Data collection

Importing libraries

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

Importing dataset

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

Out[2]:

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Blood Pressure	Hea Ra
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

head

In [3]:
```
# to display first 8 dataset values
da=data.head(8)
da
```

Out[3]:

	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
5	6	Male	28	Software Engineer	5.9	4	30	8	Obese	140/90	85
6	7	Male	29	Teacher	6.3	6	40	7	Obese	140/90	82
7	8	Male	29	Doctor	7.8	7	75	6	Normal	120/80	70

info

In [4]:
```
# to identify missing values
data.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
```

describe

In [5]:
```
# to display summary of the dataset
data.describe()
```

Out[5]:

	Person ID	Age	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	Heart Rate	Daily Steps
count	374.000000	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	6816.844920
std	108.108742	8.673133	0.795657	1.196956	20.830804	1.774526	4.135676	1617.915679
min	1.000000	27.000000	5.800000	4.000000	30.000000	3.000000	65.000000	3000.000000
25%	94.250000	35.250000	6.400000	6.000000	45.000000	4.000000	68.000000	5600.000000
50%	187.500000	43.000000	7.200000	7.000000	60.000000	5.000000	70.000000	7000.000000
75%	280.750000	50.000000	7.800000	8.000000	75.000000	7.000000	72.000000	8000.000000
max	374.000000	59.000000	8.500000	9.000000	90.000000	8.000000	86.000000	10000.000000

columns

In [6]:
```
# to display headings of the dataset
data.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]:
```
a=data.dropna(axis=1)
a
```

Out[7]:

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Blood Pressure	Hea Ra
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	Hea Ra
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

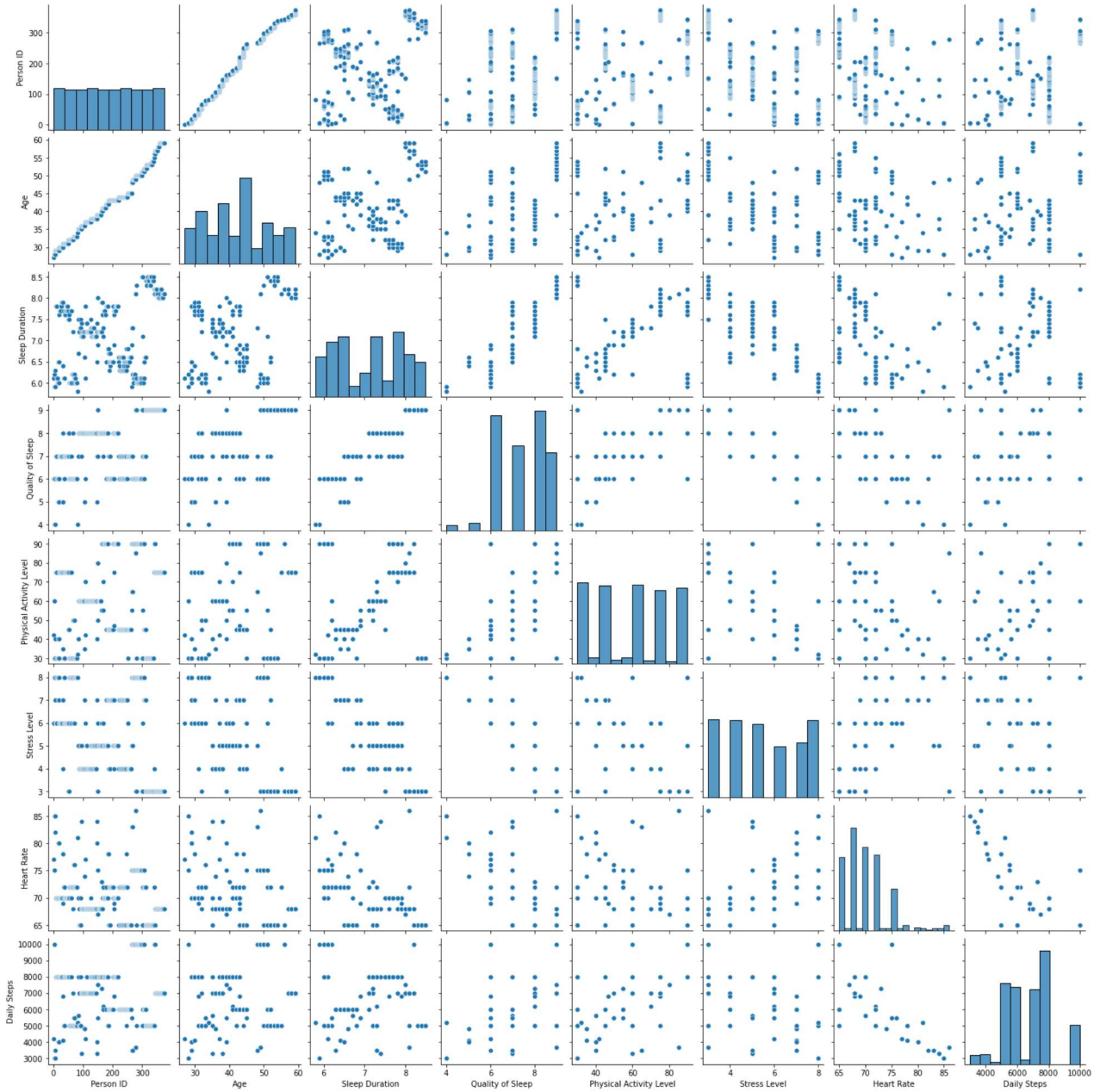
In [8]:
```
a.columns
```

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')
```

EDA and Visualization

In [9]:
```
sns.pairplot(a)
```

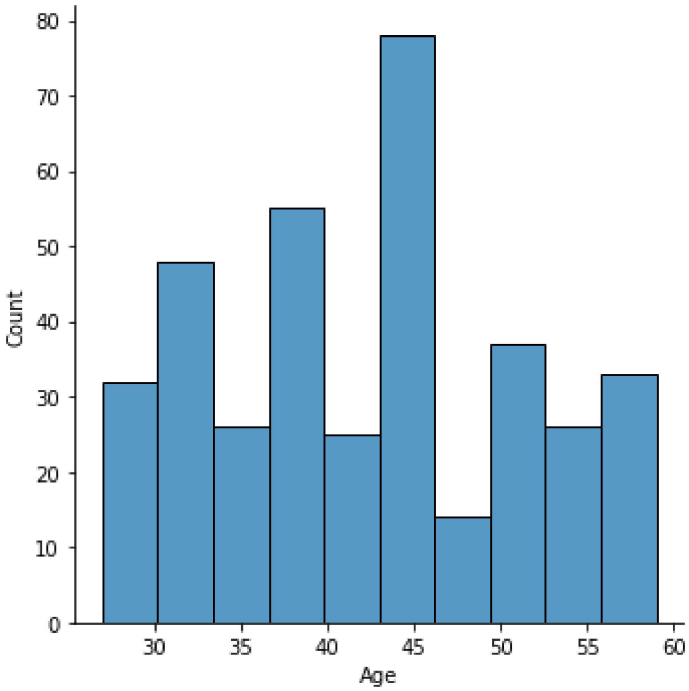
Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x1f1ec9b4f10>
```



distribution plot

In [10]:
```
sns.displot(a["Age"])
```

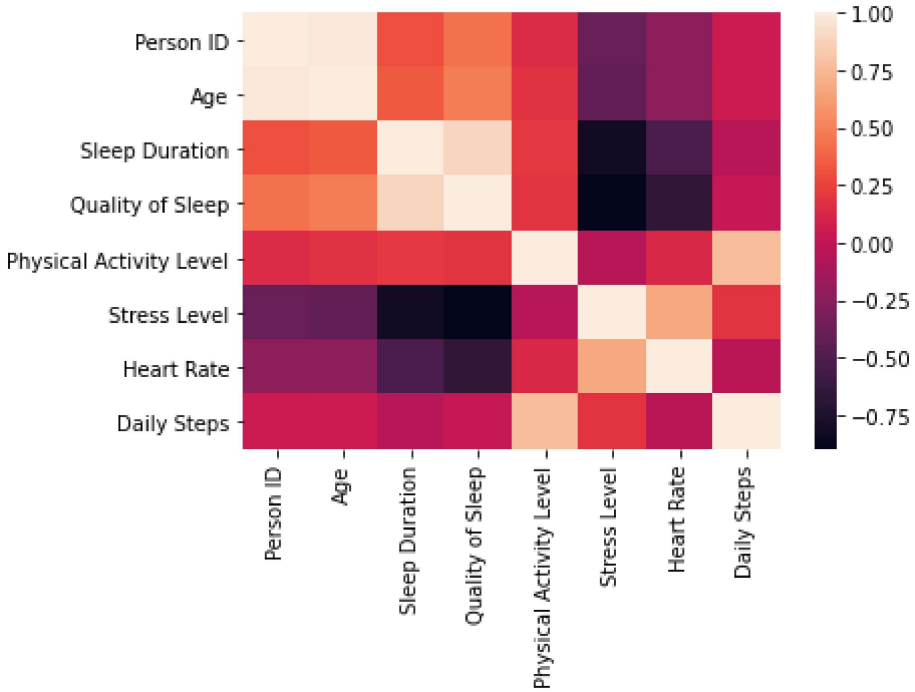
Out[10]:
```
<seaborn.axisgrid.FacetGrid at 0x1f1efe84460>
```



correlation

In [11]:
```
dat=data[['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(dat.corr())
```

Out[11]: <AxesSubplot:>



To train the model-Model Building

In [12]:
```
x=a[['Quality of Sleep']]
y=a['Stress Level']
```

In [13]:
```
# to split my dataset into training and test data
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 [14]:
```
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(x_train,y_train)
```

Out[14]:
```
LinearRegression()
```

In [15]:
```
print(lr.intercept_)
```

```
14.938556763285025
```

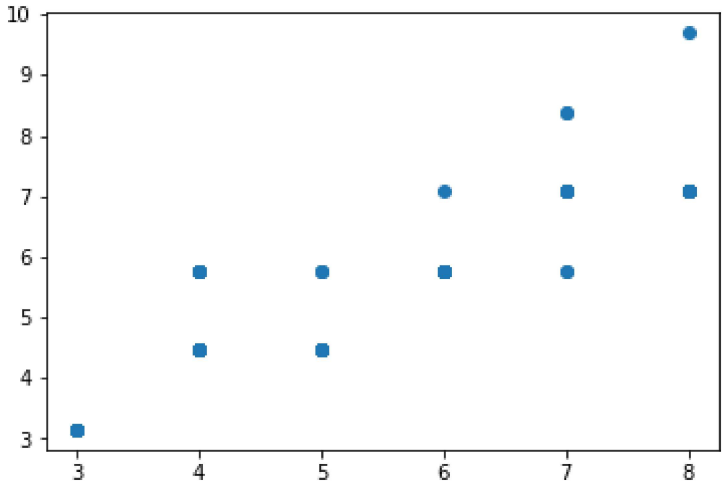
In [16]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[16]:

	Co-efficient
Quality of Sleep	-1.309692

In [17]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[17]:
```
<matplotlib.collections.PathCollection at 0x1f1f1fc2820>
```



In [18]:
```
print(lr.score(x_test,y_test))
```

```
0.8314725878297353
```

In [19]:
```
lr.score(x_train,y_train)
```

Out[19]: 0.7968707728960381

Ridge regression

In [20]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [21]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[21]: 0.8287153448247428

In [22]:
```
rr.score(x_train,y_train)
```

Out[22]: 0.7963487085019322

Lasso regression

In [23]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
la.score(x_train,y_train)
```

Out[23]: 0.0

In [24]:
```
la.score(x_test,y_test)
```

Out[24]: -0.00011391558937812185

In [25]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[25]: ElasticNet()

In [26]:
```
print(en.coef_)
```

```
[-0.72005801]
```

In [27]:
```
print(en.intercept_)
```

```
10.63490618008477
```

In [28]:
```
predict=en.predict(x_test)
```

In [29]:
```
print(en.score(x_test,y_test))
```

```
0.6409634092021227
```

In [30]:
```
from sklearn import metrics
```

In [31]:
```
print("Mean Absolute error:",metrics.mean_absolute_error(y_test,predict))
```

```
Mean Absolute error: 0.8951490551547502
```

In [32]:
```
print("Mean Squared error:",metrics.mean_squared_error(y_test,predict))
```

```
Mean Squared error: 1.128199872282405
```

In [33]:
```
print("Root squared error:",np.sqrt(metrics.mean_squared_error(y_test,predict)))
```

```
Root squared error: 1.0621675349408892
```