**IBM Capstone Project: The Battle of the Neighborhoods**[**¶**](https://render.githubusercontent.com/view/ipynb?commit=e6e9570b1cdd64cdd8747ce5465c12383e84a517&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f74756c61796361676c6179616e2f436f7572736572615f43617073746f6e652f653665393537306231636464363463646438373437636535343635633132333833653834613531372f49424d5f41737369676e6d656e745f426174746c655f6f665f4e65696768626f72686f6f64732e6970796e62&nwo=tulaycaglayan%2FCoursera_Capstone&path=IBM_Assignment_Battle_of_Neighborhoods.ipynb&repository_id=212115637&repository_type=Repository#IBM-Capstone-Project:-The-Battle-of-the-Neighborhoods)

**Open new Turkish Restaurant in Bucharest, Romania**[**¶**](https://render.githubusercontent.com/view/ipynb?commit=e6e9570b1cdd64cdd8747ce5465c12383e84a517&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f74756c61796361676c6179616e2f436f7572736572615f43617073746f6e652f653665393537306231636464363463646438373437636535343635633132333833653834613531372f49424d5f41737369676e6d656e745f426174746c655f6f665f4e65696768626f72686f6f64732e6970796e62&nwo=tulaycaglayan%2FCoursera_Capstone&path=IBM_Assignment_Battle_of_Neighborhoods.ipynb&repository_id=212115637&repository_type=Repository#Open-new-Turkish-Restaurant-in-Bucharest,-Romania)

**Introduction**[**¶**](https://render.githubusercontent.com/view/ipynb?commit=e6e9570b1cdd64cdd8747ce5465c12383e84a517&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f74756c61796361676c6179616e2f436f7572736572615f43617073746f6e652f653665393537306231636464363463646438373437636535343635633132333833653834613531372f49424d5f41737369676e6d656e745f426174746c655f6f665f4e65696768626f72686f6f64732e6970796e62&nwo=tulaycaglayan%2FCoursera_Capstone&path=IBM_Assignment_Battle_of_Neighborhoods.ipynb&repository_id=212115637&repository_type=Repository#Introduction)

In this project, I will determine which place is good for opening Turkish Restaurant in Bucharest, Romania

* I will convert address data into their equivalent latitude and longitude values.
* For Bucharest neighborhood data, I will use wikipedia , <https://en.wikipedia.org/wiki/Category:Districts> of Bucharest
* I will use the Foursquare API to explore Bucharest neighborhoods and to get venues in neighborhoods.
* I will use the Foursquare API to get venue ratings and likes in neighborhoods.
* I will use the *k*-means clustering and Agglomerative algorithms to complete clustering task
* I will use the Folium library to visualize the neighborhoods, venues , clusters in Bucharest

**Table of Contents**[**¶**](https://render.githubusercontent.com/view/ipynb?commit=e6e9570b1cdd64cdd8747ce5465c12383e84a517&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f74756c61796361676c6179616e2f436f7572736572615f43617073746f6e652f653665393537306231636464363463646438373437636535343635633132333833653834613531372f49424d5f41737369676e6d656e745f426174746c655f6f665f4e65696768626f72686f6f64732e6970796e62&nwo=tulaycaglayan%2FCoursera_Capstone&path=IBM_Assignment_Battle_of_Neighborhoods.ipynb&repository_id=212115637&repository_type=Repository#Table-of-Contents)

1. [Download and Explore Neigborhood Dataset](https://render.githubusercontent.com/view/ipynb?commit=e6e9570b1cdd64cdd8747ce5465c12383e84a517&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f74756c61796361676c6179616e2f436f7572736572615f43617073746f6e652f653665393537306231636464363463646438373437636535343635633132333833653834613531372f49424d5f41737369676e6d656e745f426174746c655f6f665f4e65696768626f72686f6f64732e6970796e62&nwo=tulaycaglayan%2FCoursera_Capstone&path=IBM_Assignment_Battle_of_Neighborhoods.ipynb&repository_id=212115637&repository_type=Repository#item1)
2. [Load and Analyze Venues of Neighborhoods in Bucharest](https://render.githubusercontent.com/view/ipynb?commit=e6e9570b1cdd64cdd8747ce5465c12383e84a517&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f74756c61796361676c6179616e2f436f7572736572615f43617073746f6e652f653665393537306231636464363463646438373437636535343635633132333833653834613531372f49424d5f41737369676e6d656e745f426174746c655f6f665f4e65696768626f72686f6f64732e6970796e62&nwo=tulaycaglayan%2FCoursera_Capstone&path=IBM_Assignment_Battle_of_Neighborhoods.ipynb&repository_id=212115637&repository_type=Repository#item2)
3. [Cluster Neighborhoods using Agglomerative Clustering](https://render.githubusercontent.com/view/ipynb?commit=e6e9570b1cdd64cdd8747ce5465c12383e84a517&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f74756c61796361676c6179616e2f436f7572736572615f43617073746f6e652f653665393537306231636464363463646438373437636535343635633132333833653834613531372f49424d5f41737369676e6d656e745f426174746c655f6f665f4e65696768626f72686f6f64732e6970796e62&nwo=tulaycaglayan%2FCoursera_Capstone&path=IBM_Assignment_Battle_of_Neighborhoods.ipynb&repository_id=212115637&repository_type=Repository#item3)
4. [Discussion](https://render.githubusercontent.com/view/ipynb?commit=e6e9570b1cdd64cdd8747ce5465c12383e84a517&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f74756c61796361676c6179616e2f436f7572736572615f43617073746f6e652f653665393537306231636464363463646438373437636535343635633132333833653834613531372f49424d5f41737369676e6d656e745f426174746c655f6f665f4e65696768626f72686f6f64732e6970796e62&nwo=tulaycaglayan%2FCoursera_Capstone&path=IBM_Assignment_Battle_of_Neighborhoods.ipynb&repository_id=212115637&repository_type=Repository#item4)
5. [Conclusion](https://render.githubusercontent.com/view/ipynb?commit=e6e9570b1cdd64cdd8747ce5465c12383e84a517&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f74756c61796361676c6179616e2f436f7572736572615f43617073746f6e652f653665393537306231636464363463646438373437636535343635633132333833653834613531372f49424d5f41737369676e6d656e745f426174746c655f6f665f4e65696768626f72686f6f64732e6970796e62&nwo=tulaycaglayan%2FCoursera_Capstone&path=IBM_Assignment_Battle_of_Neighborhoods.ipynb&repository_id=212115637&repository_type=Repository#item5)

**1. Download and Explore Neigborhood Dataset**[**¶**](https://render.githubusercontent.com/view/ipynb?commit=e6e9570b1cdd64cdd8747ce5465c12383e84a517&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f74756c61796361676c6179616e2f436f7572736572615f43617073746f6e652f653665393537306231636464363463646438373437636535343635633132333833653834613531372f49424d5f41737369676e6d656e745f426174746c655f6f665f4e65696768626f72686f6f64732e6970796e62&nwo=tulaycaglayan%2FCoursera_Capstone&path=IBM_Assignment_Battle_of_Neighborhoods.ipynb&repository_id=212115637&repository_type=Repository#1.-Download-and-Explore-Neigborhood-Dataset)

**Import Libraries**[**¶**](https://render.githubusercontent.com/view/ipynb?commit=e6e9570b1cdd64cdd8747ce5465c12383e84a517&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f74756c61796361676c6179616e2f436f7572736572615f43617073746f6e652f653665393537306231636464363463646438373437636535343635633132333833653834613531372f49424d5f41737369676e6d656e745f426174746c655f6f665f4e65696768626f72686f6f64732e6970796e62&nwo=tulaycaglayan%2FCoursera_Capstone&path=IBM_Assignment_Battle_of_Neighborhoods.ipynb&repository_id=212115637&repository_type=Repository#Import-Libraries)

In [2]:

#install libraries

!pip install geopy

!pip install folium

!pip install geocoder

In [3]:

#import libraries

import numpy as np # library to handle data in a vectorized manner

import pandas as pd # library for data analsysis

pd.set\_option('display.max\_columns', None)

pd.set\_option('display.max\_rows', None)

import matplotlib.pyplot as plt # for graphical usage

import json # library to handle JSON files

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

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

import geocoder # to get coordinates

import requests # library to handle requests

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

# 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

import folium # map rendering library

from folium import plugins

from folium.plugins import HeatMap

# main documentation page: http://beautiful-soup-4.readthedocs.io/en/latest/

# how to use the BeautifulSoup package: https://www.youtube.com/watch?v=ng2o98k983k video

from bs4 import BeautifulSoup

import pandas as pd

import requests

print('Libraries imported.')

Libraries imported.

**Get geocoordinates of Bucharest, Romania**[**¶**](https://render.githubusercontent.com/view/ipynb?commit=e6e9570b1cdd64cdd8747ce5465c12383e84a517&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f74756c61796361676c6179616e2f436f7572736572615f43617073746f6e652f653665393537306231636464363463646438373437636535343635633132333833653834613531372f49424d5f41737369676e6d656e745f426174746c655f6f665f4e65696768626f72686f6f64732e6970796e62&nwo=tulaycaglayan%2FCoursera_Capstone&path=IBM_Assignment_Battle_of_Neighborhoods.ipynb&repository_id=212115637&repository_type=Repository#Get-geocoordinates-of-Bucharest,-Romania)

In [4]:

# get coordinates of Bucharest

bucharest\_address = 'Bucharest, Romania'

geolocator = Nominatim(user\_agent="bucharest\_explorer")

location = geolocator.geocode(bucharest\_address)

latitude = location.latitude

longitude = location.longitude

bucharest\_center = [latitude, longitude ]

print('The geograpical coordinate of {} are {}, {}.'.format(bucharest\_address, latitude, longitude))

The geograpical coordinate of Bucharest, Romania are 44.4361414, 26.1027202.

**Get neighborhood data of Bucharest**[**¶**](https://render.githubusercontent.com/view/ipynb?commit=e6e9570b1cdd64cdd8747ce5465c12383e84a517&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f74756c61796361676c6179616e2f436f7572736572615f43617073746f6e652f653665393537306231636464363463646438373437636535343635633132333833653834613531372f49424d5f41737369676e6d656e745f426174746c655f6f665f4e65696768626f72686f6f64732e6970796e62&nwo=tulaycaglayan%2FCoursera_Capstone&path=IBM_Assignment_Battle_of_Neighborhoods.ipynb&repository_id=212115637&repository_type=Repository#Get-neighborhood-data-of-Bucharest)

I used wikiPedia, "Category:Districts of Bucharest" for getting neighborhoods of Bucharest .

In [5]:

# Read Bucharest neighborhood data

url = "https://en.wikipedia.org/wiki/Category:Districts of Bucharest"

source = requests.get(url).text

soup = BeautifulSoup(source,'lxml')

neighborhoodList = []

# append the data into the list

for row in soup.find\_all("div", class\_="mw-category")[0].findAll("li"):

neighborhoodList.append(row.text.replace(', Bucharest',''))

df\_neighborhood = pd.DataFrame({"Neighborhood": neighborhoodList})

print("There are {} neighborhoods in {}".format(df\_neighborhood.shape[0], bucharest\_address))

There are 40 neighborhoods in Bucharest, Romania

**Get sector of neighborhood and population of sector**[**¶**](https://render.githubusercontent.com/view/ipynb?commit=e6e9570b1cdd64cdd8747ce5465c12383e84a517&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f74756c61796361676c6179616e2f436f7572736572615f43617073746f6e652f653665393537306231636464363463646438373437636535343635633132333833653834613531372f49424d5f41737369676e6d656e745f426174746c655f6f665f4e65696768626f72686f6f64732e6970796e62&nwo=tulaycaglayan%2FCoursera_Capstone&path=IBM_Assignment_Battle_of_Neighborhoods.ipynb&repository_id=212115637&repository_type=Repository#Get-sector-of-neighborhood-and-population-of-sector)

In [6]:

# Read Bucharest sector data from wikipedia

url = "https://en.wikipedia.org/wiki/Sectors of Bucharest"

source = requests.get(url).text

soup = BeautifulSoup(source,'lxml')

sectorPopList = []

sectorPopulationList = []

for row in soup.find\_all("tbody"):

header = str(row.findAll("th"))

if "Population (October 2011)" in header:

i = 0

for td in row.find\_all("td"):

i+=1

if i==2:

sectorPopList.append(td.text.replace("\n",""))

if i==3:

sectorPopulationList.append(td.text.replace("\n",""))

i=0

df\_sectorPop = pd.DataFrame({"Sector": sectorPopList, "Population": sectorPopulationList})

sectorNeigList =[]

sectorNeigborList =[]

for row in soup.find\_all("ul"):

if sectorPopList[0] in row.text:

for s in row.text.split("\n"):

sectorNeigList.append(s.split(":")[0])

sectorNeigborList.append(s.split(":")[1])

df\_sector= pd.DataFrame({"Sector": sectorNeigList, "Neigborhoods": sectorNeigborList}).merge(df\_sectorPop,on='Sector' )

print("There are {} Sector in {}".format(df\_sector.shape[0], bucharest\_address))

df\_sector

There are 6 Sector in Bucharest, Romania

Out[6]:

|  | **Sector** | **Neigborhoods** | **Population** |
| --- | --- | --- | --- |
| **0** | Sector 1 | Dorobanți, Băneasa, Aviației, Pipera, Aviator... | 225,453 |
| **1** | Sector 2 | Pantelimon, Colentina, Iancului, Tei, Floreas... | 345,370 |
| **2** | Sector 3 | Vitan, Dudești, Titan, Centrul Civic, Balta A... | 385,439 |
| **3** | Sector 4 | Berceni, Olteniței, Văcărești, Timpuri Noi, T... | 287,828 |
| **4** | Sector 5 | Rahova, Ferentari, Giurgiului, Cotroceni, 13 ... | 271,575 |
| **5** | Sector 6 | Giulești, Crângași, Drumul Taberei, Militari,... | 367,760 |

**Set sector of neighborhoods**[**¶**](https://render.githubusercontent.com/view/ipynb?commit=e6e9570b1cdd64cdd8747ce5465c12383e84a517&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f74756c61796361676c6179616e2f436f7572736572615f43617073746f6e652f653665393537306231636464363463646438373437636535343635633132333833653834613531372f49424d5f41737369676e6d656e745f426174746c655f6f665f4e65696768626f72686f6f64732e6970796e62&nwo=tulaycaglayan%2FCoursera_Capstone&path=IBM_Assignment_Battle_of_Neighborhoods.ipynb&repository_id=212115637&repository_type=Repository#Set-sector-of-neighborhoods)

In [7]:

def getSector(row):

for i in range(df\_sector.shape[0]):

if row["Neighborhood"] in df\_sector.iloc[i].Neigborhoods:

return pd.Series([df\_sector.iloc[i].Sector, df\_sector.iloc[i].Population], index = ['Sector','SectorPopulation'])

df\_neighborhood[["Sector","SectorPopulation"]] =df\_neighborhood.apply(getSector, axis=1)

df\_neighborhood.head(5)

Out[7]:

|  | **Neighborhood** | **Sector** | **SectorPopulation** |
| --- | --- | --- | --- |
| **0** | Aviației | Sector 1 | 225,453 |
| **1** | Băneasa | Sector 1 | 225,453 |
| **2** | Berceni | Sector 4 | 287,828 |
| **3** | Bucureștii Noi | Sector 1 | 225,453 |
| **4** | Centrul Civic | Sector 3 | 385,439 |

**Get geographical coordinates of neighborhoods**[**¶**](https://render.githubusercontent.com/view/ipynb?commit=e6e9570b1cdd64cdd8747ce5465c12383e84a517&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f74756c61796361676c6179616e2f436f7572736572615f43617073746f6e652f653665393537306231636464363463646438373437636535343635633132333833653834613531372f49424d5f41737369676e6d656e745f426174746c655f6f665f4e65696768626f72686f6f64732e6970796e62&nwo=tulaycaglayan%2FCoursera_Capstone&path=IBM_Assignment_Battle_of_Neighborhoods.ipynb&repository_id=212115637&repository_type=Repository#Get-geographical-coordinates-of-neighborhoods)

I use python geocoder library to get geograpical coordinates of neighborhoods

In [8]:

# define a function to get coordinates

def get\_latlng(neighborhood):

# initialize your variable to None

lat\_lng\_coords = None

# loop until you get the coordinates

while(lat\_lng\_coords is None):

g = geocoder.arcgis('{}, {}'.format(neighborhood,bucharest\_address))

lat\_lng\_coords = g.latlng

return lat\_lng\_coords

coords = [ get\_latlng(neighborhood) for neighborhood in df\_neighborhood["Neighborhood"].tolist() ]

df\_coords = pd.DataFrame(coords, columns=['Latitude', 'Longitude'])

# merge the coordinates into the original dataframe

df\_neighborhood['Latitude'] = df\_coords['Latitude']

df\_neighborhood['Longitude'] = df\_coords['Longitude']

print("Geographical coordinates of five neighborhoods are as below")

df\_neighborhood.head()

Geographical coordinates of five neighborhoods are as below

Out[8]:

|  | **Neighborhood** | **Sector** | **SectorPopulation** | **Latitude** | **Longitude** |
| --- | --- | --- | --- | --- | --- |
| **0** | Aviației | Sector 1 | 225,453 | 44.485790 | 26.101219 |
| **1** | Băneasa | Sector 1 | 225,453 | 44.493952 | 26.080518 |
| **2** | Berceni | Sector 4 | 287,828 | 44.386430 | 26.128490 |
| **3** | Bucureștii Noi | Sector 1 | 225,453 | 44.480413 | 26.042807 |
| **4** | Centrul Civic | Sector 3 | 385,439 | 44.434300 | 26.094670 |

**Create Bucharest map with neighborhoods superimposed**[**¶**](https://render.githubusercontent.com/view/ipynb?commit=e6e9570b1cdd64cdd8747ce5465c12383e84a517&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f74756c61796361676c6179616e2f436f7572736572615f43617073746f6e652f653665393537306231636464363463646438373437636535343635633132333833653834613531372f49424d5f41737369676e6d656e745f426174746c655f6f665f4e65696768626f72686f6f64732e6970796e62&nwo=tulaycaglayan%2FCoursera_Capstone&path=IBM_Assignment_Battle_of_Neighborhoods.ipynb&repository_id=212115637&repository_type=Repository#Create-Bucharest-map-with-neighborhoods-superimposed)

Let's see locations of neighborhoods on map. In here, I use folium library to draw map

In [85]:

#create map of Bucharest neighborhoods using latitude and longitude values

map\_bucharest= folium.Map(location=[latitude, longitude], zoom\_start=11)

# add markers to map

for lat, lng, neighborhood in zip(df\_neighborhood['Latitude'], df\_neighborhood['Longitude'], df\_neighborhood['Neighborhood']):

label = '{}'.format(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\_bucharest)

map\_bucharest

Out[85]:

**2. Load Venues and Analyze Neighborhoods in Bucharest**[**¶**](https://render.githubusercontent.com/view/ipynb?commit=e6e9570b1cdd64cdd8747ce5465c12383e84a517&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f74756c61796361676c6179616e2f436f7572736572615f43617073746f6e652f653665393537306231636464363463646438373437636535343635633132333833653834613531372f49424d5f41737369676e6d656e745f426174746c655f6f665f4e65696768626f72686f6f64732e6970796e62&nwo=tulaycaglayan%2FCoursera_Capstone&path=IBM_Assignment_Battle_of_Neighborhoods.ipynb&repository_id=212115637&repository_type=Repository#2.-Load-Venues-and-Analyze-Neighborhoods-in-Bucharest)

We will use Foursquare API for getting venues of neighborhoods

**Explore Venues of Bucharest neighborhoods**[**¶**](https://render.githubusercontent.com/view/ipynb?commit=e6e9570b1cdd64cdd8747ce5465c12383e84a517&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f74756c61796361676c6179616e2f436f7572736572615f43617073746f6e652f653665393537306231636464363463646438373437636535343635633132333833653834613531372f49424d5f41737369676e6d656e745f426174746c655f6f665f4e65696768626f72686f6f64732e6970796e62&nwo=tulaycaglayan%2FCoursera_Capstone&path=IBM_Assignment_Battle_of_Neighborhoods.ipynb&repository_id=212115637&repository_type=Repository#Explore-Venues-of-Bucharest-neighborhoods)

I get venue list with 1 km distance to neighborhood's center

In [89]:

# @hidden\_cell

In [11]:

LIMIT = 100

def getNeighborhoodVenues( latitude, longitude,neighborhood, radius=1000 ):

venues = []

for lat, long, neighborhood in zip(latitude, longitude ,neighborhood):

# 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,

long,

radius,

LIMIT)

# make the GET request

results = requests.get(url).json()["response"]['groups'][0]['items']

# return only relevant information for each nearby venue

for venue in results:

venues.append((

neighborhood,

lat,

long,

venue['venue']['name'],

venue['venue']['id'],

venue['venue']['location']['lat'],

venue['venue']['location']['lng'],

venue['venue']['location']['distance'],

venue['venue']['categories'][0]['name']))

# convert the venues list into a DataFrame

venues = pd.DataFrame(venues)

# define the column names

venues.columns = ['Neighborhood', 'Latitude', 'Longitude', 'VenueName', 'VenueId', 'VenueLatitude', 'VenueLongitude','VenueDistance','VenueCategory']

return venues

neighborhood\_venues = getNeighborhoodVenues (df\_neighborhood['Latitude'], df\_neighborhood['Longitude'], df\_neighborhood['Neighborhood'] )

print('There are {} unique venue categories. Some of them are as below:'.format(len(neighborhood\_venues['VenueCategory'].unique())))

neighborhood\_venues.head()

There are 225 unique venue categories. Some of them are as below:

Out[11]:

|  | **Neighborhood** | **Latitude** | **Longitude** | **VenueName** | **VenueId** | **VenueLatitude** | **VenueLongitude** | **VenueDistance** | **VenueCategory** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | Aviației | 44.48579 | 26.101219 | LIDL | 583956246d349d0574eb02ac | 44.488396 | 26.094375 | 616 | Supermarket |
| **1** | Aviației | 44.48579 | 26.101219 | Mega Image Concept Store | 56348b62498e53f51a0a4e0e | 44.479783 | 26.102568 | 677 | Supermarket |
| **2** | Aviației | 44.48579 | 26.101219 | Flying Pig | 58a2fc95d0bb3e516a2363b7 | 44.479454 | 26.102837 | 716 | Burger Joint |
| **3** | Aviației | 44.48579 | 26.101219 | Starbucks | 525fd077498eed1c5a52c1d6 | 44.478522 | 26.102503 | 815 | Coffee Shop |
| **4** | Aviației | 44.48579 | 26.101219 | Mega Image | 4eb5452b30f8d0f18c41dfec | 44.487255 | 26.092758 | 691 | Supermarket |

In [14]:

print('Top 10 distinct venue counts are as below')

neighborhood\_venues[['VenueId','VenueCategory']].drop\_duplicates().groupby('VenueCategory').count()[['VenueId']].rename(columns={"VenueId": "Count"}).sort\_values(by=['Count'], ascending=False)[:10]

Top 10 distinct venue counts are as below

Out[14]:

|  | **Count** |
| --- | --- |
| **VenueCategory** |  |
| **Café** | 76 |
| **Restaurant** | 72 |
| **Italian Restaurant** | 63 |
| **Coffee Shop** | 54 |
| **Supermarket** | 53 |
| **Pizza Place** | 51 |
| **Gym** | 41 |
| **Hotel** | 40 |
| **Romanian Restaurant** | 36 |
| **Gym / Fitness Center** | 34 |

**Filter restaurants out of all venues**[**¶**](https://render.githubusercontent.com/view/ipynb?commit=e6e9570b1cdd64cdd8747ce5465c12383e84a517&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f74756c61796361676c6179616e2f436f7572736572615f43617073746f6e652f653665393537306231636464363463646438373437636535343635633132333833653834613531372f49424d5f41737369676e6d656e745f426174746c655f6f665f4e65696768626f72686f6f64732e6970796e62&nwo=tulaycaglayan%2FCoursera_Capstone&path=IBM_Assignment_Battle_of_Neighborhoods.ipynb&repository_id=212115637&repository_type=Repository#Filter-restaurants-out-of-all-venues)

Let's filter venues and get only restaurants

In [12]:

restaurant\_list =['Restaurant', 'Burger Joint','Café','Fried Chicken Joint','Pizza Place']

turkish\_restaurant\_list = ['Turkish Restaurant', 'Doner Restaurant']

# Filter restaurants

neighborhood\_venues['RestFlag']=False

for restCat in restaurant\_list:

neighborhood\_venues['RestFlag'] = neighborhood\_venues['RestFlag'] | neighborhood\_venues['VenueCategory'].str.contains(restCat)

neighborhood\_restaurants = neighborhood\_venues[neighborhood\_venues['RestFlag'] == True].iloc[:,:-1]

turkish\_restaurants = neighborhood\_restaurants[ neighborhood\_restaurants['VenueCategory'].isin(turkish\_restaurant\_list) ]

other\_restaurants = neighborhood\_restaurants[ ~neighborhood\_restaurants['VenueCategory'].isin(turkish\_restaurant\_list) ]

print('Total number of restaurants:', len(neighborhood\_restaurants['VenueId'].unique()))

print('Total number of Turkish restaurants:', len(turkish\_restaurants['VenueId'].unique()))

print('Percentage of Turkish restaurants: {:.2f}%'.format(len(turkish\_restaurants['VenueId'].unique()) / len(neighborhood\_restaurants['VenueId'].unique()) \* 100))

Total number of restaurants: 502

Total number of Turkish restaurants: 21

Percentage of Turkish restaurants: 4.18%

In [13]:

# get counts of restaurants in each Neighborhood

df\_rest\_counts = neighborhood\_restaurants.groupby(['Neighborhood']).count().rename(columns={"VenueCategory": "RestaurantCount"})[['RestaurantCount']]

#find neighborhoods that does not have any restaurant

noRestList = list(set(neighborhood\_venues['Neighborhood']) - set(neighborhood\_restaurants['Neighborhood']))

#if exists , append neighborhoods without any restaurant to df\_rest\_counts

if noRestList != []:

df\_rest\_counts = df\_rest\_counts.append (pd.DataFrame( {'Neighborhood' : noRestList , 'RestaurantCount': [0] \* len(noRestList) } ).set\_index('Neighborhood'))

df\_rest\_counts.reset\_index(inplace=True)

#####

# get counts of Turkish restaurants in each Neighborhood

df\_turk\_rest\_counts = turkish\_restaurants.groupby(['Neighborhood']).count().rename(columns={"VenueCategory": "TurkRestaurantCount"})[['TurkRestaurantCount']]

#find neighborhoods that does not have any restaurant

noRestList = list(set(neighborhood\_venues['Neighborhood']) - set(turkish\_restaurants['Neighborhood']))

#if exists , append neighborhoods without any restaurant to df\_rest\_counts

if noRestList != []:

df\_turk\_rest\_counts = df\_turk\_rest\_counts.append (pd.DataFrame( {'Neighborhood' : noRestList , 'TurkRestaurantCount': [0] \* len(noRestList) } ).set\_index('Neighborhood'))

df\_turk\_rest\_counts.reset\_index(inplace=True)

df\_rest\_counts= df\_rest\_counts.merge(df\_turk\_rest\_counts).set\_index('Neighborhood')

df\_rest\_counts= df\_rest\_counts.sort\_values(by=['RestaurantCount'],ascending =False)

print('{} neighborhoods do not have any Turkish restaurant'.format(len(noRestList)))

######

#Draw graph

df\_rest\_counts[['RestaurantCount','TurkRestaurantCount']].plot(kind='bar',figsize=(15,5))

plt.title('Restaurant Counts of Neighborhoods')

plt.xlabel('Neighborhood')

plt.ylabel('Restaurant Counts')

plt.show()

19 neighborhoods do not have any Turkish restaurant

We can see from the Graph :

* Vitan is the neihgborhood having most restaurants
* Dorobanti is has lots of restaurants , but no Tukish restaurant
* Odai and Gluesti are neihgborhoods having least number of restaurants
* 19 neihgborhoods do not have any Turkish restaurant. we can choose one of these to open resaturant. To filter more let's continue

**Get the restaurant's overall rating**[**¶**](https://render.githubusercontent.com/view/ipynb?commit=e6e9570b1cdd64cdd8747ce5465c12383e84a517&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f74756c61796361676c6179616e2f436f7572736572615f43617073746f6e652f653665393537306231636464363463646438373437636535343635633132333833653834613531372f49424d5f41737369676e6d656e745f426174746c655f6f665f4e65696768626f72686f6f64732e6970796e62&nwo=tulaycaglayan%2FCoursera_Capstone&path=IBM_Assignment_Battle_of_Neighborhoods.ipynb&repository_id=212115637&repository_type=Repository#Get-the-restaurant's-overall-rating)

I used Foursquare API for getting likes and rates of restaurants

In [14]:

def getVenueRaitings(venues):

raitings = []

likes =[]

for venueId in venues:

# create the API request URL

url = 'https://api.foursquare.com/v2/venues/{}?client\_id={}&client\_secret={}&v={}'.format(venueId, CLIENT\_ID, CLIENT\_SECRET, VERSION)

result = requests.get(url).json()

try:

rating = result['response']['venue']['rating']

likes = result['response']['venue']['likes']['count']

except:

rating = None

likes = None

raitings.append((venueId, rating,likes))

# convert the venues list into a DataFrame

rating = pd.DataFrame(raitings)

# define the column names

rating.columns = ['VenueId', 'VenueRating','VenueLikes']

return rating

restaurants\_raitings = getVenueRaitings(neighborhood\_restaurants['VenueId'].drop\_duplicates())

restaurants\_raitings.head()

Out[14]:

|  | **VenueId** | **VenueRating** | **VenueLikes** |
| --- | --- | --- | --- |
| **0** | 58a2fc95d0bb3e516a2363b7 | 8.3 | 22.0 |
| **1** | 53623e44498ed583ede334f5 | 8.2 | 130.0 |
| **2** | 55b8a459498efbbb6ca3526d | 7.7 | 8.0 |
| **3** | 58bbf8764f1069627380fed5 | 7.7 | 21.0 |
| **4** | 59162d2d2be42556981e1e4a | 7.6 | 68.0 |

In [15]:

#merge raitings to restaurants

neighborhood\_restaurants = neighborhood\_restaurants.merge(restaurants\_raitings)

neighborhood\_restaurants.head()

Out[15]:

|  | **Neighborhood** | **Latitude** | **Longitude** | **VenueName** | **VenueId** | **VenueLatitude** | **VenueLongitude** | **VenueDistance** | **VenueCategory** | **VenueRating** | **VenueLikes** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | Aviației | 44.485790 | 26.101219 | Flying Pig | 58a2fc95d0bb3e516a2363b7 | 44.479454 | 26.102837 | 716 | Burger Joint | 8.3 | 22.0 |
| **1** | Floreasca | 44.476308 | 26.103289 | Flying Pig | 58a2fc95d0bb3e516a2363b7 | 44.479454 | 26.102837 | 351 | Burger Joint | 8.3 | 22.0 |
| **2** | Aviației | 44.485790 | 26.101219 | trickSHOT | 53623e44498ed583ede334f5 | 44.478378 | 26.103135 | 838 | Restaurant | 8.2 | 130.0 |
| **3** | Floreasca | 44.476308 | 26.103289 | trickSHOT | 53623e44498ed583ede334f5 | 44.478378 | 26.103135 | 230 | Restaurant | 8.2 | 130.0 |
| **4** | Aviației | 44.485790 | 26.101219 | Toàn's | 55b8a459498efbbb6ca3526d | 44.478370 | 26.103411 | 844 | Vietnamese Restaurant | 7.7 | 8.0 |

In [16]:

#merge raitings to turkish restaurants

turkish\_restaurants = turkish\_restaurants.merge(restaurants\_raitings)

turkish\_restaurants.head(5)

Out[16]:

|  | **Neighborhood** | **Latitude** | **Longitude** | **VenueName** | **VenueId** | **VenueLatitude** | **VenueLongitude** | **VenueDistance** | **VenueCategory** | **VenueRating** | **VenueLikes** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | Aviației | 44.485790 | 26.101219 | Istanbul Taksim | 55acc94f498e0a64af6b19af | 44.478560 | 26.103505 | 825 | Turkish Restaurant | 7.3 | 12.0 |
| **1** | Floreasca | 44.476308 | 26.103289 | Istanbul Taksim | 55acc94f498e0a64af6b19af | 44.478560 | 26.103505 | 251 | Turkish Restaurant | 7.3 | 12.0 |
| **2** | Băneasa | 44.493952 | 26.080518 | Shaormeria Băneasa | 5137a298e4b0523475c45b54 | 44.494460 | 26.080462 | 56 | Doner Restaurant | 7.3 | 40.0 |
| **3** | Centrul Civic | 44.434300 | 26.094670 | Dristor Kebap | 4dc58dd7887717c8802694a1 | 44.429973 | 26.100381 | 661 | Doner Restaurant | 7.9 | 466.0 |
| **4** | Lipscani | 44.432155 | 26.104057 | Dristor Kebap | 4dc58dd7887717c8802694a1 | 44.429973 | 26.100381 | 379 | Doner Restaurant | 7.9 | 466.0 |

**Show restaurants on head map**[**¶**](https://render.githubusercontent.com/view/ipynb?commit=e6e9570b1cdd64cdd8747ce5465c12383e84a517&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f74756c61796361676c6179616e2f436f7572736572615f43617073746f6e652f653665393537306231636464363463646438373437636535343635633132333833653834613531372f49424d5f41737369676e6d656e745f426174746c655f6f665f4e65696768626f72686f6f64732e6970796e62&nwo=tulaycaglayan%2FCoursera_Capstone&path=IBM_Assignment_Battle_of_Neighborhoods.ipynb&repository_id=212115637&repository_type=Repository#Show-restaurants-on-head-map)

Let's crete a map showing heatmap / density of restaurants and try to extract some meaningfull info from that. Also, let's show borders of Bucharest on our map and a few circles indicating distance of 2km, 4km, 6km and 10km from Bucharest center

Red markers are Turkish restaurants

Blue markers are Turkish restaurants with rating less than 7 restaurants

In [86]:

bucharest\_center = [latitude, longitude]

map\_restaurant= folium.Map(location=bucharest\_center, zoom\_start=12)

df\_neighborhood\_noturkrest = df\_neighborhood[df\_neighborhood['Neighborhood'].isin(noRestList)]

folium.TileLayer('cartodbpositron').add\_to(map\_bucharest)

HeatMap(neighborhood\_restaurants[['VenueLatitude','VenueLongitude']]).add\_to(map\_restaurant)

folium.Marker(bucharest\_center).add\_to(map\_bucharest)

folium.Circle(bucharest\_center, radius=2000, fill=False, color='white').add\_to(map\_restaurant)

folium.Circle(bucharest\_center, radius=4000, fill=False, color='white').add\_to(map\_restaurant)

folium.Circle(bucharest\_center, radius=6000, fill=False, color='white').add\_to(map\_restaurant)

folium.Circle(bucharest\_center, radius=10000, fill=False, color='black').add\_to(map\_restaurant)

for lat, lon, neig, name in zip(turkish\_restaurants['VenueLatitude'], turkish\_restaurants['VenueLongitude'], turkish\_restaurants['Neighborhood'], turkish\_restaurants['VenueName']):

label = folium.Popup(str(name) + ' - ' + str(neig), parse\_html=True)

folium.CircleMarker(

[lat, lon],

radius=5,

popup=label,

color='red',

fill=True,

fill\_color='#3186cc',

fill\_opacity=0.7).add\_to(map\_restaurant)

raiting\_lt\_7 = turkish\_restaurants[turkish\_restaurants['VenueRating'] <7]

for lat, lon, neig, name in zip(raiting\_lt\_7['VenueLatitude'], raiting\_lt\_7['VenueLongitude'], raiting\_lt\_7['Neighborhood'], raiting\_lt\_7['VenueName']):

label = folium.Popup(str(name) + ' - ' + str(neig), parse\_html=True)

folium.CircleMarker(

[lat, lon],

radius=5,

popup=label,

color='blue',

fill=True,

fill\_color='#3186cc',

fill\_opacity=0.7).add\_to(map\_restaurant)

map\_restaurant

Out[86]:

From heat map , we can see that there are

* Too many restaurants in 4 km to the center
* Enough restaurants between 4-6 km to the center
* There are **not many**restaurants after **6 km to center**
* Turkish restaurants are superimposed with red and blue dots. Blue dots are Turkish restaurant with raiting <7
* **Turkish restaurants are located in South, North , East, between 2-4 Km distance around center**
* There are **not Turkish Restaurants in West Part.**
* 3 Turkish restaurants gets bad points. That means people does not prefer these restaurant too much and any turkish restaurant may be opened around

**Show neighborhoods which has no Turkish restaurant on head map**[**¶**](https://render.githubusercontent.com/view/ipynb?commit=e6e9570b1cdd64cdd8747ce5465c12383e84a517&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f74756c61796361676c6179616e2f436f7572736572615f43617073746f6e652f653665393537306231636464363463646438373437636535343635633132333833653834613531372f49424d5f41737369676e6d656e745f426174746c655f6f665f4e65696768626f72686f6f64732e6970796e62&nwo=tulaycaglayan%2FCoursera_Capstone&path=IBM_Assignment_Battle_of_Neighborhoods.ipynb&repository_id=212115637&repository_type=Repository#Show-neighborhoods-which-has-no-Turkish-restaurant-on-head-map)

Cyan markers are Neigborhoods without any turkish restaurants

In [88]:

bucharest\_center = [latitude, longitude]

map\_restaurant= folium.Map(location=bucharest\_center, zoom\_start=12)

df\_neighborhood\_noturkrest = df\_neighborhood[df\_neighborhood['Neighborhood'].isin(noRestList)]

folium.TileLayer('cartodbpositron').add\_to(map\_bucharest)

HeatMap(neighborhood\_restaurants[['VenueLatitude','VenueLongitude']]).add\_to(map\_restaurant)

folium.Marker(bucharest\_center).add\_to(map\_bucharest)

folium.Circle(bucharest\_center, radius=2000, fill=False, color='white').add\_to(map\_restaurant)

folium.Circle(bucharest\_center, radius=4000, fill=False, color='white').add\_to(map\_restaurant)

folium.Circle(bucharest\_center, radius=6000, fill=False, color='white').add\_to(map\_restaurant)

folium.Circle(bucharest\_center, radius=10000, fill=False, color='black').add\_to(map\_restaurant)

for lat, lon, neig in zip(df\_neighborhood\_noturkrest['Latitude'], df\_neighborhood\_noturkrest['Longitude'], df\_neighborhood\_noturkrest['Neighborhood']):

label = folium.Popup(str(neig), parse\_html=True)

folium.CircleMarker(

[lat, lon],

radius=5,

popup=label,

color='cyan',

fill=True,

fill\_color='cyan',

fill\_opacity=0.7).add\_to(map\_restaurant)

map\_restaurant

Out[88]:

**3. Cluster Neigborhoods using Agglomerative Clustering**[**¶**](https://render.githubusercontent.com/view/ipynb?commit=e6e9570b1cdd64cdd8747ce5465c12383e84a517&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f74756c61796361676c6179616e2f436f7572736572615f43617073746f6e652f653665393537306231636464363463646438373437636535343635633132333833653834613531372f49424d5f41737369676e6d656e745f426174746c655f6f665f4e65696768626f72686f6f64732e6970796e62&nwo=tulaycaglayan%2FCoursera_Capstone&path=IBM_Assignment_Battle_of_Neighborhoods.ipynb&repository_id=212115637&repository_type=Repository#3.-Cluster-Neigborhoods-using-Agglomerative-Clustering)

In [36]:

# one hot encoding

onehot = pd.get\_dummies(neighborhood\_venues[['VenueCategory']], prefix="", prefix\_sep="")

# add neighborhood column back to dataframe

onehot['Neighborhood'] = neighborhood\_venues['Neighborhood']

venues\_grouped = onehot.groupby(["Neighborhood"]).sum().reset\_index()

# move neighborhood column to the first column and filter only restaurant columns

fixed\_columns =['Neighborhood'] + list(neighborhood\_venues['VenueCategory'].unique())

venues\_grouped = venues\_grouped[fixed\_columns]

print("{} neighborhoods' venue category are shown in {} columns as below".format(venues\_grouped.shape[0],venues\_grouped.shape[1]-1))

venues\_grouped.head()

40 neighborhoods' venue category are shown in 225 columns as below

Out[36]:

|  | **Neighborhood** | **Supermarket** | **Burger Joint** | **Coffee Shop** | **Dessert Shop** | **Beer Garden** | **Roof Deck** | **Restaurant** | **Grocery Store** | **Gym / Fitness Center** | **Shopping Mall** | **Bookstore** | **Hotel** | **Candy Store** | **Vietnamese Restaurant** | **Café** | **Pie Shop** | **Bakery** | **Salon / Barbershop** | **Salad Place** | **Clothing Store** | **Sushi Restaurant** | **Steakhouse** | **Italian Restaurant** | **Turkish Restaurant** | **Toy / Game Store** | **Sandwich Place** | **Pizza Place** | **Middle Eastern Restaurant** | **Lebanese Restaurant** | **Casino** | **Spanish Restaurant** | **Spa** | **Event Space** | **Fried Chicken Joint** | **Veterinarian** | **Lounge** | **Chinese Restaurant** | **Romanian Restaurant** | **Pub** | **Stadium** | **Metro Station** | **Tennis Stadium** | **Mongolian Restaurant** | **Eastern European Restaurant** | **Men's Store** | **Cocktail Bar** | **Donut Shop** | **Nightclub** | **Comfort Food Restaurant** | **Gym** | **Doner Restaurant** | **Vegetarian / Vegan Restaurant** | **Lake** | **Park** | **Theme Restaurant** | **Tunnel** | **Farmers Market** | **Bed & Breakfast** | **Airport Terminal** | **Bus Stop** | **Food & Drink Shop**Indoor Play AreaNature PreserveFruit & Vegetable StoreFast Food RestaurantElectronics StoreJewelry StoreFountainDepartment StoreGas StationMobile Phone ShopKorean RestaurantShop & ServiceGastropubOutdoor SculptureIndie TheaterTheaterMonasteryBistroSkating RinkIce Cream ShopHistoric SiteUsed BookstoreChocolate ShopPlazaArt MuseumBarPalaceWine BarBeer BarHookah BarConcert HallTea RoomRock ClubCosmetics ShopHostelArt GalleryMusic VenueBoutiqueSwiss RestaurantHistory MuseumMediterranean RestaurantHardware StoreDiscount StoreBus StationGift ShopPharmacyTennis CourtJazz ClubAccessories StoreOpera HouseMarketHotel BarIndian RestaurantPedestrian PlazaMusic StoreGardenPoolIndie Movie TheaterAustralian RestaurantSoccer FieldShoe StoreLight Rail StationScenic LookoutCupcake ShopFrench RestaurantClimbing GymSoccer StadiumBowling AlleySmoke ShopConvenience StoreFurniture / Home StoreOutlet MallFlower ShopJapanese RestaurantJuice BarFish MarketScandinavian RestaurantCheese ShopAsian RestaurantModern European RestaurantCreperieSeafood RestaurantMolecular Gastronomy RestaurantWine ShopGerman RestaurantDrugstoreAuto WorkshopPet StoreMultiplexEye DoctorSkate ParkGreek RestaurantIntersectionKids StoreWater ParkBreweryPlaygroundButcherHealth Food StoreAmerican RestaurantDinerAuto DealershipHealth & Beauty ServiceScience MuseumLingerie StoreRamen RestaurantCafeteriaCable CarSports ClubMuseumFood TruckArts & Crafts StoreMovie TheaterBagel ShopKebab RestaurantExhibitPublic ArtSnack PlaceTattoo ParlorSporting Goods ShopGourmet ShopKaraoke BarChurchHospitalWomen's StoreIrish PubBreakfast SpotCamera StoreBridal ShopBBQ JointHungarian RestaurantBeachAuto GarageFood CourtAthletics & SportsFoodGo Kart TrackIT ServicesTrackGym PoolRecreation CenterBike ShopBasketball CourtPaper / Office Supplies StorePool HallSports BarLaundromatBaby StoreBuffetTaco PlaceFish & Chips ShopWatch ShopRecording StudioATMSoup PlaceCircusFalafel RestaurantDance StudioPet CaféLeather Goods StoreGaming CafeCultural CenterDog RunTram Station |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 13 Septembrie | 1 | 0 | 2 | 0 | 0 | 0 | 5 | 0 | 2 | 0 | 0 | 3 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 2 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 2 | 0 | 2 | 3 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 2 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 000010001000000000200001411100002010010000001001110011000000100000000000000000000020000000000000000010000000000000000000000000000010000000000000000000111100000000000 |
| **1** | Aviației | 4 | 3 | 3 | 1 | 1 | 1 | 4 | 1 | 1 | 1 | 1 | 4 | 1 | 1 | 5 | 1 | 5 | 1 | 1 | 3 | 2 | 3 | 1 | 1 | 1 | 1 | 4 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 |
| **2** | Berceni | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 3 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 1 | 0 | 0 | 2 | 0 | 0 | 0 | 001112110000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 |
| **3** | Bucureștii Noi | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 2 | 0 | 0 | 1 | 0 | 0 | 0 | 000000001111100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 |
| **4** | Băneasa | 1 | 0 | 0 | 1 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 2 | 1 | 110000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 |

In [41]:

# create a new dataframe with most common venue catrgories

def return\_most\_common\_venues(row, num\_top\_venues):

row\_categories = row

row\_categories\_sorted = row\_categories.sort\_values(ascending=False)

return row\_categories\_sorted.index.values[0:num\_top\_venues]

num\_top\_venues = 10

columns = ['Neighborhood','Total Number of Venues']

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

# create columns according to number of top venues

for ind in np.arange(num\_top\_venues):

try:

columns.append('{}{} Most Common Restaurant'.format(ind+1, indicators[ind]))

except:

columns.append('{}th Most Common Restaurant'.format(ind+1))

# create a new dataframe

venues\_most = pd.DataFrame(columns = columns)

for ind in range(venues\_grouped.shape[0]):

venues\_most.loc[ind, 'Neighborhood'] = venues\_grouped.iloc[ind].Neighborhood

venues\_most.loc[ind, 'Total Number of Venues'] = venues\_grouped.iloc[ind,1:].sum()

venues\_most.iloc[ind, 2:] = return\_most\_common\_venues(venues\_grouped.iloc[ind, 1:], num\_top\_venues)

venues\_most.head()

Out[41]:

|  | **Neighborhood** | **Total Number of Venues** | **1st Most Common Restaurant** | **2nd Most Common Restaurant** | **3rd Most Common Restaurant** | **4th Most Common Restaurant** | **5th Most Common Restaurant** | **6th Most Common Restaurant** | **7th Most Common Restaurant** | **8th Most Common Restaurant** | **9th Most Common Restaurant** | **10th Most Common Restaurant** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 13 Septembrie | 69 | Restaurant | Plaza | Café | Hotel | Pub | Seafood Restaurant | Tea Room | Comfort Food Restaurant | Romanian Restaurant | Lounge |
| **1** | Aviației | 80 | Café | Bakery | Restaurant | Pizza Place | Hotel | Supermarket | Burger Joint | Steakhouse | Coffee Shop | Clothing Store |
| **2** | Berceni | 27 | Pizza Place | Eastern European Restaurant | Electronics Store | Farmers Market | Grocery Store | Gym / Fitness Center | Gym | Fruit & Vegetable Store | Clothing Store | Metro Station |
| **3** | Bucureștii Noi | 14 | Gym | Park | Supermarket | Spa | Metro Station | Shop & Service | Korean Restaurant | Mobile Phone Shop | Department Store | Gas Station |
| **4** | Băneasa | 30 | Park | Hotel | Romanian Restaurant | Airport Terminal | Restaurant | Café | Italian Restaurant | Pizza Place | Middle Eastern Restaurant | Nightclub |

**Cluster and find similarities between neighborhoods**[**¶**](https://render.githubusercontent.com/view/ipynb?commit=e6e9570b1cdd64cdd8747ce5465c12383e84a517&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f74756c61796361676c6179616e2f436f7572736572615f43617073746f6e652f653665393537306231636464363463646438373437636535343635633132333833653834613531372f49424d5f41737369676e6d656e745f426174746c655f6f665f4e65696768626f72686f6f64732e6970796e62&nwo=tulaycaglayan%2FCoursera_Capstone&path=IBM_Assignment_Battle_of_Neighborhoods.ipynb&repository_id=212115637&repository_type=Repository#Cluster-and-find-similarities-between-neighborhoods)

I will use hierarchical agglomerative clustering method to compare neighborhoods among neighborhoods

First, find the number of clusters. Let's use scipy library to create the dendrograms for our dataset

In [53]:

import scipy.cluster.hierarchy as shc

data = venues\_grouped.iloc[:,3:]

plt.figure(figsize=(10, 6))

plt.title('Hierarchical Clustering Dendrogram')

plt.xlabel('Neighborhoods')

plt.ylabel('Distance')

plt.axhline(y=20, c='k')

dend = shc.dendrogram(shc.linkage(data, method='ward'))

According to the above graph, I decide to separate our neighborhoods into 6 clusters (cut at distance of 20, horizontal black line). I will use the hierarchical agglomerative clustering of the sklearn.cluster library to cluster these neighborhoods.

In [58]:

from sklearn.cluster import AgglomerativeClustering

kclusters= 6

cluster = AgglomerativeClustering(n\_clusters=kclusters, affinity='euclidean', linkage='ward')

clusterresult = cluster.fit\_predict(data)

venues\_grouped['NeighborhoodCluster'] = clusterresult

venues\_cluster= df\_neighborhood.merge(venues\_grouped[['Neighborhood','NeighborhoodCluster']])

venues\_cluster =venues\_cluster.merge(venues\_most[['Neighborhood','1st Most Common Restaurant','2nd Most Common Restaurant','3rd Most Common Restaurant']])

venues\_cluster.head()

Out[58]:

|  | **Neighborhood** | **Sector** | **SectorPopulation** | **Latitude** | **Longitude** | **NeighborhoodCluster** | **1st Most Common Restaurant** | **2nd Most Common Restaurant** | **3rd Most Common Restaurant** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | Aviației | Sector 1 | 225,453 | 44.485790 | 26.101219 | 0 | Café | Bakery | Restaurant |
| **1** | Băneasa | Sector 1 | 225,453 | 44.493952 | 26.080518 | 2 | Park | Hotel | Romanian Restaurant |
| **2** | Berceni | Sector 4 | 287,828 | 44.386430 | 26.128490 | 2 | Pizza Place | Eastern European Restaurant | Electronics Store |
| **3** | Bucureștii Noi | Sector 1 | 225,453 | 44.480413 | 26.042807 | 2 | Gym | Park | Supermarket |
| **4** | Centrul Civic | Sector 3 | 385,439 | 44.434300 | 26.094670 | 3 | Coffee Shop | Hotel | Theater |

In [87]:

# 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

for lat, lon, poi, cluster, sector,SectorPopulation in zip(venues\_cluster['Latitude'], venues\_cluster['Longitude'], venues\_cluster['Neighborhood'], venues\_cluster['NeighborhoodCluster'], venues\_cluster['Sector'],venues\_cluster['SectorPopulation']):

label = folium.Popup(str(poi) + ' - Cluster ' + str(cluster)+ ' ' + str(sector) + ' ' + str(SectorPopulation), parse\_html=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)

folium.Marker(bucharest\_center).add\_to(map\_clusters)

folium.Circle(bucharest\_center, radius=2000, fill=False, color='white').add\_to(map\_clusters)

folium.Circle(bucharest\_center, radius=4000, fill=False, color='white').add\_to(map\_clusters)

folium.Circle(bucharest\_center, radius=6000, fill=False, color='white').add\_to(map\_clusters)

folium.Circle(bucharest\_center, radius=10000, fill=False, color='black').add\_to(map\_clusters)

map\_clusters

Out[87]:

In [83]:

print(venues\_cluster.groupby(['NeighborhoodCluster','1st Most Common Restaurant' ]).count()[['Neighborhood']].rename(columns={"Neighborhood": "Neighborhood Count"}))

venues\_cluster.groupby(['NeighborhoodCluster','2nd Most Common Restaurant' ]).count()[['Neighborhood']].rename(columns={"Neighborhood": "Neighborhood Count"})

Neighborhood Count

NeighborhoodCluster 1st Most Common Restaurant

0 Café 1

Coffee Shop 2

Italian Restaurant 2

Lounge 1

Pizza Place 1

Pub 1

Supermarket 2

1 Café 1

Restaurant 3

2 Burger Joint 1

Bus Station 1

Café 2

Clothing Store 1

Grocery Store 1

Gym 1

Gym / Fitness Center 1

Park 2

Pedestrian Plaza 1

Pizza Place 2

Supermarket 4

3 Coffee Shop 5

4 Café 2

5 Café 2

Out[83]:

|  |  | **Neighborhood Count** |
| --- | --- | --- |
| **NeighborhoodCluster** | **2nd Most Common Restaurant** |  |
| **0** | **Bakery** | 2 |
| **Burger Joint** | 1 |  |
| **Café** | 2 |  |
| **Gym** | 2 |  |
| **Italian Restaurant** | 1 |  |
| **Pizza Place** | 1 |  |
| **Supermarket** | 1 |  |
| **1** | **Bus Station** | 2 |
| **Hotel** | 1 |  |
| **Plaza** | 1 |  |
| **2** | **Bar** | 2 |
| **Bus Station** | 1 |  |
| **Café** | 1 |  |
| **Clothing Store** | 1 |  |
| **Dessert Shop** | 1 |  |
| **Eastern European Restaurant** | 3 |  |
| **Hotel** | 1 |  |
| **Museum** | 1 |  |
| **Park** | 2 |  |
| **Pizza Place** | 1 |  |
| **Pool** | 1 |  |
| **Pub** | 1 |  |
| **Restaurant** | 1 |  |
| **3** | **Café** | 1 |
| **Hotel** | 3 |  |
| **Pub** | 1 |  |
| **4** | **Supermarket** | 2 |
| **5** | **Coffee Shop** | 2 |

**We can see cluster categories as below**

Cluster 0 (Red) : Italian Restaurant , Pizza, Café  
Cluster 1 (Purple) : Restaurant , Bar  
Cluster 2 (BLue) : Park , Plaza, Clothing stores  
Cluster 3 (Cyan) : Coffee Shop, Hotel, Pub  
Cluster 4 (Green) : Café, Suprmarket  
Cluster 5 (Orange) : Café

**4. Discussion**[**¶**](https://render.githubusercontent.com/view/ipynb?commit=e6e9570b1cdd64cdd8747ce5465c12383e84a517&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f74756c61796361676c6179616e2f436f7572736572615f43617073746f6e652f653665393537306231636464363463646438373437636535343635633132333833653834613531372f49424d5f41737369676e6d656e745f426174746c655f6f665f4e65696768626f72686f6f64732e6970796e62&nwo=tulaycaglayan%2FCoursera_Capstone&path=IBM_Assignment_Battle_of_Neighborhoods.ipynb&repository_id=212115637&repository_type=Repository#4.-Discussion)

Cluster 2 suffers from restaurant. Especially there is no Turkish restaurant in west part inluding Cluster 2 . We can think about Cluster 2 , West part

Cluster 0 especially likes Itallian tastes , may be this part will not like Turkish tastes . We can only think about 1 location may be .

Cluster 1, 2, 3 are centers full of restaurants as we saw in the heatmap also

**5. Conclusion**[**¶**](https://render.githubusercontent.com/view/ipynb?commit=e6e9570b1cdd64cdd8747ce5465c12383e84a517&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f74756c61796361676c6179616e2f436f7572736572615f43617073746f6e652f653665393537306231636464363463646438373437636535343635633132333833653834613531372f49424d5f41737369676e6d656e745f426174746c655f6f665f4e65696768626f72686f6f64732e6970796e62&nwo=tulaycaglayan%2FCoursera_Capstone&path=IBM_Assignment_Battle_of_Neighborhoods.ipynb&repository_id=212115637&repository_type=Repository#5.-Conclusion)

Possible Neighborhoods to set up a Turkish restaurant :

• Neighborhood : Regie , Cluster 0, Sector 6 , Population : 367760

I choosed Regie , since west part of Bucharest has not Turkish restaurant and Regie is close to center . Regie is red cluster which is same with the other 2 red markers having Turk restaurant. Red clusters populations may like Turkish food.

• Neighborhood : Bucureștii Noi , Cluster 2, Sector 1 , Population : 225453

I choosed Bucharest-noi , since west part of Bucharest has not Turkish restaurant and even not too much restaurant after 6 km distance to center . One restaurant will be good for this area.

• Neighborhood : Crângași , Cluster 2, Sector 6 , Population : 367760

I choosed Crangasi , since Crangasi and west part of Bucharest has not Turkish restaurant and from bar chart we can see that this neighborhood has not too much restaurant. Also this neighborhood is in Sector 6 and this sector has biggest population which will handle one more restaurant

• Neighborhood : Tineretului , Cluster 2, Sector 4 , Population : 287828

I choosed Tineretului , since Tineretului has not Turkish restaurant and from bar chart we can see that this neighborhood has not too much restaurant.