Problem Statement

Linear Regression

Import Libraries

In [1]:

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

In [2]:

```
a=pd.read_csv("Sleep.csv")
a
```

Out[2]:

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Blood Pressure	Heart Rate
0	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	126/83	77
1	2	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75
2	3	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75
3	4	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85
4	5	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85
...	...	...	...	...	...	...	...	...	...	...	...
369	370	Female	59	Nurse	8.1	9	75	3	Overweight	140/95	68

To display top 10 rows

In [3]:

```
c=a.head(15)
c
```

Out[3]:

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	I Pre
0	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	1
1	2	Male	28	Doctor	6.2	6	60	8	Normal	1
2	3	Male	28	Doctor	6.2	6	60	8	Normal	1
3	4	Male	28	Sales Representative	5.9	4	30	8	Obese	1
4	5	Male	28	Sales Representative	5.9	4	30	8	Obese	1
5	6	Male	28	Software Engineer	5.9	4	30	8	Obese	1
6	7	Male	29	Teacher	6.3	6	40	7	Obese	1
7	8	Male	29	Doctor	7.8	7	75	6	Normal	1
8	9	Male	29	Doctor	7.8	7	75	6	Normal	1
9	10	Male	29	Doctor	7.8	7	75	6	Normal	1
10	11	Male	29	Doctor	6.1	6	30	8	Normal	1
11	12	Male	29	Doctor	7.8	7	75	6	Normal	1
12	13	Male	29	Doctor	6.1	6	30	8	Normal	1
13	14	Male	29	Doctor	6.0	6	30	8	Normal	1
14	15	Male	29	Doctor	6.0	6	30	8	Normal	1

To find Missing values

In [4]:

```
c.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15 entries, 0 to 14
Data columns (total 13 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Person ID                15 non-null     int64  
 1   Gender                   15 non-null     object 
 2   Age                      15 non-null     int64  
 3   Occupation               15 non-null     object 
 4   Sleep Duration           15 non-null     float64
 5   Quality of Sleep         15 non-null     int64  
 6   Physical Activity Level  15 non-null     int64  
 7   Stress Level             15 non-null     int64  
 8   BMI Category             15 non-null     object 
 9   Blood Pressure           15 non-null     object 
 10  Heart Rate               15 non-null     int64  
 11  Daily Steps              15 non-null     int64  
 12  Sleep Disorder           15 non-null     object 
dtypes: float64(1), int64(7), object(5)
memory usage: 1.6+ KB
```

To display summary of statistics

In [5]:

```
a.describe()
```

Out[5]:

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

To display column heading

In [6]:

```
a.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')
```

Pairplot

In [7]:

```
s=a.dropna(axis=1)
s
```

Out[7]:

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

374 rows × 13 columns

In [8]:

```
s.columns
```

Out[8]:

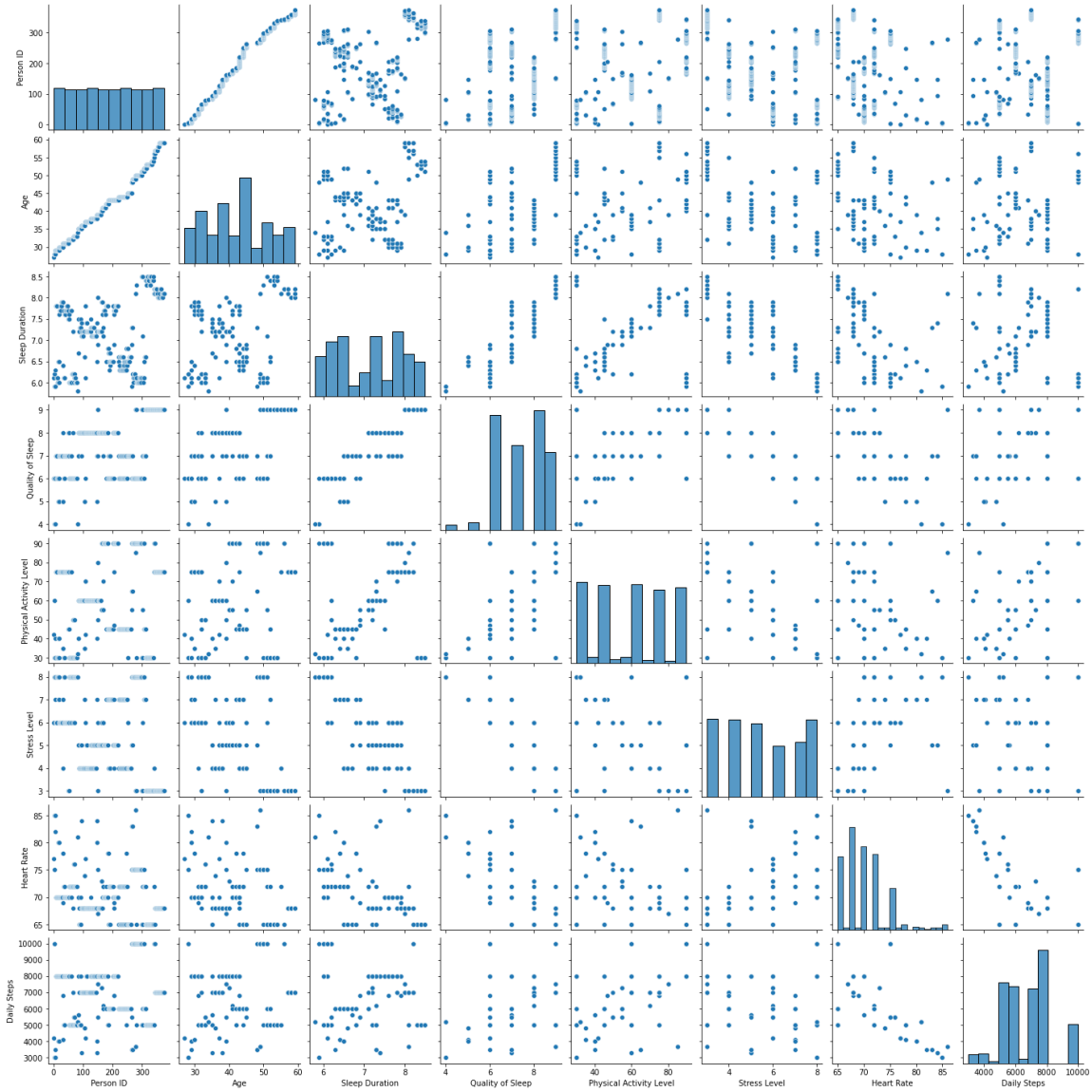
```
Index(['Person ID', 'Gender', 'Age', 'Occupation', 'Sleep Duration',
       'Quality of Sleep', 'Physical Activity Level', 'Stress Level',
       'BMI Category', 'Blood Pressure', 'Heart Rate', 'Daily Steps',
       'Sleep Disorder'],
      dtype='object')
```

In [9]:

```
sns.pairplot(a)
```

Out[9]:

```
<seaborn.axisgrid.PairGrid at 0x20748bc46d0>
```



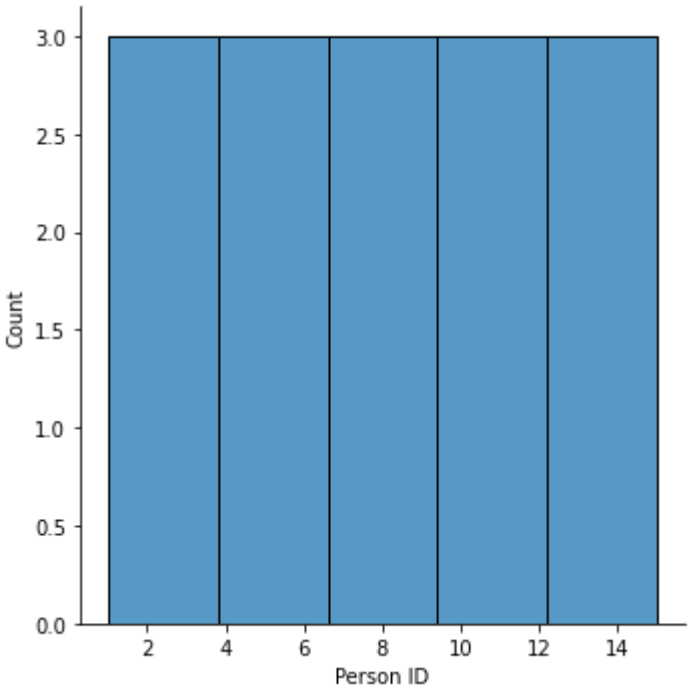
Distribution Plot

In [10]:

```
sns.displot(c['Person ID'])
```

Out[10]:

```
<seaborn.axisgrid.FacetGrid at 0x2074bb6a3d0>
```



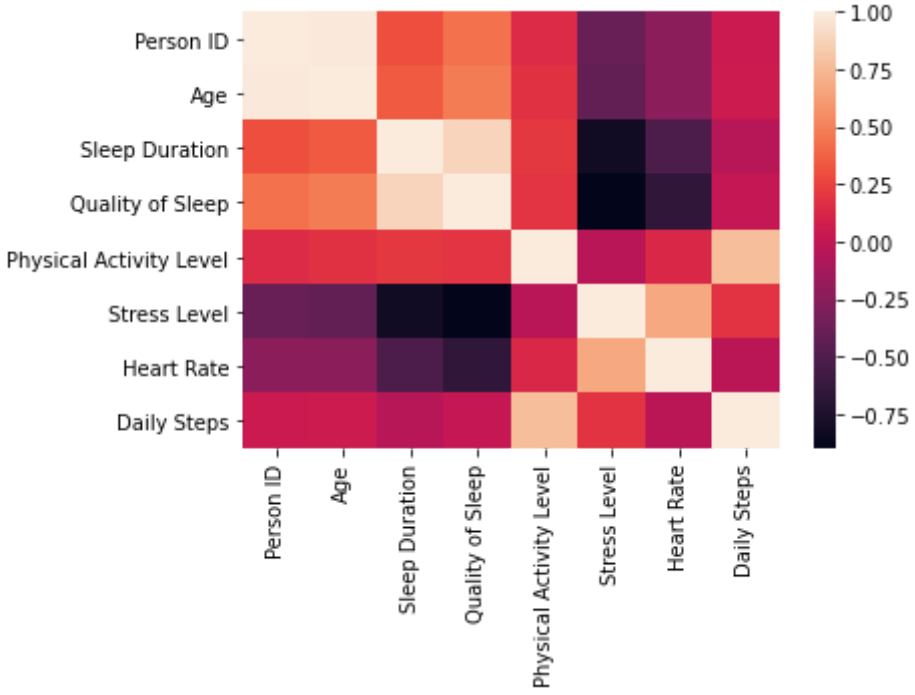
Correlation

In [11]:

```
b=a[['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']]
sns.heatmap(b.corr())
```

Out[11]:

```
<AxesSubplot:>
```



Train the model - Model Building

In [12]:

```
g=c[['Person ID']]
h=c['Age']
```

To split dataset into training end test

In [13]:

```
from sklearn.model_selection import train_test_split
g_train,g_test,h_train,h_test=train_test_split(g,h,test_size=0.6)
```

To run the model

In [14]:

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

In [15]:

```
lr=LinearRegression()
lr.fit(g_train,h_train)
```

Out[15]:

```
LinearRegression()
```

In [16]:

```
print(lr.intercept_)
```

```
27.666666666666668
```

Coeffecient

In [17]:

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

Out[17]:

	Co-effecient
Person ID	0.1

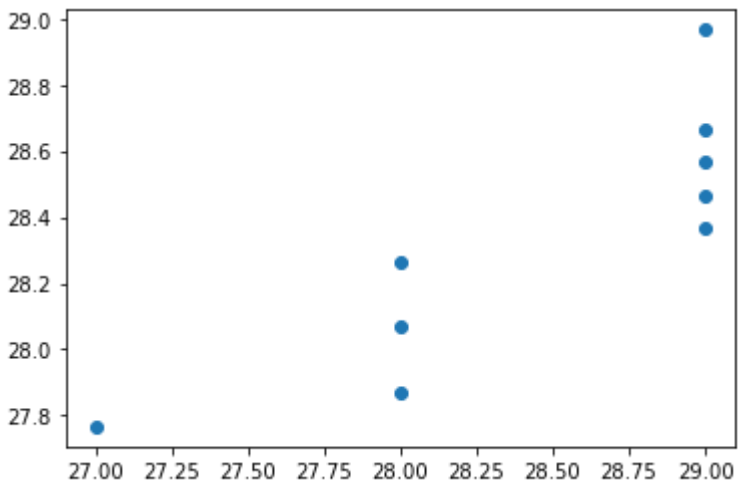
Best Fit line

In [18]:

```
prediction=lr.predict(g_test)
plt.scatter(h_test,prediction)
```

Out[18]:

```
<matplotlib.collections.PathCollection at 0x2074e0faf70>
```



To find score

In [19]:

```
print(lr.score(g_test,h_test))
```

```
0.6052631578947364
```

Import Lasso and ridge

In [20]:

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

Ridge

In [21]:

```
ri=Ridge(alpha=5)
ri.fit(g_train,h_train)
```

Out[21]:

```
Ridge(alpha=5)
```

In [22]:

```
ri.score(g_test,h_test)
```

Out[22]:

```
0.5968505263157902
```

In [23]:

```
ri.score(g_train,h_train)
```

Out[23]:

```
0.89856
```

Lasso

In [24]:

```
l=Lasso(alpha=6)
l.fit(g_train,h_train)
```

Out[24]:

```
Lasso(alpha=6)
```

In [25]:

```
l.score(g_test,h_test)
```

Out[25]:

```
-0.10526315789473784
```

In [26]:

```
ri.score(g_train,h_train)
```

Out[26]:

```
0.89856
```

ElasticNet

In [27]:

```
from sklearn.linear_model import ElasticNet
e=ElasticNet()
e.fit(g_train,h_train)
```

Out[27]:

```
ElasticNet()
```

Coeffecient,intercept

In [28]:

```
print(e.coef_)
```

```
[0.07317073]
```

In [29]:

```
print(e.intercept_)
```

```
27.9349593495935
```

Prediction

In [30]:

```
d=e.predict(g_test)
d
```

Out[30]:

```
array([28.00813008, 28.37398374, 28.5203252 , 28.44715447, 28.88617886,
       28.08130081, 28.66666667, 28.22764228, 28.59349593])
```

In [31]:

```
print(e.score(g_test,h_test))
```

```
0.5169228842480982
```

Evaluation

In [32]:

```
from sklearn import metrics
print("Mean Absolute error:",metrics.mean_absolute_error(h_test,d))
```

```
Mean Absolute error: 0.3974706413730797
```

In [33]:

```
print("Mean Squared error:",metrics.mean_squared_error(h_test,d))
```

```
Mean Squared error: 0.22662877035274404
```

In [34]:

```
print("Mean Squared error:",np.sqrt(metrics.mean_squared_error(h_test,d)))
```

```
Mean Squared error: 0.47605542781565263
```

In [ ]:

```

```