Problem Statement

Linear Regression

Import Libraries

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
a=pd.read_csv("Sleep.csv")
a
```

Out[4]:

	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
...	...	...	...	...	...	...	...	...	...	...
369	370	Female	59	Nurse	8.1	9	75	3	Overweight	140/95
370	371	Female	59	Nurse	8.0	9	75	3	Overweight	140/95
371	372	Female	59	Nurse	8.1	9	75	3	Overweight	140/95
372	373	Female	59	Nurse	8.1	9	75	3	Overweight	140/95
373	374	Female	59	Nurse	8.1	9	75	3	Overweight	140/95

374 rows × 13 columns

To display top 10 rows

In [5]:
```
c=a.head(15)
c
```

Out[5]:

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Blood Pressure	I
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	
10	11	Male	29	Doctor	6.1	6	30	8	Normal	120/80	
11	12	Male	29	Doctor	7.8	7	75	6	Normal	120/80	
12	13	Male	29	Doctor	6.1	6	30	8	Normal	120/80	
13	14	Male	29	Doctor	6.0	6	30	8	Normal	120/80	
14	15	Male	29	Doctor	6.0	6	30	8	Normal	120/80	

To find Missing values

In [6]:

```
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 [7]:
```
a.describe()
```

Out[7]:

	Person ID	Age	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	Heart Rate	Daily S
count	374.000000	374.000000	374.000000	374.000000	374.000000	374.000000	374.000000	374.00(
mean	187.500000	42.184492	7.132086	7.312834	59.171123	5.385027	70.165775	6816.84
std	108.108742	8.673133	0.795657	1.196956	20.830804	1.774526	4.135676	1617.91
min	1.000000	27.000000	5.800000	4.000000	30.000000	3.000000	65.000000	3000.00(
25%	94.250000	35.250000	6.400000	6.000000	45.000000	4.000000	68.000000	5600.00(
50%	187.500000	43.000000	7.200000	7.000000	60.000000	5.000000	70.000000	7000.00(
75%	280.750000	50.000000	7.800000	8.000000	75.000000	7.000000	72.000000	8000.00(
max	374.000000	59.000000	8.500000	9.000000	90.000000	8.000000	86.000000	10000.00(

To display column heading

In [8]:
```
a.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')
```

Pairplot

In [9]:
```
s=a.dropna(axis=1)
s
```

Out[9]:

	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
...	...	...	...	...	...	...	...	...	...	...

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Blood Pressure
369	370	Female	59	Nurse	8.1	9	75	3	Overweight	140/95
370	371	Female	59	Nurse	8.0	9	75	3	Overweight	140/95
371	372	Female	59	Nurse	8.1	9	75	3	Overweight	140/95
372	373	Female	59	Nurse	8.1	9	75	3	Overweight	140/95
373	374	Female	59	Nurse	8.1	9	75	3	Overweight	140/95

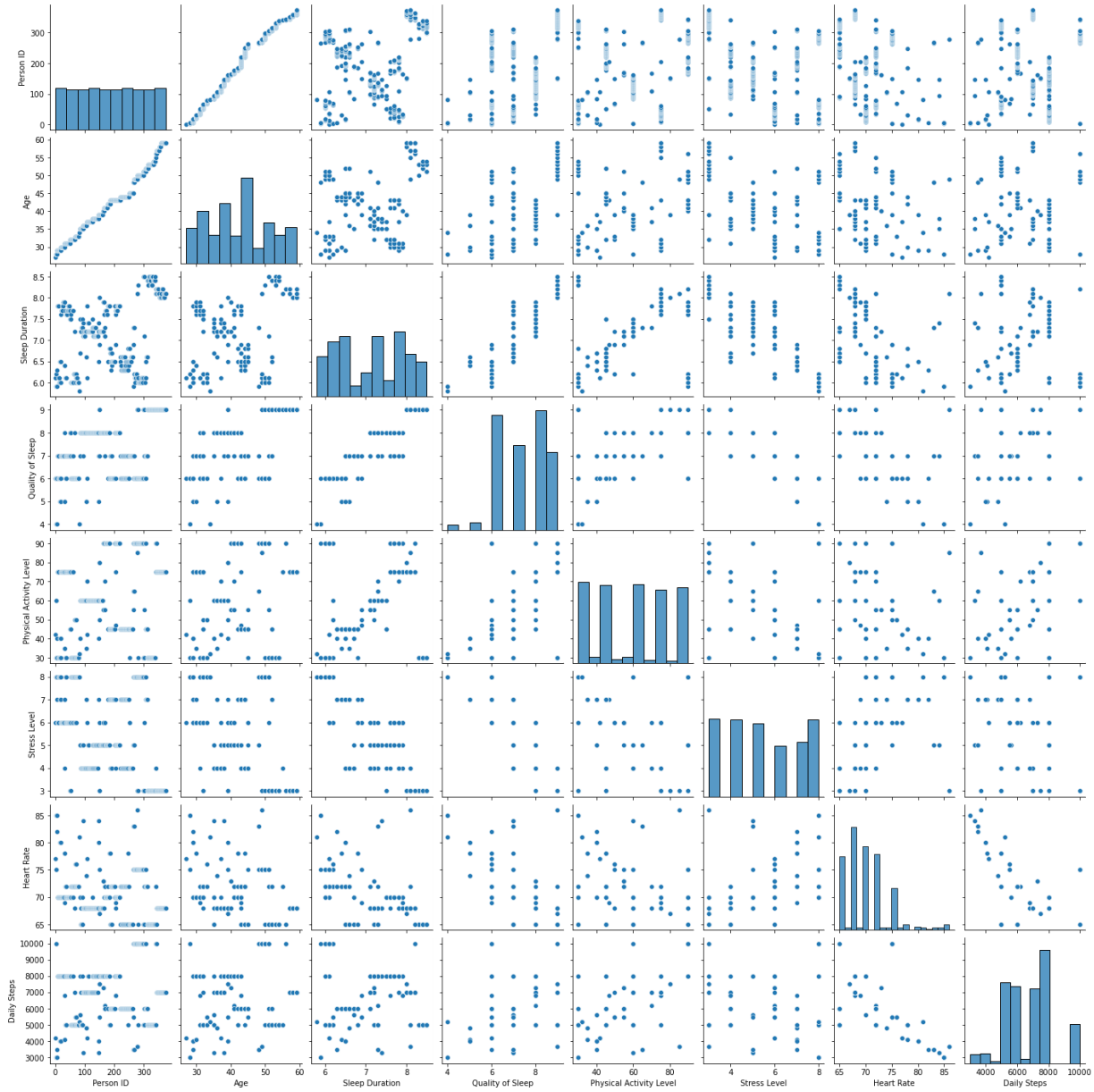
374 rows × 13 columns

In [10]:
```
s.columns
```

Out[10]:
```
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 [11]:
```
sns.pairplot(a)
```

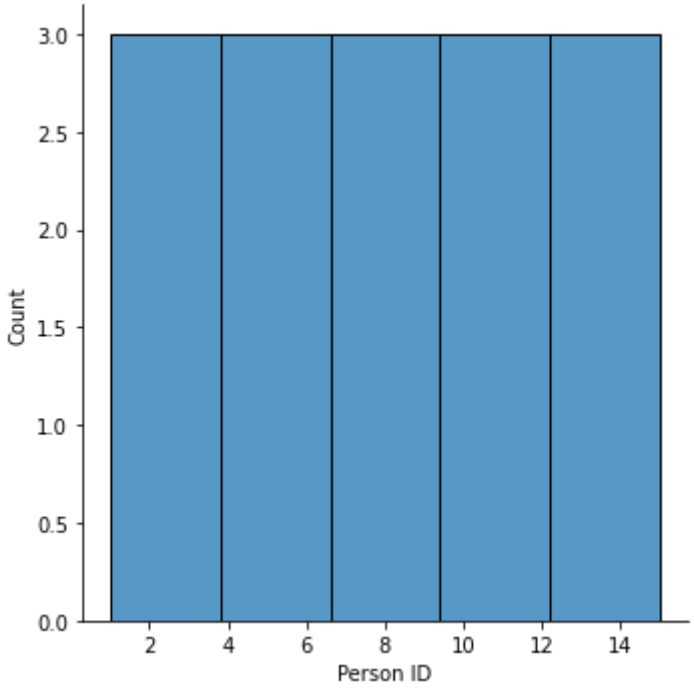
Out[11]:
```
<seaborn.axisgrid.PairGrid at 0x2366bf08ac0>
```



Distribution Plot

In [12]:
```
sns.displot(c['Person ID'])
```

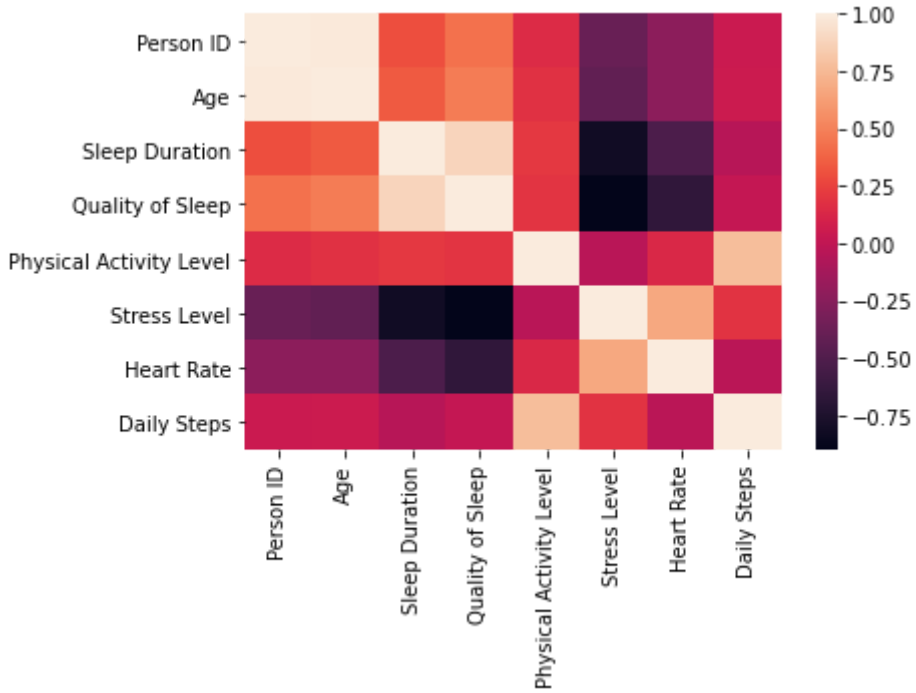
Out[12]:
```
<seaborn.axisgrid.FacetGrid at 0x2366f42d6a0>
```



Correlation

In [14]:
```
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[14]: <AxesSubplot:>



Train the model - Model Building

In [24]:
```
g=c[['Person ID']]
h=c['Age']
```

To split dataset into training end test

In [25]:
```
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 [26]:
```
from sklearn.linear_model import LinearRegression
```

In [27]:
```
lr=LinearRegression()
lr.fit(g_train,h_train)
```

Out[27]:
```
LinearRegression()
```

In [28]:
```
print(lr.intercept_)
```

```
27.412154696132596
```

Coeffecient

In [29]:
```
coeff=pd.DataFrame(lr.coef_,g.columns,columns=['Co-effecient'])
coeff
```

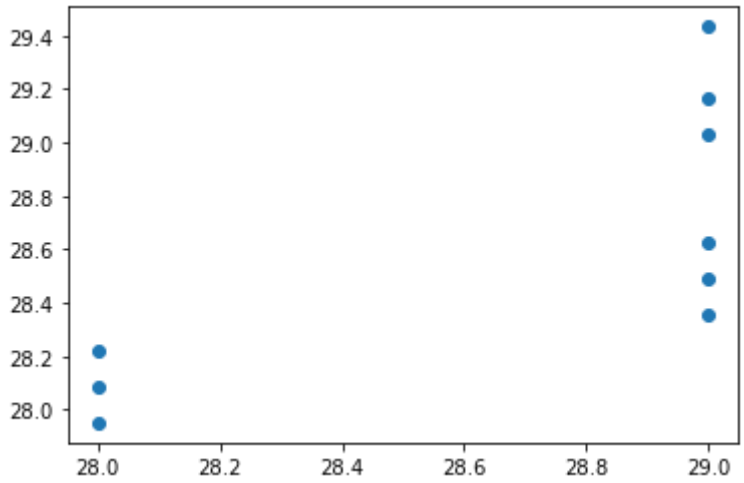
Out[29]:

	Co-effecient
Person ID	0.134807

Best Fit line

In [30]:
```
prediction=lr.predict(g_test)
plt.scatter(h_test,prediction)
```

Out[30]: <matplotlib.collections.PathCollection at 0x236719526d0>



To find score

In [31]:
```
print(lr.score(g_test,h_test))
```

```
0.4550044259943232
```

Import Lasso and ridge

In [32]:
```
from sklearn.linear_model import Ridge,Lasso
```

Ridge

In [33]:
```
ri=Ridge(alpha=5)
ri.fit(g_train,h_train)
```

Out[33]: Ridge(alpha=5)

In [34]:
```
ri.score(g_test,h_test)
```

Out[34]: 0.4665732505933836

In [35]:
```
ri.score(g_train,h_train)
```

Out[35]: 0.821473876862364

Lasso

In [36]:
```
l=Lasso(alpha=6)
l.fit(g_train,h_train)
```

Out[36]: Lasso(alpha=6)

In [37]:
```
l.score(g_test,h_test)
```

Out[37]:
```
-0.5000000000000036
```

In [38]:
```
ri.score(g_train,h_train)
```

Out[38]:
```
0.821473876862364
```

In [ ]:

In [ ]: