In [66]:
```
import numpy as np
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

In [6]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [7]:
```
df_lsoa = pd.read_csv('year_lsoa_grocery.csv')
pd.set_option('display.max_columns', None)
```

In [8]:
```
df_lsoa.head(10)
```

Out[8]:

	area_id	weight	weight_perc2.5	weight_perc25	weight_perc50	weight_perc75
0	E01000001	308.119047	35.0	150.0	250.0	400.0
1	E01000002	313.517874	40.0	150.0	250.0	400.0
2	E01000003	315.084751	35.0	150.0	250.0	400.0
3	E01000005	356.033437	38.0	150.0	280.0	450.0
4	E01000006	451.262063	36.0	180.0	325.0	500.0
5	E01000007	466.567666	30.0	165.0	325.0	500.0
6	E01000008	448.551290	30.0	150.0	300.0	500.0
7	E01000009	483.650451	37.5	170.0	300.0	500.0
8	E01000010	425.420574	34.0	166.4	300.0	500.0
9	E01000011	412.360523	34.0	155.0	300.0	500.0

Task 1: Describe the Dataset

This dataset includes 4,833 records and 202 variables and we have 199 columns of float64 datatype, 2 columns of int64 datatype, and 1 column is categorical or object datatype which is area_id, and luckily in this dataset, we have no missing values. and provides a complete picture of nutritional data, energy content from various food sources, and demographic specifics across different areas (as indicated by area_id). The nutritional information is detailed, including weight, volume, and the macronutrient profile. Each nutritional element is then broken down into statistical measurements (percentiles, standard deviations, and confidence intervals), which provide a more comprehensive picture of consumption patterns. The collection additionally measures the energy contribution of different nutrients and contains categorical consumption statistics for a variety of foods. Demographic information complements nutritional and consumption data by providing insights into each area's population characteristics, such as gender distribution, age groups, average age, and population density.

Data Value

The dataset's relevance stems from its ability to expose complex patterns of nutrient intake and eating habits across various demographics and geographic areas. Researchers can identify correlations between eating habits and demographic characteristics by combining thorough nutritional information with demographic data, which could enhance public health policy, nutritional planning, and targeted interventions. This data can help organizations in the food industry with product development, marketing initiatives, and location-based strategies. Academics and policymakers can use the dataset to investigate public health trends, the efficacy of dietary guidelines, and the influence of socioeconomic factors on nutrition

Limitations, Assumptions, or Biase

- Representativeness: Because the dataset only contains transactions made with a loyalty card, the data may be biased toward a demographic that frequents Tesco and participates in its loyalty program.
-

Geographic and demographic biases: The distribution of Tesco stores, and therefore data, may not consistently cover all locations, potentially biasing the dataset towards regions with higher store concentrations

-

Scope of Data: The dataset excludes internet purchases and purchases done without a loyalty card, which may limit the dataset's coverage of total food consumption trend e

In [9]:
```
ros, cols = df_lsoa.shape
print("Number of Records : ", ros)
print("Number of Attributes : ", cols)
```

```
Number of Records :  4833
Number of Attributes :  202
```

In [10]:
```
# Task 2: Visualzation of DataSet
```

In [11]:
```
column_lists = df_lsoa.columns.tolist()
i = 0
for col in column_lists:
    i+=1
    print(f"Column # {i} Name : {col}")
```

```
Column # 1 Name : area_id
Column # 2 Name : weight
Column # 3 Name : weight_perc2.5
Column # 4 Name : weight_perc25
Column # 5 Name : weight_perc50
Column # 6 Name : weight_perc75
Column # 7 Name : weight_perc97.5
Column # 8 Name : weight_std
Column # 9 Name : weight_ci95
Column # 10 Name : volume
Column # 11 Name : volume_perc2.5
Column # 12 Name : volume_perc25
Column # 13 Name : volume_perc50
Column # 14 Name : volume_perc75
Column # 15 Name : volume_perc97.5
Column # 16 Name : volume_std
Column # 17 Name : volume_ci95
Column # 18 Name : fat
Column # 19 Name : fat_perc2.5
Column # 20 Name : fat_perc25
Column # 21 Name : fat_perc50
Column # 22 Name : fat_perc75
Column # 23 Name : fat_perc97.5
Column # 24 Name : fat_std
Column # 25 Name : fat_ci95
Column # 26 Name : saturate
Column # 27 Name : saturate_perc2.5
Column # 28 Name : saturate_perc25
Column # 29 Name : saturate_perc50
Column # 30 Name : saturate_perc75
Column # 31 Name : saturate_perc97.5
Column # 32 Name : saturate_std
Column # 33 Name : saturate_ci95
Column # 34 Name : salt
Column # 35 Name : salt_perc2.5
Column # 36 Name : salt_perc25
Column # 37 Name : salt_perc50
Column # 38 Name : salt_perc75
Column # 39 Name : salt_perc97.5
Column # 40 Name : salt_std
Column # 41 Name : salt_ci95
Column # 42 Name : sugar
Column # 43 Name : sugar_perc2.5
Column # 44 Name : sugar_perc25
Column # 45 Name : sugar_perc50
Column # 46 Name : sugar_perc75
Column # 47 Name : sugar_perc97.5
Column # 48 Name : sugar_std
Column # 49 Name : sugar_ci95
Column # 50 Name : protein
Column # 51 Name : protein_perc2.5
Column # 52 Name : protein_perc25
Column # 53 Name : protein_perc50
Column # 54 Name : protein_perc75
Column # 55 Name : protein_perc97.5
Column # 56 Name : protein_std
Column # 57 Name : protein_ci95
Column # 58 Name : carb
Column # 59 Name : carb_perc2.5
Column # 60 Name : carb_perc25
```

```
Column # 61 Name : carb_perc50
Column # 62 Name : carb_perc75
Column # 63 Name : carb_perc97.5
Column # 64 Name : carb_std
Column # 65 Name : carb_ci95
Column # 66 Name : fibre
Column # 67 Name : fibre_perc2.5
Column # 68 Name : fibre_perc25
Column # 69 Name : fibre_perc50
Column # 70 Name : fibre_perc75
Column # 71 Name : fibre_perc97.5
Column # 72 Name : fibre_std
Column # 73 Name : fibre_ci95
Column # 74 Name : alcohol
Column # 75 Name : alcohol_perc2.5
Column # 76 Name : alcohol_perc25
Column # 77 Name : alcohol_perc50
Column # 78 Name : alcohol_perc75
Column # 79 Name : alcohol_perc97.5
Column # 80 Name : alcohol_std
Column # 81 Name : alcohol_ci95
Column # 82 Name : energy_fat
Column # 83 Name : energy_fat_perc2.5
Column # 84 Name : energy_fat_perc25
Column # 85 Name : energy_fat_perc50
Column # 86 Name : energy_fat_perc75
Column # 87 Name : energy_fat_perc97.5
Column # 88 Name : energy_fat_std
Column # 89 Name : energy_fat_ci95
Column # 90 Name : energy_saturate
Column # 91 Name : energy_saturate_perc2.5
Column # 92 Name : energy_saturate_perc25
Column # 93 Name : energy_saturate_perc50
Column # 94 Name : energy_saturate_perc75
Column # 95 Name : energy_saturate_perc97.5
Column # 96 Name : energy_saturate_std
Column # 97 Name : energy_saturate_ci95
Column # 98 Name : energy_sugar
Column # 99 Name : energy_sugar_perc2.5
Column # 100 Name : energy_sugar_perc25
Column # 101 Name : energy_sugar_perc50
Column # 102 Name : energy_sugar_perc75
Column # 103 Name : energy_sugar_perc97.5
Column # 104 Name : energy_sugar_std
Column # 105 Name : energy_sugar_ci95
Column # 106 Name : energy_protein
Column # 107 Name : energy_protein_perc2.5
Column # 108 Name : energy_protein_perc25
Column # 109 Name : energy_protein_perc50
Column # 110 Name : energy_protein_perc75
Column # 111 Name : energy_protein_perc97.5
Column # 112 Name : energy_protein_std
Column # 113 Name : energy_protein_ci95
Column # 114 Name : energy_carb
Column # 115 Name : energy_carb_perc2.5
Column # 116 Name : energy_carb_perc25
Column # 117 Name : energy_carb_perc50
Column # 118 Name : energy_carb_perc75
Column # 119 Name : energy_carb_perc97.5
Column # 120 Name : energy_carb_std
```

```
Column # 121 Name : energy_carb_ci95
Column # 122 Name : energy_fibre
Column # 123 Name : energy_fibre_perc2.5
Column # 124 Name : energy_fibre_perc25
Column # 125 Name : energy_fibre_perc50
Column # 126 Name : energy_fibre_perc75
Column # 127 Name : energy_fibre_perc97.5
Column # 128 Name : energy_fibre_std
Column # 129 Name : energy_fibre_ci95
Column # 130 Name : energy_alcohol
Column # 131 Name : energy_alcohol_perc2.5
Column # 132 Name : energy_alcohol_perc25
Column # 133 Name : energy_alcohol_perc50
Column # 134 Name : energy_alcohol_perc75
Column # 135 Name : energy_alcohol_perc97.5
Column # 136 Name : energy_alcohol_std
Column # 137 Name : energy_alcohol_ci95
Column # 138 Name : energy_tot
Column # 139 Name : energy_tot_perc2.5
Column # 140 Name : energy_tot_perc25
Column # 141 Name : energy_tot_perc50
Column # 142 Name : energy_tot_perc75
Column # 143 Name : energy_tot_perc97.5
Column # 144 Name : energy_tot_std
Column # 145 Name : energy_tot_ci95
Column # 146 Name : f_energy_fat
Column # 147 Name : f_energy_saturate
Column # 148 Name : f_energy_sugar
Column # 149 Name : f_energy_protein
Column # 150 Name : f_energy_carb
Column # 151 Name : f_energy_fibre
Column # 152 Name : f_energy_alcohol
Column # 153 Name : energy_density
Column # 154 Name : h_nutrients_weight
Column # 155 Name : h_nutrients_weight_norm
Column # 156 Name : h_nutrients_calories
Column # 157 Name : h_nutrients_calories_norm
Column # 158 Name : f_beer
Column # 159 Name : f_dairy
Column # 160 Name : f_eggs
Column # 161 Name : f_fats_oils
Column # 162 Name : f_fish
Column # 163 Name : f_fruit_veg
Column # 164 Name : f_grains
Column # 165 Name : f_meat_red
Column # 166 Name : f_poultry
Column # 167 Name : f_readymade
Column # 168 Name : f_sauces
Column # 169 Name : f_soft_drinks
Column # 170 Name : f_spirits
Column # 171 Name : f_sweets
Column # 172 Name : f_tea_coffee
Column # 173 Name : f_water
Column # 174 Name : f_wine
Column # 175 Name : f_dairy_weight
Column # 176 Name : f_eggs_weight
Column # 177 Name : f_fats_oils_weight
Column # 178 Name : f_fish_weight
Column # 179 Name : f_fruit_veg_weight
Column # 180 Name : f_grains_weight
```

```
Column # 181 Name : f_meat_red_weight
Column # 182 Name : f_poultry_weight
Column # 183 Name : f_readymade_weight
Column # 184 Name : f_sauces_weight
Column # 185 Name : f_sweets_weight
Column # 186 Name : h_items
Column # 187 Name : h_items_norm
Column # 188 Name : h_items_weight
Column # 189 Name : h_items_weight_norm
Column # 190 Name : representativeness_norm
Column # 191 Name : transaction_days
Column # 192 Name : num_transactions
Column # 193 Name : man_day
Column # 194 Name : population
Column # 195 Name : male
Column # 196 Name : female
Column # 197 Name : age_0_17
Column # 198 Name : age_18_64
Column # 199 Name : age_65+
Column # 200 Name : avg_age
Column # 201 Name : area_sq_km
Column # 202 Name : people_per_sq_km
```

In [12]:

```
df_lsoa.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4833 entries, 0 to 4832
Columns: 202 entries, area_id to people_per_sq_km
dtypes: float64(199), int64(2), object(1)
memory usage: 7.4+ MB
```

In [ ]:

1. Dataset Explanation

I wanted to know the shape of the dataset for further processing, so in this dataset I have 4833 number of observation and 202 columns/features

In [13]:

```
ros, cols = df_lsoa.shape
print("Number of Records : ", ros)
print("Number of Attributes : ", cols)
```

```
Number of Records :  4833
Number of Attributes :  202
```

List the number of columns/feature which I have for this dataset.

In [14]:

```
column_lists = df_lsoa.columns.tolist()
i = 0
for col in column_lists:
    i+=1
    print(f"Column # {i} Name : {col}")
```

```
Column # 1 Name : area_id
Column # 2 Name : weight
Column # 3 Name : weight_perc2.5
Column # 4 Name : weight_perc25
Column # 5 Name : weight_perc50
Column # 6 Name : weight_perc75
Column # 7 Name : weight_perc97.5
Column # 8 Name : weight_std
Column # 9 Name : weight_ci95
Column # 10 Name : volume
Column # 11 Name : volume_perc2.5
Column # 12 Name : volume_perc25
Column # 13 Name : volume_perc50
Column # 14 Name : volume_perc75
Column # 15 Name : volume_perc97.5
Column # 16 Name : volume_std
Column # 17 Name : volume_ci95
Column # 18 Name : fat
Column # 19 Name : fat_perc2.5
Column # 20 Name : fat_perc25
Column # 21 Name : fat_perc50
Column # 22 Name : fat_perc75
Column # 23 Name : fat_perc97.5
Column # 24 Name : fat_std
Column # 25 Name : fat_ci95
Column # 26 Name : saturate
Column # 27 Name : saturate_perc2.5
Column # 28 Name : saturate_perc25
Column # 29 Name : saturate_perc50
Column # 30 Name : saturate_perc75
Column # 31 Name : saturate_perc97.5
Column # 32 Name : saturate_std
Column # 33 Name : saturate_ci95
Column # 34 Name : salt
Column # 35 Name : salt_perc2.5
Column # 36 Name : salt_perc25
Column # 37 Name : salt_perc50
Column # 38 Name : salt_perc75
Column # 39 Name : salt_perc97.5
Column # 40 Name : salt_std
Column # 41 Name : salt_ci95
Column # 42 Name : sugar
Column # 43 Name : sugar_perc2.5
Column # 44 Name : sugar_perc25
Column # 45 Name : sugar_perc50
Column # 46 Name : sugar_perc75
Column # 47 Name : sugar_perc97.5
Column # 48 Name : sugar_std
Column # 49 Name : sugar_ci95
Column # 50 Name : protein
Column # 51 Name : protein_perc2.5
Column # 52 Name : protein_perc25
Column # 53 Name : protein_perc50
Column # 54 Name : protein_perc75
Column # 55 Name : protein_perc97.5
Column # 56 Name : protein_std
Column # 57 Name : protein_ci95
Column # 58 Name : carb
Column # 59 Name : carb_perc2.5
Column # 60 Name : carb_perc25
```

```
Column # 61 Name : carb_perc50
Column # 62 Name : carb_perc75
Column # 63 Name : carb_perc97.5
Column # 64 Name : carb_std
Column # 65 Name : carb_ci95
Column # 66 Name : fibre
Column # 67 Name : fibre_perc2.5
Column # 68 Name : fibre_perc25
Column # 69 Name : fibre_perc50
Column # 70 Name : fibre_perc75
Column # 71 Name : fibre_perc97.5
Column # 72 Name : fibre_std
Column # 73 Name : fibre_ci95
Column # 74 Name : alcohol
Column # 75 Name : alcohol_perc2.5
Column # 76 Name : alcohol_perc25
Column # 77 Name : alcohol_perc50
Column # 78 Name : alcohol_perc75
Column # 79 Name : alcohol_perc97.5
Column # 80 Name : alcohol_std
Column # 81 Name : alcohol_ci95
Column # 82 Name : energy_fat
Column # 83 Name : energy_fat_perc2.5
Column # 84 Name : energy_fat_perc25
Column # 85 Name : energy_fat_perc50
Column # 86 Name : energy_fat_perc75
Column # 87 Name : energy_fat_perc97.5
Column # 88 Name : energy_fat_std
Column # 89 Name : energy_fat_ci95
Column # 90 Name : energy_saturate
Column # 91 Name : energy_saturate_perc2.5
Column # 92 Name : energy_saturate_perc25
Column # 93 Name : energy_saturate_perc50
Column # 94 Name : energy_saturate_perc75
Column # 95 Name : energy_saturate_perc97.5
Column # 96 Name : energy_saturate_std
Column # 97 Name : energy_saturate_ci95
Column # 98 Name : energy_sugar
Column # 99 Name : energy_sugar_perc2.5
Column # 100 Name : energy_sugar_perc25
Column # 101 Name : energy_sugar_perc50
Column # 102 Name : energy_sugar_perc75
Column # 103 Name : energy_sugar_perc97.5
Column # 104 Name : energy_sugar_std
Column # 105 Name : energy_sugar_ci95
Column # 106 Name : energy_protein
Column # 107 Name : energy_protein_perc2.5
Column # 108 Name : energy_protein_perc25
Column # 109 Name : energy_protein_perc50
Column # 110 Name : energy_protein_perc75
Column # 111 Name : energy_protein_perc97.5
Column # 112 Name : energy_protein_std
Column # 113 Name : energy_protein_ci95
Column # 114 Name : energy_carb
Column # 115 Name : energy_carb_perc2.5
Column # 116 Name : energy_carb_perc25
Column # 117 Name : energy_carb_perc50
Column # 118 Name : energy_carb_perc75
Column # 119 Name : energy_carb_perc97.5
Column # 120 Name : energy_carb_std
```

```
Column # 121 Name : energy_carb_ci95
Column # 122 Name : energy_fibre
Column # 123 Name : energy_fibre_perc2.5
Column # 124 Name : energy_fibre_perc25
Column # 125 Name : energy_fibre_perc50
Column # 126 Name : energy_fibre_perc75
Column # 127 Name : energy_fibre_perc97.5
Column # 128 Name : energy_fibre_std
Column # 129 Name : energy_fibre_ci95
Column # 130 Name : energy_alcohol
Column # 131 Name : energy_alcohol_perc2.5
Column # 132 Name : energy_alcohol_perc25
Column # 133 Name : energy_alcohol_perc50
Column # 134 Name : energy_alcohol_perc75
Column # 135 Name : energy_alcohol_perc97.5
Column # 136 Name : energy_alcohol_std
Column # 137 Name : energy_alcohol_ci95
Column # 138 Name : energy_tot
Column # 139 Name : energy_tot_perc2.5
Column # 140 Name : energy_tot_perc25
Column # 141 Name : energy_tot_perc50
Column # 142 Name : energy_tot_perc75
Column # 143 Name : energy_tot_perc97.5
Column # 144 Name : energy_tot_std
Column # 145 Name : energy_tot_ci95
Column # 146 Name : f_energy_fat
Column # 147 Name : f_energy_saturate
Column # 148 Name : f_energy_sugar
Column # 149 Name : f_energy_protein
Column # 150 Name : f_energy_carb
Column # 151 Name : f_energy_fibre
Column # 152 Name : f_energy_alcohol
Column # 153 Name : energy_density
Column # 154 Name : h_nutrients_weight
Column # 155 Name : h_nutrients_weight_norm
Column # 156 Name : h_nutrients_calories
Column # 157 Name : h_nutrients_calories_norm
Column # 158 Name : f_beer
Column # 159 Name : f_dairy
Column # 160 Name : f_eggs
Column # 161 Name : f_fats_oils
Column # 162 Name : f_fish
Column # 163 Name : f_fruit_veg
Column # 164 Name : f_grains
Column # 165 Name : f_meat_red
Column # 166 Name : f_poultry
Column # 167 Name : f_readymade
Column # 168 Name : f_sauces
Column # 169 Name : f_soft_drinks
Column # 170 Name : f_spirits
Column # 171 Name : f_sweets
Column # 172 Name : f_tea_coffee
Column # 173 Name : f_water
Column # 174 Name : f_wine
Column # 175 Name : f_dairy_weight
Column # 176 Name : f_eggs_weight
Column # 177 Name : f_fats_oils_weight
Column # 178 Name : f_fish_weight
Column # 179 Name : f_fruit_veg_weight
Column # 180 Name : f_grains_weight
```

```
Column # 181 Name : f_meat_red_weight
Column # 182 Name : f_poultry_weight
Column # 183 Name : f_readymade_weight
Column # 184 Name : f_sauces_weight
Column # 185 Name : f_sweets_weight
Column # 186 Name : h_items
Column # 187 Name : h_items_norm
Column # 188 Name : h_items_weight
Column # 189 Name : h_items_weight_norm
Column # 190 Name : representativeness_norm
Column # 191 Name : transaction_days
Column # 192 Name : num_transactions
Column # 193 Name : man_day
Column # 194 Name : population
Column # 195 Name : male
Column # 196 Name : female
Column # 197 Name : age_0_17
Column # 198 Name : age_18_64
Column # 199 Name : age_65+
Column # 200 Name : avg_age
Column # 201 Name : area_sq_km
Column # 202 Name : people_per_sq_km
```

Here I wanted to know the datatype for each column so, that will be easy for us for further preprocessing here we are not getting to much usefull information about the dataset using the .info() method here we only getting the points in our dataset we have the columns of datatypes (int64, float64 and, object)

In [15]:
```
df_lsoa.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4833 entries, 0 to 4832
Columns: 202 entries, area_id to people_per_sq_km
dtypes: float64(199), int64(2), object(1)
memory usage: 7.4+ MB
```

In [16]:
```
print(df_lsoa.dtypes)
```

```
area_id               object
weight               float64
weight_perc2.5       float64
weight_perc25        float64
weight_perc50        float64
                      ...
age_18_64            float64
age_65+              float64
avg_age              float64
area_sq_km           float64
people_per_sq_km     float64
Length: 202, dtype: object
```

Here we are trying to getting the complete list of the columns with datatypes for each colums, so there is 199 columns have the (float64) datatype, 2 columns have the (int64) and 1 column have the datatype of the (object) which is 'area_id', so in total we have the 202 columns as we know.

In [17]:
```
for column_name, dtype in df_lsoa.dtypes.items():
    print(f"{column_name}: {dtype}")
```

```
area_id: object
weight: float64
weight_perc2.5: float64
weight_perc25: float64
weight_perc50: float64
weight_perc75: float64
weight_perc97.5: float64
weight_std: float64
weight_ci95: float64
volume: float64
volume_perc2.5: float64
volume_perc25: float64
volume_perc50: float64
volume_perc75: float64
volume_perc97.5: float64
volume_std: float64
volume_ci95: float64
fat: float64
fat_perc2.5: float64
fat_perc25: float64
fat_perc50: float64
fat_perc75: float64
fat_perc97.5: float64
fat_std: float64
fat_ci95: float64
saturate: float64
saturate_perc2.5: float64
saturate_perc25: float64
saturate_perc50: float64
saturate_perc75: float64
saturate_perc97.5: float64
saturate_std: float64
saturate_ci95: float64
salt: float64
salt_perc2.5: float64
salt_perc25: float64
salt_perc50: float64
salt_perc75: float64
salt_perc97.5: float64
salt_std: float64
salt_ci95: float64
sugar: float64
sugar_perc2.5: float64
sugar_perc25: float64
sugar_perc50: float64
sugar_perc75: float64
sugar_perc97.5: float64
sugar_std: float64
sugar_ci95: float64
protein: float64
protein_perc2.5: float64
protein_perc25: float64
protein_perc50: float64
protein_perc75: float64
protein_perc97.5: float64
protein_std: float64
protein_ci95: float64
carb: float64
carb_perc2.5: float64
carb_perc25: float64
```

```
carb_perc50: float64
carb_perc75: float64
carb_perc97.5: float64
carb_std: float64
carb_ci95: float64
fibre: float64
fibre_perc2.5: float64
fibre_perc25: float64
fibre_perc50: float64
fibre_perc75: float64
fibre_perc97.5: float64
fibre_std: float64
fibre_ci95: float64
alcohol: float64
alcohol_perc2.5: float64
alcohol_perc25: float64
alcohol_perc50: float64
alcohol_perc75: float64
alcohol_perc97.5: float64
alcohol_std: float64
alcohol_ci95: float64
energy_fat: float64
energy_fat_perc2.5: float64
energy_fat_perc25: float64
energy_fat_perc50: float64
energy_fat_perc75: float64
energy_fat_perc97.5: float64
energy_fat_std: float64
energy_fat_ci95: float64
energy_saturate: float64
energy_saturate_perc2.5: float64
energy_saturate_perc25: float64
energy_saturate_perc50: float64
energy_saturate_perc75: float64
energy_saturate_perc97.5: float64
energy_saturate_std: float64
energy_saturate_ci95: float64
energy_sugar: float64
energy_sugar_perc2.5: float64
energy_sugar_perc25: float64
energy_sugar_perc50: float64
energy_sugar_perc75: float64
energy_sugar_perc97.5: float64
energy_sugar_std: float64
energy_sugar_ci95: float64
energy_protein: float64
energy_protein_perc2.5: float64
energy_protein_perc25: float64
energy_protein_perc50: float64
energy_protein_perc75: float64
energy_protein_perc97.5: float64
energy_protein_std: float64
energy_protein_ci95: float64
energy_carb: float64
energy_carb_perc2.5: float64
energy_carb_perc25: float64
energy_carb_perc50: float64
energy_carb_perc75: float64
energy_carb_perc97.5: float64
energy_carb_std: float64
```

```
energy_carb_ci95: float64
energy_fibre: float64
energy_fibre_perc2.5: float64
energy_fibre_perc25: float64
energy_fibre_perc50: float64
energy_fibre_perc75: float64
energy_fibre_perc97.5: float64
energy_fibre_std: float64
energy_fibre_ci95: float64
energy_alcohol: float64
energy_alcohol_perc2.5: float64
energy_alcohol_perc25: float64
energy_alcohol_perc50: float64
energy_alcohol_perc75: float64
energy_alcohol_perc97.5: float64
energy_alcohol_std: float64
energy_alcohol_ci95: float64
energy_tot: float64
energy_tot_perc2.5: float64
energy_tot_perc25: float64
energy_tot_perc50: float64
energy_tot_perc75: float64
energy_tot_perc97.5: float64
energy_tot_std: float64
energy_tot_ci95: float64
f_energy_fat: float64
f_energy_saturate: float64
f_energy_sugar: float64
f_energy_protein: float64
f_energy_carb: float64
f_energy_fibre: float64
f_energy_alcohol: float64
energy_density: float64
h_nutrients_weight: float64
h_nutrients_weight_norm: float64
h_nutrients_calories: float64
h_nutrients_calories_norm: float64
f_beer: float64
f_dairy: float64
f_eggs: float64
f_fats_oils: float64
f_fish: float64
f_fruit_veg: float64
f_grains: float64
f_meat_red: float64
f_poultry: float64
f_readymade: float64
f_sauces: float64
f_soft_drinks: float64
f_spirits: float64
f_sweets: float64
f_tea_coffee: float64
f_water: float64
f_wine: float64
f_dairy_weight: float64
f_eggs_weight: float64
f_fats_oils_weight: float64
f_fish_weight: float64
f_fruit_veg_weight: float64
f_grains_weight: float64
```

```
f_meat_red_weight: float64
f_poultry_weight: float64
f_readymade_weight: float64
f_sauces_weight: float64
f_sweets_weight: float64
h_items: float64
h_items_norm: float64
h_items_weight: float64
h_items_weight_norm: float64
representativeness_norm: float64
transaction_days: int64
num_transactions: float64
man_day: int64
population: float64
male: float64
female: float64
age_0_17: float64
age_18_64: float64
age_65+: float64
avg_age: float64
area_sq_km: float64
people_per_sq_km: float64
```

2. Dataset Preprocessing

Now, First inspect that is there any missing value in our dataset

In [18]:
```
df_lsoa.isnull().sum()
```

Out[18]:
```
area_id              0
weight               0
weight_perc2.5       0
weight_perc25        0
weight_perc50        0
                    ..
age_18_64            0
age_65+              0
avg_age              0
area_sq_km           0
people_per_sq_km     0
Length: 202, dtype: int64
```

Here you can inspect that there is not missing value in our dataset column by column, but we have 202 features, so it's hard to go through the all features one by one.

In [19]:
```
missing_values_count = df_lsoa.isnull().sum()
for column_name, missing_count in missing_values_count.items():
    print(f"{column_name}: {missing_count}")
```

```
area_id: 0
weight: 0
weight_perc2.5: 0
weight_perc25: 0
weight_perc50: 0
weight_perc75: 0
weight_perc97.5: 0
weight_std: 0
weight_ci95: 0
volume: 0
volume_perc2.5: 0
volume_perc25: 0
volume_perc50: 0
volume_perc75: 0
volume_perc97.5: 0
volume_std: 0
volume_ci95: 0
fat: 0
fat_perc2.5: 0
fat_perc25: 0
fat_perc50: 0
fat_perc75: 0
fat_perc97.5: 0
fat_std: 0
fat_ci95: 0
saturate: 0
saturate_perc2.5: 0
saturate_perc25: 0
saturate_perc50: 0
saturate_perc75: 0
saturate_perc97.5: 0
saturate_std: 0
saturate_ci95: 0
salt: 0
salt_perc2.5: 0
salt_perc25: 0
salt_perc50: 0
salt_perc75: 0
salt_perc97.5: 0
salt_std: 0
salt_ci95: 0
sugar: 0
sugar_perc2.5: 0
sugar_perc25: 0
sugar_perc50: 0
sugar_perc75: 0
sugar_perc97.5: 0
sugar_std: 0
sugar_ci95: 0
protein: 0
protein_perc2.5: 0
protein_perc25: 0
protein_perc50: 0
protein_perc75: 0
protein_perc97.5: 0
protein_std: 0
protein_ci95: 0
carb: 0
carb_perc2.5: 0
carb_perc25: 0
```

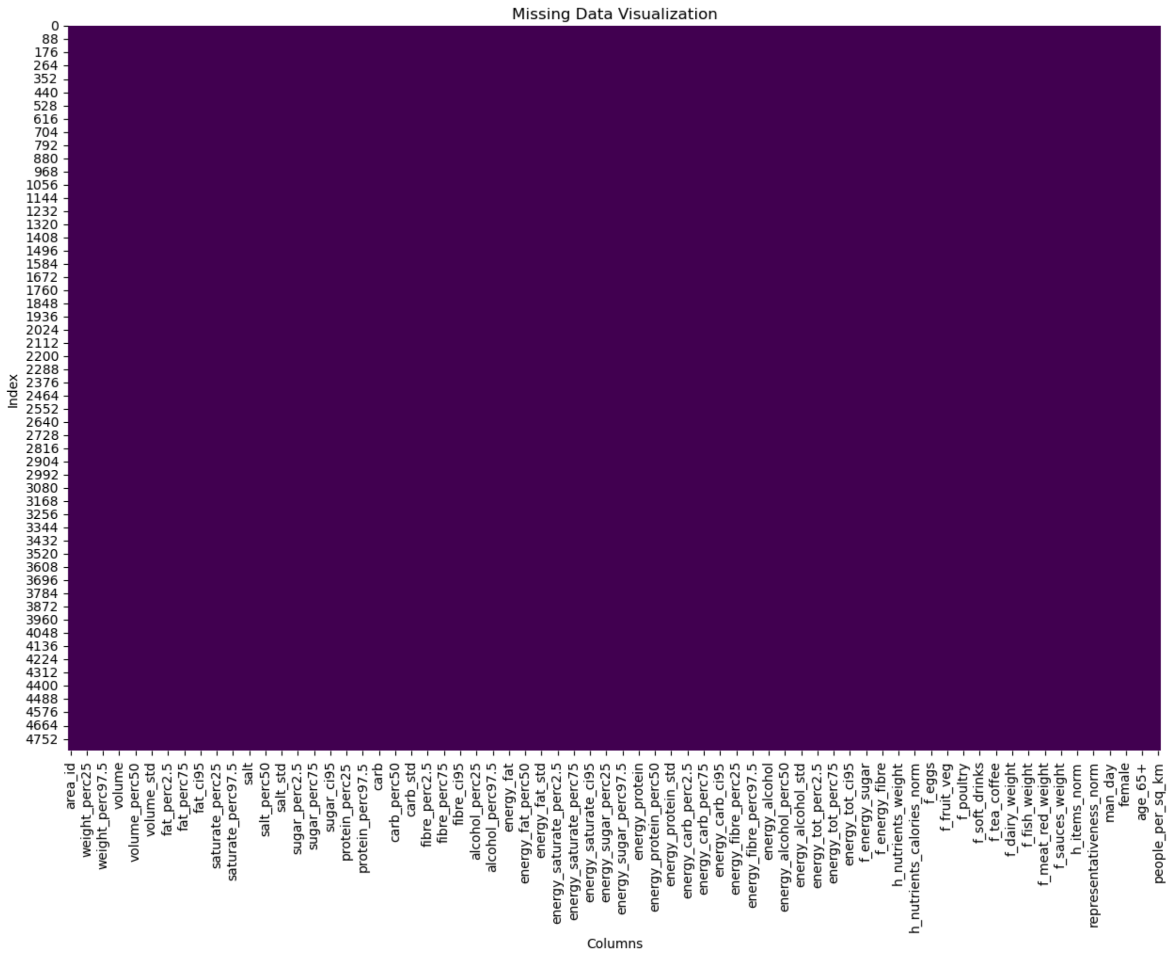
```
carb_perc50: 0
carb_perc75: 0
carb_perc97.5: 0
carb_std: 0
carb_ci95: 0
fibre: 0
fibre_perc2.5: 0
fibre_perc25: 0
fibre_perc50: 0
fibre_perc75: 0
fibre_perc97.5: 0
fibre_std: 0
fibre_ci95: 0
alcohol: 0
alcohol_perc2.5: 0
alcohol_perc25: 0
alcohol_perc50: 0
alcohol_perc75: 0
alcohol_perc97.5: 0
alcohol_std: 0
alcohol_ci95: 0
energy_fat: 0
energy_fat_perc2.5: 0
energy_fat_perc25: 0
energy_fat_perc50: 0
energy_fat_perc75: 0
energy_fat_perc97.5: 0
energy_fat_std: 0
energy_fat_ci95: 0
energy_saturate: 0
energy_saturate_perc2.5: 0
energy_saturate_perc25: 0
energy_saturate_perc50: 0
energy_saturate_perc75: 0
energy_saturate_perc97.5: 0
energy_saturate_std: 0
energy_saturate_ci95: 0
energy_sugar: 0
energy_sugar_perc2.5: 0
energy_sugar_perc25: 0
energy_sugar_perc50: 0
energy_sugar_perc75: 0
energy_sugar_perc97.5: 0
energy_sugar_std: 0
energy_sugar_ci95: 0
energy_protein: 0
energy_protein_perc2.5: 0
energy_protein_perc25: 0
energy_protein_perc50: 0
energy_protein_perc75: 0
energy_protein_perc97.5: 0
energy_protein_std: 0
energy_protein_ci95: 0
energy_carb: 0
energy_carb_perc2.5: 0
energy_carb_perc25: 0
energy_carb_perc50: 0
energy_carb_perc75: 0
energy_carb_perc97.5: 0
energy_carb_std: 0
```

```
energy_carb_ci95: 0
energy_fibre: 0
energy_fibre_perc2.5: 0
energy_fibre_perc25: 0
energy_fibre_perc50: 0
energy_fibre_perc75: 0
energy_fibre_perc97.5: 0
energy_fibre_std: 0
energy_fibre_ci95: 0
energy_alcohol: 0
energy_alcohol_perc2.5: 0
energy_alcohol_perc25: 0
energy_alcohol_perc50: 0
energy_alcohol_perc75: 0
energy_alcohol_perc97.5: 0
energy_alcohol_std: 0
energy_alcohol_ci95: 0
energy_tot: 0
energy_tot_perc2.5: 0
energy_tot_perc25: 0
energy_tot_perc50: 0
energy_tot_perc75: 0
energy_tot_perc97.5: 0
energy_tot_std: 0
energy_tot_ci95: 0
f_energy_fat: 0
f_energy_saturate: 0
f_energy_sugar: 0
f_energy_protein: 0
f_energy_carb: 0
f_energy_fibre: 0
f_energy_alcohol: 0
energy_density: 0
h_nutrients_weight: 0
h_nutrients_weight_norm: 0
h_nutrients_calories: 0
h_nutrients_calories_norm: 0
f_beer: 0
f_dairy: 0
f_eggs: 0
f_fats_oils: 0
f_fish: 0
f_fruit_veg: 0
f_grains: 0
f_meat_red: 0
f_poultry: 0
f_readymade: 0
f_sauces: 0
f_soft_drinks: 0
f_spirits: 0
f_sweets: 0
f_tea_coffee: 0
f_water: 0
f_wine: 0
f_dairy_weight: 0
f_eggs_weight: 0
f_fats_oils_weight: 0
f_fish_weight: 0
f_fruit_veg_weight: 0
f_grains_weight: 0
```

```
f_meat_red_weight: 0
f_poultry_weight: 0
f_readymade_weight: 0
f_sauces_weight: 0
f_sweets_weight: 0
h_items: 0
h_items_norm: 0
h_items_weight: 0
h_items_weight_norm: 0
representativeness_norm: 0
transaction_days: 0
num_transactions: 0
man_day: 0
population: 0
male: 0
female: 0
age_0_17: 0
age_18_64: 0
age_65+: 0
avg_age: 0
area_sq_km: 0
people_per_sq_km: 0
```

Here visually we can clearly see that there is no missing value in our dataset, which good thing so we are good to go for further processing

In [20]:
```
plt.figure(figsize=(15, 10))
sns.heatmap(df_lsoa.isnull(), cbar=False, cmap='viridis')
plt.title('Missing Data Visualization')
plt.xlabel('Columns')
plt.ylabel('Index')
plt.show()
```




```
In [21]: df_lsoa.describe()
```

Out[21]:

	weight	weight_perc2.5	weight_perc25	weight_perc50	weight_perc75	weig
count	4833.000000	4833.000000	4833.000000	4833.000000	4833.000000	
mean	371.573671	34.821362	155.787570	281.840368	461.903735	
std	52.847517	5.109351	23.762588	42.463191	48.102601	
min	164.405101	11.000000	44.000000	52.000000	174.000000	
25%	334.434314	30.000000	150.000000	250.000000	425.000000	
50%	373.254063	35.000000	154.000000	296.000000	480.000000	
75%	408.917176	40.000000	174.000000	300.000000	500.000000	
max	745.264297	60.000000	400.000000	500.000000	1500.000000	

Here From above descriptive statistics of our dataset I have notice that there is some of the variable for percentile25 there is no mean, std_deviation, min and max value and they all are zeros, which might be the case that the 25percentiles of the columns is started from 0,

Task 2: Visualization of Dataset

In [22]:
```
bins = [0, 17, 64, float('inf')]  # Adjust according to how you want to bin ages
labels = ['0-17', '18-64', '65+']

# Categorize avg_age into age groups
df_lsoa['age_group'] = pd.cut(df_lsoa['avg_age'], bins=bins, labels=labels, righ
```

I categorize the age_group based on the ave_age, so in our dataset we have the 3 age groups, (0-17, 18-64, and 65+),

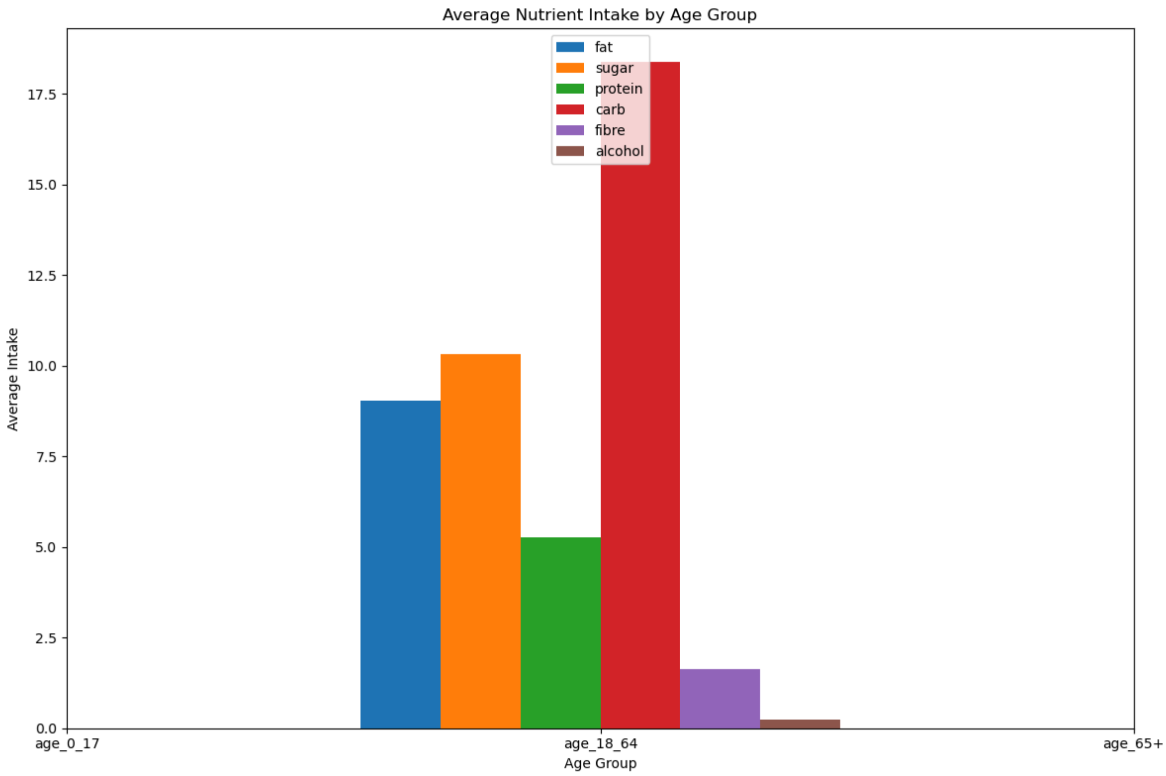
In [23]:
```
df_lsoa.head()
```

Out[23]:

	area_id	weight	weight_perc2.5	weight_perc25	weight_perc50	weight_perc75
0	E01000001	308.119047	35.0	150.0	250.0	400.0
1	E01000002	313.517874	40.0	150.0	250.0	400.0
2	E01000003	315.084751	35.0	150.0	250.0	400.0
3	E01000005	356.033437	38.0	150.0	280.0	450.0
4	E01000006	451.262063	36.0	180.0	325.0	500.0

In [24]:
```
# Calculate average nutrient intake by age group using a list for column selecti
average_nutrient_intake = df_lsoa.groupby('age_group')[['fat', 'sugar', 'protein
```

In [25]:
```
nutrients = ['fat', 'sugar', 'protein', 'carb', 'fibre', 'alcohol']
colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b']
age_groups = ['age_0_17', 'age_18_64', 'age_65+']
# Setting the positions for the bars
bar_width = 0.15  # Width of bars
n_bars = len(nutrients)  # Number of nutrients
# The x position of bars
bar_positions = np.arange(len(age_groups))
fig, ax = plt.subplots(figsize=(12, 8))
# Create bars for each nutrient
for i, nutrient in enumerate(nutrients):
    # Calculate the position for each bar
    positions = bar_positions + (i - n_bars / 2) * bar_width + bar_width / 2
    # Plotting each nutrient bar with its respective color
    ax.bar(positions, average_nutrient_intake[nutrient], width=bar_width, label=
ax.set_xticks(bar_positions)
ax.set_xticklabels(age_groups)
ax.legend()
# Adding labels and title
ax.set_xlabel('Age Group')
ax.set_ylabel('Average Intake')
ax.set_title('Average Nutrient Intake by Age Group')
plt.tight_layout()
plt.show()
```



Through visually we can see that the Most average Nutrient Intake by age group is 18-64, and so they people intake alcohol very lesser amount which is good because alcohol is not good for health, and on the other hand we also notice that they are not likely to intake the protein in higher amount, as compared to fat and sugar which is not good for health, so which means according to these insight the people are more like to intake the fat and sugar.

In [26]:
```
nutrients = ['fat', 'sugar', 'protein', 'carb', 'fibre', 'alcohol']
demographics = ['population', 'age_0_17', 'age_18_64', 'age_65+', 'avg_age']
# Creating a new DataFrame with selected columns for the correlation analysis
df_prepared = df_lsoa[nutrients + demographics]
```

In [27]:
```
df_prepared.head()
```

Out[27]:

	fat	sugar	protein	carb	fibre	alcohol	population	age_0_17
0	8.535149	9.213734	5.262429	15.158014	1.622653	0.339168	1296.0	179.0
1	8.054729	8.337412	5.351774	14.358466	1.692822	0.429261	1156.0	197.0
2	8.153757	9.414937	5.029519	15.820254	1.522523	0.521810	1350.0	152.0
3	8.339058	9.603258	5.230254	17.126487	1.612862	0.255560	1121.0	294.0
4	9.622101	11.355115	5.026295	19.903063	1.640227	0.138525	2040.0	563.0

In [28]:
```
correlation_matrix = df_prepared.corr()
```

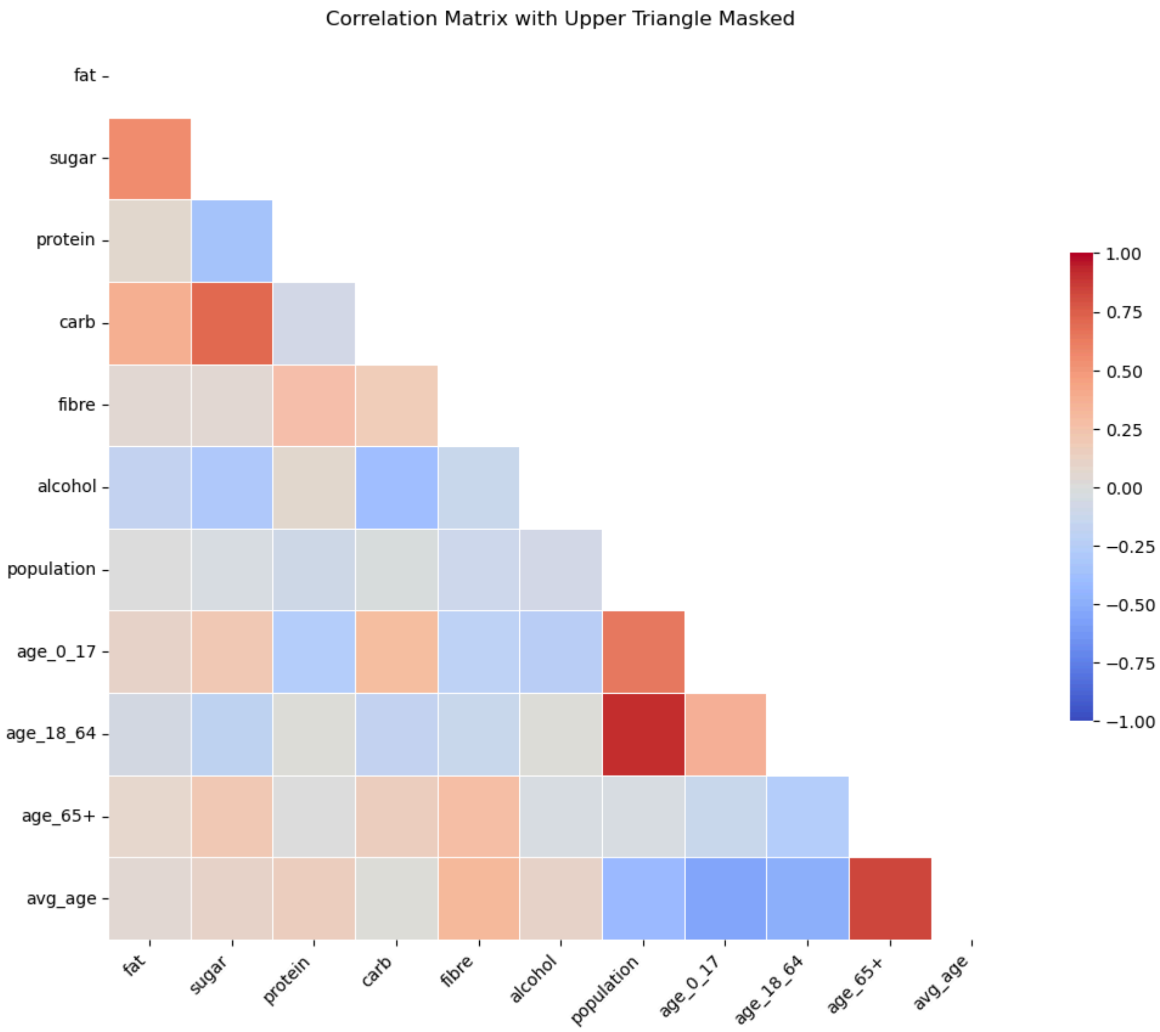
In [29]:
```
correlation_matrix
```

Out[29]:

	fat	sugar	protein	carb	fibre	alcohol	population
fat	1.000000	0.543045	0.065477	0.382284	0.059201	-0.175479	-0.001615
sugar	0.543045	1.000000	-0.360883	0.706033	0.032011	-0.282964	-0.038547
protein	0.065477	-0.360883	1.000000	-0.081434	0.258154	0.069506	-0.090561
carb	0.382284	0.706033	-0.081434	1.000000	0.173675	-0.389251	-0.010124
fibre	0.059201	0.032011	0.258154	0.173675	1.000000	-0.128872	-0.115030
alcohol	-0.175479	-0.282964	0.069506	-0.389251	-0.128872	1.000000	-0.079214
population	-0.001615	-0.038547	-0.090561	-0.010124	-0.115030	-0.079214	1.000000
age_0_17	0.093441	0.213189	-0.273007	0.294313	-0.198189	-0.239219	0.638701
age_18_64	-0.061896	-0.192235	0.007876	-0.175818	-0.124002	0.014979	0.918195
age_65+	0.074703	0.207077	0.006794	0.141835	0.271290	-0.023533	-0.036626
avg_age	0.038290	0.093013	0.156712	0.015236	0.327308	0.106084	-0.409915

In [30]:

```
mask = np.triu(np.ones_like(correlation_matrix, dtype=bool))

# Set up the matplotlib figure
plt.figure(figsize=(12, 10))

# Draw the heatmap with the mask
sns.heatmap(correlation_matrix, mask=mask, cmap='coolwarm', vmax=1, vmin=-1, cen
            square=True, linewidths=.5, cbar_kws={"shrink": .5}, annot=True, fmt

plt.xticks(rotation=45, ha="right")
plt.yticks(rotation=0)
plt.title('Correlation Matrix with Upper Triangle Masked')
plt.show()
```



The visualization gives a simple, straightforward, and visually appealing manner for understanding the relationships between variables in our dataset.

Positive Correlation:

This means two variables tend to move in the same direction means 1 variable value increases the other variable value also increases and vice versa and the shade of that is most likely close to the red shade here in our dataset the sugar and carb have highly positive correlation with fat, which means if people are more like to intake the sugar and crab they will get more fat.

Negative Correlation:

This means two variables tend to move in the opposite direction means 1 variable value increases the other variable value also increases and vice versa and the shade of that most likely in close to blue shade.

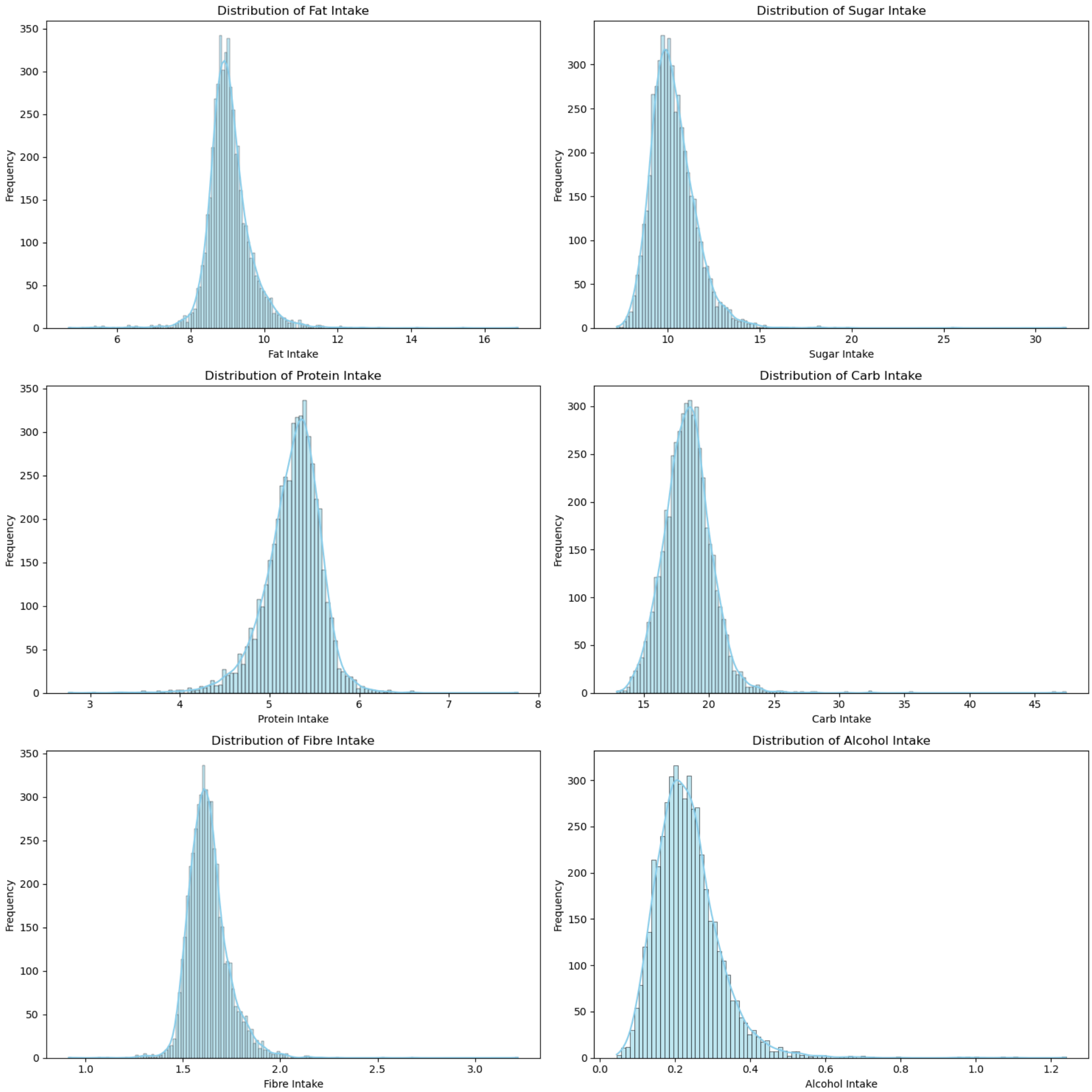
Both Positive(close to read shade) and Negative(close to blue shade):

If you see that sort of relation which means the relationship between two variables is not strictly close to linear, which means they could have more complex patterns and linearly they are not separable you can say that.

In [31]:
```
df_lsoa['age_group'].value_counts()
```

Out[31]:
```
age_group
18-64    4833
0-17        0
65+         0
Name: count, dtype: int64
```

In [32]:
```
fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(15, 15))
fig.tight_layout(pad=5.0)

# Flattening axes array for easy iteration
axes_flat = axes.flatten()

# Iterating over nutrients and plotting distribution for each
for i, nutrient in enumerate(nutrients):
    # Select the current axis
    ax = axes_flat[i]

    # Drop missing values and plot histogram
    sns.histplot(df_lsoa[nutrient].dropna(), kde=True, ax=ax, color='skyblue')

    # Setting the title for each subplot
    ax.set_title(f'Distribution of {nutrient.capitalize()} Intake')
    ax.set_xlabel(f'{nutrient.capitalize()} Intake')
    ax.set_ylabel('Frequency')

# Adjusting layout for better readability
plt.tight_layout()

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



Normal Distribution:

A fully normal distribution is symmetric around its mean, which means the majority of the data falls around it this distribution is also called Gaussian Distribution. Here in our dataset the protein intake and fat intake and the values of these distributions are more tends towards the mean value.

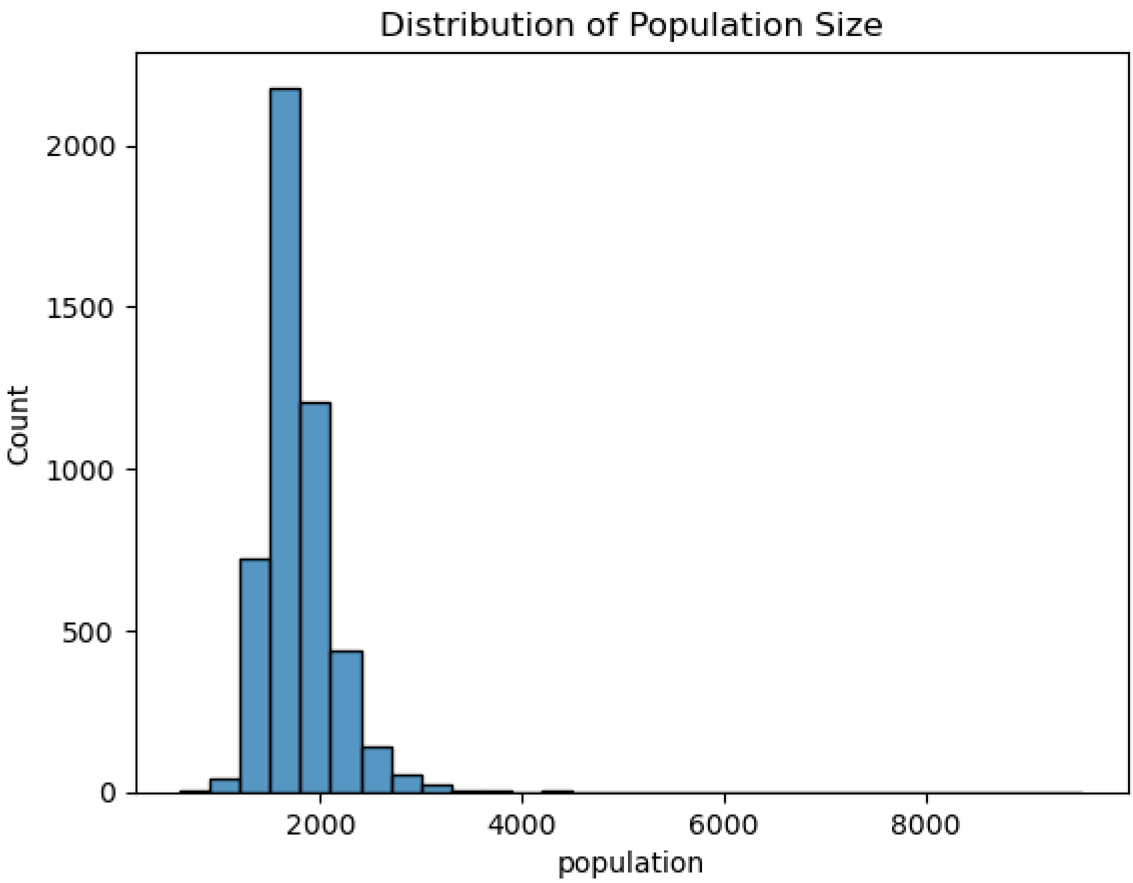
Left-Skewed Distribution (Negatively Skewed):

A left-skewed distribution, also known as a negatively skewed distribution, has a longer or fatter tail on the left side than on the right. It implies that the majority of the values (including the median) are clustered to the right of the distribution.The mean is lower than the median, which is lower than the mode. The majority of the data is focused at the upper end of the scale. Here in above figure there is no distribution is left skewed.

Right-Skewed Distribution (Positively Skewed):

A right-skewed distribution, also known as a positively skewed distribution, occurs when the right side's tail is longer or fatter than the left side. It implies that the majority of the values are clustered to the left of the distribution.The mean exceeds the median, and both exceed the mode. The majority of the data is focused toward the bottom end of the spectrum. so there is many nutrients are positively skewed distribution like 'alcohol', 'carb', 'fibre', and 'sugar'.

In [33]:
```
sns.histplot(data=df_lsoa, x='population', bins=30)
plt.title('Distribution of Population Size')
plt.show()
```



Here, the population distribution is right skewed because this distribution has long tailed to the left side

In [34]:
```
X = df_lsoa[['population', 'area_sq_km', 'people_per_sq_km']]  # Predictor varia
y = df_lsoa['fat']  # Response variable, 'fat' intake

# Splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

# Initializing and training the Random Forest model
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Making predictions on the test set
y_pred = model.predict(X_test)

# Evaluating the model
mse = mean_squared_error(y_test, y_pred)
rmse = mse ** 0.5

print(f"Test RMSE: {rmse}")
```

```
Test RMSE: 0.6374123366751582
```

In [35]:
```
X = df_lsoa.drop(columns=['area_id', 'age_group', 'fat'])  # Predictor variables
y = df_lsoa['fat']  # Response variable, 'fat' intake
# Splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = mse ** 0.5
print(f"Test RMSE: {rmse}")
```

```
Test RMSE: 0.04278214382109068
```

So when, I pass the only 3 features, to train my model to predict the 'fat' my Root Mean Square Error" is 0.6374 and when I add the multiple features, just removing the datatype object attribute I got the lower RMSE value which is 0.0427 which means by increasing the features of he model the model improves too much.

Task 3: Combining Datasets

In [36]:
```
df_income = pd.read_excel("localincomedeprivationdata.xlsx", sheet_name = "LSOA"
df_income.head()
```

Out[36]:

	LSOA code (2011)	LSOA name (2011)	Local Authority District code (2019)	Local Authority District name (2019)	Overall Index of Multiple Deprivation (IMD) Score	Index of Multiple Deprivation (IMD) Rank (where 1 is most deprived)	Index of Multiple Deprivation (IMD) Decile (where 1 is most deprived 10% of LSOAs)	Inc S (ı
0	E01031338	Adur 002A	E07000223	Adur	5.518	30006	10	(
1	E01031339	Adur 002B	E07000223	Adur	6.186	29228	9	(
2	E01031340	Adur 002C	E07000223	Adur	5.213	30309	10	(
3	E01031341	Adur 008A	E07000223	Adur	38.777	4639	2	(
4	E01031342	Adur 008B	E07000223	Adur	16.050	17896	6	(

In [37]:
```
ros, cols = df_income.shape
print("Number of Records : ", ros)
print("Number of Attributes : ", cols)
```

```
Number of Records :  32844
Number of Attributes :  15
```

In [38]:
```
df_income.columns
```

Out[38]:
```
Index(['LSOA code (2011)', 'LSOA name (2011)',
       'Local Authority District code (2019)',
       'Local Authority District name (2019)',
       'Overall Index of Multiple Deprivation (IMD) Score',
       'Index of Multiple Deprivation (IMD) Rank (where 1 is most deprived)',
       'Index of Multiple Deprivation (IMD) Decile (where 1 is most deprived 1
0% of LSOAs)',
       'Income Score (rate)', 'Income Rank (where 1 is most deprived)',
       'Income Decile (where 1 is most deprived 10% of LSOAs)',
       'Total population: mid 2015 (excluding prisoners)',
       'Dependent Children aged 0-15: mid 2015 (excluding prisoners)',
       'Population aged 16-59: mid 2015 (excluding prisoners)',
       'Older population aged 60 and over: mid 2015 (excluding prisoners)',
       'Working age population 18-59/64: for use with Employment Deprivation Do
main (excluding prisoners) '],
      dtype='object')
```

In [39]:
```
df_income.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32844 entries, 0 to 32843
Data columns (total 15 columns):
 #   Column
Non-Null Count  Dtype
---  ------
--------------  -----
 0   LSOA code (2011)
32844 non-null  object
 1   LSOA name (2011)
32844 non-null  object
 2   Local Authority District code (2019)
32844 non-null  object
 3   Local Authority District name (2019)
32844 non-null  object
 4   Overall Index of Multiple Deprivation (IMD) Score
32844 non-null  float64
 5   Index of Multiple Deprivation (IMD) Rank (where 1 is most deprived)
32844 non-null  int64
 6   Index of Multiple Deprivation (IMD) Decile (where 1 is most deprived 10% of
LSOAs)                  32844 non-null  int64
 7   Income Score (rate)
32844 non-null  float64
 8   Income Rank (where 1 is most deprived)
32844 non-null  int64
 9   Income Decile (where 1 is most deprived 10% of LSOAs)
32844 non-null  int64
 10  Total population: mid 2015 (excluding prisoners)
32844 non-null  int64
 11  Dependent Children aged 0-15: mid 2015 (excluding prisoners)
32844 non-null  int64
 12  Population aged 16-59: mid 2015 (excluding prisoners)
32844 non-null  int64
 13  Older population aged 60 and over: mid 2015 (excluding prisoners)
32844 non-null  int64
 14  Working age population 18-59/64: for use with Employment Deprivation Domain
(excluding prisoners)   32844 non-null  int64
dtypes: float64(2), int64(9), object(4)
memory usage: 3.8+ MB
```

Overview of the Income dataset:

This dataset examines socioeconomic and demographic aspects related to income and deprivation within specified geographic areas (LSOAs). The columns include identifiers such as LSOA codes and names, as well as the codes and names of the respective local authority districts. Key indicators include Overall Index of Multiple Deprivation (IMD) scores and rankings, as well as specific income-related measurements that identify places based on deprivation and income disparity.

Detailed Breakdown of Attributes:

Geographic identifiers include LSOA codes and names, as well as equivalent local authority district codes and names between 2011 and 2019. Deprivation Scores and Ranks: Comprehensive measures include the Overall IMD Score, IMD Rank, and IMD Decile, which provide a spectrum of deprivation in comparison to comparable places.

Income-specific metrics: Income Score (rate), Income Rank, and Income Decile are focused metrics that analyze the economic state of certain locations. Demographic Breakdown: Population data is segmented into dependent children (0-15 years), working-age population (16-59/64 years), and older population (60 years and up), as well as overall population estimates, which have been adjusted to remove crimina

Assumptions and Limitations:

Geographic and temporal coverage: Assumes that the dataset fully captures all relevant geographic areas as of 2019, with no notable changes in boundaries or population movements since mid-2015. Prisoners are excluded from population numbers, which may have an impact on the accuracy of demographic evaluations in locations with large prison populations. s.

In [40]:
```
filtered_df = df_income[df_income["LSOA code (2011)"] == "E01000001"]
filtered_df
```

Out[40]:

	LSOA code (2011)	LSOA name (2011)	Local Authority District code (2019)	Local Authority District name (2019)	Overall Index of Multiple Deprivation (IMD) Score	Index of Multiple Deprivation (IMD) Rank (where 1 is most deprived)	Index of Multiple Deprivation (IMD) Decile (where 1 is most deprived 10% of LSOAs)
6831	E01000001	City of London 001A	E09000001	City of London	6.208	29199	9

In [41]:
```
df_income.describe()
```

Out[41]:

	Overall Index of Multiple Deprivation (IMD) Score	Index of Multiple Deprivation (IMD) Rank (where 1 is most deprived)	Index of Multiple Deprivation (IMD) Decile (where 1 is most deprived 10% of LSOAs)	Income Score (rate)	Income Rank (where 1 is most deprived)	Inc De (where m depri 10% LSC
count	32844.000000	32844.000000	32844.000000	32844.000000	32844.000000	32844.000
mean	21.669393	16422.499208	5.500122	0.128166	16422.498478	5.500
std	15.332229	9481.390454	2.872325	0.093539	9481.389874	2.872
min	0.541000	1.000000	1.000000	0.003000	1.000000	1.000
25%	9.913750	8211.750000	3.000000	0.056000	8211.750000	3.000
50%	17.647500	16422.500000	5.500000	0.099000	16422.500000	5.500
75%	29.583000	24633.250000	8.000000	0.178000	24633.250000	8.000
max	92.735000	32844.000000	10.000000	0.609000	32844.000000	10.000

Here is the descriptive statistics for my income dataset, here we can see that std_dev, mean, min, max and percentiles values for each features.

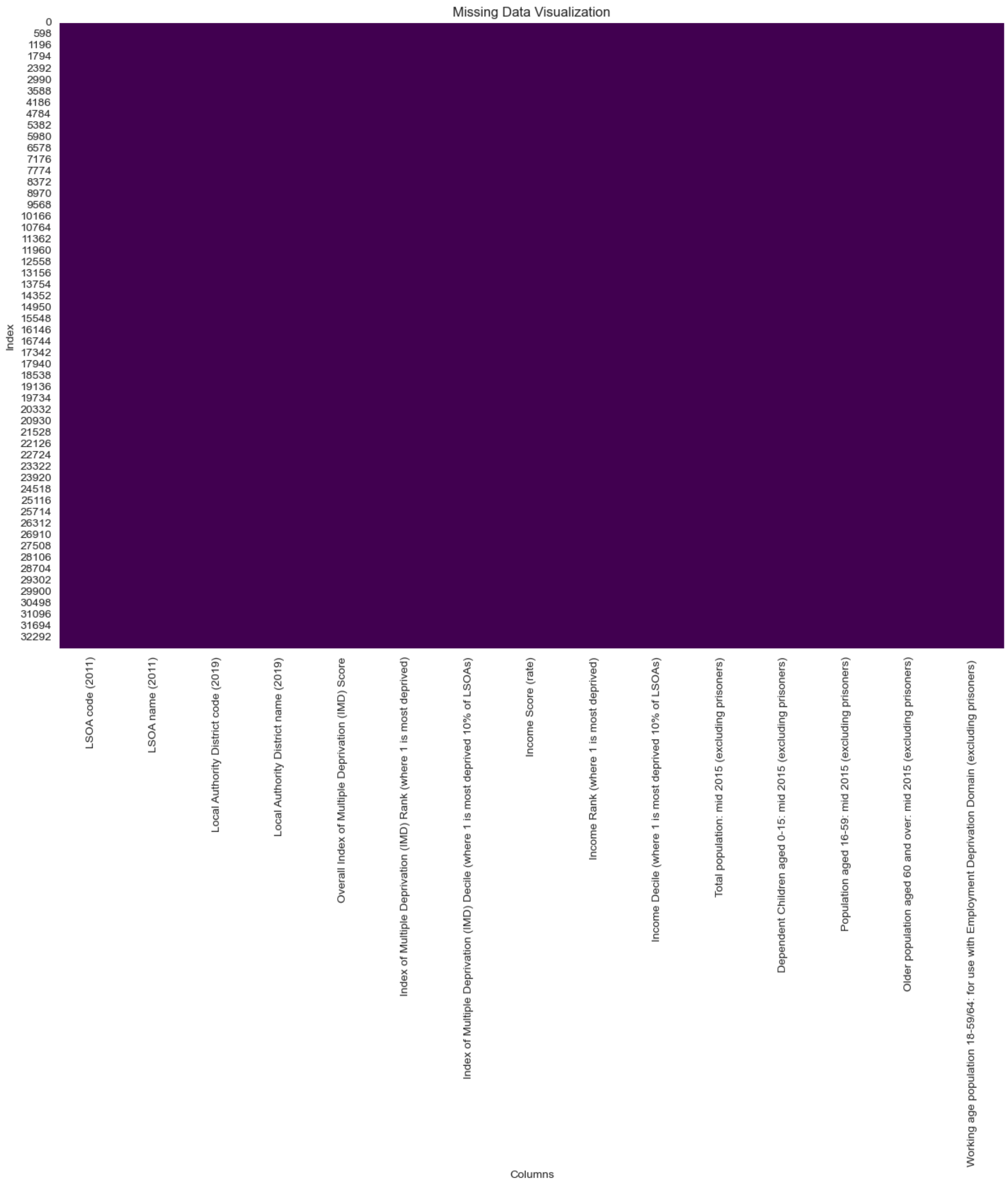
In [42]:

```
df_income.isnull().sum()
```

Out[42]:
```
LSOA code (2011)
0
LSOA name (2011)
0
Local Authority District code (2019)
0
Local Authority District name (2019)
0
Overall Index of Multiple Deprivation (IMD) Score
0
Index of Multiple Deprivation (IMD) Rank (where 1 is most deprived)
0
Index of Multiple Deprivation (IMD) Decile (where 1 is most deprived 10% of LSO
As)                    0
Income Score (rate)
0
Income Rank (where 1 is most deprived)
0
Income Decile (where 1 is most deprived 10% of LSOAs)
0
Total population: mid 2015 (excluding prisoners)
0
Dependent Children aged 0-15: mid 2015 (excluding prisoners)
0
Population aged 16-59: mid 2015 (excluding prisoners)
0
Older population aged 60 and over: mid 2015 (excluding prisoners)
0
Working age population 18-59/64: for use with Employment Deprivation Domain (ex
cluding prisoners)     0
dtype: int64
```

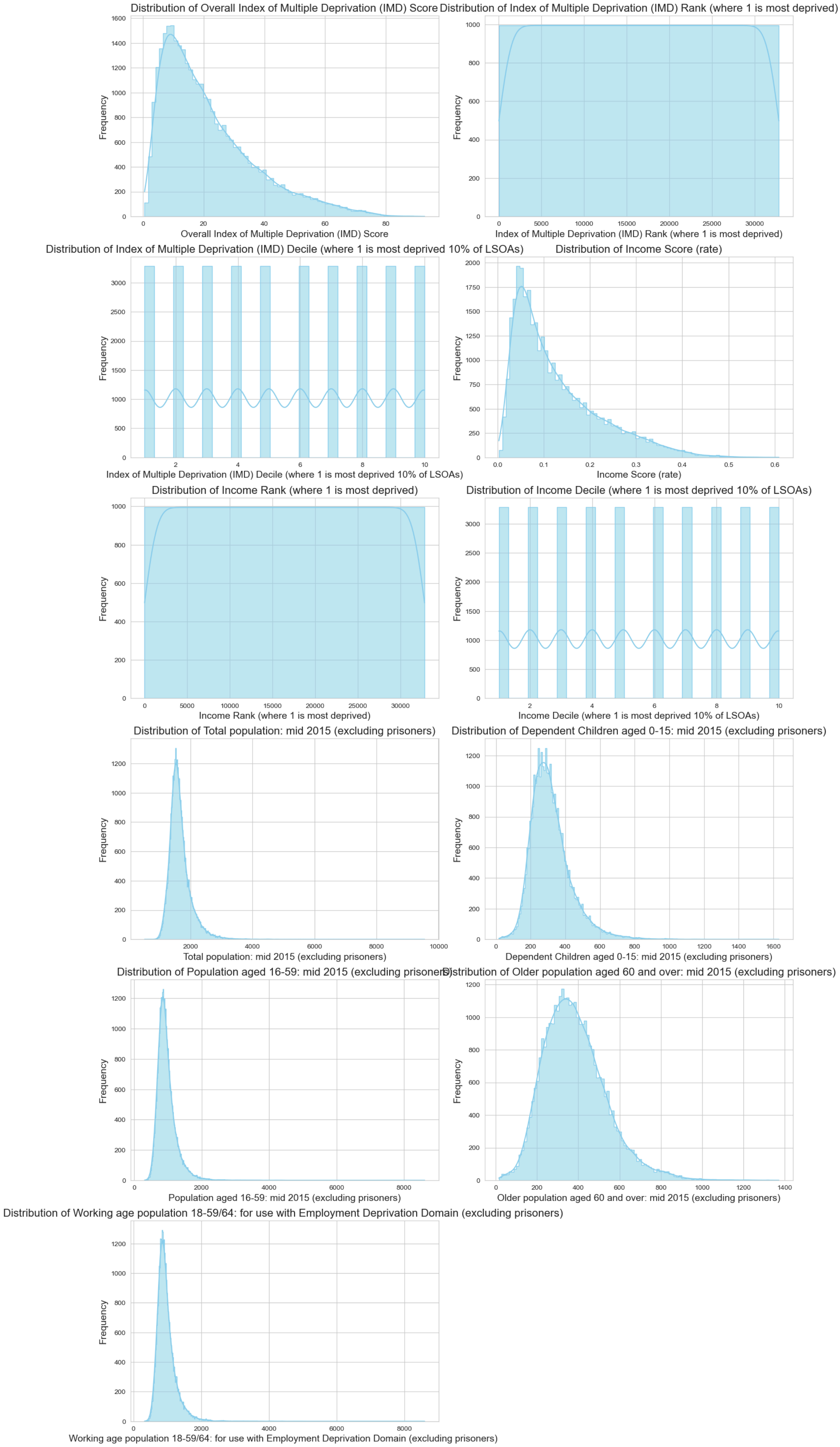
So, in our dataset there is no missing value in our dataset

In [56]:
```
plt.figure(figsize=(15, 10))
sns.heatmap(df_income.isnull(), cbar=False, cmap='viridis')
plt.title('Missing Data Visualization')
plt.xlabel('Columns')
plt.ylabel('Index')
plt.show()
```



In [43]:
```
numeric_columns = df_income.select_dtypes(include=[np.number]).columns.tolist()
categorical_columns = df_income.select_dtypes(exclude=[np.number, 'datetime']).c

# Set the aesthetic style of the plots
sns.set_style("whitegrid")

# Initialize variables for subplot layout
n_cols = 2
n_rows = max(len(numeric_columns), len(categorical_columns)) // n_cols + 1

# Plotting distributions for numeric columns
fig, axes = plt.subplots(n_rows, n_cols, figsize=(15, 5 * n_rows))
fig.tight_layout(pad=4.0)

# Flatten the axes array for easy iteration
axes_flat = axes.flatten()

# Function to remove unused subplots
```

```
def remove_unused_axes(axes_flat, start_index):
    for i in range(start_index, len(axes_flat)):
        fig.delaxes(axes_flat[i])

# Counter for the current plot
plot_counter = 0

# Plot numeric columns
for col in numeric_columns:
    sns.histplot(df_income[col], kde=True, color='skyblue', element='step', binw
    axes_flat[plot_counter].set_title(f'Distribution of {col}', fontsize=16)
    axes_flat[plot_counter].set_xlabel(col, fontsize=14)
    axes_flat[plot_counter].set_ylabel('Frequency', fontsize=14)
    plot_counter += 1
    if plot_counter >= len(axes_flat):  # Check if we've used all available subp
        remove_unused_axes(axes_flat, plot_counter)
        plt.show()
        fig, axes = plt.subplots(n_rows, n_cols, figsize=(15, 5 * n_rows))  # St
        fig.tight_layout(pad=4.0)
        axes_flat = axes.flatten()
        plot_counter = 0  # Reset counter

# Ensure unused subplots are removed and display the last figure for numeric col
if plot_counter > 0:
    remove_unused_axes(axes_flat, plot_counter)
    plt.show()
```

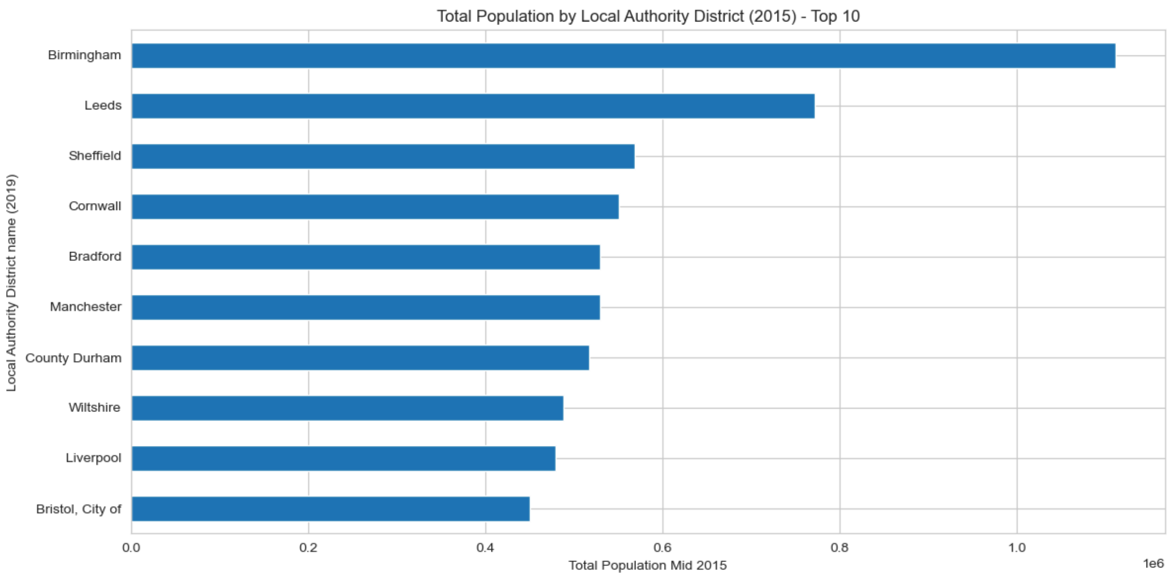

Distribution of Overall Index of Multiple Deprivation (IMD) Score
Overall Index of Multiple Deprivation (IMD) Score
Frequency
Distribution of Index of Multiple Deprivation (IMD) Rank (where 1 is most deprived)
Index of Multiple Deprivation (IMD) Rank (where 1 is most deprived)
Frequency
Distribution of Index of Multiple Deprivation (IMD) Decile (where 1 is most deprived 10% of LSOAs)
Index of Multiple Deprivation (IMD) Decile (where 1 is most deprived 10% of LSOAs)
Frequency
Distribution of Income Score (rate)
Income Score (rate)
Frequency
Distribution of Income Rank (where 1 is most deprived)
Income Rank (where 1 is most deprived)
Frequency
Distribution of Income Decile (where 1 is most deprived 10% of LSOAs)
Income Decile (where 1 is most deprived 10% of LSOAs)
Frequency
Distribution of Total population: mid 2015 (excluding prisoners)
Total population: mid 2015 (excluding prisoners)
Frequency
Distribution of Dependent Children aged 0-15: mid 2015 (excluding prisoners)
Dependent Children aged 0-15: mid 2015 (excluding prisoners)
Frequency
Distribution of Population aged 16-59: mid 2015 (excluding prisoners)
Population aged 16-59: mid 2015 (excluding prisoners)
Frequency
Distribution of Older population aged 60 and over: mid 2015 (excluding prisoners)
Older population aged 60 and over: mid 2015 (excluding prisoners)
Frequency
Distribution of Working age population 18-59/64: for use with Employment Deprivation Domain (excluding prisoners)
Working age population 18-59/64: for use with Employment Deprivation Domain (excluding prisoners)
Frequency

Here we can clearly sees that some of the distributions are not seems good because they have the some sort of categorical data like, from (1-10) scale, and some of them have the Right Skewed distrbutions which means, there tails are more tends towards to the right side of the distributions

In [44]:
```
# Check unique geographical identifiers
unique_areas = df_income['LSOA code (2011)'].nunique()
print(f"Number of unique geographical areas: {unique_areas}")

# Check for demographic group details
demographic_details = ['Total population: mid 2015 (excluding prisoners)', 'Depe
                       'Population aged 16-59: mid 2015 (excluding prisoners)',
for detail in demographic_details:
    print(f"{detail}: Range {df_income[detail].min()} to {df_income[detail].max(
```

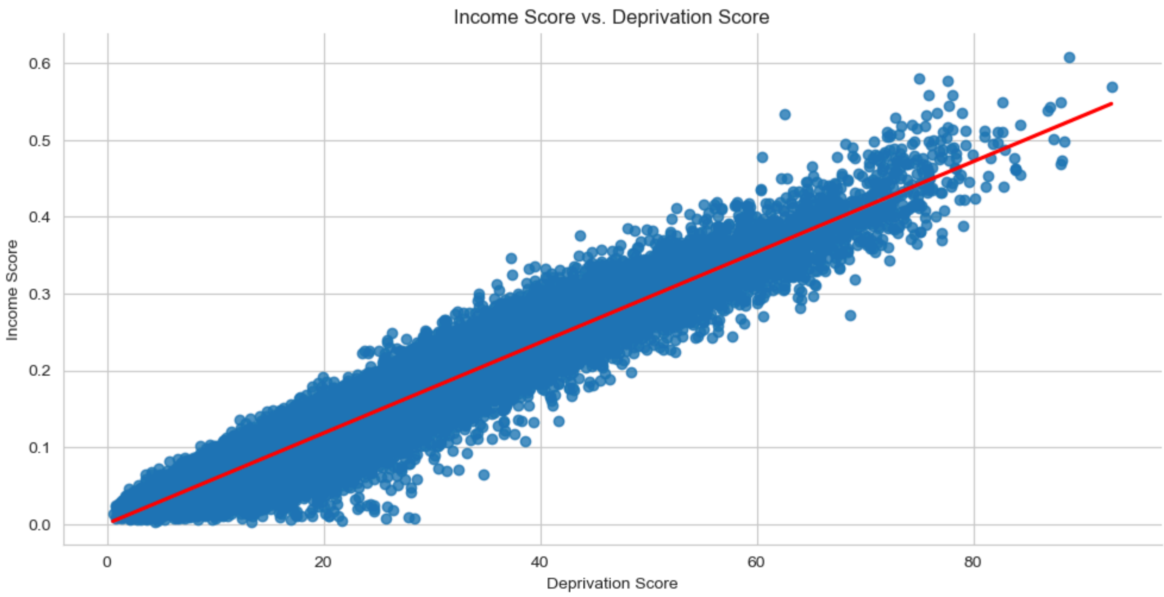
```
Number of unique geographical areas: 32844
Total population: mid 2015 (excluding prisoners): Range 523 to 9551
Dependent Children aged 0-15: mid 2015 (excluding prisoners): Range 17 to 1632
Population aged 16-59: mid 2015 (excluding prisoners): Range 310 to 8608
Older population aged 60 and over: mid 2015 (excluding prisoners): Range 15 to 13
72
```

In [45]:
```
plt.figure(figsize=(12, 6))  # Adjust the size as needed

# Assuming 'df_lsoa' is your DataFrame and focusing on the top 20 Local Authorit
top_districts = df_income.groupby("Local Authority District name (2019)")["Total

# If the dataset is large, you might consider limiting the output to the top N c
# Here's an example of limiting the plot to the top 20 districts
top_districts.tail(10).plot.barh()

plt.xlabel("Total Population Mid 2015")
plt.title("Total Population by Local Authority District (2015) - Top 10")
plt.tight_layout()  # Adjust layout to make room for the label
plt.show()
```



This visualization, notably a horizontal bar chart of the top 10 local authority districts ranked by total population in 2015, allows for a clear and succinct comparison of population sizes between districts, Key demographic data, providing a snapshot of population distribution across districts, can be used to drive future analysis, policymaking, and urban planning activities. It simplifies complex data into an understandable manner, showing substantial disparities in population sizes that may correlate with numerous socioeconomic indices so here in our case Birmingham is on the top.

In [46]:
```
sns.lmplot(x='Overall Index of Multiple Deprivation (IMD) Score', y='Income Scor
plt.title('Income Score vs. Deprivation Score')
plt.xlabel('Deprivation Score')
plt.ylabel('Income Score')
plt.show()
```



Combine the datasets :

In [47]:
```
# Rename the column in df_lsoa from 'area_id' to 'LSOA code (2011)'
df_lsoa.rename(columns={'area_id': 'LSOA code (2011)'}, inplace=True)

# Now merge the datasets
merged_data = pd.merge(df_lsoa, df_income, on='LSOA code (2011)', how='inner')

# Check the merged result
print("Merged Data Shape:", merged_data.shape)
print("Merged Data Columns:", merged_data.columns)
```

```
Merged Data Shape: (4833, 217)
Merged Data Columns: Index(['LSOA code (2011)', 'weight', 'weight_perc2.5', 'weig
ht_perc25',
       'weight_perc50', 'weight_perc75', 'weight_perc97.5', 'weight_std',
       'weight_ci95', 'volume',
       ...
       'Index of Multiple Deprivation (IMD) Rank (where 1 is most deprived)',
       'Index of Multiple Deprivation (IMD) Decile (where 1 is most deprived 10%
of LSOAs)',
       'Income Score (rate)', 'Income Rank (where 1 is most deprived)',
       'Income Decile (where 1 is most deprived 10% of LSOAs)',
       'Total population: mid 2015 (excluding prisoners)',
       'Dependent Children aged 0-15: mid 2015 (excluding prisoners)',
       'Population aged 16-59: mid 2015 (excluding prisoners)',
       'Older population aged 60 and over: mid 2015 (excluding prisoners)',
       'Working age population 18-59/64: for use with Employment Deprivation Doma
in (excluding prisoners) '],
      dtype='object', length=217)
```

Reviewed Literature Summary:

- Socioeconomic Impact on Consumer Choices: Recent research has focused on how income differences affect food purchasing habits, specifically the affordability and accessibility of nutritious food in various income categories.
- Economic and Nutritional Analysis: Research from journals such as Food Policy and The Economic Journal sheds light on the relationship between income levels and dietary choices, examining topics such as food deserts and economic limits on healthy eating.
- Cultural and Regional Buying Trends: Research highlights the impact of cultural and regional influences on purchasing patterns, pointing out considerable variances in food preferences and spending habits among demographics.

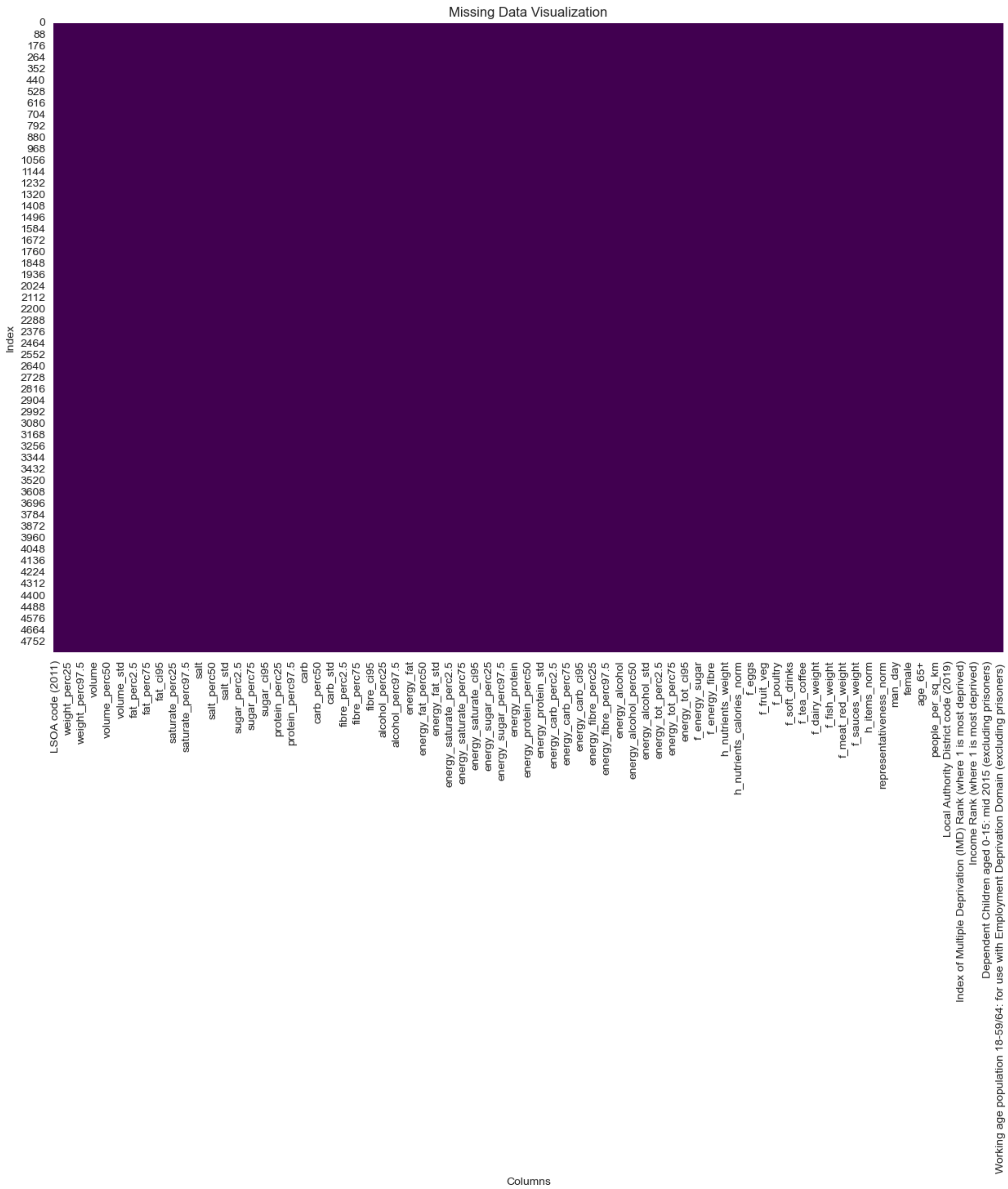
These studies are critical for understanding the complicated interplay between income and consumer purchasing decisions, since they provide a framework for examining trends in the merged dataset. The literature not only validates the observed trends, but it also aids in finding anomalies where predicted patterns do not appear.

In [48]:
```
print(merged_data.isnull().sum())
```

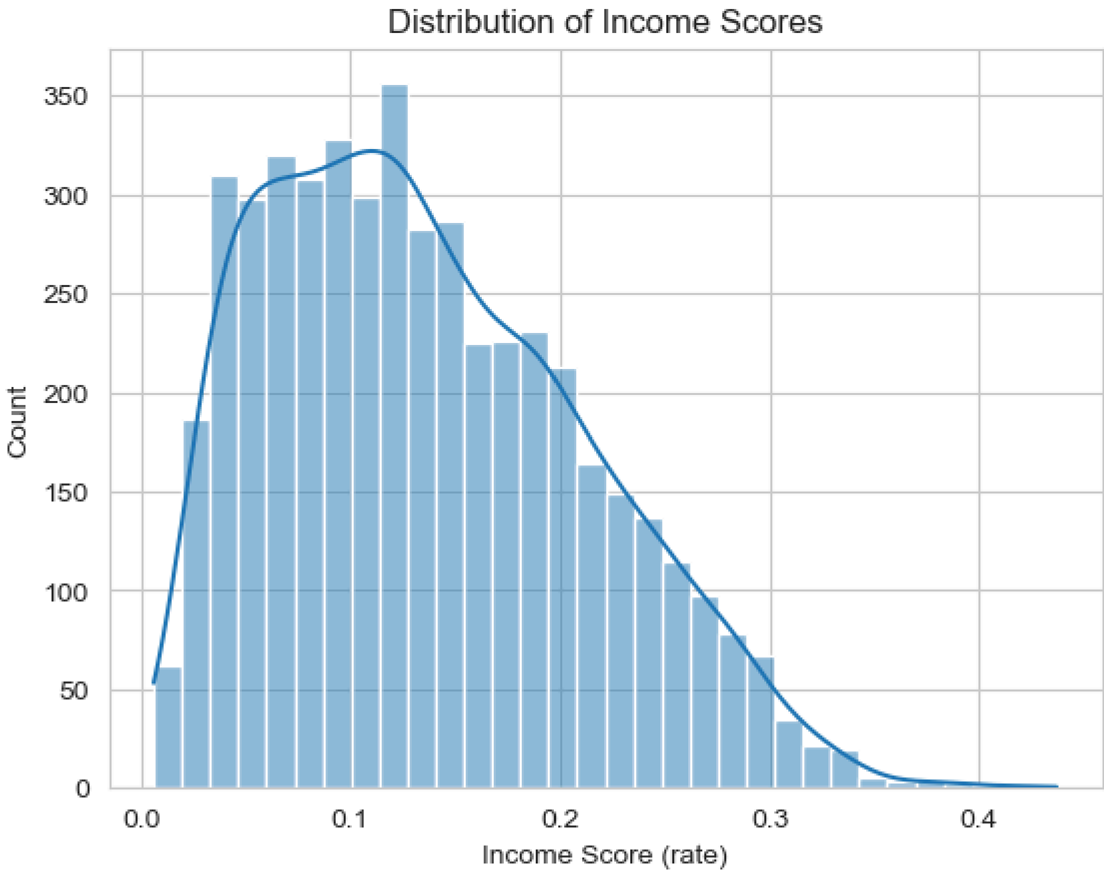
```
LSOA code (2011)
0
weight
0
weight_perc2.5
0
weight_perc25
0
weight_perc50
0

..
Total population: mid 2015 (excluding prisoners)
0
Dependent Children aged 0-15: mid 2015 (excluding prisoners)
0
Population aged 16-59: mid 2015 (excluding prisoners)
0
Older population aged 60 and over: mid 2015 (excluding prisoners)
0
Working age population 18-59/64: for use with Employment Deprivation Domain (excl
uding prisoners)     0
Length: 217, dtype: int64
```

In [49]:

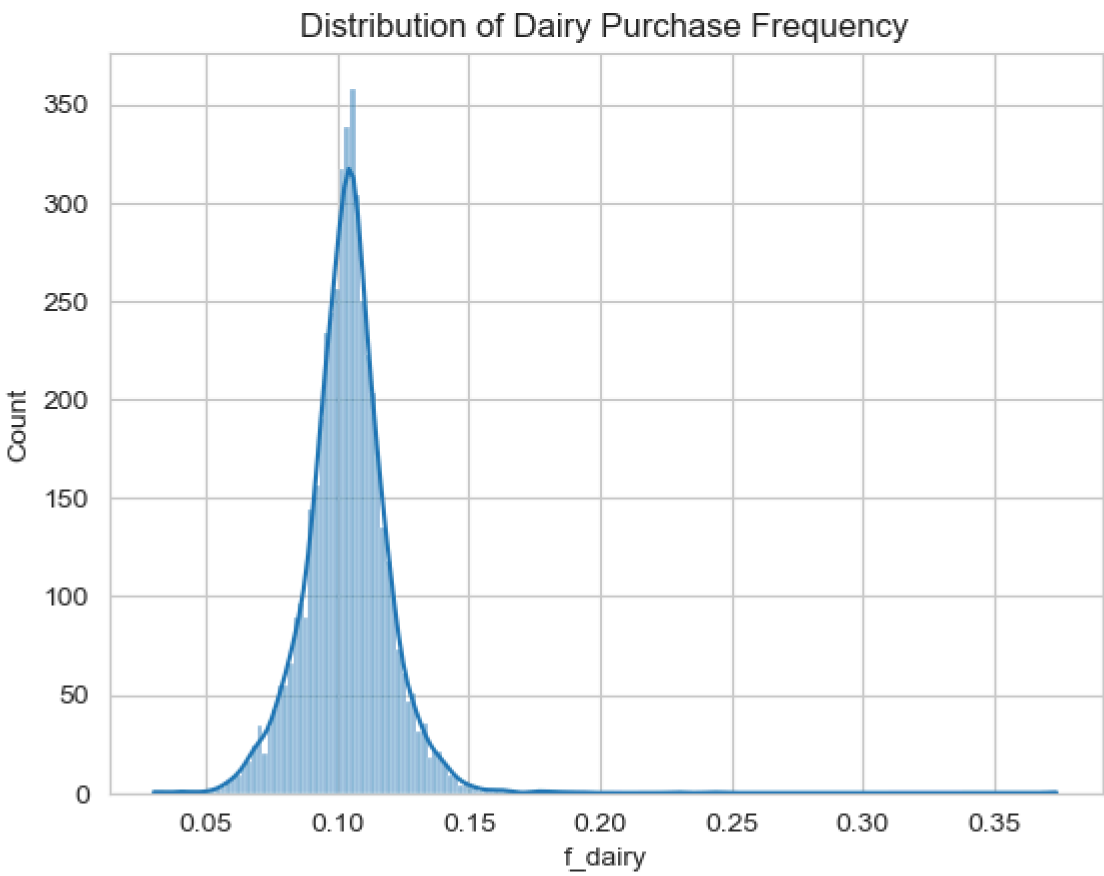
```
plt.figure(figsize=(15, 10))
sns.heatmap(merged_data.isnull(), cbar=False, cmap='viridis')
plt.title('Missing Data Visualization')
plt.xlabel('Columns')
plt.ylabel('Index')
plt.show()
```



In [50]:
```
sns.histplot(data=merged_data, x='Income Score (rate)', kde=True)
plt.title('Distribution of Income Scores')
plt.show()
```



In [51]:
```
sns.histplot(data=merged_data, x='f_dairy', kde=True)
plt.title('Distribution of Dairy Purchase Frequency')
plt.show()
```



In [52]:
```
column_lists = merged_data.columns.tolist()
i = 0
for col in column_lists:
    i+=1
    print(f"Column # {i} Name : {col}")
```

```
Column # 1 Name : LSOA code (2011)
Column # 2 Name : weight
Column # 3 Name : weight_perc2.5
Column # 4 Name : weight_perc25
Column # 5 Name : weight_perc50
Column # 6 Name : weight_perc75
Column # 7 Name : weight_perc97.5
Column # 8 Name : weight_std
Column # 9 Name : weight_ci95
Column # 10 Name : volume
Column # 11 Name : volume_perc2.5
Column # 12 Name : volume_perc25
Column # 13 Name : volume_perc50
Column # 14 Name : volume_perc75
Column # 15 Name : volume_perc97.5
Column # 16 Name : volume_std
Column # 17 Name : volume_ci95
Column # 18 Name : fat
Column # 19 Name : fat_perc2.5
Column # 20 Name : fat_perc25
Column # 21 Name : fat_perc50
Column # 22 Name : fat_perc75
Column # 23 Name : fat_perc97.5
Column # 24 Name : fat_std
Column # 25 Name : fat_ci95
Column # 26 Name : saturate
Column # 27 Name : saturate_perc2.5
Column # 28 Name : saturate_perc25
Column # 29 Name : saturate_perc50
Column # 30 Name : saturate_perc75
Column # 31 Name : saturate_perc97.5
Column # 32 Name : saturate_std
Column # 33 Name : saturate_ci95
Column # 34 Name : salt
Column # 35 Name : salt_perc2.5
Column # 36 Name : salt_perc25
Column # 37 Name : salt_perc50
Column # 38 Name : salt_perc75
Column # 39 Name : salt_perc97.5
Column # 40 Name : salt_std
Column # 41 Name : salt_ci95
Column # 42 Name : sugar
Column # 43 Name : sugar_perc2.5
Column # 44 Name : sugar_perc25
Column # 45 Name : sugar_perc50
Column # 46 Name : sugar_perc75
Column # 47 Name : sugar_perc97.5
Column # 48 Name : sugar_std
Column # 49 Name : sugar_ci95
Column # 50 Name : protein
Column # 51 Name : protein_perc2.5
Column # 52 Name : protein_perc25
Column # 53 Name : protein_perc50
Column # 54 Name : protein_perc75
Column # 55 Name : protein_perc97.5
Column # 56 Name : protein_std
Column # 57 Name : protein_ci95
Column # 58 Name : carb
Column # 59 Name : carb_perc2.5
Column # 60 Name : carb_perc25
```

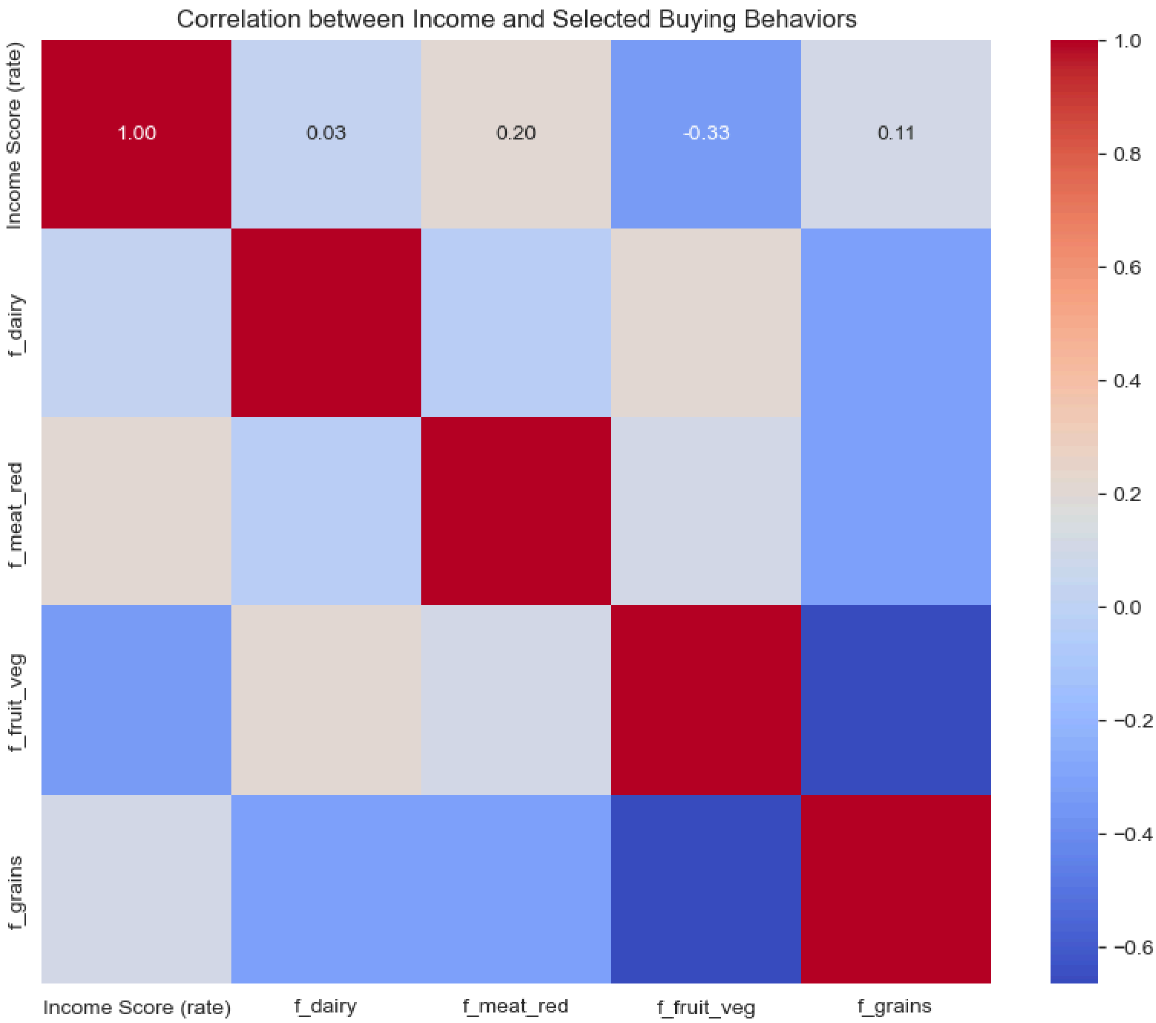
```
Column # 61 Name : carb_perc50
Column # 62 Name : carb_perc75
Column # 63 Name : carb_perc97.5
Column # 64 Name : carb_std
Column # 65 Name : carb_ci95
Column # 66 Name : fibre
Column # 67 Name : fibre_perc2.5
Column # 68 Name : fibre_perc25
Column # 69 Name : fibre_perc50
Column # 70 Name : fibre_perc75
Column # 71 Name : fibre_perc97.5
Column # 72 Name : fibre_std
Column # 73 Name : fibre_ci95
Column # 74 Name : alcohol
Column # 75 Name : alcohol_perc2.5
Column # 76 Name : alcohol_perc25
Column # 77 Name : alcohol_perc50
Column # 78 Name : alcohol_perc75
Column # 79 Name : alcohol_perc97.5
Column # 80 Name : alcohol_std
Column # 81 Name : alcohol_ci95
Column # 82 Name : energy_fat
Column # 83 Name : energy_fat_perc2.5
Column # 84 Name : energy_fat_perc25
Column # 85 Name : energy_fat_perc50
Column # 86 Name : energy_fat_perc75
Column # 87 Name : energy_fat_perc97.5
Column # 88 Name : energy_fat_std
Column # 89 Name : energy_fat_ci95
Column # 90 Name : energy_saturate
Column # 91 Name : energy_saturate_perc2.5
Column # 92 Name : energy_saturate_perc25
Column # 93 Name : energy_saturate_perc50
Column # 94 Name : energy_saturate_perc75
Column # 95 Name : energy_saturate_perc97.5
Column # 96 Name : energy_saturate_std
Column # 97 Name : energy_saturate_ci95
Column # 98 Name : energy_sugar
Column # 99 Name : energy_sugar_perc2.5
Column # 100 Name : energy_sugar_perc25
Column # 101 Name : energy_sugar_perc50
Column # 102 Name : energy_sugar_perc75
Column # 103 Name : energy_sugar_perc97.5
Column # 104 Name : energy_sugar_std
Column # 105 Name : energy_sugar_ci95
Column # 106 Name : energy_protein
Column # 107 Name : energy_protein_perc2.5
Column # 108 Name : energy_protein_perc25
Column # 109 Name : energy_protein_perc50
Column # 110 Name : energy_protein_perc75
Column # 111 Name : energy_protein_perc97.5
Column # 112 Name : energy_protein_std
Column # 113 Name : energy_protein_ci95
Column # 114 Name : energy_carb
Column # 115 Name : energy_carb_perc2.5
Column # 116 Name : energy_carb_perc25
Column # 117 Name : energy_carb_perc50
Column # 118 Name : energy_carb_perc75
Column # 119 Name : energy_carb_perc97.5
Column # 120 Name : energy_carb_std
```

```
Column # 121 Name : energy_carb_ci95
Column # 122 Name : energy_fibre
Column # 123 Name : energy_fibre_perc2.5
Column # 124 Name : energy_fibre_perc25
Column # 125 Name : energy_fibre_perc50
Column # 126 Name : energy_fibre_perc75
Column # 127 Name : energy_fibre_perc97.5
Column # 128 Name : energy_fibre_std
Column # 129 Name : energy_fibre_ci95
Column # 130 Name : energy_alcohol
Column # 131 Name : energy_alcohol_perc2.5
Column # 132 Name : energy_alcohol_perc25
Column # 133 Name : energy_alcohol_perc50
Column # 134 Name : energy_alcohol_perc75
Column # 135 Name : energy_alcohol_perc97.5
Column # 136 Name : energy_alcohol_std
Column # 137 Name : energy_alcohol_ci95
Column # 138 Name : energy_tot
Column # 139 Name : energy_tot_perc2.5
Column # 140 Name : energy_tot_perc25
Column # 141 Name : energy_tot_perc50
Column # 142 Name : energy_tot_perc75
Column # 143 Name : energy_tot_perc97.5
Column # 144 Name : energy_tot_std
Column # 145 Name : energy_tot_ci95
Column # 146 Name : f_energy_fat
Column # 147 Name : f_energy_saturate
Column # 148 Name : f_energy_sugar
Column # 149 Name : f_energy_protein
Column # 150 Name : f_energy_carb
Column # 151 Name : f_energy_fibre
Column # 152 Name : f_energy_alcohol
Column # 153 Name : energy_density
Column # 154 Name : h_nutrients_weight
Column # 155 Name : h_nutrients_weight_norm
Column # 156 Name : h_nutrients_calories
Column # 157 Name : h_nutrients_calories_norm
Column # 158 Name : f_beer
Column # 159 Name : f_dairy
Column # 160 Name : f_eggs
Column # 161 Name : f_fats_oils
Column # 162 Name : f_fish
Column # 163 Name : f_fruit_veg
Column # 164 Name : f_grains
Column # 165 Name : f_meat_red
Column # 166 Name : f_poultry
Column # 167 Name : f_readymade
Column # 168 Name : f_sauces
Column # 169 Name : f_soft_drinks
Column # 170 Name : f_spirits
Column # 171 Name : f_sweets
Column # 172 Name : f_tea_coffee
Column # 173 Name : f_water
Column # 174 Name : f_wine
Column # 175 Name : f_dairy_weight
Column # 176 Name : f_eggs_weight
Column # 177 Name : f_fats_oils_weight
Column # 178 Name : f_fish_weight
Column # 179 Name : f_fruit_veg_weight
Column # 180 Name : f_grains_weight
```

```
Column # 181 Name : f_meat_red_weight
Column # 182 Name : f_poultry_weight
Column # 183 Name : f_readymade_weight
Column # 184 Name : f_sauces_weight
Column # 185 Name : f_sweets_weight
Column # 186 Name : h_items
Column # 187 Name : h_items_norm
Column # 188 Name : h_items_weight
Column # 189 Name : h_items_weight_norm
Column # 190 Name : representativeness_norm
Column # 191 Name : transaction_days
Column # 192 Name : num_transactions
Column # 193 Name : man_day
Column # 194 Name : population
Column # 195 Name : male
Column # 196 Name : female
Column # 197 Name : age_0_17
Column # 198 Name : age_18_64
Column # 199 Name : age_65+
Column # 200 Name : avg_age
Column # 201 Name : area_sq_km
Column # 202 Name : people_per_sq_km
Column # 203 Name : age_group
Column # 204 Name : LSOA name (2011)
Column # 205 Name : Local Authority District code (2019)
Column # 206 Name : Local Authority District name (2019)
Column # 207 Name : Overall Index of Multiple Deprivation (IMD) Score
Column # 208 Name : Index of Multiple Deprivation (IMD) Rank (where 1 is most dep
rived)
Column # 209 Name : Index of Multiple Deprivation (IMD) Decile (where 1 is most d
eprived 10% of LSOAs)
Column # 210 Name : Income Score (rate)
Column # 211 Name : Income Rank (where 1 is most deprived)
Column # 212 Name : Income Decile (where 1 is most deprived 10% of LSOAs)
Column # 213 Name : Total population: mid 2015 (excluding prisoners)
Column # 214 Name : Dependent Children aged 0-15: mid 2015 (excluding prisoners)
Column # 215 Name : Population aged 16-59: mid 2015 (excluding prisoners)
Column # 216 Name : Older population aged 60 and over: mid 2015 (excluding prison
ers)
Column # 217 Name : Working age population 18-59/64: for use with Employment Depr
ivation Domain (excluding prisoners)
```

In [53]:

```python
selected_columns = ['Income Score (rate)', 'f_dairy', 'f_meat_red', 'f_fruit_veg
correlation_matrix = merged_data[selected_columns].corr()

# Plotting a heatmap of the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation between Income and Selected Buying Behaviors')
plt.show()
```



In [54]:
```
print(correlation_matrix)
```

```
                     Income Score (rate)   f_dairy  f_meat_red  f_fruit_veg  \
Income Score (rate)             1.000000  0.032263    0.195663    -0.328682
f_dairy                         0.032263  1.000000   -0.035505     0.203206
f_meat_red                      0.195663 -0.035505    1.000000     0.097302
f_fruit_veg                    -0.328682  0.203206    0.097302     1.000000
f_grains                        0.111116 -0.303350   -0.318323    -0.663758

                     f_grains
Income Score (rate)  0.111116
f_dairy             -0.303350
f_meat_red          -0.318323
f_fruit_veg         -0.663758
f_grains             1.000000
```

Weak Positive Correlations:

Dairy Purchases (f_dairy): A very modest positive correlation with income suggests that more income has little effect on increasing dairy purchases. Meat Purchases (f_meat_red): A weak positive association implies that people with greater incomes buy slightly more meat items, presumably because meat is more expensive than other foods. Grain purchase (f_grains): A minor positive association shows that increased income marginally boosts grain purchases; nevertheless, the relationship is not robust.

Moderate Negative Correlation:

Fruit and Vegetable Purchases (f_fruit_veg): Because there is a moderate negative connection with income, purchases of fruits and vegetables fall as income rises. This unanticipated tendency may represent differences in nutritional habits, shopping destinations, or lifestyle changes linked with greater income levels.

Insights and Implications

The relationships could be due to income-driven economic decisions, implying that eating preferences and buying patterns differ among income groups. Negative associations, particularly with fruits and vegetables, may reflect substitution effects, in which higher-income consumers select alternative sorts of foods or buy at different outlets.

In [57]:
```
merged_data.head()
```

Out[57]:

	LSOA code (2011)	weight	weight_perc2.5	weight_perc25	weight_perc50	weight_perc75
0	E01000001	308.119047	35.0	150.0	250.0	400.0
1	E01000002	313.517874	40.0	150.0	250.0	400.0
2	E01000003	315.084751	35.0	150.0	250.0	400.0
3	E01000005	356.033437	38.0	150.0	280.0	450.0
4	E01000006	451.262063	36.0	180.0	325.0	500.0

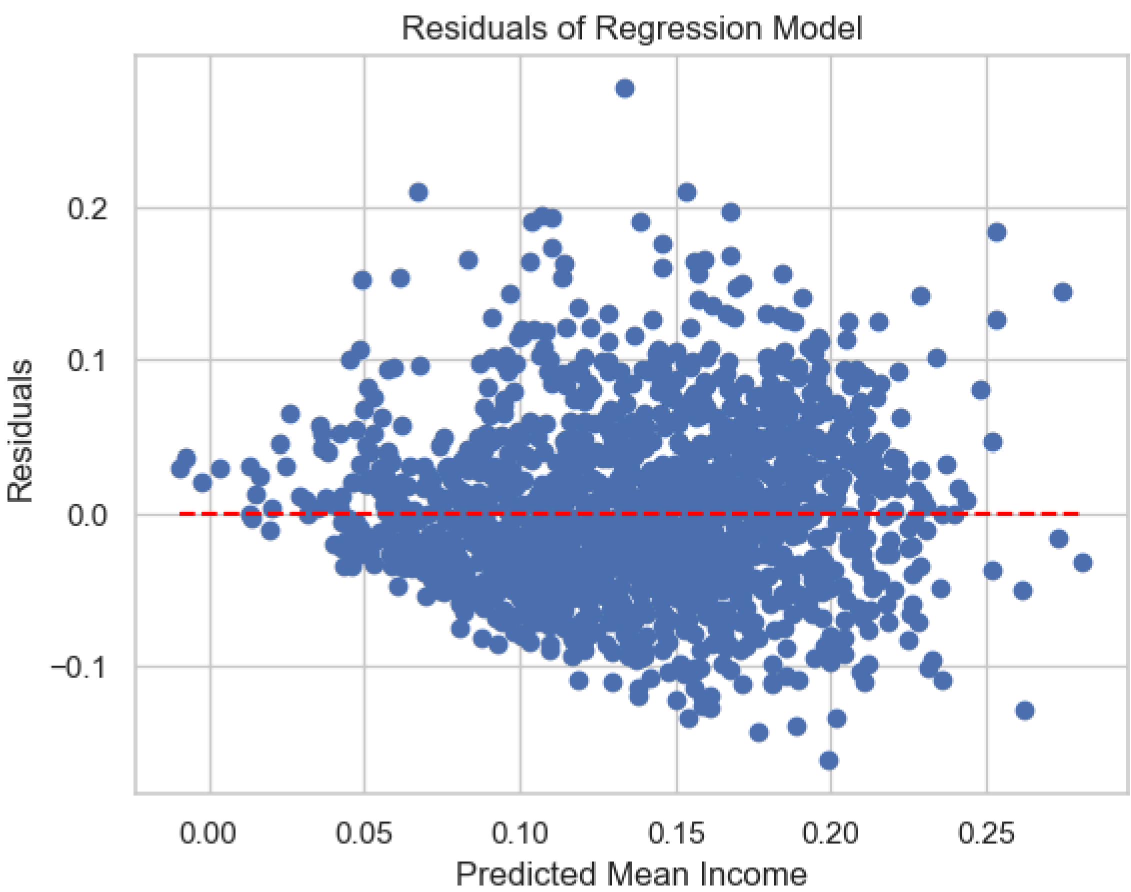
In [61]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=merged_data, x='Income Score (rate)', y='f_dairy', color='b
plt.title('Income vs. Dairy Product Purchases')
plt.xlabel('Income Score (rate)')
plt.ylabel('Dairy Purchases')
plt.grid(True)
plt.show()
```

In [65]:
```
sns.set_theme(style="whitegrid")

# Creating a scatter plot with a regression line
plt.figure(figsize=(10, 6))
scatter = sns.scatterplot(x='Income Score (rate)', y='f_dairy', data=merged_data
line = sns.regplot(x='Income Score (rate)', y='f_dairy', data=merged_data, scatt

plt.title('Income vs. Dairy Product Purchases with Regression Line', fontsize=16
plt.xlabel('Income Score (rate)', fontsize=14)
plt.ylabel('Dairy Purchases', fontsize=14)
plt.legend(labels=['Regression Line', 'Purchases'], frameon=False)
sns.despine() # Removes the top and right spines
plt.show()
```

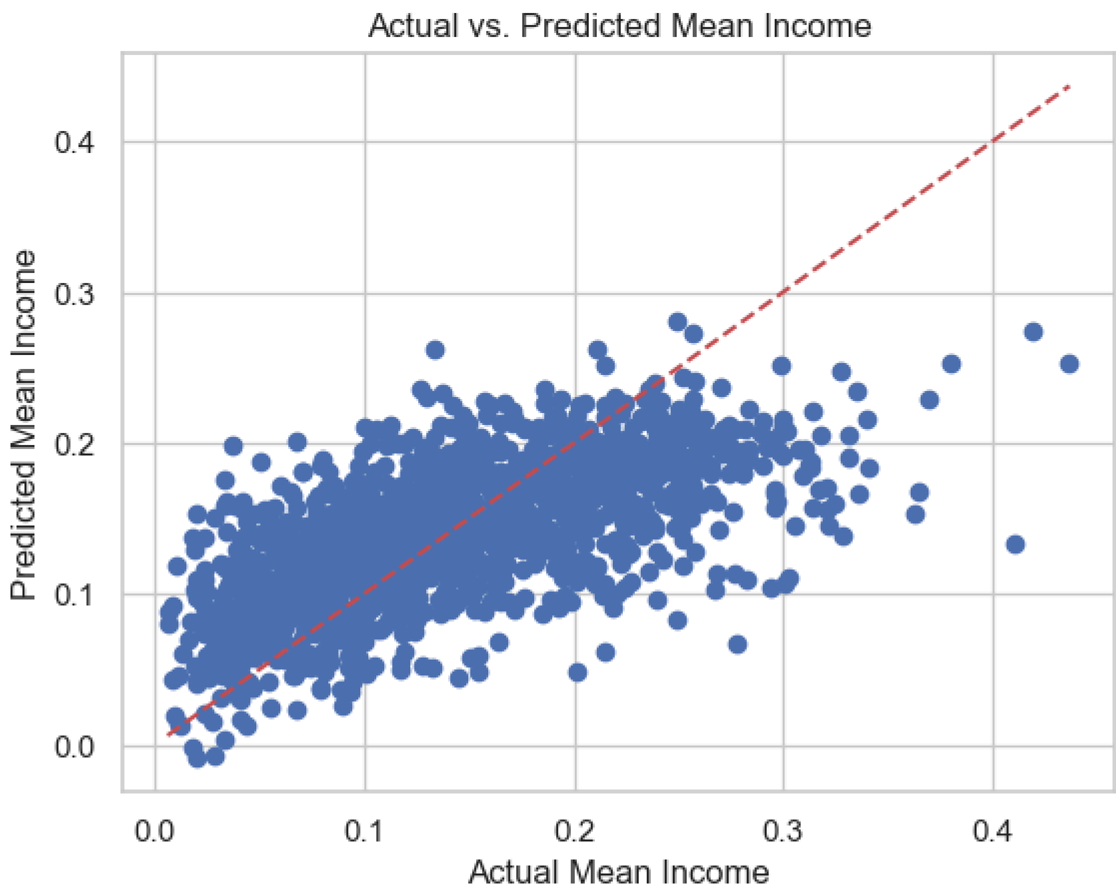
In [67]:
```
predictors = ['f_beer', 'f_dairy', 'f_fats_oils', 'f_fish', 'f_fruit_veg',
              'f_grains', 'f_meat_red', 'f_soft_drinks', 'f_spirits', 'f_sweets'
              'f_tea_coffee', 'f_water', 'f_wine']

X = merged_data[predictors]
y = merged_data['Income Score (rate)']

# Splitting the data into training and testing sets (70% train, 30% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_

# Creating and training the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Predicting on the test set
y_pred = model.predict(X_test)

# Calculating R-squared and residuals
r2 = r2_score(y_test, y_pred)
residuals = y_test - y_pred

# Output the R-squared value
print(f'R-squared value: {r2}')
```

```
R-squared value: 0.42609705855058655
```

In [68]:
```
plt.scatter(y_pred, residuals)
plt.hlines(y=0, xmin=y_pred.min(), xmax=y_pred.max(), colors='red', linestyles='
plt.xlabel('Predicted Mean Income')
plt.ylabel('Residuals')
plt.title('Residuals of Regression Model')
plt.show()
```



In [69]:
```
plt.scatter(y_test, y_pred)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], '--r')
plt.xlabel('Actual Mean Income')
plt.ylabel('Predicted Mean Income')
plt.title('Actual vs. Predicted Mean Income')
plt.show()
```



Our linear regression model predicts mean income well based on food shopping behaviors, with an R-squared value of 0.426. A visual analysis of residual plots reveals that projections are rarely off by more than £10,000, highlighting the utility of food categories as income indicators.

References:

- Darmon N, Drewnowski A. Does social class predict diet quality? Am J Clin Nutr. 2008 May;87(5):1107-17. doi: 10.1093/ajcn/87.5.1107. PMID: 18469226.
- Smith, Lisa & Subandoro, Ali. (2007). Measuring Food Security Using Household Expenditure Surveys. International Food Policy Research Institute (IFPRI), Food security in practice technical guide series.
- Asfaw A. Does consumption of processed foods explain disparities in the body weight of individuals? The case of Guatemala. Health Econ. 2011 Feb;20(2):184-95. doi: 10.1002/hec.1579. PMID: 20029821.
- Aiello, L.M., Quercia, D., Schifanella, R. et al. Tesco Grocery 1.0, a large-scale dataset of grocery purchases in London. Sci Data 7, 57 (2020). https://doi.org/10.1038/s41597-020-0397-7

- Office for National Statistics https://www.ons.gov.uk/
- What Is Linear Regression? Types, Equation, Examples, and Best Practices for 2022 https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-linear-regression/
- Sanskar Wagavkar, Introduction to the Correlation Matrix https://builtin.com/data-science/correlation-matrix
- ADAM HAYES, icies

Descriptive Statistics: Definition, Overview, Types, https://www.investopedia.com/terms/d/descriptive_statistics.asp

- Niklas Donges, Random Forest: A Complete Guide for Machine Learning https://builtin.com/data-science/random-forest-algorithm
- A deeper look at the Tesco dataset using food categories, https://p-ada-wan.github.io/ Example

In [ ]:

In [ ]: