Problem Statement

Linear Regression

Import Libraries

In [1]:

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

In [2]:

```
a=pd.read_csv("student.csv")
a
```

Out[2]:

	Student_ID	Test_1	Test_2	Test_3	Test_4	Test_5	Test_6	Test_7	Test_8	Test_9	Test_10	Test_11	T
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	

To display top 10 rows

In [3]:

```
c=a.head(15)
c
```

Out[3]:

	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	

To find Missing values

In [4]:

```
c.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15 entries, 0 to 14
Data columns (total 13 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Student_ID  15 non-null     int64
 1   Test_1      15 non-null     int64
 2   Test_2      15 non-null     int64
 3   Test_3      15 non-null     int64
 4   Test_4      15 non-null     int64
 5   Test_5      15 non-null     int64
 6   Test_6      15 non-null     int64
 7   Test_7      15 non-null     int64
 8   Test_8      15 non-null     int64
 9   Test_9      15 non-null     int64
 10  Test_10     15 non-null     int64
 11  Test_11     15 non-null     int64
 12  Test_12     15 non-null     int64
dtypes: int64(13)
memory usage: 1.6 KB
```

To display summary of statistics

In [5]:

```
a.describe()
```

Out[5]:

	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

To display column heading

In [6]:

```
a.columns
```

Out[6]:

```
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')
```

Pairplot

In [7]:

```
s=a.dropna(axis=1)
s
```

Out[7]:

	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 [8]:

```
s.columns
```

Out[8]:

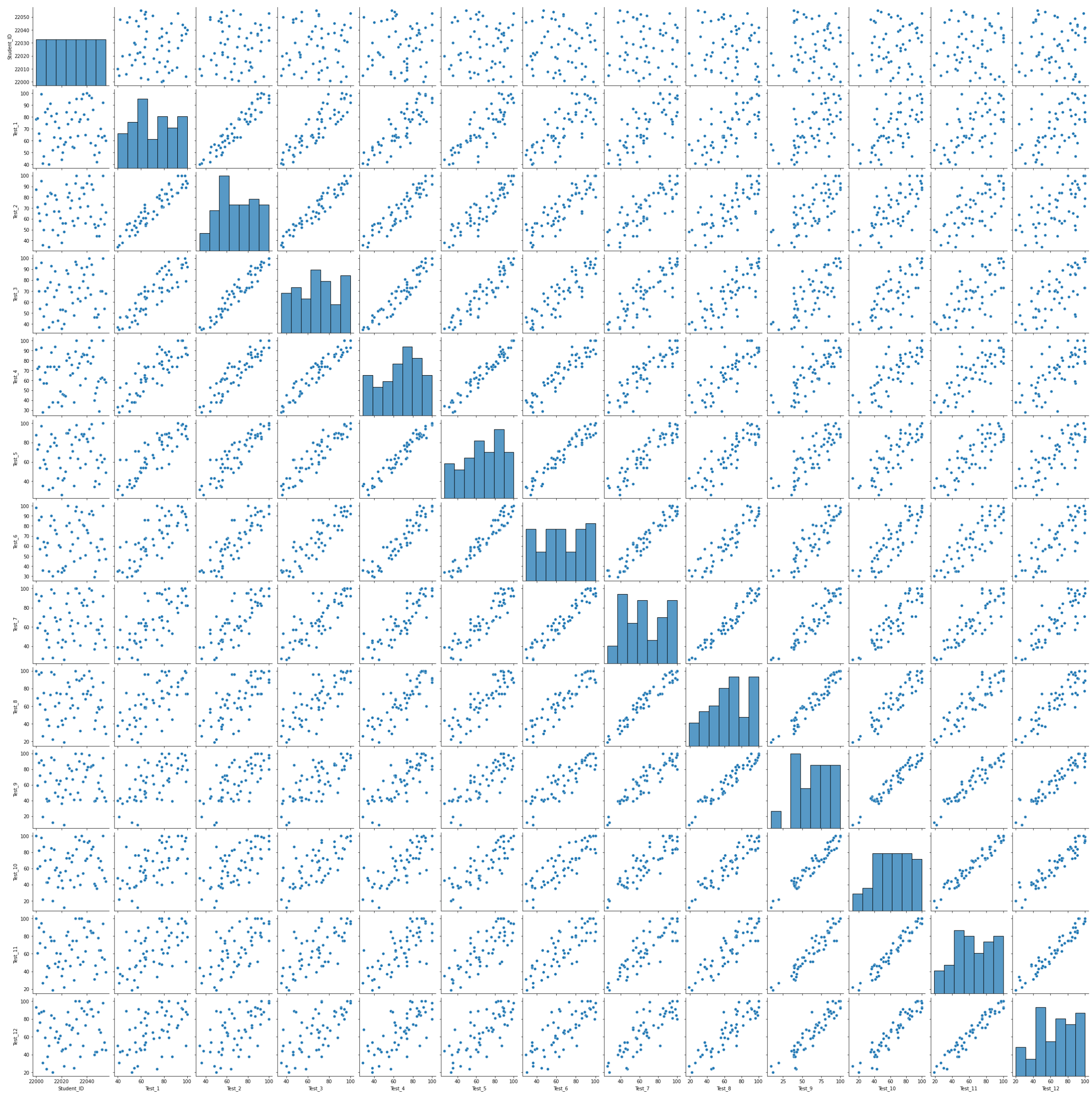
```
Index(['Student_ID', 'Test_1', 'Test_2', 'Test_3', 'Test_4', 'Test_5',
       'Test_6', 'Test_7', 'Test_8', 'Test_9', 'Test_10', 'Test_11',
       'Test_12'],
      dtype='object')
```

In [9]:

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

Out[9]:

```
<seaborn.axisgrid.PairGrid at 0x281bcbbc820>
```



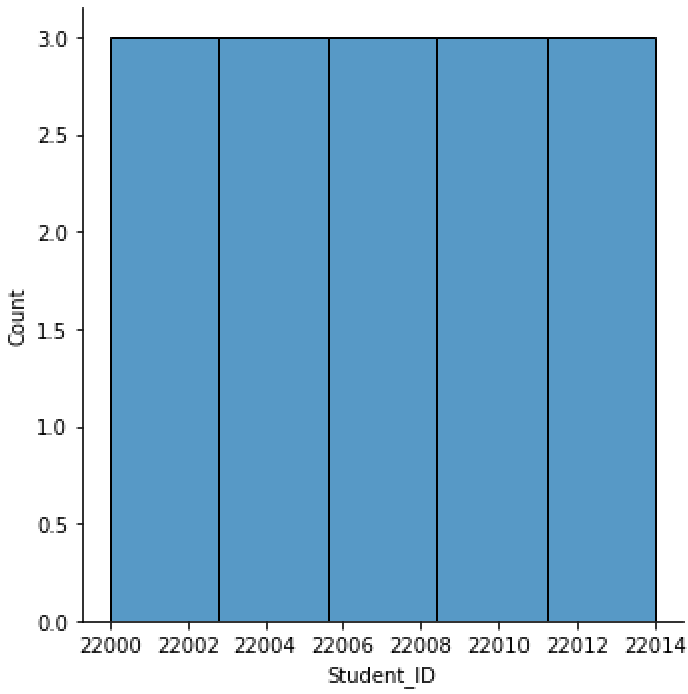
Distribution Plot

In [10]:

```
sns.displot(c['Student_ID'])
```

Out[10]:

```
<seaborn.axisgrid.FacetGrid at 0x281c3046e80>
```



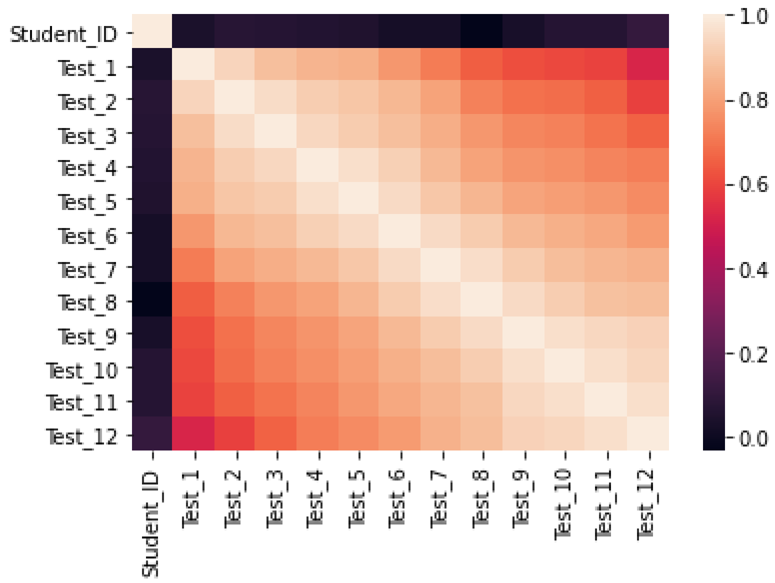
Correlation

In [11]:

```
b=a[['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']]
sns.heatmap(b.corr())
```

Out[11]:

<AxesSubplot:>



Train the model - Model Building

In [12]:

```
g=c[['Student_ID', 'Test_1', 'Test_2', 'Test_3', 'Test_4', 'Test_5',
       'Test_6', 'Test_7', 'Test_8', 'Test_9', 'Test_10', 'Test_11']]
h=c['Test_12']
```

To split dataset into training end test

In [13]:

```
from sklearn.model_selection import train_test_split
g_train,g_test,h_train,h_test=train_test_split(g,h,test_size=0.6)
```

To run the model

In [14]:

```
from sklearn.linear_model import LinearRegression
```

In [15]:

```
lr=LinearRegression()
lr.fit(g_train,h_train)
```

Out[15]:

```
LinearRegression()
```

In [16]:

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

```
4061.167388674988
```

Coeffecient

In [17]:

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

Out[17]:

	Co-effecient
Student_ID	-0.184322
Test_1	-0.482841
Test_2	-0.019378
Test_3	-0.111938
Test_4	0.439475
Test_5	0.111839
Test_6	0.010664
Test_7	0.043803
Test_8	-0.288414
Test_9	0.190001
Test_10	-0.070461
Test_11	1.097065

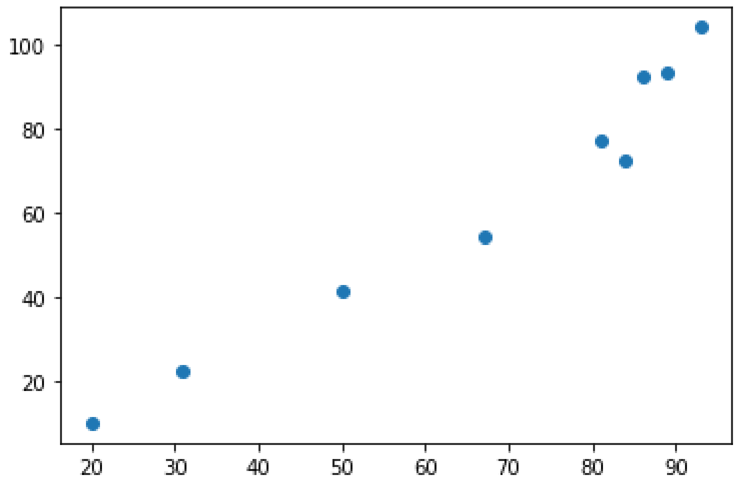
Best Fit line

In [18]:

```
prediction=lr.predict(g_test)
plt.scatter(h_test,prediction)
```

Out[18]:

```
<matplotlib.collections.PathCollection at 0x281c6af9850>
```



To find score

In [19]:

```
print(lr.score(g_test,h_test))
```

```
0.8737248964903332
```

Import Lasso and ridge

In [20]:

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

Ridge

In [21]:

```
ri=Ridge(alpha=5)
ri.fit(g_train,h_train)
```

Out[21]:

```
Ridge(alpha=5)
```

In [22]:

```
ri.score(g_test,h_test)
```

Out[22]:

```
0.8778095893429323
```

In [23]:

```
ri.score(g_train,h_train)
```

Out[23]:

```
0.9999074184901608
```

Lasso

In [24]:

```
l=Lasso(alpha=6)
l.fit(g_train,h_train)
```

Out[24]:

```
Lasso(alpha=6)
```

In [25]:

```
l.score(g_test,h_test)
```

Out[25]:

```
0.9198910720370842
```

In [26]:

```
ri.score(g_train,h_train)
```

Out[26]:

```
0.9999074184901608
```

ElasticNet

In [27]:

```
from sklearn.linear_model import ElasticNet
e=ElasticNet()
e.fit(g_train,h_train)
```

Out[27]:

```
ElasticNet()
```

Coeffecient,intercept

In [28]:

```
print(e.coef_)
```

```
[-0.         -0.54909831 -0.         -0.          0.35804753  0.
 -0.         -0.         -0.0608745  -0.         -0.05845723  1.15973189]
```

In [29]:

```
print(e.intercept_)
```

```
10.649263197957694
```

Evaluation

In [30]:

```
d=e.predict(g_test)
d
```

Out[30]:

```
array([ 15.22847945,  78.8355757 ,  26.60552813,  56.08578934,
        94.64271632,  46.40992402,  94.17619514, 104.44193528,
        73.66493361])
```

In [31]:

```
print(e.score(g_test,h_test))
```

```
0.9106839859688632
```

Evaluation

In [32]:

```
from sklearn import metrics
print("Mean Absolute error:",metrics.mean_absolute_error(h_test,d))
```

```
Mean Absolute error: 6.825624053034491
```

In [33]:

```
print("Mean Squared error:",metrics.mean_squared_error(h_test,d))
```

```
Mean Squared error: 57.55479837492857
```

In [34]:

```
print("Mean Squared error:",np.sqrt(metrics.mean_squared_error(h_test,d)))
```

```
Mean Squared error: 7.58648788141974
```

In [ ]:

```

```