In [115…

```
pip install -U notebook-as-pdf
```

```
Collecting notebook-as-pdf
  Downloading notebook_as_pdf-0.5.0-py3-none-any.whl (6.5 kB)
Requirement already satisfied: nbconvert in /Users/kruthikaramesh/opt/anaconda3/envs/new
env/lib/python3.11/site-packages (from notebook-as-pdf) (5.5.0)
Requirement already satisfied: pyppeteer in /Users/kruthikaramesh/opt/anaconda3/envs/new
env/lib/python3.11/site-packages (from notebook-as-pdf) (1.0.2)
Collecting PyPDF2 (from notebook-as-pdf)
  Downloading pypdf2-3.0.1-py3-none-any.whl (232 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 232.6/232.6 kB 3.6 MB/s eta 0:00:00a 0:00:0
1
Requirement already satisfied: mistune>=0.8.1 in /Users/kruthikaramesh/opt/anaconda3/env
s/newenv/lib/python3.11/site-packages (from nbconvert->notebook-as-pdf) (0.8.4)
Requirement already satisfied: jinja2>=2.4 in /Users/kruthikaramesh/opt/anaconda3/envs/n
ewenv/lib/python3.11/site-packages (from nbconvert->notebook-as-pdf) (3.1.2)
Requirement already satisfied: pygments in /Users/kruthikaramesh/opt/anaconda3/envs/newe
nv/lib/python3.11/site-packages (from nbconvert->notebook-as-pdf) (2.15.1)
Requirement already satisfied: traitlets>=4.2 in /Users/kruthikaramesh/opt/anaconda3/env
s/newenv/lib/python3.11/site-packages (from nbconvert->notebook-as-pdf) (5.7.1)
Requirement already satisfied: jupyter-core in /Users/kruthikaramesh/opt/anaconda3/envs/
newenv/lib/python3.11/site-packages (from nbconvert->notebook-as-pdf) (5.3.0)
Requirement already satisfied: nbformat>=4.4 in /Users/kruthikaramesh/opt/anaconda3/env
s/newenv/lib/python3.11/site-packages (from nbconvert->notebook-as-pdf) (5.7.0)
Requirement already satisfied: entrypoints>=0.2.2 in /Users/kruthikaramesh/opt/anaconda
3/envs/newenv/lib/python3.11/site-packages (from nbconvert->notebook-as-pdf) (0.4)
Requirement already satisfied: bleach in /Users/kruthikaramesh/opt/anaconda3/envs/newen
v/lib/python3.11/site-packages (from nbconvert->notebook-as-pdf) (4.1.0)
Requirement already satisfied: pandocfilters>=1.4.1 in /Users/kruthikaramesh/opt/anacond
a3/envs/newenv/lib/python3.11/site-packages (from nbconvert->notebook-as-pdf) (1.5.0)
Requirement already satisfied: testpath in /Users/kruthikaramesh/opt/anaconda3/envs/newe
nv/lib/python3.11/site-packages (from nbconvert->notebook-as-pdf) (0.6.0)
Requirement already satisfied: defusedxml in /Users/kruthikaramesh/opt/anaconda3/envs/ne
wenv/lib/python3.11/site-packages (from nbconvert->notebook-as-pdf) (0.7.1)
Requirement already satisfied: appdirs<2.0.0,>=1.4.3 in /Users/kruthikaramesh/opt/anacon
da3/envs/newenv/lib/python3.11/site-packages (from pyppeteer->notebook-as-pdf) (1.4.4)
Requirement already satisfied: certifi>=2021 in /Users/kruthikaramesh/opt/anaconda3/env
s/newenv/lib/python3.11/site-packages (from pyppeteer->notebook-as-pdf) (2023.7.22)
Requirement already satisfied: importlib-metadata>=1.4 in /Users/kruthikaramesh/opt/anac
onda3/envs/newenv/lib/python3.11/site-packages (from pyppeteer->notebook-as-pdf) (6.6.0)
Collecting pyee<9.0.0,>=8.1.0 (from pyppeteer->notebook-as-pdf)
  Using cached pyee-8.2.2-py2.py3-none-any.whl (12 kB)
Requirement already satisfied: tqdm<5.0.0,>=4.42.1 in /Users/kruthikaramesh/opt/anaconda
3/envs/newenv/lib/python3.11/site-packages (from pyppeteer->notebook-as-pdf) (4.65.0)
Requirement already satisfied: urllib3<2.0.0,>=1.25.8 in /Users/kruthikaramesh/opt/anaco
nda3/envs/newenv/lib/python3.11/site-packages (from pyppeteer->notebook-as-pdf) (1.26.1
5)
Requirement already satisfied: websockets<11.0,>=10.0 in /Users/kruthikaramesh/opt/anaco
nda3/envs/newenv/lib/python3.11/site-packages (from pyppeteer->notebook-as-pdf) (10.4)
Requirement already satisfied: zipp>=0.5 in /Users/kruthikaramesh/opt/anaconda3/envs/new
env/lib/python3.11/site-packages (from importlib-metadata>=1.4->pyppeteer->notebook-as-p
df) (3.15.0)
Requirement already satisfied: MarkupSafe>=2.0 in /Users/kruthikaramesh/opt/anaconda3/en
vs/newenv/lib/python3.11/site-packages (from jinja2>=2.4->nbconvert->notebook-as-pdf)
 (2.1.2)
Requirement already satisfied: fastjsonschema in /Users/kruthikaramesh/opt/anaconda3/env
s/newenv/lib/python3.11/site-packages (from nbformat>=4.4->nbconvert->notebook-as-pdf)
 (2.16.2)
Requirement already satisfied: jsonschema>=2.6 in /Users/kruthikaramesh/opt/anaconda3/en
vs/newenv/lib/python3.11/site-packages (from nbformat>=4.4->nbconvert->notebook-as-pdf)
 (4.17.3)
Requirement already satisfied: packaging in /Users/kruthikaramesh/opt/anaconda3/envs/new
env/lib/python3.11/site-packages (from bleach->nbconvert->notebook-as-pdf) (23.1)
Requirement already satisfied: six>=1.9.0 in /Users/kruthikaramesh/opt/anaconda3/envs/ne
```

```
wenv/lib/python3.11/site-packages (from bleach->nbconvert->notebook-as-pdf) (1.16.0)
Requirement already satisfied: webencodings in /Users/kruthikaramesh/opt/anaconda3/envs/
newenv/lib/python3.11/site-packages (from bleach->nbconvert->notebook-as-pdf) (0.5.1)
Requirement already satisfied: platformdirs>=2.5 in /Users/kruthikaramesh/opt/anaconda3/
envs/newenv/lib/python3.11/site-packages (from jupyter-core->nbconvert->notebook-as-pdf)
 (3.5.0)
Requirement already satisfied: attrs>=17.4.0 in /Users/kruthikaramesh/opt/anaconda3/env
s/newenv/lib/python3.11/site-packages (from jsonschema>=2.6->nbformat>=4.4->nbconvert->n
otebook-as-pdf) (23.1.0)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in /Users/k
ruthikaramesh/opt/anaconda3/envs/newenv/lib/python3.11/site-packages (from jsonschema>=
2.6->nbformat>=4.4->nbconvert->notebook-as-pdf) (0.18.0)
Installing collected packages: pyee, PyPDF2, notebook-as-pdf
  Attempting uninstall: pyee
    Found existing installation: pyee 9.0.4
    Uninstalling pyee-9.0.4:
      Successfully uninstalled pyee-9.0.4
ERROR: pip's dependency resolver does not currently take into account all the packages t
hat are installed. This behaviour is the source of the following dependency conflicts.
playwright 1.33.0 requires pyee==9.0.4, but you have pyee 8.2.2 which is incompatible.
Successfully installed PyPDF2-3.0.1 notebook-as-pdf-0.5.0 pyee-8.2.2
Note: you may need to restart the kernel to use updated packages.
```

In [55]:

```
!pip install plotly
!pip install matplotlib_scalebar
!pip install adjustText
```

```
Requirement already satisfied: plotly in /Users/kruthikaramesh/opt/anaconda3/lib/python
3.9/site-packages (5.15.0)
Requirement already satisfied: tenacity>=6.2.0 in /Users/kruthikaramesh/opt/anaconda3/li
b/python3.9/site-packages (from plotly) (8.2.2)
Requirement already satisfied: packaging in /Users/kruthikaramesh/opt/anaconda3/lib/pyth
on3.9/site-packages (from plotly) (23.0)
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: adjustText in /Users/kruthikaramesh/opt/anaconda3/lib/pyt
hon3.9/site-packages (0.8)
Requirement already satisfied: numpy in /Users/kruthikaramesh/opt/anaconda3/lib/python3.
9/site-packages (from adjustText) (1.25.0)
```

```
Requirement already satisfied: matplotlib in /Users/kruthikaramesh/opt/anaconda3/lib/pyt
hon3.9/site-packages (from adjustText) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /Users/kruthikaramesh/opt/anaconda3/l
ib/python3.9/site-packages (from matplotlib->adjustText) (1.0.5)
Requirement already satisfied: cycler>=0.10 in /Users/kruthikaramesh/opt/anaconda3/lib/p
ython3.9/site-packages (from matplotlib->adjustText) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /Users/kruthikaramesh/opt/anaconda3/
lib/python3.9/site-packages (from matplotlib->adjustText) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /Users/kruthikaramesh/opt/anaconda3/
lib/python3.9/site-packages (from matplotlib->adjustText) (1.4.4)
Requirement already satisfied: packaging>=20.0 in /Users/kruthikaramesh/opt/anaconda3/li
b/python3.9/site-packages (from matplotlib->adjustText) (23.0)
Requirement already satisfied: pillow>=6.2.0 in /Users/kruthikaramesh/opt/anaconda3/lib/
python3.9/site-packages (from matplotlib->adjustText) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /Users/kruthikaramesh/opt/anaconda3/l
ib/python3.9/site-packages (from matplotlib->adjustText) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in /Users/kruthikaramesh/opt/anacond
a3/lib/python3.9/site-packages (from matplotlib->adjustText) (2.8.2)
Requirement already satisfied: importlib-resources>=3.2.0 in /Users/kruthikaramesh/opt/a
naconda3/lib/python3.9/site-packages (from matplotlib->adjustText) (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->adjustText) (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->adjustText) (1.16.0)
```

In [2]:

```
pip install seaborn
```

```
Requirement already satisfied: seaborn in /Users/kruthikaramesh/opt/anaconda3/envs/newen
v/lib/python3.11/site-packages (0.12.2)
Requirement already satisfied: numpy!=1.24.0,>=1.17 in /Users/kruthikaramesh/opt/anacond
a3/envs/newenv/lib/python3.11/site-packages (from seaborn) (1.24.3)
Requirement already satisfied: pandas>=0.25 in /Users/kruthikaramesh/opt/anaconda3/envs/
newenv/lib/python3.11/site-packages (from seaborn) (2.0.1)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in /Users/kruthikaramesh/opt/anac
onda3/envs/newenv/lib/python3.11/site-packages (from seaborn) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /Users/kruthikaramesh/opt/anaconda3/e
nvs/newenv/lib/python3.11/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.0.7)
Requirement already satisfied: cycler>=0.10 in /Users/kruthikaramesh/opt/anaconda3/envs/
newenv/lib/python3.11/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /Users/kruthikaramesh/opt/anaconda3/
envs/newenv/lib/python3.11/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (4.39.
4)
Requirement already satisfied: kiwisolver>=1.0.1 in /Users/kruthikaramesh/opt/anaconda3/
envs/newenv/lib/python3.11/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in /Users/kruthikaramesh/opt/anaconda3/en
vs/newenv/lib/python3.11/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (23.1)
Requirement already satisfied: pillow>=6.2.0 in /Users/kruthikaramesh/opt/anaconda3/env
s/newenv/lib/python3.11/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (9.5.0)
Requirement already satisfied: pyparsing>=2.3.1 in /Users/kruthikaramesh/opt/anaconda3/e
nvs/newenv/lib/python3.11/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in /Users/kruthikaramesh/opt/anacond
a3/envs/newenv/lib/python3.11/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (2.
8.2)
Requirement already satisfied: pytz>=2020.1 in /Users/kruthikaramesh/opt/anaconda3/envs/
newenv/lib/python3.11/site-packages (from pandas>=0.25->seaborn) (2023.3)
Requirement already satisfied: tzdata>=2022.1 in /Users/kruthikaramesh/opt/anaconda3/env
s/newenv/lib/python3.11/site-packages (from pandas>=0.25->seaborn) (2023.3)
Requirement already satisfied: six>=1.5 in /Users/kruthikaramesh/opt/anaconda3/envs/newe
nv/lib/python3.11/site-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.1->sea
born) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:

```python
import pandas as pd
import numpy as np
import os
import fiona
import folium
import requests
import matplotlib
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
from math import floor
from scipy.spatial import KDTree
import math
```

```
/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 incompatibl
e with the GEOS version PyGEOS was compiled with (3.10.4-CAPI-1.16.2). Conversions betwe
en both will be slow.
  warnings.warn(
/var/folders/j_/yy54qztd2qv_yhxvrrgcbd1c0000gn/T/ipykernel_90477/2649238859.py:8: Deprec
ationWarning: 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 swit
ch to Shapely. To force to use Shapely 2.0 now, you can either uninstall PyGEOS or set t
he environment variable USE_PYGEOS=0. You can do this before starting the Python proces
s, 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 i
s installed. If you only have PyGEOS installed to get speed-ups, this switch should be s
mooth. 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 PyGE
OS to Shapely 2.0 (https://shapely.readthedocs.io/en/latest/migration_pygeos.html).
  import geopandas as gpd
```

In [4]:

```
from sklearn.metrics.pairwise import haversine_distances
```

In [5]:

```
def boundary_check(data,longitude,latitude):
    to_del = []
    long_del_east = data[data[longitude] > 0.4].index.to_list()
    long_del_west = data[data[longitude] <-0.5].index.to_list()
    lat_del_upper = data[data[latitude] >51.7].index.to_list()
    lat_del_lower = data[data[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)
    to_del = set(to_del)
    return to_del
```

Reading the dataset

1. Restaurant Data

In [6]:

```
raw_restaurant_data = pd.read_csv("London Population /Cleaned_restaurants.csv")
```

```
In [7]: nonnulll_raw_restaurant_data = raw_restaurant_data.dropna(subset=['Latitude','Longitude'
```

```
In [8]: print(f'Before deletion of rows: {len(nonnulll_raw_restaurant_data)}')
        rest_del = boundary_check(nonnulll_raw_restaurant_data,'Longitude','Latitude')
        nonnulll_raw_restaurant_data = nonnulll_raw_restaurant_data.drop(index=rest_del)
        print(f'After deletion of rows: {len(nonnulll_raw_restaurant_data)}')
```

```
Before deletion of rows: 64323
After deletion of rows: 64287
```

```
In [9]: # Define the string values
        string_values = ['restaurant', 'Pub', 'Catering', 'Hotel','caterer','Takeaway','Farmers/

        # Create a new column
        nonnulll_raw_restaurant_data['category_restaurant'] = nonnulll_raw_restaurant_data['Busi
```

```
In [10]: category_restaurant = nonnulll_raw_restaurant_data.loc[nonnulll_raw_restaurant_data.cate
```

```
In [11]: category_restaurant['Business Name '] = category_restaurant['Business Name '].astype(str
```

```
/var/folders/j_/yy54qztd2qv_yhxvrrgcbd1c0000gn/T/ipykernel_90477/2999336237.py:1: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  category_restaurant['Business Name '] = category_restaurant['Business Name '].astype(s
tr)
```

Importing the final file

```
In [12]: cuisine_restaurants = pd.read_csv("Final_combining_Restaurants .csv")
```

```
In [13]: import pandas as pd

         # Assuming your first dataset is named 'df1' and the second dataset is named 'df2'

         # Merge the two datasets based on 'Latitude' and 'Longitude', and keep only the desired
         merged_df = cuisine_restaurants.merge(nonnulll_raw_restaurant_data[['Latitude', 'Longitu
```

```
In [14]: merged_df = merged_df.drop_duplicates(subset=['Address '], keep='first')
```

Importing Post Code file

```
In [15]: london_postcode = pd.read_csv('London Population /london_postcodes-ons-postcodes-directo
```

```
In [16]: postcode_olsua = london_postcode[['pcds','oslaua']]
```

```
In [17]: merged_data = pd.merge(merged_df, postcode_olsua, left_on='Post Code', right_on='pcds',
```

```
In [18]: nonna_merged_data = merged_data.dropna(subset = ['Latitude'])
```

Data Preprocessing

We have noticed Rating value and hyginee columns containing NAN and Awaiting inspection values. For these rows we will perform numerical imputation using regression to predict the rating values based on the location. This is done instead of deleting the rows to prevent data loss. Imptuing with mean i not condisered as it can introduce bias.

In [19]:

```
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)
```

In [20]:

```
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
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

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

Exploratory Data Analysis

Impact of Population density

Here we are trying to analyse how population density changes with region.

In [21]:

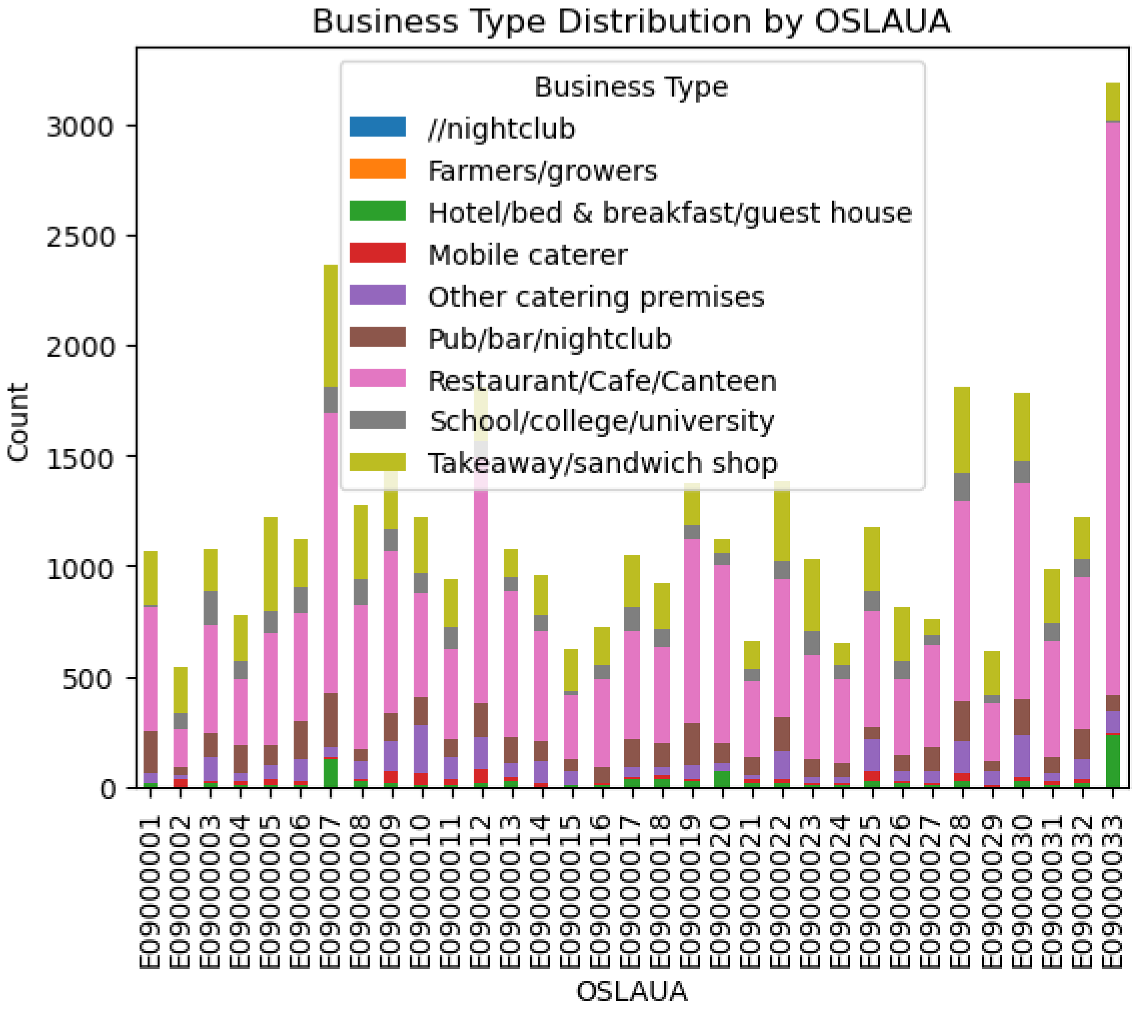
```
pop_den = pd.read_csv("London_df.csv")
pop_den["pop_dens"] = pop_den.Population/pop_den.Area
pop_den = pop_den.drop(['Indian_population','Lat','Lng','Population', 'Dist_from_center'
pop_den.rename(columns={"Borough":"Lower tier local authorities"},inplace=True)
```

Visualization of how restaruants are location based on the business type

In [23]:

```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming you have the data stored in a DataFrame called 'data'
grouped_data = nonna_merged_data.groupby('oslaua')['Business Type'].value_counts().unsta

plt.figure(figsize=(15, 5))
# Plotting the grouped data
grouped_data.plot(kind='bar', stacked=True)

# Setting the plot labels and title
plt.xlabel('OSLAUA')
plt.ylabel('Count')
plt.title('Business Type Distribution by OSLAUA')
plt.savefig('osla_regions_mapz_population_density.png')
# Display the plot
plt.show()
```

```
<Figure size 1500x500 with 0 Axes>
```

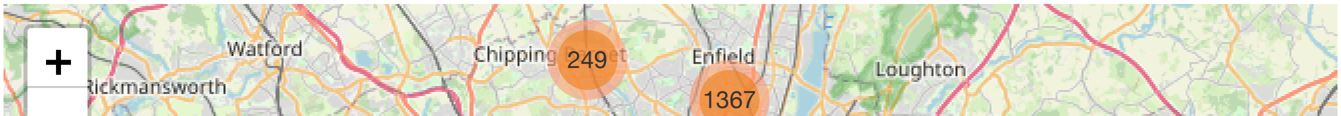


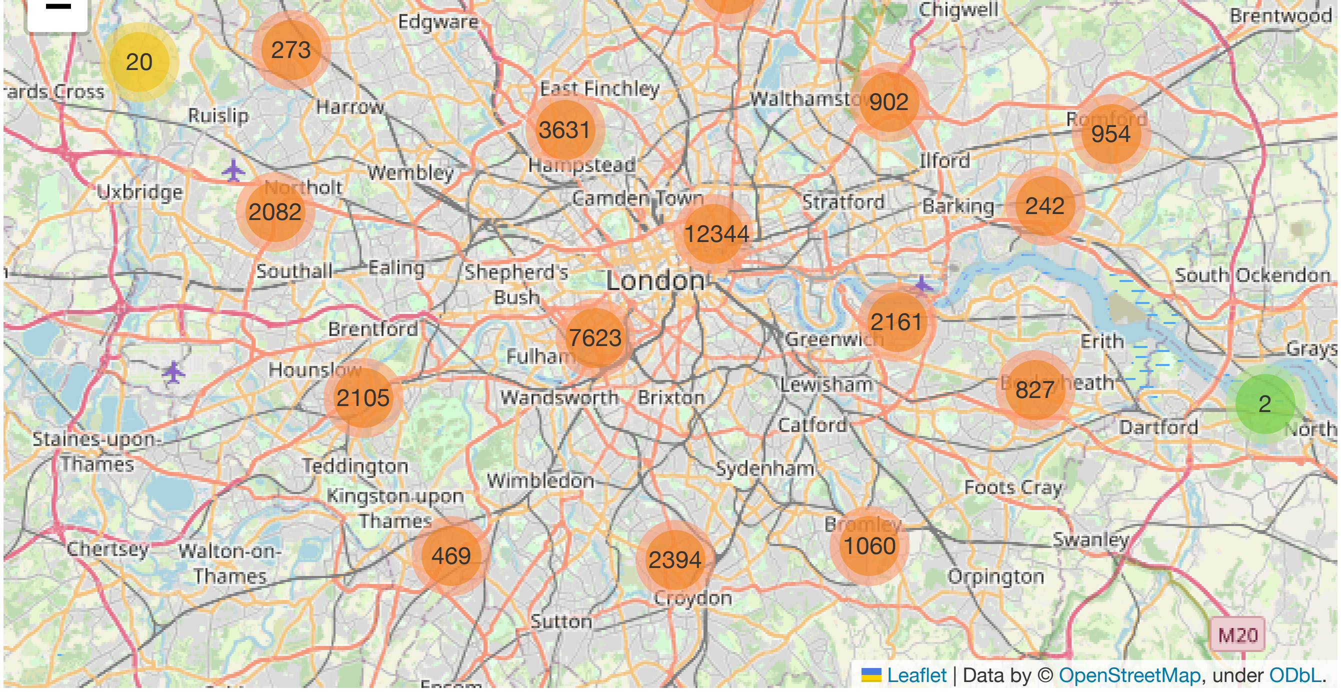
Plotting restaurants across London

In [114…

```
import folium
from folium.plugins import MarkerCluster

# Create a base map centered around London
london_map = folium.Map(location=[51.5074, -0.1278], zoom_start=10)

# Create a MarkerCluster layer
marker_cluster = MarkerCluster().add_to(london_map)

# Iterate over the dataset and add markers for each location
for index, row in nonna_merged_data.iterrows():
    oslaua = row['oslaua']
    latitude = row['Latitude']
    longitude = row['Longitude']

    tooltip = f"OSLAUA: {oslaua}"

    folium.Marker(
        location=[latitude, longitude],
        tooltip=tooltip,
        icon=folium.Icon(color='blue')
    ).add_to(marker_cluster)

# Display the map
london_map
```

Out[114]:





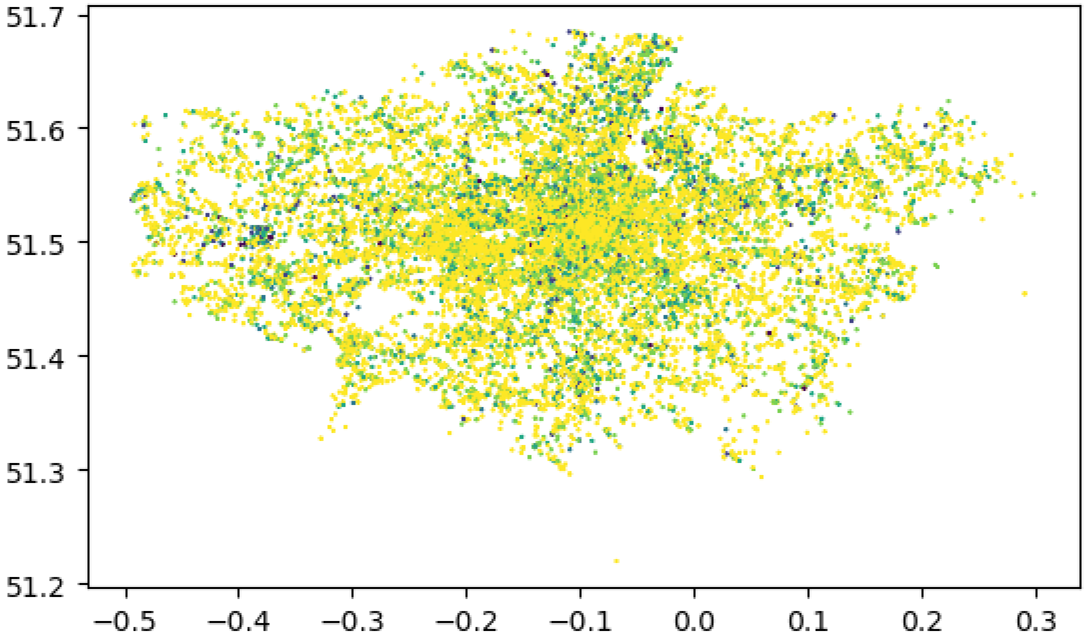
Ploting rating values for the restaurants

```
In [29]: gdp_restaurant = gpd.GeoDataFrame(nonna_merged_data, geometry=gpd.points_from_xy(nonna_m
         print(gdp_restaurant.geometry.crs)
         print(gdp_restaurant.total_bounds)
         hyg_with_rating = gdp_restaurant.copy()
         hyg_with_rating['Rating Value'] = hyg_with_rating['Rating Value'].astype('int32')   # con
         print(hyg_with_rating['Rating Value'].dtypes)
```

```
epsg:27700
[-0.4934373 51.2194365  0.298872   51.684863 ]
int32
```

```
In [31]: hyg_with_rating.plot(column='Rating Value',   marker='*', markersize=0.5)
```

```
Out[31]: <Axes: >
```



```
In [32]: # Group the data by "Predicted Cuisine Type" and count occurrences
         cuisine_counts = nonna_merged_data['Predicted Cuisine Type'].value_counts()
```

```
# Select the top 30 cuisines and their corresponding occurrence counts
top_30_cuisines = cuisine_counts.head(50)
```

Loading population data

In [33]:

```
london_population = pd.read_csv("London Population /Local Authority Code wise data.csv")
london_population = london_population.drop(['Number of Employees','Number of Unemployees
analysis_data = pd.merge(nonna_merged_data,london_population, on = 'oslaua', how = 'oute
drop_columns = ['pcds','Lower tier local authorities Code','Unnamed: 0_y']
analysis_data.drop(columns = drop_columns, inplace = True)
```

In [34]:

```
analysis_data.columns
```

Out[34]:

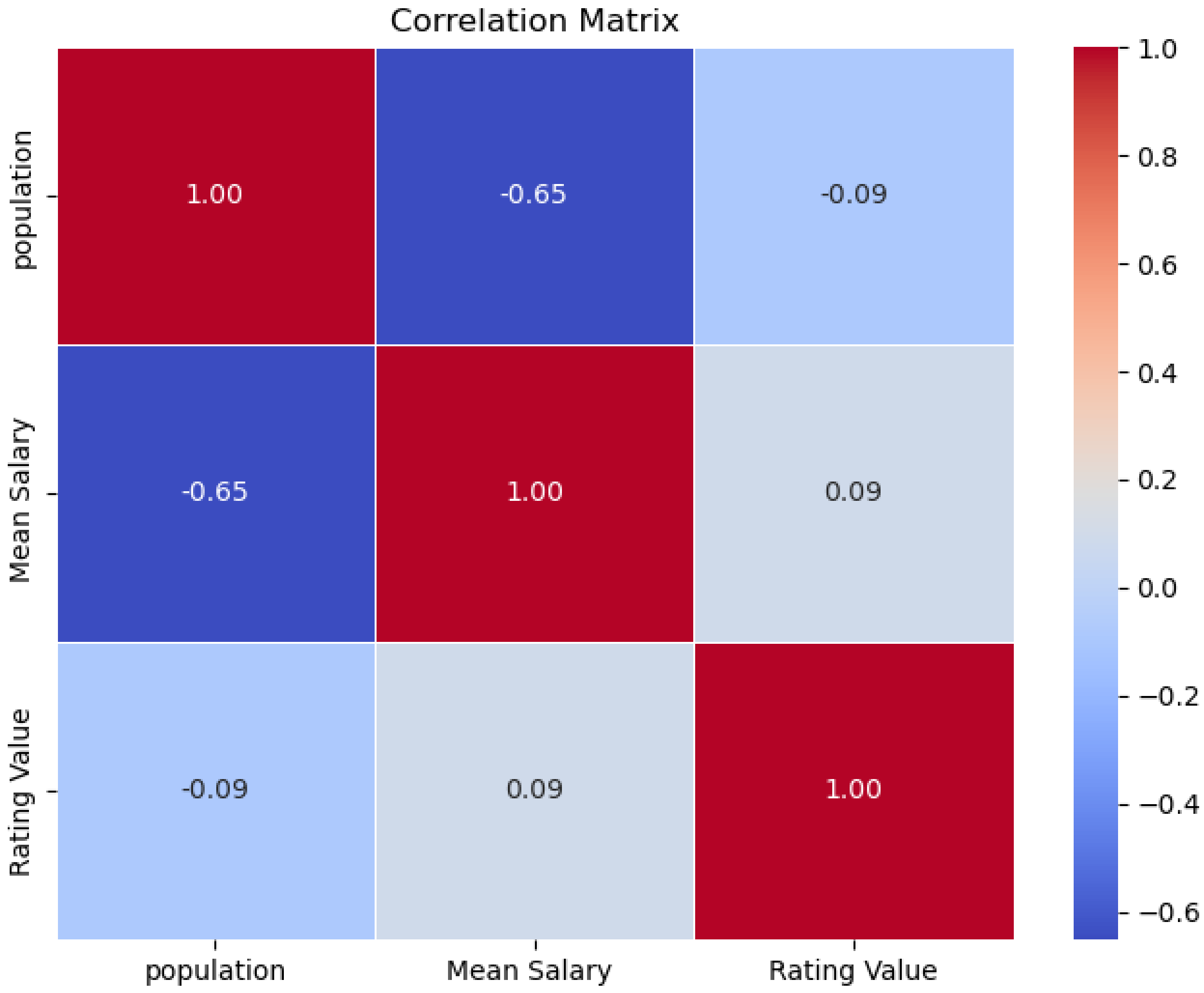
```
Index(['Unnamed: 0_x', 'Business Name ', 'Business Type', 'Rating Value',
       'Hygiene', 'processed_text', 'Predicted Cuisine Type', 'Latitude',
       'Longitude', 'Address ', 'Post Code', 'oslaua',
       'Lower tier local authorities', 'Mean Salary ', 'population', 'lat',
       'long'],
      dtype='object')
```

Analysis

Correlation

In [35]:

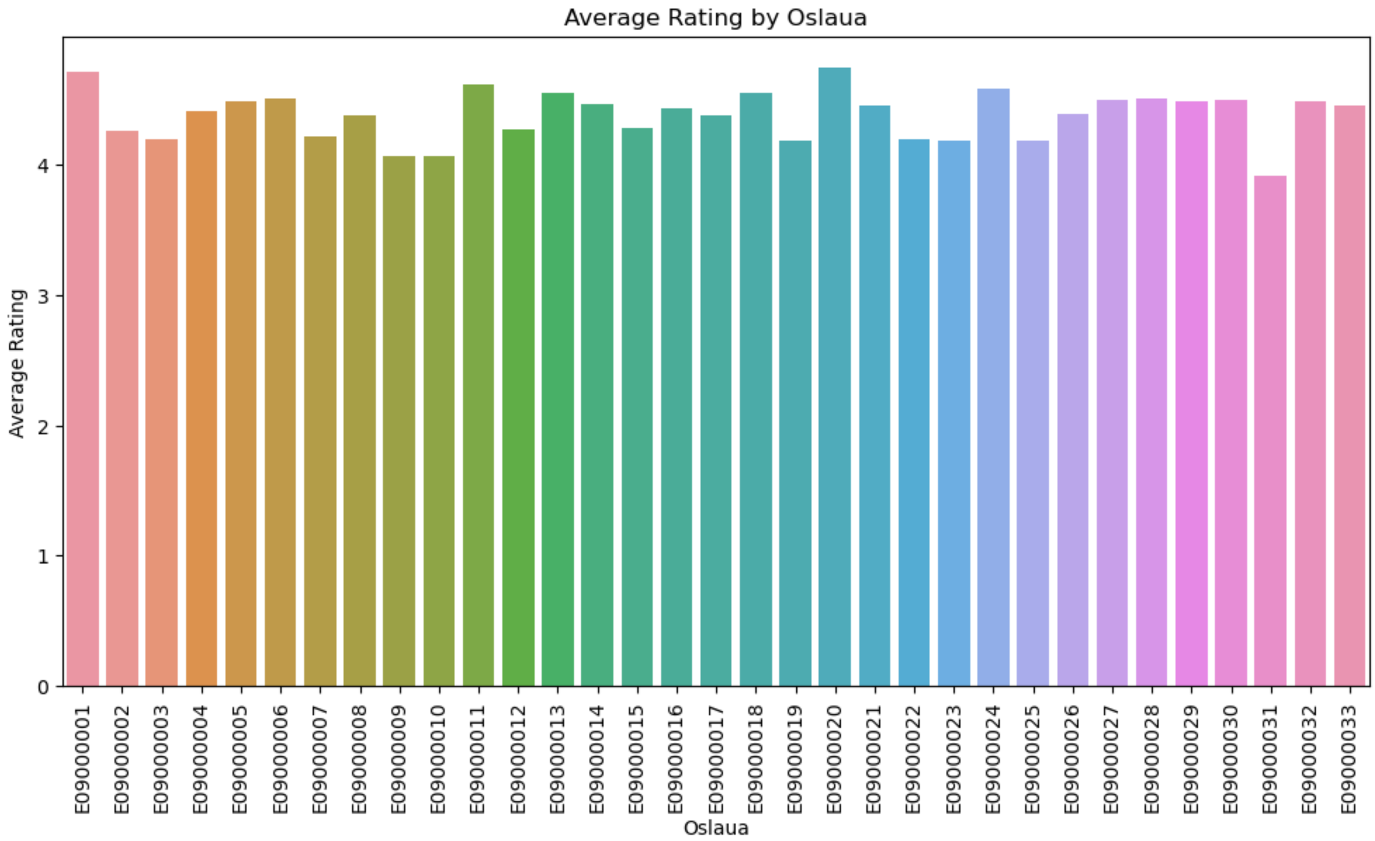
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming you have the 'analysis_data' dataset loaded in a DataFrame named 'data'

# Calculate the correlation matrix
correlation_matrix = analysis_data[['population', 'Mean Salary ', 'Rating Value']].corr(

# Plot the correlation matrix as a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()
```



Analysis of each OSLAUA region

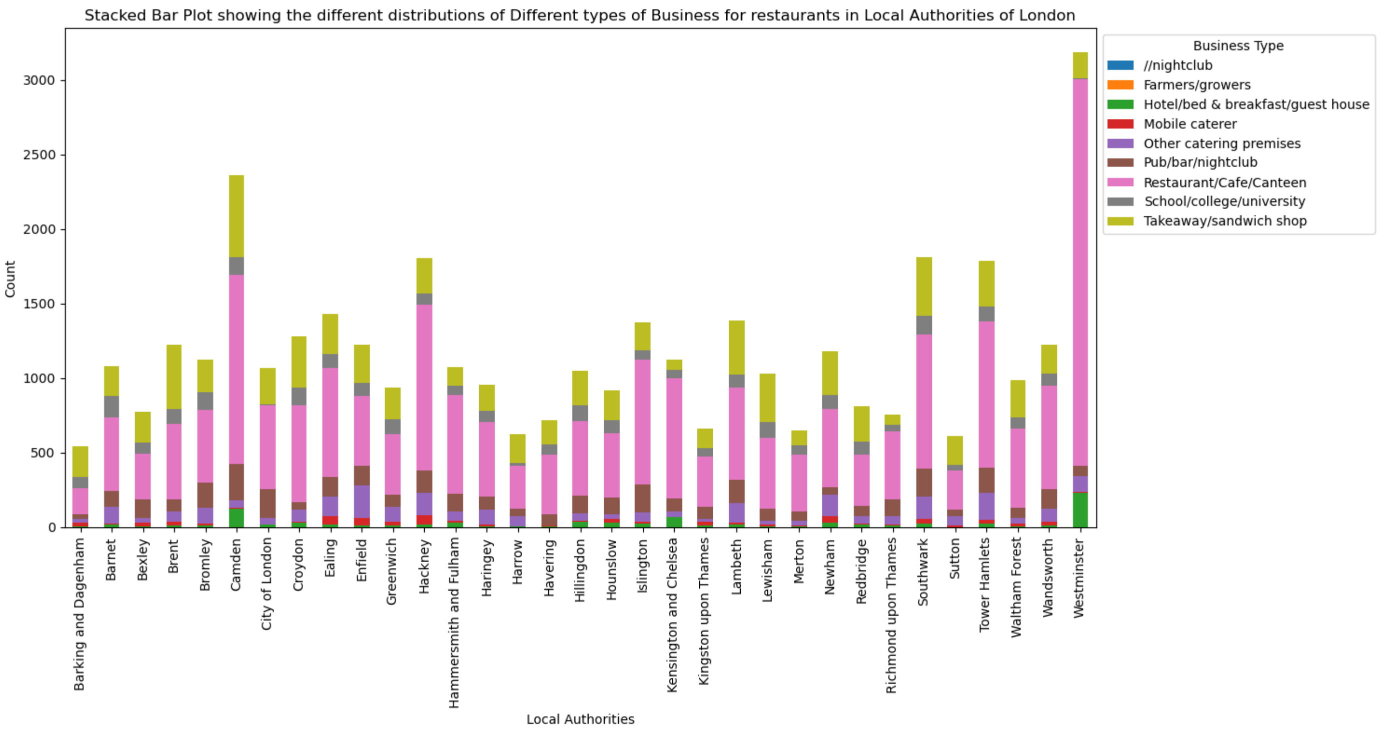
In [37]:

```
# Convert 'Rating Value' column to numeric type (float)
analysis_data['Rating Value'] = pd.to_numeric(analysis_data['Rating Value'], errors='coe

# Group the data by Oslaua and calculate the average rating for each group
average_rating_by_oslaua = analysis_data.groupby('oslaua')['Rating Value'].mean().reset_

# Sort the data by average rating in descending order
average_rating_by_oslaua = average_rating_by_oslaua.sort_values(by='Rating Value', ascen
average_rating_by_oslaua = average_rating_by_oslaua.sort_values('oslaua')


# Create a bar plot to visualize the average rating for each Oslaua
plt.figure(figsize=(12, 6))
sns.barplot(data=average_rating_by_oslaua, x='oslaua', y='Rating Value')
plt.title('Average Rating by Oslaua')
plt.xlabel('Oslaua')
plt.ylabel('Average Rating')
plt.xticks(rotation=90)
plt.show()
```



In [38]:

```
grouped_data = analysis_data.groupby(['Lower tier local authorities'])['Business Type'].
# Define a custom color map with 9 different colors for 9 business types
color_map = {
    '//nightclub': 'tab:blue',
    'Farmers/growers': 'tab:orange',
    'Hotel/bed & breakfast/guest house': 'tab:green',
    'Mobile caterer': 'tab:red',
    'Other catering premises': 'tab:purple',
    'Pub/bar/nightclub': 'tab:brown',
    'Restaurant/Cafe/Canteen': 'tab:pink',
    'School/college/university': 'tab:gray',
    'Takeaway/sandwich shop': 'tab:olive'
}

title = 'Stacked Bar Plot showing the different distributions of Different types of Busi

# Plot the stacked bar plot with different colors for each business type
ax = grouped_data.plot.bar(stacked=True, color=[color_map[col] for col in grouped_data.c

# Add labels to the plot
plt.xlabel('Local Authorities')
plt.ylabel('Count')
plt.legend(title='Business Type', bbox_to_anchor=(1, 1))

plt.savefig('osla_regions_mapz_business_type.png')

plt.tight_layout()
plt.show()
```



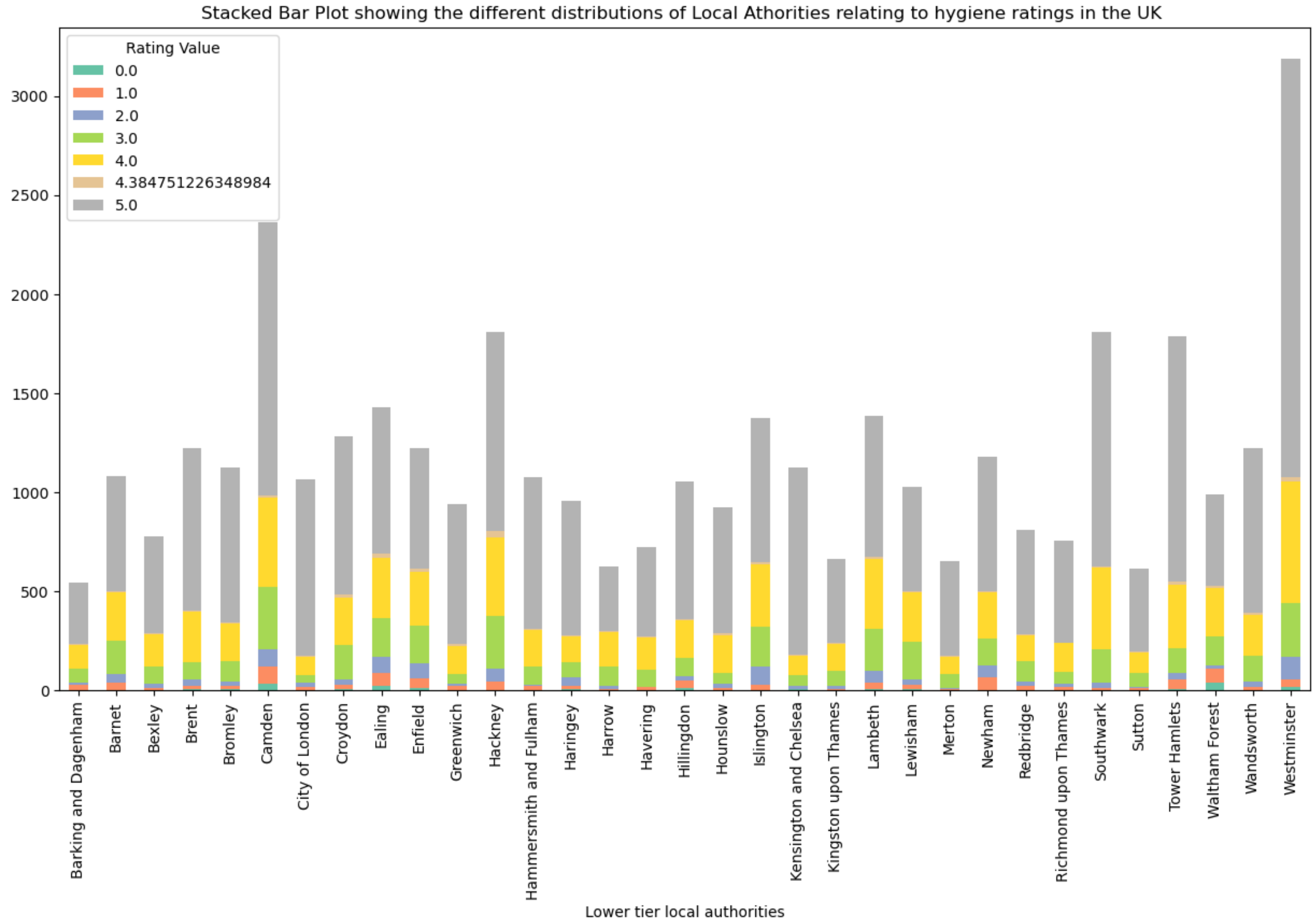
Visualization of how restaurants are located based on their rating

In [39]:

```
newFrame = analysis_data.groupby(['Lower tier local authorities'])['Rating Value'].value
title = 'Stacked Bar Plot showing the different distributions of Local Athorities relati

newFrame.plot.bar(stacked=True,  colormap ='Set2', title=title,figsize=(15,8))
```

Out[39]:

```
<Axes: title={'center': 'Stacked Bar Plot showing the different distributions of Local A
thorities relating to hygiene ratings in the UK '}, xlabel='Lower tier local authoritie
s'>
```


Stacked Bar Plot showing the different distributions of Local Athorities relating to hygiene ratings in the UK

In [40]:

```
import pandas as pd

# Assuming your data is stored in the 'analysis_data' DataFrame

# Convert 'Rating Value' column to numeric (if it contains any non-numeric values, they
analysis_data['Rating Value'] = pd.to_numeric(analysis_data['Rating Value'], errors='coe

# Group by the specified columns and calculate the mean of 'Rating Value'
average_ratings = analysis_data.groupby(['oslaua','lat','long','Lower tier local authori
```

Finding the center locations for OSLAUA

In [41]:

```
# Assuming you have the 'data' DataFrame loaded

# Group by OSLAUA and calculate the mean latitude and longitude for each group
center_locations = analysis_data.groupby('oslaua')[['Latitude', 'Longitude']].mean().res

# Rename the columns for clarity
center_locations.rename(columns={'Latitude': 'CenterLatitude', 'Longitude': 'CenterLongi
```

In [42]:

```
oslaua_average_rating = pd.merge(average_ratings,center_locations, on = 'oslaua', how =
oslaua_average_rating.drop(columns = ['lat','long'], inplace = True)
```

In [44]:

```
oslaua_average_rating
```

Out[44]:

	oslaua	Lower tier local authorities	Mean Salary	Rating Value	CenterLatitude	CenterLongitude
0	E09000001	City of London	94475.00000	4.716265	51.514646	-0.090344
1	E09000002	Barking and Dagenham	40064.00000	4.257502	51.547158	0.123973

2	E09000003	Barnet	54376.96682	4.197182	51.608302	-0.204421
3	E09000004	Bexley	45518.49315	4.417767	51.457631	0.134516
4	E09000005	Brent	43597.05202	4.492598	51.554119	-0.258364
5	E09000006	Bromley	52578.32487	4.505310	51.393071	0.025351
6	E09000007	Camden	67596.24060	4.215424	51.533683	-0.141707
7	E09000008	Croydon	46493.00000	4.383895	51.373840	-0.093970
8	E09000009	Ealing	47412.50000	4.072077	51.516268	-0.317113
9	E09000010	Enfield	45572.13115	4.068105	51.640694	-0.080040
10	E09000011	Greenwich	47122.18543	4.613092	51.477582	0.043141
11	E09000012	Hackney	45812.70833	4.270469	51.544316	-0.069927
12	E09000013	Hammersmith and Fulham	60644.51327	4.553561	51.493381	-0.218309
13	E09000014	Haringey	49649.86207	4.467339	51.589327	-0.101989
14	E09000015	Harrow	48687.73723	4.286579	51.588687	-0.339138
15	E09000016	Havering	46186.66667	4.431733	51.568744	0.204315
16	E09000017	Hillingdon	46634.65839	4.381835	51.537575	-0.440139
17	E09000018	Hounslow	46852.67606	4.552327	51.473139	-0.346601
18	E09000019	Islington	56228.94309	4.189674	51.542460	-0.107374
19	E09000020	Kensington and Chelsea	105737.08740	4.748271	51.499868	-0.187552
20	E09000021	Kingston upon Thames	53576.73469	4.452374	51.399037	-0.290776
21	E09000022	Lambeth	49160.28090	4.195694	51.463656	-0.118460
22	E09000023	Lewisham	45202.01183	4.191486	51.453747	-0.023660
23	E09000024	Merton	56379.51613	4.591007	51.412223	-0.194497
24	E09000025	Newham	39810.67073	4.191592	51.533617	0.027160
25	E09000026	Redbridge	47226.39752	4.394493	51.577428	0.068921
26	E09000027	Richmond upon Thames	66515.13043	4.495580	51.449931	-0.315159
27	E09000028	Southwark	50077.77108	4.515317	51.486130	-0.080586
28	E09000029	Sutton	48484.54545	4.492334	51.366729	-0.183961
29	E09000030	Tower Hamlets	50673.88889	4.496512	51.517308	-0.046229
30	E09000031	Waltham Forest	42692.91667	3.913405	51.585769	-0.012216
31	E09000032	Wandsworth	61769.49721	4.492460	51.453097	-0.176085
32	E09000033	Westminster	79718.43750	4.451544	51.511912	-0.148680

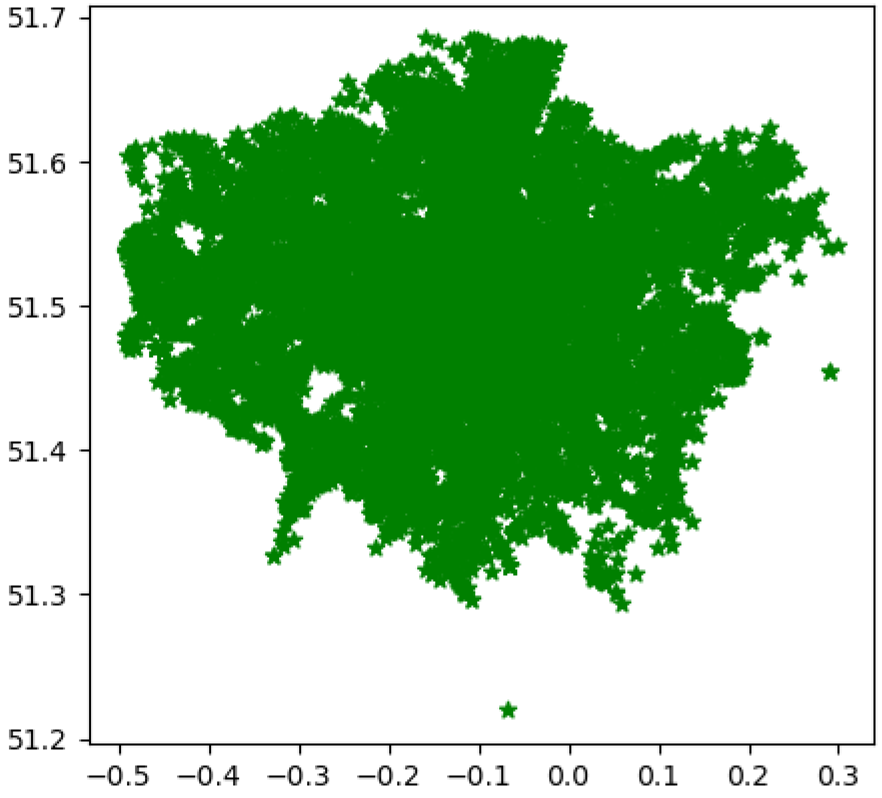
In [53]:

```
pop_merge = pd.merge(pop_den ,oslaua_average_rating, on = 'Lower tier local authorities'
```

Plotting geographic information with respect to OSLAUA

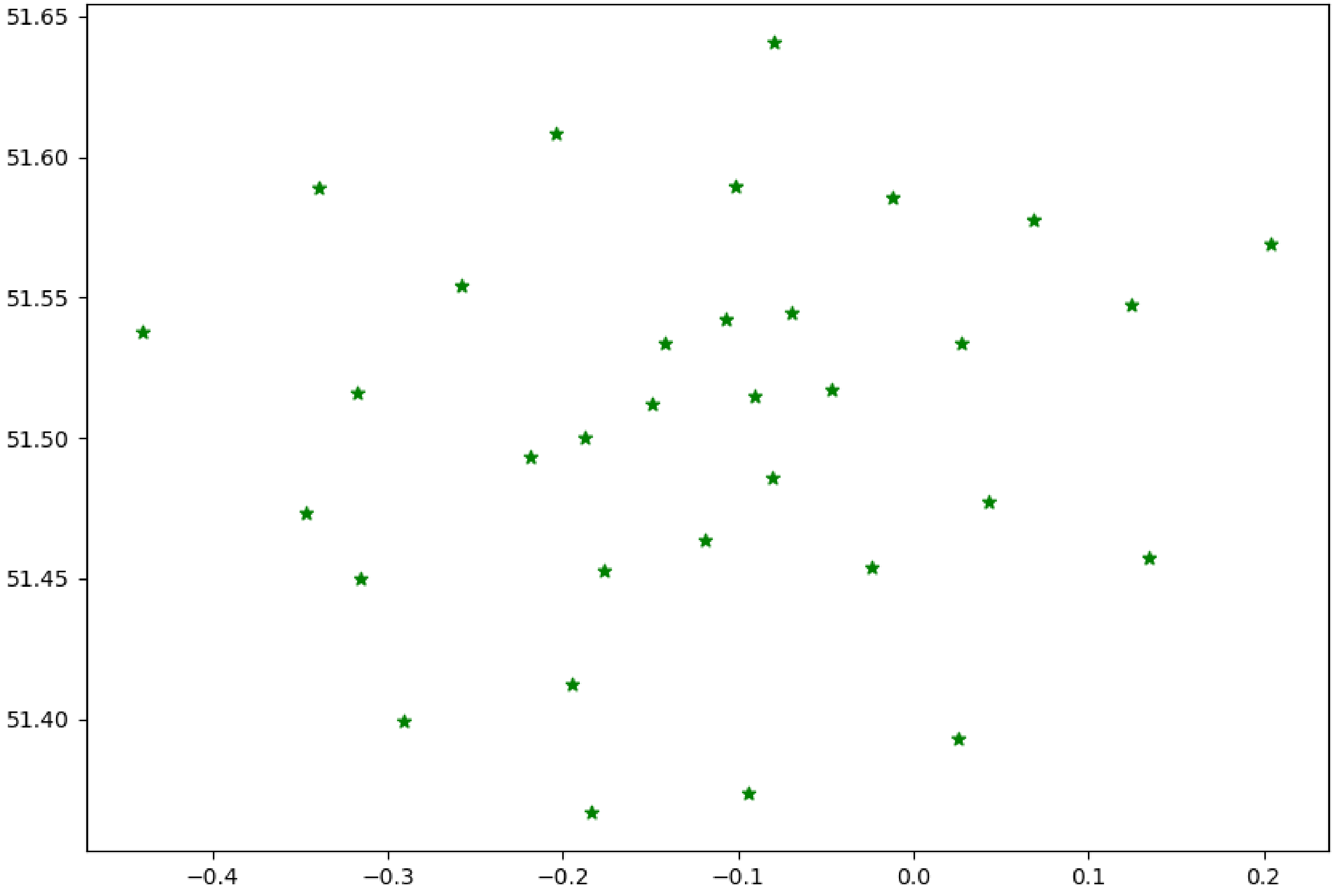
In [45]:

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



In [46]:

```
gdf_oslaua = gpd.GeoDataFrame(oslaua_average_rating, geometry= gpd.points_from_xy(oslaua
gdf_oslaua.crs = "EPSG:4326" #Adding crs information to geodataframe
gdf_oslaua.plot(marker = '*', color = 'green') #Plotting the geodataframe
plt.rcParams['figure.figsize'] = [10, 10]
```



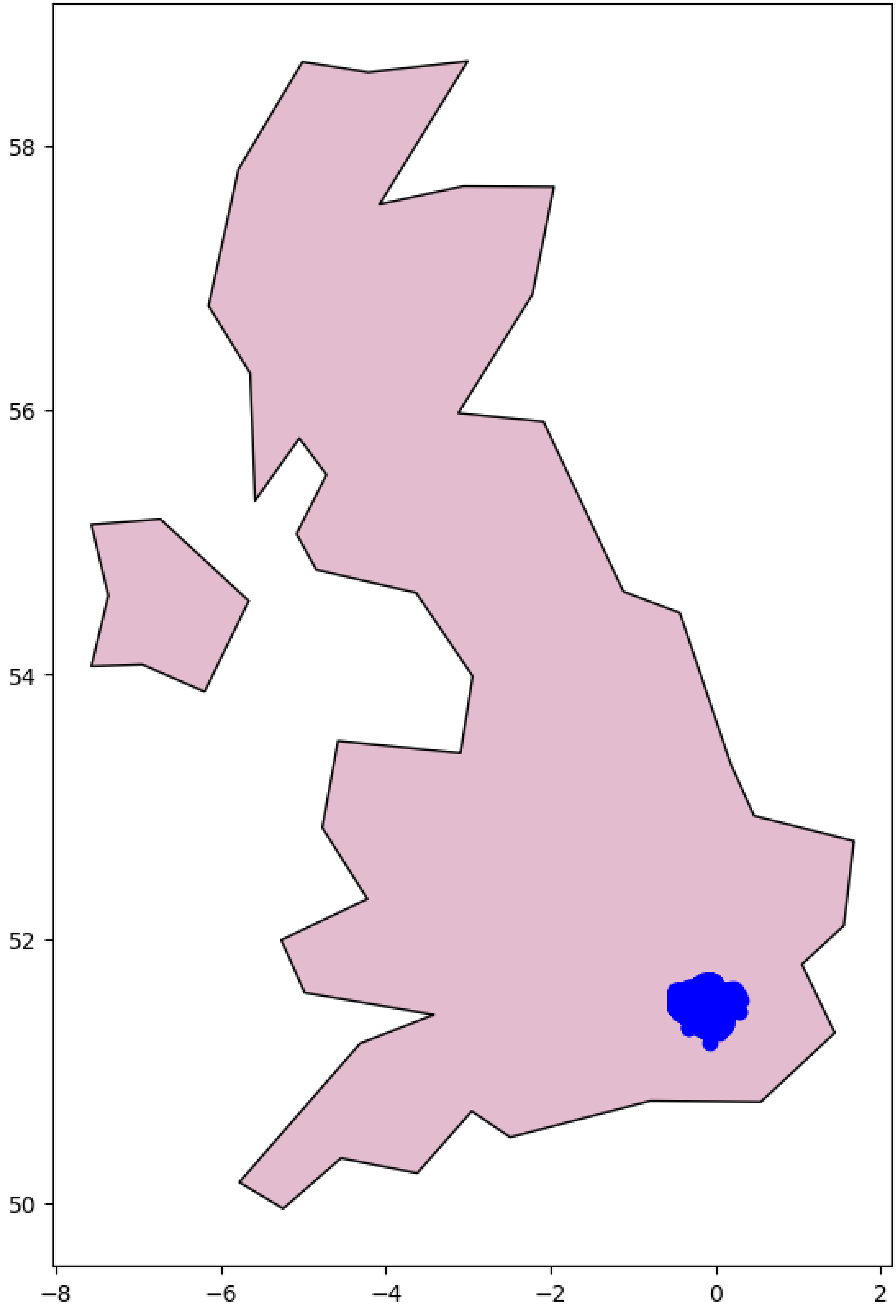
In [47]:

```
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
uk = world[world['name'] == 'United Kingdom']
```

```
ax = uk.plot(color ='#e3bccf', edgecolor = 'black')
plt.rcParams['figure.figsize'] = [10, 10]

# for x, y, label in zip(uk.geometry.representative_point().x, uk.geometry.representativ
#     ax.annotate(label, xy=(x, y))

gdf.plot(ax=ax, color = 'blue')
```

```
/var/folders/j_/yy54qztd2qv_yhxvrrgcbd1c0000gn/T/ipykernel_90477/2488275631.py:1: Future
Warning: The geopandas.dataset module is deprecated and will be removed in GeoPandas 1.
0. You can get the original 'naturalearth_lowres' data from https://www.naturalearthdat
a.com/downloads/110m-cultural-vectors/.
  world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
```

```
Out[47]: <Axes: >
```



In [48]:

```
gpd.options.use_pygeos = True  # optional, for improved performance
gpd.options._config['SHAPE_RESTORE_SHX'] = 'YES'
```

```
/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 incompatibl
e with the GEOS version PyGEOS was compiled with (3.10.4-CAPI-1.16.2). Conversions betwe
en both will be slow.
  warnings.warn(
```

In [112…

```
london_shapefile = gpd.read_file("London Population /London_Borough_Excluding_MHW.shp")
london_shapefile['geometry'].to_crs(epsg=4326)#, allow_override=True)
london_new = london_shapefile.to_crs(epsg=4326)#, allow_override=True)
# ax = gdf_oslaua.plot(alpha=0.1, color='green')
# london_new.plot(ax=ax, color = '#C6A619')
```

Plotting Population density

In [51]:

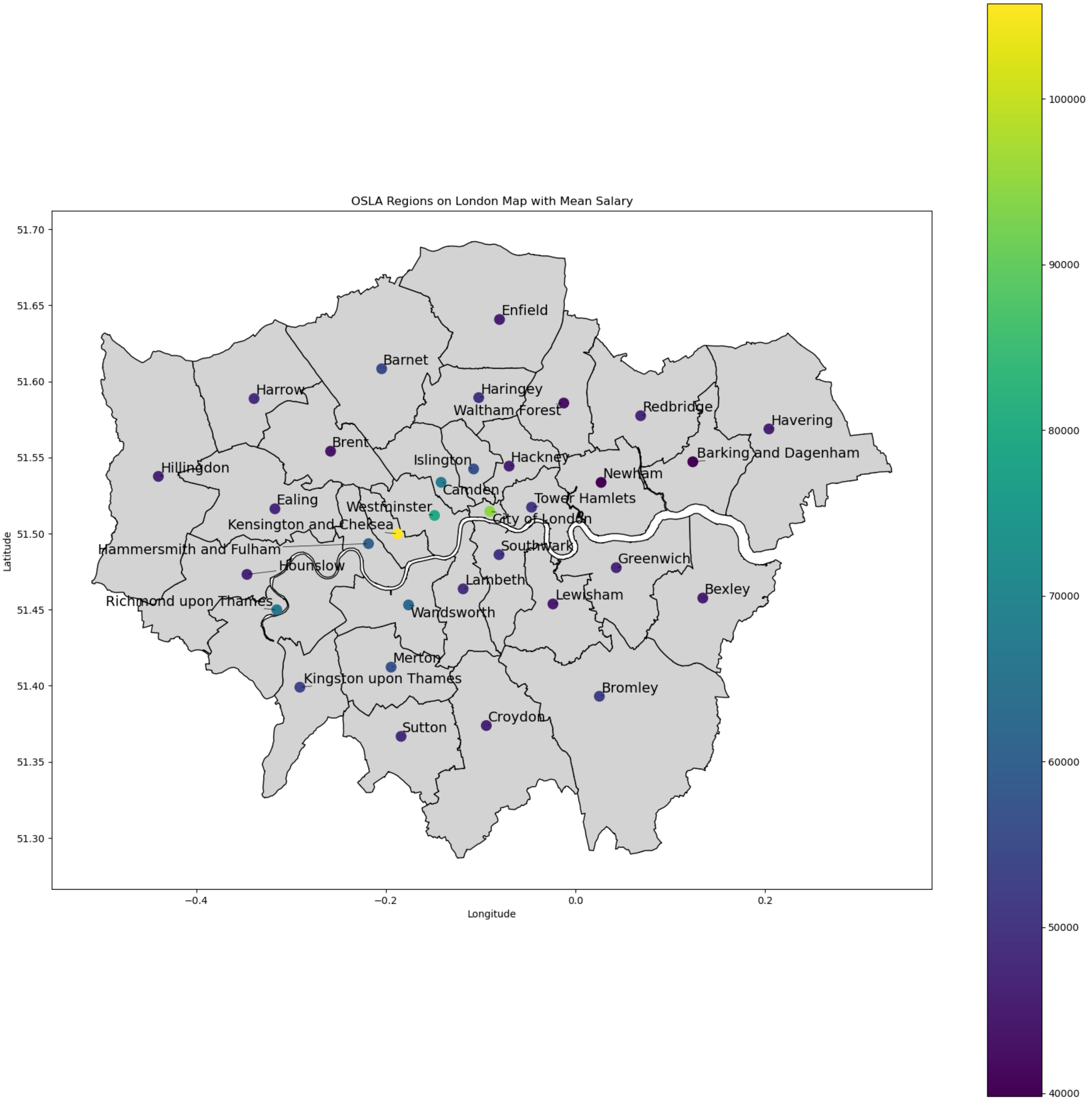
```
gdf_oslaua.columns
```

Out[51]:

```
Index(['oslaua', 'Lower tier local authorities', 'Mean Salary ',
       'Rating Value', 'CenterLatitude', 'CenterLongitude', 'geometry'],
      dtype='object')
```

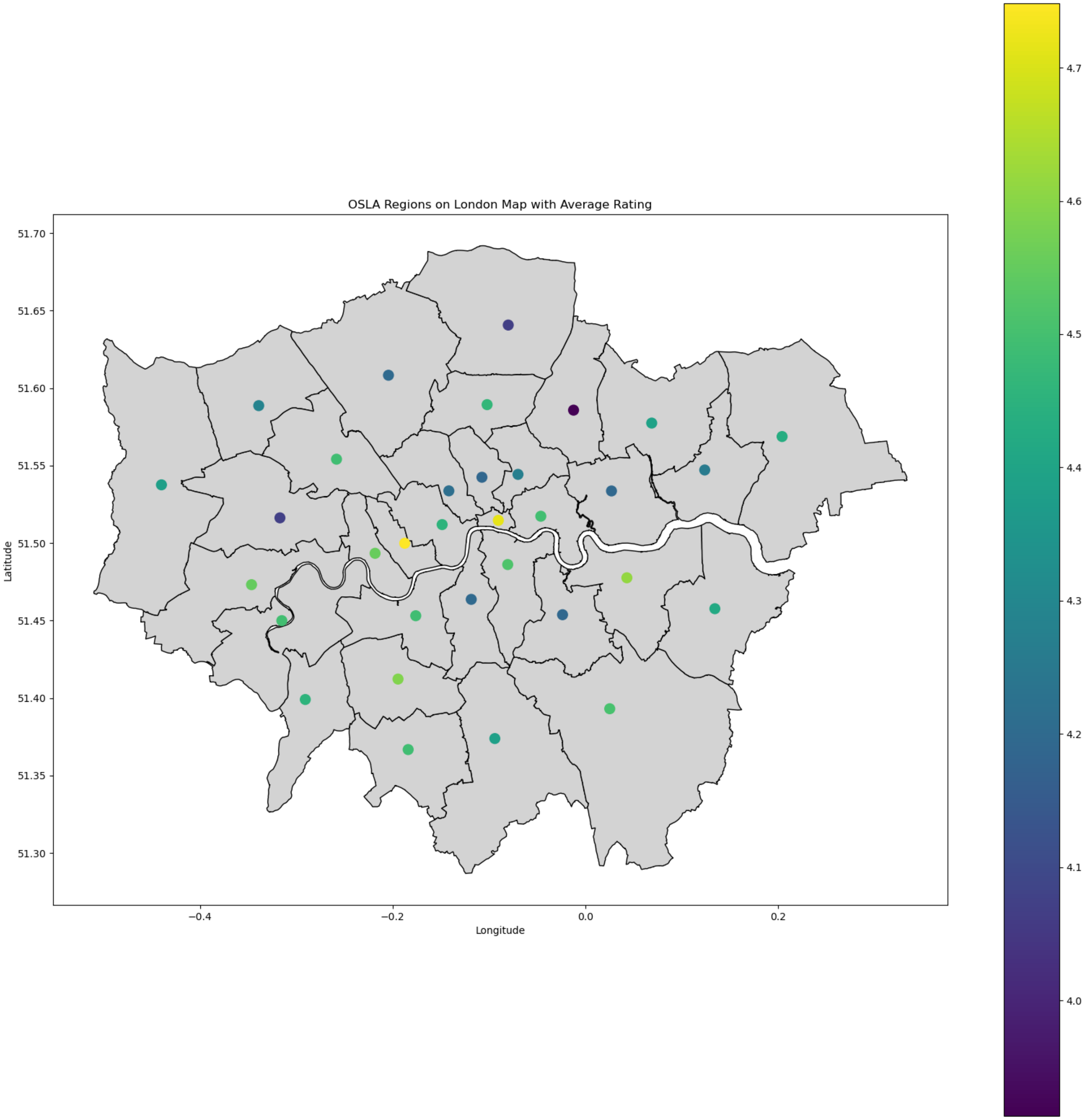
In [56]:

```
import matplotlib.pyplot as plt
from adjustText import adjust_text  # Import the adjust_text function

# Assuming you have the 'london_new' and 'gdf_oslaua' DataFrames loaded

fig, ax = plt.subplots(figsize=(20, 20))

# Plot the entire London region with a single color
london_new.plot(ax=ax, color='lightgrey', edgecolor='black')

# Plot the OSLA regions on the London map, color-coded by average rating
gdf_oslaua.plot(ax=ax, column='Mean Salary ', cmap='viridis', legend=True, markersize=10

# Set axis labels and title
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
ax.set_title('OSLA Regions on London Map with Mean Salary')

# Add labels for Lower tier local authorities and use adjust_text to avoid overlaps
texts = []
for x, y, label in zip(gdf_oslaua['CenterLongitude'], gdf_oslaua['CenterLatitude'], gdf_
    texts.append(ax.text(x, y, label, fontsize=14, color='black'))

# Use adjust_text to automatically adjust the labels to avoid overlaps
adjust_text(texts, arrowprops=dict(arrowstyle="-", color='black', lw=0.5))

# Save the plot as an image (e.g., PNG format)
plt.savefig('osla_regions_map_Mean_Salary.png')

# Show the plot (optional, you can comment this line if you only want to save the image)
plt.show()
```



In [57]:

```
fig, ax = plt.subplots(figsize=(20, 20))
london_new.plot(ax=ax, color='lightgrey', edgecolor='black')

# Plot the OSLA regions on the London map, color-coded by average rating
gdf_oslaua.plot(ax=ax, column='Rating Value', cmap='viridis', legend=True, markersize=10

# Set axis labels and title
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
ax.set_title('OSLA Regions on London Map with Average Rating')

plt.show()
```



Grouping the data to understand the popularity of cuisines for each OSLAUA region.

In [58]:

```
# Group the data by 'Lower tier local authorities' and extract the list of predicted cui
cuisine_group = analysis_data.groupby('Lower tier local authorities')['Predicted Cuisine

# Rename the columns for clarity
cuisine_group.rename(columns={'Predicted Cuisine Type': 'Cuisine Types'}, inplace=True)
```

In [59]:

```
def count_cuisine_words(cuisine_list):
    word_count = {}
    excluded_words = ["and", "new",'traditional','brunch','canteen','caterers','restaura

    for cuisine in cuisine_list:
        words = cuisine.lower().split()
        for word in words:
            if word not in excluded_words:
                word_count[word] = word_count.get(word, 0) + 1
    return word_count

# Apply the updated count_cuisine_words function to each row in the dataset
cuisine_group['Cuisine Word Count'] = cuisine_group['Cuisine Types'].apply(count_cuisine
```

In [60]:

```
cuisine_group.columns
```

Out[60]:

```
Index(['Lower tier local authorities', 'Cuisine Types', 'Cuisine Word Count'], dtype='ob
ject')
```

In [61]:

```
# Sort the word counts in descending order and get the top 10 words
top_10_words = cuisine_group['Cuisine Word Count'].apply(lambda x: sorted(x.items(), key
```

Demographics

In [64]:

```
ethnic_group = pd.read_csv('London Population /Ethinic_groups_london.csv')
ethnic_group['borough'].unique()
```

Out[64]:

```
array(['City of London', 'Barking and Dagenham', 'Barnet', 'Bexley',
       'Brent', 'Bromley', 'Camden', 'Croydon', 'Ealing', 'Enfield',
       'Greenwich', 'Hackney', 'Hammersmith and Fulham', 'Haringey',
       'Harrow', 'Havering', 'Hillingdon', 'Hounslow', 'Islington',
       'Kensington and Chelsea', 'Kingston upon Thames', 'Lambeth',
       'Lewisham', 'Merton', 'Newham', 'Redbridge',
       'Richmond upon Thames', 'Southwark', 'Sutton', 'Tower Hamlets',
       'Waltham Forest', 'Wandsworth', 'Westminster', 'Inner London',
       'Outer London', 'Greater London'], dtype=object)
```

In [65]:

```
ethnic_group.drop(columns = ['2011', '2012', '2013', '2014', '2015', '2016', '2017', '20
       '2031', '2032', '2033', '2034', '2035', '2036', '2037', '2038', '2039',
       '2040', '2041', '2042', '2043', '2044', '2045', '2046', '2047', '2048',
       '2049'], inplace = True)
```

In [66]:

```
ethnic_group['age'].unique()
```

Out[66]:

```
array(['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12',
       '13', '14', '15', '16', '17', '18', '19', '20', '21', '22', '23',
       '24', '25', '26', '27', '28', '29', '30', '31', '32', '33', '34',
       '35', '36', '37', '38', '39', '40', '41', '42', '43', '44', '45',
       '46', '47', '48', '49', '50', '51', '52', '53', '54', '55', '56',
       '57', '58', '59', '60', '61', '62', '63', '64', '65', '66', '67',
       '68', '69', '70', '71', '72', '73', '74', '75', '76', '77', '78',
       '79', '80', '81', '82', '83', '84', '85', '86', '87', '88', '89',
       '90', 'All ages'], dtype=object)
```

Determining the age groups for each location

In [67]:

```
age_groups = {
    '0-10': range(0, 11),
    '11- 17' : range(11, 18),
    '18-30': range(18, 31),
    '31-50': range(31, 51),
    '51-70': range(51, 71),
    '80 and above': range(80, 120)  # Assuming the maximum age is 119
}

# Function to map age to the corresponding age group
def map_age_to_group(age):
    if age == 'All ages':
        return 'All ages'

    try:
        age = int(age)
    except ValueError:
        return 'Unknown'

    for group, age_range in age_groups.items():
        if age in age_range:
            return group
    return 'Unknown'

# Apply the mapping function to create a new column 'age_group'
ethnic_group['age_group'] = ethnic_group['age'].map(map_age_to_group)

# Print unique age groups to check the mapping
#print(your_dataframe['age_group'].unique())

# Group by 'age_group' and 'ethnic_group', and sum the '2023' column
result = ethnic_group.groupby(['age_group', 'ethnic_group','borough'])['2023'].sum().res
```

In [68]:

```
result
```

Out[68]:

	age_group	ethnic_group	borough	2023
0	0-10	All persons	Barking and Dagenham	44725
1	0-10	All persons	Barnet	59025
2	0-10	All persons	Bexley	37184
3	0-10	All persons	Brent	51648
4	0-10	All persons	Bromley	49098
...	...	...	...	...
5467	Unknown	White Irish	Sutton	449
5468	Unknown	White Irish	Tower Hamlets	283
5469	Unknown	White Irish	Waltham Forest	433
5470	Unknown	White Irish	Wandsworth	600
5471	Unknown	White Irish	Westminster	454

5472 rows × 4 columns

In [69]:

```
result_age = result[result['age_group']!='All ages']
result_age = result_age[result_age['age_group']!='Unknown']
result_age = result_age[result_age['borough'] != 'Greater London']
result_age = result_age[result_age['borough'] != 'Inner London']
result_age = result_age[result_age['borough'] != 'Outer London' ]
result_age = result_age[result_age['borough'] != 'Barking and Dagenham' ]
result_age = result_age[result_age['borough'] != 'Bexley' ]
result_age = result_age[result_age['borough'] != 'City of London' ]
result_age = result_age[result_age['borough'] != 'Greenwich' ]
result_age = result_age[result_age['borough'] != 'Haringey']
result_age = result_age[result_age['borough'] != 'Lewisham' ]
result_age = result_age[result_age['borough'] != 'Newham' ]
result_age = result_age[result_age['borough'] != 'Richmond upon Thames']
result_age = result_age[result_age['borough'] != 'Sutton']
```

In [71]:

```
result_age_group = result_age.groupby(['borough','age_group'])['2023'].sum().reset_index
```

In [72]:

```
result_age_group.to_csv("age_group_cuisine.csv")
```

In [73]:

```
result_age_group
```

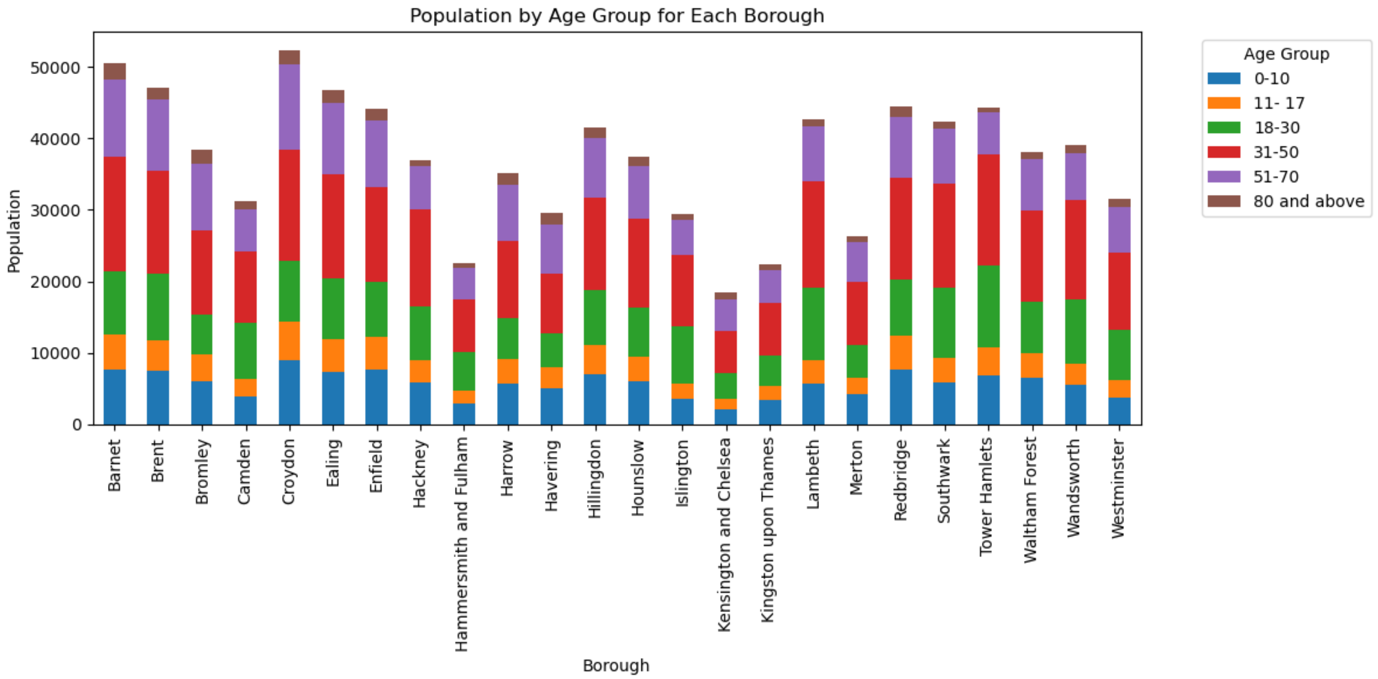
Out[73]:

	borough	age_group	2023
0	Barnet	0-10	146637
1	Barnet	11- 17	93610
2	Barnet	18-30	168108
3	Barnet	31-50	301598
4	Barnet	51-70	207477
...	...	...	...
139	Westminster	11- 17	46287
140	Westminster	18-30	133234
141	Westminster	31-50	205357
142	Westminster	51-70	121602
143	Westminster	80 and above	21221

144 rows × 3 columns

In [74]:

```
result_age['2023'] = pd.to_numeric(result_age['2023'])

# Pivot the DataFrame to have 'age_group' as columns and 'borough' as rows
pivot_df = result_age.pivot_table(index='borough', columns='age_group', values='2023', f

# Plotting the data as a stacked bar plot
pivot_df.plot(kind='bar', stacked=True, figsize=(12, 6))

# Customizing the plot
plt.xlabel('Borough')
plt.ylabel('Population')
plt.title('Population by Age Group for Each Borough')
plt.legend(title='Age Group', bbox_to_anchor=(1.05, 1), loc='upper left')

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



Determining the Ethnic groups for each region

In [75]:
```
# Group by 'borough' and 'ethnic_group', then calculate the sum of '2023' for each group
grouped_df = ethnic_group.groupby(['borough', 'ethnic_group'])['2023'].sum().reset_index
```

In [76]:
```
grouped_df
```

Out[76]:

	borough	ethnic_group	2023
0	Barking and Dagenham	All persons	461974
1	Barking and Dagenham	Arab	2352
2	Barking and Dagenham	BAME	259386
3	Barking and Dagenham	Bangladeshi	32520
4	Barking and Dagenham	Black African	85521
...	...	...	...
679	Westminster	White & Asian	9562
680	Westminster	White & Black African	4446
681	Westminster	White & Black Caribbean	2964
682	Westminster	White British	144730
683	Westminster	White Irish	8747

684 rows × 3 columns

In [77]:
```
filtered_df = grouped_df[grouped_df['ethnic_group'] != 'All persons']
filtered_df = filtered_df[filtered_df['borough'] != 'Greater London']
filtered_df = filtered_df[filtered_df['borough'] != 'Inner London']
filtered_df = filtered_df[filtered_df['borough'] != 'Outer London']
```

In [78]:
```
# Convert '2023' to numeric type
filtered_df['2023'] = pd.to_numeric(filtered_df['2023'])


top_5_ethnic_groups = filtered_df.groupby('borough').apply(lambda x: x.nlargest(5, colum
# Reset index to have a clean output
top_5_ethnic_groups.reset_index(drop=True, inplace=True)
```

In [79]:
```
top_5_ethnic_groups = top_5_ethnic_groups[top_5_ethnic_groups['ethnic_group']!='BAME']
```

In [80]:
```
top_5_ethnic_groups
```

Out[80]:

	borough	ethnic_group	2023
1	Barking and Dagenham	White British	143745
2	Barking and Dagenham	Black African	85521
3	Barking and Dagenham	Other White	56096
4	Barking and Dagenham	Bangladeshi	32520
6	Barnet	White British	316307
...	...	...	...
159	Wandsworth	Black Caribbean	23208
161	Westminster	Other White	159578

162	Westminster	White British	144730
163	Westminster	Arab	49884
164	Westminster	Other Asian	27312

132 rows × 3 columns

In [81]:

```
grouped_ethnic_groups = top_5_ethnic_groups.groupby('borough').agg({'ethnic_group': list
```

In [82]:

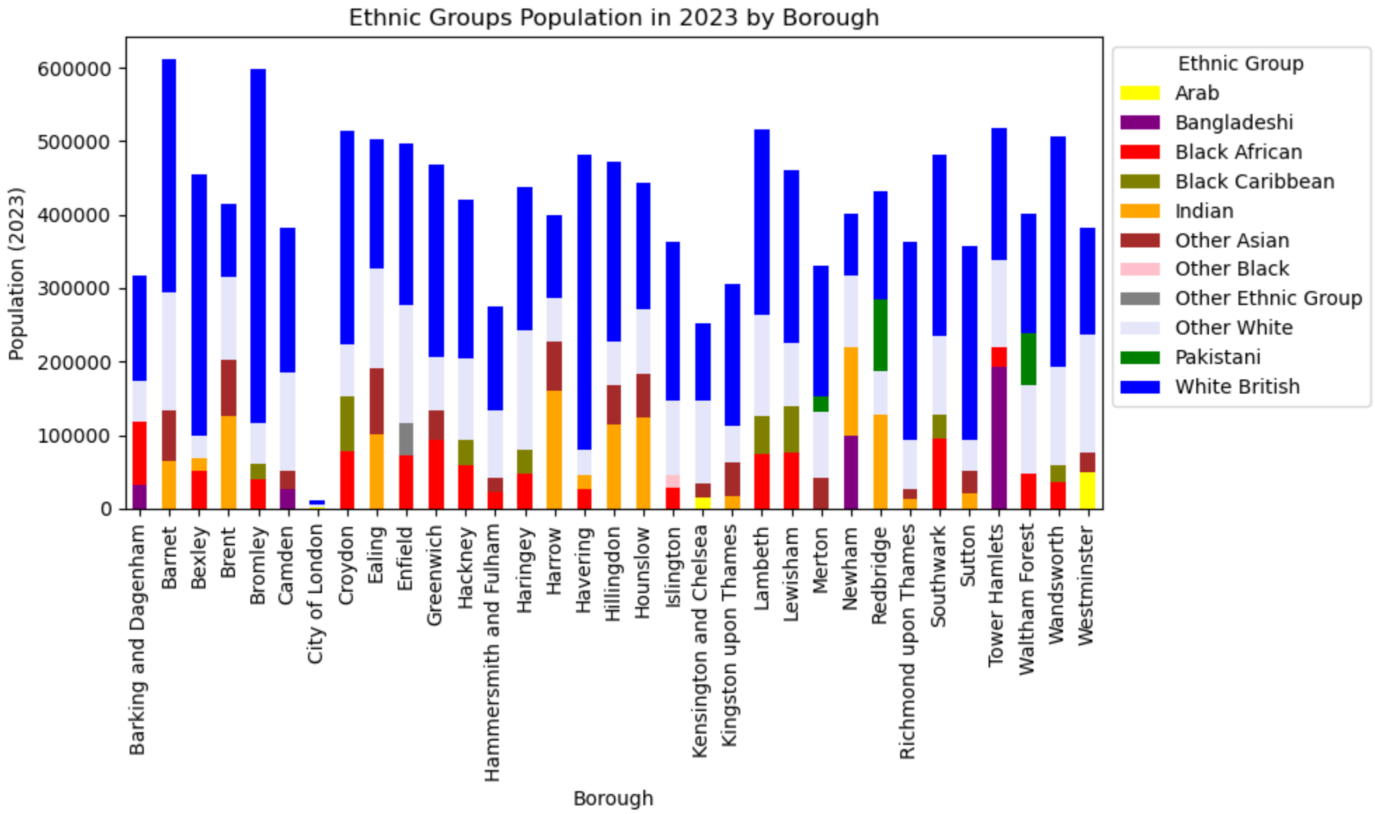
```
grouped_ethnic_groups['borough'].unique()
```

Out[82]:

```
array(['Barking and Dagenham', 'Barnet', 'Bexley', 'Brent', 'Bromley',
       'Camden', 'City of London', 'Croydon', 'Ealing', 'Enfield',
       'Greenwich', 'Hackney', 'Hammersmith and Fulham', 'Haringey',
       'Harrow', 'Havering', 'Hillingdon', 'Hounslow', 'Islington',
       'Kensington and Chelsea', 'Kingston upon Thames', 'Lambeth',
       'Lewisham', 'Merton', 'Newham', 'Redbridge',
       'Richmond upon Thames', 'Southwark', 'Sutton', 'Tower Hamlets',
       'Waltham Forest', 'Wandsworth', 'Westminster'], dtype=object)
```

In [84]:

```
import matplotlib.pyplot as plt
import pandas as pd

# Assume you have the DataFrame 'top_5_ethnic_groups' with the data

# Define custom colors for ethnic groups
colors = {
    'Arab':'yellow',
    'Other Asian': 'brown',
    'Other Black' : 'pink',
    'Other Ethnic Group': 'grey',
    'Other White': 'lavender',
    'Black Caribbean' : 'olive',
    'Pakistani' : 'light blue',
    'White British': 'blue',
    'Indian': 'orange',
    'Pakistani': 'green',
    'Black African': 'red',
    'Bangladeshi': 'purple'
}

# Pivot the DataFrame
pivot_df = top_5_ethnic_groups.pivot(index='borough', columns='ethnic_group', values='20

# Plot the stacked bar graph using custom colors
ax = pivot_df.plot(kind='bar', stacked=True, figsize=(10, 6), color=[colors[col] for col
plt.xlabel('Borough')
plt.ylabel('Population (2023)')
plt.title('Ethnic Groups Population in 2023 by Borough')
plt.legend(title='Ethnic Group', bbox_to_anchor=(1, 1))
plt.tight_layout()
plt.show()
```

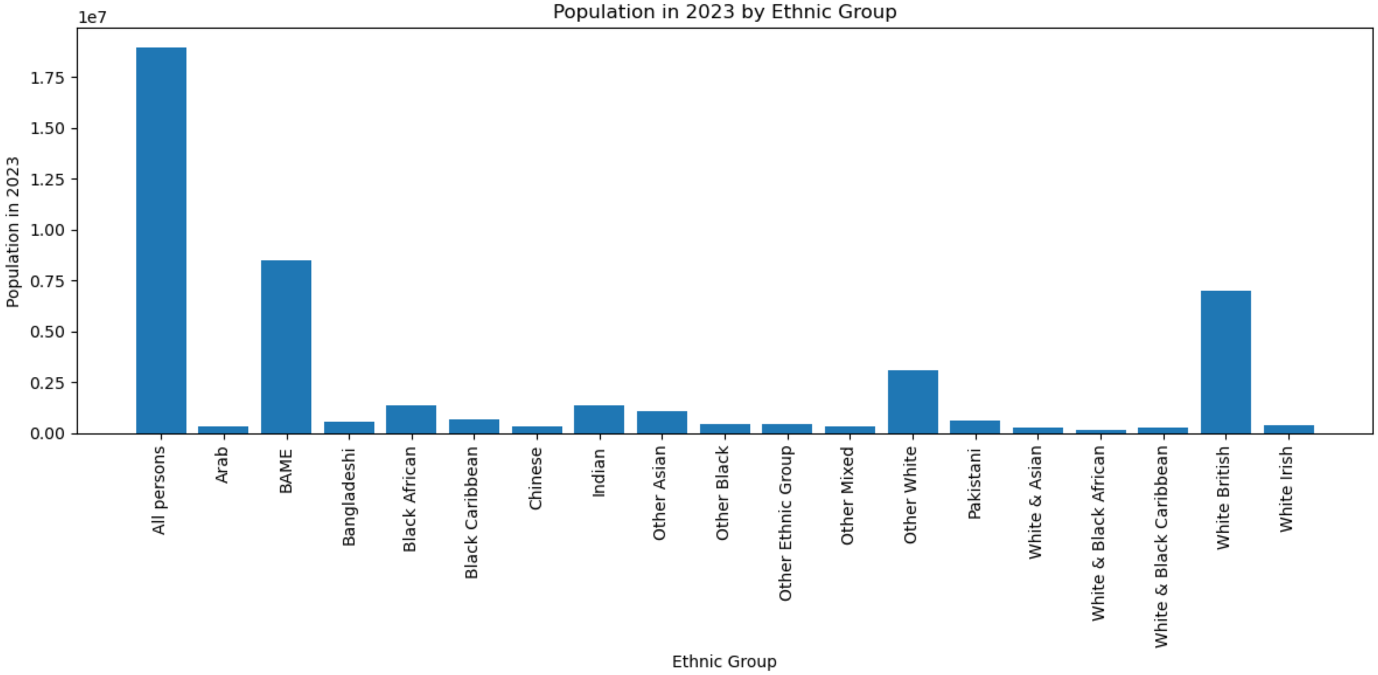


In [89]:

```
# Calculate the total population in 2023
total_population = grouped_df['2023'].sum()

# Calculate the percentage for each ethnic group
grouped_df['Percentage'] = (grouped_df['2023'] / total_population) * 100
```

In [92]:

```
plt.figure(figsize=(12, 6))
plt.bar(grouped_df['ethnic_group'], grouped_df['2023'])
plt.xticks(rotation=90)
plt.xlabel('Ethnic Group')
plt.ylabel('Population in 2023')
plt.title('Population in 2023 by Ethnic Group')
plt.tight_layout()
plt.show()
```



Determining the top 5 ethnic groups for each region

In [93]:

```
user_data = pd.read_csv('London Population /london_postcodes-ons-postcodes-directory-feb
user_data = user_data[['pcds','oslaua','oa11','lat','long']]
user_data = user_data.dropna(subset = ['lat','long'])
```

In [94]:

```
london_population = pd.read_csv("London Population /Local Authority Code wise data.csv")
london_population = london_population.drop(['Number of Employees','Number of Unemployees
```

In [95]:

```
user_data_pop = pd.merge(user_data,london_population, left_on = 'oslaua', right_on = 'Lo
user_data = user_data_pop.dropna()
```

In [97]:

```
candidate_location = pd.read_csv("London Population /Candidate_location.csv")
```

In [98]:

```
matched_locations = []
for idx, row in candidate_location.iterrows():
    match = user_data[(user_data['lat_x'] == row['Latitude']) & (user_data['long_x'] ==
    if not match.empty:
        matched_locations.append(match.values[0])
    else:
        matched_locations.append(None)

candidate_location['oslaua'] = matched_locations
```

In [101…

```
cuisine_type = pd.read_csv("London Population /Table for cuisine.csv")
cuisine_type['Borough'].unique()
```

Out[101]:

```
array(['Camden', 'Southwark', 'Westminster', 'Redbridge', 'Harrow',
       'Croydon', 'Kingston upon Thames', 'Enfield', 'Bromley',
       'Islington', 'Waltham Forest', 'Hammersmith and Fulham',
       'Tower Hamlets ', 'Merton', 'Hackney', 'Ealing',
       'Kensington and Chelsea', 'Barnet', 'Brent', 'Lambeth',
       'Hillingdon', 'Hounslow', 'Havering', 'Wandsworth'], dtype=object)
```

In [102…

```
# Assuming you have a DataFrame called 'cuisine_type' with a column 'Borough'
cuisine_type['Borough'] = cuisine_type['Borough'].replace('Tower Hamlets ', 'Tower Hamle
```

In [104…

```
# Group by 'Borough' and find top 5 ethnic groups for each borough
top_5_ethnic_groups = top_5_ethnic_groups.groupby('borough').apply(lambda x: x.nlargest(

# Reset the index to remove the multi-index
top_5_ethnic_groups.reset_index(drop=True, inplace=True)

# Merge the top 5 ethnic groups with dataset 2 based on the 'Borough' column
result = pd.merge(cuisine_type, top_5_ethnic_groups, left_on='Borough', right_on='boroug

# Drop unnecessary columns and rename columns
result.drop(['borough', 'ethnic_group'], axis=1, inplace=True)
#result.rename(columns={'ethnic_group_x': 'Ethnic Group', '2023': 'Population'}, inplace
```

In [105…

```
# Merge the top 5 ethnic groups with dataset 2 based on the 'Borough' column
result = pd.merge(cuisine_type, grouped_ethnic_groups, left_on='Borough', right_on='boro
```

In [107…

```
sorted_dataframe = result.sort_values(by='Index')
```

In [108…

```
sorted_dataframe.drop(columns = ['borough'])
```

Out[108]:

	Unnamed: 0	Index	Borough	Top 5 Cuisine Types	ethnic_group
0	0	1	Camden	american, indian, asian, bar, mediterranean	[White British, Other White, Other Asian, Bang...

1	1	2	Camden	tea, coffee, caterers, asian, bar	[White British, Other White, Other Asian, Bang...
5	2	3	Southwark	cafe, bar, european, chicken, sea	[White British, Other White, Black African, Bl...
6	3	4	Westminster	italian, bar, mediterranean, european, chicken	[Other White, White British, Arab, Other Asian]
2	4	5	Camden	chinese, indian, asian, bar, mediterranean	[White British, Other White, Other Asian, Bang...
9	5	6	Redbridge	bars, chinese, indian, bar, mediterranean	[White British, Indian, Pakistani, Other White]
10	6	7	Harrow	breakfast, chinese, caterers, asian, bar	[Indian, White British, Other Asian, Other White]
11	7	8	Croydon	cafe, pizza, caterers, indian, asian	[White British, Black African, Black Caribbean...
13	8	9	Kingston upon Thames	tea, coffee, caterers, mediterranean, european	[White British, Other White, Other Asian, Indian]
3	9	10	Camden	indian, asian, bar, mediterranean, european	[White British, Other White, Other Asian, Bang...
14	10	11	Enfield	cafe, italian, fast, asian, bar	[White British, Other White, Black African, Ot...
15	11	12	Bromley	breakfast, coffee, italian, indian, asian	[White British, Other White, Black African, Bl...
16	12	13	Islington	chinese, asian, bar, mediterranean, european	[White British, Other White, Black African, Ot...
17	13	14	Waltham Forest	chinese, caterers, italian, asian, bar	[White British, Other White, Pakistani, Black ...
7	14	15	Westminster	american, caterers, indian, fast, food	[Other White, White British, Arab, Other Asian]
18	15	16	Hammersmith and Fulham	bars, chinese, caterers, italian, asian	[White British, Other White, Black African, Ot...
20	16	17	Tower Hamlets	cafe, breakfast, indian, bar, mediterranean	[Bangladeshi, White British, Other White, Blac...
22	17	18	Merton	bar, mediterranean, european, food, pub	[White British, Other White, Other Asian, Paki...
23	18	19	Hackney	bars, chinese, caterers, bar, pub	[White British, Other White, Black African, Bl...
24	19	20	Ealing	caterers, indian, bar, mediterranean, european	[White British, Other White, Indian, Other Asian]
8	20	21	Westminster	tea, coffee, indian, fast, asian	[Other White, White British, Arab, Other Asian]
28	21	22	Kensington and Chelsea	italian, indian, bar, mediterranean, european	[Other White, White British, Other Asian, Arab]
29	22	23	Barnet	cafe, tea, coffee, caterers, indian	[White British, Other White, Other Asian, Indian]
4	23	24	Camden	tea, coffee, caterers, indian, fast	[White British, Other White, Other Asian, Bang...
25	24	25	Ealing	caterers, bar, mediterranean, european, food	[White British, Other White, Indian, Other Asian]
21	25	26	Tower Hamlets	cafe, italian, bar, food, pub	[Bangladeshi, White British, Other White, Blac...

32	26	27	Brent	chinese, asian, mediterranean, pub, bakeries	[Indian, Other White, White British, Other Asian]
30	27	28	Barnet	cafe, bars, tea, coffee, bar	[White British, Other White, Other Asian, Indian]
19	28	29	Hammersmith and Fulham	breakfast, caterers, italian, indian, bar	[White British, Other White, Black African, Ot...
33	29	30	Brent	tea, coffee, asian, bar, pub	[Indian, Other White, White British, Other Asian]
34	30	31	Lambeth	mediterranean, european, food, pub, chicken	[White British, Other White, Black African, Bl...
35	31	32	Hillingdon	bars, indian, fast, bar, mediterranean	[White British, Indian, Other White, Other Asian]
37	32	33	Hounslow	pan, chinese, caterers, bar, mediterranean	[White British, Indian, Other White, Other Asian]
36	33	34	Hillingdon	italian, indian, asian, bar, mediterranean	[White British, Indian, Other White, Other Asian]
38	34	35	Havering	bars, bar, mediterranean, european, chicken	[White British, Other White, Black African, In...
31	35	36	Barnet	mediterranean, pub, sea, bakeries, japanese	[White British, Other White, Other Asian, Indian]
26	36	37	Ealing	cafe, caterers, italian, bar, mediterranean	[White British, Other White, Indian, Other Asian]
39	37	38	Wandsworth	caterers, indian, bar, mediterranean, pub	[White British, Other White, Black African, Bl...
27	38	39	Ealing	indian, fast, bar, mediterranean, european	[White British, Other White, Indian, Other Asian]
12	39	40	Croydon	fast, bar, mediterranean, european, food	[White British, Black African, Black Caribbean...

In [109…

```
sorted_dataframe.to_csv("Cuisine_ethnical.csv")
```

In [ ]:

```

```