourhood', 'Postcode', 'Latitude', 'Longitude']].copy()

df_tor.head()
```

Out[15]:

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

In [16]:

```
address = 'Toronto, Canada'

geolocator = Nominatim()
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of the City of Toronto are {}, {}.'.format(latitude, long
```

```
C:\Users\GANTAYASHWANTHINI\anaconda3\lib\site-packages\ipykernel_launcher.p
y:3: DeprecationWarning: Using Nominatim with the default "geopy/1.21.0" `us
er_agent` is strongly discouraged, as it violates Nominatim's ToS https://op
erations.osmfoundation.org/policies/nominatim/ (https://operations.osmfounda
tion.org/policies/nominatim/) and may possibly cause 403 and 429 HTTP error
s. Please specify a custom `user_agent` with `Nominatim(user_agent="my-appli
cation")` or by overriding the default `user_agent`: `geopy.geocoders.option
s.default_user_agent = "my-application"`. In geopy 2.0 this will become an e
xception.
  This is separate from the ipykernel package so we can avoid doing imports
 until

The geograpical coordinate of the City of Toronto are 43.6534817, -79.383934
7.
```

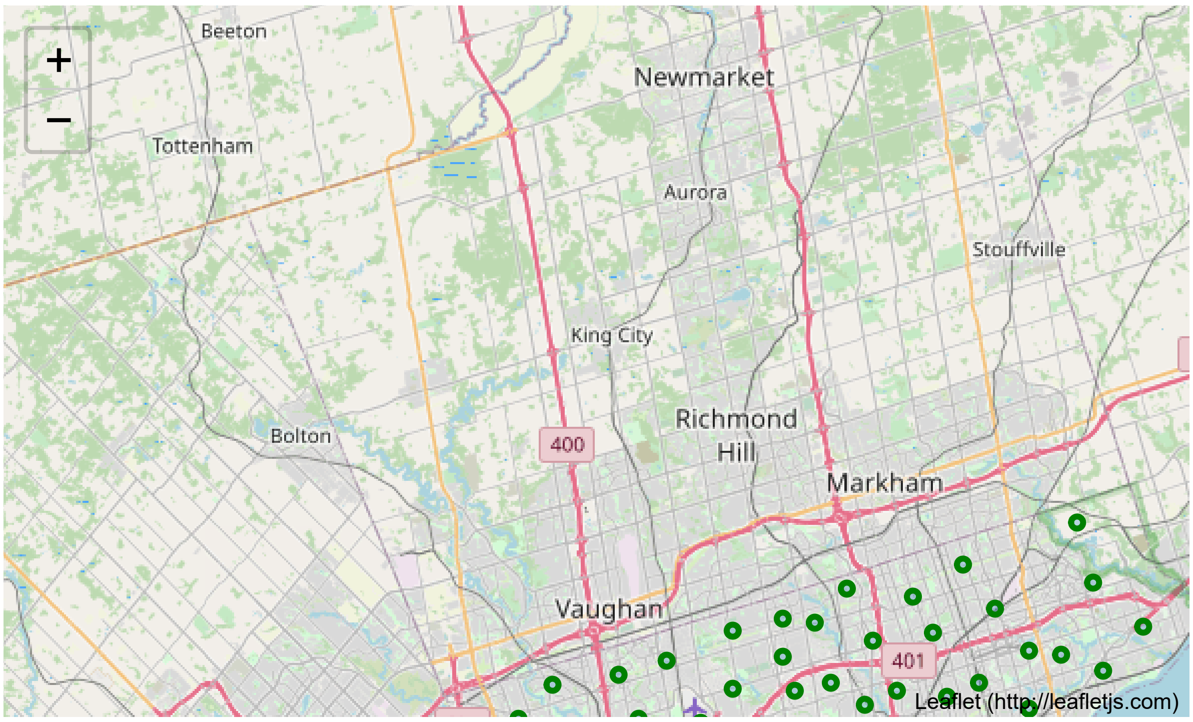
In [17]:

```
# 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_tor['Latitude'], df_tor['Longitude'], df_tor[
    label = '{}, {}'.format(neighborhood, borough)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=3,
        popup=label,
        color='green',
        fill=True,
        fill_color='#3199cc',
        fill_opacity=0.3,
        parse_html=False).add_to(map_toronto)

map_toronto
```

Out[17]:



In [18]:

```
CLIENT_ID = 'not going to share that' # your Foursquare ID
CLIENT_SECRET = 'or this' # 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: not going to share that
CLIENT_SECRET:or this
```

In [19]:

```
df_t4 = df_tor[df_tor['Borough'].str.contains('Toronto')]

to_data = df_t4.reset_index(drop=True)
to_data
```

Out[19]:

	Borough	Neighbourhood	Postcode	Latitude	Longitude
0	East Toronto	The Beaches	M4E	43.676357	-79.293031
1	East Toronto	The Danforth West / Riverdale	M4K	43.679557	-79.352188
2	East Toronto	India Bazaar / The Beaches West	M4L	43.668999	-79.315572
3	East Toronto	Studio District	M4M	43.659526	-79.340923
4	Central Toronto	Lawrence Park	M4N	43.728020	-79.388790
5	Central Toronto	Davisville North	M4P	43.712751	-79.390197
6	Central Toronto	North Toronto West	M4R	43.715383	-79.405678
7	Central Toronto	Davisville	M4S	43.704324	-79.388790
8	Central Toronto	Moore Park / Summerhill East	M4T	43.689574	-79.383160
9	Central Toronto	Summerhill West / Rathnelly / South Hill / For...	M4V	43.686412	-79.400049
10	Downtown Toronto	Rosedale	M4W	43.679563	-79.377529
11	Downtown Toronto	St. James Town / Cabbagetown	M4X	43.667967	-79.367675
12	Downtown Toronto	Church and Wellesley	M4Y	43.665860	-79.383160
13	Downtown Toronto	Regent Park / Harbourfront	M5A	43.654260	-79.360636
14	Downtown Toronto	Garden District / Ryerson	M5B	43.657162	-79.378937
15	Downtown Toronto	St. James Town	M5C	43.651494	-79.375418
16	Downtown Toronto	Berczy Park	M5E	43.644771	-79.373306
17	Downtown Toronto	Central Bay Street	M5G	43.657952	-79.387383
18	Downtown Toronto	Richmond / Adelaide / King	M5H	43.650571	-79.384568
19	Downtown Toronto	Harbourfront East / Union Station / Toronto Is...	M5J	43.640816	-79.381752
20	Downtown Toronto	Toronto Dominion Centre / Design Exchange	M5K	43.647177	-79.381576
21	Downtown Toronto	Commerce Court / Victoria Hotel	M5L	43.648198	-79.379817
22	Central Toronto	Roselawn	M5N	43.711695	-79.416936
23	Central Toronto	Forest Hill North & West	M5P	43.696948	-79.411307
24	Central Toronto	The Annex / North Midtown / Yorkville	M5R	43.672710	-79.405678

	Borough	Neighbourhood	Postcode	Latitude	Longitude
25	Downtown Toronto	University of Toronto / Harbord	M5S	43.662696	-79.400049
26	Downtown Toronto	Kensington Market / Chinatown / Grange Park	M5T	43.653206	-79.400049
27	Downtown Toronto	CN Tower / King and Spadina / Railway Lands / ...	M5V	43.628947	-79.394420
28	Downtown Toronto	Stn A PO Boxes	M5W	43.646435	-79.374846
29	Downtown Toronto	First Canadian Place / Underground city	M5X	43.648429	-79.382280
30	Downtown Toronto	Christie	M6G	43.669542	-79.422564
31	West Toronto	Dufferin / Dovercourt Village	M6H	43.669005	-79.442259
32	West Toronto	Little Portugal / Trinity	M6J	43.647927	-79.419750
33	West Toronto	Brockton / Parkdale Village / Exhibition Place	M6K	43.636847	-79.428191
34	West Toronto	High Park / The Junction South	M6P	43.661608	-79.464763
35	West Toronto	Parkdale / Roncesvalles	M6R	43.648960	-79.456325
36	West Toronto	Runnymede / Swansea	M6S	43.651571	-79.484450
37	Downtown Toronto	Queen's Park / Ontario Provincial Government	M7A	43.662301	-79.389494
38	East Toronto	Business reply mail Processing Centre	M7Y	43.662744	-79.321558

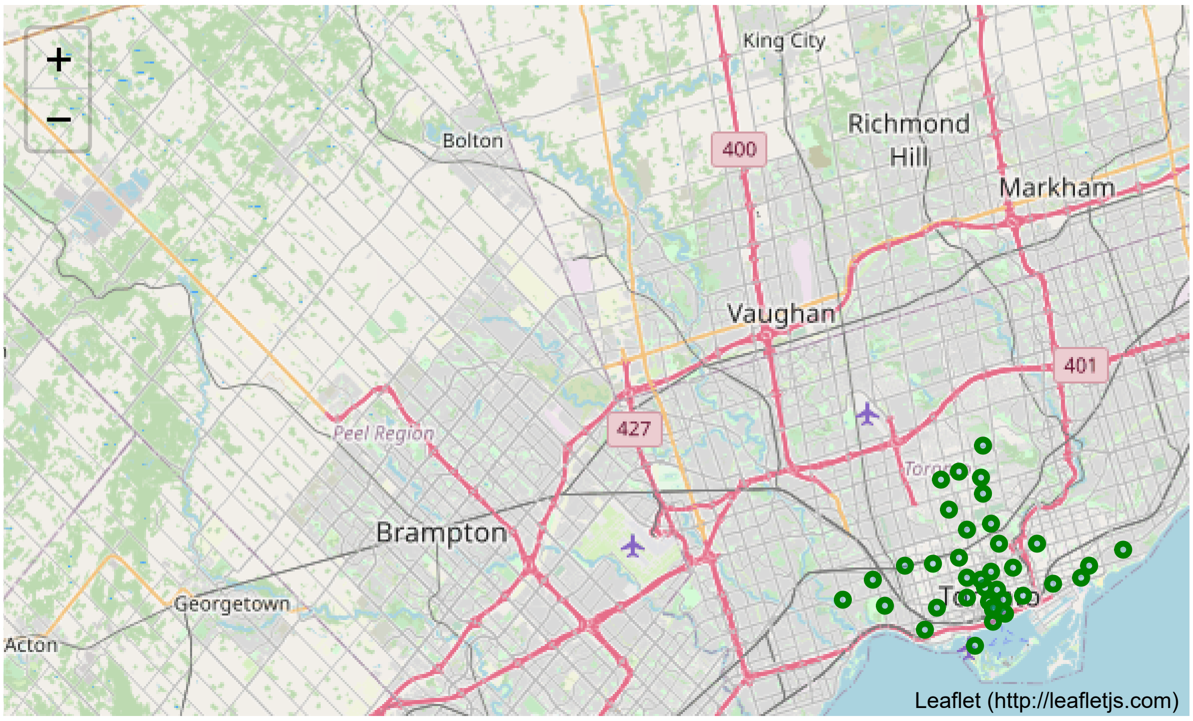
In [20]:

```
map_tohood = folium.Map(location=[latitude, longitude], zoom_start=10)

# add markers to map
for lat, lng, borough, neighborhood in zip(to_data['Latitude'], to_data['Longitude'], to_da
    label = '{}, {}'.format(neighborhood, borough)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=3,
        popup=label,
        color='green',
        fill=True,
        fill_color='#3199cc',
        fill_opacity=0.3,
        parse_html=False).add_to(map_tohood)

map_tohood
```

Out[20]:



In [21]:

```
to_data.loc[0, 'Neighbourhood']
```

Out[21]:

```
'The Beaches'
```

In [22]:

```
neighbourhood_latitude = to_data.loc[0, 'Latitude'] # neighbourhood latitude value
neighbourhood_longitude = to_data.loc[0, 'Longitude'] # neighbourhood longitude value

neighbourhood_name = to_data.loc[0, 'Neighbourhood'] # neighbourhood name

print('Latitude and longitude values of {} are {}, {}.'.format(neighbourhood_name,
                                                               neighbourhood_latitude,
                                                               neighbourhood_longitude))
```

```
Latitude and longitude values of The Beaches are 43.67635739999999, -79.2930
312.
```

In [23]:

```
LIMIT = 100
radius = 500

url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll=
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    neighbourhood_latitude,
    neighbourhood_longitude,
    radius,
    LIMIT)
url
```

Out[23]:

```
'https://api.foursquare.com/v2/venues/explore?&client_id=not going to share
that&client_secret=or this&v=20180605&ll=43.67635739999999,-79.2930312&radiu
s=500&limit=100'
```

In [24]:

```

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

Out[24]:

```
{'meta': {'code': 400,
  'errorType': 'invalid_auth',
  'errorDetail': 'Missing access credentials. See https://developer.foursqua
re.com/docs/api/configuration/authentication (https://developer.foursquare.c
om/docs/api/configuration/authentication) for details.',
  'requestId': '5eb3bcd329ce6a001b07cafe'},
 'response': {}}
```

In [25]:

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

Out[25]:

```
{'meta': {'code': 400,
  'errorType': 'invalid_auth',
  'errorDetail': 'Missing access credentials. See https://developer.foursqua
re.com/docs/api/configuration/authentication (https://developer.foursquare.c
om/docs/api/configuration/authentication) for details.',
  'requestId': '5eb3bf0db57e88001b4be144'},
 'response': {}}
```

In [26]:

```


LIMIT = 100
radius = 500

url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll=
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    neighbourhood_latitude,
    neighbourhood_longitude,
    radius,
    LIMIT)
url
```

Out[26]:

```
'https://api.foursquare.com/v2/venues/explore?&client_id=not going to share
that&client_secret=or this&v=20180605&ll=43.67635739999999,-79.2930312&radiu
s=500&limit=100'
```

In [27]:

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

Out[27]:

```
{'meta': {'code': 400,
  'errorType': 'invalid_auth',
  'errorDetail': 'Missing access credentials. See https://developer.foursqua
re.com/docs/api/configuration/authentication (https://developer.foursquare.c
om/docs/api/configuration/authentication) for details.',
  'requestId': '5eb3be320f5968001b43e1a2'},
 'response': {}}
```

In [31]:

```
# 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']
```

In [32]:

```

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
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()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-32-561c05f0fdd1> in <module>
----> 1 venues = results['response']['groups'][0]['items']
      2 
      3 nearby_venues = json_normalize(venues) # flatten JSON
      4 
      5 # filter columns

KeyError: 'groups'
```

In [33]:

```

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

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-33-8d3c8155afff> in <module>
----> 1 print('{} venues were returned by Foursquare.'.format(nearby_venues.
shape[0]))

NameError: name 'nearby_venues' is not defined
```

In [ ]:

```

```