Capstone Project - The Battle of Neighborhoods (Week 2)

Introduction

New York City comprises 5 boroughs sitting where the Hudson River meets the Atlantic Ocean. At its core is Manhattan, a densely populated borough that’s among the world’s major commercial, financial and cultural centers. Its iconic sites include skyscrapers such as the Empire State Building and sprawling Central Park. Broadway theater is staged in neon-lit Times Square.

London, the capital of England and the United Kingdom, is a 21st-century city with history stretching back to Roman times. At its centre stand the imposing Houses of Parliament, the iconic ‘Big Ben’ clock tower and Westminster Abbey, site of British monarch coronations. Across the Thames River, the London Eye observation wheel provides panoramic views of the South Bank cultural complex, and the entire city.

Description of the problem

We will explored New York City and London and segmented and clustered their neighborhoods. Both cities are very diverse and are very similar. Both cities are a densely populated boroughs that’s among the world’s major commercial, financial and cultural centers. . We will to compare the neighborhoods of the two cities and determine how similar or dissimilar they are. We will define that people like to do more in the cities, which places are often visited. Knowing this information we can think of how to use this. For exemple, open a new restaurant or supermarket, entertainment center or gift shop. As we can see in the next task that although there are Mexican restaurants in London, but they are not popular, entertainment is centrally located and almost none in areas farther from the center. We may also use this information for advertising purposes, etc

Description of Data.

This project will rely on public data from Wikipedia and Foursquare.

London is the capital of and largest city in England and the United Kingdom. It is administered by the City of London and 32 London boroughs.

We will get information about the areas of London https://en.wikipedia.org/wiki/List_of_areas_of_London (https://en.wikipedia.org/wiki/List_of_areas_of_London)

I will use dataset https://geo.nyu.edu/catalog/nyu_2451_34572 (https://geo.nyu.edu/catalog/nyu_2451_34572) for information about boroughs of NYC

In [2]:
```
# library for BeautifulSoup
from bs4 import BeautifulSoup

import numpy as np
import pandas as pd
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

# library to handle JSON files
import json

!pip -q install geopy
# conda install -c conda-forge geopy --yes # uncomment this line if you
 haven't completed the Foursquare API lab

# convert an address into latitude and longitude values
from geopy.geocoders import Nominatim

# library to handle requests
import requests

# tranform JSON file into a pandas dataframe
from pandas.io.json import json_normalize

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

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

# install the Geocoder
!pip -q install geocoder
import geocoder

# import time
import time

# !conda install -c conda-forge folium=0.5.0 --yes # uncomment this line
if you haven't completed the Foursquare API lab
!pip -q install folium

import folium # map rendering library

from PIL import Image # converting images into arrays

%matplotlib inline

import matplotlib as mpl
import matplotlib.pyplot as plt

mpl.style.use('ggplot') # optional: for ggplot-like style

# check for latest version of Matplotlib
#print ('Matplotlib version: ', mpl.__version__) # >= 2.0.0
```

```
# install wordcloud
!conda install -c conda-forge wordcloud==1.4.1 --yes
from wordcloud import WordCloud, STOPWORDS

print ('...Done')
```

```
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.5.12
  latest version: 4.7.10

Please update conda by running

    $ conda update -n base conda



# All requested packages already installed.

...Done
```

London

In [3]:
```
# download data and parse it:
r = requests.get('https://en.wikipedia.org/wiki/List_of_areas_of_London'
)
soup = BeautifulSoup(r.text, 'html.parser')
table=soup.find('table', attrs={'class':'wikitable sortable'})
```

In [4]:
```
#get headers:
headers=table.findAll('th')
for i, head in enumerate(headers): headers[i]=str(headers[i]).replace("<
th>","").replace("</th>","").replace("\n","")
#headers
```

In [5]:
```
#Find all items and skip first one:
rows=table.findAll('tr')
rows=rows[1:len(rows)]
#rows
```

In [6]:
```
# skip all meta symbols and line feeds between rows:
for i, row in enumerate(rows): rows[i] = str(rows[i]).replace("\n</td></
tr>","").replace("<tr>\n<td>","")
#rows
```

In [7]:
```
# make dataframe, expand rows and drop the old one:
df=pd.DataFrame(rows)
df[headers] = df[0].str.split("</td>\n<td>", n = 7, expand = True)
df.drop(columns=[0],inplace=True)#

df.rename(columns={'Location': 'neighborhoods', 'London\xa0borough': 'borough', 'Post town': 'posttown', 'Postcode\xa0district': 'postcode'}, inplace=True )
df.drop(columns={'OS grid ref'},inplace=True)
df.head(3)
```

Out[7]:

	neighborhoods	borough	posttown	postcode	Dial code
0	<a href="/wiki/Abbey_Wood" title="Abbey Wood">...	Bexley, Greenwich <sup class="reference" id="...	LONDON	SE2	020
1	<a href="/wiki/Acton,_London" title="Acton, Lo...	Ealing, Hammersmith and Fulham<sup class="refe...	LONDON	W3, W4	020
2	<a href="/wiki/Addington,_London" title="Addin...	Croydon<sup class="reference" id="cite_ref-mil...	CROYDON	CR0	020

In [8]:
```
df.update(df.neighborhoods.loc[lambda x: x.str.contains('title')].str.extract('title=\"([^\"]*)',expand=False))
# delete Toronto annotation from Neighbourhood:
df.update(df.neighborhoods.loc[lambda x: x.str.contains('London')].str.replace(", London",""))
```

In [9]:
```
for i in range(0, df.shape[0]-1):
    #print(df.borough.get_values()[i])
    c = df.borough.get_values()[i].split('<')[0]
    df.borough[i] = c

df = df.drop('borough', axis=1).join(df['borough'].str.split(',', expand=True).stack().reset_index(level=1, drop=True).rename('borough'))
df = df.drop('posttown', axis=1).join(df['posttown'].str.split(',', expand=True).stack().reset_index(level=1, drop=True).rename('posttown'))

df = df.drop('postcode', axis=1).join(df['postcode'].str.split(',', expand=True).stack().reset_index(level=1, drop=True).rename('postcode'))
```

In [10]:
```
df.head()
```

Out[10]:

	neighborhoods	Dial code	borough	posttown	postcode
0	Abbey Wood	020	Bexley	LONDON	SE2
0	Abbey Wood	020	Bexley	LONDON	SE2
0	Abbey Wood	020	Bexley	LONDON	SE2
0	Abbey Wood	020	Bexley	LONDON	SE2
0	Abbey Wood	020	Bexley	LONDON	SE2

In [11]:
```
df.shape
```

Out[11]: (2856, 5)

In [12]:
```
df.drop_duplicates(keep = False, inplace = True)
```

In [13]:
```
df.head()
```

Out[13]:

	neighborhoods	Dial code	borough	posttown	postcode
2	Addington	020	Croydon	CROYDON	CR0
3	Addiscombe	020	Croydon	CROYDON	CR0
5	Aldborough Hatch	020	Redbridge	ILFORD	IG2
6	Aldgate	020	City	LONDON	EC3
7	Aldwych	020	Westminster	LONDON	WC2

In [14]:
```
df.shape
```

Out[14]: (578, 5)

Now, only the Boroughs with London Post-town will be used for our search of location. Therefore, all the non-post-town are dropped.

In [15]:
```
df_london = df
df_london = df_london[df_london['posttown'].str.contains('LONDON')]

df_london.drop_duplicates(keep = False, inplace = True)
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:4: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-d
ocs/stable/indexing.html#indexing-view-versus-copy
  after removing the cwd from sys.path.
```

In [16]:
```
df_london.head()
```

Out[16]:

	neighborhoods	Dial code	borough	posttown	postcode
6	Aldgate	020	City	LONDON	EC3
7	Aldwych	020	Westminster	LONDON	WC2
9	Anerley	020	Bromley	LONDON	SE20
10	Angel	020	Islington	LONDON	EC1
10	Angel	020	Islington	LONDON	N1

Geocoder dont read my whole data, and i divide my dataset on smaller parts

In [17]:
```
# Defining a function to use --> get_latlng()'''
def get_latlng(arcgis_geocoder):

    # Initialize the Location (lat. and long.) to "None"
    lat_lng_coords = None

    # While loop helps to create a continous run until all the location
 coordinates are geocoded
    while(lat_lng_coords is None):
        g = geocoder.arcgis('{}, London, United Kingdom'.format(arcgis_g
eocoder))
        lat_lng_coords = g.latlng
    return lat_lng_coords
# Geocoder ends here
```

In [18]:
```
# New dataframe for  postcodes started with "W"
df_w = df_london[df_london['postcode'].str.startswith(('W'))].reset_inde
x(drop=True)
```

In [19]:
```
df_w.head()
```

Out[19]:

	neighborhoods	Dial code	borough	posttown	postcode
0	Aldwych	020	Westminster	LONDON	WC2
1	Bayswater	020	Westminster	LONDON	W2
2	Bedford Park	020	Ealing	LONDON	W4
3	Bloomsbury	020	Camden	LONDON	WC1
4	Charing Cross	020	Westminster	LONDON	WC2

In [20]:
```
postcode = df_w['postcode']
postcode
coordinates = [get_latlng(postcode) for postcode in postcode.tolist()]
```

In [21]:
```
df_with_coordinates = df_w

# The obtained coordinates (latitude and longitude) are joined with the
 dataframe as shown
df_with_coordinates = pd.DataFrame(coordinates, columns = ['Latitude',
'Longitude'])
df_w['Latitude'] = df_with_coordinates['Latitude']
df_w['Longitude'] = df_with_coordinates['Longitude']
```

In [22]:
```
df_w.head()
```

Out[22]:

	neighborhoods	Dial code	borough	posttown	postcode	Latitude	Longitude
0	Aldwych	020	Westminster	LONDON	WC2	51.51651	-0.11968
1	Bayswater	020	Westminster	LONDON	W2	51.51494	-0.18048
2	Bedford Park	020	Ealing	LONDON	W4	51.48944	-0.26194
3	Bloomsbury	020	Camden	LONDON	WC1	51.52450	-0.12273
4	Charing Cross	020	Westminster	LONDON	WC2	51.51651	-0.11968

In [23]:
```
# # New dataframe for  postcodes started with "S"
df_s = df_london[df_london['postcode'].str.startswith(('S'))].reset_inde
x(drop=True)
```

In [24]:
```
df_s.head()
```

Out[24]:

	neighborhoods	Dial code	borough	posttown	postcode
0	Anerley	020	Bromley	LONDON	SE20
1	Balham	020	Wandsworth	LONDON	SW12
2	Bankside	020	Southwark	LONDON	SE1
3	Barnes	020	Richmond upon Thames	LONDON	SW13
4	Battersea	020	Wandsworth	LONDON	SW11

In [25]:
```
postcode = df_s['postcode']
postcode
coordinates = [get_latlng(postcode) for postcode in postcode.tolist()]
```

In [26]:
```
df_with_coordinates_s = df_s

# The obtained coordinates (latitude and longitude) are joined with the
 dataframe as shown
df_with_coordinates_s = pd.DataFrame(coordinates, columns = ['Latitude',
'Longitude'])
df_s['Latitude'] = df_with_coordinates_s['Latitude']
df_s['Longitude'] = df_with_coordinates_s['Longitude']
```

In [27]:
```
df_s.head()
```

Out[27]:

	neighborhoods	Dial code	borough	posttown	postcode	Latitude	Longitude
0	Anerley	020	Bromley	LONDON	SE20	51.41009	-0.05683
1	Balham	020	Wandsworth	LONDON	SW12	51.44822	-0.14839
2	Bankside	020	Southwark	LONDON	SE1	51.49960	-0.09613
3	Barnes	020	Richmond upon Thames	LONDON	SW13	51.47457	-0.24212
4	Battersea	020	Wandsworth	LONDON	SW11	51.46760	-0.16290

In [28]:
```
# df_london_allpart = df_s and df_w
df_london_allpart = df_s.append(df_w, ignore_index=True)
```

In [29]:
```
df_london_allpart.shape
```

Out[29]: (129, 7)

In [30]:
```
# New dataframe for  postcodes started with "E"
df_e = df_london[df_london['postcode'].str.startswith(('E'))].reset_inde
x(drop=True)
```

In [31]:
```
postcode = df_e['postcode']
postcode
coordinates = [get_latlng(postcode) for postcode in postcode.tolist()]
```

In [32]:
```
f_with_coordinates_e = df_e

# The obtained coordinates (latitude and longitude) are joined with the
 dataframe as shown
df_with_coordinates_e = pd.DataFrame(coordinates, columns = ['Latitude',
'Longitude'])
df_e['Latitude'] = df_with_coordinates_e['Latitude']
df_e['Longitude'] = df_with_coordinates_e['Longitude']
```

In [33]:
```
df_london_allpart = df_london_allpart.append(df_e, ignore_index=True)
```

In [34]:
```
# New dataframe for  postcodes started with "N"
df_n = df_london[df_london['postcode'].str.startswith(('N'))].reset_inde
x(drop=True)
```

In [35]:
```
postcode = df_n['postcode']
postcode
coordinates = [get_latlng(postcode) for postcode in postcode.tolist()]
```

In [36]:
```
df_with_coordinates_s = df_n

# The obtained coordinates (latitude and longitude) are joined with the
 dataframe as shown
df_with_coordinates_n = pd.DataFrame(coordinates, columns = ['Latitude',
'Longitude'])
df_n['Latitude'] = df_with_coordinates_n['Latitude']
df_n['Longitude'] = df_with_coordinates_n['Longitude']
```

In [37]:
```
df_london_allpart = df_london_allpart.append(df_n, ignore_index=True)
```

In [38]:
```
#df_london_allpart.head(10)
```

In [39]:
```
# New dataframe for  postcodes started with "d"
df_d = df_london[df_london['postcode'].str.startswith(('D'))].reset_inde
x(drop=True)
```

In [40]:
```
postcode = df_d['postcode']
postcode
coordinates = [get_latlng(postcode) for postcode in postcode.tolist()]
```

In [41]:
```
df_with_coordinates_s = df_d

# The obtained coordinates (latitude and longitude) are joined with the
 dataframe as shown
df_with_coordinates_d = pd.DataFrame(coordinates, columns = ['Latitude',
'Longitude'])
df_d['Latitude'] = df_with_coordinates_d['Latitude']
df_d['Longitude'] = df_with_coordinates_d['Longitude']
```

In [42]:
```
df_london_allpart = df_london_allpart.append(df_d, ignore_index=True)
```

In [43]:
```
# New dataframe for  postcodes started with "I"/ same =E18
df_i = df_london[df_london['postcode'].str.startswith(('I'))].reset_inde
x(drop=True)
```

In [44]:
```
postcode = df_i['postcode']
postcode
coordinates = [get_latlng(postcode) for postcode in postcode.tolist()]
```

In [45]:
```
df_with_coordinates_s = df_i

# The obtained coordinates (latitude and longitude) are joined with the
 dataframe as shown
df_with_coordinates_i = pd.DataFrame(coordinates, columns = ['Latitude',
'Longitude'])
df_i['Latitude'] = df_with_coordinates_i['Latitude']
df_i['Longitude'] = df_with_coordinates_i['Longitude']
```

In [46]:
```
df_london_allpart = df_london_allpart.append(df_i, ignore_index=True)
```

In [47]:
```
df_london_allpart.head()
```

Out[47]:

	neighborhoods	Dial code	borough	posttown	postcode	Latitude	Longitude
0	Anerley	020	Bromley	LONDON	SE20	51.41009	-0.05683
1	Balham	020	Wandsworth	LONDON	SW12	51.44822	-0.14839
2	Bankside	020	Southwark	LONDON	SE1	51.49960	-0.09613
3	Barnes	020	Richmond upon Thames	LONDON	SW13	51.47457	-0.24212
4	Battersea	020	Wandsworth	LONDON	SW11	51.46760	-0.16290

In [48]:
```
df_london_allpart['borough'].unique()
```

Out[48]:
```
array(['Bromley', 'Wandsworth', 'Southwark', 'Richmond upon Thames',
       'Westminster', 'Lewisham', 'Greenwich', 'Lambeth',
       'Kensington and Chelsea', 'Merton', 'Bexley',
       'Hammersmith and Fulham', 'Kingston upon Thames', 'Croydon',
       'Ealing', 'Camden', 'Hounslow', 'Camden and Islington', 'City',
       'Islington', 'Tower Hamlets', 'Waltham Forest', 'Newham', 'Hackn
ey',
       'Islington &amp; City', 'Redbridge', 'Enfield', 'Haringey',
       'Barnet', 'Brent', 'Haringey and Barnet', 'Dartford'], dtype=obj
ect)
```

In [49]:
```
print('The dataframe has {} boroughs and {} neighborhoods.'.format(
        len(df_london_allpart['borough'].unique()),
        df_london_allpart.shape[0]
    )
)
```

```
The dataframe has 32 boroughs and 285 neighborhoods.
```

Use geopy library to get the latitude and longitude values of London. In order to define an instance of the geocoder, we need to define a user_agent. We will name our agent ny_explorer, as shown below.

In [50]:
```
address = 'London, uk'

geolocator = Nominatim(user_agent="uk_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of London, uk {}, {}.'.format(latitude
, longitude))
```

```
The geograpical coordinate of London, uk 51.4893335, -0.14405508452768
7.
```

Create a map of London with borough superimposed on top.

In [51]:
```
# create map of London using latitude and longitude values
map_london = folium.Map(location=[latitude, longitude], zoom_start=10)

# add markers to map
for lat, lng, label in zip(df_london_allpart['Latitude'], df_london_allp
art['Longitude'], df_london_allpart['borough']):
    label = '{}.format(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_london)

#map_london
```

New York

Download and Explore Dataset 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.

The link to the dataset: https://geo.nyu.edu/catalog/nyu_2451_34572 (https://geo.nyu.edu/catalog/nyu_2451_34572)

In [52]:
```
# load the data
!wget -q -O 'newyork_data.json' https://cocl.us/new_york_dataset
print('Data downloaded!')
```

```
Data downloaded!
```

In [53]:
```
with open('newyork_data.json') as json_data:
    newyork_data = json.load(json_data)
```

In [54]:
```
neighborhoods_data = newyork_data['features']
```

In [55]:
```
# define the dataframe columns
column_names = ['Borough', 'Neighborhood', 'Latitude', 'Longitude']

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

In [56]:
```
neighborhoods
```

Out[56]:

	Borough	Neighborhood	Latitude	Longitude

In [57]:
```
# let's loop through the data and fill the dataframe one row at a time.
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)
```

In [58]:
```
neighborhoods.head()
```

Out[58]:

	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 [59]:
```
print('The dataframe has {} boroughs and {} neighborhoods.'.format(
        len(neighborhoods['Borough'].unique()),
        neighborhoods.shape[0]
    )
)
```

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

Use geopy library to get the latitude and longitude values of New York City. In order to define an instance of the geocoder, we need to define a user_agent. We will name our agent ny_explorer, as shown below.

In [60]:
```
address = 'New York City, NY'

geolocator = Nominatim(user_agent="ny_explorer")
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 of New York with neighborhoods superimposed on top.

In [61]:
```
# create map of New York using latitude and longitude values
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
```

Methodology

Data Exploration

Create a new dataframe of the borough Kensington and Chelsea.

In [62]:
```
new_data = df_london_allpart[df_london_allpart['borough'] == 'Kensington
and Chelsea'].reset_index(drop=True)
new_data = new_data.drop_duplicates()
new_data.head()
```

Out[62]:

	neighborhoods	Dial code	borough	posttown	postcode	Latitude	Longitude
0	Brompton	020	Kensington and Chelsea	LONDON	SW3	51.49014	-0.16248
1	Chelsea	020	Kensington and Chelsea	LONDON	SW3	51.49014	-0.16248
2	Earls Court	020	Kensington and Chelsea	LONDON	SW5	51.49004	-0.18971
3	Kensington	020	Kensington and Chelsea	LONDON	SW7	51.49807	-0.17404
4	South Kensington	020	Kensington and Chelsea	LONDON	SW7	51.49807	-0.17404

In [63]:
```
address = 'Kensington and Chelsea, uk'
#address = 'City of London, uk'
#address = 'Islington, uk'

geolocator = Nominatim(user_agent="uk_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of London, uk {}, {}.'.format(latitude
, longitude))
```

```
The geograpical coordinate of London, uk 51.4989948, -0.1991229.
```

In [64]:
```
new_data.head()
```

Out[64]:

	neighborhoods	Dial code	borough	posttown	postcode	Latitude	Longitude
0	Brompton	020	Kensington and Chelsea	LONDON	SW3	51.49014	-0.16248
1	Chelsea	020	Kensington and Chelsea	LONDON	SW3	51.49014	-0.16248
2	Earls Court	020	Kensington and Chelsea	LONDON	SW5	51.49004	-0.18971
3	Kensington	020	Kensington and Chelsea	LONDON	SW7	51.49807	-0.17404
4	South Kensington	020	Kensington and Chelsea	LONDON	SW7	51.49807	-0.17404

In [65]:
```
 #create map of North York using latitude and longitude values  #new_dat
a['neighborhoods']
map_london_borough = folium.Map(location=[latitude, longitude], zoom_sta
rt=12)

# add markers to map
for lat, lng, label in zip(new_data['Latitude'], new_data['Longitude'],
new_data['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_london_borough)

#map_london_borough
```

Use geopy library to get the latitude and longitude values borough Manhattan.

In [66]:
```
address = 'Manhattan, usa'

geolocator = Nominatim(user_agent="ny_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of Manhattan are {}, {}.'.format(latit
ude, longitude))
```

```
The geograpical coordinate of Manhattan are 40.7900869, -73.9598295.
```

In [67]:
```
ny_data = neighborhoods[neighborhoods['Borough'] == 'Manhattan'].reset_i
ndex(drop=True)
ny_data.head()
```

Out[67]:

	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

Let's visualizat Manhattan

In [68]:
```
# create map of Manhattan using latitude and longitude values
map_nyc_m = folium.Map(location=[latitude, longitude], zoom_start=11)

# add markers to map
for lat, lng, label in zip(ny_data['Latitude'], ny_data['Longitude'], ny
_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_nyc_m)

#map_nyc_m
```

Next, we are going to start utilizing the Foursquare API to explore the neighborhoods and segment them.

In [69]:
```
CLIENT_ID = 'B3D1FREXU3FMFKG0XFFFWLZH1UBNQKQGVTG4XWBI3N32354V' # your Fo
ursquare ID
CLIENT_SECRET = 'UAFKLDYGA1SQEBZYO4P5DYUAS4DBRF5QA53DURWY03FTRQP3' # you
r Foursquare Secret
VERSION = '20180604'
LIMIT = 30
print('Your credentails:')
print('CLIENT_ID: ' + CLIENT_ID)
print('CLIENT_SECRET:' + CLIENT_SECRET)
```

```
Your credentails:
CLIENT_ID: B3D1FREXU3FMFKG0XFFFWLZH1UBNQKQGVTG4XWBI3N32354V
CLIENT_SECRET:UAFKLDYGA1SQEBZYO4P5DYUAS4DBRF5QA53DURWY03FTRQP3
```

In [70]:
```
address = 'Manhattan, usa'

geolocator = Nominatim(user_agent="ny_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of Manhattan, usa are {}, {}.'.format(
latitude, longitude))
```

```
The geograpical coordinate of Manhattan, usa are 40.7900869, -73.959829
5.
```

Get the neighborhood's latitude and longitude values.

Now, let's get the top 200 venues that are in Manhattan within a radius of 1000 meters. First, let's create the GET request URL. Name your URL url.

In [71]:
```
LIMIT = 200 # limit of number of venues returned by Foursquare API
radius = 1000 # define radius
url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client
_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    latitude,
    longitude,
    radius,
    LIMIT)
url # display URL
```

Out[71]:
```
'https://api.foursquare.com/v2/venues/explore?&client_id=B3D1FREXU3FMFK
G0XFFFWLZH1UBNQKQGVTG4XWBI3N32354V&client_secret=UAFKLDYGA1SQEBZYO4P5DY
UAS4DBRF5QA53DURWY03FTRQP3&v=20180604&ll=40.7900869,-73.9598295&radius=
1000&limit=200'
```

Send the GET request and examine the resutls

In [72]:
```
results = requests.get(url).json()
#results
```

In [73]:
```
# function that extracts the category of the venue
def get_category_type(row):
    try:
        categories_list = row['categories']
    except:
        categories_list = row['venue.categories']

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

Now we are ready to clean the json and structure it into a pandas dataframe.

In [74]:
```
venues = results['response']['groups'][0]['items']

nearby_venues = json_normalize(venues) # flatten JSON

# filter columns
filtered_columns = ['venue.name', 'venue.categories', 'venue.location.la
t', 'venue.location.lng']
nearby_venues =nearby_venues.loc[:, filtered_columns]

# filter the category for each row
nearby_venues['venue.categories'] = nearby_venues.apply(get_category_typ
e, axis=1)

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

nearby_venues.head()
```

Out[74]:

	name	categories	lat	lng
0	North Meadow	Park	40.792027	-73.959853
1	Central Park Tennis Center	Tennis Court	40.789313	-73.961862
2	East Meadow	Field	40.790160	-73.955498
3	Central Park - Gate Of All Saints	Park	40.791591	-73.964795
4	The Jewish Museum	Museum	40.785276	-73.957411

And how many venues were returned by Foursquare?

In [75]:
```
print('{} venues were returned by Foursquare.'.format(nearby_venues.shap
e[0]))
```

```
100 venues were returned by Foursquare.
```

Let's create a function to repeat the same process to all the neighborhoods.

In [76]:
```
def getNearbyVenues(names, latitudes, longitudes, radius=500):

    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)
```

Create a new dataframe called ny_venues.

In [77]:
```
ny_venues = getNearbyVenues(names=ny_data['Neighborhood'],
                                   latitudes=ny_data['Latitude'],
                                   longitudes=ny_data['Longitude']
                                  )
```

```
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
Carnegie Hill
Noho
Civic Center
Midtown South
Sutton Place
Turtle Bay
Tudor City
Stuyvesant Town
Flatiron
Hudson Yards
```

In [78]:
```
print(ny_venues.shape)
```

```
(3327, 7)
```

In [79]:
```
ny_venues.head(5)
```

Out[79]:

	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	Starbucks	40.877531	-73.905582	Coffee Shop
4	Marble Hill	40.876551	-73.91066	Dunkin'	40.877136	-73.906666	Donut Shop

Let's check how many venues were returned for each neighborhood

In [80]:
```
ny_venues.groupby('Neighborhood').count()
```

Out[80]:

	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Neighborhood						
Battery Park City	100	100	100	100	100	100
Carnegie Hill	100	100	100	100	100	100
Central Harlem	42	42	42	42	42	42
Chelsea	100	100	100	100	100	100
Chinatown	100	100	100	100	100	100
Civic Center	100	100	100	100	100	100
Clinton	100	100	100	100	100	100
East Harlem	43	43	43	43	43	43
East Village	100	100	100	100	100	100
Financial District	100	100	100	100	100	100
Flatiron	100	100	100	100	100	100
Gramercy	100	100	100	100	100	100
Greenwich Village	100	100	100	100	100	100
Hamilton Heights	62	62	62	62	62	62
Hudson Yards	76	76	76	76	76	76
Inwood	57	57	57	57	57	57
Lenox Hill	100	100	100	100	100	100
Lincoln Square	100	100	100	100	100	100
Little Italy	100	100	100	100	100	100
Lower East Side	59	59	59	59	59	59
Manhattan Valley	56	56	56	56	56	56
Manhattanville	41	41	41	41	41	41
Marble Hill	26	26	26	26	26	26
Midtown	100	100	100	100	100	100

	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Neighborhood						
Midtown South	100	100	100	100	100	100
Morningside Heights	41	41	41	41	41	41
Murray Hill	100	100	100	100	100	100
Noho	100	100	100	100	100	100
Roosevelt Island	30	30	30	30	30	30
Soho	100	100	100	100	100	100
Stuyvesant Town	18	18	18	18	18	18
Sutton Place	100	100	100	100	100	100
Tribeca	100	100	100	100	100	100
Tudor City	85	85	85	85	85	85
Turtle Bay	100	100	100	100	100	100
Upper East Side	100	100	100	100	100	100
Upper West Side	100	100	100	100	100	100
Washington Heights	91	91	91	91	91	91
West Village	100	100	100	100	100	100
Yorkville	100	100	100	100	100	100

Let's find out how many unique categories can be curated from all the returned venues

In [81]:
```
print('There are {} uniques categories.'.format(len(ny_venues['Venue Cat
egory'].unique())))
```

```
There are 339 uniques categories.
```

Analyze Each Neighborhood

In [82]:
```
# one hot encoding
ny_onehot = pd.get_dummies(ny_venues[['Venue Category']], prefix="", pre
fix_sep="")

# add neighborhood column back to dataframe
ny_onehot['Neighborhood'] = ny_venues['Neighborhood']

# move neighborhood column to the first column
fixed_columns = [ny_onehot.columns[-1]] + list(ny_onehot.columns[:-1])
ny_onehot = ny_onehot[fixed_columns]

ny_onehot.head()
```

Out[82]:

	Neighborhood	Accessories Store	Adult Boutique	Afghan Restaurant	African Restaurant	American Restaurant	Antique Shop	A
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 [83]:
```
ny_onehot.shape
```

Out[83]: (3327, 340)

Next, let's group rows by neighborhood and by taking the mean of the frequency of occurrence of each category

In [84]:
```
ny_grouped = ny_onehot.groupby('Neighborhood').mean().reset_index()
ny_grouped.head(10)
```

Out[84]:

	Neighborhood	Accessories Store	Adult Boutique	Afghan Restaurant	African Restaurant	American Restaurant	Antique Shop	A
0	Battery Park City	0.00	0.0	0.0	0.000000	0.010000	0.00	0.
1	Carnegie Hill	0.00	0.0	0.0	0.000000	0.010000	0.00	0.
2	Central Harlem	0.00	0.0	0.0	0.071429	0.047619	0.00	0.
3	Chelsea	0.00	0.0	0.0	0.000000	0.030000	0.01	0.
4	Chinatown	0.00	0.0	0.0	0.000000	0.040000	0.00	0.
5	Civic Center	0.00	0.0	0.0	0.000000	0.030000	0.01	0.
6	Clinton	0.00	0.0	0.0	0.000000	0.040000	0.00	0.
7	East Harlem	0.00	0.0	0.0	0.000000	0.000000	0.00	0.
8	East Village	0.00	0.0	0.0	0.000000	0.020000	0.01	0.
9	Financial District	0.01	0.0	0.0	0.000000	0.030000	0.00	0.

Let's confirm the new size

In [85]:
```
ny_grouped.shape
```

Out[85]: (40, 340)

Let's print each neighborhood along with the top 5 most common venues¶

In [86]:
```
num_top_venues = 5

for hood in ny_grouped['Neighborhood']:
    print("----"+hood+"----")
    temp = ny_grouped[ny_grouped['Neighborhood'] == hood].T.reset_index
()
    temp.columns = ['venue','freq']
    temp = temp.iloc[1:]
    temp['freq'] = temp['freq'].astype(float)
    temp = temp.round({'freq': 2})
    print(temp.sort_values('freq', ascending=False).reset_index(drop=Tru
e).head(num_top_venues))
    print('\n')
```

```
----Battery Park City----
           venue  freq
0           Park  0.08
1    Coffee Shop  0.06
2          Hotel  0.05
3  Memorial Site  0.04
4            Gym  0.04


----Carnegie Hill----
         venue  freq
0  Coffee Shop  0.06
1  Pizza Place  0.06
2          Bar  0.04
3         Café  0.04
4          Gym  0.03


----Central Harlem----
                  venue  freq
0   African Restaurant  0.07
1  Fried Chicken Joint  0.05
2   Chinese Restaurant  0.05
3                  Bar  0.05
4  American Restaurant  0.05


----Chelsea----
                venue  freq
0         Coffee Shop  0.06
1  Italian Restaurant  0.05
2      Ice Cream Shop  0.05
3           Nightclub  0.04
4              Bakery  0.04


----Chinatown----
                    venue  freq
0      Chinese Restaurant  0.09
1            Cocktail Bar  0.05
2  Vietnamese Restaurant  0.04
3      Salon / Barbershop  0.04
4     American Restaurant  0.04


----Civic Center----
                   venue  freq
0  Gym / Fitness Center  0.05
1    Italian Restaurant  0.04
2        Sandwich Place  0.04
3                 Hotel  0.04
4     French Restaurant  0.04


----Clinton----
                   venue  freq
0               Theater  0.12
```

```
1  Gym / Fitness Center  0.05
2    Italian Restaurant  0.04
3   American Restaurant  0.04
4                 Hotel  0.04


----East Harlem----
                        venue  freq
0          Mexican Restaurant  0.12
1                      Bakery  0.09
2               Deli / Bodega  0.07
3                         Spa  0.05
4  Latin American Restaurant  0.05


----East Village----
                 venue  freq
0                  Bar  0.06
1             Wine Bar  0.05
2  Mexican Restaurant  0.04
3  Chinese Restaurant  0.04
4         Pizza Place  0.04


----Financial District----
          venue  freq
0  Coffee Shop  0.08
1   Steakhouse  0.04
2        Hotel  0.04
3    Wine Shop  0.04
4          Gym  0.04


----Flatiron----
                       venue  freq
0                        Gym  0.05
1                Yoga Studio  0.04
2  New American Restaurant  0.04
3      American Restaurant  0.04
4      Japanese Restaurant  0.04


----Gramercy----
                  venue  freq
0                   Bar  0.07
1   Italian Restaurant  0.04
2  American Restaurant  0.04
3          Pizza Place  0.04
4           Bagel Shop  0.04


----Greenwich Village----
                 venue  freq
0  Italian Restaurant  0.13
1      Clothing Store  0.04
2    Sushi Restaurant  0.04
3  Seafood Restaurant  0.03
```

```
4  Chinese Restaurant  0.03


----Hamilton Heights----
                venue  freq
0                Café  0.06
1  Mexican Restaurant  0.06
2         Pizza Place  0.06
3         Coffee Shop  0.05
4                Park  0.03


----Hudson Yards----
                  venue  freq
0   American Restaurant  0.07
1    Italian Restaurant  0.05
2  Gym / Fitness Center  0.05
3                  Café  0.05
4                 Hotel  0.04


----Inwood----
                venue  freq
0  Mexican Restaurant  0.09
1                Café  0.07
2              Bakery  0.05
3       Deli / Bodega  0.05
4         Pizza Place  0.05


----Lenox Hill----
                  venue  freq
0           Coffee Shop  0.07
1    Italian Restaurant  0.05
2     Sushi Restaurant  0.05
3           Pizza Place  0.05
4  Gym / Fitness Center  0.03


----Lincoln Square----
                  venue  freq
0  Gym / Fitness Center  0.06
1               Theater  0.06
2          Concert Hall  0.05
3                 Plaza  0.05
4                  Café  0.05


----Little Italy----
                venue  freq
0              Bakery  0.05
1                Café  0.04
2  Italian Restaurant  0.03
3  Salon / Barbershop  0.03
4      Clothing Store  0.03
```

```
----Lower East Side----
                  venue  freq
0           Coffee Shop  0.07
1                  Café  0.05
2    Chinese Restaurant  0.05
3           Pizza Place  0.05
4      Ramen Restaurant  0.05


----Manhattan Valley----
                 venue  freq
0          Pizza Place  0.05
1          Coffee Shop  0.05
2    Indian Restaurant  0.05
3          Yoga Studio  0.04
4                 Café  0.04


----Manhattanville----
                  venue  freq
0    Italian Restaurant  0.05
1    Seafood Restaurant  0.05
2    Mexican Restaurant  0.05
3           Coffee Shop  0.05
4          Liquor Store  0.05


----Marble Hill----
               venue  freq
0     Sandwich Place  0.12
1        Coffee Shop  0.08
2     Discount Store  0.08
3        Yoga Studio  0.04
4    Supplement Shop  0.04


----Midtown----
              venue  freq
0             Hotel  0.07
1    Clothing Store  0.04
2      Cocktail Bar  0.04
3       Coffee Shop  0.04
4           Theater  0.04


----Midtown South----
                   venue  freq
0      Korean Restaurant  0.14
1                  Hotel  0.07
2    Japanese Restaurant  0.04
3              Hotel Bar  0.04
4         Cosmetics Shop  0.04


----Morningside Heights----
                   venue  freq
0              Bookstore  0.07
```

```
1                 Park  0.07
2  American Restaurant  0.07
3          Coffee Shop  0.07
4       Sandwich Place  0.05


----Murray Hill----
                 venue  freq
0          Coffee Shop  0.05
1  Japanese Restaurant  0.04
2       Sandwich Place  0.04
3                Hotel  0.04
4   Italian Restaurant  0.03


----Noho----
                venue  freq
0  Italian Restaurant  0.06
1   French Restaurant  0.05
2    Sushi Restaurant  0.04
3        Cocktail Bar  0.04
4           Gift Shop  0.03


----Roosevelt Island----
              venue  freq
0       Coffee Shop  0.07
1    Sandwich Place  0.07
2          Bus Line  0.03
3               Gym  0.03
4  Greek Restaurant  0.03


----Soho----
            venue  freq
0  Clothing Store  0.10
1        Boutique  0.06
2     Art Gallery  0.04
3   Women's Store  0.04
4      Shoe Store  0.04


----Stuyvesant Town----
            venue  freq
0      Playground  0.11
1             Bar  0.11
2            Park  0.11
3  Baseball Field  0.06
4    Cocktail Bar  0.06


----Sutton Place----
                    venue  freq
0    Gym / Fitness Center  0.06
1      Italian Restaurant  0.04
2       Indian Restaurant  0.04
3  Furniture / Home Store  0.04
```

```
4             Pizza Place  0.03


----Tribeca----
                 venue  freq
0   Italian Restaurant  0.05
1                  Spa  0.05
2                 Café  0.05
3                 Park  0.05
4  American Restaurant  0.04


----Tudor City----
                venue  freq
0  Mexican Restaurant  0.06
1    Greek Restaurant  0.06
2                Park  0.06
3         Pizza Place  0.05
4                Café  0.05


----Turtle Bay----
                venue  freq
0  Italian Restaurant  0.06
1         Coffee Shop  0.05
2    Sushi Restaurant  0.05
3          Steakhouse  0.05
4            Wine Bar  0.04


----Upper East Side----
                venue  freq
0  Italian Restaurant  0.08
1             Exhibit  0.07
2         Art Gallery  0.05
3              Bakery  0.04
4         Coffee Shop  0.04


----Upper West Side----
                venue  freq
0  Italian Restaurant  0.06
1            Wine Bar  0.04
2                 Bar  0.04
3      Cosmetics Shop  0.03
4   Indian Restaurant  0.03


----Washington Heights----
               venue  freq
0               Café  0.05
1      Deli / Bodega  0.04
2      Grocery Store  0.04
3  Mobile Phone Shop  0.04
4             Bakery  0.04
```

```
----West Village----
                    venue  freq
0      Italian Restaurant  0.09
1          Cosmetics Shop  0.05
2  New American Restaurant  0.05
3                    Park  0.04
4                Wine Bar  0.04


----Yorkville----
               venue  freq
0  Italian Restaurant  0.06
1                 Gym  0.06
2         Coffee Shop  0.06
3                 Bar  0.05
4   Sushi Restaurant  0.04
```

Let's put that into a pandas dataframe

In [87]:
```
#a function to sort the venues in descending order

def return_most_common_venues(row, num_top_venues):
    row_categories = row.iloc[1:]
    row_categories_sorted = row_categories.sort_values(ascending=False)

    return row_categories_sorted.index.values[0:num_top_venues]
```

In [88]:
```
#create the new dataframe and display the top 10 venues for each neighbo
rhood.
num_top_venues = 10
word_string3 = ''
indicators = ['st', 'nd', 'rd']

# create columns according to number of top venues
columns = ['Neighborhood']
for ind in np.arange(num_top_venues):
    try:
        columns.append('{}{} Most Common Venue'.format(ind+1, indicators
[ind]))
    except:
        columns.append('{}th Most Common Venue'.format(ind+1))

# create a new dataframe
neighborhoods_venues_sorted = pd.DataFrame(columns=columns)
neighborhoods_venues_sorted['Neighborhood'] = ny_grouped['Neighborhood']

for ind in np.arange(ny_grouped.shape[0]):
    neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venue
s(ny_grouped.iloc[ind, :], num_top_venues)
    word_string3 = word_string3 + neighborhoods_venues_sorted.iloc[ind,
1:] + ' '

neighborhoods_venues_sorted.head()
```

Out[88]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue
0	Battery Park City	Park	Coffee Shop	Hotel	Gym	Memorial Site	Italian Restaurant
1	Carnegie Hill	Pizza Place	Coffee Shop	Bar	Café	Yoga Studio	Grocery Store
2	Central Harlem	African Restaurant	Public Art	Cosmetics Shop	American Restaurant	Bar	Seafood Restaurant
3	Chelsea	Coffee Shop	Italian Restaurant	Ice Cream Shop	Nightclub	Bakery	Art Gallery
4	Chinatown	Chinese Restaurant	Cocktail Bar	American Restaurant	Vietnamese Restaurant	Salon / Barbershop	Ice Cream Shop

Vizualization with word cloud

In [89]:
```
stopwords = set(STOPWORDS)
```

In [90]:
```
word_string4 = ''
for i in range(0, num_top_venues ):
    word_string4 = word_string4 + word_string3[i]
```

In [92]:
```
# create the word cloud
stopwords.add('Restaurant')
wordcloud = WordCloud(background_color='white', stopwords = stopwords).g
enerate(word_string4)

print('Word cloud created!')

# display the cloud
fig = plt.figure()
fig.set_figwidth(12)
fig.set_figheight(15)

plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

```
Word cloud created!
```

4. Cluster Neighborhoods

Run k-means to cluster the neighborhood into 5 clusters.

In [93]:
```
# set number of clusters
kclusters = 5

ny_grouped_clustering = ny_grouped.drop('Neighborhood', 1)

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

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:10]
```

Out[93]:
```
array([2, 1, 1, 2, 1, 2, 2, 0, 1, 1], dtype=int32)
```

Let's create a new dataframe that includes the cluster as well as the top 10 venues for each neighborhood.

In [94]:
```
# add clustering labels
neighborhoods_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)

ny_merged = ny_data

# merge toronto_grouped with toronto_data to add latitude/longitude for
 each neighborhood
ny_merged = ny_merged.join(neighborhoods_venues_sorted.set_index('Neighb
orhood'), on='Neighborhood')

ny_merged.head() # check the last columns!
```

Out[94]:

	Borough	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd M Com Ve
0	Manhattan	Marble Hill	40.876551	-73.910660	4	Sandwich Place	Discount Store	Coffee Shop
1	Manhattan	Chinatown	40.715618	-73.994279	1	Chinese Restaurant	Cocktail Bar	Americ Restau
2	Manhattan	Washington Heights	40.851903	-73.936900	0	Café	Grocery Store	Deli / Bodeg
3	Manhattan	Inwood	40.867684	-73.921210	0	Mexican Restaurant	Café	Loung
4	Manhattan	Hamilton Heights	40.823604	-73.949688	0	Mexican Restaurant	Café	Pizza Place

Finally, let's visualize the resulting clusters

In [95]:
```
# 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(ny_merged['Latitude'], ny_merged['Long
itude'], ny_merged['Neighborhood'], ny_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
```

Examining the clusters

Let's try and see each cluster and the most common venue among each.

In [96]:
```
# For Cluster 0
result = ny_merged.loc[ny_merged['Cluster Labels'] == 0, ny_merged.colum
ns[[1] + list(range(5, ny_merged.shape[1]))]]
print("For cluster {}, the distribution of venues is as:\n{}".format(0,
result['1st Most Common Venue'].value_counts()))
result
```

```
For cluster 0, the distribution of venues is as:
Mexican Restaurant    4
Café                  1
Italian Restaurant    1
Coffee Shop           1
Name: 1st Most Common Venue, dtype: int64
```

Out[96]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	C
2	Washington Heights	Café	Grocery Store	Deli / Bodega	Mobile Phone Shop	Bakery	Gym	Ta R
3	Inwood	Mexican Restaurant	Café	Lounge	Deli / Bodega	Bakery	Pizza Place	R
4	Hamilton Heights	Mexican Restaurant	Café	Pizza Place	Coffee Shop	Yoga Studio	Park	S Pl
5	Manhattanville	Italian Restaurant	Park	Mexican Restaurant	Seafood Restaurant	Coffee Shop	Liquor Store	C R
7	East Harlem	Mexican Restaurant	Bakery	Deli / Bodega	Latin American Restaurant	Spa	Thai Restaurant	C
25	Manhattan Valley	Coffee Shop	Pizza Place	Indian Restaurant	Yoga Studio	Mexican Restaurant	Thai Restaurant	D B
36	Tudor City	Mexican Restaurant	Park	Greek Restaurant	Pizza Place	Café	Deli / Bodega	H

In [97]:
```
# For Cluster 1
result = ny_merged.loc[ny_merged['Cluster Labels'] == 1, ny_merged.colum
ns[[1] + list(range(5, ny_merged.shape[1]))]]
print("For cluster {}, the distribution of venues is as:\n{}".format(1,
result['1st Most Common Venue'].value_counts()))
result
```

```
For cluster 1, the distribution of venues is as:
Coffee Shop            4
Italian Restaurant     3
Bar                    2
Gym / Fitness Center   1
Korean Restaurant      1
Pizza Place            1
Chinese Restaurant     1
African Restaurant     1
Sandwich Place         1
Name: 1st Most Common Venue, dtype: int64
```

Out[97]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Mo Comm Ven
1	Chinatown	Chinese Restaurant	Cocktail Bar	American Restaurant	Vietnamese Restaurant	Salon / Barbershop	Ice Crea Shop
6	Central Harlem	African Restaurant	Public Art	Cosmetics Shop	American Restaurant	Bar	Seafood Restaur
9	Yorkville	Italian Restaurant	Coffee Shop	Gym	Bar	Pizza Place	Sushi Restaur
10	Lenox Hill	Coffee Shop	Italian Restaurant	Pizza Place	Sushi Restaurant	Burger Joint	Gym
11	Roosevelt Island	Sandwich Place	Coffee Shop	Dry Cleaner	Gym / Fitness Center	Gym	Greek Restaur
12	Upper West Side	Italian Restaurant	Wine Bar	Bar	Vegetarian / Vegan Restaurant	Mediterranean Restaurant	Bakery
16	Murray Hill	Coffee Shop	Sandwich Place	Hotel	Japanese Restaurant	French Restaurant	Bar
19	East Village	Bar	Wine Bar	Chinese Restaurant	Mexican Restaurant	Ice Cream Shop	Pizza Place
20	Lower East Side	Coffee Shop	Chinese Restaurant	Café	Ramen Restaurant	Pizza Place	Japanes Restaur
27	Gramercy	Bar	Italian Restaurant	American Restaurant	Pizza Place	Bagel Shop	Cocktail Bar
29	Financial District	Coffee Shop	Hotel	Gym	Wine Shop	Steakhouse	Cocktail Bar
30	Carnegie Hill	Pizza Place	Coffee Shop	Bar	Café	Yoga Studio	Grocery Store
33	Midtown South	Korean Restaurant	Hotel	Dessert Shop	Japanese Restaurant	Hotel Bar	Cosmeti Shop
34	Sutton Place	Gym / Fitness Center	Italian Restaurant	Furniture / Home Store	Indian Restaurant	Juice Bar	Gym

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Mo Comm Ven
35	Turtle Bay	Italian Restaurant	Coffee Shop	Steakhouse	Sushi Restaurant	Wine Bar	Ramen Restaur

In [98]:
```
# For Cluster 2
result = ny_merged.loc[ny_merged['Cluster Labels'] == 2, ny_merged.colum
ns[[1] + list(range(5, ny_merged.shape[1]))]]
print("For cluster {}, the distribution of venues is as:\n{}".format(2,
result['1st Most Common Venue'].value_counts()))
result
```

```
For cluster 2, the distribution of venues is as:
Italian Restaurant      5
Park                    2
Theater                 2
Bakery                  1
Hotel                   1
Coffee Shop             1
Clothing Store          1
Gym                     1
Gym / Fitness Center    1
American Restaurant     1
Name: 1st Most Common Venue, dtype: int64
```

Out[98]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue
8	Upper East Side	Italian Restaurant	Exhibit	Art Gallery	Bakery	Coffee Shop	Gym / Fitness Center
13	Lincoln Square	Theater	Gym / Fitness Center	Café	Plaza	Concert Hall	Italian Restaurant
14	Clinton	Theater	Gym / Fitness Center	Italian Restaurant	American Restaurant	Hotel	Wine Shop
15	Midtown	Hotel	Coffee Shop	Cocktail Bar	Clothing Store	Theater	Sporting Goods Shop
17	Chelsea	Coffee Shop	Italian Restaurant	Ice Cream Shop	Nightclub	Bakery	Art Gallery
18	Greenwich Village	Italian Restaurant	Sushi Restaurant	Clothing Store	Chinese Restaurant	Cosmetics Shop	Café
21	Tribeca	Italian Restaurant	Spa	Park	Café	American Restaurant	Boutique
22	Little Italy	Bakery	Café	Italian Restaurant	Bubble Tea Shop	Clothing Store	Cocktail Bar
23	Soho	Clothing Store	Boutique	Art Gallery	Shoe Store	Women's Store	Italian Restaurant
24	West Village	Italian Restaurant	Cosmetics Shop	New American Restaurant	Park	Jazz Club	Wine Bar
26	Morningside Heights	Park	Coffee Shop	American Restaurant	Bookstore	Food Truck	Tennis Court
28	Battery Park City	Park	Coffee Shop	Hotel	Gym	Memorial Site	Italian Restaurant
31	Noho	Italian Restaurant	French Restaurant	Sushi Restaurant	Cocktail Bar	Bookstore	Grocery Store
32	Civic Center	Gym / Fitness Center	Italian Restaurant	Coffee Shop	French Restaurant	Hotel	Sandwich Place

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue
38	Flatiron	Gym	Yoga Studio	American Restaurant	Clothing Store	Gym / Fitness Center	Japanese Restaurant
39	Hudson Yards	American Restaurant	Italian Restaurant	Café	Gym / Fitness Center	Hotel	Spanish Restaurant

In [99]:

```
# For Cluster 3
result = ny_merged.loc[ny_merged['Cluster Labels'] == 3, ny_merged.colum
ns[[1] + list(range(5, ny_merged.shape[1]))]]
print("For cluster {}, the distribution of venues is as:\n{}".format(3,
result['1st Most Common Venue'].value_counts()))
result
```

```
For cluster 3, the distribution of venues is as:
Bar    1
Name: 1st Most Common Venue, dtype: int64
```

Out[99]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Com V
37	Stuyvesant Town	Bar	Park	Playground	Pet Service	Gas Station	Boat or Ferry	Germ Resta

In [100]:

```
#For Cluster 4
#result = ny_merged.loc[ny_merged['Cluster Labels'] == 4, ny_merged.colu
mns[[1] + list(range(5, ny_merged.shape[1]))]]
#print("For cluster {}, the distribution of venues is as:\n{}".format(4,
result['1st Most Common Venue'].value_counts()))
#result
```

Let's explore the Kensington and Chelsea, uk in our dataframe.

Get the neighborhood's name.

In [101]:
```
address = 'Kensington and Chelsea, uk'

geolocator = Nominatim(user_agent="uk_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geograpical coordinate of Kensington and Chelsea, uk {}, {}.'
.format(latitude, longitude))
```

```
The geograpical coordinate of Kensington and Chelsea, uk 51.4989948, -
0.1991229.
```

In [102]:
```
LIMIT = 200 # limit of number of venues returned by Foursquare API
radius = 1500 # define radius
url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client
_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    latitude,
    longitude,
    radius,
    LIMIT)
url # display URL
```

Out[102]:
```
'https://api.foursquare.com/v2/venues/explore?&client_id=B3D1FREXU3FMFK
G0XFFFWLZH1UBNQKQGVTG4XWBI3N32354V&client_secret=UAFKLDYGA1SQEBZYO4P5DY
UAS4DBRF5QA53DURWY03FTRQP3&v=20180604&ll=51.4989948,-0.1991229&radius=1
500&limit=200'
```

In [103]:
```
results1 = requests.get(url).json()
#results1
```

In [104]:
```
# function that extracts the category of the venue
def get_category_type(row):
    try:
        categories_list = row['categories']
    except:
        categories_list = row['venue.categories']

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

In [105]:
```
venues = results1['response']['groups'][0]['items']

nearby_venues = json_normalize(venues) # flatten JSON

# filter columns
filtered_columns = ['venue.name', 'venue.categories', 'venue.location.la
t', 'venue.location.lng']
nearby_venues =nearby_venues.loc[:, filtered_columns]

# filter the category for each row
nearby_venues['venue.categories'] = nearby_venues.apply(get_category_typ
e, axis=1)

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

nearby_venues.head()
```

Out[105]:

	name	categories	lat	lng
0	Core Collective	Gym / Fitness Center	51.499589	-0.198630
1	The Design Museum	Museum	51.499785	-0.199641
2	Café Phillies	Café	51.499726	-0.197747
3	Leighton House Museum	History Museum	51.498591	-0.203118
4	The Scarsdale Tavern	Pub	51.496975	-0.199024

In [106]:
```
print('{} venues were returned by Foursquare.'.format(nearby_venues.shap
e[0]))
```

```
100 venues were returned by Foursquare.
```

In [107]:
```
def getNearbyVenues(names, latitudes, longitudes, radius=1500):

    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 [108]:
```
uk_venues = getNearbyVenues(names=new_data['neighborhoods'],
                                   latitudes=new_data['Latitude'],
                                   longitudes=new_data['Longitude'])
```

```
Brompton
Chelsea
Earls Court
Kensington
South Kensington
West Brompton
Holland Park
North Kensington
Notting Hill
```

In [109]:
```
print(uk_venues.shape)
```

```
(900, 7)
```

In [110]:
```
uk_venues.head()
```

Out[110]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Catego
0	Brompton	51.49014	-0.16248	Venchi	51.489239	-0.164265	Ice Crea Shop
1	Brompton	51.49014	-0.16248	Saturday Farmers' Market	51.490917	-0.160329	Farmers Market
2	Brompton	51.49014	-0.16248	Duke of York Square	51.491272	-0.159827	Plaza
3	Brompton	51.49014	-0.16248	The Five Fields	51.491770	-0.161191	Restaura
4	Brompton	51.49014	-0.16248	Amorino	51.489455	-0.163803	Ice Crea Shop

In [111]:
```
uk_venues.groupby('Neighborhood').count()
```

Out[111]:

	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Neighborhood						
Brompton	100	100	100	100	100	100
Chelsea	100	100	100	100	100	100
Earls Court	100	100	100	100	100	100
Holland Park	100	100	100	100	100	100
Kensington	100	100	100	100	100	100
North Kensington	100	100	100	100	100	100
Notting Hill	100	100	100	100	100	100
South Kensington	100	100	100	100	100	100
West Brompton	100	100	100	100	100	100

In [112]:
```
print('There are {} uniques categories.'.format(len(uk_venues['Venue Cat
egory'].unique())))
```

```
There are 135 uniques categories.
```

In [113]:
```
# one hot encoding
uk_onehot = pd.get_dummies(uk_venues[['Venue Category']], prefix="", pre
fix_sep="")

# add neighborhood column back to dataframe
uk_onehot['Neighborhood'] = uk_venues['Neighborhood']

# move neighborhood column to the first column
fixed_columns = [uk_onehot.columns[-1]] + list(uk_onehot.columns[:-1])
uk_onehot = uk_onehot[fixed_columns]

uk_onehot.head()
```

Out[113]:

	Neighborhood	American Restaurant	Antique Shop	Argentinian Restaurant	Art Gallery	Art Museum	Asian Restaurant	Austra Restau
0	Brompton	0	0	0	0	0	0	0
1	Brompton	0	0	0	0	0	0	0
2	Brompton	0	0	0	0	0	0	0
3	Brompton	0	0	0	0	0	0	0
4	Brompton	0	0	0	0	0	0	0

In [114]:
```
uk_onehot.shape
```

Out[114]: (900, 136)

In [115]:
```
uk_grouped = uk_onehot.groupby('Neighborhood').mean().reset_index()
uk_grouped
```

Out[115]:

	Neighborhood	American Restaurant	Antique Shop	Argentinian Restaurant	Art Gallery	Art Museum	Asian Restaurant	Austr Restau
0	Brompton	0.01	0.00	0.01	0.02	0.01	0.00	0.00
1	Chelsea	0.01	0.00	0.01	0.02	0.01	0.00	0.00
2	Earls Court	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	Holland Park	0.00	0.00	0.00	0.01	0.00	0.00	0.00
4	Kensington	0.00	0.00	0.01	0.02	0.01	0.00	0.00
5	North Kensington	0.00	0.00	0.00	0.00	0.00	0.01	0.00
6	Notting Hill	0.00	0.01	0.00	0.02	0.00	0.01	0.01
7	South Kensington	0.00	0.00	0.01	0.02	0.01	0.00	0.00
8	West Brompton	0.01	0.00	0.00	0.00	0.00	0.00	0.00

In [116]:
```
num_top_venues = 5

for hood in uk_grouped['Neighborhood']:
    print("----"+hood+"----")
    temp = uk_grouped[uk_grouped['Neighborhood'] == hood].T.reset_index
()
    temp.columns = ['venue','freq']
    temp = temp.iloc[1:]
    temp['freq'] = temp['freq'].astype(float)
    temp = temp.round({'freq': 2})
    print(temp.sort_values('freq', ascending=False).reset_index(drop=Tru
e).head(num_top_venues))
    print('\n')
```

```
----Brompton----
               venue  freq
0               Café  0.07
1  French Restaurant  0.05
2     Ice Cream Shop  0.04
3             Bakery  0.04
4              Hotel  0.03


----Chelsea----
               venue  freq
0               Café  0.07
1  French Restaurant  0.05
2     Ice Cream Shop  0.04
3             Bakery  0.04
4              Hotel  0.03


----Earls Court----
                  venue  freq
0                 Hotel  0.08
1    Italian Restaurant  0.05
2           Pizza Place  0.05
3  Gym / Fitness Center  0.04
4                   Pub  0.04


----Holland Park----
                venue  freq
0                 Pub  0.07
1              Bakery  0.04
2  Italian Restaurant  0.04
3               Hotel  0.04
4          Restaurant  0.04


----Kensington----
                 venue  freq
0   Italian Restaurant  0.08
1                 Café  0.07
2                Hotel  0.06
3       Science Museum  0.04
4  Japanese Restaurant  0.03


----North Kensington----
                  venue  freq
0                   Pub  0.10
1  Gym / Fitness Center  0.06
2    Italian Restaurant  0.06
3                Bakery  0.05
4          Cocktail Bar  0.04


----Notting Hill----
                  venue  freq
0                   Pub  0.08
```

```
1    Italian Restaurant  0.06
2  Gym / Fitness Center  0.05
3                Bakery  0.04
4                  Café  0.03


----South Kensington----
                  venue  freq
0    Italian Restaurant  0.08
1                  Café  0.07
2                 Hotel  0.06
3        Science Museum  0.04
4   Japanese Restaurant  0.03


----West Brompton----
                venue  freq
0  Italian Restaurant  0.07
1                Café  0.06
2              Bakery  0.05
3                 Pub  0.04
4               Hotel  0.04
```

In [117]:

```
#a function to sort the venues in descending order

def return_most_common_venues(row, num_top_venues):
    row_categories = row.iloc[1:]
    row_categories_sorted = row_categories.sort_values(ascending=False)

    return row_categories_sorted.index.values[0:num_top_venues]
```

In [118]:
```
#create the new dataframe and display the top 10 venues for each neighbo
rhood.
num_top_venues = 10
word_string1 = ''
indicators = ['st', 'nd', 'rd']

# create columns according to number of top venues
columns = ['Neighborhood']
for ind in np.arange(num_top_venues):
    try:
        columns.append('{}{} Most Common Venue'.format(ind+1, indicators
[ind]))
    except:
        columns.append('{}th Most Common Venue'.format(ind+1))

# create a new dataframe
neighborhoods_venues_sorted_uk = pd.DataFrame(columns=columns)
neighborhoods_venues_sorted_uk['Neighborhood'] = uk_grouped['Neighborhoo
d']

for ind in np.arange(uk_grouped.shape[0]):
    neighborhoods_venues_sorted_uk.iloc[ind, 1:] = return_most_common_ve
nues(uk_grouped.iloc[ind, :], num_top_venues)
    word_string1 = word_string1 + neighborhoods_venues_sorted_uk.iloc[in
d, 1:] + ' '

neighborhoods_venues_sorted_uk.head()
```

Out[118]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7 C
0	Brompton	Café	French Restaurant	Ice Cream Shop	Bakery	Japanese Restaurant	Italian Restaurant	Co Ba
1	Chelsea	Café	French Restaurant	Ice Cream Shop	Bakery	Japanese Restaurant	Italian Restaurant	Co Ba
2	Earls Court	Hotel	Pizza Place	Italian Restaurant	Pub	Gym / Fitness Center	Thai Restaurant	Ba
3	Holland Park	Pub	Bakery	Hotel	Café	Italian Restaurant	Restaurant	Enį Re:
4	Kensington	Italian Restaurant	Café	Hotel	Science Museum	Japanese Restaurant	Burger Joint	Ga

In [119]:
```
word_string2 = ''
for i in range(0, num_top_venues ):
    word_string2 = word_string2 + word_string1[i]
#print (word_string2)
```

In [120]:
```
# display the generated text
stopwords.add('Restaurant')
#word_string

# create the word cloud
wordcloud = WordCloud(background_color='white',stopwords = stopwords).ge
nerate(word_string2)

print('Word cloud created!')

# display the cloud
fig = plt.figure()
fig.set_figwidth(14)
fig.set_figheight(18)

plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

```
Word cloud created!
```

In [121]:
```
# set number of clusters
kclusters = 4

uk_grouped_clustering = uk_grouped.drop('Neighborhood', 1)

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

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:10]
```

Out[121]:
```
array([2, 2, 3, 3, 1, 0, 0, 1, 3], dtype=int32)
```

In [122]:
```
uk_merged = new_data
# add clustering labels
uk_merged['Cluster Labels'] = kmeans.labels_

uk_merged = uk_merged.join(neighborhoods_venues_sorted_uk.set_index('Nei
ghborhood'), on='neighborhoods')

uk_merged.head() # check the last columns!
```

Out[122]:

	neighborhoods	Dial code	borough	posttown	postcode	Latitude	Longitude	Clust Labe
0	Brompton	020	Kensington and Chelsea	LONDON	SW3	51.49014	-0.16248	2
1	Chelsea	020	Kensington and Chelsea	LONDON	SW3	51.49014	-0.16248	2
2	Earls Court	020	Kensington and Chelsea	LONDON	SW5	51.49004	-0.18971	3
3	Kensington	020	Kensington and Chelsea	LONDON	SW7	51.49807	-0.17404	3
4	South Kensington	020	Kensington and Chelsea	LONDON	SW7	51.49807	-0.17404	1

In [124]:
```
# 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(uk_merged['Latitude'], uk_merged['Long
itude'], uk_merged['neighborhoods'], uk_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
```

In [125]:
```
# For Cluster 0
result = uk_merged.loc[uk_merged['Cluster Labels'] == 0, uk_merged.colum
ns[[1] + list(range(5, uk_merged.shape[1]))]]
print("For cluster {}, the distribution of venues is as:\n{}".format(0,
result['1st Most Common Venue'].value_counts()))
result
```

```
For cluster 0, the distribution of venues is as:
Italian Restaurant    1
Pub                   1
Name: 1st Most Common Venue, dtype: int64
```

Out[125]:

	Dial code	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5 C
5	020	51.48563	-0.18144	0	Italian Restaurant	Café	Bakery	Pub	Ho
6	020	51.50162	-0.19173	0	Pub	Bakery	Hotel	Café	Ital Re

In [126]:
```
# For Cluster 1
result = uk_merged.loc[uk_merged['Cluster Labels'] == 1, uk_merged.colum
ns[[1] + list(range(5, uk_merged.shape[1]))]]
print("For cluster {}, the distribution of venues is as:\n{}".format(1,
result['1st Most Common Venue'].value_counts()))
result
```

```
For cluster 1, the distribution of venues is as:
Italian Restaurant    1
Pub                   1
Name: 1st Most Common Venue, dtype: int64
```

Out[126]:

	Dial code	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5 C
4	020	51.49807	-0.17404	1	Italian Restaurant	Café	Hotel	Science Museum	Ja Re
7	020	51.52346	-0.21353	1	Pub	Gym / Fitness Center	Italian Restaurant	Bakery	Pi Pl

In [127]:
```
# For Cluster 2
result = uk_merged.loc[uk_merged['Cluster Labels'] == 2, uk_merged.colum
ns[[1] + list(range(5, uk_merged.shape[1]))]]
print("For cluster {}, the distribution of venues is as:\n{}".format(2,
result['1st Most Common Venue'].value_counts()))
result
```

```
For cluster 2, the distribution of venues is as:
Café    2
Name: 1st Most Common Venue, dtype: int64
```

Out[127]:

	Dial code	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5 C
0	020	51.49014	-0.16248	2	Café	French Restaurant	Ice Cream Shop	Bakery	Jap Res
1	020	51.49014	-0.16248	2	Café	French Restaurant	Ice Cream Shop	Bakery	Jap Res

In [128]:
```
# For Cluster 3
result = uk_merged.loc[uk_merged['Cluster Labels'] == 3, uk_merged.colum
ns[[1] + list(range(5, uk_merged.shape[1]))]]
print("For cluster {}, the distribution of venues is as:\n{}".format(3,
result['1st Most Common Venue'].value_counts()))
result
```

```
For cluster 3, the distribution of venues is as:
Pub                   1
Italian Restaurant    1
Hotel                 1
Name: 1st Most Common Venue, dtype: int64
```

Out[128]:

	Dial code	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	
2	020	51.49004	-0.18971	3	Hotel	Pizza Place	Italian Restaurant	Pub	C F C
3	020	51.49807	-0.17404	3	Italian Restaurant	Café	Hotel	Science Museum	J F
8	020	51.51244	-0.20639	3	Pub	Italian Restaurant	Gym / Fitness Center	Bakery	F

Entertainment in NYC

In [129]:
```
#address = '102 North End Ave, New York, NY'
address = '575 5th Ave, New York'

geolocator = Nominatim(user_agent="foursquare_agent")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print(latitude, longitude)
```

```
40.7563907 -73.9782057
```

In [130]:
```
#let's define a query to search for entertainment that is within 2000 me
tres in nyc.
search_query = 'Escape Games'
radius = 2000
print(search_query + ' .... OK!')

# Define the corresponding URL
url = 'https://api.foursquare.com/v2/venues/search?client_id={}&client_s
ecret={}&ll={},{}&v={}&query={}&radius={}&limit={}'.format(CLIENT_ID, CL
IENT_SECRET, latitude, longitude, VERSION, search_query, radius, LIMIT)
url
```

```
Escape Games .... OK!
```

Out[130]:
```
'https://api.foursquare.com/v2/venues/search?client_id=B3D1FREXU3FMFKG0
XFFFWLZH1UBNQKQGVTG4XWBI3N32354V&client_secret=UAFKLDYGA1SQEBZYO4P5DYUA
S4DBRF5QA53DURWY03FTRQP3&ll=40.7563907,-73.9782057&v=20180604&query=Esc
ape Games&radius=2000&limit=200'
```

In [131]:
```
#  Send the GET Request and examine the results
results = requests.get(url).json()
#results
```

In [133]:
```
# assign relevant part of JSON to venues
venues = results['response']['venues']

# tranform venues into a dataframe
dataframe = json_normalize(venues)
#dataframe.head()
```

In [134]:
```
# keep only columns that include venue name, and anything that is associ
ated with location
filtered_columns = ['name', 'categories'] + [col for col in dataframe.co
lumns if col.startswith('location.')]
dataframe_filtered = dataframe.loc[:, filtered_columns]

# function that extracts the category of the venue
def get_category_type(row):
    try:
        categories_list = row['categories']
    except:
        categories_list = row['venue.categories']

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

# filter the category for each row
dataframe_filtered['categories'] = dataframe_filtered.apply(get_category
_type, axis=1)

# clean column names by keeping only last term
dataframe_filtered.columns = [column.split('.')[-1] for column in datafr
ame_filtered.columns]

dataframe_filtered.head()
```

Out[134]:

	name	categories	address	cc	city	country	crossStreet	distance	formattedAc
0	Mission Escape Games	General Entertainment	265 W 37th St	US	New York	United States	NaN	1172	[265 W 37th New York, N 10018, Unite Sta...
1	Riddle Me Out Escape Games NYC	General Entertainment	435 5th Avenue, 4th Floor	US	New York	United States	NaN	656	[435 5th Aver 4th Floor, Ne York, NY 100
2	Escape Day Spa & Skin Care	Spa	101 W 55th St	US	New York	United States	6th Ave, NW corner	767	[101 W 55th Ave, NW corn New York,...
3	Exit Escape Room NYC	General Entertainment	246 W 38th St Fl 7	US	New York	United States	NaN	1058	[246 W 38th 7, New York, 10018, Unite
4	BBDO NY - Escape Hatch	Office	1285 Avenue of the Americas	US	New York	United States	W 51st St	318	[1285 Avenue the Americas 51st St), New

In [135]:
```
dataframe_filtered1 = dataframe_filtered[dataframe_filtered.categories =
='General Entertainment']
dataframe_filtered1.head()
```

Out[135]:

	name	categories	address	cc	city	country	crossStreet	distance	formatt
0	Mission Escape Games	General Entertainment	265 W 37th St	US	New York	United States	NaN	1172	[265 W New Yo 10018, Sta...
1	Riddle Me Out Escape Games NYC	General Entertainment	435 5th Avenue, 4th Floor	US	New York	United States	NaN	656	[435 5th 4th Floo York, NY
3	Exit Escape Room NYC	General Entertainment	246 W 38th St Fl 7	US	New York	United States	NaN	1058	[246 W 7, New 10018,
6	Escape Room Madness	General Entertainment	38 West 32nd Street, 5th Floor, Ste 500	US	New York	United States	32nd street & Broadway	1212	[38 Wes Street, 5 Ste 500
7	Escape Entertainment	General Entertainment	4th Floor, 39 W 32nd St	US	New York	United States	NaN	1214	[4th Flo 32nd St NY 100

In [136]:
```
venues_ett_map = folium.Map(location=[latitude, longitude], zoom_start=1
3) # generate map centred around the Conrad Hotel

# add the Italian restaurants as blue circle markers
for lat, lng, label in zip(dataframe_filtered.lat, dataframe_filtered.ln
g, dataframe_filtered.categories):
    folium.features.CircleMarker(
        [lat, lng],
        radius=3,
        color='blue',
        popup=label,
        fill = True,
        fill_color='blue',
        fill_opacity=0.6
    ).add_to(venues_ett_map)

# add the Italian restaurants as blue circle markers
for lat, lng, label in zip(dataframe_filtered1.lat, dataframe_filtered1.
lng, dataframe_filtered1.categories):
    folium.features.CircleMarker(
        [lat, lng],
        radius=3,
        color='red',
        popup=label,
        fill = True,
        fill_color='red',
        fill_opacity=0.6
    ).add_to(venues_ett_map)


# display map
#venues_ett_map
```

Results

Analyzing the results we can see that people in different borough of London and NYC often visit identical places, such as Italian Restaurant, Coffee Shop, Park, Pizza Place, Hotel, Gym, Fitness Center, But there are also differences in preferences, such as French Restaurant, Pub, Japanese Restaurant, Cocktail Bar, Boutique for borough of London and for borough of NYC - American Restaurant, Wine Shop, Chinese Restaurant, Sushi Restaurant, Taco Place. Also using Foursquare API and visualization we can easily see the information that we need, for example, the placement of Escape room. They are popular now, and how we can see on map there are only a few in Manhattan.

Discussion

Based on our result, we can conclude that there are few Escape rooms and if we want to open a Escape room then the best place in Manhattan is near the center and above. If to analyze Brooklyn we can see only 2 of them. That is a good decision and we are independ from choose a place at this moment.

Also analyzing area of London and New York, we see that the British prefer French Restaurant, Pub, Japanese Restaurant while the American prefer Mexican Restaurant, Chinese Restaurant, Sushi Restaurant, Taco Place.

Conclusion

Using Foursquare API, we can captured data of common places all around the world. Using it, we refer back to our main objectives, which is to determine;

the similarity or dissimilarirty of both cities classification of area located inside the city whether it is residential, tourism places, or others. Using visualization libraries we can do different graphs for easy understanding of the material.