Coursework Coversheet

School of Geography. FACULTY OF ENVIRONMENT

GEOG5403 Creative Coding for Urban Problems

UNIVERSITY OF LEEDS

Student ID	201722343, 201733915, 201798764, 201782291, 201788996	Mark	
Group Number	Six	Less deduction (state reason)	
Assignment title	Understanding the food environments in Leeds		
Marker	Dr. Jiaqi Ge	Final Mark	

Justification of mark (using specific text from criteria):

To improve your work for next time:
1.
2.
3.

Additional comments:

Understanding the food environments in Leeds

Group 6

Jinzhe Liu, Haolin Kang, Yuchi Lai, Xiaoyan Xin, Zhihao Zhang

Background on the topic

What is Food Environment?

The food environment is a region shaped by physical and social factors that influence people‘s access to, affordability, and adequacy of food. These factors include the number and type of supermarkets, the economic conditions of the community, the population culture, and government policies. The food environment has a significant impact on the health and eating habits of residents. (Rideout, Mah, and Minaker, 2015)

Physical factors

- Availability: Access to healthy and unhealthy options in stores, restaurants, etc.
- Accessibility: Distance and ease of reaching food stores and markets.
- Walkability: Neighborhood design's impact on walking/biking access to food.

Social factors

- Government Policies: Food labeling, marketing, pricing regulations influencing accessibility.
- Income & Status: Community's economic makeup and its impact on food choices.
- Culture & Values: Beliefs and traditions surrounding food consumption.
- Nutrition Resources: Availability of education and support for healthy eating.

Approaches to the challenge

- Understand the health statue of residents in Leeds

  Adult Obesity Rate Dataset which recorded in 2023/01

- Understand the IMD statue of residents in Leeds

- **Analysing the food environment in Leeds, through…**

  Method 1:
  Food and Beverages advertisements

  Method 2:
  Distribution of Healthy and Unhealthy Food Accessibility in Leeds

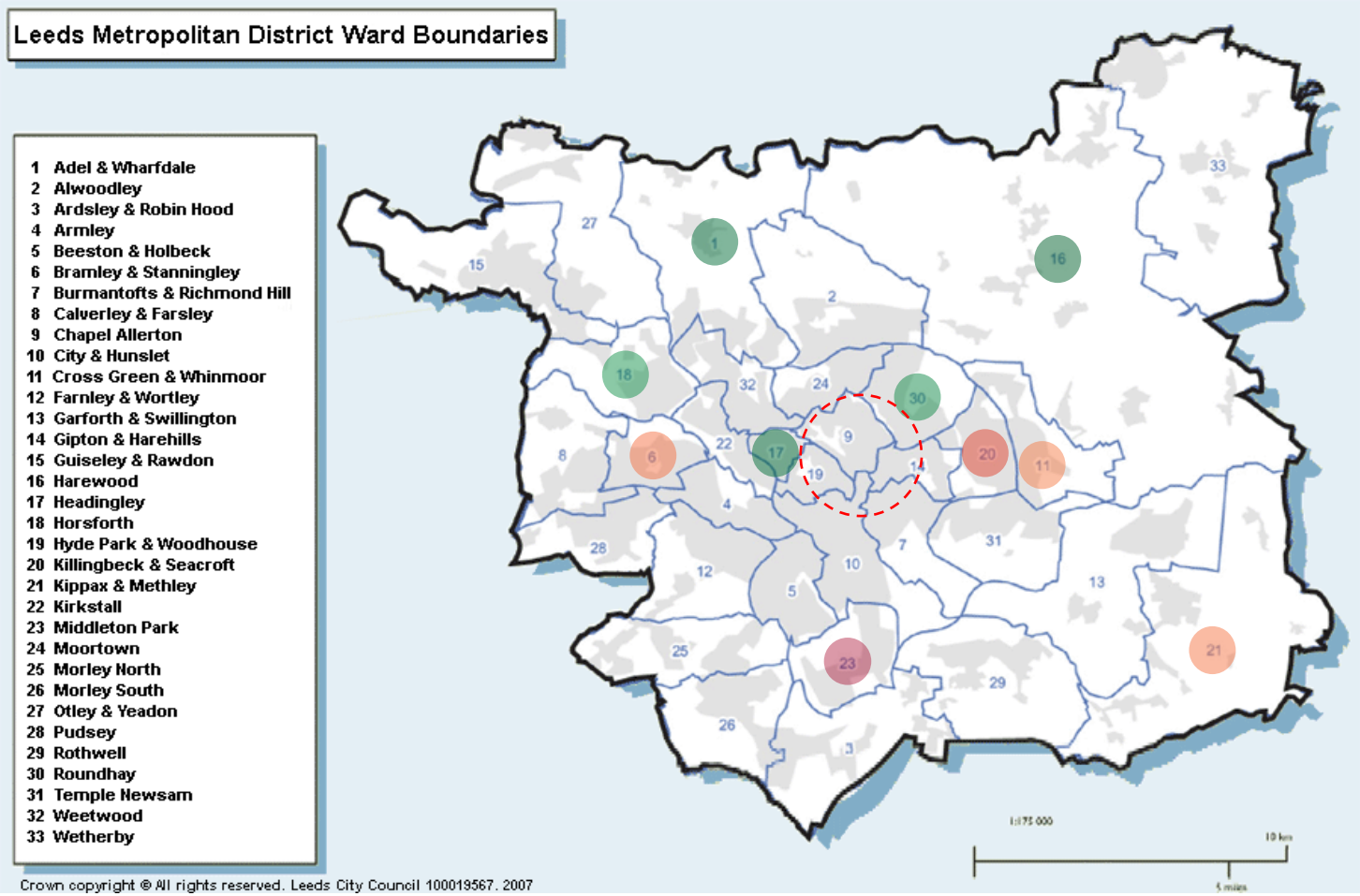
Data used

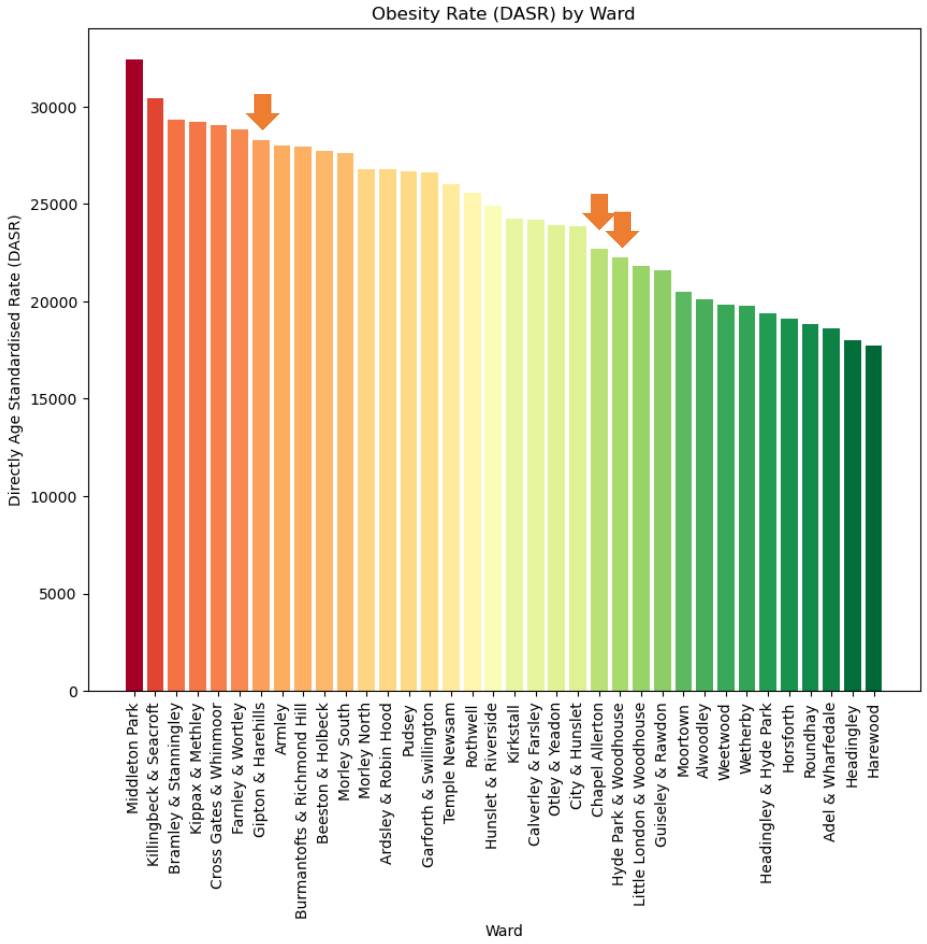
Data which is provided in class

1. **MAAPAdvertData2023.xlsx**
   We utilize the longitude and latitude information from this dataset to determine the locations of advertising assets.
2. **MAAP_advertlocations.csv**
   This dataset, which includes LSOA and IMD values, helps us assess the specific health rates in Leeds areas. A lower IMD value indicates a the area was most deprived.
3. **MAAP_AreaAttributes2023.xlsx**
   By using the LSOA and ADid data from this file, we connect datasets 2 and 3 to gather information on advertisements.

4. **Adult Obesity rates.xlsx**
   This dataset is used to analyze the adult obesity rates among residents. The data is sourced from the Leeds City Council.
5. **Restaurant data from OpenStreetMap**
   We utilize this data, which includes postcode, and geometry, to determine the locations of restaurants, shops, markets, etc. Using OpenStreetMap labels, we categorize these datas into healthy and unhealthy shops.

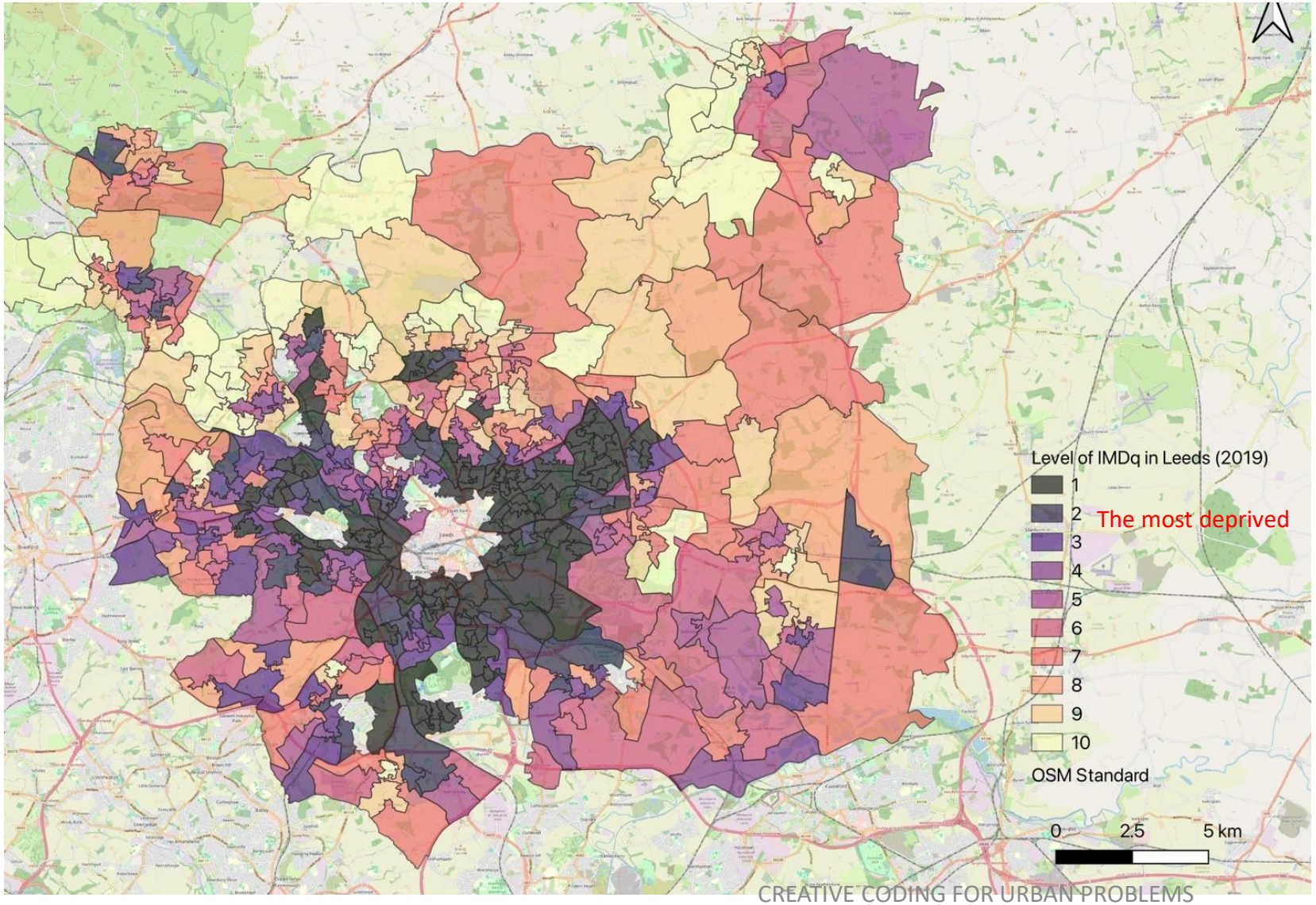
Distribution of Adult Obesity Rates in Leeds





*Directly Age Standardised Rates (DASR) per 100,000. Age standardised rates compensate for differing age structures by weighting them to meet the European Standard Population (2013). Rates can then be compared for different areas, or even across area types.

Distribution of IMD Score in Leeds



The plot was processed by QGIS.

According to Fraser and Edwards (2010), 27.1% of children are overweight or obese, with a positive correlation between fast food outlet density and childhood obesity rates, but no significant relationship with the distance to the nearest fast food outlet.

Methods 1

Understand the food environment in Leeds, through…

Food and Beverages advertisements

How many advertisements promote unhealthy food?

How do we define unhealthy food?

We use the Nutrient Profile Model (NPM) Score:

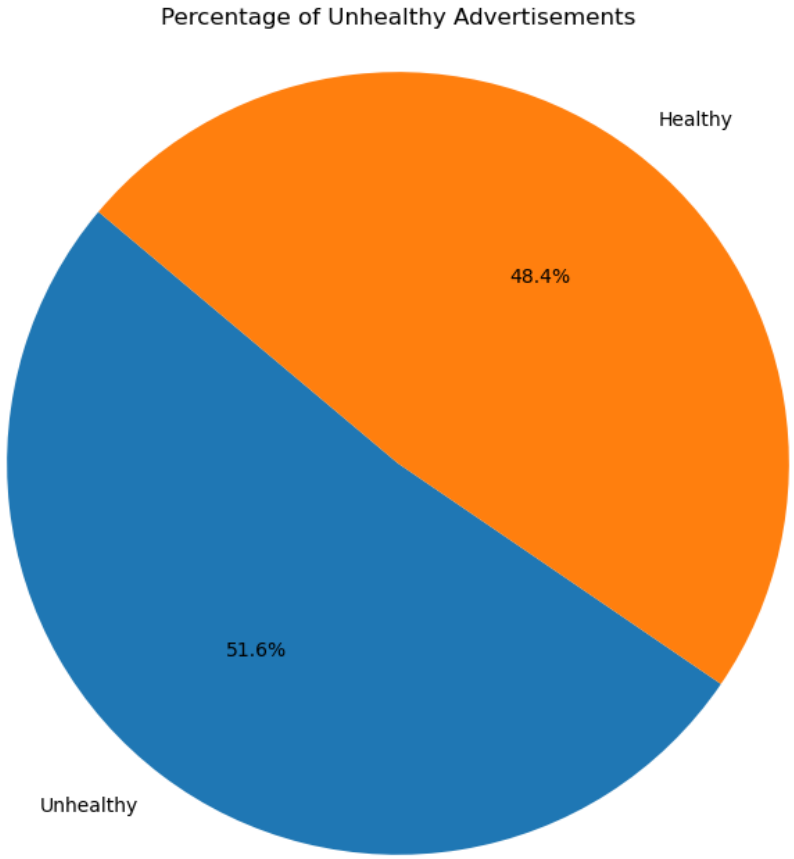
Beverages

NPM score ≥ 1

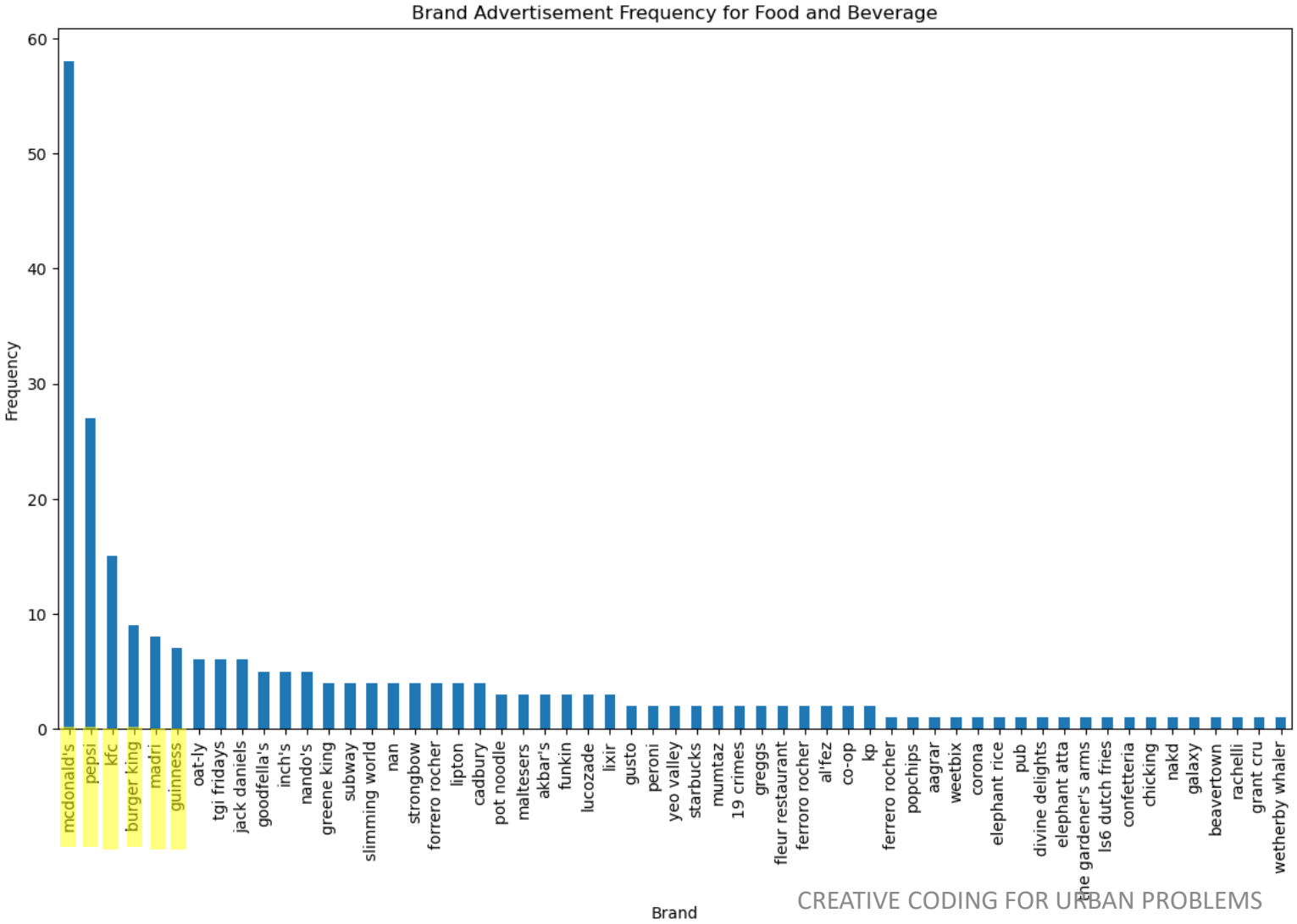
Foods

NPM score ≥ 4

Source: https://npmcalculator.cdrc.ac.uk/



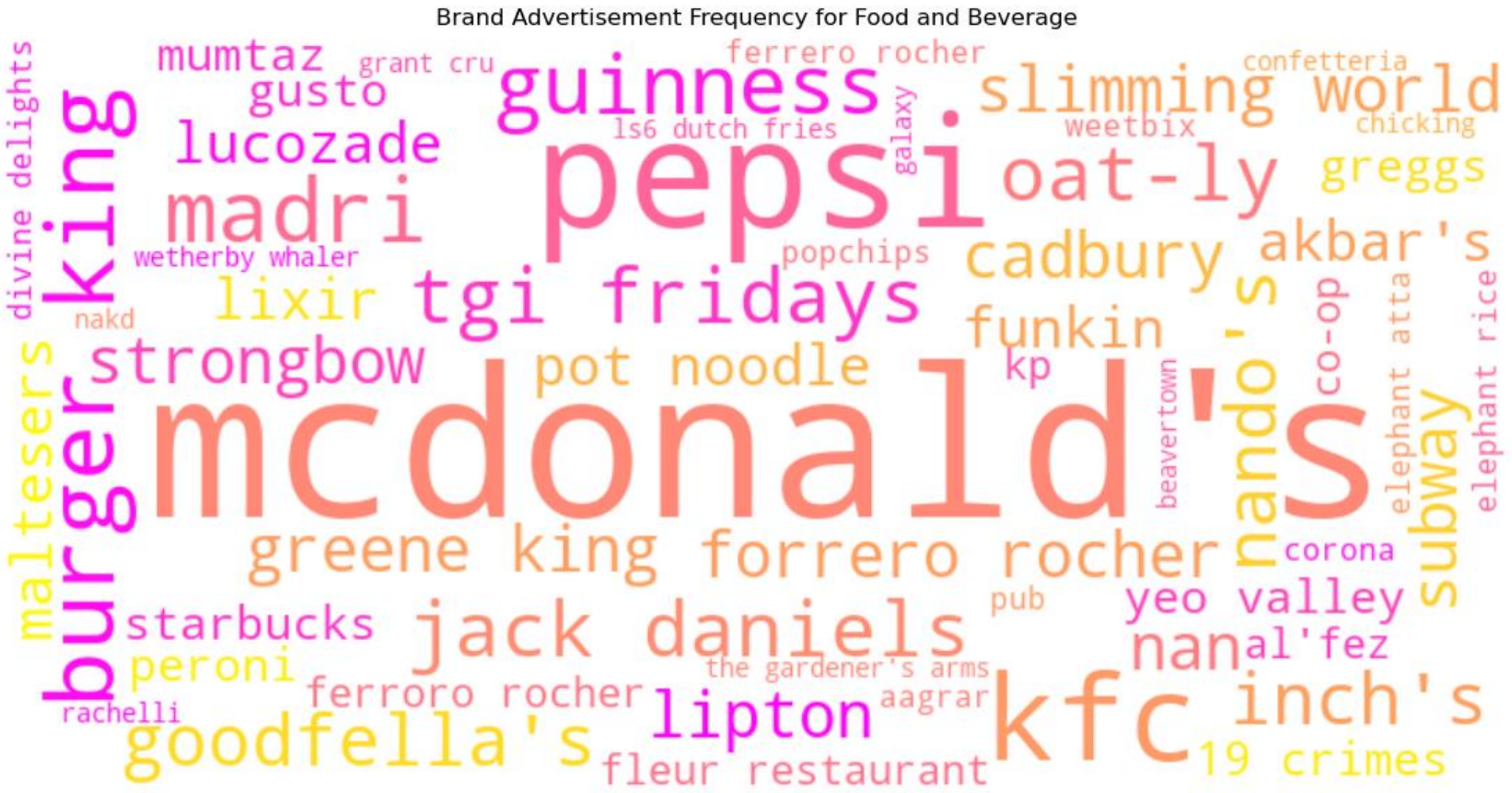
Which brands were commonly featured in the advertisements?



From May 17, 2023, to June 20, 2023, the number of advertisements for each brand.

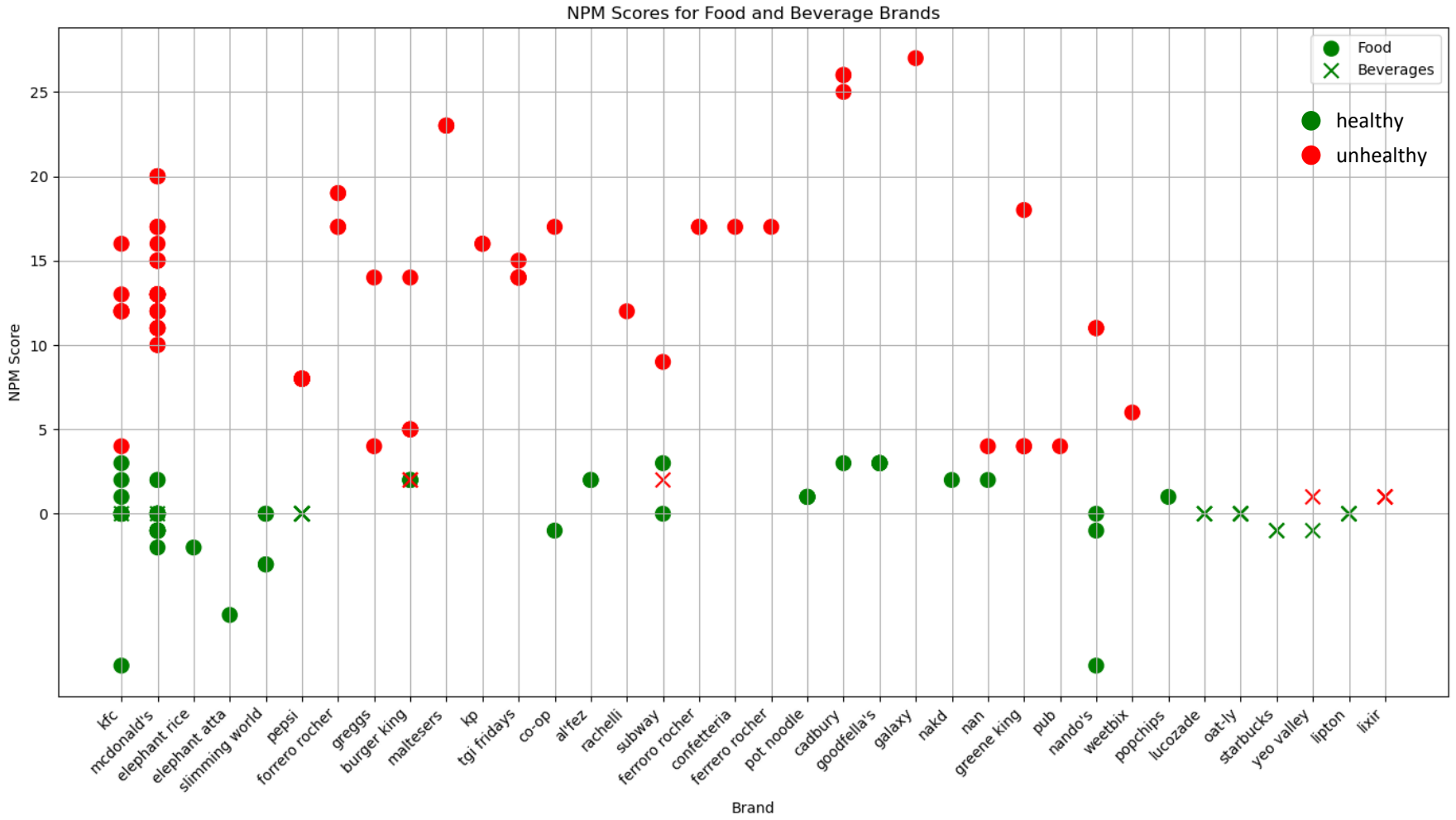
Brand	Number of Ads
McDonald's	57
Pepsi	27
KFC	14
Burger King	8
Mandri	7
Guinness	6

Which brands were commonly featured in the advertisements?



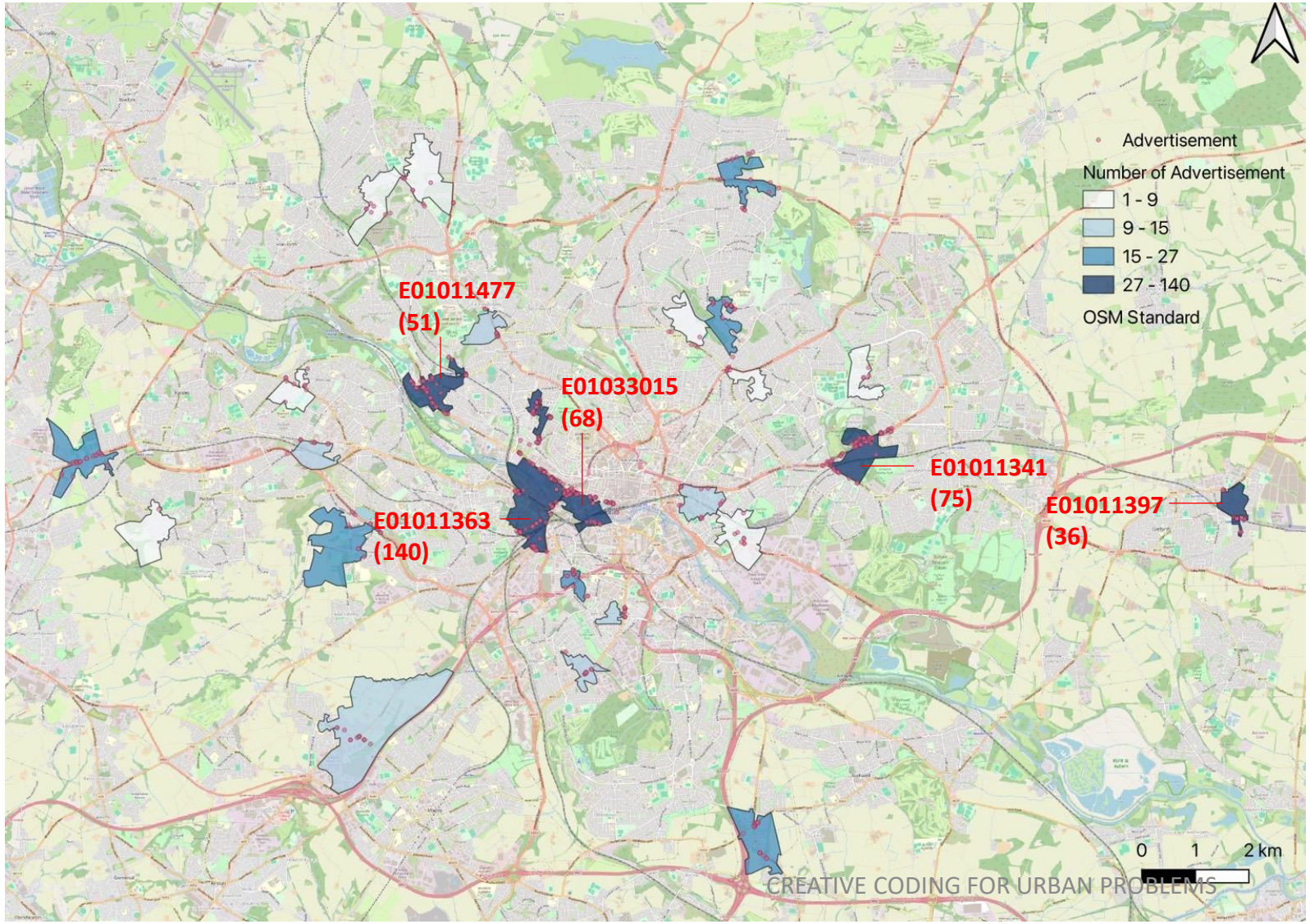
The brands McDonald's, Pepsi, KFC, Burger King, and Madri stand out for their high frequency of appearance. These brands are known for their products that are rich in fats, sugars, and additives—components often linked with unhealthy diets.

Which brand advertises the most unhealthy food?



Some product doesn't contain NPM Score.

Distribution of Advertisement Assets



Area has the most advertisement assets are…

LSOA21CD	Number of Assets
E01011363	140
E01011341	75
E01033015	68
E01011477	51
E01011397	36

The plot was processed by QGIS.

Solutions

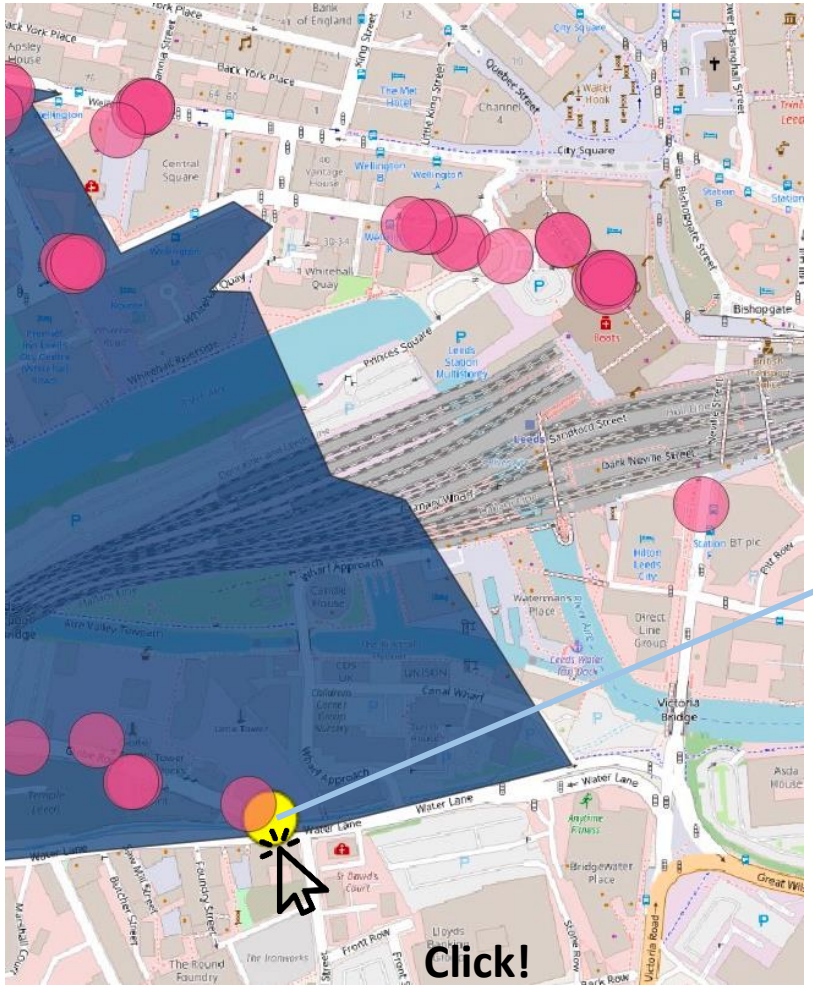
Over 50% of food and beverage advertisements are for unhealthy food. This suggests that individuals active in areas with a higher density of advertising assets are more likely to be exposed to unhealthy food options, such as the city centre in Leeds(E01011363, E01033015).

What Actions Can Different Stakeholders Take?

- **Brand Companies** – The balance between healthy and unhealthy food advertisements should be regulated each month.

- **Asset Owners** (Private and Government) – The balance of healthy and unhealthy food advertisements displayed on their assets should be regulated to ensure that both types of food are promoted, allowing the public to be exposed to both options. Additionally, the advertising fees could be adjusted based on whether the product is healthy. Asset owners can encourage advertisements for healthy food by reducing fees, while increasing fees for promoting unhealthy food.

Future Idea: A Dashboard for Asset Owners



The plot was processed by QGIS.

Dashboard Information

Current Information

ADid	HR_GWWW_35_01
ASSETid	HR_GWWW_32
LSOA	E01033015
Codedate	2023/5/30
IMGid	IMG_5513.JPG
IMGdate	2023/5/25
Adtype	6
Adsize	2
Adprodtype	1
Adprod	McNugget
Adbrand	McDonald's
NPMscore	5

Future Information

UnhAd_asset(%)	Percentage of Unhealthy Food Advertisements on the Assets
UnhAd_areas(%)	Percentage of Unhealthy Food Advertisements in the area
Addu	Ad Display Duration
Adeff	Ad Effectiveness

These columns can assist owners in managing their advertising displays.

Limitations

1. **Data Collection Time Frame**
   The dataset MAAPAdvertData2023.xlsx spans only from May 17, 2023, to June 20, 2023, offering limited insights due to the short duration.

2. **Ad Effectiveness Uncertainty**
   The impact and effectiveness of the advertisements remain unmeasured.

3. **Geographical Concentration of Ad Assets**
   The distribution of advertising assets is confined to specific areas within Leeds.

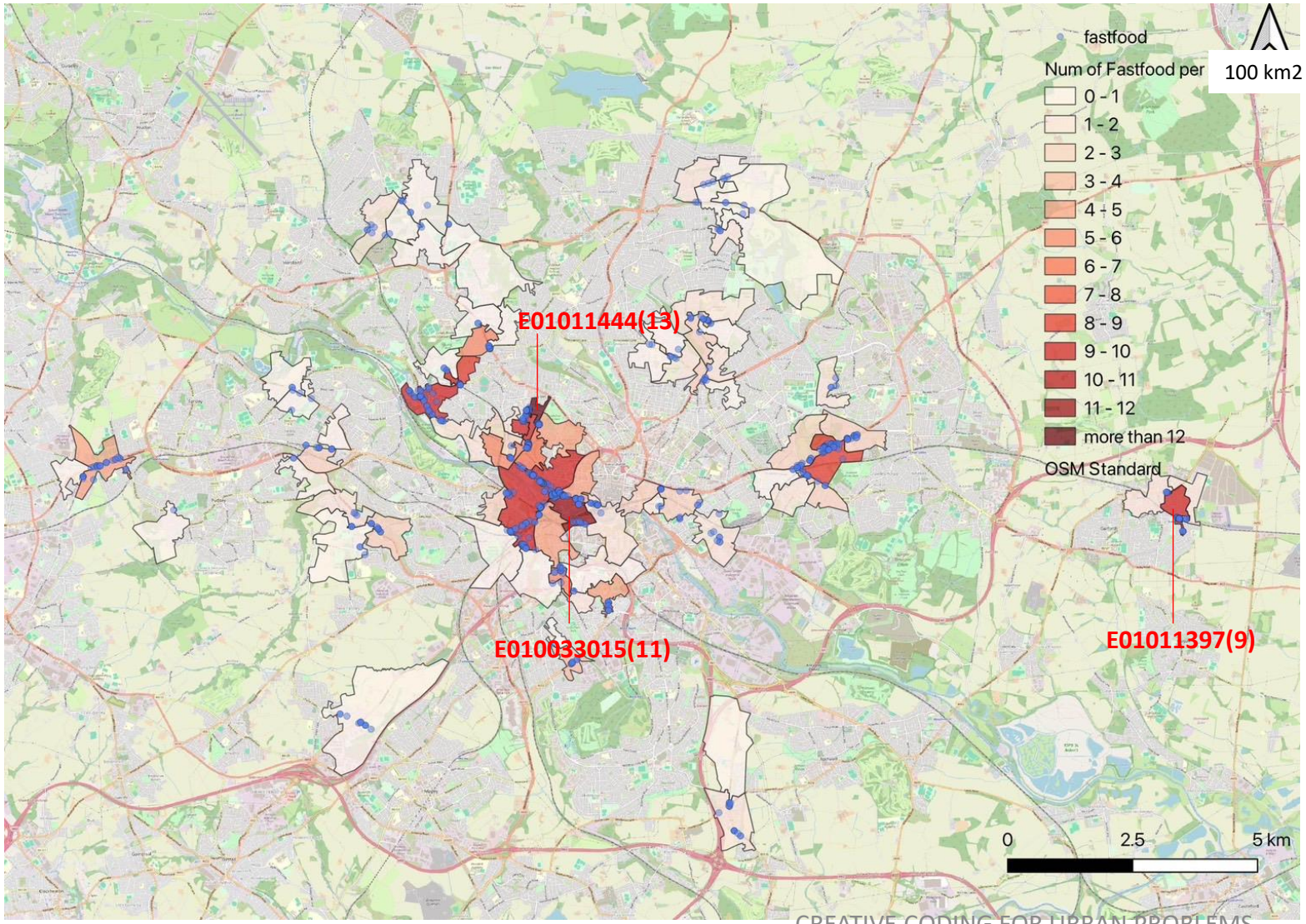
4. **Exclusion of Social Media Advertising**
   Current analysis only accounts for physical advertising facilities, overlooking the influence of social media advertisements.

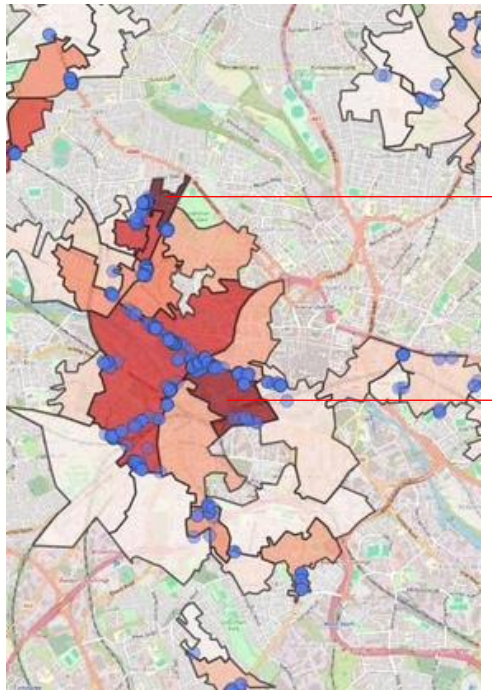
Methods 2

Understand the food environment in Leeds, through…

Distribution of Healthy and Unhealthy Food Accessibility in Leeds

Distribution of fast food in Leeds

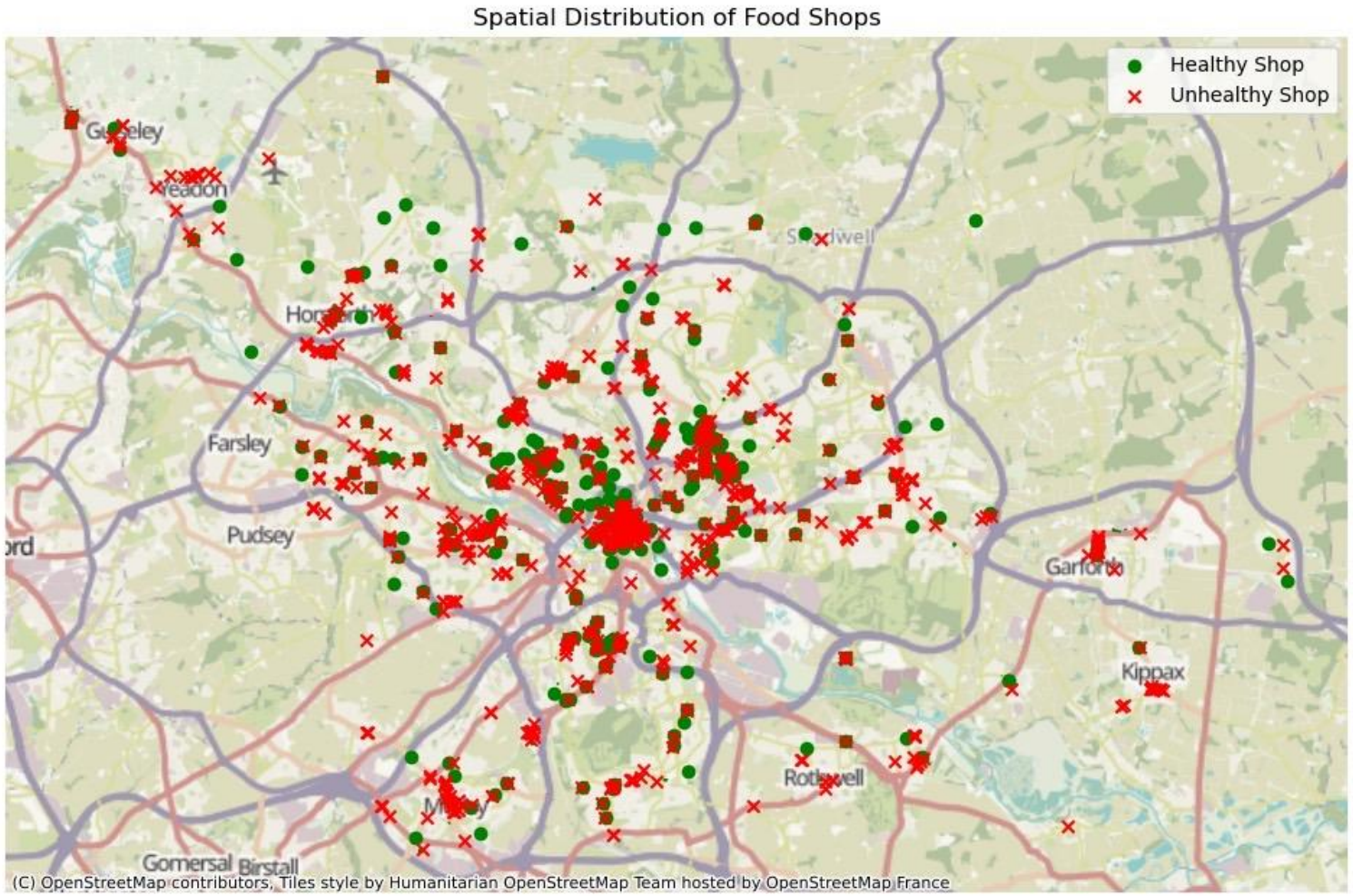




The areas with the highest concentration of fast-food restaurants are E01011444, E01033015, and E01011397, featuring 13, 11, and 9 fast-food restaurants per 100 km2, respectively.

The plot was processed by QGIS.

Distribution of food shops in Leeds



- **Healthy Food Accessibility**

  Category in OpenStreetMap, where people can get vegetable and fruit.

  Amenity = marketplace
  shop = supermarket, convenience, health_food, greengrocer

- **UnHealthy Food Accessibility**

  Category in OpenStreetMap.

  Amenity = fast_food
  shop = confectionery, chocolate, wine

The definition of each label on OpenStreetMap

Healthy Food Label

Amenity = marketplace

shop = greengrocer

shop = convenience

shop = supermarket

shop = health_food

Unhealthy Food Label

Amenity = fast_food

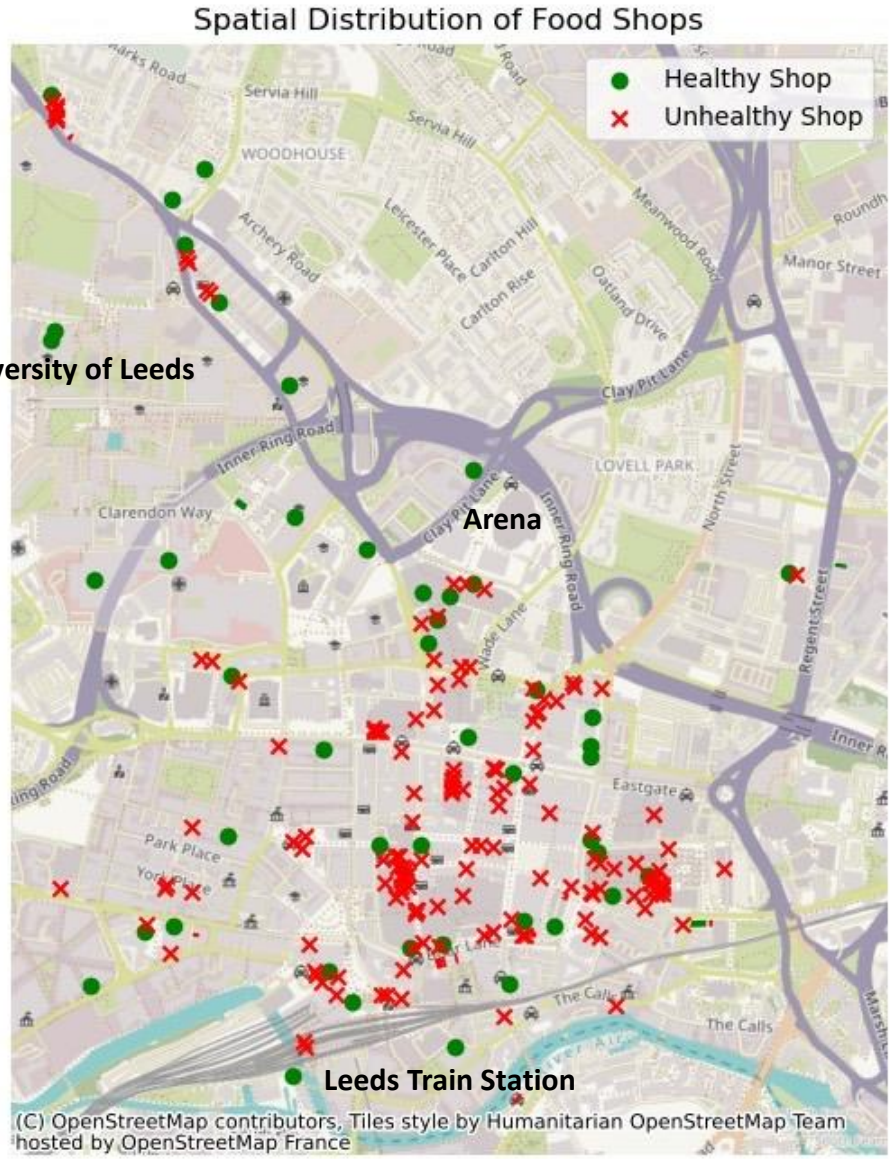
shop = confectionery

shop = wine

shop = chocolate

Source: OpenStreetMap Map Features: https://wiki.openstreetmap.org/wiki/Zh-hans:Map_Features#%E8%AE%BE%E6%96%BD(Amenity)

Distribution of food shops in LS1 and LS2 in Leeds

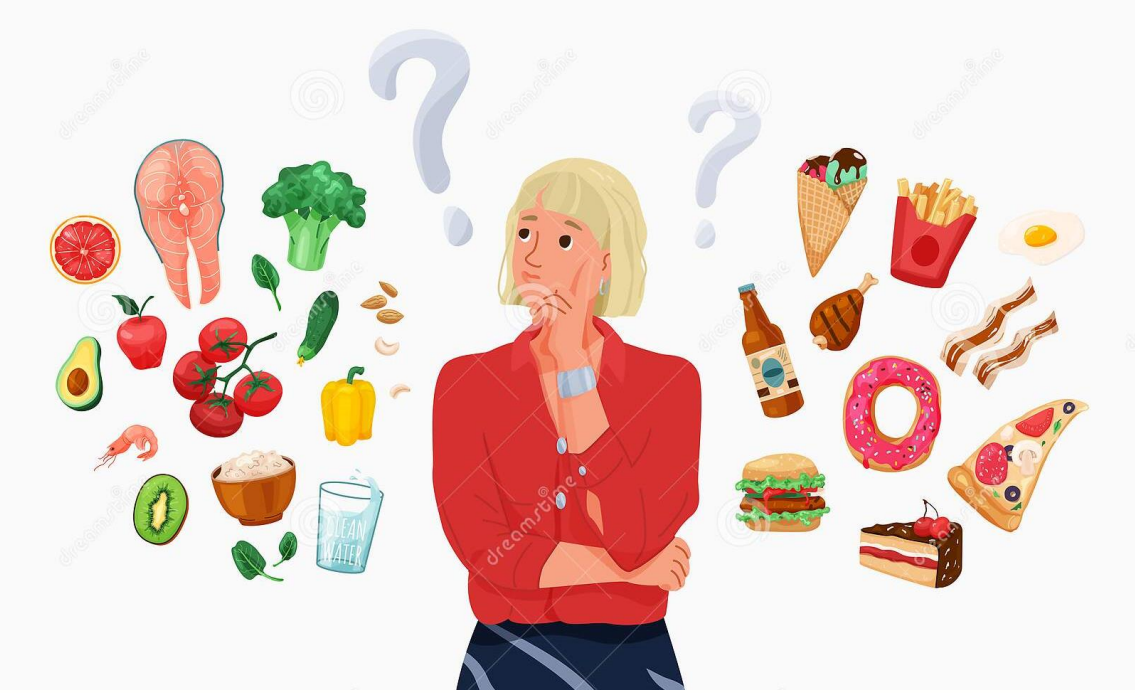


LS1 Area

LS2 Area

Solutions

Food Accessibility

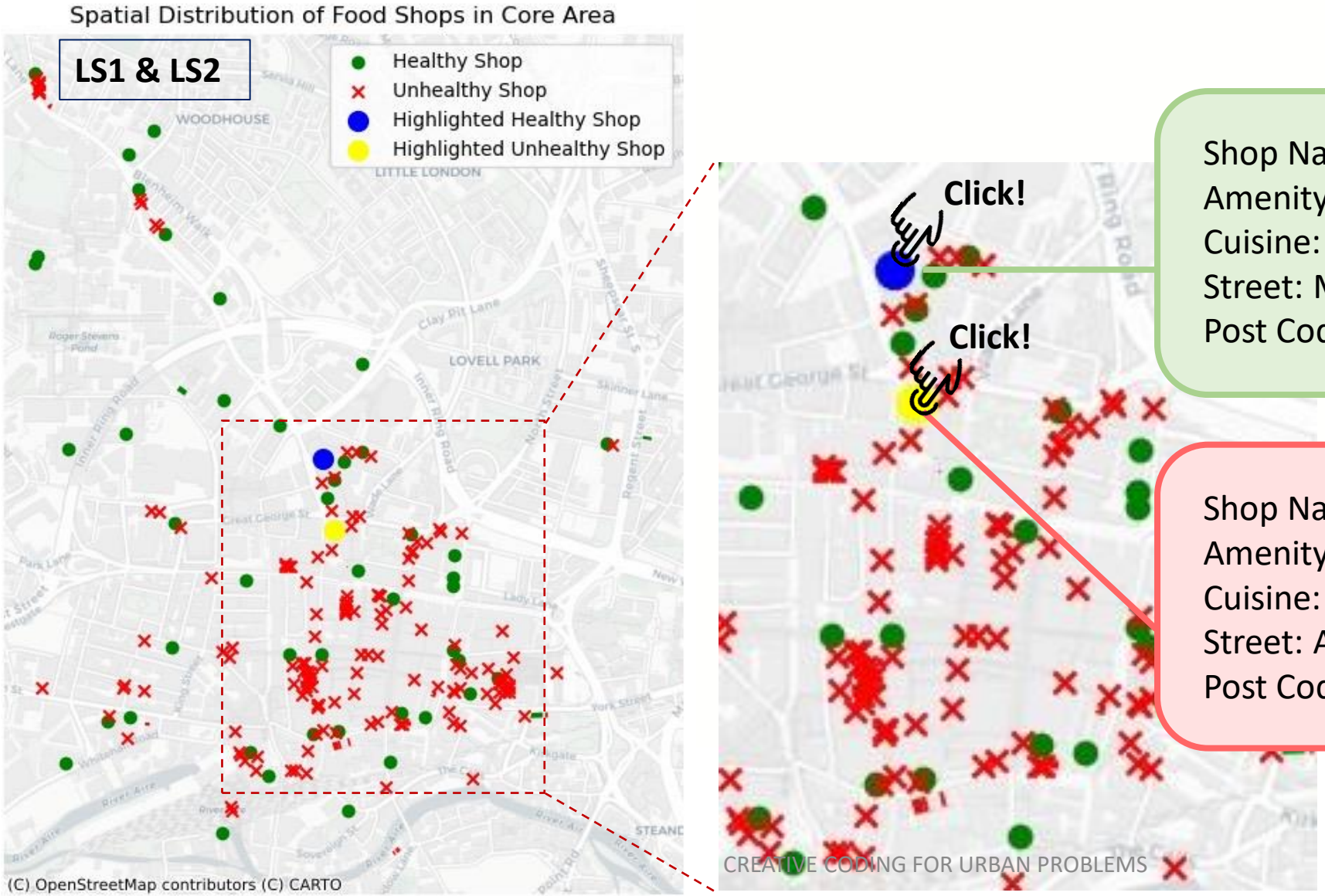


Picture Source: https://www.mylot.com/post/3442699/can-you-still-think-of-what-food-to-prepare-when-you-don-t-feel

People's choice of food is often confined to their daily routines and the areas they are familiar with. Utilizing the labels from OpenStreetMap, we have categorized restaurants into two groups: "healthy" and "unhealthy." Our visualization aims to provide residents with deeper insights into where they can find healthy dining options and recommend places close to their neighbourhoods. By interacting with the dots displayed on the map, users can explore different options.

Areas such as E01011444, E01033015, and E01011397, which are noted for having the highest number of fast-food restaurants, are highlighted to offer a clearer understanding of the dietary landscape within these locales.

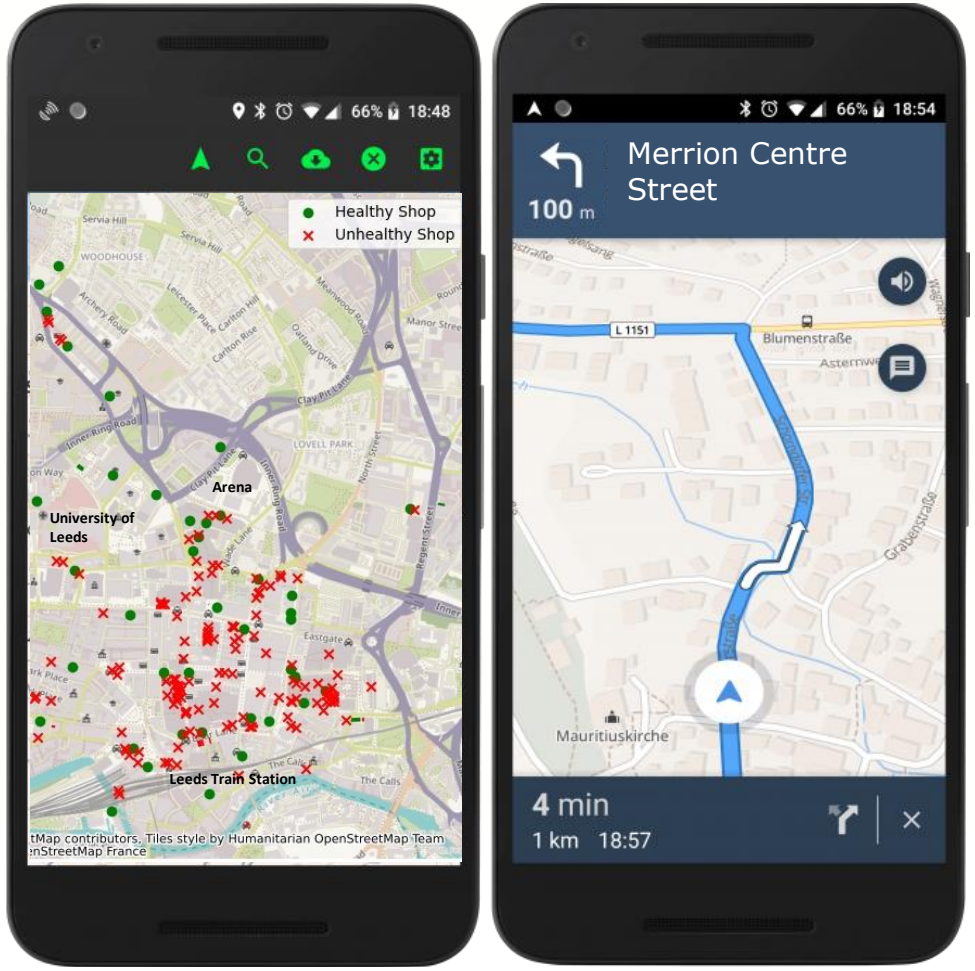
Solutions: Plots for Leeds resident



Shop Name: Morrisons
Amenity/Shop: Supermarket
Cuisine: NaN
Street: Merrion Centre
Post Code: LS2 8PL

Shop Name: McDonald's
Amenity /Shop : Fast Food
Cuisine: Burger
Street: Albion Street
Post Code: LS2 8LQ

Future Idea: An App Guide to Healthy and Unhealthy Eateries for Residents of Leeds



Unlike Google Maps and Apple Maps, this app exclusively displays healthy and unhealthy food options on the map, allowing users to easily identify the locations of healthy food.

In addition to the current restaurant information, the app will also provide data on the **average calories** consumed by customers and the **average Nutrient Profiling Model (NPM) score** of the food at each location.

This feature aims to offer additional health-related insights to assist individuals in making informed food choices.

Limitations

1. **Selection Bias in Shops**
   Our analysis selectively focuses on shops with a strong identity in Leeds, such as fast food restaurants and health shops. Shops lacking a distinct identity were excluded from the current analysis, potentially limiting the scope of our findings.

2. **Discrepancy in Data Granularity**
   The health data on residents, specifically the Adult Obesity Rate, covers a broader area (Ward level), whereas the advertisement asset data and restaurant data are subjected to a more detailed analysis. This discrepancy in the granularity of data sources limits our ability to conduct a comprehensive analysis of residents' health. The health data accessible to us at this point is not adequately suited for a detailed examination of residents' health outcomes.

References

1. Antrum, C.J., Waring, M.E., Cohen, J.F.W. and Cooksey Stowers, K., 2023. Within-store fast food marketing: The association between food swamps and unhealthy advertisement. Preventive Medicine Reports, 35, 102349.

2. Ares, G., Alcaire, F., Antúnez, L., Natero, V., de León, C., Gugliucci, V., Machín, L. and Otterbring, T., 2023. Exposure effects to unfamiliar food advertisements on YouTube: A randomized controlled trial among adolescents. Food Quality and Preference, 111, 104983.

3. Fraser, L.K. and Edwards, K.L., 2010. The association between the geography of fast food outlets and childhood obesity rates in Leeds, UK. Health & Place, 16(6), pp.1124-1128.

4. Rideout, R., Mah, C.L. and Minaker, L. (2015). Food environments: An introduction for public health practice. [chromeextension://efaidnbmnnnibpcajpcglclefindmkaj/https://ncceh.ca/sites/default/files/Food_Environments_Public_Health_Practice_Dec_2015.pdf]

Thank You!

Understanding the food environments in Leeds

1. Group Information

Group number: 6

1. 201722343
2. 201733915
3. 201798764
4. 201782291
5. 201788996

2. Import libaraies

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import geopandas as gpd
import contextily as ctx
from math import radians, cos, sin, sqrt, atan2
from shapely import wkt
from shapely.geometry import Point
```

3. Load the dataset

In [2]:
```
advert_location = pd.read_csv( 'MAAP_advertlocations.csv', encoding='latin1')
advert_detail = pd.read_csv( 'MAAPAdvertData2023.csv', encoding='latin1')
obesity=pd.read_csv('Adult_Obesity_rates.csv')
restaurant = pd.read_csv('fast_food_restaurants_leeds.csv', encoding='latin1')
```

4. Data Exploration

Look at basic information of each dataset

In [3]:
```
advert_location.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 675 entries, 0 to 674
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Unnamed: 0   675 non-null    int64  
 1   IMGid        675 non-null    object 
 2   Latitude     671 non-null    float64
 3   Longitude    671 non-null    float64
 4   LSOA_naming  675 non-null    object 
 5   lsoa_code    675 non-null    object 
 6   IMD_Q        675 non-null    object 
 7   LSOA_name    675 non-null    object 
dtypes: float64(2), int64(1), object(5)
memory usage: 42.3+ KB
```

In [4]:
```
advert_detail.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 926 entries, 0 to 925
Data columns (total 36 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Rinitials        926 non-null    object 
 1   Cinitials        926 non-null    object 
 2   Codedate         926 non-null    object 
 3   LSOA             926 non-null    object 
 4   IMGid            926 non-null    object 
 5   IMGdate          926 non-null    object 
 6   ADid             926 non-null    object 
 7   ASSETid          926 non-null    object 
 8   Long             0 non-null      float64
 9   Lat              0 non-null      float64
 10  ADtype           925 non-null    float64
 11  ADsize           925 non-null    float64
 12  ADprodtype       919 non-null    float64
 13  ADprod           920 non-null    object 
 14  ADbrand          903 non-null    object 
 15  Brandad          917 non-null    float64
 16  Admanagement     622 non-null    object 
 17  Price¡ê          172 non-null    float64
 18  Pricesourcelink  171 non-null    object 
 19  Unit             181 non-null    object 
 20  Portion          165 non-null    object 
 21  Totalwt          187 non-null    float64
 22  Wtsource         189 non-null    object 
 23  Nutritionsource  191 non-null    object 
 24  Nutsourcelink    192 non-null    object 
 25  Fatdensity       191 non-null    float64
 26  Satfatdensity    191 non-null    float64
 27  Sugardensity     191 non-null    float64
 28  Sodiumdensity    191 non-null    float64
 29  Ekcaldensity     191 non-null    float64
 30  Fibredensity     191 non-null    float64
 31  Proteindensity   191 non-null    float64
 32  FVN              191 non-null    float64
 33  NPMscore         191 non-null    float64
 34  NPMstatus        191 non-null    float64
 35  ADcompliance     190 non-null    float64
dtypes: float64(19), object(17)
memory usage: 260.6+ KB
```

In [5]:
```
obesity.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2553 entries, 0 to 2552
Data columns (total 7 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   geography_type    2553 non-null   object 
 1   geography         2553 non-null   object 
 2   auditdate         2553 non-null   int64  
 3   age_range         2553 non-null   object 
 4   Sum of dsr        2553 non-null   float64
 5   Sum of lower95CI  2553 non-null   float64
 6   Sum of upper95CI  2553 non-null   float64
dtypes: float64(3), int64(1), object(3)
memory usage: 139.7+ KB
```

In [6]:
```
restaurant.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1808 entries, 0 to 1807
Columns: 158 entries, addr:city to healthy shop
dtypes: float64(10), int64(1), object(147)
memory usage: 2.2+ MB
```

Look at the first few columns or the last few columns of each dataset

In [7]:
```
advert_location.head()
```

Out[7]:

	Unnamed: 0	IMGid	Latitude	Longitude	LSOA_naming	lsoa_c
0	0	IMG_5123	53.805711	-1.628422	E01011287_Q3_SpringValleys	E01011
1	0	IMG_5124	53.805664	-1.628519	E01011287_Q3_SpringValleys	E01011
2	0	IMG_5125	53.805928	-1.632267	E01011287_Q3_SpringValleys	E01011
3	0	IMG_5126	53.805928	-1.632200	E01011287_Q3_SpringValleys	E01011
4	0	IMG_5127	53.805908	-1.632492	E01011287_Q3_SpringValleys	E01011

In [8]:
```
advert_location.tail()
```

Out[8]:

	Unnamed: 0	IMGid	Latitude	Longitude	L
670	0	20230523_113538	53.797741	-1.522829	E01033021_Q2_UpperAccor
671	0	20230523_113831	53.797844	-1.520840	E01033021_Q2_UpperAccor
672	0	20230523_115503	53.795664	-1.516769	E01033021_Q2_UpperAccor
673	0	20230523_115524	53.795188	-1.517370	E01033021_Q2_UpperAccor
674	0	20230523_115524	53.795188	-1.517370	E01033021_Q2_UpperAccor

In [9]:
```
advert_detail.head()
```

Out[9]:

	Rinitials	Cinitials	Codedate	LSOA	IMGid	IMGdate	Al
0	IW	IW	2023/5/17	E01011597	IMG_4652.JPG	2023/5/11	CF_GLMR_01
1	IW	IW	2023/5/17	E01011597	IMG_4653.JPG	2023/5/11	CF_GLMR_02
2	IW	IW	2023/5/17	E01011597	IMG_4657.JPG	2023/5/11	CF_GLMR_03
3	IW	IW	2023/5/17	E01011597	IMG_4658.JPG	2023/5/11	CF_GLMR_04
4	IW	IW	2023/5/17	E01011597	IMG_4659.JPG	2023/5/11	CF_GLMR_05

5 rows × 36 columns

In [10]:
```
obesity.head()
```

Out[10]:

	geography_type	geography	auditdate	age_range	Sum of dsr	Sum of lower95CI	Su uppe
0	Community Committee	Inner East	201307	16+	27352.76	26881.6769	27829
1	Community Committee	Inner East	201310	16+	27277.53	26808.4482	27752
2	Community Committee	Inner East	201401	16+	27266.77	26796.3394	27743
3	Community Committee	Inner East	201404	16+	27264.18	26798.2996	27735
4	Community Committee	Inner East	201407	16+	27264.00	26798.9098	27734

In [11]:
```
restaurant.tail()
```

Out[11]:

	addr:city	addr:housenumber	addr:postcode	addr:street	amenity	cuisine
1803	NaN	NaN	NaN	NaN	NaN	NaN
1804	Leeds	97	LS8 5AJ	Roundhay Road	NaN	NaN
1805	Leeds	60	LS18 4AP	Town Street	NaN	NaN
1806	Leeds	57	LS8 1AP	Street Lane	NaN	NaN
1807	Leeds	203	LS11 5EG	Dewsbury Road	NaN	NaN

5 rows × 158 columns

Look at statistical information of each dataset

In [12]:
```
obesity.describe()
```

Out[12]:

	auditdate	Sum of dsr	Sum of lower95CI	Sum of upper95CI
count	2553.000000	2553.000000	2553.000000	2553.000000
mean	201776.457109	23975.508042	23338.625322	24627.330133
std	278.320420	3676.522090	3660.735064	3716.716300
min	201207.000000	16120.130000	14253.455600	17144.709200
25%	201510.000000	20404.820000	19830.488600	20966.098300
50%	201801.000000	24407.180000	23852.588000	25049.868100
75%	202007.000000	26907.720000	26252.312400	27625.629400
max	202301.000000	32948.740000	32174.671800	34878.341600

In [13]:
```
restaurant.describe()
```

Out[13]:

	level	ele	ele:msl	old_fhrs:id	layer	ref:UK:leedscc:bi
count	51.000000	2.000000	2.000000	2.000000	6.000000	1.
mean	0.823529	49.750000	49.750000	616865.000000	1.833333	4424.
std	0.865006	2.899138	2.899138	419325.634952	0.983192	NaN
min	-1.000000	47.700000	47.700000	320357.000000	1.000000	4424.
25%	0.000000	48.725000	48.725000	468611.000000	1.000000	4424.
50%	1.000000	49.750000	49.750000	616865.000000	1.500000	4424.
75%	1.500000	50.775000	50.775000	765119.000000	2.750000	4424.
max	2.000000	51.800000	51.800000	913373.000000	3.000000	4424.

In [14]:
```
advert_location.describe()
```

Out[14]:

	Unnamed: 0	Latitude	Longitude
count	675.0	671.000000	671.000000
mean	0.0	53.801380	-1.550777
std	0.0	0.019782	0.063984
min	0.0	53.735665	-1.704161
25%	0.0	53.792263	-1.574479
50%	0.0	53.800561	-1.564819
75%	0.0	53.811310	-1.514625
max	0.0	53.854278	-1.368975

In [15]:
```
advert_detail.describe()
```

Out[15]:

	Long	Lat	ADtype	ADsize	ADprodtype	Brandad	Price¡ê
count	0.0	0.0	925.000000	925.000000	919.000000	917.000000	172.000000
mean	NaN	NaN	6.894054	1.916757	4.047878	0.294438	4.789244
std	NaN	NaN	4.498150	0.612452	1.606052	0.456039	7.486794
min	NaN	NaN	1.000000	1.000000	1.000000	0.000000	0.750000
25%	NaN	NaN	2.000000	2.000000	3.000000	0.000000	1.990000
50%	NaN	NaN	6.000000	2.000000	5.000000	0.000000	2.870000
75%	NaN	NaN	10.000000	2.000000	5.000000	1.000000	5.097500
max	NaN	NaN	14.000000	3.000000	5.000000	1.000000	85.000000

5. Data Wrangling

Merge two datasets related to advertisements

In [16]:
```
# Match the format
advert_detail['IMGid_Clean'] = advert_detail['IMGid'].str.replace(r'\.JPG', '',
advert_location['IMGid_Clean'] = advert_location['IMGid']

# Merge two datasets
advert = pd.merge(advert_detail, advert_location, on='IMGid_Clean', how='inner')
advert.head()
```

Out[16]:

	Rinitials	Cinitials	Codedate	LSOA	IMGid_x	IMGdate	AI
0	IW	IW	2023/5/17	E01011597	IMG_4652.JPG	2023/5/11	CF_GLMR_01
1	IW	IW	2023/5/17	E01011597	IMG_4653.JPG	2023/5/11	CF_GLMR_02
2	IW	IW	2023/5/17	E01011597	IMG_4657.JPG	2023/5/11	CF_GLMR_03
3	IW	IW	2023/5/17	E01011597	IMG_4658.JPG	2023/5/11	CF_GLMR_04
4	IW	IW	2023/5/17	E01011597	IMG_4659.JPG	2023/5/11	CF_GLMR_05

5 rows × 45 columns

In [17]:
```
# Filter data with median ADprodtype values of 1,2,3
if 'ADprodtype' in advert.columns:
    advert_cleaned = advert[advert['ADprodtype'].isin([1, 2, 3])]
else:
    advert_cleaned = pd.DataFrame()

advert_cleaned.head()
```

Out[17]:

	Rinitials	Cinitials	Codedate	LSOA	IMGid_x	IMGdate	A
3	IW	IW	2023/5/17	E01011597	IMG_4658.JPG	2023/5/11	CF_GLMR_0
5	IW	IW	2023/5/17	E01011597	IMG_4660.JPG	2023/5/11	CF_GLMR_0
6	IW	IW	2023/5/17	E01011597	IMG_4660.JPG	2023/5/11	CF_GLMR_0
8	IW	IW	2023/5/17	E01011597	IMG_4662.JPG	2023/5/11	CF_GLMR_0
12	IW	IW	2023/5/17	E01011597	IMG_4666.JPG	2023/5/11	CF_GLMR_0

5 rows × 45 columns

In addition to longitude and latitude, for numeric columns, we utilize the mean of each column to fill in the missing values. For categorical columns, we utilize the mode of each column to fill in the missing values.

In [18]:
```
for column in advert_cleaned.select_dtypes(include=[np.number]).columns:
    if column not in ['Latitude', 'Longitude']:
        advert_cleaned[column].fillna(advert_cleaned[column].mean(), inplace=Tru

for column in advert_cleaned.select_dtypes(include=['object', 'category']).colum
    advert_cleaned[column].fillna(advert_cleaned[column].mode()[0], inplace=True

# Check the result after filling in the missing value
advert_cleaned.isnull().sum()
```

```
C:\Users\111\AppData\Local\Temp\ipykernel_26244\1537538335.py:3: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  advert_cleaned[column].fillna(advert_cleaned[column].mean(), inplace=True)
C:\Users\111\AppData\Local\Temp\ipykernel_26244\1537538335.py:6: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  advert_cleaned[column].fillna(advert_cleaned[column].mode()[0], inplace=True)
```

Out[18]:
```
Rinitials            0
Cinitials            0
Codedate             0
LSOA                 0
IMGid_x              0
IMGdate              0
ADid                 0
ASSETid              0
Long               236
Lat                236
ADtype               0
ADsize               0
ADprodtype           0
ADprod               0
ADbrand              0
Brandad              0
Admanagement         0
Price¡ê              0
Pricesourcelink      0
Unit                 0
Portion              0
Totalwt              0
Wtsource             0
Nutritionsource      0
Nutsourcelink        0
Fatdensity           0
Satfatdensity        0
Sugardensity         0
Sodiumdensity        0
Ekcaldensity         0
Fibredensity         0
Proteindensity       0
FVN                  0
NPMscore             0
NPMstatus            0
ADcompliance         0
IMGid_Clean          0
Unnamed: 0           0
IMGid_y              0
Latitude             0
Longitude            0
LSOA_naming          0
lsoa_code            0
IMD_Q                0
LSOA_name            0
dtype: int64
```

In [19]:
```python
# Use Google Maps to get the longitude and latitude of the centre locations in v
areas_coordinates = {
    "Inner East": (53.799, -1.534),
    "Inner North East": (53.830, -1.520),
    "Inner North West": (53.830, -1.570),
    "Inner South": (53.760, -1.540),
    "Inner West": (53.810, -1.580),
    "Outer East": (53.800, -1.450),
    "Outer North East": (53.850, -1.460),
    "Outer North West": (53.850, -1.620),
    "Outer South": (53.720, -1.540),
    "Outer West": (53.810, -1.650)
}
```

```
# Calculate the distance between two locations by using Haversine
def haversine(lat1, lon1, lat2, lon2):
    R = 6371.0 # The average radius of the Earth
    dlat = radians(lat2 - lat1)
    dlon = radians(lon2 - lon1)
    a = sin(dlat / 2)**2 + cos(radians(lat1)) * cos(radians(lat2)) * sin(dlon /
    c = 2 * atan2(sqrt(a), sqrt(1 - a))
    distance = R * c
    return distance


advert_cleaned['Nearest Area'] = None # Add a new column to save the location


# Calculate the distance of each location from the center locations and find the
for index, row in advert_cleaned.iterrows():
    min_distance = float('inf')
    nearest_area = None
    for area, (lat, lon) in areas_coordinates.items():
        distance = haversine(row['Latitude'], row['Longitude'], lat, lon)
        if distance < min_distance:
            min_distance = distance
            nearest_area = area
    advert_cleaned.at[index, 'Nearest Area'] = nearest_area
```

```
C:\Users\111\AppData\Local\Temp\ipykernel_26244\2542133472.py:26: SettingWithCopy
Warning:
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/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  advert_cleaned['Nearest Area'] = None # Add a new column to save the location
```

After selecting columns what we want to analyze, we can get the cleaned dateset of advertisements

In [20]:
```
advert_cleaned=advert_cleaned[['Latitude','Longitude','IMD_Q','Nearest Area']]
advert_cleaned.head()
```

Out[20]:

	Latitude	Longitude	IMD_Q	Nearest Area
3	53.804011	-1.694178	Q2	Outer West
5	53.803447	-1.696975	Q2	Outer West
6	53.803447	-1.696975	Q2	Outer West
8	53.802867	-1.699847	Q2	Outer West
12	53.802836	-1.699847	Q2	Outer West

Now,we can clean the dataset of restaurants in Leeds

In [21]:
```
# Filter for rows where 'addr:city' in Leeds
restaurant = restaurant[restaurant['addr:city'] == 'Leeds']
# Filter for postcodes LS1 to LS10 using regex to capture LS1 to LS10
```

```
restaurant = restaurant[restaurant['addr:postcode'].str.match(r'^LS([1-2]|)\s',
restaurant.head()
```

Out[21]:

	addr:city	addr:housenumber	addr:postcode	addr:street	amenity	cu
45	Leeds	12-??Q????	LS1 6DN	Call Lane	fast_food	
46	Leeds	1-1a	LS2 7DA	Crown Street	fast_food	
55	Leeds	35	LS1 6HD	Briggate	fast_food	b
56	Leeds	8	LS1 6LQ	Thornton's Arcade	fast_food	san
57	Leeds	59	LS1 6LR	The Headrow	fast_food	sandwich;b

5 rows × 158 columns

In [22]:
```
# Use only the required columns
restaurant=restaurant[['addr:postcode','geometry','healthy shop']]
restaurant.head()
```

Out[22]:

	addr:postcode	geometry	healthy shop
45	LS1 6DN	POINT (-1.5401096 53.7964295)	0
46	LS2 7DA	POINT (-1.5399345 53.7961637)	0
55	LS1 6HD	POINT (-1.5428864 53.7961961)	0
56	LS1 6LQ	POINT (-1.5425279 53.7987443)	0
57	LS1 6LR	POINT (-1.5428149 53.7990671)	0

In [23]:
```
# Extract longitude and latitude columns
def extract_coordinates(row):
    if 'POINT' in row:
        coords = row.split('(')[1].split(')')[0].split(' ')
        return pd.Series([coords[0], coords[1]])
    elif 'POLYGON' in row:
        coords = row.split('((')[1].split(')')[0].split(', ')
        return pd.Series([coords[0].split(' ')[0], coords[0].split(' ')[1]])
    else:
        return pd.Series([None, None])

restaurant[['longitude', 'latitude']] =restaurant['geometry'].apply(extract_coor
restaurant.head()
```

Out[23]:

	addr:postcode	geometry	healthy shop	longitude	latitude
45	LS1 6DN	POINT (-1.5401096 53.7964295)	0	-1.5401096	53.7964295
46	LS2 7DA	POINT (-1.5399345 53.7961637)	0	-1.5399345	53.7961637
55	LS1 6HD	POINT (-1.5428864 53.7961961)	0	-1.5428864	53.7961961
56	LS1 6LQ	POINT (-1.5425279 53.7987443)	0	-1.5425279	53.7987443
57	LS1 6LR	POINT (-1.5428149 53.7990671)	0	-1.5428149	53.7990671

In [24]:
```
# Modify the type of latitude and longitude
restaurant['longitude'] = restaurant['longitude'].astype(float)
restaurant['latitude'] = restaurant['latitude'].astype(float)
restaurant.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 207 entries, 45 to 1801
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   addr:postcode  207 non-null    object 
 1   geometry       207 non-null    object 
 2   healthy shop   207 non-null    int64  
 3   longitude      207 non-null    float64
 4   latitude       207 non-null    float64
dtypes: float64(2), int64(1), object(2)
memory usage: 9.7+ KB
```

In [25]:
```
restaurant['Nearest Area'] = None

for index, row in restaurant.iterrows():
    min_distance = float('inf')
    nearest_area = None
    for area, (lat, lon) in areas_coordinates.items():
        distance = haversine(row['latitude'], row['longitude'], lat, lon)
        if distance < min_distance:
            min_distance = distance
            nearest_area = area
    restaurant.at[index, 'Nearest Area'] = nearest_area
```

We can get the cleaned dateset of restaurants in Leeds

In [26]:
```
restaurant.head()
```

Out[26]:

	addr:postcode	geometry	healthy shop	longitude	latitude	Nearest Area
45	LS1 6DN	POINT (-1.5401096 53.7964295)	0	-1.540110	53.796430	Inner East
46	LS2 7DA	POINT (-1.5399345 53.7961637)	0	-1.539934	53.796164	Inner East
55	LS1 6HD	POINT (-1.5428864 53.7961961)	0	-1.542886	53.796196	Inner East
56	LS1 6LQ	POINT (-1.5425279 53.7987443)	0	-1.542528	53.798744	Inner East
57	LS1 6LR	POINT (-1.5428149 53.7990671)	0	-1.542815	53.799067	Inner East

In [27]:
```
restaurant.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 207 entries, 45 to 1801
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   addr:postcode  207 non-null    object 
 1   geometry       207 non-null    object 
 2   healthy shop   207 non-null    int64  
 3   longitude      207 non-null    float64
 4   latitude       207 non-null    float64
 5   Nearest Area   207 non-null    object 
dtypes: float64(2), int64(1), object(3)
memory usage: 19.4+ KB
```

6. Data Analysis

Before beginning to data analysis, we need to have an understanding of each cleaned dataset again.

In [28]:
```
restaurant.head()
```

Out[28]:

	addr:postcode	geometry	healthy shop	longitude	latitude	Nearest Area
45	LS1 6DN	POINT (-1.5401096 53.7964295)	0	-1.540110	53.796430	Inner East
46	LS2 7DA	POINT (-1.5399345 53.7961637)	0	-1.539934	53.796164	Inner East
55	LS1 6HD	POINT (-1.5428864 53.7961961)	0	-1.542886	53.796196	Inner East
56	LS1 6LQ	POINT (-1.5425279 53.7987443)	0	-1.542528	53.798744	Inner East
57	LS1 6LR	POINT (-1.5428149 53.7990671)	0	-1.542815	53.799067	Inner East

In [29]:
```
advert_cleaned.head()
```

Out[29]:

	Latitude	Longitude	IMD_Q	Nearest Area
3	53.804011	-1.694178	Q2	Outer West
5	53.803447	-1.696975	Q2	Outer West
6	53.803447	-1.696975	Q2	Outer West
8	53.802867	-1.699847	Q2	Outer West
12	53.802836	-1.699847	Q2	Outer West

In [30]:
```
obesity.head()
```

Out[30]:

	geography_type	geography	auditdate	age_range	Sum of dsr	Sum of lower95CI	Su upper
0	Community Committee	Inner East	201307	16+	27352.76	26881.6769	27829
1	Community Committee	Inner East	201310	16+	27277.53	26808.4482	27752
2	Community Committee	Inner East	201401	16+	27266.77	26796.3394	27743
3	Community Committee	Inner East	201404	16+	27264.18	26798.2996	27735
4	Community Committee	Inner East	201407	16+	27264.00	26798.9098	27734

In [31]:
```
restaurant.head()
```

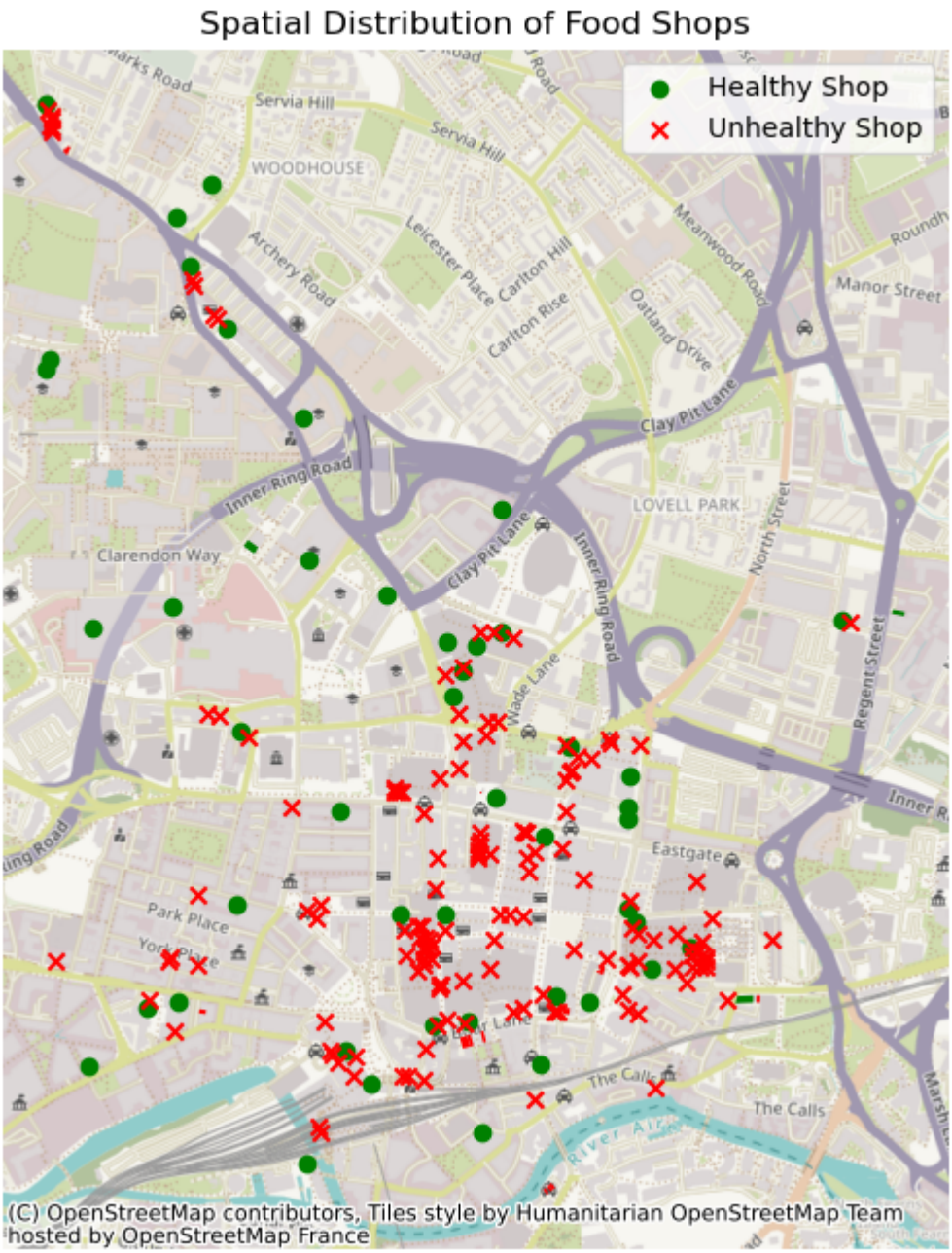
Out[31]:

	addr:postcode	geometry	healthy shop	longitude	latitude	Nearest Area
45	LS1 6DN	POINT (-1.5401096 53.7964295)	0	-1.540110	53.796430	Inner East
46	LS2 7DA	POINT (-1.5399345 53.7961637)	0	-1.539934	53.796164	Inner East
55	LS1 6HD	POINT (-1.5428864 53.7961961)	0	-1.542886	53.796196	Inner East
56	LS1 6LQ	POINT (-1.5425279 53.7987443)	0	-1.542528	53.798744	Inner East
57	LS1 6LR	POINT (-1.5428149 53.7990671)	0	-1.542815	53.799067	Inner East

In the first step, we want to understand the distribution of health shops in each area.

In [32]:
```
# Converts the 'geometry' into shapely's geometry object
restaurant['geo_geometry'] = restaurant['geometry'].apply(wkt.loads)
gdf = gpd.GeoDataFrame(restaurant, geometry='geo_geometry')

# Set the initial coordinate reference system CRS of gdf to WGS 84 (EPSG:4326)
gdf.crs = "EPSG:4326"

# Create two geodataframes: one for the health food store and one for the unheal
gdf_healthy = gdf[gdf['healthy shop'] == 1]
gdf_unhealthy = gdf[gdf['healthy shop'] == 0]

# Convert the coordinates of the data set to a Web Mercator to match the base ma
gdf_healthy = gdf_healthy.to_crs(epsg=3857)
gdf_unhealthy = gdf_unhealthy.to_crs(epsg=3857)

# Draw the spatial distribution
fig, ax = plt.subplots(figsize=(12, 8))
gdf_healthy.plot(ax=ax, color='green', marker='o', label='Healthy Shop')
gdf_unhealthy.plot(ax=ax, color='red', marker='x', label='Unhealthy Shop')

# Add the map
ctx.add_basemap(ax)
ax.set_title('Spatial Distribution of Food Shops')
ax.legend()
ax.set_axis_off()
plt.show()
```

```
C:\Users\111\AppData\Local\Temp\ipykernel_26244\1783292374.py:24: UserWarning: Le
gend does not support handles for PatchCollection instances.
See: https://matplotlib.org/stable/tutorials/intermediate/legend_guide.html#imple
menting-a-custom-legend-handler
  ax.legend()
```



Next, we need to understand the distribution of the number of healthy shops and unhealthy shops in different areas.

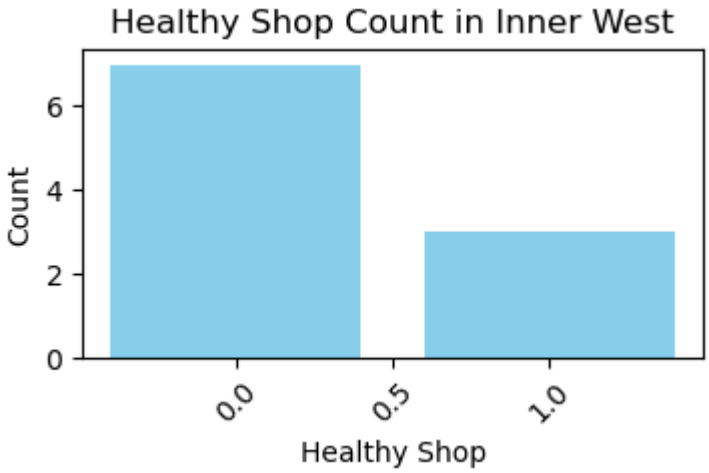
In [33]:
```
# The distribution of the number of healthy shops and unhealthy shops in differe
restaurant_type = restaurant.groupby(['Nearest Area', 'healthy shop']).size().re
restaurant_type.head()
```

Out[33]:

	Nearest Area	healthy shop	Count
0	Inner East	0	151
1	Inner East	1	46
2	Inner West	0	7
3	Inner West	1	3

In [34]:
```
# Draw a chart for each area to show its food store type
for area, group in restaurant_type.groupby('Nearest Area'):
    plt.figure(figsize=(4, 2))
```

```
    plt.bar(group['healthy shop'], group['Count'], color='skyblue')
    plt.title(f'Healthy Shop Count in {area}')
    plt.xlabel('Healthy Shop')
    plt.ylabel('Count')
    plt.xticks(rotation=45)
    plt.show()
```



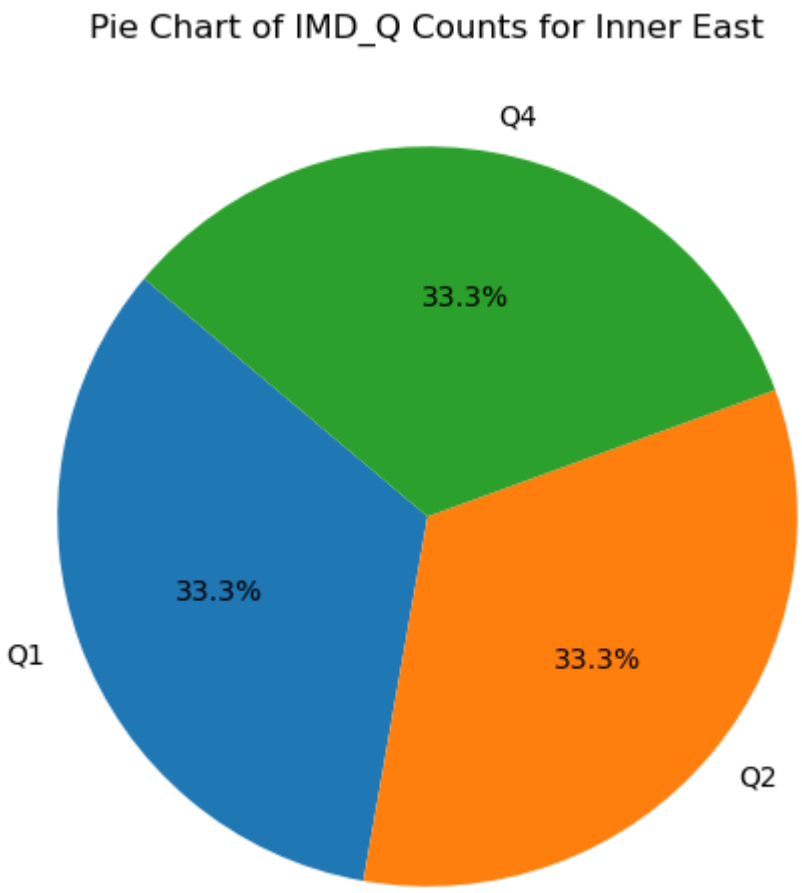
In [35]:
```
# Depending on different areas, we can get different numbers of IMD types of ads
grouped_advert_cleaned = advert_cleaned.groupby(['Nearest Area', 'IMD_Q']).size(
grouped_advert_cleaned.head()
```

Out[35]:

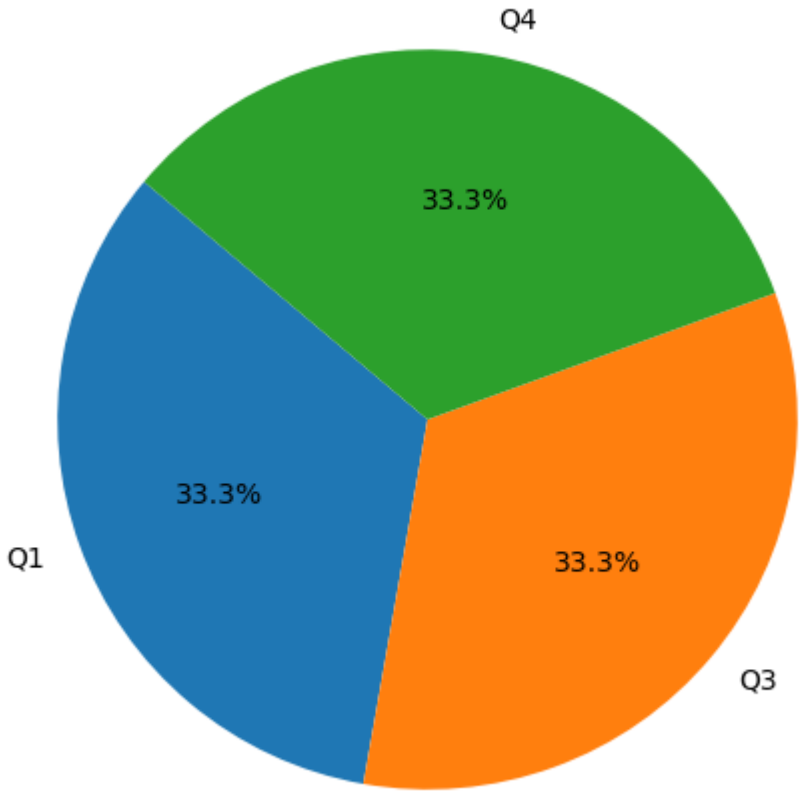
	Nearest Area	IMD_Q	Count
0	Inner East	Q1	6
1	Inner East	Q2	3
2	Inner East	Q4	24
3	Inner North East	Q1	3
4	Inner North East	Q3	2

In [36]:
```
# Visualize different amounts of ads in different areas
unique_areas = grouped_advert_cleaned['Nearest Area'].unique()
for area in unique_areas:
    area_data = grouped_advert_cleaned[grouped_advert_cleaned['Nearest Area'] ==
    area_counts = area_data['IMD_Q'].value_counts()

    plt.figure(figsize=(6, 6))
```

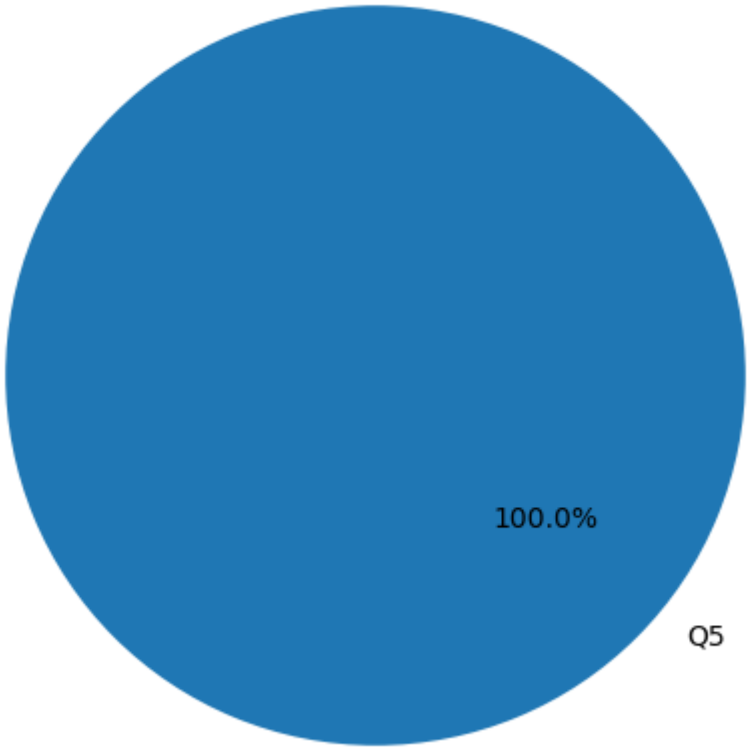
```
    plt.pie(area_counts, labels=area_counts.index, autopct='%1.1f%%', startangle
    plt.title(f'Pie Chart of IMD_Q Counts for {area}')
    plt.show()
```



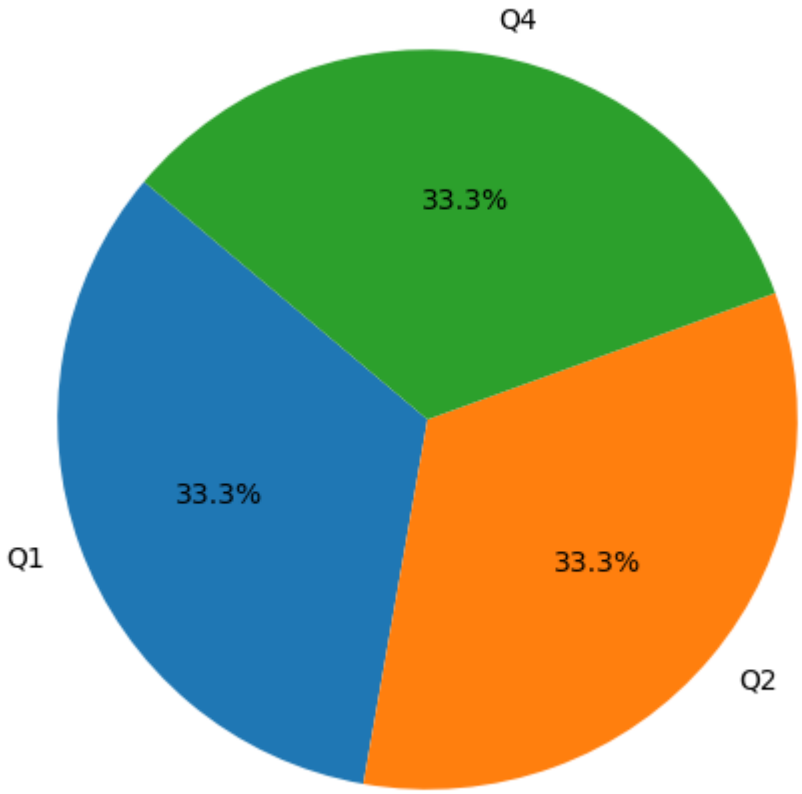
Pie Chart of IMD_Q Counts for Inner North East



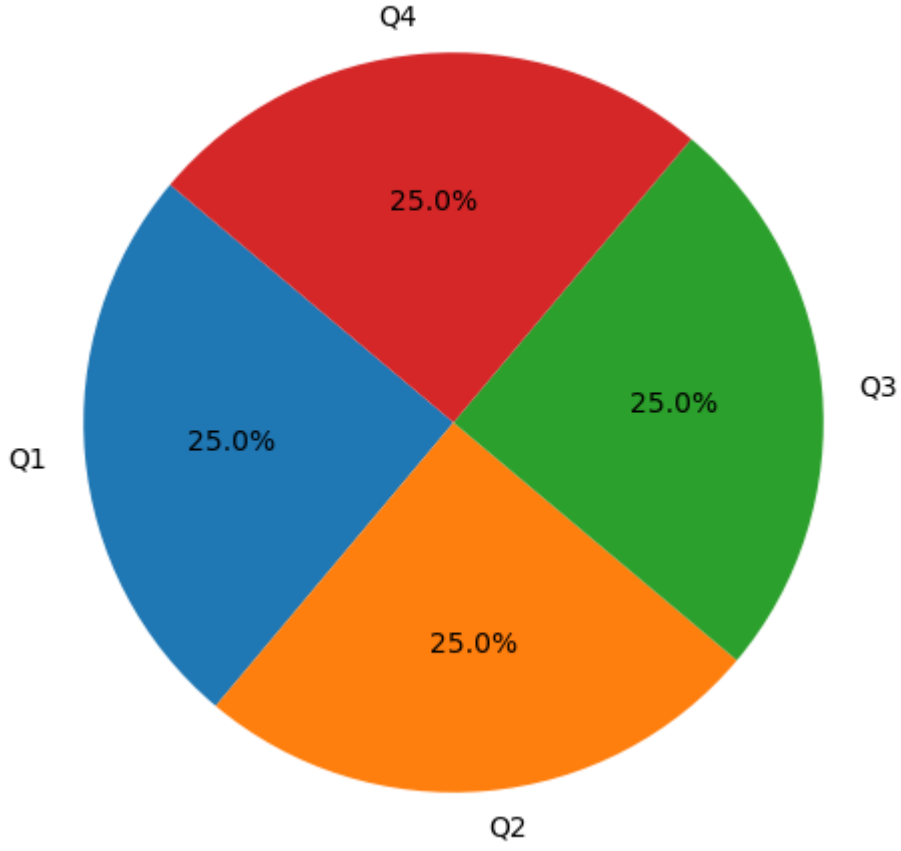
Pie Chart of IMD_Q Counts for Inner North West



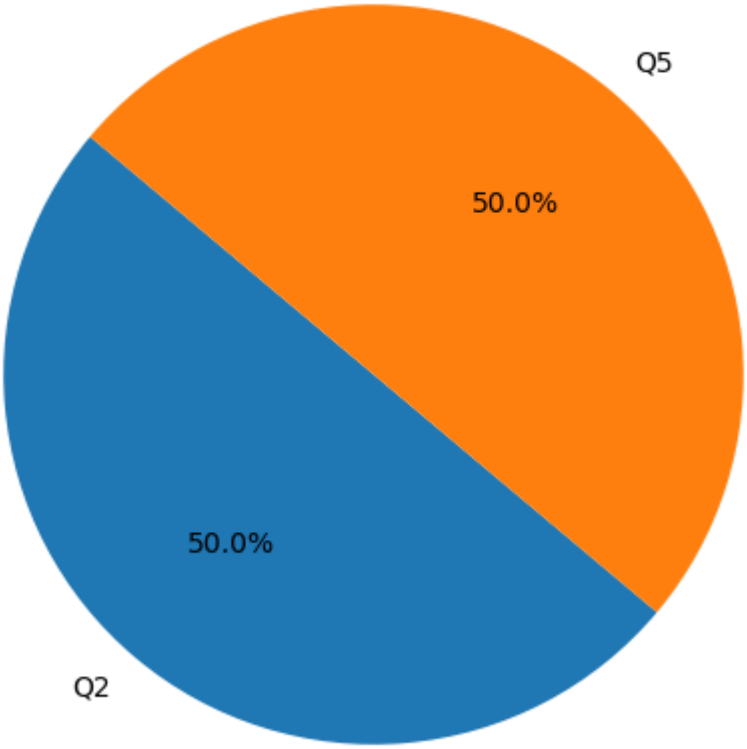
Pie Chart of IMD_Q Counts for Inner South



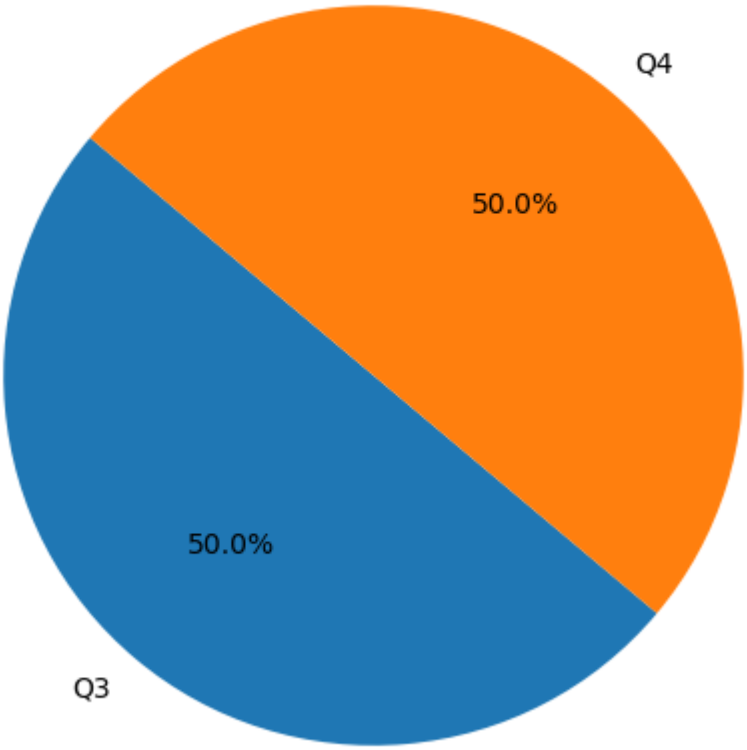
Pie Chart of IMD_Q Counts for Inner West



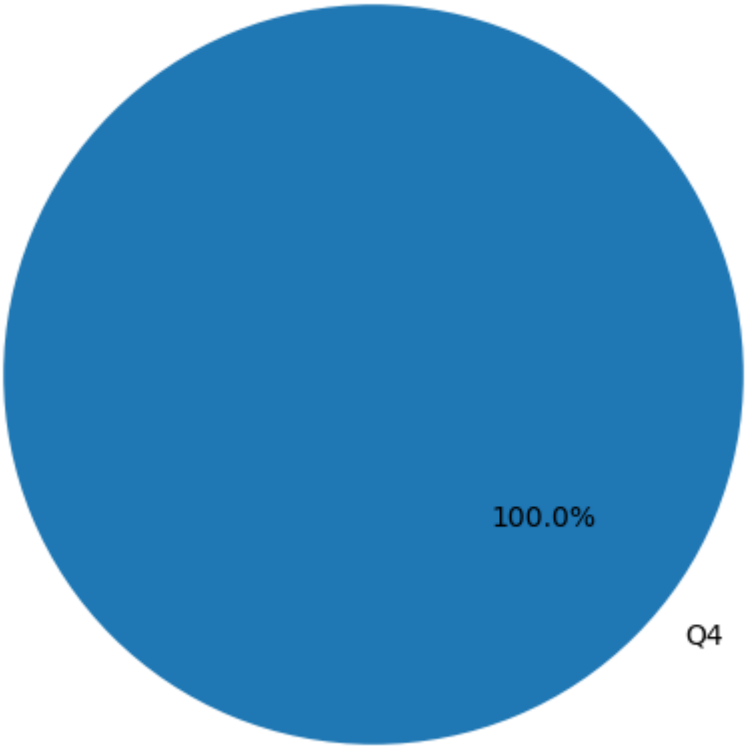
Pie Chart of IMD_Q Counts for Outer East



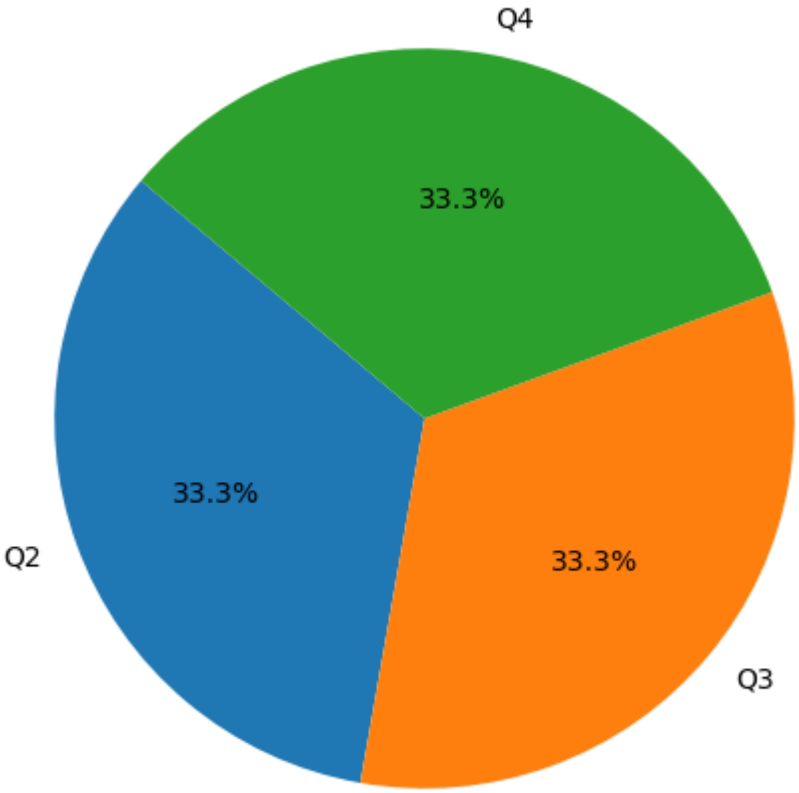
Pie Chart of IMD_Q Counts for Outer North West



Pie Chart of IMD_Q Counts for Outer South



Pie Chart of IMD_Q Counts for Outer West



Due to dataset limitations, some LSOAs do not correctly reflect geographic locations. We divide them into seven regions according to their latitude and longitude.

In [37]:
```
regions=["Inner East","Inner North East","Inner North West","Inner South","Inner
regions
```

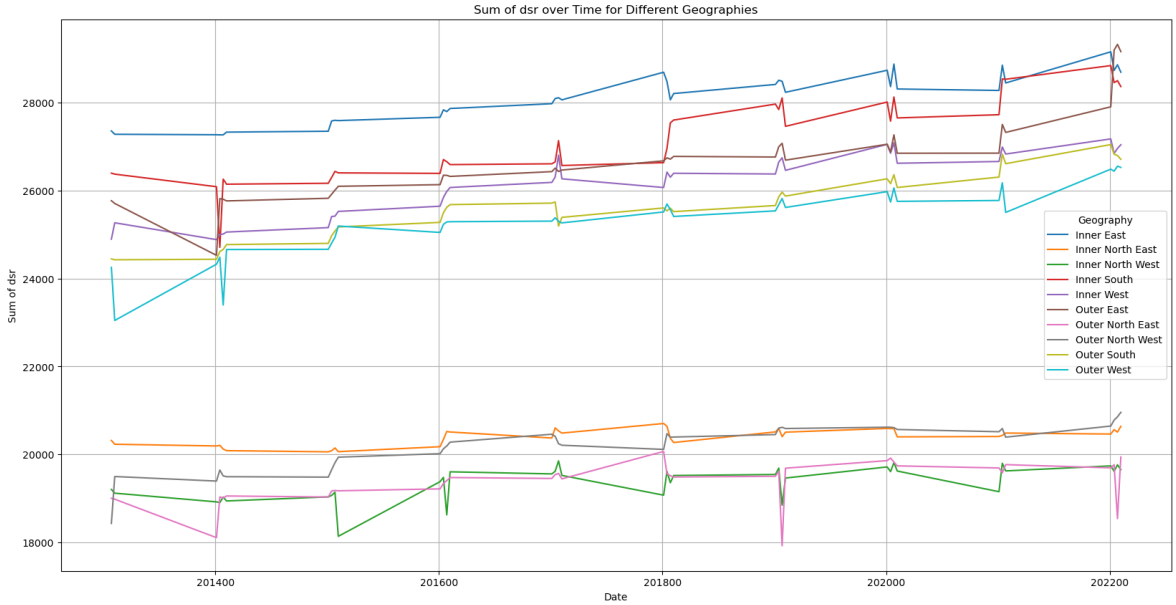
Out[37]:
```
['Inner East',
 'Inner North East',
 'Inner North West',
 'Inner South',
 'Inner West',
 'Outer East',
 'Outer North East',
 'Outer North West',
 'Outer South',
 'Outer West']
```

In [38]:
```
obesity=obesity[obesity['geography'].isin (regions)]
obesity.head()
```

Out[38]:

	geography_type	geography	auditdate	age_range	Sum of dsr	Sum of lower95CI	Su uppe
0	Community Committee	Inner East	201307	16+	27352.76	26881.6769	27829
1	Community Committee	Inner East	201310	16+	27277.53	26808.4482	27752
2	Community Committee	Inner East	201401	16+	27266.77	26796.3394	27743
3	Community Committee	Inner East	201404	16+	27264.18	26798.2996	27735
4	Community Committee	Inner East	201407	16+	27264.00	26798.9098	27734

In [39]:
```
# Depending on different areas, we can get the trend of the obesity rate
fig, ax = plt.subplots(figsize=(20, 10))

for name, group in obesity.groupby('geography'):
    group.plot(x='auditdate', y='Sum of dsr', ax=ax, label=name)

plt.title('Sum of dsr over Time for Different Geographies')
plt.xlabel('Date')
plt.ylabel('Sum of dsr')
plt.legend(title='Geography')
plt.grid(True)
plt.show()
```



To facilitate the analysis, we will calculate the average obesity rate of different areas in recent years as the obesity rate information of the area.

In [40]:
```
grouped_obesity= obesity.groupby('geography')['Sum of dsr'].mean().reset_index(n
grouped_obesity
```

Out[40]:

	geography	Average DSR
0	Inner East	28096.314054
1	Inner North East	20398.245135
2	Inner North West	19361.947027
3	Inner South	27160.301081
4	Inner West	26167.033514
5	Outer East	26699.624054
6	Outer North East	19376.057568
7	Outer North West	20205.503243
8	Outer South	25649.288919
9	Outer West	25341.790000

In [41]:
```
# To unify the geographic information with other dataset, we renamed the name of
grouped_obesity=grouped_obesity.rename(columns={'geography':'Nearest Area'})
grouped_obesity
```

Out[41]:

	Nearest Area	Average DSR
0	Inner East	28096.314054
1	Inner North East	20398.245135
2	Inner North West	19361.947027
3	Inner South	27160.301081
4	Inner West	26167.033514
5	Outer East	26699.624054
6	Outer North East	19376.057568
7	Outer North West	20205.503243
8	Outer South	25649.288919
9	Outer West	25341.790000

Similarly, the number of datasets in the relevant columns of the dataset is calculated depending on different areas.

In [42]:
```
grouped_restaurant = restaurant_type.groupby('Nearest Area')['Count'].sum().rese
grouped_restaurant.head()
```

Out[42]:

	Nearest Area	Total Count
0	Inner East	197
1	Inner West	10

Now, we would like to have an understanding of the correlation between different datasets.

The two datasets of restaurans and obesity rates were merged. We need to explore the correlation between obesity rates and the number of restaurans.

In [46]:
```
merged_rs_obesity = pd.merge(grouped_restaurant, grouped_obesity, on='Nearest Ar
merged_rs_obesity
```

Out[46]:

	Nearest Area	Total Count	Average DSR
0	Inner East	197	28096.314054
1	Inner West	10	26167.033514

In [47]:
```
correlation_rs_obesity = merged_rs_obesity['Average DSR'].corr(merged_rs_obesity
correlation_rs_obesity
```

Out[47]:
```
1.0
```

The two datasets of advertisements and obesity rates were merged. We now explore the correlation between obesity rates and the number of advertisements.

In [48]:
```
grouped_advert_cleaned= advert_cleaned.groupby('Nearest Area').size().reset_inde
grouped_advert_cleaned
```

Out[48]:

	Nearest Area	Count
0	Inner East	33
1	Inner North East	6
2	Inner North West	6
3	Inner South	16
4	Inner West	70
5	Outer East	62
6	Outer North West	6
7	Outer South	2
8	Outer West	35

In [49]:
```
merged_obesity_ad= pd.merge(grouped_obesity, grouped_advert_cleaned,on='Nearest
merged_obesity_ad
```

Out[49]:

	Nearest Area	Average DSR	Count
0	Inner East	28096.314054	33
1	Inner North East	20398.245135	6
2	Inner North West	19361.947027	6
3	Inner South	27160.301081	16
4	Inner West	26167.033514	70
5	Outer East	26699.624054	62
6	Outer North West	20205.503243	6
7	Outer South	25649.288919	2
8	Outer West	25341.790000	35

In [50]:
```
correlation_ad_obesity = merged_obesity_ad['Average DSR'].corr(merged_obesity_ad
correlation_ad_obesity
```

Out[50]:
```
0.5821196980237057
```

In [1]:
```
import pandas as pd
import numpy as np
```

In [63]:
```
df = pd.read_csv('Adult Obesity rates.csv')
df_ward_coordinates = pd.read_csv('infuse_ward_lyr_2011.csv')
```

In [64]:
```
df.head()
```

Out[64]:

	geography_type	geography	auditdate	age_range	Sum of dsr	Sum of lower95CI	Sum of upper95CI
0	Community Committee	Inner East	201307	16+	27352.76	26881.6769	27829.7892
1	Community Committee	Inner East	201310	16+	27277.53	26808.4482	27752.5204
2	Community Committee	Inner East	201401	16+	27266.77	26796.3394	27743.1488
3	Community Committee	Inner East	201404	16+	27264.18	26798.2996	27735.8898
4	Community Committee	Inner East	201407	16+	27264.00	26798.9098	27734.9115

In [66]:
```
df_ward = df[df['geography_type'] == 'Ward']
df_ward.head()
```

Out[66]:

	geography_type	geography	auditdate	age_range	Sum of dsr	Sum of lower95CI	Sum of upper95CI
1299	Ward	Adel & Wharfedale	201307	16+	18041.21	17356.2985	18745.5729
1300	Ward	Adel & Wharfedale	201310	16+	17991.38	17309.3400	18692.7368
1301	Ward	Adel & Wharfedale	201401	16+	17904.36	17225.9337	18602.0023
1302	Ward	Adel & Wharfedale	201404	16+	17869.03	17192.0308	18565.2096
1303	Ward	Adel & Wharfedale	201407	16+	18074.13	17394.4591	18772.9182

In [68]:
```
df_ward['obesity_rate'] = df_ward['Sum of dsr'] / 100000
```

```
C:\Users\USER\AppData\Local\Temp\ipykernel_24368\1039559739.py:1: SettingWithCopyWarning:
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
  df_ward['obesity_rate'] = df_ward['Sum of dsr'] / 100000
```

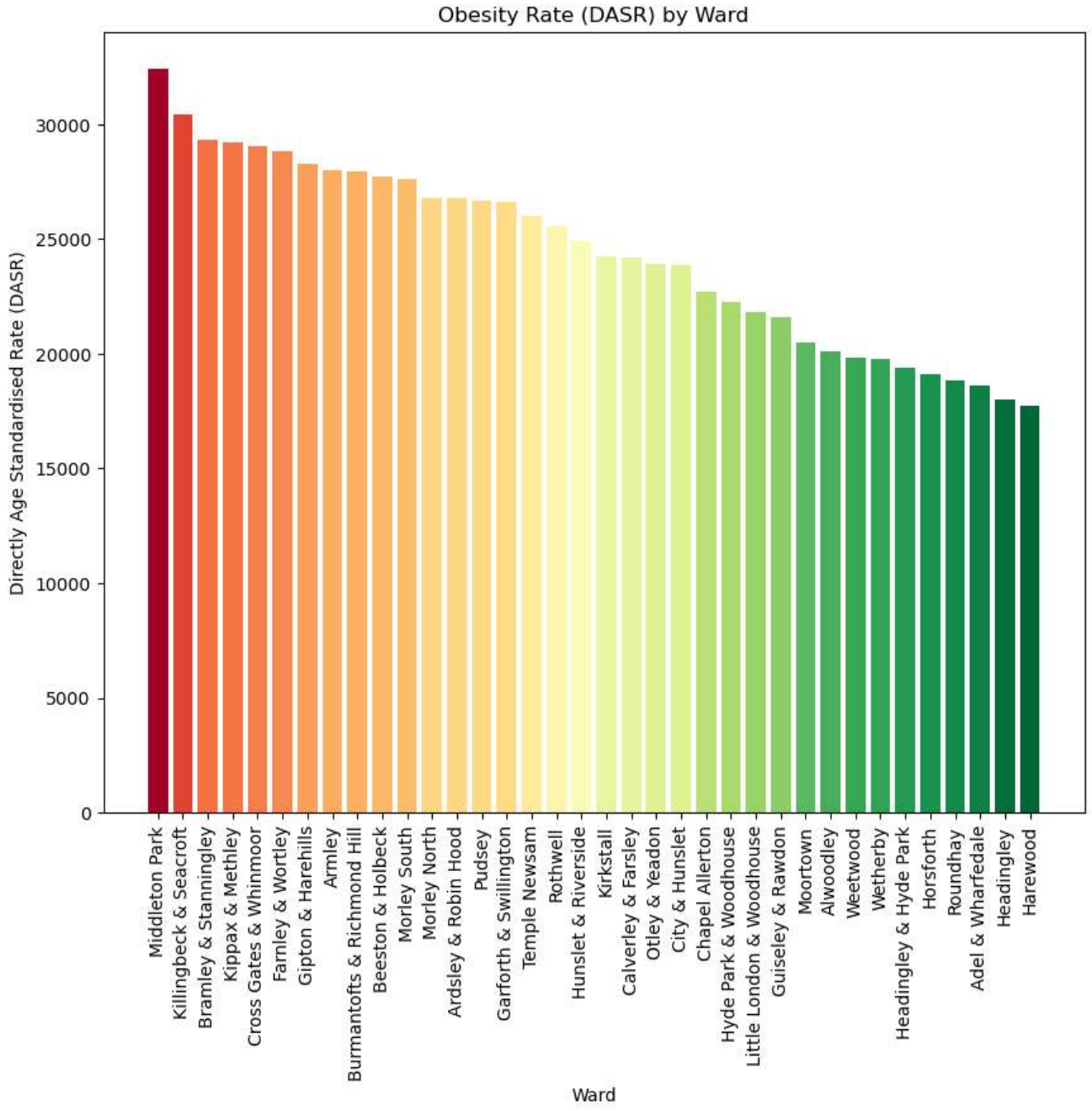
In [71]:
```
df_ward_latest_dates = df_ward.groupby('geography')['auditdate'].max().reset_index()

# 将原始DataFrame和含最新日期的DataFrame进行合并，以获取完整的记录
df_ward = pd.merge(df_ward, df_ward_latest_dates, on=['geography', 'auditdate'])
df_ward
```

Out[71]:

	geography_type	geography	auditdate	age_range	Sum of dsr	Sum of lower95CI	Sum of upper95CI	obesity_rate
0	Ward	Adel & Wharfedale	202301	16+	18594.18	17939.8869	19265.9155	0.185942
1	Ward	Alwoodley	202301	16+	20112.04	19465.6129	20774.2893	0.201120
2	Ward	Ardsley & Robin Hood	202301	16+	26766.44	26029.4263	27518.8761	0.267664
3	Ward	Armley	202301	16+	27997.35	27214.1285	28796.6859	0.279973
4	Ward	Beeston & Holbeck	202301	16+	27750.22	26989.7096	28525.9581	0.277502
5	Ward	Bramley & Stanningley	202301	16+	29332.13	28548.0116	30132.0916	0.293321
6	Ward	Burmantofts & Richmond Hill	202301	16+	27964.05	27174.9287	28769.3368	0.279641
7	Ward	Calverley & Farsley	202301	16+	24173.08	23478.0065	24883.4030	0.241731
8	Ward	Chapel Allerton	202301	16+	22695.51	22010.3232	23395.9199	0.226955
9	Ward	City & Hunslet	201804	16+	23849.37	23037.2428	24678.1451	0.238494
10	Ward	Cross Gates & Whinmoor	202301	16+	29036.69	28275.6360	29812.9355	0.290367
11	Ward	Farnley & Wortley	202301	16+	28824.25	28076.3695	29586.8733	0.288243
12	Ward	Garforth & Swillington	202301	16+	26625.45	25859.7335	27407.7832	0.266255
13	Ward	Gipton & Harehills	202301	16+	28303.95	27524.7592	29097.8919	0.283039
14	Ward	Guiseley & Rawdon	202301	16+	21572.78	20924.9557	22235.4797	0.215728
15	Ward	Harewood	202301	16+	17758.63	17099.6613	18435.7230	0.177586
16	Ward	Headingley	201804	16+	17983.98	16759.5961	19254.3084	0.179840
17	Ward	Headingley & Hyde Park	202301	16+	19406.72	18416.9993	20425.0920	0.194067
18	Ward	Horsforth	202301	16+	19098.92	18437.9845	19777.3189	0.190989
19	Ward	Hunslet & Riverside	202301	16+	24896.28	24079.5160	25731.1482	0.248963
20	Ward	Hyde Park & Woodhouse	201804	16+	22241.35	21192.2497	23317.7293	0.222413
21	Ward	Killingbeck & Seacroft	202301	16+	30414.46	29647.7714	31195.7455	0.304145
22	Ward	Kippax & Methley	202301	16+	29221.46	28304.7048	30160.2227	0.292215
23	Ward	Kirkstall	202301	16+	24245.98	23422.5389	25089.5224	0.242460
24	Ward	Little London & Woodhouse	202301	16+	21827.60	20716.7011	22965.2199	0.218276
25	Ward	Middleton Park	202301	16+	32398.85	31637.6093	33173.4807	0.323989
26	Ward	Moortown	202301	16+	20511.12	19851.5446	21186.8640	0.205111
27	Ward	Morley North	202301	16+	26800.64	26080.7214	27535.3293	0.268006
28	Ward	Morley South	202301	16+	27630.82	26887.9739	28388.8765	0.276308
29	Ward	Otley & Yeadon	202301	16+	23908.64	23210.7661	24621.8438	0.239086
30	Ward	Pudsey	202301	16+	26689.94	26026.6982	27365.7513	0.266899
31	Ward	Rothwell	202301	16+	25555.90	24794.2434	26334.8964	0.255559
32	Ward	Roundhay	202301	16+	18813.95	18199.9925	19443.2414	0.188140

	geography_type	geography	auditdate	age_range	Sum of dsr	Sum of lower95CI	Sum of upper95CI	obesity_rate
33	Ward	Temple Newsam	202301	16+	25994.80	25254.0093	26751.7307	0.259948
34	Ward	Weetwood	202301	16+	19835.51	19132.0442	20558.0002	0.198355
35	Ward	Wetherby	202301	16+	19753.58	19073.6806	20450.7257	0.197536

In [79]:
```
import matplotlib.pyplot as plt
import matplotlib.cm as cm

# Sort the data by obesity rate in descending order
df_sorted = df_ward.sort_values(by='Sum of dsr', ascending=False)

# Create a color map from green to red
norm = plt.Normalize(df_sorted['Sum of dsr'].min(), df_sorted['Sum of dsr'].max())
cmap = cm.RdYlGn_r  # Red to Green reversed color map
colors = cmap(norm(df_sorted['Sum of dsr']))  # Apply normalization and colormap to obesity rates

# Plot the chart
plt.figure(figsize=(10, 8))  # Adjust the size of the figure
plt.bar(df_sorted['geography'], df_sorted['Sum of dsr'], color=colors)  # Assuming the geography names are in the 'geography' column
plt.xlabel('Ward')  # Label for the X-axis
plt.ylabel('Directly Age Standardised Rate (DASR)')  # Label for the Y-axis
plt.title('Obesity Rate (DASR) by Ward')  # Title of the chart
plt.xticks(rotation=90)  # Rotate labels for better readability if geography names are long
plt.show()
```


Obesity Rate (DASR) by Ward
Directly Age Standardised Rate (DASR)
0
5000
10000
15000
20000
25000
30000
Middleton Park
Killingbeck & Seacroft
Bramley & Stanningley
Kippax & Methley
Cross Gates & Whinmoor
Farnley & Wortley
Gipton & Harehills
Armley
Burmantofts & Richmond Hill
Beeston & Holbeck
Morley South
Morley North
Ardsley & Robin Hood
Pudsey
Garforth & Swillington
Temple Newsam
Rothwell
Hunslet & Riverside
Kirkstall
Calverley & Farsley
Otley & Yeadon
City & Hunslet
Chapel Allerton
Hyde Park & Woodhouse
Little London & Woodhouse
Guiseley & Rawdon
Moortown
Alwoodley
Weetwood
Wetherby
Headingley & Hyde Park
Horsforth
Roundhay
Adel & Wharfedale
Headingley
Harewood
Ward

In [2]:
```
!pip install osmnx
```

```
Collecting osmnx
  Obtaining dependency information for osmnx from https://files.pythonhosted.org/packages/5e/ab/2e29d26454a8a9bdedb7a9be8920c6a9b88e5e6caae15cbba0d50a134697/osmnx-1.9.1-py3-none
-any.whl.metadata
  Downloading osmnx-1.9.1-py3-none-any.whl.metadata (4.9 kB)
Requirement already satisfied: geopandas>=0.12 in d:\software\anaconda\anaconda3\lib\site-packages (from osmnx) (0.14.1)
Requirement already satisfied: networkx>=2.5 in d:\software\anaconda\anaconda3\lib\site-packages (from osmnx) (3.1)
Requirement already satisfied: numpy>=1.20 in d:\software\anaconda\anaconda3\lib\site-packages (from osmnx) (1.24.3)
Requirement already satisfied: pandas>=1.1 in d:\software\anaconda\anaconda3\lib\site-packages (from osmnx) (2.0.3)
Requirement already satisfied: requests>=2.27 in d:\software\anaconda\anaconda3\lib\site-packages (from osmnx) (2.31.0)
Requirement already satisfied: shapely>=2.0 in d:\software\anaconda\anaconda3\lib\site-packages (from osmnx) (2.0.2)
Requirement already satisfied: fiona>=1.8.21 in d:\software\anaconda\anaconda3\lib\site-packages (from geopandas>=0.12->osmnx) (1.9.5)
Requirement already satisfied: packaging in d:\software\anaconda\anaconda3\lib\site-packages (from geopandas>=0.12->osmnx) (23.1)
Requirement already satisfied: pyproj>=3.3.0 in d:\software\anaconda\anaconda3\lib\site-packages (from geopandas>=0.12->osmnx) (3.6.1)
Requirement already satisfied: python-dateutil>=2.8.2 in d:\software\anaconda\anaconda3\lib\site-packages (from pandas>=1.1->osmnx) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in d:\software\anaconda\anaconda3\lib\site-packages (from pandas>=1.1->osmnx) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in d:\software\anaconda\anaconda3\lib\site-packages (from pandas>=1.1->osmnx) (2023.3)
Requirement already satisfied: charset-normalizer<4,>=2 in d:\software\anaconda\anaconda3\lib\site-packages (from requests>=2.27->osmnx) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in d:\software\anaconda\anaconda3\lib\site-packages (from requests>=2.27->osmnx) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in d:\software\anaconda\anaconda3\lib\site-packages (from requests>=2.27->osmnx) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in d:\software\anaconda\anaconda3\lib\site-packages (from requests>=2.27->osmnx) (2023.11.17)
Requirement already satisfied: attrs>=19.2.0 in d:\software\anaconda\anaconda3\lib\site-packages (from fiona>=1.8.21->geopandas>=0.12->osmnx) (22.1.0)
Requirement already satisfied: click~=8.0 in d:\software\anaconda\anaconda3\lib\site-packages (from fiona>=1.8.21->geopandas>=0.12->osmnx) (8.0.4)
Requirement already satisfied: click-plugins>=1.0 in d:\software\anaconda\anaconda3\lib\site-packages (from fiona>=1.8.21->geopandas>=0.12->osmnx) (1.1.1)
Requirement already satisfied: cligj>=0.5 in d:\software\anaconda\anaconda3\lib\site-packages (from fiona>=1.8.21->geopandas>=0.12->osmnx) (0.7.2)
Requirement already satisfied: six in d:\software\anaconda\anaconda3\lib\site-packages (from fiona>=1.8.21->geopandas>=0.12->osmnx) (1.16.0)
Requirement already satisfied: setuptools in d:\software\anaconda\anaconda3\lib\site-packages (from fiona>=1.8.21->geopandas>=0.12->osmnx) (68.0.0)
Requirement already satisfied: colorama in d:\software\anaconda\anaconda3\lib\site-packages (from click~=8.0->fiona>=1.8.21->geopandas>=0.12->osmnx) (0.4.6)
Downloading osmnx-1.9.1-py3-none-any.whl (104 kB)
   ---------------------------------------- 0.0/104.3 kB ? eta -:--:--
   --- ------------------------------------ 10.2/104.3 kB ? eta -:--:--
   ---------------------------------------- 104.3/104.3 kB 2.0 MB/s eta 0:00:00
Installing collected packages: osmnx
Successfully installed osmnx-1.9.1
```

In [5]:
```
import osmnx as ox

# Define the area and query tags
place = 'Leeds, England'  # Example city
tags = {'amenity': 'fast_food'}  # Querying for fast food restaurants

# Use osmnx to get fast food restaurant data
fast_food_restaurants = ox.geometries_from_place(place, tags)

# Save the data to a CSV file
fast_food_restaurants.to_csv('fast_food_restaurants_leeds.csv', index=False)

print("Data has been saved to 'fast_food_restaurants_leeds.csv'")
```

```
C:\Users\USER\AppData\Local\Temp\ipykernel_31668\4004164512.py:8: FutureWarning: The `geometries` module and `geometries_from_X` functions have been renamed the `features` modul
e and `features_from_X` functions. Use these instead. The `geometries` module and function names are deprecated and will be removed in the v2.0.0 release.
  fast_food_restaurants = ox.geometries_from_place(place, tags)
Data has been saved to 'fast_food_restaurants_leeds.csv'
```

In [4]:
```
fast_food_restaurants.head()
```

Out[4]:

element_type	osmid	addr:city	addr:housenumber	addr:postcode	addr:street	amenity	cuisine	fhrs:id	name	opening_hours	takeaway	...	source:building	happycow:id	building:part	disused:sl
node	27476846	Wetherby	2	LS22 6LE	Castle Gate	fast_food	pizza	320834	Moji's	Mo-Su 17:00-24:00	only	...	NaN	NaN	NaN	N
	28096528	Otley	9	LS21 3AB	Bondgate	fast_food	fish_and_chips	1595369	Uncle Joe's	NaN	NaN	...	NaN	NaN	NaN	N
	28098318	Otley	60	LS21 1BT	Leeds Road	fast_food	chinese	1544482	Super Wok	NaN	NaN	...	NaN	NaN	NaN	N
	28903051	Otley	14	LS21 3AE	Newmarket	fast_food	pizza	1601682	Impero Pizza	NaN	yes	...	NaN	NaN	NaN	N
	29047757	Leeds	137A	LS20 8LY	Otley Road	fast_food	sandwich	323001	Subway	Mo-Th 08:00-19:00; Fr,Sa 08:00-20:00; Su 09:00...	yes	...	NaN	NaN	NaN	N

5 rows × 107 columns

In [9]:

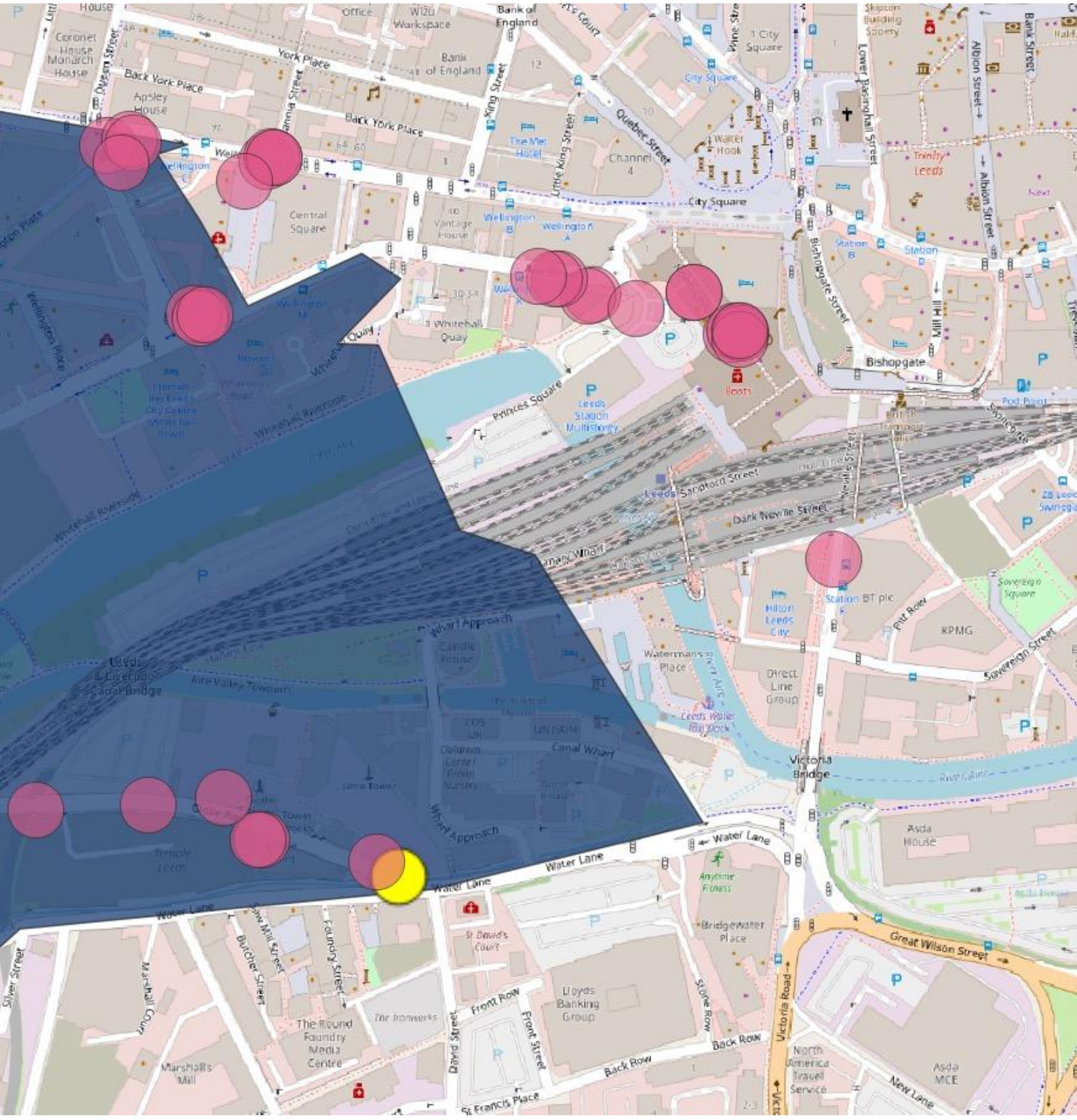
```
import osmnx as ox
import geopandas as gpd
import pandas as pd

# Define the area of interest
place = 'Leeds, England'

# Define the queries for the different amenities and shops
queries = {
    'fast_food': {'amenity': 'fast_food'},
    'confectionery': {'shop': 'confectionery'},
    'chocolate': {'shop': 'chocolate'},
    'wine': {'shop': 'wine'},
    'marketplace': {'amenity': 'marketplace'},
    'supermarket': {'shop': 'supermarket'},
    'convenience': {'shop': 'convenience'},
    'health_food': {'shop': 'health_food'},
    'greengrocer': {'shop': 'greengrocer'}
}

# Initialize a list to hold the dataframes
all_dataframes = []

for label, tags in queries.items():
    try:
        # Retrieve data for the current query
        data = ox.geometries_from_place(place, tags)
        # Add a column to indicate the type of amenity or shop for each entry
        data['category'] = label
        # Append the results to the list
        all_dataframes.append(data)
    except Exception as e:
        print(f"An error occurred while retrieving {label}: {e}")
```

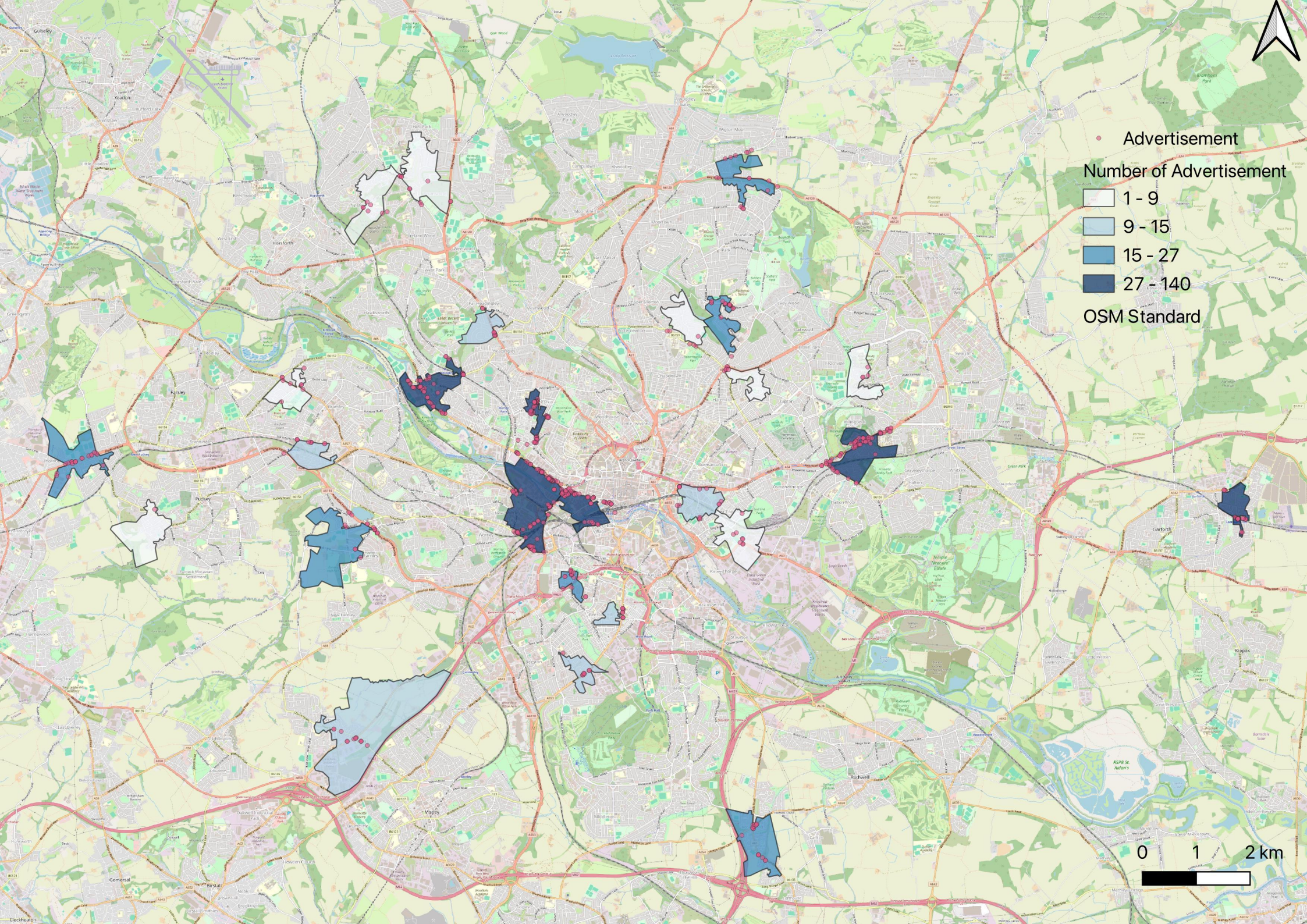
```python
# Check if we have any data to concatenate
if all_dataframes:
    # Concatenate all dataframes into one GeoDataFrame
    all_data = pd.concat(all_dataframes, ignore_index=True)

    # Save the combined data to a single CSV file
    all_data.to_csv('amenities_shops_leeds.csv', index=False)

    print("Data has been saved to 'amenities_shops_leeds.csv'")
else:
    print("No data was retrieved.")
```

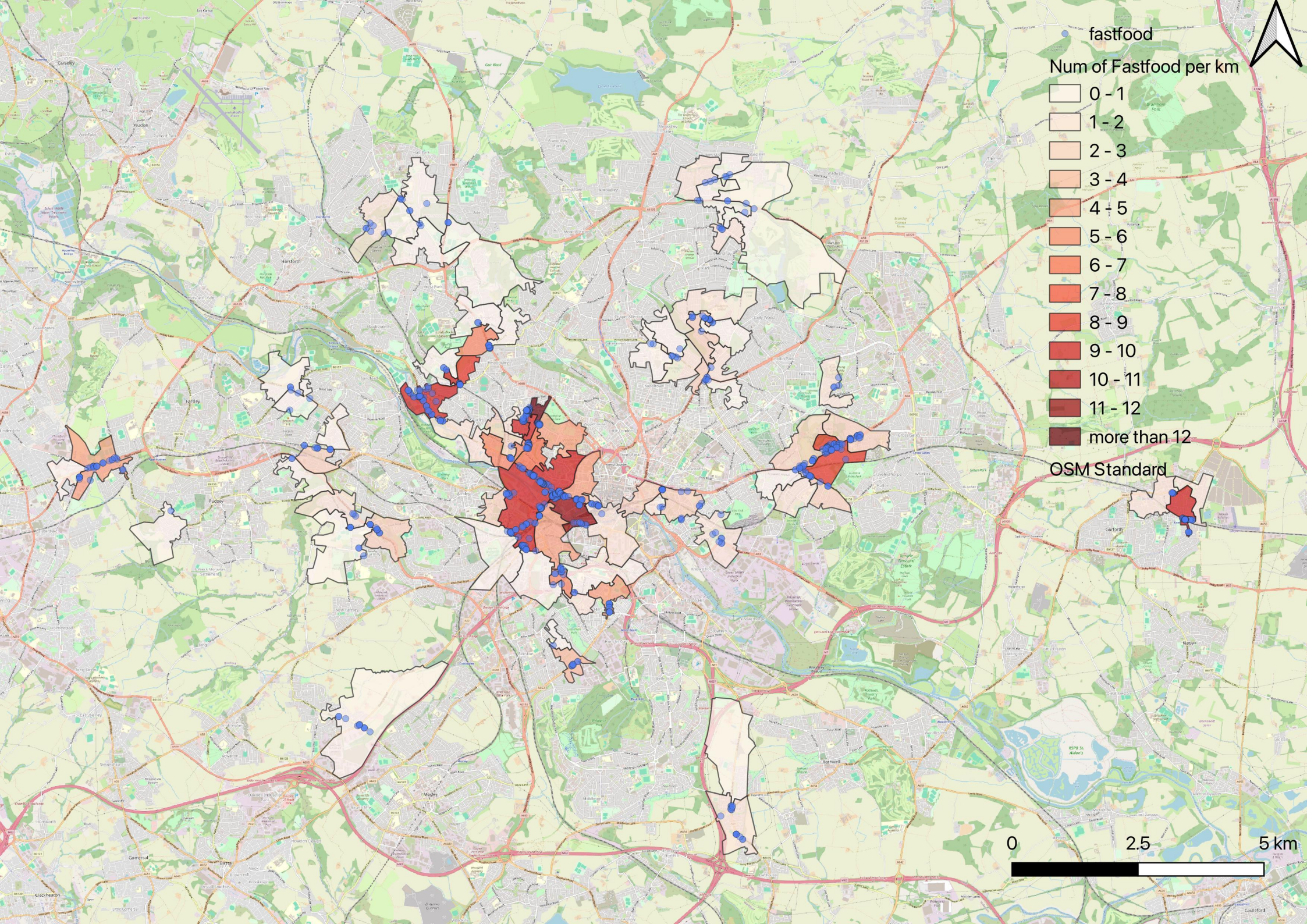
```
C:\Users\USER\AppData\Local\Temp\ipykernel_24144\1354263997.py:27: FutureWarning: The `geometries` module and `geometries_from_X` functions have been renamed the `features` module and `features_from_X` functions. Use these instead. The `geometries` module and function names are deprecated and will be removed in the v2.0.0 release.
  data = ox.geometries_from_place(place, tags)
C:\Users\USER\AppData\Local\Temp\ipykernel_24144\1354263997.py:27: FutureWarning: The `geometries` module and `geometries_from_X` functions have been renamed the `features` module and `features_from_X` functions. Use these instead. The `geometries` module and function names are deprecated and will be removed in the v2.0.0 release.
  data = ox.geometries_from_place(place, tags)
C:\Users\USER\AppData\Local\Temp\ipykernel_24144\1354263997.py:27: FutureWarning: The `geometries` module and `geometries_from_X` functions have been renamed the `features` module and `features_from_X` functions. Use these instead. The `geometries` module and function names are deprecated and will be removed in the v2.0.0 release.
  data = ox.geometries_from_place(place, tags)
C:\Users\USER\AppData\Local\Temp\ipykernel_24144\1354263997.py:27: FutureWarning: The `geometries` module and `geometries_from_X` functions have been renamed the `features` module and `features_from_X` functions. Use these instead. The `geometries` module and function names are deprecated and will be removed in the v2.0.0 release.
  data = ox.geometries_from_place(place, tags)
C:\Users\USER\AppData\Local\Temp\ipykernel_24144\1354263997.py:27: FutureWarning: The `geometries` module and `geometries_from_X` functions have been renamed the `features` module and `features_from_X` functions. Use these instead. The `geometries` module and function names are deprecated and will be removed in the v2.0.0 release.
  data = ox.geometries_from_place(place, tags)
C:\Users\USER\AppData\Local\Temp\ipykernel_24144\1354263997.py:27: FutureWarning: The `geometries` module and `geometries_from_X` functions have been renamed the `features` module and `features_from_X` functions. Use these instead. The `geometries` module and function names are deprecated and will be removed in the v2.0.0 release.
  data = ox.geometries_from_place(place, tags)
C:\Users\USER\AppData\Local\Temp\ipykernel_24144\1354263997.py:27: FutureWarning: The `geometries` module and `geometries_from_X` functions have been renamed the `features` module and `features_from_X` functions. Use these instead. The `geometries` module and function names are deprecated and will be removed in the v2.0.0 release.
  data = ox.geometries_from_place(place, tags)
C:\Users\USER\AppData\Local\Temp\ipykernel_24144\1354263997.py:27: FutureWarning: The `geometries` module and `geometries_from_X` functions have been renamed the `features` module and `features_from_X` functions. Use these instead. The `geometries` module and function names are deprecated and will be removed in the v2.0.0 release.
  data = ox.geometries_from_place(place, tags)
C:\Users\USER\AppData\Local\Temp\ipykernel_24144\1354263997.py:27: FutureWarning: The `geometries` module and `geometries_from_X` functions have been renamed the `features` module and `features_from_X` functions. Use these instead. The `geometries` module and function names are deprecated and will be removed in the v2.0.0 release.
  data = ox.geometries_from_place(place, tags)
Data has been saved to 'amenities_shops_leeds.csv'
```

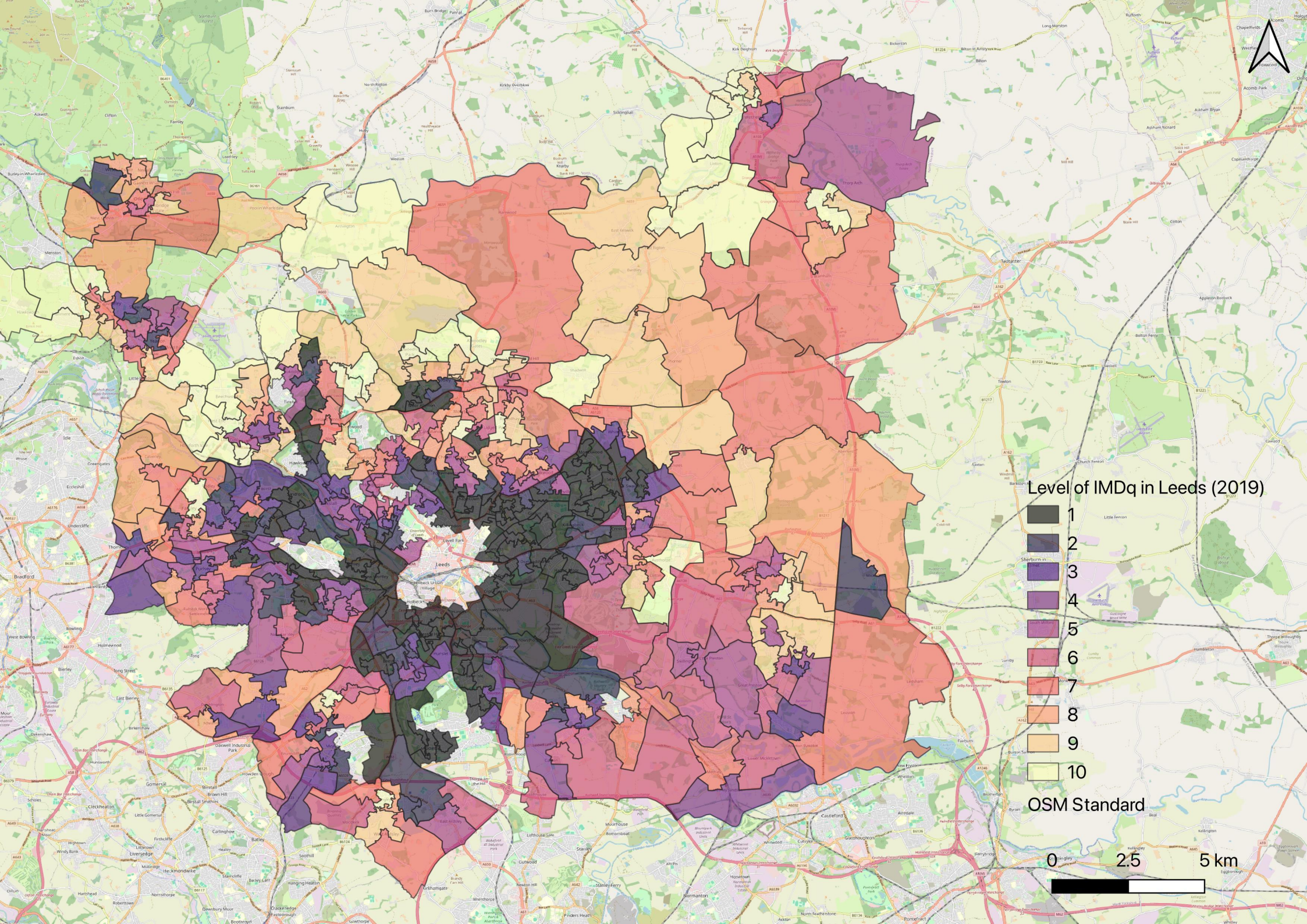
In [ ]:

The last four pictures are generated by QGIS




Advertisement
Number of Advertisement
1 - 9
9 - 15
15 - 27
27 - 140
OSM Standard
0
1
2 km


fastfood
Num of Fastfood per km
0 - 1
1 - 2
2 - 3
3 - 4
4 - 5
5 - 6
6 - 7
7 - 8
8 - 9
9 - 10
10 - 11
11 - 12
more than 12
OSM Standard
0
2.5
5 km


Level of IMDq in Leeds (2019)
1
2
3
4
5
6
7
8
9
10
OSM Standard
0
2.5
5 km