In [1]:

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

In [2]:

```
s=pd.read_csv(r"C:\Users\user\Downloads\16_Sleep_health_and_lifestyle_dataset - 16_Sleep_health_and_lifestyle_dataset.cs
s
```

Out[2]:

	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
0	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	126/83	77	4200	None
1	2	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75	10000	None
2	3	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75	10000	None
3	4	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85	3000	Sleep Apnea
4	5	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85	3000	Sleep Apnea
...	...	...	...	...	...	...	...	...	...	...	...	...	...
369	370	Female	59	Nurse	8.1	9	75	3	Overweight	140/95	68	7000	Sleep Apnea
370	371	Female	59	Nurse	8.0	9	75	3	Overweight	140/95	68	7000	Sleep Apnea
371	372	Female	59	Nurse	8.1	9	75	3	Overweight	140/95	68	7000	Sleep Apnea
372	373	Female	59	Nurse	8.1	9	75	3	Overweight	140/95	68	7000	Sleep Apnea
373	374	Female	59	Nurse	8.1	9	75	3	Overweight	140/95	68	7000	Sleep Apnea

374 rows × 13 columns

In [3]:

```
s=s.head(100)
s
```

Out[3]:

	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
0	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	126/83	77	4200	None
1	2	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75	10000	None
2	3	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75	10000	None
3	4	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85	3000	Sleep Apnea
4	5	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85	3000	Sleep Apnea
...	...	...	...	...	...	...	...	...	...	...	...	...	...
95	96	Female	36	Accountant	7.1	8	60	4	Normal	115/75	68	7000	None
96	97	Female	36	Accountant	7.2	8	60	4	Normal	115/75	68	7000	None
97	98	Female	36	Accountant	7.1	8	60	4	Normal	115/75	68	7000	None
98	99	Female	36	Teacher	7.1	8	60	4	Normal	115/75	68	7000	None
99	100	Female	36	Teacher	7.1	8	60	4	Normal	115/75	68	7000	None

100 rows × 13 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 13 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Person ID                100 non-null    int64  
 1   Gender                   100 non-null    object 
 2   Age                      100 non-null    int64  
 3   Occupation               100 non-null    object 
 4   Sleep Duration           100 non-null    float64
 5   Quality of Sleep         100 non-null    int64  
 6   Physical Activity Level  100 non-null    int64  
 7   Stress Level             100 non-null    int64  
 8   BMI Category             100 non-null    object 
 9   Blood Pressure           100 non-null    object 
 10  Heart Rate               100 non-null    int64  
 11  Daily Steps              100 non-null    int64  
 12  Sleep Disorder           100 non-null    object 
dtypes: float64(1), int64(7), object(5)
memory usage: 10.3+ KB
```

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

Out[5]:

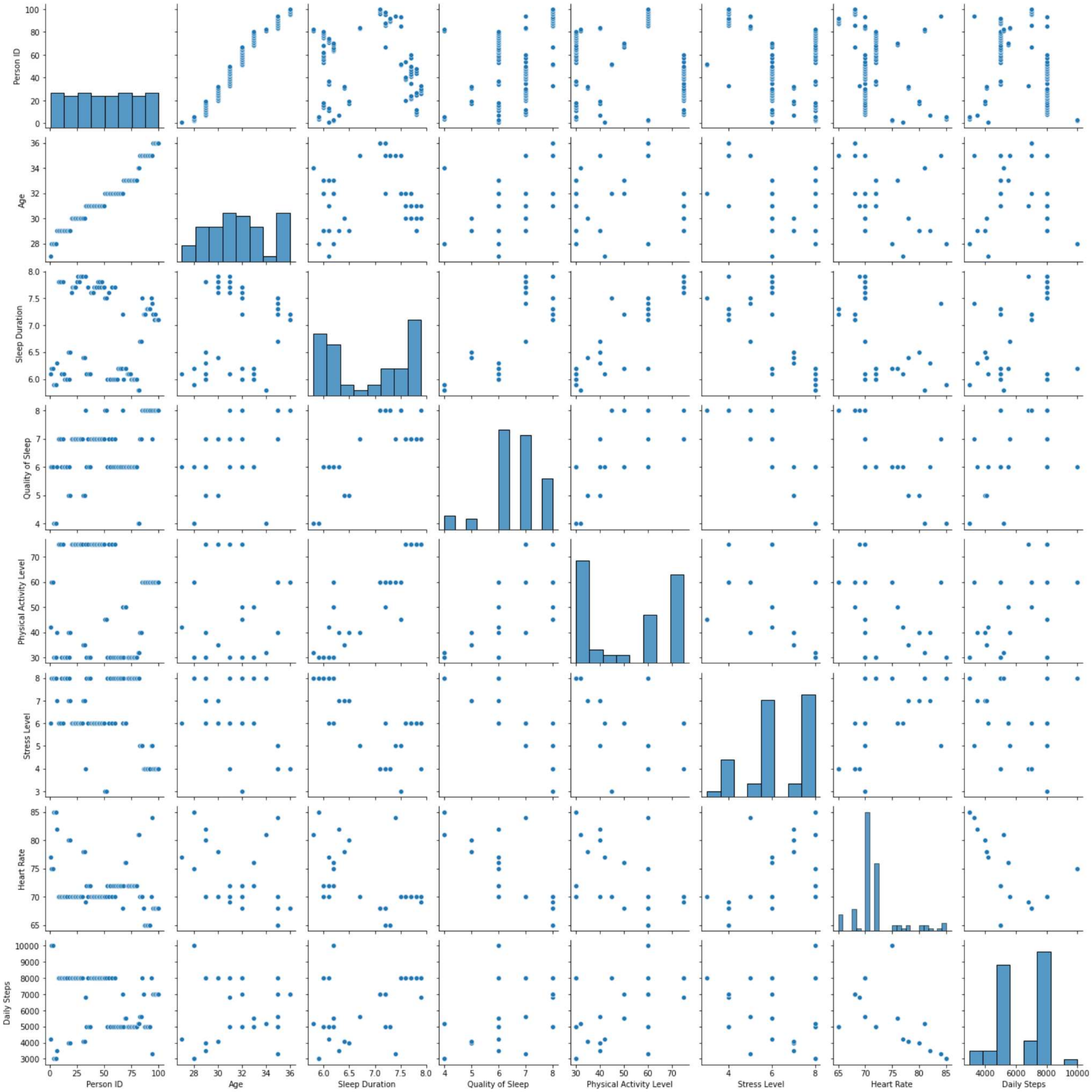
	Person ID	Age	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	Heart Rate	Daily Steps
count	100.000000	100.00000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000
mean	50.500000	31.69000	6.871000	6.590000	51.910000	6.420000	71.610000	6426.000000
std	29.011492	2.26388	0.766903	1.005992	19.429279	1.485145	4.240009	1689.517294
min	1.000000	27.00000	5.800000	4.000000	30.000000	3.000000	65.000000	3000.000000
25%	25.750000	30.00000	6.100000	6.000000	30.000000	6.000000	70.000000	5000.000000
50%	50.500000	31.50000	7.100000	7.000000	60.000000	6.000000	70.000000	7000.000000
75%	75.250000	33.00000	7.700000	7.000000	75.000000	8.000000	72.000000	8000.000000
max	100.000000	36.00000	7.900000	8.000000	75.000000	8.000000	85.000000	10000.000000

In [6]:
```
s.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(s)
```

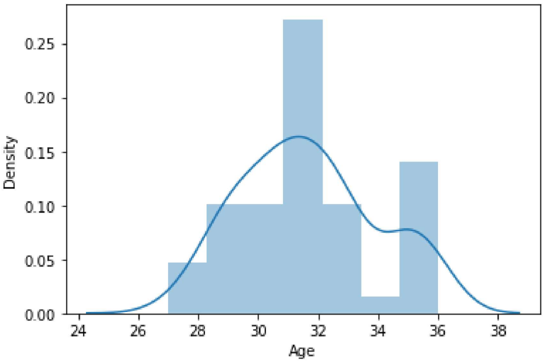
```
Out[7]: <seaborn.axisgrid.PairGrid at 0x1bbe85fb850>
```



```
In [8]: sns.distplot(s['Age'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated fun
ction and will be removed 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)
```

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



```
In [9]: s1=s[['Person ID','Age','Sleep Duration','Quality of Sleep','Stress Level']]
        s
```

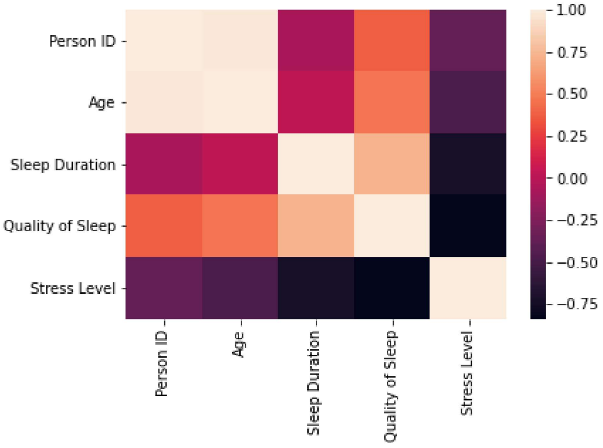
Out[9]:

	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
0	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	126/83	77	4200	None
1	2	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75	10000	None
2	3	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75	10000	None
3	4	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85	3000	Sleep Apnea
4	5	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85	3000	Sleep Apnea
...	...	...	...	...	...	...	...	...	...	...	...	...	...
95	96	Female	36	Accountant	7.1	8	60	4	Normal	115/75	68	7000	None
96	97	Female	36	Accountant	7.2	8	60	4	Normal	115/75	68	7000	None
97	98	Female	36	Accountant	7.1	8	60	4	Normal	115/75	68	7000	None
98	99	Female	36	Teacher	7.1	8	60	4	Normal	115/75	68	7000	None
99	100	Female	36	Teacher	7.1	8	60	4	Normal	115/75	68	7000	None

100 rows × 13 columns

```
In [10]: sns.heatmap(s1.corr())
```

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



In [11]:

```
x=s1[['Person ID','Age','Sleep Duration','Quality of Sleep','Stress Level']]
y=s1['Stress Level']
```

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]:

```
lr.intercept_
```

Out[14]:

```
-5.5067062021407764e-14
```

In [15]:

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

Out[15]:

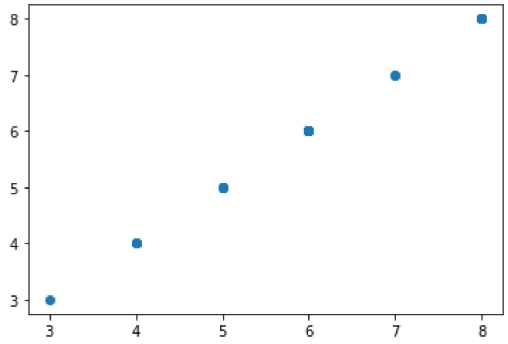
	Co-efficient
Person ID	-2.266715e-16
Age	2.789289e-15
Sleep Duration	-1.649334e-15
Quality of Sleep	-8.884300e-16
Stress Level	1.000000e+00

In [16]:

```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[16]:

```
<matplotlib.collections.PathCollection at 0x1bbf1a08670>
```



In [17]:

```
print(lr.score(x_test,y_test))
```

```
1.0
```

In [18]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [19]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [20]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[20]:

```
0.9781426348036008
```

In [21]:

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

Out[21]:

```
-0.03476371537208034
```

In [22]:

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

Out[22]:

```
ElasticNet()
```

```
In [23]: print(en.coef_)
```

```
[-0.00428925 -0.         -0.         -0.          0.61743786]
```

```
In [24]: print(en.intercept_)
```

```
2.7101852037236402
```

```
In [25]: print(en.predict(x_test))
```

```
[7.62395257 4.79390399 6.18319277 6.17032501 6.32473807 5.43278811
 7.38375447 6.95933294 6.25182079 6.20034977 6.89928341 5.39847409
 7.63253107 6.38049834 6.11456474 7.34086196 6.41052311 4.33945769
 7.60250631 5.44136661 7.40091148 7.58106005 6.31187032 7.64110957
 6.11885399 5.43707736 4.75958998 6.21321753 7.63682032 4.76816848]
```

```
In [26]: print(en.score(x_test,y_test))
```

```
0.8837501365595812
```

```
In [27]: from sklearn import metrics
```

```
In [28]: print("Mean Absolute Error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error 1.1694349192718315e-15
```

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

```
Mean squared Error 2.4520426470619784e-30
```

```
In [30]: print("Root Mean squared Error",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
Root Mean squared Error 1.5658999479730429e-15
```

```
In [ ]:
```