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

Data Collection

In [1]:

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

In [88]:

```
#import the dataset
data=pd.read_csv(r"C:\Users\user\Desktop\Vicky\16_Sleep_health_and_lifestyle_dataset.csv"
```

In [89]:

```
#to display top 10 rows
data.head()
```

Out[89]:

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Blood Pressure	Heart Rate	Dail Step
0	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	126/83	77	420
1	2	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75	1000
2	3	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75	1000
3	4	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85	300
4	5	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85	300

In [90]:

```
#to display null values
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 [91]:

```
data.shape
```

Out[91]:

```
(374, 13)
```

In [92]:

```
#to display summary of statistics
data.describe()
```

Out[92]:

	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 [93]:

```
#to display columns name
data.columns
```

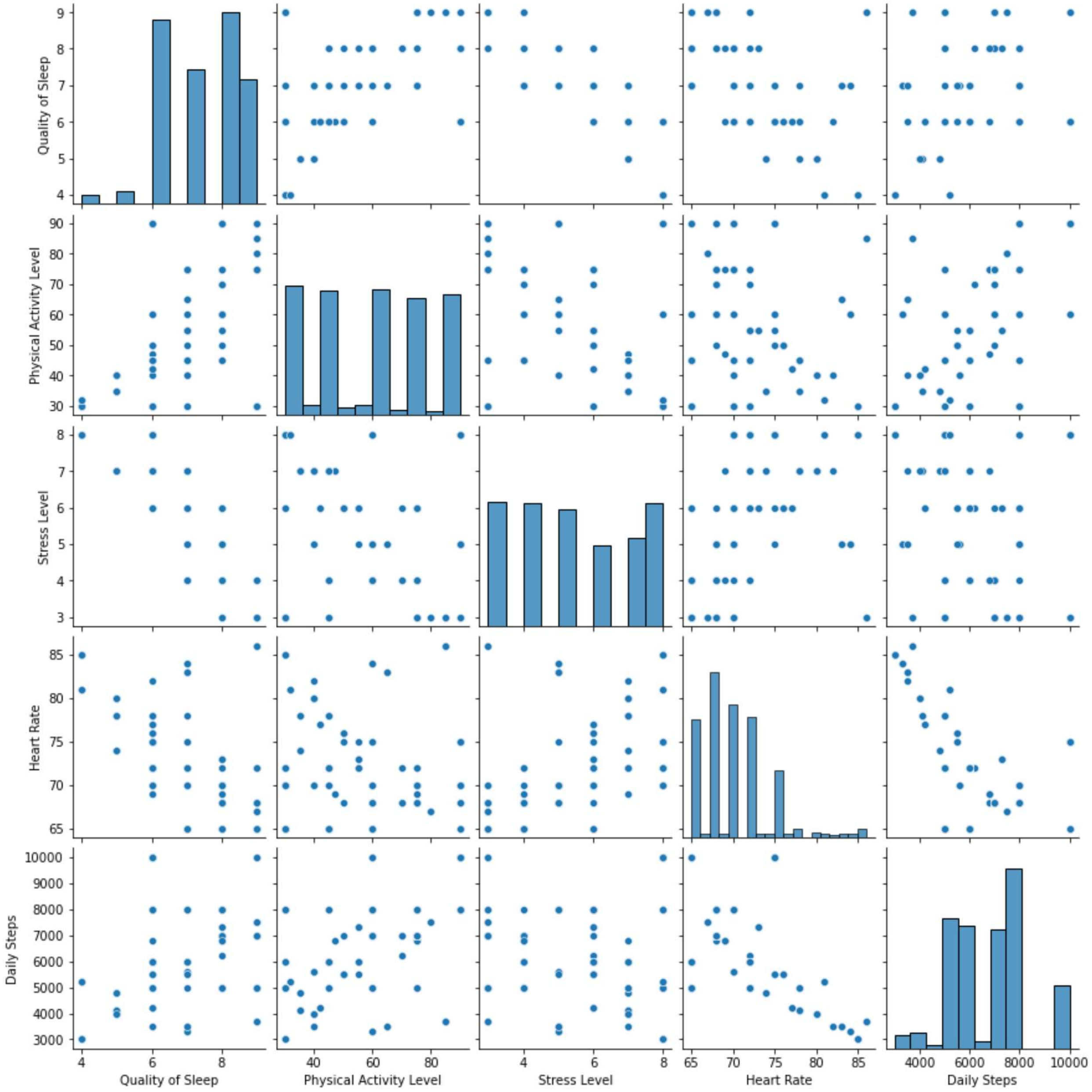
Out[93]:

```
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 [94]:
```
data1=data[['Quality of Sleep', 'Physical Activity Level', 'Stress Level',
           'BMI Category', 'Blood Pressure', 'Heart Rate', 'Daily Steps']]
```

In [95]:
```
sns.pairplot(data1)
```

Out[95]:
```
<seaborn.axisgrid.PairGrid at 0x2388eb4d6d0>
```

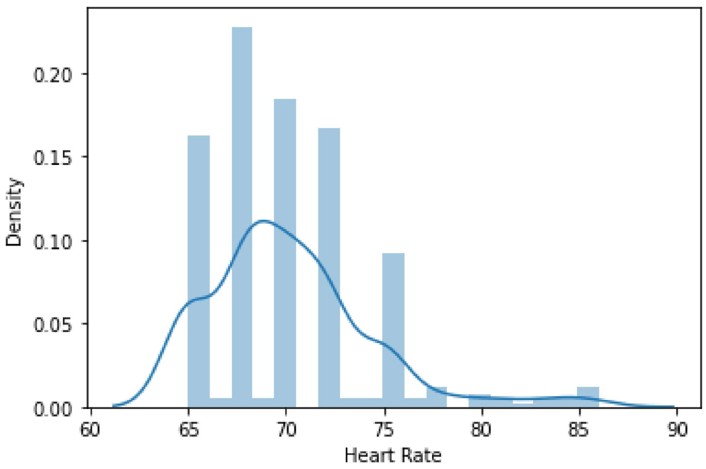


EDA and Visualization

In [96]:
```
sns.distplot(data['Heart Rate'])
```

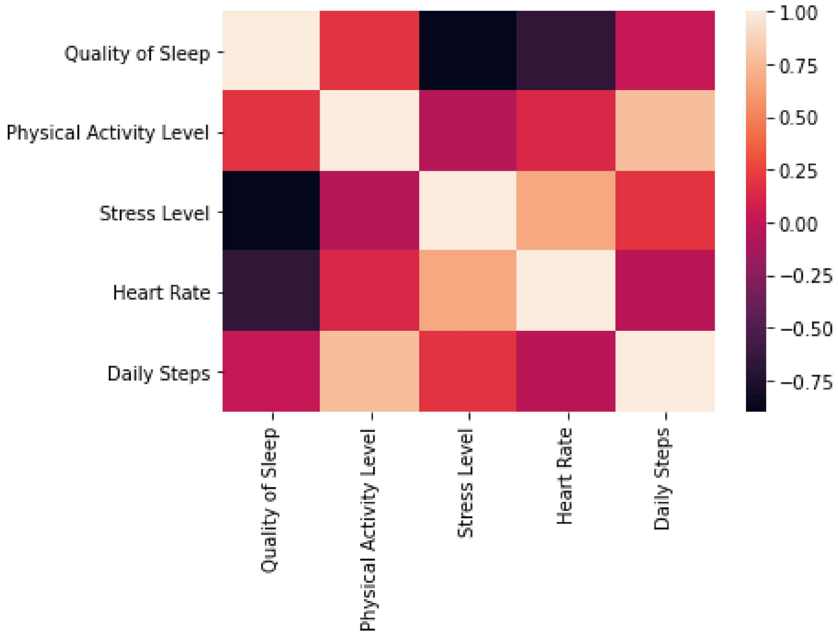
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function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

Out[96]:
```
<AxesSubplot:xlabel='Heart Rate', ylabel='Density'>
```



In [97]:
```
sns.heatmap(data1.corr())
```

Out[97]:
```
<AxesSubplot:>
```



To train the model

we are going to train the linear regression model ;We need to split the two variable x and y where x in independent variable (input) and y is dependent of x(output) so we could ignore address columns as it is not requires for our model

In [107]:
```
x=data1[[ 'Stress Level',
       'Quality of Sleep']]
y=data1["Heart Rate"]
```

In [108]:
```

#To split test and train data
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.6)
```

In [109]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[109]:
```
LinearRegression()
```

In [110]:
```
lr.intercept_
```

Out[110]:
```
74.77545651247416
```

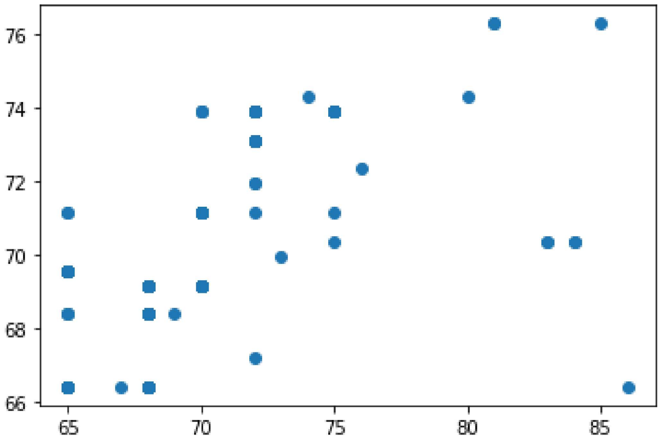
In [111]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=["Co-efficient"])
coeff
```

Out[111]:

	Co-efficient
Stress Level	0.786365
Quality of Sleep	-1.193034

In [112]:
```
prediction = lr.predict(x_train)
plt.scatter(y_train,prediction)
```

Out[112]:
```
<matplotlib.collections.PathCollection at 0x2388fdea670>
```



In [113]:
```
lr.score(x_test,y_test)
```

Out[113]:
```
0.538868100004067
```

In [114]:
```
lr.score(x_train,y_train)
```

Out[114]:
```
0.3797847342476639
```

In [115]:
```
from sklearn.linear_model import Ridge,Lasso
```

```
In [116]: rr=Ridge(alpha=10)
          rr.fit(x_train,y_train)
          rr.score(x_test,y_test)

Out[116]: 0.5389932880791577
```

```
In [117]: la=Lasso(alpha=10)
          la.fit(x_train,y_train)
          la.score(x_test,y_test)

Out[117]: -0.00341410751305693
```