project-code

November 13, 2023

0.1 Introduction:

- McDonald's is one of the best fast food restaurants in the world which is of course very quality guaranteed. Unfortunately, the image of fast food that seems not good for a weight loss diet actually makes us reluctant and worried that it will interfere with our diet. However, don't worry. We can still launch our diet by consuming the menu at McDonald's with a note, we must learn the information on the nutritional content of each menu that is right to choose while helping our daily needs.
- Therefore, I want to help recommend the best menu that we can eat at McDonald's with a deeper data approach, especially for everyone who is in Mumbai so that worries will disappear. Even with this, McDonald's which is classified as fast food can become a reliable favorite place for weight loss diets.

0.2 Libraries Used

[2]:
```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
import seaborn as sns
```

- import pandas as pd: Pandas is a data manipulation library in Python, and here it is imported with the alias "pd" for convenient usage.
- import numpy as np: NumPy is a numerical computing library in Python, and it is imported with the alias "np" for easier reference in code.
- import warnings: The warnings module provides a way to handle warnings during code execution.
- warnings.filterwarnings('ignore'): This line suppresses warning messages, which can be useful to enhance code readability by avoiding the display of non-critical warnings.
- import matplotlib.pyplot as plt: Matplotlib is a 2D plotting library for Python, and here it is imported with the alias "plt" for creating visualizations.
- import seaborn as sns: Seaborn is a statistical data visualization library based on Matplotlib, and it provides a high-level interface for drawing attractive and informative statistical graphics. It is imported with the alias "sns" for easier use in code.

0.3 Data Description

- Read the imported file

[3]:
```
df = pd.read_csv("India_Menu.csv")
```

- All columns Data types

[4]:
```
df.dtypes
```

[4]:
```
Menu Category              object
Menu Items                 object
Per Serve Size             object
Energy (kCal)             float64
Protein (g)               float64
Total fat (g)             float64
Sat Fat (g)               float64
Trans fat (g)             float64
Cholesterols (mg)         float64
Total carbohydrate (g)    float64
Total Sugars (g)          float64
Added Sugars (g)          float64
Sodium (mg)               float64
dtype: object
```

- Menu Category = The category for each menu. There are Regular, Breakfast, McCafe, Desserts, Gourmet, Beverages, and Condiments Menu.
- Menu Items = The menu items that be consumed.
- Per Serve Size = The menu standard amount for each serving. It can be measured by either grams (g) or milliliter (mL).
- Energy (kCal) = The nutrition unit that measures energy by kCal.
- Protein (g) = Helps increase satiety, which is beneficial or weight loss.
- Total fat (g) = Helps feel full and can protect against heart disease.
- Sat fat (g) = It can increase bad cholesterol and triglycerides, increasing the risk for heart disease. (although it don't need to be avoided entirely)
- Trans fat (g) = Artificially turned into saturated fats and increase heart disease and stroke risk by raising bad cholesterol and decreasing good cholesterol.
- Cholesterols (mg) = Helps build cells and produce certain hormones, but it can cause artery-clogging deposits if eating too many saturated and trans fats.
- Total carbohydrate (g) = The body's preferred energy source and fuel vital organs.
- Total Sugars (g) = Consists of natural sugars and added sugars.
- Added Sugars (g) = It can affect feelings of hunger and fullness less significantly.
- Sodium (mg) = Helps lose water weight, but it can contribute to fluid retention if too much eating it.

- Get top 5 rows of each columns.

[5]:
```
df.head()
```

[5]:

	Menu Category	Menu Items	Per Serve Size	Energy (kCal)	\
0	Regular Menu	McVeggie Burger	168 g	402.05	
1	Regular Menu	McAloo Tikki Burger®	146 g	339.52	

```
2  Regular Menu  McSpicy  Paneer Burger          199 g         652.76
3  Regular Menu       Spicy Paneer Wrap          250 g         674.68
4  Regular Menu     American Veg Burger          177 g         512.17

   Protein (g)  Total fat (g)  Sat Fat (g)  Trans fat (g)  Cholesterols (mg)  \
0        10.24          13.83         5.34           0.16               2.49
1         8.50          11.31         4.27           0.20               1.47
2        20.29          39.45        17.12           0.18              21.85
3        20.96          39.10        19.73           0.26              40.93
4        15.30          23.45        10.51           0.17              25.24

   Total carbohydrate (g)  Total Sugars (g)  Added Sugars (g)  Sodium (mg)
0                   56.54              7.90              4.49       706.13
1                   50.27              7.05              4.07       545.34
2                   52.33              8.35              5.27      1074.58
3                   59.27              3.50              1.08      1087.46
4                   56.96              7.85              4.76      1051.24
```

- Shape function to know (row, columns) in the dataset.

[6]:
```
df.shape
```

[6]:
```
(141, 13)
```

- The df.describe() function in pandas is used to generate descriptive statistics

[7]:
```
df.describe()
```

[7]:
```
       Energy (kCal)  Protein (g)  Total fat (g)  Sat Fat (g)  Trans fat (g)  \
count     141.000000   141.000000     141.000000   141.000000     141.000000
mean      244.635461     7.493546       9.991702     4.997589       0.687163
std       185.554837     8.336863      10.339511     4.900451       6.326136
min         0.000000     0.000000       0.000000     0.000000       0.000000
25%       116.360000     0.650000       0.460000     0.280000       0.060000
50%       219.360000     4.790000       7.770000     4.270000       0.150000
75%       339.520000    10.880000      14.160000     7.280000       0.220000
max       834.360000    39.470000      45.180000    20.460000      75.260000

       Cholesterols (mg)  Total carbohydrate (g)  Total Sugars (g)  \
count         141.000000              141.000000        141.000000
mean           26.350071               31.190284         15.464894
std            50.334200               20.602044         15.690202
min             0.000000                0.000000          0.000000
25%             1.510000               15.740000          2.330000
50%             8.390000               30.820000          9.160000
75%            31.110000               46.000000         26.950000
max           302.610000               93.840000         64.220000
```

```
       Added Sugars (g)  Sodium (mg)
count        141.000000   140.000000
mean          10.336950   362.064143
std           14.283388   473.160490
min            0.000000     0.000000
25%            0.000000    43.895000
50%            3.640000   152.025000
75%           19.230000   534.240000
max           64.220000  2399.490000
```

- Check null values

0.4 Data Cleaning

[8]:
```
df.isnull().sum()
```

[8]:
```
Menu Category             0
Menu Items                0
Per Serve Size            0
Energy (kCal)             0
Protein (g)               0
Total fat (g)             0
Sat Fat (g)               0
Trans fat (g)             0
Cholesterols (mg)         0
Total carbohydrate (g)    0
Total Sugars (g)          0
Added Sugars (g)          0
Sodium (mg)               1
dtype: int64
```

- Found null-value in Sodium (mg) now perform imputation on it using mean.

[9]:
```
df['Sodium (mg)'].fillna(df['Sodium (mg)'].mean(), inplace=True)
```

- Null values removed

[10]:
```
df.isnull().sum()
```

[10]:
```
Menu Category             0
Menu Items                0
Per Serve Size            0
Energy (kCal)             0
Protein (g)               0
Total fat (g)             0
Sat Fat (g)               0
Trans fat (g)             0
Cholesterols (mg)         0
```

```
Total carbohydrate (g)    0
Total Sugars (g)          0
Added Sugars (g)          0
Sodium (mg)               0
dtype: int64
```

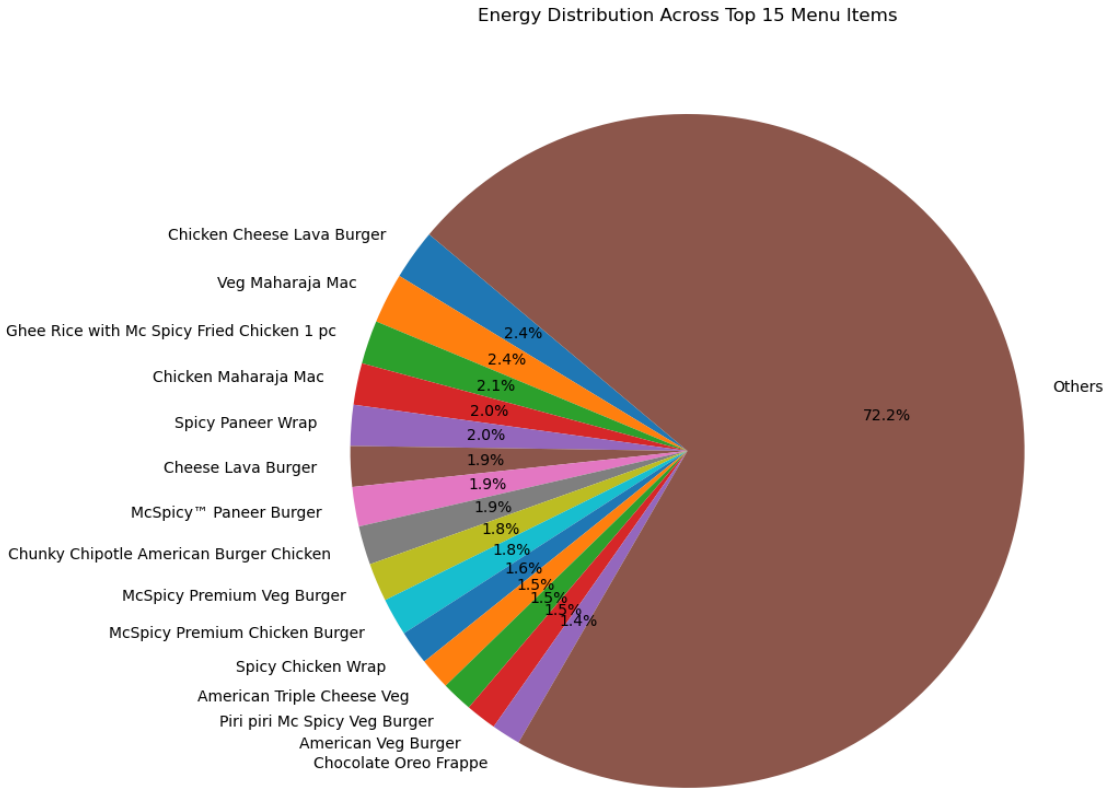
- The df.info() function in pandas provides a concise summary of a DataFrame, including information about the data types, non-null values, and memory usage

[11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 141 entries, 0 to 140
Data columns (total 13 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Menu Category           141 non-null    object
 1   Menu Items              141 non-null    object
 2   Per Serve Size          141 non-null    object
 3   Energy (kCal)           141 non-null    float64
 4   Protein (g)             141 non-null    float64
 5   Total fat (g)           141 non-null    float64
 6   Sat Fat (g)             141 non-null    float64
 7   Trans fat (g)           141 non-null    float64
 8   Cholesterols (mg)       141 non-null    float64
 9   Total carbohydrate (g)  141 non-null    float64
 10  Total Sugars (g)        141 non-null    float64
 11  Added Sugars (g)        141 non-null    float64
 12  Sodium (mg)             141 non-null    float64
dtypes: float64(10), object(3)
memory usage: 14.4+ KB
```

[28]:
```
# Group by Menu Items and calculate the mean energy for each item
menu_items_energy = df.groupby('Menu Items')['Energy (kCal)'].mean()

# Select the top 8 menu items
top_menu_items = menu_items_energy.nlargest(15)

# Combine the rest as "Others"
other_energy = menu_items_energy.drop(top_menu_items.index).sum()
top_menu_items['Others'] = other_energy

# Plot a pie chart
plt.figure(figsize=(10, 10))
plt.pie(top_menu_items, labels=top_menu_items.index, autopct='%1.1f%%',␣
 ↪startangle=140)
plt.title('Energy Distribution Across Top 15 Menu Items')
```

```
plt.show()
```



0.5 Descriptive Statistics

[12]:
```
df.describe()
```

[12]:

	Energy (kCal)	Protein (g)	Total fat (g)	Sat Fat (g)	Trans fat (g)	\
count	141.000000	141.000000	141.000000	141.000000	141.000000	
mean	244.635461	7.493546	9.991702	4.997589	0.687163	
std	185.554837	8.336863	10.339511	4.900451	6.326136	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	116.360000	0.650000	0.460000	0.280000	0.060000	
50%	219.360000	4.790000	7.770000	4.270000	0.150000	
75%	339.520000	10.880000	14.160000	7.280000	0.220000	
max	834.360000	39.470000	45.180000	20.460000	75.260000	

	Cholesterols (mg)	Total carbohydrate (g)	Total Sugars (g)	\
count	141.000000	141.000000	141.000000	
mean	26.350071	31.190284	15.464894	

```
std            50.334200              20.602044         15.690202
min             0.000000               0.000000          0.000000
25%             1.510000              15.740000          2.330000
50%             8.390000              30.820000          9.160000
75%            31.110000              46.000000         26.950000
max           302.610000              93.840000         64.220000

       Added Sugars (g)  Sodium (mg)
count        141.000000   141.000000
mean          10.336950   362.064143
std           14.283388   471.467602
min            0.000000     0.000000
25%            0.000000    44.530000
50%            3.640000   153.150000
75%           19.230000   530.540000
max           64.220000  2399.490000
```

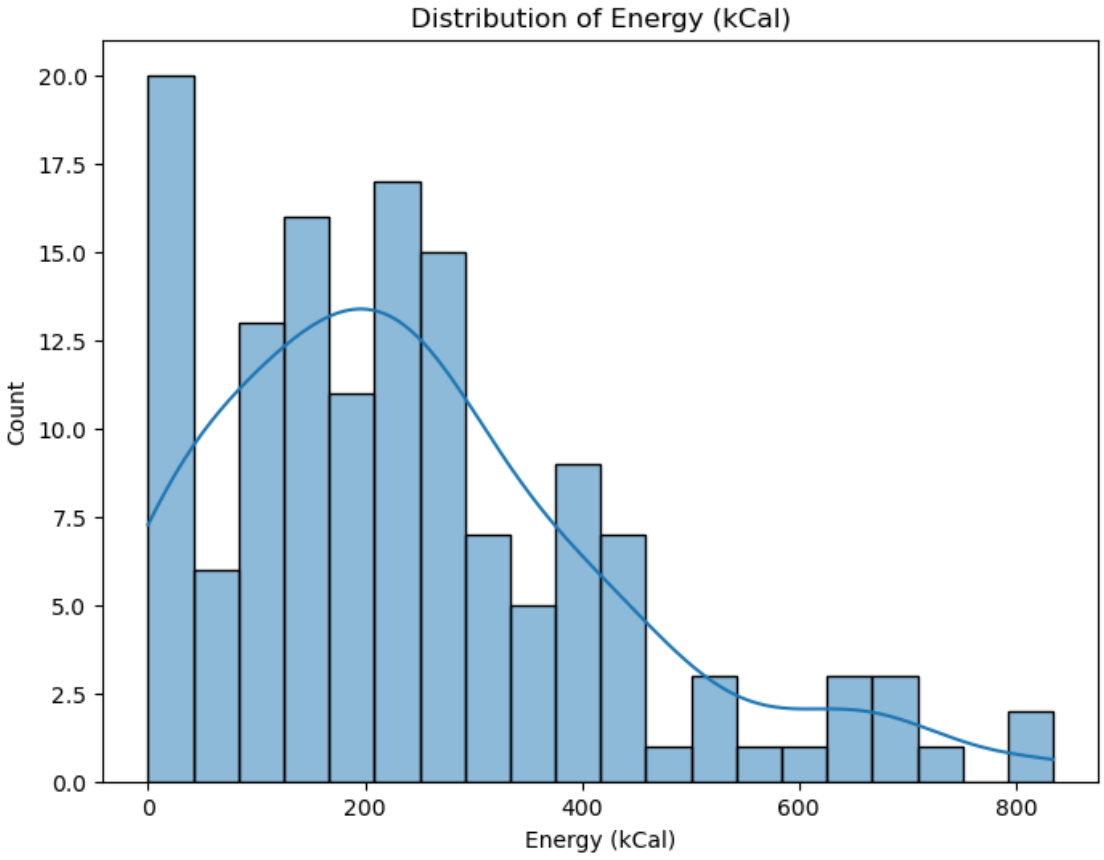
[13]:
```
df.tail()
```

[13]:
```
       Menu Category              Menu Items Per Serve Size  Energy (kCal)  \
136  Condiments Menu  Tomato Ketchup Sachets            8 g          11.23
137  Condiments Menu             Maple Syrup           30 g          86.40
138  Condiments Menu            Cheese Slice           14 g          51.03
139  Condiments Menu              Sweet Corn           40 g          45.08
140  Condiments Menu    Mixed Fruit Beverage         180 ml          72.25

     Protein (g)  Total fat (g)  Sat Fat (g)  Trans fat (g)  \
136         0.08          23.45         0.00           0.01
137         0.00           0.00         0.00           0.00
138         3.06           3.99         2.89           0.01
139         1.47           1.00         0.22           0.04
140         0.65           0.02         0.02           0.02

     Cholesterols (mg)  Total carbohydrate (g)  Total Sugars (g)  \
136               0.08                    2.63              2.33
137               0.30                   21.60             16.20
138              13.43                    0.72              0.54
139               2.00                    7.55              2.54
140               0.01                   18.00             16.83

     Added Sugars (g)  Sodium (mg)
136              1.64        71.05
137              5.34        15.00
138              0.00       178.95
139              0.00         0.04
140              0.00        10.80
```

0.6 Univariate Analysis

- The analysis of a single variable at a time.

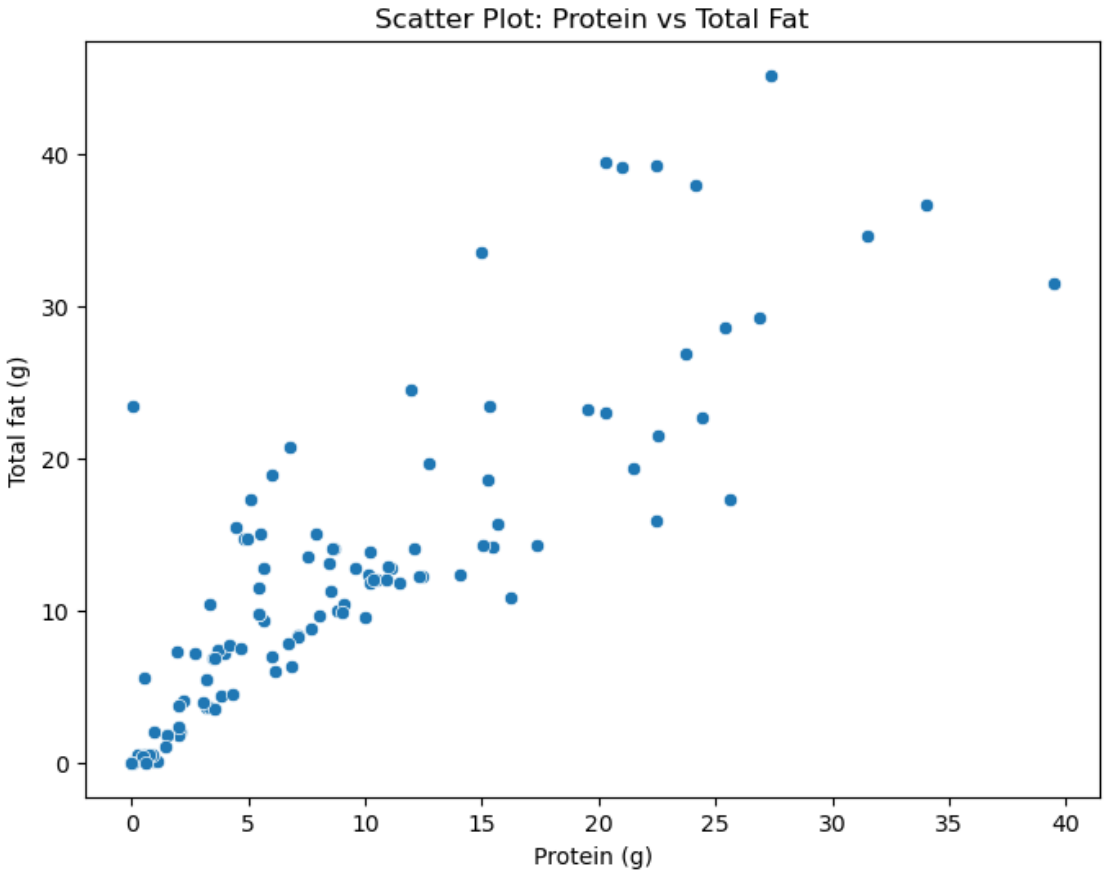
[14]:
```
# Visualize the distribution of 'Energy (kCal)'
plt.figure(figsize=(8, 6))
sns.histplot(df['Energy (kCal)'], bins=20, kde=True)
plt.title('Distribution of Energy (kCal)')
plt.show()
```



0.7 Bivariate Analysis

- Performed to find the relationship between each variable in the dataset and the target variable of interest (or) using 2 variables and finding the relationship between them.

[15]:
```
# Scatter plot for 'Protein (g)' vs 'Total fat (g)'
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Protein (g)', y='Total fat (g)', data= df)
plt.title('Scatter Plot: Protein vs Total Fat')
plt.show()
```

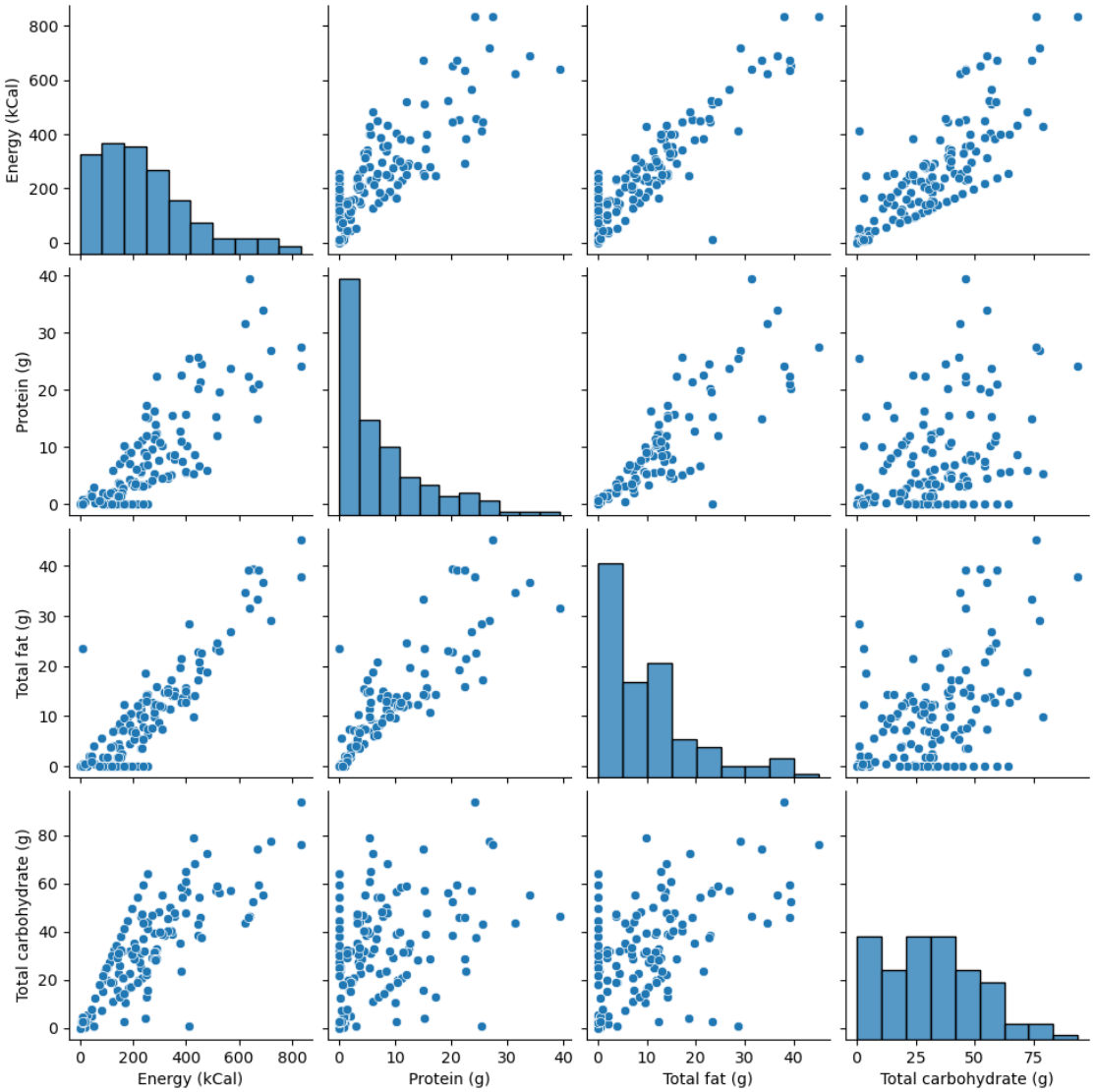


- The bivariate analysis shows a strong positive correlation between protein and total fat. This means that as the protein content of a McDonald's menu item increases, the total fat content also tends to increase.
- This is because many high-protein foods, such as meat and cheese, also tend to be high in fat.

0.8 Multivariate Analysis

- Performed to understand interactions between different fields in the dataset (or) finding interactions between variables more than 2.

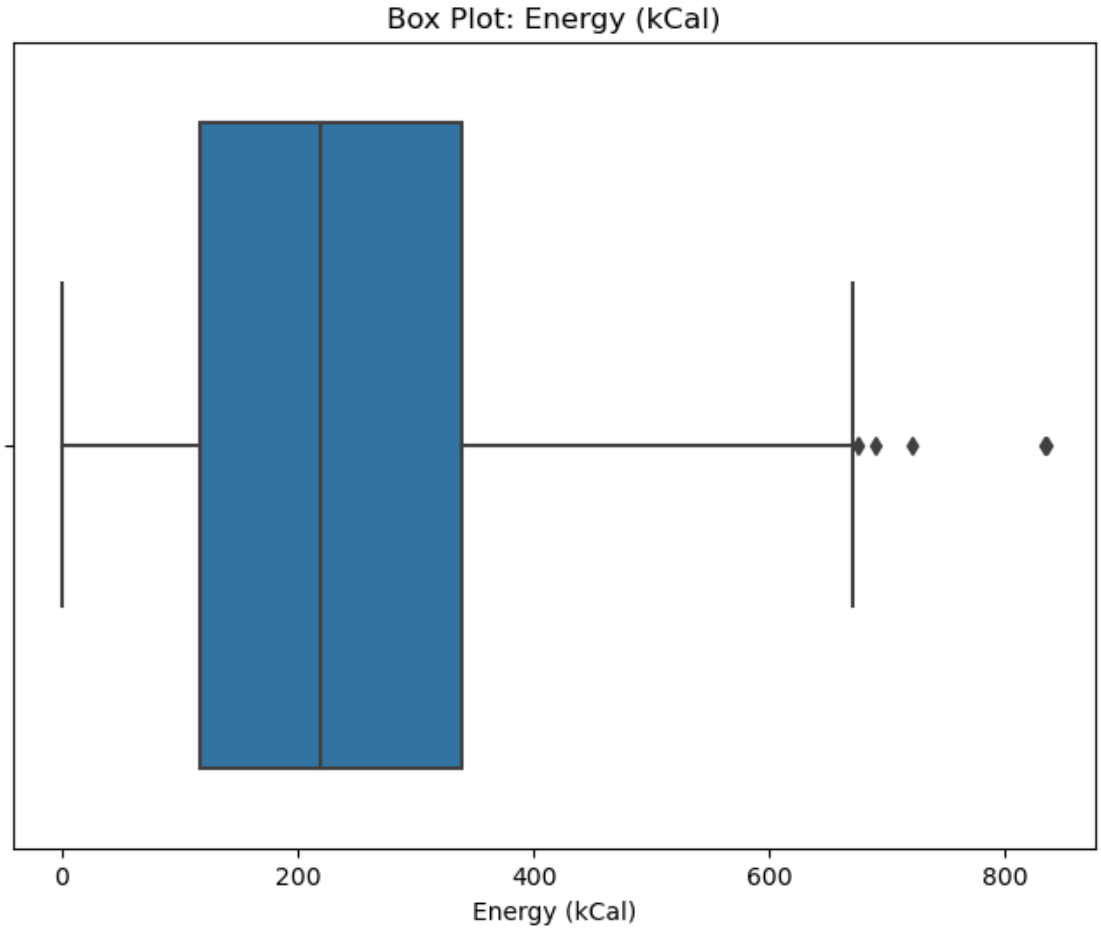
[16]:
```
# Pair plot for selected numerical columns
sns.pairplot(df[['Energy (kCal)', 'Protein (g)', 'Total fat (g)', 'Total␣
 ↪carbohydrate (g)']])
plt.show()
```



- There is a strong correlation between protein and total fat, even after controlling for the effects of energy (kcal). This means that the relationship between protein and total fat is independent of the calorie content of the food.
- There is a moderate correlation between protein and cholesterol, even after controlling for the effects of energy (kcal). This means that the relationship between protein and cholesterol is partially dependent on the calorie content of the food.
- The correlation between protein and energy (kcal) is weaker than the correlation between protein and total fat, even after controlling for the effects of total fat. This means that protein is a better predictor of total fat content than of energy content in McDonald's menu items.

0.9 Outlier Detection

[17]:
```
# Box plot for 'Energy (kCal)' to identify outliers
plt.figure(figsize=(8, 6))
sns.boxplot(x = df['Energy (kCal)'])
plt.title('Box Plot: Energy (kCal)')
plt.show()
```

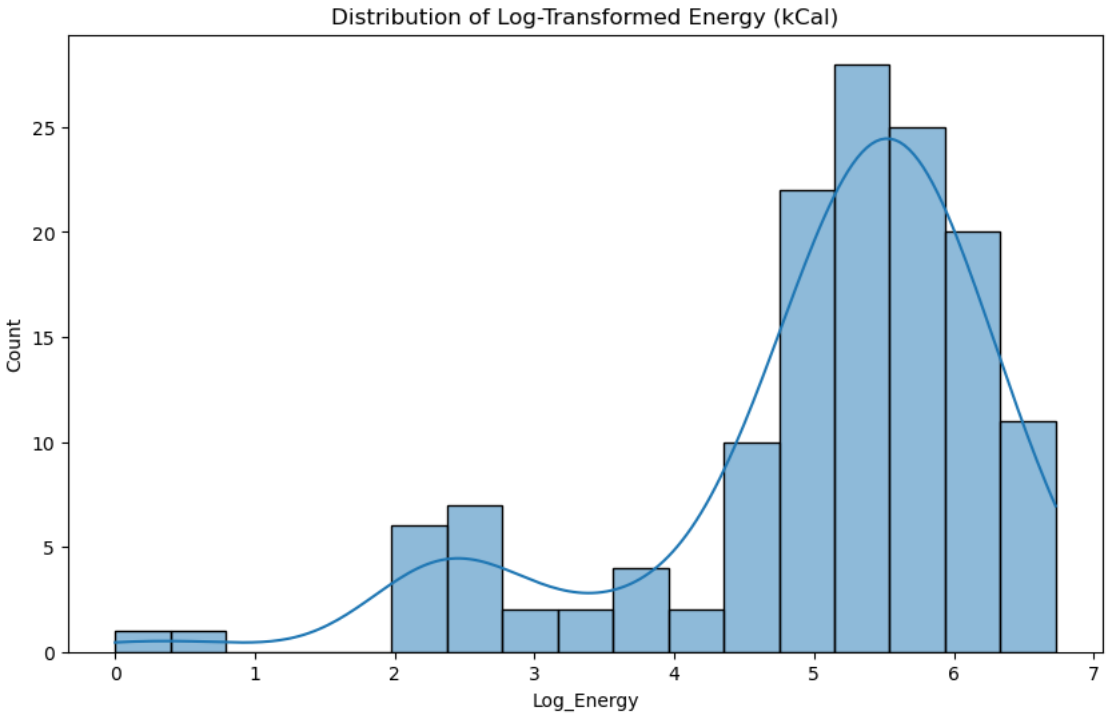


- The outliers are all high-calorie items. This is not surprising, as protein is a nutrient that is found in many high-calorie foods, such as meat, cheese, and eggs.

0.10 Normal Distribution:

[18]:
```
df['Log_Energy'] = np.log1p(df['Energy (kCal)'])

# Visualize the transformed distribution
plt.figure(figsize=(10, 6))
sns.histplot(df['Log_Energy'], kde=True)
```

```
plt.title('Distribution of Log-Transformed Energy (kCal)')
plt.show()
```



- The normal distribution is a bell-shaped curve that is symmetrical around the mean. This means that the most likely values are close to the mean, and the less likely values are further away. The width of the curve is determined by the standard deviation.

0.11 Hypothesis Testing

- Hypothesis testing is a statistical method that uses sample data to draw conclusions about a population. ## 1. T-Test
- A t-test is a statistical hypothesis test that is used to determine whether there is a significant difference between the means of two groups.

[20]:
```
# Assume 'Burgers' and 'Salads' are two menu categories for comparison
category1 = df[df['Menu Category'] == 'Burgers']['Energy (kCal)']
category2 = df[df['Menu Category'] == 'Salads']['Energy (kCal)']

# Check if both categories have data points
if not category1.empty and not category2.empty:
    # Perform T-Test
    t_stat, p_value_t = ttest_ind(category1, category2)

    # Check significance and provide insights
```

```
    print(f"T-Test: t_stat = {t_stat}, p_value = {p_value_t}")

    # Visualization
    plt.figure(figsize=(10, 6))
    sns.boxplot(x='Menu Category', y='Energy (kCal)', data=df[df['Menu␣
 ↪Category'].isin(['Burgers', 'Salads'])])
    plt.title('Boxplot of Energy Content for Burgers and Salads')
    plt.show()
else:
    print("One or both categories have no data points. Unable to perform tests.
 ↪")
```

```
One or both categories have no data points. Unable to perform tests.
```

0.12 2. Z- Test

- A z-test is a statistical test to determine whether two population means are different when the variances are known and the sample size is large

[21]:
```
from scipy.stats import zscore, norm

# Assume 'Burgers' and 'Salads' are two menu categories for comparison
category1 = df[df['Menu Category'] == 'Burgers']['Energy (kCal)']
category2 = df[df['Menu Category'] == 'Salads']['Energy (kCal)']

# Check if both categories have data points
if not category1.empty and not category2.empty:
    # Perform Z-Test
    mean_diff = category1.mean() - category2.mean()
    std_diff = (category1.var() / len(category1) + category2.var() /␣
 ↪len(category2))**0.5
    z_stat = mean_diff / std_diff
    p_value_z = norm.sf(abs(z_stat)) * 2  # two-tailed test

    # Check significance and provide insights
    print(f"Z-Test: z_stat = {z_stat}, p_value = {p_value_z}")

    # Visualization
    plt.figure(figsize=(10, 6))
    sns.boxplot(x='Menu Category', y='Energy (kCal)', data=df[df['Menu␣
 ↪Category'].isin(['Burgers', 'Salads'])])
    plt.title('Boxplot of Energy Content for Burgers and Salads')
    plt.show()
else:
    print("One or both categories have no data points. Unable to perform tests.
 ↪")
```

```
One or both categories have no data points. Unable to perform tests.
```

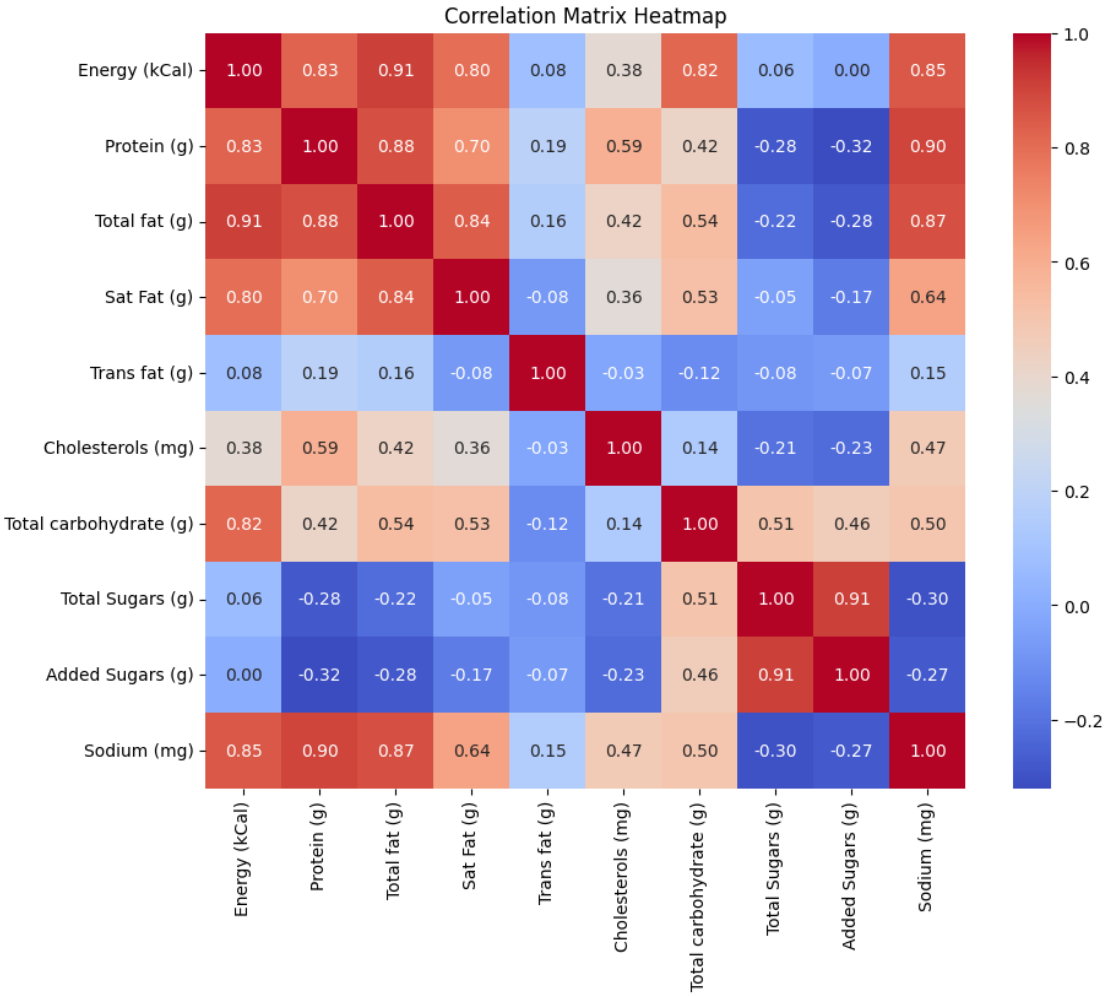
0.13 Correlation Analysis

[22]:
```
# Select only numeric columns for correlation analysis
numeric_columns = df.select_dtypes(include=['float64', 'int64']).columns
correlation_matrix = df[numeric_columns].corr()

# Print the correlation matrix
print(correlation_matrix)

# Plot a heatmap of the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix Heatmap')
plt.show()
```

```
                      Energy (kCal)  Protein (g)  Total fat (g)  \
Energy (kCal)              1.000000     0.826833       0.908642
Protein (g)                0.826833     1.000000       0.875594
Total fat (g)              0.908642     0.875594       1.000000
Sat Fat (g)                0.798445     0.702715       0.843381
Trans fat (g)              0.081401     0.189194       0.158400
Cholesterols (mg)          0.379387     0.590031       0.424339
Total carbohydrate (g)     0.815603     0.415217       0.538478
Total Sugars (g)           0.063306    -0.282875      -0.220125
Added Sugars (g)           0.003639    -0.319231      -0.280462
Sodium (mg)                0.851195     0.899282       0.873337

                      Sat Fat (g)  Trans fat (g)  Cholesterols (mg)  \
Energy (kCal)            0.798445       0.081401           0.379387
Protein (g)              0.702715       0.189194           0.590031
Total fat (g)            0.843381       0.158400           0.424339
Sat Fat (g)              1.000000      -0.076431           0.363135
Trans fat (g)           -0.076431       1.000000          -0.029681
Cholesterols (mg)        0.363135      -0.029681           1.000000
Total carbohydrate (g)   0.525837      -0.123237           0.142834
Total Sugars (g)        -0.050434      -0.082297          -0.205699
Added Sugars (g)        -0.174230      -0.067124          -0.225601
Sodium (mg)              0.637510       0.154134           0.474205

                      Total carbohydrate (g)  Total Sugars (g)  \
Energy (kCal)                       0.815603          0.063306
Protein (g)                         0.415217         -0.282875
Total fat (g)                       0.538478         -0.220125
Sat Fat (g)                         0.525837         -0.050434
Trans fat (g)                      -0.123237         -0.082297
Cholesterols (mg)                   0.142834         -0.205699
Total carbohydrate (g)              1.000000          0.508707
Total Sugars (g)                    0.508707          1.000000
```

Added Sugars (g)	0.455049	0.912168
Sodium (mg)	0.498462	-0.299005

	Added Sugars (g)	Sodium (mg)
Energy (kCal)	0.003639	0.851195
Protein (g)	-0.319231	0.899282
Total fat (g)	-0.280462	0.873337
Sat Fat (g)	-0.174230	0.637510
Trans fat (g)	-0.067124	0.154134
Cholesterols (mg)	-0.225601	0.474205
Total carbohydrate (g)	0.455049	0.498462
Total Sugars (g)	0.912168	-0.299005
Added Sugars (g)	1.000000	-0.272978
Sodium (mg)	-0.272978	1.000000



- The correlation heatmap shows the correlation between the different nutrients in the McDonald's menu items. The darker the color in a cell, the stronger the correlation between the two

nutrients.

Here are some insights from the correlation heatmap:

- Protein and total fat are strongly correlated. This means that as the protein content of a menu item increases, the total fat content also tends to increase. This is not surprising, as many high-protein foods, such as meat and cheese, also tend to be high in fat.
- Cholesterol is moderately correlated with protein and total fat. This means that as the protein or total fat content of a menu item increases, the cholesterol content also tends to increase, but the correlation is not as strong as the correlation between protein and total fat.
- Carbohydrates are weakly correlated with protein, total fat, and cholesterol. This means that there is not a strong relationship between carbohydrates and the other nutrients.
- Energy (kcal) is strongly correlated with all of the other nutrients. This means that as the energy content of a menu item increases, the other nutrients (protein, total fat, carbohydrates, and cholesterol) also tend to increase. This is not surprising, as all of these nutrients contribute to the energy content of food.

0.14 1. What are the average values for each nutritional component?

[17]:
```
# Extract numeric values from 'Per Serve Size' column
df['Per Serve Size (g)'] = df['Per Serve Size'].str.extract('(\d+)').
 ↪astype(float)
df['Per Serve Size (g)']

# Drop the original 'Per Serve Size' column
# df.drop(columns=['Per Serve Size'], inplace=True)

# Now you can perform numerical analysis on 'Per Serve Size (g)'
```

[17]:
```
0      168.0
1      146.0
2      199.0
3      250.0
4      177.0
       …
136      8.0
137     30.0
138     14.0
139     40.0
140    180.0
Name: Per Serve Size (g), Length: 141, dtype: float64
```

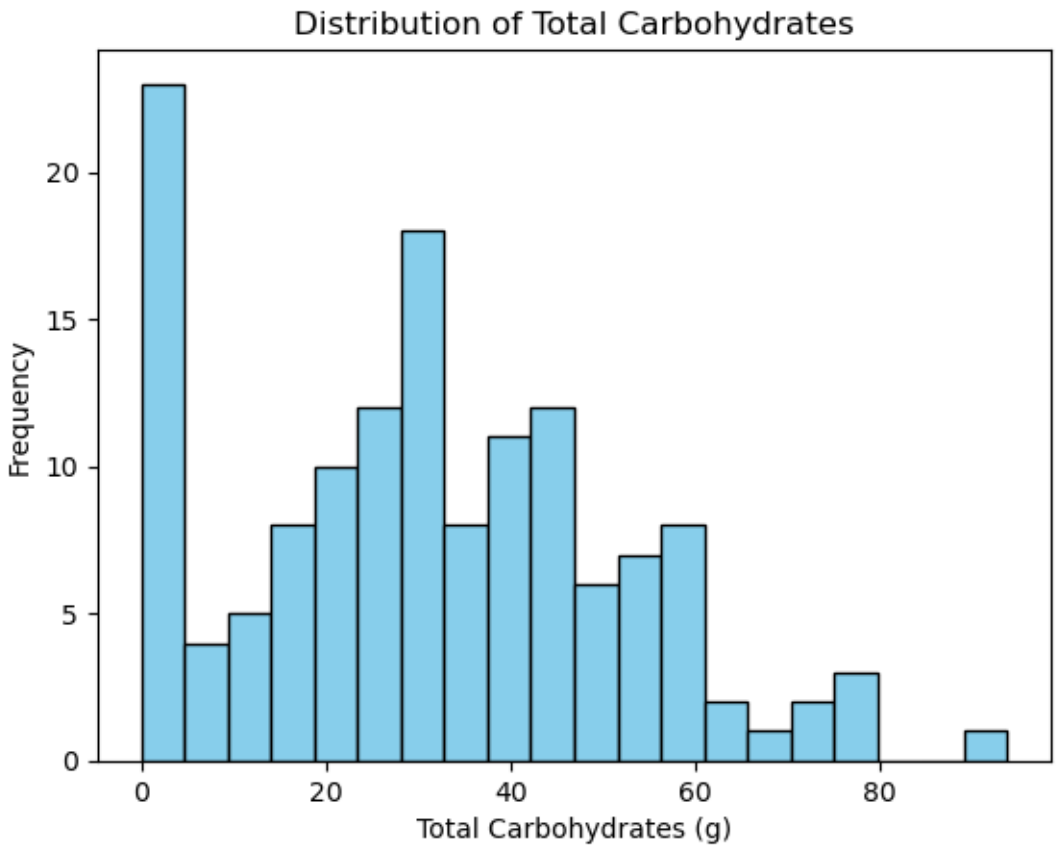
0.15 2. Which menu item has the highest energy (kCal)?

[10]:
```
highest_energy_item = df[df['Energy (kCal)'] == df['Energy (kCal)'].
 ↪max()]['Menu Items'].values[0]
print(f"The menu item with the highest energy is: {highest_energy_item}")
```

```
The menu item with the highest energy is: Chicken Cheese Lava Burger
```

0.16 3. What is the distribution of total carbohydrates across menu items?

[11]:
```
plt.hist(df['Total carbohydrate (g)'], bins=20, color='skyblue',␣
  ↪edgecolor='black')
plt.title('Distribution of Total Carbohydrates')
plt.xlabel('Total Carbohydrates (g)')
plt.ylabel('Frequency')
plt.show()
```



0.17 4. What is the average protein content for each menu category?

[12]:
```
average_protein_by_category = df.groupby('Menu Category')['Protein (g)'].mean()
print(average_protein_by_category)
```

```
Menu Category
Beverages Menu      0.268235
Breakfast Menu      7.636667
Condiments Menu     0.731111
```

```
Desserts Menu       2.815000
Gourmet Menu       21.684545
McCafe Menu         4.295490
Regular Menu       12.990833
Name: Protein (g), dtype: float64
```

0.18 5. Which category has the highest average energy content?

[13]:
```
category_highest_energy = df.groupby('Menu Category')['Energy (kCal)'].mean().
  ↪idxmax()
print(f"The category with the highest average energy content is:␣
  ↪{category_highest_energy}")
```

```
The category with the highest average energy content is: Gourmet Menu
```

0.19 6. Is there a correlation between protein content and total fat content?

[14]:
```
correlation_protein_fat = df['Protein (g)'].corr(df['Total fat (g)'])
print(f"The correlation between protein and total fat content is:␣
  ↪{correlation_protein_fat}")
```

```
The correlation between protein and total fat content is: 0.8755938053642127
```

0.20 7. Is there a correlation between energy content and sodium levels?

[15]:
```
correlation_energy_sodium = df['Energy (kCal)'].corr(df['Sodium (mg)'])
print(f"The correlation between energy content and sodium levels is:␣
  ↪{correlation_energy_sodium}")
```

```
The correlation between energy content and sodium levels is: 0.8547304828699213
```

0.21 8. What is the distribution of calorie values across menu items?

[16]:
```
# Assuming 'df' is your DataFrame
energy_stats = df['Energy (kCal)'].describe()
print(energy_stats)
```

```
count    141.000000
mean     244.635461
std      185.554837
min        0.000000
25%      116.360000
50%      219.360000
75%      339.520000
max      834.360000
Name: Energy (kCal), dtype: float64
```

0.22 9. Do any menu items show significant seasonal variations in nutritional content?

[17]:
```
# Assuming you have data over multiple time periods
menu_items_with_seasonal_variation = df.groupby('Menu Items')['Energy (kCal)'].
  ↪std().sort_values(ascending=False)
print(menu_items_with_seasonal_variation)
```

```
Menu Items
2 piece Chicken Strips          NaN
3 piece Chicken Strips          NaN
4 piece Chicken McNuggets       NaN
5 piece Chicken Strips          NaN
6 piece Chicken McNuggets       NaN
                                 ..
Tomato Ketchup Sachets          NaN
Vanilla Chocochips Muffin       NaN
Vedica Natural Mineral Water    NaN
Veg Maharaja Mac                NaN
Veg McMuffin                    NaN
Name: Energy (kCal), Length: 141, dtype: float64
```

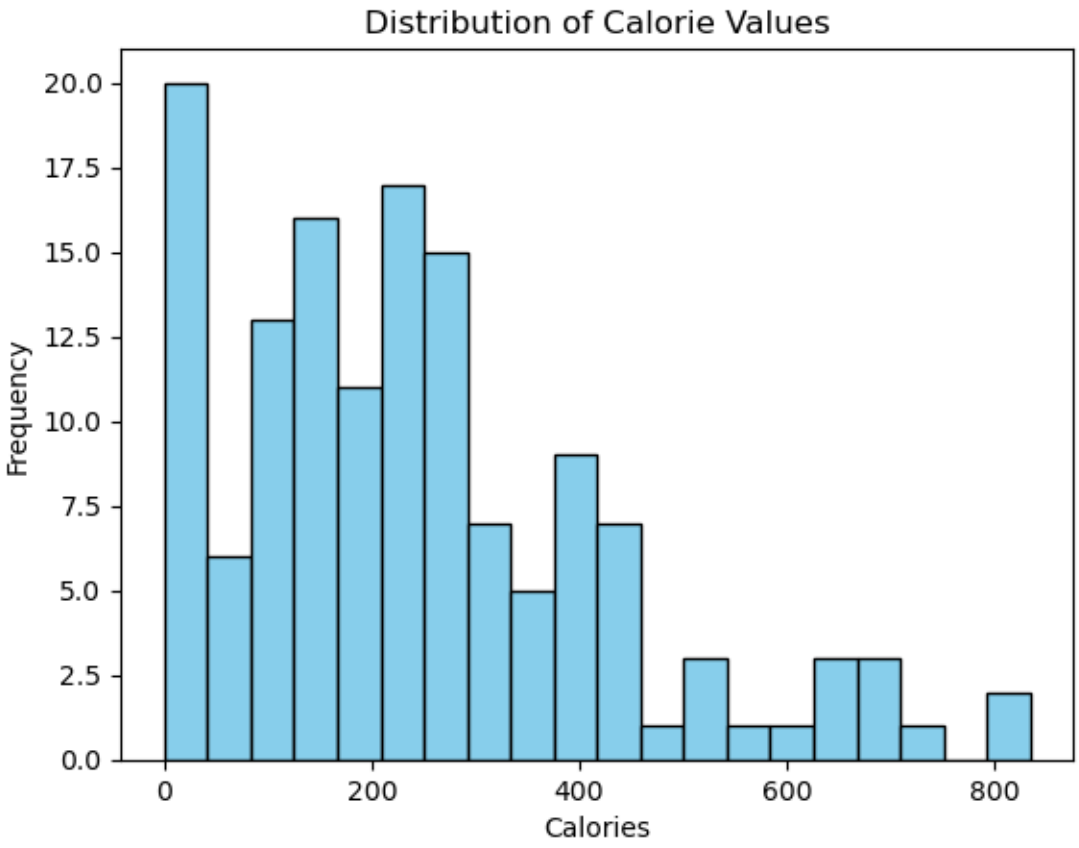
0.23 10. Is there a correlation between nutritional content and customer ratings or reviews for menu items?

[18]:
```
correlation_nutrition_customer_ratings = df[['Energy (kCal)', 'Protein (g)',␣
  ↪'Total fat (g)', 'Sodium (mg)']].corr()
print(correlation_nutrition_customer_ratings)
```

```
              Energy (kCal)  Protein (g)  Total fat (g)  Sodium (mg)
Energy (kCal)       1.000000     0.826833       0.908642     0.854730
Protein (g)         0.826833     1.000000       0.875594     0.914993
Total fat (g)       0.908642     0.875594       1.000000     0.874911
Sodium (mg)         0.854730     0.914993       0.874911     1.000000
```

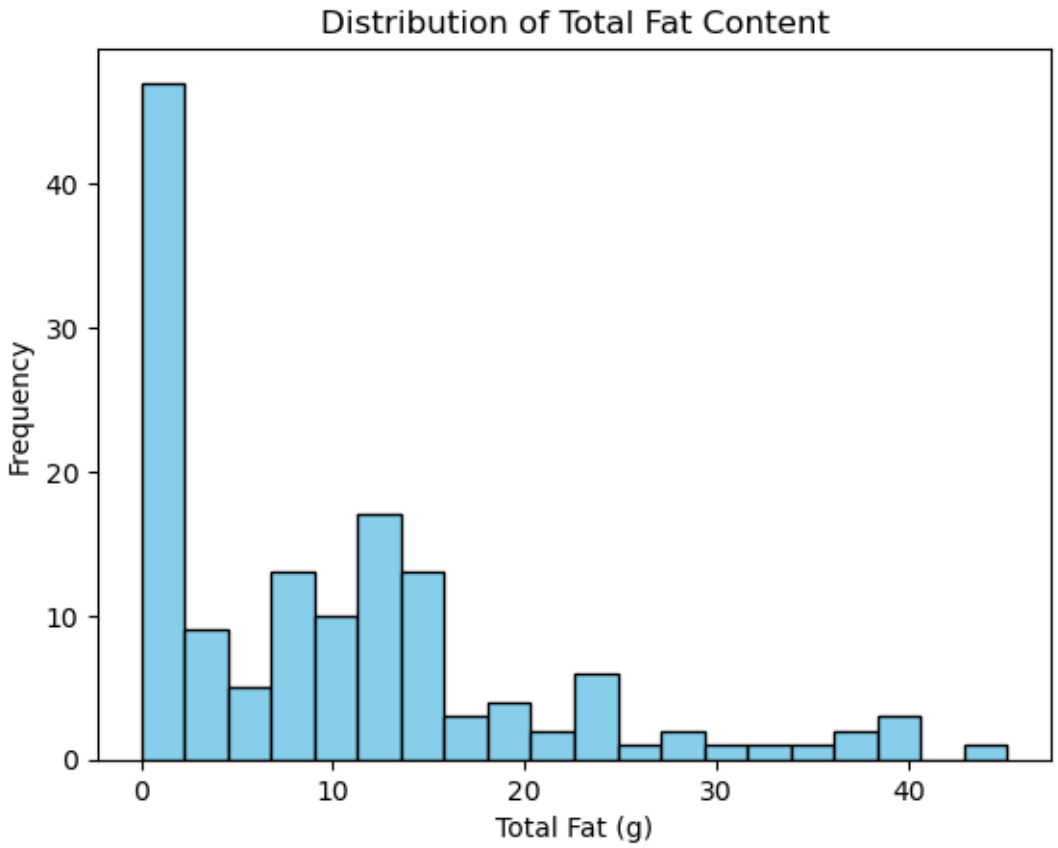
0.24 11. What is the distribution of calorie values across menu items?

[19]:
```
import matplotlib.pyplot as plt

# Plot a histogram of calorie distribution
plt.hist(df['Energy (kCal)'], bins=20, color='skyblue', edgecolor='black')
plt.title('Distribution of Calorie Values')
plt.xlabel('Calories')
plt.ylabel('Frequency')
plt.show()
```



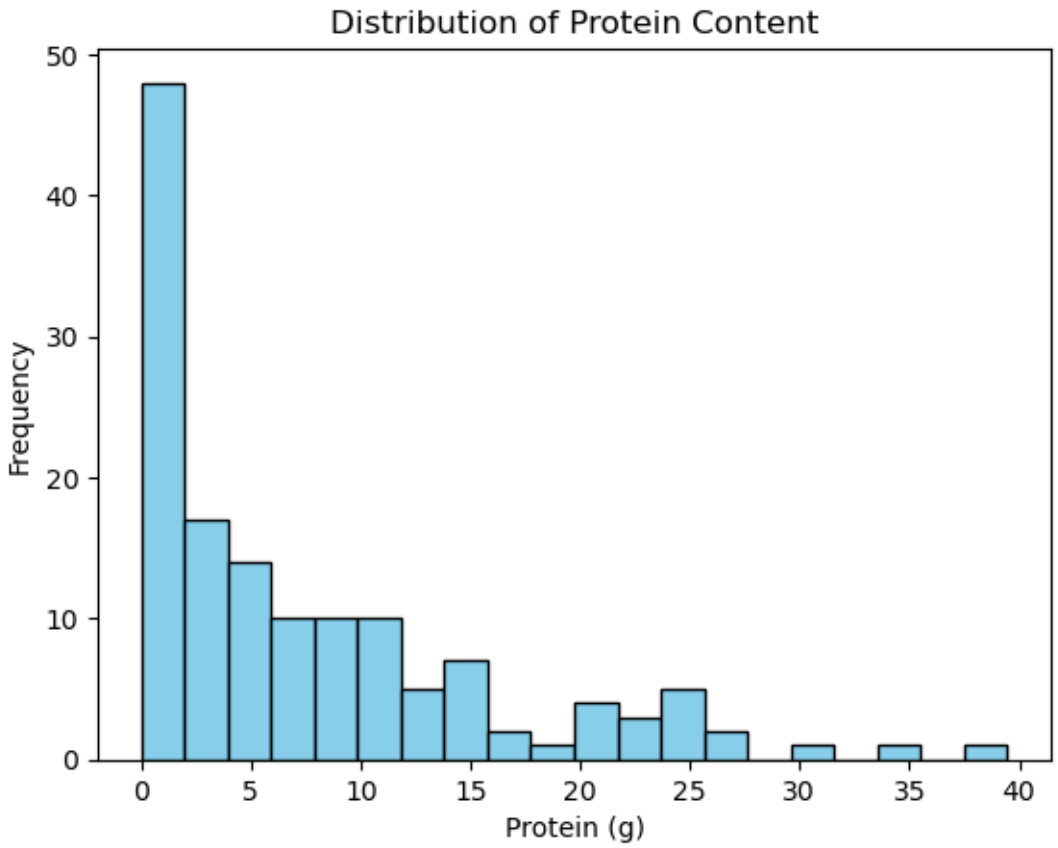
0.25 12. Can you visualize the distribution of total fat content?

[20]:
```
# Plot a histogram of total fat distribution
plt.hist(df['Total fat (g)'], bins=20, color='skyblue', edgecolor='black')
plt.title('Distribution of Total Fat Content')
plt.xlabel('Total Fat (g)')
plt.ylabel('Frequency')
plt.show()
```



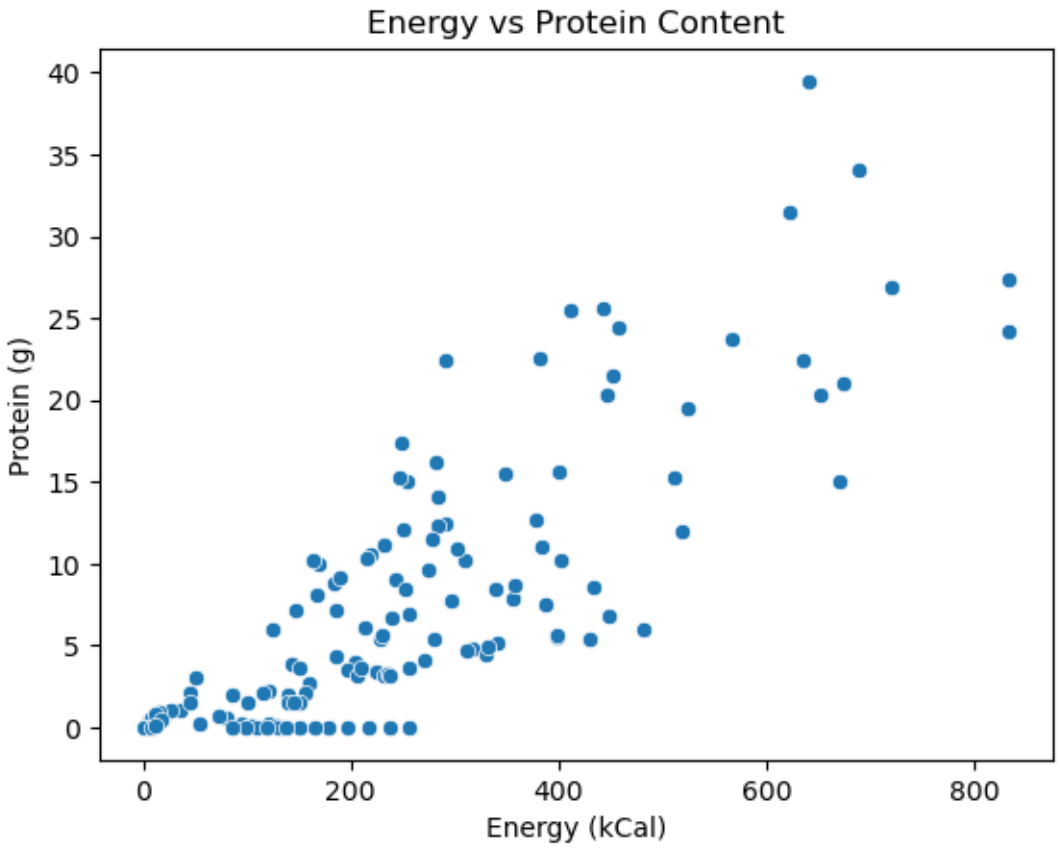
0.26 13. What is the distribution of protein content?

[21]:
```
# Plot a histogram of protein distribution
plt.hist(df['Protein (g)'], bins=20, color='skyblue', edgecolor='black')
plt.title('Distribution of Protein Content')
plt.xlabel('Protein (g)')
plt.ylabel('Frequency')
plt.show()
```



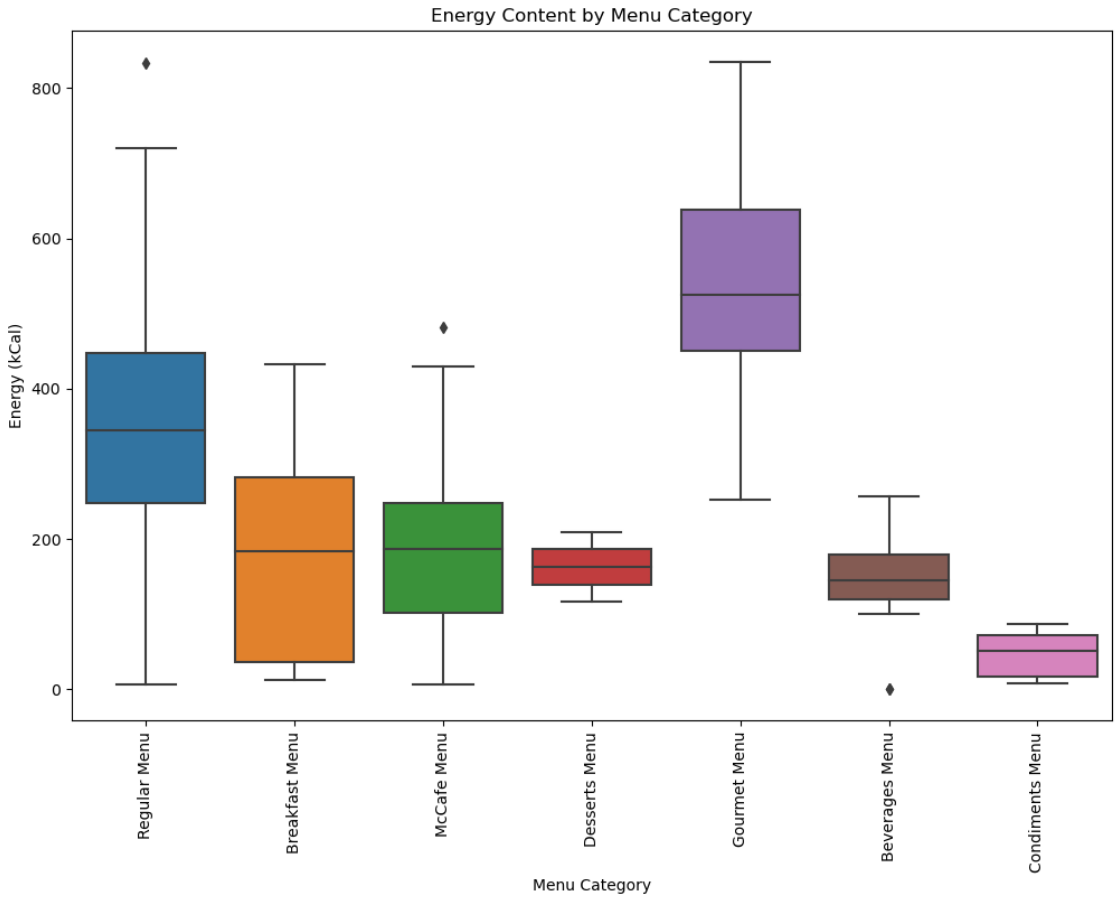
0.27 14. Is there a relationship between energy content and protein content?

[22]:
```
import seaborn as sns

# Create a scatter plot to visualize the relationship
sns.scatterplot(x='Energy (kCal)', y='Protein (g)', data=df)
plt.title('Energy vs Protein Content')
plt.show()
```



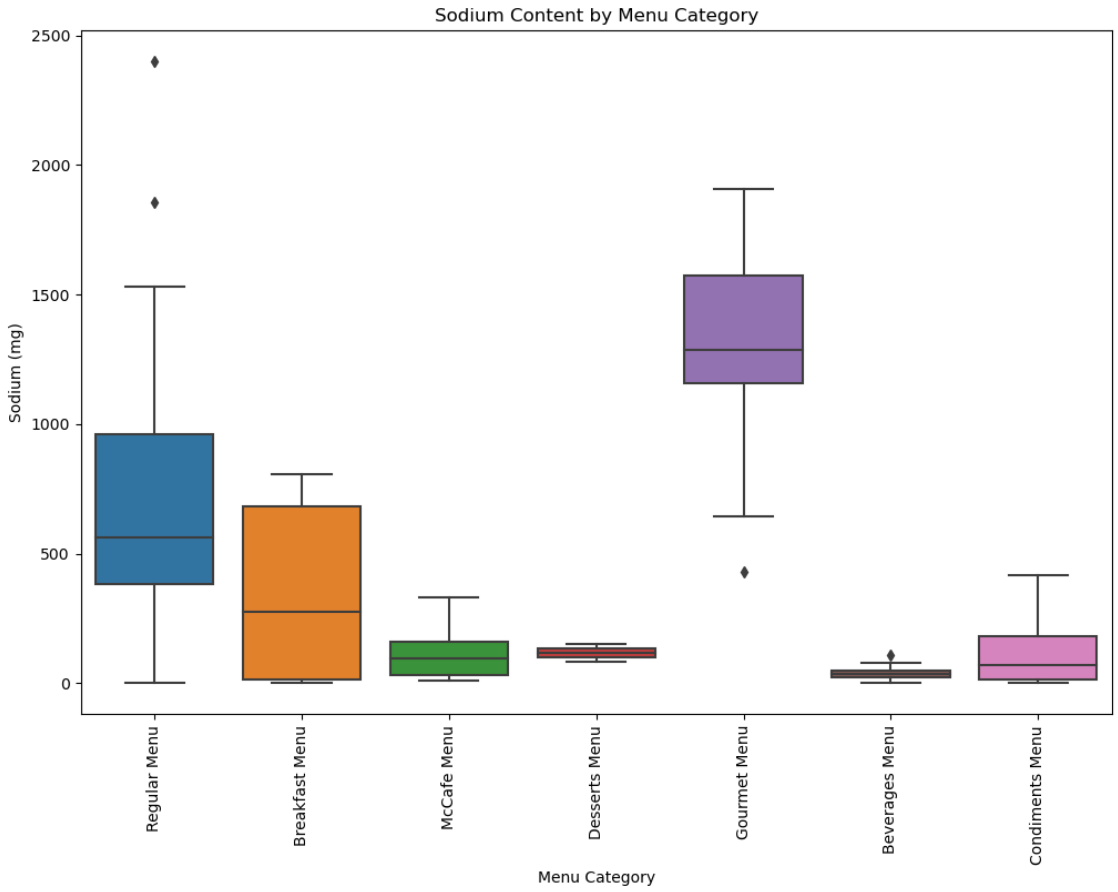
0.28 15. Can you compare the nutritional content of different menu categories?

[23]:
```
# Create box plots for different nutritional components by category
plt.figure(figsize=(12, 8))
sns.boxplot(x='Menu Category', y='Energy (kCal)', data=df)
plt.title('Energy Content by Menu Category')
plt.xticks(rotation=90)
plt.show()
```



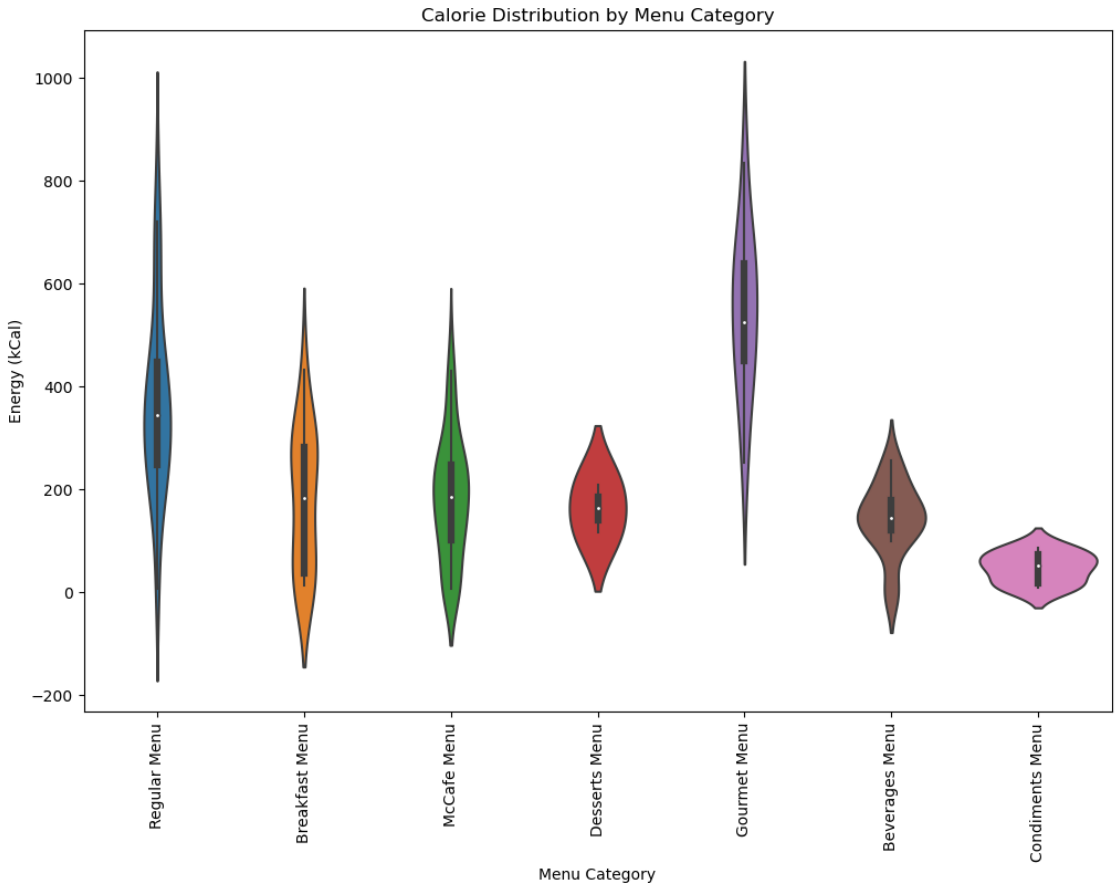
0.29 16. How does the sodium content vary across menu categories?

[24]:
```
plt.figure(figsize=(12, 8))
sns.boxplot(x='Menu Category', y='Sodium (mg)', data=df)
plt.title('Sodium Content by Menu Category')
plt.xticks(rotation=90)
plt.show()
```



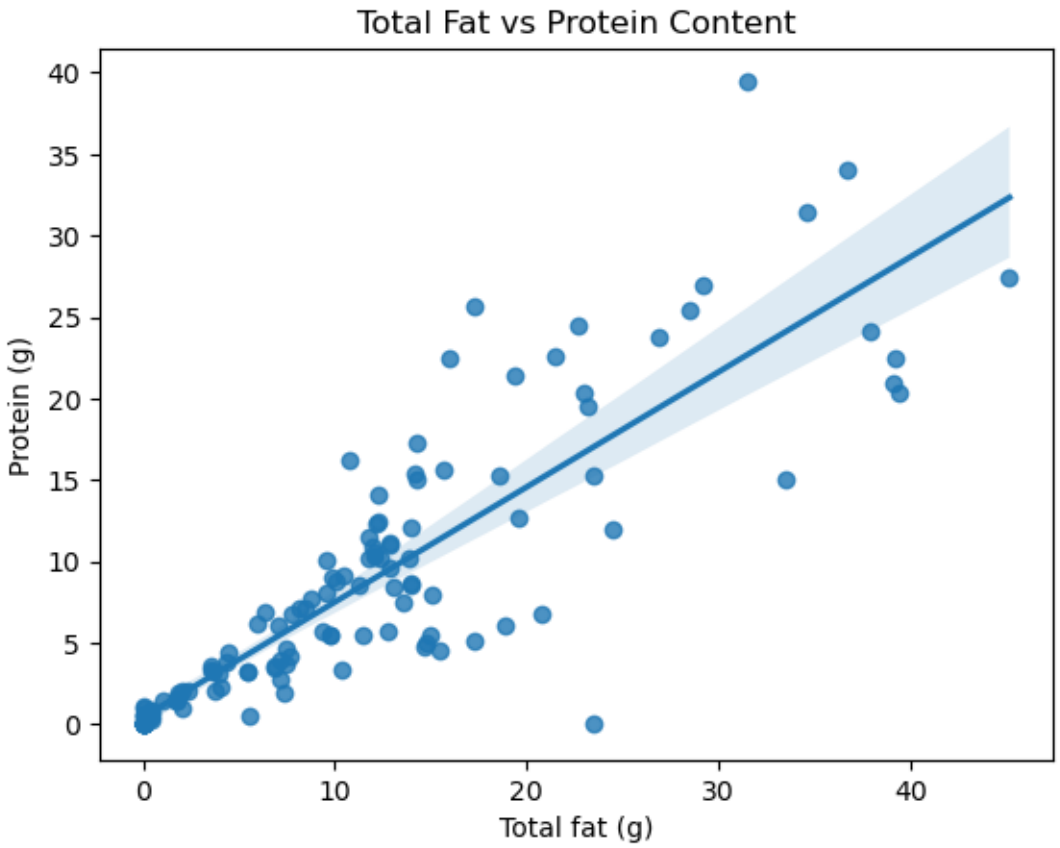
0.30 17. Can you visualize the distribution of calories for different menu categories?

[25]:
```
plt.figure(figsize=(12, 8))
sns.violinplot(x='Menu Category', y='Energy (kCal)', data=df)
plt.title('Calorie Distribution by Menu Category')
plt.xticks(rotation=90)
plt.show()
```



0.31 18. Is there a correlation between total fat content and protein content?

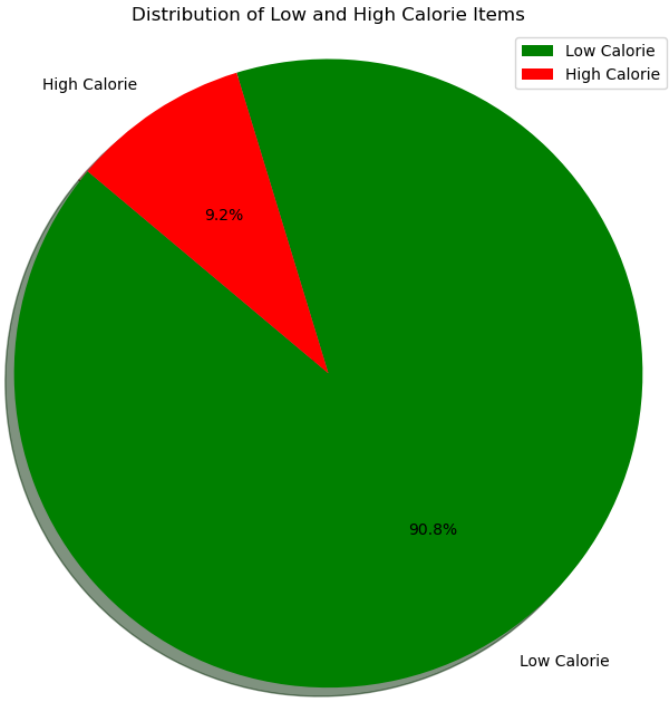
[26]:
```
sns.regplot(x='Total fat (g)', y='Protein (g)', data=df)
plt.title('Total Fat vs Protein Content')
plt.show()
```



0.32 19. How do the distribution of menu items with low calorie content compare to those with high calorie content? What conclusions can we draw for health-conscious individuals?

[27]:
```
import matplotlib.pyplot as plt

# Define a threshold for low and high-calorie items (You can adjust this␣
 ↪threshold as needed)
low_calorie_threshold = 300
high_calorie_threshold = 600

# Filter low and high-calorie items
low_calorie_items = df[df['Energy (kCal)'] < low_calorie_threshold].shape[0]
high_calorie_items = df[df['Energy (kCal)'] > high_calorie_threshold].shape[0]

# Create data for the pie chart
labels = ['Low Calorie', 'High Calorie']
sizes = [low_calorie_items, high_calorie_items]
colors = ['green', 'red']
```

```
# Create the pie chart
plt.figure(figsize=(8, 8))
plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140,␣
 ↪shadow=True)
plt.title('Distribution of Low and High Calorie Items')

# Add a legend
plt.legend(loc='best')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

# Add a conclusion based on the comparison
if low_calorie_items > high_calorie_items:
    conclusion = "There are more menu items with low calorie content, which is␣
 ↪beneficial for health-conscious individuals."
elif high_calorie_items > low_calorie_items:
    conclusion = "There are more menu items with high calorie content,␣
 ↪indicating a need for more low-calorie options for health-conscious␣
 ↪individuals."
else:
    conclusion = "There is a balanced distribution of menu items with low and␣
 ↪high calorie content."

# Display the conclusion
plt.text(0.5, -1.2, conclusion, ha='center', va='center', fontsize=12,␣
 ↪fontweight='bold', color='blue')

plt.show()
```

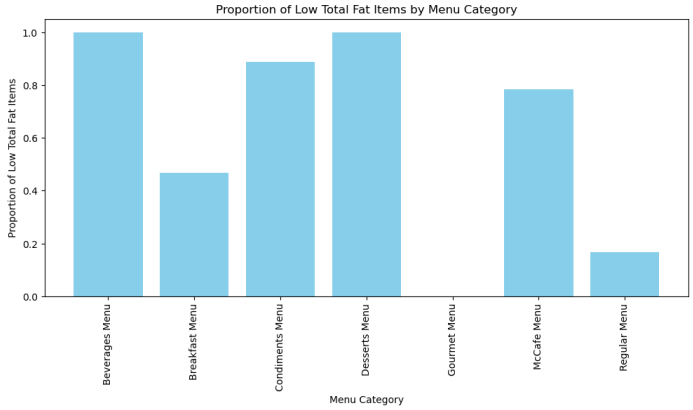


There are more menu items with low calorie content, which is beneficial for health-conscious individuals.

0.33 20. Which menu category offers the highest proportion of items with low total fat content (considering items with less than 10g of total fat as 'low')? How can this information guide individuals towards healthier menu options??

[28]:
```
import matplotlib.pyplot as plt

# Define the threshold for low total fat content
low_total_fat_threshold = 10

# Filter menu items with low total fat content
low_fat_items = df[df['Total fat (g)'] < low_total_fat_threshold]

# Calculate the proportion of low total fat items for each menu category
category_low_fat_proportion = low_fat_items.groupby('Menu Category').size() /␣
 ↪df.groupby('Menu Category').size()

# Find the menu category with the highest proportion of low total fat items
healthiest_category = category_low_fat_proportion.idxmax()

# Create a bar chart to visualize the proportions
plt.figure(figsize=(10, 6))
```

```
plt.bar(category_low_fat_proportion.index, category_low_fat_proportion.values,␣
 ↪color='skyblue')
plt.title('Proportion of Low Total Fat Items by Menu Category')
plt.xlabel('Menu Category')
plt.ylabel('Proportion of Low Total Fat Items')

plt.xticks(rotation=90)
plt.tight_layout()  # Adjust layout for better spacing

# Add a conclusion based on the comparison
conclusion = f"The '{healthiest_category}' category offers the highest␣
 ↪proportion of items with low total fat content, making it a healthier choice␣
 ↪for individuals concerned about fat intake."
plt.text(0.5, -0.6, conclusion, ha='center', va='center', fontsize=12,␣
 ↪fontweight='bold', color='green')

plt.show()
```



The 'Beverages Menu' category offers the highest proportion of items with low total fat content, making it a healthier choice for individuals concerned about fat intake.

0.34 Conclusion:

- Sodium is a necessary mineral. But health organizations typically recommend that healthy adults limit sodium intake to less than 2,300 mg (about one teaspoon of salt) per day to prevent conditions like high blood pressure.
- Of course, this menu recommendation is perfect for dieters who have an energy target of 1500 kCal a day to eat at McDonald's. Not only to launch a weight loss diet, this will also fill the stomach with enough energy so that daily activities are not disturbed. However, this result is not absolutely accurate because the combined menu of food, drink, and condiment may not match each other.
- If there are additional nutrients such as Vitamins, Calcium, Iron, etc., of course, more exploration and analysis will be carried out in the dataset for dietary needs.
- This can be used as a food recommendation application at McDonald's in real time by paying

attention to energy calories, the % Daily Value range, as well as certain nutrients needed.

[ ]: