Coursework Coversheet

School of Geography. FACULTY OF ENVIRONMENT

GEOG5403 Creative Coding for Urban Problems

UNIVERSITY OF LEEDS

Student ID	201413609, 201760120, 201650287, 201348673, 201788996, 201762009.	Mark	
Group Number	Group 9	Less deduction (state reason)	
Assignment title	Exploring the Impact of Proximity, Accessibility and Active Travel on Physical Activity	Final Mark	
Marker	Dr. Jiaqi Ge		

Justification of mark (using specific text from criteria):

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

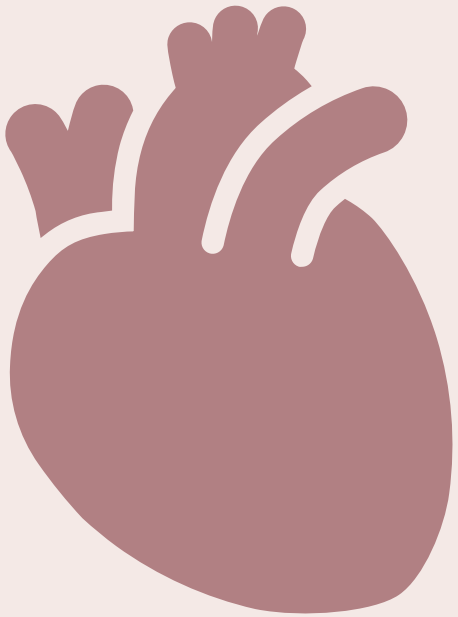
Additional comments:

EXPLORING THE IMPACT OF PROXIMITY, ACCESSIBILITY AND ACTIVE TRAVEL ON PHYSICAL ACTIVITY

Group 9
Jazmin Mzera, Jingyi Dou, Kuangyi Pei,
Mengzhu He, Oladeji Olajubu, Zhihao Zhang

BACKGROUND

- In today's world, combating chronic diseases and unhealthy lifestyles is a major public health challenge (CDC, 2023).
- To address this, communities are promoting physical activity. Facilities like sports centres and parks play a crucial role in encouraging physical activity (Cohen et al., 2007).
- Yet, their impact depends on proximity and accessibility, which influence people's exercise habits. Analyzing these factors helps understand participation levels and travel convenience.



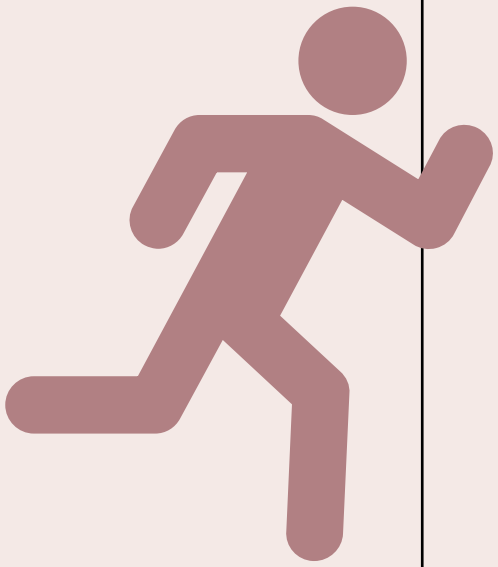
BACKGROUND

Proximity

- Research shows that distance to physical activity centres influences how often people engage in exercise. Living closer to these centres encourages regular physical activity (Jilcott et al., 2007).

Accessibility

- Accessibility is crucial for exercise participation. It includes physical distance and transportation options.
- Better accessibility, such as public transport and bike lanes, encourages more people to use these facilities for exercise (Nikolajsen et al., 2021).
- Analyzing travel distances and transportation options helps assess centre accessibility (Broaddus & Pendall, 2012).



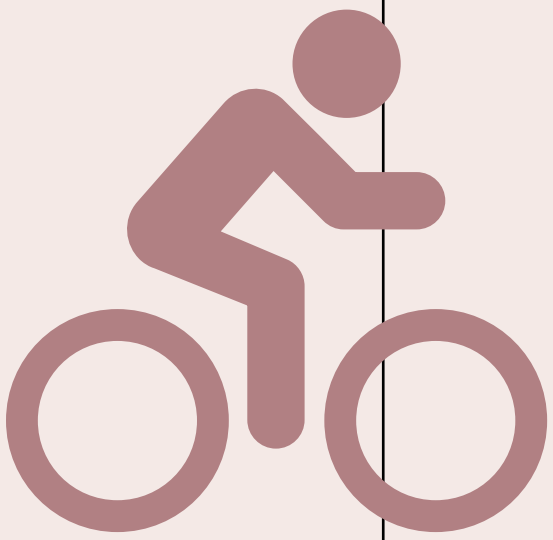
BACKGROUND

Importance of Active Travel Modes

- Active travel modes, such as walking and cycling, are vital for promoting physical activity and have positive effects on both individual health and the environment (WHO, 2022).

Significance for Urban Planning and Public Health Strategies

- Understanding the proximity and accessibility of physical activity centres is crucial for urban planners and public health policymakers.
- Strategically planning the locations of these centres can improve community health by providing easier access to physical activity facilities (CDC, 2012).
- Improving transportation infrastructure, including bike lanes and pedestrian-friendly routes, can further enhance access to these centres and encourage greater participation in physical exercise.



APPROACHES WITH EIGHT DOMAINS

- **Accessibility – 3 domains**

Investigate the density of parks, sports centres and fitness centres within 1km of each residential area in different areas of Leeds.

- **Proximity – 3 domains**

Investigate the distance of each residential area in different parts of Leeds to the nearest park, sports centres and fitness centres.

- **Mobility – 1 domain**

Investigate the transport facilities provided by each physical activity centre in different areas of Leeds, such as whether there is a car park, cycle park, bike hire, or repair workshop.

- **Disabled – 1 domain**

Find out what facilities each physical activity centre in different parts of Leeds has for people with disabilities, such as whether it has toilets for people with disabilities (disabled).

METHODS

Based on the values of the eight domains, we assign them a domain score ranging from 0 to 5 to facilitate the calculation of the final index. The following are the assignment rules:

Proximity	Score
Distance <= 500	5
500< Distance <= 1000	4
1000< Distance <= 1500	3
1500< Distance <= 2000	2
Distance > 2000	1

Accessibility	Score
Density >= 8	5
6<= Density < 8	4
4<= Density < 6	3
2<= Density < 4	2
Density < 2	1

Mobility	Score
Number of services == 4	5
Number of services == 3	4
Number of services == 2	3
Number of services == 1	2
Number of services == 0	1

Disabled	Score
Proportion of services <= 1.0	5
Proportion of services <= 0.8	4
Proportion of services <= 0.6	3
Proportion of services <= 0.4	2
Proportion of services <= 0.2	1

METHODS FOR CREATING INDEX

- Cohen et al. (2007) said parks promote physical activity in minority communities by providing facilities, supervised activities, and also by providing destinations to which people can walk, even if they are sedentary once there.
- According to Kaczynski and Henderson (2008), park density is positively associated with physical activity because people are more likely to visit parks when there are more options available.
- Giles-Corti and Donovan (2002) discovered that access to sports facilities is associated with increased physical activity, particularly for activities that require specialized spaces or equipment.

Domains	Weight
Proximity of parks	1/6
Accessibility of parks	1/6
Disabled	1/6
Proximity of fitness centres	1/10
Proximity of sports centres	1/10
Accessibility of fitness centres	1/10
Accessibility of sports centres	1/10
Mobility	1/10

DATA USED

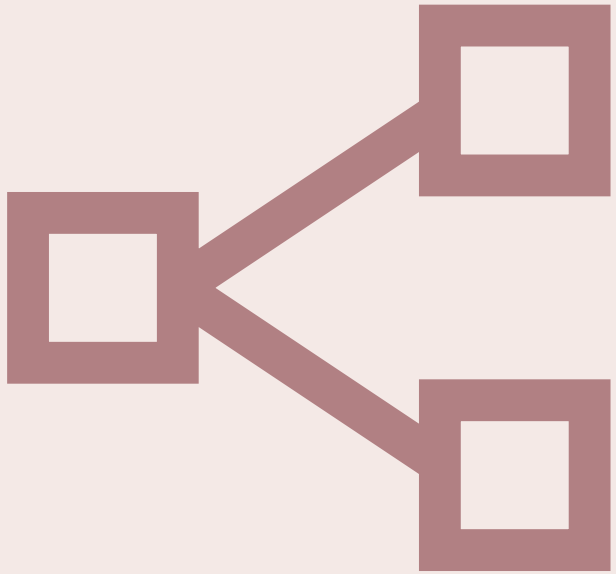
To carry out this analysis we will be using 6 datasets to explore the impact of proximity, accessibility and active travel on physical activity in Leeds.

1. **residential_leeds.csv;**
2. **parks_leeds.csv;**
3. **centre_leeds.csv;**
4. **fitness_centre_leeds.csv**

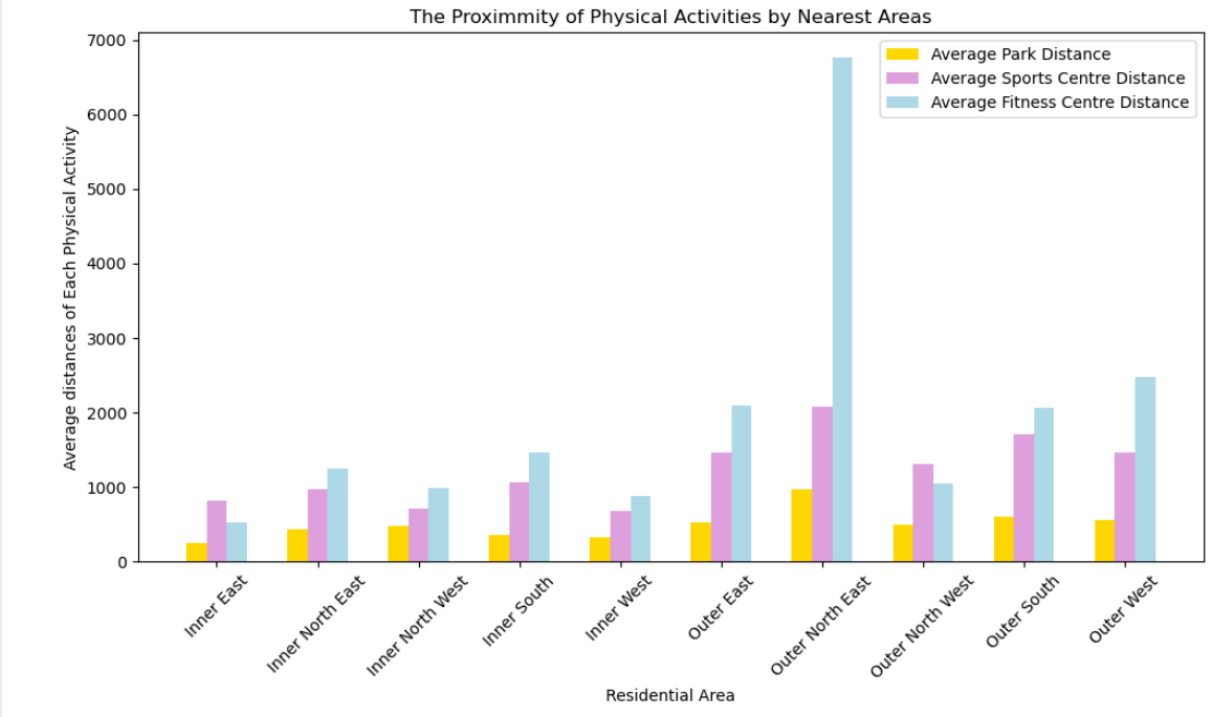
We utilize the above four datasets to determine the latitude and longitude of each residential area, park, fitness centre, and sports centre in Leeds.

1. **Geographic.csv**
2. **sites.csv**

We utilize the above two datasets to determine the information on the facilities for the disabled, mobility, latitude, and longitude of each physical activity in Leeds.



AVERAGE DISTANCE TO PARKS, SPORTS CENTERS AND FITNESS CENTERS



- The Inner East Area has the shortest average distance to parks, the Inner West Area is closest to sports centres, and the Inner East Area again leads in proximity to fitness centres. Conversely, the Outer North-East Area shows the longest distances for all three types of facilities.

- These findings underscore differences in physical activity accessibility among regions. Residents in Inner East, Inner West, and Inner North-West Areas enjoy easier access, while those in Outer North-East, Outer West, and Outer South Areas face greater distances. Highlighting disparities in accessibility.

AVERAGE DENSITY OF PARKS, SPORTS CENTERS AND FITNESS CENTERS

Parks:

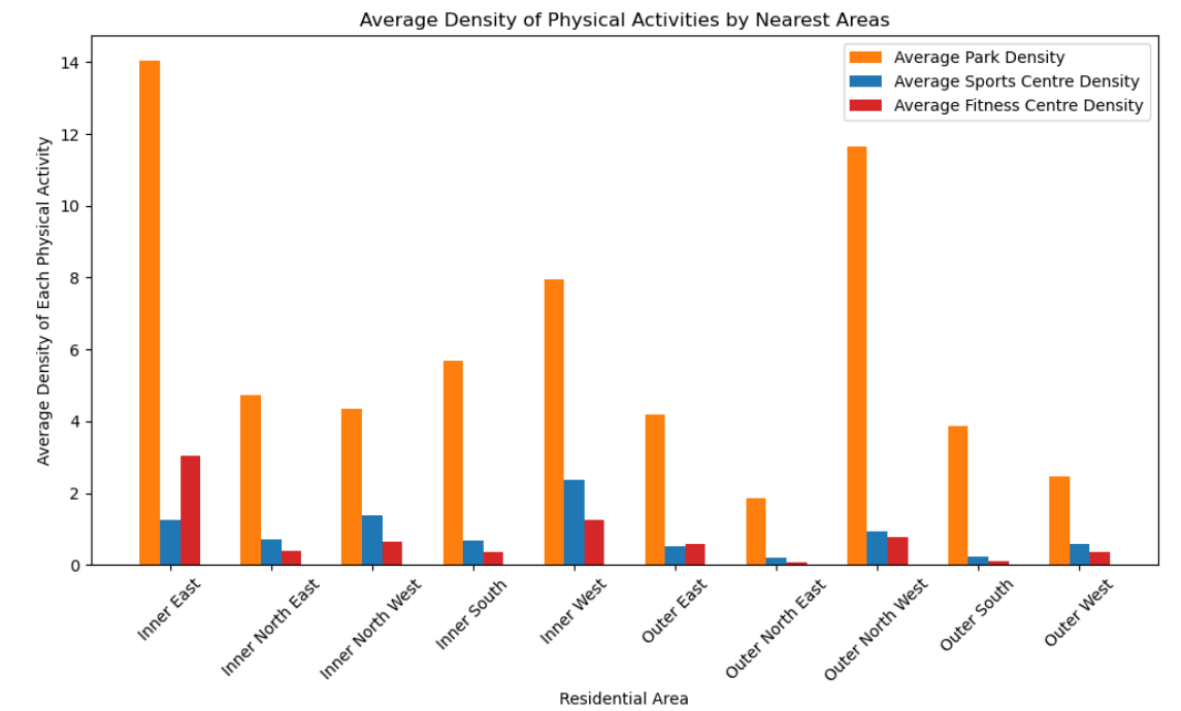
- Inner East, Outer North-West, and Inner West Areas show the highest park density, providing residents with abundant leisure options.
- Conversely, Outer North-East and Outer West Areas exhibit the lowest park density, potentially necessitating longer travel for outdoor activities.

Sports Centres:

- Inner West Area boasts the highest density, offering ample opportunities for physical activity.
- Outer North-East and Outer South Areas have lower densities, indicating limited access to sports facilities.

Fitness Centres:

- Inner East Area has the highest density, suggesting easier access to fitness facilities.
- Outer North-East and Outer South Areas show lower densities, potentially requiring longer commutes for fitness options.



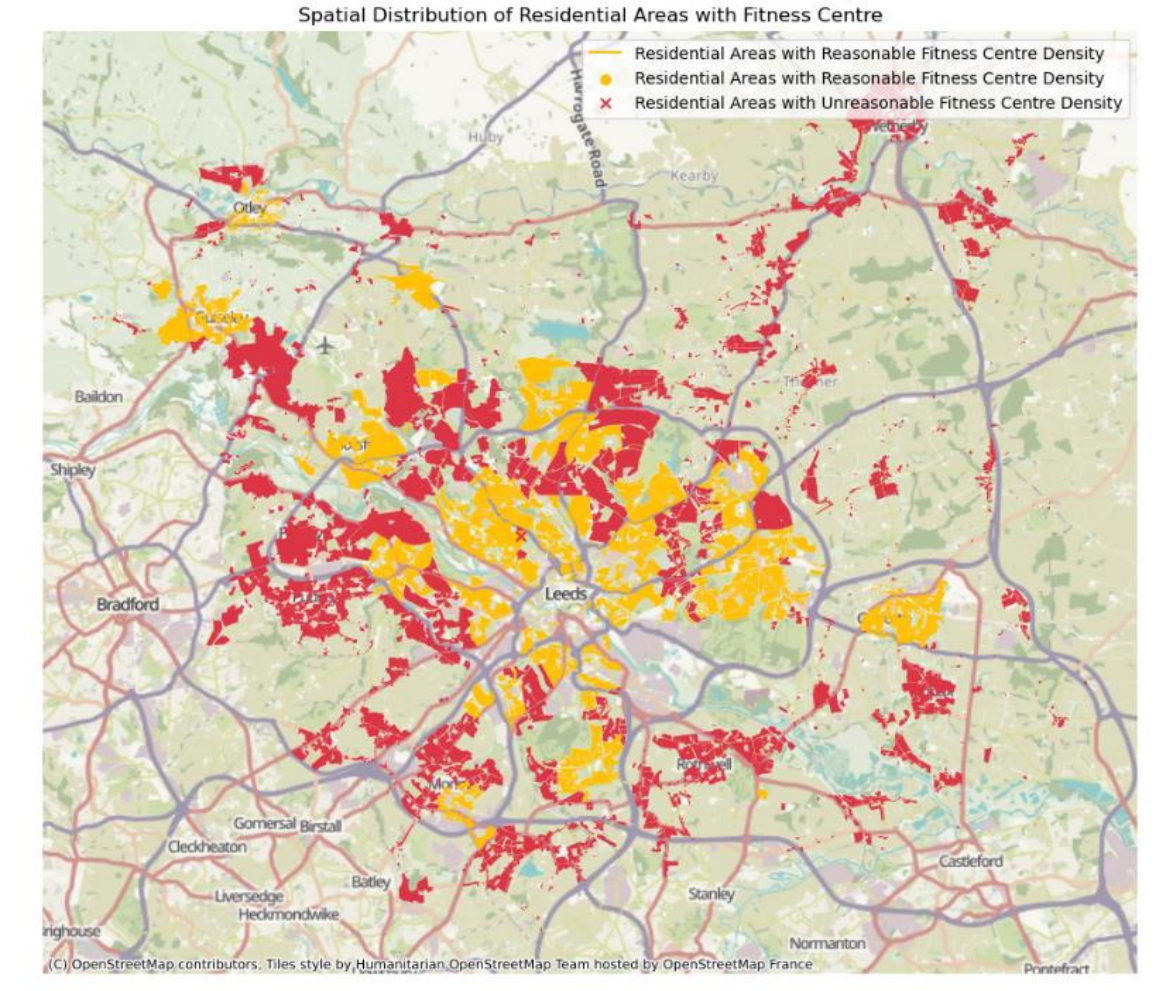
SPATIAL DISTRIBUTION OF RESIDENTIAL AREAS WITH FITNESS CENTERS

1. Distribution

- Residential areas with an inadequate density of fitness centres are notably abundant, primarily clustered in the outskirts of the urban centre.
- Residential areas with a sufficient density of fitness centres concentrated in the urban centre and spreading outward.

2. Implications for Urban Planning

- Urban planners and policymakers may need to focus on expanding fitness facilities in these regions with inadequate fitness centre density to improve residents' health and well-being.
- In areas with sufficient density, efforts can be directed towards enhancing the management and quality of fitness centres, ensuring residents can fully utilize these resources to improve the overall health of the city.



SPATIAL DISTRIBUTION OF RESIDENTIAL AREAS WITH PARKS

1. Widespread Park Distribution

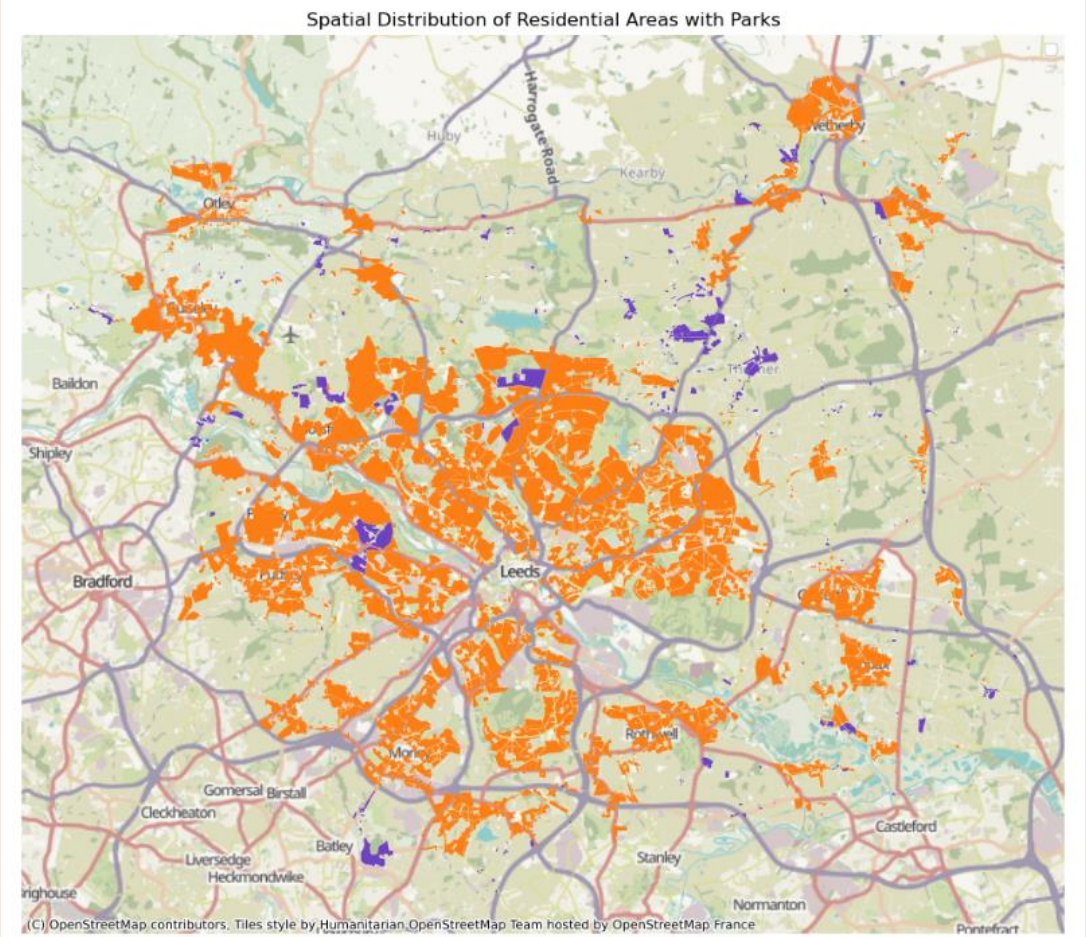
- Residential areas with ample park density are spread throughout the urban landscape, indicating a plentiful supply across the city.

2. Disparity in Park Density

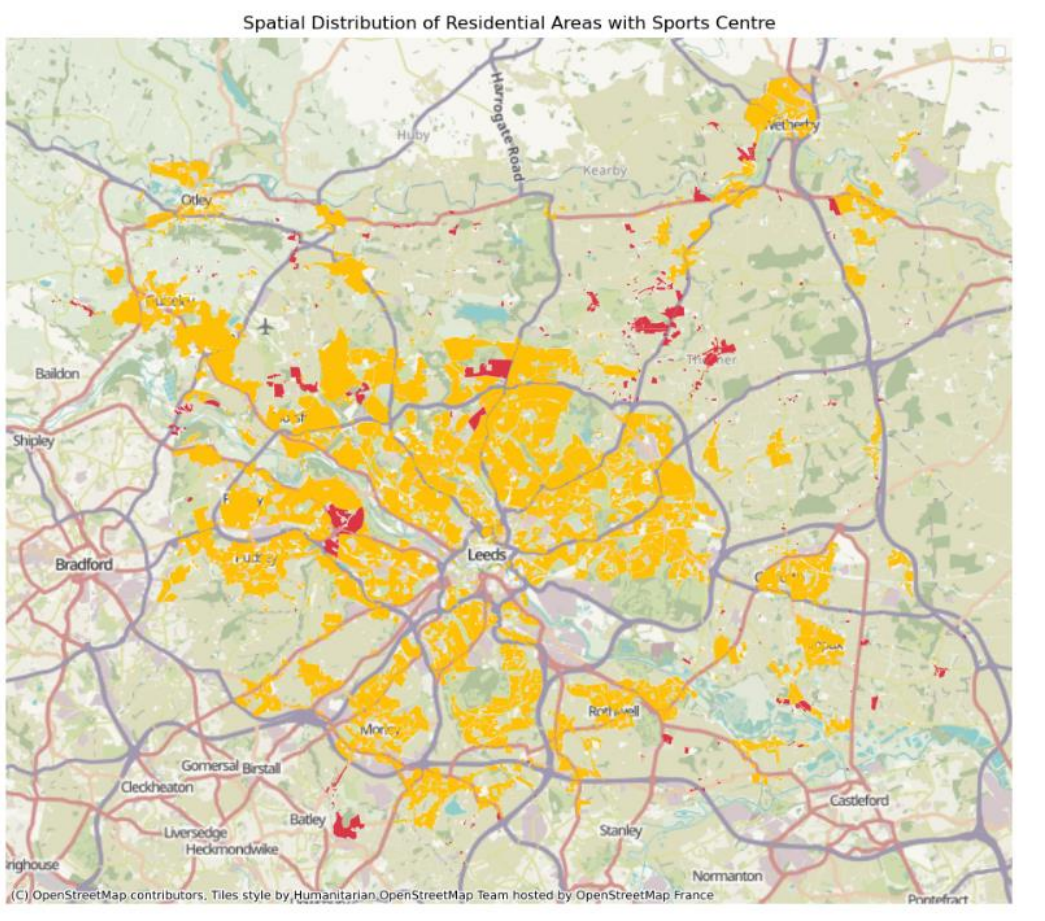
- Areas with reasonable park density are found across diverse city regions, while those with insufficient density may be limited and concentrated in specific locales.

3. Implications for Urban Planning

- Urban planners should address areas lacking sufficient park density by implementing strategies to enhance green spaces. Measures like expanding park areas, upgrading facilities, and increasing overall green infrastructure can enrich residents' leisure options and promote both physical and mental well-being.

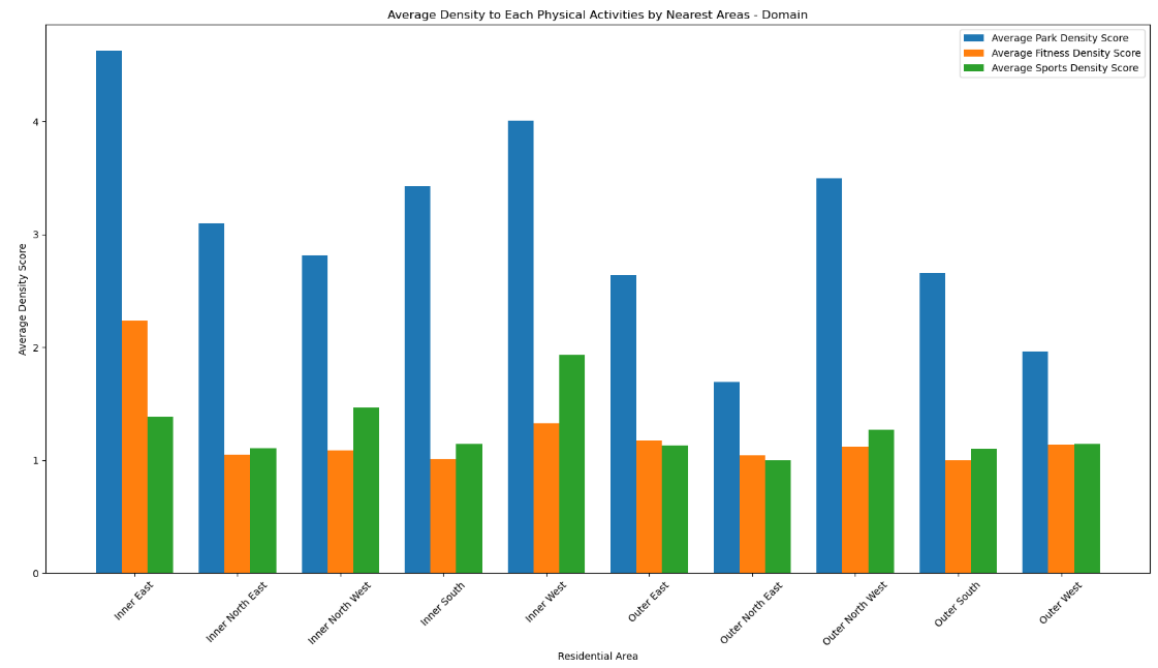


SPATIAL DISTRIBUTION OF RESIDENTIAL AREAS WITH SPORTS CENTERS



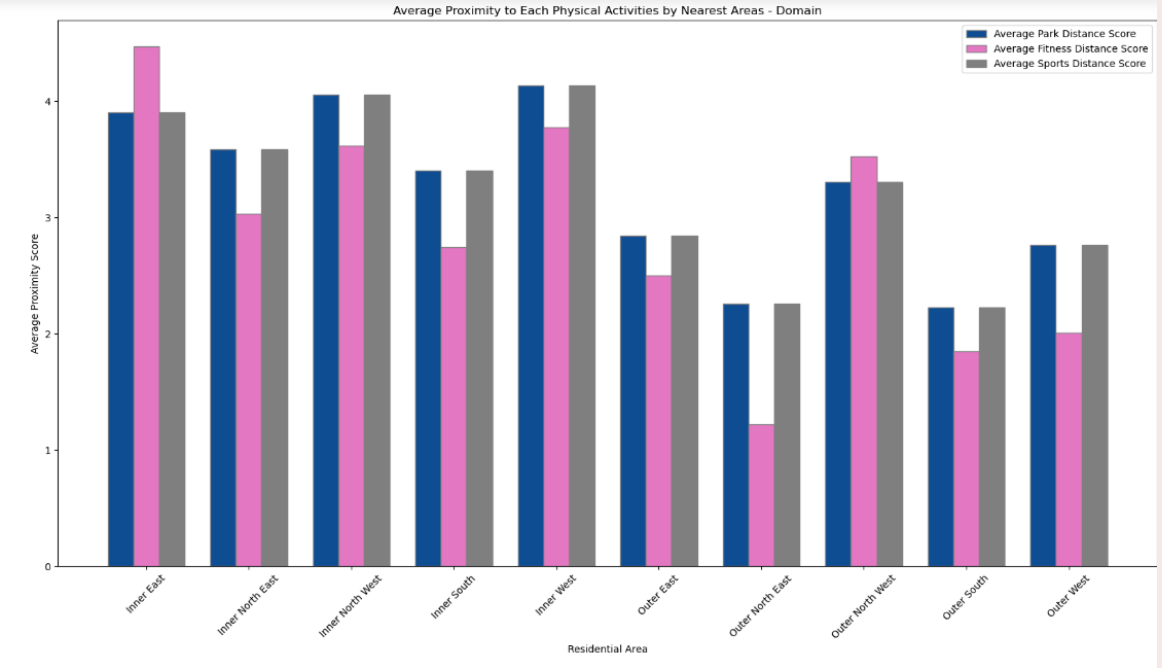
- Few residential areas exhibit unreasonable density, primarily clustered on the city outskirts, possibly due to limited investment or simplified planning in these peripheral regions.
- In contrast, residential areas with reasonable density are more widespread, found in both core urban areas and regions with developed infrastructure, likely benefiting from greater investment and facility planning.
- This distribution pattern may stem from various factors including urban planning, resource allocation, and land use. Areas with reasonable density may have benefited from more effective urban planning and resource allocation

Average Density Score to Each Physical Activities



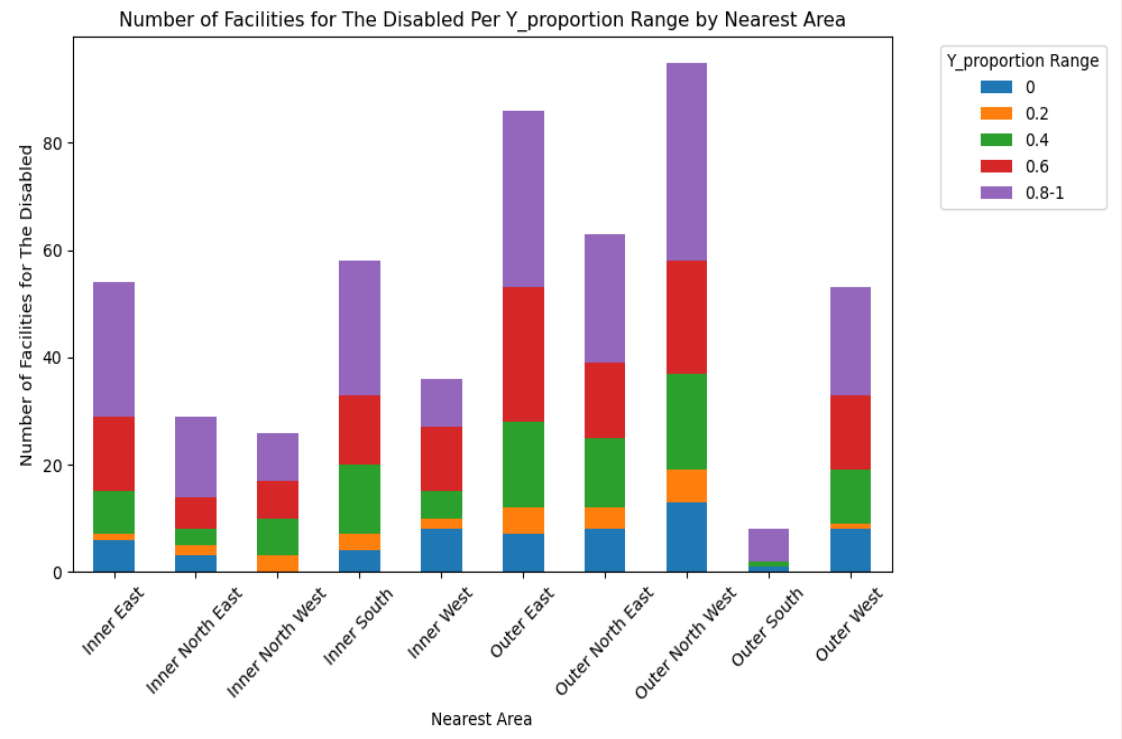
- Inner East Area stands out with a higher average fitness centre density score, suggesting enhanced accessibility to fitness centres.
- Inner West Area exhibits a higher average sports centre density score, whereas Outer North-East Area has a lower score in comparison.

Average Proximity Score to Each Physical Activities by Nearest Area



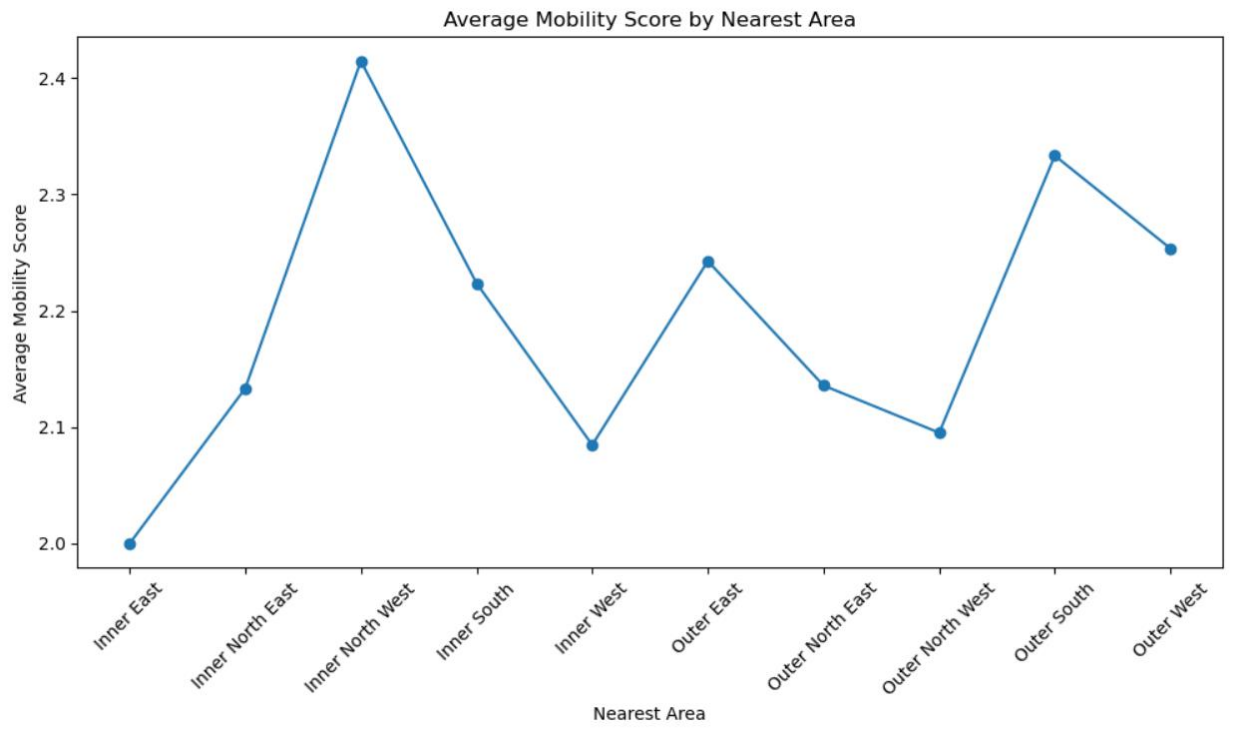
- Inner East, Inner West, and Outer North-West Areas show higher average scores for both sports and fitness centre proximity.
- Outer North East and Outer South Areas exhibit lower scores for both categories.

Number of Facilities for the Disabled by Nearest Area



- The proportion of facilities available to the disabled in most areas is above 0.6, which proves that support for disabled people in Leeds is adequate. Especially in Outer North West and Outer East.

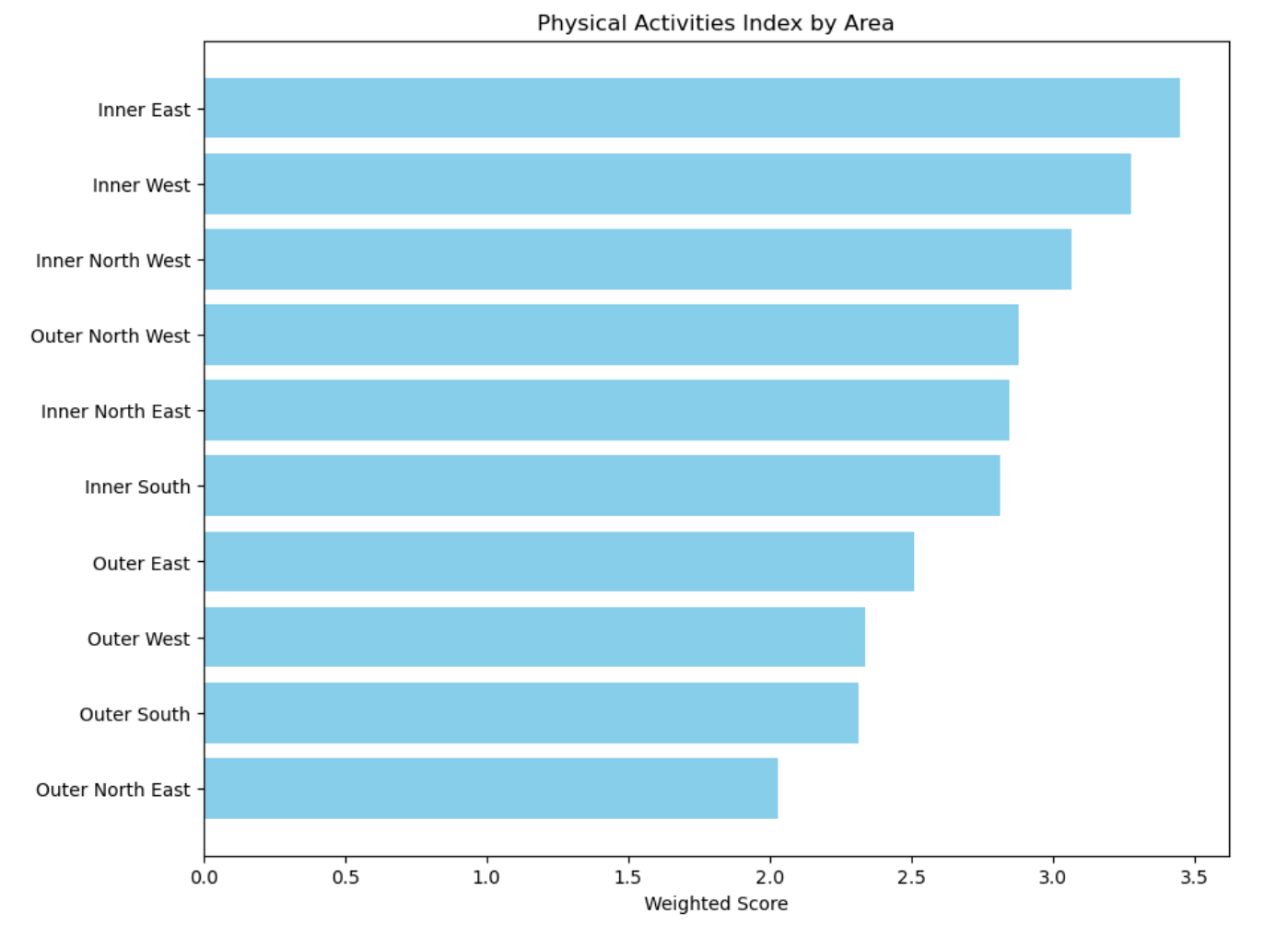
Average Mobility Score by Nearest Area



- Outer South and Inner North West show higher average scores for mobility.

- Inner East exhibits a very low score for mobility. This demonstrates the need for the area to focus on mobility improvements in physical activity facilities, such as increasing the number of parking lots

Final Index by different areas



- Most areas have an index of more than 2.5 scores - that is, 50 per cent, indicating that most parts of Leeds have good physical activity facilities.
- Among them, Inner East and Inner West, Inner North West are the highest, with more than 3 scores.
- It was lower in the Outer North East, with a score of less than 2.5, indicating a need for policymakers to improve relevant facilities in the region.

LIMITATIONS

Demographic Considerations

- The research does not take into account demographic characteristics. However, Demographics can play a significant role in the shaping of the relationship between physical activity, as research has shown that factors such as age, and income levels can influence individuals' likelihood to engage in physical activity and their mode of transport.

Smart Data or Mobile Data

- Utilising smart data from devices like smartwatches could have enriched the analysis by providing real-time, individual-level data on physical activity patterns. This could have offered valuable insights into participant’s daily routines and behaviours, potentially leading to more nuanced conclusions.

FUTURE WORKS AND IDEAS

Public Transportation Improvement:

- Collaborate with transportation authorities to create efficient routes connecting residential areas to exercise facilities.
- Collaborate with city planners to design urban spaces that prioritise walking and cycling. While monitoring changes in physical activity levels and overall health outcomes in the communities by improved routes.

Create an application:

- People can use this application to simplify the process of finding suitable sports spots.
- Selecting a spot on the map, users will be able to get information, such as the distance from the current location, real-time traffic conditions around the spots, detailed information on available equipment, and whether it is suitable for disabilities.
- The application will include a search function, allowing people to effectively search, view, and choose sports areas based on their requirements.

REFERENCES

- CDC (2023a) *Access to places for physical activity*, *Centers for Disease Control and Prevention*. Available at: https://www.cdc.gov/physicalactivity/community-strategies/access-to-places-for-physical-activity.html (Accessed: February 26, 2024).
- CDC (2023b) *Benefits of physical activity*, *Centers for Disease Control and Prevention*. Available at: https://www.cdc.gov/physicalactivity/basics/pa-health/index.htm (Accessed: February 26, 2024).
- Cohen, D. A. *et al.* (2007) "Contribution of public parks to physical activity," *American journal of public health*, 97(3), pp. 509–514. doi: 10.2105/AJPH.2005.072447.
- *Cycling and walking can help reduce physical inactivity and air pollution, save lives and mitigate climate change* (no date) *Who.int*. Available at: https://www.who.int/europe/news/item/07-06-2022-cycling-and-walking-can-help-reduce-physical-inactivity-and-air-pollution--save-lives-and-mitigate-climate-change (Accessed: February 26, 2024).
- Giles-Corti, B. and Donovan, R. J. (2002) "The relative influence of individual, social and physical environment determinants of physical activity," *Social science & medicine (1982)*, 54(12), pp. 1793–1812. doi: 10.1016/s0277-9536(01)00150-2.
- Jilcott, S. B. *et al.* (2007) "Association between physical activity and proximity to physical activity resources among low-income, midlife women," *Preventing chronic disease*, 4(1), p. A04.
- Kaczynski, A. T. and Henderson, K. A. (2007) "Environmental correlates of physical activity: A review of evidence about parks and recreation," *Leisure sciences*, 29(4), pp. 315–354. doi: 10.1080/01490400701394865.
- Kumar, H. *et al.* (2019) "Means as well as ends: some critical insights for UK sport policy on the impact of facility ownership and configuration on sports participation," *International Journal of Sport Policy and Politics*, 11(3), pp. 415–432. doi: 10.1080/19406940.2018.1522660.
- Tomer, A. and George, C. (2023) *Building for proximity: The role of activity centers in reducing total miles traveled*, *Brookings*. Available at: https://www.brookings.edu/articles/building-for-proximity-the-role-of-activity-centers-in-reducing-total-miles-traveled/ (Accessed: February 26, 2024).

Exploring the Impact of Proximity, Accessibility and Active Travel on Physical Activity

1. Group information

Group Number: 9

Student ID:

1. 201413609
2. 201760120
3. 201650287
4. 201348673
5. 201788996
6. 201762009

2. Import libraries

In [1]:
```
import pandas as pd
import geopandas as gpd
import contextily as ctx
import numpy as np
import folium
import matplotlib.pyplot as plt
from geopy.distance import geodesic
from math import radians, cos, sin, sqrt, atan2
from mpl_toolkits.basemap import Basemap
from shapely import wkt
from shapely.geometry import Point
from folium.plugins import HeatMap
```

3. Load the dataset

In [2]:
```
residential=pd.read_csv('residential_leeds.csv')
park=pd.read_csv('parks_leeds.csv')
sports=pd.read_csv('sports_centre_leeds.csv')
fitness=pd.read_csv('fitness_centre_leeds.csv')
Geographic=pd.read_csv('Geographic.csv')
site=pd.read_csv('sites.csv')
```

4. Data exploration and data wrangling

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2536 entries, 0 to 2535
Data columns (total 62 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   geometry                  2536 non-null   object
 1   name                      177 non-null    object
 2   building                  1 non-null      object
 3   landuse                   2536 non-null   object
 4   access                    3 non-null      object
 5   amenity                   3 non-null      object
 6   brand                     1 non-null      object
 7   brand:wikidata            1 non-null      object
 8   brand:wikipedia           1 non-null      object
 9   operator                  19 non-null     object
 10  note                      16 non-null     object
 11  source                    478 non-null    object
 12  wheelchair                2 non-null      object
 13  fixme                     6 non-null      object
 14  addr:city                 9 non-null      object
 15  addr:housename            1 non-null      object
 16  addr:postcode             7 non-null      object
 17  addr:street               6 non-null      object
 18  addr:suburb               2 non-null      object
 19  alt_name                  7 non-null      object
 20  fhrs:id                   2 non-null      float64
 21  internet_access           1 non-null      object
 22  internet_access:fee       1 non-null      object
 23  internet_access:operator  1 non-null      object
 24  internet_access:ssid      1 non-null      object
 25  phone                     1 non-null      object
 26  tourism                   1 non-null      object
 27  description               5 non-null      object
 28  natural                   1 non-null      object
 29  addr:housenumber          3 non-null      float64
 30  nodes                     2534 non-null   object
 31  disused                   1 non-null      object
 32  residential               17 non-null     object
 33  old_name                  13 non-null     object
 34  religion                  1 non-null      object
 35  website                   6 non-null      object
 36  wikidata                  4 non-null      object
 37  wikipedia                 3 non-null      object
 38  leisure                   1 non-null      object
 39  operator:type             1 non-null      object
 40  social_facility           2 non-null      object
 41  social_facility:for       2 non-null      object
 42  source:name               19 non-null     object
 43  source:old_name           3 non-null      object
 44  construction              10 non-null     object
 45  place                     22 non-null     object
 46  layer                     1 non-null      float64
 47  contact:website           1 non-null      object
 48  surface                   6 non-null      object
 49  landuse_2                 1 non-null      object
 50  denomination              1 non-null      object
 51  area                      2 non-null      object
 52  name:etymology            2 non-null      object
 53  name:etymology:wikidata   2 non-null      object
 54  name:etymology:wikipedia  2 non-null      object
```

```
 55  start_date                2 non-null      float64
 56  official_name             1 non-null      object
 57  ways                      21 non-null     object
 58  type                      21 non-null     object
 59  centroid                  2536 non-null   object
 60  longitude                 2536 non-null   float64
 61  latitude                  2536 non-null   float64
dtypes: float64(6), object(56)
memory usage: 1.2+ MB
```

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

Out[4]:

	geometry	name	building	landuse	access	amenity	brand	bran
0	POINT (-1.5656563 53.8137474)	High Lea Court	yes	residential	NaN	NaN	NaN	
1	POINT (-1.5439758 53.8042438)	Premier Inn	NaN	residential	NaN	NaN	Premier Inn	
2	POLYGON ((-1.6941674 53.9032485, -1.6944606 53...	NaN	NaN	residential	NaN	NaN	NaN	
3	POLYGON ((-1.6703437 53.9094396, -1.6693126 53...	NaN	NaN	residential	NaN	NaN	NaN	
4	POLYGON ((-1.7063184 53.9005304, -1.7054907 53...	Hawthornden	NaN	residential	NaN	NaN	NaN	

5 rows × 62 columns

In [5]:
```
# Save only the columns we need
residential=residential[['name','geometry','longitude','latitude']]
residential.head()
```

Out[5]:

	name	geometry	longitude	latitude
0	High Lea Court	POINT (-1.5656563 53.8137474)	-1.565656	53.813747
1	Premier Inn	POINT (-1.5439758 53.8042438)	-1.543976	53.804244
2	NaN	POLYGON ((-1.6941674 53.9032485, -1.6944606 53...	-1.694657	53.903368
3	NaN	POLYGON ((-1.6703437 53.9094396, -1.6693126 53...	-1.669387	53.908497
4	Hawthornden	POLYGON ((-1.7063184 53.9005304, -1.7054907 53...	-1.705811	53.900309

In [6]:

```
park.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 530 entries, 0 to 529
Data columns (total 62 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   created_by         3 non-null      object
 1   geometry           530 non-null    object
 2   barrier            1 non-null      object
 3   access             19 non-null     object
 4   leisure            530 non-null    object
 5   name               190 non-null    object
 6   amenity            1 non-null      object
 7   traffic_calming    1 non-null      object
 8   highway            3 non-null      object
 9   lit                4 non-null      object
 10  source             115 non-null    object
 11  operator           12 non-null     object
 12  description        10 non-null     object
 13  note               7 non-null      object
 14  surface            22 non-null     object
 15  natural            2 non-null      object
 16  opening_hours      8 non-null      object
 17  tourism            3 non-null      object
 18  website            7 non-null      object
 19  nodes              524 non-null    object
 20  loc_name           1 non-null      object
 21  wikidata           16 non-null     object
 22  wikipedia          7 non-null      object
 23  landuse            11 non-null     object
 24  layer              9 non-null      float64
 25  fee                10 non-null     object
 26  alt_name           3 non-null      object
 27  old_name           3 non-null      object
 28  image              1 non-null      object
 29  historic           4 non-null      object
 30  memorial           1 non-null      object
 31  area               4 non-null      object
 32  dog                6 non-null      object
 33  fixme              4 non-null      object
 34  man_made           2 non-null      object
 35  other_name         1 non-null      object
 36  source:name        9 non-null      object
 37  note:name          1 non-null      object
 38  addr:city          3 non-null      object
 39  addr:postcode      2 non-null      object
 40  addr:street        3 non-null      object
 41  park_ride          1 non-null      object
 42  parking            3 non-null      object
 43  wikimedia_commons  1 non-null      object
 44  contact:facebook   1 non-null      object
 45  attraction         1 non-null      object
 46  cafe               1 non-null      object
 47  start_date         1 non-null      float64
 48  building           1 non-null      object
 49  location           1 non-null      object
 50  website:map        1 non-null      object
 51  bridge:structure   2 non-null      object
 52  heritage           1 non-null      object
 53  level              3 non-null      float64
 54  addr:housenumber   1 non-null      float64
```

```
 55  proposed           3 non-null      object
 56  designation        1 non-null      object
 57  ways               9 non-null      object
 58  type               9 non-null      object
 59  centroid           530 non-null    object
 60  longitude          530 non-null    float64
 61  latitude           530 non-null    float64
dtypes: float64(6), object(56)
memory usage: 256.8+ KB
```

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

Out[7]:

	created_by	geometry	barrier	access	leisure	name	amenity	traffic_cal
0	NaN	POINT (-1.5547318 53.7827922)	NaN	NaN	park	Holbeck Moor Park	NaN	
1	NaN	POINT (-1.4936125 53.8191167)	NaN	NaN	park	Baileys Hiding Den	NaN	
2	NaN	POINT (-1.3821555 53.7682646)	NaN	NaN	park	Devils Canyon	NaN	
3	NaN	POINT (-1.5350531 53.798485)	NaN	NaN	park	Playhouse Gardens	NaN	
4	NaN	POINT (-1.5692673 53.8273674)	NaN	NaN	park	Monk Bridge Green	NaN	

5 rows × 62 columns

In [8]:
```
# Save only the columns we need
park=park[['name','geometry','longitude','latitude']]
park.head()
```

Out[8]:

	name	geometry	longitude	latitude
0	Holbeck Moor Park	POINT (-1.5547318 53.7827922)	-1.554732	53.782792
1	Baileys Hiding Den	POINT (-1.4936125 53.8191167)	-1.493613	53.819117
2	Devils Canyon	POINT (-1.3821555 53.7682646)	-1.382156	53.768265
3	Playhouse Gardens	POINT (-1.5350531 53.798485)	-1.535053	53.798485
4	Monk Bridge Green	POINT (-1.5692673 53.8273674)	-1.569267	53.827367

In [9]:
```
sports.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 82 entries, 0 to 81
Data columns (total 49 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   geometry           82 non-null     object
 1   leisure            82 non-null     object
 2   name               74 non-null     object
 3   addr:city          30 non-null     object
 4   addr:postcode      30 non-null     object
 5   addr:street        31 non-null     object
 6   building           42 non-null     object
 7   wheelchair         5 non-null      object
 8   wikidata           8 non-null      object
 9   wikipedia          6 non-null      object
 10  addr:suburb        20 non-null     object
 11  fhrs:id            19 non-null     float64
 12  old_name           5 non-null      object
 13  sport              40 non-null     object
 14  website            11 non-null     object
 15  addr:housename     5 non-null      object
 16  club               2 non-null      object
 17  nodes              74 non-null     object
 18  operator           6 non-null      object
 19  email              2 non-null      object
 20  fax                1 non-null      object
 21  phone              4 non-null      object
 22  source             6 non-null      object
 23  contact:facebook   3 non-null      object
 24  url                1 non-null      object
 25  contact:website    1 non-null      object
 26  building:levels    1 non-null      float64
 27  addr:housenumber   5 non-null      object
 28  opening_hours      3 non-null      object
 29  landuse            1 non-null      object
 30  contact:instagram  1 non-null      object
 31  contact:twitter    1 non-null      object
 32  depth              1 non-null      float64
 33  fee                2 non-null      object
 34  addr:place         1 non-null      object
 35  addr:substreet     1 non-null      object
 36  addr:town          1 non-null      object
 37  addr:unit          1 non-null      float64
 38  bowls              3 non-null      object
 39  alt_name           1 non-null      object
 40  brand              1 non-null      object
 41  internet_access    1 non-null      object
 42  addr:parentstreet  1 non-null      object
 43  climbing:boulder   1 non-null      object
 44  ways               1 non-null      object
 45  type               1 non-null      object
 46  centroid           82 non-null     object
 47  longitude          82 non-null     float64
 48  latitude           82 non-null     float64
dtypes: float64(6), object(43)
memory usage: 31.5+ KB
```

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

Out[10]:

	geometry	leisure	name	addr:city	addr:postcode	addr:street	buil
0	POINT (-1.3897263 53.9260262)	sports_centre	Wetherby Leisure Centre	NaN	NaN	NaN	
1	POINT (-1.7032339 53.8001037)	sports_centre	Esporta	NaN	NaN	NaN	
2	POINT (-1.5422539 53.764693)	sports_centre	John Charles Centre For Sport	Leeds	LS11 5DJ	Middleton Grove	
3	POINT (-1.4741962 53.9002031)	sports_centre	NaN	NaN	NaN	NaN	
4	POINT (-1.6887946 53.8671099)	sports_centre	DW Fitness	Leeds	LS19 7EN	Haworth Lane	

5 rows × 49 columns

In [11]:

```
# Save only the columns we need
sports=sports[['name','geometry','longitude','latitude']]
sports.head()
```

Out[11]:

	name	geometry	longitude	latitude
0	Wetherby Leisure Centre	POINT (-1.3897263 53.9260262)	-1.389726	53.926026
1	Esporta	POINT (-1.7032339 53.8001037)	-1.703234	53.800104
2	John Charles Centre For Sport	POINT (-1.5422539 53.764693)	-1.542254	53.764693
3	NaN	POINT (-1.4741962 53.9002031)	-1.474196	53.900203
4	DW Fitness	POINT (-1.6887946 53.8671099)	-1.688795	53.867110

In [12]:

```
fitness.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 55 entries, 0 to 54
Data columns (total 51 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   addr:city                 35 non-null     object
 1   addr:postcode             35 non-null     object
 2   addr:street               33 non-null     object
 3   fhrs:id                   8 non-null      float64
 4   leisure                   55 non-null     object
 5   name                      53 non-null     object
 6   source                    5 non-null      object
 7   geometry                  55 non-null     object
 8   addr:housename            14 non-null     object
 9   addr:housenumber          16 non-null     object
 10  brand                     20 non-null     object
 11  brand:wikidata            20 non-null     object
 12  covered                   1 non-null      object
 13  sport                     10 non-null     object
 14  brand:wikipedia           13 non-null     object
 15  contact:website           7 non-null      object
 16  level                     2 non-null      float64
 17  operator                  1 non-null      object
 18  addr:suburb               17 non-null     object
 19  internet_access           1 non-null      object
 20  name:en                   1 non-null      object
 21  website                   13 non-null     object
 22  check_date:opening_hours  1 non-null      object
 23  opening_hours:signed      1 non-null      object
 24  branch                    5 non-null      object
 25  building                  23 non-null     object
 26  addr:unit                 5 non-null      object
 27  addr:place                2 non-null      object
 28  female                    1 non-null      object
 29  male                      1 non-null      object
 30  opening_hours             8 non-null      object
 31  gender                    2 non-null      object
 32  phone                     6 non-null      object
 33  nodes                     26 non-null     object
 34  old_name                  3 non-null      object
 35  source:building           3 non-null      object
 36  url                       1 non-null      object
 37  swimming_pool             1 non-null      object
 38  historic                  1 non-null      object
 39  landuse                   1 non-null      object
 40  climbing:boulder          1 non-null      object
 41  climbing:indoor           1 non-null      object
 42  outdoor_seating           1 non-null      object
 43  building:levels           2 non-null      float64
 44  building:material         2 non-null      object
 45  roof:shape                2 non-null      object
 46  building:part             1 non-null      object
 47  shop                      1 non-null      object
 48  centroid                  55 non-null     object
 49  longitude                 55 non-null     float64
 50  latitude                  55 non-null     float64
dtypes: float64(5), object(46)
memory usage: 22.0+ KB
```

In [13]:
```
fitness.head()
```

Out[13]:

	addr:city	addr:postcode	addr:street	fhrs:id	leisure	name	s
0	Leeds	LS4 2EZ	Kirkstall Road	467019.0	fitness_centre	COACH Gym	Bing;s
1	Leeds	LS1 8TL	The Headrow	NaN	fitness_centre	Nuffield Health Fitness & Wellbeing	
2	NaN	NaN	NaN	NaN	fitness_centre	PureGym	
3	Leeds	LS27 0HG	Texas Street	1065674.0	fitness_centre	Activ zone	
4	NaN	NaN	NaN	NaN	fitness_centre	Re-ba Fitness	

5 rows × 51 columns

In [14]:

```
# Save only the columns we need
fitness=fitness[['name','geometry','longitude','latitude']]
fitness.head()
```

Out[14]:

	name	geometry	longitude	latitude
0	COACH Gym	POINT (-1.5868169 53.807151)	-1.586817	53.807151
1	Nuffield Health Fitness & Wellbeing	POINT (-1.5461011 53.8001647)	-1.546101	53.800165
2	PureGym	POINT (-1.5305892 53.8037762)	-1.530589	53.803776
3	Activ zone	POINT (-1.5919556 53.7350438)	-1.591956	53.735044
4	Re-ba Fitness	POINT (-1.3885449 53.792249)	-1.388545	53.792249

In [15]:

```
# Only save information in Leeds according to the postcode
site=site[site['PostCode'].str.startswith('LS')]
site.head()
```

Out[15]:

	SiteID	SiteName	TOIDLink	DisabilityExist	DisabilityParam	Di
76	1000087	VILLAGE GYM (LEEDS NORTH)	osgb1000032499592	1	Y,Y,Y,Y,Y,Y,Y,Y,Y,Y	
103	1000115	DAVID LLOYD (LEEDS)	osgb1000032590682	1	Y,Y,Y,Y,Y,Y,Y,Y,Y,Y	
111	1000124	DW FITNESS FIRST (YEADON) (CLOSED)	osgb1000032196306	1	Y,Y,Y,Y,Y,Y,Y,Y,Y,Y	P
117	1000130	COOKRIDGE HALL GOLF CLUB	osgb1000032413168	1	Y,Y,Y,Y,Y,Y,Y,Y,Y,Y	
191	1000228	ARMLEY SQUASH CLUB (CLOSED)	NaN	2	NaN	

5 rows × 39 columns

In [16]:
```
site.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 687 entries, 76 to 41836
Data columns (total 39 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   SiteID                      687 non-null    int64
 1   SiteName                    687 non-null    object
 2   TOIDLink                    490 non-null    object
 3   DisabilityExist             687 non-null    int64
 4   DisabilityParam             641 non-null    object
 5   DisabilityNotes             91 non-null     object
 6   CarParkExist                687 non-null    int64
 7   NumCarPark                  687 non-null    int64
 8   RecEntryDate                687 non-null    object
 9   StartDate                   45 non-null     object
 10  RecEndDate                  72 non-null     object
 11  RecLastChkdDate             687 non-null    object
 12  RecAuditDate                687 non-null    object
 13  OwnerTypeID                 687 non-null    int64
 14  OwnerOrgName                241 non-null    object
 15  MgtTypeID                   687 non-null    int64
 16  MgtType                     687 non-null    object
 17  OperatorName                12 non-null     object
 18  SubBuildingName             27 non-null     object
 19  BuildingName                158 non-null    object
 20  BuildingNumber              687 non-null    int64
 21  DependentThoroughfare       14 non-null     object
 22  ThoroughfareName            677 non-null    object
 23  DoubleDependentLocality     26 non-null     object
 24  DependentLocality           344 non-null    object
 25  PostTown                    687 non-null    object
 26  PostCode                    687 non-null    object
 27  EducationPhaseID            687 non-null    int64
 28  PublicNotes                 87 non-null     object
 29  CyclePark                   687 non-null    int64
 30  BikeHireFlag                687 non-null    int64
 31  RepairWorkshopFlag          687 non-null    int64
 32  NurseryFlag                 687 non-null    int64
 33  FirstAid                    687 non-null    int64
 34  ClosureCode                 687 non-null    int64
 35  ClosureReason               72 non-null     object
 36  ClosureNotes                0 non-null      float64
 37  ChangingPlacesToiletsExist  687 non-null    int64
 38  UPRN                        687 non-null    int64
dtypes: float64(1), int64(16), object(22)
memory usage: 214.7+ KB
```

In [17]:

```python
# Save only the columns we need and delete duplicates
Geographic=Geographic[['SiteID','Ward Name','Latitude','Longitude']]
Geographic=Geographic.drop_duplicates()
Geographic.head()
```

Out[17]:

	SiteID	Ward Name	Latitude	Longitude
0	1000001	Harrogate New Park	54.000375	-1.555473
8	1000002	South Chafford	51.490948	0.291765
15	1000003	Branston	52.782956	-1.652521
22	1000004	Moulsecoomb and Bevendean	50.859143	-0.089653
32	1000005	Bishopsgate	51.520049	-0.082017

In [18]:
```
Geographic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 41850 entries, 0 to 119383
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   SiteID     41850 non-null  int64
 1   Ward Name  41850 non-null  object
 2   Latitude   41850 non-null  float64
 3   Longitude  41850 non-null  float64
dtypes: float64(2), int64(1), object(1)
memory usage: 1.6+ MB
```

6. Data analytics and visualization

In [19]:
```
# Calculate the distance and find the nearest physical activities
def find_nearby_activities(lat, lon, destinations):
    distances = destinations.apply(lambda row: geodesic((lat, lon), (row['latitu
    closest_lat = destinations.iloc[distances.idxmin()]['latitude']
    closest_lon = destinations.iloc[distances.idxmin()]['longitude']
    closest_distance = distances.min()
    density_within_1km = (distances <= 1000).sum()
    return closest_lat, closest_lon, closest_distance, density_within_1km
```

In [20]:
```
closest_park_lat = []
closest_park_lon = []
closest_sports_lat = []
closest_sports_lon = []
closest_fitness_lat = []
closest_fitness_lon = []
park_distance = []
sports_distance = []
fitness_distance = []
park_density = []
sports_density = []
fitness_density = []

# Calculate the nearest sports centres, fitness centres, and parks to each resid
# Calculate density for sports centres, fitness centres, and parks within 1km of
for index, residence in residential.iterrows():
    park_lat, park_lon, park_dist, park_dens = find_nearby_activities(residence[
    sports_lat, sports_lon, sports_dist, sports_dens = find_nearby_activities(re
    fitness_lat, fitness_lon, fitness_dist, fitness_dens = find_nearby_activitie

    closest_park_lat.append(park_lat)
    closest_park_lon.append(park_lon)
```

```
    closest_sports_lat.append(sports_lat)
    closest_sports_lon.append(sports_lon)
    closest_fitness_lat.append(fitness_lat)
    closest_fitness_lon.append(fitness_lon)
    park_distance.append(park_dist)
    sports_distance.append(sports_dist)
    fitness_distance.append(fitness_dist)
    park_density.append(park_dens)
    sports_density.append(sports_dens)
    fitness_density.append(fitness_dens)
```

In [21]:

```
# Add some columns to save this information
residential['closest_park_lat'] = closest_park_lat
residential['closest_park_lon'] = closest_park_lon
residential['closest_sports_lat'] = closest_sports_lat
residential['closest_sports_lon'] = closest_sports_lon
residential['closest_fitness_lat'] = closest_fitness_lat
residential['closest_fitness_lon'] = closest_fitness_lon
residential['park_distance'] = park_distance
residential['sports_distance'] = sports_distance
residential['fitness_distance'] = fitness_distance
residential['park_density'] = park_density
residential['sports_density'] = sports_density
residential['fitness_density'] = fitness_density
residential.head()
```

Out[21]:

	name	geometry	longitude	latitude	closest_park_lat	closest_park_l
0	High Lea Court	POINT (-1.5656563 53.8137474)	-1.565656	53.813747	53.815207	-1.5648
1	Premier Inn	POINT (-1.5439758 53.8042438)	-1.543976	53.804244	53.803507	-1.5452
2	NaN	POLYGON ((-1.6941674 53.9032485, -1.6944606 53...	-1.694657	53.903368	53.904403	-1.6963
3	NaN	POLYGON ((-1.6703437 53.9094396, -1.6693126 53...	-1.669387	53.908497	53.909601	-1.6698
4	Hawthornden	POLYGON ((-1.7063184 53.9005304, -1.7054907 53...	-1.705811	53.900309	53.901612	-1.7038

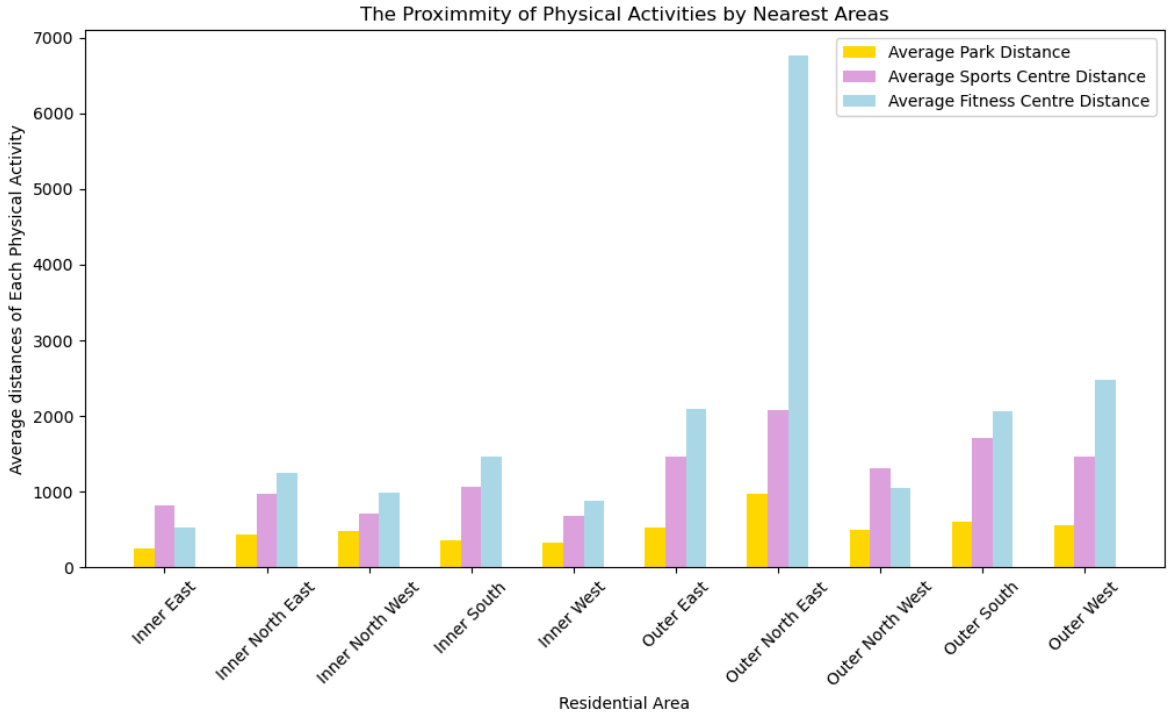
In [22]:

```
# 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


residential['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 residential.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
    residential.at[index, 'Nearest Area'] = nearest_area
```

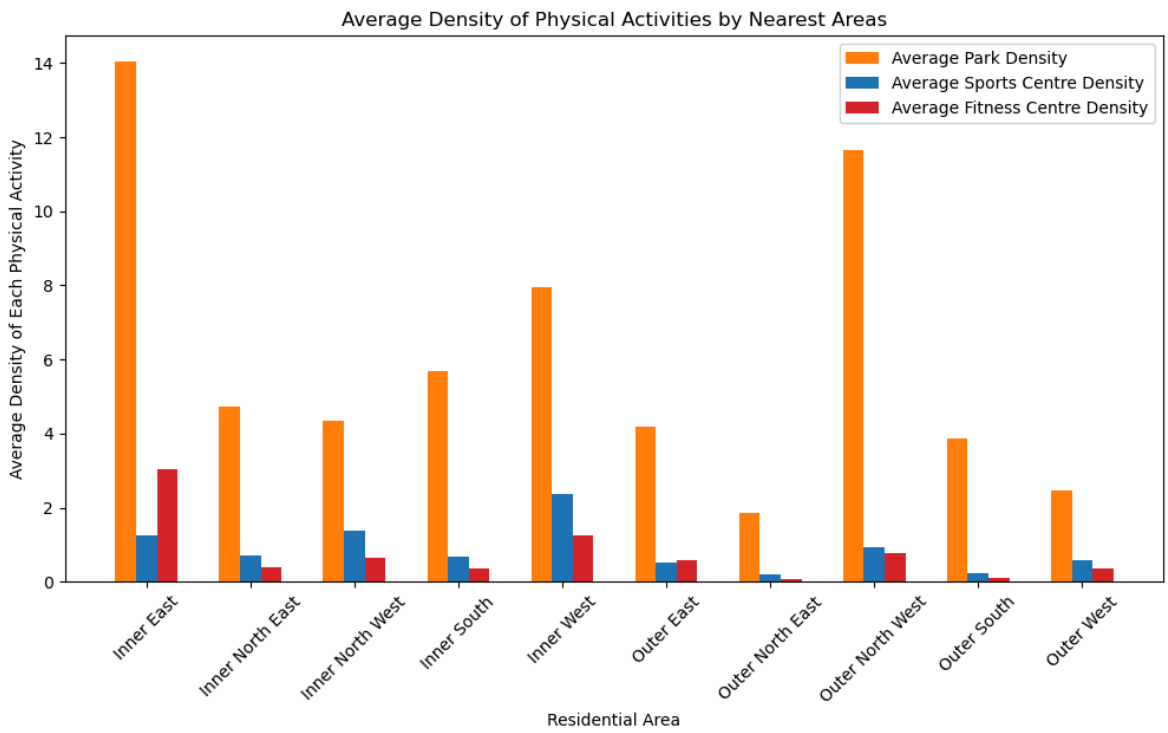
In [23]:

```
# Calculate the average distances of each Nearest Area - Proximmity
average_distances = residential.groupby('Nearest Area').agg({'park_distance': 'm
                                                            'sports_distance':

fig, ax = plt.subplots(figsize=(12, 6))
areas = average_distances['Nearest Area']
index = np.arange(len(areas))
bar_width = 0.2

ax.bar(index - bar_width, average_distances['park_distance'], bar_width, label='
ax.bar(index, average_distances['sports_distance'], bar_width, label='Average Sp
ax.bar(index + bar_width, average_distances['fitness_distance'], bar_width, labe

ax.set_xlabel('Residential Area')
ax.set_ylabel('Average distances of Each Physical Activity')
ax.set_title('The Proximmity of Physical Activities by Nearest Areas')
ax.set_xticks(index)
ax.set_xticklabels(areas, rotation=45)
ax.legend()

plt.show()
```



In [24]:

```
# Calculate the average distances of each Nearest Area
average_densities = residential.groupby('Nearest Area').agg({'park_density': 'me
                                                            'sports_density': '

fig, ax = plt.subplots(figsize=(12, 6))
areas = average_densities['Nearest Area']
index = np.arange(len(areas))
bar_width = 0.2

ax.bar(index - bar_width, average_densities['park_density'], bar_width, label='A
ax.bar(index, average_densities['sports_density'], bar_width, label='Average Spo
ax.bar(index + bar_width, average_densities['fitness_density'], bar_width, label

ax.set_xlabel('Residential Area')
ax.set_ylabel('Average Density of Each Physical Activity')
ax.set_title('Average Density of Physical Activities by Nearest Areas')
ax.set_xticks(index)
ax.set_xticklabels(areas, rotation=45)
ax.legend()

plt.show()
```



In [25]:
```
# If the physical activity density in the area is 0, it is fail
residential['park_density_flag']=residential['park_density'].apply(lambda x: 'Pa
residential['sports_density_flag']=residential['park_density'].apply(lambda x: '
residential['fitness_density_flag']=residential['fitness_density'].apply(lambda
residential.head()
```

Out[25]:

	name	geometry	longitude	latitude	closest_park_lat	closest_park_l
0	High Lea Court	POINT (-1.5656563 53.8137474)	-1.565656	53.813747	53.815207	-1.5648
1	Premier Inn	POINT (-1.5439758 53.8042438)	-1.543976	53.804244	53.803507	-1.5452
2	NaN	POLYGON ((-1.6941674 53.9032485, -1.6944606 53...	-1.694657	53.903368	53.904403	-1.6963
3	NaN	POLYGON ((-1.6703437 53.9094396, -1.6693126 53...	-1.669387	53.908497	53.909601	-1.6698
4	Hawthornden	POLYGON ((-1.7063184 53.9005304, -1.7054907 53...	-1.705811	53.900309	53.901612	-1.7038

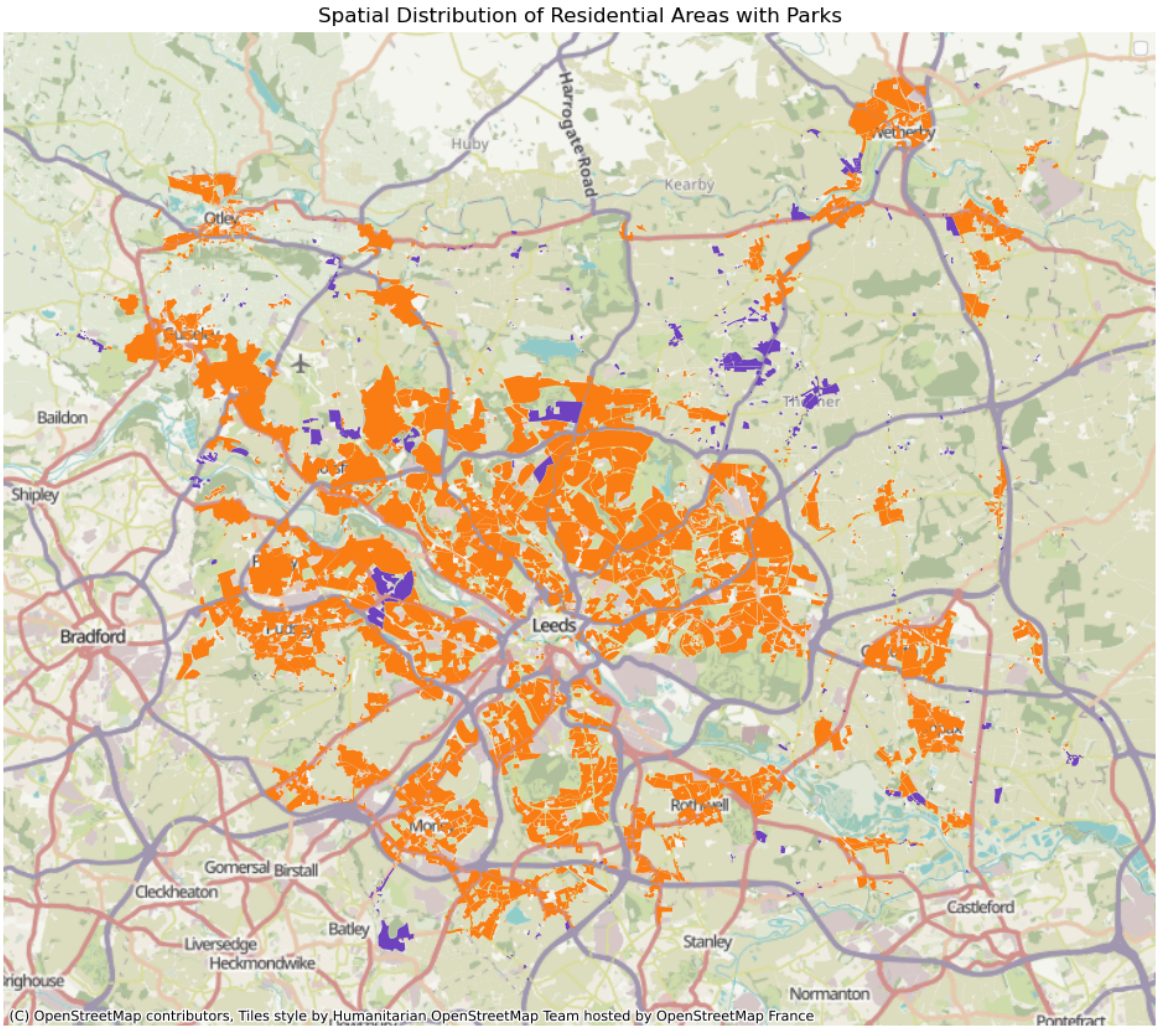
In [26]:
```
# Converts the 'geometry' into shapely's geometry object
residential['geo_geometry'] = residential['geometry'].apply(wkt.loads)
```

```
gdf = gpd.GeoDataFrame(residential, 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 residential areas with reasonable park densi
gdf_reasonable_park = gdf[gdf['park_density_flag'] == 'Pass']
gdf_unreasonable_park= gdf[gdf['park_density_flag'] == 'Fail']
gdf_reasonable_fitness = gdf[gdf['fitness_density_flag'] == 'Pass']
gdf_unreasonable_fitness= gdf[gdf['fitness_density_flag'] == 'Fail']
gdf_reasonable_sports = gdf[gdf['sports_density_flag'] == 'Pass']
gdf_unreasonable_sports= gdf[gdf['sports_density_flag'] == 'Fail']
```

In [27]:

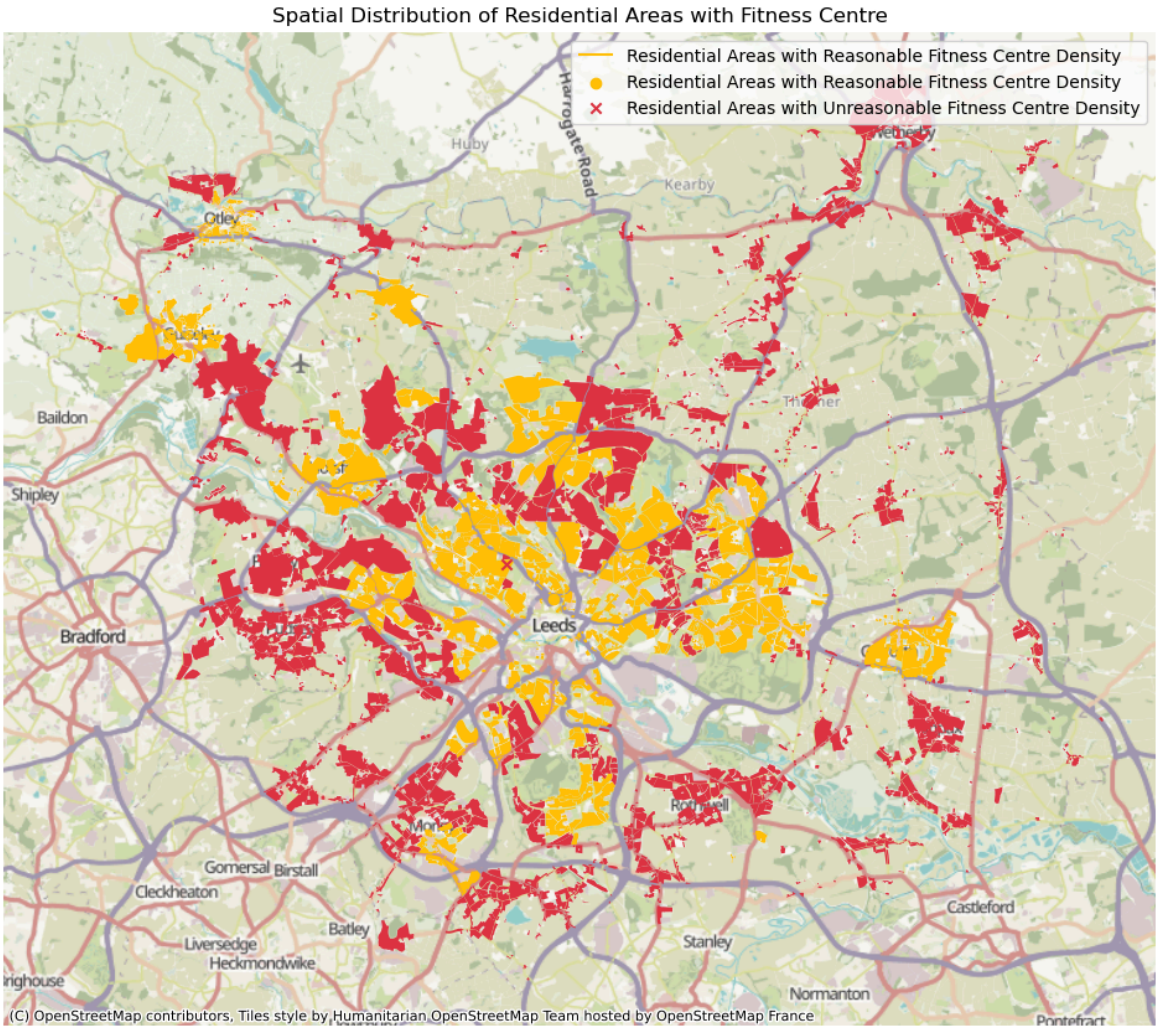
```
fig, ax = plt.subplots(figsize=(12, 24))
# Convert the coordinates of the data set to a Web Mercator to match the base ma
gdf_reasonable_park = gdf_reasonable_park.to_crs(epsg=3857)
gdf_unreasonable_park = gdf_unreasonable_park.to_crs(epsg=3857)

# Draw the spatial distribution of parks
gdf_reasonable_park.plot(ax=ax, color='#fd7e14', marker='o')
gdf_unreasonable_park.plot(ax=ax, color='#6f42c1', marker='x')

ctx.add_basemap(ax)
ax.set_title('Spatial Distribution of Residential Areas with Parks')
ax.legend()
ax.set_axis_off()
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label sta
rt with an underscore are ignored when legend() is called with no argument.
```



In [28]:
```
# Draw the spatial distribution of Fitness Centre
fig, ax1 = plt.subplots(figsize=(12, 24))

gdf_reasonable_fitness = gdf_reasonable_fitness.to_crs(epsg=3857)
gdf_unreasonable_fitness = gdf_unreasonable_fitness.to_crs(epsg=3857)

gdf_reasonable_fitness.plot(ax=ax1, color='#ffc107', marker='o', label='Resident
gdf_unreasonable_fitness.plot(ax=ax1, color='#dc3545', marker='x', label='Reside

ctx.add_basemap(ax1)
ax1.set_title('Spatial Distribution of Residential Areas with Fitness Centre')
ax1.legend()
ax1.set_axis_off()
```

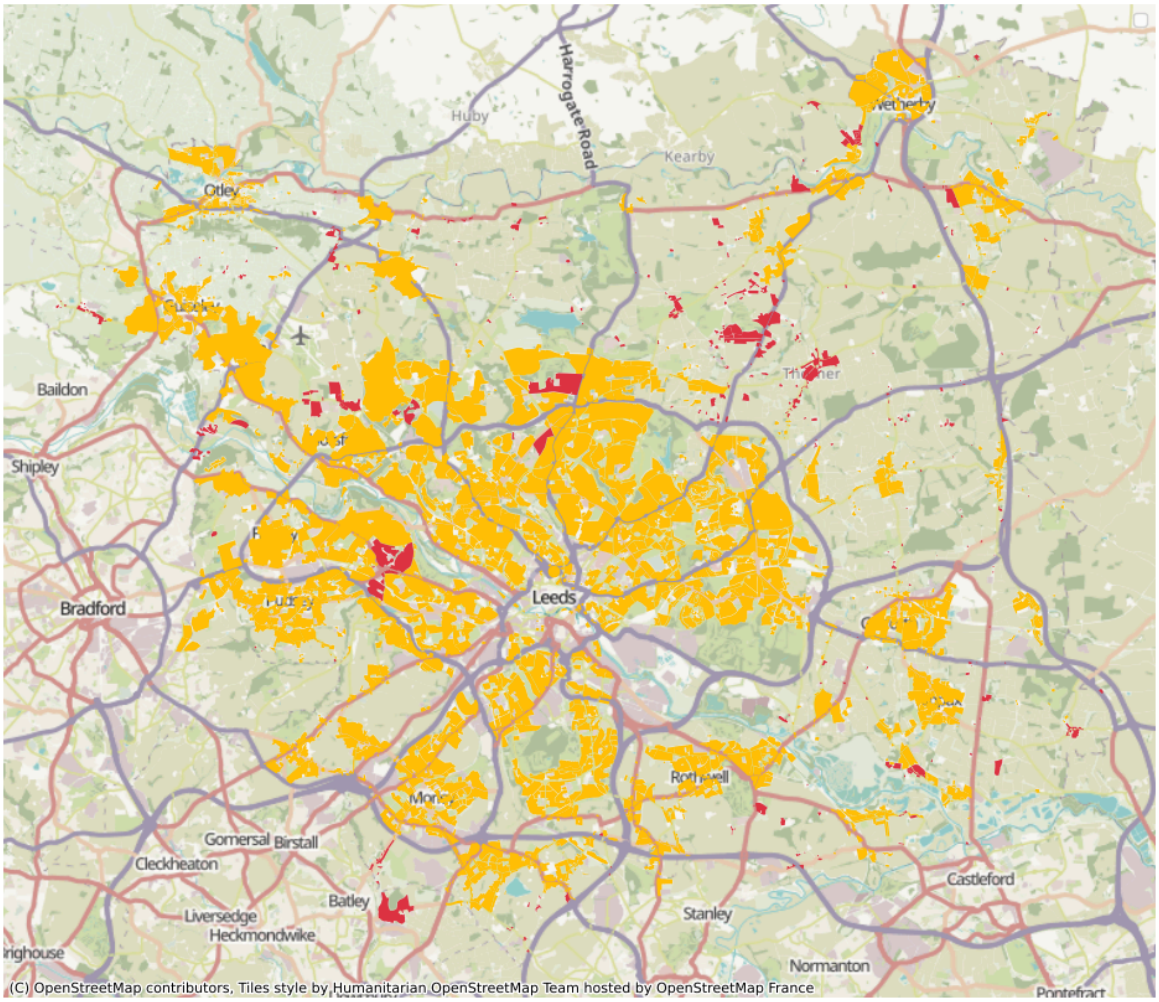
```
C:\Users\111\AppData\Local\Temp\ipykernel_22148\508422901.py:12: UserWarning: Leg
end does not support handles for PatchCollection instances.
See: https://matplotlib.org/stable/tutorials/intermediate/legend_guide.html#imple
menting-a-custom-legend-handler
  ax1.legend()
```



In [29]:
```
# Draw the spatial distribution of Sports Centre
fig, ax = plt.subplots(figsize=(12, 24))

gdf_reasonable_sports = gdf_reasonable_sports.to_crs(epsg=3857)
gdf_unreasonable_sports = gdf_unreasonable_sports.to_crs(epsg=3857)

gdf_reasonable_sports.plot(ax=ax, color='#ffc107', marker='o')
gdf_unreasonable_sports.plot(ax=ax, color='#dc3545', marker='x')

ctx.add_basemap(ax)
ax.set_title('Spatial Distribution of Residential Areas with Sports Centre')
ax.legend()
ax.set_axis_off()
```

```
No artists with labels found to put in legend.  Note that artists whose label sta
rt with an underscore are ignored when legend() is called with no argument.
```

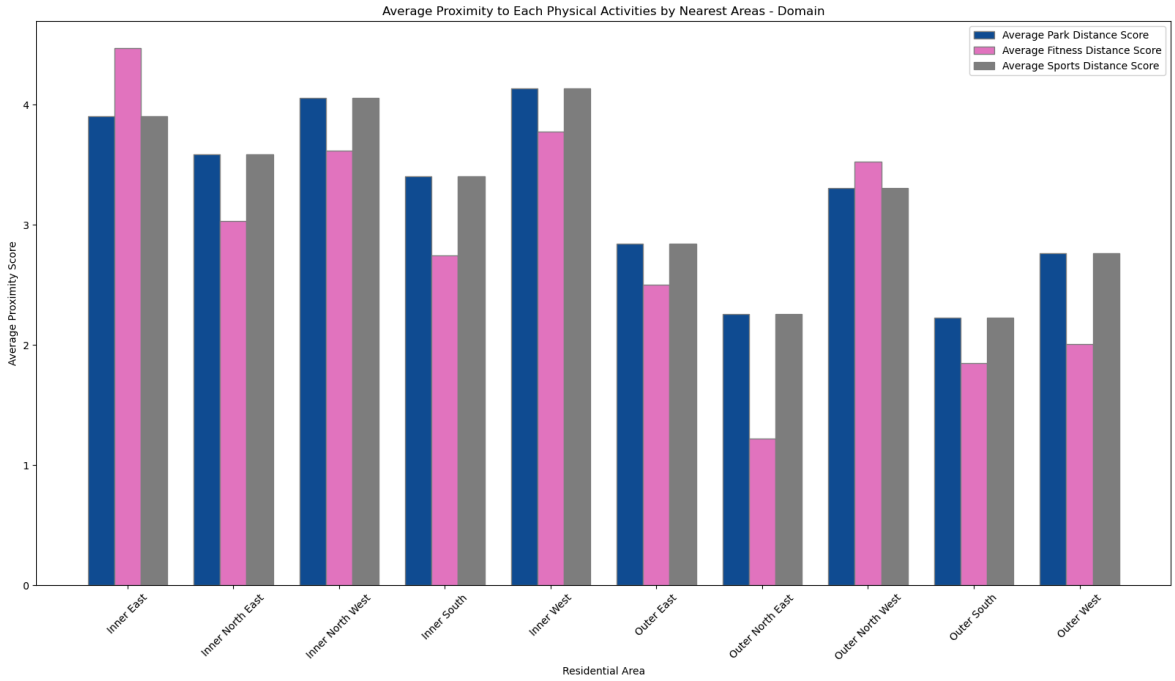
Spatial Distribution of Residential Areas with Sports Centre



In [30]:

```
# Define a scoring system based on distance
def score_distance(distance):
    if distance <= 500:
        return 5  # Within easy walking distance
    elif distance <= 1000:
        return 4  # Within a moderate walk
    elif distance <= 1500:
        return 3  # Within a longer walk / short bike ride
    elif distance <= 2000:
        return 2  # Within biking distance
    else:
        return 1  # Requires transportation


# Apply the scoring system to each distance
residential['distance_park_score'] = residential['sports_distance'].apply(score_
residential['distance_fitness_score'] = residential['fitness_distance'].apply(sc
residential['distance_sports_score'] = residential['sports_distance'].apply(scor

# Calculate the average distances of each Nearest Area
average_densities = residential.groupby('Nearest Area').agg({'distance_fitness_s
                                                            'distance_sports_sc

fig, ax = plt.subplots(figsize=(20, 10))
areas = average_densities['Nearest Area']
index = np.arange(len(areas))

# Width of the bar
```

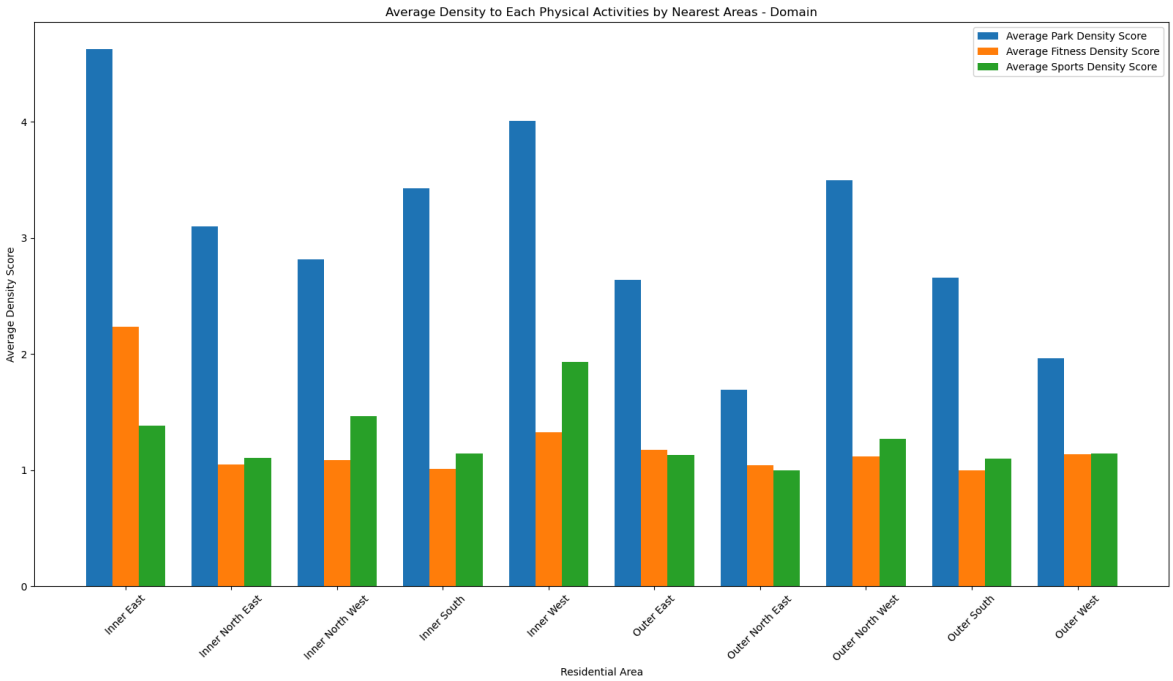
```
bar_width = 0.25

# Colors for the bars to simulate a gradient-like effect
colors = ['#0f4d92', '#e377c2', '#7f7f7f']

# Setting the position of bars on the x-axis
r1 = index
r2 = [x + bar_width for x in r1]
r3 = [x + bar_width for x in r2]

# Creating the bar chart
ax.bar(r1, average_densities['distance_park_score'], color=colors[0], width=bar_
       label='Average Park Distance Score')
ax.bar(r2, average_densities['distance_fitness_score'], color=colors[1], width=b
       label='Average Fitness Distance Score')
ax.bar(r3, average_densities['distance_sports_score'], color=colors[2], width=ba
       label='Average Sports Distance Score')

# Adding labels
ax.set_xlabel('Residential Area')
ax.set_ylabel('Average Proximity Score')
ax.set_title('Average Proximity to Each Physical Activities by Nearest Areas - D
ax.set_xticks([r + bar_width for r in range(len(areas))])
ax.set_xticklabels(areas, rotation=45)
ax.legend()

plt.show()
```



In [31]:

```
# Define a scoring system based on density
def score_density(density):
    if density >= 8:
        return 5
    elif density >= 6:
        return 4
    elif density >= 4:
        return 3
    elif density >= 2:
        return 2
    else:
```

```
        return 1


# Apply the scoring system to each density
residential['density_park_score'] = residential['park_density'].apply(score_dens
residential['density_fitness_centre_score'] = residential['fitness_density'].app
residential['density_sports_centre_score'] = residential['sports_density'].apply

average_densities = residential.groupby('Nearest Area').agg({'density_park_score
                                                            'density_fitness_ce
                                                            'density_sports_cen

fig, ax = plt.subplots(figsize=(20, 10))
areas = average_densities['Nearest Area']
index = np.arange(len(areas))

# Width of the bar
bar_width = 0.25

# Setting the position of bars on the x-axis
r1 = index
r2 = [x + bar_width for x in r1]
r3 = [x + bar_width for x in r2]

# Creating the bar chart
ax.bar(r1, average_densities['density_park_score'], width=bar_width, label='Aver
ax.bar(r2, average_densities['density_fitness_centre_score'], width=bar_width, l
ax.bar(r3, average_densities['density_sports_centre_score'], width=bar_width, la

# Adding labels
ax.set_xlabel('Residential Area')
ax.set_ylabel('Average Density Score')
ax.set_title('Average Density to Each Physical Activities by Nearest Areas - Dom
ax.set_xticks([r + bar_width for r in range(len(areas))])
ax.set_xticklabels(areas, rotation=45)
ax.legend()

plt.show()
```



In [32]:
```
# Mobility
site_mobility=site[['SiteID','CarParkExist','NumCarPark','CyclePark','BikeHireFl
# Merge the two datasets
site_mobility=pd.merge(site_mobility,Geographic,on='SiteID',how='inner')
replace = ['CarParkExist', 'CyclePark', 'BikeHireFlag', 'RepairWorkshopFlag']
# a conditional assignment to the specified column
for column in replace:
    site_mobility[column] = site_mobility[column].apply(lambda x: 0 if x == 2 el

site_mobility['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 site_mobility.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
    site_mobility.at[index, 'Nearest Area'] = nearest_area
site_mobility.head()
```

Out[32]:

	SiteID	CarParkExist	NumCarPark	CyclePark	BikeHireFlag	RepairWorkshopFl
0	1000087	1	300	1	0	
1	1000115	1	315	1	0	
2	1000124	1	140	1	0	
3	1000130	1	90	1	0	
4	1000228	1	30	0	0	

An interactive map displaying markers for each site, with popups showing the car parks, capacity of car parks and cycle parks at each location.

In [33]:
```
# Extract locations where Car Parks exist
car_park_locations = site_mobility[site_mobility['CarParkExist'] == 1][['Latitud
# Extract locations where Cycle Parks exist
cycle_park_locations = site_mobility[site_mobility['CyclePark'] == 1][['Latitude
# Extract locations where Bike Hire is available
bike_hire_locations = site_mobility[site_mobility['BikeHireFlag'] == 1][['Latitu
# Extract locations where Repair Workshop is available
repair_workshop_locations = site_mobility[site_mobility['RepairWorkshopFlag'] ==

# Create a map centered around the average latitude and longitude
```

```
map_center = site_mobility[['Latitude', 'Longitude']].mean().values
m = folium.Map(location=map_center, zoom_start=12)

# Add a heatmap for each type of location
heatmaps = {
    'Car Park Facilities': car_park_locations,
    'Cycle Park Facilities': cycle_park_locations,
    'Bike Hire Facilities': bike_hire_locations,
    'Repair Workshop Facilities': repair_workshop_locations
}

for name, locations in heatmaps.items():
    HeatMap(locations, name=name, radius=15).add_to(m)

# Add layer control to toggle between heatmaps
folium.LayerControl().add_to(m)

# Display the map
m
```

Out[33]: Make this Notebook Trusted to load map: File -> Trust Notebook

An interactive map displaying markers for each site, with popups showing the car parks, capacity of car parks and cycle parks at each location.

In [34]:
```
map_center = site_mobility[['Latitude', 'Longitude']].mean().values
m = folium.Map(location=map_center, zoom_start=12)

# Add markers to the map for each site
for idx, row in site_mobility.iterrows():
    # Construct a popup message including all relevant variables
    popup_message = folium.Popup(
        f"SiteID: {row['SiteID']}<br>"
        f"Car Park Exists: {'Yes' if row['CarParkExist'] == 1 else 'No'}<br>"
        f"Number of Car Parks: {row['NumCarPark']}<br>"
        f"Cycle Park: {'Yes' if row['CyclePark'] == 1 else 'No'}",
        max_width=300
    )

    # Add the marker to the map
    folium.Marker(
```

```
        [row['Latitude'], row['Longitude']],
        popup=popup_message
    ).add_to(m)

# Display the map
m
```

Out[34]: Make this Notebook Trusted to load map: File -> Trust Notebook

In [35]:

```
def score_mobility(mobility):
    if mobility == 4:
        return 5
    elif mobility == 3:
        return 4
    elif mobility == 2:
        return 3
    elif mobility == 1:
        return 2
    else:
        return 1

site_mobility['mobility_score'] = (
    site_mobility['CarParkExist'] +
    site_mobility['CyclePark'] +
    site_mobility['BikeHireFlag']+
    site_mobility['RepairWorkshopFlag']
)
site_mobility['mobility_score'] = site_mobility['mobility_score'].apply(score_mo
site_mobility.head()
```

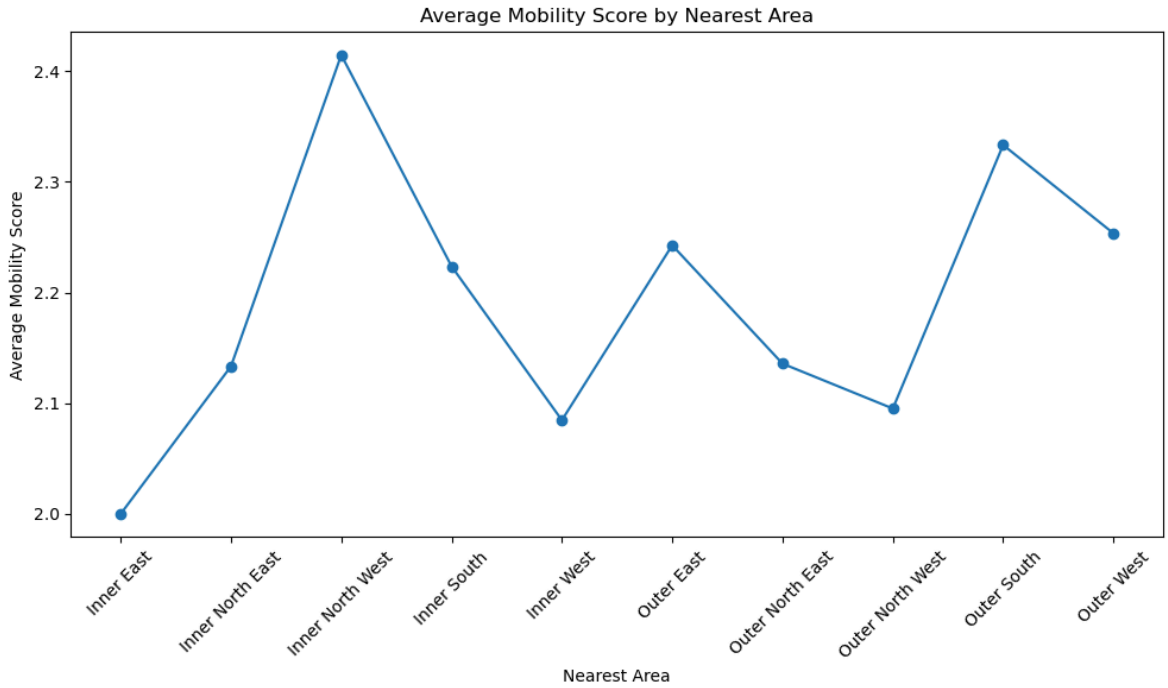
Out[35]:

	SiteID	CarParkExist	NumCarPark	CyclePark	BikeHireFlag	RepairWorkshopFl
0	1000087	1	300	1	0	
1	1000115	1	315	1	0	
2	1000124	1	140	1	0	
3	1000130	1	90	1	0	
4	1000228	1	30	0	0	

In [36]:

```
average_mobility_scores = site_mobility.groupby('Nearest Area')['mobility_score'

fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(average_mobility_scores['Nearest Area'], average_mobility_scores['mobili

ax.set_xlabel('Nearest Area')
ax.set_ylabel('Average Mobility Score')
ax.set_title('Average Mobility Score by Nearest Area')
ax.set_xticklabels(average_mobility_scores['Nearest Area'], rotation=45)

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

```
C:\Users\111\AppData\Local\Temp\ipykernel_22148\3133168800.py:9: UserWarning: Fix
edFormatter should only be used together with FixedLocator
  ax.set_xticklabels(average_mobility_scores['Nearest Area'], rotation=45)
```



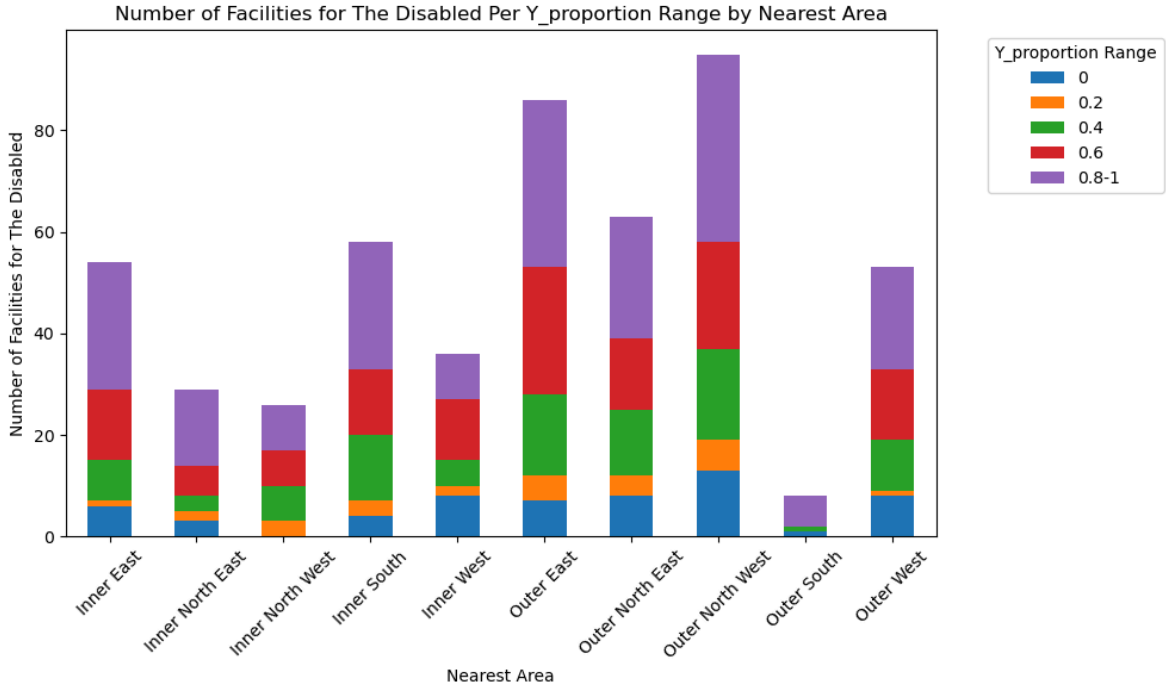
In [37]:
```
site_disabled=pd.merge(site,Geographic,on='SiteID',how='inner')
site_disabled=site_disabled[['SiteID','DisabilityParam','Latitude','Longitude']]
site_disabled=site_disabled.dropna()
# Calculate the ratio of "Y"
site_disabled['Y_proportion'] = site_disabled['DisabilityParam'].apply(lambda x:

site_disabled['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 site_disabled.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
    site_disabled.at[index, 'Nearest Area'] = nearest_area

site_disabled.head()
```

Out[37]:

	SiteID	DisabilityParam	Latitude	Longitude	Y_proportion	Nearest Area
0	1000087	Y,Y,Y,Y,Y,Y,Y,Y,Y,Y	53.834342	-1.589730	1.0	Inner North West
1	1000115	Y,Y,Y,Y,Y,Y,Y,Y,Y,Y	53.843665	-1.550199	1.0	Inner North West
2	1000124	Y,Y,Y,Y,Y,Y,Y,Y,Y,Y	53.867185	-1.688831	1.0	Outer North West
3	1000130	Y,Y,Y,Y,Y,Y,Y,Y,Y,Y	53.862232	-1.611261	1.0	Outer North West
5	1000274	N,N,N,N,N,N,N,N,N,N	53.925079	-1.827912	0.0	Outer North West

In [38]:
```
# Divide Y_proportion to the specified range
site_disabled['Y_proportion_range'] = pd.cut(site_disabled['Y_proportion'], bins
                                             right=False, labels=['0', '0.2', '0

# Group and count the number of each area and Y_proportion interval
grouped = site_disabled.groupby(['Nearest Area', 'Y_proportion_range']).size().u

grouped.plot(kind='bar', stacked=True, figsize=(10, 6))
plt.title('Number of Facilities for The Disabled Per Y_proportion Range by Neare
plt.ylabel('Number of Facilities for The Disabled')
plt.xlabel('Nearest Area')
plt.xticks(rotation=45)
plt.legend(title='Y_proportion Range', bbox_to_anchor=(1.05, 1), loc='upper left
plt.tight_layout()
plt.show()
```

```
C:\Users\111\AppData\Local\Temp\ipykernel_22148\2843668294.py:5: FutureWarning: T
he default of observed=False is deprecated and will be changed to True in a futur
e version of pandas. Pass observed=False to retain current behavior or observed=T
rue to adopt the future default and silence this warning.
  grouped = site_disabled.groupby(['Nearest Area', 'Y_proportion_range']).size().
unstack(fill_value=0)
```



In [39]:
```
def score_disabled(disabled):
    if disabled <=0.2:
        return 1
    elif disabled <=0.4:
        return 2
    elif disabled <=0.5:
        return 3
    elif disabled <=0.8:
        return 4
    else:
        return 5

site_disabled['disabled_score'] = site_disabled['Y_proportion'].apply(score_disa
site_disabled.head()
```

Out[39]:

	SiteID	DisabilityParam	Latitude	Longitude	Y_proportion	Nearest Area	Y_
0	1000087	Y,Y,Y,Y,Y,Y,Y,Y,Y,Y	53.834342	-1.589730	1.0	Inner North West	
1	1000115	Y,Y,Y,Y,Y,Y,Y,Y,Y,Y	53.843665	-1.550199	1.0	Inner North West	
2	1000124	Y,Y,Y,Y,Y,Y,Y,Y,Y,Y	53.867185	-1.688831	1.0	Outer North West	
3	1000130	Y,Y,Y,Y,Y,Y,Y,Y,Y,Y	53.862232	-1.611261	1.0	Outer North West	
5	1000274	N,N,N,N,N,N,N,N,N,N	53.925079	-1.827912	0.0	Outer North West	

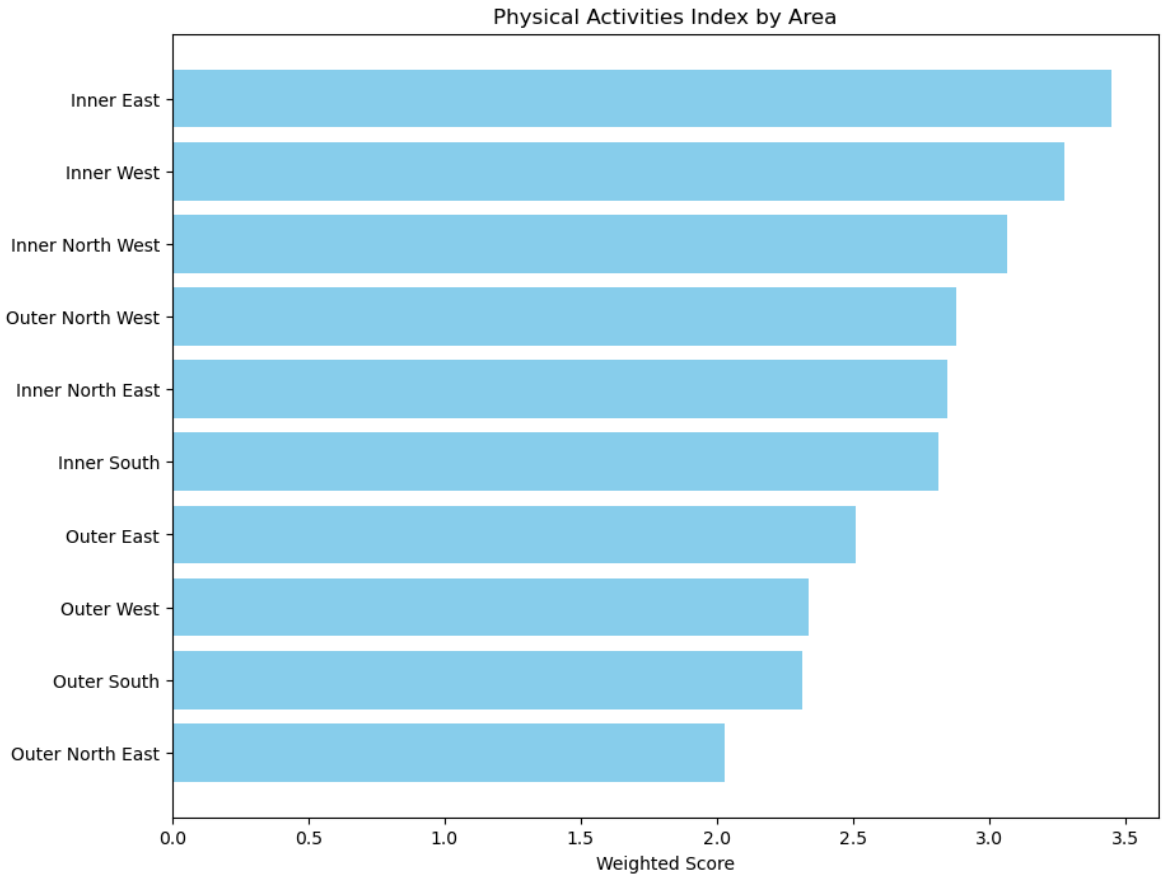
In [40]:

```
# Initialize physical_activities_index DataFrame with areas from areas_coordinat
physical_activities_index = pd.DataFrame({
    'Nearest Area': list(areas_coordinates.keys())
})

# List of scores to calculate averages for
physical_activities_domain = ['density_park_score', 'density_fitness_centre_scor
           'distance_park_score', 'distance_fitness_score', 'distance_sports_sco
           'mobility_score', 'disabled_score']

# Calculate averages for scores related to residential locations and merge them
for column in physical_activities_domain[:6]:
    averages = residential.groupby('Nearest Area')[column].mean().reset_index(na
    physical_activities_index = physical_activities_index.merge(averages, on='Ne

# Calculate the average mobility score from site_mobility and merge into the Dat
averages_mobility = site_mobility.groupby('Nearest Area')['mobility_score'].mean
physical_activities_index = physical_activities_index.merge(averages_mobility, o

# Calculate the average disabled score from site_disabled and merge into the Dat
averages_disabled = site_disabled.groupby('Nearest Area')['disabled_score'].mean
physical_activities_index = physical_activities_index.merge(averages_disabled, o

# Display the first few rows to check the result
physical_activities_index
```

Out[40]:

	Nearest Area	density_park_score	density_fitness_centre_score	density_sports_centre_s
0	Inner East	4.626016	2.235772	1.38
1	Inner North East	3.099174	1.049587	1.10
2	Inner North West	2.816901	1.084507	1.46
3	Inner South	3.429104	1.011194	1.14
4	Inner West	4.008621	1.327586	1.93
5	Outer East	2.638430	1.177686	1.13
6	Outer North East	1.689342	1.040816	1.00
7	Outer North West	3.493802	1.119835	1.27
8	Outer South	2.659341	1.000000	1.09
9	Outer West	1.963415	1.134146	1.14

In [41]:

```
# Define weights
weights = {
    'density_park_score': (1./6.),
    'density_fitness_centre_score': 0.1,
    'density_sports_centre_score': 0.1,
    'distance_park_score': (1./6.),
    'distance_fitness_score': 0.1,
    'distance_sports_score': 0.1,
    'mobility_score': 0.1,
    'disabled_score': (1./6.)
}

# Calculate each domain weighted score
physical_activities_index['weighted_score'] = 0
for index, weight in weights.items():
    physical_activities_index['weighted_score'] += physical_activities_index[ind

# In descending order of score
physical_activities_index_sorted = physical_activities_index.sort_values(by='wei

plt.figure(figsize=(10, 8))
plt.barh(physical_activities_index_sorted['Nearest Area'], physical_activities_i
plt.xlabel('Weighted Score')
plt.title('Physical Activities Index by Area')
```

```
plt.gca().invert_yaxis()
plt.show()
```