In [2]:
```
import pandas as pd
```

In [3]:
```
import numpy as np
```

In [4]:
```
import matplotlib.pyplot as plt
```

In [5]:
```

import seaborn as sns
```

In [6]:
```
data=pd.read_csv('gym_members_exercise_tracking.csv')
```

In [7]:
```
data
```

Out[7]:

	Age	Gender	Weight (kg)	Height (m)	Max_BPM	Avg_BPM	Resting_BPM	Session_Duration (hours)	Calor
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	0.59	
4	38	Male	46.1	1.79	188	158	68	0.64	
...	...	...	...	...	...	...	...	...	
968	24	Male	87.1	1.74	187	158	67	1.57	
969	25	Male	66.6	1.61	184	166	56	1.38	
970	59	Female	60.4	1.76	194	120	53	1.72	
971	32	Male	126.4	1.83	198	146	62	1.10	
972	46	Male	88.7	1.63	166	146	66	0.75	

973 rows × 15 columns

In [8]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 973 entries, 0 to 972
Data columns (total 15 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   Age                            973 non-null    int64  
 1   Gender                         973 non-null    object 
 2   Weight (kg)                    973 non-null    float64
 3   Height (m)                     973 non-null    float64
 4   Max_BPM                        973 non-null    int64  
 5   Avg_BPM                        973 non-null    int64  
 6   Resting_BPM                    973 non-null    int64  
 7   Session_Duration (hours)       973 non-null    float64
 8   Calories_Burned                973 non-null    float64
 9   Workout_Type                   973 non-null    object 
 10  Fat_Percentage                 973 non-null    float64
 11  Water_Intake (liters)          973 non-null    float64
 12  Workout_Frequency (days/week)  973 non-null    int64  
 13  Experience_Level               973 non-null    int64  
 14  BMI                            973 non-null    float64
dtypes: float64(7), int64(6), object(2)
memory usage: 114.2+ KB
```

In [9]:
```
data.describe()
```

Out[9]:

	Age	Weight (kg)	Height (m)	Max_BPM	Avg_BPM	Resting_BPM	Session_Durati (hou
count	973.000000	973.000000	973.00000	973.000000	973.000000	973.000000	973.0000
mean	38.683453	73.854676	1.72258	179.883864	143.766701	62.223022	1.2564
std	12.180928	21.207500	0.12772	11.525686	14.345101	7.327060	0.3430
min	18.000000	40.000000	1.50000	160.000000	120.000000	50.000000	0.5000
25%	28.000000	58.100000	1.62000	170.000000	131.000000	56.000000	1.0400
50%	40.000000	70.000000	1.71000	180.000000	143.000000	62.000000	1.2600
75%	49.000000	86.000000	1.80000	190.000000	156.000000	68.000000	1.4600
max	59.000000	129.900000	2.00000	199.000000	169.000000	74.000000	2.0000

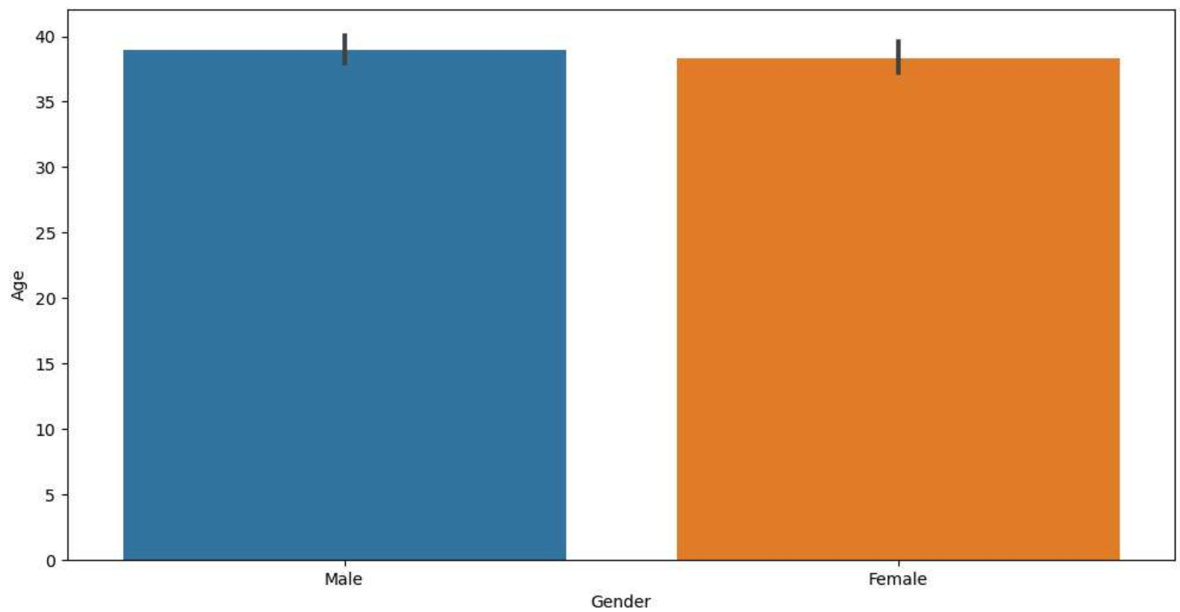
In [10]:
```
data.duplicated().sum()
```

Out[10]:
```
0
```

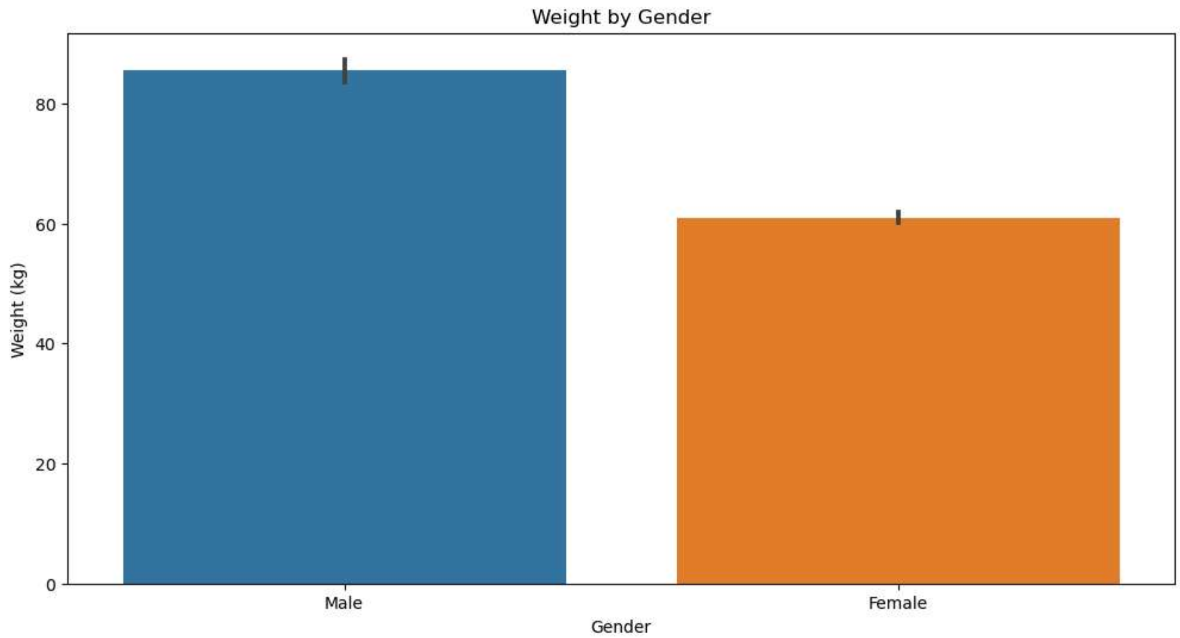
In [11]:
```
data.isna().sum()
```

Out[11]:
```
Age                              0
Gender                           0
Weight (kg)                      0
Height (m)                       0
Max_BPM                          0
Avg_BPM                          0
Resting_BPM                      0
Session_Duration (hours)         0
Calories_Burned                  0
Workout_Type                     0
Fat_Percentage                   0
Water_Intake (liters)            0
Workout_Frequency (days/week)    0
Experience_Level                 0
BMI                              0
dtype: int64
```

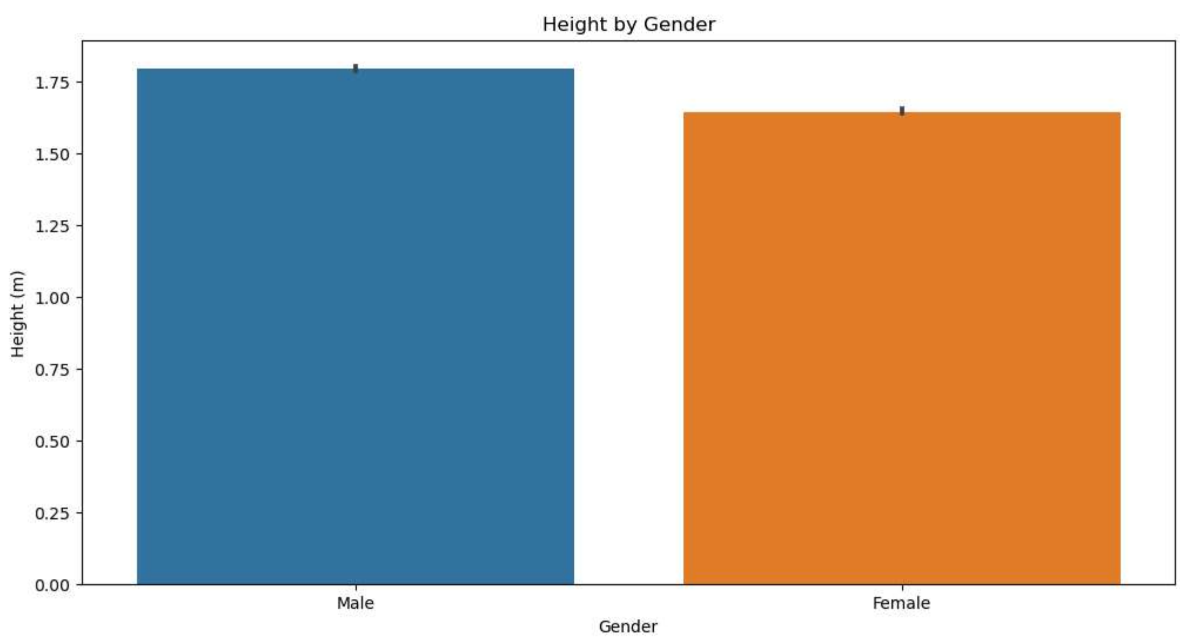
In [12]:
```
plt.figure(figsize=(12,6))
sns.barplot(x='Gender',y='Age',data=data)
plt.show()
```



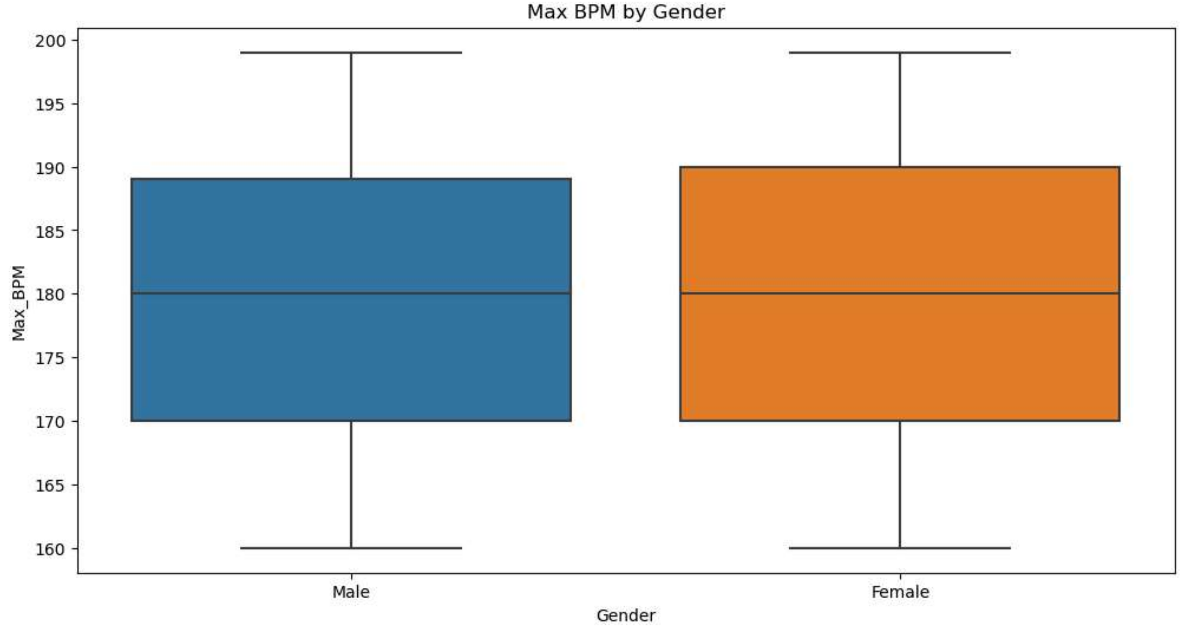
In [13]:
```
# gender vs weight
plt.figure(figsize=(12,6))
sns.barplot(x='Gender',y='Weight (kg)',data=data)
plt.title('Weight by Gender')
plt.show()
```



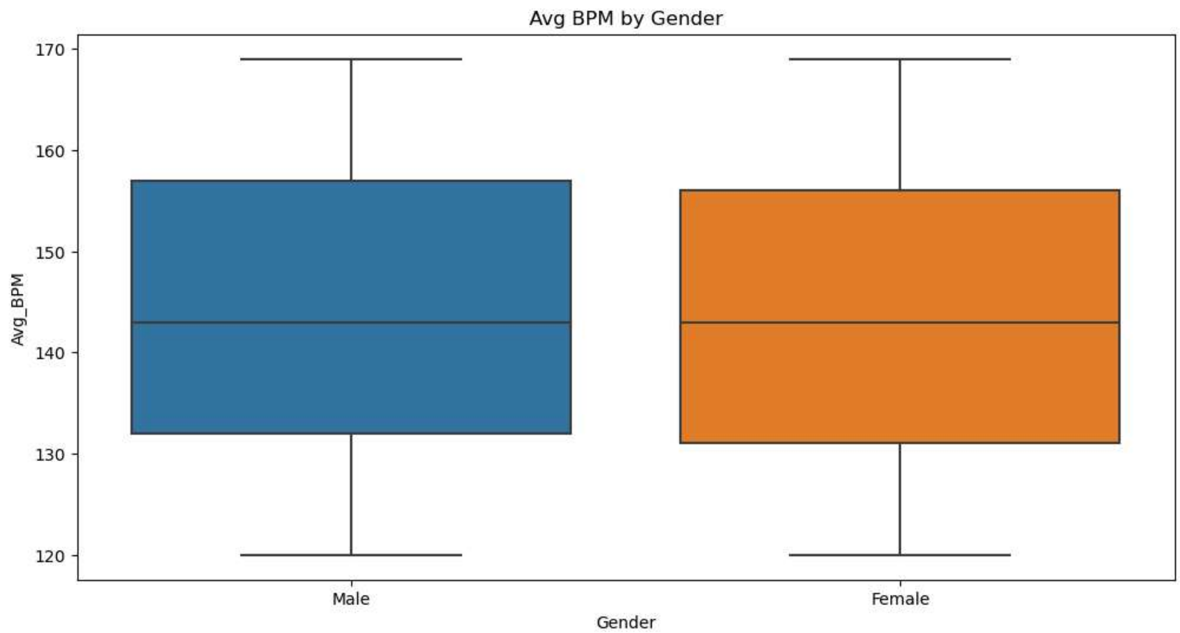
In [14]:
```
# gender vs hight
plt.figure(figsize=(12,6))
sns.barplot(x='Gender',y='Height (m)',data=data)
plt.title('Height by Gender')
plt.show()
```



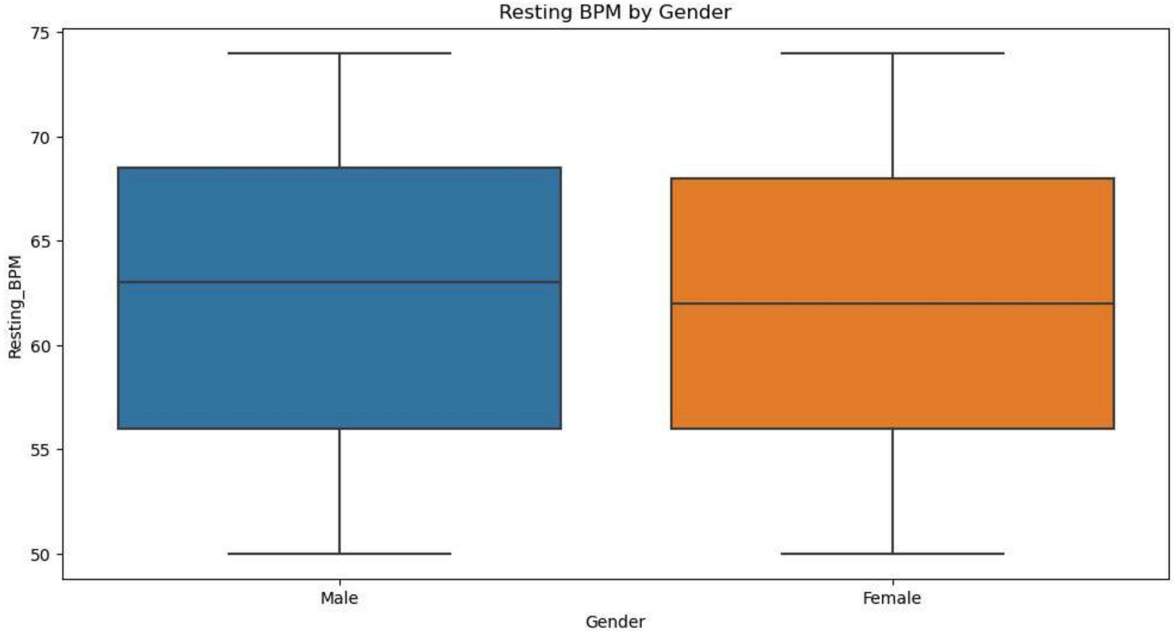
In [15]:
```
#Max BPM by Gender'
plt.figure(figsize=(12,6))
sns.boxplot(x='Gender',y='Max_BPM',data=data)
plt.title('Max BPM by Gender')
plt.show()
```



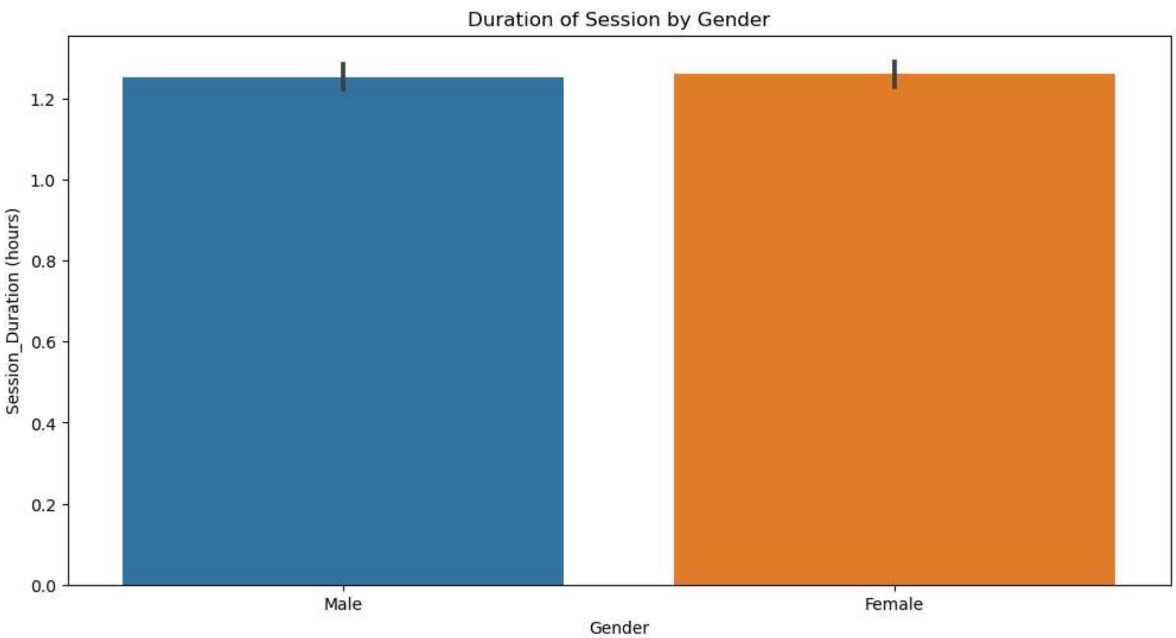
In [16]:
```
#Avg BPM by Gende
plt.figure(figsize=(12,6))
sns.boxplot(x='Gender',y='Avg_BPM',data=data)
plt.title('Avg BPM by Gender')
plt.show()
```



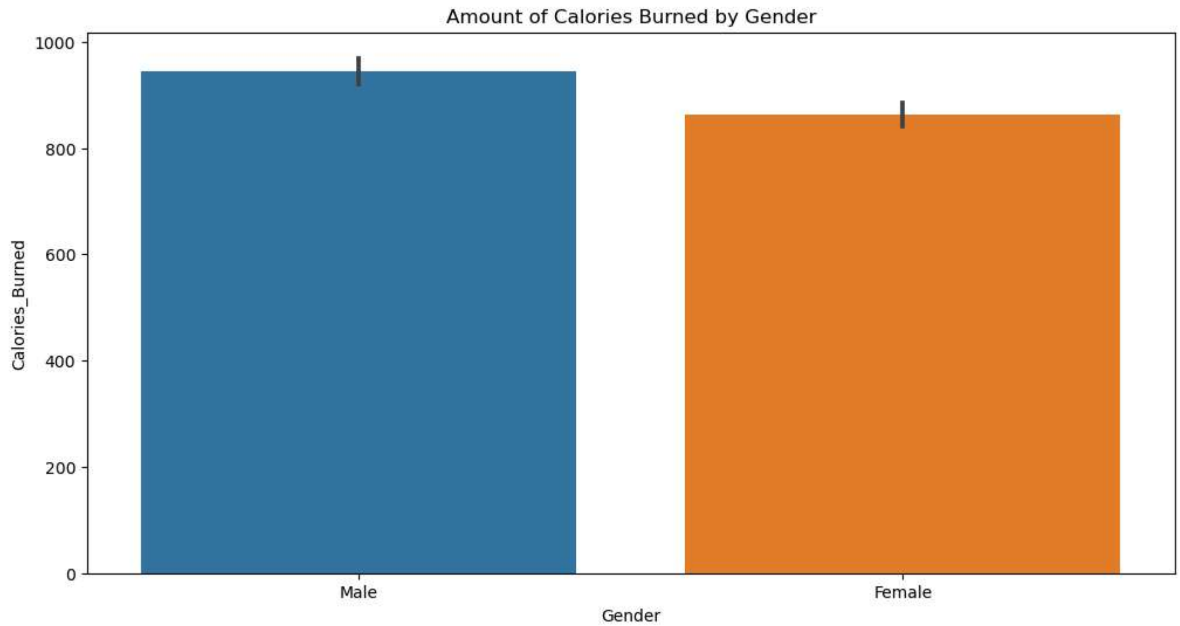
In [17]:
```
#Resting BPM by Gender
plt.figure(figsize=(12,6))
sns.boxplot(x='Gender',y='Resting_BPM',data=data)
plt.title('Resting BPM by Gender')
plt.show()
```



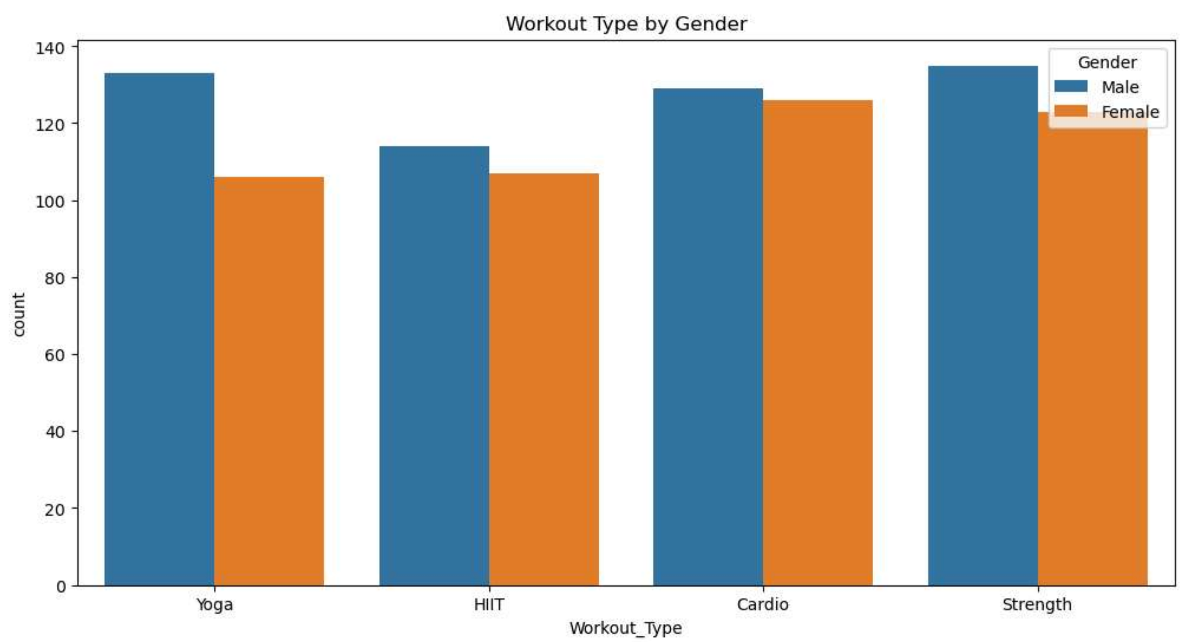
In [18]:
```
#Duration of Session by Gender
plt.figure(figsize=(12,6))
sns.barplot(x='Gender',y='Session_Duration (hours)',data=data)
plt.title('Duration of Session by Gender')
plt.show()
```



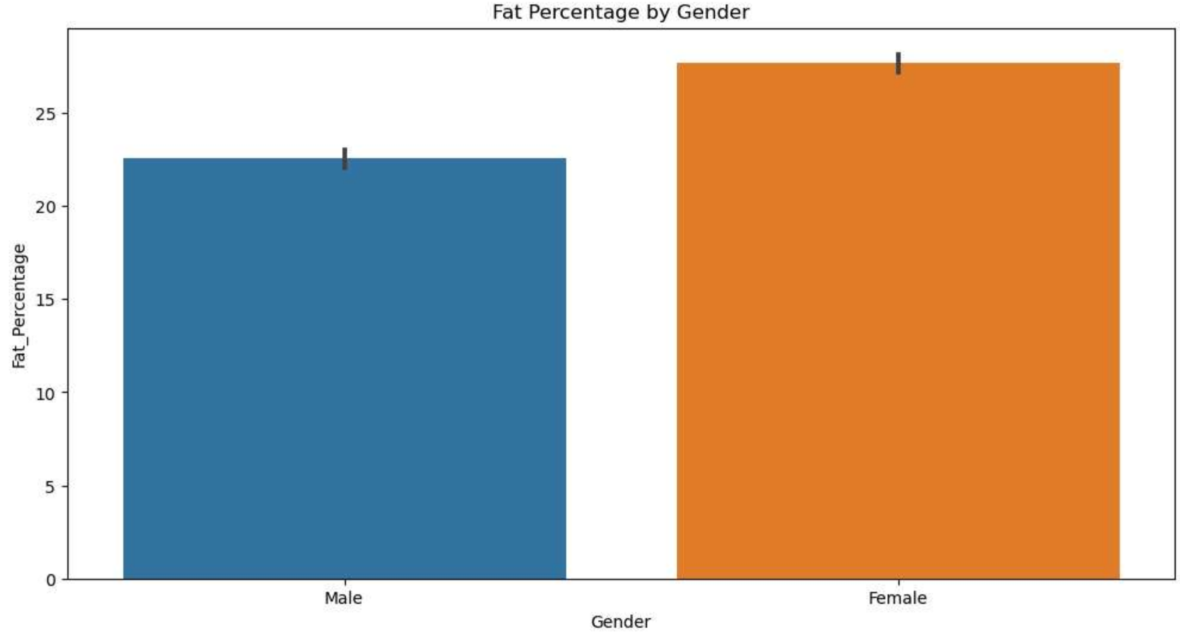
In [19]:
```
#Amount of Calories Burned by Gender
plt.figure(figsize=(12,6))
sns.barplot(x='Gender',y='Calories_Burned',data=data)
plt.title('Amount of Calories Burned by Gender')
plt.show()
```



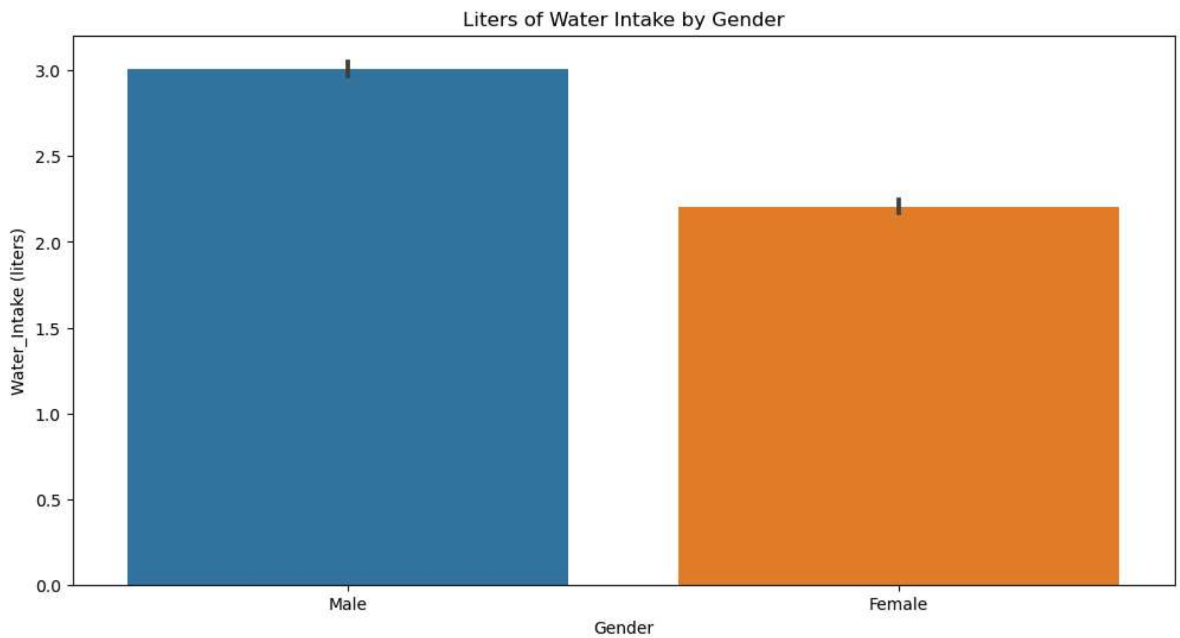
In [20]:
```
#Workout Type by Gender
plt.figure(figsize=(12,6))
sns.countplot(hue='Gender',x='Workout_Type',data=data)
plt.title('Workout Type by Gender')
plt.show()
```



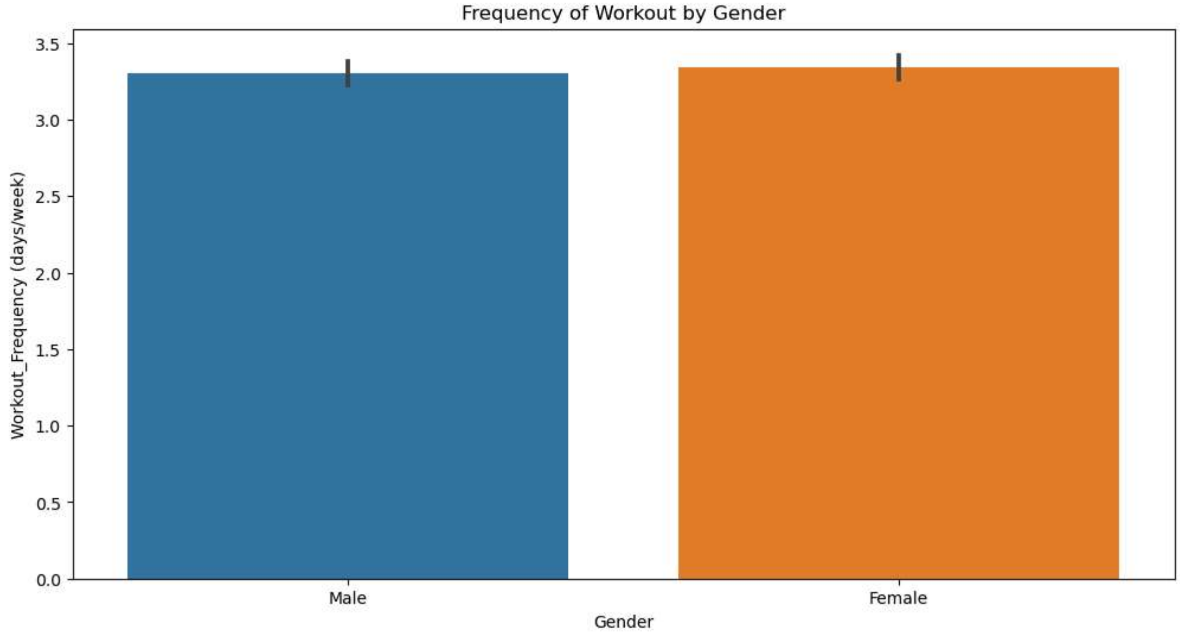
In [21]:
```
#Fat Percentage by Gender
plt.figure(figsize=(12,6))
sns.barplot(x='Gender',y='Fat_Percentage',data=data)
plt.title('Fat Percentage by Gender')
plt.show()
```



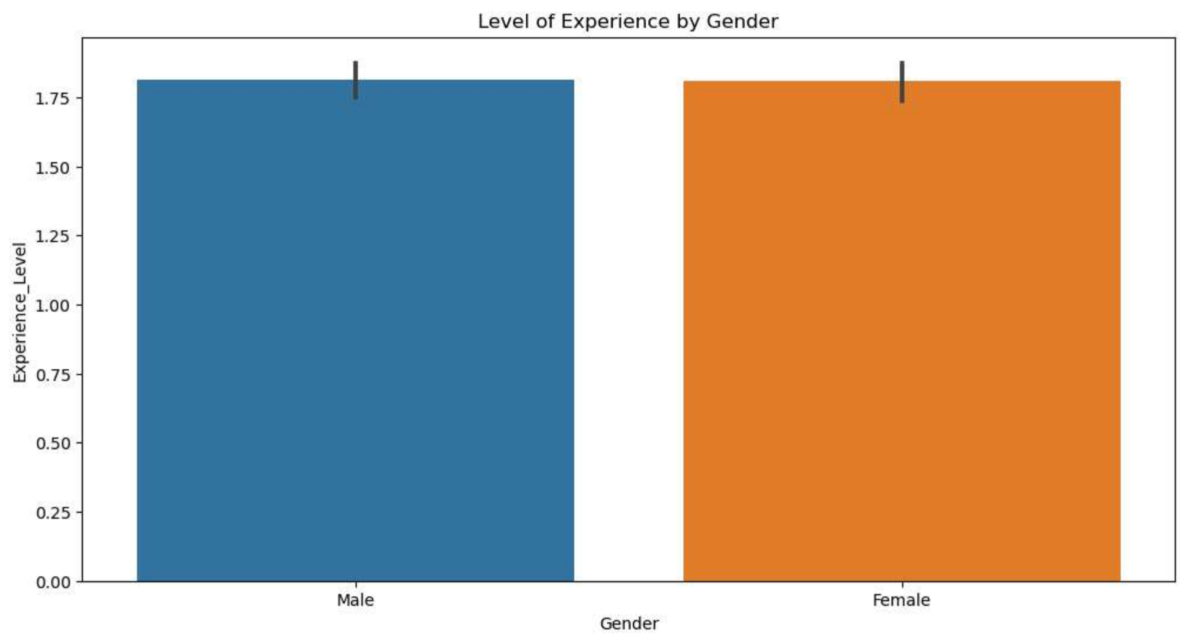
In [22]:
```
#Liters of Water Intake by Gender
plt.figure(figsize=(12,6))
sns.barplot(x='Gender',y='Water_Intake (liters)',data=data)
plt.title('Liters of Water Intake by Gender')
plt.show()
```



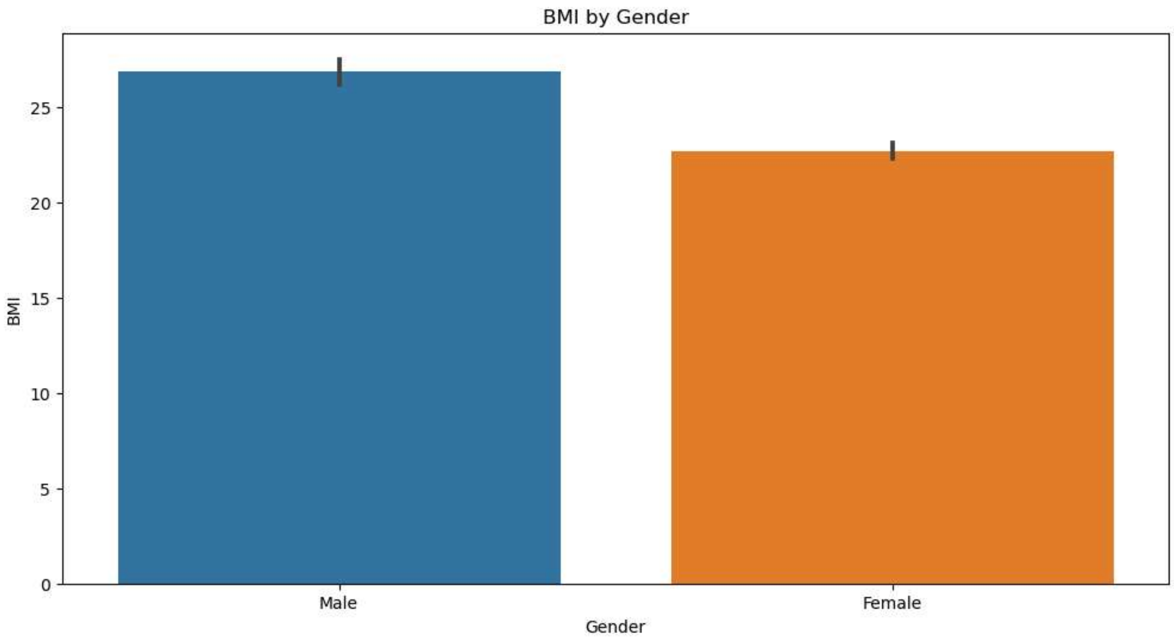
In [23]:
```
#Frequency of Workout by Gender
plt.figure(figsize=(12,6))
sns.barplot(x='Gender',y='Workout_Frequency (days/week)',data=data)
plt.title('Frequency of Workout by Gender')
plt.show()
```



In [24]:
```
#Level of Experience by Gender
plt.figure(figsize=(12,6))
sns.barplot(x='Gender',y='Experience_Level',data=data)
plt.title('Level of Experience by Gender')
plt.show()
```



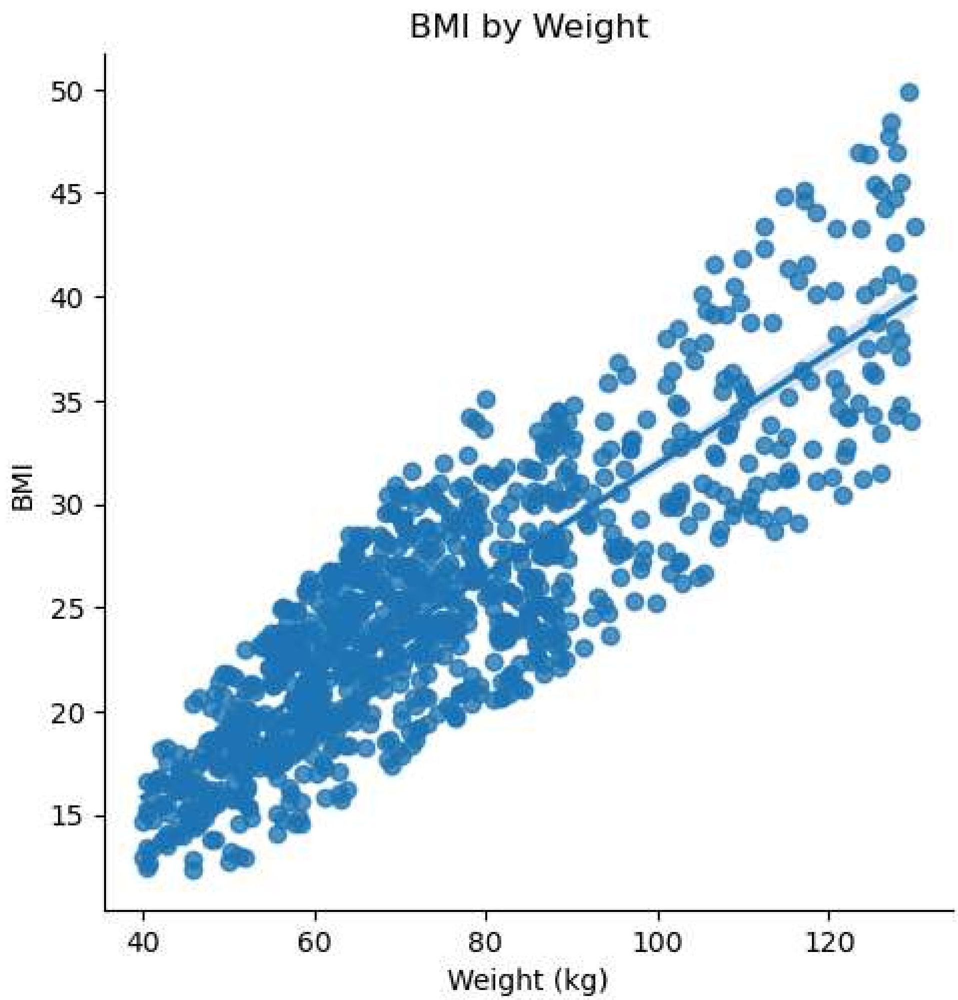
In [25]:
```
#BMI by Gender
plt.figure(figsize=(12,6))
sns.barplot(x='Gender',y='BMI',data=data)
plt.title('BMI by Gender')
plt.show()
```



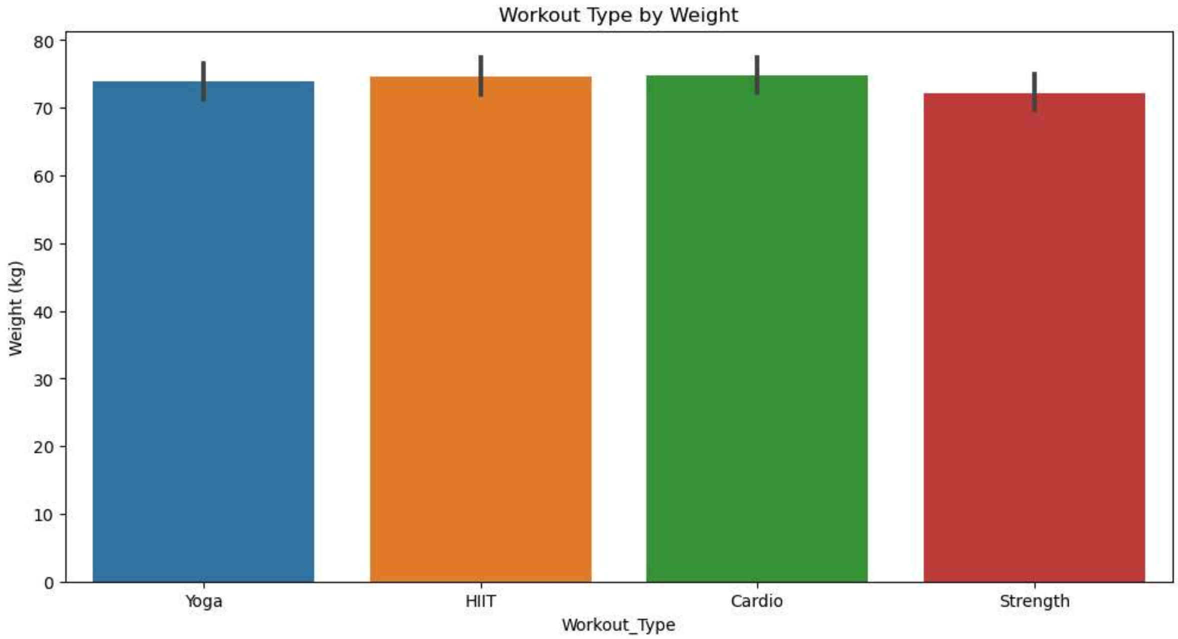
In [26]:
```
#BMI by Weight
plt.figure(figsize=(12,6))
sns.lmplot(x='Weight (kg)',y='BMI',data=data)
plt.title('BMI by Weight')
plt.show()
```

```
C:\Users\hp\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning:
The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

```
<Figure size 1200x600 with 0 Axes>
```



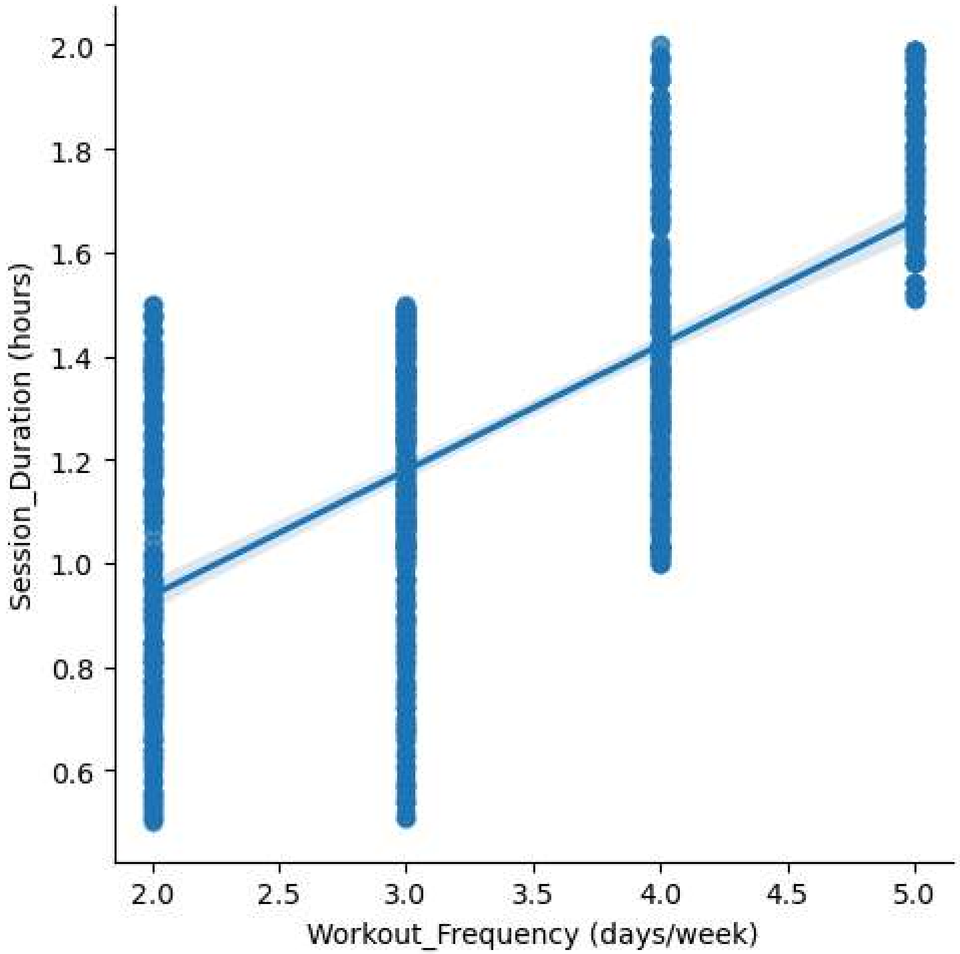
In [27]:
```
#Workout Type by Weight
plt.figure(figsize=(12,6))
sns.barplot(y='Weight (kg)',x='Workout_Type',data=data)
plt.title('Workout Type by Weight')
plt.show()
```



In [28]:
```
plt.figure(figsize=(12,6))
sns.lmplot(x='Workout_Frequency (days/week)',y='Session_Duration (hours)',data
plt.show()
```

```
C:\Users\hp\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning:
The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

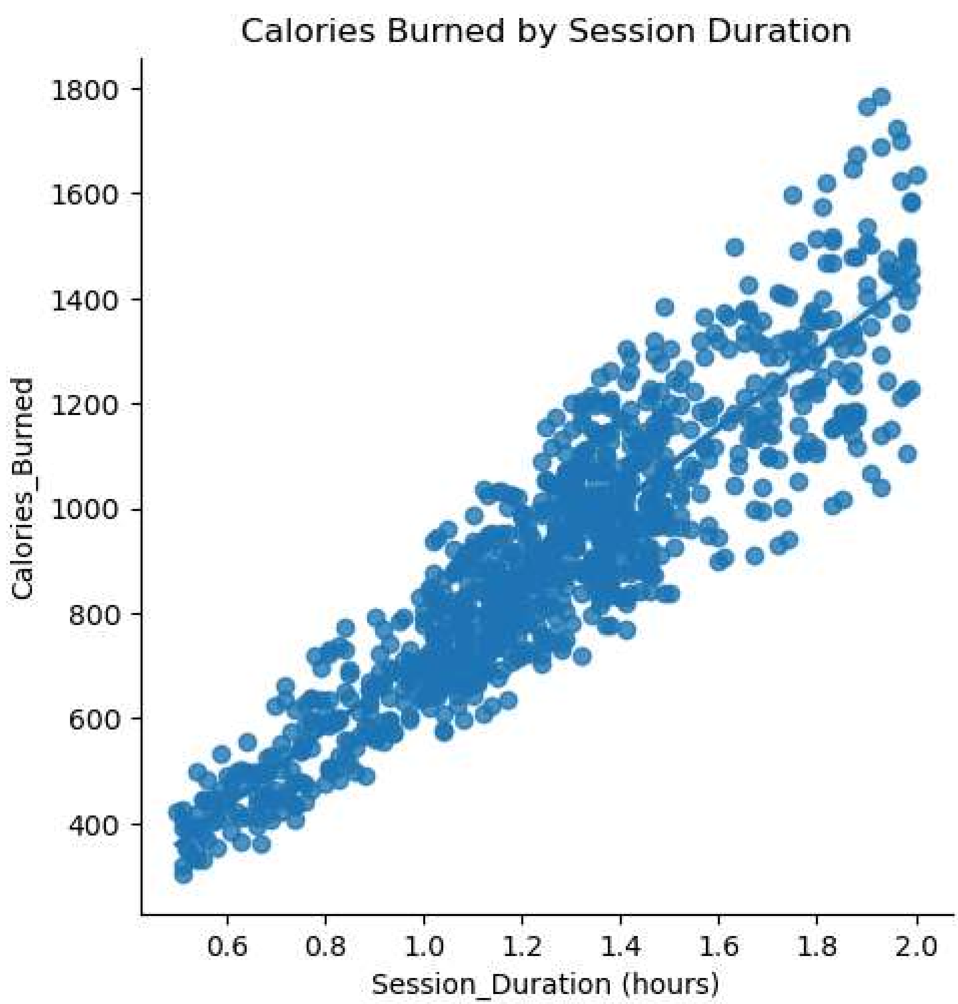
```
<Figure size 1200x600 with 0 Axes>
```



In [29]:
```
plt.figure(figsize=(12,6))
sns.lmplot(y='Calories_Burned',x='Session_Duration (hours)',data=data)
plt.title('Calories Burned by Session Duration')
plt.show()
```

```
C:\Users\hp\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning:
The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

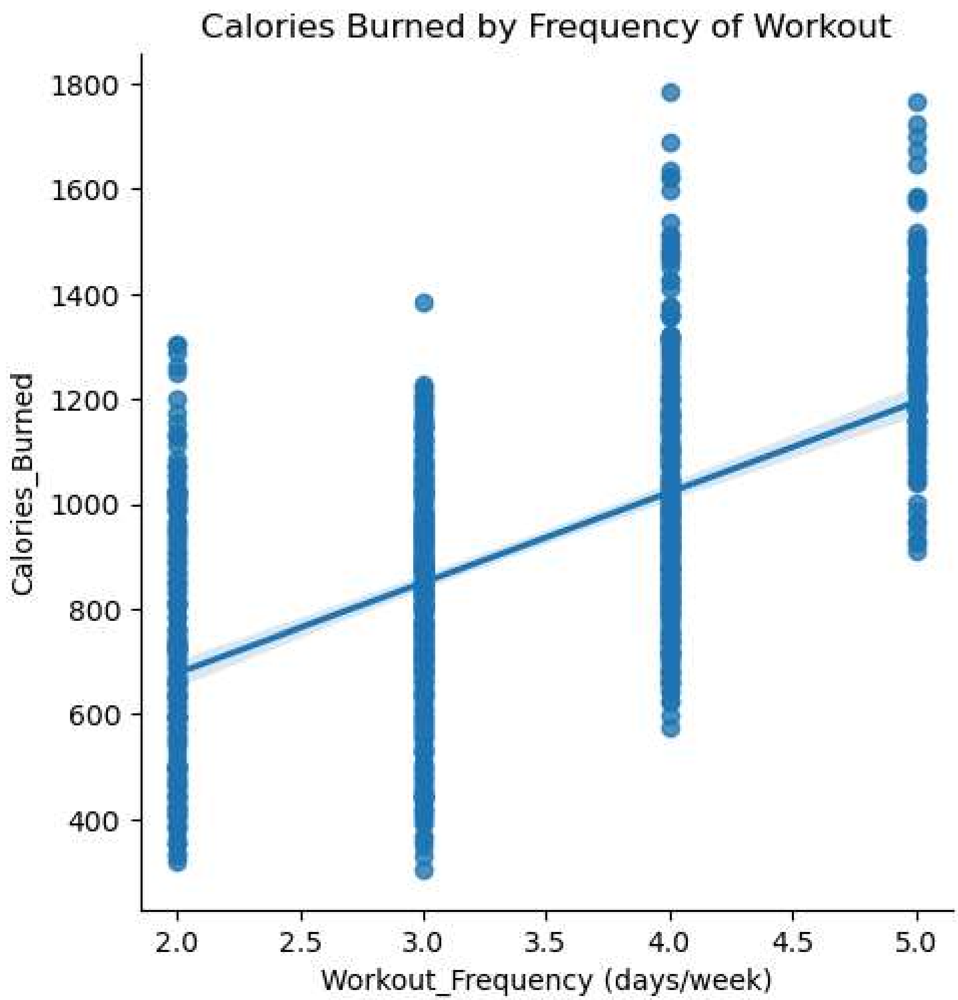
```
<Figure size 1200x600 with 0 Axes>
```



In [30]:
```
#Calories Burned by Frequency of Workout
plt.figure(figsize=(12,6))
sns.lmplot(x='Workout_Frequency (days/week)',y='Calories_Burned',data=data)
plt.title('Calories Burned by Frequency of Workout')
plt.show()
```

```
C:\Users\hp\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning:
The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

```
<Figure size 1200x600 with 0 Axes>
```



In [31]:
```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
warnings.simplefilter("ignore", category=pd.errors.SettingWithCopyWarning)
```

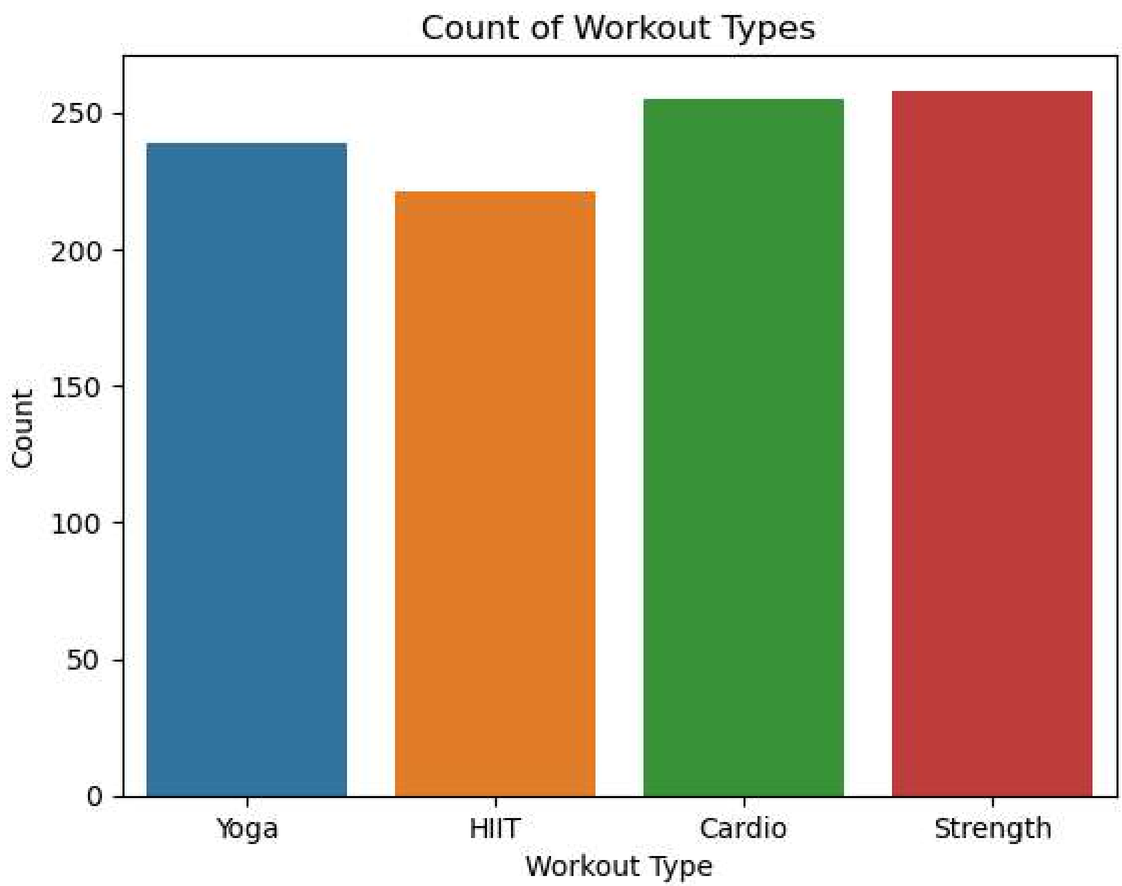
Comparing data between the Overweight and Healthy class and suggesting some improvements to reduce weigh

WHAT KIND OF TRAINING DO THEY MOSTLY DO?

In [32]:
```
# Create a countplot to visualize the count of categories in 'Category' column
sns.countplot(x='Workout_Type', data=data)

# Set the title and labels
plt.title('Count of Workout Types')
plt.xlabel('Workout Type')
plt.ylabel('Count')

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

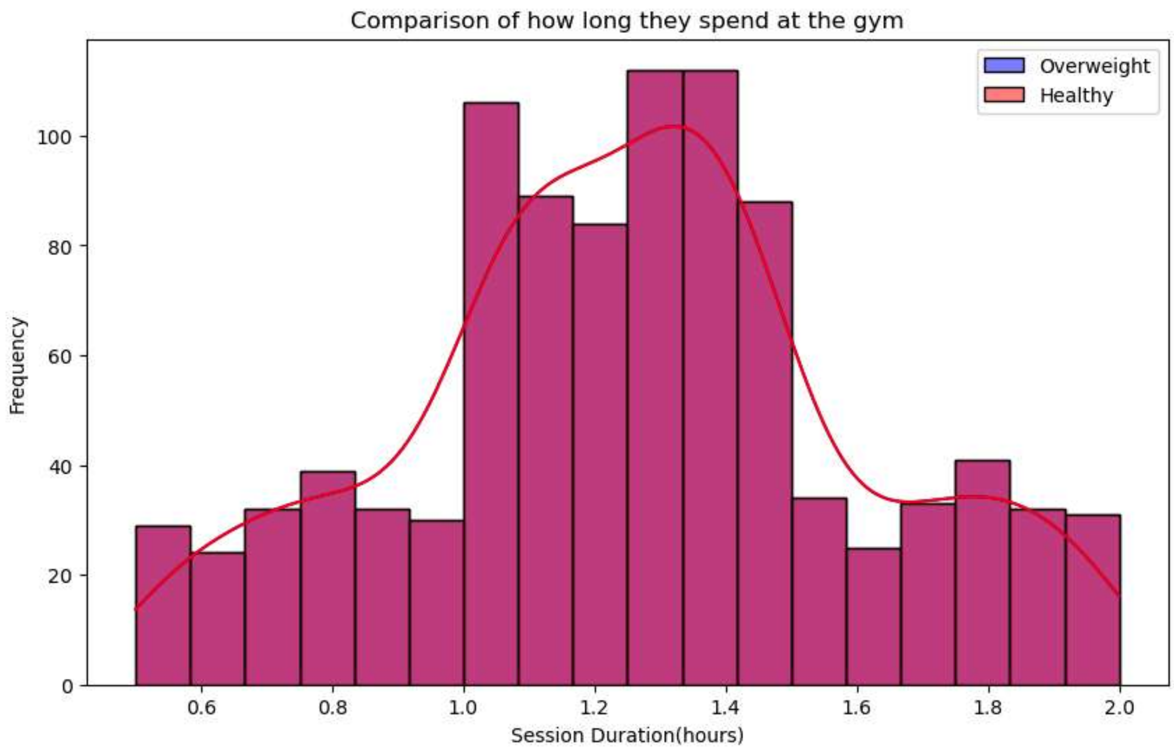


HOW LONG DO THEY SPEND THEIR TIME AT THE GYM?

In [33]:
```
plt.figure(figsize=(10, 6))
sns.histplot(data['Session_Duration (hours)'], color='blue', label='Overweight
sns.histplot(data['Session_Duration (hours)'], color='red', label='Healthy', k

# Adding labels and title
plt.title('Comparison of how long they spend at the gym')
plt.xlabel('Session Duration(hours)')
plt.ylabel('Frequency')
plt.legend()

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



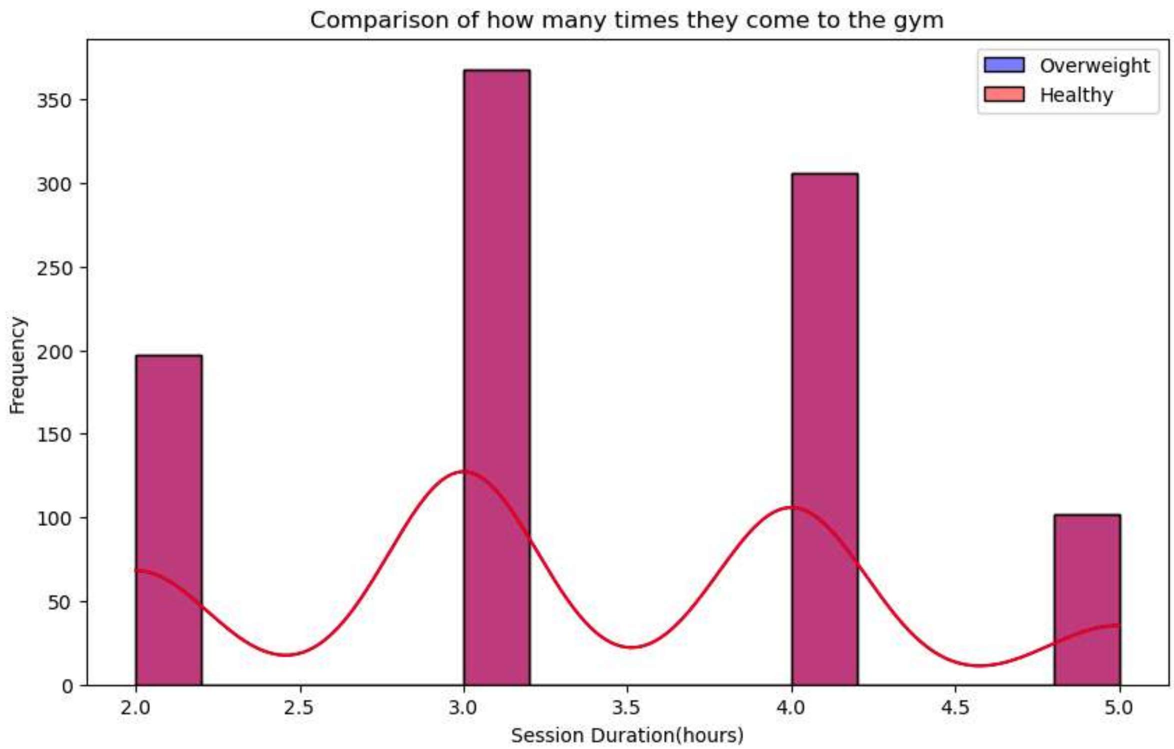
In [34]:
```
#The healthier ones seems to be spending more hours at the gym than their over
```

HOW MANY TIMES IN A WEEK DO THEY GO TO THE GYM?

In [35]:
```
# Plotting the histograms
plt.figure(figsize=(10, 6))
sns.histplot(data['Workout_Frequency (days/week)'], color='blue', label='Overw
sns.histplot(data['Workout_Frequency (days/week)'], color='red', label='Health

# Adding labels and title
plt.title('Comparison of how many times they come to the gym')
plt.xlabel('Session Duration(hours)')
plt.ylabel('Frequency')
plt.legend()

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



WHAT IS THEIR AVERAGE EXPERIENCE LEVEL?

In [36]:
```
print("Overweight:",data['Experience_Level'].mean(),"Healthy:",data['Experienc
```

```
Overweight: 1.8098663926002057 Healthy: 1.8098663926002057
```

In [37]:
```
numeric_data = data.select_dtypes(include='number')
```

handling outliers

In [38]:
```
Q1 = data['Calories_Burned'].quantile(0.25)
Q3 = data['Calories_Burned'].quantile(0.75)
IQR = Q3-Q1

min = Q1 - 1.5* IQR
MAX = Q3 + 1.5* IQR

data['Calories_Burned'] = np.where(data['Calories_Burned'] < min, min, data['C
data['Calories_Burned'] = np.where(data['Calories_Burned'] > MAX, MAX, data['C
```

In [39]:
```
Q1 = data['BMI'].quantile(0.25)
Q3 = data['BMI'].quantile(0.75)
IQR = Q3-Q1

min = Q1 - 1.5* IQR
MAX = Q3 + 1.5* IQR

data['BMI'] = np.where(data['BMI'] < min, min, data['BMI'])
data['BMI'] = np.where(data['BMI'] > MAX, MAX, data['BMI'])
```

Corelation

In [40]:
```
data['Gender'].value_counts()
```

Out[40]:
```
Gender
Male      511
Female    462
Name: count, dtype: int64
```

In [41]:
```
changes = {'Male': 0, 'Female':1}
```

In [42]:
```
data['Gender'] = data['Gender'].map(changes)
data['Gender'].head()
```

Out[42]:
```
0    0
1    1
2    1
3    0
4    0
Name: Gender, dtype: int64
```

In [43]:
```
data['Workout_Type'].value_counts()
```

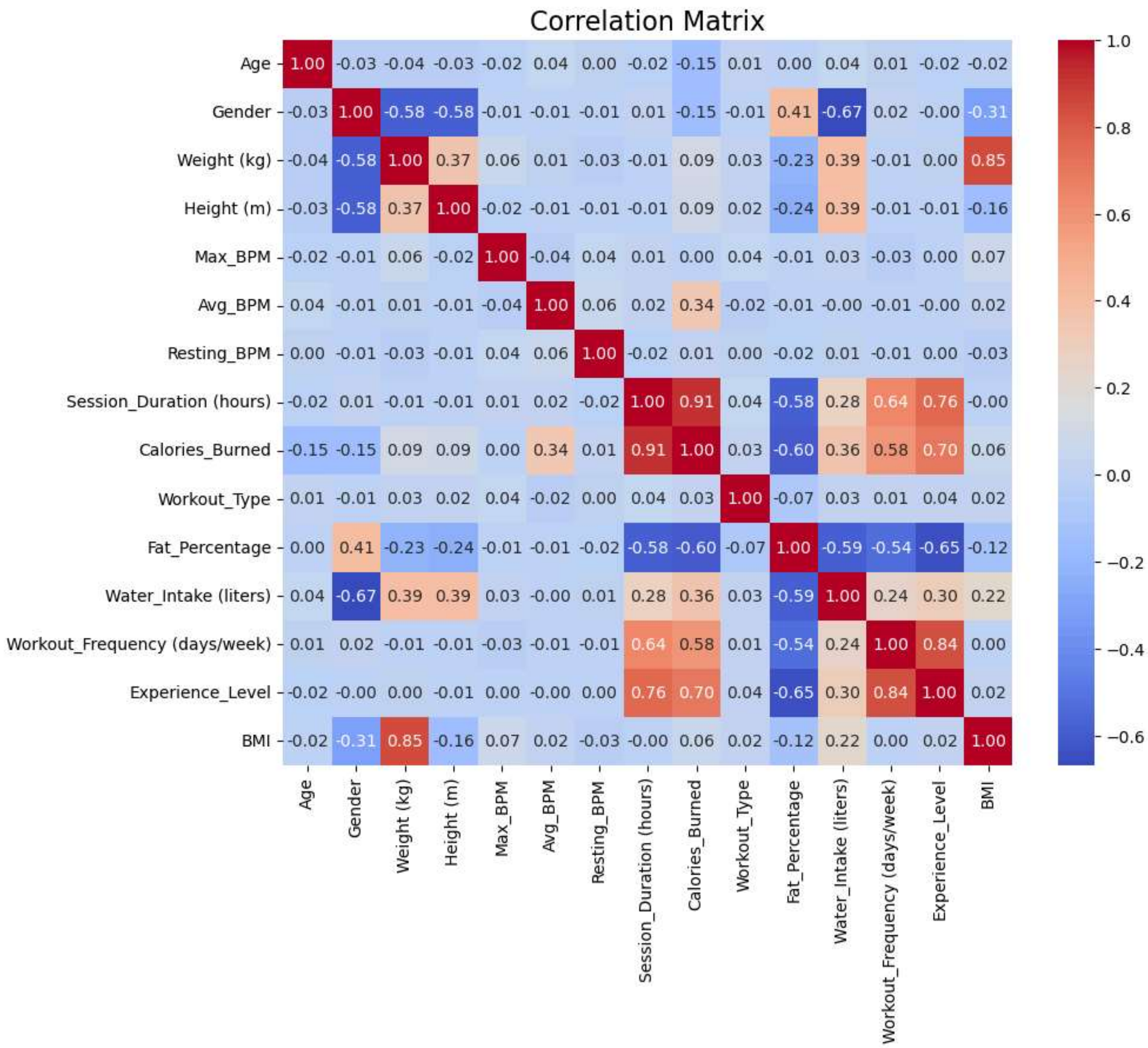
Out[43]:
```
Workout_Type
Strength    258
Cardio      255
Yoga        239
HIIT        221
Name: count, dtype: int64
```

In [44]:
```
changes2 = {'Strength': 0, 'Cardio':1, 'Yoga':2, 'HIIT':3}
```

In [45]:
```
data['Workout_Type'] = data['Workout_Type'].map(changes2)
data['Workout_Type'].head()
```

Out[45]:
```
0    2
1    3
2    1
3    0
4    0
Name: Workout_Type, dtype: int64
```

In [46]:
```
correlation_matrix = data.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix', fontsize=16)
plt.show()
```



Models

In [47]:
```
threshold = 0.03
correlation = data.corr()
high_corr_features = correlation.index[abs(correlation['Calories_Burned']) > t
high_corr_features.remove('Calories_Burned')
print('Selected features based on corrolation with target:')
print(high_corr_features)
X_selected = data[high_corr_features]
y = data['Calories_Burned']
```

```
Selected features based on corrolation with target:
['Age', 'Gender', 'Weight (kg)', 'Height (m)', 'Avg_BPM', 'Session_Duration
(hours)', 'Fat_Percentage', 'Water_Intake (liters)', 'Workout_Frequency (day
s/week)', 'Experience_Level', 'BMI']
```

In [48]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_rep
import warnings
warnings.filterwarnings('ignore')
from sklearn.preprocessing import MinMaxScaler
```

In [49]:
```
scaler = MinMaxScaler()
X = scaler.fit_transform(X_selected)
```

In [50]:
```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
```

In [51]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.1, shuff
```

In [52]:
```
model = LogisticRegression()
```

In [53]:
```
X_train.shape
```

Out[53]:
```
(875, 11)
```

In [54]:
```
model.fit(X_train, Y_train)
```

Out[54]:
```
LogisticRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [55]:
```
model.score(X_train, Y_train)
```

Out[55]:
```
0.04342857142857143
```

In [56]:
```

y_pred = model.predict(X_test)
```

In [57]:
```


print(accuracy_score(y_pred, Y_test))
```

```
0.01020408163265306
```

In [63]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Selected features based on correlation analysis
selected_features = [
    'Session_Duration (hours)', 'Experience_Level', 'Workout_Frequency (days/w
    'Water_Intake (liters)', 'Avg_BPM', 'Fat_Percentage', 'Gender'
]

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando

# Scale features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Initialize and train Random Forest Regressor
rf_regressor = RandomForestRegressor(random_state=42, n_estimators=100)
rf_regressor.fit(X_train_scaled, y_train)

# Predictions
y_train_pred = rf_regressor.predict(X_train_scaled)
y_test_pred = rf_regressor.predict(X_test_scaled)

# Evaluate the model
train_r2 = r2_score(y_train, y_train_pred)
test_r2 = r2_score(y_test, y_test_pred)
train_mse = mean_squared_error(y_train, y_train_pred)
test_mse = mean_squared_error(y_test, y_test_pred)
train_mae = mean_absolute_error(y_train, y_train_pred)
test_mae = mean_absolute_error(y_test, y_test_pred)

print("Training R^2:", train_r2)
print("Testing R^2:", test_r2)
print("Training MSE:", train_mse)
print("Testing MSE:", test_mse)
print("Training MAE:", train_mae)
print("Testing MAE:", test_mae)
```

```
Training R^2: 0.9966591492479834
Testing R^2: 0.9761296949544783
Training MSE: 236.55093097686378
Testing MSE: 1957.5072374358974
Training MAE: 11.756645244215939
Testing MAE: 34.138615384615385
```

In [ ]:

In [ ]: