In [1]:

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

In [2]:

```
calories_data=pd.read_csv('E:/calories.csv')
calories_data.head(3)
```

Out[2]:

	User_ID	Calories
0	14733363	231.0
1	14861698	66.0
2	11179863	26.0

In [3]:

```
exercise_data=pd.read_csv('E:/exercise.csv')
exercise_data.head(3)
```

Out[3]:

	User_ID	Gender	Age	Height	Weight	Duration	Heart_Rate	Body_Temp
0	14733363	male	68	190.0	94.0	29.0	105.0	40.8
1	14861698	female	20	166.0	60.0	14.0	94.0	40.3
2	11179863	male	69	179.0	79.0	5.0	88.0	38.7

In [4]:

```
new_df=exercise_data.merge(calories_data,on='User_ID')
new_df.head(3)
```

Out[4]:

	User_ID	Gender	Age	Height	Weight	Duration	Heart_Rate	Body_Temp	Calories
0	14733363	male	68	190.0	94.0	29.0	105.0	40.8	231.0
1	14861698	female	20	166.0	60.0	14.0	94.0	40.3	66.0
2	11179863	male	69	179.0	79.0	5.0	88.0	38.7	26.0

In [5]:

```
new_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15000 entries, 0 to 14999
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   User_ID     15000 non-null  int64  
 1   Gender      15000 non-null  object 
 2   Age         15000 non-null  int64  
 3   Height      15000 non-null  float64
 4   Weight      15000 non-null  float64
 5   Duration    15000 non-null  float64
 6   Heart_Rate  15000 non-null  float64
 7   Body_Temp   15000 non-null  float64
 8   Calories    15000 non-null  float64
dtypes: float64(6), int64(2), object(1)
memory usage: 1.1+ MB
```

In [6]:

```
new_df.isna().sum()
```

Out[6]:

```
User_ID       0
Gender        0
Age           0
Height        0
Weight        0
Duration      0
Heart_Rate    0
Body_Temp     0
Calories      0
dtype: int64
```

In [7]:

```
new_df.describe()
```

Out[7]:

	User_ID	Age	Height	Weight	Duration	Heart_Rate	Body_Temp	Calories
count	1.500000e+04	15000.000000	15000.000000	15000.000000	15000.000000	15000.000000	15000.000000	15000.000000
mean	1.497736e+07	42.789800	174.465133	74.966867	15.530600	95.518533	40.025453	89.539533
std	2.872851e+06	16.980264	14.258114	15.035657	8.319203	9.583328	0.779230	62.456978
min	1.000116e+07	20.000000	123.000000	36.000000	1.000000	67.000000	37.100000	1.000000
25%	1.247419e+07	28.000000	164.000000	63.000000	8.000000	88.000000	39.600000	35.000000
50%	1.499728e+07	39.000000	175.000000	74.000000	16.000000	96.000000	40.200000	79.000000
75%	1.744928e+07	56.000000	185.000000	87.000000	23.000000	103.000000	40.600000	138.000000
max	1.999965e+07	79.000000	222.000000	132.000000	30.000000	128.000000	41.500000	314.000000

In [8]:

```
new_df.drop('User_ID',axis=1,inplace=True)
```

In [9]:

```
new_df.head(3)
```

Out[9]:

	Gender	Age	Height	Weight	Duration	Heart_Rate	Body_Temp	Calories
0	male	68	190.0	94.0	29.0	105.0	40.8	231.0
1	female	20	166.0	60.0	14.0	94.0	40.3	66.0
2	male	69	179.0	79.0	5.0	88.0	38.7	26.0

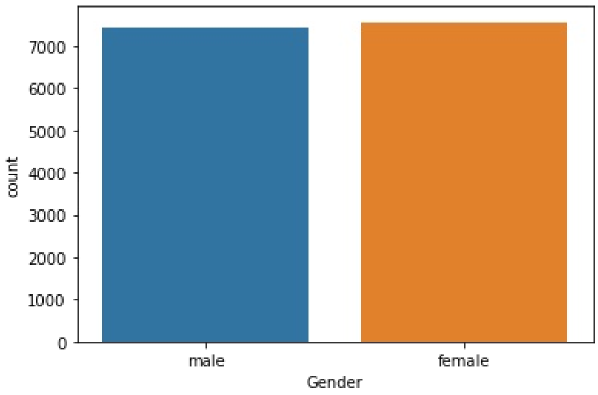
In [10]:

```
#Checking how many males and females are there

sns.countplot(new_df.Gender)

#its giving equal distribution for both
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable
as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other argu
ments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[10]:

```
<AxesSubplot:xlabel='Gender', ylabel='count'>
```



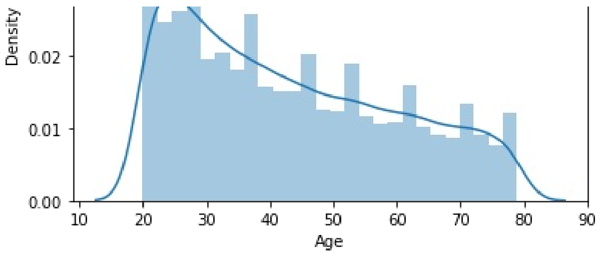
In [11]:

```
#finding the distribution of "Age" column and

sns.distplot(new_df.Age)
#as age increases less people comes to gym
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecat
ed function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-lev
el function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[11]:

```
<AxesSubplot:xlabel='Age', ylabel='Density'>
```



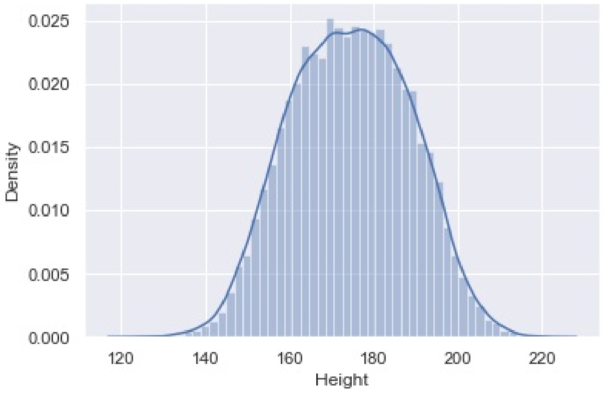
In [12]:

```
sns.set()
sns.distplot(new_df['Height'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecat
ed function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-lev
el function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[12]:

```
<AxesSubplot:xlabel='Height', ylabel='Density'>
```



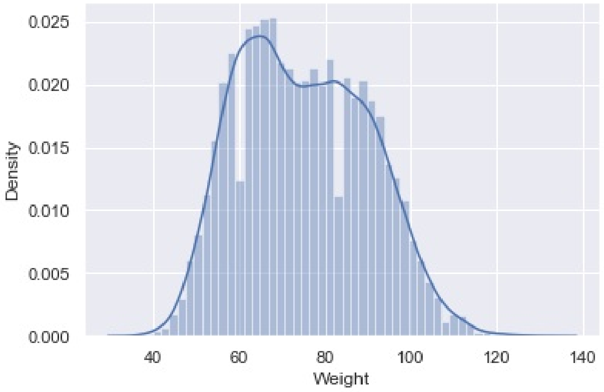
In [13]:

```
sns.distplot(new_df.Weight)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecat
ed function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-lev
el function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[13]:

```
<AxesSubplot:xlabel='Weight', ylabel='Density'>
```



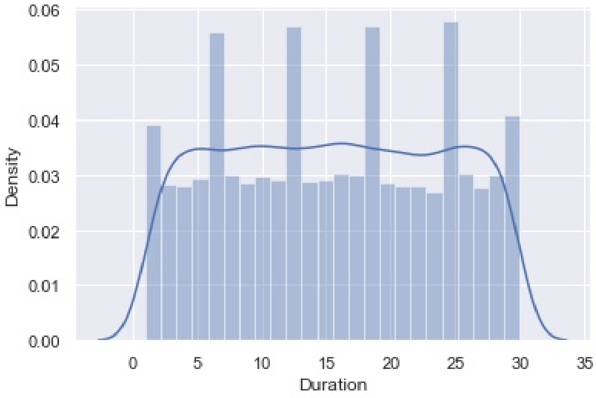
In [14]:

```
sns.distplot(new_df.Duration)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecat
ed function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-lev
el function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[14]:

```
<AxesSubplot:xlabel='Duration', ylabel='Density'>
```



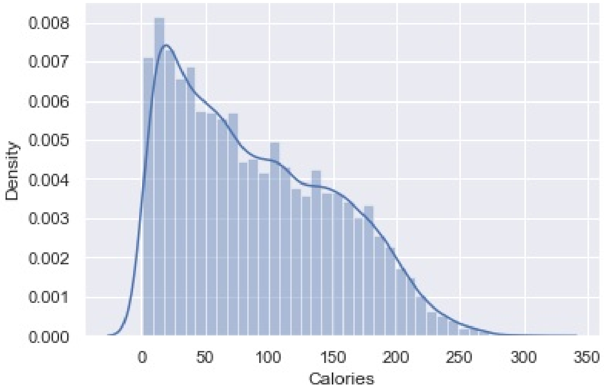
In [15]:

```
sns.distplot(new_df.Calories)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecat
ed function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-lev
el function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[15]:

```
<AxesSubplot:xlabel='Calories', ylabel='Density'>
```



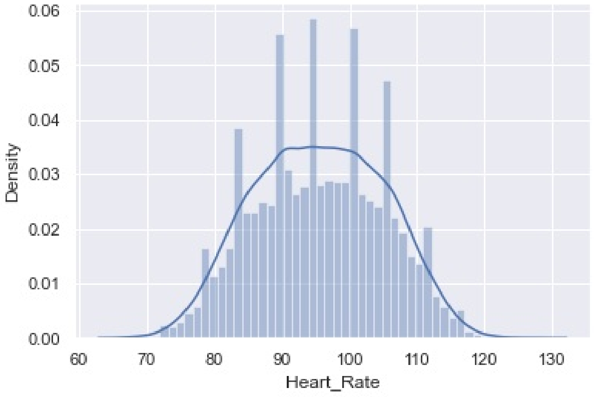
In [16]:

```
sns.distplot(new_df.Heart_Rate)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecat
ed function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-lev
el function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[16]:

```
<AxesSubplot:xlabel='Heart_Rate', ylabel='Density'>
```



In [17]:

```
#Finding corelation to check how strongly data is related to each other
cor_realtion=new_df.corr()
```

In [18]:

```
cor_realtion
```

```
#we infer that duration and calories burnt are correlated and body_temp and calories and negatively corelated sam
```

Out[18]:

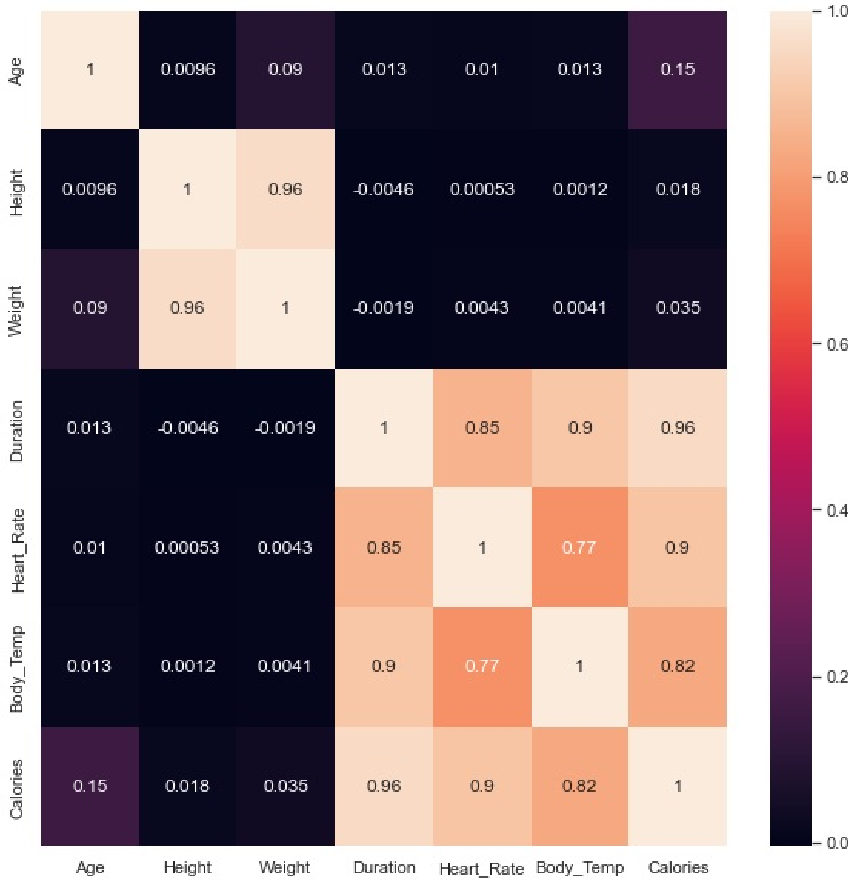
	Age	Height	Weight	Duration	Heart_Rate	Body_Temp	Calories
Age	1.000000	0.009554	0.090094	0.013247	0.010482	0.013175	0.154395
Height	0.009554	1.000000	0.958451	-0.004625	0.000528	0.001200	0.017537
Weight	0.090094	0.958451	1.000000	-0.001884	0.004311	0.004095	0.035481
Duration	0.013247	-0.004625	-0.001884	1.000000	0.852869	0.903167	0.955421
Heart_Rate	0.010482	0.000528	0.004311	0.852869	1.000000	0.771529	0.897882
Body_Temp	0.013175	0.001200	0.004095	0.903167	0.771529	1.000000	0.824558
Calories	0.154395	0.017537	0.035481	0.955421	0.897882	0.824558	1.000000

In [19]:

```
#Building heatmap for corelation
plt.figure(figsize=(10,10))
sns.heatmap(data=cor_realtion,annot=True)
#we infer that if values are high means positively corelated or else negatively
```

Out[19]:

```
<AxesSubplot:>
```



In [20]:

```
new_df.replace({"Gender":{'male':0,'female':1}}, inplace=True)
```

In [21]:

```
new_df.head()
```

Out[21]:

	Gender	Age	Height	Weight	Duration	Heart_Rate	Body_Temp	Calories
0	0	68	190.0	94.0	29.0	105.0	40.8	231.0
1	1	20	166.0	60.0	14.0	94.0	40.3	66.0
2	0	69	179.0	79.0	5.0	88.0	38.7	26.0
3	1	34	179.0	71.0	13.0	100.0	40.5	71.0
4	1	27	154.0	58.0	10.0	81.0	39.8	35.0

In [22]:

```
from sklearn.model_selection import train_test_split
```

In [23]:

```
X=new_df.drop('Calories',axis=1)
y=new_df.Calories
```

In [24]:

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=0)
```

In [25]:

```
new_df.shape
```

Out[25]:

```
(15000, 8)
```

In [26]:

```
len(X_train)
```

Out[26]:

```
12000
```

In [27]:

```
len(y_train)
```

Out[27]:

```
12000
```

MODEL-1 : LINEAR REGRESSION

In [28]:

```
from sklearn.linear_model import LinearRegression
```

In [29]:

```
reg=LinearRegression()
```

In [30]:

```
reg.fit(X_train,y_train)
```

Out[30]:

```
LinearRegression()
```

In [31]:

```
reg.score(X_test,y_test)
```

Out[31]:

```
0.969214323020104
```

In [33]:

```
y_predicted=reg.predict(X_test)
```

In [34]:

```
y_predicted
```

Out[34]:

```
array([ 37.87701502,   4.06170735, 110.99156716, ...,  28.10508645,
       150.91974893, 146.45426893])
```

In [35]:

```
from sklearn.metrics import mean_absolute_error,mean_squared_error
```

Accuracy score is only for classification problems. For regression problems you can use: R2 Score, MSE (Mean Squared Error), RMSE (Root Mean Squared Error), so we cant use confusion matrix in Regression Problem

In [36]:

```
mae = mean_absolute_error(y_test,y_predicted)
mae
```

Out[36]:

```
8.090679636313151
```

In [37]:

```
mse=mean_squared_error(y_test,y_predicted)
mse
```

Out[37]:

```
118.79074609385707
```

In [56]:

```
new_df.columns
```

Out[56]:

```
Index(['Gender', 'Age', 'Height', 'Weight', 'Duration', 'Heart_Rate',
       'Body_Temp', 'Calories'],
      dtype='object')
```

START YOUR PREDICTION

In [59]:

```
Gender=int(input('Enter Gender: ')) # 0 for male and 1 for female
Age=int(input('Enter Age: '))
Height=float(input('Enter Height: '))
Weight=float(input('Enter Weight: '))
Duration=float(input('Enter Duration: '))
Heart_Rate=float(input('Enter Heart-Rate: '))
Body_Temp=float(input('Enter Body-Temp: '))
X_array = ([[Gender,Age,Height,Weight,Duration,Heart_Rate,Body_Temp]])
y_pred = reg.predict(X_array)
print(y_pred)
```

```
Enter Gender: 0
Enter Age: 34
Enter Height: 222
Enter Weight: 132
Enter Duration: 30
Enter Heart-Rate: 128
Enter Body-Temp: 41.5
[229.3258177]
```

MODEL-2 : XGBOOST

In [38]:

```
!pip install xgboost
from xgboost import XGBRegressor
```

```
Requirement already satisfied: xgboost in c:\programdata\anaconda3\lib\site-packages (1.5.2)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-packages (from xgboost) (1.20.3)
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-packages (from xgboost) (1.7.1)
```

XGBoost is a powerful approach for building supervised regression models. The validity of this statement can be inferred by knowing about its (XGBoost) objective function and base learners.

The objective function contains loss function and a regularization term. It tells about the difference between actual values and predicted values, i.e how far the model results are from the real values. The most common loss functions in XGBoost for regression problems is reg:linear, and that for binary classification is reg:logistics.

XGBoost is one of the ensemble learning methods.

In [39]:

```
model = XGBRegressor()
```

In [40]:

```
model.fit(X_train,y_train)
```

Out[40]:

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
             gamma=0, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.300000012,
             max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=100, n_jobs=8,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
             validate_parameters=1, verbosity=None)
```

In [41]:

```
model.score(X_test,y_test)
```

Out[41]:

```
0.998801753229742
```

In [42]:

```
y_predicted=model.predict(X_test)
```

In [43]:

```
mae = mean_absolute_error(y_test,y_predicted)
mae
```

Out[43]:

```
1.511337571144104
```

In [44]:

```
mse=mean_squared_error(y_test,y_predicted)
mse
```

Out[44]:

```
4.623599082666398
```

START YOUR PREDICTION

In [60]:

```
Gender=int(input('Enter Gender: ')) # 0 for male and 1 for female
Age=int(input('Enter Age: '))
Height=float(input('Enter Height: '))
Weight=float(input('Enter Weight: '))
Duration=float(input('Enter Duration: '))
Heart_Rate=float(input('Enter Heart-Rate: '))
Body_Temp=float(input('Enter Body-Temp: '))
X_array = np.array([[Gender,Age,Height,Weight,Duration,Heart_Rate,Body_Temp]]).reshape(1,-1)
y_pred = model.predict(X_array)
y_pred
```

```
Enter Gender: 0
Enter Age: 34
Enter Height: 222
Enter Weight: 132
Enter Duration: 30
Enter Heart-Rate: 128
Enter Body-Temp: 41.5
```

Out[60]:

```
array([285.59732], dtype=float32)
```

In [ ]:

```

```

In [ ]:

```

```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js