Problem Statement ¶

A real estate agent want help to predict the house price for regions in USA.He gave us the dataset to work on to use linear regression model.Create a model that helps him to estimate of what the house would sell for

Import libraries

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

In [2]:
```
# To import dataset
df=pd.read_csv('17 student csv')
df
```

Out[2]:

	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
10	22010	40	34	37	33	31	35	39	38	40	48
11	22011	91	84	78	74	76	80	80	73	75	71
12	22012	81	83	93	88	89	90	99	99	95	85
13	22013	52	50	42	38	33	30	28	22	12	20
14	22014	63	67	65	74	80	86	95	96	92	83
15	22015	76	82	88	94	85	76	70	60	50	58
16	22016	83	78	71	71	77	72	66	75	66	61
17	22017	55	45	43	38	43	35	44	37	45	37
18	22018	71	67	76	74	64	61	57	64	61	51
19	22019	62	61	53	49	54	59	68	74	65	55
20	22020	44	38	36	34	26	34	39	44	36	45
21	22021	50	56	53	46	41	38	47	39	44	36
22	22022	57	48	40	45	43	36	26	19	9	12
23	22023	59	56	52	44	50	40	45	46	54	57
24	22024	84	92	89	80	90	80	84	74	68	73
25	22025	74	80	86	87	90	100	95	87	85	79
26	22026	92	84	74	83	93	83	75	82	81	73
27	22027	63	70	74	65	64	55	61	58	48	46
28	22028	78	77	69	76	78	74	67	69	78	68
29	22029	55	58	59	67	71	62	53	61	67	76
30	22030	54	54	48	38	35	45	46	47	41	37
31	22031	84	93	97	89	86	95	100	100	100	99
32	22032	95	100	94	100	98	99	100	90	80	84
33	22033	64	61	63	73	63	68	64	58	50	51
34	22034	76	79	73	77	83	86	95	89	90	95

	Student_ID	Test_1	Test_2	Test_3	Test_4	Test_5	Test_6	Test_7	Test_8	Test_9	Test_10
35	22035	78	71	61	55	54	48	41	32	41	40
36	22036	95	89	91	84	89	94	85	91	100	100
37	22037	99	89	79	87	87	81	82	74	64	54
38	22038	82	83	85	86	89	80	88	95	87	93
39	22039	65	56	64	62	58	51	61	68	70	70
40	22040	100	93	92	86	84	76	82	74	79	72
41	22041	78	72	73	79	81	73	71	77	83	92
42	22042	98	100	100	93	94	92	100	100	98	94
43	22043	58	62	67	77	71	63	64	73	83	76
44	22044	96	92	94	100	99	95	98	92	84	84
45	22045	86	87	85	84	85	91	86	82	85	87
46	22046	48	55	46	40	34	29	37	34	39	41
47	22047	56	52	54	47	40	35	43	44	40	39
48	22048	42	44	46	53	62	59	57	53	43	35
49	22049	64	54	49	59	54	55	57	59	63	73
50	22050	50	44	37	29	37	46	53	57	55	61
51	22051	70	60	70	62	67	67	68	67	72	69
52	22052	63	73	70	63	60	67	61	59	52	58
53	22053	92	100	100	100	100	100	92	87	94	100
54	22054	64	55	54	61	63	57	47	37	44	48
55	22055	60	66	68	58	49	47	39	29	39	44

In [3]:
```
# To display top 10 rows
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	T
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	

Data Cleaning and Pre-Processing

In [4]:
```
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]:
```
df.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	5
mean	22027.500000	70.750000	69.196429	68.089286	67.446429	67.303571	66.000000	6
std	16.309506	17.009356	17.712266	18.838333	19.807179	20.746890	21.054043	2
min	22000.000000	40.000000	34.000000	35.000000	28.000000	26.000000	29.000000	2
25%	22013.750000	57.750000	55.750000	53.000000	54.500000	53.750000	50.250000	4
50%	22027.500000	70.500000	68.500000	70.000000	71.500000	69.000000	65.500000	6
75%	22041.250000	84.000000	83.250000	85.000000	84.000000	85.250000	83.750000	8
max	22055.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	10

In [6]:
```
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')
```

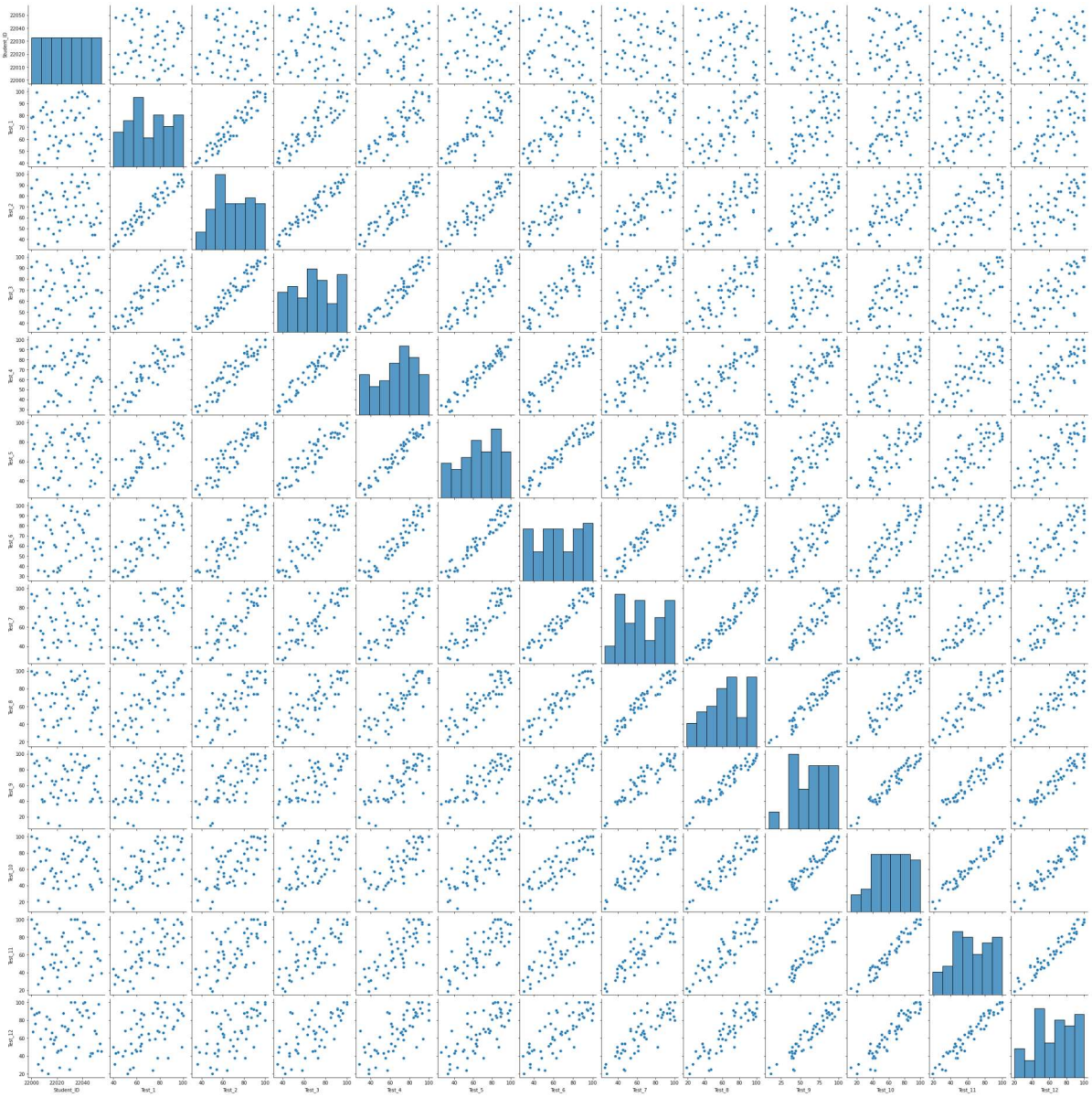
In [7]:
```
a = df.dropna(axis='columns')
a.columns
```

Out[7]:
```
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 [8]:
```
sns.pairplot(a)
```

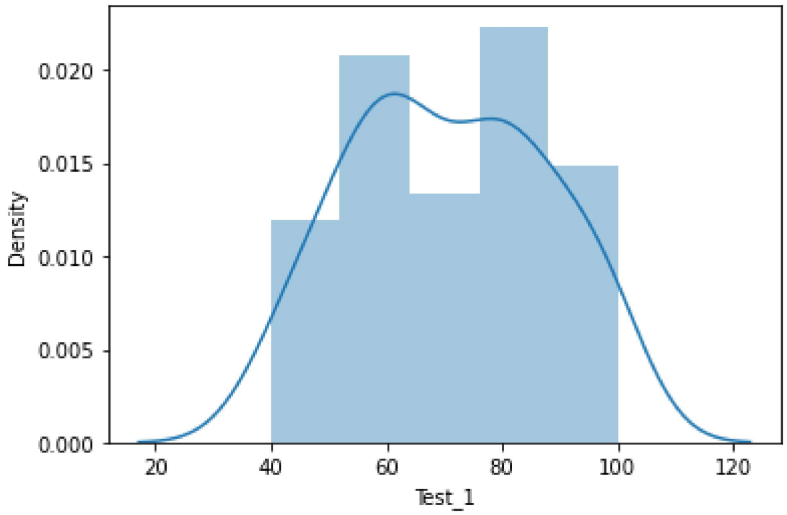
Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x107715e7400>
```



In [10]:
```
sns.distplot(a['Test_1'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)
```

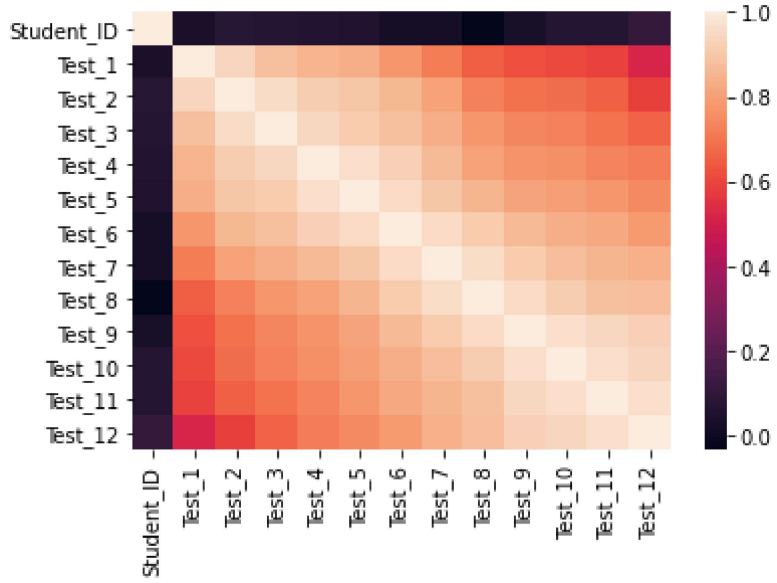
Out[10]: <AxesSubplot:xlabel='Test_1', ylabel='Density'>



In [11]:
```
a1=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']]
```

In [12]:
```
sns.heatmap(a1.corr())
```

Out[12]: <AxesSubplot:>



To Train the Model - Model Building

We are going to train Linear Regression model;We need to split out data into two variables x and y where x is independent variable (input) and y is dependent on x(output). We could ignore address column as it is not required for our model.

In [13]:
```
x=a1[['Student_ID', 'Test_2', 'Test_3', 'Test_4', 'Test_5',
       'Test_6', 'Test_7', 'Test_8', 'Test_9', 'Test_10', 'Test_11',
       'Test_12']]
y=a1['Test_1']
```

To split my dataset into training and test data

In [14]:
```
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 [15]:
```
from sklearn.linear_model import LinearRegression

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

Out[15]:
```
LinearRegression()
```

In [16]:
```
print(lr.intercept_)
```

```
-496.2386938508267
```

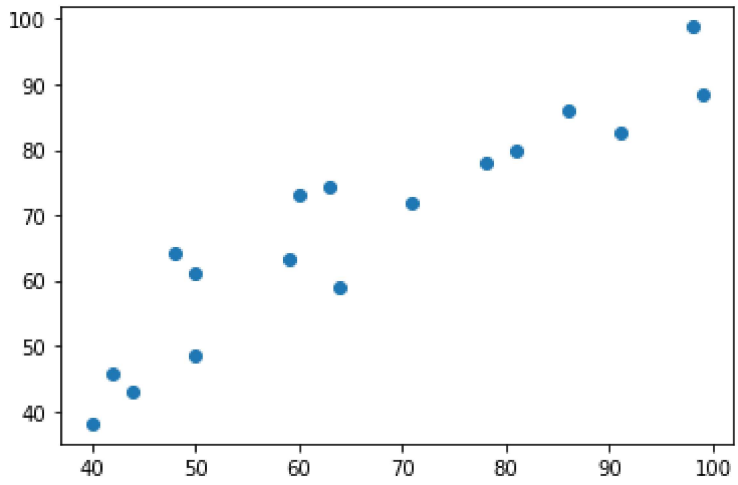
In [17]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[17]:

	Co-efficient
Student_ID	0.022984
Test_2	1.158434
Test_3	-0.044753
Test_4	-0.078670
Test_5	0.108553
Test_6	-0.086070
Test_7	-0.375757
Test_8	0.095003
Test_9	0.160400
Test_10	-0.059013
Test_11	-0.096441
Test_12	0.093856

In [18]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[18]:
```
<matplotlib.collections.PathCollection at 0x1077b7792e0>
```



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

```
0.8507480992411061
```

ACCURACY

In [20]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [21]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
rr.score(x_train,y_train)
```

Out[21]:
```
0.8971263353150887
```

In [22]:
```
rr.score(x_test,y_test)
```

Out[22]:
```
0.8518048344905961
```

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

Out[23]:
```
Lasso(alpha=10)
```

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

Out[24]:
```
0.8728605995704233
```

In [ ]:
```

```