In [37]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from xgboost import XGBRegressor
from sklearn import metrics
```

In [3]:
```
calories=pd.read_csv("C:\\Users\\Pranav\\Desktop\\DATA SCIENCE DATA\\CVC file\\calories\\calories.csv")
calories.head()
```

Out[3]:

	User_ID	Calories
0	14733363	231.0
1	14861698	66.0
2	11179863	26.0
3	16180408	71.0
4	17771927	35.0

In [5]:
```
exercises=pd.read_csv("C:\\Users\\Pranav\\Desktop\\DATA SCIENCE DATA\\CVC file\\calories\\exercise.csv")
exercises.head()
```

Out[5]:

	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
3	16180408	female	34	179.0	71.0	13.0	100.0	40.5
4	17771927	female	27	154.0	58.0	10.0	81.0	39.8

In [9]:
```
#Combining the two Dataframe
calories_data=pd.concat([exercises,calories['Calories']],axis=1)
calories_data.head()
```

Out[9]:

	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
3	16180408	female	34	179.0	71.0	13.0	100.0	40.5	71.0
4	17771927	female	27	154.0	58.0	10.0	81.0	39.8	35.0

In [10]:
```
#shape of data
calories_data.shape
```

Out[10]:
```
(15000, 9)
```

In [11]:
```
#information about data
calories_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 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: 1.0+ MB
```

In [15]:
```
## get some statistical measures about the data
calories_data.describe()
```

Out[15]:

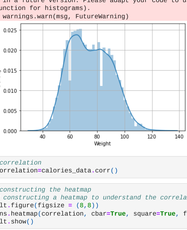
	User_ID	Age	Height	Weight	Duration	Heart_Rate	Body_Temp	Calories
count	1.500000e+04	15000.000000	15000.000000	15000.000000	15000.000000	15000.000000	15000.000000	15000.000000
mean	1.497736e+07	42.789800	174.465133	74.966867	15.530600	95.518533	40.025453	89.539533
std	2.872851e+06	16.980264	14.258114	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 [14]:
```
#find null value in dataset
calories_data.isnull().sum()
```

Out[14]:
```
User_ID       0
Gender        0
Age           0
Height        0
Weight        0
Duration      0
Heart_Rate    0
Body_Temp     0
Calories      0
dtype: int64
```

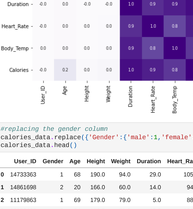
In [19]:
```
#count plot
sns.countplot(calories_data['Gender'])
plt.grid(True)
plt.show()
```

```
C:\Users\Pranav\Searches\hjhkh\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From vers
ion 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misin
terpretation.
  warnings.warn(
```



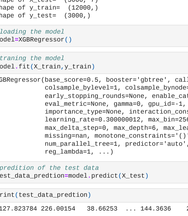
In [21]:
```
#finding the distribution of age column
sns.distplot(calories_data['Age'])
plt.grid(True)
plt.show()
```

```
C:\Users\Pranav\Searches\hjhkh\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be remove
d in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level
function for histograms).
  warnings.warn(msg, FutureWarning)
```



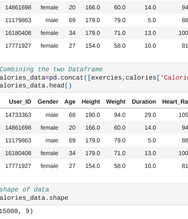
In [22]:
```
#finding the distribution of heighgt column
sns.distplot(calories_data['Height'])
plt.grid(True)
plt.show()
```

```
C:\Users\Pranav\Searches\hjhkh\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be remove
d in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level
function for histograms).
  warnings.warn(msg, FutureWarning)
```



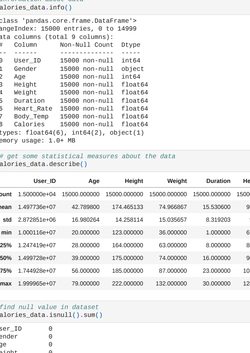
In [23]:
```
#finding the distribution of weight column
sns.distplot(calories_data['Weight'])
plt.grid(True)
plt.show()
```

```
C:\Users\Pranav\Searches\hjhkh\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be remove
d in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level
function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [24]:
```
#correlation
correlation=calories_data.corr()
```

In [25]:
```
#constructing the heatmap
# constructing a heatmap to understand the correlatiom
plt.figure(figsize = (8,8))
sns.heatmap(correlation, cbar=True, square=True, fmt='.1f',annot=True, annot_kws={'size':8}, cmap='Purples')
plt.show()
```



In [27]:
```
#replacing the gender column
calories_data.replace({'Gender':{'male':1,'female':2}},inplace=True)
calories_data.head()
```

Out[27]:

	User_ID	Gender	Age	Height	Weight	Duration	Heart_Rate	Body_Temp	Calories
0	14733363	1	68	190.0	94.0	29.0	105.0	40.8	231.0
1	14861698	2	20	166.0	60.0	14.0	94.0	40.3	66.0
2	11179863	1	69	179.0	79.0	5.0	88.0	38.7	26.0
3	16180408	2	34	179.0	71.0	13.0	100.0	40.5	71.0
4	17771927	2	27	154.0	58.0	10.0	81.0	39.8	35.0

In [29]:
```
#Separating features and Target
X=calories_data.drop(columns=['User_ID','Calories'],axis=1)
y=calories_data['Calories']
```

In [30]:
```
#Splitting into Training data and Test Data
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=2)
print("shape of X_train= ",X_train.shape)
print("shape of X_test= ",X_test.shape)
print("shape of y_train= ",y_train.shape)
print("shape of y_test= ",y_test.shape)
```

```
shape of X_train=  (12000, 7)
shape of X_test=  (3000, 7)
shape of y_train=  (12000,)
shape of y_test=  (3000,)
```

In [32]:
```
#loading the model
model=XGBRegressor()
```

Out[33]:
```
#traning the model
model.fit(X_train,y_train)
```

```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=100, n_jobs=0,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, ...)
```

In [34]:
```
#prediction of the test data
test_data_prediction=model.predict(X_test)
```

In [35]:
```
print(test_data_prediction)
```

```
[127.823784 226.00154   38.66253  ... 144.3636    22.767195  89.87375 ]
```

In [38]:
```
#mean absolute error
mae=metrics.mean_absolute_error(y_test,test_data_prediction)
```

In [39]:
```
print('Mean absolute error:',mae)
```

```
Mean absolute error: 1.4807048829992613
```

In [ ]: