In [1]:

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

14. DataSet 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(10)
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
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa

In [4]:

```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [5]:

```
a.describe()
```

Out[5]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [6]:

```
a.columns
```

Out[6]:

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

In [7]:

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

Out[7]:

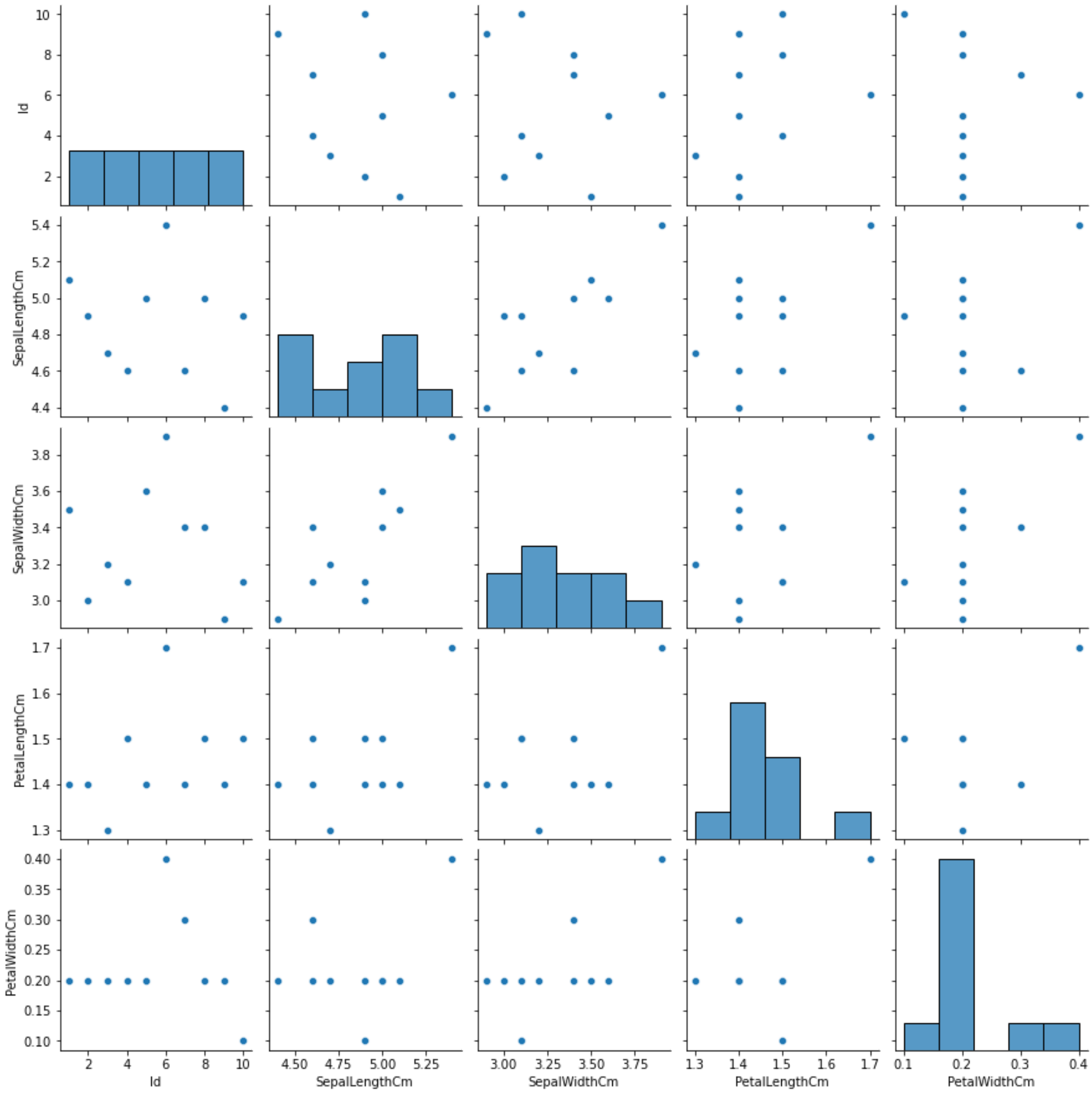
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	1	5.1	3.5	1.4	0.2
1	2	4.9	3.0	1.4	0.2
2	3	4.7	3.2	1.3	0.2
3	4	4.6	3.1	1.5	0.2
4	5	5.0	3.6	1.4	0.2
5	6	5.4	3.9	1.7	0.4
6	7	4.6	3.4	1.4	0.3
7	8	5.0	3.4	1.5	0.2
8	9	4.4	2.9	1.4	0.2
9	10	4.9	3.1	1.5	0.1

In [8]:

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

Out[8]:

<seaborn.axisgrid.PairGrid at 0x23868165550>



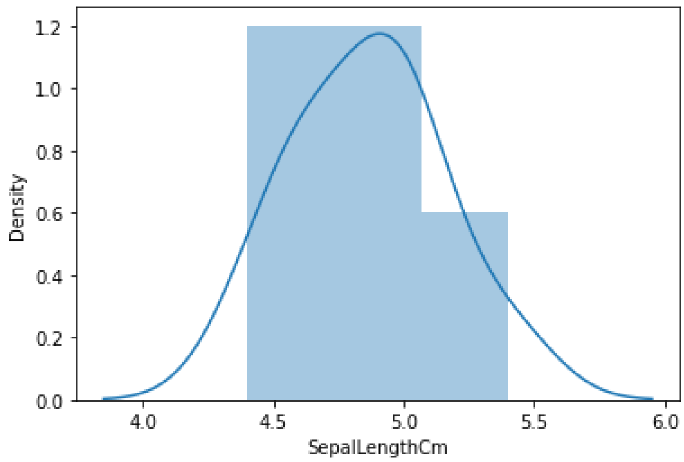
In [9]:

```
sns.distplot(c['SepalLengthCm'])
```

```
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 flexibili
ty) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[9]:

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

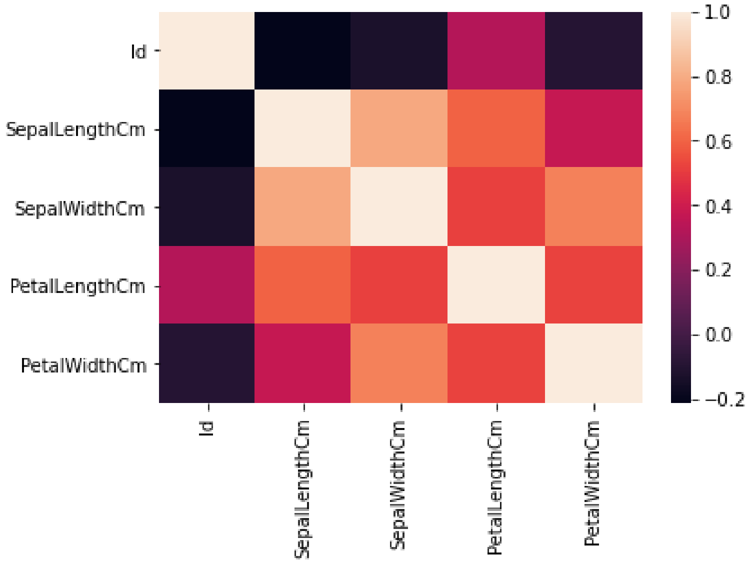


In [10]:

```
sns.heatmap(c.corr())
```

Out[10]:

```
<AxesSubplot:>
```



In [11]:

```
x=b[['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]
y=b['SepalWidthCm']
```

In [12]:

```
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 [13]:

```
from sklearn.linear_model import LinearRegression

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

Out[13]:

```
LinearRegression()
```

In [14]:

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

```
-2.6645352591003757e-15
```

In [15]:

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

Out[15]:

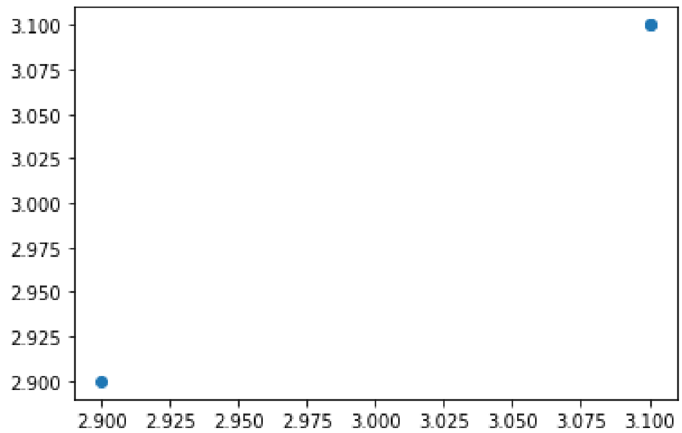
	Co-efficient
Id	4.297420e-17
SepalLengthCm	1.603101e-15
SepalWidthCm	1.000000e+00
PetalLengthCm	-3.060282e-15
PetalWidthCm	2.052200e-15

In [16]:

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

Out[16]:

```
<matplotlib.collections.PathCollection at 0x2386a17fca0>
```



In [17]:

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

```
1.0
```

In [18]:

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

In [19]:

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

Out[19]:

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

In [20]:

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

Out[20]:

```
-26.301964396877597
```

In [21]:

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

Out[21]:

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

In [22]:

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

Out[22]:

```
-17.573979591836686
```

15. DataSet Horse_Racing

In [23]:

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

Out[23]:

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

27008 rows × 21 columns

In [24]:

```
b=a.head(10)
b
```

Out[24]:

	Dato	Track	Race Number	Distance	Surface	Prize money	Starting position	Jockey	Jockey weight	Country	...	TrainerNa
0	03.09.2017	Sha Tin	10	1400	Gress	1310000	6	K C Leung	52	Sverige	...	CH
1	16.09.2017	Sha Tin	10	1400	Gress	1310000	14	C Y Ho	52	Sverige	...	CH
2	14.10.2017	Sha Tin	10	1400	Gress	1310000	8	C Y Ho	52	Sverige	...	CH
3	11.11.2017	Sha Tin	9	1600	Gress	1310000	13	Brett Prebble	54	Sverige	...	CH
4	26.11.2017	Sha Tin	9	1600	Gress	1310000	9	C Y Ho	52	Sverige	...	CH
5	10.12.2017	Sha Tin	1	1800	Gress	1310000	4	C Y Ho	52	Sverige	...	CH
6	01.01.2018	Sha Tin	9	1800	Gress	1310000	9	C Schofield	54	Sverige	...	CH
7	04.02.2018	Sha Tin	5	1800	Gress	1310000	6	Joao Moreira	57	Sverige	...	CH
8	03.03.2018	Sha Tin	8	1800	Gress	1310000	3	C Y Ho	56	Sverige	...	CH
9	11.03.2018	Sha Tin	10	1600	Gress	1310000	8	C Y Ho	57	Sverige	...	CH

10 rows × 21 columns

In [25]:

```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27008 entries, 0 to 27007
Data columns (total 21 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Dato               27008 non-null  object
 1   Track              27008 non-null  object
 2   Race Number        27008 non-null  int64 
 3   Distance           27008 non-null  int64 
 4   Surface            27008 non-null  object
 5   Prize money        27008 non-null  int64 
 6   Starting position  27008 non-null  int64 
 7   Jockey             27008 non-null  object
 8   Jockey weight      27008 non-null  int64 
 9   Country            27008 non-null  object
 10  Horse age          27008 non-null  int64 
 11  TrainerName        27008 non-null  object
 12  Race time          27008 non-null  object
 13  Path               27008 non-null  int64 
 14  Final place        27008 non-null  int64 
 15  FGrating           27008 non-null  int64 
 16  Odds               27008 non-null  object
 17  RaceType           27008 non-null  object
 18  HorseId            27008 non-null  int64 
 19  JockeyId           27008 non-null  int64 
 20  TrainerID          27008 non-null  int64 
dtypes: int64(12), object(9)
memory usage: 4.3+ MB
```

In [26]:

```
a.describe()
```

Out[26]:

	Race Number	Distance	Prize money	Starting position	Jockey weight	Horse age	Path	
count	27008.000000	27008.000000	2.700800e+04	27008.000000	27008.000000	27008.000000	27008.000000	2
mean	5.268624	1401.666173	1.479445e+06	6.741447	55.867373	5.246408	1.678021	
std	2.780088	276.065045	2.162109e+06	3.691071	2.737006	1.519880	1.631784	
min	1.000000	1000.000000	6.600000e+05	1.000000	47.000000	2.000000	0.000000	
25%	3.000000	1200.000000	9.200000e+05	4.000000	54.000000	4.000000	0.000000	
50%	5.000000	1400.000000	9.670000e+05	7.000000	56.000000	5.000000	1.000000	
75%	8.000000	1650.000000	1.450000e+06	10.000000	58.000000	6.000000	3.000000	
max	11.000000	2400.000000	2.800000e+07	14.000000	63.000000	12.000000	11.000000	

In [27]:

```
a.columns
```

Out[27]:

```
Index(['Dato', 'Track', 'Race Number', 'Distance', 'Surface', 'Prize money',
       'Starting position', 'Jockey', 'Jockey weight', 'Country', 'Horse age',
       'TrainerName', 'Race time', 'Path', 'Final place', 'FGrating', 'Odds',
       'RaceType', 'HorseId', 'JockeyId', 'TrainerID'],
      dtype='object')
```

In [28]:

```
c=b[['Race Number', 'Distance', 'Surface', 'Prize money',
       'Starting position']]
c
```

Out[28]:

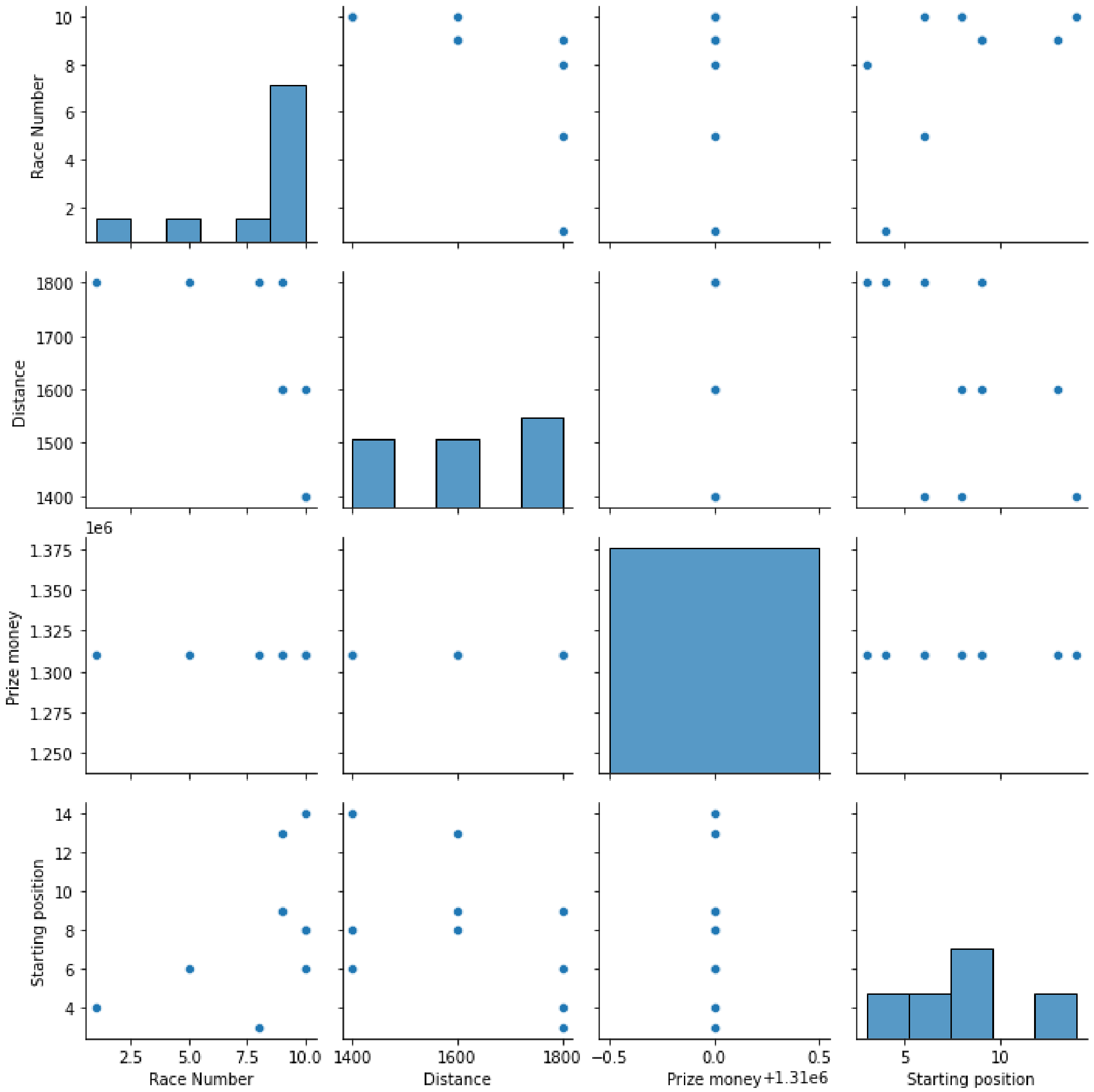
	Race Number	Distance	Surface	Prize money	Starting position
0	10	1400	Gress	1310000	6
1	10	1400	Gress	1310000	14
2	10	1400	Gress	1310000	8
3	9	1600	Gress	1310000	13
4	9	1600	Gress	1310000	9
5	1	1800	Gress	1310000	4
6	9	1800	Gress	1310000	9
7	5	1800	Gress	1310000	6
8	8	1800	Gress	1310000	3
9	10	1600	Gress	1310000	8

In [29]:

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

Out[29]:

<seaborn.axisgrid.PairGrid at 0x2386a9a1d30>



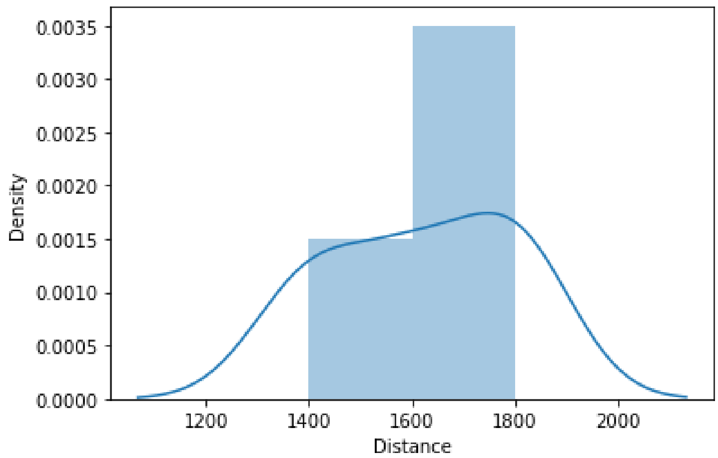
In [30]:

```
sns.distplot(c['Distance'])
```

```
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 flexibili
ty) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[30]:

<AxesSubplot:xlabel='Distance', ylabel='Density'>



In [31]:

```
sns.heatmap(c.corr())
```

Out[31]:

<AxesSubplot:>

In [32]:

```
x=b[['Race Number', 'Distance', 'Prize money',
       'Starting position']]
y=b['Path']
```

In [33]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [34]:

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

Out[34]:

```
LinearRegression()
```

In [35]:

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

```
-12.625878843832108
```

In [36]:

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

Out[36]:

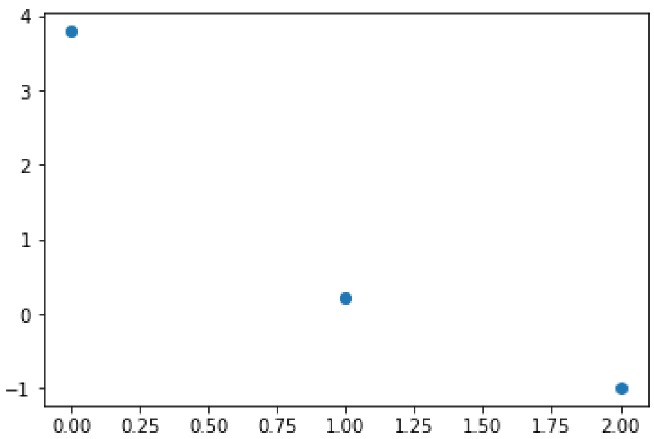
	Co-efficient
Race Number	-0.040835
Distance	0.006498
Prize money	0.000000
Starting position	0.491442

In [37]:

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

Out[37]:

```
<matplotlib.collections.PathCollection at 0x2386b5bc220>
```



In [38]:

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

```
-10.961275690329224
```

In [39]:

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

In [40]:

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

Out[40]:

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

In [41]:

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

Out[41]:

```
-6.7333257426802895
```

In [42]:

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

Out[42]:

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

In [43]:

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

Out[43]:

```
-0.9372285899653976
```

16. DataSet Sleep_Health

In [44]:

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

Out[44]:

	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 [45]:

```
b=a.head(10)
b
```

Out[45]:

	Person ID	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Blood Pressure	Heart Rate	Da Ste
0	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	126/83	77	42
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(

In [46]:

```
a.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 [47]:

```
a.describe()
```

Out[47]:

	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 [48]:

```
a.columns
```

Out[48]:

```
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 [49]:

```
c=b[['Sleep Duration',
       'Quality of Sleep', 'Physical Activity Level', 'Stress Level']]
c
```

Out[49]:

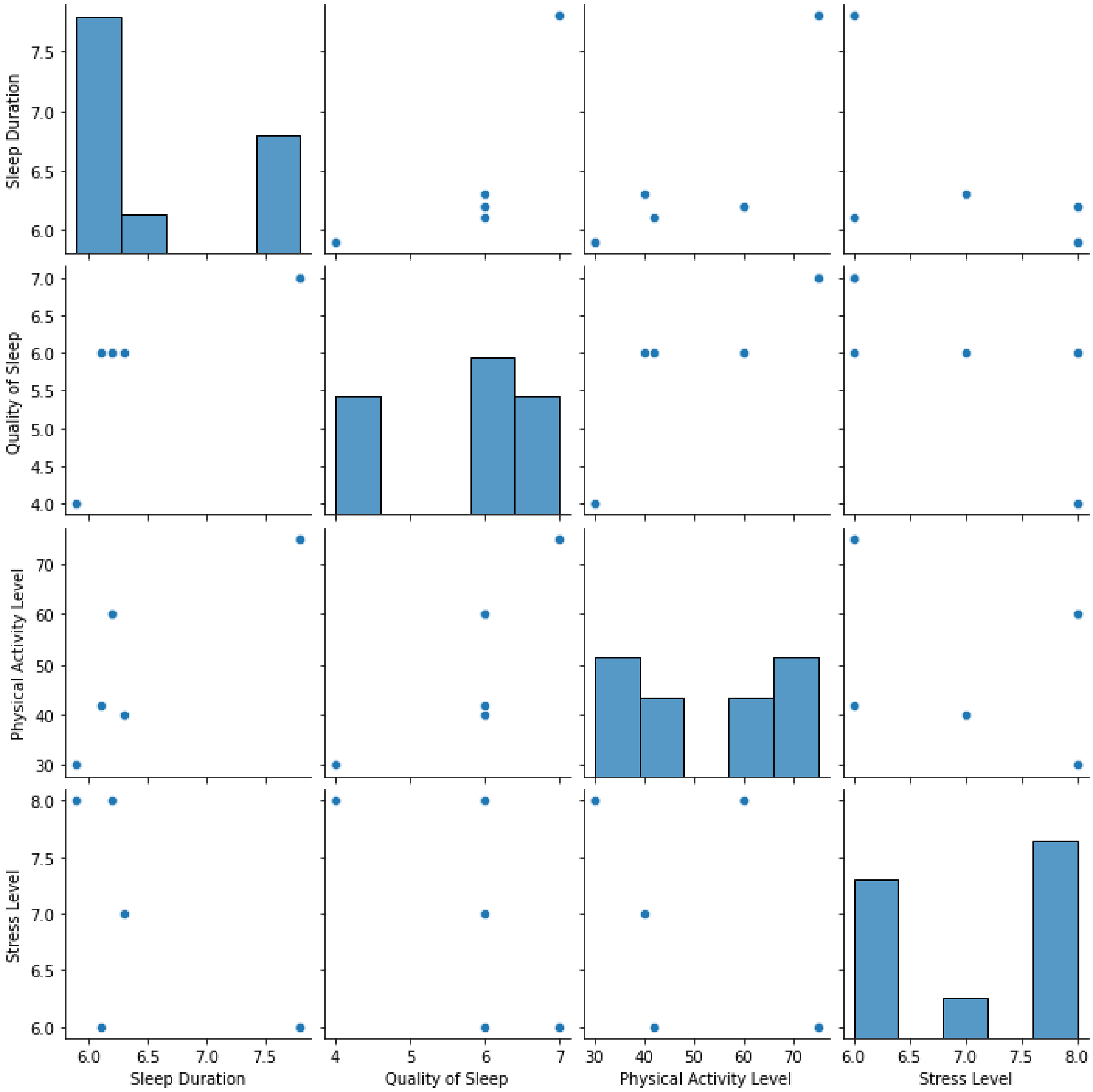
	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level
0	6.1	6	42	6
1	6.2	6	60	8
2	6.2	6	60	8
3	5.9	4	30	8
4	5.9	4	30	8
5	5.9	4	30	8
6	6.3	6	40	7
7	7.8	7	75	6
8	7.8	7	75	6
9	7.8	7	75	6

In [50]:

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

Out[50]:

<seaborn.axisgrid.PairGrid at 0x2386b603d30>



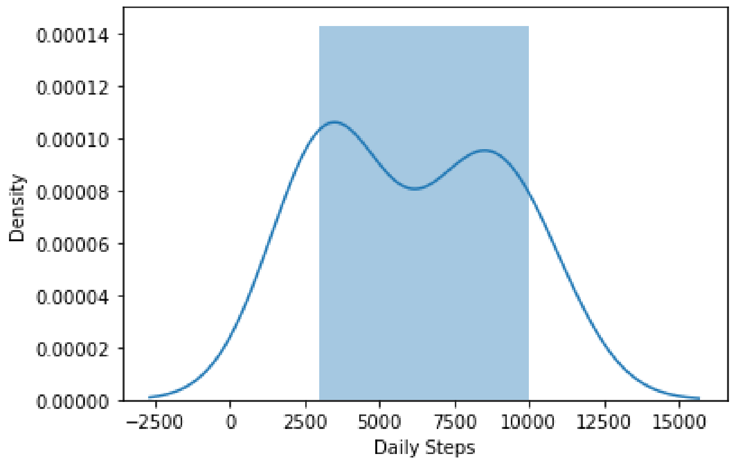
In [51]:

```
sns.distplot(b['Daily Steps'])
```

```
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 flexibili
ty) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[51]:

<AxesSubplot:xlabel='Daily Steps', ylabel='Density'>

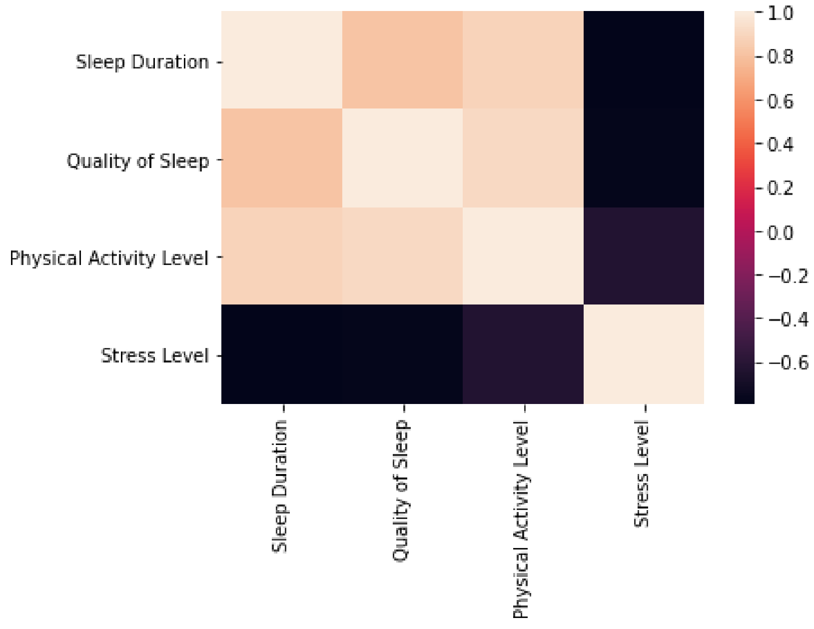


In [52]:

```
sns.heatmap(c.corr())
```

Out[52]:

<AxesSubplot:>



In [53]:

```
x=b[['Age', 'Sleep Duration',
       'Quality of Sleep', 'Physical Activity Level', 'Stress Level']]
y=b['Heart Rate']
```

In [54]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [55]:

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

Out[55]:

```
LinearRegression()
```

In [56]:

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

```
95.04919710143575
```

In [57]:

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

Out[57]:

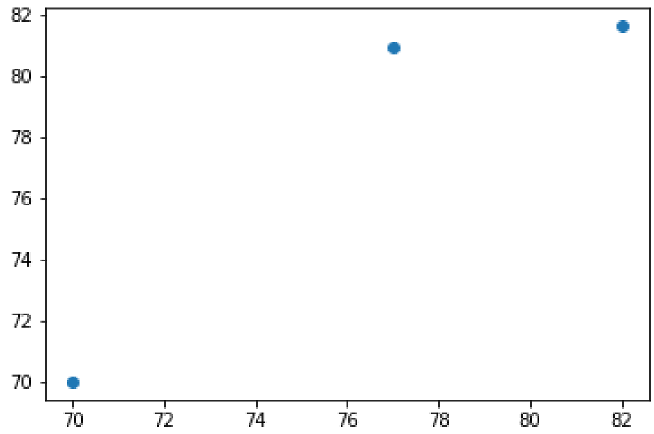
	Co-efficient
Age	0.000677
Sleep Duration	-0.002336
Quality of Sleep	-0.022122
Physical Activity Level	-0.331835
Stress Level	-0.001355

In [58]:

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

Out[58]:

```
<matplotlib.collections.PathCollection at 0x2386c1204c0>
```



In [59]:

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

```
0.7807291681997746
```

In [60]:

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

In [61]:

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

Out[61]:

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

In [62]:

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

Out[62]:

```
0.7828704468307911
```

In [63]:

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

Out[63]:

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

In [64]:

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

Out[64]:

```
0.7955467828863413
```

17. DataSet Student_Marks

In [65]:

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

Out[65]:

	Student_ID	Test_1	Test_2	Test_3	Test_4	Test_5	Test_6	Test_7	Test_8	Test_9	Test_10	Test_11	Tes
0	22000	78	87	91	91	88	98	94	100	100	100	100	
1	22001	79	71	81	72	73	68	59	69	59	60	61	
2	22002	66	65	70	74	78	86	87	96	88	82	90	
3	22003	60	58	54	61	54	57	64	62	72	63	72	
4	22004	99	95	96	93	97	89	92	98	91	98	95	
5	22005	41	36	35	28	35	36	27	26	19	22	27	
6	22006	47	50	47	57	62	64	71	75	85	87	85	
7	22007	84	74	70	68	58	59	56	56	64	70	67	
8	22008	74	64	58	57	53	51	47	45	42	43	34	
9	22009	87	81	73	74	71	63	53	45	39	43	46	
10	22010	40	34	37	33	31	35	39	38	40	48	44	
11	22011	91	84	78	74	76	80	80	73	75	71	79	
12	22012	81	83	93	88	89	90	99	99	95	85	75	
13	22013	52	50	42	38	33	30	28	22	12	20	19	
14	22014	63	67	65	74	80	86	95	96	92	83	75	
15	22015	76	82	88	94	85	76	70	60	50	58	49	
16	22016	83	78	71	71	77	72	66	75	66	61	61	
17	22017	55	45	43	38	43	35	44	37	45	37	45	
18	22018	71	67	76	74	64	61	57	64	61	51	51	
19	22019	62	61	53	49	54	59	68	74	65	55	60	
20	22020	44	38	36	34	26	34	39	44	36	45	35	
21	22021	50	56	53	46	41	38	47	39	44	36	43	
22	22022	57	48	40	45	43	36	26	19	9	12	22	
23	22023	59	56	52	44	50	40	45	46	54	57	52	
24	22024	84	92	89	80	90	80	84	74	68	73	81	
25	22025	74	80	86	87	90	100	95	87	85	79	85	
26	22026	92	84	74	83	93	83	75	82	81	73	70	
27	22027	63	70	74	65	64	55	61	58	48	46	46	
28	22028	78	77	69	76	78	74	67	69	78	68	65	
29	22029	55	58	59	67	71	62	53	61	67	76	75	
30	22030	54	54	48	38	35	45	46	47	41	37	30	
31	22031	84	93	97	89	86	95	100	100	100	99	100	
32	22032	95	100	94	100	98	99	100	90	80	84	75	
33	22033	64	61	63	73	63	68	64	58	50	51	56	
34	22034	76	79	73	77	83	86	95	89	90	95	100	
35	22035	78	71	61	55	54	48	41	32	41	40	48	
36	22036	95	89	91	84	89	94	85	91	100	100	100	
37	22037	99	89	79	87	87	81	82	74	64	54	51	
38	22038	82	83	85	86	89	80	88	95	87	93	90	
39	22039	65	56	64	62	58	51	61	68	70	70	63	
40	22040	100	93	92	86	84	76	82	74	79	72	79	

	Student_ID	Test_1	Test_2	Test_3	Test_4	Test_5	Test_6	Test_7	Test_8	Test_9	Test_10	Test_11	Tes
41	22041	78	72	73	79	81	73	71	77	83	92	97	
42	22042	98	100	100	93	94	92	100	100	98	94	97	
43	22043	58	62	67	77	71	63	64	73	83	76	86	
44	22044	96	92	94	100	99	95	98	92	84	84	84	
45	22045	86	87	85	84	85	91	86	82	85	87	84	
46	22046	48	55	46	40	34	29	37	34	39	41	31	
47	22047	56	52	54	47	40	35	43	44	40	39	47	
48	22048	42	44	46	53	62	59	57	53	43	35	37	
49	22049	64	54	49	59	54	55	57	59	63	73	78	
50	22050	50	44	37	29	37	46	53	57	55	61	64	
51	22051	70	60	70	62	67	67	68	67	72	69	64	
52	22052	63	73	70	63	60	67	61	59	52	58	56	
53	22053	92	100	100	100	100	100	92	87	94	100	94	
54	22054	64	55	54	61	63	57	47	37	44	48	54	
55	22055	60	66	68	58	49	47	39	29	39	44	39	

In [66]:

```
b=a.head(10)
b
```

Out[66]:

	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
0	22000	78	87	91	91	88	98	94	100	100	100	100	
1	22001	79	71	81	72	73	68	59	69	59	60	61	
2	22002	66	65	70	74	78	86	87	96	88	82	90	
3	22003	60	58	54	61	54	57	64	62	72	63	72	
4	22004	99	95	96	93	97	89	92	98	91	98	95	
5	22005	41	36	35	28	35	36	27	26	19	22	27	
6	22006	47	50	47	57	62	64	71	75	85	87	85	
7	22007	84	74	70	68	58	59	56	56	64	70	67	
8	22008	74	64	58	57	53	51	47	45	42	43	34	
9	22009	87	81	73	74	71	63	53	45	39	43	46	

In [67]:

```
a.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 [68]:

```
a.describe()
```

Out[68]:

	Student_ID	Test_1	Test_2	Test_3	Test_4	Test_5	Test_6	Test_7	
count	56.000000	56.000000	56.000000	56.000000	56.000000	56.000000	56.000000	56.000000	5
mean	22027.500000	70.750000	69.196429	68.089286	67.446429	67.303571	66.000000	66.160714	6
std	16.309506	17.009356	17.712266	18.838333	19.807179	20.746890	21.054043	21.427914	2
min	22000.000000	40.000000	34.000000	35.000000	28.000000	26.000000	29.000000	26.000000	1
25%	22013.750000	57.750000	55.750000	53.000000	54.500000	53.750000	50.250000	47.000000	4
50%	22027.500000	70.500000	68.500000	70.000000	71.500000	69.000000	65.500000	64.000000	6
75%	22041.250000	84.000000	83.250000	85.000000	84.000000	85.250000	83.750000	85.250000	8
max	22055.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	10

In [69]:

```
a.columns
```

Out[69]:

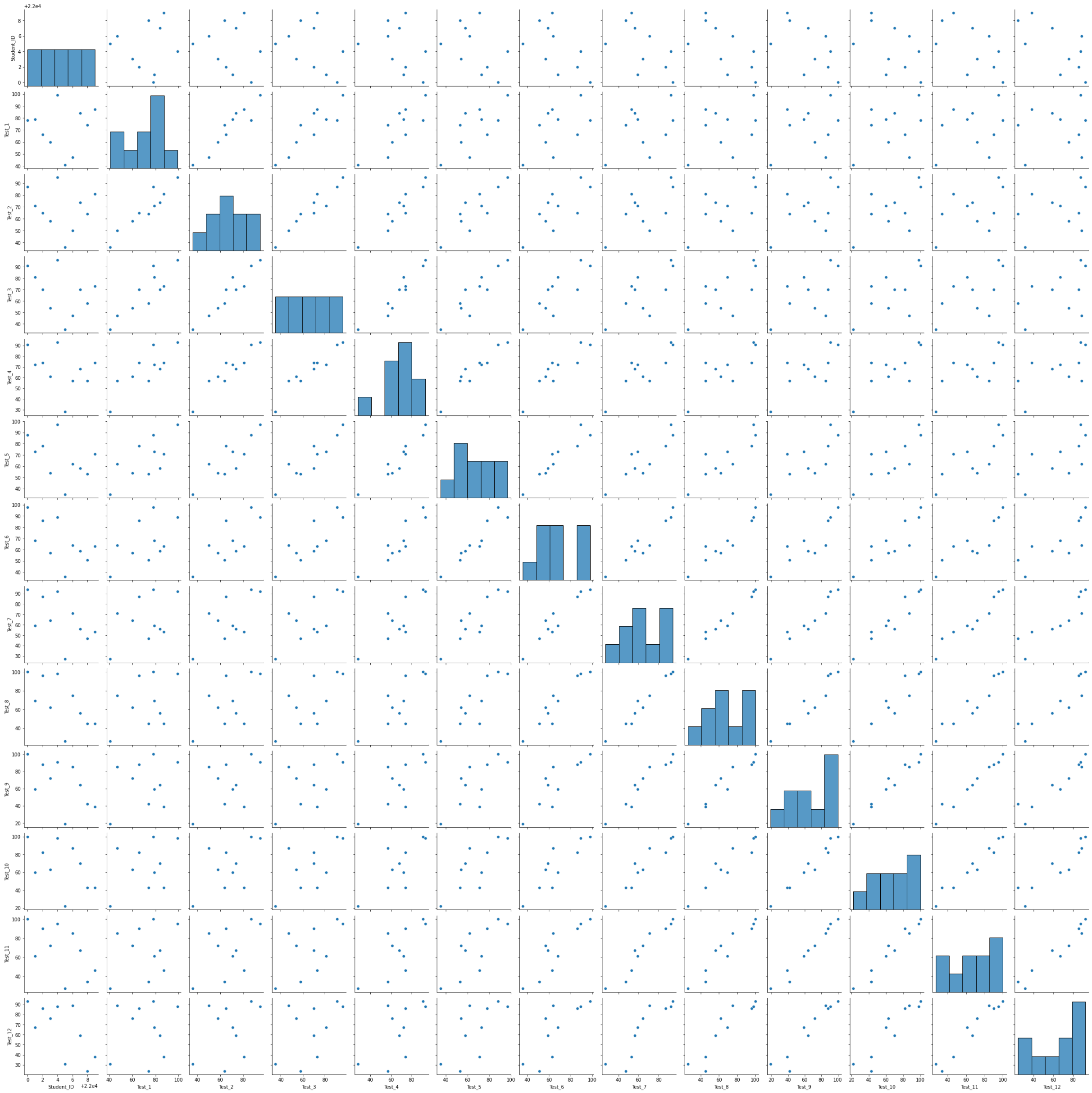
```
Index(['Student_ID', 'Test_1', 'Test_2', 'Test_3', 'Test_4', 'Test_5',
       'Test_6', 'Test_7', 'Test_8', 'Test_9', 'Test_10', 'Test_11',
       'Test_12'],
      dtype='object')
```

In [70]:

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

Out[70]:

<seaborn.axisgrid.PairGrid at 0x23868a866a0>



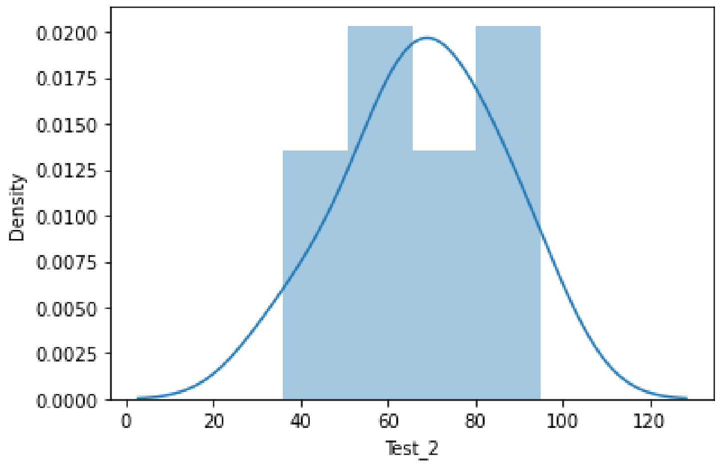
In [71]:

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

```
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 flexibili
ty) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[71]:

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

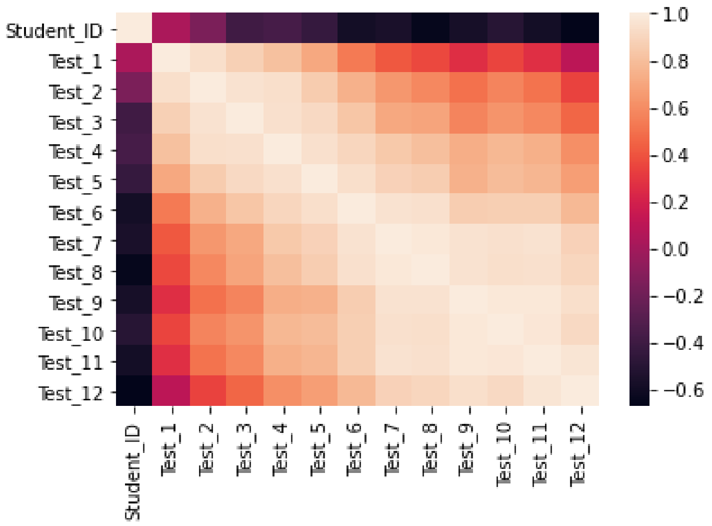


In [72]:

```
sns.heatmap(b.corr())
```

Out[72]:

```
<AxesSubplot:>
```



In [73]:

```
x=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']]
y=b['Test_6']
```

In [74]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [75]:

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

Out[75]:

```
LinearRegression()
```

In [76]:

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

```
-110.29919768327323
```

In [77]:

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

Out[77]:

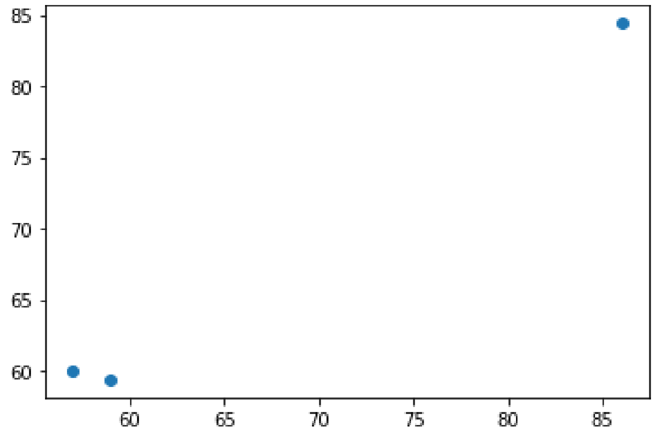
	Co-efficient
Student_ID	0.005353
Test_1	-0.290005
Test_2	0.179604
Test_3	0.221374
Test_4	0.120571
Test_5	-0.037931
Test_6	0.526857
Test_7	0.171807
Test_8	0.042485
Test_9	0.007245
Test_10	-0.105161
Test_11	0.143197
Test_12	-0.073277

In [78]:

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

Out[78]:

```
<matplotlib.collections.PathCollection at 0x238750614c0>
```



In [79]:

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

```
0.9770094119141627
```

In [80]:

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

In [81]:

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

Out[81]:

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

In [82]:

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

Out[82]:

```
0.9623085100430313
```

In [83]:

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

Out[83]:

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

In [84]:

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

Out[84]:

```
0.9956518976162068
```

18. DataSet World_Data

In [85]:

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

Out[85]:

	Country	Density\n(P/Km2)	Abbreviation	Agricultural Land(%)	Land Area(Km2)	Armed Forces size	Birth Rate	Calling Code	Capital/Maj Ci
0	Afghanistan	60	AF	58.10%	652,230	323,000	32.49	93.0	Kab
1	Albania	105	AL	43.10%	28,748	9,000	11.78	355.0	Tirar
2	Algeria	18	DZ	17.40%	2,381,741	317,000	24.28	213.0	Algie
3	Andorra	164	AD	40.00%	468	NaN	7.20	376.0	Andorra Vel
4	Angola	26	AO	47.50%	1,246,700	117,000	40.73	244.0	Luanc
...	...	...	...	...	...	...	...	...	
190	Venezuela	32	VE	24.50%	912,050	343,000	17.88	58.0	Caraca
191	Vietnam	314	VN	39.30%	331,210	522,000	16.75	84.0	Han
192	Yemen	56	YE	44.60%	527,968	40,000	30.45	967.0	Sana
193	Zambia	25	ZM	32.10%	752,618	16,000	36.19	260.0	Lusak
194	Zimbabwe	38	ZW	41.90%	390,757	51,000	30.68	263.0	Harai

195 rows × 35 columns

In [86]:

```
b=a.fillna(value=51)
b
```

Out[86]:

	Country	Density\n(P/Km2)	Abbreviation	Agricultural Land(%)	Land Area(Km2)	Armed Forces size	Birth Rate	Calling Code	Capital/Majo Ci
0	Afghanistan	60	AF	58.10%	652,230	323,000	32.49	93.0	Kab
1	Albania	105	AL	43.10%	28,748	9,000	11.78	355.0	Tirar
2	Algeria	18	DZ	17.40%	2,381,741	317,000	24.28	213.0	Algie
3	Andorra	164	AD	40.00%	468	51	7.20	376.0	Andorra Vel
4	Angola	26	AO	47.50%	1,246,700	117,000	40.73	244.0	Luanc
...	...	...	...	...	...	...	...	...	
190	Venezuela	32	VE	24.50%	912,050	343,000	17.88	58.0	Caraca
191	Vietnam	314	VN	39.30%	331,210	522,000	16.75	84.0	Han
192	Yemen	56	YE	44.60%	527,968	40,000	30.45	967.0	Sana
193	Zambia	25	ZM	32.10%	752,618	16,000	36.19	260.0	Lusak
194	Zimbabwe	38	ZW	41.90%	390,757	51,000	30.68	263.0	Harai

195 rows × 35 columns

In [87]:

```
c=b.head(10)
c
```

Out[87]:

	Country	Density\n(P/Km2)	Abbreviation	Agricultural Land(%)	Land Area(Km2)	Armed Forces size	Birth Rate	Calling Code	Capital/Major City
0	Afghanistan	60	AF	58.10%	652,230	323,000	32.49	93.0	Kabul
1	Albania	105	AL	43.10%	28,748	9,000	11.78	355.0	Tirana
2	Algeria	18	DZ	17.40%	2,381,741	317,000	24.28	213.0	Algiers
3	Andorra	164	AD	40.00%	468	51	7.20	376.0	Andorra la Vella
4	Angola	26	AO	47.50%	1,246,700	117,000	40.73	244.0	Luanda
5	Antigua and Barbuda	223	AG	20.50%	443	0	15.33	1.0	St. John's, Saint John
6	Argentina	17	AR	54.30%	2,780,400	105,000	17.02	54.0	Buenos Aires
7	Armenia	104	AM	58.90%	29,743	49,000	13.99	374.0	Yerevan
8	Australia	3	AU	48.20%	7,741,220	58,000	12.60	61.0	Canberra
9	Austria	109	AT	32.40%	83,871	21,000	9.70	43.0	Vienna

10 rows × 35 columns

In [88]:

```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 195 entries, 0 to 194
Data columns (total 35 columns):
 #   Column                                   Non-Null Count  Dtype  
---  ------                                   --------------  -----  
 0   Country                                  195 non-null    object 
 1   Density
(P/Km2)                          195 non-null    object 
 2   Abbreviation                             188 non-null    object 
 3   Agricultural Land( %)                    188 non-null    object 
 4   Land Area(Km2)                           194 non-null    object 
 5   Armed Forces size                        171 non-null    object 
 6   Birth Rate                               189 non-null    float64
 7   Calling Code                             194 non-null    float64
 8   Capital/Major City                       192 non-null    object 
 9   Co2-Emissions                            188 non-null    object 
 10  CPI                                      178 non-null    object 
 11  CPI Change (%)                           179 non-null    object 
 12  Currency-Code                            180 non-null    object 
 13  Fertility Rate                           188 non-null    float64
 14  Forested Area (%)                        188 non-null    object 
 15  Gasoline Price                           175 non-null    object 
 16  GDP                                      193 non-null    object 
 17  Gross primary education enrollment (%)   188 non-null    object 
 18  Gross tertiary education enrollment (%)  183 non-null    object 
 19  Infant mortality                         189 non-null    float64
 20  Largest city                             189 non-null    object 
 21  Life expectancy                          187 non-null    float64
 22  Maternal mortality ratio                 181 non-null    float64
 23  Minimum wage                             150 non-null    object 
 24  Official language                        194 non-null    object 
 25  Out of pocket health expenditure         188 non-null    object 
 26  Physicians per thousand                  188 non-null    float64
 27  Population                               194 non-null    object 
 28  Population: Labor force participation (%)  176 non-null    object 
 29  Tax revenue (%)                          169 non-null    object 
 30  Total tax rate                           183 non-null    object 
 31  Unemployment rate                        176 non-null    object 
 32  Urban_population                         190 non-null    object 
 33  Latitude                                 194 non-null    float64
 34  Longitude                                194 non-null    float64
dtypes: float64(9), object(26)
memory usage: 53.4+ KB
```

In [89]:

```
a.describe()
```

Out[89]:

	Birth Rate	Calling Code	Fertility Rate	Infant mortality	Life expectancy	Maternal mortality ratio	Physicians per thousand	Latitude	
count	189.000000	194.000000	188.000000	189.000000	187.000000	181.000000	188.000000	194.000000	1
mean	20.214974	360.546392	2.698138	21.332804	72.279679	160.392265	1.839840	19.092351	
std	9.945774	323.236419	1.282267	19.548058	7.483661	233.502024	1.684261	23.961779	
min	5.900000	1.000000	0.980000	1.400000	52.800000	2.000000	0.010000	-40.900557	-1
25%	11.300000	82.500000	1.705000	6.000000	67.000000	13.000000	0.332500	4.544175	
50%	17.950000	255.500000	2.245000	14.000000	73.200000	53.000000	1.460000	17.273849	
75%	28.750000	506.750000	3.597500	32.700000	77.500000	186.000000	2.935000	40.124603	
max	46.080000	1876.000000	6.910000	84.500000	85.400000	1150.000000	8.420000	64.963051	1

In [90]:

```
c.columns
```

Out[90]:

```
Index(['Country', 'Density\n(P/Km2)', 'Abbreviation', 'Agricultural Land( %)',
       'Land Area(Km2)', 'Armed Forces size', 'Birth Rate', 'Calling Code',
       'Capital/Major City', 'Co2-Emissions', 'CPI', 'CPI Change (%)',
       'Currency-Code', 'Fertility Rate', 'Forested Area (%)',
       'Gasoline Price', 'GDP', 'Gross primary education enrollment (%)',
       'Gross tertiary education enrollment (%)', 'Infant mortality',
       'Largest city', 'Life expectancy', 'Maternal mortality ratio',
       'Minimum wage', 'Official language', 'Out of pocket health expenditure',
       'Physicians per thousand', 'Population',
       'Population: Labor force participation (%)', 'Tax revenue (%)',
       'Total tax rate', 'Unemployment rate', 'Urban_population', 'Latitude',
       'Longitude'],
      dtype='object')
```

In [91]:

```
d=c[['Density\n(P/Km2)','Urban_population', 'Latitude',
       'Longitude']]
d
```

Out[91]:

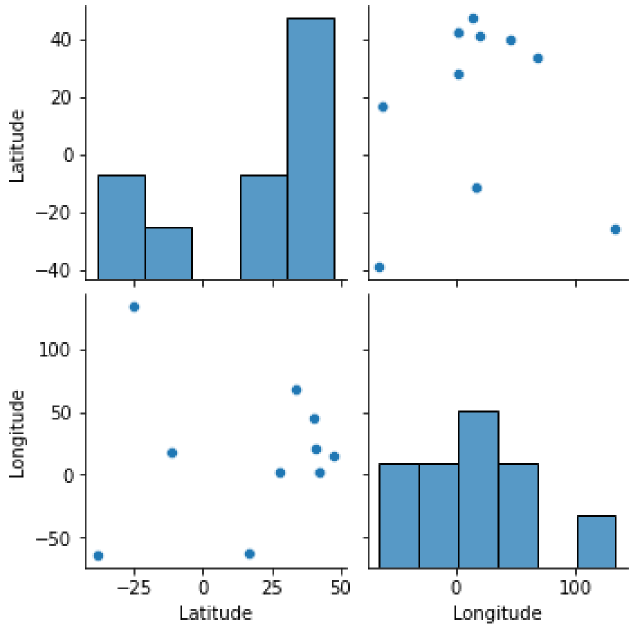
	Density\n(P/Km2)	Urban_population	Latitude	Longitude
0	60	9,797,273	33.939110	67.709953
1	105	1,747,593	41.153332	20.168331
2	18	31,510,100	28.033886	1.659626
3	164	67,873	42.506285	1.521801
4	26	21,061,025	-11.202692	17.873887
5	223	23,800	17.060816	-61.796428
6	17	41,339,571	-38.416097	-63.616672
7	104	1,869,848	40.069099	45.038189
8	3	21,844,756	-25.274398	133.775136
9	109	5,194,416	47.516231	14.550072

In [92]:

```
sns.pairplot(d)
```

Out[92]:

<seaborn.axisgrid.PairGrid at 0x238764c8580>



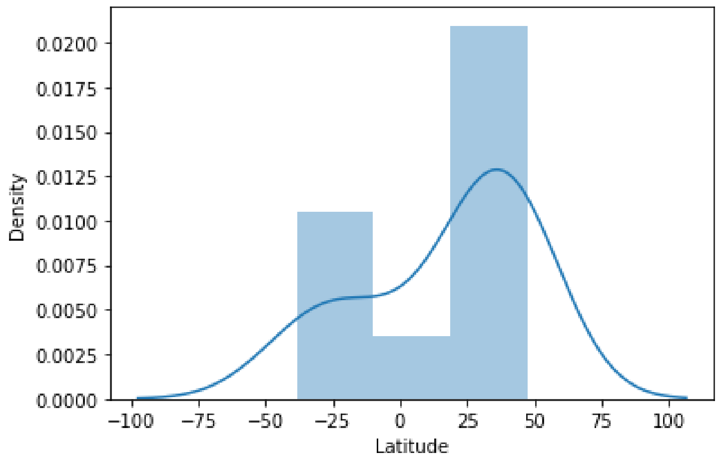
In [93]:

```
sns.distplot(d['Latitude'])
```

```
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 flexibili
ty) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[93]:

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

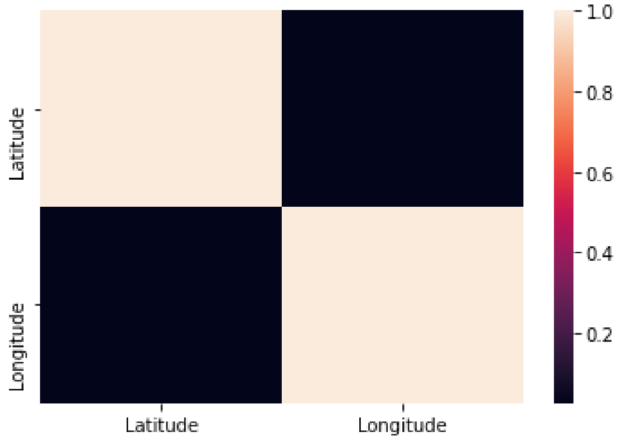


In [94]:

```
sns.heatmap(d.corr())
```

Out[94]:

```
<AxesSubplot:>
```



In [95]:

```
x=d[['Density\n(P/Km2)','Latitude',
       'Longitude']]
y=d['Latitude']
```

In [96]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [97]:

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

Out[97]:

```
LinearRegression()
```

In [98]:

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

```
1.0658141036401503e-14
```

In [99]:

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

Out[99]:

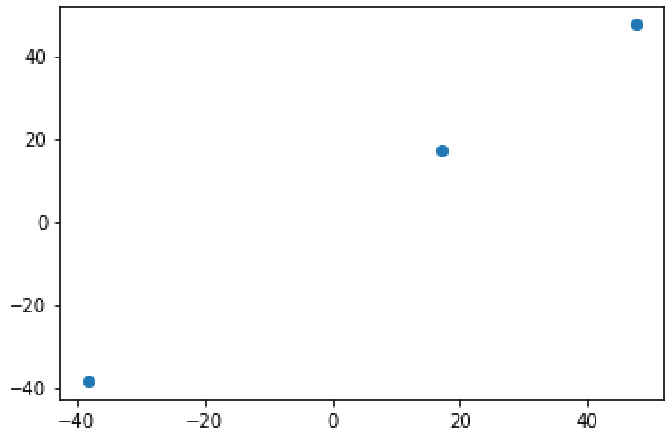
	Co-efficient
Density\n(P/Km2)	-2.982559e-16
Latitude	1.000000e+00
Longitude	-1.825955e-17

In [100]:

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

Out[100]:

```
<matplotlib.collections.PathCollection at 0x238767ac550>
```



In [101]:

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

```
1.0
```

In [102]:

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

In [103]:

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

Out[103]:

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

In [104]:

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

Out[104]:

```
0.9999126263897781
```

In [105]:

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

Out[105]:

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

In [106]:

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

Out[106]:

```
0.9994308979728221
```

In [ ]:

```

```