Finding Healthy Fast Foods and Restaurants:

Project By: Ali Motamednejad
Date: Apr 2023

Project Description:

The target is to get the healthiest among a list of food served in some restaurants. It has always been my concern to eat healthy meals and getting good amount of energy.
The results of this project can also be useful for the people who are interested in healthy foods and finding the restaurants which serve healthy meals.

Definition of Healthy food:

Generally, healthy foods include less salt, sugar and fat. On the other hand,they have more vitamin, calcium, protein and fiber.
They also leave more feeling of full.

Libraries:

In [1]:
```
#  For Analysis
import numpy as np
import pandas as pd
import math
import re

#  For Visualization
import plotly.express as px
import plotly.subplots as sp
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.graph_objects as go
from matplotlib.ticker import FixedLocator
import matplotlib.colors as mcolors

#  For Reporting
from IPython.display import display, Markdown, HTML
from PIL import Image
```

Loading Data

In [2]:
```
df_food = pd.read_csv('C:/One Drive/OneDrive/Desktop/Data Science/Hamideh/1/data_fastfood_calories.csv')
df_sale = pd.read_csv('C:/One Drive/OneDrive/Desktop/Data Science/Hamideh/1/data_fastfood_sales.csv')
```

The initial Observation of the Data:

Fast Food Data:

In [3]:
```
df_food.sample(10)
```

Out[3]:

	restaurant	item	calories	cal_fat	total_fat	sat_fat	trans_fat	cholesterol	sodium	total_carb	fiber	sugar	protein	vit_a	vit_c	calcium
479	Taco Bell	Triple Layer Nachos	320	140	15	1.5	0.0	0	600	41	6.0	2	7.0	NaN	NaN	NaN
158	Arbys	Loaded Italian Sandwich	680	360	40	14.0	0.5	100	2270	49	3.0	7	32.0	NaN	NaN	NaN
9	Mcdonalds	Double Quarter Pounder® with Cheese	770	400	45	21.0	2.5	175	1290	42	3.0	10	51.0	20.0	6.0	20.0
225	Burger King	Chicken BLT Salad w/ Crispy Chicken	690	430	48	12.0	1.0	100	1750	31	4.0	8	35.0	NaN	NaN	NaN
428	Taco Bell	Beef Spicy Cheesy Core Burrito	570	210	23	10.0	0.5	45	1760	68	7.0	5	22.0	15.0	6.0	35.0
430	Taco Bell	XXL Grilled Stuft Burrito - Beef	880	380	42	14.0	1.0	75	2020	94	12.0	6	31.0	NaN	NaN	NaN
156	Arbys	Half Pound Roast Beef Sandwich	610	270	30	12.0	2.0	130	2040	38	2.0	5	48.0	NaN	NaN	NaN
183	Arbys	Jalapeno Roast Beef 'n Cheese Slider	240	90	11	4.5	0.0	30	670	21	1.0	1	14.0	NaN	NaN	NaN
259	Burger King	Spicy Crispy Chicken Jr.	410	220	25	4.5	0.0	35	850	35	2.0	5	12.0	NaN	NaN	NaN
110	Sonic	Chicken Strip Sandwich	450	220	24	4.0	0.0	35	740	43	1.0	4	19.0	0.0	0.0	4.0

Sale Data:

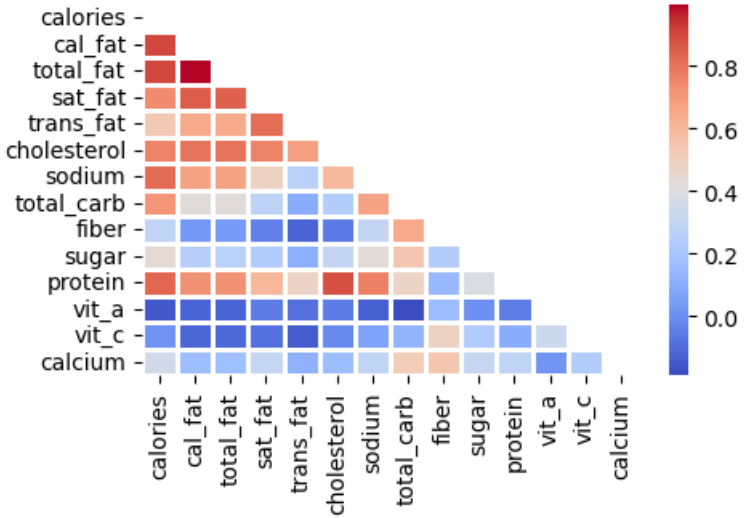
In [4]:
```
title = "<h2 style='text-align:left;'><b><u><font color='purple'>Sample of Sale Dataframe</font></u></b></h2>"
centered_title = title.center(80)  # Adjust the width as needed

display(HTML(centered_title))
display(df_sale.sample(10))
```

Sample of Sale Dataframe

	restaurant	average_sales	us_sales	num_company_stores	num_franchised_stores	unit_count
12	Arbys	1130.00	3634.00	1075	2340	3415
5	Burger King	1387.81	10028.32	50	7196	7266
15	Baskin-Robbins	360.72	606.00	0	2560	2560
9	Little Caesars	815.00	3530.58	535	3797	4332
8	Dominos	1000.00	5900.00	392	5195	5587
16	Chipotle	1940.00	4476.41	2371	0	2371
2	Starbucks	945.27	13167.61	8222	5708	13930
1	Mcdonalds	2670.32	37480.67	842	13194	14036
11	Sonic	1250.00	4408.16	228	3365	3593
6	Taco Bell	1500.00	9790.15	647	5799	6446

The investigation of the relationship between different variables:

In [5]:
```
plt.figure(figsize=(5, 3))
mask = np.triu(np.ones_like(df_food.corr(numeric_only = True), dtype=bool))
sns.heatmap(df_food.corr(numeric_only = True),lw=1,cmap="coolwarm",mask=mask)

plt.show()
```



Based on the heatmap:

- It uses colors to represent the values of the variables. The warmer colors indicate higher values and cooler colors indicate lower values.

- Each row and column corresponds to a specific food compounds in the dataset.
- By analyzing the heatmap, patterns and correlations could be identified between the variables. For example, there is a strong positive correlation between calorie and fat content, it means the more value of fat in the meals, the more calories one can obtain.

Findings:

1- There are no considerable relationship between Vitamins, Calcium, Protein, Fiber and Calorie

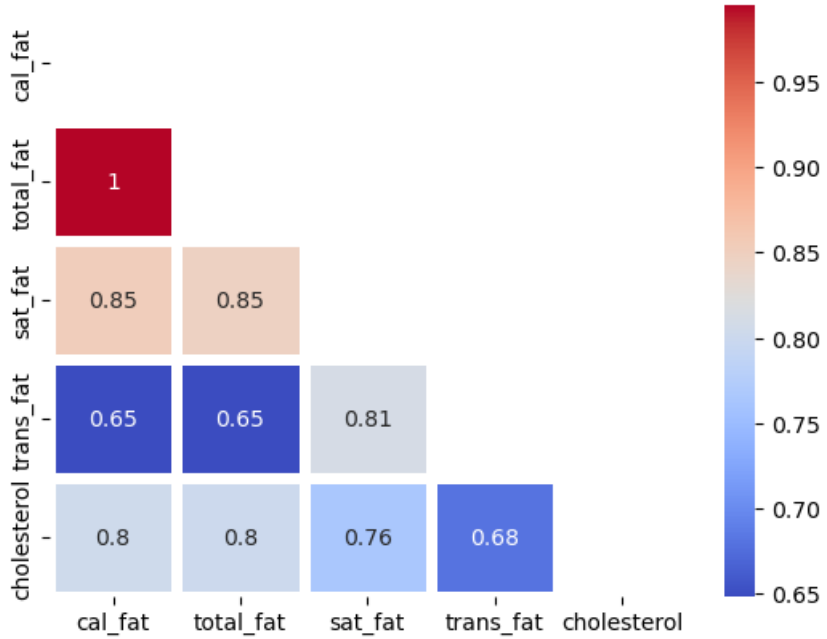
2- There are strong positive correlations between cholesterol, fat, sodium, and calorie. It means the higher values of cholesterol in a meal, the higher values of fat, sodium, and calorie.

For example "American Brewhouse King" has high values of cal_fat, cholestrol, sodium and calorie and negligible amount of vitamin, calcium and fiber.However, "Grilled Chicken Garden Greens Salad" has low amount of Cholesterol, fat and Calorie.

Optimizing the Variables:

Regarding many variables in this dataset, it is better to study on possibility of optimizing them.
On the first step, the relationship of "cal_fat", "total_fat", "sat_fat", "trans_fat" and "cholestrol" is investigated:

In [6]:
```
df_fat = df_food[['cal_fat','total_fat','sat_fat','trans_fat','cholesterol']]

mask = np.triu(np.ones_like(df_fat.corr(), dtype=bool))

sns.heatmap(df_fat.corr(numeric_only = True),lw=5,cmap="coolwarm",annot=True,mask=mask)

plt.show()
```



Findings:

Cholestrol will be considered as a representative variable among other kind of fats, because of strong correlation.

Removing foods which are for more than a person (Meals contain 5 pieces and more)

In [7]:
```
df_food1 = df_food[df_food['item'].str.contains('[5-9]\s*[P p]iece|[1-9]\d+\s*[P p]iece') == False]
print("There are",str(len(df_food1.index)),"foods for one person in the dataframe")
```

```
There are 491 foods for one person in the dataframe
```

In [8]:
```
df_food1.sample(10)
```

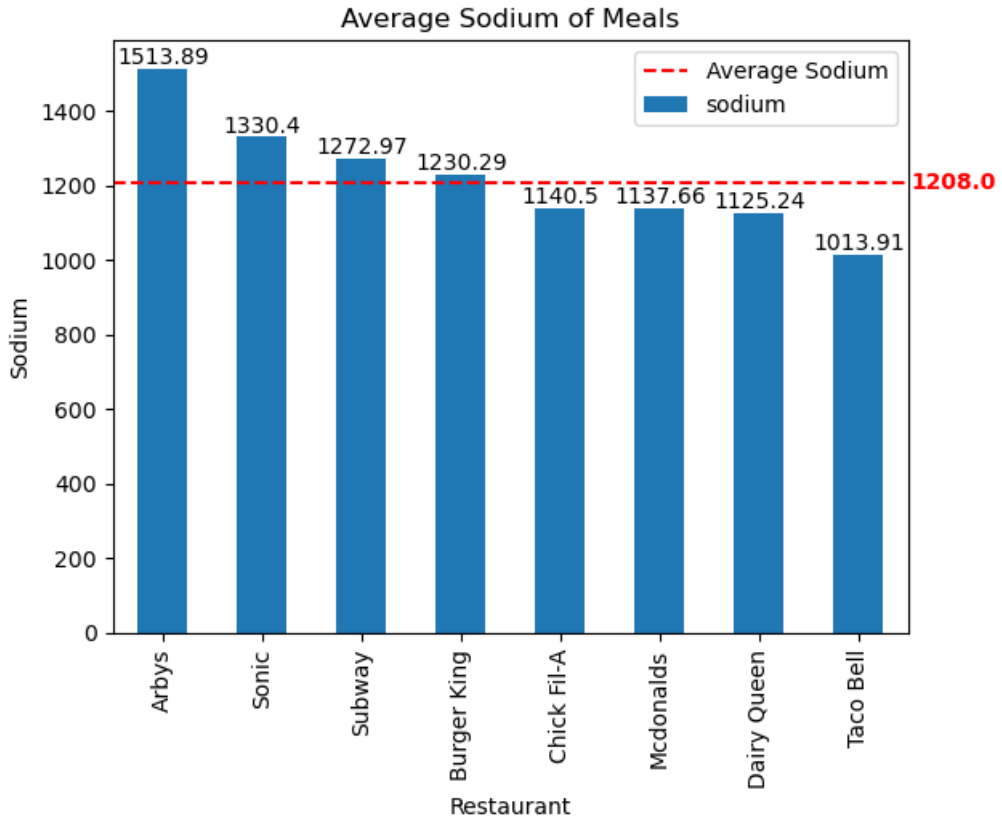
Out[8]:

	restaurant	item	calories	cal_fat	total_fat	sat_fat	trans_fat	cholesterol	sodium	total_carb	fiber	sugar	protein	vit_a	vit_c	calcium
123	Sonic	3 Piece Super Crunch Chicken Strips	330	140	16	3.0	0.0	55	670	25	2.0	0	22.0	1.0	2.0	1.0
335	Subway	Kids Mini Sub Roast Beef	200	25	3	1.0	0.0	25	390	30	4.0	5	14.0	6.0	15.0	20.0
306	Subway	6" BBQ Rib Sandwich	430	160	18	6.0	0.0	50	590	47	5.0	8	19.0	8.0	20.0	30.0
286	Dairy Queen	Deluxe Double Hamburger	540	240	26	13.0	1.0	100	750	34	1.0	9	29.0	15.0	6.0	4.0
489	Taco Bell	Cheese Roll-Up	190	80	9	5.0	0.0	20	450	18	2.0	1	9.0	NaN	NaN	NaN
412	Taco Bell	Cantina Power Burrito - Veggie	740	230	26	5.0	0.0	10	1750	107	17.0	8	20.0	NaN	NaN	NaN
511	Taco Bell	Express Taco Salad w/ Chips	580	260	29	9.0	1.0	60	1270	59	8.0	7	23.0	NaN	NaN	NaN
435	Taco Bell	Cool Ranch® Doritos® Locos Taco	160	90	10	3.5	0.0	25	350	13	2.0	1	8.0	NaN	NaN	NaN
313	Subway	6" Black Forest Ham	290	40	5	1.0	0.0	20	830	46	5.0	8	18.0	8.0	20.0	30.0
459	Taco Bell	Mild Naked Chicken Chalupa	440	270	30	7.0	0.0	70	1090	22	3.0	1	20.0	6.0	4.0	6.0

Comparison of different restaurants based on sodium in the meals:

In [9]:

```
average_sodium = df_food1['sodium'].mean()
ax = df_food1.groupby(['restaurant'])['sodium'].mean().sort_values(ascending=False).plot(kind='bar')
ax.set_ylabel('Sodium')
ax.set_xlabel('Restaurant')
ax.bar_label(ax.containers[0], label_type='edge')
ax.set_title('Average Sodium of Meals')

# Add a horizontal line for the average sodium
ax.axhline(y=average_sodium, color='red', linestyle='--', label='Average Sodium')

# Add the value of average sodium on the vertical axis
ax.text(ax.get_xlim()[1] + 0.02, average_sodium, f'{average_sodium:.1f}', ha='left', va='center', color='red', fontweight='bold')# Show the legend
ax.legend()

plt.show()
```



In [10]:

```
# Create the pivot table
pivot_table = df_food1.pivot_table(values=['cholesterol', 'sodium', 'calories', 'sugar'],
                                   index='restaurant',
                                   aggfunc={'cholesterol': 'mean', 'sodium': 'median', 'calories': 'mean', 'sugar': 'mean'})

pivot_table = pivot_table.rename(columns={'calories': 'Average<br>Calories', 'sodium': 'Average<br>Sodium', 'cholesterol': 'Average<br>Cholesterol', 'sugar': 'Averag

# Define a function to apply the desired color formatting
def color_formatting(val, col_avg):
    if val > col_avg:
        return 'color: red'
    else:
        return 'color: blue'

# Formatting and decoration
formatted_pivot_table = pivot_table.style \
    .format({'Average<br>Calories': '{:.1f}', 'Average<br>Sodium': '{:.1f}', 'Average<br>Cholesterol': '{:.1f}', 'Average<br>Sugar': '{:.1f}'}) \
    .apply(lambda col: [color_formatting(val, col.mean()) for val in col], subset=pd.IndexSlice[:, ['Average<br>Calories', 'Average<br>Sodium', 'Average<br>Cholester

formatted_pivot_table = formatted_pivot_table.set_properties(**{'font-size': '10pt', 'text-align': 'center'}) \
    .set_table_styles([{'selector': 'th', 'props': [('font-size', '10pt')]}]) \
```

```
    .set_caption('Pivot Table - The Average Compounds of Meals')

# Add the note
formatted_pivot_table = formatted_pivot_table.set_caption('Pivot Table - The Average Compounds of Meals<br><br>Note: Red values are greater than the average of each

# Display the formatted pivot table
formatted_pivot_table
```

Out[10]:

Pivot Table - The Average Compounds of Meals

Note: Red values are greater than the average of each column and blue ones are smaller

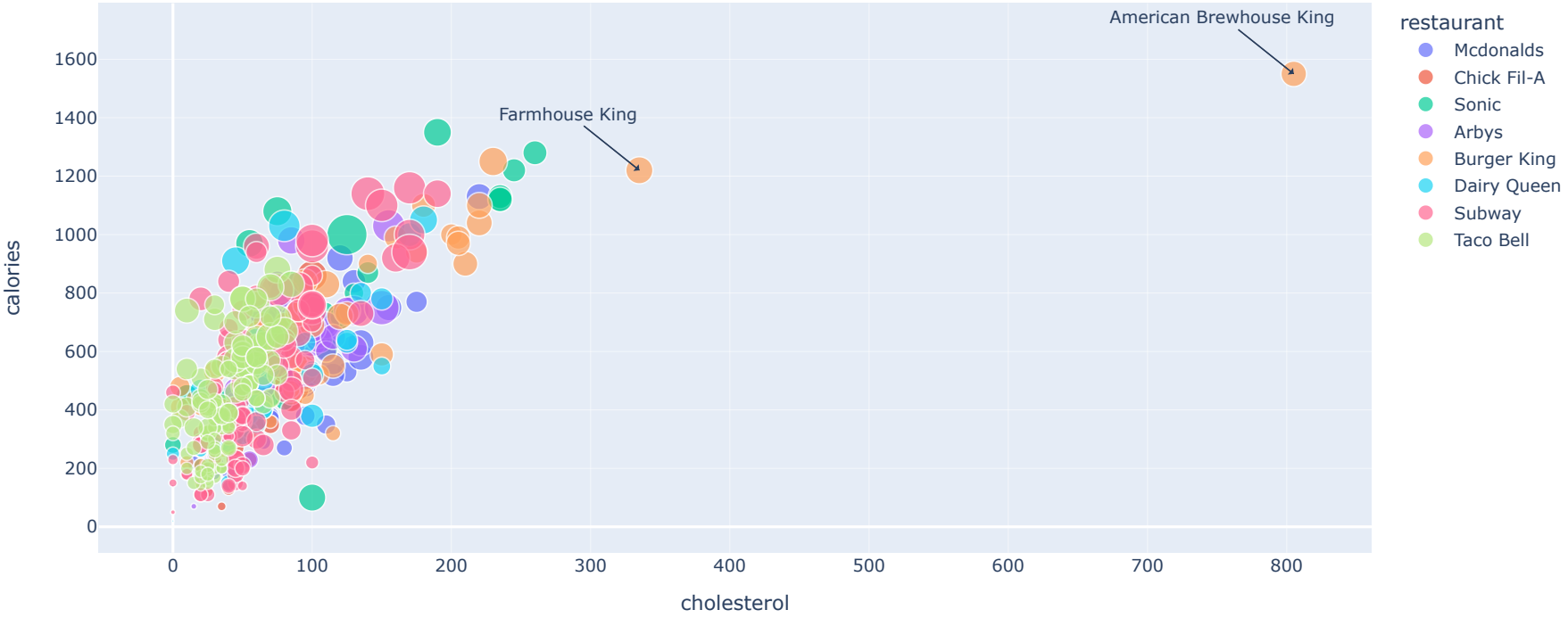
restaurant	Average Calories	Average Cholesterol	Average Sodium	Average Sugar
Arbys	531.5	70.4	1470.0	7.7
Burger King	608.2	101.3	1150.0	8.4
Chick Fil-A	405.5	69.5	1125.0	5.4
Dairy Queen	502.2	70.4	1020.0	6.4
Mcdonalds	524.7	87.6	1090.0	10.2
Sonic	625.4	88.6	1220.0	6.7
Subway	503.0	61.3	1130.0	10.1
Taco Bell	443.7	39.0	960.0	3.7

Scatter Diagram Based on Calories, Cholestrol and Sodium:

In [11]:

```
fig = px.scatter(df_food1, 'cholesterol', 'calories',  color='restaurant', size='sodium', hover_data='item')

# calculate quartiles and interquartile range
q1_x, q3_x = df_food1['cholesterol'].quantile([0.1, 0.9])
iqr_x = q3_x - q1_x
q1_y, q3_y = df_food1['calories'].quantile([0.25, 0.75])
iqr_y = q3_y - q1_y

# identify outliers
x_outliers = (df_food1['cholesterol'] < q1_x - 1.5*iqr_x) | (df_food1['cholesterol'] > q3_x + 1.5*iqr_x)
y_outliers =    (df_food1['calories'] < q1_y - 1.5*iqr_y) | (df_food1['calories']    > q3_y + 1.5*iqr_y)
xy_outliers = df_food1[x_outliers & y_outliers]

# add annotations for outliers
for index, row in xy_outliers.iterrows():
    fig.add_annotation(
        x=row['cholesterol'], y=row['calories'], text=row['item'],
        showarrow=True, arrowhead=1, ax=-50, ay=-40
```

```
    )

fig.show()
```

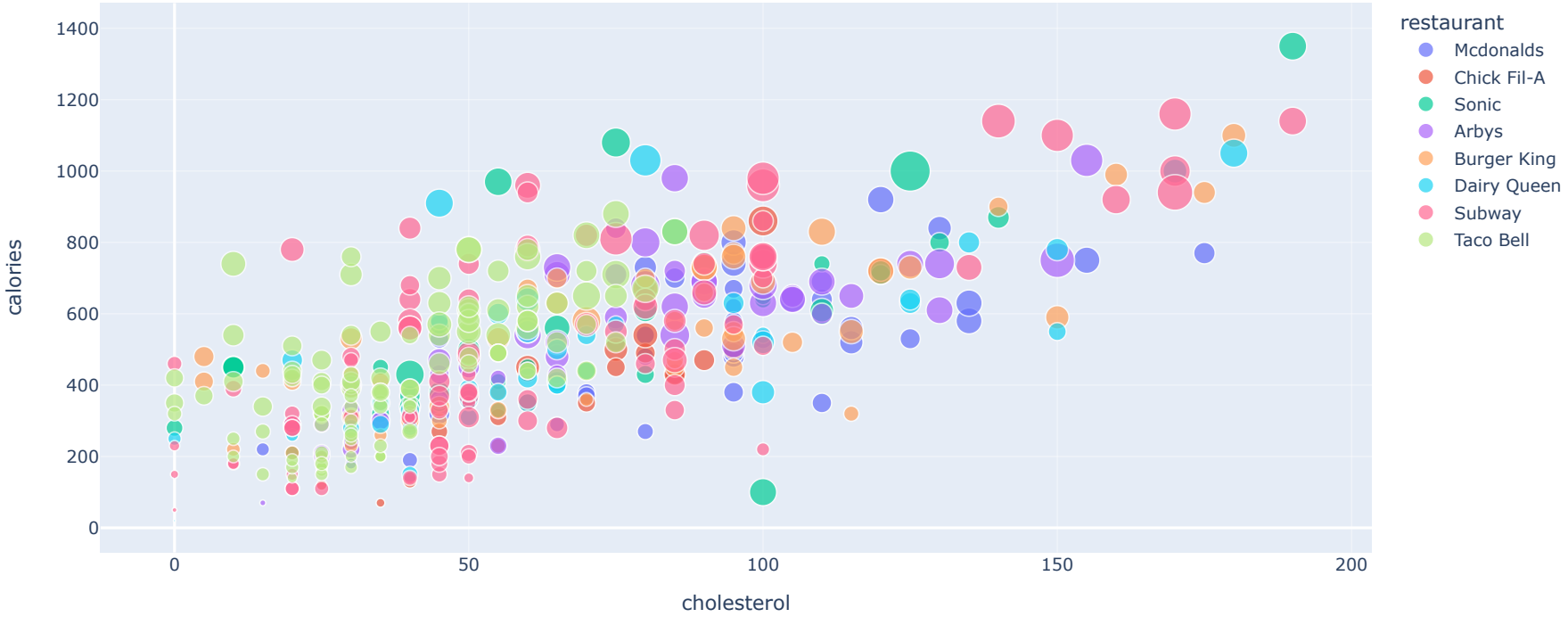

American Brewhouse King
Farmhouse King
restaurant
Mcdonalds
Chick Fil-A
Sonic
Arbys
Burger King
Dairy Queen
Subway
Taco Bell
calories
cholesterol
0
200
400
600
800
1000
1200
1400
1600
0
100
200
300
400
500
600
700
800


American
BREWHOUSE
KING
BURGER
KING
Limited time only at participating restaurants.


FARM
HOUSE
KING
BURGER
KING
Limited time only at participating restaurants.

Removing Outliers From Previous Scatter Diagram:

In [12]:
```
fig = px.scatter(df_food1[(df_food1['cholesterol'] < 200) & (df_food1['calories'] < 1400)], 'cholesterol', 'calories',  color='restaurant', size='sodium', hover_data
fig.show()
```



Assumptions:

1- The foods with cholesterol<300 and calorie>1000 are considered as low fat and energetic meals

2- The foods containing more than 2,300 mg sodium is considered as non-healthy meals because adults limit sodium intake is less than 2,300 mg per day.

3- The criteria for healthy and energetic food will be as follow:
sodium < 2300
cholesterol < 300
calorie > 1000

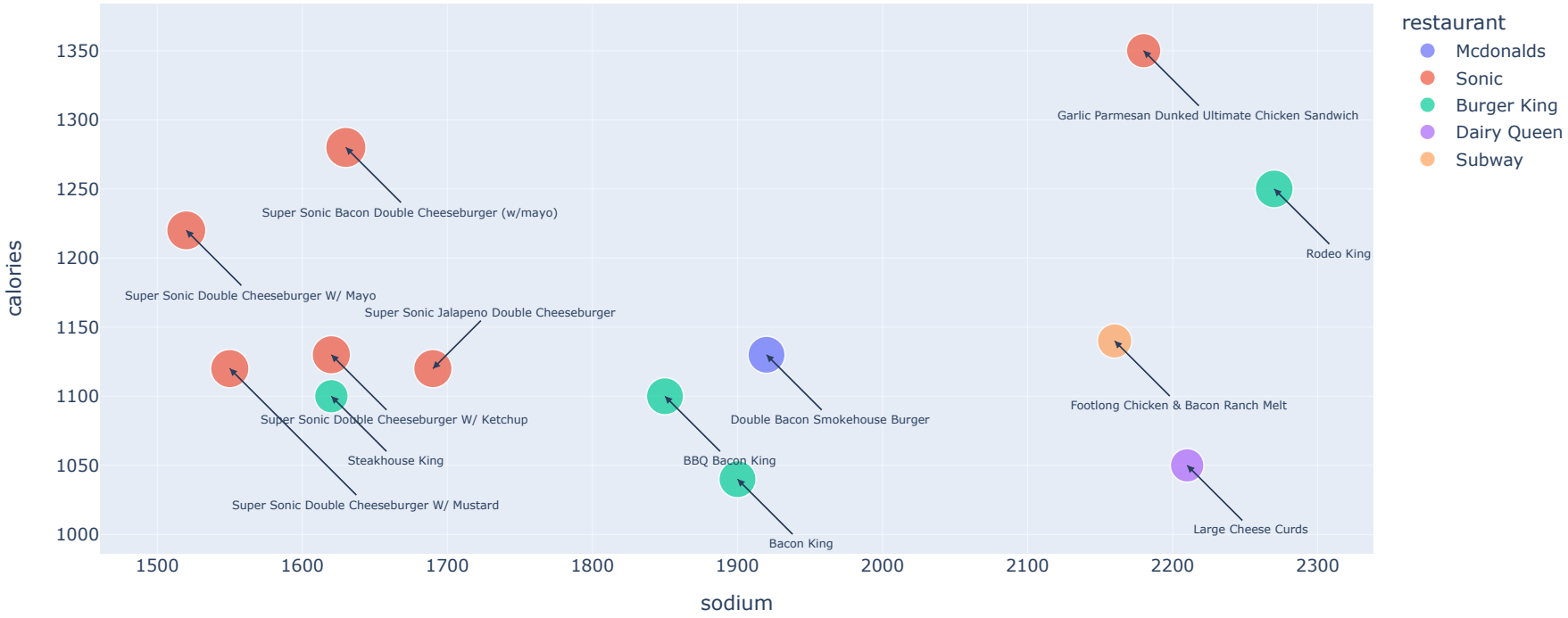
The number of healthy and energetic foods in data set:

In [13]:
```
# https://kanoki.org/2020/01/21/pandas-dataframe-filter-with-multiple-conditions/
filtered = df_food1.query('calories > 1000 & cholesterol<300 & sodium<2300')
len(filtered. index)

# df_unhealthy

print(str(len(filtered. index)) + " Meals out of " + str(len(df_food1. index)) + " Meals are Considered Healthy" + " ("+ str(round(len(filtered. index)/len(df_food1.
# darsad
```

```
13 Meals out of 491 Meals are Considered Healthy (3.0%)
```

Healthy foods:

In [14]:
```
fig = px.scatter(filtered, 'sodium', 'calories',  color='restaurant', size='cholesterol', hover_data='item')

for i, row in filtered.iterrows():

    if row['item']=='Super Sonic Double Cheeseburger W/ Mustard':
        fig.add_annotation(x=row['sodium'], y=row['calories'], text=row['item'], showarrow=True,
                           arrowhead=2, ax=95, ay=95, font=dict(size=8))
    elif row['item']=='Super Sonic Jalapeno Double Cheeseburger':
        fig.add_annotation(x=row['sodium'], y=row['calories'], text=row['item'], showarrow=True,
                           arrowhead=2, ax=40, ay=-40, font=dict(size=8))
    else:
        fig.add_annotation(x=row['sodium'], y=row['calories'], text=row['item'], showarrow=True,
                           arrowhead=2, ax=45, ay=45, font=dict(size=8))

fig.show()
```



Salads:

In [15]:
```
df_salad = df_food[df_food['item'].str.contains("salad", flags=re.IGNORECASE) == True]
print("There are",len(df_salad. index),"salads in the dataset")
df_salad.sample(10)
```

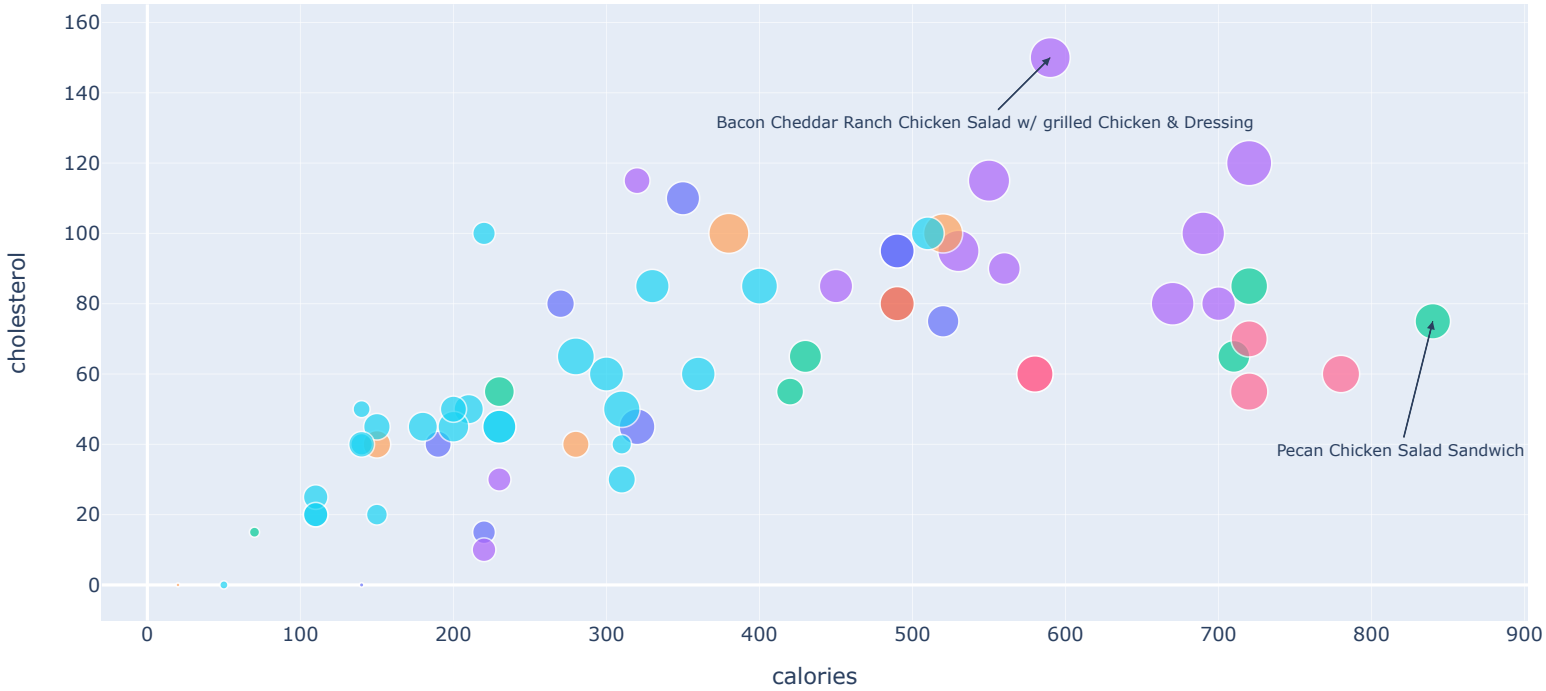
```
There are 64 salads in the dataset
```

Out[15]:

	restaurant	item	calories	cal_fat	total_fat	sat_fat	trans_fat	cholesterol	sodium	total_carb	fiber	sugar	protein	vit_a	vit_c	calcium
191	Arbys	Super Greek Salad	720	480	53	15.0	0.0	85	1310	39	5.0	7	22.0	NaN	NaN	NaN
298	Dairy Queen	Grilled Chicken BLT Salad	380	170	19	9.0	0.0	100	1540	11	3.0	6	42.0	NaN	NaN	NaN
230	Burger King	Garden Grilled Chicken Salad w/ Grilled Chicke...	320	120	14	6.0	0.0	115	650	16	2.0	4	36.0	NaN	NaN	NaN
294	Dairy Queen	Crispy Chicken BLT Salad	520	280	31	10.0	0.0	100	1470	25	9.0	6	37.0	NaN	NaN	NaN
372	Subway	Black Forest Ham Salad	110	25	3	1.0	0.0	20	590	11	4.0	6	12.0	25.0	45.0	4.0
381	Subway	Meatball Marinara Salad	310	150	17	7.0	1.0	30	720	25	6.0	10	16.0	60.0	70.0	10.0
160	Arbys	Pecan Chicken Salad Sandwich	840	400	44	6.0	0.5	75	1210	81	6.0	20	33.0	10.0	8.0	25.0
391	Subway	Turkey Breast Salad	110	20	2	1.0	0.0	20	570	11	4.0	5	12.0	25.0	45.0	6.0
383	Subway	Roast Beef Salad	140	30	4	1.0	0.0	40	450	10	4.0	5	18.0	25.0	45.0	4.0
56	Mcdonalds	Premium Southwest Salad w/ Crispy Chicken	520	230	25	6.0	0.0	75	960	46	8.0	9	28.0	180.0	40.0	20.0

In [16]:

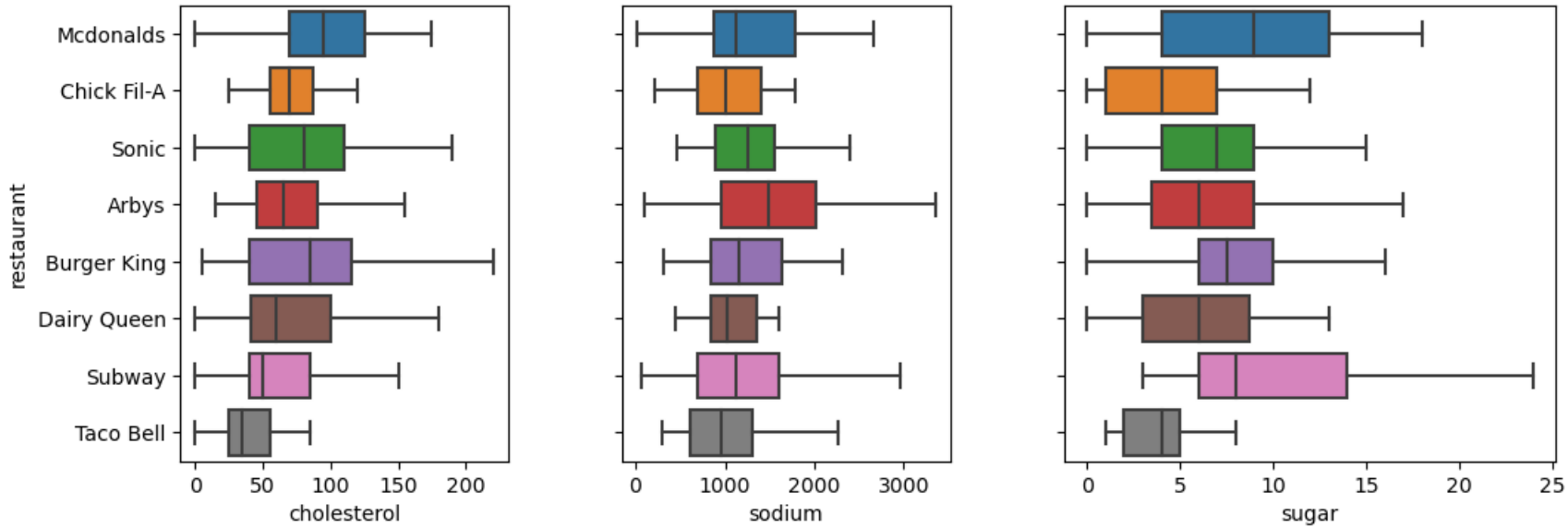
```
fig = px.scatter(df_salad, 'calories', 'cholesterol',  color='restaurant', size='sodium', hover_data='item')
for i, row in df_salad.iterrows():

    if row['cholesterol']==150:
        fig.add_annotation(x=row['calories'], y=row['cholesterol'], text=row['item'], showarrow=True,
                           arrowhead=2, ax=-40, ay=40, font=dict(size=10))

    elif row['calories']==840:
        fig.add_annotation(x=row['calories'], y=row['cholesterol'], text=row['item'], showarrow=True,
                           arrowhead=2, ax=-20, ay=80, font=dict(size=10))
fig.show()
```



Comparison of restaurants based on Sodium, Sugar and Cholestrol of the Meals:

In [17]:
```
fig, axs = plt.subplots(ncols=3, figsize=(12, 4), gridspec_kw={'width_ratios': [2, 2, 3]})

sns.boxplot(y='restaurant', x='cholesterol', data=df_food, ax=axs[0], showfliers=False)
sns.boxplot(y='restaurant', x='sodium', data=df_food, ax=axs[1], showfliers=False)
```

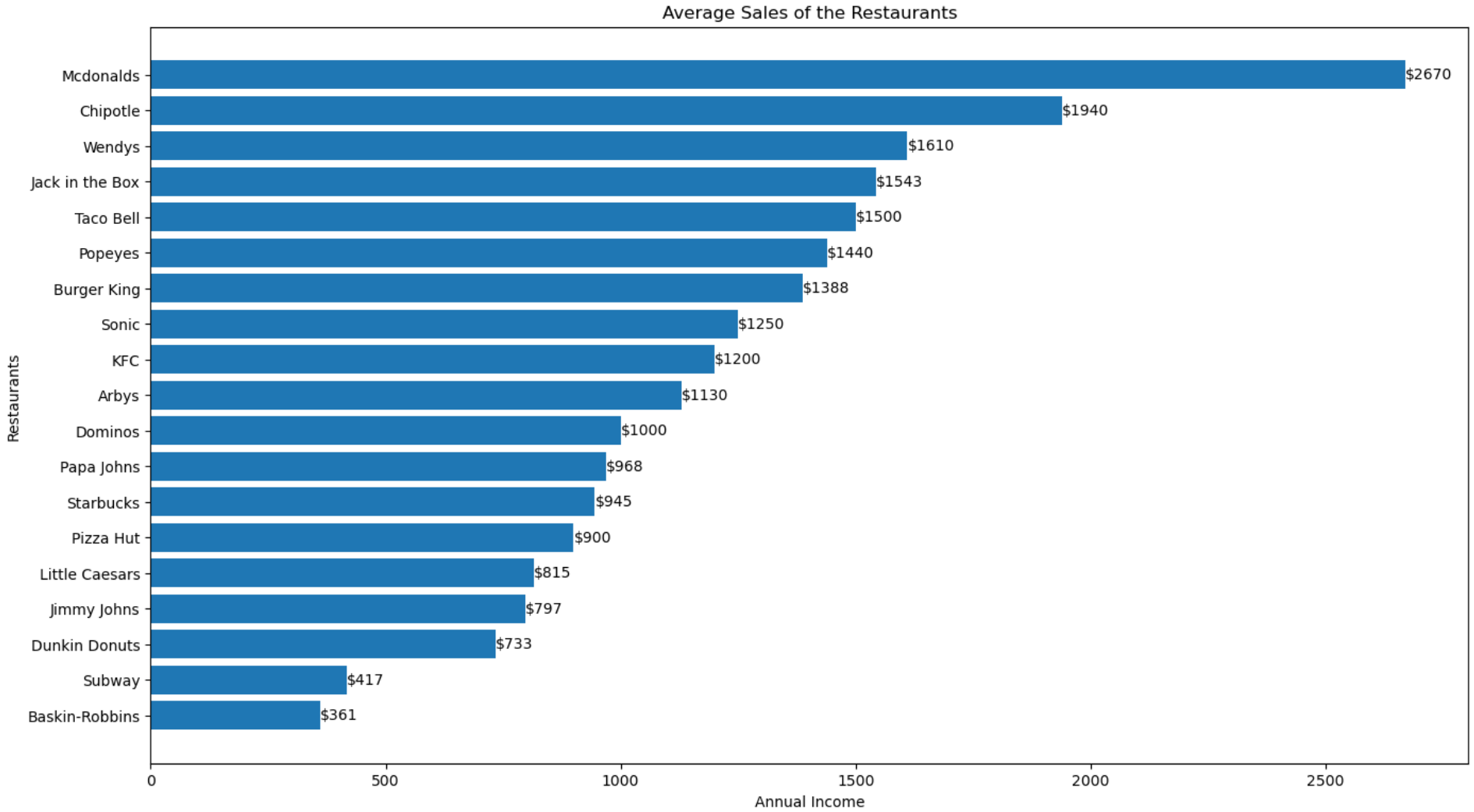
```
sns.boxplot(y='restaurant', x='sugar', data=df_food, ax=axs[2], showfliers=False)

axs[0].xaxis.set_major_locator(FixedLocator(axs[0].get_xticks()))
axs[0].tick_params(axis='y', labelleft=True)

axs[1].xaxis.set_major_locator(FixedLocator(axs[1].get_xticks()))
axs[1].tick_params(axis='y', labelleft=False)

axs[2].xaxis.set_major_locator(FixedLocator(axs[2].get_xticks()))
axs[2].tick_params(axis='y', labelleft=False)

axs[1].set_ylabel('')  # Hide y-axis label for the second plot
axs[2].set_ylabel('')  # Hide y-axis label for the third plot

axs[0].set_title('')  # Remove title for the first plot
axs[1].set_title('')  # Remove title for the second plot
axs[2].set_title('')  # Remove title for the third plot

plt.subplots_adjust(wspace=0.3)  # Adjust the spacing between subplots

plt.show()
```



Comparison of Average Sales of the Restaurants:

In [18]:
```
df_sale_sorted = df_sale.sort_values('average_sales', ascending=True)
num_bars = len(df_sale_sorted)
fig, ax = plt.subplots(figsize=(16, 9))
ax.barh(df_sale_sorted['restaurant'], df_sale_sorted['average_sales'])
plt.xlabel('Annual Income')
```

```
plt.ylabel('Restaurants')
plt.title('Average Sales of the Restaurants')
for i, v in enumerate(df_sale_sorted['average_sales']):
    ax.text(v, i, '${:.0f}'.format(v), ha='left', va='center')
plt.show()
```



Extracting Veggie and Regular Foods:

In [19]:
```
df_veggie = df_food1[df_food1['item'].str.contains('Veg',flags=re.IGNORECASE) == True]
print("There are",str(len(df_veggie.index)),"veggie foods in the dataframe")
```

```
There are 13 veggie foods in the dataframe
```

In [20]:
```
title = "<h2 style='text-align:center;'><b><u><font color='purple'>Sample of Veggie Items (13 rows)</font></u></b></h2>"
centered_title = title.center(80)  # Adjust the width as needed

display(HTML(centered_title))
display(df_veggie.sample(13))
```

Sample of Veggie Items (13 rows)

	restaurant	item	calories	cal_fat	total_fat	sat_fat	trans_fat	cholesterol	sodium	total_carb	fiber	sugar	protein	vit_a	vit_c	calcium
365	Subway	Footlong Veggie Delite	460	40	6	2.0	0.0	0	620	88	10.0	12	16.0	16.0	40.0	60.0
366	Subway	6" Veggie Patty	390	70	7	1.0	0.0	10	800	56	8.0	8	23.0	15.0	20.0	35.0
103	Sonic	Veggie Burger W/ Ketchup	450	130	14	4.0	0.0	10	1410	67	5.0	11	15.0	6.0	8.0	25.0
367	Subway	Footlong Veggie Patty	780	140	14	2.0	0.0	20	1600	112	16.0	16	46.0	30.0	20.0	70.0
105	Sonic	Veggie Burger W/ Mustard	450	130	14	4.0	0.0	10	1300	64	5.0	8	15.0	6.0	8.0	25.0
363	Subway	Kids Mini Sub Veggie Delite	150	15	2	0.0	0.0	0	190	29	3.0	4	6.0	6.0	15.0	20.0
104	Sonic	Veggie Burger With Mustard	450	130	14	4.0	0.0	10	1350	64	5.0	8	15.0	6.0	8.0	27.0
487	Taco Bell	Cantina Power Bowl - Veggie	540	190	21	3.0	0.0	10	1310	75	14.0	4	14.0	NaN	NaN	NaN
237	Burger King	BK VEGGIE Burger	410	150	16	3.0	0.0	5	1030	44	7.0	8	22.0	NaN	NaN	NaN
396	Subway	Cheese & Veggies Pizza	740	230	25	11.0	0.0	50	1270	100	5.0	9	36.0	35.0	30.0	60.0
392	Subway	Veggie Delite Salad	50	10	1	0.0	0.0	0	65	9	4.0	4	3.0	25.0	45.0	4.0
364	Subway	6" Veggie Delite	230	20	3	1.0	0.0	0	310	44	5.0	6	8.0	8.0	20.0	30.0
412	Taco Bell	Cantina Power Burrito - Veggie	740	230	26	5.0	0.0	10	1750	107	17.0	8	20.0	NaN	NaN	NaN

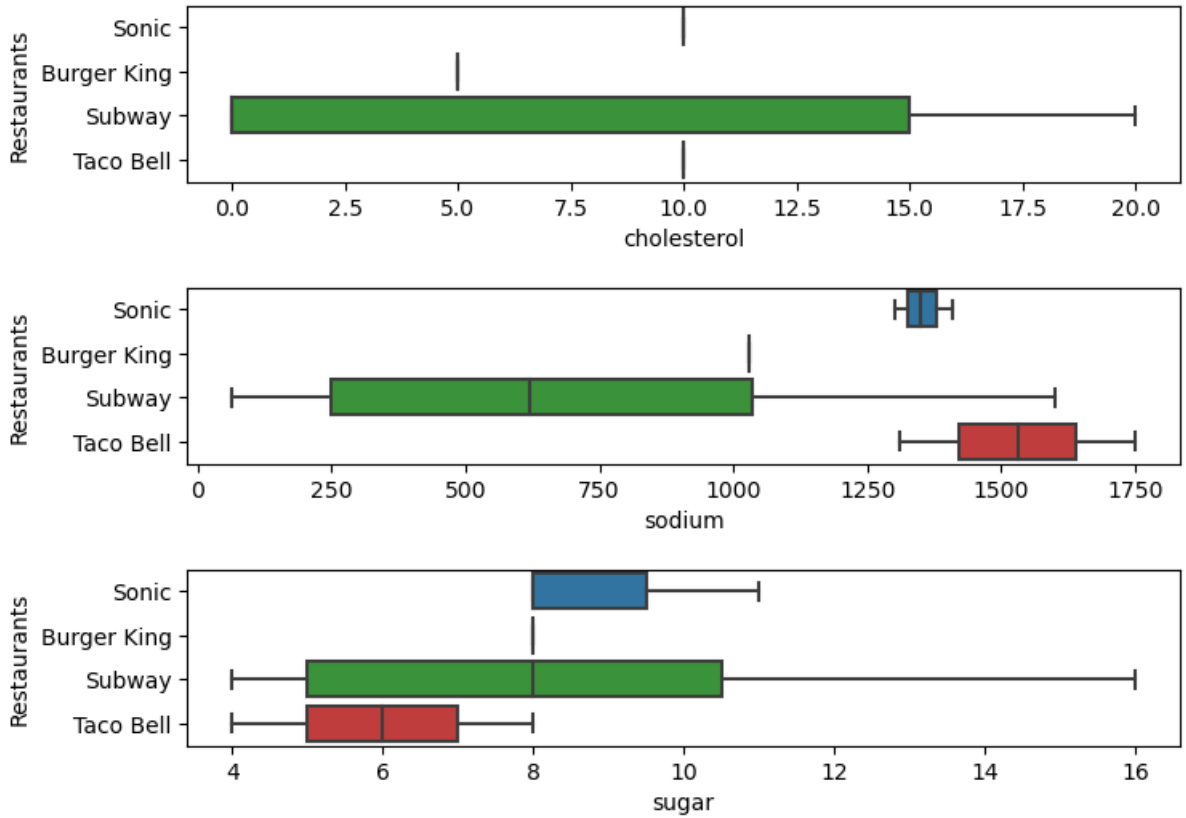
In [21]:
```
df_regular = df_food1[df_food1['item'].str.contains('Veg',flags=re.IGNORECASE) == False]
print("There are",str(len(df_regular.index)),"regular foods (for one person) in the dataframe")
```

```
There are 478 regular foods (for one person) in the dataframe
```

Study on Veggie Food:

In [22]:
```
fig, axs = plt.subplots(ncols=1, nrows=3, figsize=(8, 6), sharey=True)

sns.boxplot(y='restaurant', x='cholesterol', data=df_veggie, ax=axs[0], showfliers=False)
sns.boxplot(y='restaurant', x='sodium', data=df_veggie, ax=axs[1], showfliers=False)
sns.boxplot(y='restaurant', x='sugar', data=df_veggie, ax=axs[2], showfliers=False)

axs[0].xaxis.set_major_locator(FixedLocator(axs[0].get_xticks()))
axs[0].tick_params(axis='y', labelleft=True)

axs[1].xaxis.set_major_locator(FixedLocator(axs[1].get_xticks()))
axs[1].tick_params(axis='y', labelleft=True)
```

```
axs[2].xaxis.set_major_locator(FixedLocator(axs[2].get_xticks()))
axs[2].tick_params(axis='y', labelleft=True)

axs[0].set_ylabel('Restaurants')
axs[1].set_ylabel('Restaurants')
axs[2].set_ylabel('Restaurants')

plt.subplots_adjust(hspace=0.6)  # Adjust the spacing between subplots

plt.show()
```



In [23]:
```
import plotly.subplots as sp
import plotly.graph_objects as go
import pandas as pd

# Create subplots with 1 row and 2 columns
fig = sp.make_subplots(rows=1, cols=2, subplot_titles=("Ordinary", "Giant"), horizontal_spacing=0.1,
                       column_widths=[0.45, 0.45], shared_yaxes=False,
                       specs=[[{'type': 'scatter'}, {'type': 'scatter'}]])
```

```python
# Add the first scatter plot to the first subplot and customize hover information
df_sale_ord=df_sale[df_sale['us_sales'] < 5000]
fig.add_trace(go.Scatter(x=df_sale_ord['unit_count'], y=df_sale_ord['us_sales'], mode='markers',
                         marker=dict(size=15, color=df_sale_ord['average_sales'], showscale=True,
                                     colorbar=dict(x=-0.25, y=0.5, title='Average Sales')),
                         hovertemplate='%{text}<br>Average Sale: %{customdata}',
                         text=df_sale_ord['restaurant'],
                         customdata=df_sale_ord['average_sales'],
                         showlegend=False),
              row=1, col=1)

# Add the second scatter plot to the second subplot and customize hover information
df_sale_gia=df_sale[df_sale['us_sales'] >= 5000]
fig.add_trace(go.Scatter(x=df_sale_gia['unit_count'], y=df_sale_gia['us_sales'], mode='markers',
                         marker=dict(size=15, color=df_sale_gia['average_sales'], showscale=True,
                                     colorbar=dict(x=1.05, y=0.5, title='Average Sales')),
                         hovertemplate='%{text}<br>Average Sale: %{customdata}',
                         text=df_sale_gia['restaurant'],
                         customdata=df_sale_gia['average_sales'],
                         showlegend=False),
              row=1, col=2)

# Update the layout of the subplots with axis titles
fig.update_layout(height=500, width=900)

# Add y-axis titles to subplots
fig.update_yaxes(title_text="US Sales", row=1, col=1)

# Add x-axis titles to subplots
fig.update_xaxes(title_text="Unit Count", row=1, col=1)
fig.update_xaxes(title_text="Unit Count", row=1, col=2)

# Show the combined scatter plots
fig.show()
```


Ordinary
Giant
Average Sales
US Sales
1800
1600
1400
1200
1000
800
600
400
4000
3000
2000
1000
3000
4000
Unit Count
40k
35k
30k
25k
20k
15k
10k
5k
5k
10k
15k
20k
25k
Unit Count
Average Sales
2500
2000
1500
1000
500