In [1]:

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

In [2]:

```
df1=pd.read_csv(r'C:\Users\user\Downloads\16_Sleep_health_and_lifestyle_dataset.csv')
df1
```

Out[2]:

	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 [3]:

```
df=df1.head(100)
df
```

Out[3]:

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Pre
0	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	1
1	2	Male	28	Doctor	6.2	6	60	8	Normal	1
2	3	Male	28	Doctor	6.2	6	60	8	Normal	1
3	4	Male	28	Sales Representative	5.9	4	30	8	Obese	1
4	5	Male	28	Sales Representative	5.9	4	30	8	Obese	1
...	...	...	...	...	...	...	...	...	...	
95	96	Female	36	Accountant	7.1	8	60	4	Normal	1
96	97	Female	36	Accountant	7.2	8	60	4	Normal	1
97	98	Female	36	Accountant	7.1	8	60	4	Normal	1
98	99	Female	36	Teacher	7.1	8	60	4	Normal	1
99	100	Female	36	Teacher	7.1	8	60	4	Normal	1

100 rows × 13 columns

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 13 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Person ID                100 non-null    int64  
 1   Gender                   100 non-null    object 
 2   Age                      100 non-null    int64  
 3   Occupation               100 non-null    object 
 4   Sleep Duration           100 non-null    float64
 5   Quality of Sleep         100 non-null    int64  
 6   Physical Activity Level  100 non-null    int64  
 7   Stress Level             100 non-null    int64  
 8   BMI Category             100 non-null    object 
 9   Blood Pressure           100 non-null    object 
 10  Heart Rate               100 non-null    int64  
 11  Daily Steps              100 non-null    int64  
 12  Sleep Disorder           100 non-null    object 
dtypes: float64(1), int64(7), object(5)
memory usage: 10.3+ KB
```

In [5]:

```
df.describe()
```

Out[5]:

	Person ID	Age	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	Heart Rate	
count	100.000000	100.00000	100.000000	100.000000	100.000000	100.000000	100.000000	
mean	50.500000	31.69000	6.871000	6.590000	51.910000	6.420000	71.610000	6
std	29.011492	2.26388	0.766903	1.005992	19.429279	1.485145	4.240009	1
min	1.000000	27.00000	5.800000	4.000000	30.000000	3.000000	65.000000	3
25%	25.750000	30.00000	6.100000	6.000000	30.000000	6.000000	70.000000	5
50%	50.500000	31.50000	7.100000	7.000000	60.000000	6.000000	70.000000	7
75%	75.250000	33.00000	7.700000	7.000000	75.000000	8.000000	72.000000	8
max	100.000000	36.00000	7.900000	8.000000	75.000000	8.000000	85.000000	10

In [6]:

```
df.columns
```

Out[6]:

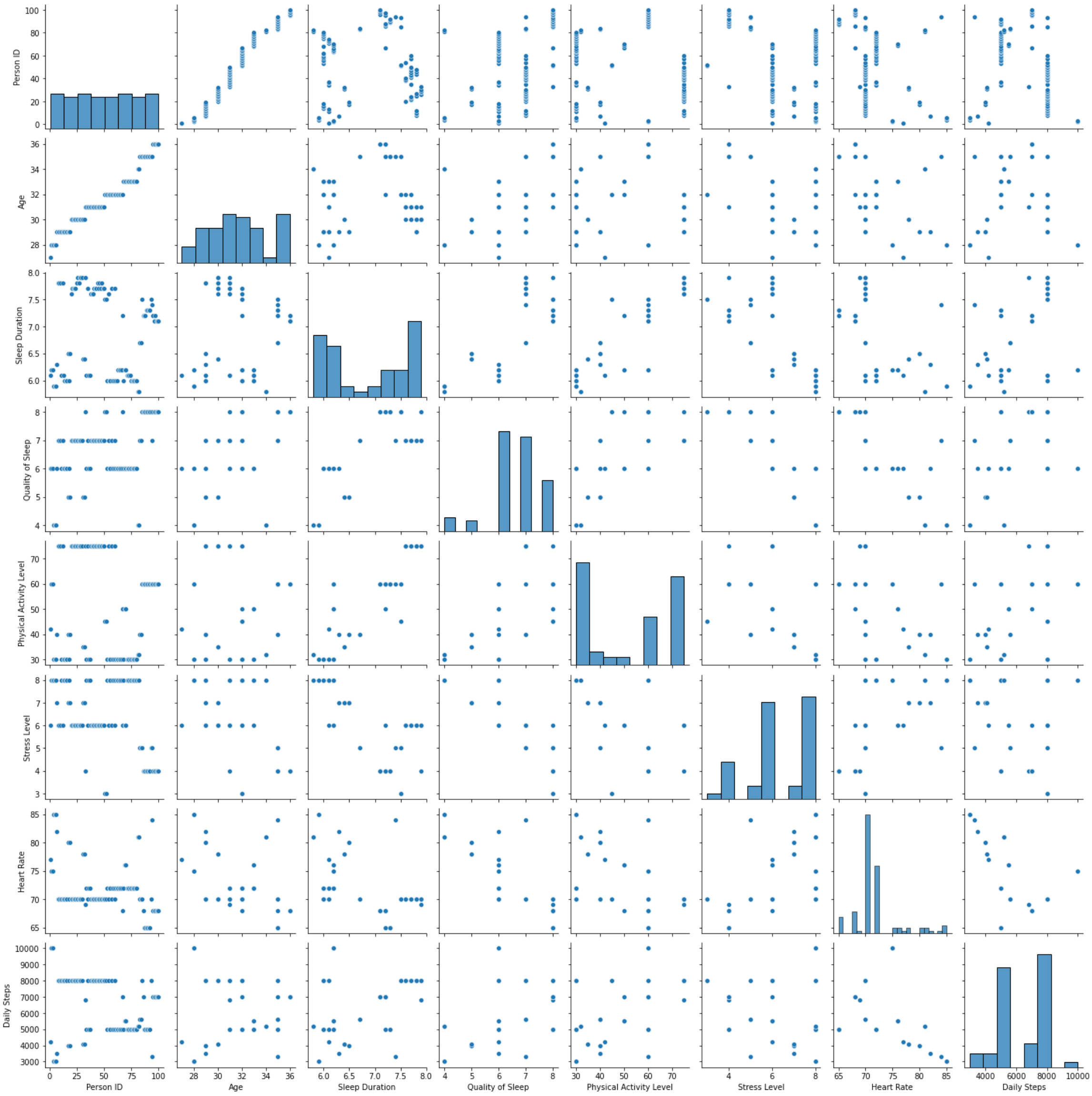
```
Index(['Person ID', 'Gender', 'Age', 'Occupation', 'Sleep Duration',
       'Quality of Sleep', 'Physical Activity Level', 'Stress Level',
       'BMI Category', 'Blood Pressure', 'Heart Rate', 'Daily Steps',
       'Sleep Disorder'],
      dtype='object')
```

In [7]:

```
sns.pairplot(df)
```

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x1dd3f873ac0>
```



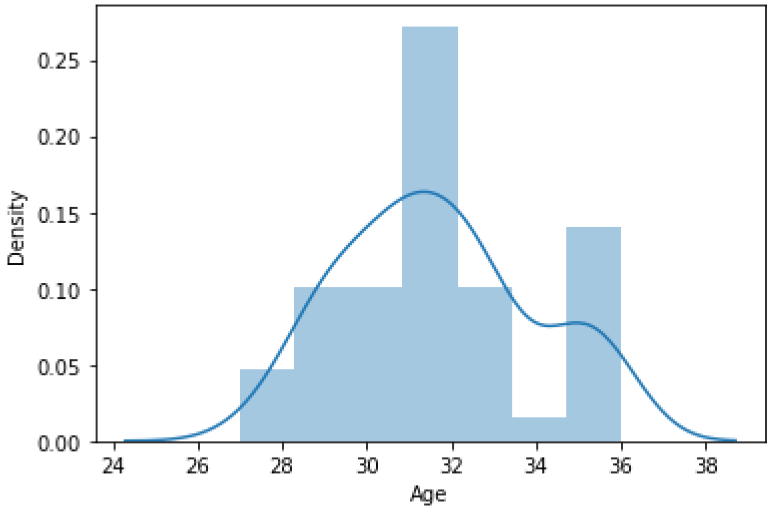
In [8]:

```
sns.distplot(df['Age'])
```

```
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 fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[8]:

```
<AxesSubplot:xlabel='Age', ylabel='Density'>
```

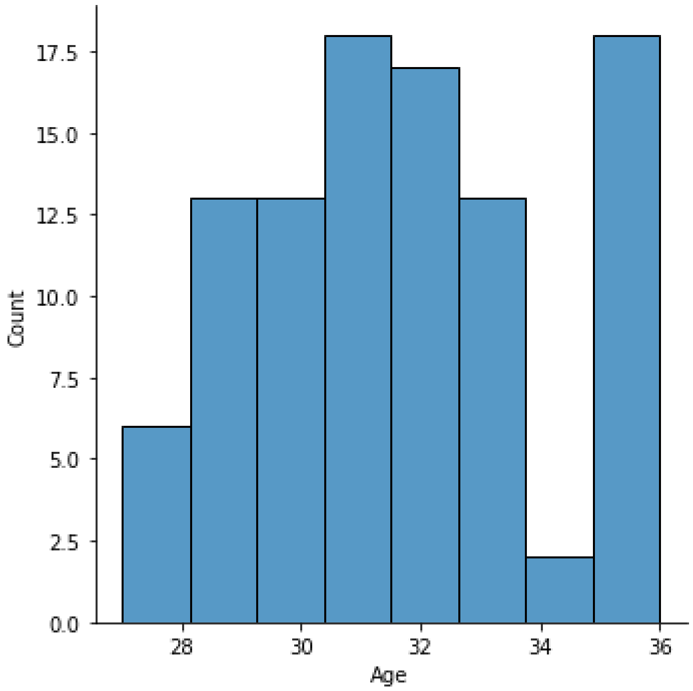


In [9]:

```
sns.displot(df["Age"])
```

Out[9]:

```
<seaborn.axisgrid.FacetGrid at 0x1dd43676910>
```



In [10]:

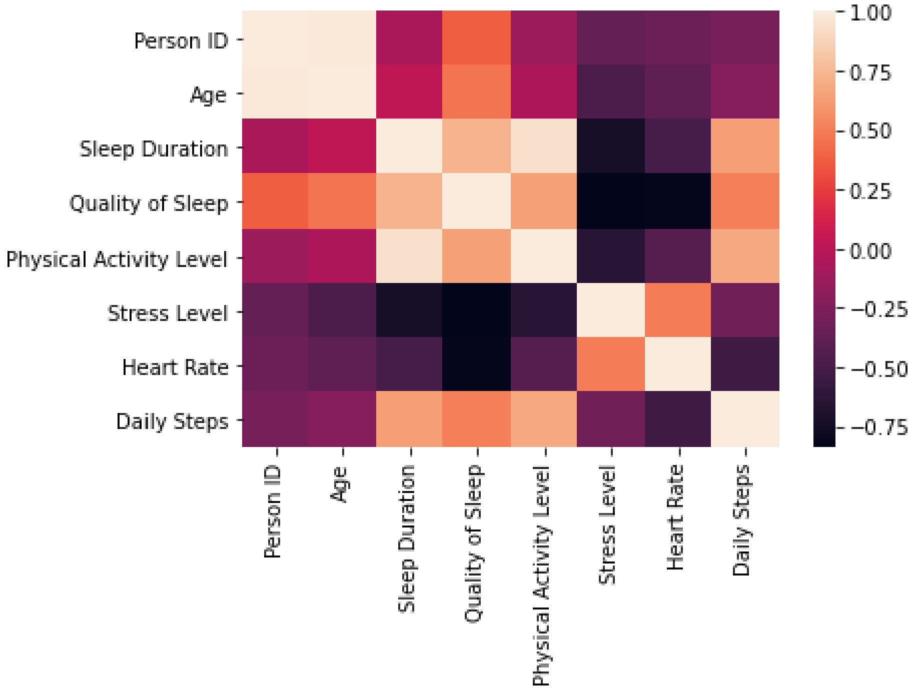
```
df1=df[['Person ID', 'Gender', 'Age', 'Occupation', 'Sleep Duration',
       'Quality of Sleep', 'Physical Activity Level', 'Stress Level',
       'BMI Category', 'Blood Pressure', 'Heart Rate', 'Daily Steps',
       'Sleep Disorder']]
```

In [11]:

```
sns.heatmap(df1.corr())
```

Out[11]:

```
<AxesSubplot:>
```



In [12]:

```
df2=df.dropna(axis=1)
df2
```

Out[12]:

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	I Pre
0	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	1
1	2	Male	28	Doctor	6.2	6	60	8	Normal	1
2	3	Male	28	Doctor	6.2	6	60	8	Normal	1
3	4	Male	28	Sales Representative	5.9	4	30	8	Obese	1
4	5	Male	28	Sales Representative	5.9	4	30	8	Obese	1
...	...	...	...	...	...	...	...	...	...	
95	96	Female	36	Accountant	7.1	8	60	4	Normal	1
96	97	Female	36	Accountant	7.2	8	60	4	Normal	1
97	98	Female	36	Accountant	7.1	8	60	4	Normal	1
98	99	Female	36	Teacher	7.1	8	60	4	Normal	1
99	100	Female	36	Teacher	7.1	8	60	4	Normal	1

100 rows × 13 columns

In [13]:

```
x=df2[['Person ID', 'Sleep Duration',
       'Quality of Sleep', 'Physical Activity Level', 'Stress Level', 'Heart Rate', 'Dail
y=df2[['Age']]
```

In [14]:

```
from sklearn.model_selection import train_test_split
```

In [15]:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [16]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)#ValueError: Input contains NaN, infinity or a value too large for
```

Out[16]:

```
LinearRegression()
```

In [17]:

```
print(lr.intercept_)
```

```
[36.5307651]
```

In [18]:

```
coef= pd.DataFrame(lr.coef_)
coef
```

Out[18]:

	0	1	2	3	4	5	6
0	0.068178	-0.726162	-0.020979	0.015462	-0.357912	-0.021145	-0.000022

In [19]:

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

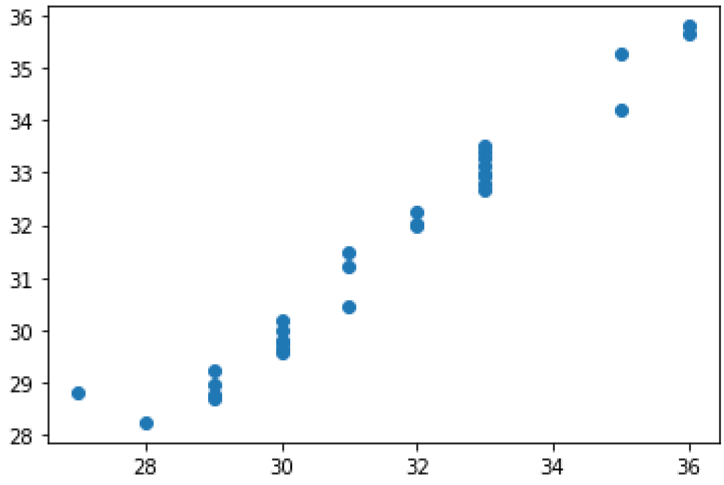
```
0.9600125564551515
```

In [20]:

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

Out[20]:

```
<matplotlib.collections.PathCollection at 0x1dd44f4b7c0>
```



In [21]:

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

Out[21]:

```
0.9600125564551515
```

In [22]:

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

Out[22]:

```
0.979857066694694
```

In [23]:

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

In [24]:

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

Out[24]:

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

In [25]:

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

Out[25]:

```
0.9699957173182402
```

In [26]:

```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[26]:

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

In [27]:

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

Out[27]:

```
0.9300736272958144
```

Elastic Net

In [28]:

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

Out[28]:

```
ElasticNet()
```

In [29]:

```
print(en.coef_)
```

```
[ 7.64351669e-02  0.00000000e+00  0.00000000e+00  4.23550707e-03
 -0.00000000e+00 -0.00000000e+00  1.09760954e-05]
```

In [30]:

```
print(en.intercept_)
```

```
[27.55857198]
```

In [31]:

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

```
[32.17375735 28.72839544 32.25019252 29.79848778 31.70936695 33.39672002
 28.10074131 30.56861884 34.39750035 27.85899804 33.77889585 35.22731109
 30.12130689 33.47315519 29.64561744 29.87492294 34.82318307 28.65196027
 35.38018143 28.84353829 32.63236835 31.40362628 33.10474237 30.18066361
 33.62602552 29.14927896 33.16741452 32.93810902 29.56918228 33.32028485]
```

In [32]:

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

```
0.9636569960225496
```

Evaluation Metrics

In [33]:

```
from sklearn import metrics
```

In [34]:

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

```
Mean Absolute Error: 0.3666843443665551
```

In [35]:

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

```
Mean Squared Error: 0.18833752283425376
```

In [36]:

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

```
Root Mean Squared Error: 0.43397871242061375
```