Importing needed packages

In [1]:
```
!pip install matplotlib-scalebar
!pip install fiona
!pip install folium
```

```
Requirement already satisfied: matplotlib-scalebar in /Users/kruthikaramesh/opt/anaconda
3/lib/python3.9/site-packages (0.8.1)
Requirement already satisfied: matplotlib in /Users/kruthikaramesh/opt/anaconda3/lib/pyt
hon3.9/site-packages (from matplotlib-scalebar) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /Users/kruthikaramesh/opt/anaconda3/l
ib/python3.9/site-packages (from matplotlib->matplotlib-scalebar) (1.0.5)
Requirement already satisfied: cycler>=0.10 in /Users/kruthikaramesh/opt/anaconda3/lib/p
ython3.9/site-packages (from matplotlib->matplotlib-scalebar) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /Users/kruthikaramesh/opt/anaconda3/
lib/python3.9/site-packages (from matplotlib->matplotlib-scalebar) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /Users/kruthikaramesh/opt/anaconda3/
lib/python3.9/site-packages (from matplotlib->matplotlib-scalebar) (1.4.4)
Requirement already satisfied: numpy>=1.20 in /Users/kruthikaramesh/opt/anaconda3/lib/py
thon3.9/site-packages (from matplotlib->matplotlib-scalebar) (1.25.0)
Requirement already satisfied: packaging>=20.0 in /Users/kruthikaramesh/opt/anaconda3/li
b/python3.9/site-packages (from matplotlib->matplotlib-scalebar) (23.0)
Requirement already satisfied: pillow>=6.2.0 in /Users/kruthikaramesh/opt/anaconda3/lib/
python3.9/site-packages (from matplotlib->matplotlib-scalebar) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /Users/kruthikaramesh/opt/anaconda3/l
ib/python3.9/site-packages (from matplotlib->matplotlib-scalebar) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in /Users/kruthikaramesh/opt/anacond
a3/lib/python3.9/site-packages (from matplotlib->matplotlib-scalebar) (2.8.2)
Requirement already satisfied: importlib-resources>=3.2.0 in /Users/kruthikaramesh/opt/a
naconda3/lib/python3.9/site-packages (from matplotlib->matplotlib-scalebar) (5.2.0)
Requirement already satisfied: zipp>=3.1.0 in /Users/kruthikaramesh/opt/anaconda3/lib/py
thon3.9/site-packages (from importlib-resources>=3.2.0->matplotlib->matplotlib-scalebar)
(3.11.0)
Requirement already satisfied: six>=1.5 in /Users/kruthikaramesh/opt/anaconda3/lib/pytho
n3.9/site-packages (from python-dateutil>=2.7->matplotlib->matplotlib-scalebar) (1.16.0)
Requirement already satisfied: fiona in /Users/kruthikaramesh/opt/anaconda3/lib/python3.
9/site-packages (1.9.4.post1)
Requirement already satisfied: attrs>=19.2.0 in /Users/kruthikaramesh/opt/anaconda3/lib/
python3.9/site-packages (from fiona) (22.1.0)
Requirement already satisfied: certifi in /Users/kruthikaramesh/opt/anaconda3/lib/python
3.9/site-packages (from fiona) (2023.7.22)
Requirement already satisfied: click~=8.0 in /Users/kruthikaramesh/opt/anaconda3/lib/pyt
hon3.9/site-packages (from fiona) (8.0.4)
Requirement already satisfied: click-plugins>=1.0 in /Users/kruthikaramesh/opt/anaconda
3/lib/python3.9/site-packages (from fiona) (1.1.1)
Requirement already satisfied: cligj>=0.5 in /Users/kruthikaramesh/opt/anaconda3/lib/pyt
hon3.9/site-packages (from fiona) (0.7.2)
Requirement already satisfied: six in /Users/kruthikaramesh/opt/anaconda3/lib/python3.9/
site-packages (from fiona) (1.16.0)
Requirement already satisfied: importlib-metadata in /Users/kruthikaramesh/opt/anaconda
3/lib/python3.9/site-packages (from fiona) (6.8.0)
Requirement already satisfied: zipp>=0.5 in /Users/kruthikaramesh/opt/anaconda3/lib/pyth
on3.9/site-packages (from importlib-metadata->fiona) (3.11.0)
Requirement already satisfied: folium in /Users/kruthikaramesh/opt/anaconda3/lib/python
3.9/site-packages (0.14.0)
Requirement already satisfied: branca>=0.6.0 in /Users/kruthikaramesh/opt/anaconda3/lib/
python3.9/site-packages (from folium) (0.6.0)
Requirement already satisfied: jinja2>=2.9 in /Users/kruthikaramesh/opt/anaconda3/lib/py
thon3.9/site-packages (from folium) (3.1.2)
Requirement already satisfied: numpy in /Users/kruthikaramesh/opt/anaconda3/lib/python3.
9/site-packages (from folium) (1.25.0)
```

```
Requirement already satisfied: requests in /Users/kruthikaramesh/opt/anaconda3/lib/python3.9/site-packages (from folium) (2.30.0)
Requirement already satisfied: MarkupSafe>=2.0 in /Users/kruthikaramesh/opt/anaconda3/lib/python3.9/site-packages (from jinja2>=2.9->folium) (2.1.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /Users/kruthikaramesh/opt/anaconda3/lib/python3.9/site-packages (from requests->folium) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /Users/kruthikaramesh/opt/anaconda3/lib/python3.9/site-packages (from requests->folium) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /Users/kruthikaramesh/opt/anaconda3/lib/python3.9/site-packages (from requests->folium) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in /Users/kruthikaramesh/opt/anaconda3/lib/python3.9/site-packages (from requests->folium) (2023.7.22)
```

In [2]:

```
import os
import fiona
import folium
import requests
import matplotlib
#import osmnx as ox
import pandas as pd
import geopandas as gpd
import plotly.express as px
from pyproj import Transformer
from folium.plugins import Draw
import matplotlib.pyplot as plt
from scipy.spatial import ConvexHull
from folium.plugins import BeautifyIcon
from sklearn.cluster import MiniBatchKMeans
from matplotlib_scalebar.scalebar import ScaleBar
from matplotlib.font_manager import FontProperties
from shapely.geometry import Point, LineString, Polygon
```

```
/Users/kruthikaramesh/opt/anaconda3/envs/newenv/lib/python3.11/site-packages/geopandas/_compat.py:124: UserWarning: The Shapely GEOS version (3.11.2-CAPI-1.17.2) is incompatible with the GEOS version PyGEOS was compiled with (3.10.4-CAPI-1.16.2). Conversions between both will be slow.
  warnings.warn(
/var/folders/j_/yy54qztd2qv_yhxvrrgcbd1c0000gn/T/ipykernel_87893/1713895721.py:8: DeprecationWarning: Shapely 2.0 is installed, but because PyGEOS is also installed, GeoPandas still uses PyGEOS by default. However, starting with version 0.14, the default will switch to Shapely. To force to use Shapely 2.0 now, you can either uninstall PyGEOS or set the environment variable USE_PYGEOS=0. You can do this before starting the Python process, or in your code before importing geopandas:

import os
os.environ['USE_PYGEOS'] = '0'
import geopandas

In the next release, GeoPandas will switch to using Shapely by default, even if PyGEOS is installed. If you only have PyGEOS installed to get speed-ups, this switch should be smooth. However, if you are using PyGEOS directly (calling PyGEOS functions on geometries from GeoPandas), this will then stop working and you are encouraged to migrate from PyGEOS to Shapely 2.0 (https://shapely.readthedocs.io/en/latest/migration_pygeos.html).
  import geopandas as gpd
```

Reading the restaurant data

In [3]:

```
nonna_merged_data = pd.read_csv('London Population /Cleaned_restaurants.csv') #path chan
```

In [4]:

```
import numpy as np
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
```

```
from sklearn.linear_model import LinearRegression
# Select the 'Rating Value' column
rating_col = nonna_merged_data['Rating Value']

# Create a mask for rows with missing values
missing_mask = rating_col.eq('AwaitingPublication') | rating_col.eq('AwaitingInspection'

# Create a copy of the 'Rating Value' column for imputation
rating_imputed = rating_col.copy()

# Map the string values to NaN
rating_imputed[missing_mask] = np.nan

# Convert the 'Rating Value' column to float type
rating_imputed = rating_imputed.astype(float)

# Creating a regression imputer
imputer = IterativeImputer(estimator=LinearRegression(), random_state=0)

# Fit and transform the 'Rating Value' column with missing values
rating_imputed = imputer.fit_transform(rating_imputed.values.reshape(-1, 1))

# Flatten the imputed values
rating_imputed = rating_imputed.flatten()

# Assign the imputed values back to the original DataFrame
nonna_merged_data.loc[missing_mask, 'Rating Value'] = rating_imputed[:missing_mask.sum()

# Save the modified DataFrame to a CSV file
#nonna_merged_data.to_csv('new_rating_imputed_data.csv', index=False)



# Select the 'Rating Value' column
hygiene_col = nonna_merged_data['Hygiene']

# Create a mask for rows with missing values
missing_hygiene =  hygiene_col.isnull()

# Create a copy of the 'Rating Value' column for imputation
hygiene_imputed = hygiene_col.copy()

# Map the string values to NaN
hygiene_imputed[missing_hygiene] = np.nan

# Convert the 'Rating Value' column to float type
hygiene_imputed = hygiene_imputed.astype(float)

# Creating a regression imputer
imputer = IterativeImputer(estimator=LinearRegression(), random_state=0)

# Fit and transform the 'Rating Value' column with missing values
hygiene_imputed = imputer.fit_transform(hygiene_imputed.values.reshape(-1, 1))

# Flatten the imputed values
hygiene_imputed = hygiene_imputed.flatten()

# Assign the imputed values back to the original DataFrame
nonna_merged_data.loc[missing_hygiene, 'Hygiene'] = hygiene_imputed[:missing_hygiene.sum
```

In [5]:
```
nonna_merged_data.columns
```

Out[5]:
```
Index(['Unnamed: 0', 'Address ', 'Business Name ', 'Business Type',
       'BusinessTypeID', 'FHRSID', 'Latitude', 'Longitude',
       'Local Authority BusinessID', 'Local Authority Code',
```

```
       'Local Authority Email Address', 'Local Authority Name',
       'Local Authority WebSite', 'New Rating Pending', 'Post Code',
       'Rating Key', 'Rating Value', 'Scheme Type', 'Confidence In Management',
       'Hygiene', ' Structural', '/Header/ItemCount/#agg'],
      dtype='object')
```

In [6]:
```
columns_drop = [' Structural','Scheme Type','Confidence In Management','Local Authority
nonna_merged_data.drop(columns = columns_drop, inplace = True)
```

In [7]:
```
nonna_merged_data.dropna(subset = ['Latitude','Longitude'])
```

Out[7]:

	Unnamed: 0	Address	Business Name	Business Type	BusinessTypeID	FHRSID	Latitude	L
1	1	Rio Cinema Dalston 103-107 Kingsland High Stre...	"Rio Cinema"	Restaurant/Cafe/Canteen	1.0	468824.0	51.549539	-
2	2	Railway Arch 214 Ponsford Street Hackney London	% Arabica	Manufacturers/packers	7839.0	1220366.0	51.547348	-
3	3	Basement To Ground Floor 33 Broadway Market Ha...	%Arabica	Other catering premises	7841.0	1225476.0	51.536510	-
4	4	45-47 lower Clapton road Hackney London	&organic ltd	Restaurant/Cafe/Canteen	1.0	1527059.0	51.550962	-
6	6	485 kingsland road Hackney London	@las coffee house	Restaurant/Cafe/Canteen	1.0	1578908.0	51.544256	-
...	...	...	...	...	...	...	...	
78778	78778	2 Castle Street Kingston Upon Thames	Yoriya	Takeaway/sandwich shop	7844.0	1385803.0	51.411049	-
78779	78779	96 Burlington Road New Malden	You Me Korean Restaurant	Restaurant/Cafe/Canteen	1.0	1386083.0	51.398814	-
78781	78781	43 Market Place Kingston Upon Thames	Zizzi And Coco Di Mama	Restaurant/Cafe/Canteen	1.0	1386687.0	51.408839	-
78782	78782	Kings Oak Primary	Zone All Sports	School/college/university	7845.0	1594642.0	51.404842	-

		School Dickerage Lane New Ma...	And Care At Kings Oak Primary ...					
78784	78784	Malden Parochial C Of E Primary School The Man...	Zone All Sports And Care At Malden Parochial S...	School/college/university	7845.0	1594647.0	51.381095	-

64323 rows × 17 columns

In [119…
```
gdf = gpd.GeoDataFrame(nonna_merged_data, geometry= gpd.points_from_xy(nonna_merged_data
```

In [9]:
```
gdf.crs = "EPSG:4326" #Adding crs information to geodataframe
#gdf.plot(marker = '*', color = 'green') #Plotting the geodataframe
#plt.rcParams['figure.figsize'] = [10, 10]
```

Visualizing restaurant data

From the plot above it can be seen that not all the points are in London, and will be removed...

To do this, we need another dataset that has the London spatial data information, ie London shapefile...

In [11]:
```
london = gpd.read_file("London Population /London_Borough_Excluding_MHW.shp")
london.crs #crs here is projected wherea~s points geodataframe ('gdf') is geographic crs
```

Out[11]:
```
<Projected CRS: PROJCS["OSGB36 / British National Grid",GEOGCS["OS ...>
Name: OSGB36 / British National Grid
Axis Info [cartesian]:
- [east]: Easting (metre)
- [north]: Northing (metre)
Area of Use:
- undefined
Coordinate Operation:
- name: unnamed
- method: Transverse Mercator
Datum: Ordnance Survey of Great Britain 1936
- Ellipsoid: Airy 1830
- Prime Meridian: Greenwich
```

In [12]:
```
gdf.crs
```

Out[12]:
```
<Geographic 2D CRS: EPSG:4326>
Name: WGS 84
Axis Info [ellipsoidal]:
- Lat[north]: Geodetic latitude (degree)
- Lon[east]: Geodetic longitude (degree)
Area of Use:
- name: World.
- bounds: (-180.0, -90.0, 180.0, 90.0)
Datum: World Geodetic System 1984 ensemble
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```

In [120…
```
london['geometry'].to_crs(epsg=4326)#, allow_override=True)
```

```
london_new = london.to_crs(epsg=4326)#, allow_override=True)
```

Breakdown of spatial datasets we have so far...

1. gdf
2. london
3. london_new

In [14]:
```
london_new.crs #geographic coordinate system
```

Out[14]:
```
<Geographic 2D CRS: EPSG:4326>
Name: WGS 84
Axis Info [ellipsoidal]:
- Lat[north]: Geodetic latitude (degree)
- Lon[east]: Geodetic longitude (degree)
Area of Use:
- name: World.
- bounds: (-180.0, -90.0, 180.0, 90.0)
Datum: World Geodetic System 1984 ensemble
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```

In [15]:
```
# ax = london.plot()
# gdf.plot(ax=ax)

#ax = gdf.plot(alpha=0.1, color='green')
#london_new.plot(ax=ax, color = '#C6A619')
```

Plot is not very intuitive, london boundary is shown by a small yellow dot!

In [16]:
```
#london_new.plot(color='#C6A619')
```

From the london plot above, it is seen that the highest latitude value is about 51.7 and the lowest value about 51.2 the lowest longitude value is about -0.2 and the highest value about 0.4

In [17]:
```
to_del = []
long_del_east = gdf[gdf['Longitude'] > 0.4].index.to_list()
long_del_west = gdf[gdf['Longitude'] <-0.5].index.to_list()
lat_del_upper = gdf[gdf['Latitude'] >51.7].index.to_list()
lat_del_lower = gdf[gdf['Latitude'] <51.2].index.to_list()

to_del.extend(long_del_east)
to_del.extend(long_del_west)
to_del.extend(lat_del_upper)
to_del.extend(lat_del_lower)
print(len(to_del))
to_del = set(to_del)
print(len(to_del))
```

```
53
36
```

In [18]:
```
print(f'Before deletion of rows: {len(gdf)}')
gdf = gdf.drop(index=to_del)
print(f'After deletion of rows: {len(gdf)}')
```

```
Before deletion of rows: 78785
After deletion of rows: 78749
```

In [19]:
```
#gdf.plot() #looks much better, and within the london area range
```

In [20]:
```
# ax = london_new.plot(edgecolor='blue',color='#F1C40F')
```

```
# gdf.plot(ax=ax, marker='*')
# plt.rcParams['figure.figsize']= [8, 8]  #Some points are shown outside the boundary of
```

In [21]:
```
gdf.geometry
```

Out[21]:
```
0                    POINT EMPTY
1        POINT (-0.07557 51.54954)
2        POINT (-0.04853 51.54735)
3        POINT (-0.06205 51.53651)
4        POINT (-0.05220 51.55096)
                  ...
78780                POINT EMPTY
78781    POINT (-0.30618 51.40884)
78782    POINT (-0.27453 51.40484)
78783                POINT EMPTY
78784    POINT (-0.25778 51.38109)
Name: geometry, Length: 78749, dtype: geometry
```

In [22]:
```
gdf.dropna(subset = ['Business Name '])
```

Out[22]:

	Unnamed: 0	Address	Business Name	Business Type	BusinessTypeID	FHRSID	Latitude	Lo
0	0	432-434 Kingsland Road, London E8 4AA Hackney...	"MU"	Restaurant/Cafe/Canteen	1.0	1584416.0	NaN	
1	1	Rio Cinema Dalston 103-107 Kingsland High Stre...	"Rio Cinema"	Restaurant/Cafe/Canteen	1.0	468824.0	51.549539	-0
2	2	Railway Arch 214 Ponsford Street Hackney London	% Arabica	Manufacturers/packers	7839.0	1220366.0	51.547348	-0
3	3	Basement To Ground Floor 33 Broadway Market Ha...	%Arabica	Other catering premises	7841.0	1225476.0	51.536510	-0
4	4	45-47 lower Clapton road Hackney London	&organic ltd	Restaurant/Cafe/Canteen	1.0	1527059.0	51.550962	-0
...	...	...	...	...	...	...	...	
78780	78780		Zays Bakes	Other catering premises	7841.0	1498540.0	NaN	
78781	78781	43 Market Place Kingston	Zizzi And Coco Di Mama	Restaurant/Cafe/Canteen	1.0	1386687.0	51.408839	-0

		Upon Thames						
78782	78782	Kings Oak Primary School Dickerage Lane New Ma...	Zone All Sports And Care At Kings Oak Primary ...	School/college/university	7845.0	1594642.0	51.404842	-0
78783	78783	Malden Manor Primary And Nursery School Sheeph...	Zone All Sports And Care At Malden Manor Primary	School/college/university	7845.0	1594643.0	NaN	
78784	78784	Malden Parochial C Of E Primary School The Man...	Zone All Sports And Care At Malden Parochial S...	School/college/university	7845.0	1594647.0	51.381095	-0

78748 rows × 18 columns

Geopandas and folium plots

Functions for folium plots

Tube Data

In [23]:
```
coordinates_df = pd.read_csv("London Population /Candidate_location.csv")
```

In [121…
```
gdf_influential_candidate = gpd.GeoDataFrame(coordinates_df, geometry= gpd.points_from_x
```

Reading london tube data (stations and lines)

Tube data downloaded from downloaded from https://www.doogal.co.uk/london_stations

In [123…
```
tube_df = pd.read_csv('London Population /London tube lines.csv')
```

In [122…
```
# gpd.read_file('London_Train_Lines.kml')

gpd.io.file.fiona.drvsupport.supported_drivers['KML'] = 'rw'
tube_linesgdf = gpd.read_file('London Population /London_Train_Lines (1).kml', driver='K
```

In [124…
```
# gpd.read_file('London_Train_Lines.kml')
```

```
gpd.io.file.fiona.drvsupport.supported_drivers['KML'] = 'rw'
tube_stnsgdf = gpd.read_file('London Population /london_tube_stations.kml', driver='KML'
```

In [125…
```
# gpd.read_file('London_Train_Lines.kml')

gpd.io.file.fiona.drvsupport.supported_drivers['KML'] = 'rw'
tube_stnsgdf_distance = gpd.read_file('London Population /london_tube_stations.kml', dri
```

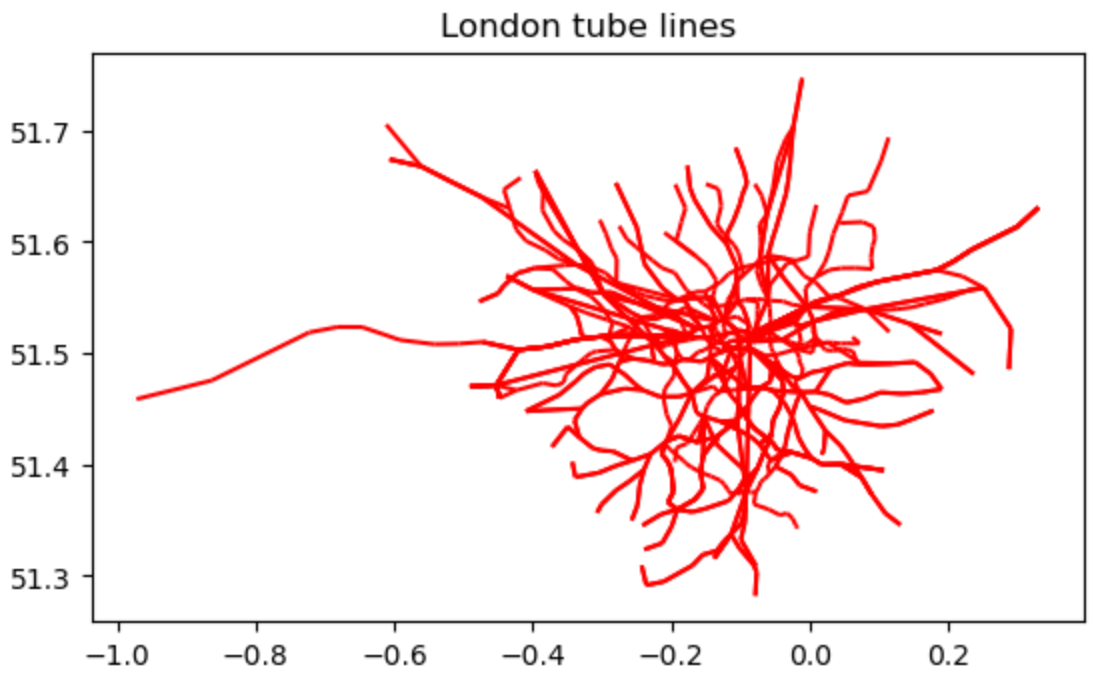
In [126…
```
import pandas as pd
from shapely.geometry import Point

# function to extract latitude and longitude from the Point object
def get_lat_lon(point):
    return point.x, point.y

# Convert the "geometry" column to Point objects
tube_stnsgdf_distance['geometry'] = tube_stnsgdf_distance['geometry'].apply(Point)

# Extract latitude and longitude from the Point objects and create new columns
tube_stnsgdf_distance['Latitude'], tube_stnsgdf_distance['Longitude'] = zip(*tube_stnsgd
```

In [30]:
```
[i for i in tube_linesgdf.loc[[0]].geometry[0].coords]
```

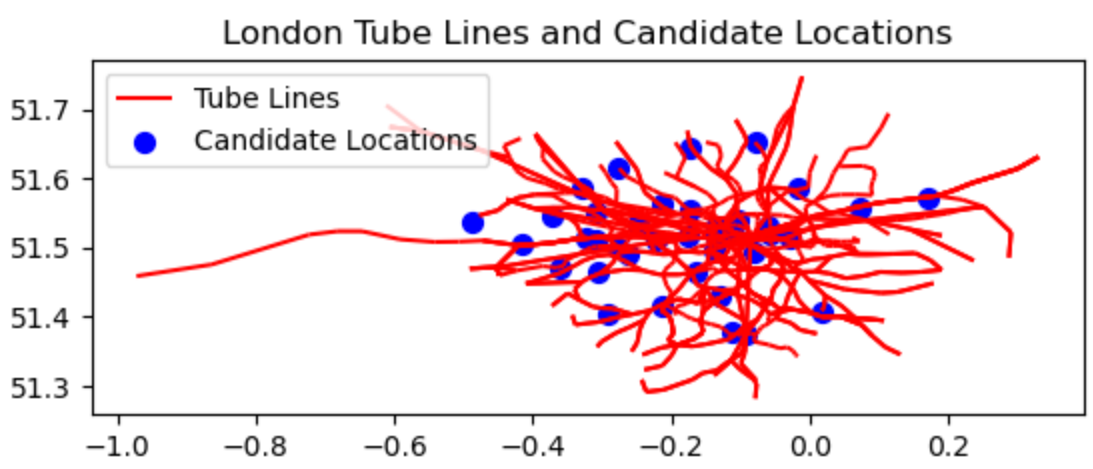
Out[30]:
```
[(-0.15690419298618452, 51.52312936605881, 1.0),
 (-0.1453444097205957, 51.523810437772845, 1.0)]
```

In [31]:
```
tube_linesgdf.plot(color='red')
plt.title('London tube lines')
```

Out[31]:
```
Text(0.5, 1.0, 'London tube lines')
```



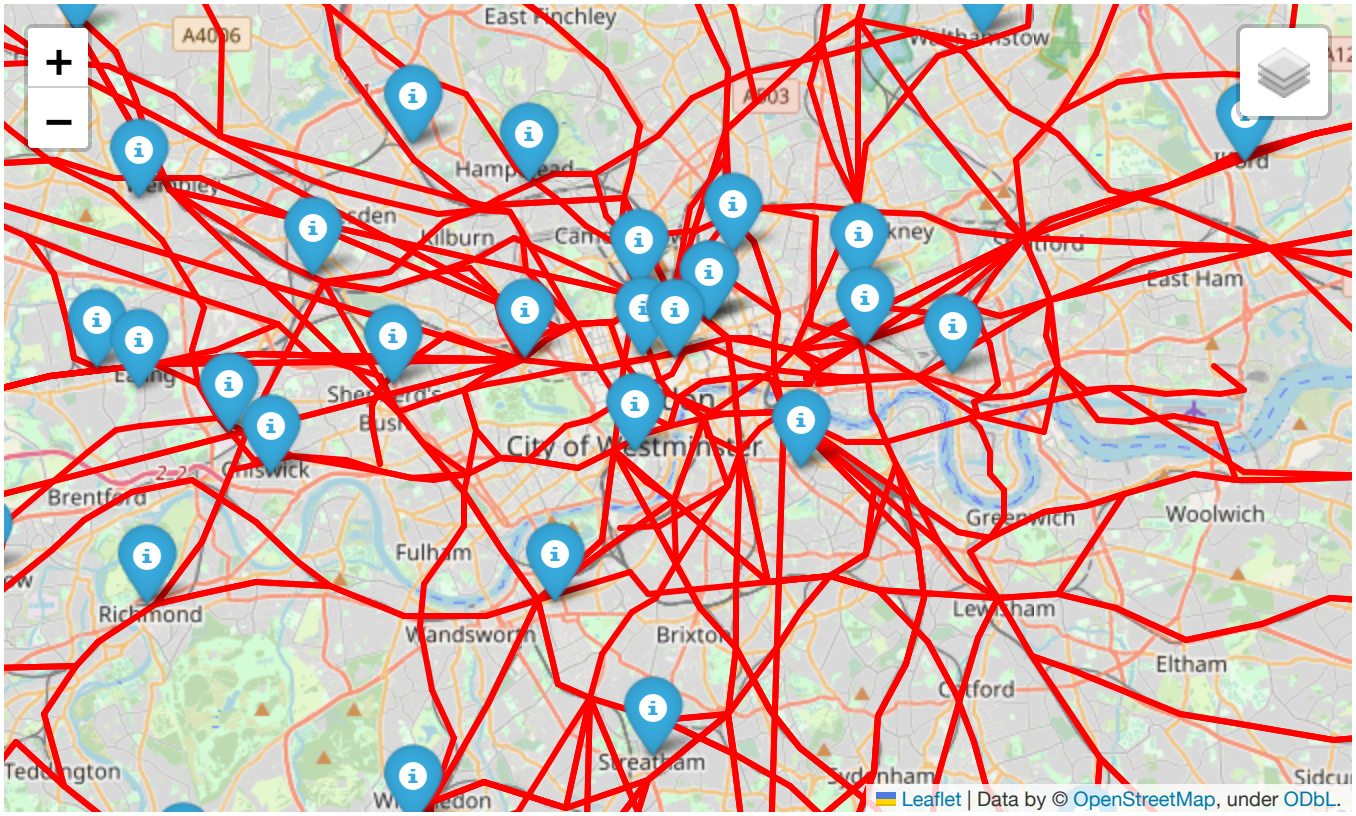
In [32]:
```
#showing what the tube lines look like in the real world
tube_linesgdf.plot(color='red',label='Tube Lines')
## Plotting the candidate locations
gdf_influential_candidate.plot(ax=plt.gca(), color='blue', marker='o', markersize=50, la

# Set the title and legend
plt.title('London Tube Lines and Candidate Locations')
plt.legend()

# Show the plot
plt.show()
```



In [33]:

```
# Create a folium map centered on London
m = folium.Map(location=[51.5074, -0.1278], zoom_start=11)

# Plot tube lines on the map
folium.GeoJson(tube_linesgdf, name='Tube Lines', style_function=lambda x: {'color': 'red

# Plot candidate locations on the map
for idx, row in gdf_influential_candidate.iterrows():
    folium.Marker([row['Latitude'], row['Longitude']], popup=f'Candidate Location {idx+1

# Add layer control to toggle between tube lines and candidate locations
folium.LayerControl().add_to(m)

# Display the map
m
```

Out[33]:

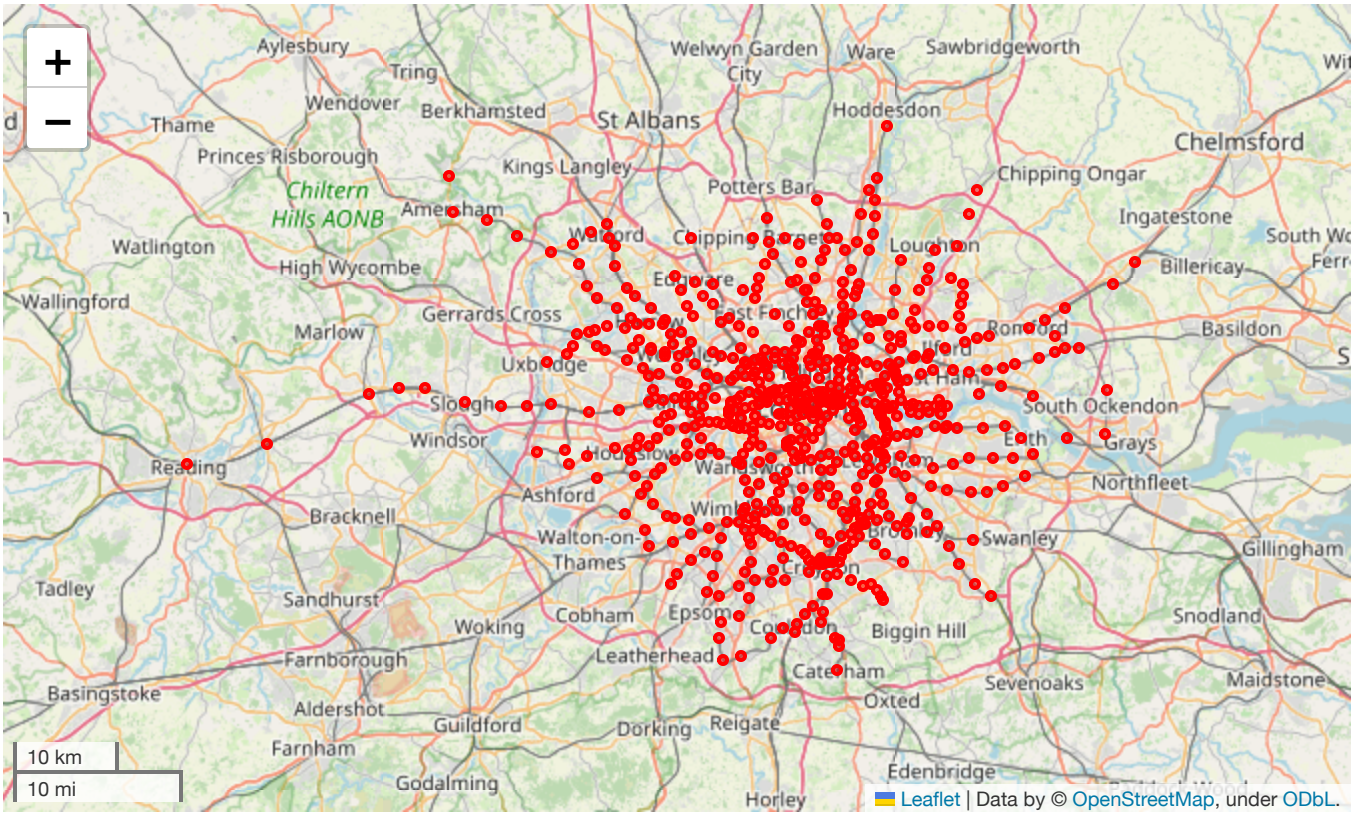


Obtaining from and to locations from tube lines to be added to a folium feature group

[list(tube_linesgdf.apply(lambda row: row.loc[[i for i in tube_linesgdf['indexx']]]).geometry[index_no].coords) for index_no in tube_linesgdf.index]

In [34]:
```
tube_stnsgdf.explore(color='red')  #showing what the points look like in the real world
```

Out[34]:



In [35]:
```
tube_stnsgdf.columns
```

Out[35]:
```
Index(['Name', 'Description', 'geometry'], dtype='object')
```

In [127…
```
# # Create a folium map centered on London
# m = folium.Map(location=[51.5074, -0.1278], zoom_start=11)

# # Plot tube lines on the map
# folium.GeoJson(tube_stnsgdf, name='Tube stations', style_function=lambda x: {'color':

# # Plot candidate locations on the map
# for idx, row in gdf_influential_candidate.iterrows():
#     folium.Marker([row['Latitude'], row['Longitude']], popup=f'Candidate Location {idx

# # Add layer control to toggle between tube lines and candidate locations
# folium.LayerControl().add_to(m)

# # Display the map
# m
```

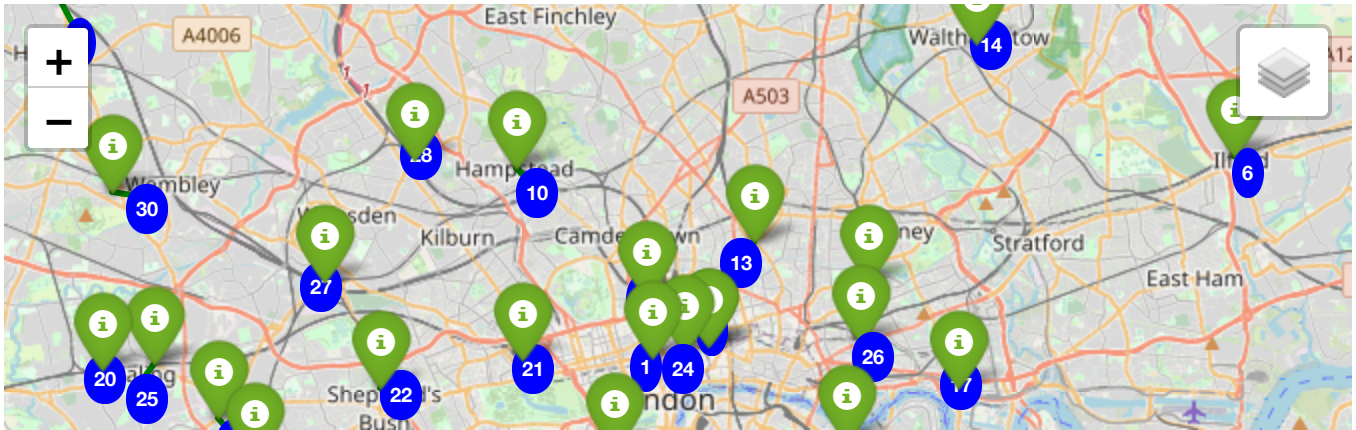
In [37]:
```
pip install geopy
```

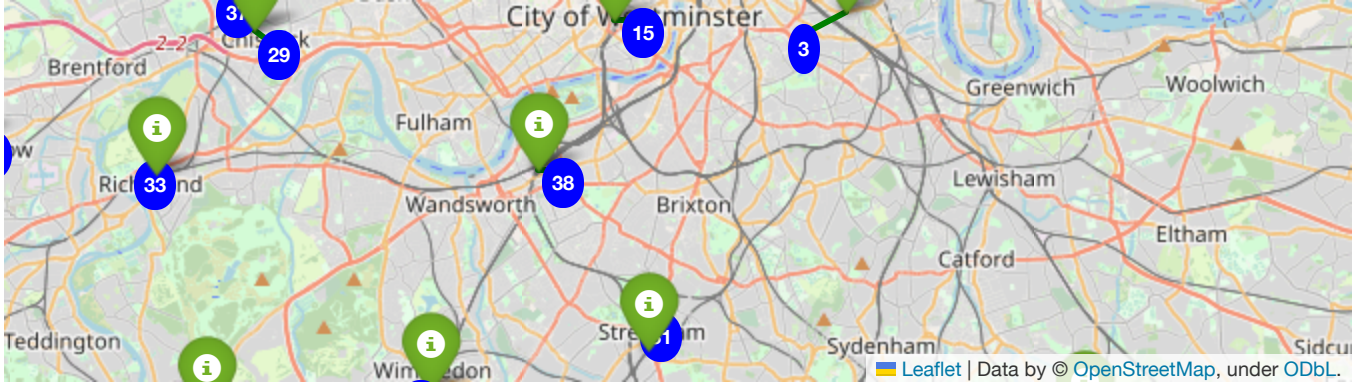
```
Requirement already satisfied: geopy in /Users/kruthikaramesh/opt/anaconda3/envs/newenv/
lib/python3.11/site-packages (2.3.0)
Requirement already satisfied: geographiclib<3,>=1.52 in /Users/kruthikaramesh/opt/anaco
nda3/envs/newenv/lib/python3.11/site-packages (from geopy) (2.0)
Note: you may need to restart the kernel to use updated packages.
```

In [39]:
```
from geopy.distance import geodesic
import folium
import pandas as pd
from math import radians
# Create a folium map centered on London
m = folium.Map(location=[51.5074, -0.1278], zoom_start=11)
```

```
# Plot candidate locations on the map and calculate distance to the nearest tube station
for idx, row in gdf_influential_candidate.iterrows():
    candidate_coords = (row['Latitude'], row['Longitude'])
    icon_html = f'<div style="display:inline-block;padding:5px;background-color:blue;col
    folium.Marker(candidate_coords, icon=folium.DivIcon(html=icon_html)).add_to(m)

    # Calculate distance to the nearest tube station
    nearest_tube_station = tube_stnsgdf_distance.iloc[
        tube_stnsgdf_distance.apply(lambda tube: geodesic(candidate_coords, (tube['geome
    ]
    distance_km = geodesic(candidate_coords, (nearest_tube_station['geometry'].y, neares

    # Add a line between candidate location and nearest tube station
    folium.PolyLine([candidate_coords, (nearest_tube_station['geometry'].y, nearest_tube
    folium.Marker((nearest_tube_station['geometry'].y, nearest_tube_station['geometry'].

# Add layer control to toggle between tube stations and candidate locations
folium.LayerControl().add_to(m)

# Create the DataFrame for the table
table_data = {
    'Candidate Location': [],
    'Candidate_latitude': [],
    'Candidate_longitude': [],
    'Nearest Tube Station': [],
    'Distance (km)': [],
    'Latitude': [],
    'Longitude': []
}

# Populate the table_data dictionary
for idx, row in gdf_influential_candidate.iterrows():
    candidate_coords = (row['Latitude'], row['Longitude'])
    nearest_tube_station = tube_stnsgdf_distance.iloc[
        tube_stnsgdf_distance.apply(lambda tube: geodesic(candidate_coords, (tube['geome
    ]
    distance_km = geodesic(candidate_coords, (nearest_tube_station['geometry'].y, neares

    table_data['Candidate Location'].append(f'Candidate Location {idx+1}')
    table_data['Candidate_latitude'].append(candidate_coords[0])
    table_data['Candidate_longitude'].append(candidate_coords[1])
    table_data['Nearest Tube Station'].append(nearest_tube_station['Name'])
    table_data['Distance (km)'].append(distance_km)
    table_data['Latitude'].append(nearest_tube_station['Latitude'])
    table_data['Longitude'].append(nearest_tube_station['Longitude'])

# Create the DataFrame
table_df = pd.DataFrame(table_data)

# Display the map and the DataFrame
m
```

Out[39]:





Distance between existing location and tube stations.

In [40]:
```
existing_location = pd.read_csv('Distance_candidate_existing.csv')
```

In [41]:
```
existing_location.columns
```

Out[41]:
```
Index(['Unnamed: 0', 'Candidate Latitude', 'Candidate Longitude',
       'Restaurant Name', 'Restaurant Cuisine Type', 'Restaurant Latitude',
       'Restaurant Longitude', 'Distance'],
      dtype='object')
```

In [42]:
```
merge_candidate_location_tube = pd.merge(existing_location,table_df, left_on = ['Candida
```

In [44]:
```
from sklearn.metrics.pairwise import haversine_distances
from math import radians
# Function to calculate distance between two points using haversine formula
def calculate_distance(row):
    # Convert latitude and longitude from degrees to radians
    lat1, lon1, lat2, lon2 = map(radians, [row['Restaurant Latitude'], row['Restaurant L

    # Haversine formula
    distances = haversine_distances(np.array([[lat1, lon1], [lat2, lon2]]))

    # Radius of the Earth in kilometers
    radius = 6371.0

    # Calculate the distance
    distance = distances[0, 1] * radius
    return distance

# Assuming you have the DataFrame named 'merge_candidate_location_tube' with the given c
merge_candidate_location_tube['Distance to Tube Station (km)'] = merge_candidate_locatio
```

In [128…
```
# Group by "Candidate Location" and calculate the average of "Distance to Tube Station (
average_distance_per_location = merge_candidate_location_tube.groupby('Candidate Locatio
```

In [47]:
```
tube_station_distance = pd.merge(table_df,average_distance_per_location, on = 'Candidate
```

In [48]:
```
tube_station_distance.to_csv("tube_stattion_distance.csv")
```

In [50]:
```
tube_station_distance = [['Candidate Location','Nearest Tube Station','Distance (km)','D
```

In [51]:
```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors


# Define a function to get the color based on the value of "Distance to Tube Station (km
```

```
def get_color(val):
    norm = plt.Normalize(merge_candidate_location_tube['Distance to Tube Station (km)'].
                         merge_candidate_location_tube['Distance to Tube Station (km)'].
    cmap = plt.cm.get_cmap('RdYlGn')  # You can choose any other colormap you prefer

    rgba_color = cmap(norm(val))
    color = mcolors.rgb2hex(rgba_color)
    return f'background-color: {color};'

# Apply the get_color function to the entire DataFrame
styled_df = merge_candidate_location_tube.style.apply(
    lambda row: [get_color(val) for val in row],
    subset=['Distance to Tube Station (km)']
)
```

In [129…

```
import pandas as pd
from math import radians, sin, cos, sqrt, atan2
from sklearn.metrics.pairwise import haversine_distances

# Function to calculate distance between two points using haversine formula
def calculate_distance(lat1, lon1, lat2, lon2):
    # Convert latitude and longitude from degrees to radians
    lat1, lon1, lat2, lon2 = map(radians, [lat1, lon1, lat2, lon2])

    # Haversine formula
    distances = haversine_distances(np.array([[lat1, lon1], [lat2, lon2]]))

    # Radius of the Earth in kilometers
    radius = 6371.0

    # Calculate the distance
    distance = distances[0, 1] * radius
    return distance

# Group Dataset 1 by candidate latitude and longitude
grouped_dataset1 = existing_location.groupby(['Candidate Latitude', 'Candidate Longitude

# Create a list to store the results
result = []

# Iterate through each group in Dataset 1
for group_name, group_data in grouped_dataset1:
    candidate_lat, candidate_lon = group_name

    # Get the corresponding candidate location from Dataset 2
    candidate_location = table_df.iloc[len(result)]

    # Calculate the distance for each restaurant in the group with the candidate locatio
    group_data['Distance from Candidate Location'] = group_data.apply(
        lambda row: calculate_distance(candidate_lat, candidate_lon, row['Restaurant Lat
        axis=1
    )

    # Append the updated group_data to the result list
    result.append(group_data)

# Concatenate the results back to a single DataFrame
final_result = pd.concat(result)
```

In [53]:

```
grouped_dataset1
```

Out[53]:

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x17df2ab50>
```

In [55]:

```
table_df.columns
```

Out[55]:
```
Index(['Candidate Location', 'Candidate_latitude', 'Candidate_longitude',
       'Nearest Tube Station', 'Distance (km)', 'Latitude', 'Longitude'],
      dtype='object')
```

In [56]:
```
table_df['Distance (km)'].describe()
```

Out[56]:
```
count    40.000000
mean      0.443337
std       0.294863
min       0.006812
25%       0.244696
50%       0.382542
75%       0.601850
max       1.241891
Name: Distance (km), dtype: float64
```

In [57]:
```
tube_stnsgdf.columns
```

Out[57]:
```
Index(['Name', 'Description', 'geometry'], dtype='object')
```

Connecting London tube lines and station points with restaurant data so that layers can be visualized

London Bus stops

In [59]:
```
bus_stops = pd.read_csv('London Population /bus-stops-10-06-15.csv')
```

In [61]:
```
import pyproj

def convert_easting_northing_to_lat_lon(df):
    # Define the British National Grid (BNG) and WGS84 (latitude and longitude) coordina
    bng = pyproj.Proj(init='epsg:27700')  # British National Grid
    wgs84 = pyproj.Proj(init='epsg:4326')  # WGS84 (latitude and longitude)

    # Convert Easting and Northing to Latitude and Longitude
    df['Longitude'], df['Latitude'] = pyproj.transform(bng, wgs84, df['Location_Easting'



    return df
# Call the function to add 'Latitude' and 'Longitude' columns
bus_stops = convert_easting_northing_to_lat_lon(bus_stops)
```

```
/Users/kruthikaramesh/opt/anaconda3/envs/newenv/lib/python3.11/site-packages/pyproj/crs/
crs.py:141: FutureWarning: '+init=<authority>:<code>' syntax is deprecated. '<authority
>:<code>' is the preferred initialization method. When making the change, be mindful of
axis order changes: https://pyproj4.github.io/pyproj/stable/gotchas.html#axis-order-chan
ges-in-proj-6
  in_crs_string = _prepare_from_proj_string(in_crs_string)
/Users/kruthikaramesh/opt/anaconda3/envs/newenv/lib/python3.11/site-packages/pyproj/crs/
crs.py:141: FutureWarning: '+init=<authority>:<code>' syntax is deprecated. '<authority
>:<code>' is the preferred initialization method. When making the change, be mindful of
axis order changes: https://pyproj4.github.io/pyproj/stable/gotchas.html#axis-order-chan
ges-in-proj-6
  in_crs_string = _prepare_from_proj_string(in_crs_string)
/var/folders/j_/yy54qztd2qv_yhxvrrgcbd1c0000gn/T/ipykernel_87893/3364698988.py:9: Future
Warning: This function is deprecated. See: https://pyproj4.github.io/pyproj/stable/gotch
as.html#upgrading-to-pyproj-2-from-pyproj-1
  df['Longitude'], df['Latitude'] = pyproj.transform(bng, wgs84, df['Location_Easting'].
values, df['Location_Northing'].values)
```

In [62]:
```
bus_stops = bus_stops[['Latitude','Longitude']]
```

In [130…
```
bus_stnsgdf = gpd.GeoDataFrame(bus_stops, geometry= gpd.points_from_xy(bus_stops.Longitu
```

In [64]:
```
bus_stnsgdf = bus_stnsgdf.replace([np.inf, -np.inf], np.nan)
bus_stnsgdf = bus_stnsgdf.dropna()
```

In [136…
```
# from geopy.distance import geodesic
# import pandas as pd

# # Create a list to store the data for the new dataset
# candidate_bus_distances = []

# # Calculate the distance between each candidate location and all bus stops
# for idx, candidate_row in gdf_influential_candidate.iterrows():
#     candidate_coords = (candidate_row['geometry'].y, candidate_row['geometry'].x)

#     # Create a list to store the distances and geometries for this candidate location
#     distances = []

#     for bus_idx, bus_row in bus_stnsgdf.iterrows():
#         bus_coords = (bus_row['geometry'].y, bus_row['geometry'].x)
#         distance_km = geodesic(candidate_coords, bus_coords).km
#         distances.append((bus_idx, distance_km, bus_row['geometry']))

#     # Sort the distances and take the 5 closest bus stops
#     distances.sort(key=lambda x: x[1])
#     closest_bus_stops = distances[:5]

#     # Append the data to the list
#     for bus_idx, distance_km, bus_geometry in closest_bus_stops:
#         candidate_bus_distances.append({
#             'Candidate Location': f'Candidate Location {idx+1}',
#             'Bus Stop Index': bus_idx,
#             'Distance (km)': distance_km,
#             'Bus Stop Geometry': bus_geometry
#         })

# # Create the new dataset
# candidate_bus_distances_df = pd.DataFrame(candidate_bus_distances)
```

In [68]:
```
candidate_bus_distances_df['Distance (km)'].describe()
```

Out[68]:
```
count    200.000000
mean       0.192471
std        0.107254
min        0.014490
25%        0.113824
50%        0.183627
75%        0.235593
max        0.556960
Name: Distance (km), dtype: float64
```

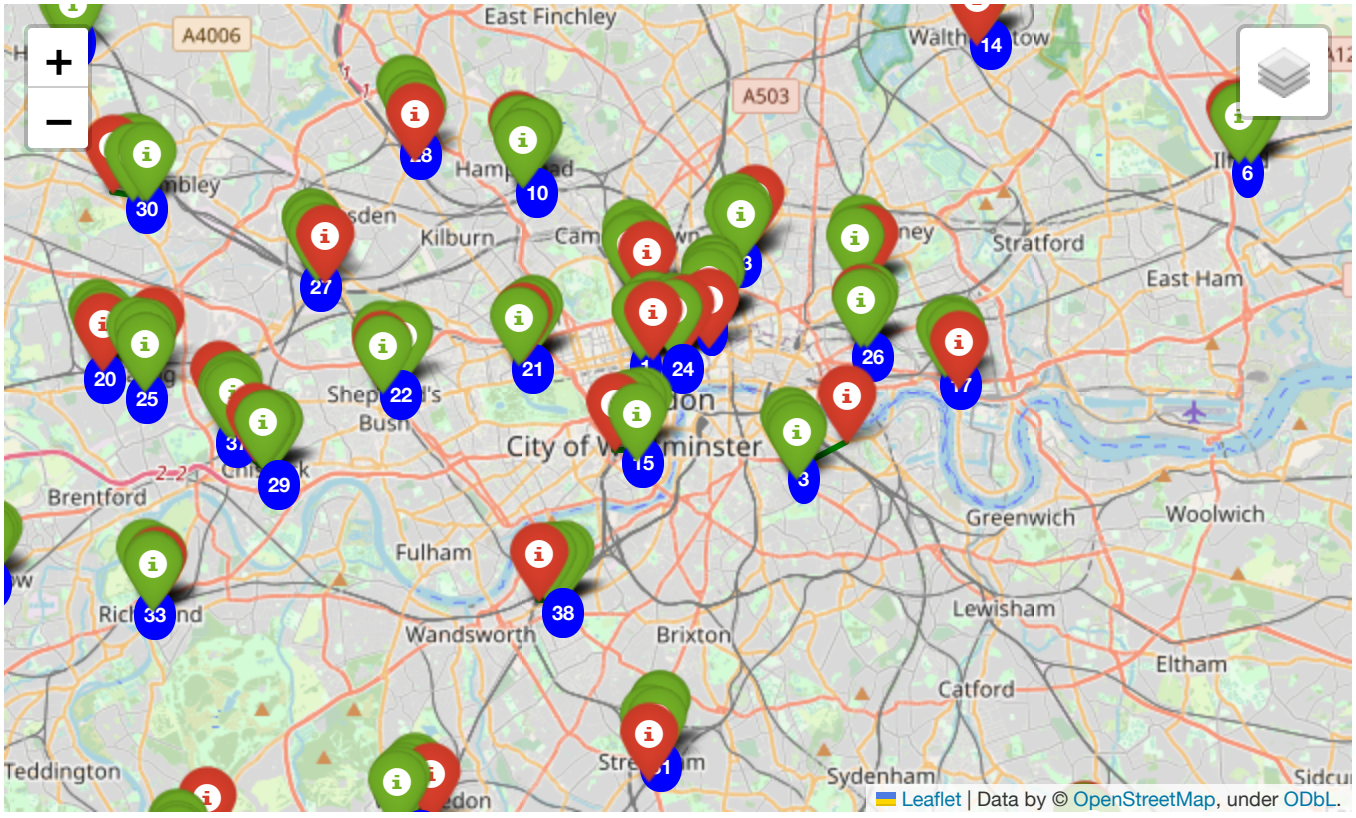
In [69]:
```
tube_stnsgdf.columns
```

Out[69]:
```
Index(['Name', 'Description', 'geometry'], dtype='object')
```

In [70]:
```
import folium

# Create a folium map centered on London
m = folium.Map(location=[51.5074, -0.1278], zoom_start=11)

# # Plot tube stations on the map
# for idx, row in tube_stnsgdf.iterrows():
#     tube_station_coords = (row['geometry'].y, row['geometry'].x)
#     folium.Marker(tube_station_coords, popup=f'Tube Station: {row["Name"]}', icon=foli
```

```
# Plot bus stations on the map
for idx, row in candidate_bus_distances_df.iterrows():
    bus_station_coords = (row['Bus Stop Geometry'].y, row['Bus Stop Geometry'].x)
    folium.Marker(bus_station_coords, popup=f'Bus Station: {idx}', icon=folium.Icon(colo

# Plot candidate locations on the map
for idx, row in gdf_influential_candidate.iterrows():
    candidate_coords = (row['Latitude'], row['Longitude'])
    icon_html = f'<div style="display:inline-block;padding:5px;background-color:blue;col
    folium.Marker(candidate_coords, icon=folium.DivIcon(html=icon_html)).add_to(m)
    # Calculate distance to the nearest tube station
    nearest_tube_station = tube_stnsgdf_distance.iloc[
        tube_stnsgdf_distance.apply(lambda tube: geodesic(candidate_coords, (tube['geome
    ]
    distance_km = geodesic(candidate_coords, (nearest_tube_station['geometry'].y, neares

    # Add a line between candidate location and nearest tube station
    folium.PolyLine([candidate_coords, (nearest_tube_station['geometry'].y, nearest_tube
    folium.Marker((nearest_tube_station['geometry'].y, nearest_tube_station['geometry'].


    # Add a line between candidate location and nearest tube station
#     tube_station_coords = (tube_stnsgdf.loc[row['Bus Stop Index']]['geometry'].y, tube
#     folium.PolyLine([candidate_coords, tube_station_coords], color='green').add_to(m)

    # Add a line between candidate location and nearest bus station
#     bus_station_coords = (candidate_bus_distances_df.loc[row['Bus Stop Index']]['geome
#     folium.PolyLine([candidate_coords, bus_station_coords], color='purple').add_to(m)

# Add layer control to toggle between tube stations, bus stations, and candidate locatio
folium.LayerControl().add_to(m)

# Display the map
m
```

Out[70]:



Cultural sites and open sites

In [72]:
```
tourist_attraction = pd.read_csv('GiGL_SpacesToVisit.csv')
```

In [73]:
```
tourist_attraction.columns
```

Out[73]:
```
Index(['SiteName', 'SiteID', 'PrimaryUse', 'Borough', 'AreaHa', 'Easting',
       'Northing', 'Qualifier', 'Access', 'DataSource', 'DataVersion',
       'ReleaseDate', 'Copyright'],
      dtype='object')
```

In [74]:
```
tourist_attraction = tourist_attraction[['SiteName', 'Borough','Easting','Northing']]
```

In [75]:
```
def convert_easting_northing_to_lat_lon(df):
    # Define the British National Grid (BNG) and WGS84 (latitude and longitude) coordina
    bng = pyproj.Proj(init='epsg:27700')  # British National Grid
    wgs84 = pyproj.Proj(init='epsg:4326')  # WGS84 (latitude and longitude)

    # Convert Easting and Northing to Latitude and Longitude
    df['Longitude'], df['Latitude'] = pyproj.transform(bng, wgs84, df['Easting'].values,


    return df
# Call the function to add 'Latitude' and 'Longitude' columns
tourist_attraction = convert_easting_northing_to_lat_lon(tourist_attraction)
```

```
/Users/kruthikaramesh/opt/anaconda3/envs/newenv/lib/python3.11/site-packages/pyproj/crs/
crs.py:141: FutureWarning: '+init=<authority>:<code>' syntax is deprecated. '<authority
>:<code>' is the preferred initialization method. When making the change, be mindful of
axis order changes: https://pyproj4.github.io/pyproj/stable/gotchas.html#axis-order-chan
ges-in-proj-6
  in_crs_string = _prepare_from_proj_string(in_crs_string)
/Users/kruthikaramesh/opt/anaconda3/envs/newenv/lib/python3.11/site-packages/pyproj/crs/
crs.py:141: FutureWarning: '+init=<authority>:<code>' syntax is deprecated. '<authority
>:<code>' is the preferred initialization method. When making the change, be mindful of
axis order changes: https://pyproj4.github.io/pyproj/stable/gotchas.html#axis-order-chan
ges-in-proj-6
  in_crs_string = _prepare_from_proj_string(in_crs_string)
/var/folders/j_/yy54qztd2qv_yhxvrrgcbd1c0000gn/T/ipykernel_87893/1146460687.py:7: Future
Warning: This function is deprecated. See: https://pyproj4.github.io/pyproj/stable/gotch
as.html#upgrading-to-pyproj-2-from-pyproj-1
  df['Longitude'], df['Latitude'] = pyproj.transform(bng, wgs84, df['Easting'].values, d
f['Northing'].values)
```

In [132…
```
tourist_attraction_gdf = gpd.GeoDataFrame(tourist_attraction, geometry= gpd.points_from_
```

In [77]:
```
from geopy.distance import geodesic
import pandas as pd

# Create a list to store the data for the new dataset
candidate_bus_distances = []

# Calculate the distance between each candidate location and all bus stops
for idx, candidate_row in gdf_influential_candidate.iterrows():
    candidate_coords = (candidate_row['geometry'].y, candidate_row['geometry'].x)

    # Create a list to store the distances and geometries for this candidate location
    distances = []

    for space_idx, space_row in tourist_attraction_gdf.iterrows():
        space_coords = (space_row['geometry'].y, space_row['geometry'].x)
        distance_km = geodesic(candidate_coords, space_coords).km
        distances.append((space_idx, distance_km, space_row['geometry']))

    # Sort the distances and take the 5 closest bus stops
    distances.sort(key=lambda x: x[1])
```

```
    closest_open_spaces = distances[:5]

    # Append the data to the list
    for space_idx, distance_km, space_geometry in closest_open_spaces:
        candidate_bus_distances.append({
            'Candidate Location': f'Candidate Location {idx+1}',
            'OpenSpace Index': space_idx,
            'Distance (km)': distance_km,
            'OpenSpace Geometry': space_geometry
        })

# Create the new dataset
tourist_distances_df = pd.DataFrame(candidate_bus_distances)
```

In [78]:
```
from geopy.distance import geodesic
import pandas as pd

# Create a list to store the data for the new dataset
candidate_bus_distances = []

# Calculate the distance between each candidate location and all bus stops
for idx, candidate_row in gdf_influential_candidate.iterrows():
    candidate_coords = (candidate_row['geometry'].y, candidate_row['geometry'].x)

    # Create a dictionary to store the distances and geometries for this candidate locat
    distances_dict = {}

    for space_idx, space_row in tourist_attraction_gdf.iterrows():
        space_coords = (space_row['geometry'].y, space_row['geometry'].x)
        distance_km = geodesic(candidate_coords, space_coords).km
        distances_dict[space_idx] = (distance_km, space_row['geometry'])

    # Sort the distances and take the 5 closest bus stops
    closest_open_space = sorted(distances_dict.items(), key=lambda x: x[1][0])[:5]

    # Append the data to the list
    for open_idx, (distance_km, open_geometry) in closest_open_space:
        candidate_bus_distances.append({
            'Candidate Location': f'Candidate Location {idx+1}',
            'OpenSpace Index': open_idx,
            'Distance OpenSpaces (km)': distance_km,
            'OpenSpace Geometry': open_geometry,
            'Top Bus Stop Latitude': open_geometry.y,
            'Top Bus Stop Longitude': open_geometry.x,
            'Candidate Latitude': candidate_coords[0],
            'Candidate Longitude': candidate_coords[1]
        })

# Create the new dataset
tourist_distances_df = pd.DataFrame(candidate_bus_distances)
```

In [118…
```
# Group by "Candidate Location" and calculate the average of "Distance to Tube Station (
average_distance_per_candidate = tourist_distances_df.groupby('Candidate Location')['Dis
```

Distance between OPenspaces and candidate location and Existing Restaurant

In [81]:
```
merge_candidate_location_openspaces = pd.merge(existing_location,tourist_distances_df, o
```

In [82]:
```
merge_candidate_location_openspaces = merge_candidate_location_openspaces[['Candidate La
```

In [83]:
```
merge_candidate_location_openspaces.columns
```

Out[83]:
```
Index(['Candidate Latitude', 'Candidate Longitude', 'Restaurant Name',
```

```
       'Restaurant Latitude', 'Restaurant Longitude', 'Candidate Location',
       'Distance OpenSpaces (km)', 'Top Bus Stop Latitude',
       'Top Bus Stop Longitude'],
      dtype='object')
```

In [84]:
```
# Function to calculate distance between two points using haversine formula
def calculate_distance(row):
    # Convert latitude and longitude from degrees to radians
    lat1, lon1, lat2, lon2 = map(radians, [row['Restaurant Latitude'], row['Restaurant L

    # Haversine formula
    distances = haversine_distances(np.array([[lat1, lon1], [lat2, lon2]]))

    # Radius of the Earth in kilometers
    radius = 6371.0

    # Calculate the distance
    distance = distances[0, 1] * radius
    return distance

# Assuming you have the DataFrame named 'merge_candidate_location_tube' with the given c
merge_candidate_location_openspaces['Distance to OPenSpaces (Rest)(km)'] = merge_candida
```

In [117…
```
# Group by "Candidate Location" and calculate the average of "Distance to Tube Station (
average_distance_per_location = merge_candidate_location_openspaces.groupby('Candidate L
```

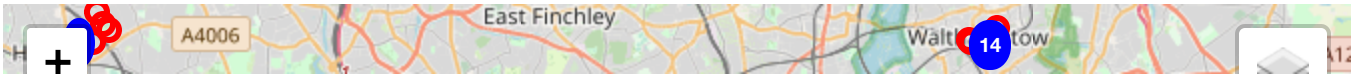
In [86]:
```
openSpace_station_distance = pd.merge(average_distance_per_candidate,average_distance_pe
```

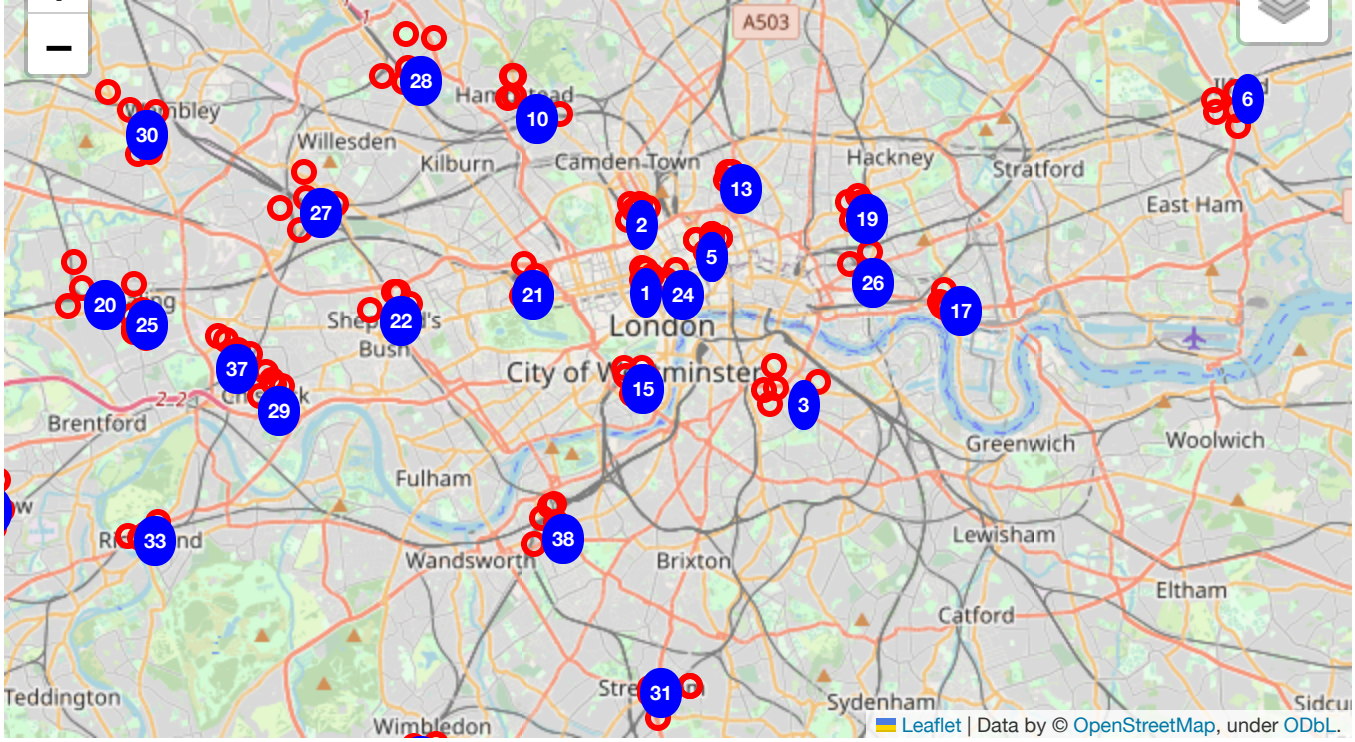
In [87]:
```
openSpace_station_distance_sort = pd.merge(table_df,openSpace_station_distance, on = 'Ca
```

In [89]:
```
openSpace_station_distance_sort.to_csv("Distance_open_Spaces.csv")
```

In [90]:
```
 from geopy.distance import geodesic

# Create a folium map centered on London
m = folium.Map(location=[51.5074, -0.1278], zoom_start=11)


# Plot bus stations on the map
for idx, row in tourist_distances_df.iterrows():
    bus_station_coords = (row['OpenSpace Geometry'].y, row['OpenSpace Geometry'].x)
    folium.CircleMarker(bus_station_coords, radius=5, color='red', popup=f'Bus Station:

candidate_locations = []
nearest_tube_stations = []
nearest_bus_stations = []
distances_tube = []
distances_bus = []

# Plot candidate locations on the map and calculate distance to the nearest tube and bus
for idx, row in gdf_influential_candidate.iterrows():
    candidate_coords = (row['Latitude'], row['Longitude'])
    icon_html = f'<div style="display:inline-block;padding:5px;background-color:blue;col
    folium.Marker(candidate_coords, icon=folium.DivIcon(html=icon_html)).add_to(m)
    #Add a line between candidate location and nearest bus station
   #bus_station_coords = (tourist_distances_df.loc[row['Bus']]['geometry'].y, tourist_di
    #folium.PolyLine([candidate_coords, bus_station_coords], color='purple').add_to(m)
folium.LayerControl().add_to(m)

# Display the map
m
```

Out[90]:





In [133…
```
# import folium

# # Create a folium map centered on London
# m = folium.Map(location=[51.5074, -0.1278], zoom_start=11)

# # Plot candidate locations on the map and label them with candidate index
# for idx, row in gdf_influential_candidate.iterrows():
#     candidate_coords = (row['Latitude'], row['Longitude'])
#     tooltip = f'Candidate Location {idx}'  # Tooltip to display the candidate index
#     icon_html = f'<div style="display:inline-block;padding:5px;background-color:blue;c
#     folium.Marker(candidate_coords, icon=folium.DivIcon(html=icon_html)).add_to(m)

# # Display the map
# m
```

Determining Schools nearby the candidate location

In [93]:
```
schools = pd.read_csv('London Population /all_schools_xy_2016.csv')
```

In [94]:
```
schools.columns
```

Out[94]:
```
Index(['OBJECTID', 'URN', 'SCHOOL_NAM', 'TYPE', 'PHASE', 'ADDRESS', 'TOWN',
       'POSTCODE', 'STATUS', 'GENDER', 'EASTING', 'NORTHING', 'Longitude ',
       'Latitude '],
      dtype='object')
```

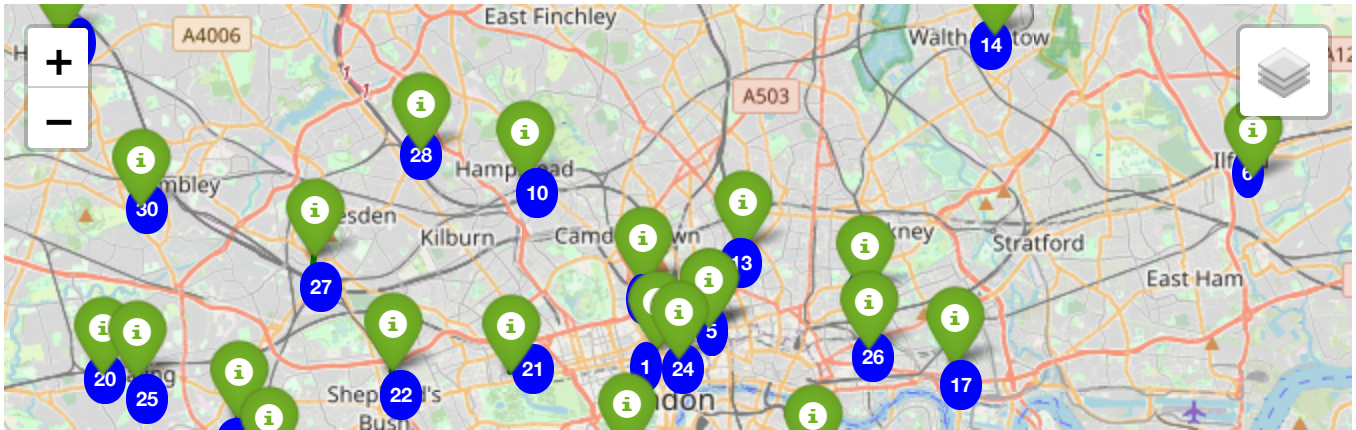
In [95]:
```
schools = schools[['SCHOOL_NAM','Longitude ','Latitude ']]
```

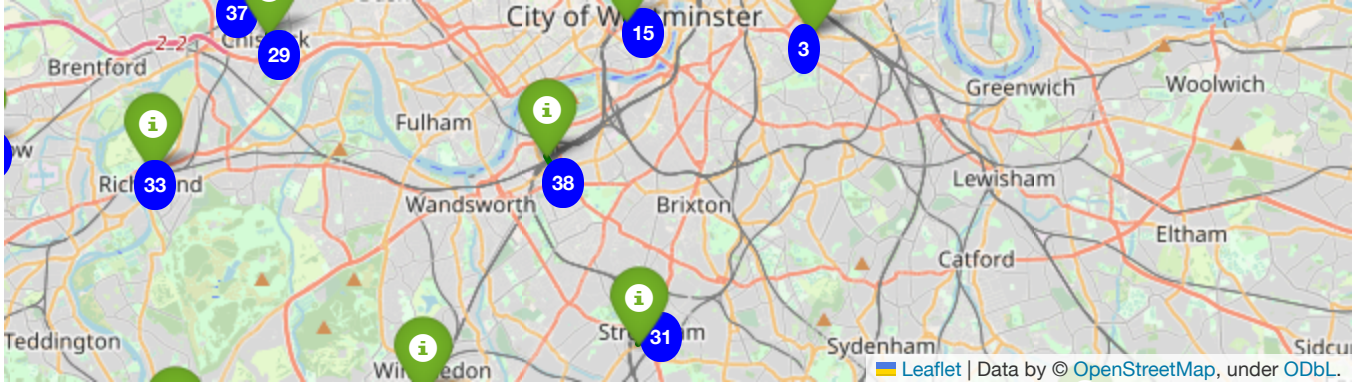
In [96]:
```
column_mapping = {
    'Longitude ': 'Longitude',
    'Latitude ': 'Latitude'
}

# Use the rename() method to replace the column names
schools.rename(columns=column_mapping, inplace=True)
```

In [134…
```
schools_gdf = gpd.GeoDataFrame(schools, geometry= gpd.points_from_xy(schools.Longitude,
```

In [98]:
```
from geopy.distance import geodesic
# Create a folium map centered on London
m = folium.Map(location=[51.5074, -0.1278], zoom_start=11)


candidate_locations = []
candidate_latitude = []
candidate_longitude = []
nearest_tube_stations = []
nearest_school_latitude = []
nearest_school_longitude = []
distances = []
# Plot candidate locations on the map and calculate distance to the nearest tube station
for idx, row in gdf_influential_candidate.iterrows():
    candidate_coords = (row['Latitude'], row['Longitude'])
    candidate_latitude.append(row['Latitude'])
    candidate_longitude.append(row['Longitude'])
    icon_html = f'<div style="display:inline-block;padding:5px;background-color:blue;col
    folium.Marker(candidate_coords, icon=folium.DivIcon(html=icon_html)).add_to(m)

    # Calculate distance to the nearest tube station
    nearest_school = schools_gdf.iloc[
        schools_gdf.apply(lambda tube: geodesic(candidate_coords, (tube['geometry'].y, t
    ]
    distance_km = geodesic(candidate_coords, (nearest_school['geometry'].y, nearest_scho
     # Add data to the lists
    candidate_locations.append(f'Candidate Location {idx+1}')
    nearest_school.append(nearest_school['SCHOOL_NAM'])
    distances.append(distance_km)
    nearest_school_latitude.append(nearest_school['geometry'].y)
    nearest_school_longitude.append(nearest_school['geometry'].x)
    # Add a line between candidate location and nearest tube station
    folium.PolyLine([candidate_coords, (nearest_school['geometry'].y, nearest_school['ge
    folium.Marker((nearest_school['geometry'].y, nearest_school['geometry'].x), popup=f'

# Add layer control to toggle between tube stations and candidate locations
folium.LayerControl().add_to(m)

# Create the DataFrame for the table
table_data = {
    'Candidate Location': candidate_locations,
    "Candidate Latitude": candidate_latitude,
    "Candidate Longitude": candidate_longitude,
    'Nearest School': nearest_school,
    'Nearest School Latitude': nearest_school_latitude,
    'Nearest School Longitude': nearest_school_longitude,
    'Distance (km)': distances

}
table_school = pd.DataFrame(table_data)

m
```

Out[98]:





In [99]:
```
table_school['Distance (km)'].describe()
```

Out[99]:
```
count    40.000000
mean      0.311129
std       0.153236
min       0.082324
25%       0.197369
50%       0.274332
75%       0.420903
max       0.720212
Name: Distance (km), dtype: float64
```

Distance ratio for candidate and existing restaurant from schools

In [101…
```
merge_candidate_location_schools = pd.merge(existing_location,table_school, on = ['Candi
```

In [102…
```
merge_candidate_location_schools.dropna(subset = ['Nearest School Latitude','Nearest Sch
```

Out[102]:

	Unnamed: 0	Candidate Latitude	Candidate Longitude	Restaurant Name	Restaurant Cuisine Type	Restaurant Latitude	Restaurant Longitude	Distance	Candi Loca
0	0	51.375744	-0.091776	Porter & Sorter	Nightlife Bars	51.375996	-0.091879	0.028880	Cand Locat
1	1	51.375744	-0.091776	AMT Coffee	Coffee Tea Caf	51.374989	-0.092898	0.114547	Cand Locat
2	2	51.375744	-0.091776	Costa Coffee	Cafe	51.374989	-0.092898	0.114547	Cand Locat
3	3	51.375744	-0.091776	Sodexo Ltd @ Defence And Govt	Pan Restaurant	51.376292	-0.093524	0.135754	Cand Locat
4	4	51.375744	-0.091776	Big Mikes Calypso Kitchen	Caribbean, Healthy, Jamaican	51.374892	-0.093697	0.163585	Cand Locat
...	...	...	...	...	...	...	...	...	
1195	1195	51.651840	-0.077056	Silverways School	Pan Restaurant	51.652896	-0.081317	0.316531	Cand Loc
1196	1196	51.651840	-0.077056	St Andrews Church Hall	Pan Restaurant	51.653003	-0.081410	0.327032	Cand Loc
1197	1197	51.651840	-0.077056	Caffe Nero	Cafe	51.651517	-0.081790	0.328567	Cand Loc

1198	1198	51.651840	-0.077056	The Fresh Bean Trading Company	Coffee Tea	51.654663	-0.075575	0.330142	Cand Loc
1199	1199	51.651840	-0.077056	En	Pan British	51.649782	-0.080751	0.342568	Cand Loc

1200 rows × 13 columns

In [103…
```
# Function to calculate distance between two points using haversine formula
def calculate_distance(row):
    # Convert latitude and longitude from degrees to radians
    lat1, lon1, lat2, lon2 = map(radians, [row['Restaurant Latitude'], row['Restaurant L

    # Haversine formula
    distances = haversine_distances(np.array([[lat1, lon1], [lat2, lon2]]))

    # Radius of the Earth in kilometers
    radius = 6371.0

    # Calculate the distance
    distance = distances[0, 1] * radius
    return distance


valid_rows = merge_candidate_location_schools.dropna(subset=['Restaurant Latitude', 'Res

# Calculate distance for valid rows
valid_rows['Distance to Schools (Rest)(km)'] = valid_rows.apply(calculate_distance, axis
#merge_candidate_location_schools['Distance to Schools (Rest)(km)'] = merge_candidate_lo
```

In [135…
```
# Group by "Candidate Location" and calculate the average of "Distance to Tube Station (
average_distance_per_school = valid_rows.groupby('Candidate Location')['Distance to Scho
```

In [105…
```
school_station_distance = pd.merge(table_df,average_distance_per_school, on = 'Candidate
```

In [106…
```
school_station_distance.to_csv("Distance_from_school.csv")
```

To plot the candidate locations which have a higher score when given equal weightage.

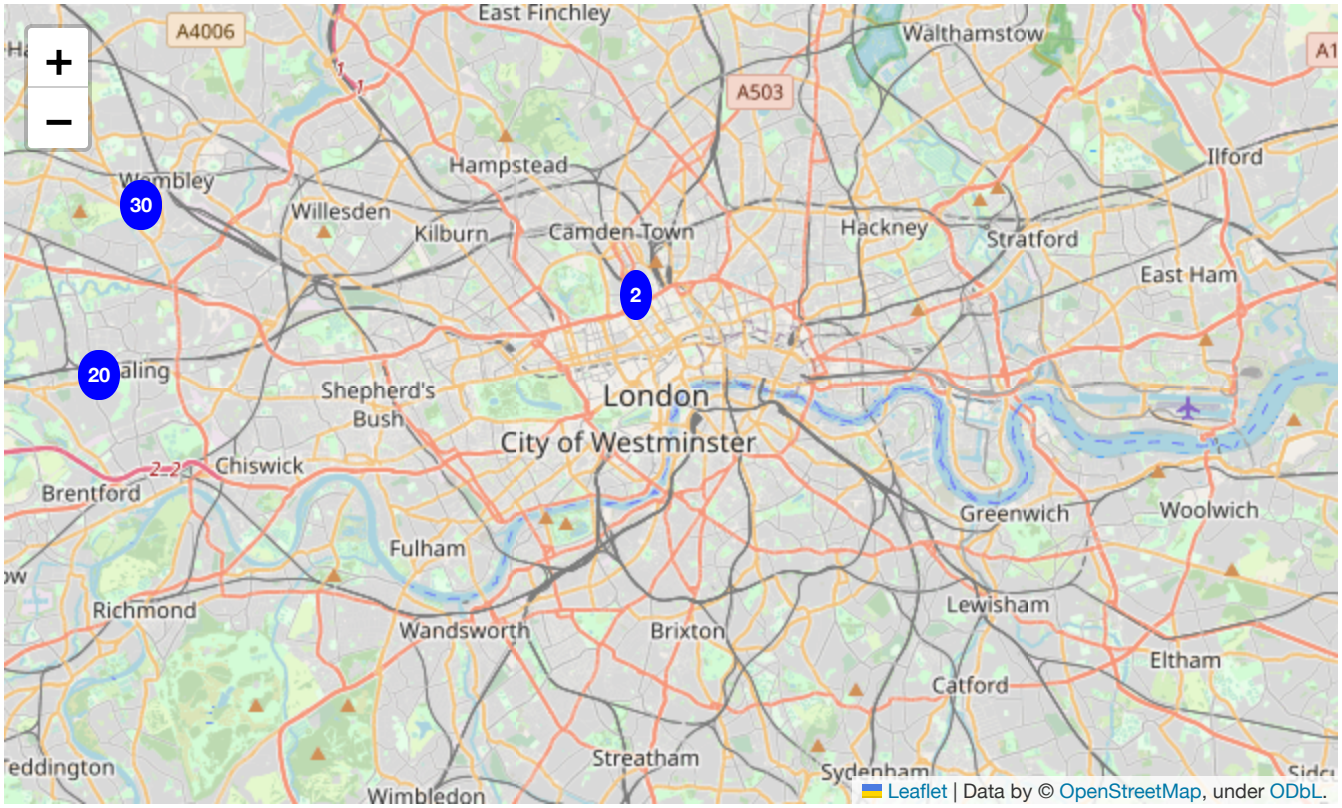
In [107…
```
score_candidate = pd.read_csv("Average Scores of Candidate Locations.csv")
```

In [108…
```
score_candidate_coordinates = pd.merge(score_candidate,table_school, on = 'Candidate Loc
```

In [110…
```
# Create a folium map centered on London
m = folium.Map(location=[51.5074, -0.1278], zoom_start=11)

# Loop through candidate locations
for idx, row in score_candidate_coordinates.iterrows():
    score = row['Average_score']

    # Only plot if the score is higher than or equal to 0.7
    if score >= 0.7:
        candidate_coords = (row['Candidate Latitude'], row['Candidate Longitude'])
        icon_html = f'<div style="display:inline-block;padding:5px;background-color:blue
        folium.Marker(candidate_coords, icon=folium.DivIcon(html=icon_html)).add_to(m)
```

```
# Show the map
m
```

Out[110]:



Plotting candidate locations when weightage is higher for Proximity to tube station

In [111…

```
score_tube_candidate = pd.read_csv("Weight for proximity tube station .csv")
```

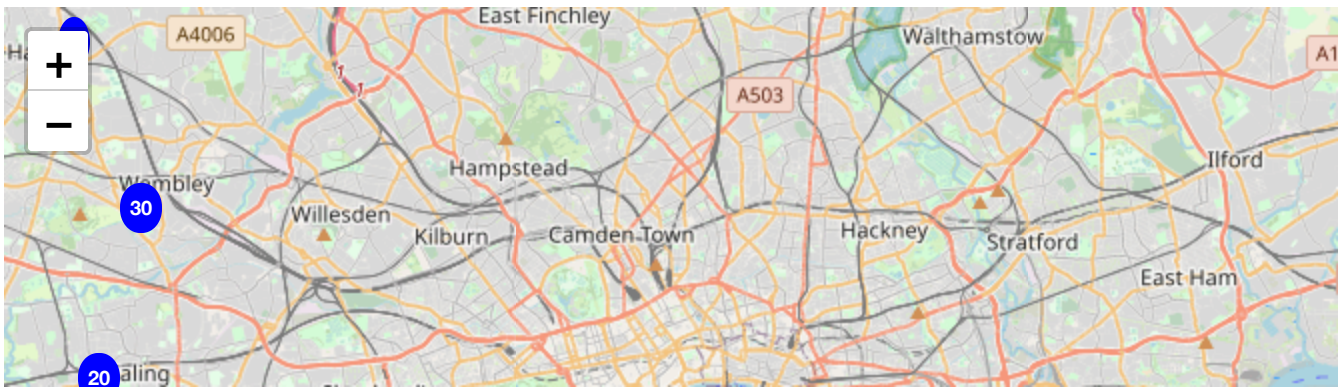
In [112…

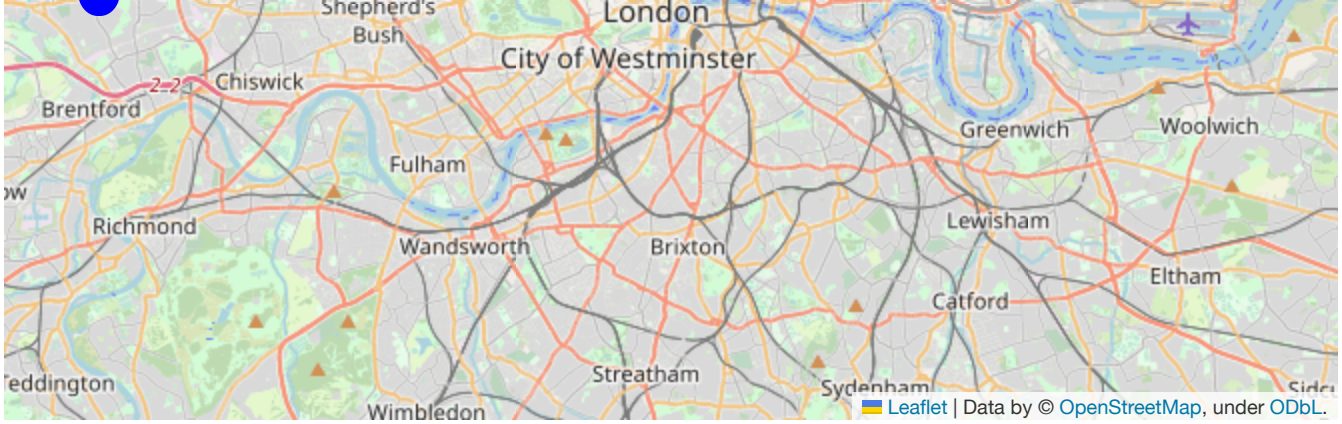
```
score_tube_candidate_coordinates = pd.merge(score_tube_candidate,table_school, on = 'Can
```

In [114…

```
# Create a folium map centered on London
m = folium.Map(location=[51.5074, -0.1278], zoom_start=11)

# Loop through candidate locations
for idx, row in score_tube_candidate_coordinates.iterrows():
    score = row['Sc1_Averagescore']

    # Only plot if the score is higher than or equal to 0.7
    if score >= 0.7:
        candidate_coords = (row['Candidate Latitude'], row['Candidate Longitude'])
        icon_html = f'<div style="display:inline-block;padding:5px;background-color:blue
        folium.Marker(candidate_coords, icon=folium.DivIcon(html=icon_html)).add_to(m)

# Show the map
m
```

Out[114]:




Shepherd's
Bush
London
City of Westminster
Chiswick
Brentford
Greenwich
Woolwich
Fulham
Richmond
Wandsworth
Brixton
Lewisham
Eltham
Catford
Streatham
Wimbledon
Leaflet | Data by © OpenStreetMap, under ODbL.

In [ ]: