The Battle of 'Neighborhoods'

Liverpool & Manchester

OPENING OF A RESTAURANT

BUSINESS IDEA:

- Recommending a firm to open a restaurant in Liverpool city or Manchester.
- Shortlisting of places based on the present competition in both the cities by looking for neighbourhoods having lesser frequency of restaurants.

TARGET AUDIENCE: a FIRM asking for recommendation for opening a restaurant in Liverpool or Manchester.

METHOD | TARGET:

- **if (the frequencies of restaurants in a neighbourhood is less) :**

  lesser competition + more benefits of opening a restaurant in that location.

DESCRIPTION

- Extracting the neighbourhood & coordinates of a city.
- Searching for restaurants in the nearby areas and extracting it for each neighbourhood.
- Applying k-means to cluster these locations based on the frequency of restaurants available.
- Displaying them on a map.
- Repeat the same for the next desired city.
- Shortlist the neighbourhood for both the cities.

THE DATA SECTION

- Wikipedia: extracting neighbourhood for the locations.
- Foursquare: extracting venues for each neighbourhood.

WIKIPEDIA: Here are the codes used to extract data from Wikipedia and store it in a dataframe

Importing Libraries and scraping data from Wikipedia ¶

In [8]:
```
# import the library we use to open URLs
import urllib.request
# specify which URL/web page we are going to be scraping
url = "https://en.wikipedia.org/wiki/Category:Areas_of_Liverpool"
# open the url using urllib.request and put the HTML into the page variable
page = urllib.request.urlopen(url)
# import the BeautifulSoup library so we can parse HTML and XML documents
from bs4 import BeautifulSoup
# parse the HTML from our URL into the BeautifulSoup parse tree format
soup = BeautifulSoup(page, "lxml")
#Then we use Beautiful Soup to parse the HTML data we stored in our 'url' variable and store it in a new variable called 'soup' in the Beautiful Soup forn
#Jupyter Notebook prefers we specify a parser format so we use the "lxml" library option
#print(soup.prettify())
#to beautify the way data is presented
import pandas as pd
```

Printing title and viewing it

In [9]:
```
soup.title.string
```

Out[9]: 'Category:Areas of Liverpool - Wikipedia'

In [10]:
```
print(soup.prettify())
```

RESULTS OF THE DATA RECEIVED:

```
      Aigburth
     </a>
    </li>
    <li>
     <a href="/wiki/Allerton,_Liverpool" title="Allerton, Liverpool">
      Allerton, Liverpool
     </a>
    </li>
    <li>
     <a href="/wiki/Anfield_(suburb)" title="Anfield (suburb)">
      Anfield (suburb)
     </a>
    </li>
   </ul>
  </div>
  <div class="mw-category-group">
   <h3>
    B
   </h3>
```

Extracting data

In [11]:
```
# create a list to store neighborhood data
neighborhoodList = []
```

In [12]:
```
# append the data into the list
for row in soup.find_all("div", class_="mw-category")[0].findAll("li"):
    neighborhoodList.append(row.text)
```

In [13]:
```
# create a new DataFrame from the list
lp_df = pd.DataFrame({"Neighborhood": neighborhoodList})

lp_df.head()
```

Out[13]:

	Neighborhood
0	Aigburth
1	Allerton, Liverpool
2	Anfield (suburb)
3	Belle Vale, Liverpool
4	Broadgreen

Creating a Dataframe

In [22]:
```
import numpy as np
import pandas as pd
data={'Neighborhood': n,
      'Latitude': c,
      'Longitude': d}
df= pd.DataFrame(data)
df.head()
```

Out[22]:

	Neighborhood	Latitude	Longitude
0	Aigburth	53.369504	-2.931818
1	Allerton	39.915319	-87.933215
2	Anfield	53.430836	-2.960910
3	Belle Vale	53.395074	-2.864178
4	Broadgreen	51.564941	-1.777782

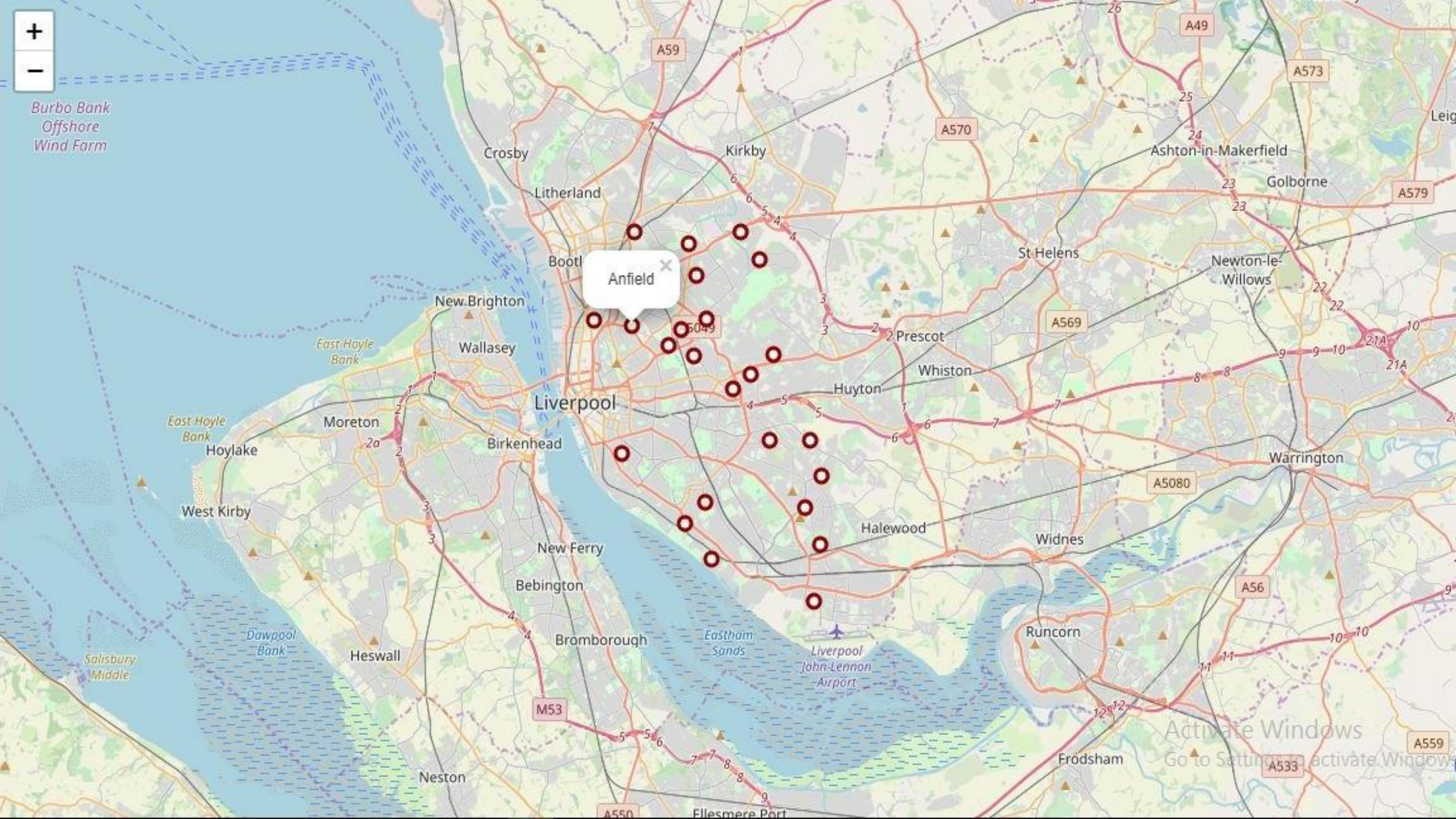
Displaying the Neighborhoods

Getting a map of Liverpool

In [24]:
```
# create map of Liverpool using latitude and longitude values
map_lp = folium.Map(location=[latitude, longitude], zoom_start=11)

# add markers to map
for lat, lng, neighborhood in zip(df['Latitude'], df['Longitude'], df['Neighborhood']):
    label = '{}'.format(neighborhood)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='darkred',
        fill=True,
        fill_color='white',
        fill_opacity=0.7).add_to(map_lp)

map_lp
```


+
−
Burbo Bank Offshore Wind Farm
Crosby
Litherland
Kirkby
Bootl
Anfield
New Brighton
Wallasey
East Hoyle Bank
Fast Hoyle Bank
Liverpool
Moreton
Hoylake
Birkenhead
West Kirby
New Ferry
Bebington
Bromborough
Heswall
Dawpool Bank
Salisbury Middle
Neston
Eastham Sands
Liverpool John Lennon Airport
Ellesmere Port
Halewood
Huyton
Whiston
Prescot
St Helens
Ashton-in-Makerfield
Golborne
Newton-le-Willows
Warrington
Widnes
Runcorn
Frodsham
A59
A570
A49
A573
A579
A569
A5080
A56
A559
A533
A550
M53
Activate Windows
Go to Settings to activate Windows

FOURSQUARE: Here are the codes utilising Foursquare API calls:

In [26]:
```
radius = 2000
LIMIT = 100

venues = []

for lat, long, neighborhood in zip(df['Latitude'], df['Longitude'], df['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']['location']['lat'],
            venue['venue']['location']['lng'],
            venue['venue']['categories'][0]['name']))
```

Storing the data collected in a data frame and viewing the categories.

In [27]:
```python
# convert the venues list into a new DataFrame
venues_df = pd.DataFrame(venues)

# define the column names
venues_df.columns = ['Neighborhood', 'Latitude', 'Longitude', 'VenueName', 'VenueLatitude', 'VenueLongitude', 'VenueCategory']

print(venues_df.shape)
venues_df.head()
```

(1656, 7)

Out[27]:

	Neighborhood	Latitude	Longitude	VenueName	VenueLatitude	VenueLongitude	VenueCategory
0	Aigburth	53.369504	-2.931818	Otterspool Promenade	53.362505	-2.931786	Other Great Outdoors
1	Aigburth	53.369504	-2.931818	Sefton Park	53.381713	-2.936611	Park
2	Aigburth	53.369504	-2.931818	Steves Chippy	53.373487	-2.934005	Fast Food Restaurant
3	Aigburth	53.369504	-2.931818	The Palm House	53.381339	-2.935269	Botanical Garden
4	Aigburth	53.369504	-2.931818	Mossley Hill Athletics Club	53.374798	-2.919895	Athletics & Sports

In [28]:
```python
# print out the list of categories
venues_df['VenueCategory'].unique()[:50]
```

Out[28]:
```
array(['Other Great Outdoors', 'Park', 'Fast Food Restaurant',
       'Botanical Garden', 'Athletics & Sports', 'Turkish Restaurant',
       'Historic Site', 'Restaurant', 'Indian Restaurant', 'Wine Bar',
       'Bar', 'Italian Restaurant', 'Café', 'Gym / Fitness Center',
       'Cricket Ground', 'Discount Store', 'Pharmacy',
       'English Restaurant', 'Grocery Store', 'Sandwich Place', 'Pub',
       'Playground', 'Gas Station', 'Tapas Restaurant', 'Supermarket',
       'Coffee Shop', 'Hotel', 'Pizza Place', 'Train Station',
       'Outdoor Sculpture', 'Convenience Store', 'Fish & Chips Shop',
       'Gastropub', 'Tennis Court', 'Ice Cream Shop', 'Music Venue',
       'Music Store', 'Gift Shop', 'Soccer Stadium', 'Souvenir Shop',
```

In [29]:
```
# one hot encoding
lp_onehot = pd.get_dummies(venues_df[['VenueCategory']], prefix="", prefix_sep="")

# add neighborhood column back to dataframe
lp_onehot['Neighborhoods'] = venues_df['Neighborhood']

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

lp_onehot.head()
```

Out[29]:

	Neighborhoods	Afghan Restaurant	African Restaurant	Airport	Airport Lounge	American Restaurant	Antique Shop	Art Gallery	Art Museum	Arts & Crafts Store	...	University	Vegetarian / Vegan Restaurant	Video Game Store	Vietnamese Restaurant	Warehouse Store	Wine Bar	Wine Shop	Wom S
0	Aigburth	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
1	Aigburth	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
2	Aigburth	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
3	Aigburth	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
4	Aigburth	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	

5 rows × 232 columns

Taking the frequencies of each venue in a location

```
In [30]: lp_grouped = lp_onehot.groupby(["Neighborhoods"]).mean().reset_index()
         lp_grouped.head(10)
```

Out[30]:

	Neighborhoods	Afghan Restaurant	African Restaurant	Airport	Airport Lounge	American Restaurant	Antique Shop	Art Gallery	Art Museum	Arts & Crafts Store	...	University	Vegetarian / Vegan Restaurant	Video Game Store	Vietnamese Restaurant	Warehouse Store	Wine Bar	Wine Shop
0	Aigburth	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	...	0.0	0.00	0.000000	0.00	0.000000	0.037736	0.00
1	Allerton	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	...	0.0	0.00	0.000000	0.00	0.000000	0.000000	0.00
2	Anfield	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	...	0.0	0.00	0.000000	0.00	0.019608	0.000000	0.00
3	Belle Vale	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	...	0.0	0.00	0.000000	0.00	0.043478	0.000000	0.00
4	Broadgreen	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	...	0.0	0.00	0.010101	0.00	0.020202	0.000000	0.00
5	Canning	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	...	0.0	0.00	0.000000	0.00	0.000000	0.000000	0.00
6	Childwall	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	...	0.0	0.00	0.000000	0.00	0.000000	0.000000	0.00
7	Chinatown	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.01	...	0.0	0.02	0.000000	0.01	0.000000	0.010000	0.02
8	Clubmoor	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	...	0.0	0.00	0.000000	0.00	0.023256	0.000000	0.00
9	Croxteth	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	...	0.0	0.00	0.000000	0.00	0.000000	0.000000	0.00

10 rows × 232 columns

Selecting Restaurant as search query

In [31]:
```
lp_food = lp_grouped[["Neighborhoods","Restaurant"]]
```

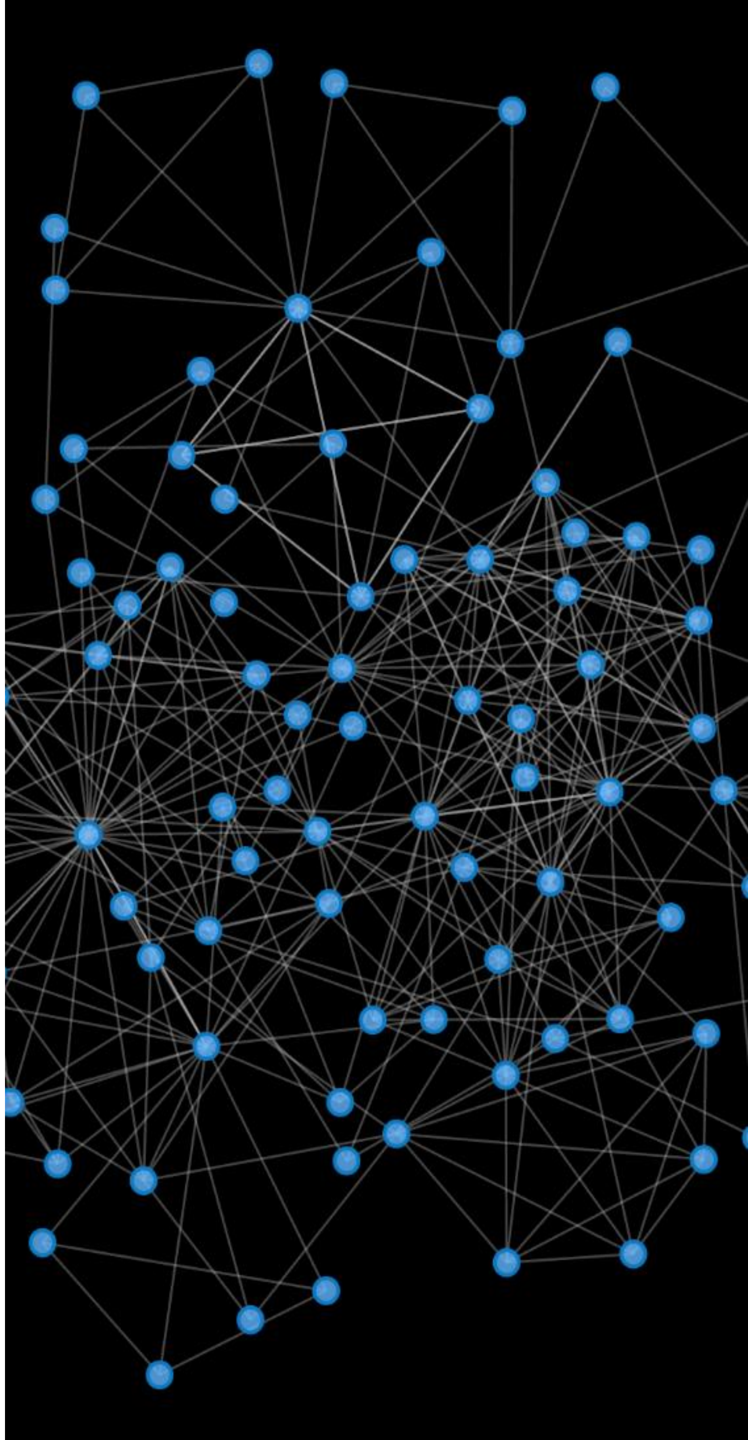
In [32]:
```
lp_food.head()
```

Out[32]:

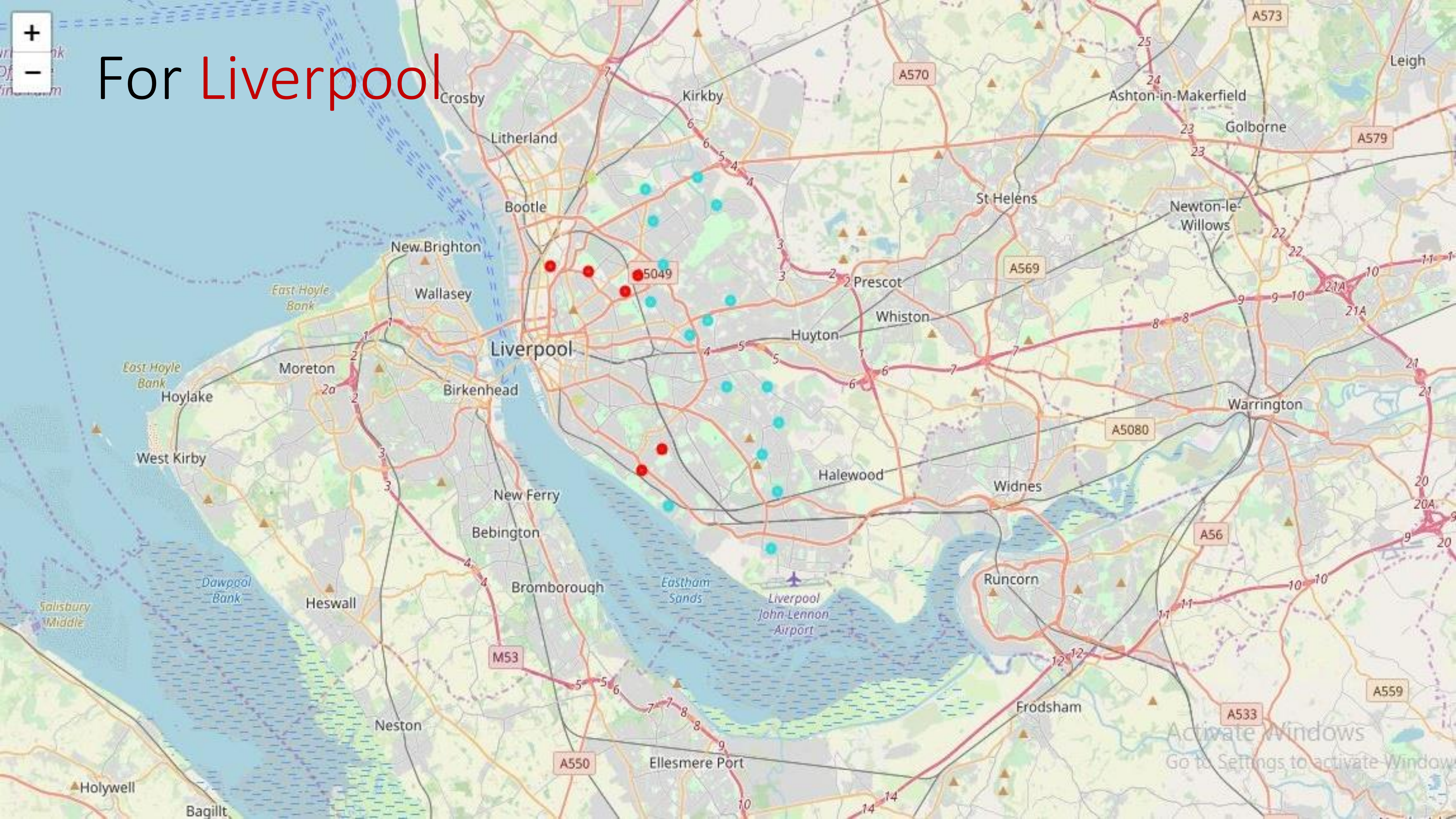
	Neighborhoods	Restaurant
0	Aigburth	0.018868
1	Allerton	0.000000
2	Anfield	0.019608
3	Belle Vale	0.000000
4	Broadgreen	0.010101

METHODOLOGY

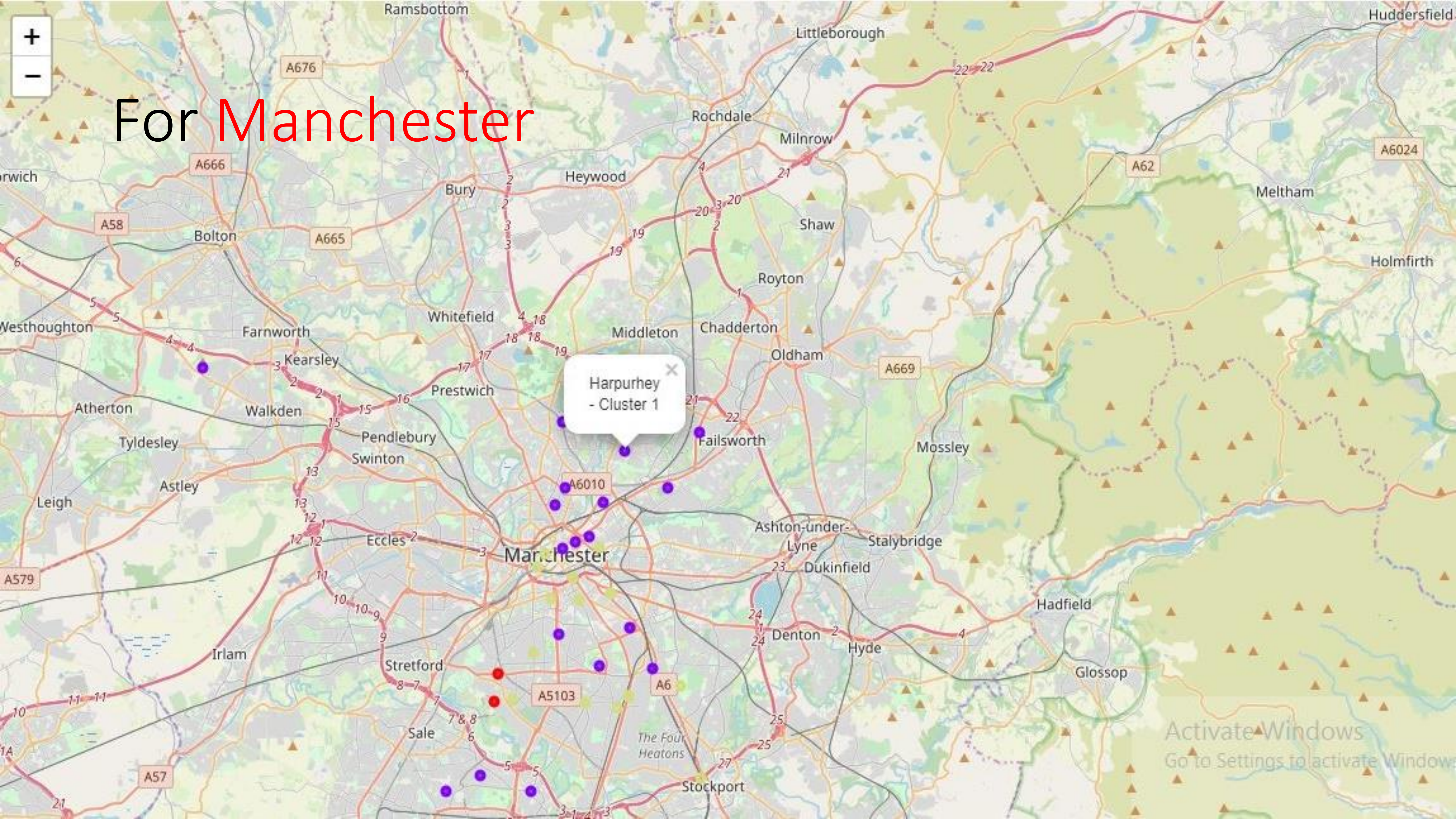
- In this project the use of k-means clustering is done.
- One of the algorithms that can be used for segmentation is K-means clustering.
- K-means can group data only unsupervised based on the similarity of customers to each other.
- K-means is a type of partitioning clustering.
- That is, it divides the data into k non-overlapping subsets or clusters without any cluster internal structure or labels. This means, it's an unsupervised algorithm.
- Objects within a cluster are very similar and objects across different clusters are very different or dissimilar.



For Liverpool



For Manchester



RESULT: Shortlisting the places

- In the end the neighbourhood from both the cities having less frequency of restaurants were chosen.
- This closes the loop, satisfying the initial condition:
- **if (the frequencies of restaurants in a neighbourhood is less) :**

  lesser competition + more benefits of opening a restaurant in that location.

Out[228]:

	Neighborhood	Latitude	Longitude
0	Ardwick	53.467675	-2.216010
1	Ardwick Green	53.467675	-2.216010
2	Bradford	53.794423	-1.751919
3	Burnage	53.435605	-2.205955
4	Burnage	53.435605	-2.205955
5	Castlefield	53.475822	-2.255700
6	Chorlton Park	53.434827	-2.269240
7	Chorlton-on-Medlock	53.465704	-2.233098
8	Circle Square Manchester	53.472337	-2.236694
9	Great Heaton	53.410148	-2.166866
10	Highfield Country Park	53.439075	-2.178117
11	Hulme	53.466031	-2.248166
12	Ladybarn	53.432233	-2.212339
13	Merseybank	53.414180	-2.995938
14	New Islington	53.482120	-2.221699
15	Spinningfields	53.480015	-2.251799
16	Whalley Range	53.449363	-2.257469
17	Withington	53.433582	-2.229308
18	Aigburth	53.369504	-2.931818
19	Anfield	53.430836	-2.960910
20	Clubmoor	53.429620	-2.934187
21	Kirkdale	53.432550	-2.981540
22	Mossley Hill	53.376114	-2.920953
23	Tuebrook	53.424701	-2.940850

CONCLUSION

- Got the shortlisted places in a single data frame.
- Providing the firm with the beneficial location in their desired cities and now its up to to them to choose from.

THANK YOU