Halal Food Provider

In Ohio , USA

Select A location to start the business

Section One : Introduction

This project will study the data from two cointies in Ohio, then compare between them (Cuyahoga county and franklin county) to decide which area is better to start a business of Halal food provider

Why These two counties ?

These two counties are two major counties in Ohio '\n'

Franklin county :

a. As of 2019 census estimates, it is the most populous county in Ohio.

b. Columbus (a city in the county) is the Ohio state capital.

c. The most populous city in the U.S. state of Ohio

Cuyahoga county :

a. As of the 2019 United States Census estimates, it is the second most populous county in Ohio

b. Cleveland (a city in the county) with an estimated 2019 population, is the second-largest city in Ohio as estimated in 2019.

Information reference : https://en.wikipedia.org/ (https://en.wikipedia.org/)

Section Two : Data Source

the source of data will be depend on published data on the web, as well as we will depend on Fouresquare API to get the venues. So we devided the data on two three categories As below :

1. A list of all Zip codes and cities in each county :

For this requierment we depend on https://www.zipdatamaps.com/ (https://www.zipdatamaps.com/) where we find a table of Cities and zip codes for each county , We searched for Franklin County and Cuyahoga County

2. A list of zip codes and Geo coordination :

For this requierment we extract the data from https://public.opendatasoft.com (https://public.opendatasoft.com). The issue we face the list we filtered has all cities in Ohio and no way to extract only the Geo coordination points for the cities of two county we are study, (In the code and Methodology section we will talk about this issue)

3. We need to study venues depend on zip codes :

For this requierment we used Fouresquare to list the venues for the each city or zip code which will be depending on Geo Coordination points

Section Three : Methodology and Codes

We divided the code in Two parts :

First part : We work on Franklin county data

Second part : We work on Cuyahoga County data

Step One : Install the required Depedencies such as "numpy", "panda" and etc...

In [ ]:
```
# Download all the dependencies that I need - for Now and i may i download oth
ers during the project:
import numpy as np

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

import json

!conda install -c conda-forge geopy --yes
from geopy.geocoders import Nominatim

import requests
from pandas.io.json import json_normalize


import matplotlib.cm as cm
import matplotlib.colors as colors


from sklearn.cluster import KMeans

!pip install folium
import folium

print('We are good to Go ')
```

Step Two : Import the data from Website (Data Source) :

We have two Data source tables to be imported as we mentioned in the data source section

We need to install and import some libraries

In [2]:
```
import pandas as pd
import requests
from bs4 import BeautifulSoup
from tabulate import tabulate

print("We are Good to Go ")
```

```
We are Good to Go
```

Code below to read and import the data table for all zip codes and cities for each county [in part one for Franklin County and part two for Cuyahoga County]

In [3]:
```
res = requests.get("https://www.zipdatamaps.com/franklin-oh-county-zipcodes")
soup = BeautifulSoup(res.content,'lxml')
table = soup.find_all('table')
df_F = pd.read_html(str(table)) # df_F for dataframe Franklin county
df_F= (pd.DataFrame(df_F[1]))
df_F.columns = df_F.columns.droplevel()
print('The list of columns :', list(df_F.columns),'\nThe shape of table :', df
_F.shape,'\n\a Below part of the table : ' )

df_F.head()
```

```
The list of columns : ['ZIP Code', 'ZIP Code Name', 'Population', 'ZIP Type']
The shape of table : (69, 4)
 Below part of the table :
```

Out[3]:

	ZIP Code	ZIP Code Name	Population	ZIP Type
0	43002.0	Amlin	2262.0	Non-Unique
1	43004.0	Blacklick	22727.0	Non-Unique
2	43016.0	Dublin	31306.0	Non-Unique
3	43017.0	Dublin	37626.0	Non-Unique
4	43026.0	Hilliard	54017.0	Non-Unique

Code to clean the data fram : we droped the columns with data dose not add value to our study which is in columns "Population" and " Zip Type"

In [4]:
```
# Clean the Dataframe
df_F.drop(columns=['Population', 'ZIP Type'],inplace=True)
df_F.dropna(inplace=True)
print('The list of columns :', list(df_F.columns),'\nThe shape of table :', df
_F.shape,'\n\a Below part of the table : ' )
df_F.head(12)
```

```
The list of columns : ['ZIP Code', 'ZIP Code Name']
The shape of table : (68, 2)
 Below part of the table :
```

Out[4]:

	ZIP Code	ZIP Code Name
0	43002.0	Amlin
1	43004.0	Blacklick
2	43016.0	Dublin
3	43017.0	Dublin
4	43026.0	Hilliard
6	43054.0	New Albany
7	43065.0	Powell
8	43068.0	Reynoldsburg
9	43081.0	Westerville
10	43085.0	Columbus
11	43110.0	Canal Winchester
12	43119.0	Galloway

Read the data set for the Geo Coordination from from geojson file on the website https://public.opendatasoft.com (https://public.opendatasoft.com)

code below to install some libraries needed and to read data from geojson file

In [5]:
```
!pip install geopandas
import geopandas as gpd
df_G = gpd.read_file('https://public.opendatasoft.com/explore/dataset/us-zip-code-latitude-and-longitude/download/?format=geojson&refine.state=OH&timezone=America/New_York&lang=en')
print('The list of columns :', list(df_G.columns),'\nThe shape of table :', df_G.shape,'\n\a Below part of the table : ' )
df_G.head(12)
```

```
Collecting geopandas
  Downloading geopandas-0.8.1-py2.py3-none-any.whl (962 kB)
     |████████████████████████████████| 962 kB 11.6 MB/s eta 0:00:01
Collecting pyproj>=2.2.0
  Downloading pyproj-3.0.0.post1-cp37-cp37m-manylinux2010_x86_64.whl (6.4 MB)
     |████████████████████████████████| 6.4 MB 25.8 MB/s eta 0:00:01
Collecting shapely
  Downloading Shapely-1.7.1-cp37-cp37m-manylinux1_x86_64.whl (1.0 MB)
     |████████████████████████████████| 1.0 MB 51.9 MB/s eta 0:00:01
Collecting fiona
  Downloading Fiona-1.8.18-cp37-cp37m-manylinux1_x86_64.whl (14.8 MB)
     |████████████████████████████████| 14.8 MB 27.6 MB/s eta 0:00:01
Requirement already satisfied: pandas>=0.23.0 in /opt/conda/envs/Python-3.7-m
ain/lib/python3.7/site-packages (from geopandas) (1.0.5)
Requirement already satisfied: certifi in /opt/conda/envs/Python-3.7-main/li
b/python3.7/site-packages (from pyproj>=2.2.0->geopandas) (2020.12.5)
Requirement already satisfied: click<8,>=4.0 in /opt/conda/envs/Python-3.7-ma
in/lib/python3.7/site-packages (from fiona->geopandas) (7.1.2)
Collecting cligj>=0.5
  Downloading cligj-0.7.1-py3-none-any.whl (7.1 kB)
Requirement already satisfied: attrs>=17 in /opt/conda/envs/Python-3.7-main/l
ib/python3.7/site-packages (from fiona->geopandas) (19.3.0)
Requirement already satisfied: six>=1.7 in /opt/conda/envs/Python-3.7-main/li
b/python3.7/site-packages (from fiona->geopandas) (1.15.0)
Collecting munch
  Downloading munch-2.5.0-py2.py3-none-any.whl (10 kB)
Collecting click-plugins>=1.0
  Downloading click_plugins-1.1.1-py2.py3-none-any.whl (7.5 kB)
Requirement already satisfied: pytz>=2017.2 in /opt/conda/envs/Python-3.7-mai
n/lib/python3.7/site-packages (from pandas>=0.23.0->geopandas) (2020.1)
Requirement already satisfied: numpy>=1.13.3 in /opt/conda/envs/Python-3.7-ma
in/lib/python3.7/site-packages (from pandas>=0.23.0->geopandas) (1.18.5)
Requirement already satisfied: python-dateutil>=2.6.1 in /opt/conda/envs/Pyth
on-3.7-main/lib/python3.7/site-packages (from pandas>=0.23.0->geopandas) (2.
8.1)
Installing collected packages: pyproj, shapely, cligj, munch, click-plugins,
fiona, geopandas
Successfully installed click-plugins-1.1.1 cligj-0.7.1 fiona-1.8.18 geopandas
-0.8.1 munch-2.5.0 pyproj-3.0.0.post1 shapely-1.7.1
The list of columns : ['city', 'zip', 'dst', 'longitude', 'state', 'latitud
e', 'timezone', 'geometry']
The shape of table : (1498, 8)
 Below part of the table :
```

Out[5]:

	city	zip	dst	longitude	state	latitude	timezone	geometry
0	Yorkshire	45388	1	-84.479380	OH	40.328535	-5	POINT (-84.47938 40.32854)
1	Delphos	45833	1	-84.341780	OH	40.841409	-5	POINT (-84.34178 40.84141)
2	Cleveland	44185	1	-81.672797	OH	41.685744	-5	POINT (-81.67280 41.68574)
3	Willoughby	44096	1	-81.249027	OH	41.910676	-5	POINT (-81.24903 41.91068)
4	Batavia	45103	1	-84.153190	OH	39.082894	-5	POINT (-84.15319 39.08289)
5	Hebron	43098	1	-82.482659	OH	40.095148	-5	POINT (-82.48266 40.09515)
6	Lakeside Marblehead	43440	1	-82.758810	OH	41.533186	-5	POINT (-82.75881 41.53319)
7	Nankin	44848	1	-82.281695	OH	40.920617	-5	POINT (-82.28169 40.92062)
8	Cincinnati	45271	1	-84.538220	OH	39.166759	-5	POINT (-84.53822 39.16676)
9	Wheelersburg	45694	1	-82.800400	OH	38.729816	-5	POINT (-82.80040 38.72982)
10	Cleveland	44115	1	-81.671250	OH	41.493501	-5	POINT (-81.67125 41.49350)
11	Cleveland	44125	1	-81.607930	OH	41.414403	-5	POINT (-81.60793 41.41440)

Clean the data fram : we dropped the columns we dont need in our study, columns are:

'dst','state','timezone', 'geometry'

In [6]:
```
df_G.drop(columns=['dst','state','timezone', 'geometry'],inplace=True)
df_G.dropna(inplace=True)
print('The list of columns :', list(df_G.columns),'\nThe shape of table :', df
_G.shape,'\n\a Below part of the table : ' )
df_G.head(12)
```

```
The list of columns : ['city', 'zip', 'longitude', 'latitude']
The shape of table : (1498, 4)
 Below part of the table :
```

Out[6]:

	city	zip	longitude	latitude
0	Yorkshire	45388	-84.479380	40.328535
1	Delphos	45833	-84.341780	40.841409
2	Cleveland	44185	-81.672797	41.685744
3	Willoughby	44096	-81.249027	41.910676
4	Batavia	45103	-84.153190	39.082894
5	Hebron	43098	-82.482659	40.095148
6	Lakeside Marblehead	43440	-82.758810	41.533186
7	Nankin	44848	-82.281695	40.920617
8	Cincinnati	45271	-84.538220	39.166759
9	Wheelersburg	45694	-82.800400	38.729816
10	Cleveland	44115	-81.671250	41.493501
11	Cleveland	44125	-81.607930	41.414403

Code to change the zip columns ("zip" in df_G & "Zip Code" in df_F) data to int64

In [7]:
```
df_G['zip'] = df_G['zip'].apply(pd.to_numeric, errors='coerce')
#df_G = df_G.set_index(['zip'])
df_G.head(10)
```

Out[7]:

	city	zip	longitude	latitude
0	Yorkshire	45388	-84.479380	40.328535
1	Delphos	45833	-84.341780	40.841409
2	Cleveland	44185	-81.672797	41.685744
3	Willoughby	44096	-81.249027	41.910676
4	Batavia	45103	-84.153190	39.082894
5	Hebron	43098	-82.482659	40.095148
6	Lakeside Marblehead	43440	-82.758810	41.533186
7	Nankin	44848	-82.281695	40.920617
8	Cincinnati	45271	-84.538220	39.166759
9	Wheelersburg	45694	-82.800400	38.729816

In [8]:
```
datatypes = df_G["zip"].dtypes
datatypes
```

Out[8]:
```
dtype('int64')
```

In [9]:
```
# Clean the table of zip codes for Franklin
df_F['ZIP Code'] = df_F['ZIP Code'].astype(int)
df_F.rename(columns={"ZIP Code": "zip"}, inplace = 'true')
df_F.head(12)
```

Out[9]:

	zip	ZIP Code Name
0	43002	Amlin
1	43004	Blacklick
2	43016	Dublin
3	43017	Dublin
4	43026	Hilliard
6	43054	New Albany
7	43065	Powell
8	43068	Reynoldsburg
9	43081	Westerville
10	43085	Columbus
11	43110	Canal Winchester
12	43119	Galloway

In [10]:
```
datatypes = df_F["zip"].dtypes
datatypes
```

Out[10]:
```
dtype('int64')
```

In [11]:
```
#DF_Final = pd.merge(df_F, df_G, how="left", on=["ZIP Code"])
#DF_Final.head(12)
```

Merge two Data sets :

Why ?

The data imported for the zip codes and city is df_F for franklin and df_F_C for Cuyahoga.

The data imported for zip codes and Geo Coordination is df_G

but as we mentioned earlier the df_G has the Geo Coordination points for each city in Ohio.

Since we need a data set just for Franklin county cites and Cuyahoga county cities we come up with new data sets which are

1. Merge df_F and df_G to get data set for Franklin County
2. Merge df_F_C and df_G to get Data set for Cuyahoga county

In [12]:
```
DF_Final = pd.merge(df_F, df_G, how="left", on=["zip"])
DF_Final.head(12)
```

Out[12]:

	zip	ZIP Code Name	city	longitude	latitude
0	43002	Amlin	Amlin	-83.18299	40.059910
1	43004	Blacklick	Blacklick	-82.80520	39.998073
2	43016	Dublin	Dublin	-83.13983	40.089811
3	43017	Dublin	Dublin	-83.12713	40.109478
4	43026	Hilliard	Hilliard	-83.14762	40.021665
5	43054	New Albany	New Albany	-82.82462	40.080252
6	43065	Powell	Powell	-83.08415	40.155515
7	43068	Reynoldsburg	Reynoldsburg	-82.79610	39.956384
8	43081	Westerville	Westerville	-82.91606	40.109513
9	43085	Columbus	Columbus	-83.02076	40.097796
10	43110	Canal Winchester	Canal Winchester	-82.80872	39.854413
11	43119	Galloway	Galloway	-83.16670	39.939871

In [13]:
```
DF_Final.shape
DF_Final
```

Out[13]:

	zip	ZIP Code Name	city	longitude	latitude
0	43002	Amlin	Amlin	-83.182990	40.059910
1	43004	Blacklick	Blacklick	-82.805200	39.998073
2	43016	Dublin	Dublin	-83.139830	40.089811
3	43017	Dublin	Dublin	-83.127130	40.109478
4	43026	Hilliard	Hilliard	-83.147620	40.021665
5	43054	New Albany	New Albany	-82.824620	40.080252
6	43065	Powell	Powell	-83.084150	40.155515
7	43068	Reynoldsburg	Reynoldsburg	-82.796100	39.956384
8	43081	Westerville	Westerville	-82.916060	40.109513
9	43085	Columbus	Columbus	-83.020760	40.097796
10	43110	Canal Winchester	Canal Winchester	-82.808720	39.854413
11	43119	Galloway	Galloway	-83.166700	39.939871
12	43123	Grove City	Grove City	-83.079890	39.886280
13	43125	Groveport	Groveport	-82.895890	39.852363
14	43137	Lockbourne	Lockbourne	-82.981150	39.818249
15	43140	London	London	-83.436430	39.892676
16	43146	Orient	Orient	-83.146900	39.786202
17	43147	Pickerington	Pickerington	-82.756000	39.904288
18	43201	Columbus	Columbus	-83.001170	39.990764
19	43202	Columbus	Columbus	-83.008940	40.018814
20	43203	Columbus	Columbus	-82.968800	39.971800
21	43204	Columbus	Columbus	-83.082310	39.958496
22	43205	Columbus	Columbus	-82.965870	39.957265
23	43206	Columbus	Columbus	-82.980850	39.944564
24	43207	Columbus	Columbus	-82.969690	39.897695
25	43209	Columbus	Columbus	-82.928240	39.958564
26	43210	Columbus	Columbus	-83.021480	40.003681
27	43211	Columbus	Columbus	-82.971270	40.012714
28	43212	Columbus	Columbus	-83.042680	39.988114
29	43213	Columbus	Columbus	-82.874130	39.966814
30	43214	Columbus	Columbus	-83.018810	40.053063
31	43215	Columbus	Columbus	-83.004310	39.965131
32	43217	Columbus	Columbus	-82.943840	39.824831
33	43219	Columbus	Columbus	-82.925890	40.002514
34	43220	Columbus	Columbus	-83.069860	40.047273

	zip	ZIP Code Name	city	longitude	latitude
35	43221	Columbus	Columbus	-83.076550	40.020630
36	43222	Columbus	Columbus	-83.028560	39.958664
37	43223	Columbus	Columbus	-83.045580	39.935263
38	43224	Columbus	Columbus	-82.967720	40.039914
39	43227	Columbus	Columbus	-82.890630	39.944231
40	43228	Columbus	Columbus	-83.123000	39.954363
41	43229	Columbus	Columbus	-82.973260	40.085313
42	43230	Columbus	Columbus	-82.878550	40.039963
43	43231	Columbus	Columbus	-82.940640	40.076042
44	43232	Columbus	Columbus	-82.865630	39.924213
45	43235	Columbus	Columbus	-83.055670	40.099204
46	43086	Westerville	Westerville	-83.011389	39.969036
47	43109	Brice	Brice	-82.832180	39.916574
48	43126	Harrisburg	Harrisburg	-83.170850	39.810093
49	43216	Columbus	Columbus	-83.011389	39.969036
50	43218	Columbus	Columbus	-83.011389	39.969036
51	43226	Columbus	Columbus	-83.011389	39.969036
52	43234	Columbus	Columbus	-83.011389	39.969036
53	43236	Columbus	Columbus	-83.007626	40.135711
54	43069	Reynoldsburg	NaN	NaN	NaN
55	43194	Lockbourne	NaN	NaN	NaN
56	43195	Groveport	NaN	NaN	NaN
57	43199	Groveport	Groveport	-83.011389	39.969036
58	43251	Columbus	Columbus	-83.011389	39.969036
59	43260	Columbus	Columbus	-83.011389	39.969036
60	43266	Columbus	Columbus	-83.011389	39.969036
61	43268	Columbus	Columbus	-83.011389	39.969036
62	43270	Columbus	Columbus	-83.011389	39.969036
63	43271	Columbus	Columbus	-83.011389	39.969036
64	43272	Columbus	Columbus	-83.011389	39.969036
65	43279	Columbus	Columbus	-83.011389	39.969036
66	43287	Columbus	Columbus	-83.011389	39.969036
67	43291	Columbus	Columbus	-83.011389	39.969036

Clean Data Fram and Drop NAN values

In [14]:
```
#data[data.isnull()]
#DF_Final[DF_Final.isnul
DF_Final.dropna(inplace=True)
DF_Final
```

Out[14]:

	zip	ZIP Code Name	city	longitude	latitude
0	43002	Amlin	Amlin	-83.182990	40.059910
1	43004	Blacklick	Blacklick	-82.805200	39.998073
2	43016	Dublin	Dublin	-83.139830	40.089811
3	43017	Dublin	Dublin	-83.127130	40.109478
4	43026	Hilliard	Hilliard	-83.147620	40.021665
5	43054	New Albany	New Albany	-82.824620	40.080252
6	43065	Powell	Powell	-83.084150	40.155515
7	43068	Reynoldsburg	Reynoldsburg	-82.796100	39.956384
8	43081	Westerville	Westerville	-82.916060	40.109513
9	43085	Columbus	Columbus	-83.020760	40.097796
10	43110	Canal Winchester	Canal Winchester	-82.808720	39.854413
11	43119	Galloway	Galloway	-83.166700	39.939871
12	43123	Grove City	Grove City	-83.079890	39.886280
13	43125	Groveport	Groveport	-82.895890	39.852363
14	43137	Lockbourne	Lockbourne	-82.981150	39.818249
15	43140	London	London	-83.436430	39.892676
16	43146	Orient	Orient	-83.146900	39.786202
17	43147	Pickerington	Pickerington	-82.756000	39.904288
18	43201	Columbus	Columbus	-83.001170	39.990764
19	43202	Columbus	Columbus	-83.008940	40.018814
20	43203	Columbus	Columbus	-82.968800	39.971800
21	43204	Columbus	Columbus	-83.082310	39.958496
22	43205	Columbus	Columbus	-82.965870	39.957265
23	43206	Columbus	Columbus	-82.980850	39.944564
24	43207	Columbus	Columbus	-82.969690	39.897695
25	43209	Columbus	Columbus	-82.928240	39.958564
26	43210	Columbus	Columbus	-83.021480	40.003681
27	43211	Columbus	Columbus	-82.971270	40.012714
28	43212	Columbus	Columbus	-83.042680	39.988114
29	43213	Columbus	Columbus	-82.874130	39.966814
30	43214	Columbus	Columbus	-83.018810	40.053063
31	43215	Columbus	Columbus	-83.004310	39.965131
32	43217	Columbus	Columbus	-82.943840	39.824831
33	43219	Columbus	Columbus	-82.925890	40.002514
34	43220	Columbus	Columbus	-83.069860	40.047273

	zip	ZIP Code Name	city	longitude	latitude
35	43221	Columbus	Columbus	-83.076550	40.020630
36	43222	Columbus	Columbus	-83.028560	39.958664
37	43223	Columbus	Columbus	-83.045580	39.935263
38	43224	Columbus	Columbus	-82.967720	40.039914
39	43227	Columbus	Columbus	-82.890630	39.944231
40	43228	Columbus	Columbus	-83.123000	39.954363
41	43229	Columbus	Columbus	-82.973260	40.085313
42	43230	Columbus	Columbus	-82.878550	40.039963
43	43231	Columbus	Columbus	-82.940640	40.076042
44	43232	Columbus	Columbus	-82.865630	39.924213
45	43235	Columbus	Columbus	-83.055670	40.099204
46	43086	Westerville	Westerville	-83.011389	39.969036
47	43109	Brice	Brice	-82.832180	39.916574
48	43126	Harrisburg	Harrisburg	-83.170850	39.810093
49	43216	Columbus	Columbus	-83.011389	39.969036
50	43218	Columbus	Columbus	-83.011389	39.969036
51	43226	Columbus	Columbus	-83.011389	39.969036
52	43234	Columbus	Columbus	-83.011389	39.969036
53	43236	Columbus	Columbus	-83.007626	40.135711
57	43199	Groveport	Groveport	-83.011389	39.969036
58	43251	Columbus	Columbus	-83.011389	39.969036
59	43260	Columbus	Columbus	-83.011389	39.969036
60	43266	Columbus	Columbus	-83.011389	39.969036
61	43268	Columbus	Columbus	-83.011389	39.969036
62	43270	Columbus	Columbus	-83.011389	39.969036
63	43271	Columbus	Columbus	-83.011389	39.969036
64	43272	Columbus	Columbus	-83.011389	39.969036
65	43279	Columbus	Columbus	-83.011389	39.969036
66	43287	Columbus	Columbus	-83.011389	39.969036
67	43291	Columbus	Columbus	-83.011389	39.969036

Plot a map for the County

In part one for Franklin County

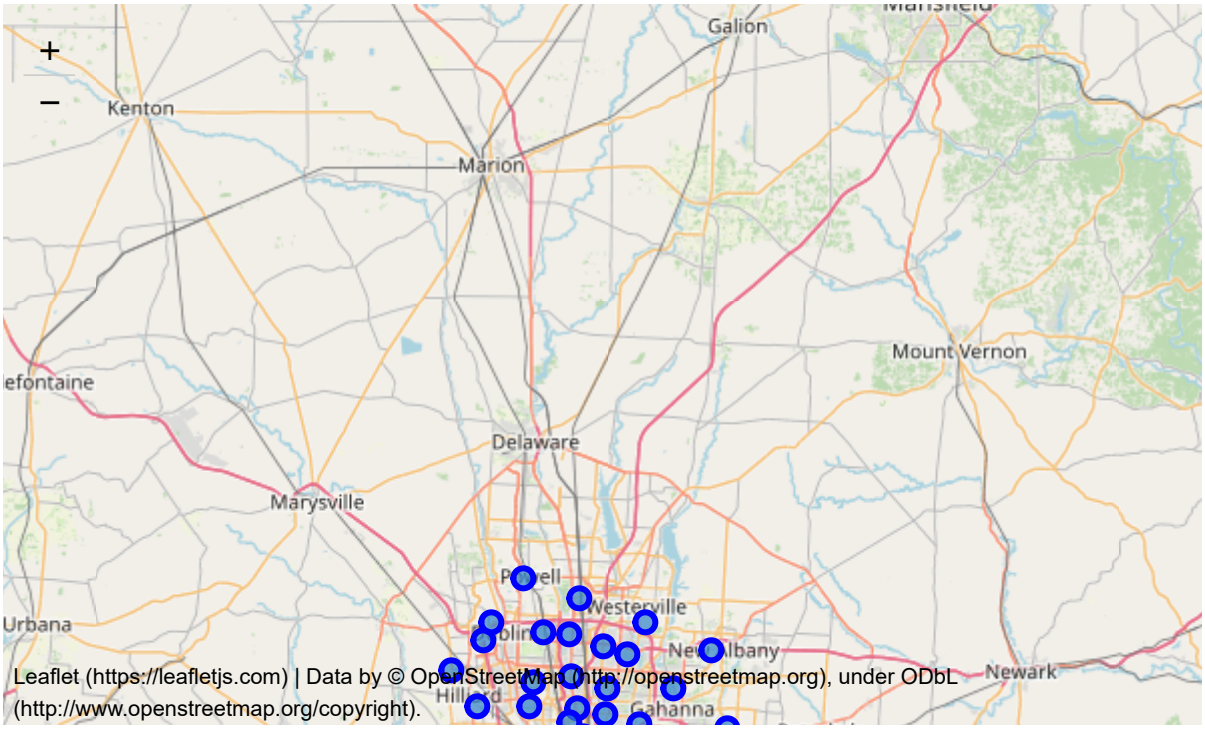
In part Two for Cuyahoga County

In [55]:
```
#create map Franklin
map_Franklin = folium.Map(location=[39.958564, -82.928240], zoom_start=10)

for lat, lng, city,code  in zip(DF_Final['latitude'], DF_Final['longitude'], D
F_Final['city'], DF_Final['zip']):
    label = '{}, {}'.format(code, city)
    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_Franklin)

map_Franklin
```

Out[55]:



Step Two : Using Foursquare to find the venues for each city

Hidden data on the below cell :

In [48]:
```
# @hidden_cell

CLIENT_ID = 'G3CWHQSIX3ENKMAYUI3NU3X1TEQOEUATGWGLPLNQH3NVWTDE'
CLIENT_SECRET = 'EO1TSYJIO4TD05PWIXFQB400X121NMGDUWBAG51MWLQHHFGB'
VERSION = '20180605'
LIMIT = 100

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

Collect the all Veneus in the County :

1. Part one for Franklin County
2. Part trwo for Cuyahoga County

In [17]:
```
## All Veneus in Franklin

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={}&clie
nt_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT)

        # make the GET request
        results = requests.get(url).json()["response"]['groups'][0]['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 = ['Neighborhood',
                  'Neighborhood Latitude',
                  'Neighborhood Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']

    return(nearby_venues)

Franklin_venues = getNearbyVenues(names=DF_Final['city'],
                                  latitudes=DF_Final['latitude'],
                                  longitudes=DF_Final['longitude']
                                 )

print(Franklin_venues.shape)
Franklin_venues.head(11)
```

```
Amlin
Blacklick
Dublin
Dublin
Hilliard
New Albany
Powell
Reynoldsburg
Westerville
Columbus
Canal Winchester
Galloway
Grove City
Groveport
Lockbourne
London
Orient
Pickerington
Columbus
Columbus
Columbus
Columbus
Columbus
Columbus
Columbus
Columbus
Columbus
Columbus
Columbus
Columbus
Columbus
Columbus
Columbus
Columbus
Columbus
Columbus
Columbus
Columbus
Columbus
Columbus
Columbus
Columbus
Columbus
Columbus
Columbus
Columbus
Westerville
Brice
Harrisburg
Columbus
Columbus
Columbus
Columbus
Columbus
Groveport
Columbus
Columbus
```

```
Columbus
Columbus
Columbus
Columbus
Columbus
Columbus
Columbus
Columbus
(761, 7)
```

Out[17]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Amlin	40.059910	-83.18299	GetGo	40.060791	-83.181626	Gas Station
1	Amlin	40.059910	-83.18299	Great Clips	40.061618	-83.181825	Business Service
2	Amlin	40.059910	-83.18299	Huntington Bank	40.061238	-83.179904	Bank
3	Amlin	40.059910	-83.18299	Redbox	40.061226	-83.179886	Video Store
4	Amlin	40.059910	-83.18299	Giant Eagle Supermarket	40.061055	-83.180127	Supermarket
5	Amlin	40.059910	-83.18299	Free To Be Me Salon	40.061848	-83.181933	Cosmetics Shop
6	Amlin	40.059910	-83.18299	Giant Eagle Supermarket	40.061257	-83.179859	Grocery Store
7	Blacklick	39.998073	-82.80520	Train Stop	39.996730	-82.810660	Light Rail Station
8	Dublin	40.089811	-83.13983	Goldfish Swim School - Dublin	40.093897	-83.140357	Swim School
9	Dublin	40.089811	-83.13983	SUBWAY	40.086861	-83.135537	Sandwich Place
10	Dublin	40.089811	-83.13983	Grainger formally Safety Solutions	40.092675	-83.139407	Sporting Goods Shop

In [18]:
```
Franklin_venues.groupby('Neighborhood').count()
```

Out[18]:

Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Amlin	7	7	7	7	7	7
Blacklick	1	1	1	1	1	1
Brice	4	4	4	4	4	4
Canal Winchester	1	1	1	1	1	1
Columbus	655	655	655	655	655	655
Dublin	11	11	11	11	11	11
Grove City	3	3	3	3	3	3
Groveport	30	30	30	30	30	30
Harrisburg	4	4	4	4	4	4
Hilliard	1	1	1	1	1	1
London	5	5	5	5	5	5
New Albany	3	3	3	3	3	3
Orient	1	1	1	1	1	1
Powell	8	8	8	8	8	8
Reynoldsburg	8	8	8	8	8	8
Westerville	19	19	19	19	19	19

Analyze Each Neighborhood

In [19]:
```
Franklin_onehot = pd.get_dummies(Franklin_venues[['Venue Category']], prefix="", prefix_sep="")

Franklin_onehot['Neighborhood'] = Franklin_venues['Neighborhood']

fixed_columns = [Franklin_onehot.columns[-1]] + list(Franklin_onehot.columns[:-1])
Franklin_onehot = Franklin_onehot[fixed_columns]

Franklin_onehot.head()
```

Out[19]:

	Neighborhood	ATM	American Restaurant	Arts & Crafts Store	Asian Restaurant	Athletics & Sports	Automotive Shop	BBQ Joint	Bakery	Ba
0	Amlin	0	0	0	0	0	0	0	0	
1	Amlin	0	0	0	0	0	0	0	0	
2	Amlin	0	0	0	0	0	0	0	0	
3	Amlin	0	0	0	0	0	0	0	0	
4	Amlin	0	0	0	0	0	0	0	0	

Important : We created a data fram with:

1. Three Venues for Franklin county ["Indian Restaurant","Mediterranean Restaurant","Turkish Restaurant"]
2. Two venues for Cuyahgo ["Indian Restaurant","Mediterranean Restaurant"]

Why ?

Our study assume that thses venues are the venues highly depend on Halal products, so we focus on spacific data.

Our result will depend to the cluster with highest sum value of those spacific venues between two county

In [20]:
```
Franklin_onehot_M = Franklin_onehot [["Neighborhood","Indian Restaurant","Mediterranean Restaurant","Turkish Restaurant"]]
Franklin_onehot_M.head()
```

Out[20]:

	Neighborhood	Indian Restaurant	Mediterranean Restaurant	Turkish Restaurant
0	Amlin	0	0	0
1	Amlin	0	0	0
2	Amlin	0	0	0
3	Amlin	0	0	0
4	Amlin	0	0	0

group rows by neighborhood and by taking the sum of the frequency of occurrence of each category

1. Part one for Franklin County
2. Part two for Cuyahogo County

In [21]:
```
Franklin_grouped = Franklin_onehot_M.groupby('Neighborhood').sum().reset_index()
Franklin_grouped
```

Out[21]:

	Neighborhood	Indian Restaurant	Mediterranean Restaurant	Turkish Restaurant
0	Amlin	0	0	0
1	Blacklick	0	0	0
2	Brice	0	0	0
3	Canal Winchester	0	0	0
4	Columbus	1	2	1
5	Dublin	0	0	0
6	Grove City	0	0	0
7	Groveport	0	0	0
8	Harrisburg	0	0	0
9	Hilliard	0	0	0
10	London	0	0	0
11	New Albany	0	0	0
12	Orient	0	0	0
13	Powell	1	0	0
14	Reynoldsburg	0	0	0
15	Westerville	0	0	0

In [22]:
```
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]
num_top_venues = 3

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

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

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

for ind in np.arange(Franklin_grouped.shape[0]):
    neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(Fran
klin_grouped.iloc[ind, :], num_top_venues)

neighborhoods_venues_sorted.head()
```

Out[22]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue
0	Amlin	Turkish Restaurant	Mediterranean Restaurant	Indian Restaurant
1	Blacklick	Turkish Restaurant	Mediterranean Restaurant	Indian Restaurant
2	Brice	Turkish Restaurant	Mediterranean Restaurant	Indian Restaurant
3	Canal Winchester	Turkish Restaurant	Mediterranean Restaurant	Indian Restaurant
4	Columbus	Mediterranean Restaurant	Turkish Restaurant	Indian Restaurant

Step Three : We will create Clusters for each county and our analysis depend on :

1. Compare the sum of freq for each venue in the same county.
2. Choose the cluster with highest sum value in the same county
3. Chosse the cluster with highest sum value between two county.

In [23]:
```
kclusters =4

Franklin_grouped_clustering = Franklin_grouped.drop('Neighborhood', 1)

kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(Franklin_grouped_clustering)

Franklin_grouped.insert(0, 'Cluster Labels', kmeans.labels_)

Franklin_merged = DF_Final

Franklin_merged = Franklin_merged.join(Franklin_grouped.set_index('Neighborhood'), on='city')

Franklin_merged.dropna(inplace=True)

Franklin_merged['Cluster Labels'] = Franklin_merged['Cluster Labels'].astype(int)

Franklin_merged.head()
```

```
/opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages/ipykernel/__main__.py:5: ConvergenceWarning: Number of distinct clusters (3) found smaller than n_clusters (4). Possibly due to duplicate points in X.
```

Out[23]:

	zip	ZIP Code Name	city	longitude	latitude	Cluster Labels	Indian Restaurant	Mediterranean Restaurant	Turkis Restaurar
0	43002	Amlin	Amlin	-83.18299	40.059910	0	0.0	0.0	0.
1	43004	Blacklick	Blacklick	-82.80520	39.998073	0	0.0	0.0	0.
2	43016	Dublin	Dublin	-83.13983	40.089811	0	0.0	0.0	0.
3	43017	Dublin	Dublin	-83.12713	40.109478	0	0.0	0.0	0.
4	43026	Hilliard	Hilliard	-83.14762	40.021665	0	0.0	0.0	0.

create cluster map

Part One for Franklin county

Part Two For Cuyahoga County

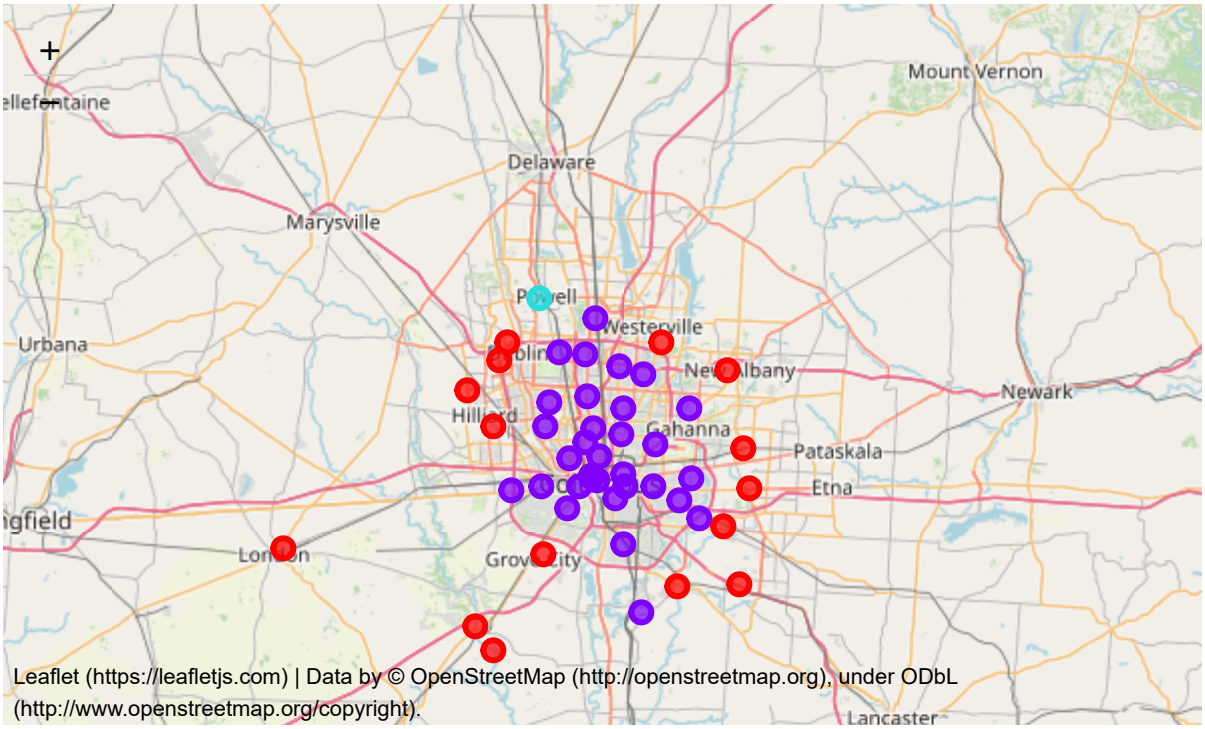
In [58]:
```
# create cluster  map
map_clusters = folium.Map(location=[39.958564, -82.928240], zoom_start=10)

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]

markers_colors = []
for lat, lon, poi, cluster in zip(Franklin_merged['latitude'], Franklin_merged
['longitude'], Franklin_merged['city'], Franklin_merged['Cluster Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=Tru
e)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)

map_clusters
```

Out[58]:



In [25]:
```
# Group by Cluster to see which cluster has the highest value
# step to clean
Franklin_merged_Cluster = Franklin_merged[["Cluster Labels","Indian Restauran
t","Mediterranean Restaurant","Turkish Restaurant"]]
Franklin_merged_Cluster.head(10)
```

Out[25]:

	Cluster Labels	Indian Restaurant	Mediterranean Restaurant	Turkish Restaurant
0	0	0.0	0.0	0.0
1	0	0.0	0.0	0.0
2	0	0.0	0.0	0.0
3	0	0.0	0.0	0.0
4	0	0.0	0.0	0.0
5	0	0.0	0.0	0.0
6	2	1.0	0.0	0.0
7	0	0.0	0.0	0.0
8	0	0.0	0.0	0.0
9	1	1.0	2.0	1.0

In [26]:
```
# Group by Cluster to see which cluster has the highest value
Franklin_merged_Cluster = Franklin_merged_Cluster.groupby('Cluster Labels').su
m().reset_index()
Franklin_merged_Cluster
```

Out[26]:

	Cluster Labels	Indian Restaurant	Mediterranean Restaurant	Turkish Restaurant
0	0	0.0	0.0	0.0
1	1	44.0	88.0	44.0
2	2	1.0	0.0	0.0

From the table above we noticed the venues of Indian , mediterranean and turkish repeated in cluster 1 more than other two cluster accordingly we are chosing cluster 1 to start the bussiness.

1. Table below to list the zip codes under cluster 1

2. Map for locations under cluster 1

In [27]:
```
# Find out which Zip codes and cities under cluster 1 as per the table below

Franklin_Cluster_1 = Franklin_merged[Franklin_merged['Cluster Labels']==1]
Franklin_Cluster_1
```

Out[27]:

	zip	ZIP Code Name	city	longitude	latitude	Cluster Labels	Indian Restaurant	Mediterranean Restaurant	T Rest
9	43085	Columbus	Columbus	-83.020760	40.097796	1	1.0	2.0	
18	43201	Columbus	Columbus	-83.001170	39.990764	1	1.0	2.0	
19	43202	Columbus	Columbus	-83.008940	40.018814	1	1.0	2.0	
20	43203	Columbus	Columbus	-82.968800	39.971800	1	1.0	2.0	
21	43204	Columbus	Columbus	-83.082310	39.958496	1	1.0	2.0	
22	43205	Columbus	Columbus	-82.965870	39.957265	1	1.0	2.0	
23	43206	Columbus	Columbus	-82.980850	39.944564	1	1.0	2.0	
24	43207	Columbus	Columbus	-82.969690	39.897695	1	1.0	2.0	
25	43209	Columbus	Columbus	-82.928240	39.958564	1	1.0	2.0	
26	43210	Columbus	Columbus	-83.021480	40.003681	1	1.0	2.0	
27	43211	Columbus	Columbus	-82.971270	40.012714	1	1.0	2.0	
28	43212	Columbus	Columbus	-83.042680	39.988114	1	1.0	2.0	
29	43213	Columbus	Columbus	-82.874130	39.966814	1	1.0	2.0	
30	43214	Columbus	Columbus	-83.018810	40.053063	1	1.0	2.0	
31	43215	Columbus	Columbus	-83.004310	39.965131	1	1.0	2.0	
32	43217	Columbus	Columbus	-82.943840	39.824831	1	1.0	2.0	
33	43219	Columbus	Columbus	-82.925890	40.002514	1	1.0	2.0	
34	43220	Columbus	Columbus	-83.069860	40.047273	1	1.0	2.0	
35	43221	Columbus	Columbus	-83.076550	40.020630	1	1.0	2.0	
36	43222	Columbus	Columbus	-83.028560	39.958664	1	1.0	2.0	
37	43223	Columbus	Columbus	-83.045580	39.935263	1	1.0	2.0	
38	43224	Columbus	Columbus	-82.967720	40.039914	1	1.0	2.0	
39	43227	Columbus	Columbus	-82.890630	39.944231	1	1.0	2.0	
40	43228	Columbus	Columbus	-83.123000	39.954363	1	1.0	2.0	
41	43229	Columbus	Columbus	-82.973260	40.085313	1	1.0	2.0	
42	43230	Columbus	Columbus	-82.878550	40.039963	1	1.0	2.0	
43	43231	Columbus	Columbus	-82.940640	40.076042	1	1.0	2.0	
44	43232	Columbus	Columbus	-82.865630	39.924213	1	1.0	2.0	
45	43235	Columbus	Columbus	-83.055670	40.099204	1	1.0	2.0	
49	43216	Columbus	Columbus	-83.011389	39.969036	1	1.0	2.0	
50	43218	Columbus	Columbus	-83.011389	39.969036	1	1.0	2.0	
51	43226	Columbus	Columbus	-83.011389	39.969036	1	1.0	2.0	
52	43234	Columbus	Columbus	-83.011389	39.969036	1	1.0	2.0	
53	43236	Columbus	Columbus	-83.007626	40.135711	1	1.0	2.0	
58	43251	Columbus	Columbus	-83.011389	39.969036	1	1.0	2.0	

	zip	ZIP Code Name	city	longitude	latitude	Cluster Labels	Indian Restaurant	Mediterranean Restaurant	T Rest
59	43260	Columbus	Columbus	-83.011389	39.969036	1	1.0	2.0	
60	43266	Columbus	Columbus	-83.011389	39.969036	1	1.0	2.0	
61	43268	Columbus	Columbus	-83.011389	39.969036	1	1.0	2.0	
62	43270	Columbus	Columbus	-83.011389	39.969036	1	1.0	2.0	
63	43271	Columbus	Columbus	-83.011389	39.969036	1	1.0	2.0	
64	43272	Columbus	Columbus	-83.011389	39.969036	1	1.0	2.0	
65	43279	Columbus	Columbus	-83.011389	39.969036	1	1.0	2.0	
66	43287	Columbus	Columbus	-83.011389	39.969036	1	1.0	2.0	
67	43291	Columbus	Columbus	-83.011389	39.969036	1	1.0	2.0	

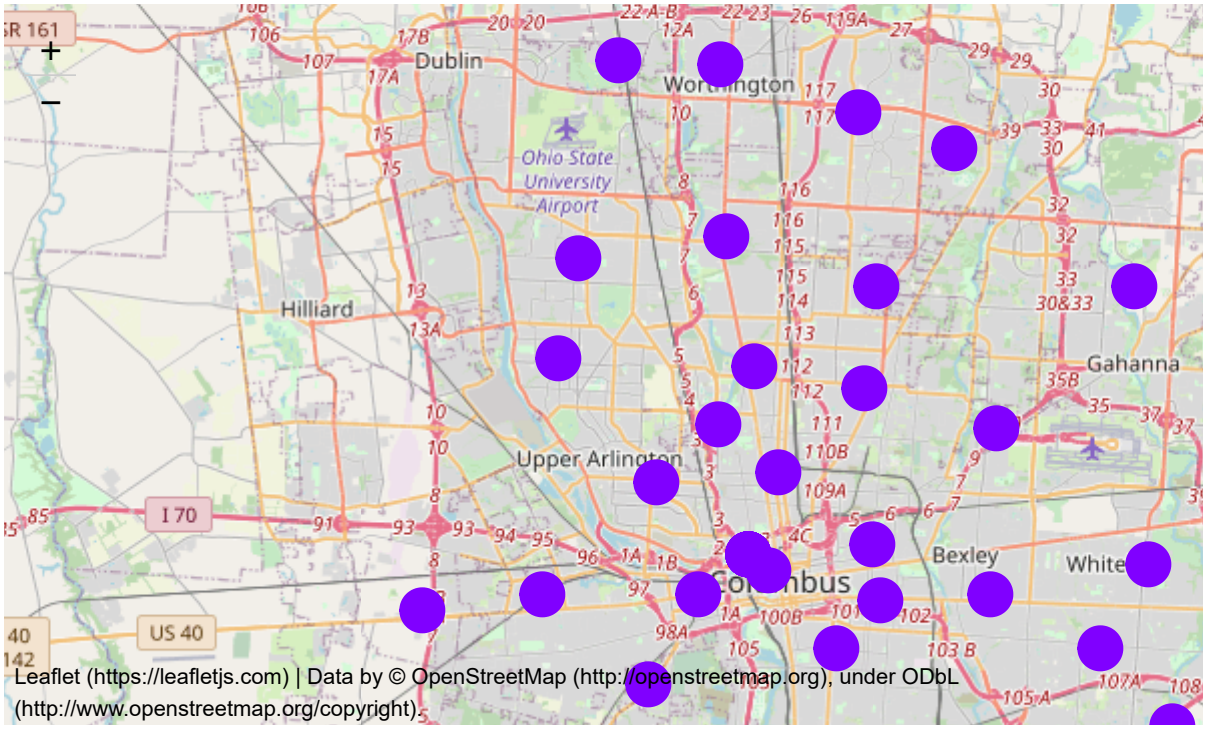
In [59]:
```
# create cluster_1  map
map_clusters_1 = folium.Map(location=[39.958564, -82.928240], zoom_start=11)

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]

markers_colors = []
for lat, lon, poi, cluster in zip(Franklin_Cluster_1['latitude'], Franklin_Clu
ster_1['longitude'], Franklin_Cluster_1['city'], Franklin_Cluster_1['Cluster L
abels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=Tru
e)
    folium.CircleMarker(
        [lat, lon],
        radius=10,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=1).add_to(map_clusters_1)

map_clusters_1
```

Out[59]:



Part Tow : Study the Cuyahoga county .

We will repeat the same steps used to analyize Franklin County Data

Download the table of zip codes of Cuyahoga county

In [29]:
```
res_1 = requests.get("https://www.zipdatamaps.com/cuyahoga-oh-county-zipcodes"
)
soup = BeautifulSoup(res_1.content,'lxml')
table = soup.find_all('table')
df_F_C = pd.read_html(str(table)) # df_F for dataframe Franklin county
df_F_C= (pd.DataFrame(df_F_C[1]))
df_F_C.columns = df_F_C.columns.droplevel()
print('The list of columns :', list(df_F_C.columns),'\nThe shape of table :',
df_F_C.shape,'\n\a Below part of the table : ' )

df_F_C.head()
```

```
The list of columns : ['ZIP Code', 'ZIP Code Name', 'Population', 'ZIP Type']
The shape of table : (65, 4)
 Below part of the table :
```

Out[29]:

	ZIP Code	ZIP Code Name	Population	ZIP Type
0	44017.0	Berea	19161.0	Non-Unique
1	44022.0	Chagrin Falls	16516.0	Non-Unique
2	44040.0	Gates Mills	3020.0	Non-Unique
3	44067.0	Northfield	20441.0	Non-Unique
4	44070.0	North Olmsted	32902.0	Non-Unique

In [30]:
```
# Clean the Dataframe : Drop Columns and remove NAN value
df_F_C.drop(columns=['Population', 'ZIP Type'],inplace=True)
df_F_C.dropna(inplace=True)
print('The list of columns :', list(df_F_C.columns),'\nThe shape of table :',
df_F_C.shape,'\n\a Below part of the table : ' )
df_F_C.head(12)
```

```
The list of columns : ['ZIP Code', 'ZIP Code Name']
The shape of table : (64, 2)
 Below part of the table :
```

Out[30]:

	ZIP Code	ZIP Code Name
0	44017.0	Berea
1	44022.0	Chagrin Falls
2	44040.0	Gates Mills
3	44067.0	Northfield
4	44070.0	North Olmsted
6	44102.0	Cleveland
7	44103.0	Cleveland
8	44104.0	Cleveland
9	44105.0	Cleveland
10	44106.0	Cleveland
11	44107.0	Lakewood
12	44108.0	Cleveland

In [31]:
```
# Clean the table of zip codes for Cuyahoga
df_F_C['ZIP Code'] = df_F_C['ZIP Code'].astype(int)
df_F_C.rename(columns={"ZIP Code": "zip"}, inplace = 'true')
df_F_C.head(12)
```

Out[31]:

	zip	ZIP Code Name
0	44017	Berea
1	44022	Chagrin Falls
2	44040	Gates Mills
3	44067	Northfield
4	44070	North Olmsted
6	44102	Cleveland
7	44103	Cleveland
8	44104	Cleveland
9	44105	Cleveland
10	44106	Cleveland
11	44107	Lakewood
12	44108	Cleveland

To Find the coordination of each zip code witll merge the dataframe for Cuyahoga county with Df_G

In [32]:
```
DF_C_Final = pd.merge(df_F_C, df_G, how="left", on=["zip"])
DF_C_Final.head(12)
```

Out[32]:

	zip	ZIP Code Name	city	longitude	latitude
0	44017	Berea	Berea	-81.86259	41.369950
1	44022	Chagrin Falls	Chagrin Falls	-81.39232	41.439320
2	44040	Gates Mills	Gates Mills	-81.41052	41.532584
3	44067	Northfield	Northfield	-81.54119	41.319704
4	44070	North Olmsted	North Olmsted	-81.91436	41.415097
5	44102	Cleveland	Cleveland	-81.73580	41.473451
6	44103	Cleveland	Cleveland	-81.64287	41.513801
7	44104	Cleveland	Cleveland	-81.62702	41.484001
8	44105	Cleveland	Cleveland	-81.62216	41.450602
9	44106	Cleveland	Cleveland	-81.60883	41.507751
10	44107	Lakewood	Lakewood	-81.80036	41.480881
11	44108	Cleveland	Cleveland	-81.60905	41.537150

In [33]:
```
DF_C_Final.shape
```

Out[33]:
```
(64, 5)
```

In [34]:
```
DF_C_Final.dropna(inplace=True)
DF_C_Final
```

Out[34]:

	zip	ZIP Code Name	city	longitude	latitude
0	44017	Berea	Berea	-81.862590	41.369950
1	44022	Chagrin Falls	Chagrin Falls	-81.392320	41.439320
2	44040	Gates Mills	Gates Mills	-81.410520	41.532584
3	44067	Northfield	Northfield	-81.541190	41.319704
4	44070	North Olmsted	North Olmsted	-81.914360	41.415097
5	44102	Cleveland	Cleveland	-81.735800	41.473451
6	44103	Cleveland	Cleveland	-81.642870	41.513801
7	44104	Cleveland	Cleveland	-81.627020	41.484001
8	44105	Cleveland	Cleveland	-81.622160	41.450602
9	44106	Cleveland	Cleveland	-81.608830	41.507751
10	44107	Lakewood	Lakewood	-81.800360	41.480881
11	44108	Cleveland	Cleveland	-81.609050	41.537150
12	44109	Cleveland	Cleveland	-81.703420	41.448951
13	44110	Cleveland	Cleveland	-81.572730	41.562781
14	44111	Cleveland	Cleveland	-81.781740	41.459399
15	44112	Cleveland	Cleveland	-81.575580	41.534101
16	44113	Cleveland	Cleveland	-81.701250	41.484688
17	44114	Cleveland	Cleveland	-81.675300	41.509880
18	44115	Cleveland	Cleveland	-81.671250	41.493501
19	44116	Rocky River	Rocky River	-81.845750	41.475297
20	44117	Euclid	Euclid	-81.527260	41.570344
21	44118	Cleveland	Cleveland	-81.554340	41.499202
22	44119	Cleveland	Cleveland	-81.545840	41.586749
23	44120	Cleveland	Cleveland	-81.582110	41.472352
24	44121	Cleveland	Cleveland	-81.533090	41.526151
25	44122	Beachwood	Beachwood	-81.522090	41.472881
26	44123	Euclid	Euclid	-81.523570	41.602498
27	44124	Cleveland	Cleveland	-81.472810	41.513752
28	44125	Cleveland	Cleveland	-81.607930	41.414403
29	44126	Cleveland	Cleveland	-81.857380	41.441697
30	44127	Cleveland	Cleveland	-81.650090	41.469951
31	44128	Cleveland	Cleveland	-81.551790	41.441153
32	44129	Cleveland	Cleveland	-81.735750	41.397701
33	44130	Cleveland	Cleveland	-81.779450	41.378051
34	44131	Independence	Independence	-81.658160	41.391753

	zip	ZIP Code Name	city	longitude	latitude
35	44132	Euclid	Euclid	-81.500410	41.608298
36	44133	North Royalton	North Royalton	-81.742040	41.313758
37	44134	Cleveland	Cleveland	-81.705070	41.393852
38	44135	Cleveland	Cleveland	-81.804670	41.432149
39	44136	Strongsville	Strongsville	-81.831970	41.312752
40	44137	Maple Heights	Maple Heights	-81.560670	41.412653
41	44138	Olmsted Falls	Olmsted Falls	-81.915070	41.373949
42	44139	Solon	Solon	-81.443650	41.388387
43	44140	Bay Village	Bay Village	-81.926580	41.484193
44	44141	Brecksville	Brecksville	-81.618350	41.307688
45	44142	Brook Park	Brookpark	-81.820550	41.400350
46	44143	Cleveland	Cleveland	-81.481750	41.550318
47	44144	Cleveland	Cleveland	-81.734880	41.437018
48	44145	Westlake	Westlake	-81.928650	41.454439
49	44146	Bedford	Bedford	-81.527090	41.389371
50	44147	Broadview Heights	Broadview Heights	-81.676590	41.321827
51	44149	Strongsville	Brunswick	-81.854005	41.316784
52	44194	Cleveland	Cleveland	-81.672797	41.685744
53	44101	Cleveland	Cleveland	-81.599648	41.523401
54	44181	Cleveland	Cleveland	-81.672797	41.685744
55	44199	Cleveland	Cleveland	-81.672797	41.685744
56	44188	Cleveland	Cleveland	-81.672797	41.685744
57	44190	Cleveland	Cleveland	-81.672797	41.685744
58	44191	Cleveland	Cleveland	-81.672797	41.685744
59	44192	Cleveland	Cleveland	-81.672797	41.685744
60	44193	Cleveland	Cleveland	-81.672797	41.685744
61	44195	Cleveland	Cleveland	-81.672797	41.685744
62	44197	Cleveland	Cleveland	-81.672797	41.685744
63	44198	Cleveland	Cleveland	-81.672797	41.685744

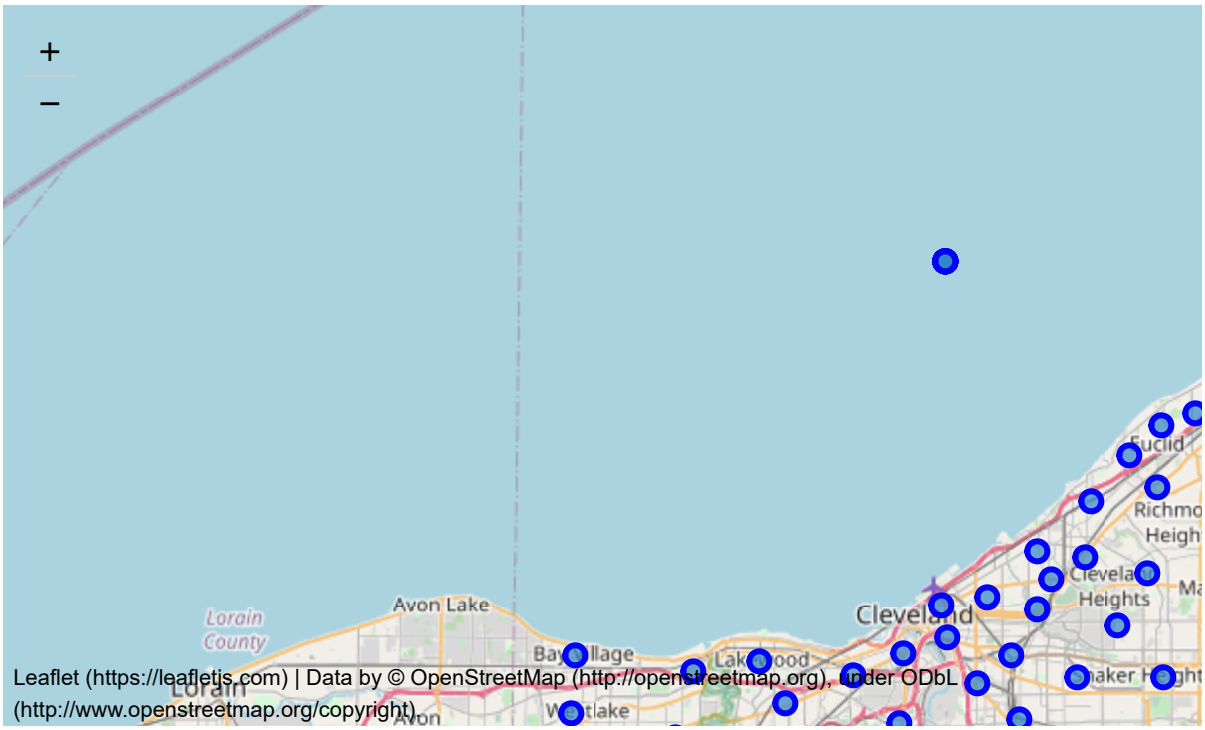
Create a map for Cuyahoga county

In [60]:
```
#create map Cuyahoga
map_Cuyahoga = folium.Map(location=[41.513801, -81.64287], zoom_start=10)

for lat, lng, city,code  in zip(DF_C_Final['latitude'], DF_C_Final['longitude'
], DF_C_Final['city'], DF_C_Final['zip']):
    label = '{}, {}'.format(code, city)
    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_Cuyahoga)

map_Cuyahoga
```

Out[60]:



Using foursquare to find the veneus in Cuyahoga County

In [36]:
```
Cuyahoga_venues = getNearbyVenues(names=DF_C_Final['city'],
                                  latitudes=DF_C_Final['latitude'],
                                  longitudes=DF_C_Final['longitude']
                                 )

print(Cuyahoga_venues.shape)
Cuyahoga_venues.head(11)
```

```
Berea
Chagrin Falls
Gates Mills
Northfield
North Olmsted
Cleveland
Cleveland
Cleveland
Cleveland
Cleveland
Lakewood
Cleveland
Cleveland
Cleveland
Cleveland
Cleveland
Cleveland
Cleveland
Cleveland
Rocky River
Euclid
Cleveland
Cleveland
Cleveland
Cleveland
Beachwood
Euclid
Cleveland
Cleveland
Cleveland
Cleveland
Cleveland
Cleveland
Cleveland
Cleveland
Independence
Euclid
North Royalton
Cleveland
Cleveland
Strongsville
Maple Heights
Olmsted Falls
Solon
Bay Village
Brecksville
Brookpark
Cleveland
Cleveland
Westlake
Bedford
Broadview Heights
Brunswick
Cleveland
Cleveland
Cleveland
Cleveland
Cleveland
```

```
Cleveland
Cleveland
Cleveland
Cleveland
Cleveland
Cleveland
Cleveland
(570, 7)
```

Out[36]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Berea	41.369950	-81.86259	Nam Wah	41.372801	-81.867126	Vietnamese Restaurant
1	Berea	41.369950	-81.86259	PNC Bank	41.371800	-81.865570	Bank
2	Berea	41.369950	-81.86259	Dollar General	41.372884	-81.867108	Discount Store
3	Berea	41.369950	-81.86259	Twice Good Wines	41.373433	-81.865712	Liquor Store
4	Berea	41.369950	-81.86259	CVS pharmacy	41.373281	-81.861017	Pharmacy
5	Berea	41.369950	-81.86259	Subway	41.371391	-81.868151	Sandwich Place
6	Berea	41.369950	-81.86259	McDonald's	41.372552	-81.866831	Fast Food Restaurant
7	Berea	41.369950	-81.86259	Zach's Steakhouse	41.373749	-81.861741	Steakhouse
8	Northfield	41.319704	-81.54119	Dragon Garden	41.316255	-81.538687	Chinese Restaurant
9	Northfield	41.319704	-81.54119	Precise Gutter Systems LLC	41.321805	-81.542382	Home Service
10	Northfield	41.319704	-81.54119	Excel Truck Service	41.322530	-81.541340	Auto Garage

In [37]:
```
Cuyahoga_venues.groupby('Neighborhood').count()
```

Out[37]:

Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
Bay Village	15	15	15	15	15	15
Beachwood	2	2	2	2	2	2
Bedford	7	7	7	7	7	7
Berea	8	8	8	8	8	8
Brookpark	7	7	7	7	7	7
Brunswick	11	11	11	11	11	11
Cleveland	305	305	305	305	305	305
Euclid	16	16	16	16	16	16
Independence	4	4	4	4	4	4
Lakewood	45	45	45	45	45	45
Maple Heights	5	5	5	5	5	5
North Olmsted	49	49	49	49	49	49
North Royalton	4	4	4	4	4	4
Northfield	7	7	7	7	7	7
Rocky River	5	5	5	5	5	5
Solon	28	28	28	28	28	28
Strongsville	51	51	51	51	51	51
Westlake	1	1	1	1	1	1

look at each zip code area

In [38]:
```
Cuyahoga_onehot = pd.get_dummies(Cuyahoga_venues[['Venue Category']], prefix="", prefix_sep="")

Cuyahoga_onehot['Neighborhood'] = Cuyahoga_venues['Neighborhood']

fixed_columns = [Cuyahoga_onehot.columns[-1]] + list(Cuyahoga_onehot.columns[:-1])
Cuyahoga_onehot = Cuyahoga_onehot[fixed_columns]

Cuyahoga_onehot.head()
```

Out[38]:

	Neighborhood	Accessories Store	American Restaurant	Antique Shop	Art Gallery	Art Museum	Arts & Crafts Store	Asian Restaurant	Athlet & Spo
0	Berea	0	0	0	0	0	0	0	
1	Berea	0	0	0	0	0	0	0	
2	Berea	0	0	0	0	0	0	0	
3	Berea	0	0	0	0	0	0	0	
4	Berea	0	0	0	0	0	0	0	

In [39]:
```
Cuyahoga_onehot_M = Cuyahoga_onehot [["Neighborhood","Indian Restaurant","Mediterranean Restaurant"]]
Cuyahoga_onehot_M.head()
```

Out[39]:

	Neighborhood	Indian Restaurant	Mediterranean Restaurant
0	Berea	0	0
1	Berea	0	0
2	Berea	0	0
3	Berea	0	0
4	Berea	0	0

Grouped by Neighborhood and take the sum of the frequancy of occurance

In [40]:
```
Cuyahoga_grouped = Cuyahoga_onehot_M.groupby('Neighborhood').sum().reset_index()
Cuyahoga_grouped
```

Out[40]:

	Neighborhood	Indian Restaurant	Mediterranean Restaurant
0	Bay Village	0	0
1	Beachwood	0	0
2	Bedford	0	0
3	Berea	0	0
4	Brookpark	0	0
5	Brunswick	0	0
6	Cleveland	2	1
7	Euclid	0	0
8	Independence	0	0
9	Lakewood	0	0
10	Maple Heights	0	0
11	North Olmsted	2	0
12	North Royalton	0	0
13	Northfield	0	0
14	Rocky River	0	0
15	Solon	0	0
16	Strongsville	0	0
17	Westlake	0	0

In [41]:
```
num_top_venues = 2

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

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

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

for ind in np.arange(Cuyahoga_grouped.shape[0]):
    neighborhoods_venues_sorted_C.iloc[ind, 1:] = return_most_common_venues(Cu
yahoga_grouped.iloc[ind, :], num_top_venues)

neighborhoods_venues_sorted_C.head()
```

Out[41]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue
0	Bay Village	Mediterranean Restaurant	Indian Restaurant
1	Beachwood	Mediterranean Restaurant	Indian Restaurant
2	Bedford	Mediterranean Restaurant	Indian Restaurant
3	Berea	Mediterranean Restaurant	Indian Restaurant
4	Brookpark	Mediterranean Restaurant	Indian Restaurant

clustering

In [42]:
```
kclusters =4

Cuyahoga_grouped_clustering = Cuyahoga_grouped.drop('Neighborhood', 1)

kmeans_1 = KMeans(n_clusters=kclusters, random_state=0).fit(Cuyahoga_grouped_clustering)

Cuyahoga_grouped.insert(0, 'Cluster Labels', kmeans_1.labels_)

Cuyahoga_merged = DF_C_Final

Cuyahoga_merged = Cuyahoga_merged.join(Cuyahoga_grouped.set_index('Neighborhood'), on='city')

Cuyahoga_merged.dropna(inplace=True)

Cuyahoga_merged['Cluster Labels'] = Cuyahoga_merged['Cluster Labels'].astype(int)

Cuyahoga_merged.head()
```

```
/opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages/ipykernel/__main__.py:5: ConvergenceWarning: Number of distinct clusters (3) found smaller than n_clusters (4). Possibly due to duplicate points in X.
```

Out[42]:

	zip	ZIP Code Name	city	longitude	latitude	Cluster Labels	Indian Restaurant	Mediterranean Restaurant
0	44017	Berea	Berea	-81.86259	41.369950	0	0.0	0.0
3	44067	Northfield	Northfield	-81.54119	41.319704	0	0.0	0.0
4	44070	North Olmsted	North Olmsted	-81.91436	41.415097	2	2.0	0.0
5	44102	Cleveland	Cleveland	-81.73580	41.473451	1	2.0	1.0
6	44103	Cleveland	Cleveland	-81.64287	41.513801	1	2.0	1.0

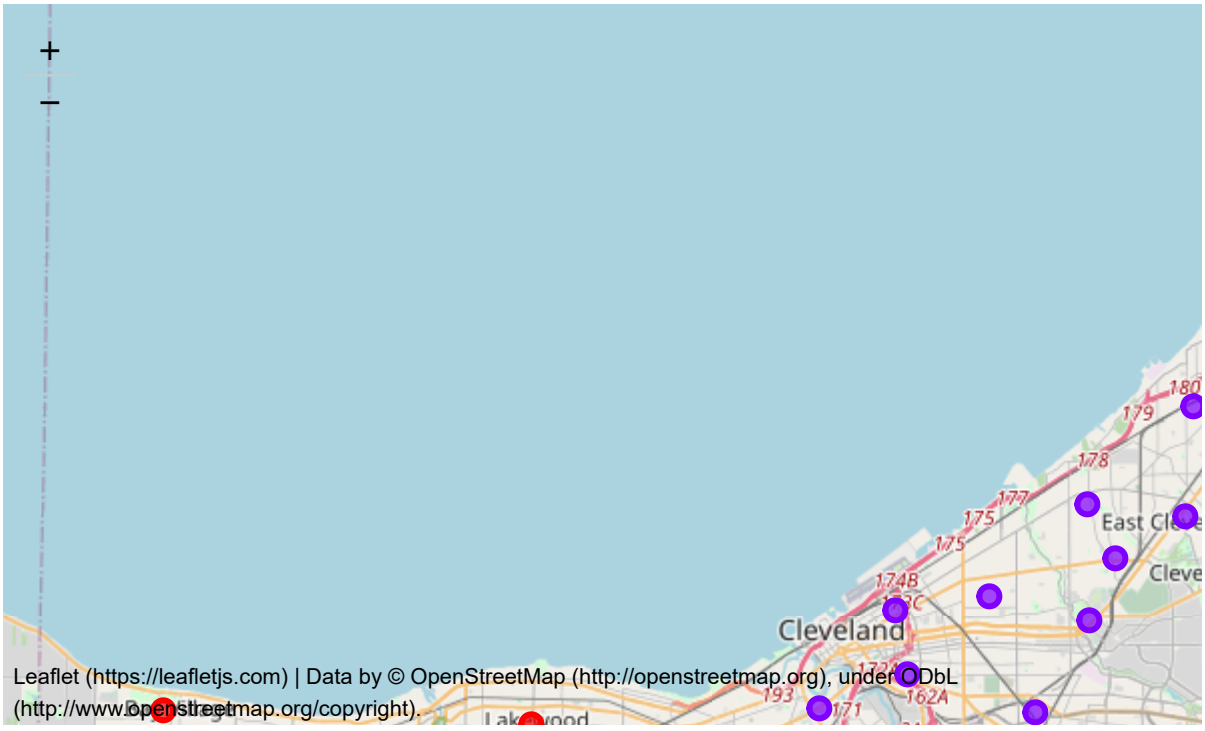
In [61]:
```
# create cluster  map for Cuyahoga
map_clusters_C = folium.Map(location=[41.513801, -81.64287], zoom_start=11)

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]

markers_colors = []
for lat, lon, poi, cluster in zip(Cuyahoga_merged['latitude'], Cuyahoga_merged
['longitude'], Cuyahoga_merged['city'], Cuyahoga_merged['Cluster Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=Tru
e)
    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_C)

map_clusters_C
```

Out[61]:



In [44]:
```
Cuyahoga_merged_Cluster = Cuyahoga_merged[["Cluster Labels","Indian Restaurant","Mediterranean Restaurant"]]
Cuyahoga_merged_Cluster.head(10)
```

Out[44]:

	Cluster Labels	Indian Restaurant	Mediterranean Restaurant
0	0	0.0	0.0
3	0	0.0	0.0
4	2	2.0	0.0
5	1	2.0	1.0
6	1	2.0	1.0
7	1	2.0	1.0
8	1	2.0	1.0
9	1	2.0	1.0
10	0	0.0	0.0
11	1	2.0	1.0

In [45]:
```
# Group by Cluster to see which cluster has the highest value
Cuyahoga_merged_Cluster = Cuyahoga_merged_Cluster.groupby('Cluster Labels').sum().reset_index()
Cuyahoga_merged_Cluster
```

Out[45]:

	Cluster Labels	Indian Restaurant	Mediterranean Restaurant
0	0	0.0	0.0
1	1	80.0	40.0
2	2	2.0	0.0

We noticed Cluster 1 in Cuyahoga county has the highest Value

In [46]:
```
# List of zip codes under cluster 1 which has more frequency of occourance

Cuyahoga_Cluster_1 = Cuyahoga_merged[Cuyahoga_merged['Cluster Labels']==1]
Cuyahoga_Cluster_1
```

Out[46]:

	zip	ZIP Code Name	city	longitude	latitude	Cluster Labels	Indian Restaurant	Mediterranean Restaurant
5	44102	Cleveland	Cleveland	-81.735800	41.473451	1	2.0	1.0
6	44103	Cleveland	Cleveland	-81.642870	41.513801	1	2.0	1.0
7	44104	Cleveland	Cleveland	-81.627020	41.484001	1	2.0	1.0
8	44105	Cleveland	Cleveland	-81.622160	41.450602	1	2.0	1.0
9	44106	Cleveland	Cleveland	-81.608830	41.507751	1	2.0	1.0
11	44108	Cleveland	Cleveland	-81.609050	41.537150	1	2.0	1.0
12	44109	Cleveland	Cleveland	-81.703420	41.448951	1	2.0	1.0
13	44110	Cleveland	Cleveland	-81.572730	41.562781	1	2.0	1.0
14	44111	Cleveland	Cleveland	-81.781740	41.459399	1	2.0	1.0
15	44112	Cleveland	Cleveland	-81.575580	41.534101	1	2.0	1.0
16	44113	Cleveland	Cleveland	-81.701250	41.484688	1	2.0	1.0
17	44114	Cleveland	Cleveland	-81.675300	41.509880	1	2.0	1.0
18	44115	Cleveland	Cleveland	-81.671250	41.493501	1	2.0	1.0
21	44118	Cleveland	Cleveland	-81.554340	41.499202	1	2.0	1.0
22	44119	Cleveland	Cleveland	-81.545840	41.586749	1	2.0	1.0
23	44120	Cleveland	Cleveland	-81.582110	41.472352	1	2.0	1.0
24	44121	Cleveland	Cleveland	-81.533090	41.526151	1	2.0	1.0
27	44124	Cleveland	Cleveland	-81.472810	41.513752	1	2.0	1.0
28	44125	Cleveland	Cleveland	-81.607930	41.414403	1	2.0	1.0
29	44126	Cleveland	Cleveland	-81.857380	41.441697	1	2.0	1.0
30	44127	Cleveland	Cleveland	-81.650090	41.469951	1	2.0	1.0
31	44128	Cleveland	Cleveland	-81.551790	41.441153	1	2.0	1.0
32	44129	Cleveland	Cleveland	-81.735750	41.397701	1	2.0	1.0
33	44130	Cleveland	Cleveland	-81.779450	41.378051	1	2.0	1.0
37	44134	Cleveland	Cleveland	-81.705070	41.393852	1	2.0	1.0
38	44135	Cleveland	Cleveland	-81.804670	41.432149	1	2.0	1.0
46	44143	Cleveland	Cleveland	-81.481750	41.550318	1	2.0	1.0
47	44144	Cleveland	Cleveland	-81.734880	41.437018	1	2.0	1.0
52	44194	Cleveland	Cleveland	-81.672797	41.685744	1	2.0	1.0
53	44101	Cleveland	Cleveland	-81.599648	41.523401	1	2.0	1.0
54	44181	Cleveland	Cleveland	-81.672797	41.685744	1	2.0	1.0
55	44199	Cleveland	Cleveland	-81.672797	41.685744	1	2.0	1.0
56	44188	Cleveland	Cleveland	-81.672797	41.685744	1	2.0	1.0
57	44190	Cleveland	Cleveland	-81.672797	41.685744	1	2.0	1.0
58	44191	Cleveland	Cleveland	-81.672797	41.685744	1	2.0	1.0

	zip	ZIP Code Name	city	longitude	latitude	Cluster Labels	Indian Restaurant	Mediterranean Restaurant
59	44192	Cleveland	Cleveland	-81.672797	41.685744	1	2.0	1.0
60	44193	Cleveland	Cleveland	-81.672797	41.685744	1	2.0	1.0
61	44195	Cleveland	Cleveland	-81.672797	41.685744	1	2.0	1.0
62	44197	Cleveland	Cleveland	-81.672797	41.685744	1	2.0	1.0
63	44198	Cleveland	Cleveland	-81.672797	41.685744	1	2.0	1.0

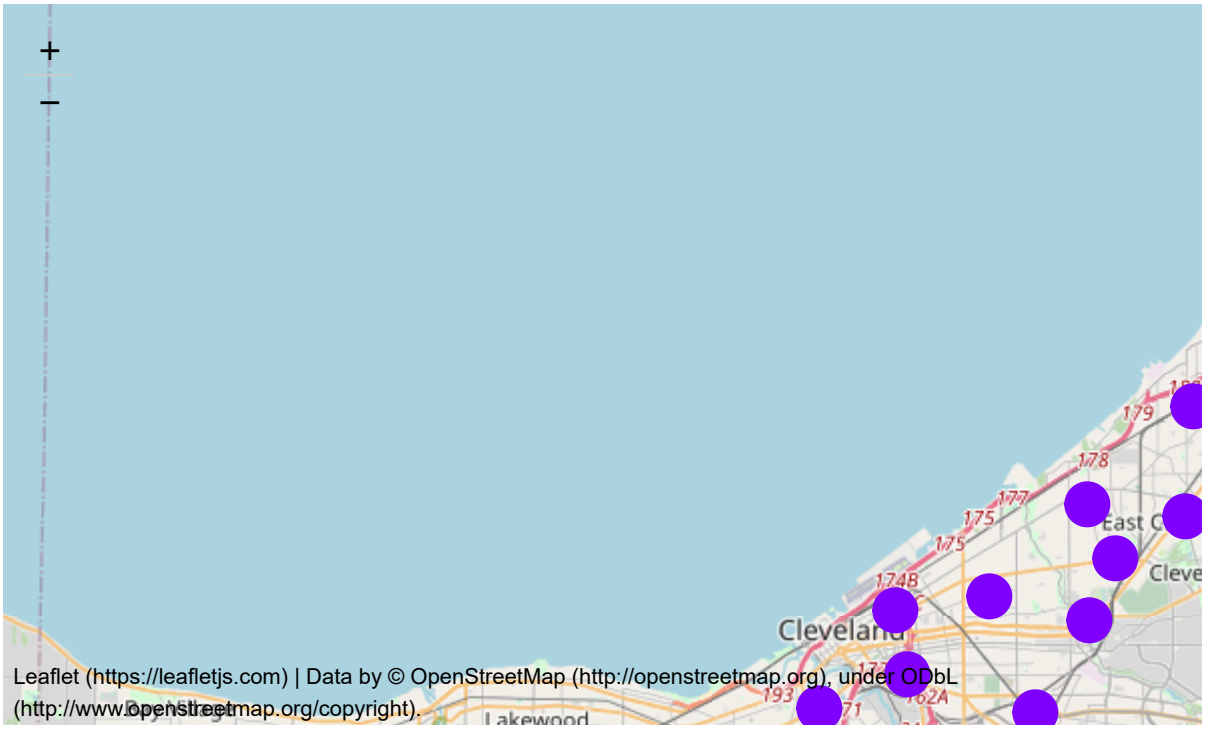
In [62]:
```
# create cluster_1  map in Cuyahoga
map_clusters_C_1 = folium.Map(location=[41.513801, -81.64287], zoom_start=11)

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(Cuyahoga_Cluster_1['latitude'], Cuyahoga_Clu
ster_1['longitude'], Cuyahoga_Cluster_1['city'], Cuyahoga_Cluster_1['Cluster L
abels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=Tru
e)
    folium.CircleMarker(
        [lat, lon],
        radius=10,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=1).add_to(map_clusters_C_1)

map_clusters_C_1
```

Out[62]:



Section Three : Results section & Discussion section

We Looked To Cluster Data from Franklin County

In [49]:
```
Franklin_merged_Cluster
```

Out[49]:

	Cluster Labels	Indian Restaurant	Mediterranean Restaurant	Turkish Restaurant
0	0	0.0	0.0	0.0
1	1	44.0	88.0	44.0
2	2	1.0	0.0	0.0

Result : in Franklin County the cluster 1 area is the best to start the business of Halal provider since the freq. of three venues is the highest which mean high demand on halal product on this cluster

We Loked to Cluster Data from Cuyahoga County

In [50]:
```
Cuyahoga_merged_Cluster
```

Out[50]:

	Cluster Labels	Indian Restaurant	Mediterranean Restaurant
0	0	0.0	0.0
1	1	80.0	40.0
2	2	2.0	0.0

Result : in Cuyahoga County the cluster 1 area is the best to start the business of Halal provider since the freq. of two venues is the highest which mean high demand on halal product on this cluster

Section Four : Recommandation

As start will be so important to notice the following :

The cities in cluster 1 in Franklin county is the best area to open the business.

In [63]:
```
## thanks Am
```

In [ ]: