Part 1

In [246]:

```python
import pandas as pd
import requests
import re
import numpy as np
from pandas.io.json import json_normalize
import folium # map rendering library
from math import sin, cos, sqrt, atan2, radians

# Scrape the Wikipedia page with the list of Postal codes within the city of T
# The table associates postcode, borough and neighbourhood
page_url = "https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M"
response = requests.get(page_url)
content = response.content

# Save the result as a string
```

In [247]:

```python
# Search the values of the table from the content of the page
# and save them in "locations"
lines = re.findall('<tr>(.*?)</tr>', content)
locations = []
for i in range(1,len(lines) - 4):
    line = re.findall('<td>(.*?)</td>', lines[i])
    location = []
    for value in range(3):
        if '<' in line[value]:
            location.append(re.findall('>(.*?)<',line[value])[0])
        else:
            location.append(line[value])
    locations.append(location)

# Transform locations into a DataFrame
locations = pd.DataFrame(locations)
```

In [248]:

Out[248]:

	PostalCode	Borough	Neighborhood
0	M1A	Not assigned	Not assigned\n
1	M2A	Not assigned	Not assigned\n
2	M3A	North York	Parkwoods
3	M4A	North York	Victoria Village
4	M5A	Downtown Toronto	Harbourfront

In [249]:

```python
# Delete "\n", "\" and "Not assigned" from the values in the DataFrame
locations = locations.replace(r'\\n?','', regex=True)
locations = locations.replace(r'Not assigned','')

```

Out[249]:

	PostalCode	Borough	Neighborhood
0	M1A		
1	M2A		
2	M3A	North York	Parkwoods
3	M4A	North York	Victoria Village
4	M5A	Downtown Toronto	Harbourfront

In [250]:

```
# Delete the rows where there is no borough
locations = locations[locations["Borough"] != ""]
```

Out[250]:

	PostalCode	Borough	Neighborhood
2	M3A	North York	Parkwoods
3	M4A	North York	Victoria Village
4	M5A	Downtown Toronto	Harbourfront
5	M6A	North York	Lawrence Heights
6	M6A	North York	Lawrence Manor

In [251]:

```
# If a cell has a borough but an empty neighborhood, then
# the neighborhood will be replaced by the borough.
locations.loc[locations["Neighborhood"] =="", "Neighborhood"] = locations.loc[
```

Part 2

In [252]:

```
# Read the csv-file with the latitude and longitude from the postal codes.
geospatial_coord = pd.read_csv("Geospatial_Coordinates.csv")
```

Out[252]:

	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 [253]:

Out[253]:

	PostalCode	Borough	Neighborhood
2	M3A	North York	Parkwoods
3	M4A	North York	Victoria Village
4	M5A	Downtown Toronto	Harbourfront
5	M6A	North York	Lawrence Heights
6	M6A	North York	Lawrence Manor

In [254]:

```
# Merge locations and geospatial_coord on "PostalCode" and "Postal Code"
# to get a DataFrame with them both
```

In [255]:

```
locations.head()
```

Out[255]:

	PostalCode	Borough	Neighborhood	Postal Code	Latitude	Longitude
0	M3A	North York	Parkwoods	M3A	43.753259	-79.329656
1	M4A	North York	Victoria Village	M4A	43.725882	-79.315572
2	M5A	Downtown Toronto	Harbourfront	M5A	43.654260	-79.360636
3	M6A	North York	Lawrence Heights	M6A	43.718518	-79.464763
4	M6A	North York	Lawrence Manor	M6A	43.718518	-79.464763

In [256]:

```python
# Postal code is twice in the table
# delete one
locations = locations.drop("Postal Code", axis = 1)
```

Out[256]:

	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	Harbourfront	43.654260	-79.360636
3	M6A	North York	Lawrence Heights	43.718518	-79.464763
4	M6A	North York	Lawrence Manor	43.718518	-79.464763

In [257]:

Out[257]:

```
(210, 5)
```

Part 3

In [258]:

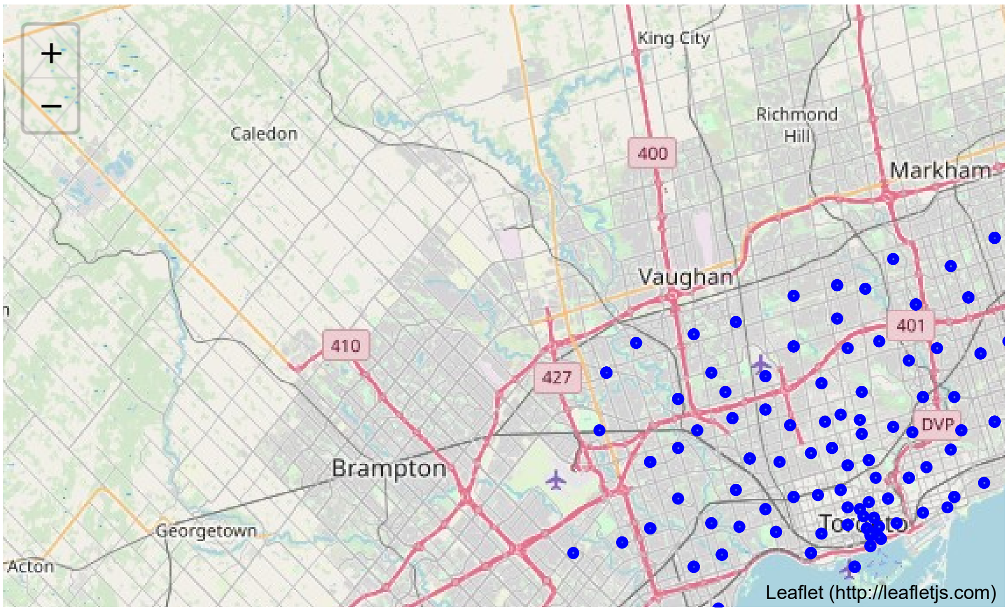
```python
from geopy.geocoders import Nominatim # convert an address into latitude and l
# Search the latitude and longitude of Toronto
# Sometimes the code doesn't work due to time out of geolocator
# In case it doesn't work, the location is also hard coded.
address = 'Toronto'
try:
    geolocator = Nominatim(user_agent="tr_explorer")
    loc = geolocator.geocode(address)
    latitude = loc.latitude
    longitude = loc.longitude
except:
    latitude = 43.653963
    longitude = -79.387207
```

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

In [259]:

```
import folium # map rendering library

# 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(locations['Latitude'], locations['L
    label = '{}, {}'.format(neighborhood, borough)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=2,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(map_toronto)

map_toronto
```

Out[259]:



Quantity of venues by location

In [260]:

```
# 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:
```

In [261]:

```
# Calcule the distance between two locations on the earth
# using latitude and longitude
# Return the distance in km
def distance_earth(lat1,lon1,lat2,lon2):
    R = 6373.0
    lat1 = radians(lat1)
    lon1 = radians(lon1)
    lat2 = radians(lat2)
    lon2 = radians(lon2)
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
    c = 2 * atan2(sqrt(a), sqrt(1 - a))
    distance = R * c
    return distance

```

In [262]:

```
# Search the needed difference between two latitudes by the same longitude (ho
# or two longitudes by the same latitude (how = "lat")
# to get a distance of 1 km
def search_opt(start, end, steps, lat1, lon1, how = "lon"):
    steps_len = (end - start) / steps
    i_opt = start
    dist_opt = distance_earth(lat1,lon1,lat1,lon1)

    for i in np.arange(start, end + steps_len, steps_len):
        if how == "lon":
            dist = distance_earth(lat1,lon1,lat1,lon1 + i)
        else:
            how = "lat"
            dist = distance_earth(lat1,lon1,lat1 + i,lon1)
        if abs(1-dist) < abs(1-dist_opt):
            i_opt = i
            dist_opt = dist
    return i_opt, dist_opt, how
```

In [263]:

```
# Sart location
loc_toronto = [43.653963, -79.387207]
loc_toronto = [43.653963+0.08, -79.387207]
[lat, lng] = loc_toronto
```

Out[263]:

```
[43.733962999999996, -79.387207]
```

In [264]:

```
# Create a latitude and longitude grid, starting at the location of Toronto
# Each point has a distance of 1km to the next one
# The distance between each point is not exactly 1km.
# It doesn't take in count the curvature of the earth
# to adapt the distance.
# Depending of the size of the grid, it could be a few meters
# more or less.

# size of the grid
size_grid = 10
lst_lat_lng = []
lst_lat_lng.append([lat, lng])

x_lng = search_opt(0, 1, 100000, lat, lng, how = "lon")[0]
y_lat = search_opt(0, 1, 100000, lat, lng, how = "lat")[0]

for i in range(size_grid):
    for j in range(size_grid):
        lst_lat_lng.append([lat + i * y_lat, lng + j * x_lng])
```

In [265]:

```
# Extract the total number of venues 1km around a given location
# Extract a maximum of 100 venues
def venues_nb(latitude, longitude):
    url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_s
        "X",
        "X",
        20191120,
        latitude,
        longitude,
        1000,
        100)
    results = requests.get(url).json()
    #print(neighborhood_latitude)
    #print(neighborhood_longitude)
    venues = results['response']['groups'][0]['items']

    nearby_venues = json_normalize(venues) # flatten JSON

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

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

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

    return nearby_venues.shape[0]
```

In [266]:

```
# Create a list with in each line the latitude, the longitude and
# the number of venues near it
lst_lat_lng_nb = []
for i in range(len(lst_lat_lng)):
    nb = venues_nb(lst_lat_lng[i][0], lst_lat_lng[i][1])
    lst_lat_lng_nb.append([lst_lat_lng[i][0], lst_lat_lng[i][1],nb])

# Display the 5 first lines
```

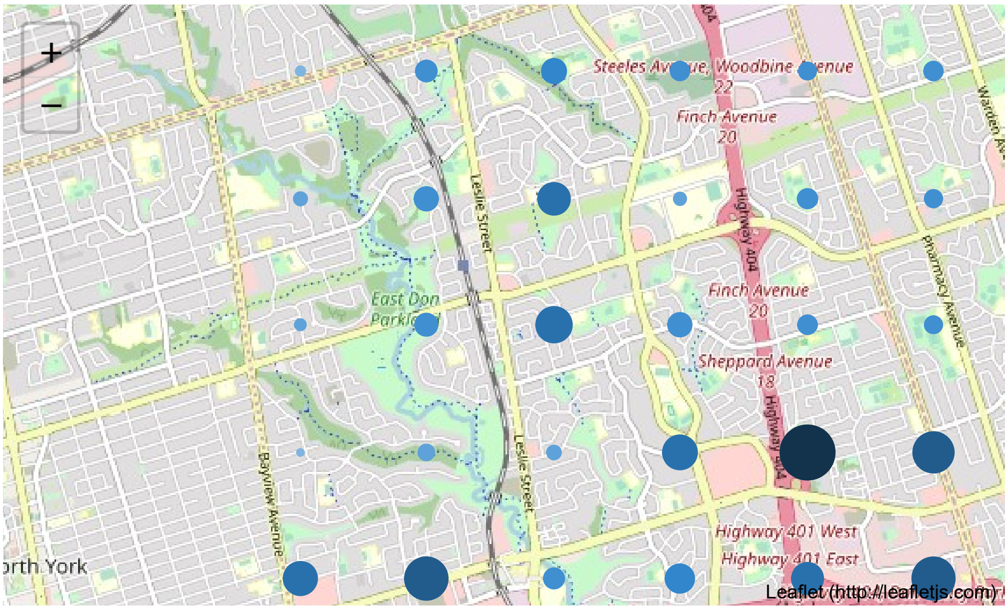
Out[266]:

```
[[43.733962999999996, -79.387207, 8],
 [43.733962999999996, -79.387207, 8],
 [43.733962999999996, -79.374767, 10],
 [43.733962999999996, -79.36232700000001, 6],
 [43.733962999999996, -79.34988700000001, 44]]
```

In [293]:

```
# Create map using latitude and longitude values of the grid
# The more venues there are in a 1km radius around each point
#   the bigger and the darker the point

lat_center, lng_center = pd.DataFrame(lst_lat_lng_nb).mean()[:2]

# map_comp = folium.Map(location=lst_lat_lng_nb[0][:2], zoom_start=12)
map_comp = folium.Map(location=[lat_center, lng_center], zoom_start=13)

color = ['#7fb4e0','#5fa2d9','#408fd1', '#3186cc', '#2971ac', '#225c8d', '#a48

# add markers to map
for point in lst_lat_lng_nb:
    label = 'Number of venues: {}.Lat: {}. Long: {}'.format(point[2],point[0],
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        point[:2],
        radius=point[2]/5,
        popup=label,
        color=color[int(point[2]/10)],
        fill=True,
        fill_color=color[int(point[2]/10)],
        fill_opacity=1,
        parse_html=False).add_to(map_comp)

map_comp
```

Out[293]:


+
−
Steeles Avenue
Woodbine Avenue
22
Finch Avenue
20
Warden Av
Leslie Street
Highway 404
Pharmacy Avenue
East Don Parkland
Finch Avenue
20
Sheppard Avenue
18
Highway 404
Bayview Avenue
Leslie Street
Highway 401 West
Highway 401 East
orth York
Leaflet (http://leafletjs.com)