DATA COLLECTION

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

In [2]:
```
# To Import Dataset
sd=pd.read_csv(r"c:\Users\user\Downloads\16_Sleep_health_and_lifestyle_dataset.
sd
```

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]:
```
# to display top 10 rows
sd.head(10)
```

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
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

DATA CLEANING AND PRE_PROCESSING

In [4]:
```
sd.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]:
```
# to display summary of statistics
sd.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

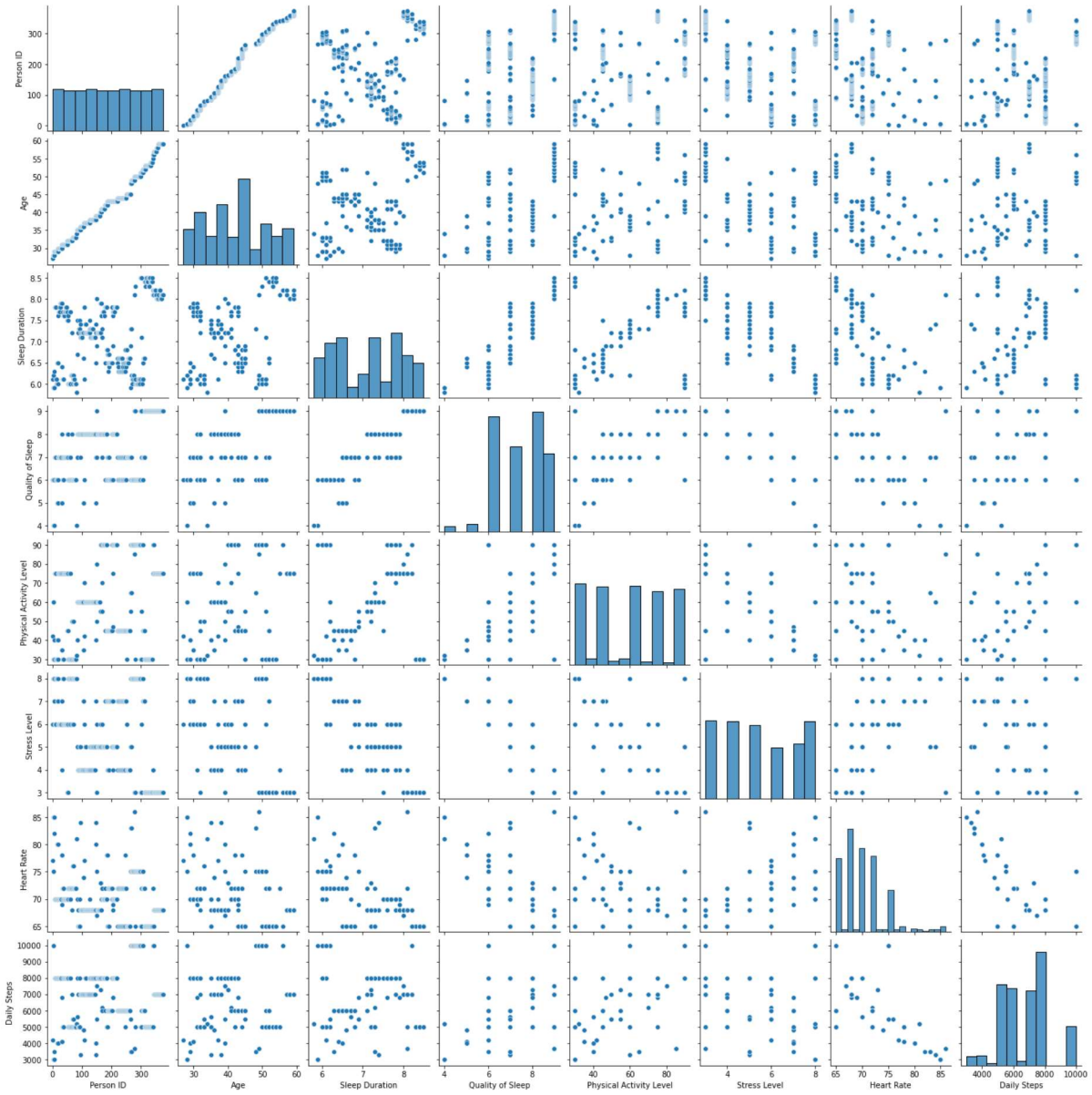
In [6]:
```
#to display colums heading
sd.columns
```

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(sd)
```

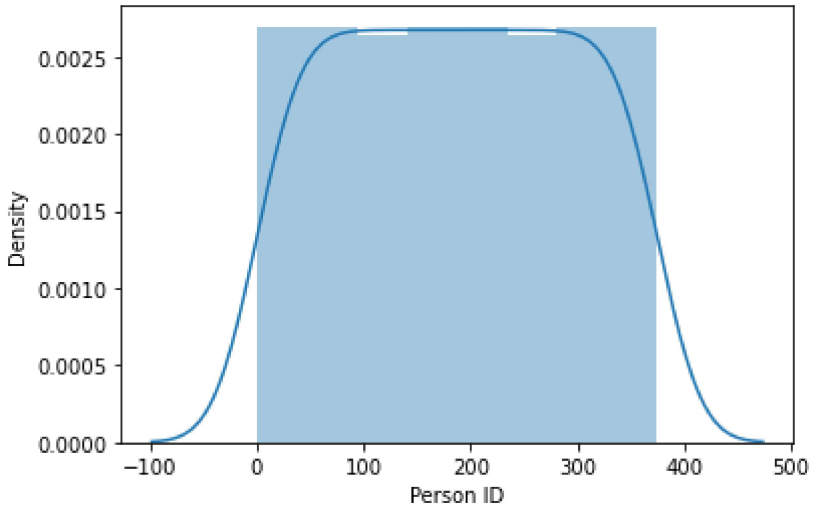
Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x16fa4944370>
```



In [8]:
```
sns.distplot(sd['Person ID'])
```

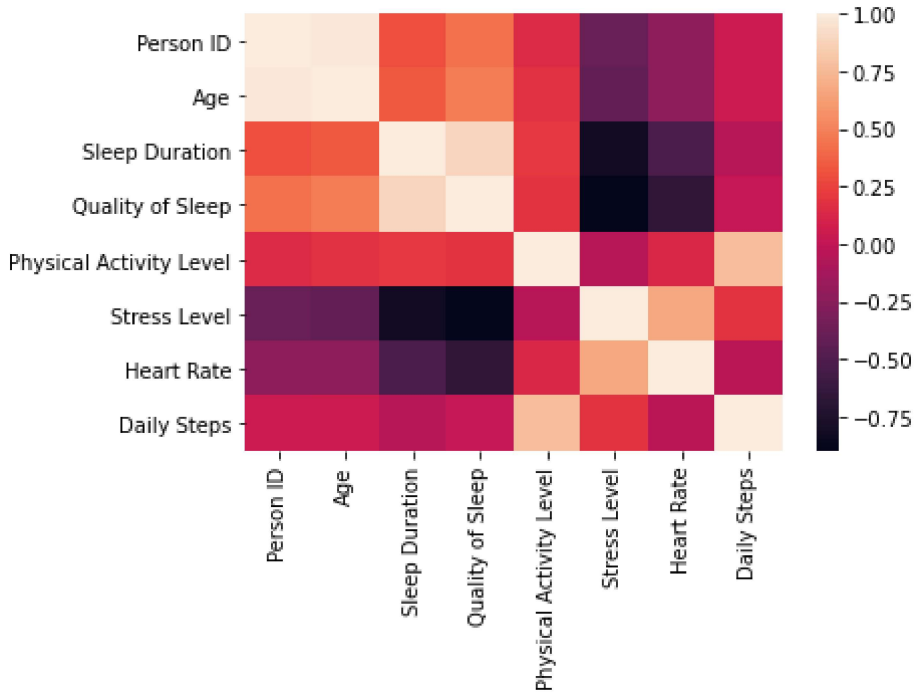
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)
```

Out[8]: <AxesSubplot:xlabel='Person ID', ylabel='Density'>



In [9]:
```
sns.heatmap(sd.corr())
```

Out[9]: <AxesSubplot:>



In [10]:
```
sd1=sd[['Person ID', 'Age','Sleep Duration',
       'Quality of Sleep', 'Physical Activity Level', 'Stress Level']]
```

TO TRAIN THE MODEL _MODEL BUILDING

we are goint train Liner Regression model; we need to split out the data into two varibles x and y where x is independent on x (output) and y is dependent on x(output) adress coloumn as it is not required our model

In [11]:
```
x= sd1[['Person ID', 'Age','Sleep Duration',
       'Quality of Sleep', 'Physical Activity Level']]
y=sd1[ 'Stress Level']
```

In [12]:
```
# To split my dataset  into training data 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 [13]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[13]:
```
LinearRegression()
```

In [14]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

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

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

```
13.084132986920197
```

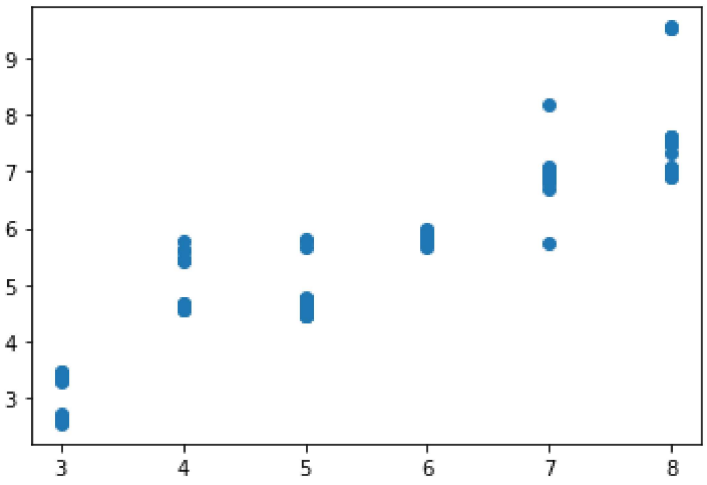
In [16]:
```
coeff= pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[16]:

	Co-efficient
Person ID	-0.007568
Age	0.091085
Sleep Duration	-0.246246
Quality of Sleep	-1.233045
Physical Activity Level	0.011033

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

Out[17]:
```
<matplotlib.collections.PathCollection at 0x16fa9f35730>
```



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

```
0.839438327324136
```

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

Out[19]:
```
0.8298677030363362
```

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

In [21]:
```
dr=Ridge(alpha=10)
dr.fit(x_train,y_train)
```

Out[21]:
```
Ridge(alpha=10)
```

In [22]:
```
dr.score(x_test,y_test)
```

Out[22]:
```
0.8383538907801309
```

In [23]:
```
dr.score(x_train,y_train)
```

Out[23]:
```
0.8288009015081673
```

In [24]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[24]:
```
Lasso(alpha=10)
```

In [25]:
```
la.score(x_test,y_test)
```

Out[25]:
```
0.048975015896049245
```

In [26]:
```
la.score(x_train,y_train)
```

Out[26]:
```
0.18584557182207373
```

ElasticNet

In [27]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

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

In [28]:
```
print(en.coef_)
```

```
[-0.00389015 -0.         -0.04416695 -0.59632909  0.00615756]
```

In [29]:
```
print(en.intercept_)
```

```
10.390365999810541
```

In [30]:
```
prediction=en.predict(x_test)
```

In [31]:
```
print(en.score(x_test,y_test))
```

```
0.5916367410289955
```

Evaluation metric

In [32]:
```
from sklearn import metrics
```

In [33]:
```
print("mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
mean Absolute Error: 0.9090583883666242
```

In [34]:
```
print("mean squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
mean squared Error: 1.229151342727824
```

In [35]:
```
print("Root mean Absolytre Error:",np.sqrt(metrics.mean_squared_error(y_test,pr
```

```
Root mean Absolytre Error: 1.1086709803759742
```

In [ ]: