Coursera Capstone Project

The Battle of Neighborhoods - Final Report

Upload Libraries Required

In [2]:

```
import numpy as np # library to handle data in a vectorized manner
import time
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
import requests # library to handle requests
from pandas.io.json import json_normalize # tranform JSON file into a pandas dataframe


#!conda install -c conda-forge geopy --yes # uncomment this line if you haven't complet
ed the Foursquare API lab
from geopy.geocoders import Nominatim # convert an address into latitude and longitude
 values

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

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

import seaborn as sns

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



print('Libraries imported.')
```

```
Libraries imported.
```

Coursera Capstone - REPORT

Content

1. Introduction Section :

   1.1 Discussion of the "backgroung situation" leading to the problem at hand:

   1.2 Problem to be resolved

   1.3 Audience for this project.

1. Data Section:

   2.1 Data of Current Situation (current residence place)

   2.2 Data required to resolve the problem

   2.3 Data sources and data manipulation

1. Methodology section :

   3.1 Process steps and strategy to resolve the problem

   3.2 Data Science Methods, machine learing, mapping tools and exploratory data analysis.

1. Results section

   Discussion of the results and how they help to take a decision.

1. Discussion section

   Elaboration and discussion on any observations and/or recommendations for improvement.

1. Conclusion section

   Desicison taken and Report Conclusion.

1. Introduction Section :

Discussion of the business problem and the audience who would be interested in this project.

1.1 Scenario and Background

I am a data scientist currently residing in Downtown Singapore. I currently live within walking distance to Downtown "Telok Ayer MRT metro station" therefore I have access to good public transportation to work. Likewise, I enjoy many ammenities in the neighborhood , such as international cousine restaurants, cafes, food shops and entertainment. I have been offered a great opportunity to work in Manhattan, NY. Although, I am very excited about it, I am a bit stress toward the process to secure a comparable place to live in Manhattan. Therefore, I decided to apply the learned skills during the Coursera course to explore ways to make sure my decision is factual and rewarding. Of course, there are alternatives to achieve the answer using available Google and Social media tools, but it rewarding doing it myself with learned tools.

1.2 Problem to be resolved:

The challenge to resolve is being able to find a rental apartment unit in Manhattan NY that offers similar characteristics and benefits to my current situation. Therefore, in order to set a basis for comparison, I want to find a renta unit subject to the following conditions:

- Apartment with min 2 bedrooms with monthly rent not to exceed US$7000/month
- Unit located within walking distance (<=1.0 mile, 1.6 km) from a subway metro station in Manhattan
- Area with ammenities and venues similar to the ones described for current location (See item 2.1)

1.3 Interested Audience

I believe this is a relevant project for a person or entity considering moving to a major city in Europe, US or Asia, since the approach and methodologies used here are applicable in all cases. The use of FourSquare data and mapping techniques combined with data analysis will help resolve the key questions arisen. Lastly, this project is a good practical case toward the development of Data Science skills.

2. Data Section:

Description of the data and its sources that will be used to solve the problem

2.1 Data of Current Situation

I Currently reside in the neighborhood of 'Mccallum Street' in Downtonw Singapore. I use Foursquare to identify the venues around the area of residence which are then shown in the Singapore map shown in methodology and execution in section 3.0 . It serves as a reference for comparison with the desired future location in Manhattan NY

2.2 Data Required to resolve the problem

In order to make a good choice of a similar apartment in Manhattan NY, the following data is required: List/Information on neighborhoods form Manhattan with their Geodata (latitud and longitud. List/Information about the subway metro stations in Manhattan with geodata. Listed apartments for rent in Manhattan area with descriptions (how many beds, price, location, address) Venues and ammenities in the Manhattan neighborhoods (e.g. top 10) 2.3 sources and manipulation The list of Manhattan neighborhoods is worked out during LAb exercise during the course. A csv file was created which will be read in order to create a dataframe and its mapping. The csv file 'mh_neigh_data.csv' has the following below data structure. The file will be directly read to the Jupiter Notebook for convenience and space savings. The clustering of neighborhoods and mapping will be shown however. An algorythm was used to determine the geodata from Nominatim . The actual algorythm coding may be shown in 'markdown' mode becasues it takes time to run.

mh_neigh_data.tail():

	Borough	Neighborhood	Latitude	Longitude
35	Manhattan	Turtle Bay	40.752042	-73.967708
36	Manhattan	Tudor City	40.746917	-73.971219
37	Manhattan	Stuyvesant Town	40.731000	-73.974052
38	Manhattan	Flatiron	40.739673	-73.990947
39	Manhattan	Hudson Yards	40.756658	-74.000111

A list of Manhattan subway metro stops was complied in Numbers (Apple excel) and it was complemeted with wikipedia data (https://en.wikipedia.org/wiki/List_of_New_York_City_Subway_stations_in_Manhattan (https://en.wikipedia.org/wiki/List_of_New_York_City_Subway_stations_in_Manhattan)) and information from NY Transit authority and Google maps (https://www.google.com/maps/search/manhattan+subway+metro+stations/@40.7837297,-74.1033043,11z/dat (https://www.google.com/maps/search/manhattan+subway+metro+stations/@40.7837297,-74.1033043,11z/dat for a final consolidated list of subway stops names and their address. The geolocation was obtained via an algorythm using Nominatim. Details will be shown in the execution of methodolody in section 3.0. The subway csv file is "MH_subway.csv"' and the data structure is: mhsub.tail(): sub_station sub_address lat long

17	190 Street Subway Station	Bennett Ave, New York, NY 10040, USA	40.858113	-73.932983
18	59 St-Lexington Av Station	E 60th St, New York, NY 10065, USA	40.762259	-73.966271
19	57 Street Station	New York, NY 10019, United States	40.764250	-73.954525
20	14 Street / 8 Av	New York, NY 10014, United States	40.730862	-73.987156
21	MTA New York City	525 11th Ave, New York, NY 10018, USA	40.759809	-73.999282

A list of places for rent was collected by web-browsing real estate companies in Manhattan : http://www.rentmanhattan.com/index.cfm?page=search&state=results (http://www.rentmanhattan.com/index.cfm?page=search&state=results) https://www.nestpick.com/search?city=new-york&page=1&order=relevance&district=manhattan&gclid=CjwKCAiAjNjgBRAgEiwAGLlf2hkP3A-cPxjZYkURqQEswQK2jKQEpv_MvKcrIhRWRzNkc_r-fGi0lxoCA7cQAvD_BwE&type=apartment&display=list (https://www.nestpick.com/search?city=new-york&page=1&order=relevance&district=manhattan&gclid=CjwKCAiAjNjgBRAgEiwAGLlf2hkP3A-cPxjZYkURqQEswQK2jKQEpv_MvKcrIhRWRzNkc_r-fGi0lxoCA7cQAvD_BwE&type=apartment&display=list) https://www.realtor.com/apartments/Manhattan_NY (https://www.realtor.com/apartments/Manhattan_NY) A csv file was compiled with the rental place that indicated: areas of Manhattan, address, number of beds, area and monthly rental price. The csv file "nnnn.csv" had the following below structure. An algorythm was used to create all the geodata using Nominatim, as shown in section 3.0. The actual algorythm coding may be shown in 'markdown' mode becasues it takes time to run. With the use of geolocator = Nominatim() , it was possible to determine the latitude and longiude for the subway metro locations as well as for the geodata for each rental place listed. The loop algorythms used are shown in the execution of data in section 3.0 "Great_circle" function from geolocator was used to calculate distances between two points , as in the case to calculate average rent price for units around each subway station and at 1.6 km radius. Foursquare is used to find the avenues at Manhattan neighborhoods in general and a cluster is created to later be able to search for the venues depending of the location shown.

2.4 How the data will be used to solve the problem

The data will be used as follows: Use Foursquare and geopy data to map top 10 venues for all Manhattan neighborhoods and clustered in groups (as per Course LAB) Use foursquare and geopy data to map the location of subway metro stations , separately and on top of the above clustered map in order to be able to identify the venues and ammenities near each metro station, or explore each subway location separately Use Foursquare and geopy data to map the location of rental places, in some form, linked to the subway locations. create a map that depicts, for instance, the average rental price per square ft, around a radious of 1.0 mile (1.6 km) around each subway station - or a similar metrics. I will be able to quickly point to the popups to know the relative price per subway area. Addresses from rental locations will be converted to geodata(lat, long) using Geopy-distance and Nominatim. Data will be searched in open data sources if available, from real estate sites if open to reading, libraries or other government agencies such as Metro New York MTA, etc.

2.5 Mapping of Data

The following maps were created to facilitate the analysis and the choice of the palace to live. Manhattan map of Neighborhoods manhattan subway metro locations Manhattan map of places for rent Manhattan map of clustered venues and neighborhoods Combined maps of Manhattan rent places with subway locations Combined maps of Manhattan rent places with subway locations and venues clusters

3. Methodology section:

This section represents the main component of the report where the data is gathered, prepared for analysis. The tools described are used here and the Notebook cells indicates the execution of steps.

The analysis and the stragegy:

The strategy is based on mapping the above described data in section 2.0, in order to facilitate the choice of at least two candidate places for rent. The choice is made based on the demands imposed : location near a subway, rental price and similar venues to Singapore. This visual approach and maps with popups labels allow quick identification of location, price and feature, thus making the selection very easy.

The procesing of these DATA and its mapping will allow to answer the key questions to make a decision:

- what is the cost of available rental places that meet the demands?
- what is the cost of rent around a mile radius from each subway metro station?
- what is the area of Manhattan with best rental pricing that meets criteria established?
- What is the distance from work place (Park Ave and 53 rd St) and the tentative future rental home?
- What are the venues of the two best places to live? How the prices compare?
- How venues distribute among Manhattan neighborhoods and around metro stations?
- Are there tradeoffs between size and price and location?
- Any other interesting statistical data findings of the real estate and overall data.

METHODOLOY EXECUTION - Mapping Data

Singapore Map - Current residence and venues in neighborhood

for comparison to future Manhattan renting place

In [3]:

```
# Shenton Way, District 01, Singapore
address = 'Mccallum Street, Singapore'
geolocator = Nominatim()
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of Singapore home are {}, {}.'.format(latitude, longi
tude))
```

```
C:\Users\ngoctrungduy.nguyen\AppData\Local\Continuum\anaconda3\lib\site-pa
ckages\ipykernel\__main__.py:3: DeprecationWarning: Using Nominatim with t
he default "geopy/1.18.1" `user_agent` is strongly discouraged, as it viol
ates Nominatim's ToS https://operations.osmfoundation.org/policies/nominat
im/ and may possibly cause 403 and 429 HTTP errors. Please specify a custo
m `user_agent` with `Nominatim(user_agent="my-application")` or by overrid
ing the default `user_agent`: `geopy.geocoders.options.default_user_agent
= "my-application"`. In geopy 2.0 this will become an exception.
  app.launch_new_instance()
```

```
The geograpical coordinate of Singapore home are 1.2795465, 103.8476049.
```

In [4]:

```
neighborhood_latitude=1.2792655
neighborhood_longitude=103.8480938
```

Dial FourSquare to find venues around current residence in Singapore

In [11]:

```
# @hidden_cell
CLIENT_ID = 'EQGFF1SZ4EAE4JZ0VVSGYL2KQ4RF1NL5TZBPWICGIMQVYULC' # your Foursquare ID
CLIENT_SECRET = 'HZRXOSCKH0LFZGJJWQWAD3GYDBQIVGFQEB04WYEZ2OG3DBGB' # your Foursquare Se
cret
VERSION = '20180405' # Foursquare API version

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

In [12]:

```
LIMIT = 100 # limit of number of venues returned by Foursquare API
radius = 500 # define radius
# create URL
url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}
&ll={},{}&radius={}&limit={}'.format(
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    neighborhood_latitude,
    neighborhood_longitude,
    radius,
    LIMIT)
url # display URL
```

Out[12]:

```
'https://api.foursquare.com/v2/venues/explore?&client_id=EQGFF1SZ4EAE4JZ0V
VSGYL2KQ4RF1NL5TZBPWICGIMQVYULC&client_secret=HZRXOSCKH0LFZGJJWQWAD3GYDBQI
VGFQEB04WYEZ2OG3DBGB&v=20180405&ll=1.2792655,103.8480938&radius=500&limit=
100'
```

In [13]:

```
# results display is hidden for report simplification
results = requests.get(url).json()
#results
```

function that extracts the category of the venue - borrow from the Foursquare lab.

In [14]:

```
def get_category_type(row):
    try:
        categories_list = row['categories']
    except:
        categories_list = row['venue.categories']

    if len(categories_list) == 0:
        return None
    else:
        return categories_list[0]['name']
```

In [15]:

```
venues = results['response']['groups'][0]['items']
SGnearby_venues = json_normalize(venues) # flatten JSON
# filter columns
filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat', 'venue.loca
tion.lng']
SGnearby_venues =SGnearby_venues.loc[:, filtered_columns]
# filter the category for each row
SGnearby_venues['venue.categories'] = SGnearby_venues.apply(get_category_type, axis=1)
# clean columns
SGnearby_venues.columns = [col.split(".")[-1] for col in SGnearby_venues.columns]

SGnearby_venues.shape
```

Out[15]:

```
(100, 4)
```

In [16]:

```
# Venues near current Singapore residence place
SGnearby_venues.head(10)
```

Out[16]:

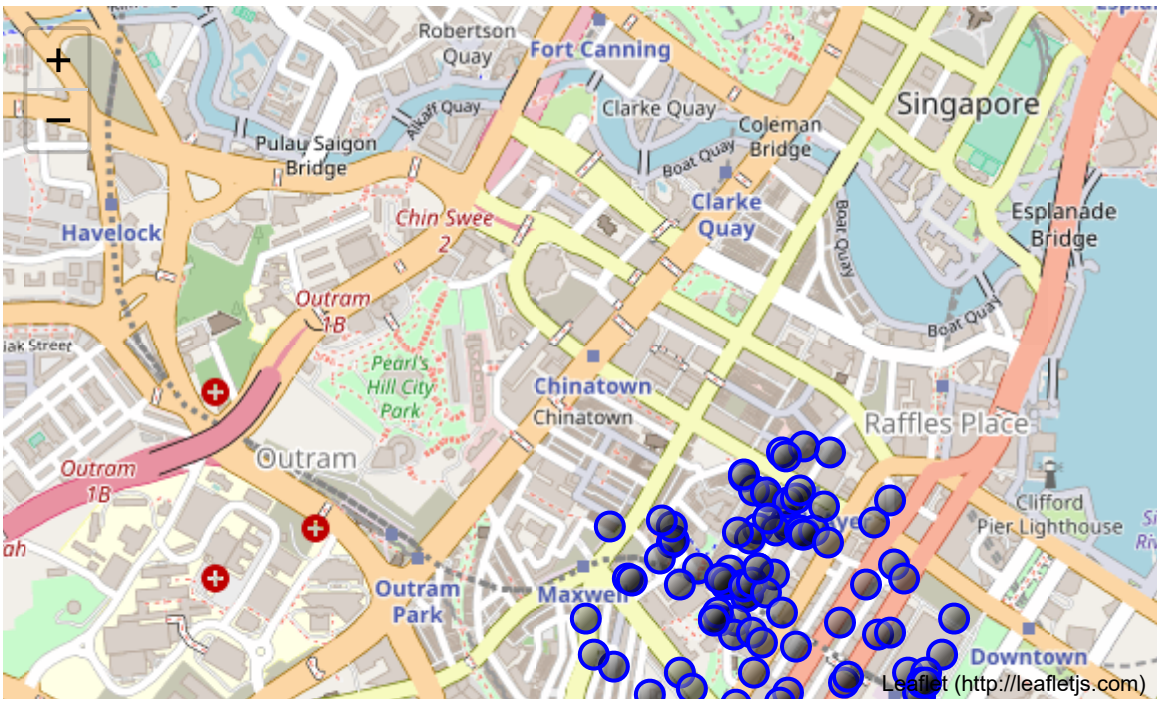
	name	categories	lat	lng
0	Napoleon Food & Wine Bar	Wine Bar	1.279925	103.847333
1	Park Bench Deli	Deli / Bodega	1.279872	103.847287
2	Native	Cocktail Bar	1.280135	103.846844
3	Freehouse	Beer Garden	1.281254	103.848513
4	PS.Cafe	Café	1.280468	103.846264
5	Matt's | The Chocolate Shop	Dessert Shop	1.280462	103.846950
6	Pepper Bowl	Asian Restaurant	1.279371	103.846710
7	왕대박 Wang Dae Bak Korean BBQ Restaurant	Korean Restaurant	1.281345	103.847551
8	Amoy Street Food Centre	Food Court	1.279368	103.847079
9	Sofitel So Singapore	Hotel	1.280124	103.849867

Map of Singapore residence place with venues in Neighborhood - for reference

In [17]:

```
latitude=1.2792655
longitude=103.8480938
# create map of Singapore place  using latitude and longitude values
map_sg = folium.Map(location=[latitude, longitude], zoom_start=18)
# add markers to map
for lat, lng, label in zip(SGnearby_venues['lat'], SGnearby_venues['lng'], SGnearby_ven
ues['name']):
    label = folium.Popup(label, parse_html=True)
    folium.RegularPolygonMarker(
        [lat, lng],
        number_of_sides=30,
        radius=7,
        popup=label,
        color='blue',
        fill_color='#0f0f0f',
        fill_opacity=0.6,
    ).add_to(map_sg)

map_sg
```

Out[17]:



MANHATTAN NEIGHBORHOODS - DATA AND MAPPING

Cluster neighborhood data was produced with Foursquare during course lab work. A csv file was produced containing the neighborhoods around the 40 Boroughs. Now, the csv file is just read for convenience and consolidation of report.

In [22]:

```
# Read csv file with clustered neighborhoods with geodata
manhattan_data  = pd.read_csv('mh_neigh_data.csv')
manhattan_data.head()
```

Out[22]:

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

In [23]:

```
manhattan_data.tail()
```

Out[23]:

	Borough	Neighborhood	Latitude	Longitude	Cluster Labels
35	Manhattan	Turtle Bay	40.752042	-73.967708	3
36	Manhattan	Tudor City	40.746917	-73.971219	3
37	Manhattan	Stuyvesant Town	40.731000	-73.974052	4
38	Manhattan	Flatiron	40.739673	-73.990947	3
39	Manhattan	Hudson Yards	40.756658	-74.000111	2

Manhattan Borough neighborhoods - data with top 10 clustered venues

In [25]:

```
manhattan_merged = pd.read_csv('manhattan_merged.csv')
manhattan_merged.head()
```

Out[25]:

	Borough	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3 C
0	Manhattan	Marble Hill	40.876551	-73.910660	2	Coffee Shop	Discount Store	Yo Stu
1	Manhattan	Chinatown	40.715618	-73.994279	2	Chinese Restaurant	Cocktail Bar	Dir Re
2	Manhattan	Washington Heights	40.851903	-73.936900	4	Café	Bakery	Mc Ph Sh
3	Manhattan	Inwood	40.867684	-73.921210	3	Mexican Restaurant	Lounge	Piz Pla
4	Manhattan	Hamilton Heights	40.823604	-73.949688	0	Mexican Restaurant	Coffee Shop	Ca

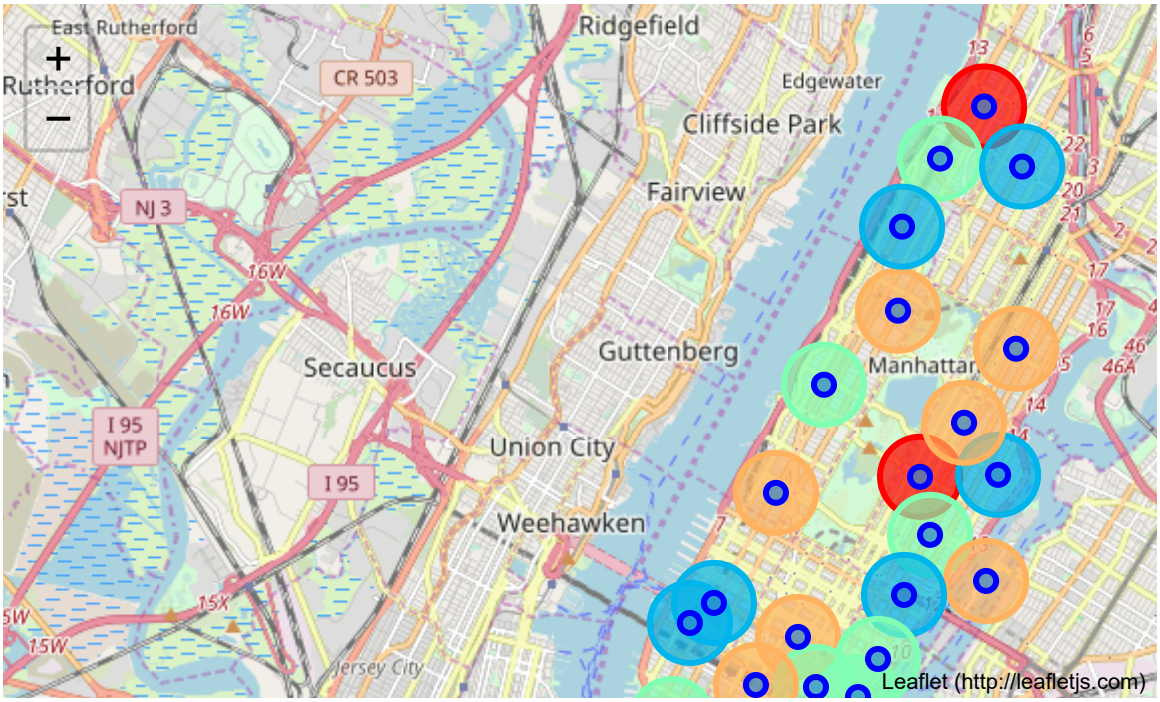
Map of Manhattan neighborhoods with top 10 clustered venues

popus allow to identify each neighborhood and the cluster of venues around it in order to proceed to examine in more detail in the next cell

In [27]:

```
# create map of Manhattan using latitude and longitude values from Nominatim
latitude= 40.7308619
longitude= -73.9871558

kclusters=5
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=13)

# 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(manhattan_merged['Latitude'], manhattan_merged['Longi
tude'], manhattan_merged['Neighborhood'], manhattan_merged['Cluster Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=20,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)
  # add markers for rental places to map
for lat, lng, label in zip(manhattan_data['Latitude'], manhattan_data['Longitude'], man
hattan_data['Neighborhood']):
    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_clusters)


map_clusters
```

Out[27]:



Examine a paticular Cluster - print venues

After examining several cluster data , I concluded that cluster # 2 resembles closer the Singapore place, therefore providing guidance as to where to look for the future apartment .

Assign a value to 'kk' to explore a given cluster.

In [28]:

```
## kk is the cluster number to explore
kk = 2
manhattan_merged.loc[manhattan_merged['Cluster Labels'] == kk, manhattan_merged.columns
[[1] + list(range(5, manhattan_merged.shape[1]))]]
```

Out[28]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th M Comm Ve
0	Marble Hill	Coffee Shop	Discount Store	Yoga Studio	Steakhouse	Supplement Shop	Tennis Stadiun
1	Chinatown	Chinese Restaurant	Cocktail Bar	Dim Sum Restaurant	American Restaurant	Vietnamese Restaurant	Salon / Barbers
6	Central Harlem	African Restaurant	Seafood Restaurant	French Restaurant	American Restaurant	Cosmetics Shop	Chinese Restaur
9	Yorkville	Coffee Shop	Gym	Bar	Italian Restaurant	Sushi Restaurant	Pizza Place
14	Clinton	Theater	Italian Restaurant	Coffee Shop	American Restaurant	Gym / Fitness Center	Hotel
23	Soho	Clothing Store	Boutique	Women's Store	Shoe Store	Men's Store	Furnitur Home Store
26	Morningside Heights	Coffee Shop	American Restaurant	Park	Bookstore	Pizza Place	Sandwi Place
34	Sutton Place	Gym / Fitness Center	Italian Restaurant	Furniture / Home Store	Indian Restaurant	Dessert Shop	Americ Restaur
39	Hudson Yards	Coffee Shop	Italian Restaurant	Hotel	Theater	American Restaurant	Café

Map of Manhattan places for rent

Several Manhattan real estate webs were webscrapped to collect rental data, as mentioned in section 2.0 . The resut was summarized in a csv file for direct reading, in order to consolidate the proces.

The initial data for 144 apartment did not have the latitude and longitude data (NaN) but the information was established in the following cell using an algorythm and Nominatim.

In [31]:

```
# csv files with rental places with basic data but still wihtout geodata ( latitude and
longitude)
# pd.read_csv(' le.csv', header=None, nrows=5)
mh_rent=pd.read_csv('MH_flats_price.csv')
mh_rent.head()
```

Out[31]:

	Address	Area	Price_per_ft2	Rooms	Area-ft2	Rent_Price	Lat	Long
0	West 105th Street	Upper West Side	2.94	5.0	3400	10000	NaN	NaN
1	East 97th Street	Upper East Side	3.57	3.0	2100	7500	NaN	NaN
2	West 105th Street	Upper West Side	1.89	4.0	2800	5300	NaN	NaN
3	CARMINE ST.	West Village	3.03	2.0	1650	5000	NaN	NaN
4	171 W 23RD ST.	Chelsea	3.45	2.0	1450	5000	NaN	NaN

In [32]:

```
mh_rent.tail()
```

Out[32]:

	Address	Area	Price_per_ft2	Rooms	Area-ft2	Rent_Price	Lat	Long
139	200 East 72nd Street	Rental in Lenox Hill	5.15	3.0	1700	8750	NaN	NaN
140	50 Murray Street	No fee rental in Tribeca	7.11	2.0	1223	8700	NaN	NaN
141	300 East 56th Street	No fee rental in Midtown East	3.87	3.0	2100	8118	NaN	NaN
142	1930 Broadway	No fee rental in Central Park West	5.06	2.0	1600	8095	NaN	NaN
143	33 West 9th Street	Rental in Greenwich Village	6.67	2.0	1500	10000	NaN	NaN

Obtain geodata (lat,long) for each rental place in Manhattan with Nominatim

Data was stored in a csv file for simplifaction report purposes and saving code processing time in future.

This coding section was 'markedown' for the report because its execution takes few minutes . Terefore, the csv saved will be be just read directly in the following cell.

for n in range(len(mh_rent)):
address= mh_rent['Address'][n] address=(mh_rent['Address'][n]+ ' , '+' Manhattan NY ') geolocator = Nominatim() location = geolocator.geocode(address) latitude = location.latitude longitude = location.longitude mh_rent['Lat'][n]=latitude mh_rent['Long'][n]=longitude

```
    #print(n,latitude,longitude)
    time.sleep(2)
```

print('Geodata completed')

save dataframe to csv file

mh_rent.to_csv('MH_rent_latlong.csv',index=False) mh_rent.shape

In [33]:

```
mh_rent=pd.read_csv('MH_rent_latlong.csv')
mh_rent.head()
```

Out[33]:

	Address	Area	Price_per_ft2	Rooms	Area-ft2	Rent_Price	Lat	Lo
0	West 105th Street	Upper West Side	2.94	5.0	3400	10000	40.799771	-73.966
1	East 97th Street	Upper East Side	3.57	3.0	2100	7500	40.788585	-73.955
2	West 105th Street	Upper West Side	1.89	4.0	2800	5300	40.799771	-73.966
3	CARMINE ST.	West Village	3.03	2.0	1650	5000	40.730523	-74.001
4	171 W 23RD ST.	Chelsea	3.45	2.0	1450	5000	40.744118	-73.995

In [34]:

```
mh_rent.tail()
```

Out[34]:

	Address	Area	Price_per_ft2	Rooms	Area-ft2	Rent_Price	Lat	
139	200 East 72nd Street	Rental in Lenox Hill	5.15	3.0	1700	8750	40.769465	-73.96
140	50 Murray Street	No fee rental in Tribeca	7.11	2.0	1223	8700	40.714051	-74.00
141	300 East 56th Street	No fee rental in Midtown East	3.87	3.0	2100	8118	40.758216	-73.96
142	1930 Broadway	No fee rental in Central Park West	5.06	2.0	1600	8095	40.772474	-73.98
143	33 West 9th Street	Rental in Greenwich Village	6.67	2.0	1500	10000	40.733691	-73.99

Manhattan apartment rent price statistics

A US 7000 Dollar per month rent is actually around the mean value - similar to Singapore! wow!

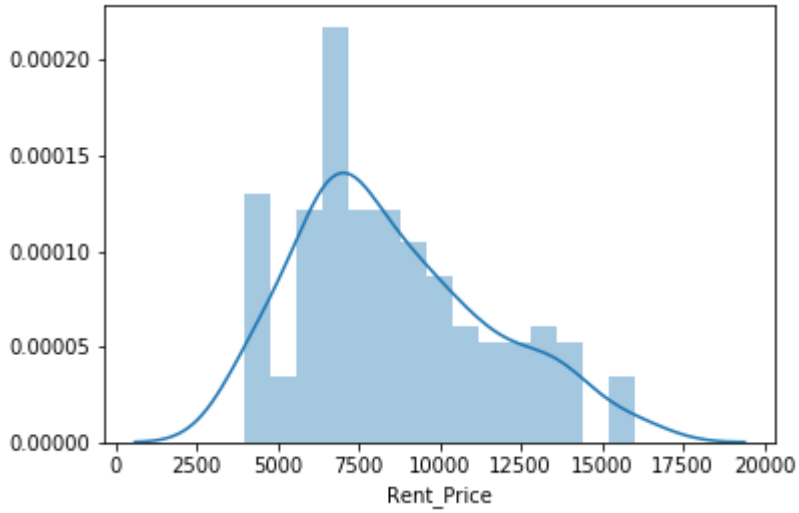
In [35]:

```
import seaborn as sns
sns.distplot(mh_rent['Rent_Price'],bins=15)
```

```
C:\Users\ngoctrungduy.nguyen\AppData\Local\Continuum\anaconda3\lib\site-pa
ckages\scipy\stats\stats.py:1713: FutureWarning: Using a non-tuple sequenc
e for multidimensional indexing is deprecated; use `arr[tuple(seq)]` inste
ad of `arr[seq]`. In the future this will be interpreted as an array inde
x, `arr[np.array(seq)]`, which will result either in an error or a differe
nt result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[35]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1edd6ef34a8>
```



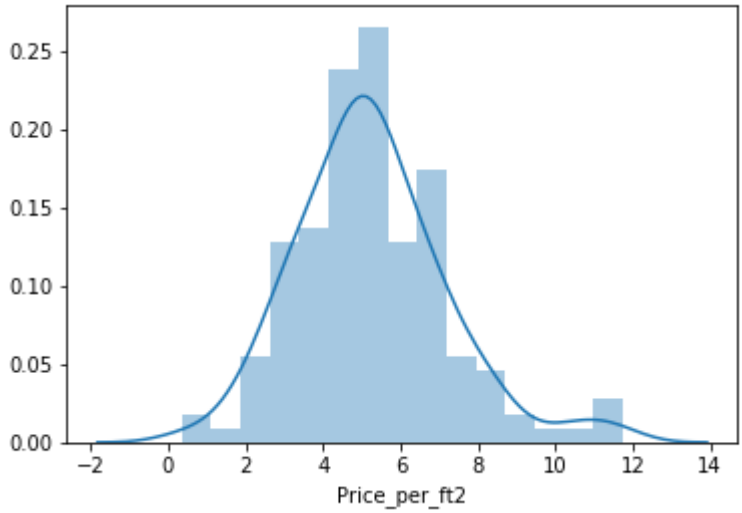
In [36]:

```
import seaborn as sns
sns.distplot(mh_rent['Price_per_ft2'],bins=15)
```

```
C:\Users\ngoctrungduy.nguyen\AppData\Local\Continuum\anaconda3\lib\site-pa
ckages\scipy\stats\stats.py:1713: FutureWarning: Using a non-tuple sequenc
e for multidimensional indexing is deprecated; use `arr[tuple(seq)]` inste
ad of `arr[seq]`. In the future this will be interpreted as an array inde
x, `arr[np.array(seq)]`, which will result either in an error or a differe
nt result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[36]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1edd6fb2e48>
```

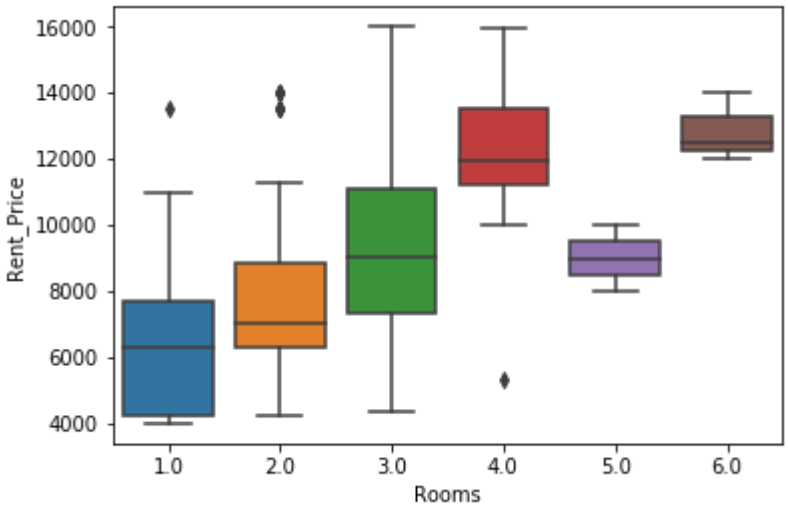


In [37]:

```
sns.boxplot(x='Rooms', y= 'Rent_Price', data=mh_rent)
```

Out[37]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1edd74e59e8>
```



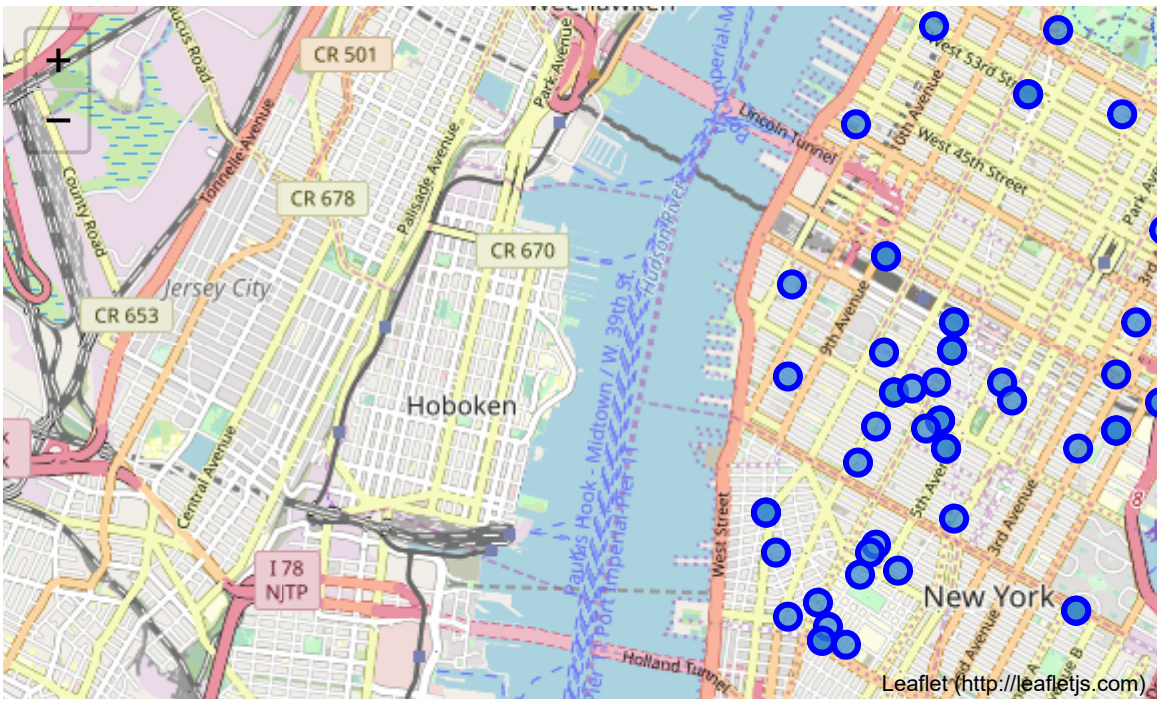
Map of Manhattan apartments for rent

The popups will indicate the address and the monthly price for rent thus making it convenient to select the target appartment with the price condition estipulated (max US7000)

In [38]:

```
# create map of Manhattan using latitude and longitude values from Nominatim
latitude= 40.7308619
longitude= -73.9871558

map_manhattan_rent = folium.Map(location=[latitude, longitude], zoom_start=12.5)

# add markers to map
for lat, lng, label in zip(mh_rent['Lat'], mh_rent['Long'],'$ ' + mh_rent['Rent_Price']
.astype(str)+ ',  '+ mh_rent['Address']):
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=6,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(map_manhattan_rent)


map_manhattan_rent
```

Out[38]:



Map of Manhattan showing the places for rent and the cluster of venues

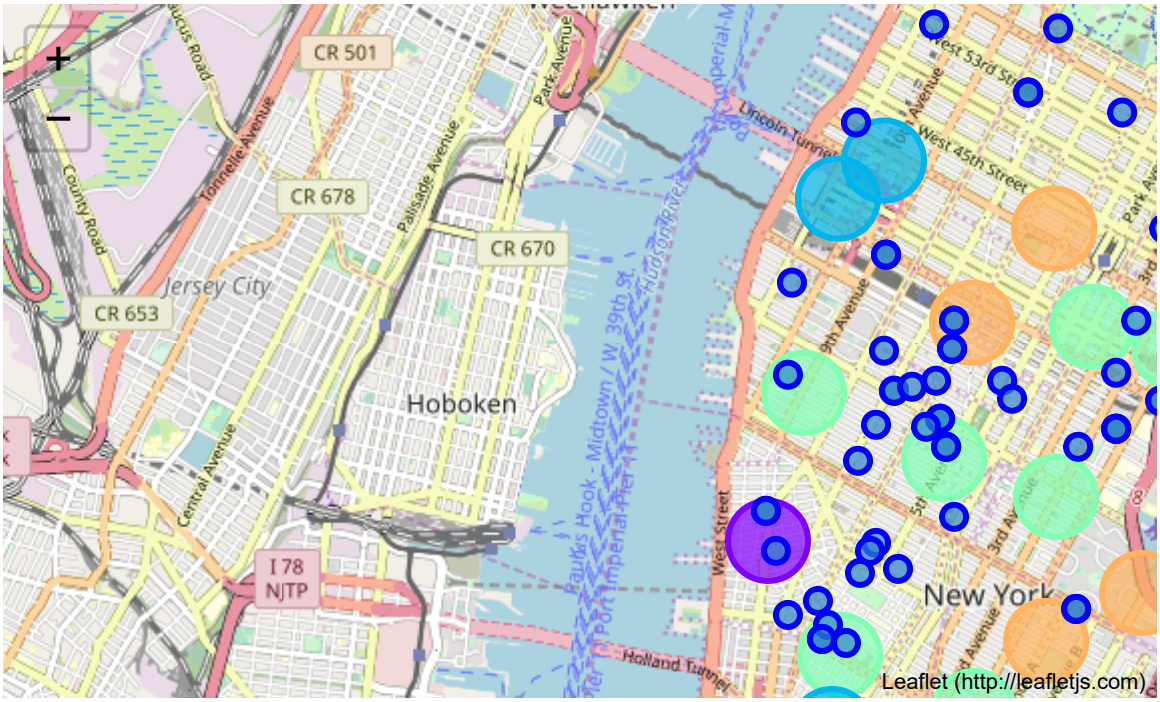
Now, one can point to a rental place for price and address location information while knowing the cluster venues around it.

This is an insightful way to explore rental possibilites

In [39]:

```
# create map of Manhattan using latitude and longitude values from Nominatim
latitude= 40.7308619
longitude= -73.9871558

# create map with clusters
kclusters=5
map_clusters2 = folium.Map(location=[latitude, longitude], zoom_start=13)

# 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(manhattan_merged['Latitude'], manhattan_merged['Longi
tude'], manhattan_merged['Neighborhood'], manhattan_merged['Cluster Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=20,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters2)

# add markers to map for rental places
for lat, lng, label in zip(mh_rent['Lat'], mh_rent['Long'],'$ ' + mh_rent['Rent_Price']
.astype(str)+ mh_rent['Address']):
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=6,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(map_clusters2)

    # Adds tool to the top right
from folium.plugins import MeasureControl
map_manhattan_rent.add_child(MeasureControl())

# FMeasurement ruler icon to establish distnces on map
from folium.plugins import FloatImage
url = ('https://media.licdn.com/mpr/mpr/shrinknp_100_100/AAEAAQAAAAAAAAlgAAAAJGE3OTA4YT
dlLTkzZjUtNDFjYy1iZThlLWQ5OTNkYzlhNzM4OQ.jpg')
FloatImage(url, bottom=5, left=85).add_to(map_manhattan_rent)

map_clusters2
```

Out[39]:



Now one can explore a particular rental place and its venues in detail

In the map above, examination of appartments with rental place below 7000/month is straightforwad while knowing the venues around it.

We could find an appartment with at the right price and in a location with desirable venues. The next step is to see if it is located near a subway metro station, in next cells work.

In [40]:

```
## kk is the cluster number to explore
kk = 3
manhattan_merged.loc[manhattan_merged['Cluster Labels'] == kk, manhattan_merged.columns
[[1] + list(range(5, manhattan_merged.shape[1]))]]
```

Out[40]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th M Comm Ve
3	Inwood	Mexican Restaurant	Lounge	Pizza Place	Café	Wine Bar	Bakery
5	Manhattanville	Deli / Bodega	Italian Restaurant	Seafood Restaurant	Mexican Restaurant	Sushi Restaurant	Beer Garden
10	Lenox Hill	Sushi Restaurant	Italian Restaurant	Coffee Shop	Gym / Fitness Center	Pizza Place	Burger Joint
12	Upper West Side	Italian Restaurant	Bar	Bakery	Vegetarian / Vegan Restaurant	Indian Restaurant	Coffee Shop
16	Murray Hill	Sandwich Place	Hotel	Japanese Restaurant	Gym / Fitness Center	Coffee Shop	Salon / Barbers
17	Chelsea	Coffee Shop	Italian Restaurant	Ice Cream Shop	Bakery	Nightclub	Theater
18	Greenwich Village	Italian Restaurant	Sushi Restaurant	French Restaurant	Clothing Store	Chinese Restaurant	Café
27	Gramercy	Italian Restaurant	Restaurant	Thrift / Vintage Store	Cocktail Bar	Bagel Shop	Coffee Shop
29	Financial District	Coffee Shop	Hotel	Gym	Wine Shop	Steakhouse	Bar
31	Noho	Italian Restaurant	French Restaurant	Cocktail Bar	Gift Shop	Bookstore	Grocery Store
32	Civic Center	Gym / Fitness Center	Bakery	Italian Restaurant	Cocktail Bar	French Restaurant	Sandwi Place
35	Turtle Bay	Italian Restaurant	Coffee Shop	Steakhouse	Wine Bar	Sushi Restaurant	Hotel
36	Tudor City	Café	Park	Pizza Place	Mexican Restaurant	Greek Restaurant	Sushi Restaur
38	Flatiron	Italian Restaurant	American Restaurant	Gym	Gym / Fitness Center	Yoga Studio	Vegetar / Vegan Restaur

Mapping Manhattan Subway locations

Manhattan subway metro locations (address) was obtained from webscrapping sites such as Wikipedia, Google and NY Metro Transit. For simplification, a csv file was produced from the 'numbers' (Apple excel) so that the reading of this file is the starting point here.

In [41]:

```
# A csv file summarized the subway station and the addresses for next step to determine geodata
mh=pd.read_csv('NYC_subway_list.csv')
mh.head()
```

Out[41]:

	sub_station	sub_address
0	Dyckman Street Subway Station	170 Nagle Ave, New York, NY 10034, USA
1	57 Street Subway Station	New York, NY 10106, USA
2	Broad St	New York, NY 10005, USA
3	175 Street Station	807 W 177th St, New York, NY 10033, USA
4	5 Av and 53 St	New York, NY 10022, USA

Add colums labeled 'lat' and 'long' to be filled with geodata

In [42]:

```
#  Add columns 'lat'  and 'long' to mh dataframe - with random temporary numbers to get started
sLength = len(mh['sub_station'])
lat = pd.Series(np.random.randn(sLength))
long=pd.Series(np.random.randn(sLength))
mh = mh.assign(lat=lat.values)
mh = mh.assign(long=long.values)
```

Algorythm to find latitude and longitud for each subway metro station and add them to dataframe

This coding has been 'Markdown' just to simplify the file report, and the csv file will be read in cell below.

for n in range(len(mh)): address= mh['sub_address'][n] geolocator = Nominatim() location = geolocator.geocode(address) latitude = location.latitude longitude = location.longitude mh['lat'][n]=latitude mh['long'][n]=longitude

```
    #print(n,latitude,longitude)
    time.sleep(2)
```

print('Geodata completed')

save dataframe to csv file

mh.to_csv('MH_subway.csv',index=False) mh.shape

Read csv file that produced the subway stations list with geodata

In [43]:

```
mh=pd.read_csv('MH_subway.csv')
print(mh.shape)
mh.head()
```

(76, 4)

Out[43]:

	sub_station	sub_address	lat	long
0	Dyckman Street Subway Station	170 Nagle Ave, New York, NY 10034, USA	40.861857	-73.924509
1	57 Street Subway Station	New York, NY 10106, USA	40.764250	-73.954525
2	Broad St	New York, NY 10005, USA	40.730862	-73.987156
3	175 Street Station	807 W 177th St, New York, NY 10033, USA	40.847991	-73.939785
4	5 Av and 53 St	New York, NY 10022, USA	40.764250	-73.954525

In [44]:

```
# removing duplicate rows and creating new set mhsub1
mhsub1=mh.drop_duplicates(subset=['lat','long'], keep="last").reset_index(drop=True)
mhsub1.shape
```

Out[44]:

```
(22, 4)
```

In [45]:

```
mhsub1.tail()
```

Out[45]:

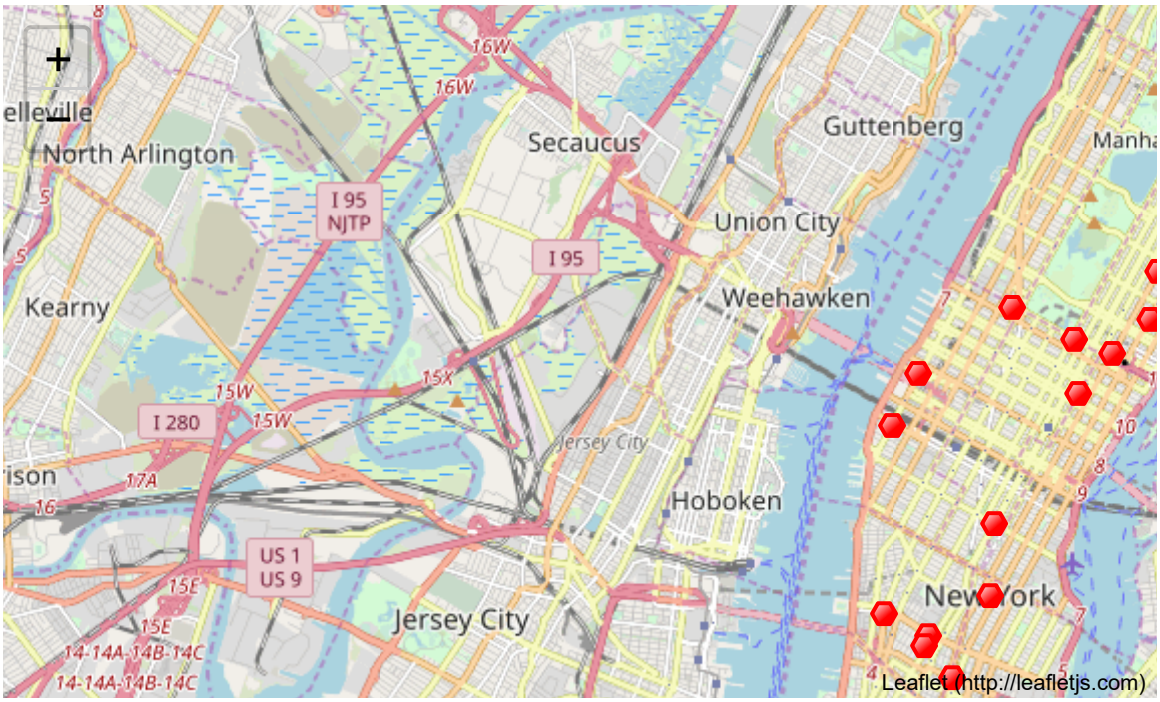
	sub_station	sub_address	lat	long
17	190 Street Subway Station	Bennett Ave, New York, NY 10040, USA	40.858113	-73.932983
18	59 St-Lexington Av Station	E 60th St, New York, NY 10065, USA	40.762259	-73.966271
19	57 Street Station	New York, NY 10019, United States	40.764250	-73.954525
20	14 Street / 8 Av	New York, NY 10014, United States	40.730862	-73.987156
21	MTA New York City	525 11th Ave, New York, NY 10018, USA	40.759809	-73.999282

MAP of Manhattan showing the location of subway stations

In [46]:

```
# map subway stations
# create map of Manhattan using latitude and longitude values obtain previoulsy via Mon
inatim geolocator
latitude=40.7308619
longitude=-73.9871558

map_mhsub1 = folium.Map(location=[latitude, longitude], zoom_start=12)

# add markers of subway locations to map
for lat, lng, label in zip(mhsub1['lat'], mhsub1['long'],  mhsub1['sub_station'].astype
(str) ):
    label = folium.Popup(label, parse_html=True)
    folium.RegularPolygonMarker(
        [lat, lng],
        number_of_sides=6,
        radius=6,
        popup=label,
        color='red',
        fill_color='red',
        fill_opacity=2.5,
    ).add_to(map_mhsub1)
map_mhsub1
```

Out[46]:



Map of Manhattan showing places for rent and the subway locations nearby

Now, we can visualize the desirable rental places and their nearest subway station. Popups display rental address and monthly rental price and the subway station name.

Notice that the icon in the top-right corner is a "ruler" that allows to measure the distance from a rental place to an specific subway station

In [47]:

```
mh_rent.head()
```

Out[47]:

	Address	Area	Price_per_ft2	Rooms	Area-ft2	Rent_Price	Lat	Lo
0	West 105th Street	Upper West Side	2.94	5.0	3400	10000	40.799771	-73.966
1	East 97th Street	Upper East Side	3.57	3.0	2100	7500	40.788585	-73.955
2	West 105th Street	Upper West Side	1.89	4.0	2800	5300	40.799771	-73.966
3	CARMINE ST.	West Village	3.03	2.0	1650	5000	40.730523	-74.001
4	171 W 23RD ST.	Chelsea	3.45	2.0	1450	5000	40.744118	-73.995

In [48]:

```
# create map of Manhattan using latitude and longitude values from Nominatim
latitude= 40.7308619
longitude= -73.9871558

map_manhattan_rent = folium.Map(location=[latitude, longitude], zoom_start=13.3)

# add markers to map
for lat, lng, label in zip(mh_rent['Lat'], mh_rent['Long'],'$ ' + mh_rent['Rent_Price']
.astype(str)+ mh_rent['Address']):
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=6,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(map_manhattan_rent)

    # add markers of subway locations to map
for lat, lng, label in zip(mhsub1['lat'], mhsub1['long'],  mhsub1['sub_station'].astype
(str) ):
    label = folium.Popup(label, parse_html=True)
    folium.RegularPolygonMarker(
        [lat, lng],
        number_of_sides=6,
        radius=6,
        popup=label,
        color='red',
        fill_color='red',
        fill_opacity=2.5,
    ).add_to(map_manhattan_rent)

    # Adds tool to the top right
from folium.plugins import MeasureControl
map_manhattan_rent.add_child(MeasureControl())

# Measurement ruler icon tool to measure distances in map
from folium.plugins import FloatImage
url = ('https://media.licdn.com/mpr/mpr/shrinknp_100_100/AAEAAQAAAAAAAAlgAAAAJGE3OTA4YT
dlLTkzZjUtNDFjYy1iZThlLWQ5OTNkYzlhNzM4OQ.jpg')
FloatImage(url, bottom=5, left=85).add_to(map_manhattan_rent)

map_manhattan_rent
```

Out[48]:

4.0 Results

ONE CONSOLIDATE MAP

Let's consolidate all the required inforamtion to make the apartment selection in one map

Map of Manhattan with rental places, subway locations and cluster of venues

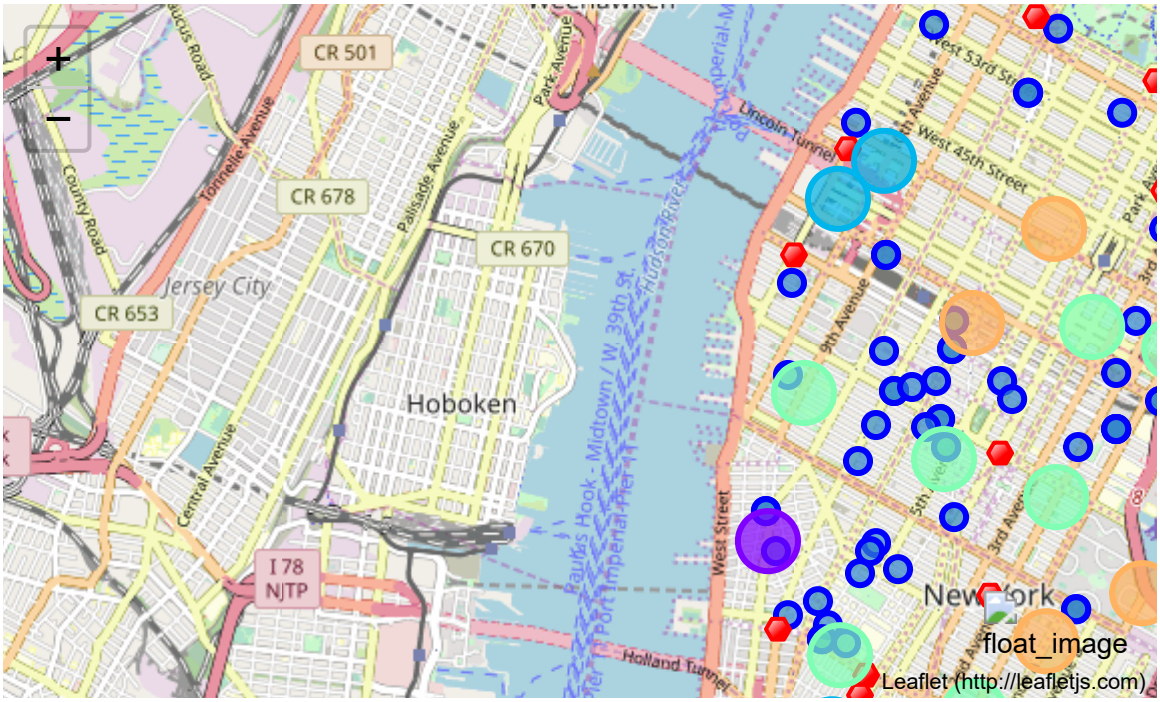
Red dots are Subway stations, Blue dots are apartments available for rent, Bubbles are the clusters of venues

In [49]:

```
# create map of Manhattan using latitude and longitude values from Nominatim
latitude= 40.7308619
longitude= -73.9871558

map_mh_one = folium.Map(location=[latitude, longitude], zoom_start=13.3)

# add markers to map
for lat, lng, label in zip(mh_rent['Lat'], mh_rent['Long'],'$ ' + mh_rent['Rent_Price']
.astype(str)+ ', '+mh_rent['Address']):
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=6,
        popup=label,
        color='blue',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(map_mh_one)

    # add markers of subway locations to map
for lat, lng, label in zip(mhsub1['lat'], mhsub1['long'],  mhsub1['sub_station'].astype
(str) ):
    label = folium.Popup(label, parse_html=True)
    folium.RegularPolygonMarker(
        [lat, lng],
        number_of_sides=6,
        radius=6,
        popup=label,
        color='red',
        fill_color='red',
        fill_opacity=2.5,
    ).add_to(map_mh_one)


# set color scheme for the clusters
kclusters=5
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(manhattan_merged['Latitude'], manhattan_merged['Longi
tude'], manhattan_merged['Neighborhood'], manhattan_merged['Cluster Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=15,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_mh_one)

    # Adds tool to the top right
from folium.plugins import MeasureControl
map_mh_one.add_child(MeasureControl())
```

```
# Measurement ruler icon tool to measure distances in map
from folium.plugins import FloatImage
url = ('https://media.licdn.com/mpr/mpr/shrinknp_100_100/AAEAAQAAAAAAAAlgAAAAJGE3OTA4YT
dlLTkzZjUtNDFjYy1iZThlLWQ5OTNkYzlhNzM4OQ.jpg')
FloatImage(url, bottom=5, left=85).add_to(map_mh_one)

map_mh_one
```

Out[49]:



Problem Resolution - Select the apartment for rent

The above consolidate map was used to explore options.

After examining, I have chosen two locations that meet the requirements which will assess to make a choice.

- Apartment 1: 305 East 63rd Street in the Sutton Place Neighborhood and near 'subway 59th Street' station, Cluster # 2 Monthly rent : 7500 Dollars
- Apartment 2: 19 Dutch Street in the Financial District Neighborhood and near 'Fulton Street Subway' station, Cluster # 3 Monthly rent : 6935 Dollars

Venues for Apartment 1 - Cluster 2

In [50]:

```
## kk is the cluster number to explore
kk = 2
manhattan_merged.loc[manhattan_merged['Cluster Labels'] == kk, manhattan_merged.columns
[[1] + list(range(5, manhattan_merged.shape[1]))]]
```

Out[50]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th M Comm Ve
0	Marble Hill	Coffee Shop	Discount Store	Yoga Studio	Steakhouse	Supplement Shop	Tennis Stadiun
1	Chinatown	Chinese Restaurant	Cocktail Bar	Dim Sum Restaurant	American Restaurant	Vietnamese Restaurant	Salon / Barbers
6	Central Harlem	African Restaurant	Seafood Restaurant	French Restaurant	American Restaurant	Cosmetics Shop	Chinese Restaur
9	Yorkville	Coffee Shop	Gym	Bar	Italian Restaurant	Sushi Restaurant	Pizza Place
14	Clinton	Theater	Italian Restaurant	Coffee Shop	American Restaurant	Gym / Fitness Center	Hotel
23	Soho	Clothing Store	Boutique	Women's Store	Shoe Store	Men's Store	Furnitur Home Store
26	Morningside Heights	Coffee Shop	American Restaurant	Park	Bookstore	Pizza Place	Sandwi Place
34	Sutton Place	Gym / Fitness Center	Italian Restaurant	Furniture / Home Store	Indian Restaurant	Dessert Shop	Americ Restaur
39	Hudson Yards	Coffee Shop	Italian Restaurant	Hotel	Theater	American Restaurant	Café

Venues for Apartment 2 - Cluster 3

In [51]:

```
## kk is the cluster number to explore
kk = 3
manhattan_merged.loc[manhattan_merged['Cluster Labels'] == kk, manhattan_merged.columns
[[1] + list(range(5, manhattan_merged.shape[1]))]]
```

Out[51]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th M Comm Ve
3	Inwood	Mexican Restaurant	Lounge	Pizza Place	Café	Wine Bar	Bakery
5	Manhattanville	Deli / Bodega	Italian Restaurant	Seafood Restaurant	Mexican Restaurant	Sushi Restaurant	Beer Garden
10	Lenox Hill	Sushi Restaurant	Italian Restaurant	Coffee Shop	Gym / Fitness Center	Pizza Place	Burger Joint
12	Upper West Side	Italian Restaurant	Bar	Bakery	Vegetarian / Vegan Restaurant	Indian Restaurant	Coffee Shop
16	Murray Hill	Sandwich Place	Hotel	Japanese Restaurant	Gym / Fitness Center	Coffee Shop	Salon / Barbers
17	Chelsea	Coffee Shop	Italian Restaurant	Ice Cream Shop	Bakery	Nightclub	Theater
18	Greenwich Village	Italian Restaurant	Sushi Restaurant	French Restaurant	Clothing Store	Chinese Restaurant	Café
27	Gramercy	Italian Restaurant	Restaurant	Thrift / Vintage Store	Cocktail Bar	Bagel Shop	Coffee Shop
29	Financial District	Coffee Shop	Hotel	Gym	Wine Shop	Steakhouse	Bar
31	Noho	Italian Restaurant	French Restaurant	Cocktail Bar	Gift Shop	Bookstore	Grocery Store
32	Civic Center	Gym / Fitness Center	Bakery	Italian Restaurant	Cocktail Bar	French Restaurant	Sandwi Place
35	Turtle Bay	Italian Restaurant	Coffee Shop	Steakhouse	Wine Bar	Sushi Restaurant	Hotel
36	Tudor City	Café	Park	Pizza Place	Mexican Restaurant	Greek Restaurant	Sushi Restaur
38	Flatiron	Italian Restaurant	American Restaurant	Gym	Gym / Fitness Center	Yoga Studio	Vegetar / Vegan Restaur

Apartment Selection

Using the "one map" above, I was able to explore all possibilities since the popups provide the information needed for a good decision.

Apartment 1 rent cost is US7500 slightly above the US7000 budget. Apt 1 is located 400 meters from subway station at 59th Street and work place (Park Ave and 53rd) is another 600 meters way. I can walk to work place and use subway for other places aroung. Venues for this apt are as of Cluster 2 and it is located in a fine district in the East side of Manhattan.

Apartment 2 rent cost is US6935, just under the US7000 budget. Apt 2 is located 60 meters from subway station at Fulton Street, but I will have to ride the subway daily to work , possibly 40-60 min ride. Venues for this apt are as of Cluster 3.¶

Based on current Singapore venues, I feel that Cluster 2 type of venues is a closer resemblance to my current place. That means that APARTMENT 1 is a better choice since the extra monthly

5.0 DISCUSSION

In general, I am positively impressed with the overall organization, content and lab works presented during the Coursera IBM Certification Course

I feel this Capstone project presented me a great opportunity to practice and apply the Data Science tools and methodologies learned.

I have created a good project that I can present as an example to show my potential.

I feel I have acquired a good starting point to become a professional Data Scientist and I will continue exploring to creating examples of practical cases.

6.0 CONCLUSIONS

I feel rewarded with the efforts, time and money spent. I believe this course with all the topics covered is well worthy of appreciation.

This project has shown me a practical application to resolve a real situation that has impacting personal and financial impact using Data Science tools.

The mapping with Folium is a very powerful technique to consolidate information and make the analysis and decision thoroughly and with confidence. I would recommend for use in similar situations.

One must keep abreast of new tools for DS that continue to appear for application in several business fields