NEIGHBORING TORONTO CITY

EVERYTHING IS IN ONE NOTEBOOK ITSELF

START OF WEEK 3 ASSIGNMENT

IMPORT LIBRARIES

In [141]:
```python
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

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 DBSCAN from clustering stage
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler

#!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.
```

GET THE DATA OF TORONTO CITY INTO A DATAFRAME

In [142]:
```python
url = 'https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M'
data = pd.read_html(url)[0]
data.head()
```

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

CLEANING UP THE DATA

In [143]:
```python
Toronto_data = data.where(data['Borough'] != 'Not assigned').dropna()
Toronto_data.head()
```

	Postal Code	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 [144]:
```python
Toronto_data.sort_values('Postal Code', ascending = True, inplace = True)
Toronto_data = Toronto_data.reset_index().drop('index', axis = 1)
Toronto_data.head()
```

	Postal Code	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

In [145]:
```python
for index, row in Toronto_data.iterrows():
    if row['Neighbourhood'] == 'Not assigned':
        row['Neighbourhood'] == row['Borough']
Toronto_data.head()
```

	Postal Code	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

In [146]:
```python
Toronto_data.shape
```

```
(103, 3)
```

TRIED FINDING THE LATITUDE AND LONGITUDE OF THE POSTAL CODE (USING GEOPY)

In [147]:
```python
# Create new columns
Toronto_data['Latitude'] = None
Toronto_data['Longitude'] = None
geolocator = Nominatim(user_agent="to_explorer")

# iterate over every row to find its latitude and logitude
for index, row in Toronto_data.iterrows():
    Neighbourhood = row['Neighbourhood'].split(',')[0]
    #print(Neighbourhood)
    address = '{}, Toronto, Ontario'.format(Neighbourhood)
    location = geolocator.geocode(address)

    if location is None:
        continue
    latitude = location.latitude
    longitude = location.longitude

    # Add it to the new columns
    row['Latitude'] = latitude
    row['Longitude'] = longitude
```

TRIED FINDING THE LATITUDE AND LONGITUDE OF THE POSTAL CODE (USING GEOCODER)

In [8]:
```python
# import library
import geocoder

# Create new columns
Toronto_data['Latitude'] = None
Toronto_data['Longitude'] = None

# initiatize variable to None
coordinates = None

# iterate over every row to find its latitude and logitude
for index, row in Toronto_data.iterrows():
    Postal_code = row['Postal code']
    # While loop till we get locations
    while (coordinates == None):
        g = geocoder.google('{}, Toronto, Ontario'.format(Postal_code))
        coordinates = g.latlng

    latitude = coordinates[0]
    longitude = coordinates[1]

    # Add it to the new columns
    row['Latitude'] = latitude
    row['Longitude'] = longitude
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_value(self, series, key)
   4410             try:
-> 4411                 return libindex.get_value_at(s, key)
   4412             except IndexError:

pandas\_libs\index.pyx in pandas._libs.index.get_value_at()

pandas\_libs\index.pyx in pandas._libs.index.get_value_at()

pandas\_libs\util.pxd in pandas._libs.util.get_value_at()

pandas\_libs\util.pxd in pandas._libs.util.validate_indexer()

TypeError: 'str' object cannot be interpreted as an integer

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent call last)
<ipython-input-8-0170bf99a1e9> in <module>
     11 # iterate over every row to find its latitude and logitude
     12 for index, row in Toronto_data.iterrows():
---> 13     Postal_code = row['Postal code']
     14     # While loop till we get locations
     15     while (coordinates == None):

~\anaconda3\lib\site-packages\pandas\core\series.py in __getitem__(self, key)
    869         key = com.apply_if_callable(key, self)
    870         try:
--> 871             result = self.index.get_value(self, key)
    872
    873             if not is_scalar(result):

~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_value(self, series, key)
   4417                     raise InvalidIndexError(key)
   4418                 else:
-> 4419                     raise e1
   4420             except Exception:
   4421                 raise e1

~\anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_value(self, series, key)
   4403         k = self._convert_scalar_indexer(k, kind="getitem")
   4404         try:
-> 4405             return self._engine.get_value(s, k, tz=getattr(series.dtype, "tz", None))
   4406         except KeyError as e1:
   4407             if len(self) > 0 and (self.holds_integer() or self.is_boolean()):

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_value()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_value()

pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'Postal code'
```

USING THE GIVEN COORDINATES FILE

In [148]:
```python
# Read the csv file containing the coordinates
coordinates_df = pd.read_csv('Geospatial_Coordinates.csv')
coordinates_df.head()
```

	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 [149]:
```python
coordinates_df.shape
```

```
(103, 3)
```

In [150]:
```python
# Adding the Latitude and Longitude values
Toronto_data['Latitude'] = coordinates_df['Latitude']
Toronto_data['Longitude'] = coordinates_df['Longitude']
Toronto_data.head()
```

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

END OF WEEK 3 ASSIGNMENT

START OF WEEK 4 ASSIGNMENT

Problem Statement

You want to open a new coffee shop in Toronto, CA. You want to find the optimal venue to set up your cafe to maximize the profit.

1) We select the top 10 neighborhood having maximum number of offices, colleges and schools.
2) Make a union list of all these neighborhood.
3) For all these neighborhood, we find which one of these has the minimum number of coffee shops.
4) Once we find the neighborhood where we want to set up our shop, we search for trending venues in the neighborhood and set up a shop there.
5) We then combine all the offices, colleges and schools data points to cluster them into groups to find outliers 6) Visualize the colleges and schools in the area and select the area where the data points are densed.
7) If not happy with the result, select the second neighborhood with minimum number of coffee shops.

We will be using the Toranto_data that we created earlier to get the neighbouhood data and the Foursquare API for location data

END OF WEEK 4 ASSIGNMENT

START OF WEEK 5 ASSIGNMENT

FIND MAXIMUM NUMBER OF OFFICES USING FOURSQUARE API

In [151]:
```python
# Make extra columns to the dataset
Toronto_data['offices'] = None
Toronto_data['colleges'] = None
Toronto_data['schools'] = None
```

In [152]:
```python
# API parameters
RADIUS = 500
LIMIT = 100
QUERY = 'office'
CLIENT_ID = 'HNAVL03NPRXQBKBJAM423EQ2U02ITQV5OS50DODNPB10OUI5' # your Foursquare ID
CLIENT_SECRET = 'LQW1BBS1HK2KMOVYN3ALEQVWIUWG0ELBSB4VQ5PJJRY2WJWF' # your Foursquare Secret
ACCESS_TOKEN = 'CEYIBG2MFAYGN3HQSUFTU41ZJBBTJS1QH1XU5YI00QEUP4NF' # your FourSquare Access Token
VERSION = '20180604'

# new lists
office_list = []
college_list = []
school_list = []

for index, row in Toronto_data.iterrows():
    LATITUDE = row['Latitude']
    LONGITUDE = row['Longitude']

    # API url
    url = 'https://api.foursquare.com/v2/venues/search?client_id={}&client_secret={}&v={}&ll={},{}&radius={}&oauth_token={}&query={}&limit={}'.format(CLIENT_ID,CLIENT_SECRET,VERSION,LATITUDE,LONGITUDE,RADIUS,ACCESS_TOKEN,QUERY,LIMIT)
    results = requests.get(url).json()
    offices = results['response']['venues']
    temp_dataframe = json_normalize(offices)
    #print(temp_dataframe)
    Number_of_offices = temp_dataframe.shape[0]
    #print(temp_dataframe.shape[0])
    office_list.append(Number_of_offices)
```

```
C:\Users\Tejas\anaconda3\lib\site-packages\ipykernel_launcher.py:23: FutureWarning: pandas.io.json.json_normalize is deprecated, use pandas.json_normalize instead
```

In [153]:
```python
# Add the list to the data frame
Toronto_data['offices'] = office_list
Toronto_data.head()
```

	Postal Code	Borough	Neighbourhood	Latitude	Longitude	offices	colleges	schools
0	M1B	Scarborough	Malvern, Rouge	43.806686	-79.194353	1	None	None
1	M1C	Scarborough	Rouge Hill, Port Union, Highland Creek	43.784535	-79.160497	1	None	None
2	M1E	Scarborough	Guildwood, Morningside, West Hill	43.763573	-79.188711	0	None	None
3	M1G	Scarborough	Woburn	43.770992	-79.216917	0	None	None
4	M1H	Scarborough	Cedarbrae	43.773136	-79.239476	2	None	None

FIND MAXIMUM NUMBER OF COLLEGS USING FOURSQUARE API

In [154]:
```python
QUERY = 'college'

for index, row in Toronto_data.iterrows():
    LATITUDE = row['Latitude']
    LONGITUDE = row['Longitude']

    # API url
    url = 'https://api.foursquare.com/v2/venues/search?client_id={}&client_secret={}&v={}&ll={},{}&radius={}&oauth_token={}&query={}&limit={}'.format(CLIENT_ID,CLIENT_SECRET,VERSION,LATITUDE,LONGITUDE,RADIUS,ACCESS_TOKEN,QUERY,LIMIT)
    results = requests.get(url).json()
    colleges = results['response']['venues']
    temp_dataframe = json_normalize(colleges)
    #print(temp_dataframe)
    Number_of_colleges = temp_dataframe.shape[0]
    #print(temp_dataframe.shape[0])
    college_list.append(Number_of_colleges)
```

```
C:\Users\Tejas\anaconda3\lib\site-packages\ipykernel_launcher.py:11: FutureWarning: pandas.io.json.json_normalize is deprecated, use pandas.json_normalize instead
  # This is added back by InteractiveShellApp.init_path()
```

In [155]:
```python
Toronto_data['colleges'] = college_list
Toronto_data.head()
```

	Postal Code	Borough	Neighbourhood	Latitude	Longitude	offices	colleges	schools
0	M1B	Scarborough	Malvern, Rouge	43.806686	-79.194353	1	0	None
1	M1C	Scarborough	Rouge Hill, Port Union, Highland Creek	43.784535	-79.160497	1	0	None
2	M1E	Scarborough	Guildwood, Morningside, West Hill	43.763573	-79.188711	0	0	None
3	M1G	Scarborough	Woburn	43.770992	-79.216917	0	0	None
4	M1H	Scarborough	Cedarbrae	43.773136	-79.239476	2	0	None

FIND MAXIMUM NUMBER OF SCHOOLS USING FOURSQUARE API

In [156]:
```python
QUERY = 'school'

for index, row in Toronto_data.iterrows():
    LATITUDE = row['Latitude']
    LONGITUDE = row['Longitude']

    # API url
    url = 'https://api.foursquare.com/v2/venues/search?client_id={}&client_secret={}&v={}&ll={},{}&radius={}&oauth_token={}&query={}&limit={}'.format(CLIENT_ID,CLIENT_SECRET,VERSION,LATITUDE,LONGITUDE,RADIUS,ACCESS_TOKEN,QUERY,LIMIT)
    results = requests.get(url).json()
    schools = results['response']['venues']
    temp_dataframe = json_normalize(schools)
    #print(temp_dataframe)
    Number_of_schools = temp_dataframe.shape[0]
    #print(temp_dataframe.shape[0])
    school_list.append(Number_of_schools)
```

```
C:\Users\Tejas\anaconda3\lib\site-packages\ipykernel_launcher.py:11: FutureWarning: pandas.io.json.json_normalize is deprecated, use pandas.json_normalize instead
  # This is added back by InteractiveShellApp.init_path()
```

In [157]:
```python
Toronto_data['schools'] = school_list
Toronto_data.head()
```

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

ARRINGING THE DATASET WITH RESPECT TO MAXIMUM NUMBER OF OFFICES, COLLEGES AND SCHOOLS

In [158]:
```python
Toronto_data['sum'] = Toronto_data['offices'] + Toronto_data['colleges'] + Toronto_data['schools']
Toronto_data.head()
```

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

In [159]:
```python
arranged_Toronto = Toronto_data.sort_values('sum', ascending = False)
arranged_Toronto.head()
```

	Postal Code	Borough	Neighbourhood	Latitude	Longitude	offices	colleges	schools	sum
54	M5B	Downtown Toronto	Garden District, Ryerson	43.657162	-79.378937	50	50	27	127
57	M5G	Downtown Toronto	Central Bay Street	43.657952	-79.387383	50	50	23	123
52	M4Y	Downtown Toronto	Church and Wellesley	43.665860	-79.383160	47	46	10	103
85	M7A	Downtown Toronto	Queen's Park, Ontario Provincial Government	43.662301	-79.389494	28	50	10	88
66	M5S	Downtown Toronto	University of Toronto, Harbord	43.662696	-79.400049	23	50	13	86

WE SELECT THE TOP 3 RESULTS FOR FURTHER ANALYSIS

In [160]:
```python
trimmed_Toronto = arranged_Toronto.head(3)
trimmed_Toronto.head()
```

	Postal Code	Borough	Neighbourhood	Latitude	Longitude	offices	colleges	schools	sum
54	M5B	Downtown Toronto	Garden District, Ryerson	43.657162	-79.378937	50	50	27	127
57	M5G	Downtown Toronto	Central Bay Street	43.657952	-79.387383	50	50	23	123
52	M4Y	Downtown Toronto	Church and Wellesley	43.665860	-79.383160	47	46	10	103

FOR THESE LOCATIONS WE WILL CHECK WHICH LOCATION HAS THE LEAST NUMBER OF COFFEE SHOPS

In [161]:
```python
QUERY = 'coffee'
coffee_list = []

for index, row in trimmed_Toronto.iterrows():
    LATITUDE = row['Latitude']
    LONGITUDE = row['Longitude']

    # API url
    url = 'https://api.foursquare.com/v2/venues/search?client_id={}&client_secret={}&v={}&ll={},{}&radius={}&oauth_token={}&query={}&limit={}'.format(CLIENT_ID,CLIENT_SECRET,VERSION,LATITUDE,LONGITUDE,RADIUS,ACCESS_TOKEN,QUERY,LIMIT)
    results = requests.get(url).json()
    coffee = results['response']['venues']
    temp_dataframe = json_normalize(coffee)
    #print(temp_dataframe)
    Number_of_coffee = temp_dataframe.shape[0]
    print(temp_dataframe.shape[0])
    coffee_list.append(Number_of_coffee)
```

```
C:\Users\Tejas\anaconda3\lib\site-packages\ipykernel_launcher.py:12: FutureWarning: pandas.io.json.json_normalize is deprecated, use pandas.json_normalize instead
  if sys.path[0] == '':

30
32
11
```

AS WE CAN SEE, THE THIRD LOCATION HAS COMPARATIVELY LESS COFFEE SHOPS AND TOTAL 103 OFFICEC, COLLEGES AND SCHOOLS.

LOOK FOR TRENDING PLACES IN THE NEIGHBORHOOD

In [162]:
```python
# define URL
url = 'https://api.foursquare.com/v2/venues/trending?client_id={}&client_secret={}&ll={},{}&v={}'.format(CLIENT_ID, CLIENT_SECRET, LATITUDE, LONGITUDE, VERSION)

# send GET request and get trending venues
results = requests.get(url).json()
results
```

```
{'meta': {'code': 200, 'requestId': '5fd114c794a8241c14989030'},
 'response': {'venues': []}}
```

In [163]:
```python
if len(results['response']['venues']) == 0:
    trending_venues_df = None

else:
    trending_venues = results['response']['venues']
    trending_venues_df = json_normalize(trending_venues)

    # Filter columns
    columns_filtered = ['name', 'categories'] + ['location.distance', 'location.city', 'location.postalCode', 'location.state', 'location.country', 'location.lat', 'location.lng']
    trending_venues_df = trending_venues_df.loc[:, columns_filtered]
```

In [164]:
```python
# Display trending values
trending_venues_df
```

GET COORDINATES OF OFFICES IN THE NEIGHBORHOOD

In [165]:
```python
# coordinates of the neighborhood
LATITUDE = trimmed_Toronto.at[52, 'Latitude']
LONGITUDE = trimmed_Toronto.at[52, 'Longitude']

# query to search for
QUERY = 'office'

# new lists
target_office_list = []
target_college_list = []
target_school_list = []

#for index, row in Toronto_data.iterrows():
#    LATITUDE = row['Latitude']
#    LONGITUDE = row['Longitude']

# API url
url = 'https://api.foursquare.com/v2/venues/search?client_id={}&client_secret={}&v={}&ll={},{}&radius={}&oauth_token={}&query={}&limit={}'.format(CLIENT_ID,CLIENT_SECRET,VERSION,LATITUDE,LONGITUDE,RADIUS,ACCESS_TOKEN,QUERY,LIMIT)
results = requests.get(url).json()
offices = results['response']['venues']
temp_dataframe = json_normalize(offices)
temp_dataframe = temp_dataframe[['name', 'location.lat', 'location.lng']]
office_data = temp_dataframe
print(office_data.shape[0])
office_data.head()
```

```
47

C:\Users\Tejas\anaconda3\lib\site-packages\ipykernel_launcher.py:21: FutureWarning: pandas.io.json.json_normalize is deprecated, use pandas.json_normalize instead
```

	name	location.lat	location.lng
0	Angela's Office, 11th floor OMP	43.670198	-79.379034
1	Martina's Office - BBDO	43.670275	-79.387297
2	CUPE 2191 Office	43.662360	-79.381737
3	Le Office	43.669748	-79.380067
4	Ken Chan Campaign Office	43.663729	-79.383380

In [166]:
```python
# make a list of lists of offices
for index, row in office_data.iterrows():
    lat = row['location.lat']
    #print(lat)
    lng = row['location.lng']
    #print(lng)
    #temp_target_office_list.append([lat, lng])
    target_office_list.append([lat, lng])
len(target_office_list)
```

```
47
```

GET COORDINATES OF COLLEGES IN THE NEIGHBORHOOD

In [167]:
```python
# query to search for
QUERY = 'college'

# API url
url = 'https://api.foursquare.com/v2/venues/search?client_id={}&client_secret={}&v={}&ll={},{}&radius={}&oauth_token={}&query={}&limit={}'.format(CLIENT_ID,CLIENT_SECRET,VERSION,LATITUDE,LONGITUDE,RADIUS,ACCESS_TOKEN,QUERY,LIMIT)
results = requests.get(url).json()
colleges = results['response']['venues']
temp_dataframe = json_normalize(colleges)
temp_dataframe = temp_dataframe[['name', 'location.lat', 'location.lng']]
college_data = temp_dataframe
print(college_data.shape[0])
college_data.head()
```

```
46

C:\Users\Tejas\anaconda3\lib\site-packages\ipykernel_launcher.py:8: FutureWarning: pandas.io.json.json_normalize is deprecated, use pandas.json_normalize instead
```

	name	location.lat	location.lng
0	College Park	43.661237	-79.383603
1	College Subway Station	43.661311	-79.382819
2	Yonge & College	43.661371	-79.383184
3	TTC Streetcar #506 College	43.664858	-79.377675
4	College Cleaners	43.662731	-79.381139

In [168]:
```python
# make a list of lists of colleges
for index, row in college_data.iterrows():
    lat = row['location.lat']
    lng = row['location.lng']
    target_college_list.append([lat, lng])
len(target_college_list)
```

```
46
```

GET COORDINATES OF SCHOOLS IN THE NEIGHBORHOOD

In [169]:
```python
# query to search for
QUERY = 'school'

# API url
url = 'https://api.foursquare.com/v2/venues/search?client_id={}&client_secret={}&v={}&ll={},{}&radius={}&oauth_token={}&query={}&limit={}'.format(CLIENT_ID,CLIENT_SECRET,VERSION,LATITUDE,LONGITUDE,RADIUS,ACCESS_TOKEN,QUERY,LIMIT)
results = requests.get(url).json()
schools = results['response']['venues']
temp_dataframe = json_normalize(schools)
temp_dataframe = temp_dataframe[['name', 'location.lat', 'location.lng']]
school_data = temp_dataframe
print(school_data.shape[0])
school_data.head()
```

```
10

C:\Users\Tejas\anaconda3\lib\site-packages\ipykernel_launcher.py:8: FutureWarning: pandas.io.json.json_normalize is deprecated, use pandas.json_normalize instead
```

	name	location.lat	location.lng
0	Church St junior Public School	43.663915	-79.379760
1	St Josephs College Secondary School	43.664259	-79.388681
2	Church Street Junior Public School	43.663483	-79.379778
3	Sol School	43.664743	-79.377530
4	Church Street Public School	43.663703	-79.380324

In [170]:
```python
# make a list of lists of schools
for index, row in school_data.iterrows():
    lat = row['location.lat']
    lng = row['location.lng']
    target_school_list.append([lat, lng])
len(target_school_list)
```

```
10
```

COMBINING THE OFFICE, COLLEGE AND SCHOOL DATAFRAME

In [171]:
```python
temp1 = office_data.append(college_data)
combine_data = temp1.append(school_data)

# Make new column for cluster
combine_data['cluster'] = None
```

In [172]:
```python
#look at combine data
print(combine_data.shape[0])
combine_data.head()
```

```
103
```

	name	location.lat	location.lng	cluster
0	Angela's Office, 11th floor OMP	43.670198	-79.379034	None
1	Martina's Office - BBDO	43.670275	-79.387297	None
2	CUPE 2191 Office	43.662360	-79.381737	None
3	Le Office	43.669748	-79.380067	None
4	Ken Chan Campaign Office	43.663729	-79.383380	None

COMBINE THE LIST OF COORDINATES TO PREPARE FOR CLUSTERING

In [173]:
```python
# Adding college list of office list
for value in target_college_list:
    target_office_list.append(value)

# Adding school list to office list
for value in target_school_list:
    target_office_list.append(value)

#target_office_list
```

In [174]:
```python
# Confirm the size
print(len(target_office_list))
print(combine_data.shape[0])
```

```
103
103
```

In [175]:
```python
# Change the name
target_list = target_office_list
```

CLUSTERING USING DBSCAN

```python
# Radius
epsilon = 0.006

minimumSamples = 91
db = DBSCAN(eps = epsilon, min_samples = minimumSamples).fit(target_list)
labels = db.labels_
labels
```

```
array([-1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0, -1,  0, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0, -1,
        0,  0,  0, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0, -1,  0,  0, -1, -1,  0,  0,  0,  0,  0,
        0,  0,  0,  0, -1,  0,  0,  0,  0,  0,  0, -1,  0,  0,  0,  0,  0,
       -1], dtype=int64)
```

```python
combine_data['cluster'] = labels
combine_data.head()       # -1 are outliers and 0 are densed areas
```

	name	location.lat	location.lng	cluster
0	Angela's Office, 11th floor OMP	43.670198	-79.379034	-1
1	Martina's Office - BBDO	43.670275	-79.387297	0
2	CUPE 2191 Office	43.662360	-79.381737	0
3	Le Office	43.669748	-79.380067	0
4	Ken Chan Campaign Office	43.663729	-79.383380	0

WE ANALYSE ALL THIS INFORMATION ON A MAP

```python
# Create a map of Church and Wellesley, Downtown Toronto
map_toronto = folium.Map(location = [LATITUDE, LONGITUDE], zoom_start = 15)

# Extract cluster and outlier data bases
cluster_df = combine_data[combine_data['cluster'] == 0]
outlier_df = combine_data[combine_data['cluster'] == -1]

# Add labels and cluster markers to the map
for lat, lng, label in zip(cluster_df['location.lat'], cluster_df['location.lng'], cluster_
df['name']):
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius = 3,
        popup = label,
        color = 'blue',
        fill = True,
        fill_color = '#3186cc',
        fill_opacity = 0.7,
        parse_html = False).add_to(map_toronto)

# Add labels and outlier markers to the map
for lat, lng, label in zip(outlier_df['location.lat'], outlier_df['location.lng'], outlier_
df['name']):
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius = 3,
        popup = label,
        color = 'red',
        fill = True,
        fill_color = '#ff0000',
        fill_opacity = 0.7,
        parse_html = False).add_to(map_toronto)
map_toronto

# Similarly adding trending places to the map if any. None in this case
```


EXCLUDING THE OUTLIERS (RED), WE CAN SEE THAT THE IDEAL PLACE TO OPEN A COFFEE SHOP WOULD BE EITHER THE INTERSECTION OF BLOOR STREET WEST AND YONGE STREE OR THE INTERSECTION OF COLLEGE STREET AN YONGE STREET.