Type *Markdown* and LaTeX: α^2

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

In [2]:
```
#import dataset
df=pd.read_csv(r"E:\154\16_Sleep_health_and_lifestyle_dataset.csv",low_memory=
df
```

Out[2]:

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

374 rows × 13 columns

In [3]:
```
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 [4]:
```
#to display top 5 rows
df.head()
```

Out[4]:

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Bloo Pressur
0	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	126/8
1	2	Male	28	Doctor	6.2	6	60	8	Normal	125/8
2	3	Male	28	Doctor	6.2	6	60	8	Normal	125/8
3	4	Male	28	Sales Representative	5.9	4	30	8	Obese	140/9
4	5	Male	28	Sales Representative	5.9	4	30	8	Obese	140/9

Data cleaning and Pre-Processing

In [5]:

```
#To find null values
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]:

```
# To display summary of statistics
df.describe()
```

Out[6]:

	Person ID	Age	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	Heart Rate	Da
count	374.000000	374.000000	374.000000	374.000000	374.000000	374.000000	374.000000	37
mean	187.500000	42.184492	7.132086	7.312834	59.171123	5.385027	70.165775	681
std	108.108742	8.673133	0.795657	1.196956	20.830804	1.774526	4.135676	161
min	1.000000	27.000000	5.800000	4.000000	30.000000	3.000000	65.000000	300
25%	94.250000	35.250000	6.400000	6.000000	45.000000	4.000000	68.000000	560
50%	187.500000	43.000000	7.200000	7.000000	60.000000	5.000000	70.000000	700
75%	280.750000	50.000000	7.800000	8.000000	75.000000	7.000000	72.000000	800
max	374.000000	59.000000	8.500000	9.000000	90.000000	8.000000	86.000000	1000

In [7]:

```
#To Display column heading
df.columns
```

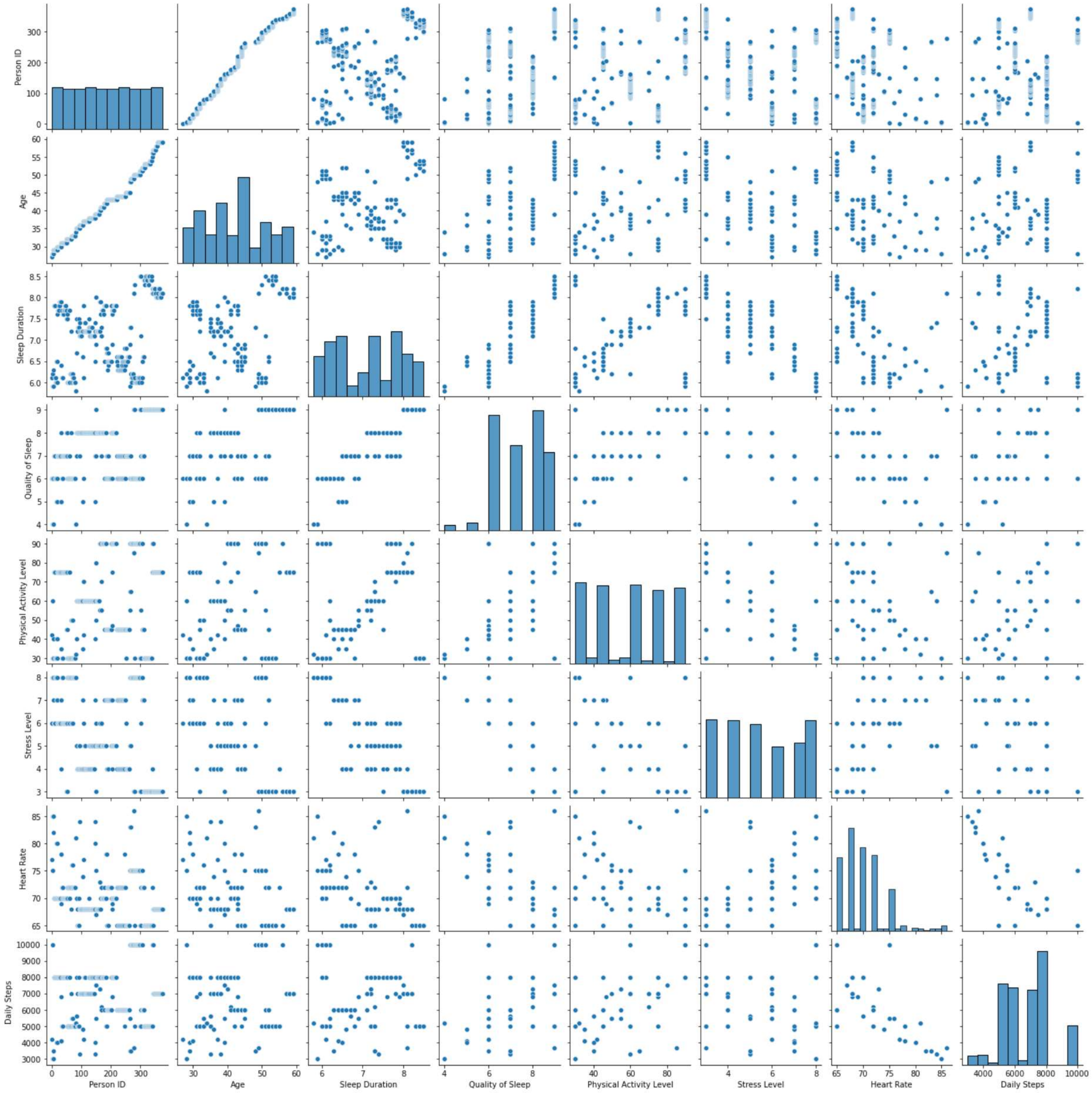
Out[7]:

```
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')
```

EDA and VISUALIZATION

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

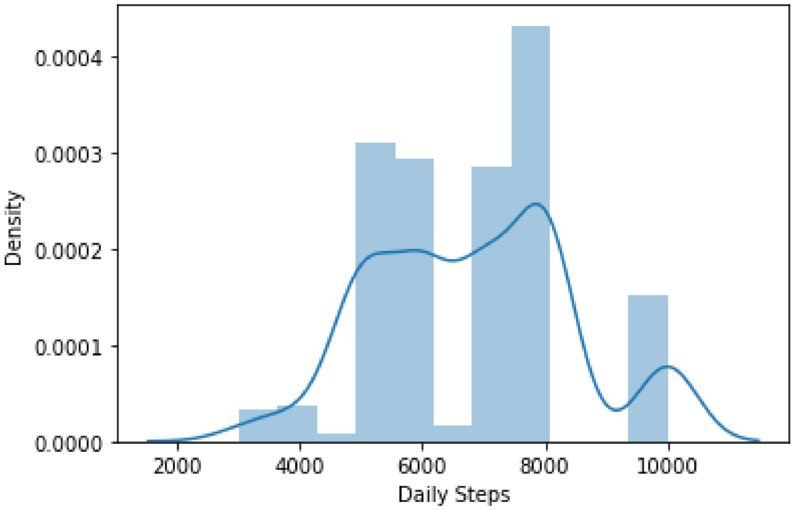
```
Out[8]: <seaborn.axisgrid.PairGrid at 0x1af76b685e0>
```



In [9]:
```
sns.distplot(df["Daily Steps"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)
```

Out[9]:
```
<AxesSubplot:xlabel='Daily Steps', ylabel='Density'>
```

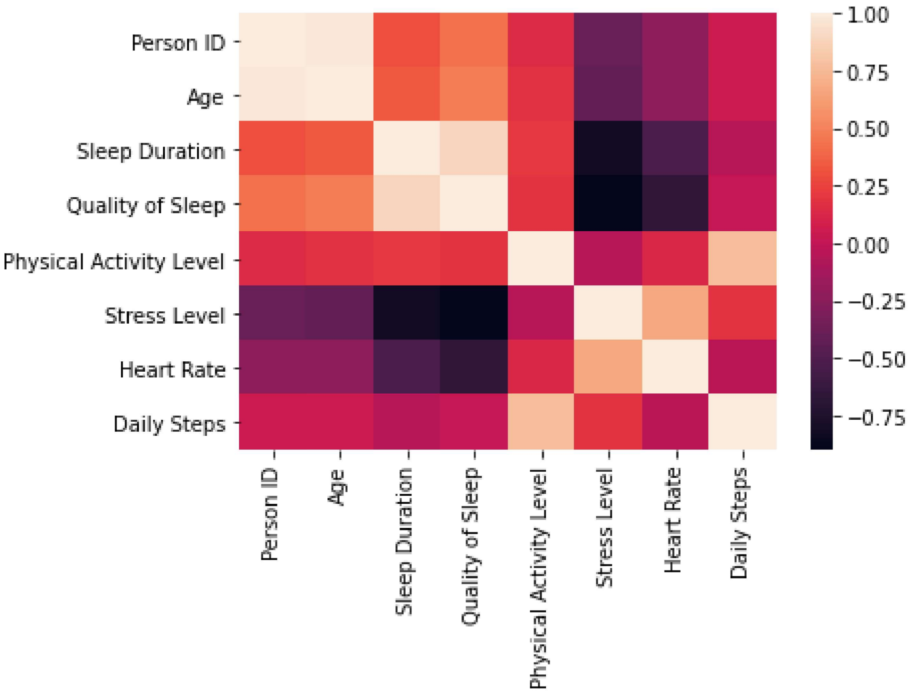


In [10]:
```
df1=df[['Person ID' , 'Age',  'Sleep Duration',
       'Quality of Sleep', 'Physical Activity Level', 'Stress Level', 'Heart R
```

Plot Using Heat Map

In [11]:
```
sns.heatmap(df1.corr())
```

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



To Train The Model-Model Building

we are going to train Linera Regression Model;We need to split out data into two variables x and y where x is independent variable(input) and y is dependent on x(output) we could ignore address column as it required for our model

In [12]:
```
x=df1[['Person ID' , 'Age',  'Sleep Duration',
       'Quality of Sleep', 'Physical Activity Level', 'Stress Level', 'Heart Ra
       ]]
y=df1["Daily Steps"]
```

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]:
```
lr.intercept_
```

Out[15]:
```
12585.044390548912
```

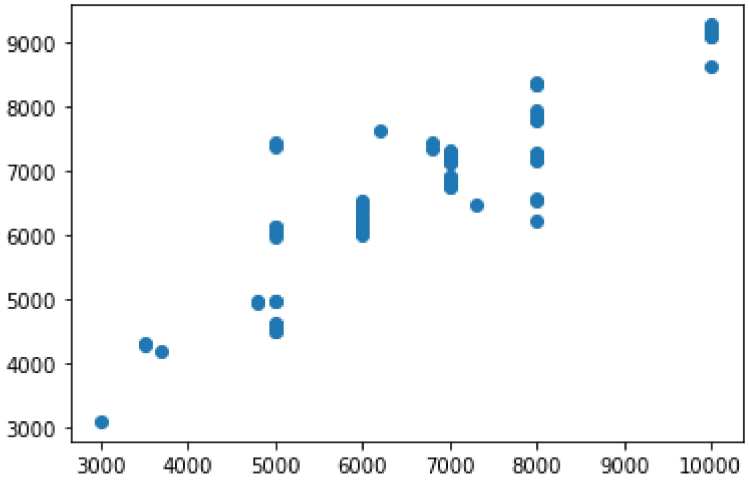
In [16]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[16]:

	Co-efficient
Person ID	-7.518145
Age	88.527426
Sleep Duration	-476.616768
Quality of Sleep	305.281429
Physical Activity Level	63.689519
Stress Level	501.652842
Heart Rate	-191.354218

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

Out[17]:
```
<matplotlib.collections.PathCollection at 0x1af7c0ef070>
```



Accuracy

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

Out[18]:
```
0.8400877081596163
```

```
In [19]: lr.score(x_train,y_train)

Out[19]: 0.7801891516263414
```

```
In [20]: from sklearn.linear_model import Ridge,Lasso
```

```
In [21]: rr=Ridge(alpha=10)
         rr.fit(x_train,y_train)

Out[21]: Ridge(alpha=10)
```

```
In [22]: rr.score(x_test,y_test)

Out[22]: 0.8369905501894674
```

```
In [23]: la =Lasso(alpha=10)
         la.fit(x_train,y_train)

Out[23]: Lasso(alpha=10)
```

```
In [24]: la.score(x_test,y_test)

Out[24]: 0.8379977837119328
```

ElasticNet

```
In [25]: from sklearn.linear_model import ElasticNet
         en = ElasticNet()
         en.fit(x_train,y_train)

Out[25]: ElasticNet()
```

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

         [  -4.37547173   47.58864674 -137.81484244  -58.3619333    64.01234878
           305.68102917 -169.85334064]
```

```
In [27]: print(en.intercept_)

         13480.17000191959
```

In [28]:

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

```
[5743.43710013 6098.00835034 8631.07111455 4720.63712869 7052.77593012
 6770.16887101 6095.77943898 8430.56042741 6222.55459258 4663.75599614
 7039.64951492 5695.42043775 4702.59369941 6140.18922537 6792.04622968
 4699.18778566 6069.52660858 6831.96701763 6205.05270564 5347.97680143
 6735.36420973 8907.4667874  7218.4340468  5668.62606501 5891.73073474
 6260.02100223 8015.93553557 8051.59437848 6783.29528621 5153.88069485
 6082.65302378 6316.78860807 7588.8449488  8047.21890675 8885.58942872
 7074.53976209 8876.06988951 4711.23111618 8902.43624662 7559.16453287
 6748.29151234 8854.96112658 8902.97778896 6788.21230028 8443.68684261
 6787.67075794 6089.25740687 7027.8332378  6161.411515   6726.52768037
 6801.33871549 8920.5932026  7004.75926775 4703.13524175 6439.18828993
 6991.63285255 6276.37039147 4915.04133577 8867.54599945 8863.71207005
 4689.46728421 7079.02876053 6774.54434274 8091.51516643 5676.72193944
 7302.03426725 6739.65409557 8452.43778608 8042.84343501 5659.1065258
 6823.7576165  7191.41262065 7220.95783811 7000.38379601 8871.69441778
 6757.15598251 8854.30605754 7380.10911419 6219.98668758 4698.21822768
 8007.1845921  4709.15583692 5659.87512154 7324.56669496 6430.15571236
 8832.42869887 8867.31894604 8836.80417061 4676.22734231 8029.17547747
 6828.24661493 7182.66167719 6171.19860958 8033.3238958  6325.53955154
 6273.14741743 8858.68152928 5721.67326816 7231.56046201 4694.92584062
 6098.06217394 6075.52974622 7313.1669858  7257.27175007 4383.49457069
 8867.43247275 5339.22585796 6071.15427449 4781.72369614 6043.27377817
 6362.86490658 4698.10470098 3668.12871867]
```

In [29]:

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

```
0.8088724753839608
```

Evaluation Metrics

In [30]:

```
from sklearn import metrics
```

In [31]:

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

```
Mean Absolute Error 529.2192458217474
```

In [32]:

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

```
Mean Squared Error: 486650.83730979863
```

In [33]:

```
print("Root Mean Absolute Error:",np.sqrt(metrics.mean_squared_error(y_test,pr
```

```
Root Mean Absolute Error: 697.6036391173706
```