Introduction & Business Problem :

Problem Background:

New York is the most populous city in the United States.It is diverse and is the financial capital of USA.It provides lot of business oppourtunities and business friendly environment.It has attracted many different players into the market. The city is a major center for banking and finance, retailing, world trade, transportation, tourism, real estate, new media, traditional media, advertising, legal services, accountancy, insurance, theater, fashion, and the arts in the United States. This also means that the market is highly competitive. As it is highly developed city so cost of doing business is also one of the highest. Thus, any new business venture or expansion needs to be analysed carefully. The insights derived from analysis will give good understanding of the business environment which help in strategically targeting the market. This will help in reduction of risk. And the Return on Investment will be reasonable.

Problem Description:

New York is famous for its excelllent cuisine. It's food culture includes an array of international cuisines influenced by the city's immigrant history. Central and Eastern European immigrants, especially Jewish immigrants - bagels, cheesecake, hot dogs, knishes, and delicatessens.

Italian immigrants:New York-style pizza and Italian cuisine. Jewish immigrants and Irish immigrants: pastrami and corned beef Chinese and other Asian restaurants Street food Middle Eastern foods such as falafel The city is home to "nearly one thousand of the finest and most diverse haute cuisine restaurants in the world", according to Michelin. So it is evident that to survive in such competitive market it is very important to startegically plan. Various factors need to be studied in order to decide on the Location such as :

New York Population

New York City Demographics Cuisine served / Menu of the competitors Segmentation of the Borough Untapped markets Saturated markets etc Target population:

The objective is to locate and recommend to the management which neighborhood of New york city will be best choice to start a restaurant. The Management also expects to understand the rationale of the recommendations made. This would interest anyone who wants to start a new restaurant in New york city.

Criteria:

The success criteria of the project will be a good recommendation of borough/Neighborhood choice based on Lack of such restaurants in that location and nearest suppliers of ingredients.

The Battle of the Neighborhoods - Week 2

Data :

We will be using the below datasets:

Data 1 : Neighborhood has a total of 5 boroughs and 306 neighborhoods. In order to segement the neighborhoods and explore them, we will essentially need a dataset that contains the 5 boroughs and the neighborhoods that exist in each borough as well as the the latitude and logitude coordinates of each neighborhood.

This dataset exists for free on the web. Link to the dataset is : (https://geo.nyu.edu/catalog/nyu_2451_34572)

Data 2: [Second data which will be used is the DOHMH Farmers Markets and Food Boxes dataset]

(https://data.cityofnewyork.us/dataset/DOHMH-Farmers-Markets-and-Food-Boxes/8vwk-6iz2 (https://data.cityofnewyork.us/dataset/DOHMH-Farmers-Markets-and-Food-Boxes/8vwk-6iz2))

Website (https://www.grownyc.org/greenmarketco/foodbox)

Data 3 : [Data from wikipedia as given below :]

(https://en.wikipedia.org/wiki/New_York_City (https://en.wikipedia.org/wiki/New_York_City)) (https://en.wikipedia.org/wiki/Economy_of_New_York_City (https://en.wikipedia.org/wiki/Economy_of_New_York_City)) (https://en.wikipedia.org/wiki/Portal:New_York_City (https://en.wikipedia.org/wiki/Portal:New_York_City)) (https://en.wikipedia.org/wiki/Cuisine_of_New_York_City (https://en.wikipedia.org/wiki/Cuisine_of_New_York_City)) (https://en.wikipedia.org/wiki/List_of_Michelin_starred_restaurants_in_New_York_City (https://en.wikipedia.org/wiki/List_of_Michelin_starred_restaurants_in_New_York_City))

Data 4 : Newyork city geographical coordinates data will be utilized as input for the Foursquare API, that will be leveraged to provision venues information for each neighborhood.We will use the Foursquare API to explore neighborhoods in New York City.

New York city geographical coordinates dataset

In [2]:
```
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 h
aven'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 pa
ndas dataframe

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

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

import csv # implements classes to read and write tabular data in CSV fo
rm

print('Libraries imported.')
```

```
Solving environment: done

## Package Plan ##

  environment location: /opt/conda/envs/DSX-Python35

  added / updated specs:
    - geopy


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    openssl-1.0.2r             |       h14c3975_0         3.1 MB  conda
-forge
    certifi-2018.8.24          |        py35_1001         139 KB  conda
-forge
    ca-certificates-2019.6.16  |       hecc5488_0         145 KB  conda
-forge
    geopy-1.20.0               |             py_0          57 KB  conda
-forge
    geographiclib-1.49         |             py_0          32 KB  conda
-forge
    ------------------------------------------------------------
                                           Total:         3.5 MB

The following NEW packages will be INSTALLED:

    geographiclib:   1.49-py_0         conda-forge
    geopy:           1.20.0-py_0       conda-forge

The following packages will be UPDATED:

    ca-certificates: 2019.1.23-0                  --> 2019.6.16-hecc54
88_0 conda-forge
    certifi:         2018.8.24-py35_1             --> 2018.8.24-py35_1
001  conda-forge

The following packages will be DOWNGRADED:

    openssl:         1.0.2s-h7b6447c_0            --> 1.0.2r-h14c3975_
0    conda-forge


Downloading and Extracting Packages
openssl-1.0.2r       | 3.1 MB    | ####################################
# | 100%
certifi-2018.8.24    | 139 KB    | ####################################
# | 100%
ca-certificates-2019 | 145 KB    | ####################################
# | 100%
geopy-1.20.0         | 57 KB     | ####################################
# | 100%
geographiclib-1.49   | 32 KB     | ####################################
# | 100%
Preparing transaction: done
```

```
Verifying transaction: done
Executing transaction: done
Solving environment: done

## Package Plan ##

  environment location: /opt/conda/envs/DSX-Python35

  added / updated specs:
    - folium=0.5.0


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    vincent-0.4.4              |             py_1          28 KB  conda
-forge
    branca-0.3.1               |             py_0          25 KB  conda
-forge
    folium-0.5.0               |             py_0          45 KB  conda
-forge
    altair-2.2.2               |          py35_1         462 KB  conda
-forge
    ------------------------------------------------------------
                                           Total:         560 KB

The following NEW packages will be INSTALLED:

    altair:  2.2.2-py35_1 conda-forge
    branca:  0.3.1-py_0   conda-forge
    folium:  0.5.0-py_0   conda-forge
    vincent: 0.4.4-py_1   conda-forge


Downloading and Extracting Packages
vincent-0.4.4        | 28 KB     | ####################################
# | 100%
branca-0.3.1         | 25 KB     | ####################################
# | 100%
folium-0.5.0         | 45 KB     | ####################################
# | 100%
altair-2.2.2         | 462 KB    | ####################################
# | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
Libraries imported.
```

In [3]:
```
!wget -q -O 'newyork_data.json' https://ibm.box.com/shared/static/fbpwbo
var7lf8p5sgddm06cgipa2rxpe.json
print('Data downloaded!')
```

```
Data downloaded!
```

Data exploratoin

In [4]:
```
with open('newyork_data.json') as json_data:
    newyork_data = json.load(json_data)
neighborhoods_data = newyork_data['features']
```

In [4]:
```
neighborhoods_data[0]
```

Out[4]:
```
{'geometry': {'coordinates': [-73.84720052054902, 40.89470517661],
  'type': 'Point'},
 'geometry_name': 'geom',
 'id': 'nyu_2451_34572.1',
 'properties': {'annoangle': 0.0,
  'annoline1': 'Wakefield',
  'annoline2': None,
  'annoline3': None,
  'bbox': [-73.84720052054902,
   40.89470517661,
   -73.84720052054902,
   40.89470517661],
  'borough': 'Bronx',
  'name': 'Wakefield',
  'stacked': 1},
 'type': 'Feature'}
```

Data Transformation

In [5]:
```
# define the dataframe columns
column_names = ['Borough', 'Neighborhood', 'Latitude', 'Longitude']

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

Out[5]:

	Borough	Neighborhood	Latitude	Longitude

In [6]:
```
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_n
ame,
                                          'Latitude': neighborhood_lat,
                                          'Longitude': neighborhood_lon

}, ignore_index=True)
neighborhoods.head()
```

Out[6]:

	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

In [7]:
```
print('The dataframe has {} boroughs and {} neighborhoods.'.format(
        len(neighborhoods['Borough'].unique()),
        neighborhoods.shape[0]
    )
)
```

```
The dataframe has 5 boroughs and 306 neighborhoods.
```

In [8]:
```
neighborhoods.to_csv('BON1_NYC_GEO.csv',index=False)
```

Get the latitude and longitude values of New York City.

In [9]:
```
address = 'New York City, NY'

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

```
The geograpical coordinate of New York City are 40.7127281, -74.006015
2.
```

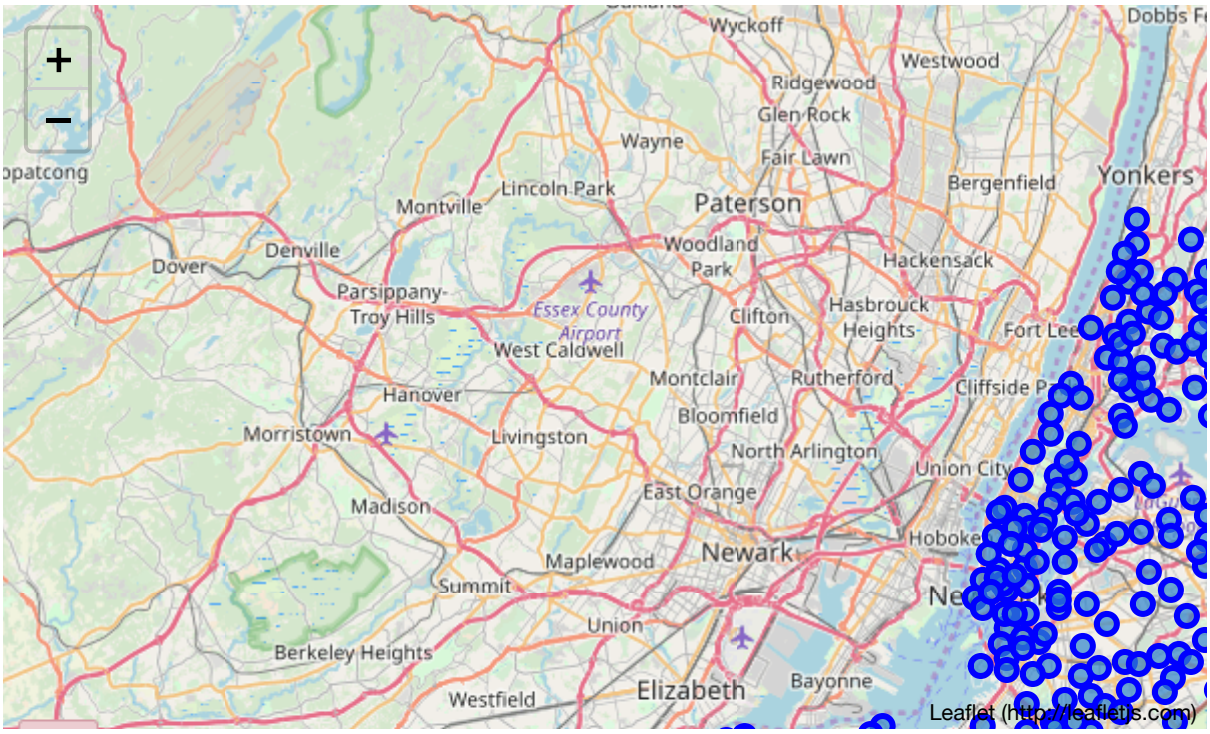
Create a map

In [94]:
```
map_NewYork = folium.Map(location=[latitude, longitude], zoom_start=10)

# add markers to map
for lat, lng, borough, neighborhood in zip(neighborhoods['Latitude'], ne
ighborhoods['Longitude'], neighborhoods['Borough'], neighborhoods['Neigh
borhood']):
    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
```

Out[94]:



Web scrapping of Population and Demographics data of New York city from Wikipedia

Population

In [11]:
```
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 h
aven'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 pa
ndas dataframe

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

# conda install -c anaconda beautiful-soup --yes
from bs4 import BeautifulSoup # package for parsing HTML and XML documen
ts

import csv # implements classes to read and write tabular data in CSV fo
rm

print('Libraries imported.')
```

```
Solving environment: done

# All requested packages already installed.

Libraries imported.
```

Web scrapping of Population data from wikipedia website

In [12]:
```
website_url = requests.get('https://en.wikipedia.org/wiki/Demographics_of_New_York_City').text
soup = BeautifulSoup(website_url,'lxml')
table = soup.find('table',{'class':'wikitable sortable'})
#print(soup.prettify())

headers = [header.text for header in table.find_all('th')]

table_rows = table.find_all('tr')
rows = []
for row in table_rows:
   td = row.find_all('td')
   row = [row.text for row in td]
   rows.append(row)

with open('POPULATION1.csv', 'w') as f:
   writer = csv.writer(f)
   writer.writerow(headers)
   writer.writerows(row for row in rows if row)
```

In [13]:
```
Pop_data=pd.read_csv('POPULATION1.csv')
Pop_data.drop(Pop_data.columns[[7,8,9,10,11]], axis=1,inplace=True)
print('Data downloaded!')
```

```
Data downloaded!
```

Data cleaning

In [14]:
```
Pop_data.columns = Pop_data.columns.str.replace(' ', '')
Pop_data.columns = Pop_data.columns.str.replace('\'','')
Pop_data.rename(columns={'Borough':'persons_sq_mi','County':'persons_sq_
km'}, inplace=True)
Pop_data
```

Out[14]:

	NewYorkCitysfiveboroughsvte	Jurisdiction	Population	GrossDomesticProduct	Landa
0	The Bronx\n	\n Bronx\n	1,471,160\n	28.787\n	19,57(
1	Brooklyn\n	\n Kings\n	2,648,771\n	63.303\n	23,90(
2	Manhattan\n	\n New York\n	1,664,727\n	629.682\n	378,2
3	Queens\n	\n Queens\n	2,358,582\n	73.842\n	31,31(
4	Staten Island\n	\n Richmond\n	479,458\n	11.249\n	23,46(
5	City of New York	8,622,698	806.863	93,574	302.6
6	State of New York	19,849,399	1,547.116	78,354	47,21
7	Sources:[14] and see individual borough articl...	NaN	NaN	NaN	NaN

In [15]:
```
Pop_data.rename(columns = {'NewYorkCitysfiveboroughsvte\n' : 'Borough',
                   'Jurisdiction\n':'County',
                   'Population\n':'Estimate_2017',
                   'Landarea\n':'square_miles',
                    'Density\n':'square_km'}, inplace=True)
Pop_data
```

Out[15]:

	Borough	County	Estimate_2017	GrossDomesticProduct	square_miles	squar
0	The Bronx\n	\n Bronx\n	1,471,160\n	28.787\n	19,570\n	42.10\
1	Brooklyn\n	\n Kings\n	2,648,771\n	63.303\n	23,900\n	70.82\
2	Manhattan\n	\n New York\n	1,664,727\n	629.682\n	378,250\n	22.83\
3	Queens\n	\n Queens\n	2,358,582\n	73.842\n	31,310\n	108.53
4	Staten Island\n	\n Richmond\n	479,458\n	11.249\n	23,460\n	58.37\
5	City of New York	8,622,698	806.863	93,574	302.64	783.83
6	State of New York	19,849,399	1,547.116	78,354	47,214	122,28
7	Sources:[14] and see individual borough articl...	NaN	NaN	NaN	NaN	NaN

In [16]:
```
Pop_data = Pop_data.fillna('')
Pop_data
```

Out[16]:

	Borough	County	Estimate_2017	GrossDomesticProduct	square_miles	squar
0	The Bronx\n	\n Bronx\n	1,471,160\n	28.787\n	19,570\n	42.10\
1	Brooklyn\n	\n Kings\n	2,648,771\n	63.303\n	23,900\n	70.82\
2	Manhattan\n	\n New York\n	1,664,727\n	629.682\n	378,250\n	22.83\
3	Queens\n	\n Queens\n	2,358,582\n	73.842\n	31,310\n	108.53
4	Staten Island\n	\n Richmond\n	479,458\n	11.249\n	23,460\n	58.37\
5	City of New York	8,622,698	806.863	93,574	302.64	783.83
6	State of New York	19,849,399	1,547.116	78,354	47,214	122,28
7	Sources:[14] and see individual borough articl...					

In [17]:
```
i = Pop_data[((Pop_data.County == 'Sources: [2] and see individual borou
gh articles'))].index
Pop_data.drop(i)
```

Out[17]:

	Borough	County	Estimate_2017	GrossDomesticProduct	square_miles	squar
0	The Bronx\n	\n Bronx\n	1,471,160\n	28.787\n	19,570\n	42.10\
1	Brooklyn\n	\n Kings\n	2,648,771\n	63.303\n	23,900\n	70.82\
2	Manhattan\n	\n New York\n	1,664,727\n	629.682\n	378,250\n	22.83\
3	Queens\n	\n Queens\n	2,358,582\n	73.842\n	31,310\n	108.53
4	Staten Island\n	\n Richmond\n	479,458\n	11.249\n	23,460\n	58.37\
5	City of New York	8,622,698	806.863	93,574	302.64	783.83
6	State of New York	19,849,399	1,547.116	78,354	47,214	122,28
7	Sources:[14] and see individual borough articl...					

In [18]:
```
Pop_data.to_csv('POPULATION.csv',index=False)
```

DEMOGRAPHICS

Web scrapping of Demographics data from wikipedia

In [19]:
```
website_url = requests.get('https://en.wikipedia.org/wiki/New_York_City'
).text
soup = BeautifulSoup(website_url,'lxml')
table = soup.find('table',{'class':'wikitable sortable collapsible'})
#print(soup.prettify())

headers = [header.text for header in table.find_all('th')]

table_rows = table.find_all('tr')
rows = []
for row in table_rows:
   td = row.find_all('td')
   row = [row.text for row in td]
   rows.append(row)

with open('DEMO.csv', 'w') as f:
   writer = csv.writer(f)
   writer.writerow(headers)
   writer.writerows(row for row in rows if row)
```

In [20]:
```
Demo_data=pd.read_csv('DEMO.csv')
print('Data downloaded!')
```

```
Data downloaded!
```

In [21]:
```
Demo_data
```

Out[21]:

	Racial composition	2010[249]	1990[251]	1970[251]	1940[251]
0	White	44.0%	52.3%	76.6%	93.6%\n
1	—Non-Hispanic	33.3%	43.2%	62.9%[252]	92.0%\n
2	Black or African American	25.5%	28.7%	21.1%	6.1%\n
3	Hispanic or Latino (of any race)	28.6%	24.4%	16.2%[252]	1.6%\n
4	Asian	12.7%	7.0%	1.2%	–\n

Data cleaning

In [22]:
```
Demo_data.columns
```

Out[22]:
```
Index(['Racial composition', '2010[249]', '1990[251]', '1970[251]',
       '1940[251]\n'],
      dtype='object')
```

In [23]:
```
Demo_data.rename(columns = {'2010[237]' : '2010',
                    '1990[239]':'1990',
                    '1970[239]':'1970',
                    '1940[239]\n':'1940',
                     }, inplace=True)
Demo_data
```

Out[23]:

	Racial composition	2010[249]	1990[251]	1970[251]	1940[251]
0	White	44.0%	52.3%	76.6%	93.6%\n
1	—Non-Hispanic	33.3%	43.2%	62.9%[252]	92.0%\n
2	Black or African American	25.5%	28.7%	21.1%	6.1%\n
3	Hispanic or Latino (of any race)	28.6%	24.4%	16.2%[252]	1.6%\n
4	Asian	12.7%	7.0%	1.2%	–\n

In [24]:
```
Demo_data.columns
```

Out[24]:
```
Index(['Racial composition', '2010[249]', '1990[251]', '1970[251]',
       '1940[251]\n'],
      dtype='object')
```

In [25]:
```
Demo_data.columns = Demo_data.columns.str.replace(' ', '')
```

In [26]:
```
Demo_data= Demo_data.replace('\n',' ', regex=True)
Demo_data
```

Out[26]:

	Racialcomposition	2010[249]	1990[251]	1970[251]	1940[251]
0	White	44.0%	52.3%	76.6%	93.6%
1	—Non-Hispanic	33.3%	43.2%	62.9%[252]	92.0%
2	Black or African American	25.5%	28.7%	21.1%	6.1%
3	Hispanic or Latino (of any race)	28.6%	24.4%	16.2%[252]	1.6%
4	Asian	12.7%	7.0%	1.2%	–

In [90]:
```
Demo_data.to_csv('DEMOGRAPHICS.csv',index=False)
```

In [ ]:
```

```

Farmers market data

In [30]:
```
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 h
aven'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 pa
ndas dataframe

# Matplotlib and associated plotting modules
import matplotlib.cm as cm
import matplotlib.colors as colors
import matplotlib.pyplot as plt
from matplotlib.ticker import NullFormatter
import matplotlib.ticker as ticker

# notice: installing seaborn might takes a few minutes
!conda install -c anaconda seaborn -y
import seaborn as sns

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

print('Libraries imported.')
```

```
Solving environment: done

# All requested packages already installed.

Solving environment: done

## Package Plan ##

  environment location: /opt/conda/envs/DSX-Python35

  added / updated specs:
    - seaborn


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    certifi-2018.8.24          |           py35_1         139 KB  anaco
nda
    openssl-1.0.2s             |       h7b6447c_0         3.1 MB  anaco
nda
    seaborn-0.9.0              |           py35_0         378 KB  anaco
nda
    ca-certificates-2019.5.15  |                0         133 KB  anaco
nda
    ------------------------------------------------------------
                                           Total:         3.8 MB

The following packages will be UPDATED:

    openssl:         1.0.2r-h14c3975_0    conda-forge --> 1.0.2s-h7b644
7c_0 anaconda
    seaborn:         0.8.0-py35h15a2772_0             --> 0.9.0-py35_0
anaconda

The following packages will be DOWNGRADED:

    ca-certificates: 2019.6.16-hecc5488_0 conda-forge --> 2019.5.15-0
anaconda
    certifi:         2018.8.24-py35_1001  conda-forge --> 2018.8.24-py3
5_1  anaconda


Downloading and Extracting Packages
certifi-2018.8.24    | 139 KB    | ####################################
# | 100%
openssl-1.0.2s       | 3.1 MB    | ####################################
# | 100%
seaborn-0.9.0        | 378 KB    | ####################################
# | 100%
ca-certificates-2019 | 133 KB    | ####################################
# | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
Solving environment: done
```

```
## Package Plan ##

  environment location: /opt/conda/envs/DSX-Python35

  added / updated specs:
    - folium=0.5.0


The following packages will be UPDATED:

    ca-certificates: 2019.5.15-0       anaconda --> 2019.6.16-hecc5488_
0 conda-forge
    certifi:         2018.8.24-py35_1  anaconda --> 2018.8.24-py35_1001
conda-forge

The following packages will be DOWNGRADED:

    openssl:         1.0.2s-h7b6447c_0 anaconda --> 1.0.2r-h14c3975_0
conda-forge

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
Libraries imported.
```

In [49]:
```
# Data from website - https://data.cityofnewyork.us/dataset/DOHMH-Farmers-Markets-and-Food-Boxes/8vwk-6iz2
body = client_697c296344734375af43f9b31ad91125.get_object(Bucket='capstone-donotdelete-pr-w81lc2lnw2glhn',Key='dohmh-farmers-markets-and-food-boxes-1.csv')['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

FM_NYC = pd.read_csv(body)
FM_NYC.head()
```

Out[49]:

	FacilityName	Service Category	Service Type	Address	Address 2	Borough	ZipCode	Latitude
0	Inwood Park Greenmarket	Farmers Markets and Food Boxes	Farmers Markets	Isham St bet Seaman & Cooper	NaN	Manhattan	10034	40.869009
1	82nd Street Greenmarket	Farmers Markets and Food Boxes	Farmers Markets	82nd St bet 1st & York Aves	NaN	Manhattan	10028	40.773448
2	1 Centre Street	Farmers Markets and Food Boxes	Food Boxes	1 Centre Street	South Building, 9th Floor	Manhattan	11101	40.713028
3	125th Street Farmers Market	Farmers Markets and Food Boxes	Farmers Markets	125th St & Adam Clayton Powell Jr Blvd	NaN	Manhattan	10027	40.808981
4	170 Farm Stand	Farmers Markets and Food Boxes	Farmers Markets	170th St & Townsend Ave	NaN	Bronx	10452	40.840095

In [50]:
```
FM_NYC.rename(columns={'Service Type':'Service_Type'}, inplace=True)
print(FM_NYC.Service_Type.unique())
```

```
['Farmers Markets' 'Food Boxes']
```

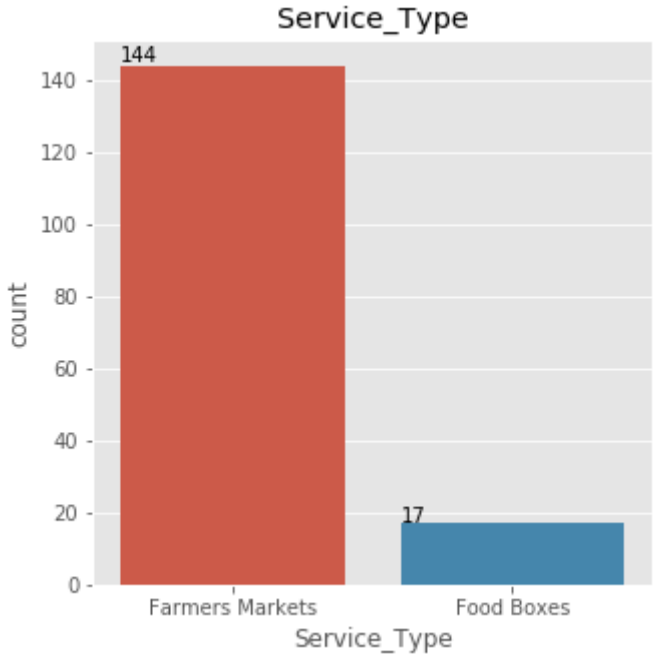
In [51]:
```
FM_NYC['Service_Type'].value_counts().to_frame()
```

Out[51]:

	Service_Type
Farmers Markets	144
Food Boxes	17

In [52]:
```
fig,ax = plt.subplots(1, 1, figsize=(5, 5))
sns.countplot(x='Service_Type',data=FM_NYC)
ax.set_title("Service_Type")
for t in ax.patches:
    if (np.isnan(float(t.get_height()))):
        ax.annotate('', (t.get_x(), 0))
    else:
        ax.annotate(str(format(int(t.get_height()), ',d')), (t.get_x(),
t.get_height()*1.01))

plt.show();
```



In [53]:
```
# FM_NYC_filtered - Dataset with only Farmers Market
FM_NYC_filtered = FM_NYC[FM_NYC['Service_Type'] == 'Farmers Markets'].co
py()
FM_NYC_filtered ['Borough'] = FM_NYC_filtered['Borough'].map(lambda x: x
.strip())
print(FM_NYC_filtered.shape)
FM_NYC_filtered.head()
```

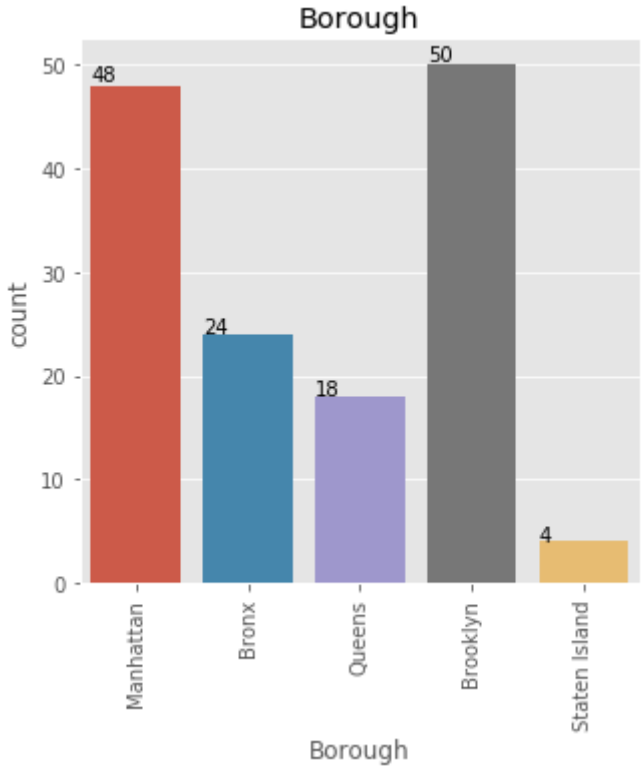
```
(144, 24)
```

Out[53]:

	FacilityName	Service Category	Service_Type	Address	Address 2	Borough	ZipCode	Lati
0	Inwood Park Greenmarket	Farmers Markets and Food Boxes	Farmers Markets	Isham St bet Seaman & Cooper	NaN	Manhattan	10034	40.86
1	82nd Street Greenmarket	Farmers Markets and Food Boxes	Farmers Markets	82nd St bet 1st & York Aves	NaN	Manhattan	10028	40.77
3	125th Street Farmers Market	Farmers Markets and Food Boxes	Farmers Markets	125th St & Adam Clayton Powell Jr Blvd	NaN	Manhattan	10027	40.80
4	170 Farm Stand	Farmers Markets and Food Boxes	Farmers Markets	170th St & Townsend Ave	NaN	Bronx	10452	40.84
5	175th Street Greenmarket	Farmers Markets and Food Boxes	Farmers Markets	175th St bet Wadsworth Ave & Broadway	NaN	Manhattan	10033	40.84

In [54]:
```
fig,ax = plt.subplots(1, 1, figsize=(5, 5))
sns.countplot(x='Borough',data=FM_NYC_filtered)
ax.set_title("Borough")
for t in ax.patches:
    if (np.isnan(float(t.get_height()))):
        ax.annotate('', (t.get_x(), 0))
    else:
        ax.annotate(str(format(int(t.get_height()), ',d')), (t.get_x(),
t.get_height()*1.01))
        ax.set_xticklabels([t.get_text().split("T")[0] for t in ax.get_x
ticklabels()])

# This sets the yticks "upright" with 0, as opposed to sideways with 90.
plt.xticks(rotation=90)
plt.show()
```



In [55]:
```
address = 'New York City, NY'

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

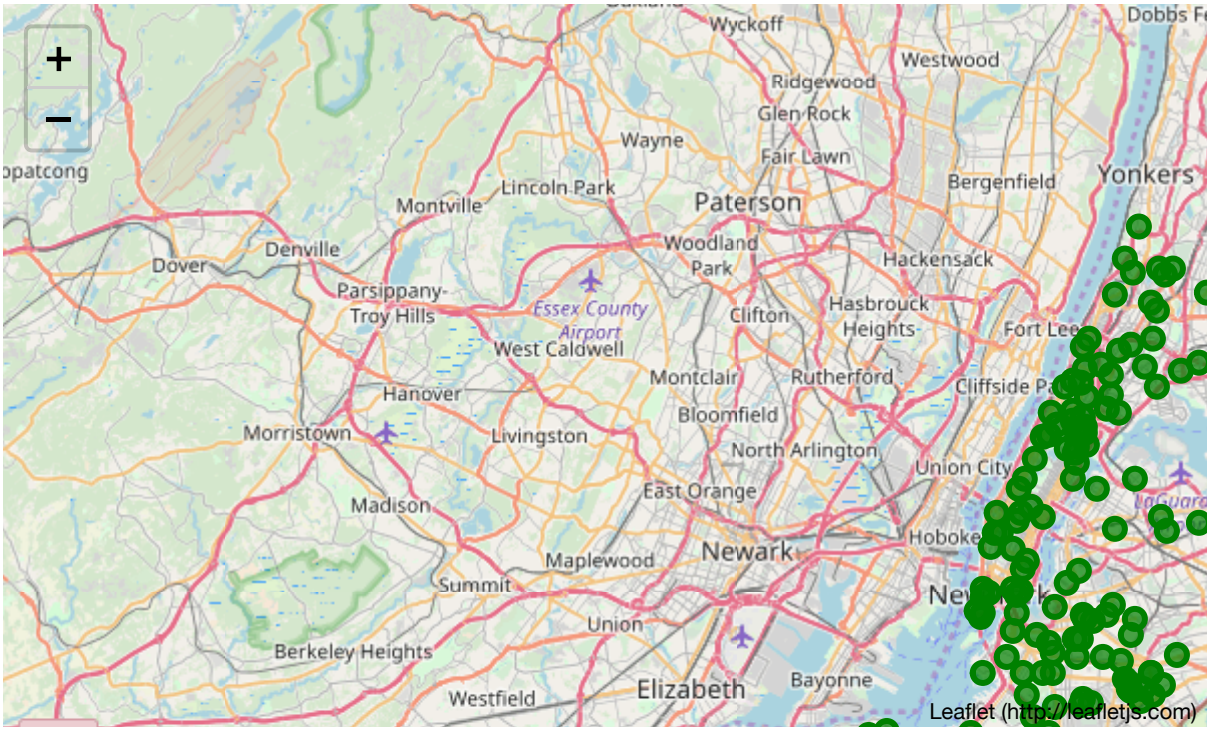
```
The geograpical coordinate of New York City are 40.7127281, -74.006015
2.
```

In [95]:
```
# create map of New York City using latitude and longitude values
map_markets = folium.Map(location=[latitude, longitude], zoom_start=10)

# add markers to map
for lat, lng, FacilityName, borough in zip(FM_NYC_filtered['Latitude'],
FM_NYC_filtered['Longitude'], FM_NYC_filtered['FacilityName'], FM_NYC_fi
ltered['Borough']):
            label = '{}, {}'.format(FacilityName, borough)
            label = folium.Popup(label, parse_html=True)
            folium.CircleMarker(
                [lat, lng],
                radius=5,
                popup=label,
                color='green',
                fill=True,
                fill_color='green',
                fill_opacity=0.7,
                parse_html = False).add_to(map_markets)

map_markets
```

Out[95]:



Segmenting and Clustering Neighborhoods - Queens and Manhattan

In [57]:
```
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 h
aven'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 pa
ndas dataframe

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

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

from sklearn.metrics import silhouette_score

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

print('Libraries imported.')
```

```
Solving environment: done

# All requested packages already installed.

Solving environment: done

# All requested packages already installed.

Libraries imported.
```

Load data

In [59]:
```
NYC_Geo=pd.read_csv('BON1_NYC_GEO.csv')
print('Data downloaded!')
NYC_Geo.head()
NYC_Geo['Borough'].value_counts().to_frame()
print(NYC_Geo.Borough.unique())
```

```
Data downloaded!
['Bronx' 'Manhattan' 'Brooklyn' 'Queens' 'Staten Island']
```

Segmenting and Clustering Neighborhoods - Queens and Manhattan

In [61]:
```
QM_Geo = NYC_Geo.loc[(NYC_Geo['Borough'] == 'Queens')|(NYC_Geo['Borough'
] == 'Manhattan')]
QM_Geo = QM_Geo.reset_index(drop=True)
QM_Geo.head()
```

Out[61]:

	Borough	Neighborhood	Latitude	Longitude
0	Manhattan	Marble Hill	40.876551	-73.910660
1	Manhattan	Chinatown	40.715618	-73.994279
2	Manhattan	Washington Heights	40.851903	-73.936900
3	Manhattan	Inwood	40.867684	-73.921210
4	Manhattan	Hamilton Heights	40.823604	-73.949688

In [62]:
```
import time
start_time = time.time()

address = 'New York City, NY'

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

print("--- %s seconds ---" % round((time.time() - start_time), 2))
```

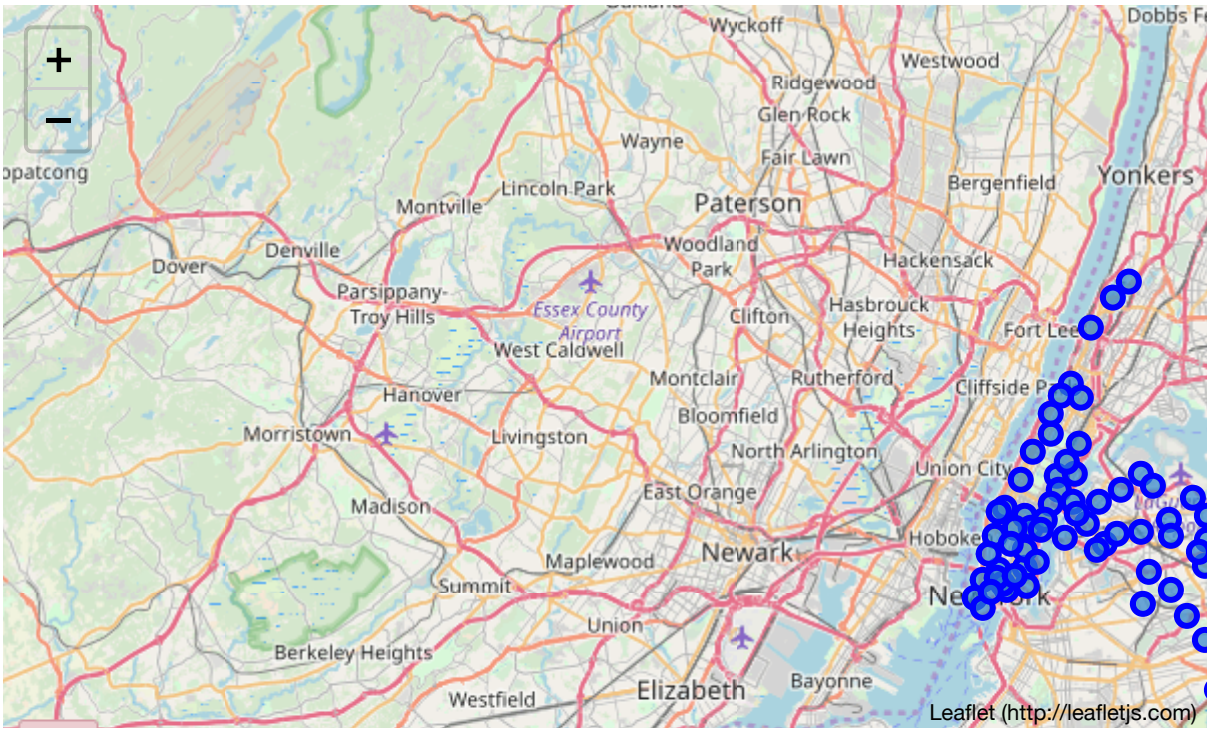
```
The geograpical coordinate of New York City are 40.7127281, -74.006015
2.
--- 0.55 seconds ---
```

In [96]:
```
# create map of Toronto using latitude and longitude values
map_QM = folium.Map(location=[latitude, longitude], zoom_start=10)

# add markers to map
for lat, lng, borough, neighborhood in zip(QM_Geo['Latitude'], QM_Geo['L
ongitude'], QM_Geo['Borough'], QM_Geo['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_QM)

map_QM
```

Out[96]:



Define Foursquare Credentials and Version

In [64]:
```
CLIENT_ID = 'OLN1BAQQBHO234LKFIU1ZNGV4Z3O3P1GS5KIMTNPJHLX1MKL' # your Fo
ursquare ID
CLIENT_SECRET = 'VDM5CGGVSUOGKMY21ETO4J1UAJH5QJEALQCJAIWUF2DJXR2T' # you
r Foursquare Secret
VERSION = '20181218' # Foursquare API version

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

```
Your credentails:
CLIENT_ID: OLN1BAQQBHO234LKFIU1ZNGV4Z3O3P1GS5KIMTNPJHLX1MKL
CLIENT_SECRET:VDM5CGGVSUOGKMY21ETO4J1UAJH5QJEALQCJAIWUF2DJXR2T
```

extract neighboors in Queens and Mahattan

In [65]:
```
def getNearbyVenues(names, latitudes, longitudes, LIMIT=200, radius=1000
):

    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]['ite
ms']

        # 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 [66]:
```
QM_venues = getNearbyVenues(names=QM_Geo['Neighborhood'],
                                    latitudes=QM_Geo['Latitude'],
                                    longitudes=QM_Geo['Longitude'],
                                    LIMIT=200)

print('The "QM_venues" dataframe has {} venues and {} unique venue type
s.'.format(
      len(QM_venues['Venue Category']),
      len(QM_venues['Venue Category'].unique())))

QM_venues.to_csv('QM_venues.csv', sep=',', encoding='UTF8')
QM_venues.head()
```

```
Marble Hill
Chinatown
Washington Heights
Inwood
Hamilton Heights
Manhattanville
Central Harlem
East Harlem
Upper East Side
Yorkville
Lenox Hill
Roosevelt Island
Upper West Side
Lincoln Square
Clinton
Midtown
Murray Hill
Chelsea
Greenwich Village
East Village
Lower East Side
Tribeca
Little Italy
Soho
West Village
Manhattan Valley
Morningside Heights
Gramercy
Battery Park City
Financial District
Astoria
Woodside
Jackson Heights
Elmhurst
Howard Beach
Corona
Forest Hills
Kew Gardens
Richmond Hill
Flushing
Long Island City
Sunnyside
East Elmhurst
Maspeth
Ridgewood
Glendale
Rego Park
Woodhaven
Ozone Park
South Ozone Park
College Point
Whitestone
Bayside
Auburndale
Little Neck
Douglaston
Glen Oaks
```

```
Bellerose
Kew Gardens Hills
Fresh Meadows
Briarwood
Jamaica Center
Oakland Gardens
Queens Village
Hollis
South Jamaica
St. Albans
Rochdale
Springfield Gardens
Cambria Heights
Rosedale
Far Rockaway
Broad Channel
Breezy Point
Steinway
Beechhurst
Bay Terrace
Edgemere
Arverne
Rockaway Beach
Neponsit
Murray Hill
Floral Park
Holliswood
Jamaica Estates
Queensboro Hill
Hillcrest
Ravenswood
Lindenwood
Laurelton
Lefrak City
Belle Harbor
Rockaway Park
Somerville
Brookville
Bellaire
North Corona
Forest Hills Gardens
Carnegie Hill
Noho
Civic Center
Midtown South
Jamaica Hills
Utopia
Pomonok
Astoria Heights
Sutton Place
Hunters Point
Turtle Bay
Tudor City
Stuyvesant Town
Flatiron
Sunnyside Gardens
Blissville
```

```
Roxbury
Middle Village
Malba
Hudson Yards
Hammels
Bayswater
Queensbridge
The "QM_venues" dataframe has 9297 venues and 417 unique venue types.
```

Out[66]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Ve Categ
0	Marble Hill	40.876551	-73.91066	Arturo's	40.874412	-73.910271	Pizza Place
1	Marble Hill	40.876551	-73.91066	Bikram Yoga	40.876844	-73.906204	Yoga Studio
2	Marble Hill	40.876551	-73.91066	Tibbett Diner	40.880404	-73.908937	Diner
3	Marble Hill	40.876551	-73.91066	Sam's Pizza	40.879435	-73.905859	Pizza Place
4	Marble Hill	40.876551	-73.91066	Starbucks	40.877531	-73.905582	Coffee Shop

In [67]:
```
colnames = ['Neighborhood', 'Neighborhood Latitude', 'Neighborhood Longi
tude', 'Venue', 'Venue Latitude', 'Venue Longitude', 'Venue Category']
QM_venues = pd.read_csv('QM_venues.csv', skiprows=1, names=colnames)
QM_venues.columns = QM_venues.columns.str.replace(' ', '')
QM_venues.head()
```

Out[67]:

	Neighborhood	NeighborhoodLatitude	NeighborhoodLongitude	Venue	VenueLatitu
0	Marble Hill	40.876551	-73.91066	Arturo's	40.874412
1	Marble Hill	40.876551	-73.91066	Bikram Yoga	40.876844
2	Marble Hill	40.876551	-73.91066	Tibbett Diner	40.880404
3	Marble Hill	40.876551	-73.91066	Sam's Pizza	40.879435
4	Marble Hill	40.876551	-73.91066	Starbucks	40.877531

Visualization

In [68]:
```
def Venues_Map(Borough_name, Borough_neighborhoods):

    # Use geopy library to get the latitude and longitude values
    geolocator = Nominatim(user_agent="Jupyter")
    Borough_location = geolocator.geocode(Borough_name) #'Brooklyn, NY'
    Borough_latitude = Borough_location.latitude
    Borough_longitude = Borough_location.longitude
    print('The geographical coordinates of "{}" are {}, {}.'.format(Boro
ugh_name, Borough_latitude, Borough_longitude))

    # To verify the number of Boroughs and Neighborhoods in the extracte
d data
    print('The "{}" dataframe has {} different venue types and {} neighb
orhoods.'.format(
          Borough_name,
          len(Borough_neighborhoods['VenueCategory'].unique()),
          len(Borough_neighborhoods['Neighborhood'].unique())))

    # create map of city using latitude and longitude values
    map_Borough = folium.Map(location=[Borough_latitude, Borough_longitu
de], zoom_start=10)

    # add markers to map
    for lat, lng, venue, category in zip(Borough_neighborhoods['VenueLat
itude'], Borough_neighborhoods['VenueLongitude'], Borough_neighborhoods[
'Venue'], Borough_neighborhoods['VenueCategory']):
        label = '{}, {}'.format(category, venue)
        label = folium.Popup(label, parse_html=True)
        folium.CircleMarker(
            [lat, lng],
            radius=0.1,
            popup=label,
            color='red',
            fill=True,
            fill_color='#FF0000',
            fill_opacity=0.3).add_to(map_Borough)

    return map_Borough
```

In [97]:
```
Venues_Map('New York City, NY', QM_venues)
```

```
The geographical coordinates of "New York City, NY" are 40.7127281, -7
4.0060152.
The "New York City, NY" dataframe has 417 different venue types and 120
neighborhoods.
```

Out[97]:

In [72]:
```
QM_venues.groupby('VenueCategory')['Venue'].count().sort_values(ascending=False)
```

Out[72]:

VenueCategory	
Pizza Place	347
Coffee Shop	260
Italian Restaurant	233
Bakery	229
Chinese Restaurant	196
Deli / Bodega	189
Park	186
Bar	168
Café	157
American Restaurant	153
Ice Cream Shop	148
Sandwich Place	148
Donut Shop	145
Mexican Restaurant	143
Gym	135
Gym / Fitness Center	134
Grocery Store	132
Pharmacy	124
Hotel	117
Bank	111
Sushi Restaurant	110
Indian Restaurant	110
Korean Restaurant	103
Japanese Restaurant	103
Latin American Restaurant	96
Thai Restaurant	91
Cosmetics Shop	89
Supermarket	87
Wine Shop	84
Spa	81
Seafood Restaurant	81
Bagel Shop	80
Fast Food Restaurant	78
Bus Station	76
Cocktail Bar	76
Diner	76
Wine Bar	72
Yoga Studio	68
Asian Restaurant	68
French Restaurant	66
Mobile Phone Shop	64
Caribbean Restaurant	64
Burger Joint	64
Food Truck	62
Beach	61
Greek Restaurant	59
Playground	56
Restaurant	56
Liquor Store	55
Fried Chicken Joint	54
Convenience Store	54
Juice Bar	50
Clothing Store	48
Dessert Shop	48
Mediterranean Restaurant	47
Theater	46

Salon / Barbershop	46
Discount Store	45
Bubble Tea Shop	43
Spanish Restaurant	42
Vegetarian / Vegan Restaurant	42
Art Gallery	40
Gourmet Shop	38
Lounge	38
Pub	37
South American Restaurant	36
New American Restaurant	36
Cycle Studio	34
Steakhouse	34
Plaza	34
Bus Stop	33
Furniture / Home Store	33
Pet Store	33
Bookstore	33
Sporting Goods Shop	32
Dance Studio	32
Department Store	32
Dog Run	31
Boutique	31
Vietnamese Restaurant	30
Peruvian Restaurant	30
Southern / Soul Food Restaurant	30
Gift Shop	30
Middle Eastern Restaurant	29
Rental Car Location	28
Women's Store	28
Scenic Lookout	28
Farmers Market	27
Intersection	27
Salad Place	27
Shoe Store	27
BBQ Joint	26
Taco Place	26
Arts & Crafts Store	25
Art Museum	23
Cuban Restaurant	22
Performing Arts Venue	21
Gas Station	21
Flower Shop	21
Speakeasy	21
Shopping Mall	21
Martial Arts Dojo	21
Ramen Restaurant	21
Breakfast Spot	20
Indie Theater	20
Cheese Shop	19
Health & Beauty Service	19
Optical Shop	18
Tapas Restaurant	18
Athletics & Sports	18
Frozen Yogurt Shop	18
Video Game Store	18
Electronics Store	18

Trail	18
Tennis Court	18
Men's Store	17
Food Court	17
Gastropub	17
Tea Room	17
Event Space	17
Jazz Club	17
Baseball Field	17
Train Station	16
Exhibit	16
Argentinian Restaurant	16
Video Store	16
Kids Store	16
Supplement Shop	16
Garden	16
Smoke Shop	16
Jewelry Store	15
Food & Drink Shop	15
Brazilian Restaurant	15
Museum	15
Harbor / Marina	15
Shipping Store	15
Hotel Bar	14
Pool	14
Music Venue	14
History Museum	14
Dumpling Restaurant	14
Lingerie Store	13
Market	13
Food	13
Nail Salon	13
Nightclub	13
Snack Place	13
Boat or Ferry	13
Shanghai Restaurant	13
Beer Bar	13
Filipino Restaurant	12
Brewery	12
Cupcake Shop	12
Paper / Office Supplies Store	12
Indie Movie Theater	12
Metro Station	12
Surf Spot	12
Falafel Restaurant	11
Sports Bar	11
Hookah Bar	11
Bus Line	11
Health Food Store	11
Boxing Gym	11
Karaoke Bar	11
Wings Joint	11
Beer Garden	11
Arepa Restaurant	10
Massage Studio	10
School	10
Australian Restaurant	10

Track	10
Building	10
Turkish Restaurant	10
Movie Theater	10
Toy / Game Store	9
Beer Store	9
Cantonese Restaurant	9
Eastern European Restaurant	9
Basketball Court	9
Noodle House	9
Hot Dog Joint	9
Miscellaneous Shop	9
Sake Bar	9
Tennis Stadium	9
Pilates Studio	9
Monument / Landmark	8
Whisky Bar	8
Accessories Store	8
Historic Site	8
Roof Deck	8
Climbing Gym	8
Burrito Place	8
Szechuan Restaurant	8
Candy Store	8
Empanada Restaurant	8
Cajun / Creole Restaurant	8
Ethiopian Restaurant	8
Bistro	8
Board Shop	7
Hardware Store	7
Soccer Field	7
Laundromat	7
German Restaurant	7
Japanese Curry Restaurant	7
Home Service	7
Moving Target	7
Automotive Shop	7
Hotpot Restaurant	6
Halal Restaurant	6
Dive Bar	6
Himalayan Restaurant	6
Colombian Restaurant	6
Comedy Club	6
Fountain	6
Tibetan Restaurant	6
Chocolate Shop	6
Creperie	6
Bowling Alley	6
Music Store	6
Pet Service	5
Fruit & Vegetable Store	5
African Restaurant	5
Airport Terminal	5
Record Shop	5
General Entertainment	5
Malay Restaurant	5
Public Art	5

Memorial Site	5
Soup Place	5
Dry Cleaner	5
Weight Loss Center	5
Pie Shop	5
Comfort Food Restaurant	5
Shop & Service	5
Moroccan Restaurant	5
Butcher	5
Flea Market	5
Indonesian Restaurant	4
Tattoo Parlor	4
College Theater	4
Thrift / Vintage Store	4
Comic Shop	4
Golf Course	4
Concert Hall	4
Airport Service	4
Fish Market	4
Residential Building (Apartment / Condo)	4
Warehouse Store	4
Hostel	4
Office	4
Kosher Restaurant	4
Outdoors & Recreation	4
Design Studio	4
Lake	4
Other Nightlife	4
Hawaiian Restaurant	4
Beach Bar	4
Sculpture Garden	4
Organic Grocery	4
Science Museum	4
Gym Pool	4
Farm	3
Czech Restaurant	3
Antique Shop	3
Fish & Chips Shop	3
Waste Facility	3
Waterfront	3
Parking	3
Perfume Shop	3
Pet Café	3
Udon Restaurant	3
Community Center	3
Dim Sum Restaurant	3
Persian Restaurant	3
Afghan Restaurant	3
Salvadoran Restaurant	3
Pool Hall	3
Outdoor Sculpture	3
Sri Lankan Restaurant	3
Lawyer	3
Rest Area	3
Lebanese Restaurant	3
Library	3
Soba Restaurant	3

Skating Rink	3
Big Box Store	3
Bike Trail	3
State / Provincial Park	3
Jewish Restaurant	3
Rock Club	3
Austrian Restaurant	3
Strip Club	3
Cambodian Restaurant	3
Bed & Breakfast	3
Irish Pub	3
Poke Place	3
Pier	3
Baseball Stadium	3
Check Cashing Service	2
Circus	2
Church	2
Business Service	2
Adult Boutique	2
Cafeteria	2
Construction & Landscaping	2
Camera Store	2
Auto Garage	2
College Academic Building	2
Auditorium	2
Arts & Entertainment	2
Airport Lounge	2
ATM	2
Pedestrian Plaza	2
Music School	2
Kebab Restaurant	2
Tailor Shop	2
TV Station	2
Street Art	2
Mini Golf	2
Molecular Gastronomy Restaurant	2
Motel	2
Motorcycle Shop	2
Multiplex	2
Neighborhood	2
Taiwanese Restaurant	2
Newsstand	2
Sports Club	2
Opera House	2
Souvlaki Shop	2
Outdoor Supply Store	2
Pakistani Restaurant	2
Post Office	2
Russian Restaurant	2
Rental Service	2
Israeli Restaurant	2
Street Food Gathering	2
Resort	2
Venezuelan Restaurant	2
Gay Bar	2
Vape Store	2
Field	2

Hobby Shop	2
Tram Station	2
Train	2
Tourist Information Center	2
English Restaurant	2
Tech Startup	2
Heliport	2
High School	2
Scandinavian Restaurant	1
Bath House	1
Basketball Stadium	1
Skate Park	1
Volleyball Court	1
Smoothie Shop	1
Romanian Restaurant	1
Soccer Stadium	1
Social Club	1
Watch Shop	1
Veterinarian	1
Tanning Salon	1
Spiritual Center	1
Baby Store	1
Ukrainian Restaurant	1
Stables	1
Animal Shelter	1
Arcade	1
Storage Facility	1
Theme Park Ride / Attraction	1
Auto Workshop	1
Tex-Mex Restaurant	1
Rock Climbing Spot	1
Swiss Restaurant	1
Synagogue	1
Used Bookstore	1
Coworking Space	1
River	1
Kitchen Supply Store	1
Insurance Office	1
Indoor Play Area	1
Gun Range	1
Club House	1
Government Building	1
Golf Driving Range	1
Gluten-free Restaurant	1
College Basketball Court	1
College Cafeteria	1
College Gym	1
College Residence Hall	1
General Travel	1
College Track	1
Garden Center	1
Food Service	1
Egyptian Restaurant	1
Drugstore	1
Caucasian Restaurant	1
Car Wash	1
Bike Shop	1

```
Laundry Service                 1
Recording Studio                1
Portuguese Restaurant           1
Polish Restaurant               1
Piano Bar                       1
Botanical Garden                1
Photography Studio              1
Daycare                         1
Pastry Shop                     1
Other Great Outdoors            1
Bridge                          1
Buffet                          1
Non-Profit                      1
National Park                   1
Modern Greek Restaurant         1
Mattress Store                  1
Locksmith                       1
Lighthouse                      1
Zoo                             1
Name: Venue, dtype: int64
```

In [73]:
```
QM_venues.groupby('Neighborhood').count()
```

Out[73]:

	NeighborhoodLatitude	NeighborhoodLongitude	Venue	VenueLatitude	V
Neighborhood					
Arverne	37	37	37	37	3
Astoria	100	100	100	100	1
Astoria Heights	78	78	78	78	7
Auburndale	100	100	100	100	1
Battery Park City	100	100	100	100	1
Bay Terrace	68	68	68	68	6
Bayside	100	100	100	100	1
Bayswater	8	8	8	8	8
Beechhurst	56	56	56	56	5
Bellaire	56	56	56	56	5
Belle Harbor	28	28	28	28	2
Bellerose	53	53	53	53	5
Blissville	69	69	69	69	6
Breezy Point	5	5	5	5	5
Briarwood	86	86	86	86	8
Broad Channel	13	13	13	13	1
Brookville	17	17	17	17	1
Cambria Heights	32	32	32	32	3
Carnegie Hill	100	100	100	100	1
Central Harlem	100	100	100	100	1
Chelsea	100	100	100	100	1
Chinatown	100	100	100	100	1
Civic Center	100	100	100	100	1
Clinton	100	100	100	100	1
College Point	79	79	79	79	7
Corona	100	100	100	100	1
Douglaston	67	67	67	67	6

	NeighborhoodLatitude	NeighborhoodLongitude	Venue	VenueLatitude	V
Neighborhood					
East Elmhurst	85	85	85	85	8
East Harlem	100	100	100	100	1
East Village	100	100	100	100	1
Edgemere	25	25	25	25	2
Elmhurst	100	100	100	100	1
Far Rockaway	29	29	29	29	2
Financial District	100	100	100	100	1
Flatiron	100	100	100	100	1
Floral Park	75	75	75	75	7
Flushing	100	100	100	100	1
Forest Hills	100	100	100	100	1
Forest Hills Gardens	100	100	100	100	1
Fresh Meadows	82	82	82	82	8
Glen Oaks	42	42	42	42	4
Glendale	62	62	62	62	6
Gramercy	100	100	100	100	1
Greenwich Village	100	100	100	100	1
Hamilton Heights	100	100	100	100	1
Hammels	57	57	57	57	5
Hillcrest	67	67	67	67	6
Hollis	32	32	32	32	3
Holliswood	48	48	48	48	4
Howard Beach	63	63	63	63	6
Hudson Yards	100	100	100	100	1
Hunters Point	100	100	100	100	1
Inwood	100	100	100	100	1

	NeighborhoodLatitude	NeighborhoodLongitude	Venue	VenueLatitude	V
Neighborhood					
Jackson Heights	100	100	100	100	1
Jamaica Center	87	87	87	87	8
Jamaica Estates	54	54	54	54	5
Jamaica Hills	100	100	100	100	1
Kew Gardens	40	40	40	40	4
Kew Gardens Hills	70	70	70	70	7
Laurelton	47	47	47	47	4
Lefrak City	100	100	100	100	1
Lenox Hill	100	100	100	100	1
Lincoln Square	100	100	100	100	1
Lindenwood	64	64	64	64	6
Little Italy	100	100	100	100	1
Little Neck	72	72	72	72	7
Long Island City	100	100	100	100	1
Lower East Side	100	100	100	100	1
Malba	55	55	55	55	5
Manhattan Valley	100	100	100	100	1
Manhattanville	100	100	100	100	1
Marble Hill	85	85	85	85	8
Maspeth	82	82	82	82	8
Middle Village	73	73	73	73	7
Midtown	100	100	100	100	1
Midtown South	100	100	100	100	1
Morningside Heights	100	100	100	100	1

	NeighborhoodLatitude	NeighborhoodLongitude	Venue	VenueLatitude	V
Neighborhood					
Murray Hill	200	200	200	200	2
Neponsit	23	23	23	23	2
Noho	100	100	100	100	1
North Corona	92	92	92	92	9
Oakland Gardens	57	57	57	57	5
Ozone Park	81	81	81	81	8
Pomonok	46	46	46	46	4
Queens Village	39	39	39	39	3
Queensboro Hill	53	53	53	53	5
Queensbridge	100	100	100	100	1
Ravenswood	100	100	100	100	1
Rego Park	100	100	100	100	1
Richmond Hill	61	61	61	61	6
Ridgewood	100	100	100	100	1
Rochdale	37	37	37	37	3
Rockaway Beach	83	83	83	83	8
Rockaway Park	50	50	50	50	5
Roosevelt Island	100	100	100	100	1
Rosedale	46	46	46	46	4
Roxbury	24	24	24	24	2
Soho	100	100	100	100	1
Somerville	25	25	25	25	2
South Jamaica	29	29	29	29	2
South Ozone Park	45	45	45	45	4
Springfield Gardens	31	31	31	31	3

	NeighborhoodLatitude	NeighborhoodLongitude	Venue	VenueLatitude	V
Neighborhood					
St. Albans	18	18	18	18	1
Steinway	100	100	100	100	1
Stuyvesant Town	100	100	100	100	1
Sunnyside	100	100	100	100	1
Sunnyside Gardens	100	100	100	100	1
Sutton Place	100	100	100	100	1
Tribeca	100	100	100	100	1
Tudor City	100	100	100	100	1
Turtle Bay	100	100	100	100	1
Upper East Side	100	100	100	100	1
Upper West Side	100	100	100	100	1
Utopia	70	70	70	70	7
Washington Heights	100	100	100	100	1
West Village	100	100	100	100	1
Whitestone	69	69	69	69	6
Woodhaven	70	70	70	70	7
Woodside	100	100	100	100	1
Yorkville	100	100	100	100	1

In [74]:
```
print('There are {} uniques categories.'.format(len(QM_venues['VenueCategory'].unique())))
```

```
There are 417 uniques categories.
```

Analyze neighboors

In [75]:
```
# one hot encoding
QM_onehot = pd.get_dummies(QM_venues[['VenueCategory']], prefix="", pref
ix_sep="")

#column lists before adding neighborhood
column_names = ['Neighborhood'] + list(QM_onehot.columns)

# add neighborhood column back to dataframe
QM_onehot['Neighborhood'] = QM_venues['Neighborhood']

# move neighborhood column to the first column
QM_onehot = QM_onehot[column_names]

QM_onehot.head()
```

Out[75]:

	Neighborhood	ATM	Accessories Store	Adult Boutique	Afghan Restaurant	African Restaurant	Airport Lounge	Airport Service
0	Marble Hill	0	0	0	0	0	0	0
1	Marble Hill	0	0	0	0	0	0	0
2	Marble Hill	0	0	0	0	0	0	0
3	Marble Hill	0	0	0	0	0	0	0
4	Marble Hill	0	0	0	0	0	0	0

In [76]:
```
restaurant_List = []
search = 'Restaurant'
for i in QM_onehot.columns :
    if search in i:
        restaurant_List.append(i)
```

In [77]:
```
restaurant_List
```

```
Out[77]: ['Afghan Restaurant',
          'African Restaurant',
          'American Restaurant',
          'Arepa Restaurant',
          'Argentinian Restaurant',
          'Asian Restaurant',
          'Australian Restaurant',
          'Austrian Restaurant',
          'Brazilian Restaurant',
          'Cajun / Creole Restaurant',
          'Cambodian Restaurant',
          'Cantonese Restaurant',
          'Caribbean Restaurant',
          'Caucasian Restaurant',
          'Chinese Restaurant',
          'Colombian Restaurant',
          'Comfort Food Restaurant',
          'Cuban Restaurant',
          'Czech Restaurant',
          'Dim Sum Restaurant',
          'Dumpling Restaurant',
          'Eastern European Restaurant',
          'Egyptian Restaurant',
          'Empanada Restaurant',
          'English Restaurant',
          'Ethiopian Restaurant',
          'Falafel Restaurant',
          'Fast Food Restaurant',
          'Filipino Restaurant',
          'French Restaurant',
          'German Restaurant',
          'Gluten-free Restaurant',
          'Greek Restaurant',
          'Halal Restaurant',
          'Hawaiian Restaurant',
          'Himalayan Restaurant',
          'Hotpot Restaurant',
          'Indian Restaurant',
          'Indonesian Restaurant',
          'Israeli Restaurant',
          'Italian Restaurant',
          'Japanese Curry Restaurant',
          'Japanese Restaurant',
          'Jewish Restaurant',
          'Kebab Restaurant',
          'Korean Restaurant',
          'Kosher Restaurant',
          'Latin American Restaurant',
          'Lebanese Restaurant',
          'Malay Restaurant',
          'Mediterranean Restaurant',
          'Mexican Restaurant',
          'Middle Eastern Restaurant',
          'Modern Greek Restaurant',
          'Molecular Gastronomy Restaurant',
          'Moroccan Restaurant',
          'New American Restaurant',
```

```
'Pakistani Restaurant',
'Persian Restaurant',
'Peruvian Restaurant',
'Polish Restaurant',
'Portuguese Restaurant',
'Ramen Restaurant',
'Restaurant',
'Romanian Restaurant',
'Russian Restaurant',
'Salvadoran Restaurant',
'Scandinavian Restaurant',
'Seafood Restaurant',
'Shanghai Restaurant',
'Soba Restaurant',
'South American Restaurant',
'Southern / Soul Food Restaurant',
'Spanish Restaurant',
'Sri Lankan Restaurant',
'Sushi Restaurant',
'Swiss Restaurant',
'Szechuan Restaurant',
'Taiwanese Restaurant',
'Tapas Restaurant',
'Tex-Mex Restaurant',
'Thai Restaurant',
'Tibetan Restaurant',
'Turkish Restaurant',
'Udon Restaurant',
'Ukrainian Restaurant',
'Vegetarian / Vegan Restaurant',
'Venezuelan Restaurant',
'Vietnamese Restaurant']
```

In [78]:
```
col_name = []
col_name = ['Neighborhood'] + restaurant_List
QM_restaurant = QM_onehot[col_name]
QM_restaurant = QM_restaurant.iloc[:,1::]
```

In [79]:
```
QM_restaurant_grouped = QM_restaurant.groupby('Neighborhood').sum().rese
t_index()
```

In [80]:
```
QM_restaurant_grouped['Total'] = QM_restaurant_grouped .sum(axis=1)
```

Clustering

In [81]:
```
QM_grouped_clustering = QM_restaurant_grouped.drop('Neighborhood', 1)

for n_cluster in range(2, 10):
    kmeans = KMeans(n_clusters=n_cluster).fit(QM_grouped_clustering)
    label = kmeans.labels_
    sil_coeff = silhouette_score(QM_grouped_clustering, label, metric='euclidean')
    print("For n_clusters={}, The Silhouette Coefficient is {}".format(n_cluster, sil_coeff))
```

```
For n_clusters=2, The Silhouette Coefficient is 0.46304731792043985
For n_clusters=3, The Silhouette Coefficient is 0.47072827708575693
For n_clusters=4, The Silhouette Coefficient is 0.3690149242157238
For n_clusters=5, The Silhouette Coefficient is 0.3734158182535759
For n_clusters=6, The Silhouette Coefficient is 0.3100164237399326
For n_clusters=7, The Silhouette Coefficient is 0.31237008136787714
For n_clusters=8, The Silhouette Coefficient is 0.22201169537578677
For n_clusters=9, The Silhouette Coefficient is 0.2562977809448113
```

In [82]:
```
# set number of clusters
kclusters = 2

QM_grouped_clustering = QM_restaurant_grouped.drop('Neighborhood', 1)

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

# check cluster labels generated for each row in the dataframe
kmeans.labels_
```

Out[82]:
```
array([1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1,
0, 0,
       1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0,
0, 1,
       1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1,
0, 1,
       0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1,
0, 1,
       1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0,
       0, 1, 1, 0, 0], dtype=int32)
```

In [85]:
```
QM_results = pd.DataFrame(kmeans.cluster_centers_)
QM_results.columns = QM_grouped_clustering.columns
QM_results.index = ['cluster0','cluster1']
QM_results['Total Sum'] = QM_results.sum(axis = 1)
QM_results
```

Out[85]:

	Afghan Restaurant	African Restaurant	American Restaurant	Arepa Restaurant	Argentinian Restaurant	Asian Restaurant	Aust Resta
cluster0	0.018182	9.090909e-02	2.000000	0.145455	0.218182	0.818182	1.818 01
cluster1	0.030769	1.387779e-17	0.661538	0.030769	0.061538	0.353846	2.775 17

In [86]:
```
QM_results_merged = pd.DataFrame(QM_restaurant_grouped['Neighborhood'])

QM_results_merged['Total'] = QM_restaurant_grouped['Total']
QM_results_merged = QM_results_merged.assign(Cluster_Labels = kmeans.lab
els_)
```

In [87]:
```
print(QM_results_merged.shape)
QM_results_merged
```

```
(120, 3)
```

Out[87]:

	Neighborhood	Total	Cluster_Labels
0	Arverne	4	1
1	Astoria	29	0
2	Astoria Heights	8	1
3	Auburndale	34	0
4	Battery Park City	7	1
5	Bay Terrace	9	1
6	Bayside	31	0
7	Bayswater	1	1
8	Beechhurst	10	1
9	Bellaire	11	1
10	Belle Harbor	3	1
11	Bellerose	5	1
12	Blissville	15	1
13	Breezy Point	1	1
14	Briarwood	26	0
15	Broad Channel	1	1
16	Brookville	3	1
17	Cambria Heights	6	1
18	Carnegie Hill	20	1
19	Central Harlem	35	0
20	Chelsea	21	1
21	Chinatown	28	0
22	Civic Center	24	0
23	Clinton	20	1
24	College Point	13	1
25	Corona	21	1
26	Douglaston	26	0
27	East Elmhurst	17	1
28	East Harlem	29	0
29	East Village	34	0
30	Edgemere	3	1
31	Elmhurst	62	0

	Neighborhood	Total	Cluster_Labels
32	Far Rockaway	5	1
33	Financial District	18	1
34	Flatiron	26	0
35	Floral Park	18	1
36	Flushing	53	0
37	Forest Hills	34	0
38	Forest Hills Gardens	30	0
39	Fresh Meadows	17	1
40	Glen Oaks	12	1
41	Glendale	9	1
42	Gramercy	41	0
43	Greenwich Village	25	0
44	Hamilton Heights	33	0
45	Hammels	7	1
46	Hillcrest	21	1
47	Hollis	5	1
48	Holliswood	10	1
49	Howard Beach	18	1
50	Hudson Yards	17	1
51	Hunters Point	29	0
52	Inwood	30	0
53	Jackson Heights	56	0
54	Jamaica Center	29	0
55	Jamaica Estates	15	1
56	Jamaica Hills	32	0
57	Kew Gardens	7	1
58	Kew Gardens Hills	14	1
59	Laurelton	14	1
60	Lefrak City	24	0
61	Lenox Hill	29	0
62	Lincoln Square	17	1
63	Lindenwood	8	1

	Neighborhood	Total	Cluster_Labels
64	Little Italy	22	0
65	Little Neck	27	0
66	Long Island City	19	1
67	Lower East Side	29	0
68	Malba	7	1
69	Manhattan Valley	32	0
70	Manhattanville	34	0
71	Marble Hill	13	1
72	Maspeth	7	1
73	Middle Village	7	1
74	Midtown	22	0
75	Midtown South	32	0
76	Morningside Heights	33	0
77	Murray Hill	72	0
78	Neponsit	3	1
79	Noho	41	0
80	North Corona	26	0
81	Oakland Gardens	15	1
82	Ozone Park	10	1
83	Pomonok	9	1
84	Queens Village	5	1
85	Queensboro Hill	20	1
86	Queensbridge	18	1
87	Ravenswood	36	0
88	Rego Park	25	0
89	Richmond Hill	20	1
90	Ridgewood	28	0
91	Rochdale	8	1
92	Rockaway Beach	19	1
93	Rockaway Park	6	1
94	Roosevelt Island	24	0
95	Rosedale	10	1

	Neighborhood	Total	Cluster_Labels
96	Roxbury	1	1
97	Soho	26	0
98	Somerville	2	1
99	South Jamaica	1	1
100	South Ozone Park	8	1
101	Springfield Gardens	5	1
102	St. Albans	6	1
103	Steinway	30	0
104	Stuyvesant Town	23	0
105	Sunnyside	31	0
106	Sunnyside Gardens	39	0
107	Sutton Place	31	0
108	Tribeca	26	0
109	Tudor City	39	0
110	Turtle Bay	37	0
111	Upper East Side	26	0
112	Upper West Side	34	0
113	Utopia	18	1
114	Washington Heights	35	0
115	West Village	33	0
116	Whitestone	13	1
117	Woodhaven	17	1
118	Woodside	42	0
119	Yorkville	31	0

In [88]:
```
QM_merged = QM_Geo

QM_merged = QM_merged.join(QM_results_merged.set_index('Neighborhood'), on='Neighborhood')

print(QM_merged.shape)
QM_merged.head(10) # check the last columns!
```

```
(121, 6)
```

Out[88]:

	Borough	Neighborhood	Latitude	Longitude	Total	Cluster_Labels
0	Manhattan	Marble Hill	40.876551	-73.910660	13	1
1	Manhattan	Chinatown	40.715618	-73.994279	28	0
2	Manhattan	Washington Heights	40.851903	-73.936900	35	0
3	Manhattan	Inwood	40.867684	-73.921210	30	0
4	Manhattan	Hamilton Heights	40.823604	-73.949688	33	0
5	Manhattan	Manhattanville	40.816934	-73.957385	34	0
6	Manhattan	Central Harlem	40.815976	-73.943211	35	0
7	Manhattan	East Harlem	40.792249	-73.944182	29	0
8	Manhattan	Upper East Side	40.775639	-73.960508	26	0
9	Manhattan	Yorkville	40.775930	-73.947118	31	0

Visualization of clusters

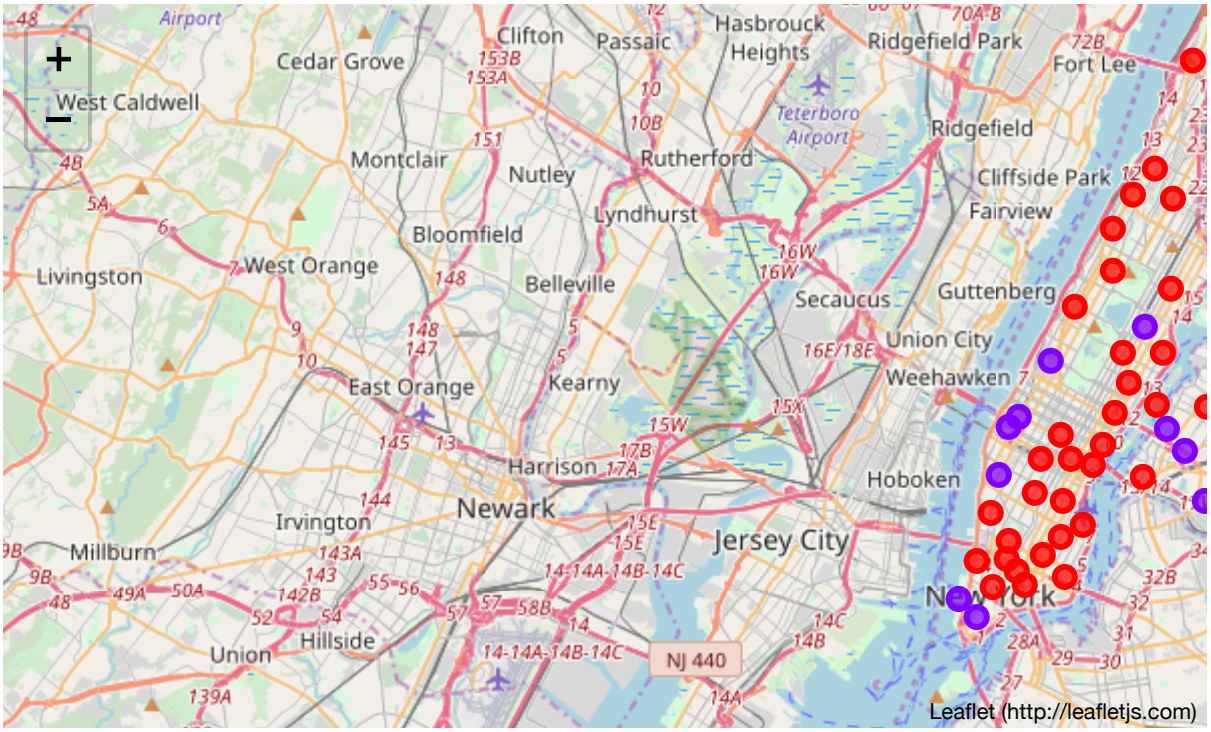
In [98]:
```
# create map
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=11)

# set color scheme for the clusters
x = np.arange(kclusters)
ys = [i+x+(i*x)**2 for i in range(kclusters)]
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(QM_merged['Latitude'], QM_merged['Long
itude'], QM_merged['Neighborhood'], QM_merged['Cluster_Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_ht
ml=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)

map_clusters
```

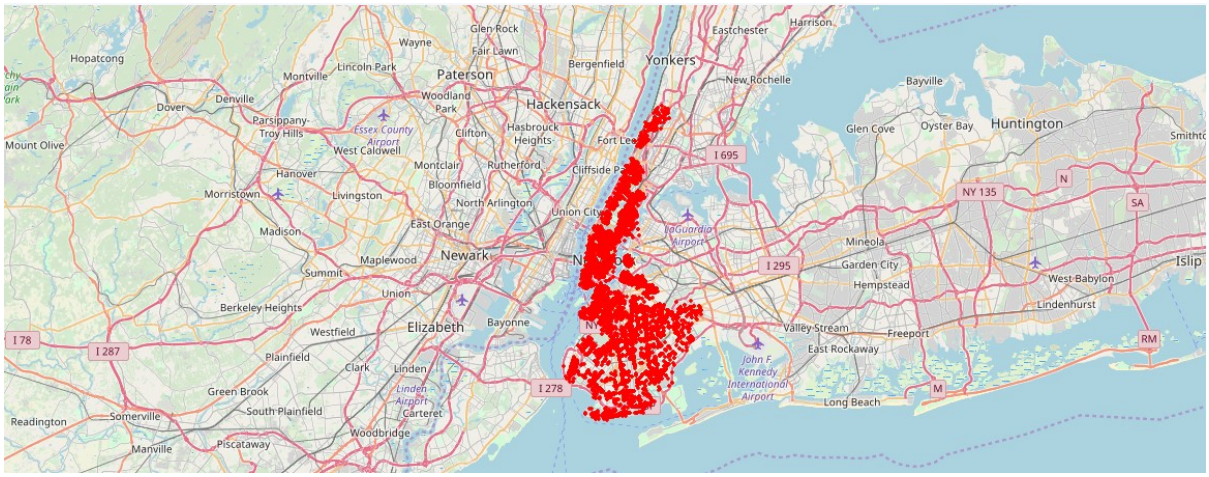
Out[98]:



Brooklyn and Manhattan Venues :

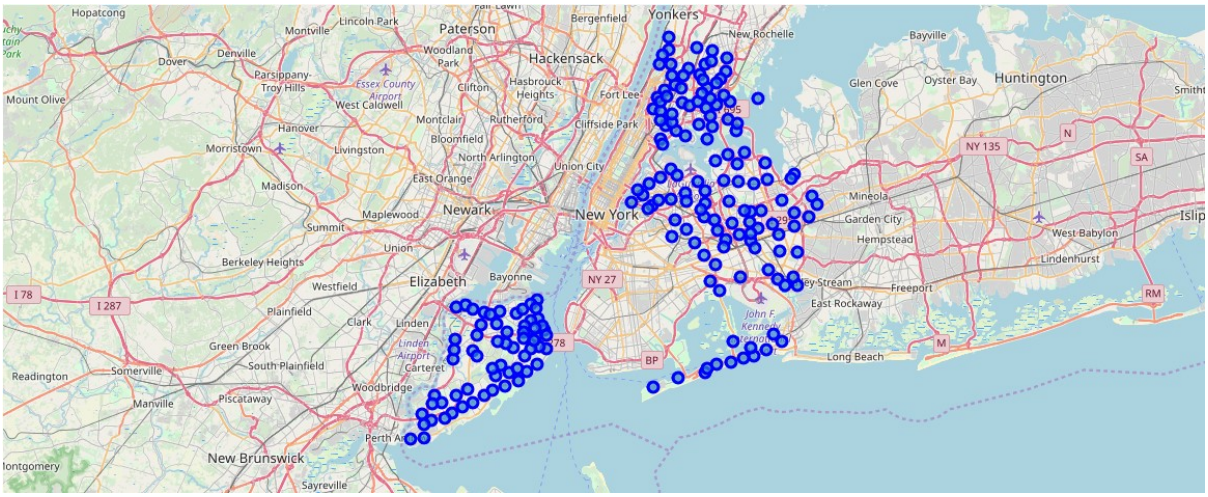
	Neighborhood	NeighborhoodLatitude	NeighborhoodLongitude	Venue	VenueLatitude	VenueLongitude	VenueCategory
0	Marble Hill	40.876551	-73.91066	Arturo's	40.874412	-73.910271	Pizza Place
1	Marble Hill	40.876551	-73.91066	Bikram Yoga	40.876844	-73.906204	Yoga Studio
2	Marble Hill	40.876551	-73.91066	Tibbett Diner	40.880404	-73.908937	Diner
3	Marble Hill	40.876551	-73.91066	Sam's Pizza	40.879435	-73.905859	Pizza Place
4	Marble Hill	40.876551	-73.91066	Loeser's Delicatessen	40.879242	-73.905471	Sandwich Place

Brooklyn and Manhattan Venues Visualization : Generated the below Brooklyn and Manhattan Venues Visualization. The "BM_venues" dataframe has 9708 venues and 397 unique venue types.



Bronx, Queens and Staten Island :

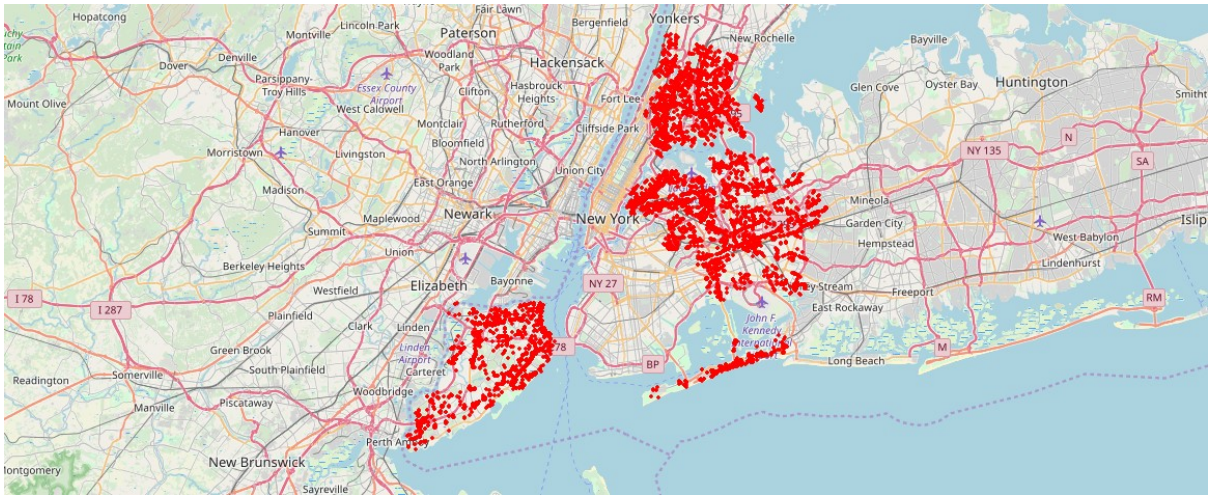
Bronx, Queens and Staten Island Neighborhoods Visualization :



Bronx, Queens and Staten Island Venues Visualization : The "BQS_venues" dataframe has 10805 venues and 387 unique venue types.

	Neighborhood	NeighborhoodLatitude	NeighborhoodLongitude	Venue	VenueLatitude	VenueLongitude	VenueCategory
0	Wakefield	40.894705	-73.847201	Lollipops Gelato	40.894123	-73.845892	Dessert Shop
1	Wakefield	40.894705	-73.847201	Ripe Kitchen & Bar	40.898152	-73.838875	Caribbean Restaurant
2	Wakefield	40.894705	-73.847201	Jackie's West Indian Bakery	40.889283	-73.843310	Caribbean Restaurant
3	Wakefield	40.894705	-73.847201	Ali's Roti Shop	40.894036	-73.856935	Caribbean Restaurant
4	Wakefield	40.894705	-73.847201	Rite Aid	40.896521	-73.844680	Pharmacy

Bronx, Queens and Staten Island Venues Map Visualization :



4.RESULTS :

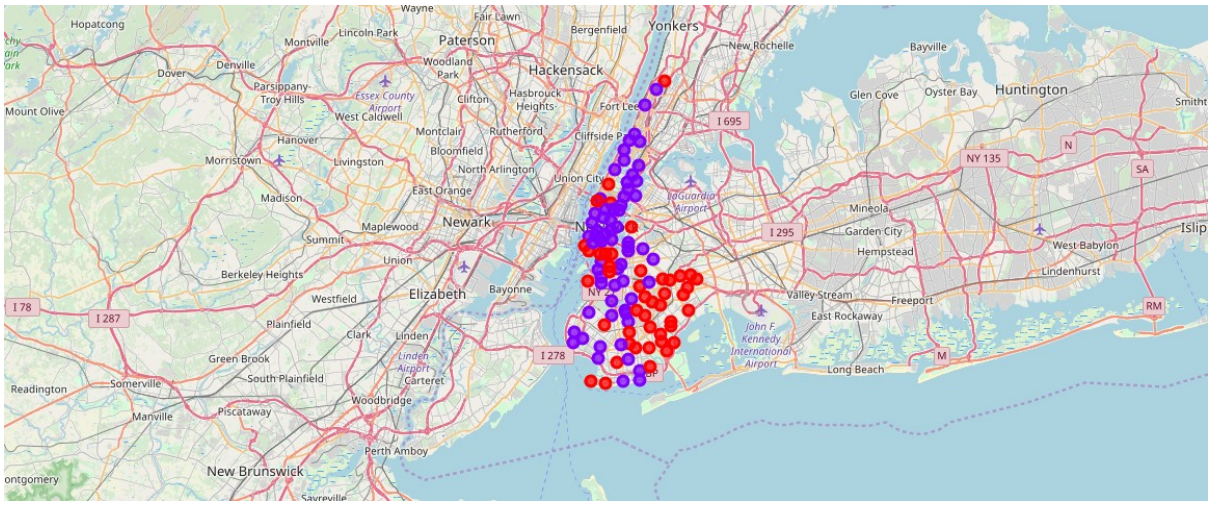
From this venues data we filtered and used only the restaurant data for Brooklyn & Manhattan clustering and Bronx, Queens and Staten Island clustering. As we focussed only on restaurants business.

Neighborhood K-Means clustering based on mean occurrence of venue category :

To cluster the neighborhoods into two clusters we used the K-Means clustering Algorithm. *k*-means clustering aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean. It uses iterative refinement approach.

Brooklyn & Manhattan :

In the below Map Visualization, we can see the different types of clusters created by using K-Means for Brooklyn & Manhattan.



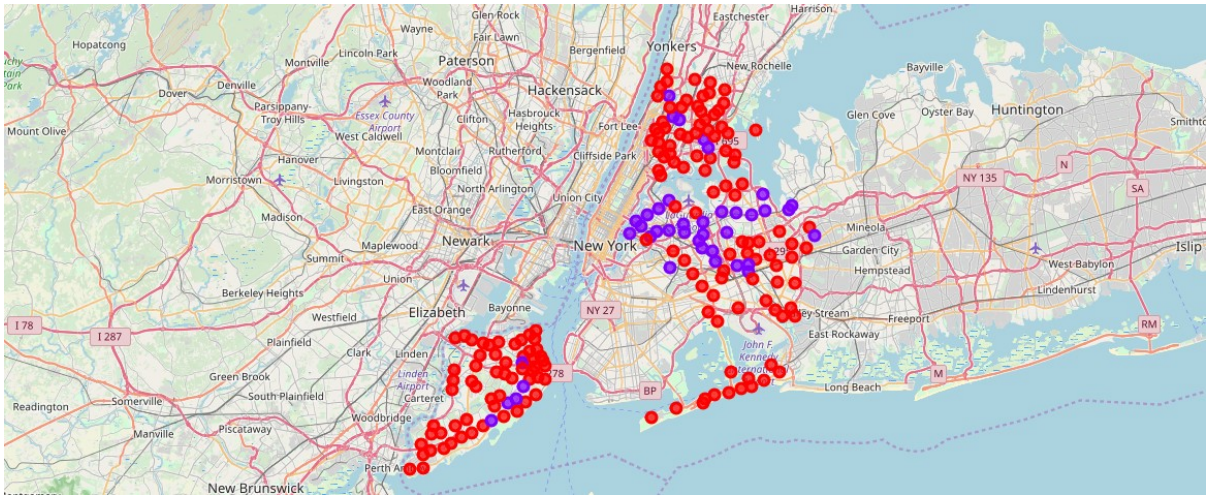
Cluster0 : The Total and Total Sum of cluster0 has smallest value. It shows that the market is not saturated.

Cluster1 : The Total and Total Sum of cluster1 has highest value. It shows that the markets are saturated. Number of restaurants are very high.

There are no untapped neighborhoods in Brooklyn and Manhattan.

Bronx, Queens and Staten Island :

In the below Map Visualization, we can see the different types of clusters created by using K-Means for Bronx, Queens and Staten Island.



Cluster0 : The Total and Total Sum of cluster0 has smallest value. It shows that the market is not saturated. There are untapped neighborhoods. List is as given below.

	Borough	Neighborhood	Latitude	Longitude	Total	Cluster_Labels
0	Staten Island	Todt Hill	40.597069	-74.111329	0	0
1	Staten Island	Port Ivory	40.639683	-74.174645	0	0
2	Staten Island	Bloomfield	40.605779	-74.187256	0	0

Cluster1 : The Total and Total Sum of cluster1 has highest value. It shows that the markets are saturated. Number of restaurants are very high.

5.DISCUSSION:

1. There is scope to increase Farmers markets in Bronx, Queens and Staten Island.
2. There is scope to explore cuisines of various countries in Bronx, Queens and Staten Island.
3. In Manhattan and Brooklyn restaurants of cuisines of many countries are available. So if risk can be taken with great menu on board. It also shows people love eating cuisines of various countries.

6.CONCLUSION:

This analysis is performed on limited data. This may be right or may be wrong. But if good amount of data is available there is scope to come up with better results. If there are lot of restaurants probably there is lot of demand. Brooklyn and Manhattan has high concentration of restaurant business. Very competitive market. Bronx, Queens and Staten Island also has good number of restaurants but not as many as required. So this can be explored.

As per the neighbourhood or restaurant type mentioned like Indian Restaurant analysis can be checked. A venue with lowest risk and competition can be identified.

.