```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from xgboost import XGBRegressor
from sklearn import metrics
```

```
calories = pd.read_csv("/cbpd.csv")
```

```
calories.head()
```

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

Next steps: Generate code with `calories` New interactive sheet

```
exercise_data = pd.read_csv('/exercise.csv')
```

```
exercise_data.head()
```

	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

Next steps: Generate code with `exercise_data` New interactive sheet

```
calories_data = pd.concat([exercise_data,calories['Calories']],axis=1)
```

```
calories_data.head()
```

	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

Next steps: Generate code with `calories_data` New interactive sheet

```
calories_data.shape
```

```
(15000, 9)
```

```
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
```

```python
calories_data.isnull().sum()
```

	0
User_ID	0
Gender	0
Age	0
Height	0
Weight	0
Duration	0
Heart_Rate	0
Body_Temp	0
Calories	0

dtype: int64

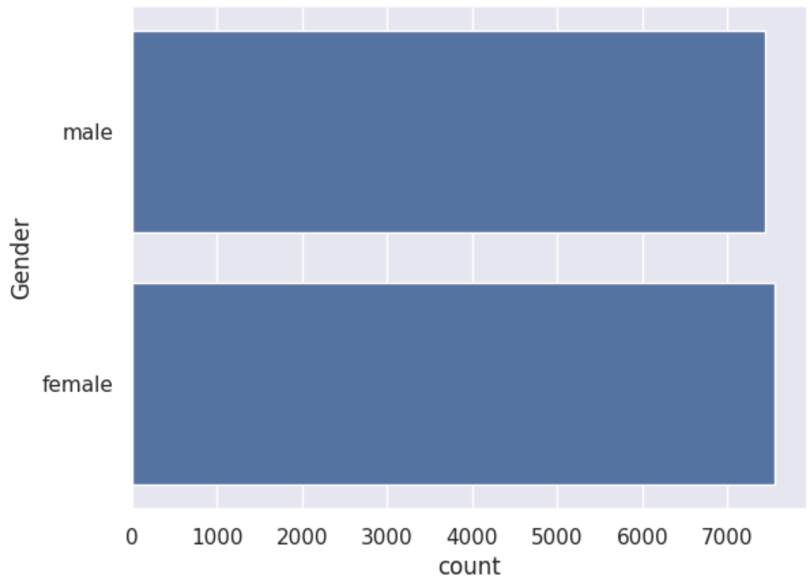
```python
calories_data.describe()
```

	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

```python
sns.set()
```

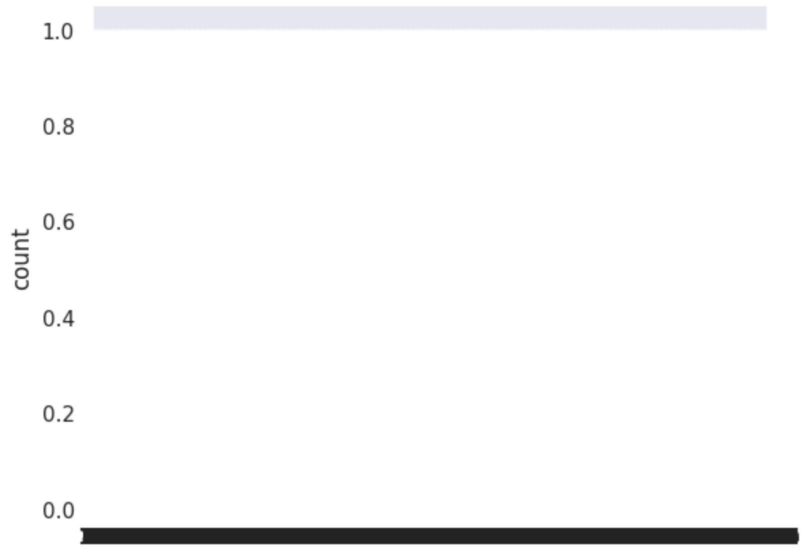
```python
sns.countplot(calories_data['Gender'])
```

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



```
sns.countplot(calories_data['Weight'])
```

```
<Axes: ylabel='count'>
```



```
sns.countplot(calories_data['Age'])
```

```
<Axes: ylabel='count'>
```

```
sns.countplot(calories_data['Height'])
```

```
<Axes: ylabel='count'>
```

```
sns.countplot(calories_data['Heart_Rate'])
```

```
<Axes: ylabel='count'>
```

```
sns.distplot(calories_data['Age'])
```

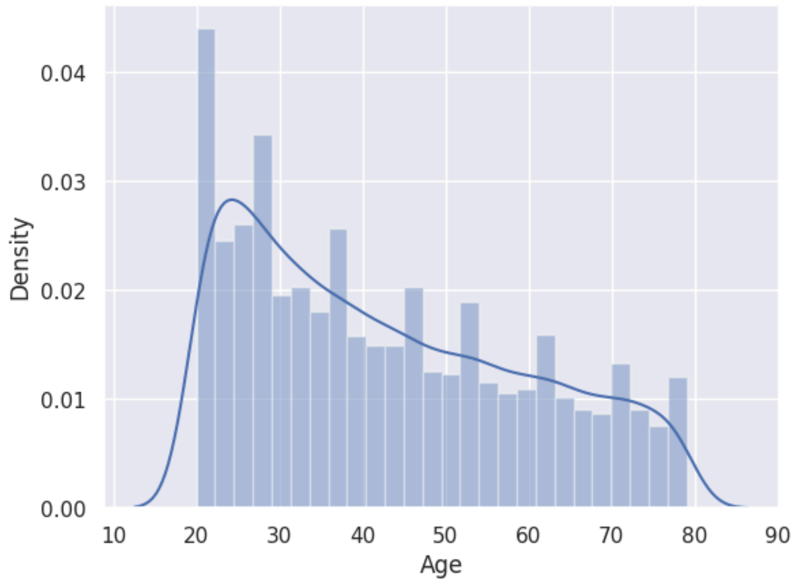
```
/tmp/ipython-input-1950677799.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(calories_data['Age'])
<Axes: xlabel='Age', ylabel='Density'>
```



```
sns.distplot(calories_data['Height'])
```

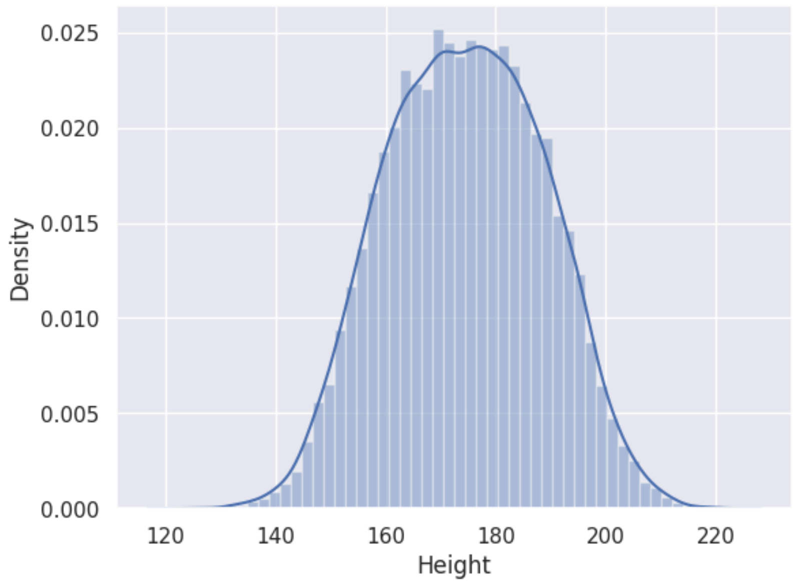
```
/tmp/ipython-input-2430521470.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(calories_data['Height'])
<Axes: xlabel='Height', ylabel='Density'>
```



```
sns.distplot(calories_data['Weight'])
```

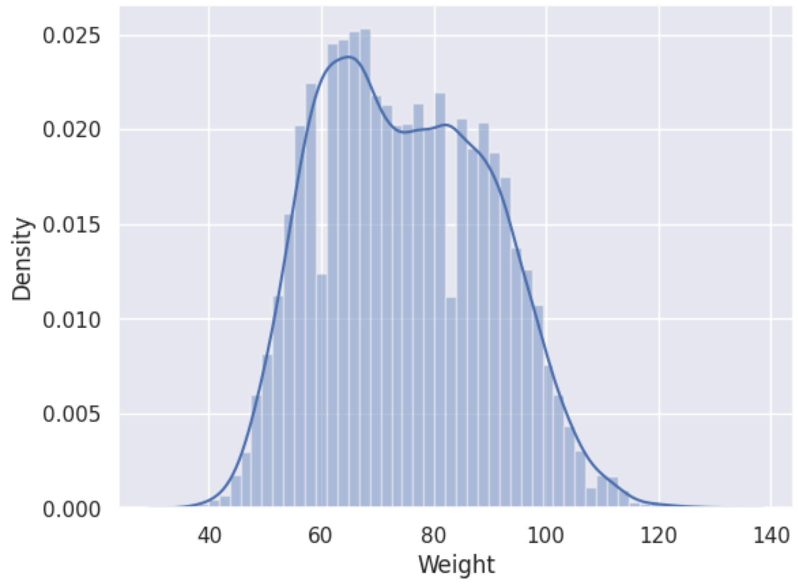
```
/tmp/ipython-input-1539919118.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(calories_data['Weight'])
<Axes: xlabel='Weight', ylabel='Density'>
```



```
sns.distplot(calories_data['Calories'])
```

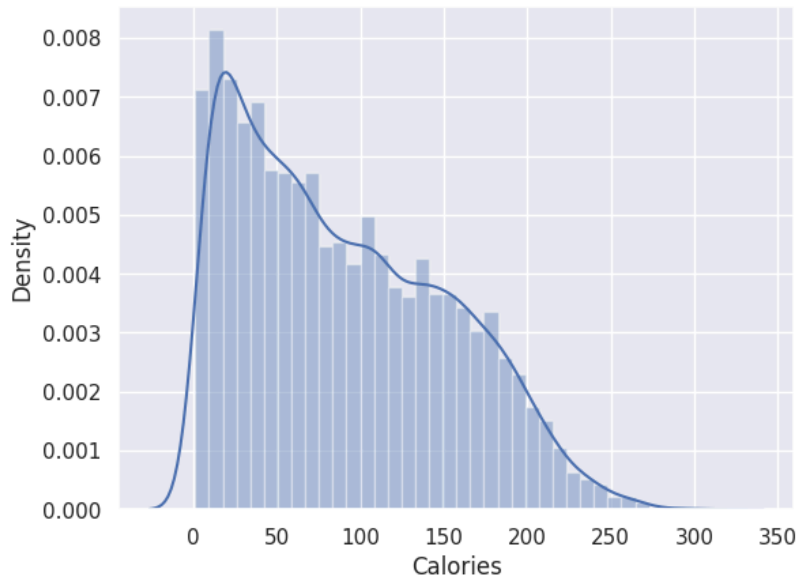
```
/tmp/ipython-input-4164852567.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(calories_data['Calories'])
<Axes: xlabel='Calories', ylabel='Density'>
```



```
correlation = calories_data.corr()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/tmp/ipython-input-669282712.py in <cell line: 0>()
----> 1 correlation = calories_data.corr()

                           3 frames
/usr/local/lib/python3.12/dist-packages/pandas/core/internals/managers.py in _interleave(self, dtype, na_value)
   1751             else:
   1752                 arr = blk.get_values(dtype)
-> 1753             result[rl.indexer] = arr
   1754             itemmask[rl.indexer] = 1
   1755 

ValueError: could not convert string to float: 'male'
```

Next steps: Explain error

```
# Create new columns for 'male' and 'female' (or whatever the categories are)
calories_encoded = pd.get_dummies(calories_data, columns=['Gender'], drop_first=True)

# Now, 'Gender_male' (or similar) will be a 0/1 column
correlation = calories_encoded.corr()
```

```
correlation = calories_data.corr()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/tmp/ipython-input-669282712.py in <cell line: 0>()
----> 1 correlation = calories_data.corr()

                           3 frames
/usr/local/lib/python3.12/dist-packages/pandas/core/internals/managers.py in _interleave(self, dtype, na_value)
   1751             else:
   1752                 arr = blk.get_values(dtype)
-> 1753             result[rl.indexer] = arr
   1754             itemmask[rl.indexer] = 1
   1755 

ValueError: could not convert string to float: 'male'
```

Next steps: Explain error

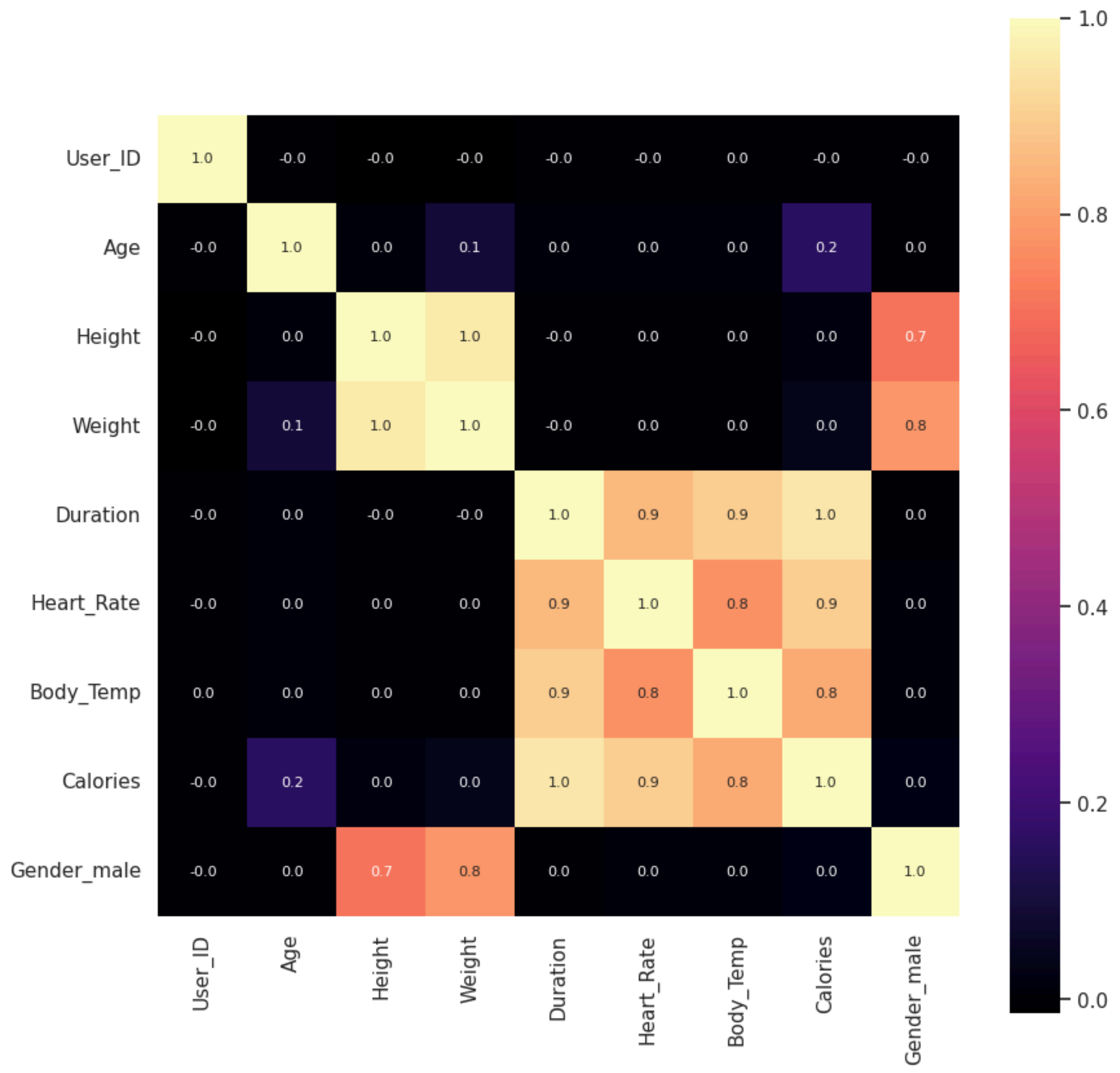
```
calories_data.head()
```

	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

Next steps: Generate code with calories_data New interactive sheet

```
plt.figure(figsize=(10,10))
sns.heatmap(correlation, cbar=True, square=True, fmt= '.1f', annot=True, annot_kws={'size':8}, cmap='magma')
```

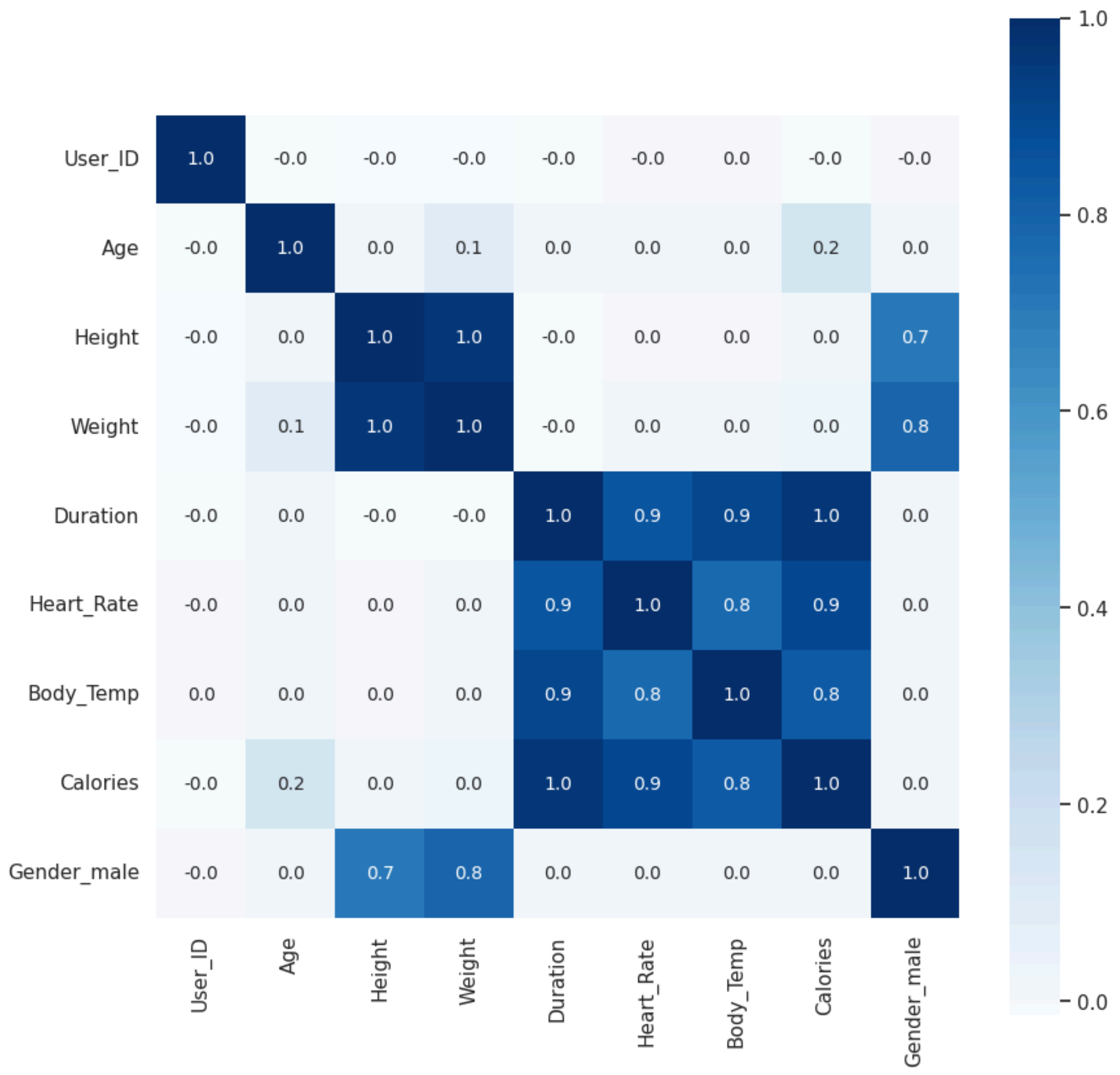
```
<Axes: >
```



	User_ID	Age	Height	Weight	Duration	Heart_Rate	Body_Temp	Calories	Gender_male
User_ID	1.0	-0.0	-0.0	-0.0	-0.0	-0.0	0.0	-0.0	-0.0
Age	-0.0	1.0	0.0	0.1	0.0	0.0	0.0	0.2	0.0
Height	-0.0	0.0	1.0	1.0	-0.0	0.0	0.0	0.0	0.7
Weight	-0.0	0.1	1.0	1.0	-0.0	0.0	0.0	0.0	0.8
Duration	-0.0	0.0	-0.0	-0.0	1.0	0.9	0.9	1.0	0.0
Heart_Rate	-0.0	0.0	0.0	0.0	0.9	1.0	0.8	0.9	0.0
Body_Temp	0.0	0.0	0.0	0.0	0.9	0.8	1.0	0.8	0.0
Calories	-0.0	0.2	0.0	0.0	1.0	0.9	0.8	1.0	0.0
Gender_male	-0.0	0.0	0.7	0.8	0.0	0.0	0.0	0.0	1.0

```
plt.figure(figsize=(10,10))
sns.heatmap(correlation, cbar=True, square=True, fmt= '.1f', annot=True, annot_kws={'size':10}, cmap='Blues')
```

```
<Axes: >
```



	User_ID	Age	Height	Weight	Duration	Heart_Rate	Body_Temp	Calories	Gender_male
User_ID	1.0	-0.0	-0.0	-0.0	-0.0	-0.0	0.0	-0.0	-0.0
Age	-0.0	1.0	0.0	0.1	0.0	0.0	0.0	0.2	0.0
Height	-0.0	0.0	1.0	1.0	-0.0	0.0	0.0	0.0	0.7
Weight	-0.0	0.1	1.0	1.0	-0.0	0.0	0.0	0.0	0.8
Duration	-0.0	0.0	-0.0	-0.0	1.0	0.9	0.9	1.0	0.0
Heart_Rate	-0.0	0.0	0.0	0.0	0.9	1.0	0.8	0.9	0.0
Body_Temp	0.0	0.0	0.0	0.0	0.9	0.8	1.0	0.8	0.0
Calories	-0.0	0.2	0.0	0.0	1.0	0.9	0.8	1.0	0.0
Gender_male	-0.0	0.0	0.7	0.8	0.0	0.0	0.0	0.0	1.0

```
#converting the text data into numerical data
calories_data.replace({"Gender":{'male':0,'female':1}},inplace=True)
```

```
/tmp/ipython-input-3949438929.py:2: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a f
  calories_data.replace({"Gender":{'male':0,'female':1}},inplace=True)
```

```
calories_data.head()
```

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

Next steps: Generate code with `calories_data` New interactive sheet

```
x=calories_data.drop(columns=['Calories','User_ID'],axis=1)
y=calories_data['Calories']
```

```
print(x)
```

```
       Gender  Age  Height  Weight  Duration  Heart_Rate  Body_Temp
0           0   68   190.0    94.0      29.0       105.0       40.8
1           1   20   166.0    60.0      14.0        94.0       40.3
2           0   69   179.0    79.0       5.0        88.0       38.7
3           1   34   179.0    71.0      13.0       100.0       40.5
4           1   27   154.0    58.0      10.0        81.0       39.8
...       ...  ...     ...     ...       ...         ...        ...
14995       1   20   193.0    86.0      11.0        92.0       40.4
14996       1   27   165.0    65.0       6.0        85.0       39.2
14997       1   43   159.0    58.0      16.0        90.0       40.1
14998       0   78   193.0    97.0       2.0        84.0       38.3
14999       0   63   173.0    79.0      18.0        92.0       40.5

[15000 rows x 7 columns]
```

```
print(y)
```

```
0        231.0
1         66.0
2         26.0
3         71.0
4         35.0
         ...
14995     45.0
14996     23.0
14997     75.0
14998     11.0
14999     98.0
Name: Calories, Length: 15000, dtype: float64
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2, random_state=2)
```

```
print(x.shape,x_train.shape,x_test.shape)
```

```
(15000, 7) (12000, 7) (3000, 7)
```

```
model=XGBRegressor()
```

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

```
XGBRegressor
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             feature_weights=None, gamma=None, grow_policy=None,
             importance_type=None, interaction_constraints=None,
             learning_rate=None, max_bin=None, max_cat_threshold=None,
             max_cat_to_onehot=None, max_delta_step=None, max_depth=None,
             max_leaves=None, min_child_weight=None, missing=nan,
             monotone_constraints=None, multi_strategy=None, n_estimators=None,
             n_jobs=None, num_parallel_tree=None, ...)
```

```
test_data_predict=model.predict(x_test)
```

```
print(test_data_predict)
```

```
[125.58828  222.11377   38.725952 ... 144.3179    23.425894  90.100494]
```

```
mae=metrics.mean_absolute_error(y_test,test_data_predict)
```

```
print("Mean Absolute Error=",mae)
```

```
Mean Absolute Error= 1.4833678883314132
```

```
calories_data['Height_m'] = calories_data['Height'] / 100
calories_data['BMI'] = calories_data['Weight'] / (calories_data['Height_m'] ** 2)
calories_data.drop(columns=['Height_m'], inplace=True)
```

```
from sklearn.model_selection import GridSearchCV
from xgboost import XGBRegressor
```

```
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.05, 0.1, 0.2]
}
```

```
xgb_model = XGBRegressor(random_state=2)
grid_search = GridSearchCV(
    estimator=xgb_model,
    param_grid=param_grid,
    scoring='neg_mean_absolute_error',
    cv=3,
    verbose=1,
    n_jobs=-1
)
```

```
grid_search.fit(x_train, y_train)
```

```
Fitting 3 folds for each of 27 candidates, totalling 81 fits
▸        GridSearchCV
                           i ?

▸      best_estimator_:
         XGBRegressor

      ▸ XGBRegressor  ?
```

```
calories_data.head()
```

	User_ID	Gender	Age	Height	Weight	Duration	Heart_Rate	Body_Temp	Calories	BMI
0	14733363	0	68	190.0	94.0	29.0	105.0	40.8	231.0	26.038781
1	14861698	1	20	166.0	60.0	14.0	94.0	40.3	66.0	21.773842
2	11179863	0	69	179.0	79.0	5.0	88.0	38.7	26.0	24.655910
3	16180408	1	34	179.0	71.0	13.0	100.0	40.5	71.0	22.159109
4	17771927	1	27	154.0	58.0	10.0	81.0	39.8	35.0	24.456063