Capstone Project - The Battle of the Neighborhoods (Week 2)

A Greek is going to Germany - and he wants to eat great greek food

Applied Data Science Capstone by IBM/Coursera

Table of contents

Introduction: Business Problem

Germany has been known as to attract many people to come and work and live there. It is the locomotive of the European economy, and a destination for many to come and live there. One of them is Hercules, a hero of our story. A Greek workaholic, known for his keen knowledge of the computer engineering... but also for his fine taste in food. Greek in particular. Gyros, Souvlaki, and Moussaka, just to name a few.

As Hercules is about to go on his journey, he is mostly concerned where he can find the highest number of Greek restaurants. He has chosen the cities of his particular interest in Germany, based on the already present IT industry:

Our task, which Hercules asked us to do, is to determine where, in the above 6 cities, is the highest number of Greek restaurants, so that he can go look for a job there. In addition to this, as Hercules is not too keen on driving, he wants to know in which city of the two, he has a (potential) shorter distances to drive to each of these restaurants, what could be his second option when making a decision where to move.

Data

We will use the FourSquare API to collect data about locations of the Greek restaurants stores in the selected cities which are: Munich, Stuttgart, Frankfurt, Berlin, Cologne, and Dusseldorf. These are the cities chosen by Hercules, as having the best IT opportunities.

We need the following information: GEO locations of the cities, can be obtained from the following link: https://latitudelongitude.org/de/ (https://latitudelongitude.org/de/) In the setting where we will be looking for a number of the cities, we could write the data-scrapping algorithm; however, as we are dealing with only 6 cities, collecting the data by hand will be much quicker. It will be stored in the CSV data file, for a later use if needed.

The data for the cities above, will be used to be obtain the number of the Greek restaurants via the **Foursquare API** utilized by the Request library in Python.

In [1]:

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

```
Libraries imported.
```

German cities, which Hercules selected

In [2]:
```python
data_de_cities = pd.read_csv("GermanCities.csv")
```

Out[2]:

	City	Lon	Lat
0	Berlin	52.52437	13.41053
1	Cologne	50.93333	6.95000
2	Dusseldorf	51.22172	6.77616
3	Frankfurt am Main	50.11552	8.68417
4	Munich	48.13743	11.57549
5	Stuttgart	48.78232	9.17702

In [20]:
```python
# Foursquare credentials
CLIENT_ID = 'hidden' # your Foursquare ID
CLIENT_SECRET = 'hidden' # your Foursquare Secret
VERSION = '20180605' # Foursquare API version

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

```
Your credentails:
CLIENT_ID: hidden
CLIENT_SECRET:hidden
```

In [4]:
```python
# collect greek restaurants
LIMIT = 500
cities = ['Berlin, Germany', 'Cologne, Germany', 'Dusseldorf, Germany', 'Frankfurt am Main, Germany', 'Munich,
results = {}
for city in cities:
    url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&near={}&limit={}&c
        CLIENT_ID,
        CLIENT_SECRET,
        VERSION,
        city,
        LIMIT,
        "4bf58dd8d48988d10e941735") # Greek restaurant ID
    results[city] = requests.get(url).json()
```

In [5]:
```python
df_venues={}
for city in cities:
    venues = pd.json_normalize(results[city]['response']['groups'][0]['items'])
    df_venues[city] = venues[['venue.name', 'venue.location.address', 'venue.location.lat', 'venue.location.ln
```

The Foursquare API will collect the data for us, by filtering on the Greek Restaurant ID (see above).

Let's first check out the total numbers by each city...

In [6]:
```python
maps = {}
for city in cities:
    city_lat = np.mean([results[city]['response']['geocode']['geometry']['bounds']['ne']['lat'],
                        results[city]['response']['geocode']['geometry']['bounds']['sw']['lat']])
    city_lng = np.mean([results[city]['response']['geocode']['geometry']['bounds']['ne']['lng'],
                        results[city]['response']['geocode']['geometry']['bounds']['sw']['lng']])
    maps[city] = folium.Map(location=[city_lat, city_lng], zoom_start=11)

    # add markers to map
    for lat, lng, label in zip(df_venues[city]['Lat'], df_venues[city]['Lng'], df_venues[city]['Name']):
        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(maps[city])
    print(f"Total number of Greek places in {city} = ", results[city]['response']['totalResults'])
```

```
Total number of Greek places in Berlin, Germany =  119
Total number of Greek places in Cologne, Germany =  53
Total number of Greek places in Dusseldorf, Germany =  53
Total number of Greek places in Frankfurt am Main, Germany =  33
Total number of Greek places in Munich, Germany =  116
Total number of Greek places in Stuttgart, Germany =  28
```

Methodology

In this project we want to see in which of the cities Hercules selected, is the highest number of the Greek restaurants, and what is the density of those restaurants.

In first step we have collected the required **data**, that is the **Greek restaurants** (according to Foursquare categorization, filtered by ID 4bf58dd8d48988d10e941735).

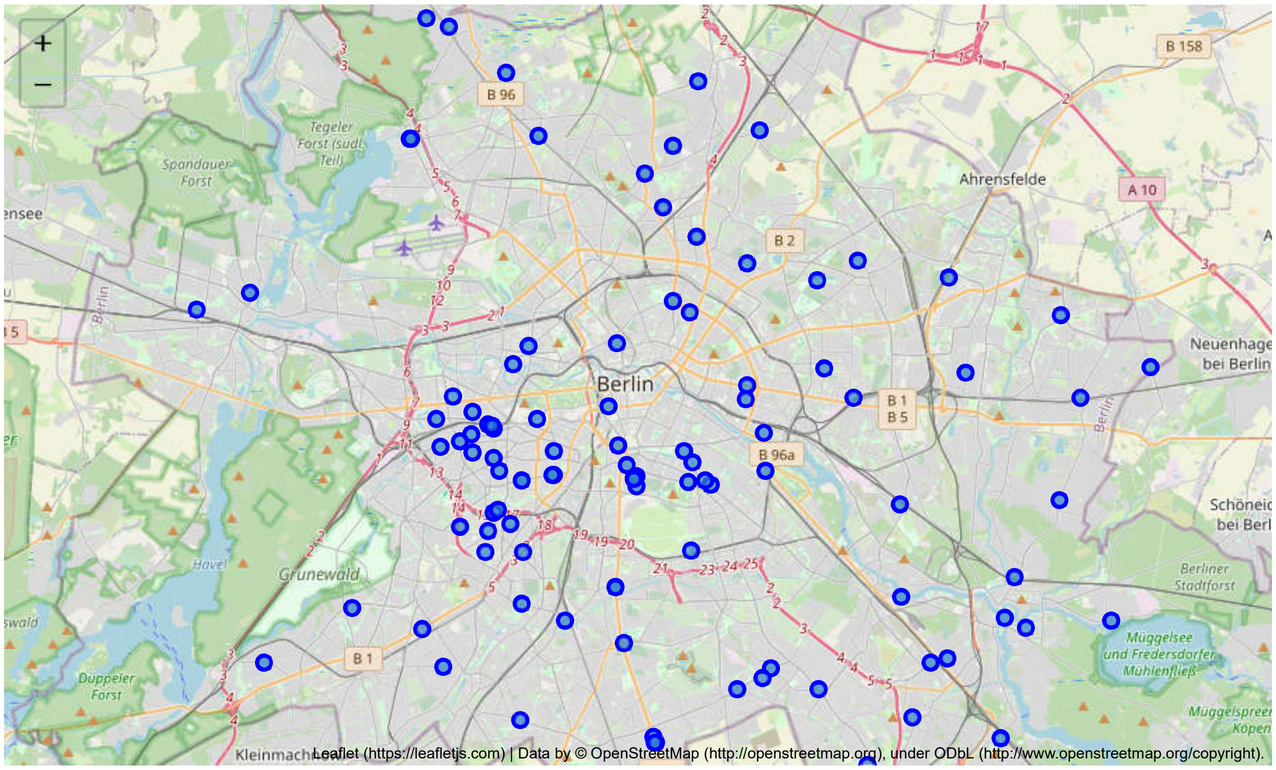
Second step in our analysis will be calculation and exploration of '**Greek restaurant density**' across different selected cities - we will calculate a center coordinate of the venues to get the mean longitude and latitude values. In the final step, we will calculate the mean of the Euclidean distance from each venue to the mean coordinates, which will be our indicator of the destiny.

Analysis

Let's have a look at the restaurants on the maps of each city, to get a visual look and feel of the Greek restaurants. Here, it can be noted, that it was the Greeks, who have opened up the Germans for different foods - and have brought a diveristy in the once dull German cousine.

In [7]:
```
[illegible]
```

Out[7]:

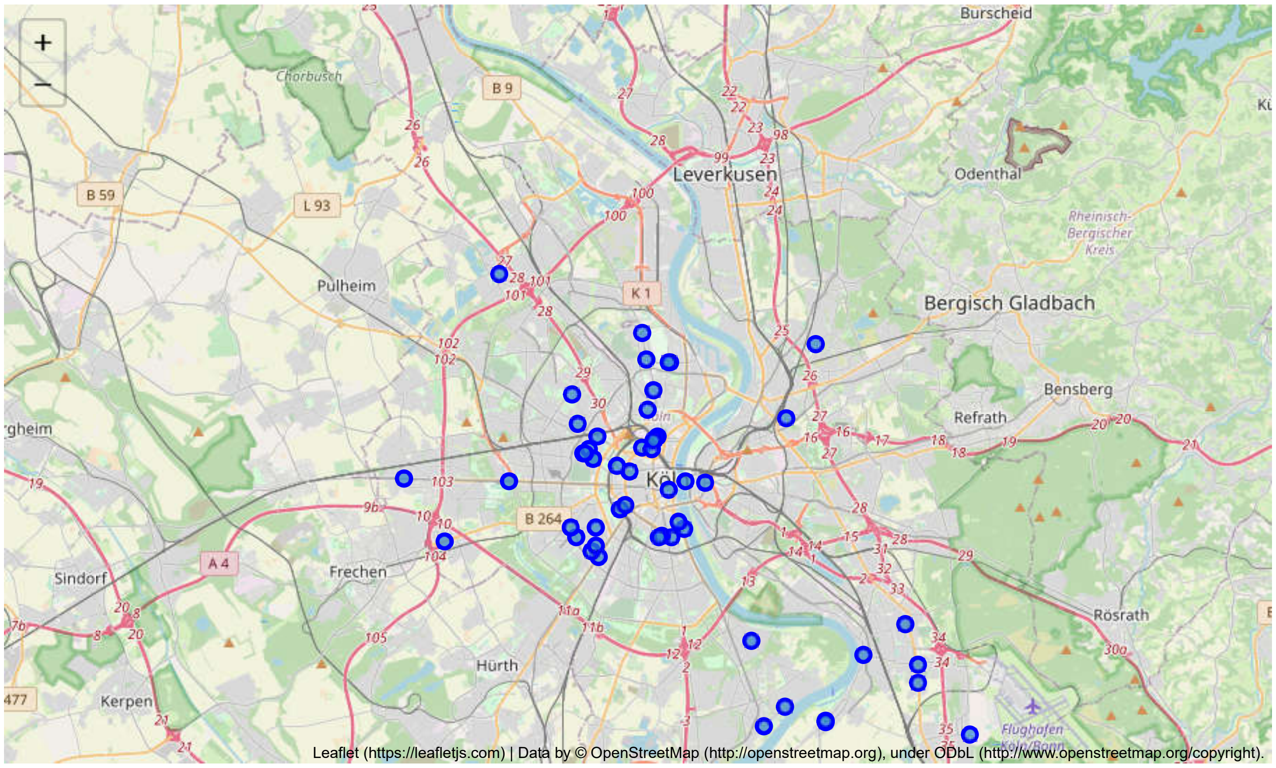


Berlin
Ahrensfelde
Tegeler Forst (südl. Teil)
Spandauer Forst
Grunewald
Düppeler Forst
Neuenhagen bei Berlin
Schöneiche bei Berlin
Berliner Stadtforst
Müggelsee und Fredersdorfer Mühlenfließ
B 96
B 158
A 10
B 2
B 1 B 5
B 96a
B 1
Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

In [8]:

```
[illegible]
```

Out[8]:

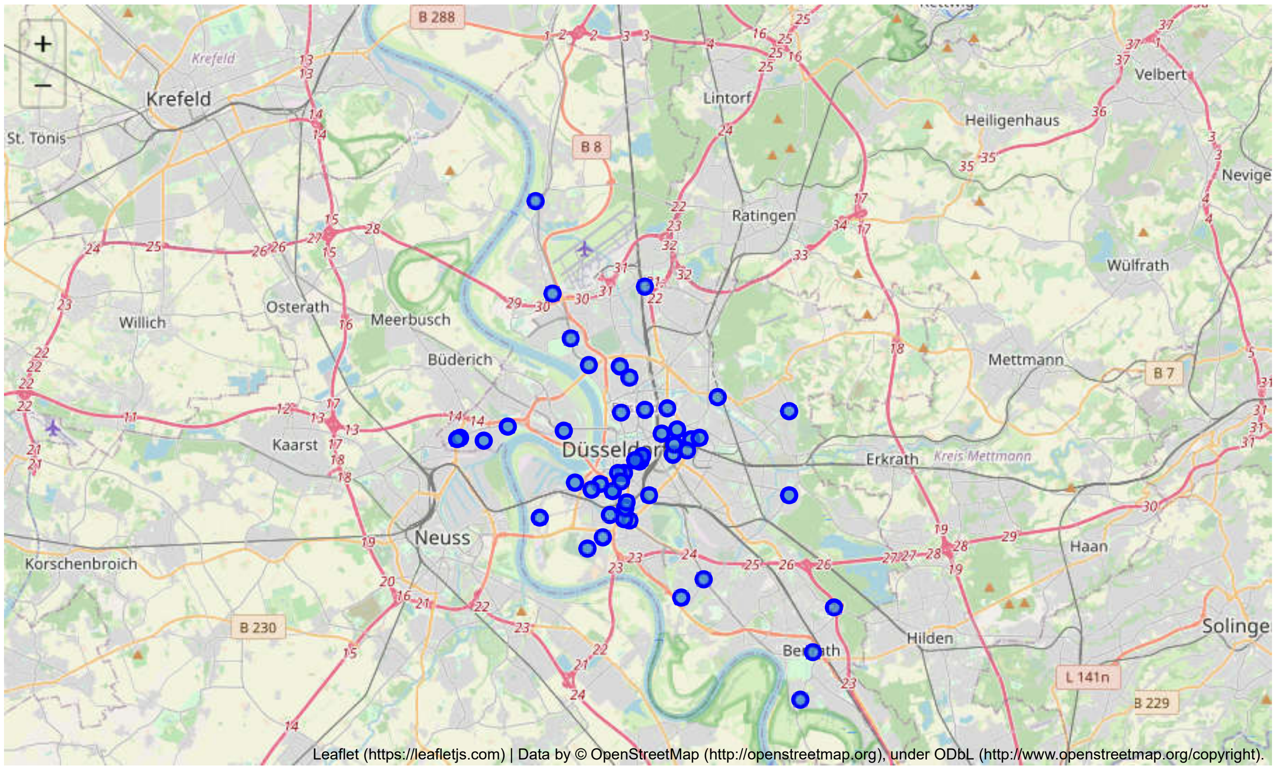


Burscheid
Leverkusen
Odenthal
Chorbusch
Pulheim
Bergisch Gladbach
Bensberg
Refrath
Köln
Frechen
Sindorf
Hürth
Kerpen
Rösrath
Flughafen Köln/Bonn
Rheinisch-Bergischer Kreis
Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

In [9]:

```
[illegible]
```

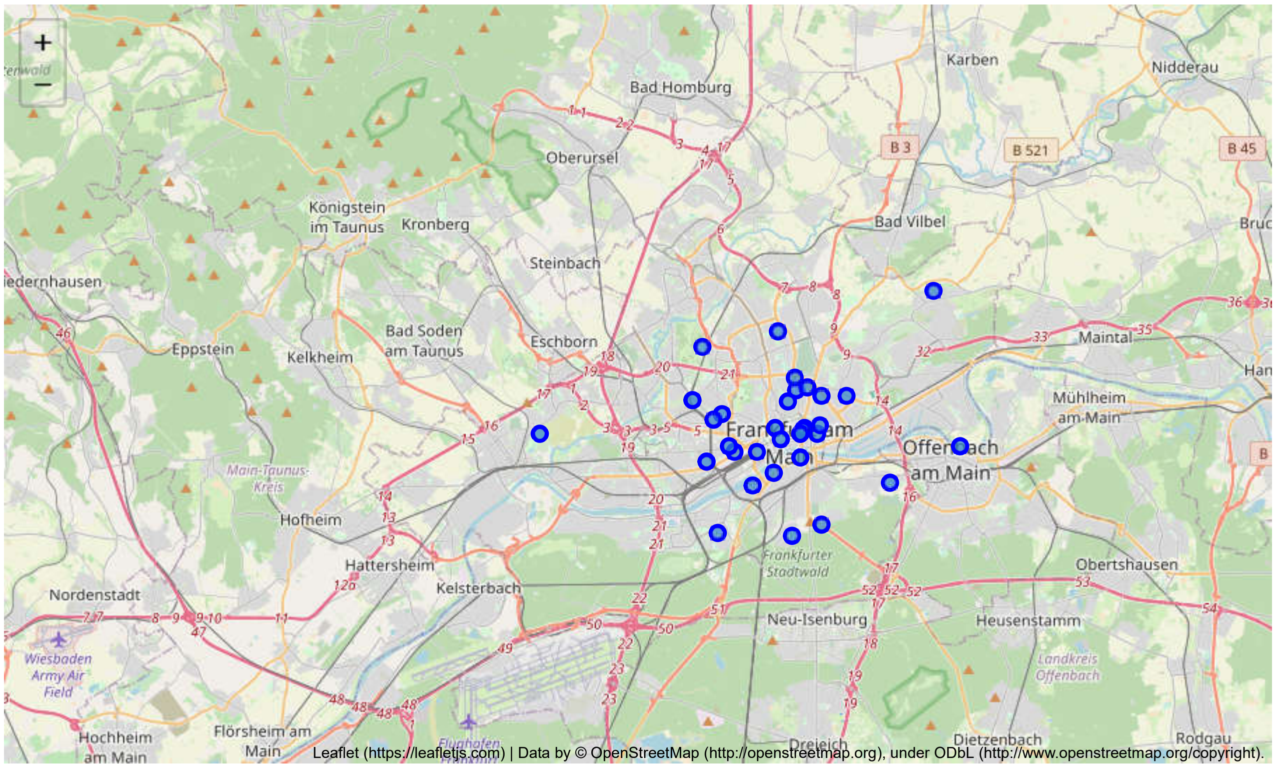
Out[9]:



In [10]:

```
[illegible]
```

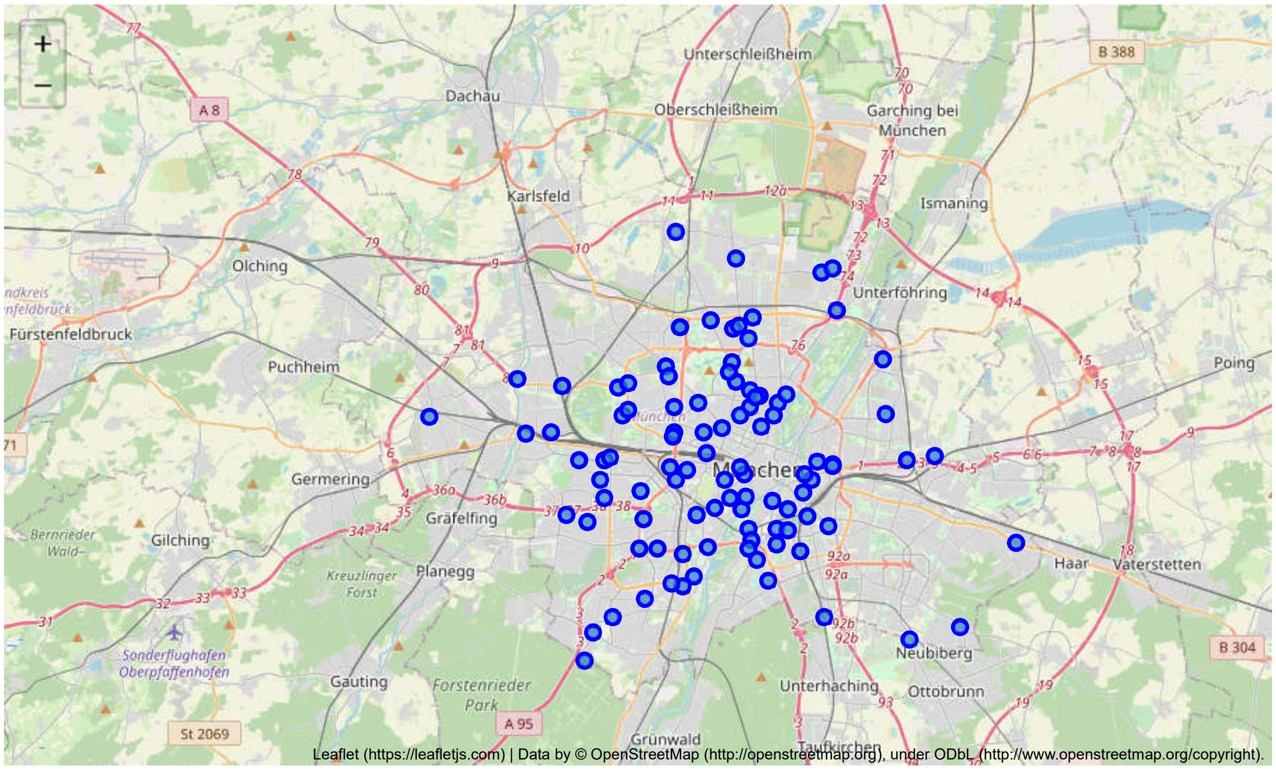
Out[10]:



In [11]:

```
[illegible]
```

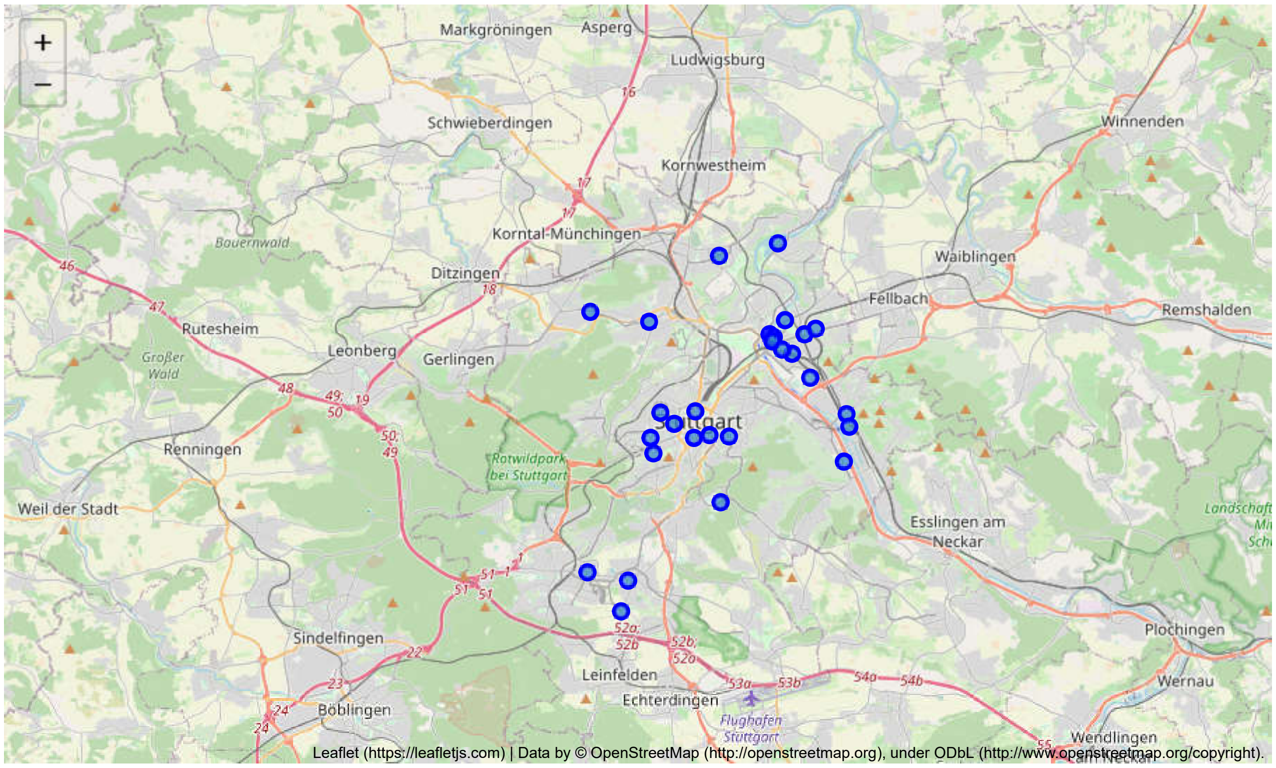
Out[11]:



In [12]:

```
[illegible]
```

Out[12]:



Initial analysis

We can see that Berlin, followed by Munich, are the most dense cities with Greek restaurants. The ones who are familiar with the German geography, might note that Düsseldorf and Cologne could also be of interest, each with 53 Greek restaurants, as the two cities are close to each other. However, this would be outside of the scope of this project, and is left to the reader to further explore this idea.

As a next step, let's have a concrete measure of this density.

For this, we will use some basic statistics.

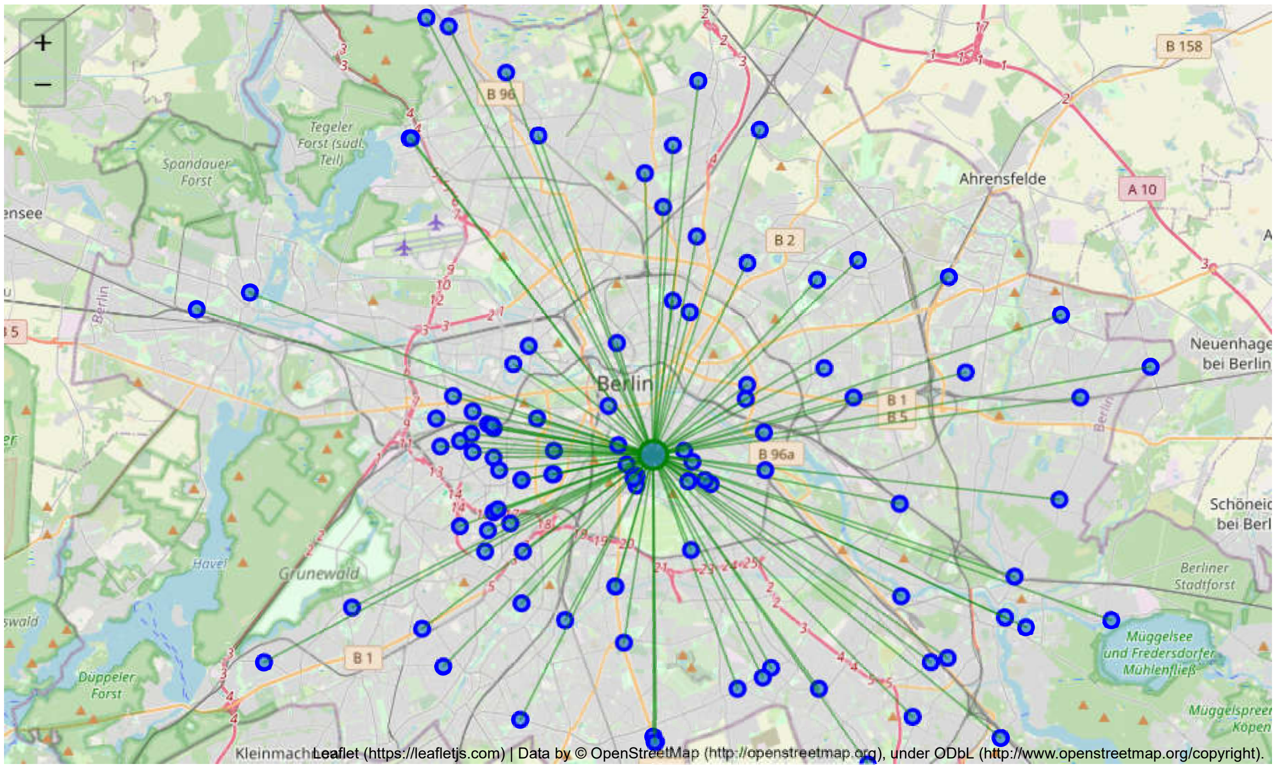
In [13]:
```python
mapsDist = {}
for city in cities:
    city_lat = np.mean([results[city]['response']['geocode']['geometry']['bounds']['ne']['lat'],
                        results[city]['response']['geocode']['geometry']['bounds']['sw']['lat']])
    city_lng = np.mean([results[city]['response']['geocode']['geometry']['bounds']['ne']['lng'],
                        results[city]['response']['geocode']['geometry']['bounds']['sw']['lng']])
    mapsDist[city] = folium.Map(location=[city_lat, city_lng], zoom_start=11)
    venues_mean_coor = [df_venues[city]['Lat'].mean(), df_venues[city]['Lng'].mean()]
    # add markers to map
    for lat, lng, label in zip(df_venues[city]['Lat'], df_venues[city]['Lng'], df_venues[city]['Name']):
        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(mapsDist[city])
        folium.PolyLine([venues_mean_coor, [lat, lng]], color="green", weight=1.5, opacity=0.5).add_to(mapsDis

    label = folium.Popup("Mean Co-ordinate", parse_html=True)
    folium.CircleMarker(
        venues_mean_coor,
        radius=10,
        popup=label,
        color='green',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(mapsDist[city])

    print(city)
    print("Mean Distance from Mean coordinates")
    print(np.mean(np.apply_along_axis(lambda x: np.linalg.norm(x - venues_mean_coor),1,df_venues[city][['Lat',
    print('')
```

```
Berlin, Germany
Mean Distance from Mean coordinates
0.09620332084639395

Cologne, Germany
Mean Distance from Mean coordinates
0.052812766251462105

Dusseldorf, Germany
Mean Distance from Mean coordinates
0.03584372950897846

Frankfurt am Main, Germany
Mean Distance from Mean coordinates
0.030341493393213657

Munich, Germany
Mean Distance from Mean coordinates
0.048757247897015174

Stuttgart, Germany
Mean Distance from Mean coordinates
0.04338877481443552
```

In [14]:

```
Dist[cities[0]]  #Berlin
```

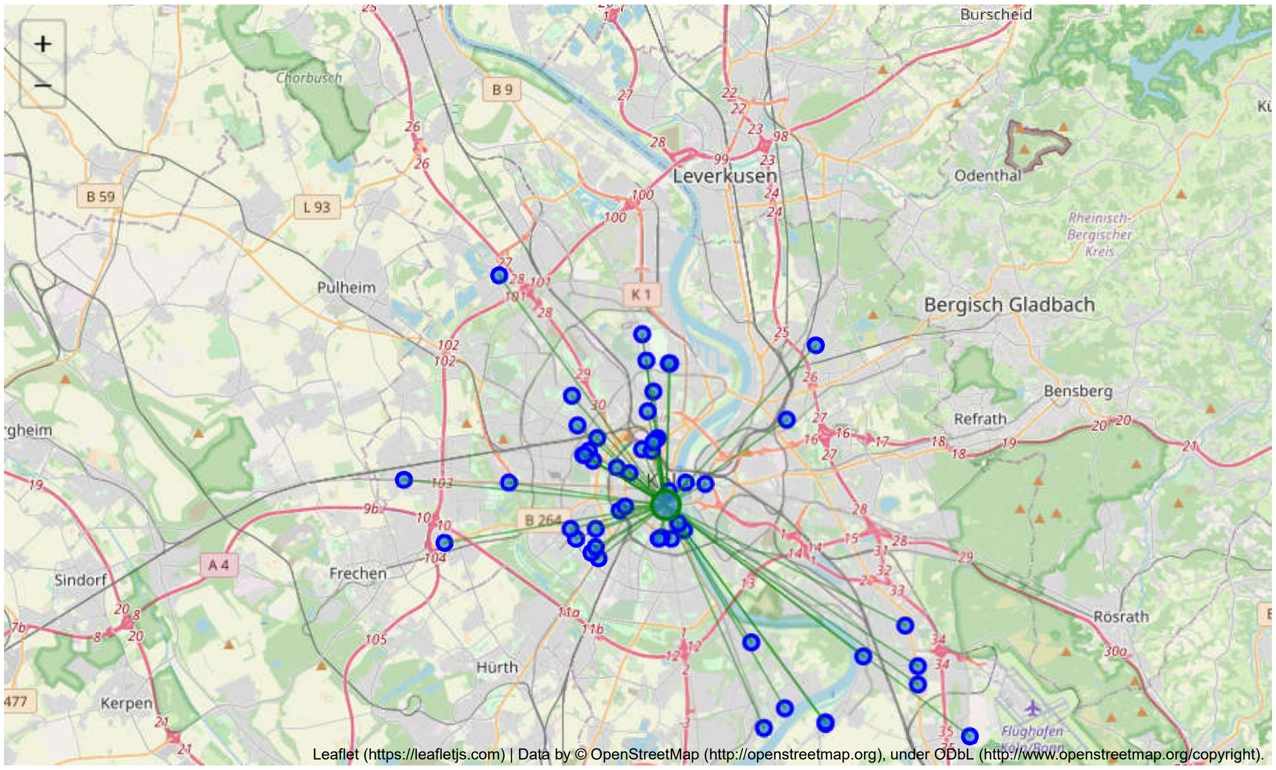
Out[14]:


Berlin
Spandauer Forst
Tegeler Forst (südl. Teil)
Ahrensfelde
Neuenhagen bei Berlin
Schöneiche bei Berlin
Grunewald
Düppeler Forst
Müggelsee und Fredersdorfer Mühlenfließ
Kleinmachnow
Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

In [15]:

```
Dist[cities[1]]  #Cologne
```

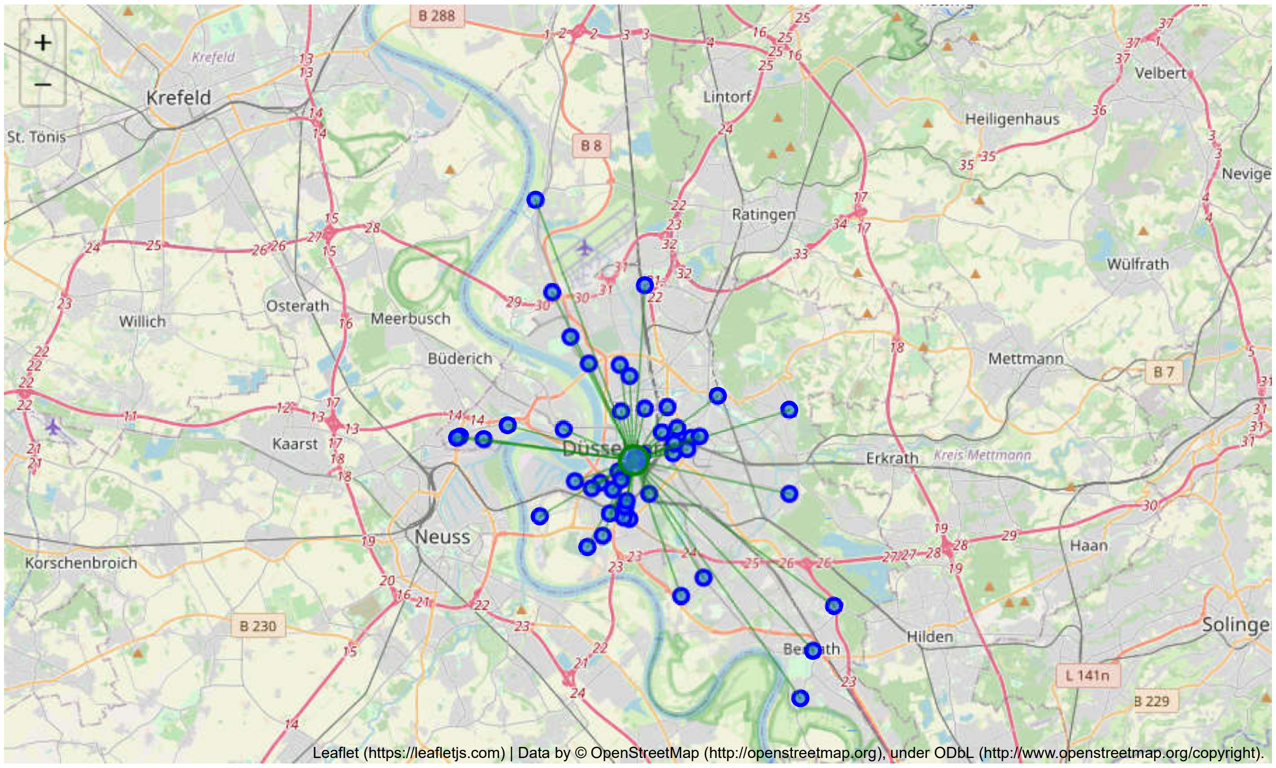
Out[15]:


Leverkusen
Burscheid
Odenthal
Bergisch Gladbach
Pulheim
Bensberg
Refrath
Frechen
Sindorf
Hürth
Kerpen
Rösrath
Flughafen Köln/Bonn
Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

In [16]:

```
[illegible]
```

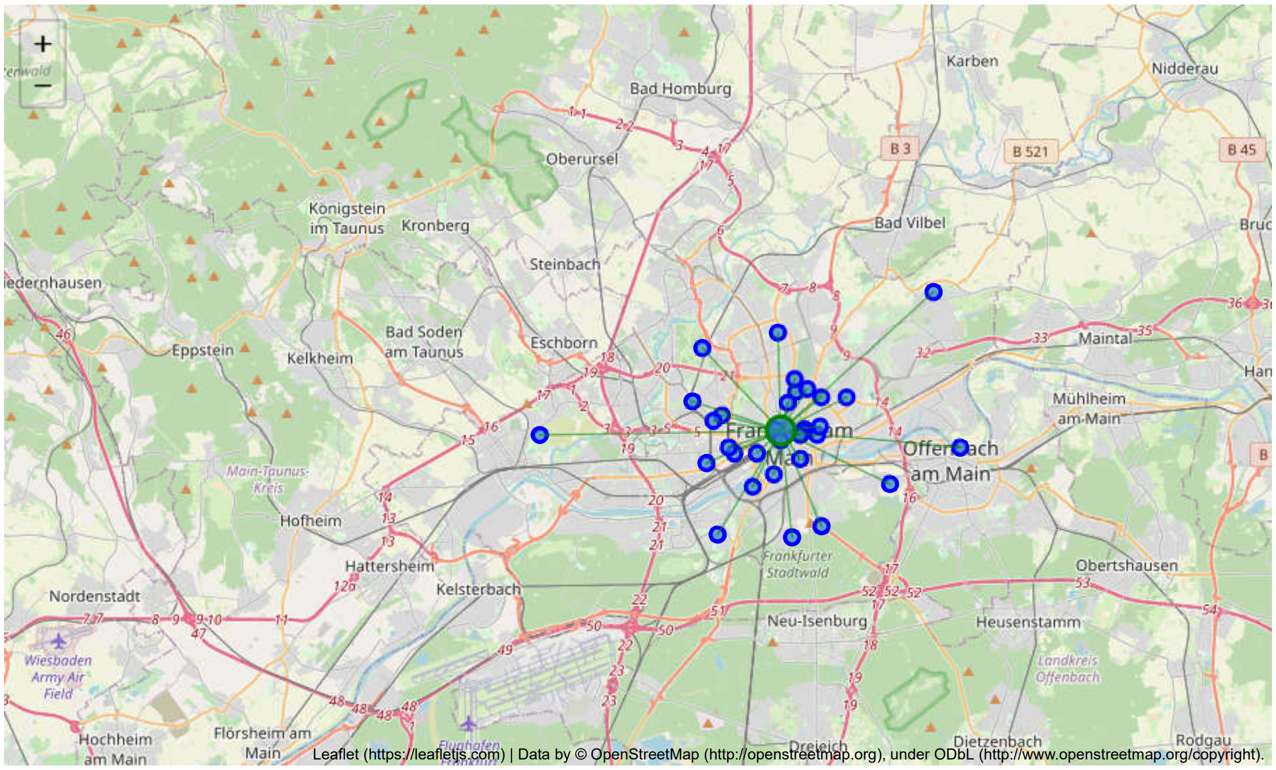
Out[16]:



In [17]:

```
[illegible]
```

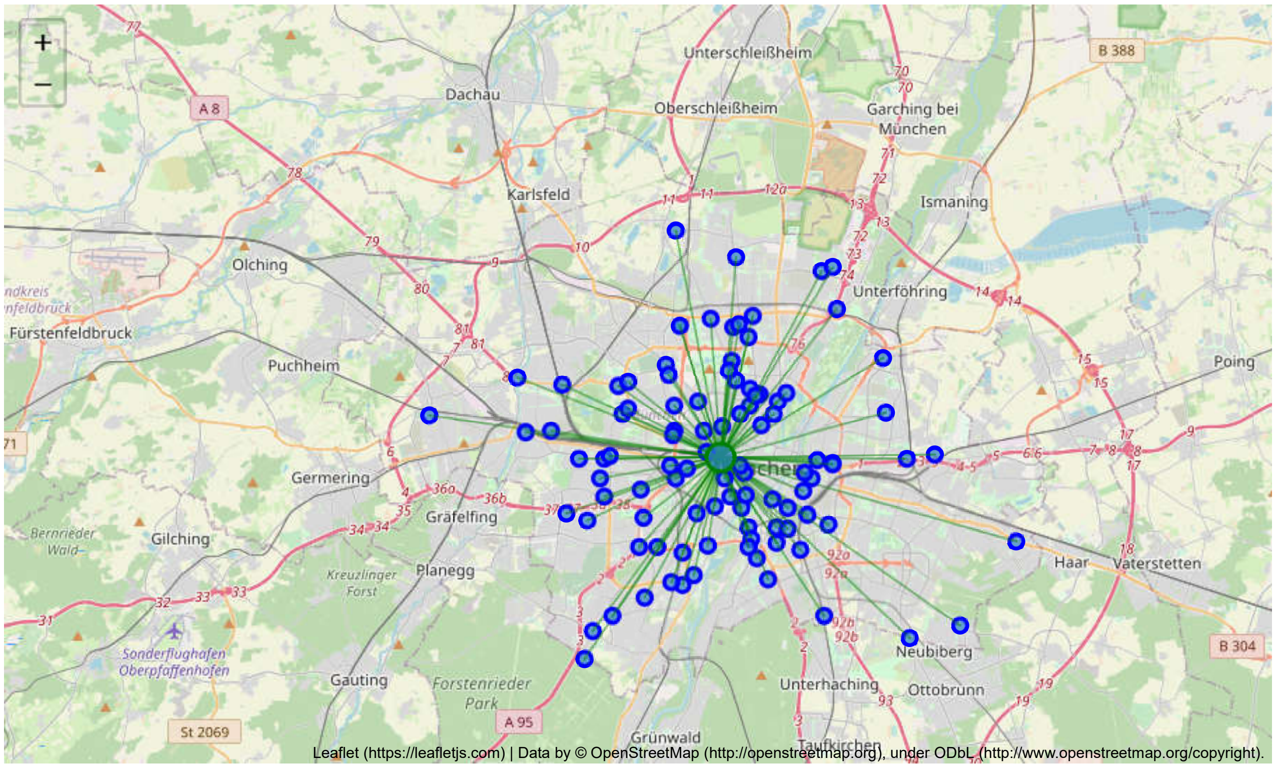
Out[17]:



In [18]:

```
Dist[cities[4]] #Munich
```

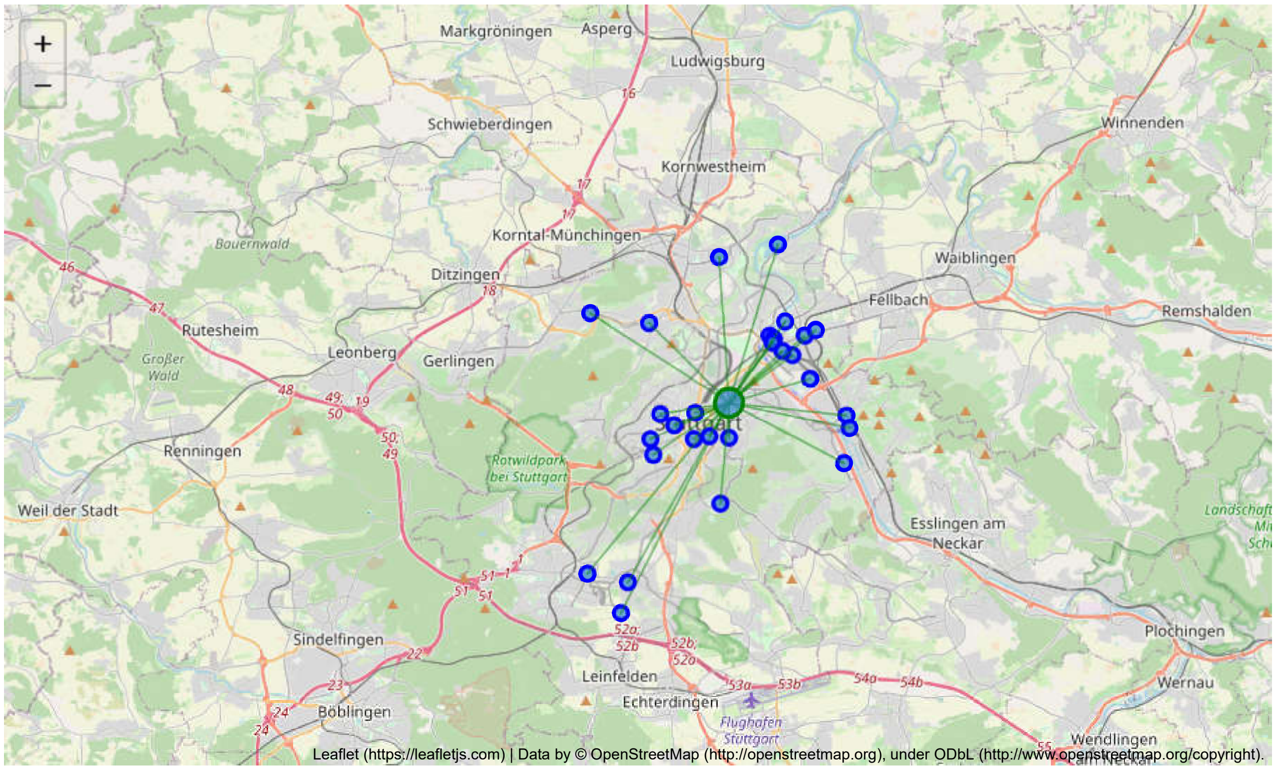
Out[18]:



In [19]:

```
Dist[cities[5]] #Stuttgart
```

Out[19]:



Results and Discussion

When looking at the data we have obtained, we can see that Berlin, with 146 Greek restaurants, is the number one candidate for Hercules to move to. The second city is Munich, with 116, what is 30 restaurants less than Berlin. Other cities haf far less Greek restaurants, and do not offer much variety Hercules is looking for.

In addition to these 2 cities, if Hercules does receive a good job offer from a company located in Cologne or Düsseldorf, he could consider these offers as well, as there is only 30 km (20 miles) between these 2 cities, which could potentially offer an additional advantage. However, this analysis is outside the scope of this exercise, and would require an additional analysis of many cities located near both Cologne and Düsseldorf, which could offer an additional number of the Greek restaurants in their vicinity.

The second point of Hercules' interest was the distance of the restaurants, taken from the geographical city center. With the average mean distance of 0.03 the winner is Frankfurt am Main, but with only 33 restaurants it is not that interesting. Similar can be said about Düsseldorf with 53 restaurants each, but with an average mean of 0.035. What could speak for Düsseldorf is the vicinity not only Cologne, a city which we have analyzed, but also a number of other cities nearyby, in this densly populated area of Germany. Leverkusen, Duisburg and Wuppertal, are a few cities one can almost immediatelly spot on the map.

In the middle of the pack we can see Munich, with an average mean of 0.048, making it very attractive for the final consideration, taking into the account a total of well over 100 Greek restaurants one can find here.

Berlin, the city with the highest number of restaurants, is having the worst mean average of 0.085. By looking at the map, this does not come as a surprise, as the restaurants are spread all over Berlin.

Conclusion

The purpose of this exercise was to examine the Greek restaurants in the 6 cities, Hercules is considering to move to. A second criteria was the distance from the city centers to each of those restaurants within the city.

Even though Berlin does offer a great deal of Greek restaurants, with an average of 0.085, which is much larger than Munich's 0.049 and its own 116 restaurans, we would recommend the Berlin as Hercules' second choice.

We are suggesting that Hercules explores **Munich** first, and the job offers he has there. 116 restaurants give Hercules a much of the variety he is looking for, and they are not that far from the city's center. **Berlin,** should be his second choice.

And with that, we conclude our exercise, and wish best of luck to our hero, Hercules, in his adventure to Germany!

```
In [ ]:
```