kaviyadevi 20106064

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

In [2]:
```
#to import dataset
data=pd.read_csv(r"C:\Users\user\Downloads\16_Sleep_health_and_lifestyle_dataset
data
```

Out[2]:

	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

In [3]:
```
#to display top 5 rows
data.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 PREPROCESSING

In [4]:
```
#
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
```

In [5]:
```
#to display summary of statistics(here to know min max value)
data.describe()
```

Out[5]:

	Person ID	Age	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	Heart Rate	Daily
count	374.000000	374.000000	374.000000	374.000000	374.000000	374.000000	374.000000	374.
mean	187.500000	42.184492	7.132086	7.312834	59.171123	5.385027	70.165775	6816.
std	108.108742	8.673133	0.795657	1.196956	20.830804	1.774526	4.135676	1617.
min	1.000000	27.000000	5.800000	4.000000	30.000000	3.000000	65.000000	3000.
25%	94.250000	35.250000	6.400000	6.000000	45.000000	4.000000	68.000000	5600.
50%	187.500000	43.000000	7.200000	7.000000	60.000000	5.000000	70.000000	7000.
75%	280.750000	50.000000	7.800000	8.000000	75.000000	7.000000	72.000000	8000.
max	374.000000	59.000000	8.500000	9.000000	90.000000	8.000000	86.000000	10000.

In [6]:
```
#to display the column heading
data.columns
```

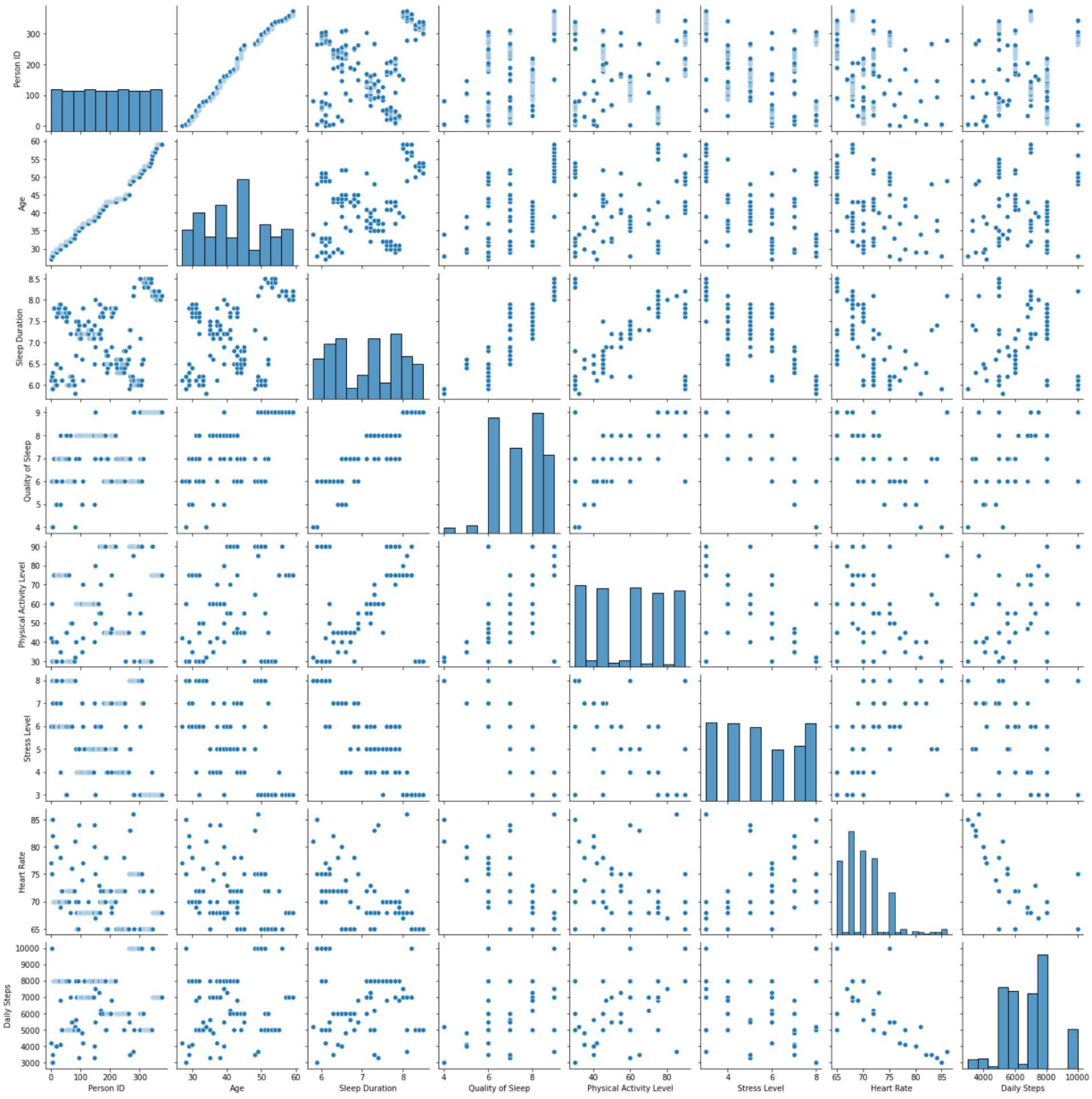
Out[6]:
```
Index(['Person ID', 'Gender', 'Age', 'Occupation', 'Sleep Duration',
       'Quality of Sleep', 'Physical Activity Level', 'Stress Level',
       'BMI Category', 'Blood Pressure', 'Heart Rate', 'Daily Steps',
       'Sleep Disorder'],
      dtype='object')
```

In [7]:
```
#here there is no missing values (identified through info() 5000 data are descrit
```

EDA and DATA VISUALIZATION

In [8]:
```
sns.pairplot(data)
```

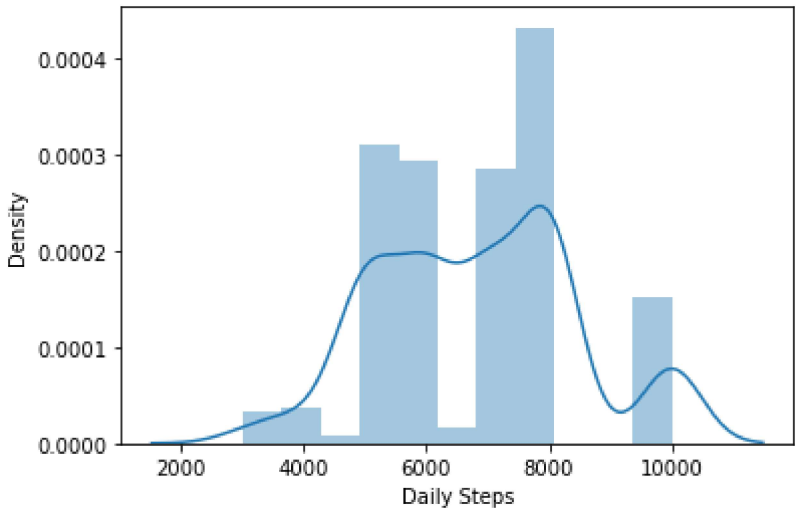
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x232a35d34f0>
```



In [9]:
```
sns.distplot(data['Daily Steps'])
```

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

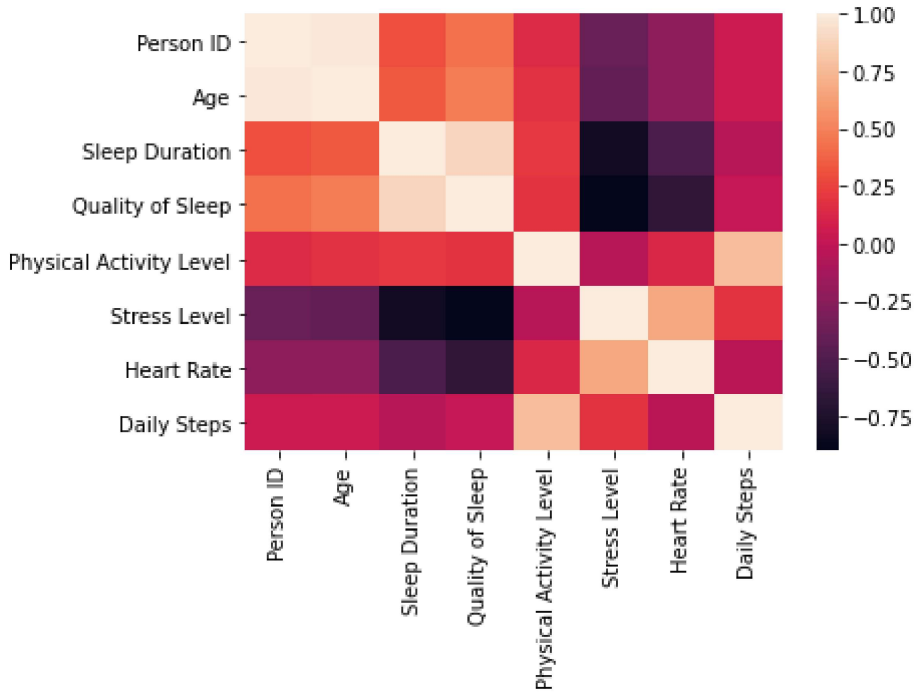
Out[9]: <AxesSubplot:xlabel='Daily Steps', ylabel='Density'>



In [10]:
```
df=data[['Person ID', 'Age', 'Occupation', 'Sleep Duration',
       'Quality of Sleep', 'Physical Activity Level', 'Stress Level',
       'BMI Category', 'Blood Pressure', 'Heart Rate', 'Daily Steps',
       'Sleep Disorder']]
```

In [11]:
```
sns.heatmap(df.corr())
```

Out[11]: <AxesSubplot:>



TRAINING MODEL

In [12]:
```
x=df[['Person ID', 'Age', 'Sleep Duration','Quality of Sleep', 'Physical Activity
y=df['Daily Steps']
```

In [13]:
```
#to split my dataset into trainning and test

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]:
```
#to find intercept
print(lr.intercept_)
```

```
13276.179851555607
```

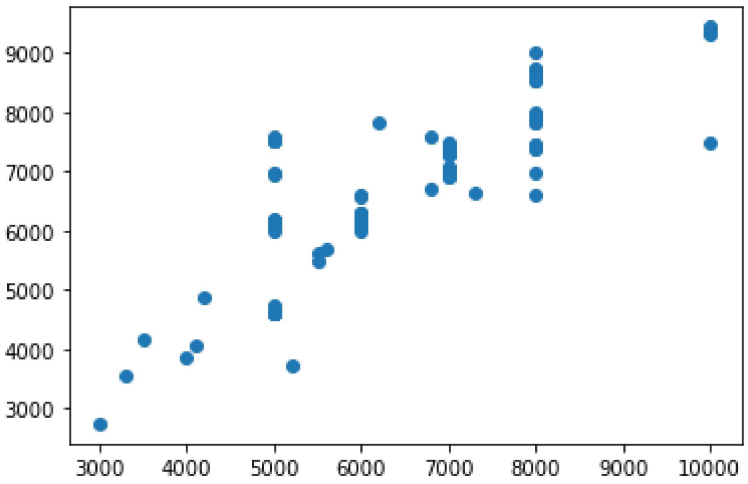
In [16]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[16]:

	Co-efficient
Person ID	-6.670189
Age	78.791777
Sleep Duration	-602.451388
Quality of Sleep	439.836998
Physical Activity Level	66.606520
Stress Level	546.369890
Heart Rate	-203.237425

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

Out[17]:
```
<matplotlib.collections.PathCollection at 0x232ac2a6e80>
```



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

```
0.6914975770433521
```

RIDGE AND LASSO REGRESSION

In [19]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [20]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[20]:
```
Ridge(alpha=10)
```

In [21]:
```
rr.score(x_test,y_test)
```

Out[21]:
```
0.6999186769236416
```

In [22]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[22]:
```
Lasso(alpha=10)
```

In [23]:
```
la.score(x_test,y_test)
```

Out[23]:
```
0.6986795130521921
```

In [24]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[24]:
```
ElasticNet()
```

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

```
[  -3.25264573   38.27340592 -164.41908556  -56.60173052   68.007223
  323.8306963  -183.31182676]
```

In [26]:
```
print(en.predict(x_test))
```

```
[7773.15183843 8768.06846069 6894.91939094 7145.6751954  6875.40351654
 5678.27558254 6437.50052434 7389.27178364 6216.45469947 5691.28616547
 7182.03396143 8179.59837148 7148.92784114 7194.75966689 9150.16593343
 8681.80430603 6188.62830634 7420.1717112  4788.92688    6404.79538712
 8763.8787922  5642.06786994 6862.39293362 6810.82268523 4752.21075419
 4339.08633457 7374.52524071 6908.86699662 7188.53925289 5708.30773699
 7443.90926062 6889.17244233 3474.19595879 6301.0234885  4547.62256991
 8714.50944325 6199.79048773 9086.3034113  7438.41882942 5684.78087401
 8182.85101721 8169.84043428 5638.19949642 7392.52442938 5714.0546856
 7416.91906546 6195.13359781 6878.65616228 5657.1539831  5675.02293681
 7423.42435693 6401.54274139 7162.87544682 9173.51411652 6444.58547877
 7395.955755   7482.84546106 4739.20017126 6096.7222935  6430.99523287
 6852.63499642 8754.120855   6814.25793613 4975.71936677 7368.01994924
 6157.14873343 5725.15062862 6921.26695533 8191.44962847 7443.11891122
 6888.67932324 5688.03351974 6231.66817442 6921.87757955 4752.79041715
 6451.09077023 8724.26738044 7399.02972084 8688.3095975  4762.72703424
 4745.70546272 6918.01430959 7415.65030929 9165.84949913 4769.36864518
 4233.90957775 6477.11193609 9182.87107065 7389.45046354 9153.99824213
 8758.31052349 8701.32018042 7178.78131569 4749.53777142 8154.73350266
 6398.29009566 8139.22861686 6176.84328771 6225.81165359 6902.18302526
 4749.71645132 8146.1348914  4623.49826861 7413.66641973 6865.64557935
 9160.10255052 8135.79729124 7418.72427513 8192.02929144 7155.4331326
 6464.10135316 5649.40275403 5718.06567419]
```

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

```
0.6916153809303172
```

In [29]:
```
from sklearn import metrics
```

In [30]:
```
print("Mean Absolute error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute error 592.2018830261603
```

In [31]:
```
print("Mean Squared error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared error 731067.8726985279
```

In [32]:
```
print("Root Mean Absolute error",np.sqrt(metrics.mean_squared_error(y_test,predic
```

```
Root Mean Absolute error 855.0250713859377
```

In [ ]:
```

```