Project Overview - Analyzing Workout and Health Data

ng project_description = """

📊 Project Overview: Analyzing Workout and Health Data 📊

This project aims to analyze a dataset containing workout and health metrics of individuals. The focus is on understanding relationships between various attributes, such as age, gender, workout types, calories burned, and body measurements.

We will explore the data through:

- Descriptive statistics.
- Visualizations such as histograms, bar charts, and correlation heatmaps.
- Uncovering trends and insights that can guide better workout plans and health strategies.

Key Steps:

1. **Data Cleaning**: Handle missing values and outliers.
2. **Descriptive Statistics**: Calculate basic statistics and distributions.
3. **Data Visualization**: Create visual representations of key data features.
4. **Correlation Analysis**: Understand relationships between different health metrics.
5. **Insights**: Draw conclusions based on the findings.

Let's beginkdown(project_description))

In [1]:
```
# Importing liprarys to analies and manuplation dataset

import numpy as numpy
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Ignoring worning if it's shown
import warnings
warnings.filterwarnings('ignore')
```

Reading or downlowding dataset

In [2]:
```
 # Reading the file fo dataset

fname = 'gym_members_exercise_tracking1.csv'
```

```
data = pd.read_csv(fname)
data
```

Out[2]:

	Age	Gender	Weight (kg)	Height (m)	Max_BPM	Avg_BPM	Resting_BPM	Session_Durati (hou
0	56	Male	88.3	1.71	180	157	60	1.
1	46	Female	74.9	1.53	179	151	66	1.
2	32	Female	68.1	1.66	167	122	54	1.
3	25	Male	53.2	1.70	190	164	56	N
4	38	Male	46.1	1.79	188	158	68	0.
...	...	...	...	...	...	...	...	
968	24	Male	87.1	1.74	187	158	67	1.
969	25	Male	66.6	1.61	184	166	56	1.
970	59	Female	60.4	1.76	194	120	53	1.
971	32	Male	126.4	1.83	198	146	62	1.
972	46	Male	88.7	1.63	400	146	66	0.

973 rows × 15 columns

In [3]:
```
# Show the 6 first raws

data.head(6)
```

Out[3]:

	Age	Gender	Weight (kg)	Height (m)	Max_BPM	Avg_BPM	Resting_BPM	Session_Duration (hours)
0	56	Male	88.3	1.71	180	157	60	1.69
1	46	Female	74.9	1.53	179	151	66	1.30
2	32	Female	68.1	1.66	167	122	54	1.11
3	25	Male	53.2	1.70	190	164	56	NaN
4	38	Male	46.1	1.79	188	158	68	0.64
5	56	Female	58.0	1.68	168	156	74	1.59

Data Quality Check

In [4]:
```
data.isnull().sum()
```

Out[4]:
```
Age                              0
Gender                           0
Weight (kg)                      0
Height (m)                       0
Max_BPM                          0
Avg_BPM                          0
Resting_BPM                      0
Session_Duration (hours)         3
Calories_Burned                  0
Workout_Type                     1
Fat_Percentage                   0
Water_Intake (liters)            0
Workout_Frequency (days/week)    0
Experience_Level                 0
BMI                              0
dtype: int64
```

In [5]:
```
# To full in null value with the mean of Session column

a = data['Session_Duration (hours)'] = data['Session_Duration (hours)'].fillna(d
a.isnull().sum()
```

Out[5]:
```
0
```

In [6]:
```
# To delete all null value in dataset

data = data.dropna()
data["Workout_Type"].isnull().sum()
```

Out[6]:
```
0
```

In [45]:
```
data.nunique()
```

Out[45]:
```
Age                              42
Gender                            2
Weight (kg)                     532
Height (m)                       51
Max_BPM                          45
Avg_BPM                          50
Resting_BPM                      25
Session_Duration (hours)        148
Calories_Burned                 621
Workout_Type                      4
Fat_Percentage                  239
Water_Intake (liters)            23
Workout_Frequency (days/week)     4
Experience_Level                  3
BMI                             770
dtype: int64
```

In [47]:
```
unique_gender_values = data['Workout_Type'].unique()
print(unique_gender_values)
```

```
['Yoga' 'HIIT' 'Cardio' 'Strength']
```

In [49]:
```
# Count the number of people for each workout type

workout_counts = data['Workout_Type'].value_counts()
print(workout_counts)
```

```
Workout_Type
Strength    257
Cardio      255
Yoga        239
HIIT        221
Name: count, dtype: int64
```

In [55]:
```
# # Count the number of people for each workout type
grouped_by_workout = data.groupby('Workout_Type')['Gender'].count()

print(grouped_by_workout)
```

```
Workout_Type
Cardio      255
HIIT        221
Strength    257
Yoga        239
Name: Gender, dtype: int64
```

In [7]:
```
data.describe()
```

Out[7]:

	Age	Weight (kg)	Height (m)	Max_BPM	Avg_BPM	Resting_BPM	Sessic
count	972.000000	972.000000	972.000000	972.000000	972.000000	972.000000	
mean	38.694444	73.798560	1.722438	181.459877	143.752058	62.213992	
std	12.182370	21.146018	0.127709	22.122527	14.345209	7.325413	
min	18.000000	40.000000	1.500000	160.000000	120.000000	50.000000	
25%	28.000000	58.100000	1.620000	170.000000	131.000000	56.000000	
50%	40.000000	69.950000	1.710000	180.000000	143.000000	62.000000	
75%	49.250000	85.925000	1.800000	190.000000	156.000000	68.000000	
max	59.000000	129.900000	2.000000	470.000000	169.000000	74.000000	

In [ ]:
```
```

In [8]:
```
def detect_outliers(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return column[(column < lower_bound) | (column > upper_bound)]
```

In [9]:
```
outliers = detect_outliers(data['Max_BPM'])
print("\n The outlyer in Max_BPM: ")
print(outliers)
```

```
 The outlyer in Max_BPM:
12     350
18     420
28     350
374    470
962    400
965    411
972    400
Name: Max_BPM, dtype: int64
```

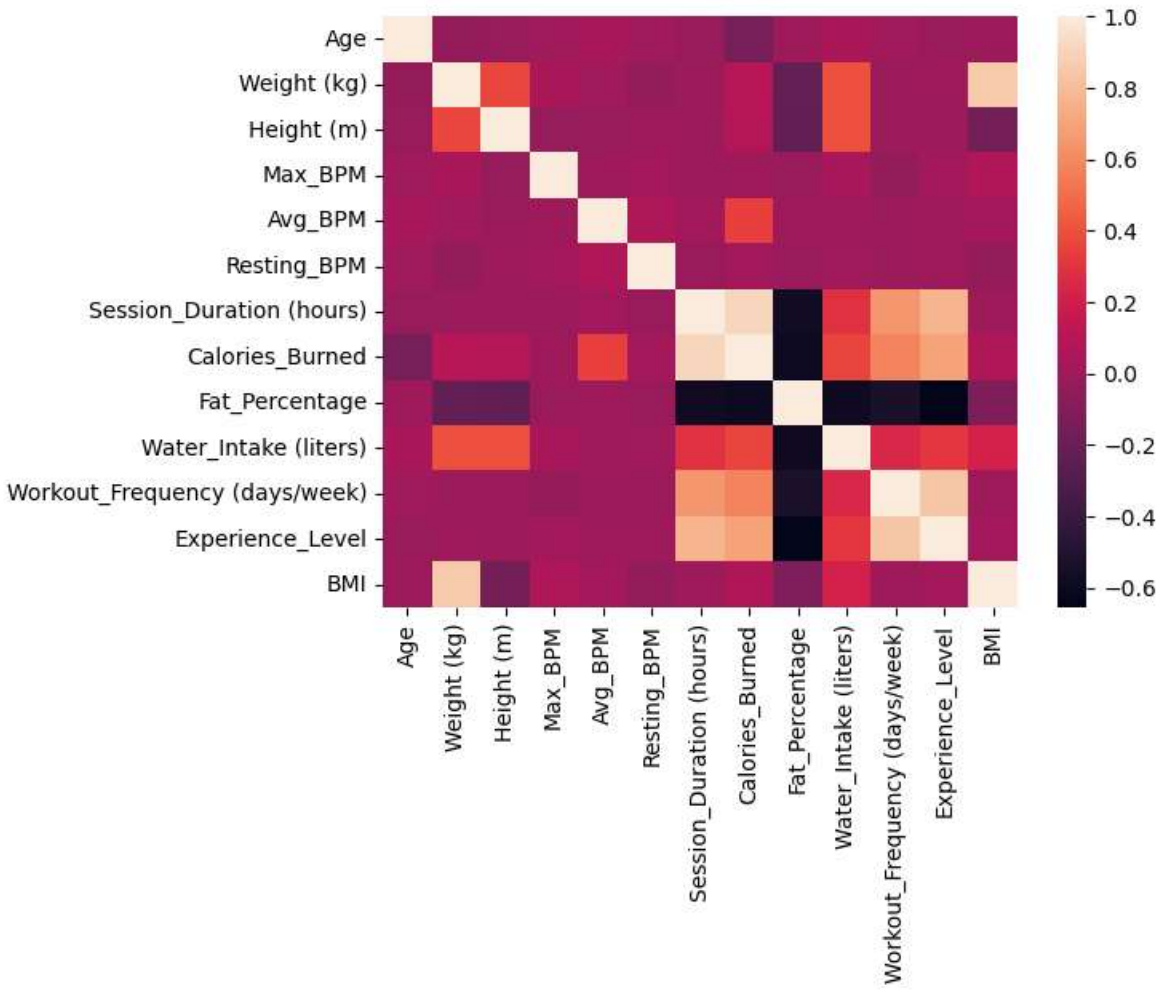
In [11]:
```
numeric_data = data.select_dtypes(include=['float64', 'int64'])
numeric_data.corr().sum()
```

Out[11]:
```
Age                              0.831868
Weight (kg)                      2.437200
Height (m)                       1.337473
Max_BPM                          1.094042
Avg_BPM                          1.444806
Resting_BPM                      0.991244
Session_Duration (hours)         2.960188
Calories_Burned                  3.375178
Fat_Percentage                  -2.579526
Water_Intake (liters)            2.670843
Workout_Frequency (days/week)    2.688580
Experience_Level                 2.949802
BMI                              1.888479
dtype: float64
```

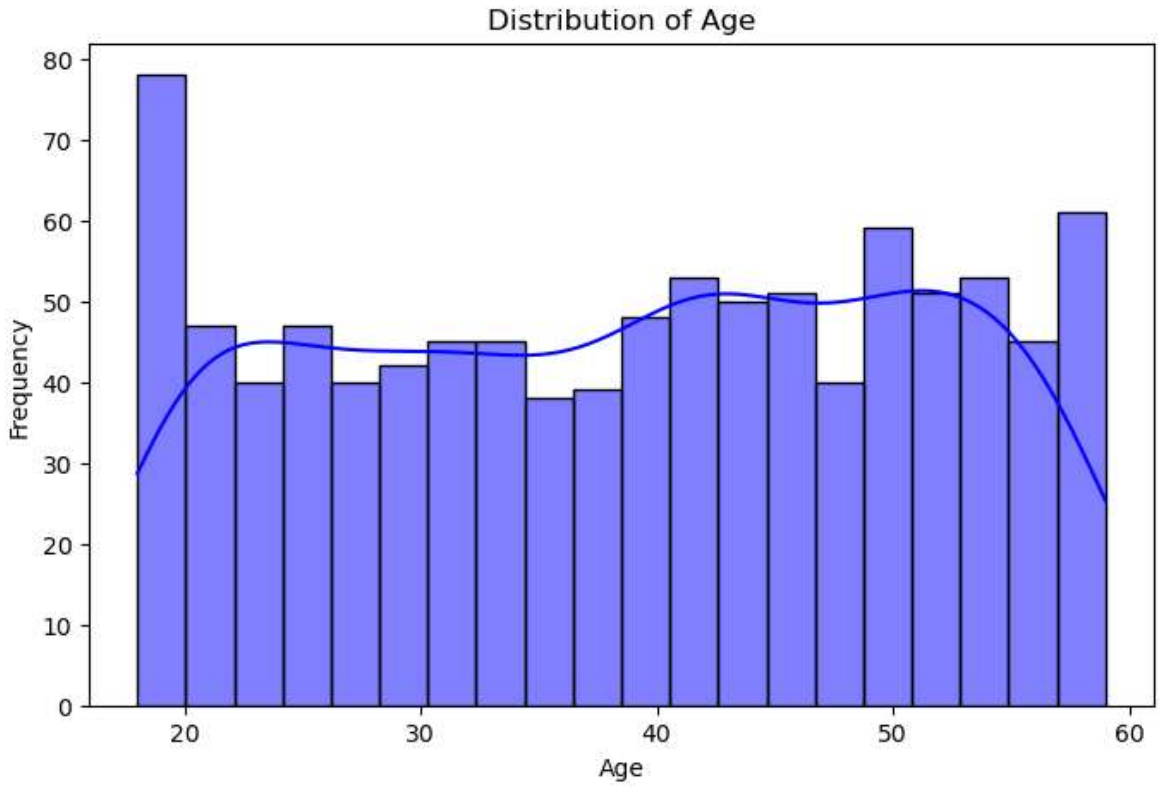
Correlation and Visualization

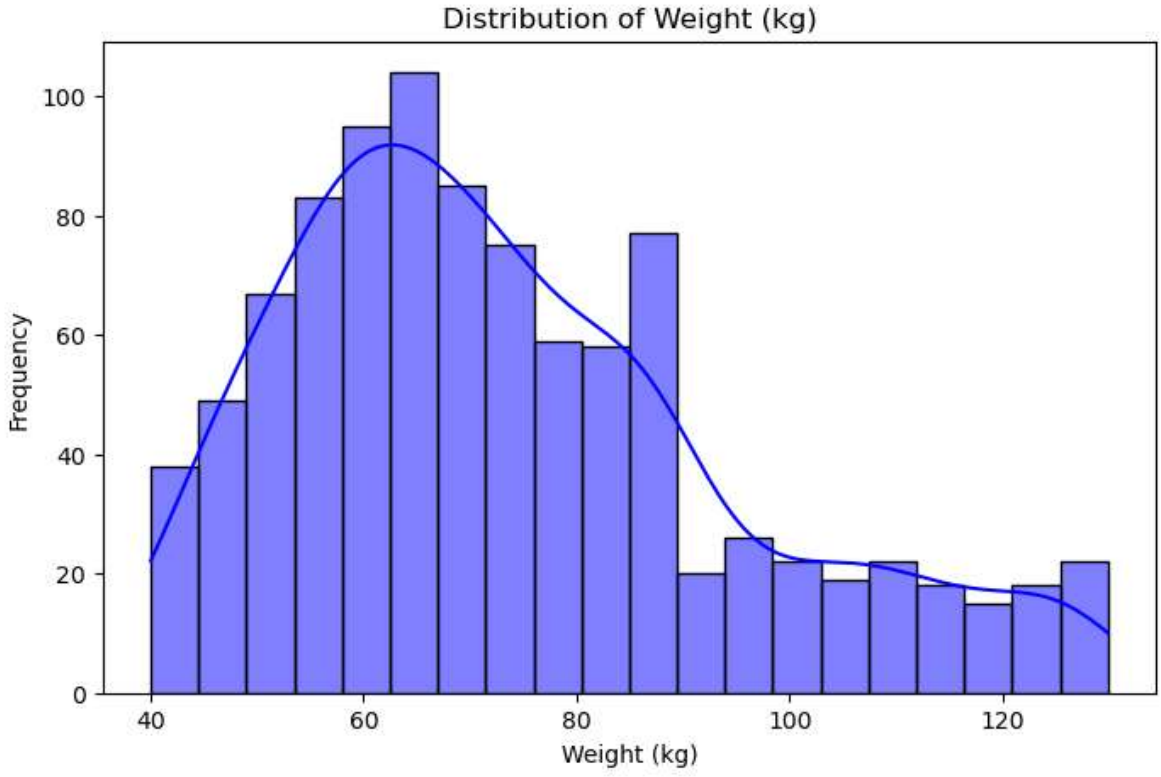
In [12]:
```
sns.heatmap(numeric_data.corr())
```

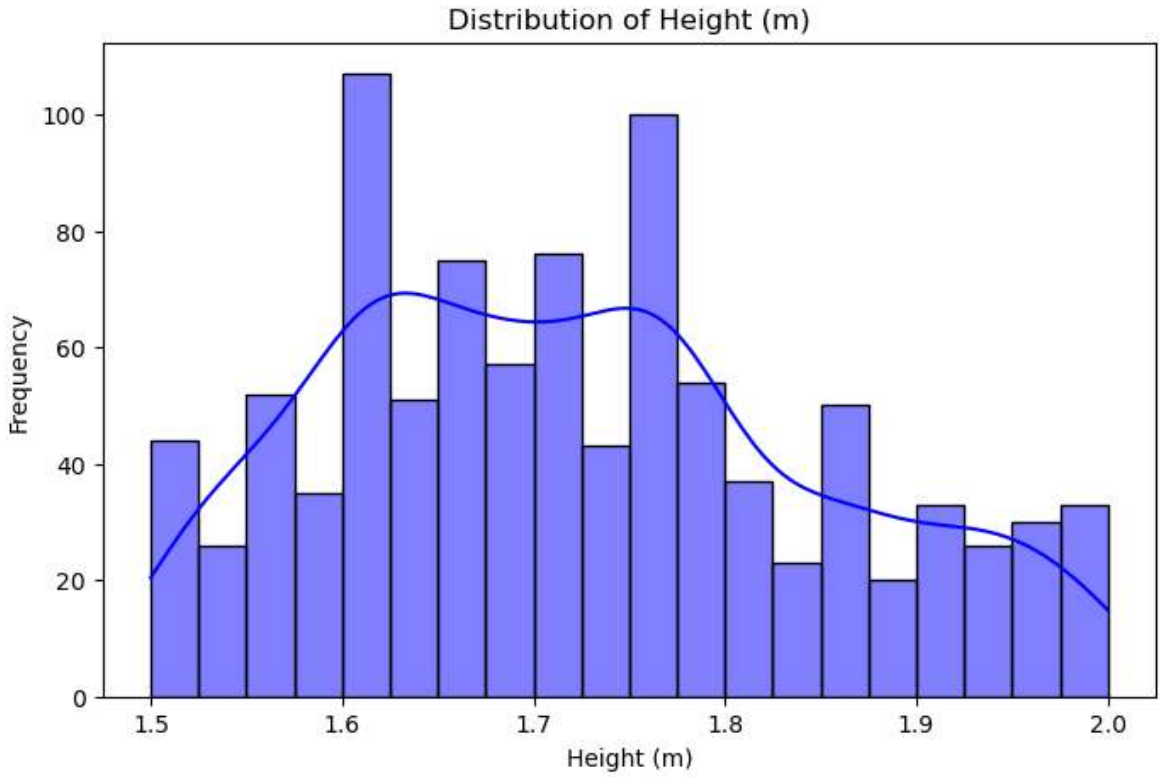
Out[12]:
```
<Axes: >
```

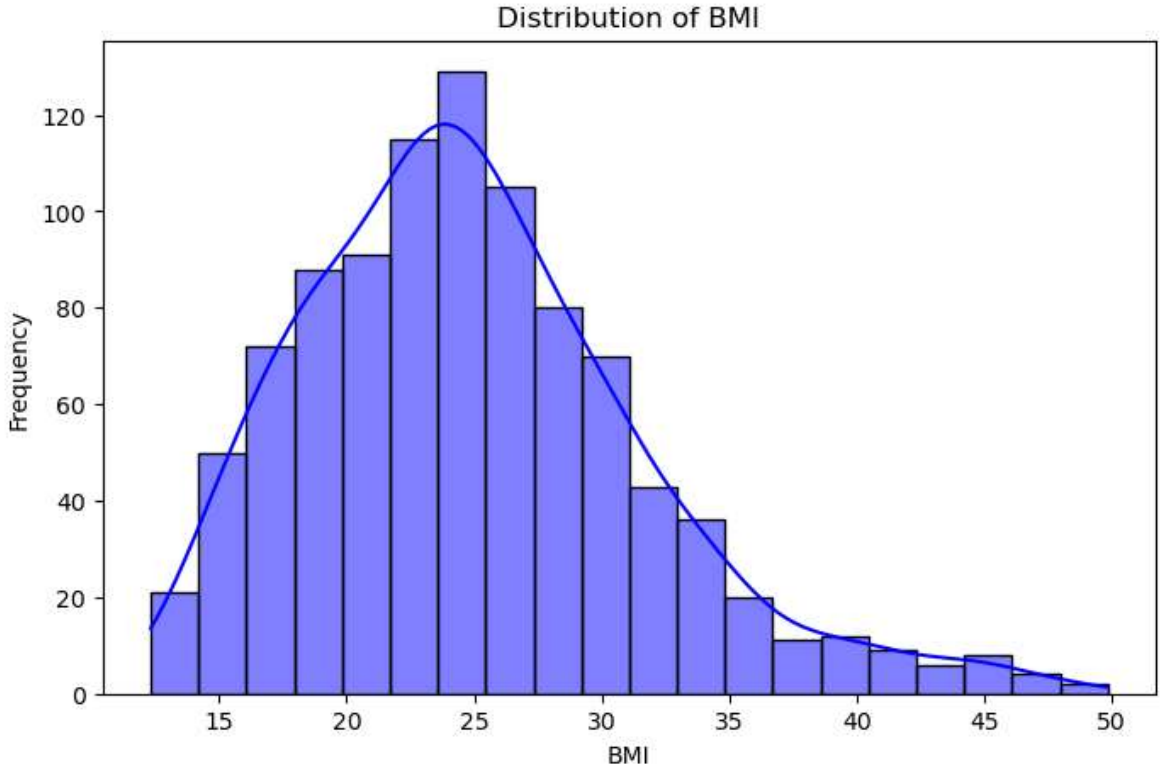


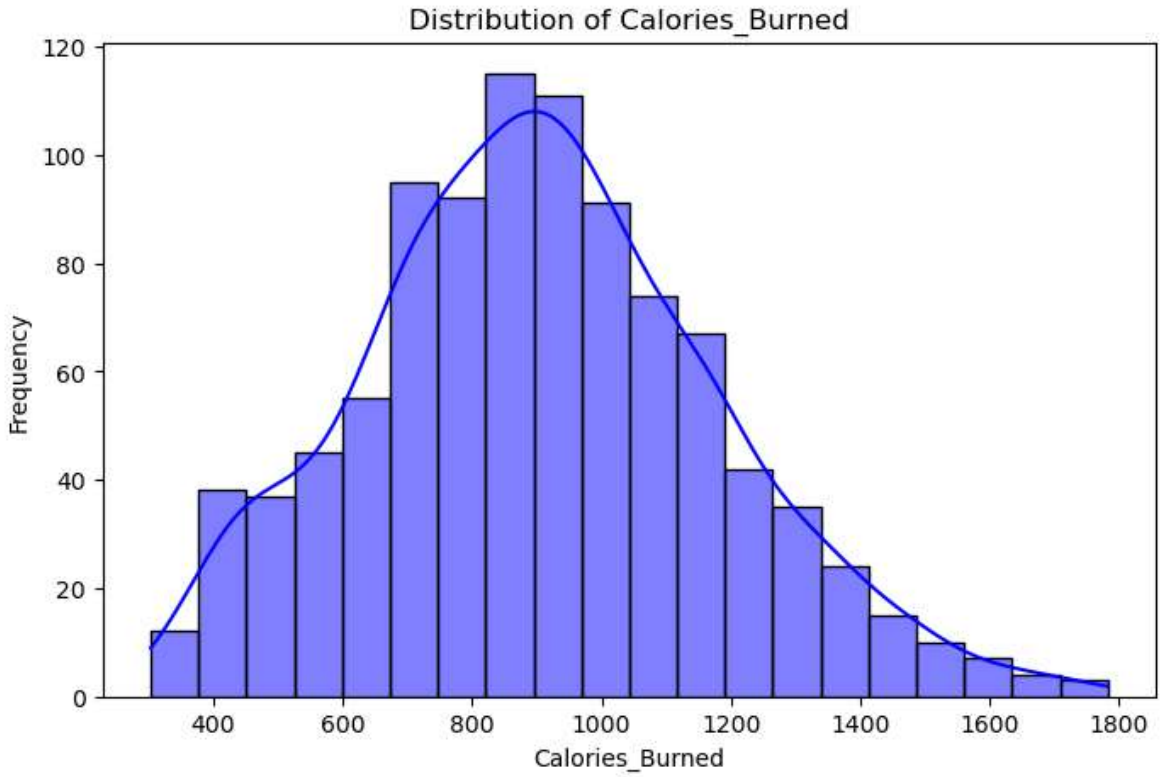
In [13]:
```
numerical_columns = ['Age', 'Weight (kg)', 'Height (m)', 'BMI', 'Calories_Burned
for col in numerical_columns:
    plt.figure(figsize=(8, 5))
    sns.histplot(data[col], kde=True, bins=20, color='blue')
    plt.title(f"Distribution of {col}")
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.show()
```


Distribution of Age
Frequency
Age
0
10
20
30
40
50
60
70
80
20
30
40
50
60

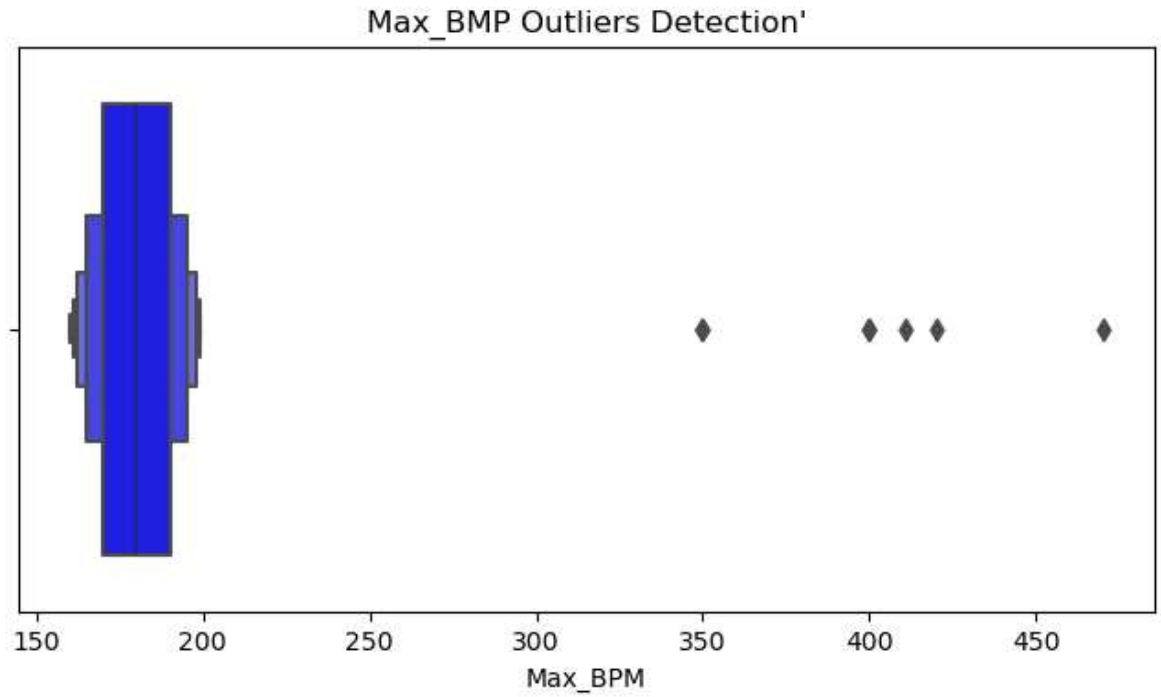

Distribution of Weight (kg)
Frequency
Weight (kg)
0
20
40
60
80
100
40
60
80
100
120


Distribution of Height (m)
Frequency
Height (m)
0
20
40
60
80
100
1.5
1.6
1.7
1.8
1.9
2.0

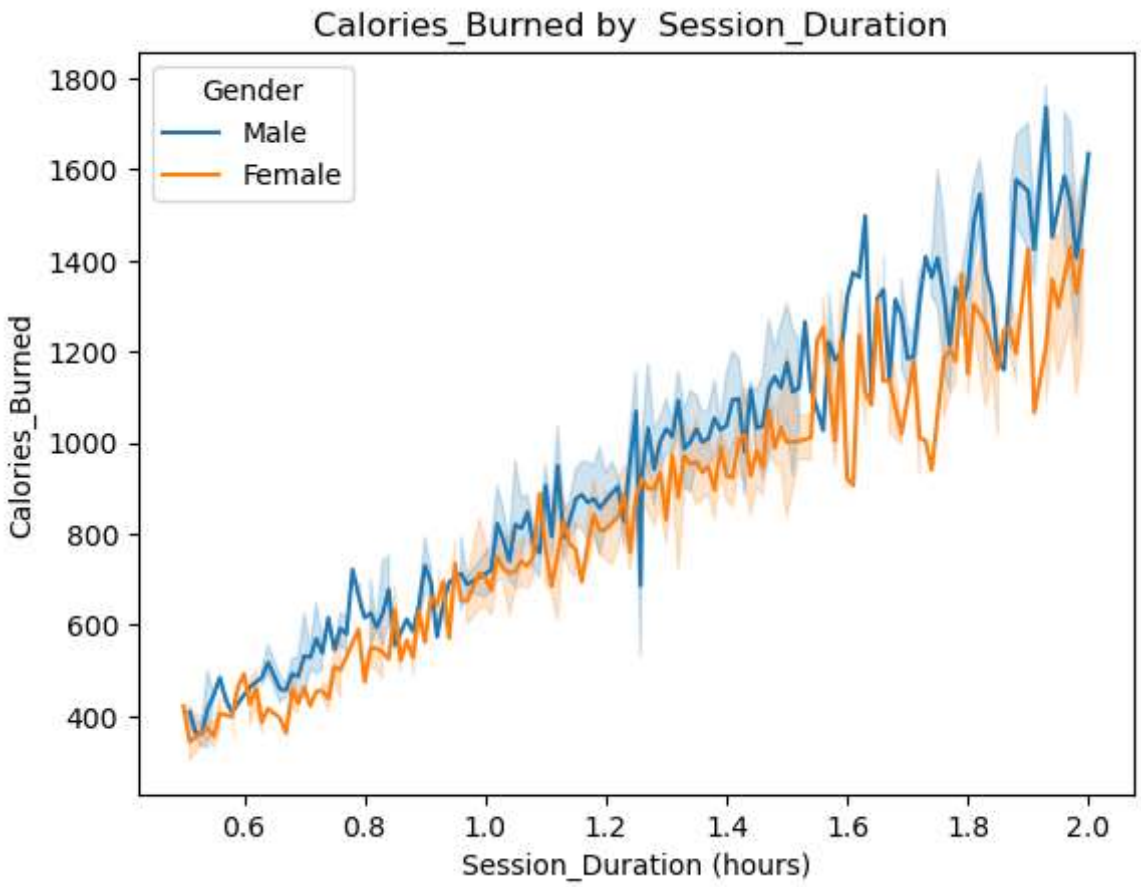

Distribution of BMI
Frequency
BMI
0
20
40
60
80
100
120
15
20
25
30
35
40
45
50



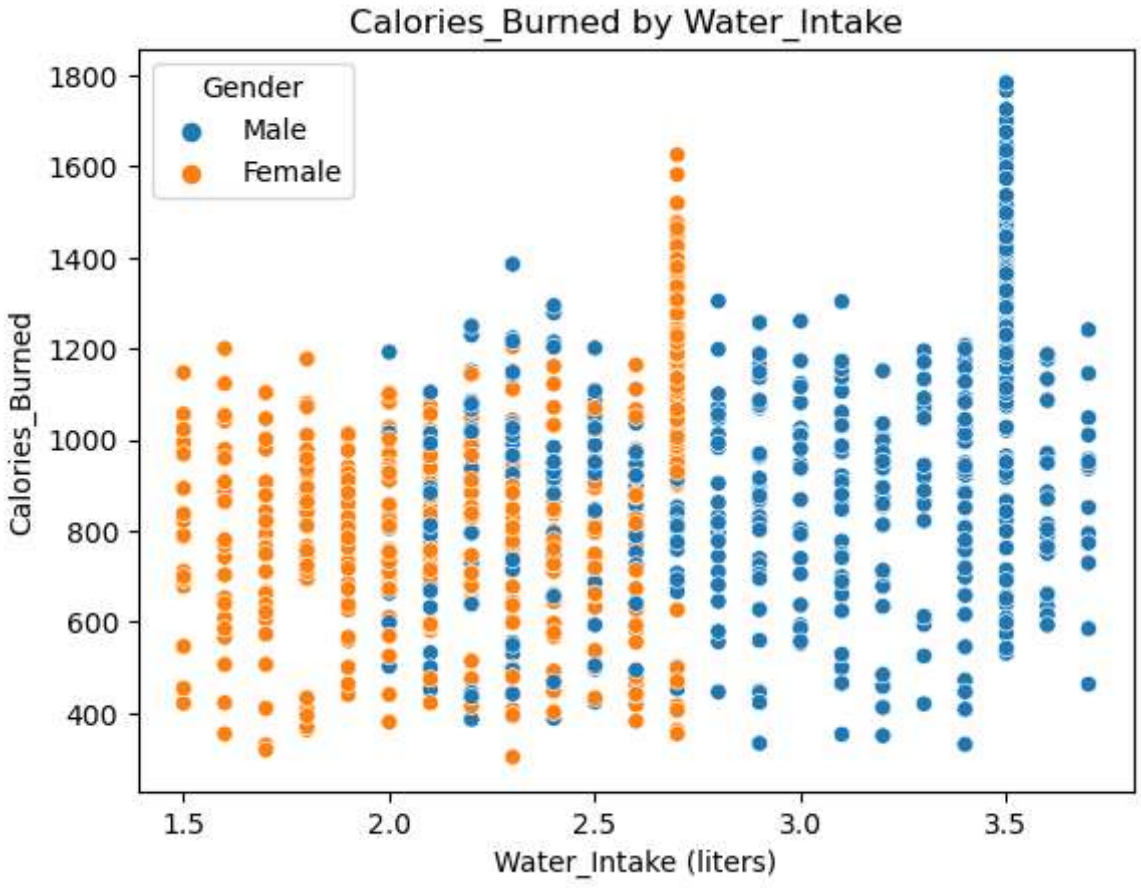
In [14]:
```
plt.figure(figsize = (8,4))
sns.boxenplot(x = data["Max_BPM"], color = 'blue')
plt.title("Max_BMP Outliers Detection'")
plt.show()
```



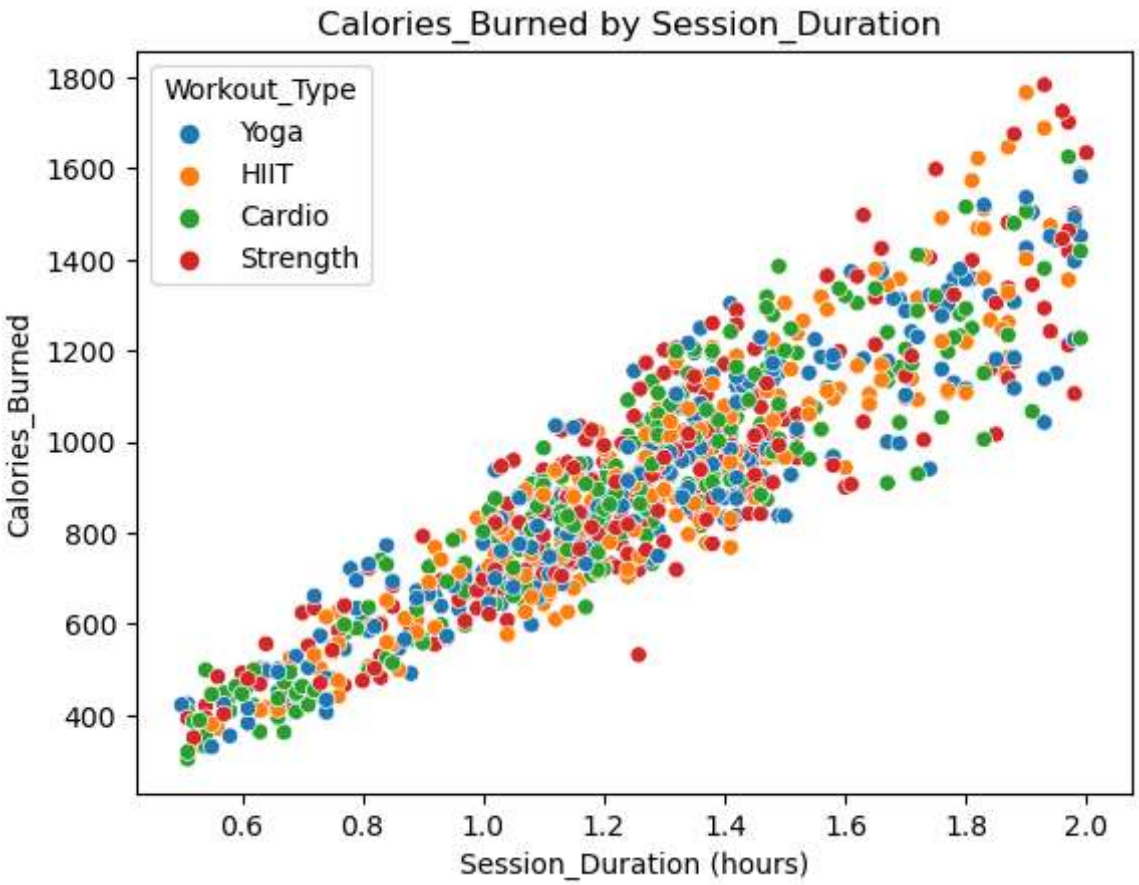
In [36]:
```
sns.lineplot(data ,x='Session_Duration (hours)',y='Calories_Burned',hue='Gender'
plt.title("Calories_Burned by  Session_Duration ")
plt.show()
```



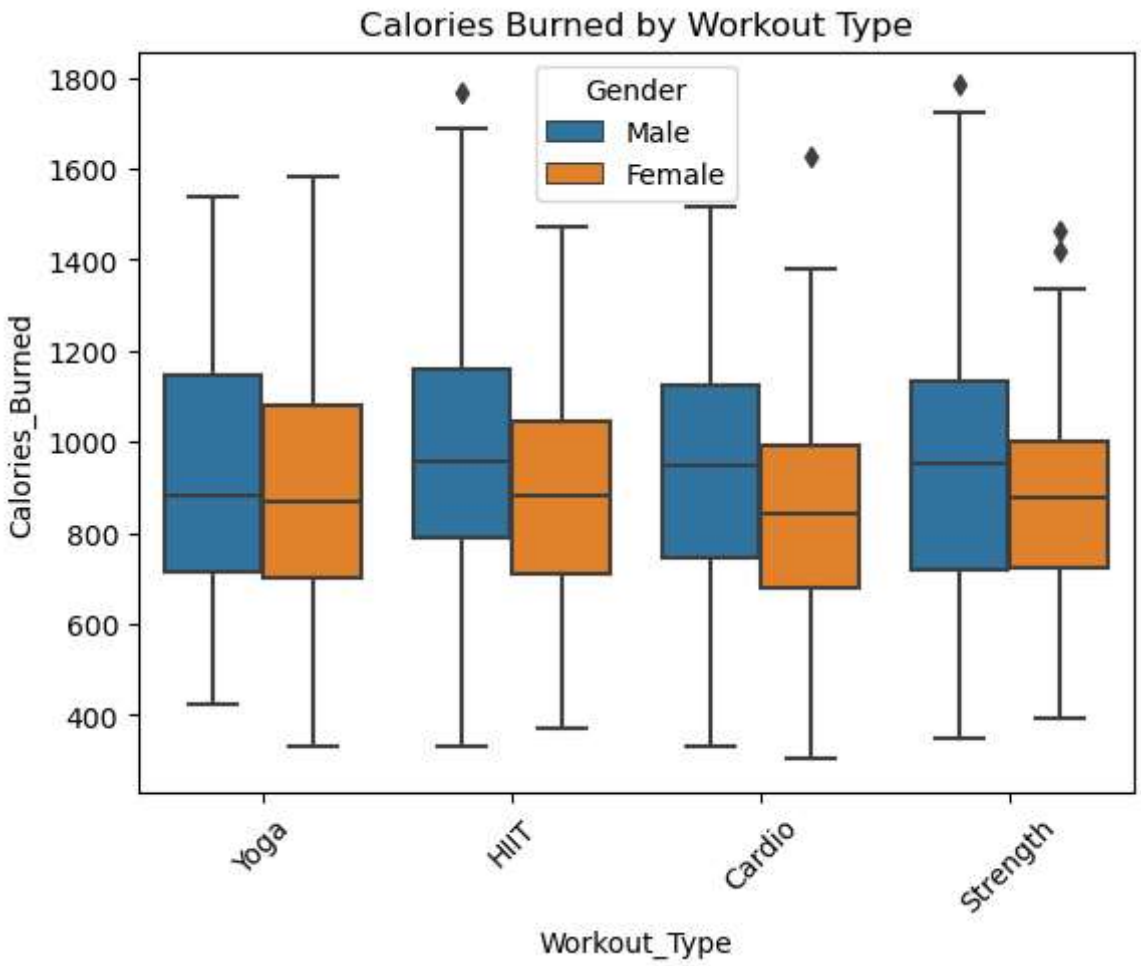
In [37]:
```
sns.scatterplot(data, x='Water_Intake (liters)',y='Calories_Burned',hue='Gender'
plt.title("Calories_Burned by Water_Intake  ")
plt.show()
```



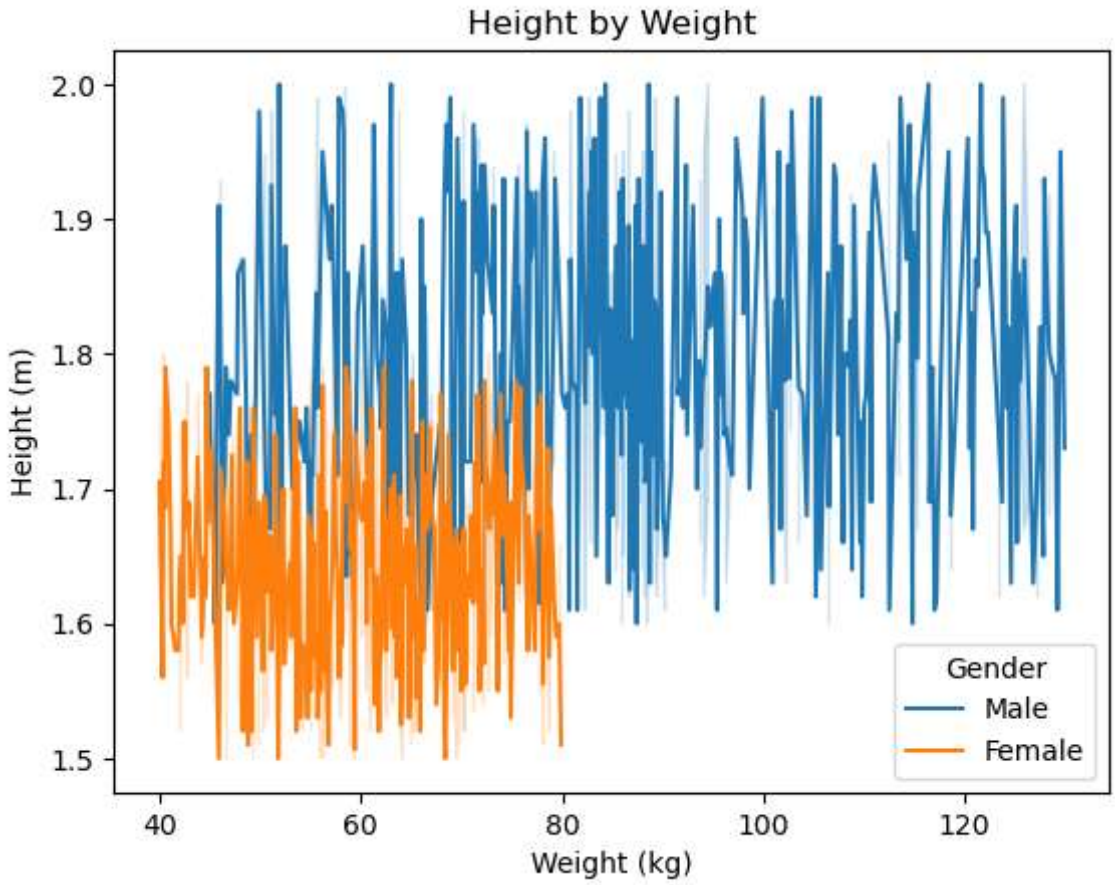
In [39]:
```
sns.scatterplot(data , x ="Session_Duration (hours)", y = "Calories_Burned", hue
plt.title("Calories_Burned by Session_Duration ")
plt.show()
```



In [40]:
```
sns.boxplot(data=data, x='Workout_Type', y='Calories_Burned', hue='Gender')
plt.title('Calories Burned by Workout Type')
plt.xticks(rotation=45)
plt.show()
```



In [53]:
```
sns.lineplot(data ,x='Weight (kg)',y='Height (m)',hue='Gender')
plt.title(" Height by Weight ")
plt.show()
```


Height by Weight
Height (m)
Weight (kg)
2.0
1.9
1.8
1.7
1.6
1.5
40
60
80
100
120
Gender
Male
Female

In [ ]: