In [1]:

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

14.IRIS

In [2]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\14_Iris.csv")
a
```

Out[2]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

In [3]:

```
b=a.head(100)
b
```

Out[3]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
95	96	5.7	3.0	4.2	1.2	Iris-versicolor
96	97	5.7	2.9	4.2	1.3	Iris-versicolor
97	98	6.2	2.9	4.3	1.3	Iris-versicolor
98	99	5.1	2.5	3.0	1.1	Iris-versicolor
99	100	5.7	2.8	4.1	1.3	Iris-versicolor

100 rows × 6 columns

In [4]:

```
b.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             100 non-null    int64  
 1   SepalLengthCm  100 non-null    float64
 2   SepalWidthCm   100 non-null    float64
 3   PetalLengthCm  100 non-null    float64
 4   PetalWidthCm   100 non-null    float64
 5   Species        100 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 4.8+ KB
```

In [5]:

```
b.describe()
```

Out[5]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	100.000000	100.000000	100.000000	100.000000	100.000000
mean	50.500000	5.471000	3.094000	2.862000	0.785000
std	29.011492	0.641698	0.476057	1.448565	0.566288
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	25.750000	5.000000	2.800000	1.500000	0.200000
50%	50.500000	5.400000	3.050000	2.450000	0.800000
75%	75.250000	5.900000	3.400000	4.325000	1.300000
max	100.000000	7.000000	4.400000	5.100000	1.800000

In [6]:

```
b.columns
```

Out[6]:

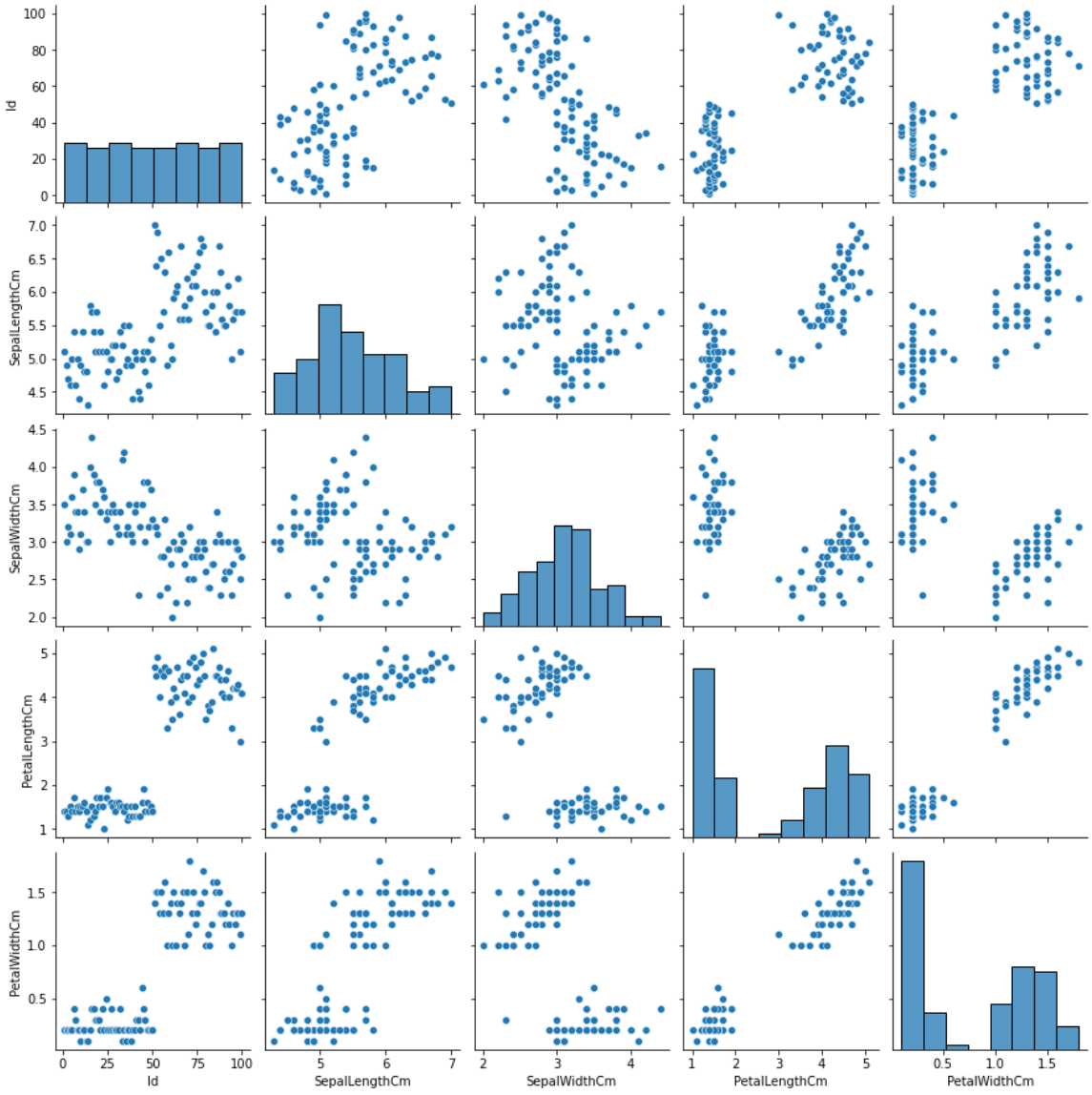
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidth
Cm',
       'Species'],
      dtype='object')
```

In [7]:

```
sns.pairplot(b)
```

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x171d5960af0>
```



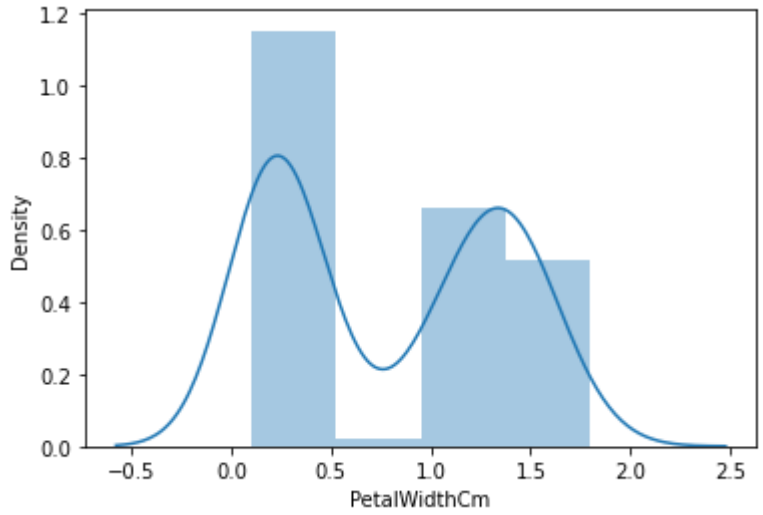
In [8]:

```
sns.distplot(b['PetalWidthCm'])
```

```
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='PetalWidthCm', ylabel='Density'>
```



In [9]:

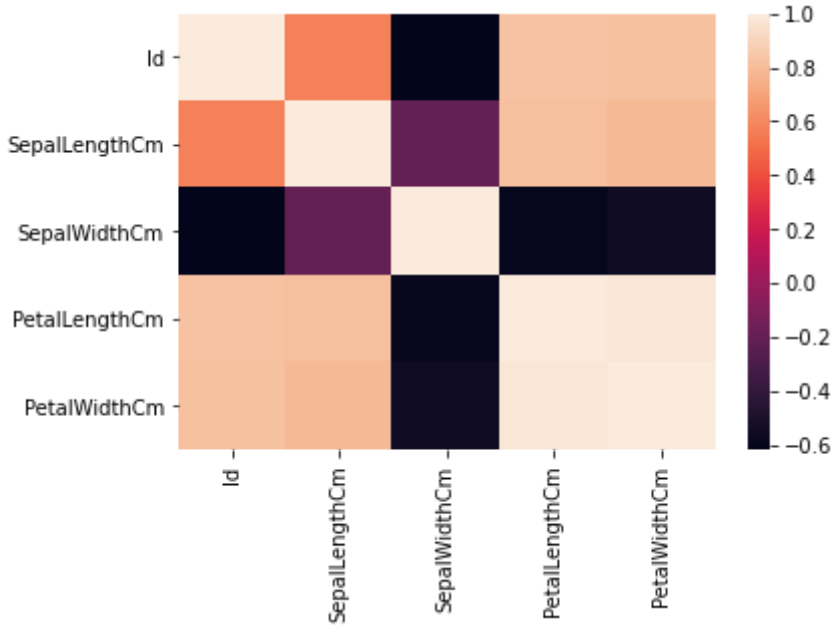
```
f=b[['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species']]
```

In [10]:

```
sns.heatmap(f.corr())
```

Out[10]:

```
<AxesSubplot:>
```



In [11]:

```
x1=f[['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm']]
y1=f['PetalWidthCm']
```

In [12]:

```
from sklearn.model_selection import train_test_split
x1_train,x1_test,y1_train,y1_test=train_test_split(x1,y1,test_size=0.5)
```

In [13]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x1_train,y1_train)
```

Out[13]:

```
LinearRegression()
```

In [14]:

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

```
-0.4226823928478569
```

In [15]:

```
r=pd.DataFrame(lr.coef_,x1.columns,columns=['Co-efficient'])
r
```

Out[15]:

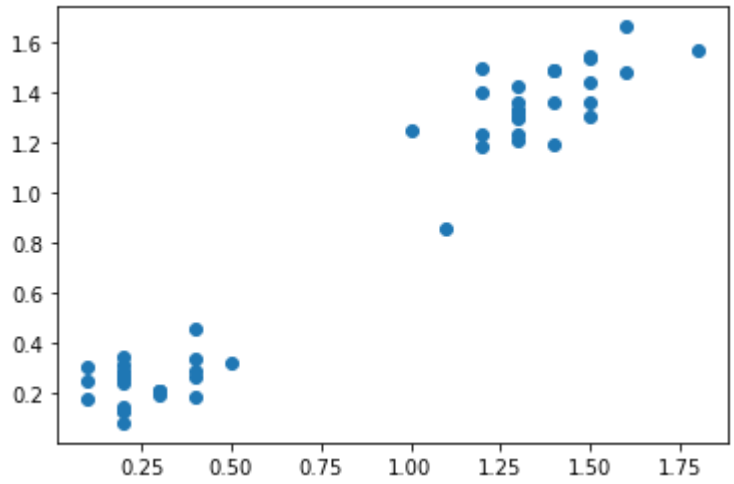
	Co-efficient
Id	0.001062
SepalLengthCm	-0.044434
SepalWidthCm	0.078301
PetalLengthCm	0.402064

In [16]:

```
u=lr.predict(x1_test)
plt.scatter(y1_test,u)
```

Out[16]:

```
<matplotlib.collections.PathCollection at 0x171d79adc10>
```



In [17]:

```
print(lr.score(x1_test,y1_test))
```

```
0.9515948448055147
```

In [18]:

```
lr.score(x1_train,y1_train)
```

Out[18]:

```
0.9693146372994317
```

RIDGE REGRESSION

In [19]:

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

In [20]:

```
rr=Ridge(alpha=10)
rr.fit(x1_train,y1_train)
```

Out[20]:

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

In [21]:

```
rr.score(x1_test,y1_test)
```

Out[21]:

```
0.9356412208675733
```

LASSO REGRESSION

In [22]:

```
la=Lasso(alpha=10)
la.fit(x1_train,y1_train)
```

Out[22]:

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

In [23]:

```
la.score(x1_test,y1_test)
```

Out[23]:

```
0.23929107734430222
```

ELASTIC NET

In [25]:

```
from sklearn.linear_model import ElasticNet
p=ElasticNet()
p.fit(x1_train,y1_train)
```

Out[25]:

```
ElasticNet()
```

In [26]:

```
print(p.coef_)
```

```
[ 0.01540332  0.         -0.          0.        ]
```

In [27]:

```
print(p.intercept_)
```

```
0.002953725623672354
```

In [28]:

```
print(p.predict(x1_test))
```

```
[1.46626882 1.51247877 0.81932951 0.37263333 1.09658921 1.32763896
 0.61908639 1.5432854  1.42005886 1.05037926 1.29683233 1.52788208
 0.11077694 1.14279916 1.28142901 0.92715273 1.37384891 0.26481011
 1.15820248 1.38925223 1.03497595 0.35723001 0.95795936 0.32642338
 1.01957263 1.40465555 0.18779353 0.20319684 0.52666649 0.91174941
 0.09537363 0.74231293 0.58827976 0.72690961 0.55747313 0.34182669
 0.7115063  0.69610298 0.24940679 0.40343996 0.23400348 1.06578258
 0.988766   1.11199253 0.28021343 0.51126318 1.43546218 1.12739585
 0.04916368 0.63448971]
```

In [33]:

```
prediction=p.predict(x1_test)
print(p.score(x1_test,y1_test))
```

```
0.6786581507031236
```

In [34]:

```
from sklearn import metrics
```

In [35]:

```
print("Mean Absolytre Error:",metrics.mean_absolute_error(y1_test,prediction))
```

```
Mean Absolytre Error: 0.27009296813033523
```

In [36]:

```
print("Mean Squared Error:",metrics.mean_squared_error(y1_test,prediction))
```

```
Mean Squared Error: 0.10417131533766419
```

In [37]:

```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y1_test,prediction)))
```

```
Root Mean Squared Error: 0.32275581379374746
```

15.Horse Racing Results

In [38]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\15_Horse Racing Results.csv - 15_Horse Racing Res
a
```

Out[38]:

	Dato	Track	Race Number	Distance	Surface	Prize money	Starting position	Jockey	Jockey weight	C
0	03.09.2017	Sha Tin	10	1400	Gress	1310000	6	K C Leung	52	
1	16.09.2017	Sha Tin	10	1400	Gress	1310000	14	C Y Ho	52	
2	14.10.2017	Sha Tin	10	1400	Gress	1310000	8	C Y Ho	52	
3	11.11.2017	Sha Tin	9	1600	Gress	1310000	13	Brett Prebble	54	
4	26.11.2017	Sha Tin	9	1600	Gress	1310000	9	C Y Ho	52	
...	...	...	...	...	...	...	...	...	...	
27003	14.06.2020	Sha Tin	11	1200	Gress	1450000	6	A Hamelin	59	A
27004	21.06.2020	Sha Tin	2	1200	Gress	967000	7	K C Leung	57	A
27005	21.06.2020	Sha Tin	4	1200	Gress	967000	6	Blake Shinn	57	A
27006	21.06.2020	Sha Tin	5	1200	Gress	967000	14	Joao Moreira	57	Z
27007	21.06.2020	Sha Tin	11	1200	Gress	1450000	7	C Schofield	55	Z

27008 rows × 21 columns

In [39]:

```
b=a.head(100)
b
```

Out[39]:

	Dato	Track	Race Number	Distance	Surface	Prize money	Starting position	Jockey	Jockey weight	Cou
0	03.09.2017	Sha Tin	10	1400	Gress	1310000	6	K C Leung	52	Sv
1	16.09.2017	Sha Tin	10	1400	Gress	1310000	14	C Y Ho	52	Sv
2	14.10.2017	Sha Tin	10	1400	Gress	1310000	8	C Y Ho	52	Sv
3	11.11.2017	Sha Tin	9	1600	Gress	1310000	13	Brett Prebble	54	Sv
4	26.11.2017	Sha Tin	9	1600	Gress	1310000	9	C Y Ho	52	Sv
...	...	...	...	...	...	...	...	...	...	
95	10.12.2017	Sha Tin	5	1200	Gress	18500000	13	Francois-Xavier Bertras	57	(B
96	10.12.2017	Sha Tin	7	1600	Gress	23000000	11	Ryan Moore	57	
97	01.10.2017	Sha Tin	7	1000	Gress	3000000	10	Brett Prebble	59	Zea
98	22.10.2017	Sha Tin	7	1200	Gress	4000000	9	Brett Prebble	59	Zea
99	19.11.2017	Sha Tin	7	1200	Gress	4000000	3	Brett Prebble	56	Zea

100 rows × 21 columns

In [40]:

```
b.describe()
```

Out[40]:

	Race Number	Distance	Prize money	Starting position	Jockey weight	Horse age	Path
count	100.000000	100.000000	1.000000e+02	100.000000	100.000000	100.00000	100.000000
mean	6.910000	1446.000000	3.562200e+06	6.170000	55.870000	6.58000	1.510000
std	2.099038	334.820923	4.486259e+06	3.440857	2.942736	1.35721	1.573101
min	1.000000	1000.000000	9.200000e+05	1.000000	49.000000	3.00000	0.000000
25%	6.000000	1200.000000	1.380000e+06	3.000000	54.000000	6.00000	0.000000
50%	7.000000	1400.000000	1.950000e+06	6.000000	56.000000	7.00000	1.000000
75%	8.000000	1650.000000	3.000000e+06	9.000000	58.000000	8.00000	3.000000
max	10.000000	2400.000000	2.300000e+07	14.000000	60.000000	9.00000	6.000000

In [41]:

```
b.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 21 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Dato               100 non-null    object
 1   Track              100 non-null    object
 2   Race Number        100 non-null    int64 
 3   Distance           100 non-null    int64 
 4   Surface            100 non-null    object
 5   Prize money        100 non-null    int64 
 6   Starting position  100 non-null    int64 
 7   Jockey             100 non-null    object
 8   Jockey weight      100 non-null    int64 
 9   Country            100 non-null    object
 10  Horse age          100 non-null    int64 
 11  TrainerName        100 non-null    object
 12  Race time          100 non-null    object
 13  Path               100 non-null    int64 
 14  Final place        100 non-null    int64 
 15  FGrating           100 non-null    int64 
 16  Odds               100 non-null    object
 17  RaceType           100 non-null    object
 18  HorseId            100 non-null    int64 
 19  JockeyId           100 non-null    int64 
 20  TrainerID          100 non-null    int64 
dtypes: int64(12), object(9)
memory usage: 16.5+ KB
```

In [42]:

```
c=b.dropna(axis=1)
c
```

Out[42]:

	Dato	Track	Race Number	Distance	Surface	Prize money	Starting position	Jockey	Jockey weight	Cou
0	03.09.2017	Sha Tin	10	1400	Gress	1310000	6	K C Leung	52	Sv
1	16.09.2017	Sha Tin	10	1400	Gress	1310000	14	C Y Ho	52	Sv
2	14.10.2017	Sha Tin	10	1400	Gress	1310000	8	C Y Ho	52	Sv
3	11.11.2017	Sha Tin	9	1600	Gress	1310000	13	Brett Prebble	54	Sv
4	26.11.2017	Sha Tin	9	1600	Gress	1310000	9	C Y Ho	52	Sv
...	...	...	...	...	...	...	...	...	...	
95	10.12.2017	Sha Tin	5	1200	Gress	18500000	13	Francois-Xavier Bertras	57	(B
96	10.12.2017	Sha Tin	7	1600	Gress	23000000	11	Ryan Moore	57	
97	01.10.2017	Sha Tin	7	1000	Gress	3000000	10	Brett Prebble	59	Zea
98	22.10.2017	Sha Tin	7	1200	Gress	4000000	9	Brett Prebble	59	Zea
99	19.11.2017	Sha Tin	7	1200	Gress	4000000	3	Brett Prebble	56	Zea

100 rows × 21 columns

In [43]:

```
c.columns
```

Out[43]:

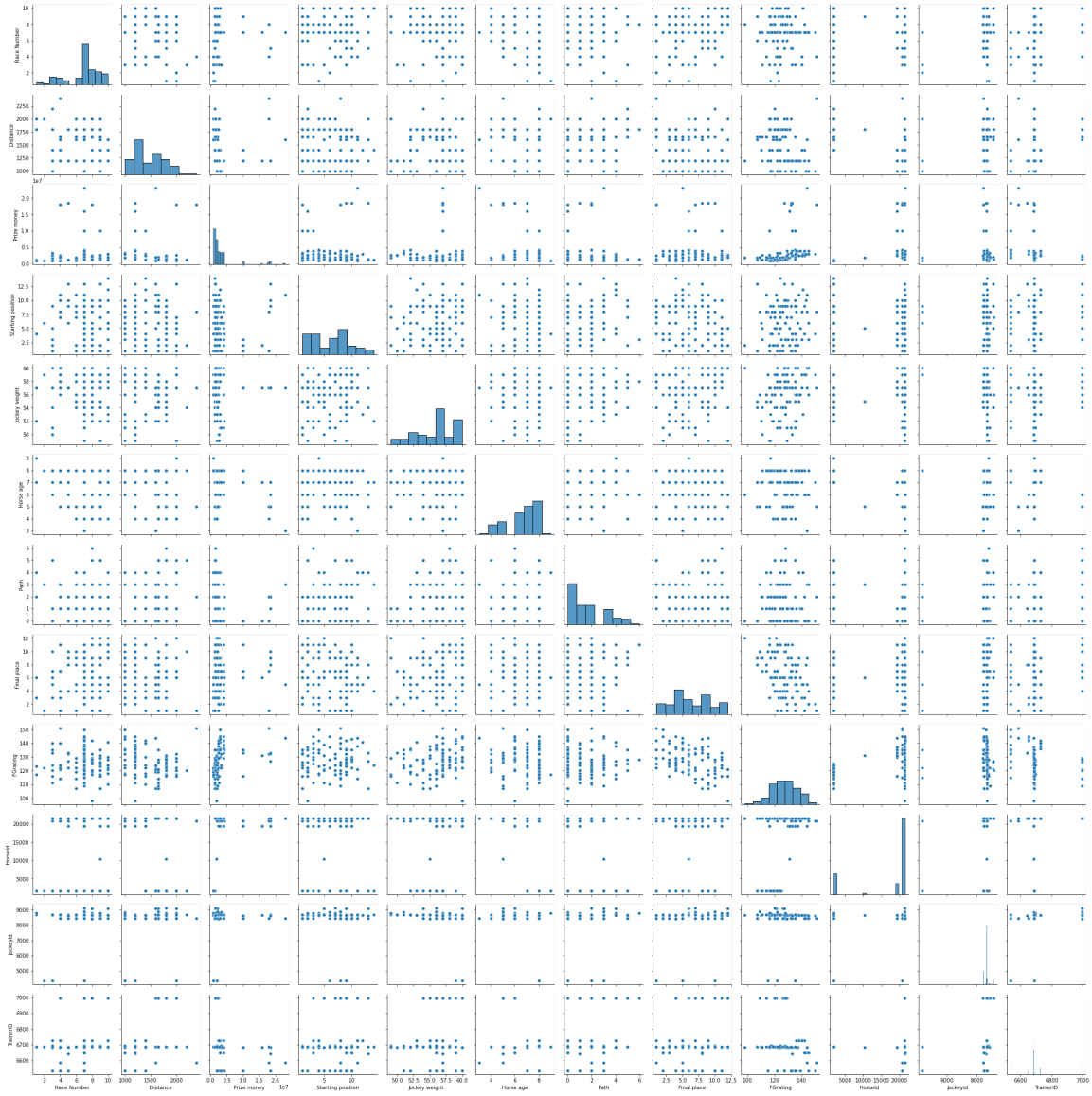
```
Index(['Dato', 'Track', 'Race Number', 'Distance', 'Surface', 'Prize mone
y',
       'Starting position', 'Jockey', 'Jockey weight', 'Country', 'Horse a
ge',
       'TrainerName', 'Race time', 'Path', 'Final place', 'FGrating', 'Odd
s',
       'RaceType', 'HorseId', 'JockeyId', 'TrainerID'],
      dtype='object')
```

In [44]:

```
sns.pairplot(c)
```

Out[44]:

```
<seaborn.axisgrid.PairGrid at 0x171d79f8160>
```



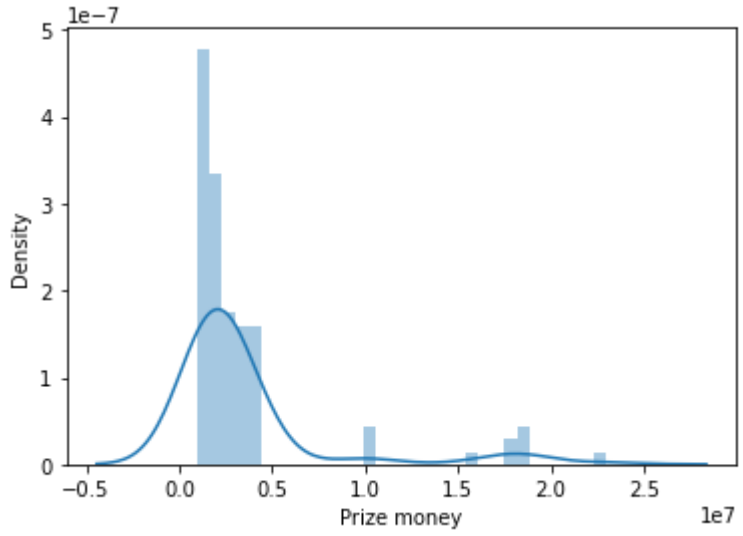
In [45]:

```
sns.distplot(c['Prize money'])
```

```
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[45]:

```
<AxesSubplot:xlabel='Prize money', ylabel='Density'>
```



In [46]:

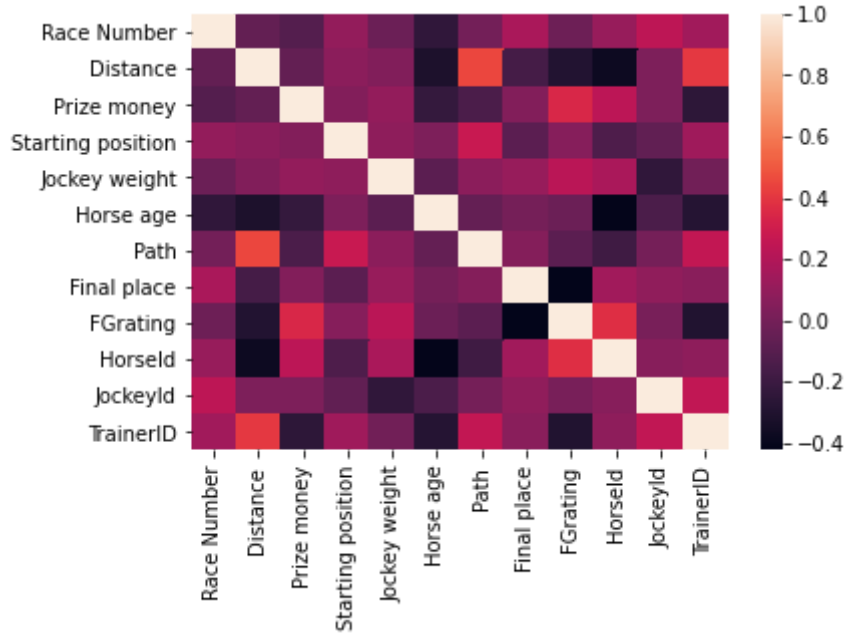
```
f=c[['Dato','Race Number', 'Distance','Prize money',
       'Starting position','Jockey weight','Horse age',
      'Race time', 'Path', 'Final place', 'FGrating', 'Odds',
       'HorseId', 'JockeyId', 'TrainerID']]
```

In [47]:

```
sns.heatmap(f.corr())
```

Out[47]:

<AxesSubplot:>



In [48]:

```
x2=f[[ 'Distance','Starting position','Jockey weight','Horse age','FGrating','TrainerID']]
y2=f['Prize money']
```

In [49]:

```
from sklearn.model_selection import train_test_split
x2_train,x2_test,y2_train,y2_test=train_test_split(x2,y2,test_size=0.5)
```

In [50]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x2_train,y2_train)
```

Out[50]:

```
LinearRegression()
```

In [51]:

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

```
-7943479.703428032
```

In [52]:

```
r=pd.DataFrame(lr.coef_,x2.columns,columns=['Co-efficient'])
r
```

Out[52]:

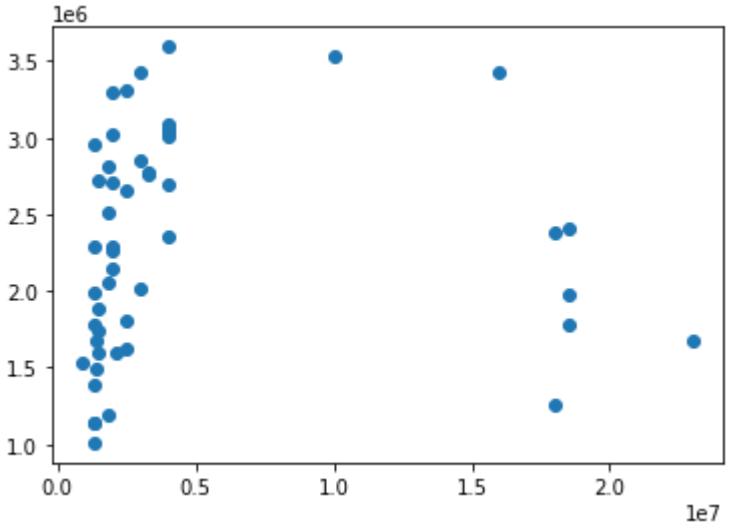
	Co-efficient
Distance	-1405.295242
Starting position	-144265.340447
Jockey weight	16290.610897
Horse age	34287.629446
FGrating	28925.877574
TrainerID	1254.837090

In [53]:

```
u=lr.predict(x2_test)
plt.scatter(y2_test,u)
```

Out[53]:

```
<matplotlib.collections.PathCollection at 0x171dfb31a00>
```



In [54]:

```
print(lr.score(x2_test,y2_test))
```

```
-0.18099745594645333
```

In [55]:

```
lr.score(x2_train,y2_train)
```

Out[55]:

```
0.20270317432237261
```

In [56]:

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

In [57]:

```
rr=Ridge(alpha=10)
rr.fit(x2_train,y2_train)
```

Out[57]:

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

In [58]:

```
rr.score(x2_test,y2_test)
```

Out[58]:

```
-0.17854010111414875
```

In [59]:

```
la=Lasso(alpha=10)
la.fit(x2_train,y2_train)
```

Out[59]:

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

In [60]:

```
la.score(x2_test,y2_test)
```

Out[60]:

```
-0.18099496539455084
```

ELASTIC NET

In [61]:

```
from sklearn.linear_model import ElasticNet
p=ElasticNet()
p.fit(x2_train,y2_train)
```

Out[61]:

```
ElasticNet()
```

In [62]:

```
print(p.coef_)
```

```
[  -1412.0454908  -137098.97426212   14253.01782617   22168.26556704
   28716.42411684    1161.8394489 ]
```

In [63]:

```
print(p.intercept_)
```

```
-7129709.166710248
```

In [64]:

```
print(p.predict(x2_test))
```

```
[2893874.22892909 3284561.6081243  2139951.88279886 2300236.85974332
 2066619.66203657 1400500.6780598  2012053.03819022 3006474.23646753
 3424680.98654762 2703709.07931783 1822122.98536334 1609180.80439627
 2523405.94488993 2369984.33402477 2680927.72586478 2785710.0442954
 2282522.67889327 1767733.98777093 3272606.42229179 2649741.03803308
 1514504.31946627 1260877.91388004 1768407.38197416 1650989.13849134
 3021463.12415376 2803876.65127175 3497843.46505108 2719847.68091414
 3050060.24664378 3585831.74535887 2251600.55928487 3396824.60664115
 2044543.45814441 1851873.11055457 1511915.51732091 1977888.39991734
 1730711.77762278 3035451.46411633 1156516.2672989  2785991.33833103
 2393346.81167774 2958553.4229894  1036481.90569666 1200433.27773343
 1610953.31873499 1737195.42596711 2443679.9453561  1294746.01786834
 3093444.57217237 1874294.40022923]
```

In [65]:

```
print(p.score(x2_test,y2_test))
```

```
-0.17561248978599076
```

EVALUATION METRICS

In [66]:

```
from sklearn import metrics
```

In [67]:

```
print("Mean Absolytre Error:",metrics.mean_absolute_error(y2_test,prediction))
```

```
Mean Absolytre Error: 4704399.176357559
```

In [68]:

```
print("Mean Squared Error:",metrics.mean_squared_error(y2_test,prediction))
```

```
Mean Squared Error: 56256648448046.99
```

In [69]:

```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y2_test,prediction)))
```

```
Root Mean Squared Error: 7500443.216773725
```

16_Sleep_health_and_lifestyle_dataset

In [71]:

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

Out[71]:

	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 [72]:

```
b=a.head(100)
b
```

Out[72]:

	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 [73]:

```
b.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 [74]:

```
b.describe()
```

Out[74]:

	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 [75]:

```
b.columns
```

Out[75]:

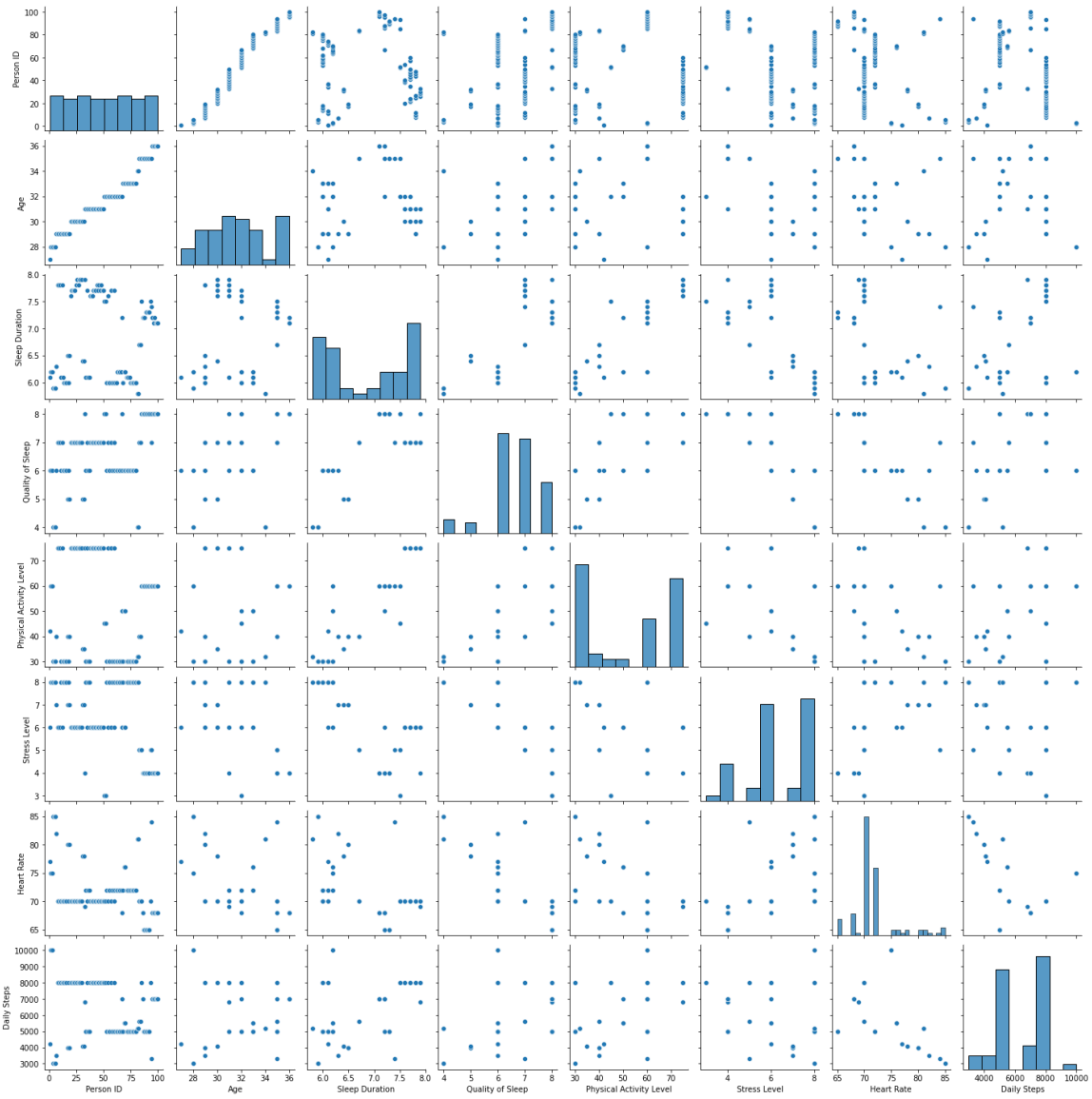
```
Index(['Person ID', 'Gender', 'Age', 'Occupation', 'Sleep Duration',
       'Quality of Sleep', 'Physical Activity Level', 'Stress Level',
       'BMI Category', 'Blood Pressure', 'Heart Rate', 'Daily Steps',
       'Sleep Disorder'],
      dtype='object')
```

In [76]:

```
sns.pairplot(b)
```

Out[76]:

```
<seaborn.axisgrid.PairGrid at 0x171dfb4b4f0>
```



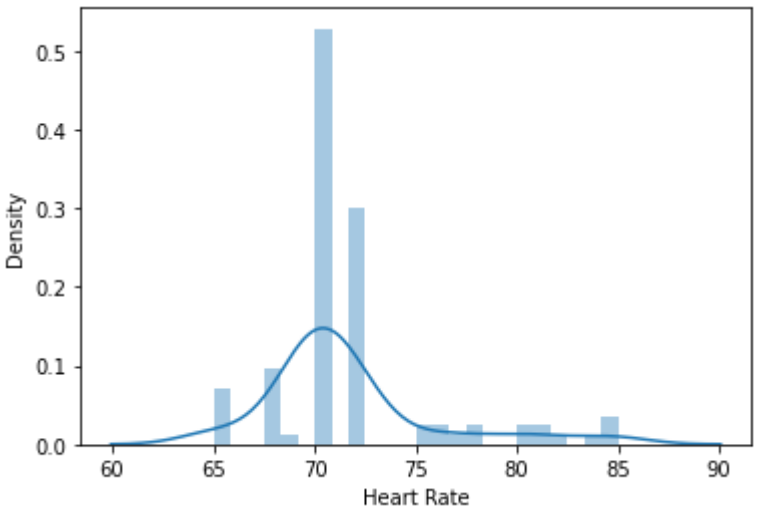
In [77]:

```
sns.distplot(b['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 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[77]:

```
<AxesSubplot:xlabel='Heart Rate', ylabel='Density'>
```



In [78]:

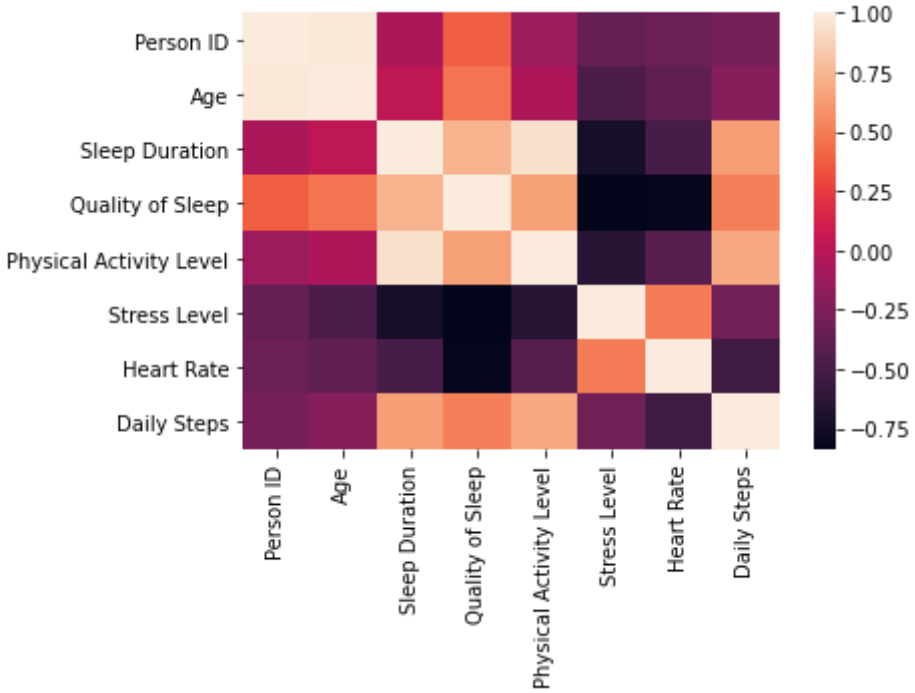
```
f=b[['Person ID', 'Gender', 'Age', 'Occupation', 'Sleep Duration',
       'Quality of Sleep', 'Physical Activity Level', 'Stress Level',
       'BMI Category', 'Blood Pressure', 'Heart Rate', 'Daily Steps',
       'Sleep Disorder']]
```

In [79]:

```
sns.heatmap(f.corr())
```

Out[79]:

<AxesSubplot:>



In [80]:

```
x3=f[['Person ID','Age','Sleep Duration',
       'Quality of Sleep', 'Physical Activity Level', 'Stress Level',
       'Heart Rate', 'Daily Steps']]
y3=f['Heart Rate']
```

In [81]:

```
from sklearn.model_selection import train_test_split
x3_train,x3_test,y3_train,y3_test=train_test_split(x3,y3,test_size=0.5)
```

In [82]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x3_train,y3_train)
```

Out[82]:

```
LinearRegression()
```

In [83]:

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

```
2.8421709430404007e-13
```

In [84]:

```
r=pd.DataFrame(lr.coef_,x3.columns,columns=['Co-efficient'])
r
```

Out[84]:

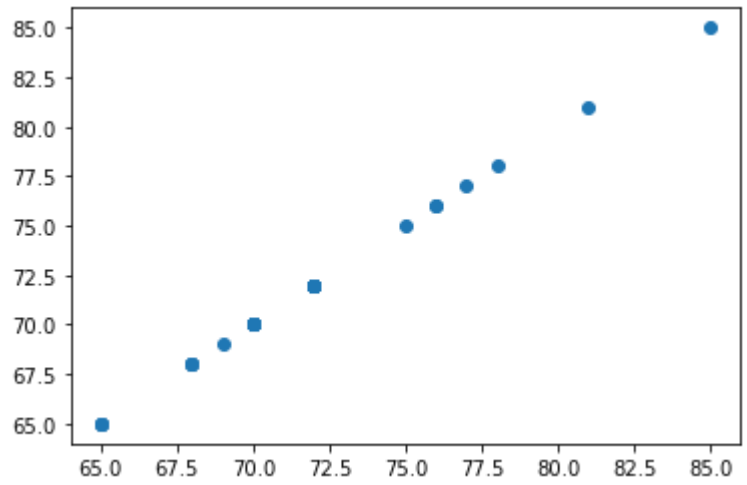
	Co-efficient
Person ID	3.338952e-16
Age	-3.645748e-15
Sleep Duration	-2.959317e-16
Quality of Sleep	-2.342765e-15
Physical Activity Level	2.160301e-17
Stress Level	-1.231670e-15
Heart Rate	1.000000e+00
Daily Steps	-2.874025e-17

In [85]:

```
u=lr.predict(x3_test)
plt.scatter(y3_test,u)
```

Out[85]:

```
<matplotlib.collections.PathCollection at 0x171e3210eb0>
```



In [86]:

```
print(lr.score(x3_test,y3_test))
```

```
1.0
```

In [87]:

```
lr.score(x3_train,y3_train)
```

Out[87]:

```
1.0
```

In [88]:

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

In [89]:

```
rr=Ridge(alpha=10)
rr.fit(x3_train,y3_train)
```

Out[89]:

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

In [90]:

```
rr.score(x3_test,y3_test)
```

Out[90]:

```
0.9995763294228194
```

In [91]:

```
la=Lasso(alpha=10)
la.fit(x3_train,y3_train)
```

Out[91]:

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

In [92]:

```
la.score(x3_test,y3_test)
```

Out[92]:

```
0.28648433285844843
```

ELASTIC NET

In [94]:

```
from sklearn.linear_model import ElasticNet
p=ElasticNet()
p.fit(x3_train,y3_train)
```

Out[94]:

```
ElasticNet()
```

In [95]:

```
print(p.coef_)
```

```
[-8.11075063e-03 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
  0.00000000e+00  0.00000000e+00  8.87540114e-01 -2.15505154e-04]
```

In [96]:

```
print(p.intercept_)
```

```
9.881434969249142
```

In [97]:

```
print(p.predict(x3_test))
```

```
[70.04998994 67.94699457 72.17148786 70.1392082  72.21204161 72.0660481
 70.07432219 70.05810069 70.1554297  72.0984911  65.74782754 70.09054369
 72.41481038 70.2040942  69.88777493 72.22826311 69.95266093 67.93077307
 69.96077168 72.18770936 69.5309019  69.59578791 68.18220634 74.26755972
 65.78027055 69.93643943 69.38861301 79.98647584 69.84722118 70.00132544
 75.58956348 70.06621144 77.9664487  77.30879134 72.20393086 70.17976195
 72.40669962 68.02810208 65.76404904 84.64338618 70.1716512  70.04187919
 72.17959861 69.87966418 75.58145273 70.1229867  65.7883813  70.12111103
 72.10660185 72.1309341 ]
```

In [98]:

```
print(p.score(x3_test,y3_test))
```

```
0.9910400441354008
```

EVALUATION METRICS

In [99]:

```
from sklearn import metrics
```

In [100]:

```
print("Mean Absolytre Error:",metrics.mean_absolute_error(y3_test,prediction))
```

```
Mean Absolytre Error: 70.23635755897878
```

In [101]:

```
print("Mean Squared Error:",metrics.mean_squared_error(y3_test,prediction))
```

```
Mean Squared Error: 4947.82351088423
```

In [102]:

```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y3_test,prediction)))
```

```
Root Mean Squared Error: 70.34076706209729
```

17_student_marks

In [103]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\17_student_marks.csv")
a
```

Out[103]:

	Student_ID	Test_1	Test_2	Test_3	Test_4	Test_5	Test_6	Test_7	Test_8	Test_9	Test_
0	22000	78	87	91	91	88	98	94	100	100	1
1	22001	79	71	81	72	73	68	59	69	59	
2	22002	66	65	70	74	78	86	87	96	88	
3	22003	60	58	54	61	54	57	64	62	72	
4	22004	99	95	96	93	97	89	92	98	91	
5	22005	41	36	35	28	35	36	27	26	19	
6	22006	47	50	47	57	62	64	71	75	85	
7	22007	84	74	70	68	58	59	56	56	64	
8	22008	74	64	58	57	53	51	47	45	42	
9	22009	87	81	73	74	71	63	53	45	39	
10	22010	40	34	37	33	31	35	39	38	40	
11	22011	91	84	78	74	76	80	80	73	75	
12	22012	81	83	93	88	89	90	99	99	95	
13	22013	52	50	42	38	33	30	28	22	12	
14	22014	63	67	65	74	80	86	95	96	92	
15	22015	76	82	88	94	85	76	70	60	50	
16	22016	83	78	71	71	77	72	66	75	66	
17	22017	55	45	43	38	43	35	44	37	45	
18	22018	71	67	76	74	64	61	57	64	61	
19	22019	62	61	53	49	54	59	68	74	65	
20	22020	44	38	36	34	26	34	39	44	36	
21	22021	50	56	53	46	41	38	47	39	44	
22	22022	57	48	40	45	43	36	26	19	9	
23	22023	59	56	52	44	50	40	45	46	54	
24	22024	84	92	89	80	90	80	84	74	68	
25	22025	74	80	86	87	90	100	95	87	85	
26	22026	92	84	74	83	93	83	75	82	81	
27	22027	63	70	74	65	64	55	61	58	48	
28	22028	78	77	69	76	78	74	67	69	78	
29	22029	55	58	59	67	71	62	53	61	67	
30	22030	54	54	48	38	35	45	46	47	41	
31	22031	84	93	97	89	86	95	100	100	100	
32	22032	95	100	94	100	98	99	100	90	80	
33	22033	64	61	63	73	63	68	64	58	50	
34	22034	76	79	73	77	83	86	95	89	90	
35	22035	78	71	61	55	54	48	41	32	41	
36	22036	95	89	91	84	89	94	85	91	100	1

	Student_ID	Test_1	Test_2	Test_3	Test_4	Test_5	Test_6	Test_7	Test_8	Test_9	Test_
37	22037	99	89	79	87	87	81	82	74	64	
38	22038	82	83	85	86	89	80	88	95	87	
39	22039	65	56	64	62	58	51	61	68	70	
40	22040	100	93	92	86	84	76	82	74	79	
41	22041	78	72	73	79	81	73	71	77	83	
42	22042	98	100	100	93	94	92	100	100	98	
43	22043	58	62	67	77	71	63	64	73	83	
44	22044	96	92	94	100	99	95	98	92	84	
45	22045	86	87	85	84	85	91	86	82	85	
46	22046	48	55	46	40	34	29	37	34	39	
47	22047	56	52	54	47	40	35	43	44	40	
48	22048	42	44	46	53	62	59	57	53	43	
49	22049	64	54	49	59	54	55	57	59	63	
50	22050	50	44	37	29	37	46	53	57	55	
51	22051	70	60	70	62	67	67	68	67	72	
52	22052	63	73	70	63	60	67	61	59	52	
53	22053	92	100	100	100	100	100	92	87	94	1
54	22054	64	55	54	61	63	57	47	37	44	
55	22055	60	66	68	58	49	47	39	29	39	

In [104]:

```
b=a.head(100)
b
```

Out[104]:

	Student_ID	Test_1	Test_2	Test_3	Test_4	Test_5	Test_6	Test_7	Test_8	Test_9	Test_
0	22000	78	87	91	91	88	98	94	100	100	1
1	22001	79	71	81	72	73	68	59	69	59	
2	22002	66	65	70	74	78	86	87	96	88	
3	22003	60	58	54	61	54	57	64	62	72	
4	22004	99	95	96	93	97	89	92	98	91	
5	22005	41	36	35	28	35	36	27	26	19	
6	22006	47	50	47	57	62	64	71	75	85	
7	22007	84	74	70	68	58	59	56	56	64	
8	22008	74	64	58	57	53	51	47	45	42	
9	22009	87	81	73	74	71	63	53	45	39	
10	22010	40	34	37	33	31	35	39	38	40	
11	22011	91	84	78	74	76	80	80	73	75	
12	22012	81	83	93	88	89	90	99	99	95	
13	22013	52	50	42	38	33	30	28	22	12	
14	22014	63	67	65	74	80	86	95	96	92	
15	22015	76	82	88	94	85	76	70	60	50	
16	22016	83	78	71	71	77	72	66	75	66	
17	22017	55	45	43	38	43	35	44	37	45	
18	22018	71	67	76	74	64	61	57	64	61	
19	22019	62	61	53	49	54	59	68	74	65	
20	22020	44	38	36	34	26	34	39	44	36	
21	22021	50	56	53	46	41	38	47	39	44	
22	22022	57	48	40	45	43	36	26	19	9	
23	22023	59	56	52	44	50	40	45	46	54	
24	22024	84	92	89	80	90	80	84	74	68	
25	22025	74	80	86	87	90	100	95	87	85	
26	22026	92	84	74	83	93	83	75	82	81	
27	22027	63	70	74	65	64	55	61	58	48	
28	22028	78	77	69	76	78	74	67	69	78	
29	22029	55	58	59	67	71	62	53	61	67	
30	22030	54	54	48	38	35	45	46	47	41	
31	22031	84	93	97	89	86	95	100	100	100	
32	22032	95	100	94	100	98	99	100	90	80	
33	22033	64	61	63	73	63	68	64	58	50	
34	22034	76	79	73	77	83	86	95	89	90	
35	22035	78	71	61	55	54	48	41	32	41	
36	22036	95	89	91	84	89	94	85	91	100	1

	Student_ID	Test_1	Test_2	Test_3	Test_4	Test_5	Test_6	Test_7	Test_8	Test_9	Test_
37	22037	99	89	79	87	87	81	82	74	64	
38	22038	82	83	85	86	89	80	88	95	87	
39	22039	65	56	64	62	58	51	61	68	70	
40	22040	100	93	92	86	84	76	82	74	79	
41	22041	78	72	73	79	81	73	71	77	83	
42	22042	98	100	100	93	94	92	100	100	98	
43	22043	58	62	67	77	71	63	64	73	83	
44	22044	96	92	94	100	99	95	98	92	84	
45	22045	86	87	85	84	85	91	86	82	85	
46	22046	48	55	46	40	34	29	37	34	39	
47	22047	56	52	54	47	40	35	43	44	40	
48	22048	42	44	46	53	62	59	57	53	43	
49	22049	64	54	49	59	54	55	57	59	63	
50	22050	50	44	37	29	37	46	53	57	55	
51	22051	70	60	70	62	67	67	68	67	72	
52	22052	63	73	70	63	60	67	61	59	52	
53	22053	92	100	100	100	100	100	92	87	94	1
54	22054	64	55	54	61	63	57	47	37	44	
55	22055	60	66	68	58	49	47	39	29	39	

In [105]:

```
b.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56 entries, 0 to 55
Data columns (total 13 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Student_ID  56 non-null     int64
 1   Test_1      56 non-null     int64
 2   Test_2      56 non-null     int64
 3   Test_3      56 non-null     int64
 4   Test_4      56 non-null     int64
 5   Test_5      56 non-null     int64
 6   Test_6      56 non-null     int64
 7   Test_7      56 non-null     int64
 8   Test_8      56 non-null     int64
 9   Test_9      56 non-null     int64
 10  Test_10     56 non-null     int64
 11  Test_11     56 non-null     int64
 12  Test_12     56 non-null     int64
dtypes: int64(13)
memory usage: 5.8 KB
```

In [106]:

```
b.describe()
```

Out[106]:

	Student_ID	Test_1	Test_2	Test_3	Test_4	Test_5	Test_6
count	56.000000	56.000000	56.000000	56.000000	56.000000	56.000000	56.000000
mean	22027.500000	70.750000	69.196429	68.089286	67.446429	67.303571	66.000000
std	16.309506	17.009356	17.712266	18.838333	19.807179	20.746890	21.054043
min	22000.000000	40.000000	34.000000	35.000000	28.000000	26.000000	29.000000
25%	22013.750000	57.750000	55.750000	53.000000	54.500000	53.750000	50.250000
50%	22027.500000	70.500000	68.500000	70.000000	71.500000	69.000000	65.500000
75%	22041.250000	84.000000	83.250000	85.000000	84.000000	85.250000	83.750000
max	22055.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000

In [107]:

```
b.columns
```

Out[107]:

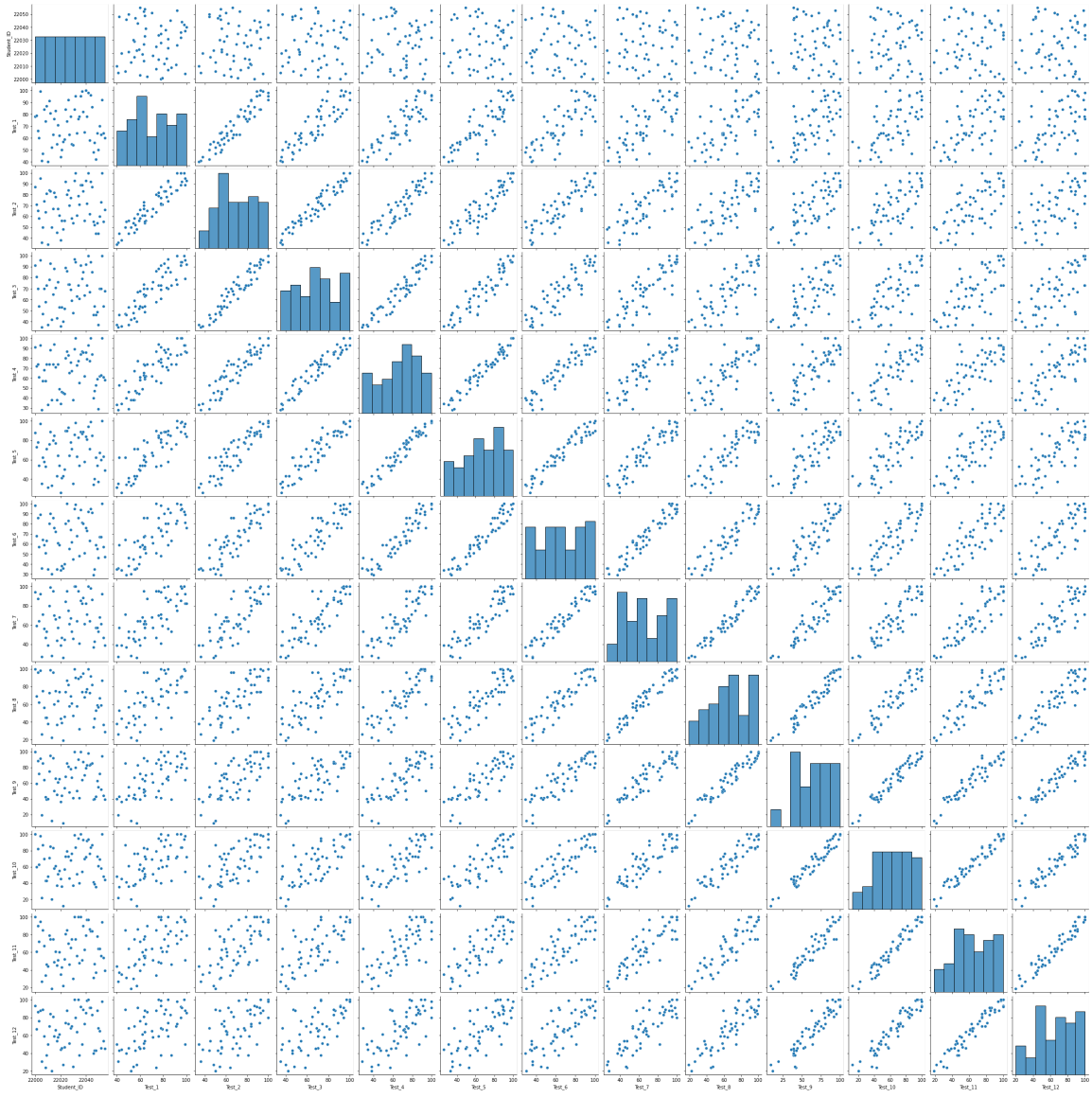
```
Index(['Student_ID', 'Test_1', 'Test_2', 'Test_3', 'Test_4', 'Test_5',
       'Test_6', 'Test_7', 'Test_8', 'Test_9', 'Test_10', 'Test_11',
       'Test_12'],
      dtype='object')
```

In [108]:

```
sns.pairplot(b)
```

Out[108]:

```
<seaborn.axisgrid.PairGrid at 0x171e3238580>
```



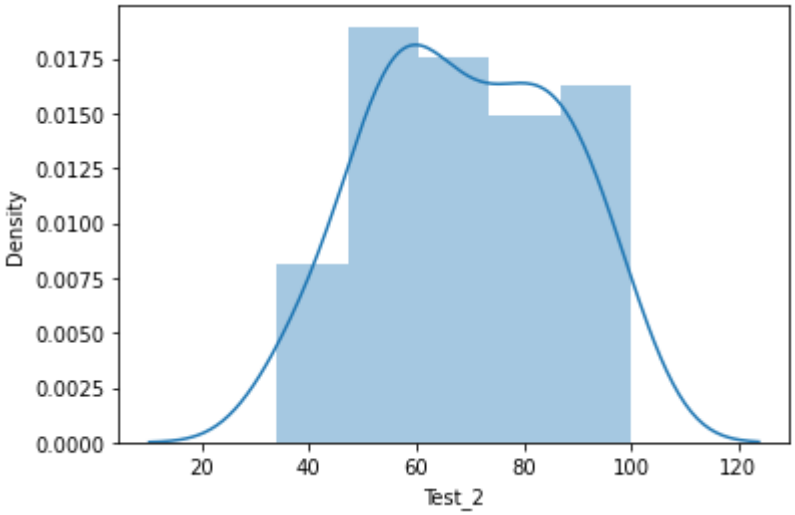
In [109]:

```
sns.distplot(b['Test_2'])
```

```
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[109]:

```
<AxesSubplot:xlabel='Test_2', ylabel='Density'>
```



In [110]:

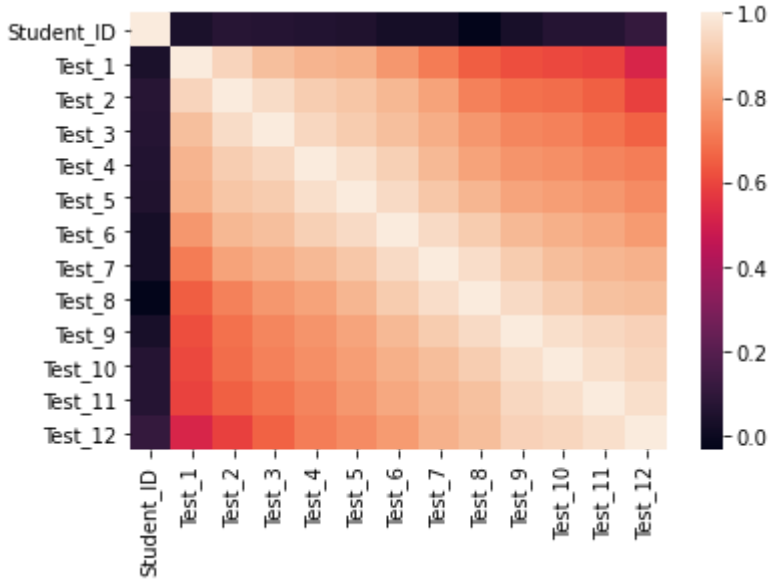
```
f=b[['Student_ID', 'Test_1', 'Test_2', 'Test_3', 'Test_4', 'Test_5',
       'Test_6', 'Test_7', 'Test_8', 'Test_9', 'Test_10', 'Test_11',
       'Test_12']]
```

In [111]:

```
sns.heatmap(f.corr())
```

Out[111]:

<AxesSubplot:>



In [112]:

```
x4=f[['Student_ID', 'Test_1','Test_3', 'Test_4', 'Test_5',
       'Test_6', 'Test_7', 'Test_8', 'Test_9', 'Test_10', 'Test_11',
       'Test_12']]
y4=f['Test_2']
```

In [113]:

```
from sklearn.model_selection import train_test_split
x4_train,x4_test,y4_train,y4_test=train_test_split(x4,y4,test_size=0.5)
```

In [114]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x4_train,y4_train)
```

Out[114]:

```
LinearRegression()
```

In [115]:

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

```
-1936.3869042265571
```

In [116]:

```
r=pd.DataFrame(lr.coef_,x4.columns,columns=['Co-efficient'])
r
```

Out[116]:

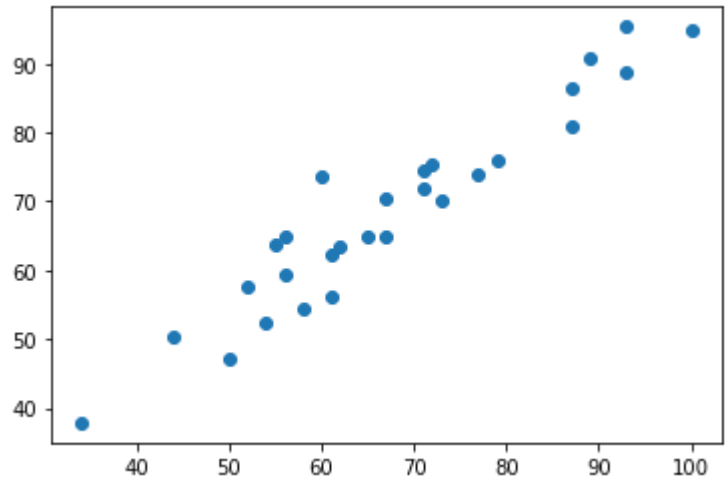
	Co-efficient
Student_ID	0.088189
Test_1	0.371436
Test_3	0.492043
Test_4	-0.063353
Test_5	0.261007
Test_6	-0.180162
Test_7	0.210883
Test_8	-0.176475
Test_9	0.069315
Test_10	0.058202
Test_11	0.067368
Test_12	-0.191963

In [118]:

```
u=lr.predict(x4_test)
plt.scatter(y4_test,u)
```

Out[118]:

```
<matplotlib.collections.PathCollection at 0x171ec20b070>
```



In [119]:

```
print(lr.score(x4_test,y4_test))
```

```
0.9075725286491936
```

In [120]:

```
lr.score(x4_train,y4_train)
```

Out[120]:

```
0.9860267918987687
```

In [121]:

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

In [122]:

```
rr=Ridge(alpha=10)
rr.fit(x4_train,y4_train)
```

Out[122]:

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

In [124]:

```
rr.score(x4_test,y4_test)
```

Out[124]:

```
0.9081261200925351
```

In [125]:

```
la=Lasso(alpha=10)
la.fit(x4_train,y4_train)
```

Out[125]:

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

In [126]:

```
la.score(x4_test,y4_test)
```

Out[126]:

```
0.9103242376616466
```

ELASTIC NET

In [170]:

```
from sklearn.linear_model import ElasticNet
p=ElasticNet()
p.fit(x4_train,y4_train)
```

Out[170]:

```
ElasticNet()
```

In [171]:

```
print(p.coef_)
```

```
[ 0.07928517  0.35948343  0.51489021 -0.          0.15815984 -0.10461042
  0.09059163 -0.07292736  0.05057222  0.          0.07337492 -0.14493778]
```

In [172]:

```
print(p.intercept_)
```

```
-1740.5648305729223
```

In [173]:

```
print(p.predict(x4_test))
```

```
[49.88745773 70.29717482 55.08295575 86.1251413  73.98719324 75.54328634
 65.1077985  89.9578371  72.44900834 57.96129177 74.95397091 47.17348935
 52.43330373 37.939942   62.72285822 89.63881295 56.77317452 94.38943327
 64.61147947 82.40823394 63.55055548 70.91961992 64.89729884 57.696127
 72.7367411  76.38735574 66.44555203 95.03758024]
```

In [174]:

```
prediction=p.predict(x4_test)
print(p.score(x4_test,y4_test))
```

```
0.9100721717195225
```

EVALUATION METRICS

In [175]:

```
from sklearn import metrics
```

In [176]:

```
print("Mean Absolytre Error:",metrics.mean_absolute_error(y4_test,prediction))
```

```
Mean Absolytre Error: 3.9116681158081894
```

In [177]:

```
print("Mean Squared Error:",metrics.mean_squared_error(y4_test,prediction))
```

```
Mean Squared Error: 22.380558855578638
```

In [178]:

```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y4_test,prediction)))
```

```
Root Mean Squared Error: 4.730809534908231
```

18_world-data-2023

In [136]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\18_world-data-2023.csv")
a
```

Out[136]:

	Country	Density\n(P/Km2)	Abbreviation	Agricultural Land(%)	Land Area(Km2)	Armed Forces size	Birth Rate	Call Cc
0	Afghanistan	60	AF	58.10%	652,230	323,000	32.49	9
1	Albania	105	AL	43.10%	28,748	9,000	11.78	35
2	Algeria	18	DZ	17.40%	2,381,741	317,000	24.28	21
3	Andorra	164	AD	40.00%	468	NaN	7.20	37
4	Angola	26	AO	47.50%	1,246,700	117,000	40.73	24
...	...	...	...	...	...	...	...	
190	Venezuela	32	VE	24.50%	912,050	343,000	17.88	5
191	Vietnam	314	VN	39.30%	331,210	522,000	16.75	8
192	Yemen	56	YE	44.60%	527,968	40,000	30.45	96
193	Zambia	25	ZM	32.10%	752,618	16,000	36.19	26
194	Zimbabwe	38	ZW	41.90%	390,757	51,000	30.68	26

195 rows × 35 columns

In [137]:

```
b=a.head(100)
b
```

Out[137]:

	Country	Density\n(P/Km2)	Abbreviation	Agricultural Land(%)	Land Area(Km2)	Armed Forces size	Birth Rate	Call Cc
0	Afghanistan	60	AF	58.10%	652,230	323,000	32.49	9
1	Albania	105	AL	43.10%	28,748	9,000	11.78	35
2	Algeria	18	DZ	17.40%	2,381,741	317,000	24.28	21
3	Andorra	164	AD	40.00%	468	NaN	7.20	37
4	Angola	26	AO	47.50%	1,246,700	117,000	40.73	24
...	...	...	...	...	...	...	...	
95	Lesotho	71	LS	77.60%	30,355	2,000	26.81	26
96	Liberia	53	LR	28.00%	111,369	2,000	33.04	23
97	Libya	4	LY	8.70%	1,759,540	0	18.83	21
98	Liechtenstein	238	LI	32.20%	160	NaN	9.90	42
99	Lithuania	43	LT	47.20%	65,300	34,000	10.00	37

100 rows × 35 columns

In [138]:

```
b.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 35 columns):
 #   Column                                      Non-Null Count  Dtype
---  ------                                      --------------  -----
 0   Country                                     100 non-null    object
 1   Density
(P/Km2)                             100 non-null    object
 2   Abbreviation                                96 non-null     object
 3   Agricultural Land( %)                       98 non-null     object
 4   Land Area(Km2)                              100 non-null    object
 5   Armed Forces size                           92 non-null     object
 6   Birth Rate                                  98 non-null     float64
 7   Calling Code                                100 non-null    float64
 8   Capital/Major City                          99 non-null     object
 9   Co2-Emissions                               98 non-null     object
 10  CPI                                         93 non-null     object
 11  CPI Change (%)                              94 non-null     object
 12  Currency-Code                               91 non-null     object
 13  Fertility Rate                              98 non-null     float64
 14  Forested Area (%)                           98 non-null     object
 15  Gasoline Price                              93 non-null     object
 16  GDP                                         99 non-null     object
 17  Gross primary education enrollment (%)      97 non-null     object
 18  Gross tertiary education enrollment (%)     95 non-null     object
 19  Infant mortality                            97 non-null     float64
 20  Largest city                                97 non-null     object
 21  Life expectancy                             97 non-null     float64
 22  Maternal mortality ratio                    95 non-null     float64
 23  Minimum wage                                80 non-null     object
 24  Official language                           100 non-null    object
 25  Out of pocket health expenditure            97 non-null     object
 26  Physicians per thousand                     97 non-null     float64
 27  Population                                  100 non-null    object
 28  Population: Labor force participation (%)   92 non-null     object
 29  Tax revenue (%)                             87 non-null     object
 30  Total tax rate                              96 non-null     object
 31  Unemployment rate                           92 non-null     object
 32  Urban_population                            98 non-null     object
 33  Latitude                                    100 non-null    float64
 34  Longitude                                   100 non-null    float64
dtypes: float64(9), object(26)
memory usage: 27.5+ KB
```

In [139]:

```
b.describe()
```

Out[139]:

	Birth Rate	Calling Code	Fertility Rate	Infant mortality	Life expectancy	Maternal mortality ratio	Physicians per thousand	
count	98.000000	100.00000	98.000000	97.000000	97.000000	95.000000	97.000000	10(
mean	20.045000	354.15000	2.651633	21.782474	72.358763	164.305263	1.916186	2(
std	9.737986	330.66234	1.239758	20.437751	7.712838	226.488977	1.809687	2:
min	7.200000	1.00000	1.270000	1.400000	52.800000	2.000000	0.040000	-3(
25%	11.420000	84.75000	1.695000	5.000000	66.600000	12.000000	0.280000	1
50%	18.125000	253.50000	2.180000	12.800000	74.100000	58.000000	1.580000	1(
75%	27.027500	501.25000	3.322500	32.900000	78.100000	244.500000	3.190000	4
max	42.170000	1876.00000	5.920000	84.500000	84.200000	1140.000000	8.420000	6(

In [140]:

```
c=b.dropna(axis=1)
c
```

Out[140]:

	Country	Density\n(P/Km2)	Land Area(Km2)	Calling Code	Official language	Population	Latitude	Lo
0	Afghanistan	60	652,230	93.0	Pashto	38,041,754	33.939110	6
1	Albania	105	28,748	355.0	Albanian	2,854,191	41.153332	2(
2	Algeria	18	2,381,741	213.0	Arabic	43,053,054	28.033886	1
3	Andorra	164	468	376.0	Catalan	77,142	42.506285	1
4	Angola	26	1,246,700	244.0	Portuguese	31,825,295	-11.202692	1
...	...	...	...	...	...	...	...	
95	Lesotho	71	30,355	266.0	English	2,125,268	-29.609988	2(
96	Liberia	53	111,369	231.0	English	4,937,374	6.428055	-9
97	Libya	4	1,759,540	218.0	Arabic	6,777,452	26.335100	1
98	Liechtenstein	238	160	423.0	German	38,019	47.141039	9
99	Lithuania	43	65,300	370.0	Lithuanian	2,786,844	55.169438	2:

100 rows × 8 columns

In [141]:

```
c.columns
```

Out[141]:

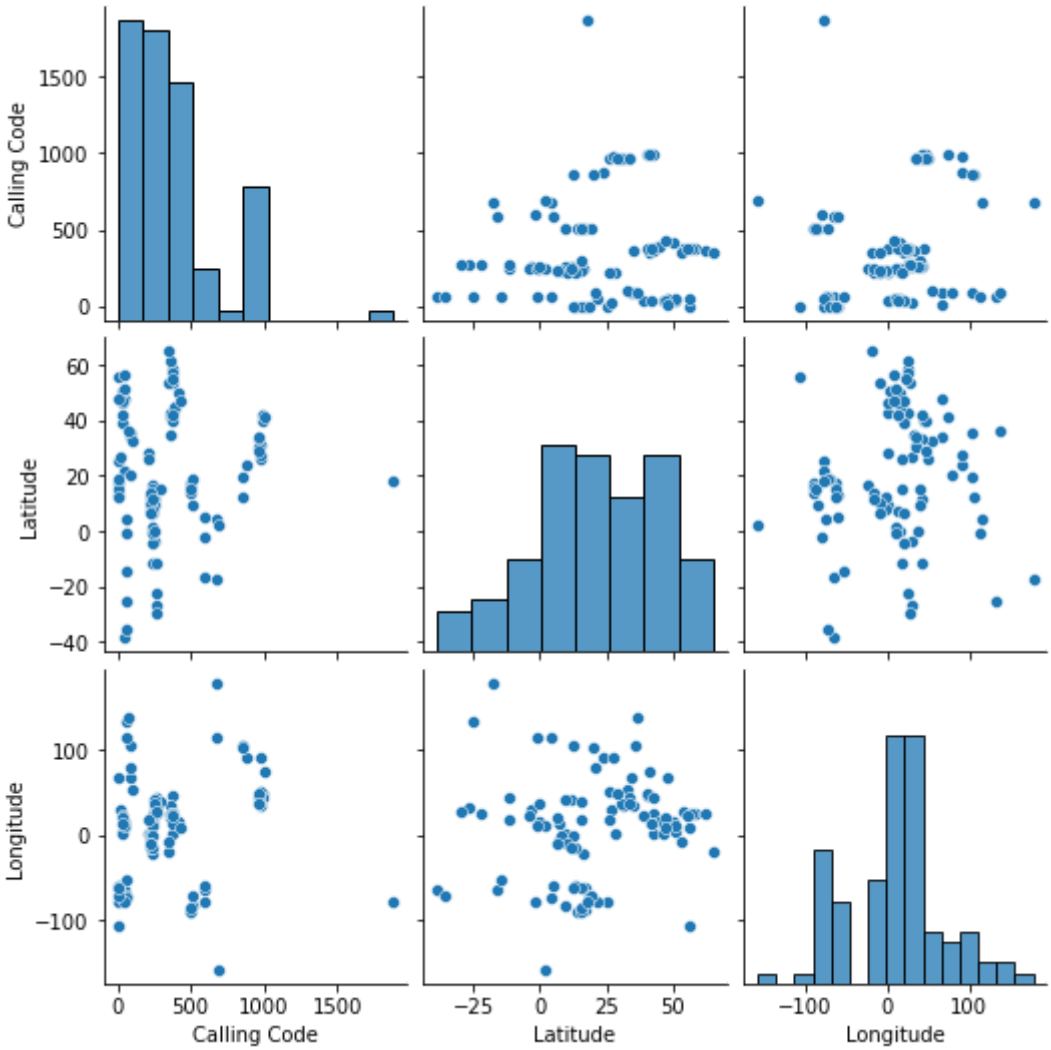
```
Index(['Country', 'Density\n(P/Km2)', 'Land Area(Km2)', 'Calling Code',
       'Official language', 'Population', 'Latitude', 'Longitude'],
      dtype='object')
```

In [142]:

```
sns.pairplot(c)
```

Out[142]:

```
<seaborn.axisgrid.PairGrid at 0x171ec34a2e0>
```



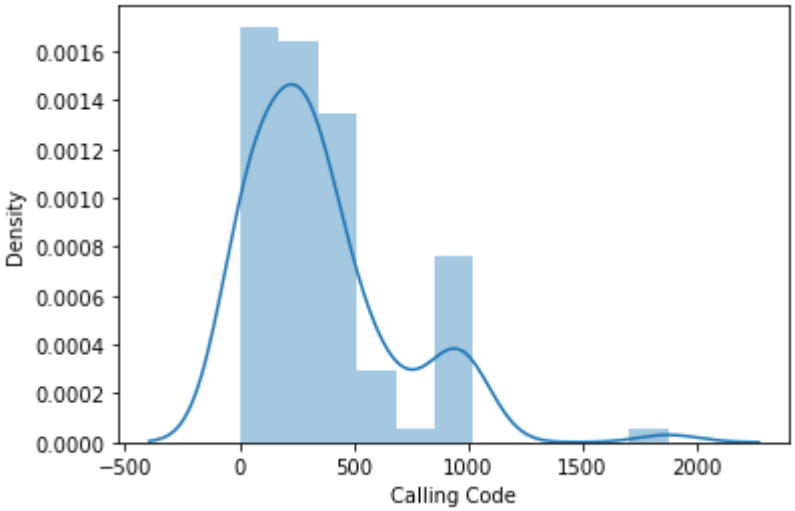
In [143]:

```
sns.distplot(c['Calling Code'])
```

```
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[143]:

```
<AxesSubplot:xlabel='Calling Code', ylabel='Density'>
```



In [144]:

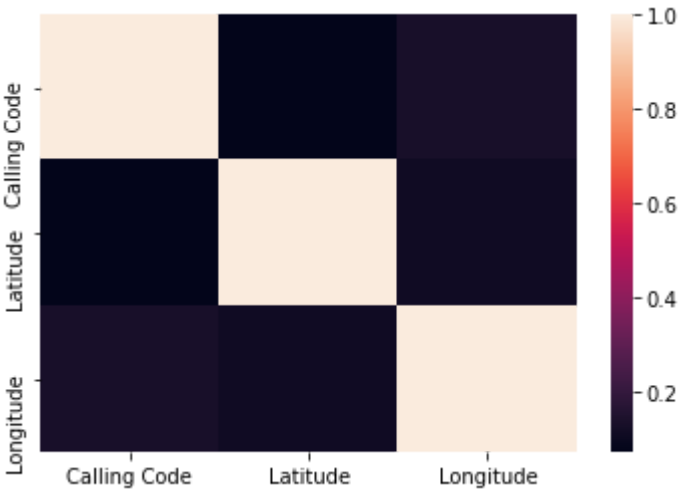
```
f=c[['Country', 'Density\n(P/Km2)', 'Land Area(Km2)', 'Calling Code',
       'Official language', 'Population', 'Latitude', 'Longitude']]
```

In [145]:

```
sns.heatmap(f.corr())
```

Out[145]:

```
<AxesSubplot:>
```



In [146]:

```
x=f[['Latitude', 'Longitude']]
y=f['Calling Code']
```

In [147]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.5)
```

In [148]:

```
from sklearn.linear_model import LinearRegression

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

Out[148]:

```
LinearRegression()
```

In [149]:

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

```
362.5847124562507
```

In [150]:

```
r=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
r
```

Out[150]:

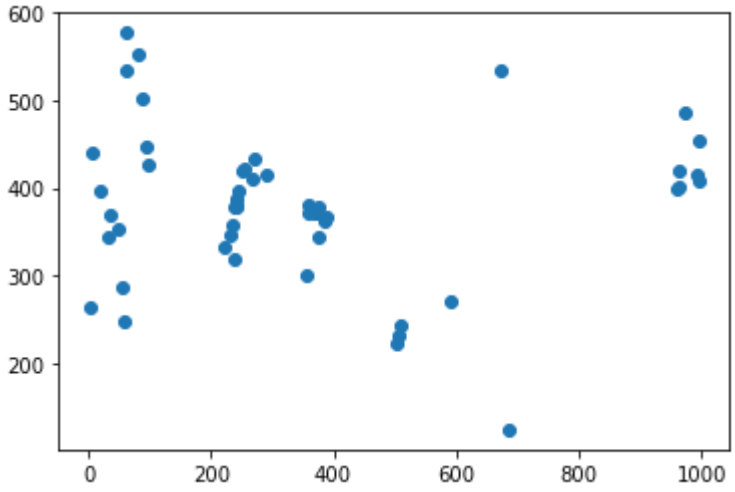
	Co-efficient
Latitude	-0.498721
Longitude	1.511282

In [151]:

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

Out[151]:

```
<matplotlib.collections.PathCollection at 0x171eca85520>
```



In [152]:

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

```
-0.10366393692391407
```

In [153]:

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

Out[153]:

```
0.06275371502200777
```

In [154]:

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

In [155]:

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

Out[155]:

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

In [156]:

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

Out[156]:

```
-0.10364803763482766
```

In [157]:

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

Out[157]:

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

In [158]:

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

Out[158]:

```
-0.10279538065354266
```

ELASTIC NET

In [159]:

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

Out[159]:

```
ElasticNet()
```

In [160]:

```
print(p.coef_)
```

```
[-0.49725868  1.51090022]
```

In [161]:

```
print(p.intercept_)
```

```
362.5559749931516
```

In [162]:

```
print(p.predict(x_test))
```

```
[395.75872307 377.5580689  397.43924134 386.58337386 377.24886466
 419.72070815 368.57156158 379.8127267  343.71864512 301.51395
 371.20471649 357.0585514  427.54819658 344.1961059  410.95537202
 231.16516868 553.43918496 221.3829101  533.6428293  363.09529168
 378.08308861 399.90575524 447.98244062 415.11503162 407.02220901
 352.90460375 332.73914364 535.07220671 485.51046653 439.79283411
 123.86187229 285.54026919 455.03085304 319.51805242 502.15230125
 243.90664037 263.34012556 421.02547825 248.02721451 419.18440012
 380.49632781 371.4450462  414.47845684 271.10146908 402.09986423
 345.11253672 433.81909344 577.24477111 367.42973912 370.66656053]
```

In [163]:

```
prediction=p.predict(x_test)
print(p.score(x_test,y_test))
```

```
-0.10358069305318973
```

EVALUATION METRICS

In [164]:

```
from sklearn import metrics
```

In [165]:

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

```
Mean Absolytre Error: 255.90098769716585
```

In [166]:

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

```
Mean Squared Error: 99755.58333391439
```

In [167]:

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

```
Root Mean Squared Error: 315.84107290521035
```

In [ ]:

```

```