In [ ]:

```
```

In [198]:

```
# IMPORT LIBRARIES
import numpy  as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [199]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\16_Sleep_health_and_lifestyle_dataset.csv")
a
```

Out[199]:

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Pr
0	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	
1	2	Male	28	Doctor	6.2	6	60	8	Normal	
2	3	Male	28	Doctor	6.2	6	60	8	Normal	
3	4	Male	28	Sales Representative	5.9	4	30	8	Obese	
4	5	Male	28	Sales Representative	5.9	4	30	8	Obese	
...	...	...	...	...	...	...	...	...	...	
369	370	Female	59	Nurse	8.1	9	75	3	Overweight	
370	371	Female	59	Nurse	8.0	9	75	3	Overweight	
371	372	Female	59	Nurse	8.1	9	75	3	Overweight	
372	373	Female	59	Nurse	8.1	9	75	3	Overweight	
373	374	Female	59	Nurse	8.1	9	75	3	Overweight	

374 rows × 13 columns

In [200]:

```
a=a.head(10)
a
```

Out[200]:

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Bl Pres
0	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	12
1	2	Male	28	Doctor	6.2	6	60	8	Normal	12
2	3	Male	28	Doctor	6.2	6	60	8	Normal	12
3	4	Male	28	Sales Representative	5.9	4	30	8	Obese	14
4	5	Male	28	Sales Representative	5.9	4	30	8	Obese	14
5	6	Male	28	Software Engineer	5.9	4	30	8	Obese	14
6	7	Male	29	Teacher	6.3	6	40	7	Obese	14
7	8	Male	29	Doctor	7.8	7	75	6	Normal	12
8	9	Male	29	Doctor	7.8	7	75	6	Normal	12
9	10	Male	29	Doctor	7.8	7	75	6	Normal	12

In [201]:

```
# to find
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 13 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Person ID                10 non-null     int64  
 1   Gender                   10 non-null     object 
 2   Age                      10 non-null     int64  
 3   Occupation               10 non-null     object 
 4   Sleep Duration           10 non-null     float64
 5   Quality of Sleep         10 non-null     int64  
 6   Physical Activity Level  10 non-null     int64  
 7   Stress Level             10 non-null     int64  
 8   BMI Category             10 non-null     object 
 9   Blood Pressure           10 non-null     object 
 10  Heart Rate               10 non-null     int64  
 11  Daily Steps              10 non-null     int64  
 12  Sleep Disorder           10 non-null     object 
dtypes: float64(1), int64(7), object(5)
memory usage: 1.1+ KB
```

In [202]:

```
# to display summary of statastic
a.describe()
```

Out[202]:

	Person ID	Age	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	Heart Rate	Daily Ste
count	10.00000	10.000000	10.000000	10.000000	10.000000	10.000000	10.00000	10.0000
mean	5.50000	28.300000	6.590000	5.700000	51.700000	7.100000	77.40000	6070.0000
std	3.02765	0.674949	0.846496	1.251666	19.465354	0.994429	6.41526	2989.6302
min	1.00000	27.000000	5.900000	4.000000	30.000000	6.000000	70.00000	3000.0000
25%	3.25000	28.000000	5.950000	4.500000	32.500000	6.000000	71.25000	3125.0000
50%	5.50000	28.000000	6.200000	6.000000	51.000000	7.500000	76.00000	6100.0000
75%	7.75000	29.000000	7.425000	6.750000	71.250000	8.000000	84.25000	8000.0000
max	10.00000	29.000000	7.800000	7.000000	75.000000	8.000000	85.00000	10000.0000

In [203]:

```
# to display colum heading
a.columns
```

Out[203]:

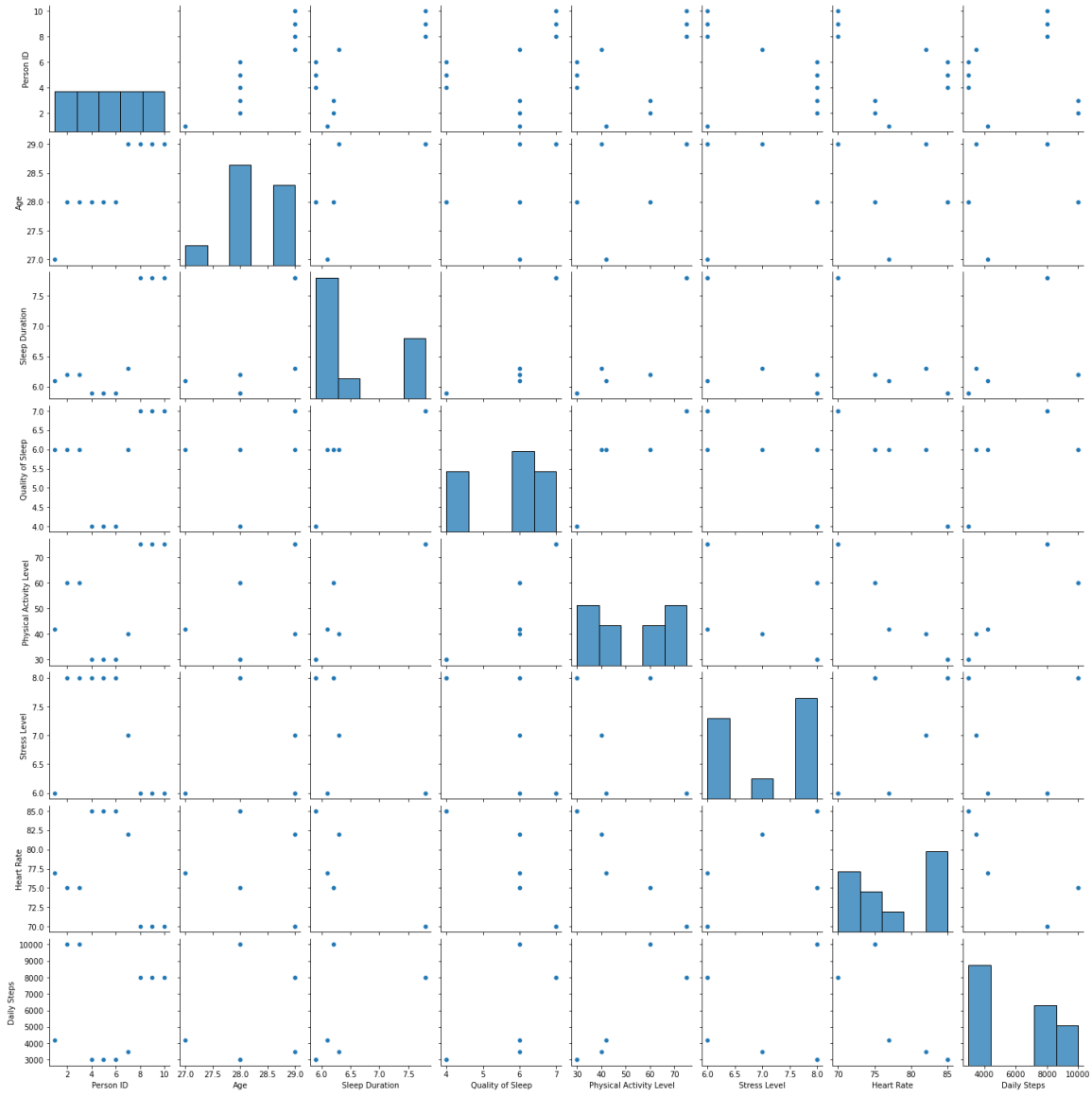
```
Index(['Person ID', 'Gender', 'Age', 'Occupation', 'Sleep Duration',
       'Quality of Sleep', 'Physical Activity Level', 'Stress Level',
       'BMI Category', 'Blood Pressure', 'Heart Rate', 'Daily Steps',
       'Sleep Disorder'],
      dtype='object')
```

In [204]:

```
sns.pairplot(a)
```

Out[204]:

```
<seaborn.axisgrid.PairGrid at 0x198cf314fa0>
```

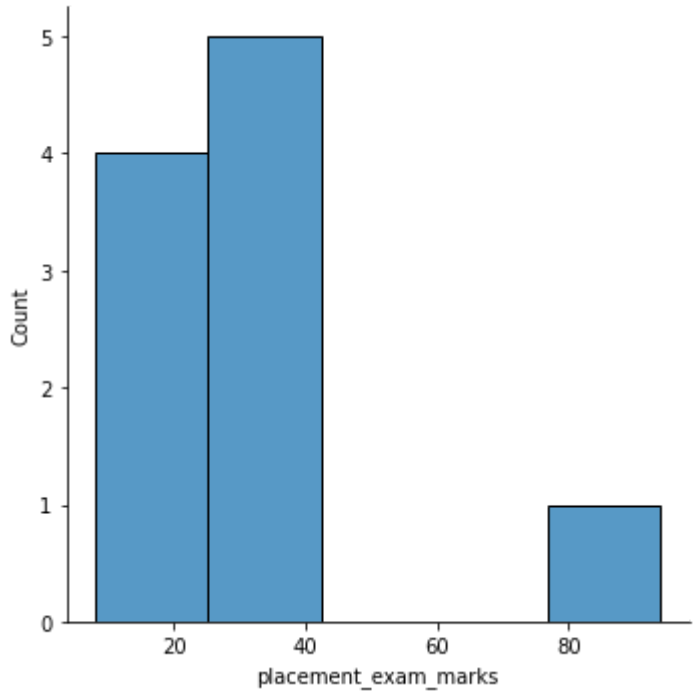


In [172]:

```
sns.displot(a["Sleep Duration"])
```

Out[172]:

```
<seaborn.axisgrid.FacetGrid at 0x198cf088ee0>
```



In [205]:

```
b=a[['Sleep Duration',
       'Quality of Sleep', 'Physical Activity Level', 'Stress Level']]
b
```

Out[205]:

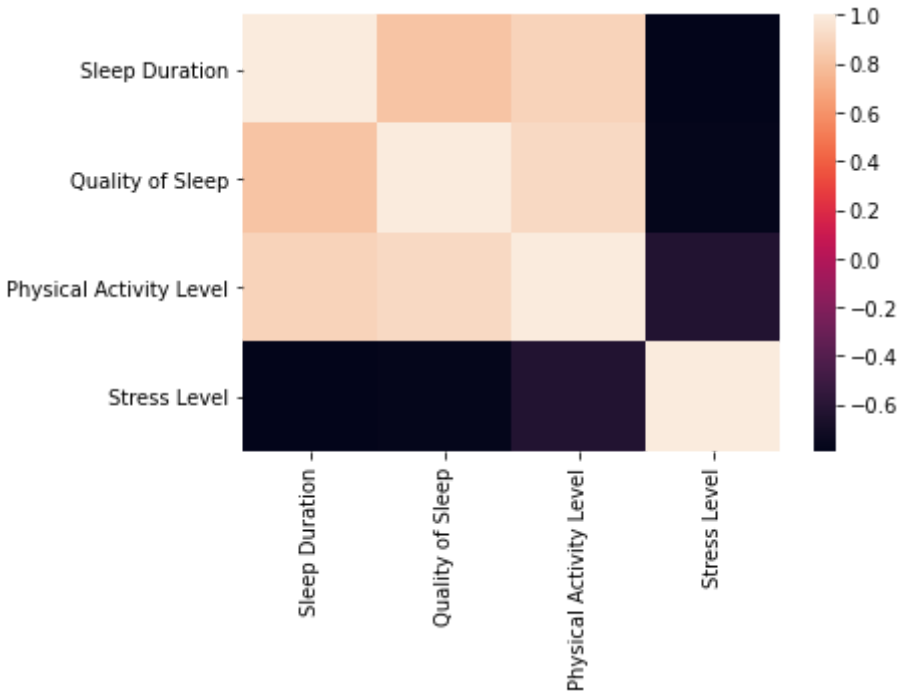
	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level
0	6.1	6	42	6
1	6.2	6	60	8
2	6.2	6	60	8
3	5.9	4	30	8
4	5.9	4	30	8
5	5.9	4	30	8
6	6.3	6	40	7
7	7.8	7	75	6
8	7.8	7	75	6
9	7.8	7	75	6

In [206]:

```
sns.heatmap(b.corr())
```

Out[206]:

<AxesSubplot:>



In [208]:

```
x=a[['Sleep Duration',
       'Quality of Sleep', 'Physical Activity Level', 'Stress Level']]
y=a['Sleep Duration']
```

In [209]:

```
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 [210]:

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[210]:

```
LinearRegression()
```

In [211]:

```
lr.intercept_
```

Out[211]:

```
7.3931193320857975
```

In [212]:

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

Out[212]:

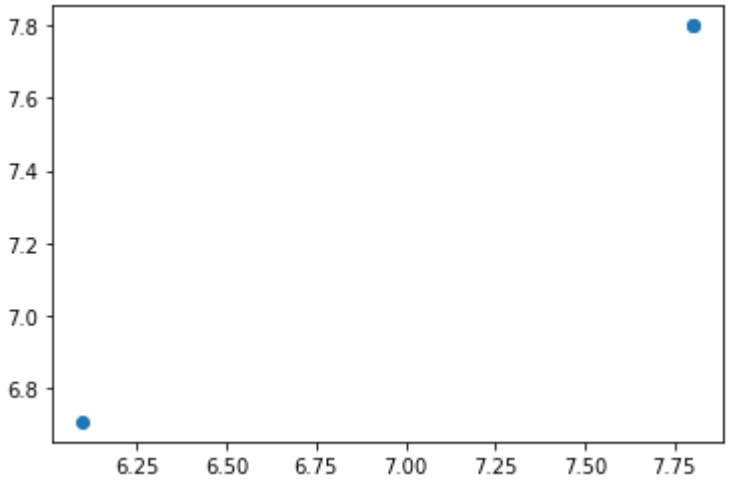
	Co-efficient
Sleep Duration	0.385826
Quality of Sleep	-0.195768
Physical Activity Level	0.019193
Stress Level	-0.445276

In [213]:

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

Out[213]:

```
<matplotlib.collections.PathCollection at 0x198d38c8be0>
```



In [214]:

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

Out[214]:

```
0.8090804258450611
```

In [215]:

```
lr.score(x_train,y_train)
```

Out[215]:

```
1.0
```

In [216]:

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

In [217]:

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

Out[217]:

```
Ridge(alpha=10)
```

In [218]:

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

Out[218]:

```
0.9998166887016773
```

In [219]:

```
la=Lasso(alpha=10)
la.fit(x_test,y_test)
```

Out[219]:

```
Lasso(alpha=10)
```

In [220]:

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

Out[220]:

```
0.3565729646258119
```

In [221]:

```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[221]:

```
ElasticNet()
```

In [222]:

```
en.coef_
```

Out[222]:

```
array([ 0.        ,  0.        ,  0.02763669, -0.        ])
```

In [223]:

```
en.intercept_
```

Out[223]:

```
5.031153455462539
```

In [224]:

```
prediction=en.predict(x_test)
prediction
```

Out[224]:

```
array([6.19189464, 7.10390557, 7.10390557])
```

In [225]:

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

Out[225]:

```
0.4926265327931927
```

EVALUATION METRICS

In [226]:

```
from sklearn import metrics
```

In [227]:

```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 0.49469450185061997
```

In [228]:

```
print("Mean Squared Error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error 0.3258465156061497
```

In [229]:

```
print("Root Mean Squared Error",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
Root Mean Squared Error 0.5708296730252814
```

In [ ]:

```

```