Part 1 - A description of the problem and a discussion of the background.

1. Problem Description

Delhi and Mumbai are two major cities in India. Both cities become a center of attention for residential, job employment, tourism, education, shopping and sports activity and are well known in India, thus becoming a best choice for local and foreign communities.

Both cities are best options for both foreign and local people to settle down, then these places could be aimed by any restaurant industry to establish and grow. In order to better make a decision about where a restaurant can be statrted, we need to analyze the food preferences of people living in the most demandin neighborhoods of Delhi and Mumbai.

The real deal is that as much as there are many fine restaurants in Delhi – Asian, Middle Eastern, Latin and American restaurants, you can struggle to find good place to dine in the finest of Mumbai that has combination of same cuisine options.

- **Delhi:** India’s capital territory, is a massive metropolitan area in the country’s north. In Old Delhi, a neighborhood dating to the 1600s, stands the imposing Mughal-era Red Fort, a symbol of India, and the sprawling Jama Masjid mosque, whose courtyard accommodates 25,000 people. Nearby is Chandni Chowk, a vibrant bazaar filled with food carts, sweets shops and spice stallsm with total ares of 1,484 km² and population 18.98 million as of 2012.

- **Mumbai:** Also known as Bombay is the capital city of the Indian state of Maharashtra. A financial center, it's India's largest city. On the Mumbai Harbour waterfront stands the iconic Gateway of India stone arch, built by the British Raj in 1924. Offshore, nearby Elephanta Island holds ancient cave temples dedicated to the Hindu god Shiva. The city's also famous as the heart of the Bollywood film industry. As of 2011 it is the most populous city in India with an estimated city proper population of 12.4 million.

2. Discussion of the Background

My client, a successful restaurant chain in Dubai is looking to expand operation into India through the most populous cities-Delhi and Mumbai. They want to create a high-end restaurant that comes with organic mix and healthy. Their target is not only to establish their restaurant chain in India, but also to provide the citizens, pro-organic and healthy eating. To them every meal counts and counts as a royal when you eat.

Since the Indian demography is so big, my client needs deeper insight from available data in other to decide where to establish their first restaurant. This company spends a lot on research and provides customers with data insight into the ingredients used at restaurants.

3. Target Audience

Considering the diversity of India, there is a high multicultural sense. India is a place where different shades live. As such, in the search for an high-end Indian-inclined restaurant, there is a high shortage.

Part 2 - A description of the data and how it will be used

to solve the problem

1. Data

The data is gathered using wikipedia and is organized into csv file for easier mainpulation. The csv files are attached to this project.

- https://github.com/pareshprakash/capstone-project/blob/master/Delhi-District.csv
- https://github.com/pareshprakash/capstone-project/blob/master/Mumbai-District.csv

One should keep in mind that the amount and accuracy of data captured using Four Square API cannot be 100%.

Dataset 1: Delhi

Delhi consists of 7 districst with 63 locations.

In [1]:

```
#import required libraries
import numpy as np
import pandas as pd

#read data for Delhi
df_delhi = pd.read_csv(r"C:\Users\shiva\Downloads\Delhi-District.csv")
df_delhi.head()
```

Out[1]:

	Pincode	Location	District	State
0	110001	Baroda House	CENTRAL DELHI	DELHI
1	110001	Bengali Market	CENTRAL DELHI	DELHI
2	110001	Bhagat Singh Market	CENTRAL DELHI	DELHI
3	110001	Connaught Place	CENTRAL DELHI	DELHI
4	110001	Constitution House	CENTRAL DELHI	DELHI

In [2]:

```
#examine data
print("Delhi dataframe has {} district and {} locations".format(
        len(df_delhi['District'].unique()),
        df_delhi.shape[0]
    )
)

#grouping data to find District with highest number of area
df_delhi.groupby('District').count()
```

```
Delhi dataframe has 7 district and 73 locations
```

Out[2]:

District	Pincode	Location	State
CENTRAL DELHI	5	5	5
EAST DELHI	7	7	7
NORTH DELHI	12	12	12
NORTH EAST DELHI	12	12	12
SOUTH DELHI	7	7	7
SOUTH WEST DELHI	20	20	20
WEST DELHI	10	10	10

Dataset 2: Mumbai

Mumbai consists 12 districts with 114 locations.

In [3]:

```
#read data for Mumbai
df_mumbai = pd.read_csv(r"C:\Users\shiva\Downloads\Mumbai-District.csv")
df_mumbai.head()
```

Out[3]:

	Pincode	Location	District	State
0	400004	Ambewadi	Ambewadi	Maharashtra
1	400004	Charni Road	Ambewadi	Maharashtra
2	400004	Chaupati	Ambewadi	Maharashtra
3	400004	Girgaon	Ambewadi	Maharashtra
4	400004	Madhavbaug	Ambewadi	Maharashtra

In [4]:

```
#examine data
print("Mumbai dataframe has {} district and {} locations".format(
        len(df_mumbai['District'].unique()),
        df_mumbai.shape[0]
    )
)

#grouping data to find District with highest number of area
df_mumbai.groupby('District').count()
```

```
Mumbai dataframe has 12 district and 114 locations
```

Out[4]:

District	Pincode	Location	State
Ambewadi	6	6	6
Andheri	7	7	7
Bhawani Shankar	10	10	10
Churchgate	16	16	16
Colaba	9	9	9
Goregaon	9	9	9
Malad	10	10	10
Mumbai Central	9	9	9
Mumbai East	17	17	17
Navi Mumbai	6	6	6
Parel Naka	6	6	6
Wadala	9	9	9

2. How data will be used to solve the problem

The data from the datasets 1 and 2 will be explored by considering the venues within the neighbourhood of London Postcode areas. These areas' restaurants would be checked in terms of the types of restaurants within a certain mile radius. Due to Foursquare restrictions, the number of venues will be limited to 100 venues. The proximity to transport connection and other amenities would be correlated. Also, accessibility and ease of supplies of organic ingredients would be considered.

In [5]:

```
!pip install geocoder
#import geocoder to add latitudes and longtitudes to each district
print('geocoder has been installed before.')
import geocoder
print('geocoder has been successfully imported.')
```

```
Requirement already satisfied: geocoder in c:\users\shiva\anaconda3\lib\site
-packages (1.38.1)
Requirement already satisfied: ratelim in c:\users\shiva\anaconda3\lib\site-
packages (from geocoder) (0.1.6)
Requirement already satisfied: six in c:\users\shiva\anaconda3\lib\site-pack
ages (from geocoder) (1.12.0)
Requirement already satisfied: future in c:\users\shiva\anaconda3\lib\site-p
ackages (from geocoder) (0.17.1)
Requirement already satisfied: requests in c:\users\shiva\anaconda3\lib\site
-packages (from geocoder) (2.21.0)
Requirement already satisfied: click in c:\users\shiva\anaconda3\lib\site-pa
ckages (from geocoder) (7.0)
Requirement already satisfied: decorator in c:\users\shiva\anaconda3\lib\sit
e-packages (from ratelim->geocoder) (4.3.0)
Requirement already satisfied: urllib3<1.25,>=1.21.1 in c:\users\shiva\anaco
nda3\lib\site-packages (from requests->geocoder) (1.24.1)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\shiva\anaconda
3\lib\site-packages (from requests->geocoder) (2018.11.29)
Requirement already satisfied: idna<2.9,>=2.5 in c:\users\shiva\anaconda3\li
b\site-packages (from requests->geocoder) (2.8)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in c:\users\shiva\anaco
nda3\lib\site-packages (from requests->geocoder) (3.0.4)
geocoder has been installed before.
geocoder has been successfully imported.
```

In [6]:

```
#function to get latitude and longitude for India
def get_latlng(postal_code):
    # initialize your variable to None
    lat_lng_coords = None
    # loop until you get the coordinates
    while(lat_lng_coords is None):
        g = geocoder.arcgis('{}, India'.format(postal_code))
        lat_lng_coords = g.latlng
    return lat_lng_coords

get_latlng('M4G')
```

Out[6]:

```
[23.379379735000043, 79.44332654800007]
```

In [7]:

```
#put new column of latitude and logitude into dataframe of Delhi
postal_delhi_codes = df_delhi['Location']
coords = [ get_latlng(postal_code) for postal_code in postal_delhi_codes.tolist() ]

#add latitude and longtitude to Delhi
df_delhi_coords = pd.DataFrame(coords, columns=['Latitude', 'Longitude'])
df_delhi['Latitude'] = df_delhi_coords['Latitude']
df_delhi['Longitude'] = df_delhi_coords['Longitude']
df_delhi.head(10)
```

Out[7]:

	Pincode	Location	District	State	Latitude	Longitude
0	110001	Baroda House	CENTRAL DELHI	DELHI	28.61648	77.22925
1	110001	Bengali Market	CENTRAL DELHI	DELHI	28.62919	77.23216
2	110001	Bhagat Singh Market	CENTRAL DELHI	DELHI	28.97528	77.71057
3	110001	Connaught Place	CENTRAL DELHI	DELHI	28.63396	77.21979
4	110001	Constitution House	CENTRAL DELHI	DELHI	-33.92413	18.42088
5	110005	Election Commission	SOUTH DELHI	DELHI	30.74108	76.77884
6	110005	Anand Parbat Indl. Area	SOUTH DELHI	DELHI	28.66585	77.17347
7	110005	Anand Parbat	SOUTH DELHI	DELHI	28.66585	77.17347
8	110005	Bank Street	SOUTH DELHI	DELHI	43.65962	-70.25125
9	110005	Desh Bandhu Gupta Road	SOUTH DELHI	DELHI	28.64519	77.21281

In [8]:

```
#add latitude and longtitude to dataframe of Mumbai
postal_codes_mumbai = df_mumbai['Location']
coords = [ get_latlng(postal_code) for postal_code in postal_codes_mumbai.tolist() ]

df_mumbai_coords = pd.DataFrame(coords, columns=['Latitude', 'Longitude'])
df_mumbai['Latitude'] = df_mumbai_coords['Latitude']
df_mumbai['Longitude'] = df_mumbai_coords['Longitude']
df_mumbai.head(10)
```

Out[8]:

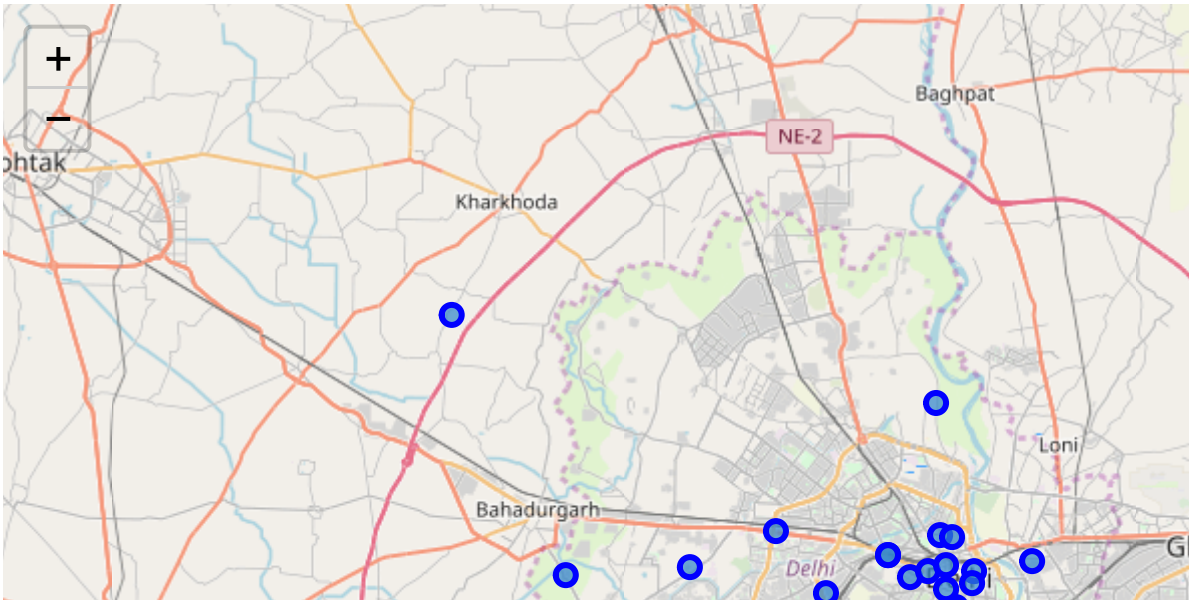
	Pincode	Location	District	State	Latitude	Longitude
0	400004	Ambewadi	Ambewadi	Maharashtra	18.01874	76.94887
1	400004	Charni Road	Ambewadi	Maharashtra	18.95719	72.82477
2	400004	Chaupati	Ambewadi	Maharashtra	21.18535	72.80715
3	400004	Girgaon	Ambewadi	Maharashtra	16.44751	74.52361
4	400004	Madhavbaug	Ambewadi	Maharashtra	23.07582	72.56212
5	400004	Opera House	Ambewadi	Maharashtra	21.23580	72.86974
6	400052	Danda	Andheri	Maharashtra	24.12784	83.94542
7	400052	Khar Colony	Andheri	Maharashtra	30.70523	76.24135
8	400052	Khar Delivery	Andheri	Maharashtra	18.52061	73.85731
9	400052	Andheri	Andheri	Maharashtra	30.38924	77.12491

In [54]:

```
from geopy.geocoders import Nominatim
import folium

address = 'Delhi, India'
geolocator = Nominatim()
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude

# create map of New York using latitude and longitude values
map_delhi = folium.Map(location=[latitude, longitude], zoom_start=10)

# add markers to map
for lat, lng, borough, neighborhood in zip(df_delhi['Latitude'], df_delhi['Longitude'], df_
    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_delhi)

map_delhi
```

```
C:\Users\shiva\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: Deprecat
ionWarning: Using Nominatim with the default "geopy/1.19.0" `user_agent` is
strongly discouraged, as it violates Nominatim's ToS https://operations.osmf
oundation.org/policies/nominatim/ (https://operations.osmfoundation.org/poli
cies/nominatim/) and may possibly cause 403 and 429 HTTP errors. Please spec
ify a custom `user_agent` with `Nominatim(user_agent="my-application")` or b
y overriding the default `user_agent`: `geopy.geocoders.options.default_user
_agent = "my-application"`. In geopy 2.0 this will become an exception.
  """
```

Out[54]:



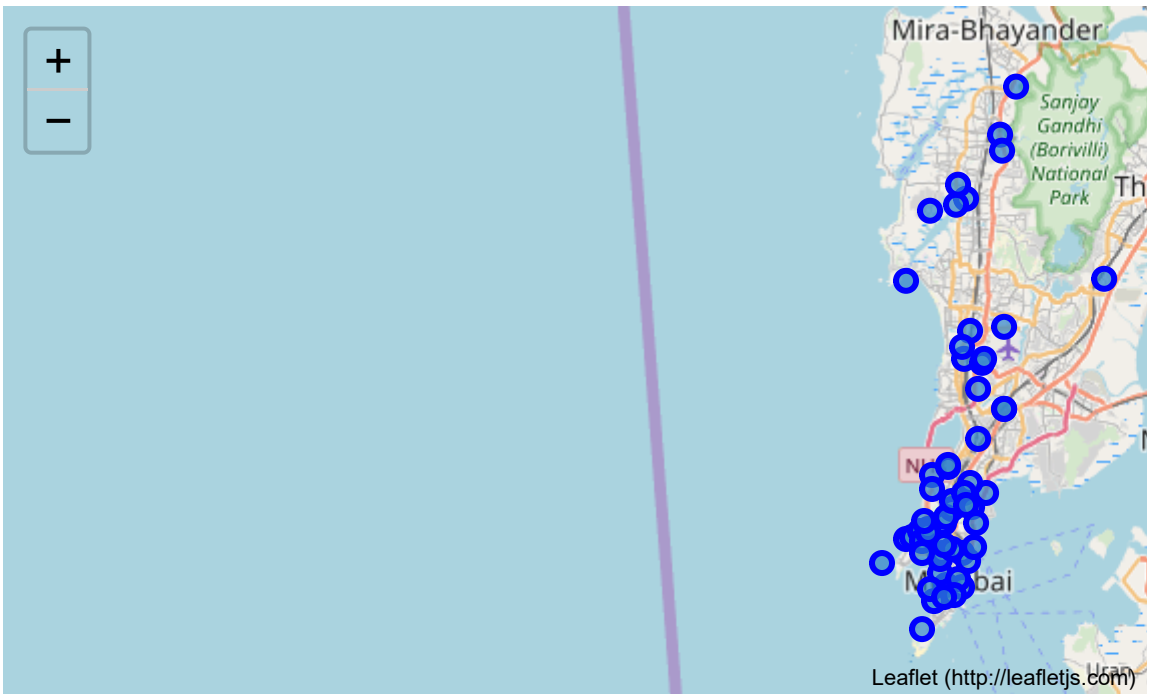

Jhajjar
New Delhi
Leaflet (http://leafletjs.com)

In [10]:

```
address = 'Mumbai, India'
geolocator = Nominatim()
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude

# create map of New York using latitude and longitude values
map_mumbai = folium.Map(location=[latitude, longitude], zoom_start=10)

# add markers to map
for lat, lng, borough, neighborhood in zip(df_mumbai['Latitude'], df_mumbai['Longitude'], d
    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_mumbai)

map_mumbai
```

```
C:\Users\shiva\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: Deprecat
ionWarning: Using Nominatim with the default "geopy/1.19.0" `user_agent` is
strongly discouraged, as it violates Nominatim's ToS https://operations.osmf
oundation.org/policies/nominatim/ (https://operations.osmfoundation.org/poli
cies/nominatim/) and may possibly cause 403 and 429 HTTP errors. Please spec
ify a custom `user_agent` with `Nominatim(user_agent="my-application")` or b
y overriding the default `user_agent`: `geopy.geocoders.options.default_user
_agent = "my-application"`. In geopy 2.0 this will become an exception.
```

Out[10]:



3. Methodology

The addresses found above were converted into their equivalent latitude and longitude coordinates. Now, the Foursquare API will be used to explore neighborhoods in both cities of Delhi and Mumbai. After that, explore function to get the most common venue categories in each neighborhood, and then this feature can be used to group the neighborhoods into clusters via K-means clustering algorithm. And also, the Folium library will be then used to visualize the neighborhoods in Delhi and Mumbai and their emerging clusters.

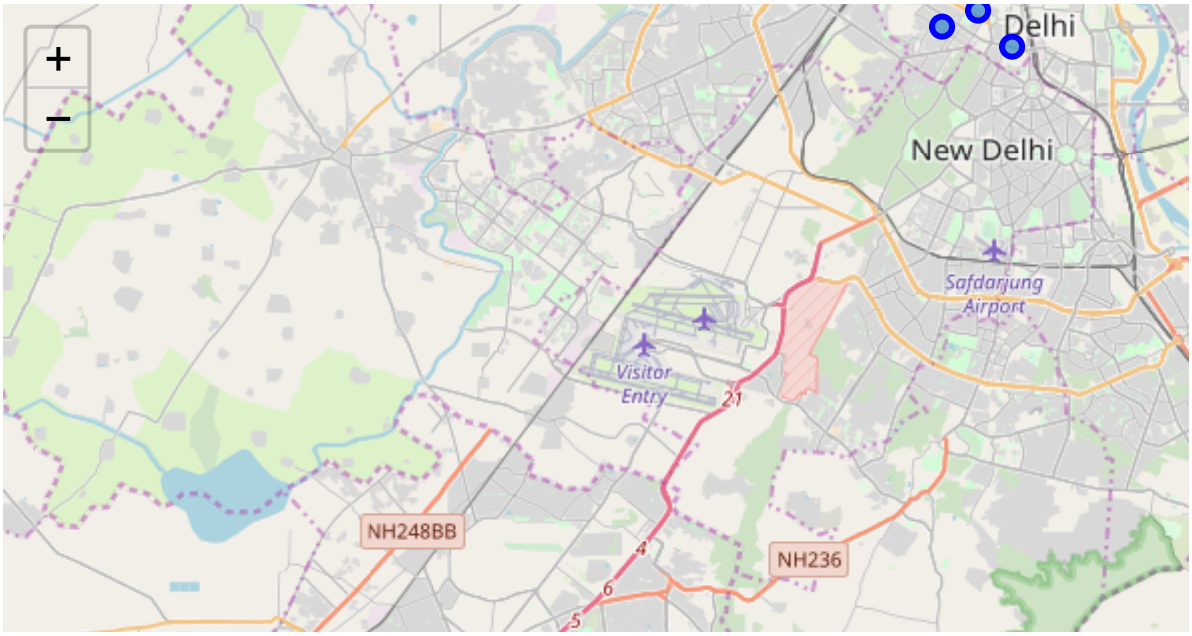
Based on dataframe analysis above, we found out that South Delhi area in Delhi and Mumbai Central area in Mumbai are both have the highest number of area within it those district.

In [11]:

```
#slice the original dataframe and create a new dataframe of the South Delhi
sDelhi = df_delhi[df_delhi['District'] == 'SOUTH DELHI'].reset_index(drop=True)

#get the geographical coordinates of Bukit Bintang, Kuala Lumpur
address = 'SOUTH DELHI, India'
geolocator = Nominatim()
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude

# create map of Bukit Bintang using latitude and longitude values
map_sDelhi = folium.Map(location=[latitude, longitude], zoom_start=11)

# add markers to map
for lat, lng, label in zip(sDelhi['Latitude'], sDelhi['Longitude'], sDelhi['Location']):
    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_sDelhi)

map_sDelhi
```

```
C:\Users\shiva\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: Deprecat
ionWarning: Using Nominatim with the default "geopy/1.19.0" `user_agent` is
strongly discouraged, as it violates Nominatim's ToS https://operations.osmf
oundation.org/policies/nominatim/ (https://operations.osmfoundation.org/poli
cies/nominatim/) and may possibly cause 403 and 429 HTTP errors. Please spec
ify a custom `user_agent` with `Nominatim(user_agent="my-application")` or b
y overriding the default `user_agent`: `geopy.geocoders.options.default_user
_agent = "my-application"`. In geopy 2.0 this will become an exception.
```

Out[11]:



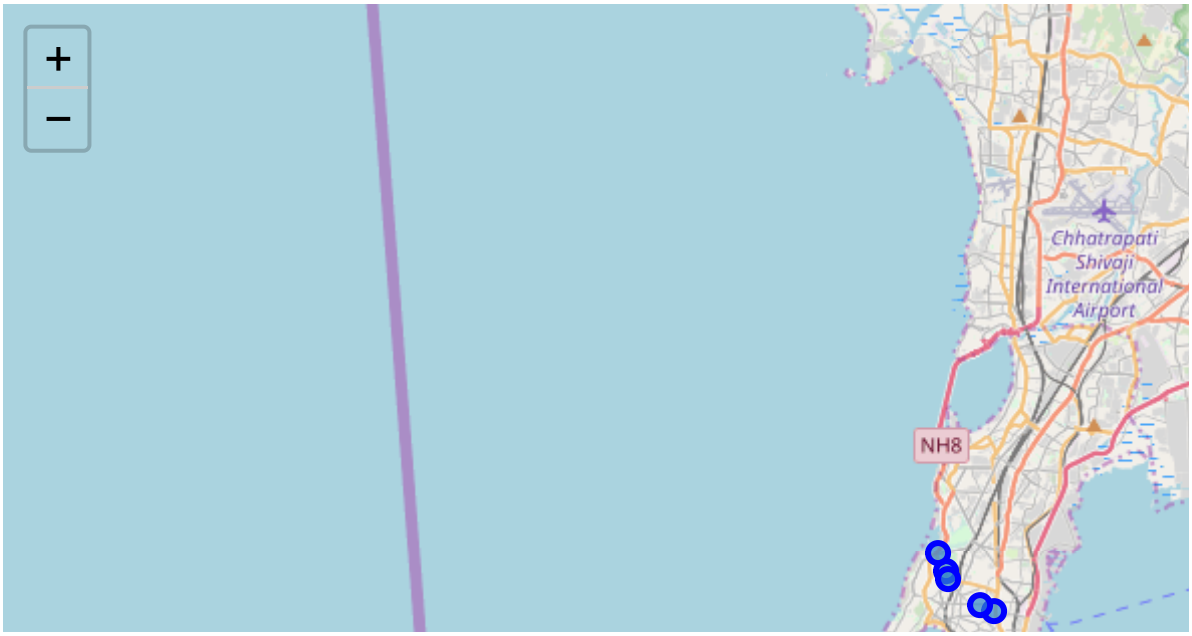

Gurugram
9
10
Asola Bhati
Leaflet (http://leafletjs.com)

In [12]:

```
#slice the original dataframe and create a new dataframe of the Jacob Circle
mCentral = df_mumbai[df_mumbai['District'] == 'Mumbai Central'].reset_index(drop=True)

#get the geographical coordinates of Manhattan
address = 'Mumbai Central, India'
geolocator = Nominatim()
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude

# create map of Bukit Bintang using latitude and longitude values
map_mCentral = folium.Map(location=[latitude, longitude], zoom_start=11)

# add markers to map
for lat, lng, label in zip(mCentral['Latitude'], mCentral['Longitude'], mCentral['Location'
    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_mCentral)

map_mCentral
```

```
C:\Users\shiva\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: Deprecat
ionWarning: Using Nominatim with the default "geopy/1.19.0" `user_agent` is
strongly discouraged, as it violates Nominatim's ToS https://operations.osmf
oundation.org/policies/nominatim/ (https://operations.osmfoundation.org/poli
cies/nominatim/) and may possibly cause 403 and 429 HTTP errors. Please spec
ify a custom `user_agent` with `Nominatim(user_agent="my-application")` or b
y overriding the default `user_agent`: `geopy.geocoders.options.default_user
_agent = "my-application"`. In geopy 2.0 this will become an exception.
```

Out[12]:



Using Foursquare API to get venues at surounding area of both Jama Masjid, Delhi and Jacob Circle, Mumbai.

In [13]:

```
import requests # library to handle requests
from pandas.io.json import json_normalize # tranform JSON file into a pandas dataframe

#Foursquare Credentials and Version
CLIENT_ID = 'OHI3T00DYQCL20NHSX3AS1LGEC4KDKNKWWTBRQBH23BJAESC'
CLIENT_SECRET = 'CHTMCPDHRQD1KXZWP3NBCP2MBAZHCZKNC24W0XBJXHN03PV2'
VERSION = '20180605'

#explore the first neighborhood in our dataframe
#Get the neighborhood's latitude and longitude values.
neighborhood_latitude = sDelhi.loc[0, 'Latitude'] # neighborhood latitude value
neighborhood_longitude = sDelhi.loc[0, 'Longitude'] # neighborhood longitude value
neighborhood_name = sDelhi.loc[0, 'Location'] # neighborhood name

#get the top 100 venues that are in Bukit Bintang within a radius of 500 meters
LIMIT = 100 # limit of number of venues returned by Foursquare API
radius = 500 # define radius
url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll=
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    neighborhood_latitude,
    neighborhood_longitude,
    radius,
    LIMIT)

#Send the GET request and examine the resutls
results = requests.get(url).json()

#borrow the get_category_type function from the Foursquare lab.
# 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']

#clean the json and structure it into a pandas dataframe
venues = results['response']['groups'][0]['items']
nearby_venues = json_normalize(venues) # flatten JSON

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

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

# clean columns
nearby_venues.columns = [col.split(".")[-1] for col in nearby_venues.columns]
print('{} venues were returned by Foursquare for South Delhi, Delhi'.format(nearby_venues.s
nearby_venues.head()
```

```
10 venues were returned by Foursquare for South Delhi, Delhi
```

Out[13]:

	name	categories	lat	lng
0	Softy Corner	Ice Cream Shop	30.740414	76.781619
1	Sector 17	Miscellaneous Shop	30.739541	76.782158
2	Indian Coffee House	Coffee Shop	30.740343	76.780902
3	Hot Millions 2	Fast Food Restaurant	30.740557	76.782547
4	Ghazal	Indian Restaurant	30.739055	76.783358

In [14]:

```
#explore the first neighborhood in our dataframe
#Get the neighborhood's latitude and longitude values.
neighborhood_latitude = mCentral.loc[0, 'Latitude'] # neighborhood latitude value
neighborhood_longitude = mCentral.loc[0, 'Longitude'] # neighborhood longitude value
neighborhood_name = mCentral.loc[0, 'Location'] # neighborhood name

#get the top 100 venues that are in Marble Hill within a radius of 500 meters
LIMIT = 100 # limit of number of venues returned by Foursquare API
radius = 500 # define radius
url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll=
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    neighborhood_latitude,
    neighborhood_longitude,
    radius,
    LIMIT)

#Send the GET request and examine the resutls
results = requests.get(url).json()

#clean the json and structure it into a pandas dataframe
venues = results['response']['groups'][0]['items']
nearby_venues = json_normalize(venues) # flatten JSON

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

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

# clean columns
nearby_venues.columns = [col.split(".")[-1] for col in nearby_venues.columns]
print('{} venues were returned by Foursquare for Mumbai Central, Mumbai'.format(nearby_venu
nearby_venues.head()
```

```
3 venues were returned by Foursquare for Mumbai Central, Mumbai
```

Out[14]:

	name	categories	lat	lng
0	CPT Square	Plaza	8.535950	76.990825
1	State Bank Atm	ATM	8.534620	76.991714
2	Rajappan Fireworks	Fireworks Store	8.534635	76.997696

In [15]:

```
#function to repeat the same process to all area
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={}&
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT)

        # make the GET request
        results = requests.get(url).json()["response"]['groups'][0]['items']

        # 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 = ['Area',
                  'Area Latitude',
                  'Area Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']

    return(nearby_venues)

#run the above function on each neighborhood and create a new dataframe
sDelhi_venues = getNearbyVenues(names=sDelhi['Location'],
                                latitudes=sDelhi['Latitude'],
                                longitudes=sDelhi['Longitude']
                               )

#check the size of the resulting dataframe
print(sDelhi_venues.shape)
sDelhi_venues.head()
```

```
Election Commission
Anand Parbat Indl. Area
Anand Parbat
Bank Street
Desh Bandhu Gupta Road
Karol Bagh
```

```
Master Prithvi Nath Marg
(166, 7)
```

Out[15]:

	Area	Area Latitude	Area Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Election Commission	30.74108	76.77884	Softy Corner	30.740414	76.781619	Ice Cream Shop
1	Election Commission	30.74108	76.77884	Sector 17	30.739541	76.782158	Miscellaneous Shop
2	Election Commission	30.74108	76.77884	Indian Coffee House	30.740343	76.780902	Coffee Shop
3	Election Commission	30.74108	76.77884	Hot Millions 2	30.740557	76.782547	Fast Food Restaurant
4	Election Commission	30.74108	76.77884	Ghazal	30.739055	76.783358	Indian Restaurant

In [16]:

```
#run the above function on each neighborhood and create a new dataframe
mCentral_venues = getNearbyVenues(names=mCentral['Location'],
                                  latitudes=mCentral['Latitude'],
                                  longitudes=mCentral['Longitude']
                                 )

#check the size of the resulting dataframe
print(mCentral_venues.shape)
mCentral_venues.head()
```

```
Bharat Nagar
Grant Road
Swami Vivekand Road
Tardeo
Falkland Road
Kamathipura
Mumbai Central
Hajiali
Tulsiwadi
(73, 7)
```

Out[16]:

	Area	Area Latitude	Area Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Bharat Nagar	8.53727	76.99409	CPT Square	8.535950	76.990825	Plaza
1	Bharat Nagar	8.53727	76.99409	State Bank Atm	8.534620	76.991714	ATM
2	Bharat Nagar	8.53727	76.99409	Rajappan Fireworks	8.534635	76.997696	Fireworks Store
3	Grant Road	18.95929	72.83108	Taj Ice Cream	18.960013	72.830779	Ice Cream Shop
4	Grant Road	18.95929	72.83108	Shalimar Restaurant	18.958180	72.832367	Indian Restaurant

In [17]:

```
#check how many venues were returned for each area
print('There are {} uniques categories in Delhi'.format(len(sDelhi_venues['Venue Category']
sDelhi_venues.groupby('Area').count()
```

```
There are 72 uniques categories in Delhi
```

Out[17]:

Area	Area Latitude	Area Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Anand Parbat	4	4	4	4	4	4
Anand Parbat Indl. Area	4	4	4	4	4	4
Bank Street	100	100	100	100	100	100
Desh Bandhu Gupta Road	36	36	36	36	36	36
Election Commission	10	10	10	10	10	10
Karol Bagh	8	8	8	8	8	8
Master Prithvi Nath Marg	4	4	4	4	4	4

In [18]:

```
#check how many venues were returned for each area
print('There are {} uniques categories in Mumbai.'.format(len(mCentral_venues['Venue Catego
mCentral_venues.groupby('Area').count()
```

```
There are 34 uniques categories in Mumbai.
```

Out[18]:

Area	Area Latitude	Area Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Bharat Nagar	3	3	3	3	3	3
Falkland Road	1	1	1	1	1	1
Grant Road	10	10	10	10	10	10
Hajiali	14	14	14	14	14	14
Kamathipura	6	6	6	6	6	6
Mumbai Central	16	16	16	16	16	16
Swami Vivekand Road	2	2	2	2	2	2
Tardeo	17	17	17	17	17	17
Tulsiwadi	4	4	4	4	4	4

4. Anazlyze Delhi

In [40]:

```
address = 'Delhi, India'

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

```
The geograpical coordinate of Delhi are 28.6517178, 77.2219388.
```

In [19]:

```
# one hot encoding
sDelhi_onehot = pd.get_dummies(sDelhi_venues[['Venue Category']], prefix="", prefix_sep="")

# add neighborhood column back to dataframe
sDelhi_onehot['Area'] = sDelhi_venues['Area']

# move neighborhood column to the first column
fixed_columns = sDelhi_onehot.columns[-1] + sDelhi_onehot.columns[:-1]

#examine the new dataframe size after one hot encoding
print('{} rows were returned after one hot encoding.'.format(sDelhi_onehot.shape[0]))

#group rows by neighborhood and by taking the mean of the frequency of occurrence of each c
sDelhi_grouped = sDelhi_onehot.groupby('Area').mean().reset_index()

#examine the new dataframe size after one hot encoding
print('{} rows were returned after grouping.'.format(sDelhi_grouped.shape[0]))
```

```
166 rows were returned after one hot encoding.
7 rows were returned after grouping.
```

In [20]:

```
#print each neighborhood along with the top 5 most common venues
num_top_venues = 5

for hood in sDelhi_grouped['Area']:
    print("----"+hood+"----")
    temp = sDelhi_grouped[sDelhi_grouped['Area'] == 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=True).head(num_top_ver
    print('\n')
```

```
----Anand Parbat  ----
               venue  freq
0   Airport Terminal  0.25
1      Train Station  0.25
2  Convenience Store  0.25
3   Business Service  0.25
4  Korean Restaurant  0.00


----Anand Parbat Indl. Area  ----
               venue  freq
0   Airport Terminal  0.25
1      Train Station  0.25
2  Convenience Store  0.25
3   Business Service  0.25
4  Korean Restaurant  0.00


----Bank Street  ----
                venue  freq
0  Seafood Restaurant  0.07
1         Coffee Shop  0.05
2               Hotel  0.05
3  Italian Restaurant  0.05
4                 Bar  0.04


----Desh Bandhu Gupta Road  ----
         venue  freq
0        Hotel  0.44
1   Restaurant  0.08
2  Pizza Place  0.06
3         Café  0.06
4       Hostel  0.03


----Election Commission  ----
                venue  freq
0      Ice Cream Shop   0.2
1   Indian Restaurant   0.2
2  Miscellaneous Shop   0.1
3      Clothing Store   0.1
4               Café    0.1
```

```
----Karol Bagh  ----
                  venue  freq
0  Fast Food Restaurant  0.50
1           Snack Place  0.12
2                 Hotel  0.12
3     Indian Restaurant  0.12
4               Bakery  0.12


----Master Prithvi Nath Marg  ----
               venue  freq
0        Coffee Shop  0.25
1  Indian Restaurant  0.25
2   Asian Restaurant  0.25
3          Gift Shop  0.25
4             Lounge  0.00
```

In [21]:

```
#put into a pandas dataframe

#write 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]

#create the new dataframe and display the top 10 venues for each neighborhood
num_top_venues = 8

indicators = ['st', 'nd', 'rd']

# create columns according to number of top venues
columns = ['Area']
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
areas_venues_sorted = pd.DataFrame(columns=columns)
areas_venues_sorted['Area'] = sDelhi_grouped['Area']

for ind in np.arange(sDelhi_grouped.shape[0]):
    areas_venues_sorted.iloc[ind, 1:] = return_most_common_venues(sDelhi_grouped.iloc[ind,

areas_venues_sorted
```

Out[21]:

	Area	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 Cor
0	Anand Parbat	Train Station	Airport Terminal	Convenience Store	Business Service	Fast Food Restaurant	Coffee Shop	Comic
1	Anand Parbat Indl. Area	Train Station	Airport Terminal	Convenience Store	Business Service	Fast Food Restaurant	Coffee Shop	Comic
2	Bank Street	Seafood Restaurant	Hotel	Coffee Shop	Italian Restaurant	Bar	Ice Cream Shop	Am Rest
3	Desh Bandhu Gupta Road	Hotel	Restaurant	Café	Pizza Place	Hostel	Indian Restaurant	Cl Rest
4	Election Commission	Indian Restaurant	Ice Cream Shop	Clothing Store	Miscellaneous Shop	Café	Fast Food Restaurant	Coffee
5	Karol Bagh	Fast Food Restaurant	Snack Place	Hotel	Indian Restaurant	Bakery	Coffee Shop	Comic
6	Master Prithvi Nath Marg	Gift Shop	Coffee Shop	Asian Restaurant	Indian Restaurant	Fast Food Restaurant	Comic Shop	Conver

5. K-Means Clustering for North Delhi, Delhi

In [22]:

```
from sklearn.cluster import KMeans

# set number of clusters
kclusters = 3

sDelhi_grouped_clustering = sDelhi_grouped.drop('Area', 1)

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

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

# add clustering labels
areas_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)
```

In [23]:

```
#create a new dataframe that includes the cluster as well as the top 10 venues for each nei
sDelhi_merged = sDelhi

# merge toronto_grouped with toronto_data to add latitude/longitude for each neighborhood
sDelhi_merged = sDelhi_merged.join(areas_venues_sorted.set_index('Area'), on='Location')

sDelhi_merged
```

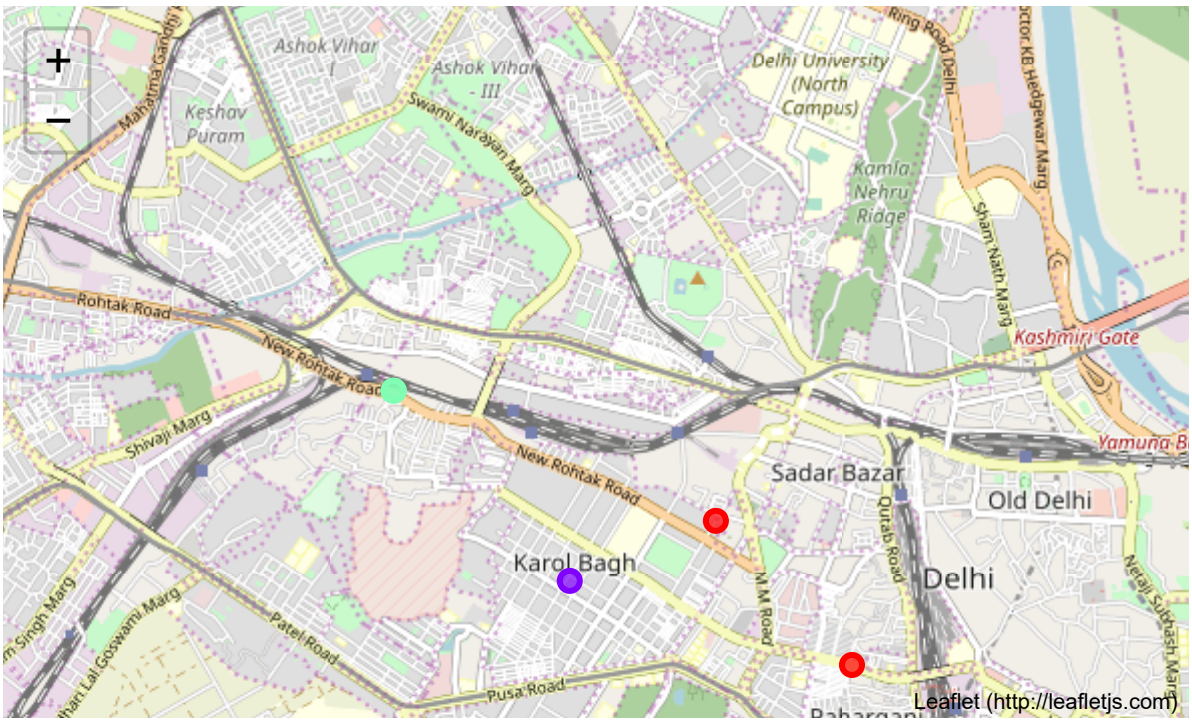
Out[23]:

	Pincode	Location	District	State	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue
0	110005	Election Commission	SOUTH DELHI	DELHI	30.74108	76.77884	0	Indian Restaurant	Ice Cream Shop
1	110005	Anand Parbat Indl. Area	SOUTH DELHI	DELHI	28.66585	77.17347	2	Train Station	Airport Terminal
2	110005	Anand Parbat	SOUTH DELHI	DELHI	28.66585	77.17347	2	Train Station	Airport Terminal
3	110005	Bank Street	SOUTH DELHI	DELHI	43.65962	-70.25125	0	Seafood Restaurant	Hotel
4	110005	Desh Bandhu Gupta Road	SOUTH DELHI	DELHI	28.64519	77.21281	0	Hotel	Restaurant
5	110005	Karol Bagh	SOUTH DELHI	DELHI	28.65156	77.18858	1	Fast Food Restaurant	Snack Place
6	110005	Master Prithvi Nath Marg	SOUTH DELHI	DELHI	28.65611	77.20108	0	Gift Shop	Coffee Shop

In [41]:

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

#Finally, let's visualize the resulting clusters
# create map 28.6517178, 77.2219388
sDelhi_clusters = folium.Map(location=[28.6517178, 77.2219388], 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(sDelhi_merged['Latitude'], sDelhi_merged['Longitude'], sD
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    cluster
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[(int)(cluster-1)],
        fill=True,
        fill_color=rainbow[(int)(cluster-1)],
        fill_opacity=0.7).add_to(sDelhi_clusters)

sDelhi_clusters
```

Out[41]:



6. Analyze Mumbai

In [42]:

```
address = 'Mumbai, India'

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

```
The geograpical coordinate of Mumbai are 18.9387711, 72.8353355.
```

In [25]:

```
# one hot encoding
mCentral_onehot = pd.get_dummies(mCentral_venues[['Venue Category']], prefix="", prefix_sep

# add neighborhood column back to dataframe
mCentral_onehot['Area'] = mCentral_venues['Area']

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

#examine the new dataframe size after one hot encoding
print('{} rows were returned after one hot encoding.'.format(mCentral_onehot.shape[0]))

#group rows by neighborhood and by taking the mean of the frequency of occurrence of each c
mCentral_grouped = mCentral_onehot.groupby('Area').mean().reset_index()

#examine the new dataframe size after one hot encoding
print('{} rows were returned after grouping.'.format(mCentral_grouped.shape[0]))
```

```
73 rows were returned after one hot encoding.
9 rows were returned after grouping.
```

In [26]:

```
#print each neighborhood along with the top 5 most common venues
num_top_venues = 5

for hood in mCentral_grouped['Area']:
    print("----"+hood+"----")
    temp = mCentral_grouped[mCentral_grouped['Area'] == 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=True).head(num_top_ver
    print('\n')
```

```
----Bharat Nagar----
             venue  freq
0              ATM  0.33
1            Plaza  0.33
2  Fireworks Store  0.33
3           Market  0.00
4      Outlet Mall  0.00


----Falkland Road----
       venue  freq
0   Pharmacy   1.0
1        ATM   0.0
2      Plaza   0.0
3  Juice Bar   0.0
4     Market   0.0


----Grant Road----
               venue  freq
0  Indian Restaurant   0.4
1       Dessert Shop   0.2
2             Arcade   0.1
3       Antique Shop   0.1
4     Ice Cream Shop   0.1


----Hajiali----
                venue  freq
0       Shopping Mall  0.14
1   Indian Restaurant  0.07
2       Deli / Bodega  0.07
3           Juice Bar  0.07
4  Italian Restaurant  0.07


----Kamathipura----
                 venue  freq
0    Indian Restaurant  0.50
1  Arts & Crafts Store  0.17
2            BBQ Joint  0.17
3         Antique Shop  0.17
4        Train Station  0.00
```

```
----Mumbai Central----
                  venue  freq
0  Fast Food Restaurant  0.12
1         Train Station  0.12
2    Chinese Restaurant  0.12
3     Indian Restaurant  0.06
4           Snack Place  0.06


----Swami Vivekand Road----
                        venue  freq
0                         ATM   0.5
1  Construction & Landscaping   0.5
2               Train Station   0.0
3                 Snack Place   0.0
4               Shopping Mall   0.0


----Tardeo----
                           venue  freq
0             Chinese Restaurant  0.18
1           Fast Food Restaurant  0.12
2                Sandwich Place  0.12
3  Vegetarian / Vegan Restaurant  0.06
4             Italian Restaurant  0.06


----Tulsiwadi----
            venue  freq
0  Sandwich Place  0.50
1       Pool Hall  0.25
2       Juice Bar  0.25
3           Plaza  0.00
4          Market  0.00
```

In [27]:

```
#create the new dataframe and display the top 10 venues for each neighborhood
num_top_venues = 8

indicators = ['st', 'nd', 'rd']

# create columns according to number of top venues
columns = ['Area']
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
areas_venues_sorted = pd.DataFrame(columns=columns)
areas_venues_sorted['Area'] = mCentral_grouped['Area']

for ind in np.arange(mCentral_grouped.shape[0]):
    areas_venues_sorted.iloc[ind, 1:] = return_most_common_venues(mCentral_grouped.iloc[inc

areas_venues_sorted.head()
```

Out[27]:

	Area	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 Mo Commc Venu
0	Bharat Nagar	ATM	Fireworks Store	Plaza	Arcade	Arts & Crafts Store	BBQ Joint	Beng Restaura
1	Falkland Road	Pharmacy	Vegetarian / Vegan Restaurant	Construction & Landscaping	Fireworks Store	Fast Food Restaurant	Dessert Shop	Departme Sto
2	Grant Road	Indian Restaurant	Dessert Shop	Ice Cream Shop	Antique Shop	Arcade	Restaurant	De Bode
3	Hajiali	Shopping Mall	Ice Cream Shop	Fast Food Restaurant	Department Store	Golf Course	Deli / Bodega	Indi Restaura
4	Kamathipura	Indian Restaurant	Antique Shop	Arts & Crafts Store	BBQ Joint	Deli / Bodega	Food Court	Firewor Sto

In [28]:

```
from sklearn.cluster import KMeans

# set number of clusters
kclusters = 3

mCentral_grouped_clustering = mCentral_grouped.drop('Area', 1)

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

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

# add clustering labels
areas_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)
```

In [29]:

```
#create a new dataframe that includes the cluster as well as the top 10 venues for each nei
mCentral_merged = mCentral

# merge toronto_grouped with toronto_data to add latitude/longitude for each neighborhood
mCentral_merged = mCentral_merged.join(areas_venues_sorted.set_index('Area'), on='Location'

mCentral_merged
```

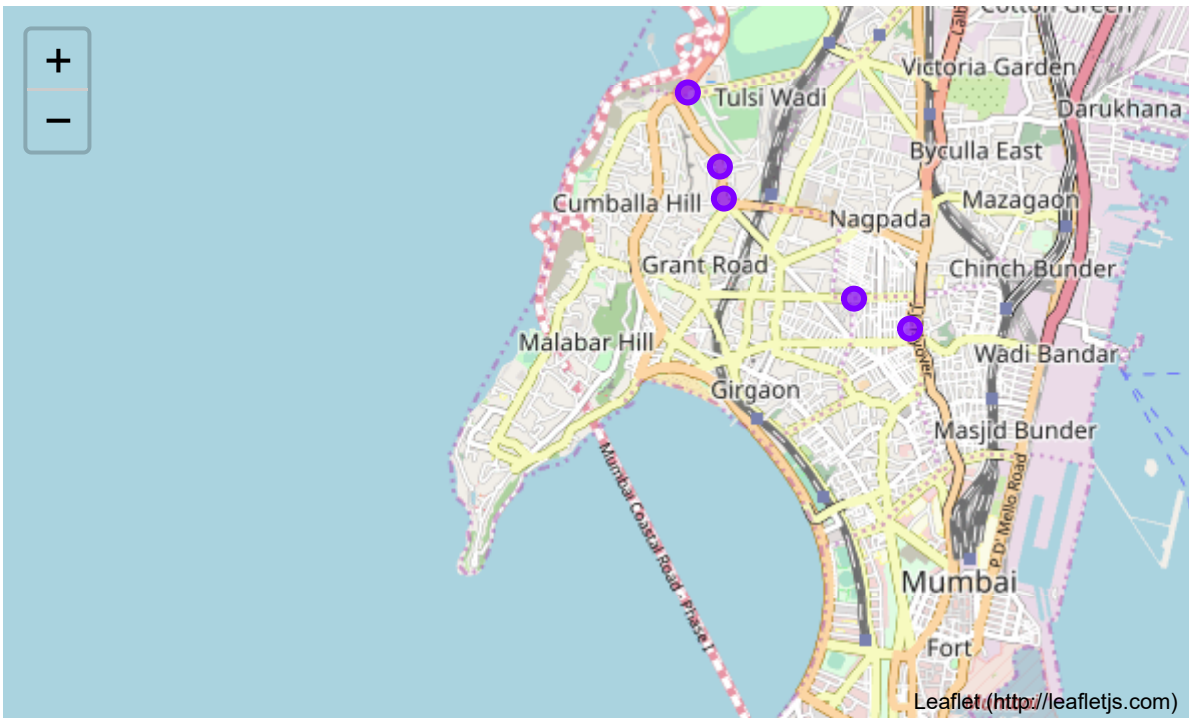
Out[29]:

	Pincode	Location	District	State	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Com V
0	400007	Bharat Nagar	Mumbai Central	Maharashtra	8.53727	76.99409	0	ATM	Firev
1	400007	Grant Road	Mumbai Central	Maharashtra	18.95929	72.83108	1	Indian Restaurant	De
2	400007	Swami Vivekand Road	Mumbai Central	Maharashtra	28.24905	77.07279	0	ATM	Constru Landsc
3	400007	Tardeo	Mumbai Central	Maharashtra	18.97243	72.81483	1	Chinese Restaurant	Fast Resta
4	400008	Falkland Road	Mumbai Central	Maharashtra	21.67448	87.55792	2	Pharmacy	Vegeta V Resta
5	400008	Kamathipura	Mumbai Central	Maharashtra	18.96172	72.82627	1	Indian Restaurant	Ar
6	400008	Mumbai Central	Mumbai Central	Maharashtra	18.96972	72.81507	1	Chinese Restaurant	Fast Resta
7	400034	Hajiali	Mumbai Central	Maharashtra	18.97834	72.81214	1	Shopping Mall	Ice C
8	400034	Tulsiwadi	Mumbai Central	Maharashtra	22.31646	73.20885	1	Sandwich Place	Juic

In [45]:

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

#Finally, let's visualize the resulting clusters
# create map 18.9387711, 72.8353355
mCentral_clusters = folium.Map(location=[18.9387711, 72.8353355], 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(mCentral_merged['Latitude'], mCentral_merged['Longitude']
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True)
    cluster
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[(int)(cluster-1)],
        fill=True,
        fill_color=rainbow[(int)(cluster-1)],
        fill_opacity=0.7).add_to(mCentral_clusters)

mCentral_clusters
```

Out[45]:



7. Results

In [46]:

```
#Cluster 1 for Delhi
sDelhi_merged.loc[sDelhi_merged['Cluster Labels'] == 0, sDelhi_merged.columns[[2] + list(ra
```

Out[46]:

	District	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6t Co
0	SOUTH DELHI	76.77884	0	Indian Restaurant	Ice Cream Shop	Clothing Store	Miscellaneous Shop	Café	Fas Res
3	SOUTH DELHI	-70.25125	0	Seafood Restaurant	Hotel	Coffee Shop	Italian Restaurant	Bar	Ice
4	SOUTH DELHI	77.21281	0	Hotel	Restaurant	Café	Pizza Place	Hostel	Res
6	SOUTH DELHI	77.20108	0	Gift Shop	Coffee Shop	Asian Restaurant	Indian Restaurant	Fast Food Restaurant	

In [47]:

```
#Cluster 2 for Delhi
sDelhi_merged.loc[sDelhi_merged['Cluster Labels'] == 1, sDelhi_merged.columns[[2] + list(ra
```

Out[47]:

	District	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue
5	SOUTH DELHI	77.18858	1	Fast Food Restaurant	Snack Place	Hotel	Indian Restaurant	Bakery	Coffee Shop

In [48]:

```
#Cluster 3 for Delhi
sDelhi_merged.loc[sDelhi_merged['Cluster Labels'] == 2, sDelhi_merged.columns[[2] + list(ra
```

Out[48]:

	District	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Mos Commor Venue
1	SOUTH DELHI	77.17347	2	Train Station	Airport Terminal	Convenience Store	Business Service	Fast Food Restaurant	Coffee Shop
2	SOUTH DELHI	77.17347	2	Train Station	Airport Terminal	Convenience Store	Business Service	Fast Food Restaurant	Coffee Shop

In [49]:

```
#Cluster 1 for Mumbai
mCentral_merged.loc[mCentral_merged['Cluster Labels'] == 0, mCentral_merged.columns[[2] + l
```

Out[49]:

	District	Longitude	Cluster Labels	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	Mumbai Central	76.99409	0	ATM	Fireworks Store	Plaza	Arcade	Arts & Crafts Store	BBQ Join
2	Mumbai Central	77.07279	0	ATM	Construction & Landscaping	Food Court	Fireworks Store	Fast Food Restaurant	Desser Shop

In [50]:

```
#Cluster 2 for Mumbai
mCentral_merged.loc[mCentral_merged['Cluster Labels'] == 1, mCentral_merged.columns[[2] + l
```

Out[50]:

	District	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th M Comm Ver
1	Mumbai Central	72.83108	1	Indian Restaurant	Dessert Shop	Ice Cream Shop	Antique Shop	Arcade	Restaur
3	Mumbai Central	72.81483	1	Chinese Restaurant	Fast Food Restaurant	Sandwich Place	Vegetarian / Vegan Restaurant	Pizza Place	Beng Restaur
5	Mumbai Central	72.82627	1	Indian Restaurant	Antique Shop	Arts & Crafts Store	BBQ Joint	Deli / Bodega	Fo Co
6	Mumbai Central	72.81507	1	Chinese Restaurant	Fast Food Restaurant	Train Station	Vegetarian / Vegan Restaurant	Pizza Place	Beng Restaur
7	Mumbai Central	72.81214	1	Shopping Mall	Ice Cream Shop	Fast Food Restaurant	Department Store	Golf Course	De Bode
8	Mumbai Central	73.20885	1	Sandwich Place	Juice Bar	Pool Hall	Vegetarian / Vegan Restaurant	Coffee Shop	Fast Fo Restaur

In [51]:

```
#Cluster 3 for Mumbai
mCentral_merged.loc[mCentral_merged['Cluster Labels'] == 2, mCentral_merged.columns[[2] + l
```

Out[51]:

	District	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Mc Comm Ven
4	Mumbai Central	87.55792	2	Pharmacy	Vegetarian / Vegan Restaurant	Construction & Landscaping	Fireworks Store	Fast Food Restaurant	Dess Sh

8. Discussion

Based on cluster for each cities above, we believe that classification for each cluster can be done better with calculation of venues categories (most common) in each cities. Refering to each clsuter, we can't deterimine clearly what represent in each cluster by using Foursquare - Most Common Venue data.

However, we can make an assumption about each cluster as follow:

Cluster 1: *Delhi:* Gift Shop
Cluster 2: *Delhi:* Restaurants
Cluster 3: *Delhi:* Train Station
Cluster 1: *Mumbai:* Indian Restuarants
Cluster 2: *Mumbai:* Mix Cuisine Restuarants
Cluster 3: *Mumbai:* Vegeterian Restuarants

What is lacking at this point is a systematic, quantitative way to identify and distinguish different district and to describe the correlation most common venues as recorded in Foursquare. The reality is however more complex: similar cities might have or might not have similar common venues. A further step in this classification would be to find a method to extract these common venues and integrate the spatial correlations between different of areas or district.

We believe that the classification we propose is an encouraging step towards a quantitative and systematic comparison of the different cities. Further studies are indeed needed in order to relate the data acquired, then observe it to more meaningful and objective results.

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 similarities or dissimilarirties in both the cities
- Classification of area located inside the city whether it is restaurant, gift place, or others

In conclusion, both cities Delhi and Mumbai are the center of attraction among India. However, to declare both cities are similar or dissimilar base on common venues visited is quite difficult. Both cities is similar in some venues also dissimilar in certain venues. And for classitification based on common venues, again we must have

more systematic or quantitative way to identify and declare this. Comparison can be made, but no such method or quantitative data to determine this. We hope in the future, a method to determine it can be establish and explore for references.