Problem statement

predicting the house price in USA.To create a model to help him estimate of what the house would sell for.

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

In [2]:
```
df=pd.read_csv("student_marks")
```

To display top 10 rows

In [3]:
```
df.head(10)
```

Out[3]:

	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
0	22000	78	87	91	91	88	98	94	100	100	100	100	93
1	22001	79	71	81	72	73	68	59	69	59	60	61	67
2	22002	66	65	70	74	78	86	87	96	88	82	90	86
3	22003	60	58	54	61	54	57	64	62	72	63	72	76
4	22004	99	95	96	93	97	89	92	98	91	98	95	88
5	22005	41	36	35	28	35	36	27	26	19	22	27	31
6	22006	47	50	47	57	62	64	71	75	85	87	85	89
7	22007	84	74	70	68	58	59	56	56	64	70	67	59
8	22008	74	64	58	57	53	51	47	45	42	43	34	24
9	22009	87	81	73	74	71	63	53	45	39	43	46	38

Data Cleaning And Pre-Processing

In [4]:
```
1 df.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 [5]:
```
# Display the statistical summary
df.describe()
```

Out[5]:

	Student_ID	Test_1	Test_2	Test_3	Test_4	Test_5	Test_6	Test_7	Test_8	Test_9	T
count	56.000000	56.000000	56.000000	56.000000	56.000000	56.000000	56.000000	56.000000	56.000000	56.000000	56.(
mean	22027.500000	70.750000	69.196429	68.089286	67.446429	67.303571	66.000000	66.160714	65.303571	64.392857	64.2
std	16.309506	17.009356	17.712266	18.838333	19.807179	20.746890	21.054043	21.427914	22.728372	23.211814	22.5
min	22000.000000	40.000000	34.000000	35.000000	28.000000	26.000000	29.000000	26.000000	19.000000	9.000000	12.(
25%	22013.750000	57.750000	55.750000	53.000000	54.500000	53.750000	50.250000	47.000000	45.750000	44.000000	45.7
50%	22027.500000	70.500000	68.500000	70.000000	71.500000	69.000000	65.500000	64.000000	67.500000	65.500000	65.5
75%	22041.250000	84.000000	83.250000	85.000000	84.000000	85.250000	83.750000	85.250000	83.250000	84.250000	83.2
max	22055.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.(

In [6]:
```
# To display the col headings
df.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')
```

In [7]:
```
cols=df.dropna(axis=1)
```

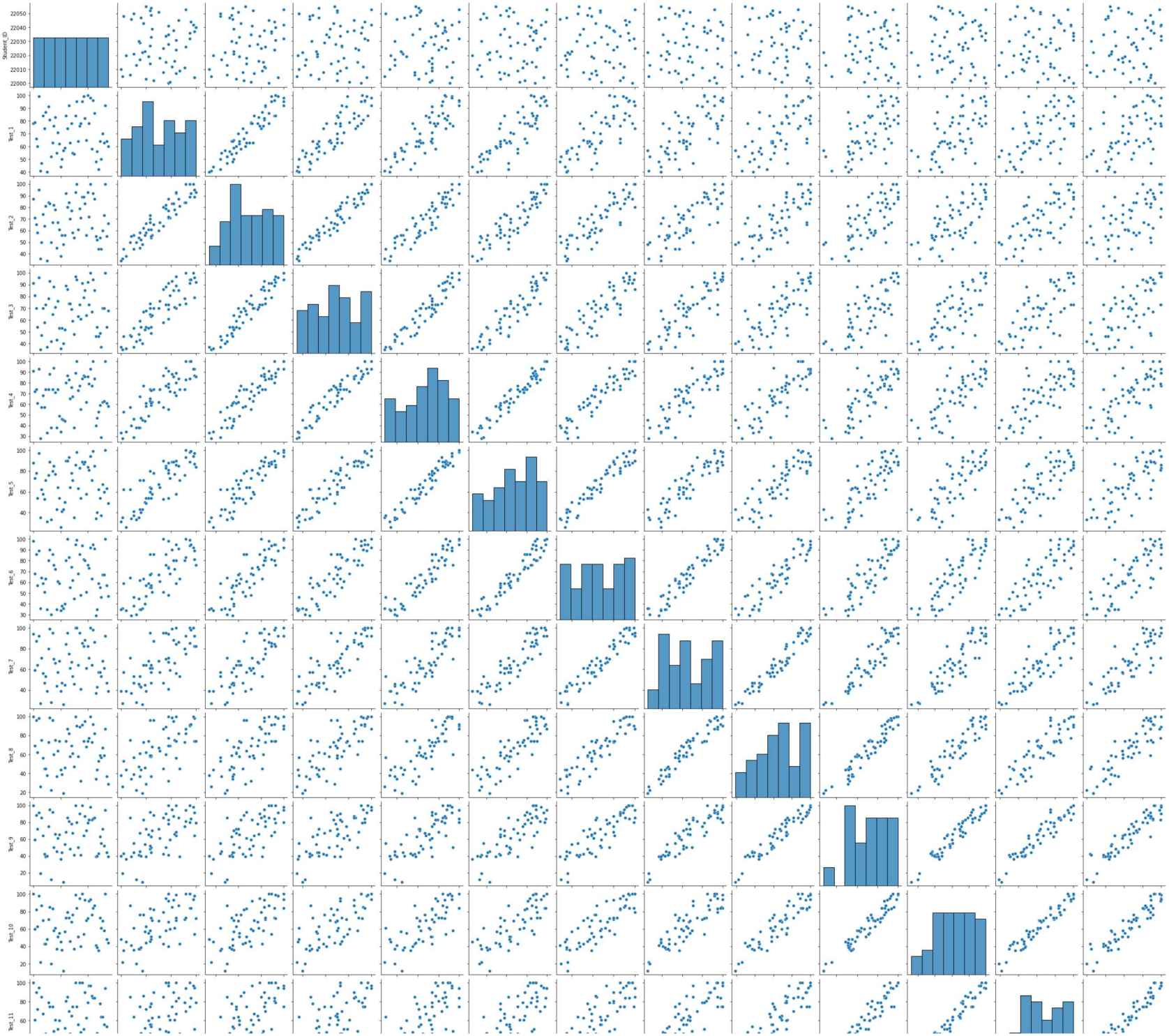
In [8]:
```
cols.columns
```

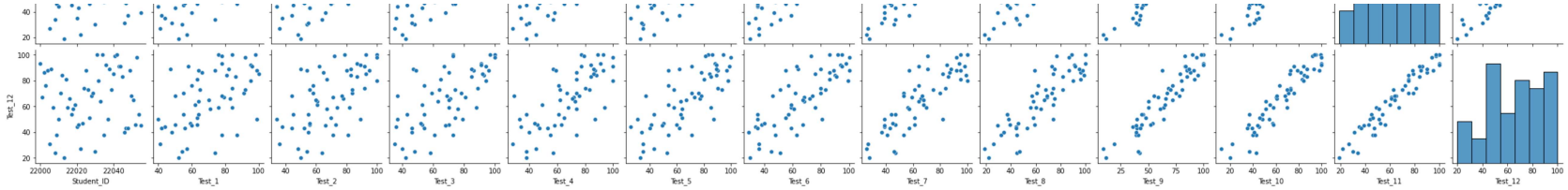
Out[8]:
```
Index(['Student_ID', 'Test_1', 'Test_2', 'Test_3', 'Test_4', 'Test_5',
       'Test_6', 'Test_7', 'Test_8', 'Test_9', 'Test_10', 'Test_11',
       'Test_12'],
      dtype='object')
```

EDA and Visualization

In [9]:
```
1 sns.pairplot(cols)
```

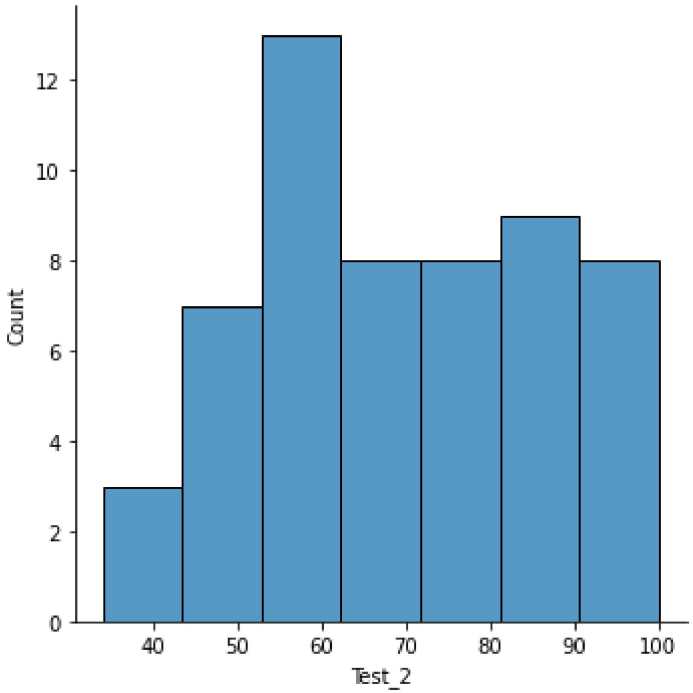
Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x1b9da614f10>
```


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



In [10]:
```
1 sns.displot(df['Test_2'])
```

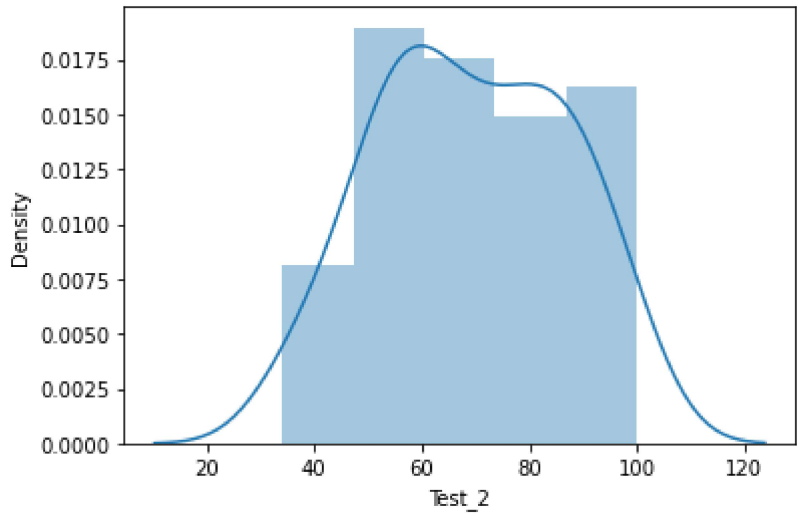
Out[10]: <seaborn.axisgrid.FacetGrid at 0x1b9e0b2b070>



In [11]:
```
# We use displot in older version we get distplot use displot
sns.distplot(df['Test_2'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a dep
recated 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 function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[11]: <AxesSubplot:xlabel='Test_2', ylabel='Density'>



In [12]:

```
df1=cols[['Test_1', 'Test_2', 'Test_3', 'Test_4']]
df1
```

Out[12]:

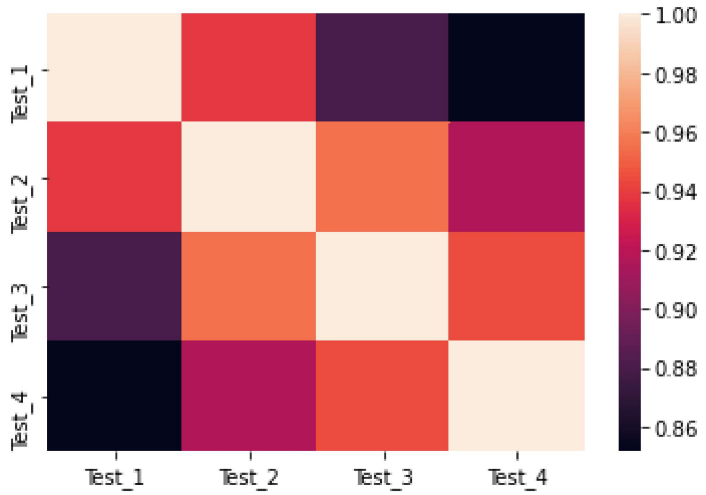
	Test_1	Test_2	Test_3	Test_4
0	78	87	91	91
1	79	71	81	72
2	66	65	70	74
3	60	58	54	61
4	99	95	96	93
5	41	36	35	28
6	47	50	47	57
7	84	74	70	68
8	74	64	58	57
9	87	81	73	74
10	40	34	37	33
11	91	84	78	74
12	81	83	93	88
13	52	50	42	38
14	63	67	65	74
15	76	82	88	94
16	83	78	71	71
17	55	45	43	38
18	71	67	76	74
19	62	61	53	49
20	44	38	36	34
21	50	56	53	46
22	57	48	40	45
23	59	56	52	44
24	84	92	89	80
25	74	80	86	87

	Test_1	Test_2	Test_3	Test_4
26	92	84	74	83
27	63	70	74	65
28	78	77	69	76
29	55	58	59	67
30	54	54	48	38
31	84	93	97	89
32	95	100	94	100
33	64	61	63	73
34	76	79	73	77
35	78	71	61	55
36	95	89	91	84
37	99	89	79	87
38	82	83	85	86
39	65	56	64	62
40	100	93	92	86
41	78	72	73	79
42	98	100	100	93
43	58	62	67	77
44	96	92	94	100
45	86	87	85	84
46	48	55	46	40
47	56	52	54	47
48	42	44	46	53
49	64	54	49	59
50	50	44	37	29
51	70	60	70	62
52	63	73	70	63

	Test_1	Test_2	Test_3	Test_4
53	92	100	100	100
54	64	55	54	61
55	60	66	68	58

In [13]:
```
1 sns.heatmap(df1.corr())
```

Out[13]: <AxesSubplot:>



To train the model - MODEL BUILD

Going to train linear regression model;We split our data into 2 variables x and y where x is independent var(input) and y is dependent on x(output), we could ignore address col as it is not required for our model

In [14]:
```
1 x=df1[['Test_1', 'Test_2', 'Test_3', 'Test_4']]
2 y=df1[['Test_1']]
```

To split the dataset into test data

In [15]:
```
# importing lib for splitting test data
from sklearn.model_selection import train_test_split
```

In [16]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [17]:
```
from sklearn.linear_model import LinearRegression

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

Out[17]:
```
LinearRegression()
```

In [18]:
```
print(lr.intercept_)
```

```
[4.26325641e-14]
```

In [19]:
```
print(lr.score(x_test,y_test))
```

```
1.0
```

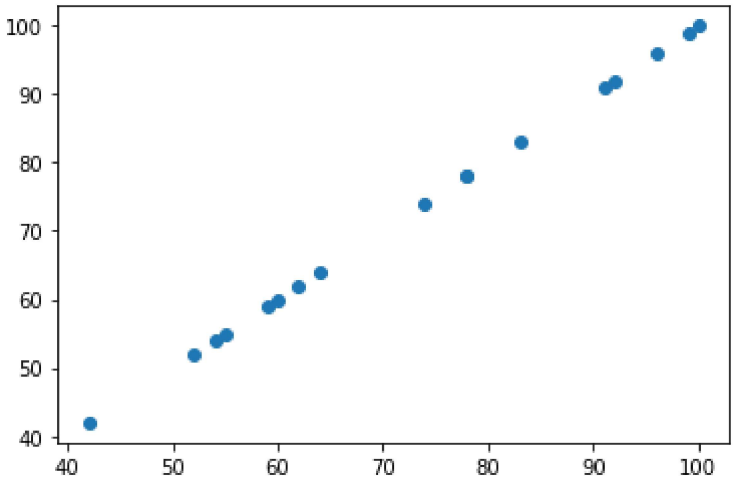
In [20]:
```
coeff=pd.DataFrame(lr.coef_)
coeff
```

Out[20]:

	0	1	2	3
0	1.0	7.445727e-16	-4.313715e-16	-4.115542e-16

In [21]:
```
pred = lr.predict(x_test)
plt.scatter(y_test,pred)
```

Out[21]: <matplotlib.collections.PathCollection at 0x1b9e45cbca0>



In [22]:
```
from sklearn.linear_model import Ridge,Lasso
```

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

Out[23]: Ridge(alpha=10)

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

Out[24]: 0.9999956286026966

In [25]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[25]: Lasso(alpha=10)

In [26]:
```
la.score(x_test,y_test)
```

Out[26]: 0.9985057160904269

ELASTIC NET

In [27]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[27]:
```
ElasticNet()
```

In [28]:
```
print(en.coef_)
```

```
[9.85576129e-01 1.02287630e-02 0.00000000e+00 4.15288145e-04]
```

In [29]:
```
print(en.intercept_)
```

```
[0.27682945]
```

In [30]:
```
prediction=en.predict(x_test)
prediction
```

Out[30]:
```
array([59.01690448, 77.92104623, 92.01423845, 99.82143211, 62.02685312,
       77.97094418, 55.10460912, 52.05400727, 95.87471286, 64.0079723 ,
       42.14310272, 98.85922052, 73.88777527, 54.06607458, 60.02999803,
       90.85420462, 82.90697714])
```

In [31]:
```
print(en.score(x_test,y_test))
```

```
0.9999708945318195
```

EVALUATION METRICS

In [32]:
```
from sklearn import metrics
```

In [33]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 0.08043806694810902
```

In [34]:
```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 0.00928595359209668
```

In [35]:
```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
Root Mean Squared Error: 0.09636365285779011
```