D13

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\16_Sleep_health_and_lifestyle_dataset
df
```

Out[2]:

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Blc Press
0	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	126
1	2	Male	28	Doctor	6.2	6	60	8	Normal	125
2	3	Male	28	Doctor	6.2	6	60	8	Normal	125
3	4	Male	28	Sales Representative	5.9	4	30	8	Obese	140
4	5	Male	28	Sales Representative	5.9	4	30	8	Obese	140
...	...	...	...	...	...	...	...	...	...	
369	370	Female	59	Nurse	8.1	9	75	3	Overweight	140
370	371	Female	59	Nurse	8.0	9	75	3	Overweight	140
371	372	Female	59	Nurse	8.1	9	75	3	Overweight	140
372	373	Female	59	Nurse	8.1	9	75	3	Overweight	140
373	374	Female	59	Nurse	8.1	9	75	3	Overweight	140

374 rows × 13 columns

In [3]:
```
df.head(10)
```

Out[3]:

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Blood Pressure
0	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	126/83
1	2	Male	28	Doctor	6.2	6	60	8	Normal	125/80
2	3	Male	28	Doctor	6.2	6	60	8	Normal	125/80
3	4	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90
4	5	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90
5	6	Male	28	Software Engineer	5.9	4	30	8	Obese	140/90
6	7	Male	29	Teacher	6.3	6	40	7	Obese	140/90
7	8	Male	29	Doctor	7.8	7	75	6	Normal	120/80
8	9	Male	29	Doctor	7.8	7	75	6	Normal	120/80
9	10	Male	29	Doctor	7.8	7	75	6	Normal	120/80

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 374 entries, 0 to 373
Data columns (total 13 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Person ID                374 non-null    int64  
 1   Gender                   374 non-null    object 
 2   Age                      374 non-null    int64  
 3   Occupation               374 non-null    object 
 4   Sleep Duration           374 non-null    float64
 5   Quality of Sleep         374 non-null    int64  
 6   Physical Activity Level  374 non-null    int64  
 7   Stress Level             374 non-null    int64  
 8   BMI Category             374 non-null    object 
 9   Blood Pressure           374 non-null    object 
 10  Heart Rate               374 non-null    int64  
 11  Daily Steps              374 non-null    int64  
 12  Sleep Disorder           374 non-null    object 
dtypes: float64(1), int64(7), object(5)
memory usage: 38.1+ KB
```

In [5]:
```
df.describe()
```

Out[5]:

	Person ID	Age	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	Heart Rate	Da
count	374.000000	374.000000	374.000000	374.000000	374.000000	374.000000	374.000000	37
mean	187.500000	42.184492	7.132086	7.312834	59.171123	5.385027	70.165775	681
std	108.108742	8.673133	0.795657	1.196956	20.830804	1.774526	4.135676	161
min	1.000000	27.000000	5.800000	4.000000	30.000000	3.000000	65.000000	30(
25%	94.250000	35.250000	6.400000	6.000000	45.000000	4.000000	68.000000	56(
50%	187.500000	43.000000	7.200000	7.000000	60.000000	5.000000	70.000000	70(
75%	280.750000	50.000000	7.800000	8.000000	75.000000	7.000000	72.000000	80(
max	374.000000	59.000000	8.500000	9.000000	90.000000	8.000000	86.000000	100(

In [6]:
```
df.columns
```

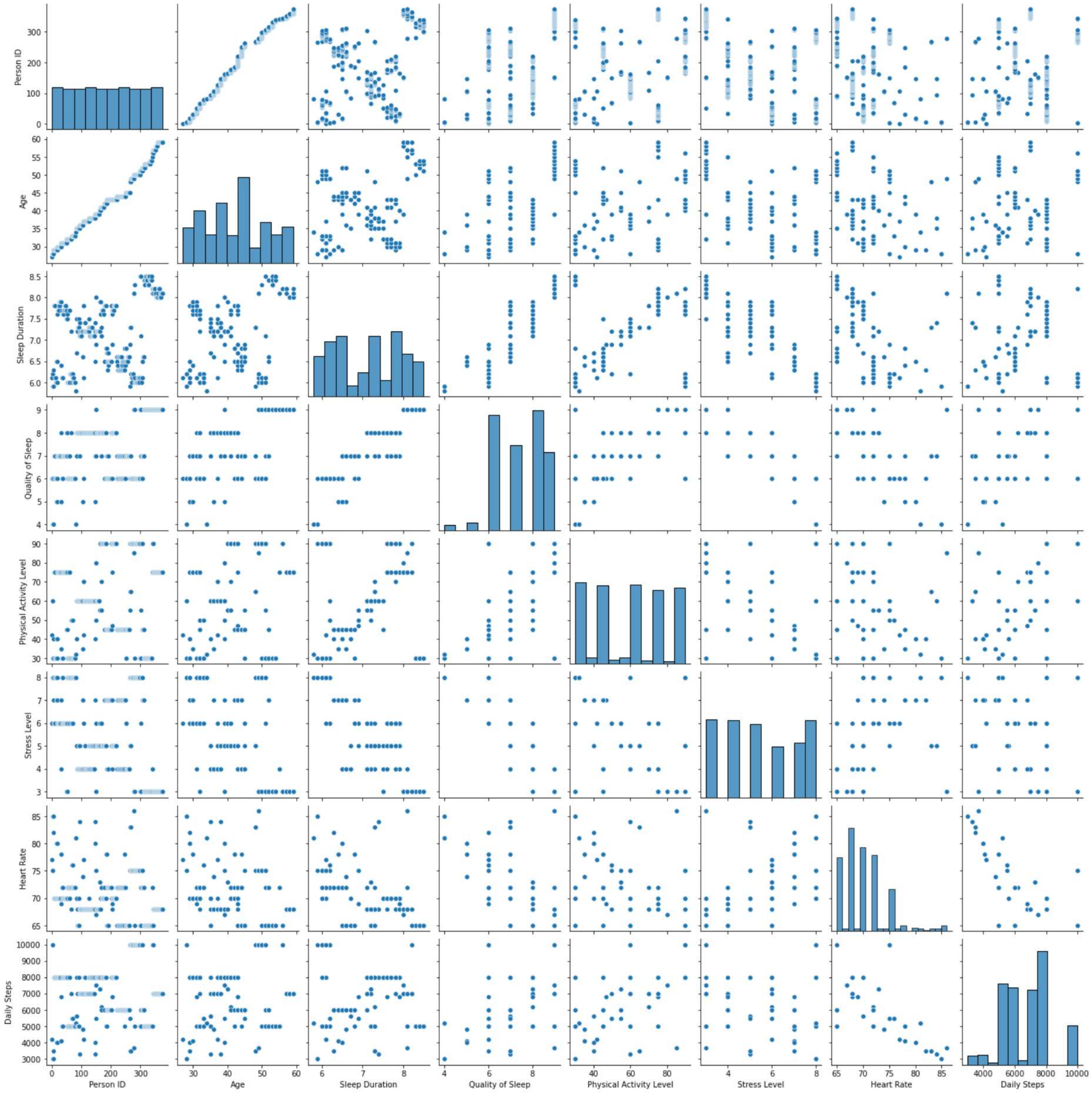
Out[6]:
```
Index(['Person ID', 'Gender', 'Age', 'Occupation', 'Sleep Duration',
       'Quality of Sleep', 'Physical Activity Level', 'Stress Level',
       'BMI Category', 'Blood Pressure', 'Heart Rate', 'Daily Steps',
       'Sleep Disorder'],
      dtype='object')
```

In [7]:

```
sns.pairplot(df)
```

Out[7]:

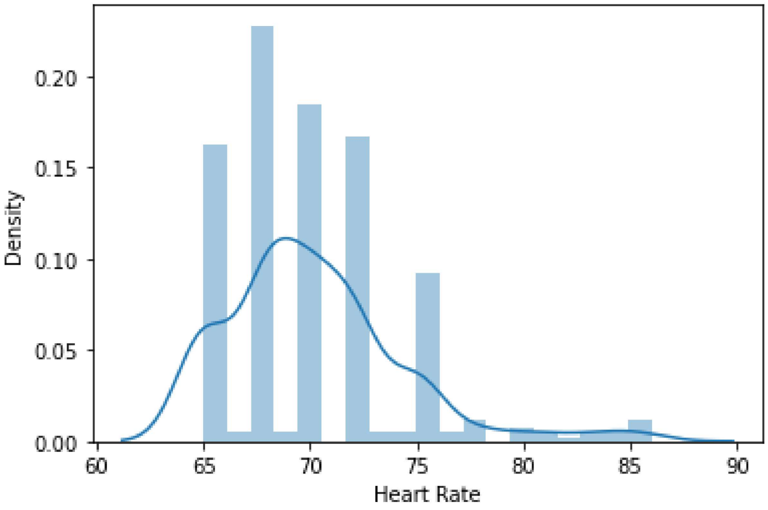
```
<seaborn.axisgrid.PairGrid at 0x227b0ae9b50>
```



In [8]:
```
sns.distplot(df["Heart Rate"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)
```

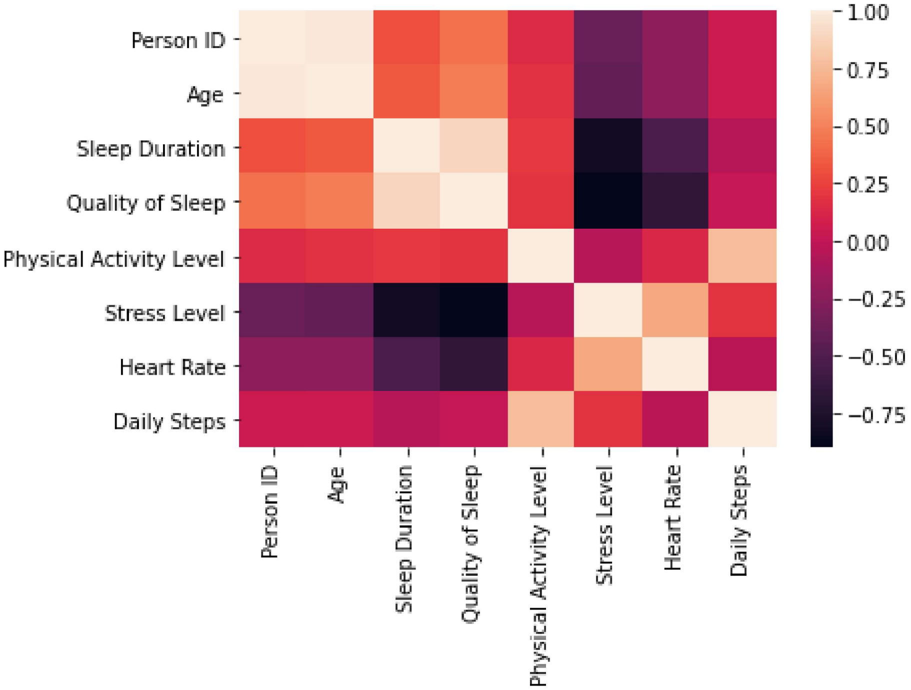
Out[8]:
```
<AxesSubplot:xlabel='Heart Rate', ylabel='Density'>
```



In [9]:
```
df1=df[['Person ID', 'Gender', 'Age', 'Occupation', 'Sleep Duration',
       'Quality of Sleep', 'Physical Activity Level', 'Stress Level',
       'BMI Category', 'Blood Pressure', 'Heart Rate', 'Daily Steps',
       'Sleep Disorder']]
```

In [10]:
```
sns.heatmap(df1.corr())
```

Out[10]:
```
<AxesSubplot:>
```



In [11]:
```
x=df1[['Person ID', 'Age', 'Sleep Duration',
       'Quality of Sleep', 'Physical Activity Level', 'Stress Level'
       , 'Daily Steps']]
y=df1['Heart Rate']
```

In [12]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [13]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[13]:
```
LinearRegression()
```

In [14]:
```
print(lr.intercept_)
```

```
59.98431856357619
```

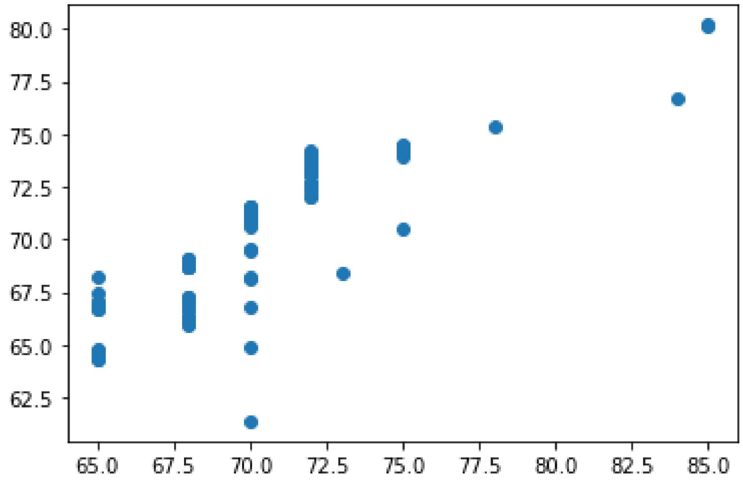
In [15]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[15]:

	Co-efficient
Person ID	-0.036545
Age	0.507671
Sleep Duration	0.183908
Quality of Sleep	-1.270134
Physical Activity Level	0.153268
Stress Level	1.412109
Daily Steps	-0.001918

In [16]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[16]:
```
<matplotlib.collections.PathCollection at 0x227b6127220>
```



In [17]:
```
print(lr.score(x_test,y_test))
```

```
0.7363283591344467
```

In [18]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [19]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[19]:
```
Ridge(alpha=10)
```

In [20]:
```
rr.score(x_test,y_test)
```

Out[20]:
```
0.7381215462447748
```

In [21]:

```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[21]:

```
Lasso(alpha=10)
```

In [22]:

```
la.score(x_test,y_test)
```

Out[22]:

```
0.10064803231278763
```

In [23]:

```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[23]:

```
ElasticNet()
```

In [24]:

```
print(en.coef_)
```

```
[-5.95217275e-04  0.00000000e+00 -0.00000000e+00 -5.31134367e-01
  1.43483113e-01  1.23901103e+00 -1.75172130e-03]
```

In [25]:

```
print(en.intercept_)
```

```
70.9296475173023
```

In [26]:
```
print(en.predict(x_train))
```

```
[71.37061795 71.43521308 67.37918065 70.3641258  67.91034265 67.91629483
 73.16748427 72.24328735 72.24923952 67.93910514 71.67653144 71.65272274
 65.22426953 67.40298934 72.22959735 72.25638213 73.16629383 73.16926992
 67.97666816 67.40537021 71.3795462  71.67712665 65.22605518 71.64608386
 71.65034188 67.3845376  67.98262033 71.64677057 72.23019257 75.41629192
 72.8851311  67.91272352 71.37537968 67.37679978 72.8815598  73.16569862
 67.93731949 71.36704664 73.49760657 72.87679806 76.14816687 67.91450918
 69.12162407 67.91748526 67.90558092 67.90915222 67.95940686 72.24269213
 71.67355535 67.89843831 71.64855622 71.64846473 65.23438822 65.23260257
 65.21950779 71.6717697  72.24626344 73.78228053 72.87620284 65.21355562
 72.21947866 67.92522309 67.39227543 67.96238294 65.22486475 67.89784309
 72.87739328 68.14899659 65.22248388 68.14959181 65.21236518 72.09677425
 71.36883229 74.76785756 68.15673442 71.38787925 67.37799021 67.40417977
 72.245073   65.22069823 68.1561392  72.88751197 73.43530385 71.37716534
 65.21415084 68.1525679  68.15911529 65.21534127 73.15438949 67.39525151
 76.44321688 73.19838333 67.92284222 72.22007388 72.88810719 71.38073664
 71.38728403 67.93236569 70.46650317 68.15316311 71.66998405 71.38609359
 68.00166728 73.16212731 72.21531214 65.21712692 72.88691675 67.92343743
 71.37002273 73.16450818 68.15792485 67.96119251 68.74551289 67.97071599
 73.15796079 73.15379427 68.15018703 72.22662127 71.64548865 65.21831736
 73.1615321  71.64736579 73.16391297 71.67415057 75.66937351 73.498797
 68.15554398 67.38632325 72.24566822 77.76355169 67.97547772 65.21891257
 72.22304996 73.1787934  71.43223699 73.84537356 72.67493078 71.6472743
 65.22545996 68.29053917 65.23319779 71.12649567 71.64498492 72.87560763
 76.44440732 67.96178773 68.16030572 74.15089932 72.8779885  73.18057905
 71.64617535 72.88870241 72.2278117  68.11733279 72.86965545 72.89286893
 65.21474605 71.43342743 75.4156967  68.14304442 67.93612906 62.80962405
 71.67296013 72.88275023 72.88929762 71.36109447 67.91212831 72.87441719
 71.64915144 73.15557992 71.36823708 68.11792801 67.89962874 71.67057926
 71.67772187 68.15435355 67.91153309 71.66760318 68.15137746 74.15149454
 71.38133186 67.91510439 68.98686575 72.22126431 72.675526   72.21769301
 72.22900213 67.90974744 67.922247   73.16331775 67.90379526 67.96357338
 72.86906024 72.89167849 67.97904903 73.15677036 67.9677399  68.15197268
 71.66819839 65.21176997 71.43283221 68.00285772 71.92642482 65.2129604
 67.37620456 71.65153231 71.64786952 71.65093709 67.39048978 75.59140005
 68.14780616 74.00393691 71.36645142 68.74491768 67.93791471 68.14363964
 67.38870412 71.43402264 71.92702004 67.92403265 75.66818308 71.37657012
 73.15498471 65.05917417 72.21650257 65.21117475 67.38989456 67.377395
 72.24745387 67.4154889  71.43461786 65.21593649 67.38334717 71.38847446
 67.97785859 74.76845278 65.21057953 67.96059729 65.24569735 68.28994396
 72.22543083 67.39406108 67.37977586 72.88394067 76.02005197 65.21653171
 73.15319905 67.95821642 71.36407056 68.15852007 72.88989284 67.92700874
 67.92105657 68.15732963 71.37776055 67.95762121 71.67117448 73.15617514
 72.87382197 72.24685865 67.39168021]
```

In [27]:
```
print(en.score(x_train,y_train))
```

```
0.6697487334633323
```

In [28]:
```
from sklearn import metrics
```

```
In [29]: print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolytre Error: 1.4446121552112152
```

```
In [30]: print("Mean Square Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Square Error: 3.76243676055674
```

```
In [31]: print("Root Mean Square Error:",np.sqrt(metrics.mean_absolute_error(y_test,pre
```

```
Root Mean Square Error: 1.201920195025949
```

```
In [ ]:
```