In [1]:

```
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 sklearn.linear_model import LinearRegression
```

In [2]:

```
df = pd.read_csv("Sleep.csv")
# .dropna(axis="columns")
df
```

Out[2]:

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Blo Pressu
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()
```

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

Data cleaning and pre processing

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	300
25%	94.250000	35.250000	6.400000	6.000000	45.000000	4.000000	68.000000	560
50%	187.500000	43.000000	7.200000	7.000000	60.000000	5.000000	70.000000	700
75%	280.750000	50.000000	7.800000	8.000000	75.000000	7.000000	72.000000	800
max	374.000000	59.000000	8.500000	9.000000	90.000000	8.000000	86.000000	1000

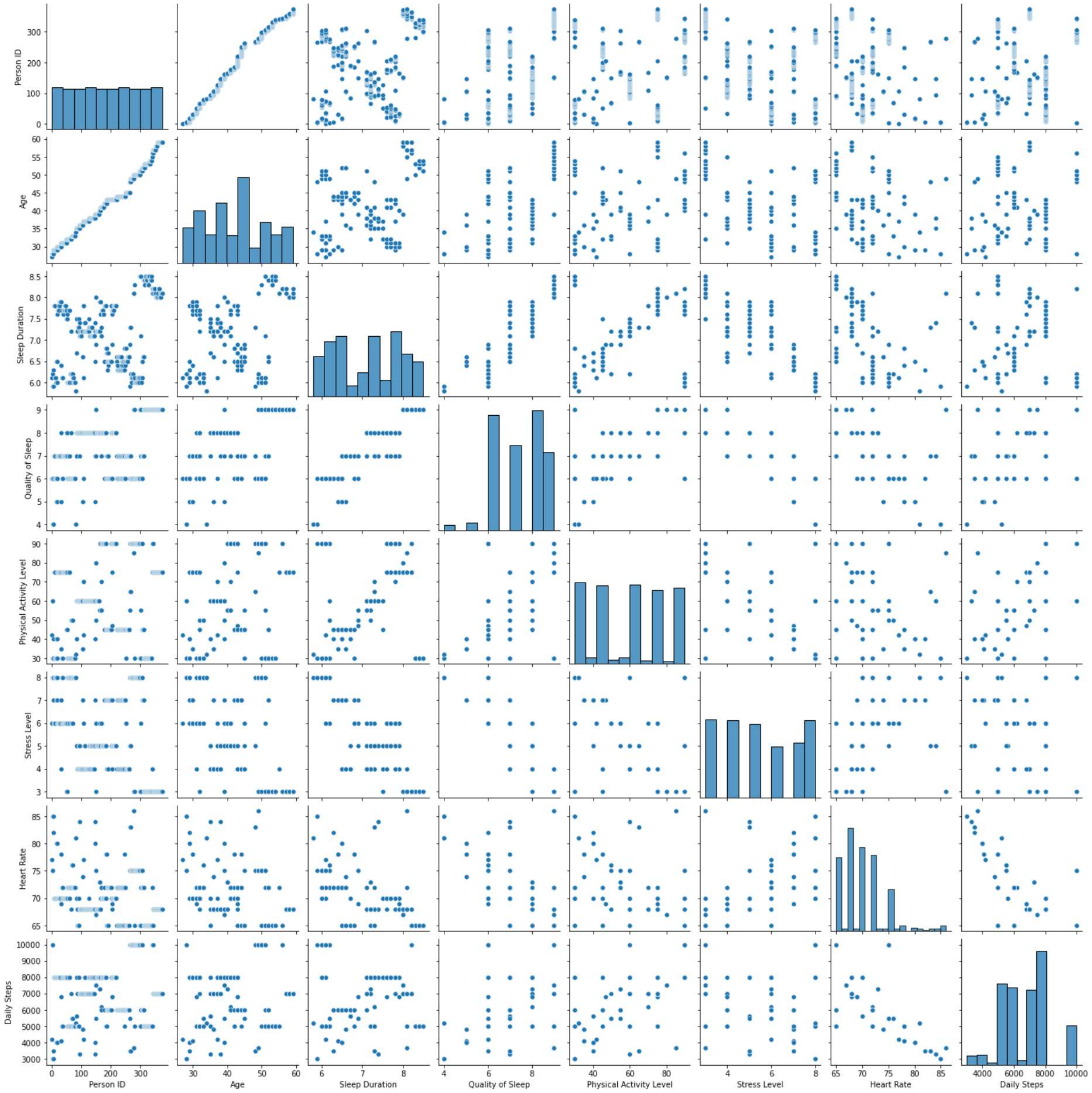
In [6]:
```
df.columns
```

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')
```

EDA and VISUALIZATION

In [7]:
```
sns.pairplot(df)
```

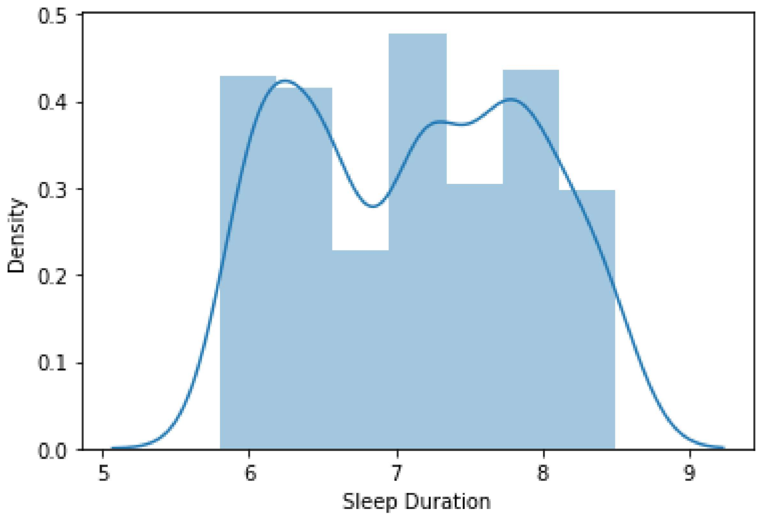
Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x23999011310>
```



```
In [8]: sns.distplot(df['Sleep Duration'])
```

```
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)
```

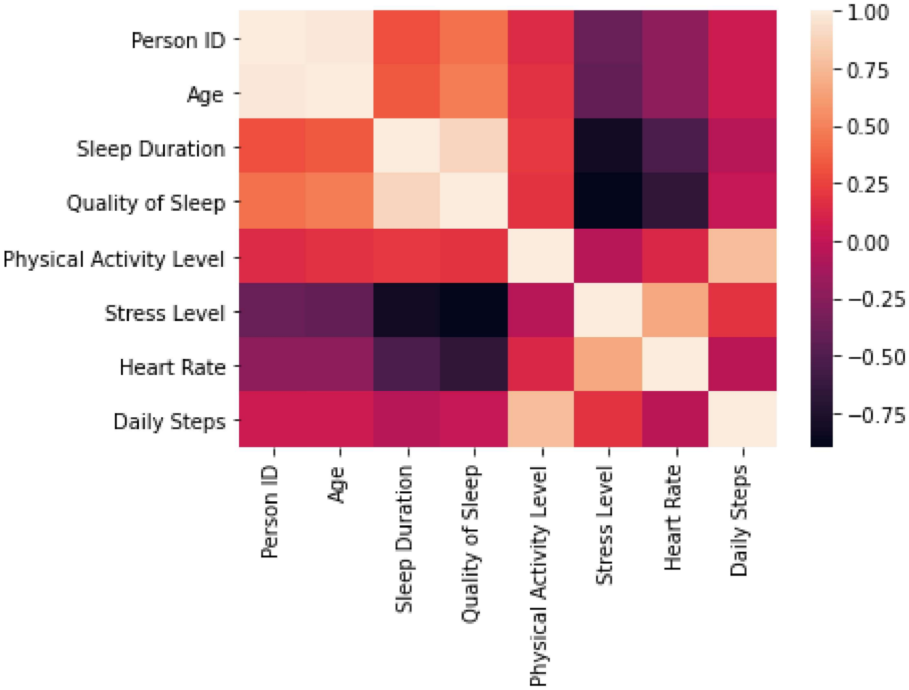
```
Out[8]: <AxesSubplot:xlabel='Sleep Duration', ylabel='Density'>
```



```
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',  'Sleep Duration',
                'Quality of Sleep','Stress Level',
                 'Heart Rate', 'Daily Steps']]
         y = df1[ 'Age']
```

split the data into training and test data

```
In [12]: x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3)
```

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

Out[13]: LinearRegression()
```

```
In [14]: lr.intercept_

Out[14]: 14.88193886989994
```

In [15]:

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

Out[15]:

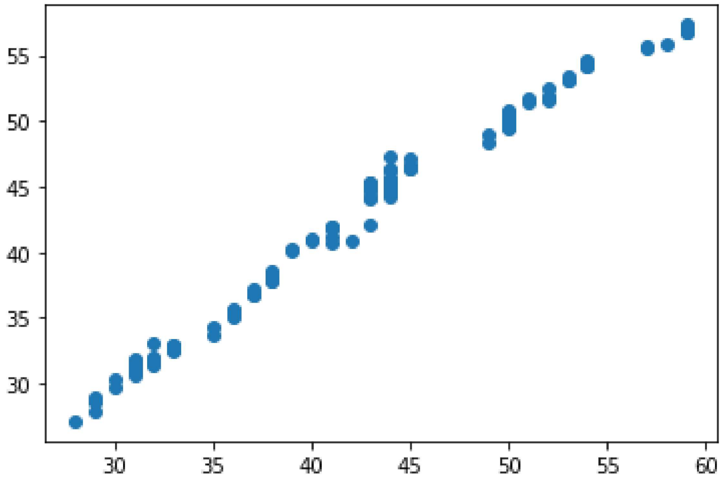
	Co-efficient
Person ID	0.077752
Sleep Duration	0.429799
Quality of Sleep	0.434789
Stress Level	0.104685
Heart Rate	0.077327
Daily Steps	0.000081

In [16]:

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

Out[16]:

```
<matplotlib.collections.PathCollection at 0x2399b116670>
```



In [17]:

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

Out[17]:

```
0.9844590789203896
```

ACURACY

In [18]:

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

In [19]:

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

Out[19]:

```
0.9860900449839386
```

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

Out[20]:
```
0.9843961753552044
```

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]:
```
0.9787346764362932
```

In [23]:
```
from sklearn.linear_model import ElasticNet
en = ElasticNet()
en.fit(x_train,y_train)
```

Out[23]:
```
ElasticNet()
```

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

```
[ 7.95488851e-02  0.00000000e+00  0.00000000e+00 -1.86403127e-03
 -0.00000000e+00  8.04251363e-05]
```

In [25]:
```
print(en.intercept_)
```

```
26.809554637021204
```

In [26]:
```

print(en.predict(x_test))
```

```
[37.54634397 48.12645721 54.01406249 28.31308121 56.87957485 38.58146726
 56.32273266 46.21728397 46.69457728 35.00176743 50.03167933 32.28789671
 40.5692016  45.50134401 37.14958733 54.17316026 53.85496472 33.16293445
 34.28407496 56.95912374 50.1907771  50.11122822 45.81953955 36.59274513
 44.54664586 46.92675559 40.17145718 36.67229402 56.6409282  33.32203222
 45.18314847 55.44769492 52.10488925 45.89349634 31.41921579 47.41051725
 35.55860963 27.35411334 45.66044178 46.92763184 31.33966691 46.13214299
 35.39951186 47.72871279 49.47977604 52.74128033 31.10102025 54.09361137
 44.14890143 51.22985151 40.64875049 31.89015229 30.62372694 52.01876049
 53.77541583 47.49006613 50.27032599 33.93272159 35.24041409 47.08672961
 48.99754383 42.71154094 43.83070589 50.98626596 35.1608652  31.8106034
 29.74868921 40.64513396 51.94085258 32.6911217  33.00383668 40.01235941
 32.92428779 44.78529251 37.78597841 32.20834783 41.36469046 30.78282471
 37.9440884  50.58852153 51.3044615  28.07816262 28.39635816 29.19184701
 30.22598252 44.62619474 43.51251035 51.22491261 40.48965272 46.29124076
 44.70574363 41.52378823 55.12949938 52.66173144 33.4015811  36.7518429
 56.48183043 53.05947587 30.70327583 31.02147137 49.47483714 51.85966272
 33.08338556 50.90671707 28.62963578 46.05259411 55.28859715 44.8648414
 52.09830937 41.43875878 41.44423934 56.56137931 38.02363728]
```

```
In [27]: print(en.score(x_test,y_test))
```

```
0.9787013008337877
```

```
In [28]: # Evaluation Metrics
         from sklearn import metrics
```

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

```
Mean Absolute Error: 0.8243892747891743
```

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

```
Mean Squared Error: 1.16793387145223
```

```
In [31]: print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,pred
```

```
Root Mean Squared Error: 1.0807098923634548
```