Problem Statement

A real estate agent want help to predict the house price for regions in USA.He gave us the dataset to work on to use linear regression model.Create a model that helps him to estimate of what the house would sell for

Import libraries

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

In [2]:
```
# To import dataset
df=pd.read_csv('16 Sleep csv')
df
```

Out[2]:

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Blood Pressure	Heart Rate
0	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	126/83	77
1	2	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75
2	3	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75
3	4	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85
4	5	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85
...	...	...	...	...	...	...	...	...	...	...	...
369	370	Female	59	Nurse	8.1	9	75	3	Overweight	140/95	68
370	371	Female	59	Nurse	8.0	9	75	3	Overweight	140/95	68
371	372	Female	59	Nurse	8.1	9	75	3	Overweight	140/95	68
372	373	Female	59	Nurse	8.1	9	75	3	Overweight	140/95	68
373	374	Female	59	Nurse	8.1	9	75	3	Overweight	140/95	68

374 rows × 13 columns

In [3]:
```
# To display top 10 rows
df.head(10)
```

Out[3]:

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Blood Pressure	Heart Rate	D St
0	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	126/83	77	4
1	2	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75	10
2	3	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75	10
3	4	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85	3
4	5	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85	3
5	6	Male	28	Software Engineer	5.9	4	30	8	Obese	140/90	85	3
6	7	Male	29	Teacher	6.3	6	40	7	Obese	140/90	82	3
7	8	Male	29	Doctor	7.8	7	75	6	Normal	120/80	70	8
8	9	Male	29	Doctor	7.8	7	75	6	Normal	120/80	70	8
9	10	Male	29	Doctor	7.8	7	75	6	Normal	120/80	70	8

Data Cleaning and Pre-Processing

In [4]:
```
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 [5]:
```
df.describe()
```

Out[5]:

	Person ID	Age	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	Heart Rate	Daily Steps
count	374.000000	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	6816.844920
std	108.108742	8.673133	0.795657	1.196956	20.830804	1.774526	4.135676	1617.915679
min	1.000000	27.000000	5.800000	4.000000	30.000000	3.000000	65.000000	3000.000000
25%	94.250000	35.250000	6.400000	6.000000	45.000000	4.000000	68.000000	5600.000000
50%	187.500000	43.000000	7.200000	7.000000	60.000000	5.000000	70.000000	7000.000000
75%	280.750000	50.000000	7.800000	8.000000	75.000000	7.000000	72.000000	8000.000000
max	374.000000	59.000000	8.500000	9.000000	90.000000	8.000000	86.000000	10000.000000

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')
```

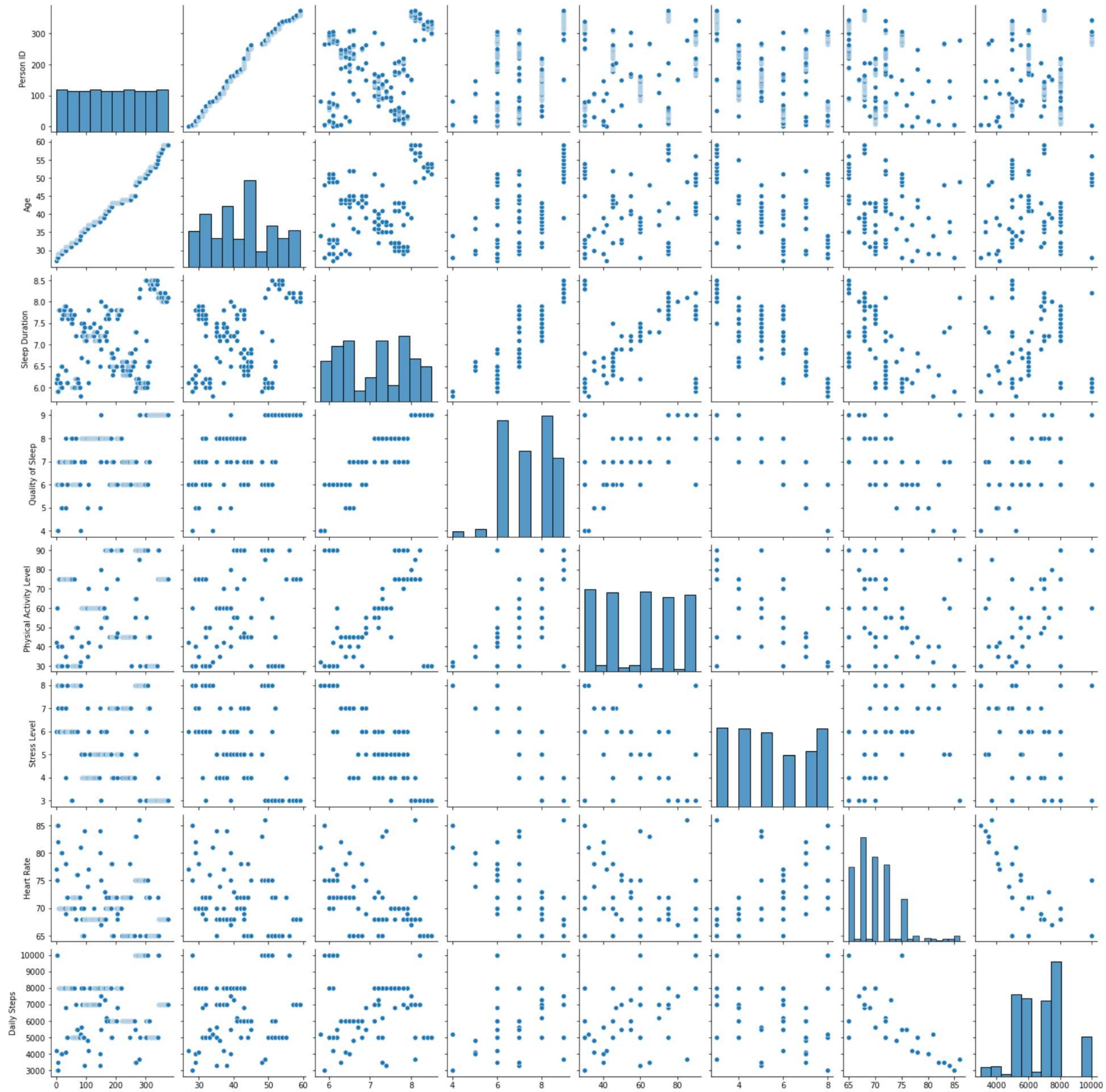
In [7]:
```
a = df.dropna(axis='columns')
a.columns
```

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')
```

EDA and Visualization

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

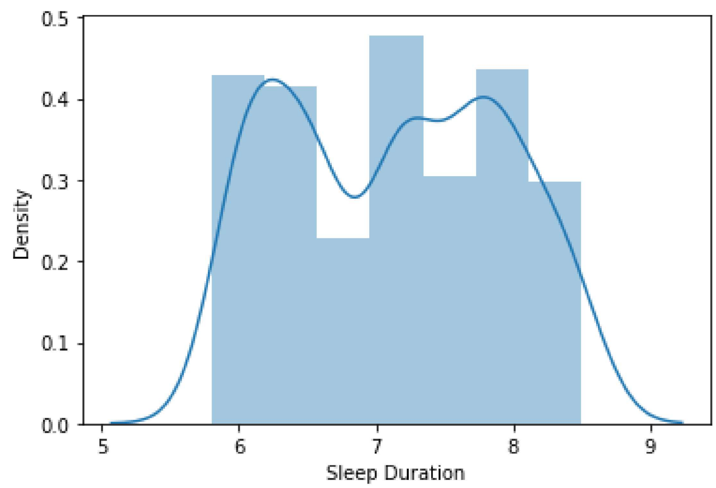
Out[8]: <seaborn.axisgrid.PairGrid at 0x27864d961f0>



In [9]:
```
sns.distplot(a['Sleep Duration'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarnin
g: `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 flexibil
ity) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

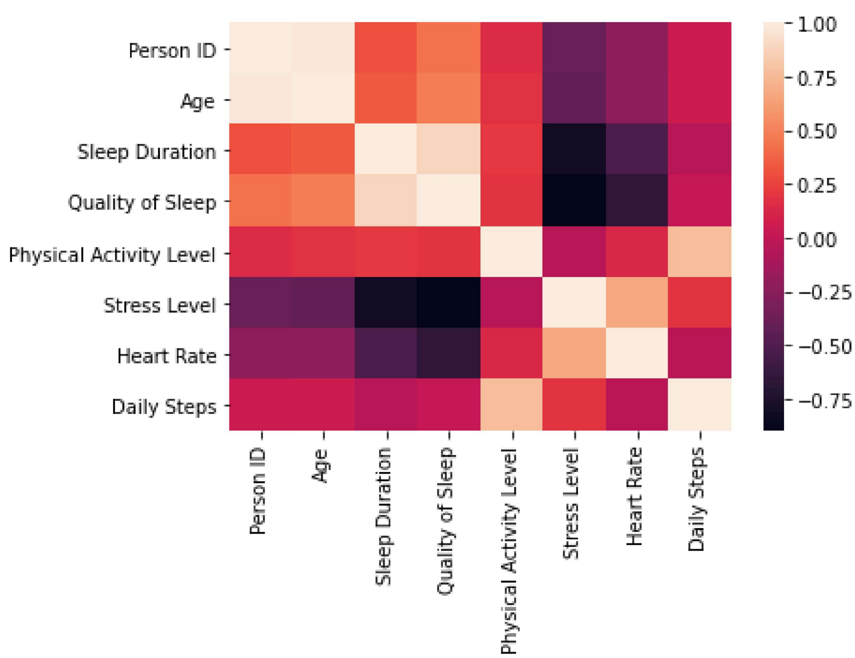
Out[9]:
```
<AxesSubplot:xlabel='Sleep Duration', ylabel='Density'>
```



In [10]:
```
a1=a[['Person ID', 'Age', 'Sleep Duration','Quality of Sleep', 'Physical Activity Level
```

In [11]:
```
sns.heatmap(a1.corr())
```

Out[11]:
```
<AxesSubplot:>
```



To Train the Model - Model Building

We are going to train Linear Regression model;We need to split out data into two variables x and y where x

In [12]:
```
x=a1[['Person ID', 'Age', 'Sleep Duration','Quality of Sleep', 'Physical Activity Level
y=a1['Sleep Duration']
```

To split my dataset into training and test data

In [13]:
```
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]:
```
print(lr.intercept_)
```

```
3.099742684753437e-13
```

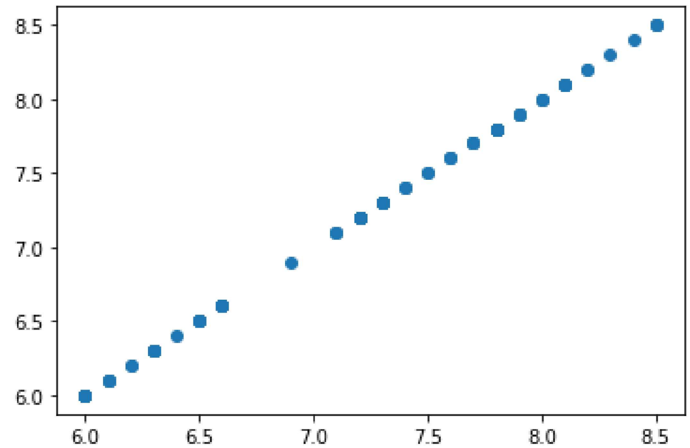
In [16]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[16]:

	Co-efficient
Person ID	9.738430e-17
Age	1.930084e-16
Sleep Duration	1.000000e+00
Quality of Sleep	-4.931132e-16
Physical Activity Level	1.385472e-17
Stress Level	-1.777827e-17
Heart Rate	3.729460e-17
Daily Steps	-4.912326e-17

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

Out[17]: <matplotlib.collections.PathCollection at 0x2786a393550>



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

1.0

ACCURACY

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

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

Out[20]: 0.9901000175495261

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

Out[21]: 0.9864939832344124

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]: 0.01650048922838787

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

Out[24]: ElasticNet()

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

```
[ 0.00096002  0.          0.          0.          0.02505612 -0.
 -0.08448836 -0.000312  ]
```

In [26]:
```
print(en.intercept_)
```

```
13.52071771606479
```

In [27]:
```
print(en.predict(x_test))
```

```
[6.93262849 7.19278867 6.70411413 7.02909079 7.82783679 6.59177357
 6.70123406 6.9047878  6.99163992 6.75802508 6.5341285  6.92974842
 7.53453571 6.91195795 6.93166846 6.61002732 7.03389091 6.28700756
 6.7036021  6.59849374 6.68299361 6.86395677 7.53841861 7.82303667
 6.65797156 6.91342801 6.6954901  7.04445117 7.20142888 6.89230749
 6.68587368 6.6090673  8.08412555 6.92251821 7.19374869 6.69835399
 5.87660498 6.93358851 6.70315411 7.56045635 7.20526898 6.99837003
 7.53457851 7.51297668 7.81535648 7.5422159  7.52881837 7.01469043
 7.00989031 7.17742829 7.01853053 6.91246799 7.52737703 6.70219409
 7.23452088 6.9301984  6.94289352 7.72991313 7.81727653 6.69739397
 8.06065641 7.53933583 7.22350943 6.91579805 7.0175705  6.68203359
 7.1860685  7.49665628 6.92878839 7.54513877 7.21198914 7.21390919
 7.53169844 6.92539828 6.90958792 7.5340972  6.88411727 7.53217715
 7.20334893 6.59081355 7.52493547 6.66187309 7.80095612 6.91054794
 6.91726811 7.03581095 6.68011354 7.51009661 7.53121713 6.8884674
 6.50963224 7.03197086 7.50049637 7.21486921 6.5888935  7.56909656
 6.70264208 6.68875375 7.5259383  7.82207664 7.45006999 7.4803007
 5.87564496 7.037731   7.03762141 6.8773971  6.58697345 7.82495671
 7.82687676 6.90670785 7.49329852 7.57389668 7.80863631]
```

In [28]:
```
print(en.score(x_test,y_test))
```

```
0.4428043811137372
```

In [29]:
```
from sklearn import metrics
print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
Mean Absolytre Error: 6.173233016216623e-14
Mean Squared Error: 5.134549029614454e-27
Root Mean Squared Error: 7.165576759490093e-14
```