


```
import numpy
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os;

import pandas as pd

# Use the raw content URL for GitHub files, ensuring to use the 'raw' version
food = pd.read_csv('https://github.com/satyanarayanan102/OIBSIP/blob/main/Nutrition-Facts-Menu-Analysis/menu_data.csv')

# Optionally, you can add error handling to skip problematic rows
# food = pd.read_csv('https://raw.githubusercontent.com/PragadishTRS/Nutrition_Menu_Analysis/main/menu_data.csv', error_bad_lines=False)
```

food #loading the dataset



	Category	Item	Serving Size	Calories	Calories from Fat	Total Fat	Total Fat (% Daily Value)	Saturated Fat	Saturated Fat (% Daily Value)	Trans Fat	...	Carbohydrates	Carbohydrates (% Daily Value)	
0	Breakfast	Egg McMuffin	4.8 oz (136 g)	300	120	13.0	20	5.0	25	0.0	...	31	10	
1	Breakfast	Egg White Delight	4.8 oz (135 g)	250	70	8.0	12	3.0	15	0.0	...	30	10	
2	Breakfast	Sausage McMuffin	3.9 oz (111 g)	370	200	23.0	35	8.0	42	0.0	...	29	10	
3	Breakfast	Sausage McMuffin with Egg	5.7 oz (161 g)	450	250	28.0	43	10.0	52	0.0	...	30	10	
4	Breakfast	Sausage McMuffin with Egg Whites	5.7 oz (161 g)	400	210	23.0	35	8.0	42	0.0	...	30	10	
...	
255	Smoothies & Shakes	McFlurry with Oreo Cookies (Small)	10.1 oz (285 g)	510	150	17.0	26	9.0	44	0.5	...	80	27	
256	Smoothies & Shakes	McFlurry with Oreo Cookies (Medium)	13.4 oz (381 g)	690	200	23.0	35	12.0	58	1.0	...	106	35	
257	Smoothies & Shakes	McFlurry with Oreo Cookies (Snack)	6.7 oz (190 g)	340	100	11.0	17	6.0	29	0.0	...	53	18	
258	Smoothies & Shakes	McFlurry with Reese's Peanut Butter Cups (Medium)	14.2 oz (403 g)	810	290	32.0	50	15.0	76	1.0	...	114	38	
259	Smoothies & Shakes	McFlurry with Reese's Peanut Butter Cups (Snack)	7.1 oz (202 g)	410	150	16.0	25	8.0	38	0.0	...	57	19	

260 rows × 24 columns

```
print("Dimension of the dataset:", food.shape)
print("Size of the dataset: ", food.size)
print("Columns in the dataset: ", food.keys())
```

```

Dimension of the dataset: (260, 24)
Size of the dataset: 6240
Columns in the dataset: Index(['Category', 'Item', 'Serving Size', 'Calories', 'Calories from Fat',
'Total Fat', 'Total Fat (% Daily Value)', 'Saturated Fat',
'Saturated Fat (% Daily Value)', 'Trans Fat', 'Cholesterol',
'Cholesterol (% Daily Value)', 'Sodium', 'Sodium (% Daily Value)',
'Carbohydrates', 'Carbohydrates (% Daily Value)', 'Dietary Fiber',
'Dietary Fiber (% Daily Value)', 'Sugars', 'Protein',
'Vitamin A (% Daily Value)', 'Vitamin C (% Daily Value)',
'Calcium (% Daily Value)', 'Iron (% Daily Value)'],
dtype='object')

```

```

print("Information of the dataset: \n",food.info())
print("After removing duplicates: \n",food.drop_duplicates(inplace=True))
print("Size of the dataset after removing duplicates: \n",food.size)

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 260 entries, 0 to 259
Data columns (total 24 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Category                             260 non-null    object
1   Item                                 260 non-null    object
2   Serving Size                         260 non-null    object
3   Calories                             260 non-null    int64
4   Calories from Fat                    260 non-null    int64
5   Total Fat                           260 non-null    float64
6   Total Fat (% Daily Value)            260 non-null    int64
7   Saturated Fat                       260 non-null    float64
8   Saturated Fat (% Daily Value)        260 non-null    int64
9   Trans Fat                           260 non-null    float64
10  Cholesterol                          260 non-null    int64
11  Cholesterol (% Daily Value)          260 non-null    int64
12  Sodium                              260 non-null    int64
13  Sodium (% Daily Value)               260 non-null    int64
14  Carbohydrates                       260 non-null    int64
15  Carbohydrates (% Daily Value)        260 non-null    int64
16  Dietary Fiber                       260 non-null    int64
17  Dietary Fiber (% Daily Value)         260 non-null    int64
18  Sugars                              260 non-null    int64
19  Protein                             260 non-null    int64
20  Vitamin A (% Daily Value)            260 non-null    int64
21  Vitamin C (% Daily Value)            260 non-null    int64
22  Calcium (% Daily Value)              260 non-null    int64
23  Iron (% Daily Value)                 260 non-null    int64
dtypes: float64(3), int64(18), object(3)
memory usage: 48.9+ KB
Information of the dataset:
None
After removing duplicates:
None
Size of the dataset after removing duplicates:
6240

```

```
print("Description of the dataset: \n",food.describe())
```

```

Description of the dataset:

```

	Calories	Calories from Fat	Total Fat	Total Fat (% Daily Value)
count	260.000000	260.000000	260.000000	260.000000
mean	368.269231	127.096154	14.165385	21.815385
std	240.269886	127.875914	14.205998	21.885199
min	0.000000	0.000000	0.000000	0.000000
25%	210.000000	20.000000	2.375000	3.750000
50%	340.000000	100.000000	11.000000	17.000000
75%	500.000000	200.000000	22.250000	35.000000
max	1880.000000	1060.000000	118.000000	182.000000

	Saturated Fat	Saturated Fat (% Daily Value)	Trans Fat	Cholesterol
count	260.000000	260.000000	260.000000	260.000000
mean	6.007692	29.965385	0.203846	54.942308
std	5.321873	26.639209	0.429133	87.269257
min	0.000000	0.000000	0.000000	0.000000
25%	1.000000	4.750000	0.000000	5.000000
50%	5.000000	24.000000	0.000000	35.000000
75%	10.000000	48.000000	0.000000	65.000000
max	20.000000	102.000000	2.500000	575.000000

	Cholesterol (% Daily Value)	Sodium	...	Carbohydrates
count	260.000000	260.000000	...	260.000000
mean	18.392308	495.750000	...	47.346154
std	29.091653	577.026323	...	28.252232
min	0.000000	0.000000	...	0.000000

25%	2.000000	107.500000	...	30.000000
50%	11.000000	190.000000	...	44.000000
75%	21.250000	865.000000	...	60.000000
max	192.000000	3600.000000	...	141.000000

	Carbohydrates (% Daily Value)	Dietary Fiber \
count	260.000000	260.000000
mean	15.780769	1.630769
std	9.419544	1.567717
min	0.000000	0.000000
25%	10.000000	0.000000
50%	15.000000	1.000000
75%	20.000000	3.000000
max	47.000000	7.000000

	Dietary Fiber (% Daily Value)	Sugars	Protein \
count	260.000000	260.000000	260.000000
mean	6.530769	29.423077	13.338462
std	6.307057	28.679797	11.426146
min	0.000000	0.000000	0.000000
25%	0.000000	5.750000	4.000000
50%	5.000000	17.500000	12.000000
75%	10.000000	48.000000	19.000000
max	28.000000	128.000000	87.000000

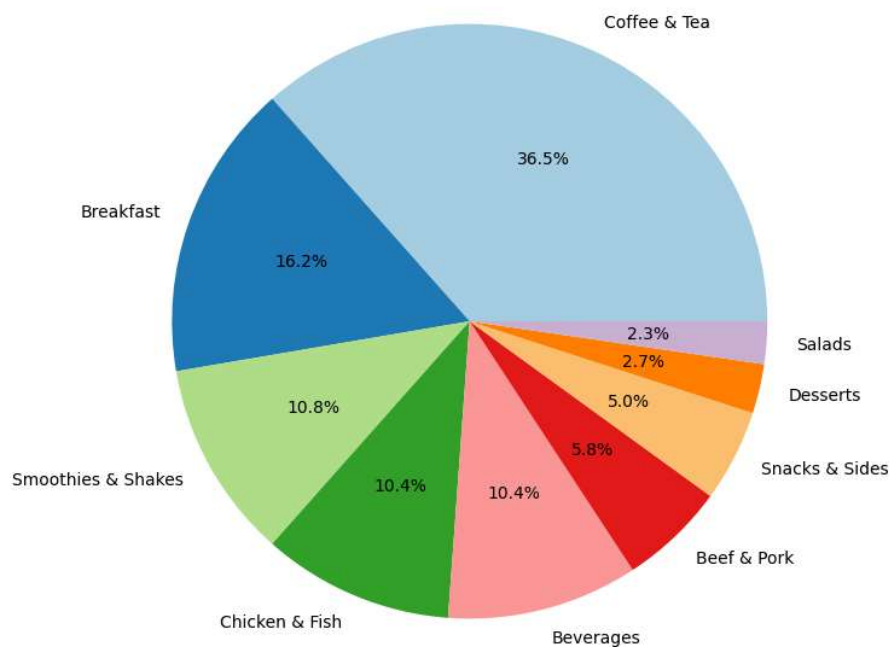
	Vitamin A (% Daily Value)	Vitamin C (% Daily Value) \
count	260.000000	260.000000
mean	13.426923	8.534615
std	24.366381	26.345542
min	0.000000	0.000000
25%	2.000000	0.000000
50%	8.000000	0.000000

```
category_counts = food['Category'].value_counts()
```

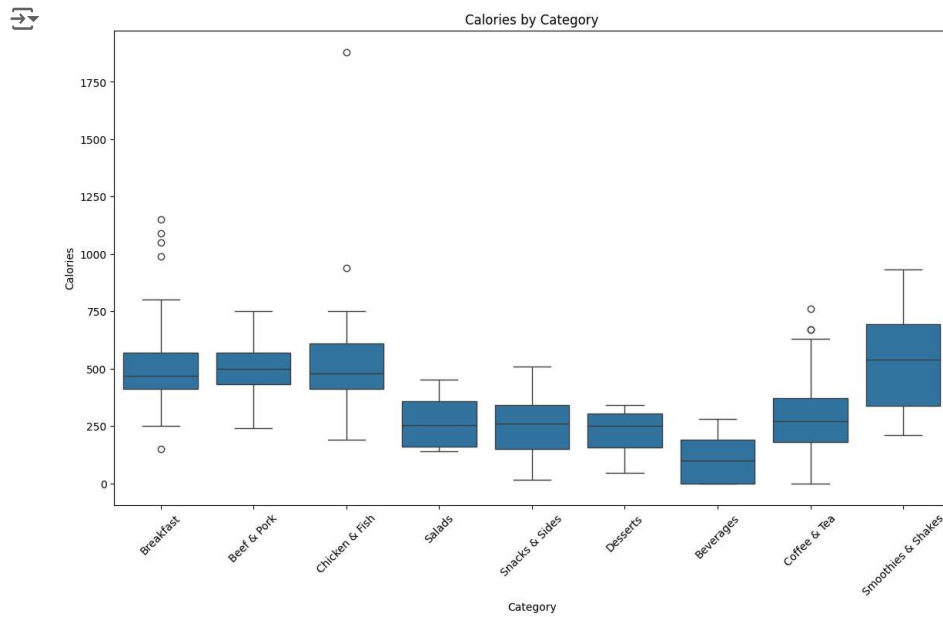
```
plt.figure(figsize=(8, 8))
plt.pie(category_counts, labels=category_counts.index, autopct='%1.1f%%', colors=plt.cm.Paired.colors)
plt.title('Proportion of Different Categories')
plt.show()
```



Proportion of Different Categories



```
plt.figure(figsize=(14, 8))
sns.boxplot(x='Category', y='Calories', data=food)
plt.xticks(rotation=45)
plt.title('Calories by Category')
plt.xlabel('Category')
plt.ylabel('Calories')
plt.show()
```



```
food['Calories'].idxmax()
```

```
82
```

```
food.at[82, 'Item']
```

```
'Chicken McNuggets (40 piece)'
```

```
food['Protein'].idxmax()
```

```
82
```

```
food['Cholesterol'].idxmax()
```

```
31
```

```
food.at[31, 'Item']
```

```
'Big Breakfast with Hotcakes (Regular Biscuit)'
```

```
food[['Saturated Fat (% Daily Value)', 'Saturated Fat']].idxmax()
```

```

↳ Saturated Fat (% Daily Value)    253
   Saturated Fat                    32
   dtype: int64

```

```

value_1 = food.iloc[253, 1]
value_2 = food.iloc[32, 1]
print(value_1, " ", value_2)

```

```

↳ McFlurry with M&M's Candies (Medium)    Big Breakfast with Hotcakes (Large Biscuit)

```

```

item_with_highest_calories = food.loc[food.groupby(['Category'])['Calories'].idxmax()]
item_with_highest_calories_sorted = item_with_highest_calories.sort_values(by='Calories', ascending=False)[['Category', 'Item', 'Calories']]
print(item_with_highest_calories_sorted)

```

```

↳
      Category                                     Item  Calories
82    Chicken & Fish          Chicken McNuggets (40 piece)    1880
32    Breakfast    Big Breakfast with Hotcakes (Large Biscuit)    1150
253  Smoothies & Shakes    McFlurry with M&M's Candies (Medium)    930
231    Coffee & Tea          Frappé Chocolate Chip (Large)    760
47    Beef & Pork          Double Quarter Pounder with Cheese    750
98    Snacks & Sides          Large French Fries    510
88    Salads    Premium Southwest Salad with Crispy Chicken    450
108    Desserts          Hot Caramel Sundae    340
112    Beverages          Coca-Cola Classic (Large)    280

```

```

item_with_lowest_calories = food.loc[food.groupby(['Category'])['Calories'].idxmin()]
item_with_lowest_calories_sorted = item_with_lowest_calories.sort_values(by='Calories', ascending=False)[['Category', 'Item', 'Calories']]
print(item_with_lowest_calories_sorted)

```

```

↳
      Category                                     Item  Calories
48    Beef & Pork          Hamburger    240
235  Smoothies & Shakes    Strawberry Banana Smoothie (Small)    210
78    Chicken & Fish          Chicken McNuggets (4 piece)    190
38    Breakfast          Hash Brown    150
84    Salads    Premium Bacon Ranch Salad (without Chicken)    140
106    Desserts          Kids Ice Cream Cone    45
101    Snacks & Sides          Apple Slices    15
114    Beverages          Diet Coke (Small)    0
137    Coffee & Tea          Iced Tea (Small)    0

```

```

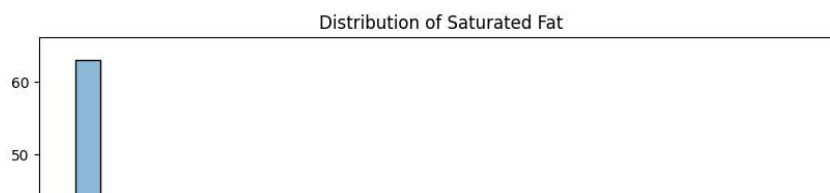
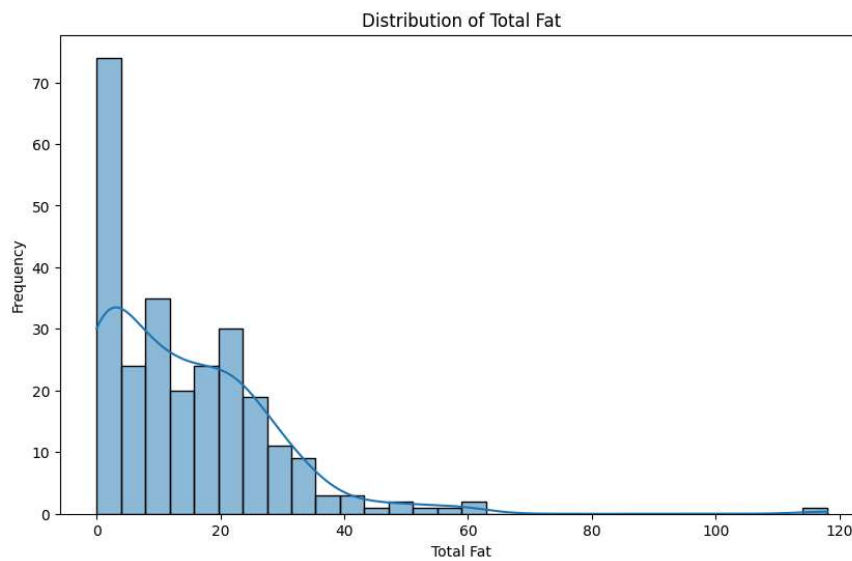
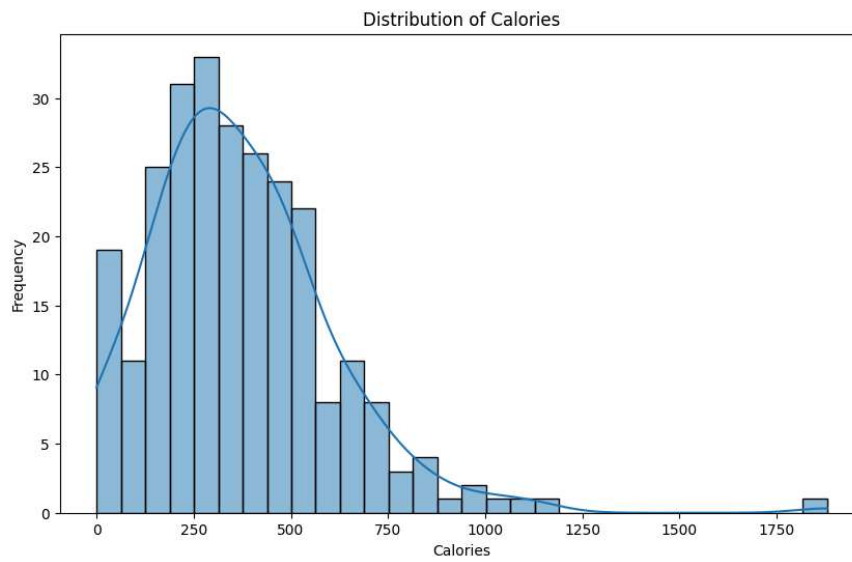
def plot_histogram(column, bins=30):
    plt.figure(figsize=(10, 6))
    sns.histplot(food[column], bins=bins, kde=True)
    plt.title(f'Distribution of {column}')
    plt.xlabel(column)
    plt.ylabel('Frequency')
    plt.show()

```

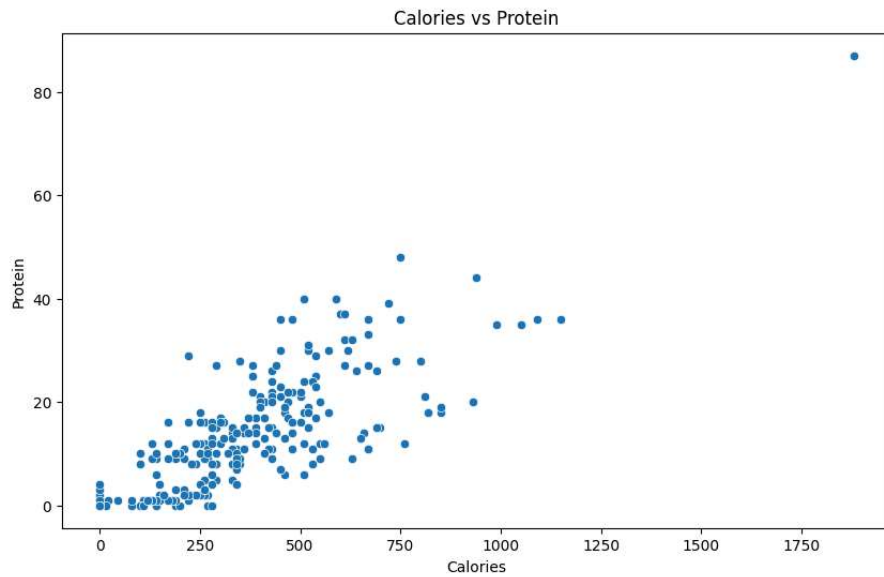
```

columns_to_plot = ['Calories', 'Total Fat', 'Saturated Fat', 'Cholesterol', 'Sodium', 'Carbohydrates', 'Sugars', 'Protein']
for column in columns_to_plot:
    plot_histogram(column)

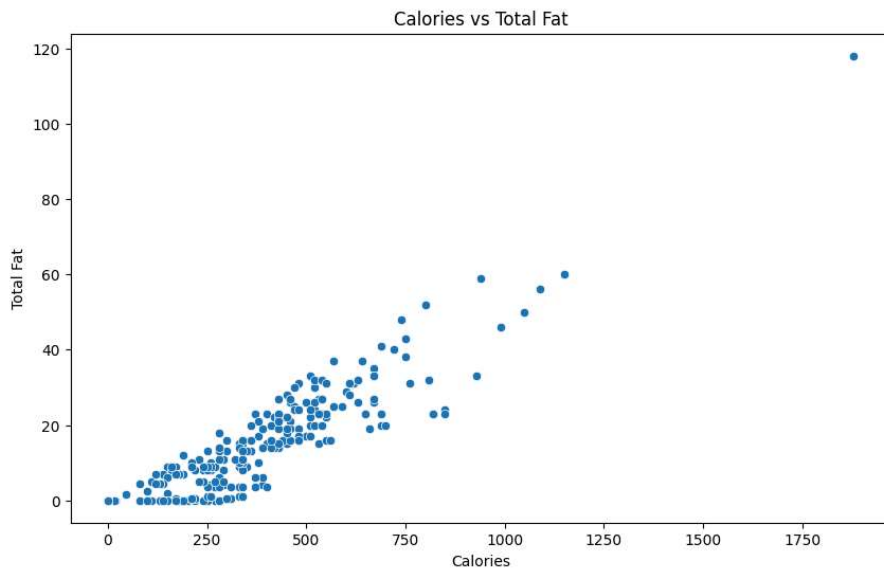
```



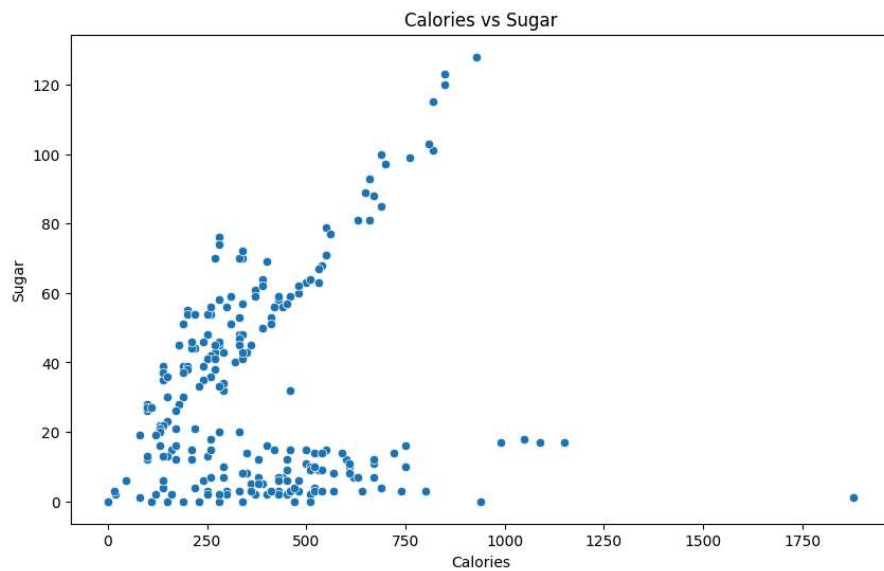
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=food['Calories'], y=food['Protein'])
plt.title('Calories vs Protein')
plt.xlabel('Calories')
plt.ylabel('Protein')
plt.show()
```



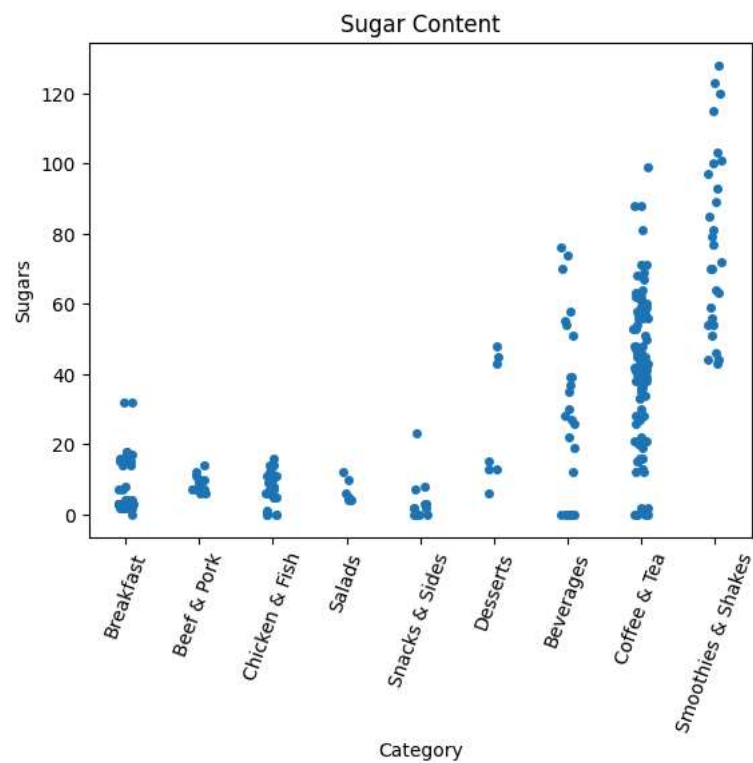
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=food['Calories'], y=food['Total Fat'])
plt.title('Calories vs Total Fat')
plt.xlabel('Calories')
plt.ylabel('Total Fat')
plt.show()
```



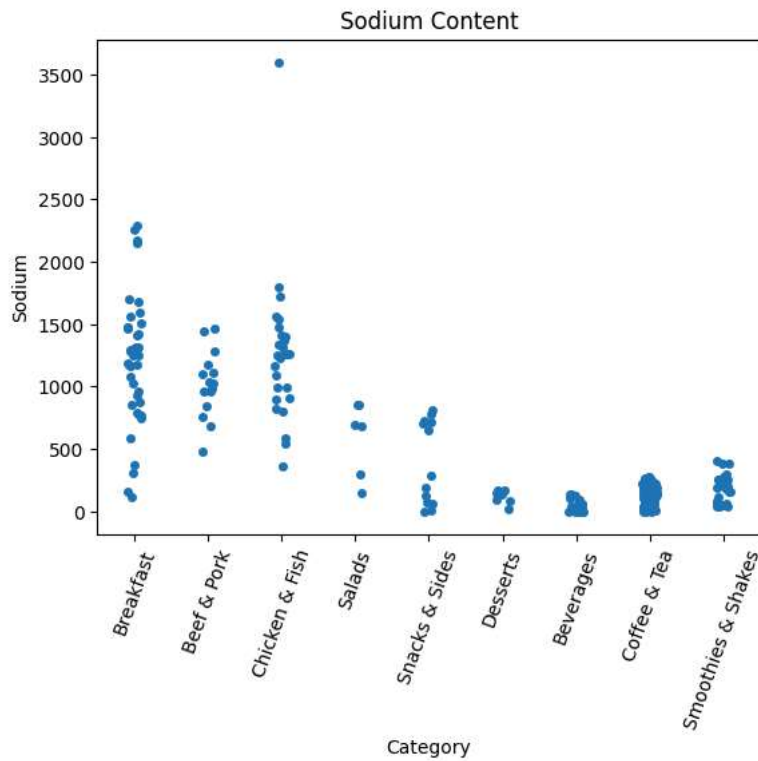
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=food['Calories'], y=food['Sugars'])
plt.title('Calories vs Sugar')
plt.xlabel('Calories')
plt.ylabel('Sugar')
plt.show()
```



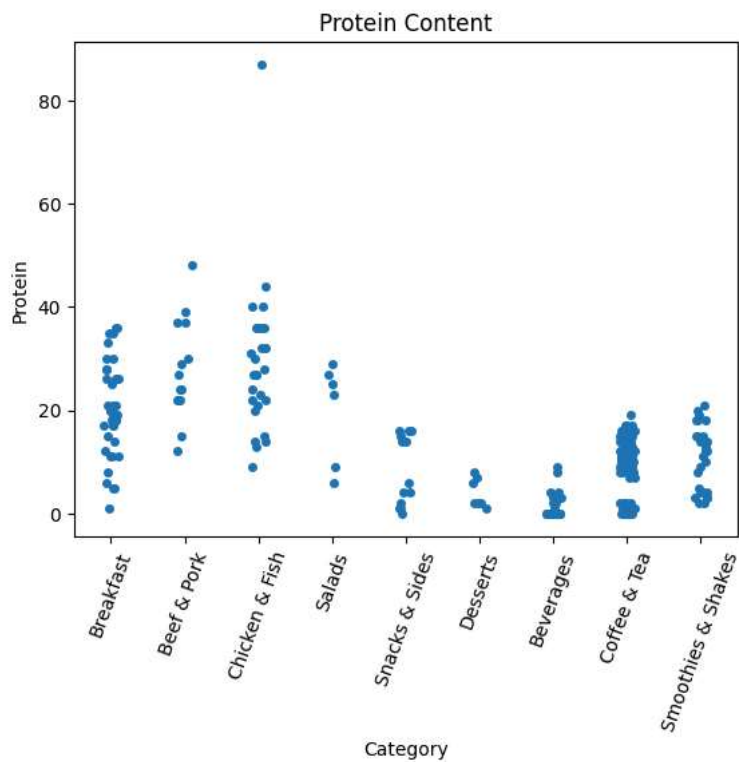
```
plot=sns.stripplot(x="Category", y='Sugars', data=food)
plt.setp(plot.get_xticklabels(), rotation=70)
plt.title('Sugar Content')
plt.show()
```



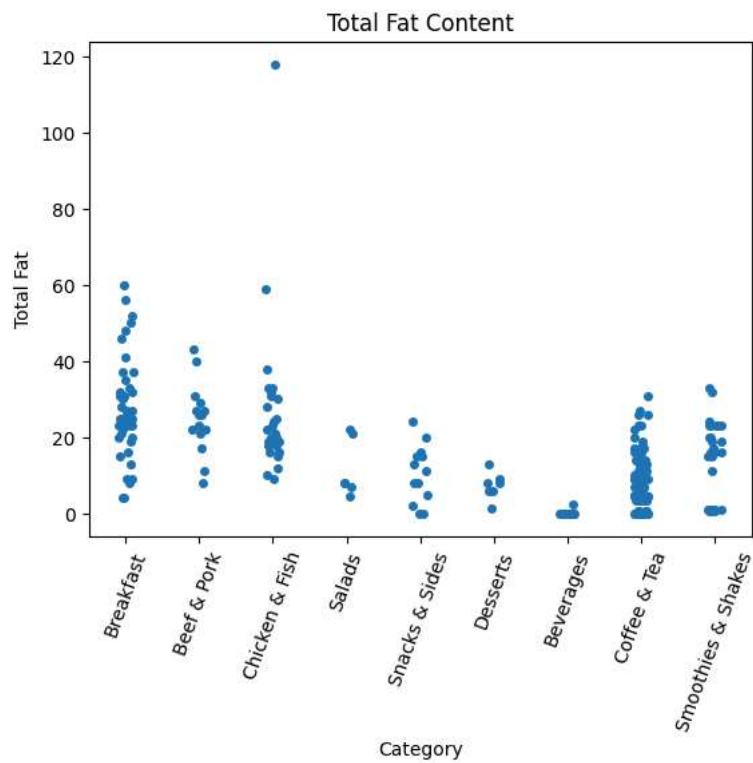
```
plot=sns.stripplot(x="Category", y='Sodium', data=food)
plt.setp(plot.get_xticklabels(), rotation=70)
plt.title('Sodium Content')
plt.show()
```

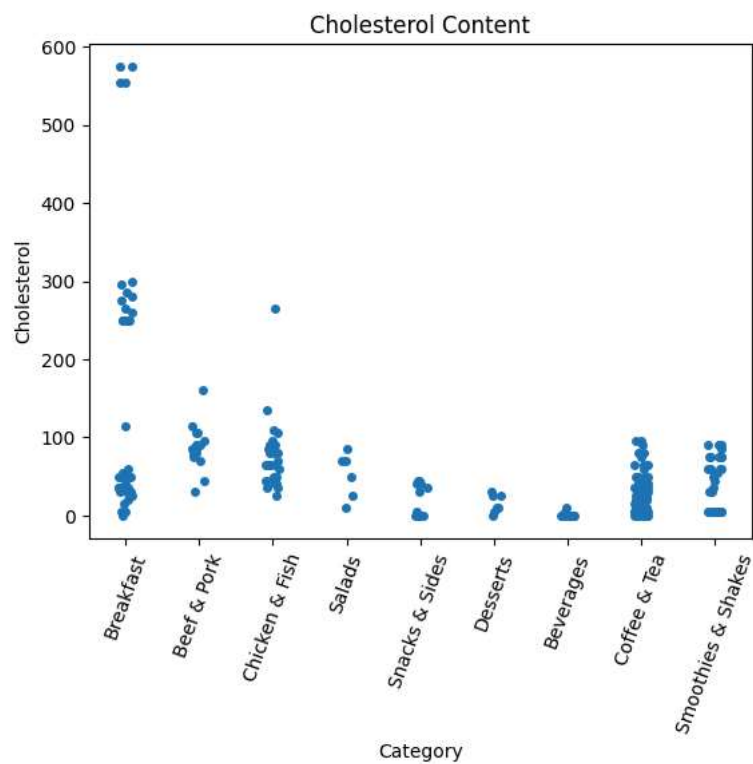
```
plot=sns.stripplot(x="Category", y='Protein', data=food)
plt.setp(plot.get_xticklabels(), rotation=70)
plt.title('Protein Content')
plt.show()
```



```
plot=sns.stripplot(x="Category", y='Total Fat', data=food)
plt.setp(plot.get_xticklabels(), rotation=70)
plt.title('Total Fat Content')
plt.show()
```

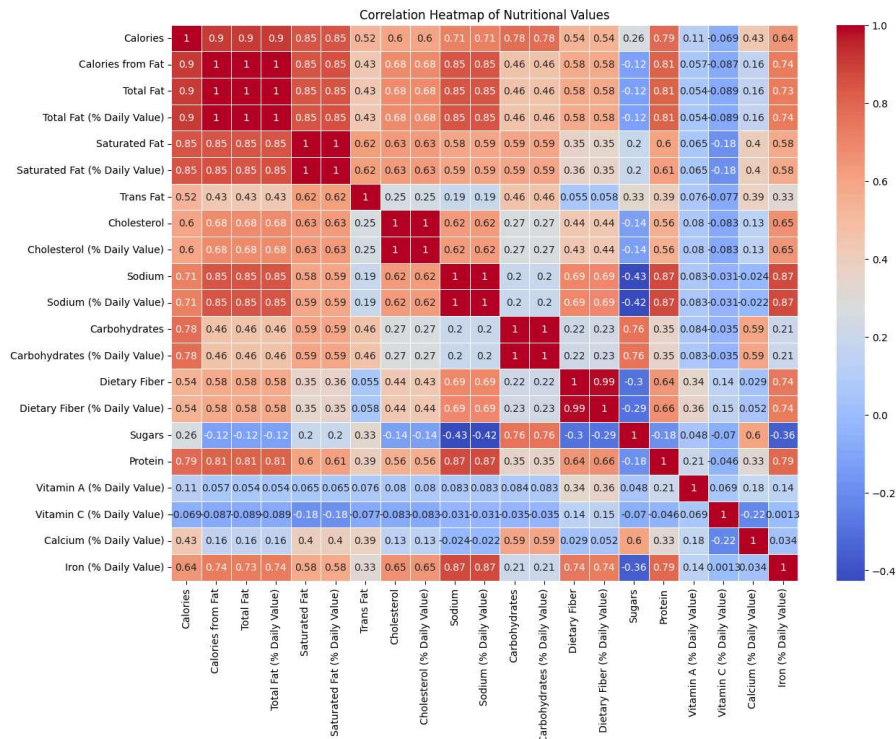


```
plot=sns.stripplot(x="Category", y='Cholesterol', data=food)
plt.setp(plot.get_xticklabels(), rotation=70)
plt.title('Cholesterol Content')
plt.show()
```



```
# Select only numeric columns for the correlation matrix
numeric_data = food.select_dtypes(include=['float64', 'int64'])

# Correlation Heatmap
plt.figure(figsize=(14, 10))
correlation_matrix = numeric_data.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap of Nutritional Values')
plt.show()
```



```
sorted_items_by_calories = food.sort_values(by=['Category', 'Calories'], ascending=[True, False]) #Sort by Calories
```

```
# Print the sorted items by category
for category, items in sorted_items_by_calories.groupby('Category'):
    pd.set_option('display.max_rows', None) # Show all rows
    pd.set_option('display.max_columns', None) # Show all columns
    pd.set_option('display.width', None) #Auto-adjust width
    print(f"Category: {category}")
    print(items[['Item', 'Calories']])
    print("\n")

# Create a bar plot to show the top items with the highest calories in each category
plt.figure(figsize=(12, 6))
top_items = sorted_items_by_calories.groupby('Category').head(1) # Select top item in each category
plt.bar(top_items['Category'], top_items['Calories'], color='skyblue')
plt.xlabel('Category')
plt.ylabel('Calories')
plt.title('Top Item with Highest Calories in Each Category')
plt.xticks(rotation=45)
plt.show()
```

```
Category: Beef & Pork
```

	Item	Calories
47	Double Quarter Pounder with Cheese	750
51	Bacon Clubhouse Burger	720
45	Quarter Pounder with Bacon Habanero Ranch	610
44	Quarter Pounder with Bacon & Cheese	600
46	Quarter Pounder Deluxe	540
42	Big Mac	530
43	Quarter Pounder with Cheese	520
56	McRib	500
53	Bacon McDouble	440
50	Double Cheeseburger	430
54	Daily Double	430
55	Jalapeño Double	430
52	McDouble	380
49	Cheeseburger	290
48	Hamburger	240

```
Category: Beverages
```

	Item	Calories
112	Coca-Cola Classic (Large)	280
128	Sprite (Large)	280
135	Minute Maid Orange Juice (Large)	280
120	Dr Pepper (Large)	270
111	Coca-Cola Classic (Medium)	200
127	Sprite (Medium)	200
119	Dr Pepper (Medium)	190
134	Minute Maid Orange Juice (Medium)	190
133	Minute Maid Orange Juice (Small)	150
110	Coca-Cola Classic (Small)	140
118	Dr Pepper (Small)	140
126	Sprite (Small)	140
131	Fat Free Chocolate Milk Jug	130
113	Coca-Cola Classic (Child)	100
121	Dr Pepper (Child)	100
129	Sprite (Child)	100
130	1% Low Fat Milk Jug	100
132	Minute Maid 100% Apple Juice Box	80
114	Diet Coke (Small)	0
115	Diet Coke (Medium)	0
116	Diet Coke (Large)	0
117	Diet Coke (Child)	0

```
# Create a pie chart to show the distribution of calories among different categories
plt.figure(figsize=(8, 8))
```