LinearRegression

In [1]:

```
import numpy as np
import pandas as pd
```

data collection

In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as pp
import seaborn as sb
```

In [3]:

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

Out[3]:

	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

first 10 rows

In [4]:

```
df.head(10)
```

Out[4]:

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	B 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

data cleaning

In [5]:

```
df.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 [6]:

```
df.describe()
```

Out[6]:

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

In [7]:

```
df.columns
```

Out[7]:

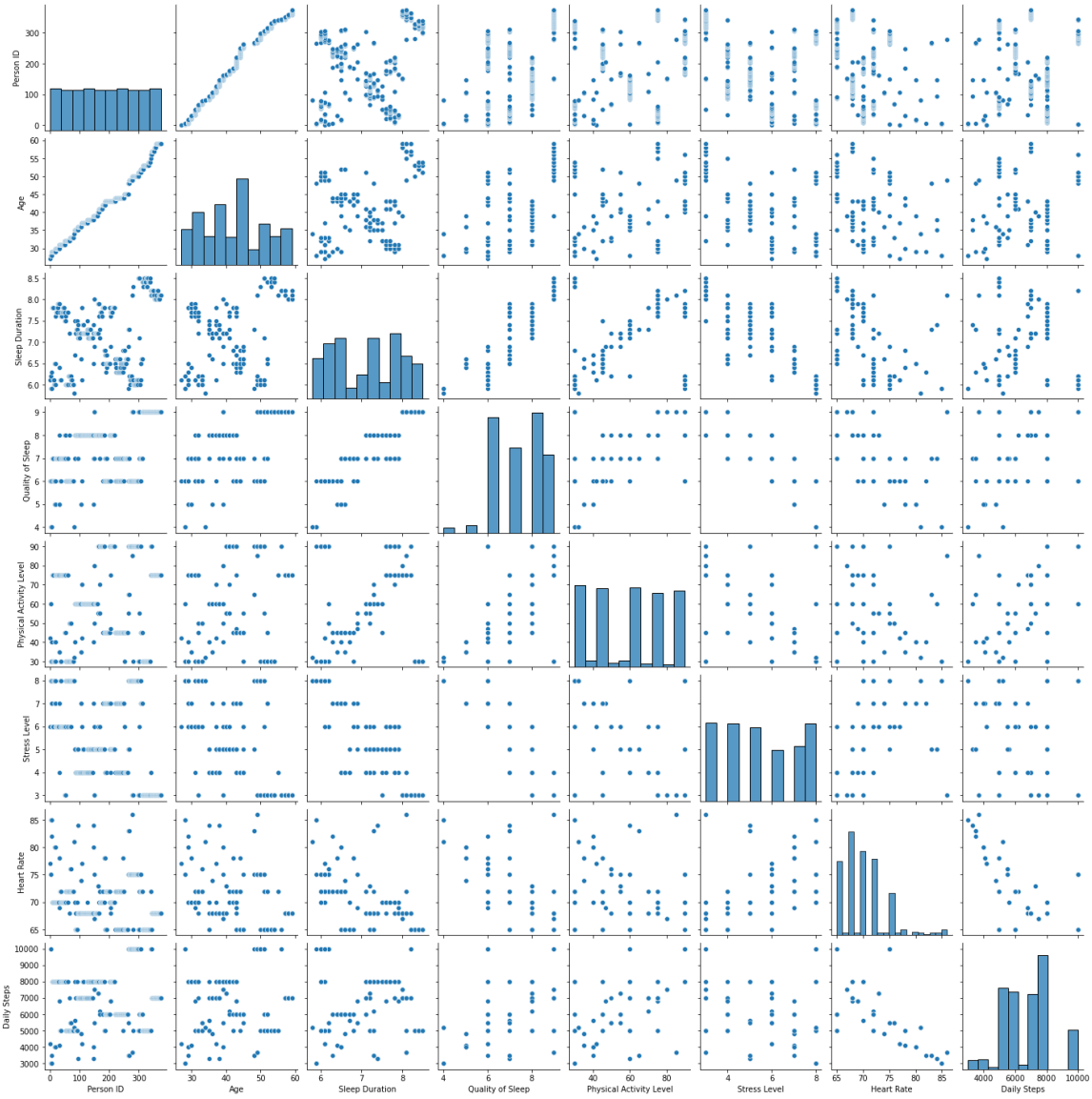
```
Index(['Person ID', 'Gender', 'Age', 'Occupation', 'Sleep Duration',
       'Quality of Sleep', 'Physical Activity Level', 'Stress Level',
       'BMI Category', 'Blood Pressure', 'Heart Rate', 'Daily Steps',
       'Sleep Disorder'],
      dtype='object')
```

In [8]:

```
sb.pairplot(df)
```

Out[8]:

```
<seaborn.axisgrid.PairGrid at 0x240fb8eaa30>
```



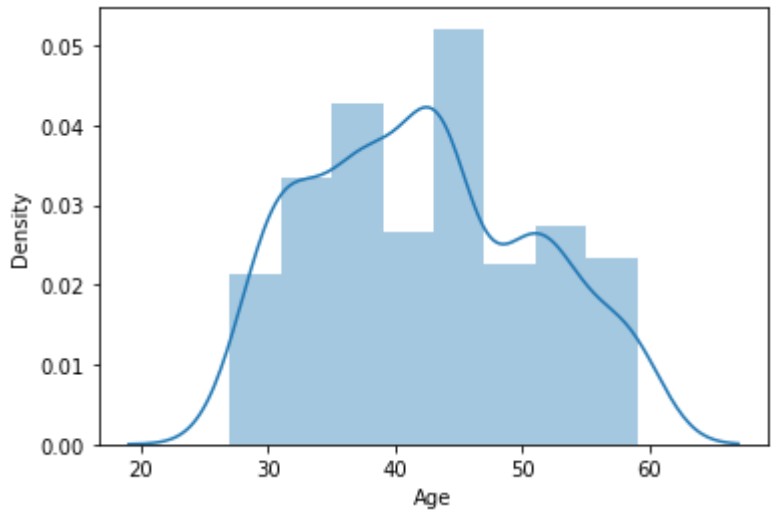
In [9]:

```
sb.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[9]:

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



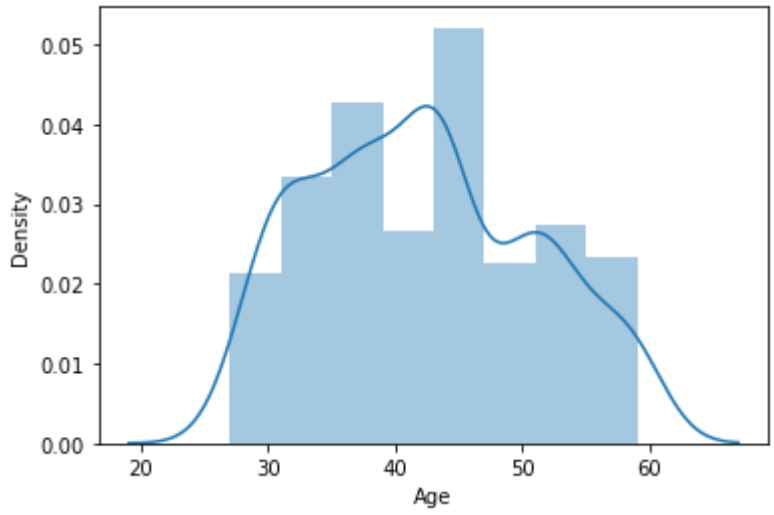
In [10]:

```
sb.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[10]:

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



In [11]:

```
df1=df[['Person ID', 'Age', 'Sleep Duration',
       'Quality of Sleep', 'Physical Activity Level', 'Stress Level', 'Heart Rate', 'Dail
df1
```

Out[11]:

	Person ID	Age	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	Heart Rate	Daily Steps
0	1	27	6.1	6	42	6	77	4200
1	2	28	6.2	6	60	8	75	10000
2	3	28	6.2	6	60	8	75	10000
3	4	28	5.9	4	30	8	85	3000
4	5	28	5.9	4	30	8	85	3000
...	...	...	...	...	...	...	...	...
369	370	59	8.1	9	75	3	68	7000
370	371	59	8.0	9	75	3	68	7000
371	372	59	8.1	9	75	3	68	7000
372	373	59	8.1	9	75	3	68	7000
373	374	59	8.1	9	75	3	68	7000

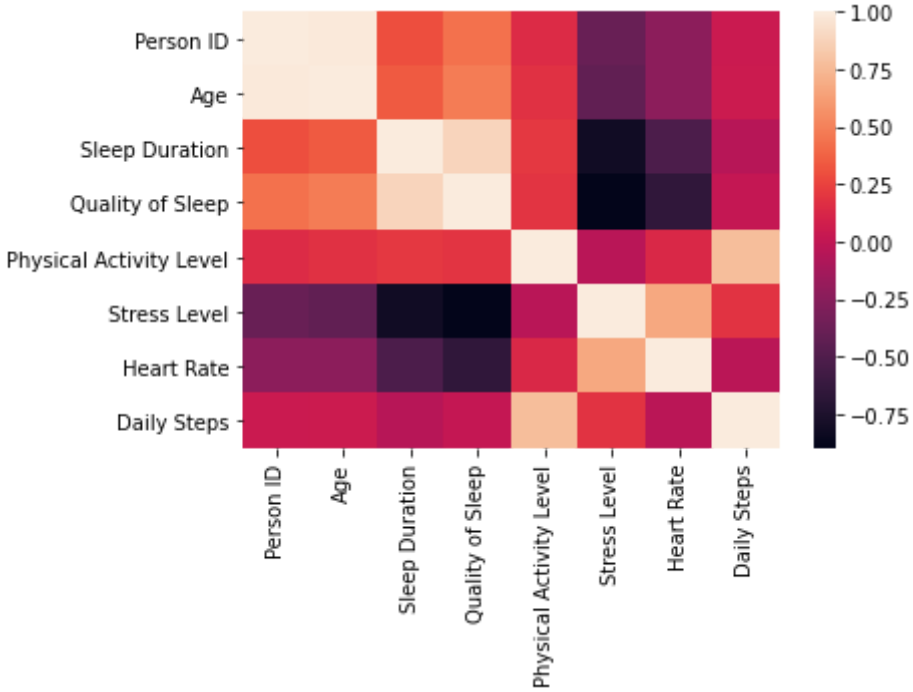
374 rows × 8 columns

In [12]:

```
sb.heatmap(df1.corr())
```

Out[12]:

```
<AxesSubplot:>
```



model building

In [13]:

```
x = df1[['Person ID', 'Age', 'Sleep Duration',
       'Quality of Sleep', 'Physical Activity Level', 'Stress Level', 'Heart Rate', 'Dail
y = df1['Age']
```

In [14]:

```
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)
```

linear regression

In [15]:

```
from sklearn.linear_model import LinearRegression

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

Out[15]:

```
LinearRegression()
```

In [16]:

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

```
3.836930773104541e-12
```

In [17]:

```
coef = pd.DataFrame(lr.coef_,x.columns,columns=['Co_efficient'])
coef
```

Out[17]:

	Co_efficient
Person ID	1.144347e-15
Age	1.000000e+00
Sleep Duration	-6.177779e-14
Quality of Sleep	-1.139433e-14
Physical Activity Level	7.401957e-15
Stress Level	-8.832246e-15
Heart Rate	-9.180148e-15
Daily Steps	-4.010221e-16

In [18]:

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

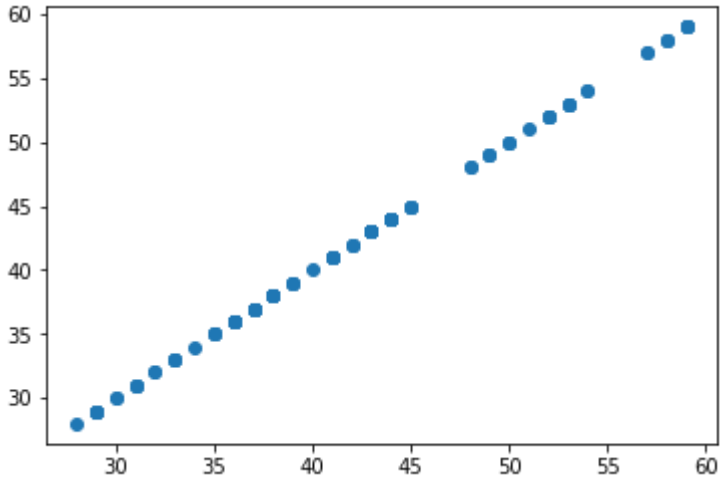
```
1.0
```

In [19]:

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

Out[19]:

```
<matplotlib.collections.PathCollection at 0x24082b0e6d0>
```



lasso and ridge regression

In [20]:

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

Out[20]:

```
1.0
```

In [21]:

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

Out[21]:

```
1.0
```

In [22]:

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

In [23]:

```
r = Ridge(alpha=10)
r.fit(x_train,y_train)
r.score(x_test,y_test)
r.score(x_train,y_train)
```

Out[23]:

```
0.9999827365842395
```

In [24]:

```
l = Lasso(alpha=10)
l.fit(x_train,y_train)
l.score(x_test,y_test)
l.score(x_train,y_train)
```

Out[24]:

```
0.9829846901505059
```

elasticnet

In [25]:

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

Out[25]:

```
ElasticNet()
```

In [26]:

```
print(e.coef_)
```

```
[ 4.32209284e-02  4.52807298e-01  0.00000000e+00  0.00000000e+00
  8.96576706e-03 -0.00000000e+00 -0.00000000e+00 -5.63666553e-05]
```

In [27]:

```
print(e.intercept_)
```

```
14.820650282393508
```

In [28]:

```
predictions = e.predict(x_test)
predictions
```

Out[28]:

```
array([44.7502445 , 55.86297343, 35.67362713, 38.68951921, 53.69517419,
       43.14653128, 29.57722893, 31.93410018, 55.8197525 , 36.4158352 ,
       35.63040621, 46.49967965, 46.54290058, 45.00957007, 32.99200047,
       40.42038117, 44.96634915, 33.71079154, 42.43893728, 37.03407393,
       27.77201138, 41.2189559 , 28.69221234, 39.39605436, 49.93579039,
       48.48088003, 38.16527035, 36.73152743, 45.58458787, 42.06075155,
       51.70071437, 51.83037716, 35.37108064, 39.05832498, 42.87114657,
       52.68049482, 50.86724073, 35.71684806, 30.93767572, 33.12166326,
       38.0788285 , 49.53393862, 50.23833689, 48.96433195, 34.29922017,
       39.30961251, 29.87977542, 51.78715623, 41.93108877, 32.86233769,
       45.18245379, 45.61466307, 30.85123386, 36.60186465, 48.39443817,
       52.63727389, 38.381375  , 45.3553375 , 28.4159558 , 28.56254956,
       30.57310642, 45.31211657, 57.89233216, 57.41690195, 43.42119835,
       36.5186386 , 35.54396435, 43.72374484, 41.26217683, 53.65195326,
       56.70476908, 49.55001486, 40.80178272, 44.66380265, 52.59405296,
       46.58612151, 43.68052392, 49.00755288, 37.60339829, 46.28357501,
       57.67622752, 57.50334381, 52.50761111, 40.9164094 , 42.56860007,
       41.08929311, 34.4296974 , 28.46163654, 39.13672879, 33.20810512,
       37.73306107, 56.83443187, 41.59714163, 42.611821  , 30.31432781,
       37.12051579, 42.74148378, 40.88177497, 38.3512998 , 36.90441114,
       28.90642362, 35.45752249, 57.97877402, 51.88192009, 35.41430156,
       28.37519468, 45.26889565, 34.77168108, 34.90134386, 56.74799001,
       44.02629134, 43.37797742, 32.06376296])
```

In [29]:

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

```
0.9932917187781416
```

In [30]:

```
from sklearn import metrics
```

mean absolute error

In [31]:

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

```
Mean Absolute Error: 0.5150409254524503
```

mean squared error

In [32]:

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

```
Mean Squared Error: 0.4609669858393404
```

root mean squared error

In [33]:

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

```
Root Mean Squared Error 0.6789454954849766
```

In [ ]:

```

```