In [ ]:

```
```

In [130]:

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

In [131]:

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

Out[131]:

	Student_ID	Test_1	Test_2	Test_3	Test_4	Test_5	Test_6	Test_7	Test_8	Test_9	Test_1
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	2
6	22006	47	50	47	57	62	64	71	75	85	(
7	22007	84	74	70	68	58	59	56	56	64	7
8	22008	74	64	58	57	53	51	47	45	42	4
9	22009	87	81	73	74	71	63	53	45	39	4
10	22010	40	34	37	33	31	35	39	38	40	4
11	22011	91	84	78	74	76	80	80	73	75	7
12	22012	81	83	93	88	89	90	99	99	95	(
13	22013	52	50	42	38	33	30	28	22	12	2
14	22014	63	67	65	74	80	86	95	96	92	(
15	22015	76	82	88	94	85	76	70	60	50	5
16	22016	83	78	71	71	77	72	66	75	66	(
17	22017	55	45	43	38	43	35	44	37	45	3
18	22018	71	67	76	74	64	61	57	64	61	5
19	22019	62	61	53	49	54	59	68	74	65	5
20	22020	44	38	36	34	26	34	39	44	36	4
21	22021	50	56	53	46	41	38	47	39	44	3
22	22022	57	48	40	45	43	36	26	19	9	1
23	22023	59	56	52	44	50	40	45	46	54	5
24	22024	84	92	89	80	90	80	84	74	68	7
25	22025	74	80	86	87	90	100	95	87	85	7
26	22026	92	84	74	83	93	83	75	82	81	7
27	22027	63	70	74	65	64	55	61	58	48	4
28	22028	78	77	69	76	78	74	67	69	78	(
29	22029	55	58	59	67	71	62	53	61	67	7
30	22030	54	54	48	38	35	45	46	47	41	3
31	22031	84	93	97	89	86	95	100	100	100	9
32	22032	95	100	94	100	98	99	100	90	80	8
33	22033	64	61	63	73	63	68	64	58	50	5
34	22034	76	79	73	77	83	86	95	89	90	9
35	22035	78	71	61	55	54	48	41	32	41	4
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_1
37	22037	99	89	79	87	87	81	82	74	64	5
38	22038	82	83	85	86	89	80	88	95	87	9
39	22039	65	56	64	62	58	51	61	68	70	7
40	22040	100	93	92	86	84	76	82	74	79	7
41	22041	78	72	73	79	81	73	71	77	83	9
42	22042	98	100	100	93	94	92	100	100	98	9
43	22043	58	62	67	77	71	63	64	73	83	7
44	22044	96	92	94	100	99	95	98	92	84	8
45	22045	86	87	85	84	85	91	86	82	85	8
46	22046	48	55	46	40	34	29	37	34	39	4
47	22047	56	52	54	47	40	35	43	44	40	3
48	22048	42	44	46	53	62	59	57	53	43	3
49	22049	64	54	49	59	54	55	57	59	63	7
50	22050	50	44	37	29	37	46	53	57	55	6
51	22051	70	60	70	62	67	67	68	67	72	6
52	22052	63	73	70	63	60	67	61	59	52	5
53	22053	92	100	100	100	100	100	92	87	94	10
54	22054	64	55	54	61	63	57	47	37	44	4
55	22055	60	66	68	58	49	47	39	29	39	4

In [132]:

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

Out[132]:

	Student_ID	Test_1	Test_2	Test_3	Test_4	Test_5	Test_6	Test_7	Test_8	Test_9	Test_10
0	22000	78	87	91	91	88	98	94	100	100	100
1	22001	79	71	81	72	73	68	59	69	59	60
2	22002	66	65	70	74	78	86	87	96	88	82
3	22003	60	58	54	61	54	57	64	62	72	63
4	22004	99	95	96	93	97	89	92	98	91	98
5	22005	41	36	35	28	35	36	27	26	19	22
6	22006	47	50	47	57	62	64	71	75	85	87
7	22007	84	74	70	68	58	59	56	56	64	70
8	22008	74	64	58	57	53	51	47	45	42	43
9	22009	87	81	73	74	71	63	53	45	39	43

In [133]:

```
# to find
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 13 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Student_ID  10 non-null     int64
 1   Test_1      10 non-null     int64
 2   Test_2      10 non-null     int64
 3   Test_3      10 non-null     int64
 4   Test_4      10 non-null     int64
 5   Test_5      10 non-null     int64
 6   Test_6      10 non-null     int64
 7   Test_7      10 non-null     int64
 8   Test_8      10 non-null     int64
 9   Test_9      10 non-null     int64
 10  Test_10     10 non-null     int64
 11  Test_11     10 non-null     int64
 12  Test_12     10 non-null     int64
dtypes: int64(13)
memory usage: 1.1 KB
```

In [134]:

```
# to display summary of statastic
a.describe()
```

Out[134]:

	Student_ID	Test_1	Test_2	Test_3	Test_4	Test_5	Test_6	Test_7
count	10.00000	10.000000	10.000000	10.000000	10.00000	10.00000	10.00000	10.000000
mean	22004.50000	71.500000	68.100000	67.500000	67.50000	66.90000	67.10000	65.000000
std	3.02765	18.106168	17.565433	19.259918	18.65029	18.28448	18.89415	21.395742
min	22000.00000	41.000000	36.000000	35.000000	28.00000	35.00000	36.00000	27.000000
25%	22002.25000	61.500000	59.500000	55.000000	58.00000	55.00000	57.50000	53.750000
50%	22004.50000	76.000000	68.000000	70.000000	70.00000	66.50000	63.50000	61.500000
75%	22006.75000	82.750000	79.250000	79.000000	74.00000	76.75000	81.50000	83.000000
max	22009.00000	99.000000	95.000000	96.000000	93.00000	97.00000	98.00000	94.000000

In [135]:

```
# to display colum heading
a.columns
```

Out[135]:

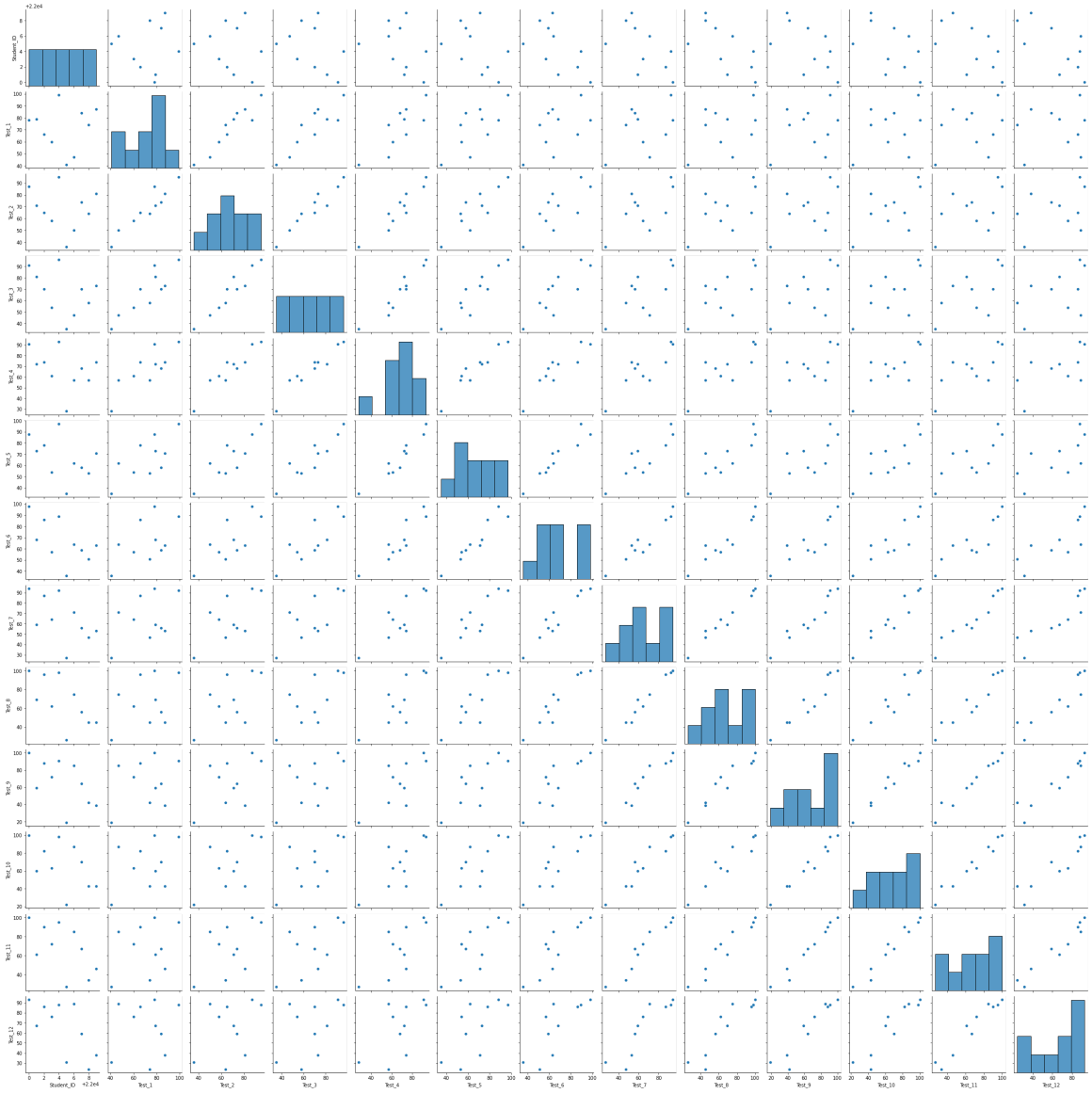
```
Index(['Student_ID', 'Test_1', 'Test_2', 'Test_3', 'Test_4', 'Test_5',
       'Test_6', 'Test_7', 'Test_8', 'Test_9', 'Test_10', 'Test_11',
       'Test_12'],
      dtype='object')
```

In [136]:

```
sns.pairplot(a)
```

Out[136]:

```
<seaborn.axisgrid.PairGrid at 0x198c57c9e50>
```

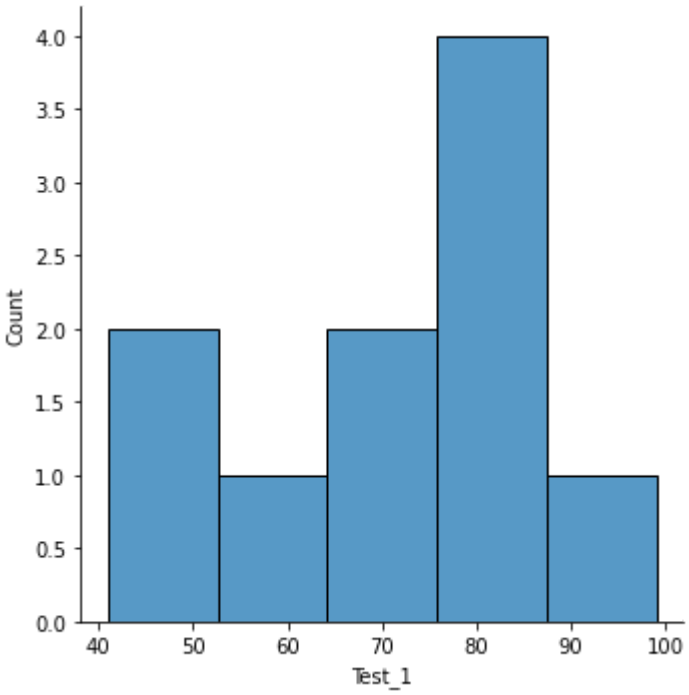


In [137]:

```
sns.displot(a["Test_1"])
```

Out[137]:

```
<seaborn.axisgrid.FacetGrid at 0x198cc1b3fd0>
```



In [140]:

```
b=a[['Test_1', 'Test_2', 'Test_3', 'Test_4', 'Test_5',
       'Test_6', 'Test_7', 'Test_8' ]]
b
```

Out[140]:

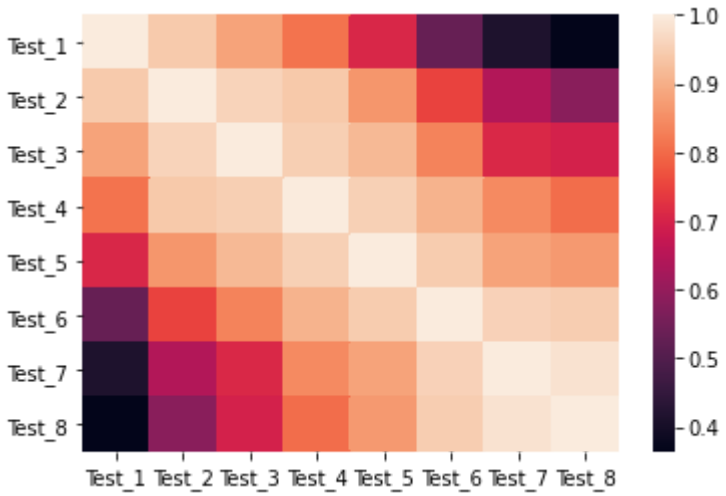
	Test_1	Test_2	Test_3	Test_4	Test_5	Test_6	Test_7	Test_8
0	78	87	91	91	88	98	94	100
1	79	71	81	72	73	68	59	69
2	66	65	70	74	78	86	87	96
3	60	58	54	61	54	57	64	62
4	99	95	96	93	97	89	92	98
5	41	36	35	28	35	36	27	26
6	47	50	47	57	62	64	71	75
7	84	74	70	68	58	59	56	56
8	74	64	58	57	53	51	47	45
9	87	81	73	74	71	63	53	45

In [141]:

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

Out[141]:

<AxesSubplot:>



In [143]:

```
x=a[['Test_1', 'Test_2', 'Test_3', 'Test_4', 'Test_5',
       'Test_6', 'Test_7', 'Test_8']]
y=a['Test_1']
```

In [144]:

```
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 [145]:

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[145]:

```
LinearRegression()
```

In [146]:

```
lr.intercept_
```

Out[146]:

```
0.6587122399825773
```

In [147]:

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

Out[147]:

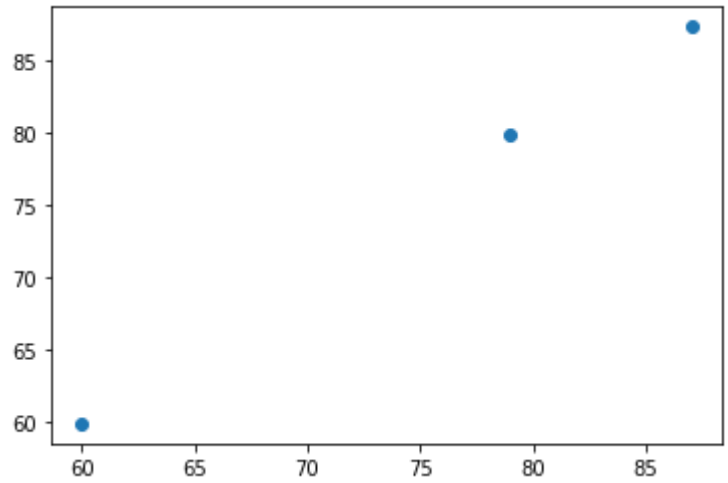
	Co-efficient
Test_1	0.985917
Test_2	-0.037144
Test_3	0.003220
Test_4	0.073824
Test_5	0.023451
Test_6	-0.014371
Test_7	-0.073683
Test_8	0.029311

In [148]:

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

Out[148]:

```
<matplotlib.collections.PathCollection at 0x198ce7c82b0>
```



In [149]:

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

Out[149]:

```
0.9976551732585991
```

In [150]:

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

Out[150]:

```
1.0
```

In [151]:

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

In [152]:

```
rr=Ridge(alpha=10)
rr.fit(x_test,y_test)
```

Out[152]:

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

In [153]:

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

Out[153]:

```
0.9999408830326352
```

In [154]:

```
la=Lasso(alpha=10)
la.fit(x_test,y_test)
```

Out[154]:

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

In [155]:

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

Out[155]:

```
0.9939176220755777
```

In [156]:

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

Out[156]:

```
ElasticNet()
```

In [157]:

```
en.coef_
```

Out[157]:

```
array([ 9.87008121e-01,  1.12226230e-02,  0.00000000e+00,  0.00000000e+00,
       -0.00000000e+00, -0.00000000e+00, -0.00000000e+00, -3.81198500e-0
4])
```

In [158]:

```
en.intercept_
```

Out[158]:

```
0.17946400214847813
```

In [159]:

```
prediction=en.predict(x_test)
prediction
```

Out[159]:

```
array([86.94104903, 78.92360907, 60.02722907])
```

In [160]:

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

Out[160]:

```
0.9999738677307175
```

EVALUATION METRICS

In [161]:

```
from sklearn import metrics
```

In [162]:

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

```
Mean Absolute Error: 0.054190322287001415
```

In [163]:

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

```
Mean Squared Error 0.0033507376391063363
```

In [164]:

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

```
Root Mean Squared Error 0.057885556394547476
```

In [ ]:

```

```