Build the SLR model with Statistics With Housing data

In [2]:

```
#import library
import pandas as pd
import numpy as np
```

In [3]:

```
df1 = pd.read_csv("/Users/myyntiimac/Desktop/House_data.csv")
df1
```

Out[3]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floor
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.(
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.(
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.(
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.(
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.(
...	...	...	...	...	...	...	...	..
21608	263000018	20140521T000000	360000.0	3	2.50	1530	1131	3.(
21609	6600060120	20150223T000000	400000.0	4	2.50	2310	5813	2.(
21610	1523300141	20140623T000000	402101.0	2	0.75	1020	1350	2.(
21611	291310100	20150116T000000	400000.0	3	2.50	1600	2388	2.(
21612	1523300157	20141015T000000	325000.0	2	0.75	1020	1076	2.(

21613 rows × 21 columns

In [4]:

```
df = df1[["sqft_living", "price"]]
df
```

Out[4]:

	sqft_living	price
0	1180	221900.0
1	2570	538000.0
2	770	180000.0
3	1960	604000.0
4	1680	510000.0
...	...	...
21608	1530	360000.0
21609	2310	400000.0
21610	1020	402101.0
21611	1600	400000.0
21612	1020	325000.0

21613 rows × 2 columns

In [5]:

```
df.mean()
```

Out[5]:

```
sqft_living      2079.899736
price          540088.141767
dtype: float64
```

In [6]:

```
df.median()
```

Out[6]:

```
sqft_living      1910.0
price          450000.0
dtype: float64
```

In [7]:

```
df.mode()
```

Out[7]:

	sqft_living	price
0	1300.0	350000.0
1	NaN	450000.0

In [8]:

```
df.isnull().any()
```

Out[8]:

```
sqft_living    False
price          False
dtype: bool
```

In [9]:

```
df.var()
```

Out[9]:

```
sqft_living    8.435337e+05
price          1.347824e+11
dtype: float64
```

In [10]:

```
df["sqft_living"].var()
```

Out[10]:

```
843533.6813681519
```

In [11]:

```
df["price"].var()
```

Out[11]:

```
134782378397.24681
```

In [12]:

```
df.std()
```

Out[12]:

```
sqft_living       918.440897
price          367127.196483
dtype: float64
```

In [13]:

```
df["sqft_living"].std()
```

Out[13]:

```
918.4408970468115
```

In [14]:

```
# for calculatingcoefficient of variation (cv )we have to import a library first
from scipy.stats import variation
```

In [16]:

```
variation(df.values)
```

Out[16]:

```
array([0.44156919, 0.6797385 ])
```

In [17]:

```
variation(df)
```

Out[17]:

```
array([0.44156919, 0.6797385 ])
```

In [18]:

```
variation(df["price"])
```

Out[18]:

```
0.6797384996837632
```

In [19]:

```
df.corr()
```

Out[19]:

	sqft_living	price
sqft_living	1.000000	0.702035
price	0.702035	1.000000

In [20]:

```
df["sqft_living"].corr(df["price"])
```

Out[20]:

```
0.7020350546118002
```

In [21]:

```
df.skew()
```

Out[21]:

```
sqft_living    1.471555
price          4.024069
dtype: float64
```

In [22]:

```
df["sqft_living"].skew()
```

Out[22]:

```
1.471555426802092
```

In [23]:

```
df.sem()
```

Out[23]:

```
sqft_living       6.247319
price          2497.232803
dtype: float64
```

In [24]:

```
# for calculating Z-score we have to import a library first
import scipy.stats as stats
```

In [25]:

```
df.apply(stats.zscore)
```

Out[25]:

	sqft_living	price
0	-0.979835	-0.866717
1	0.533634	-0.005688
2	-1.426254	-0.980849
3	-0.130550	0.174090
4	-0.435422	-0.081958
...	...	...
21608	-0.598746	-0.490545
21609	0.250539	-0.381588
21610	-1.154047	-0.375865
21611	-0.522528	-0.381588
21612	-1.154047	-0.585882

21613 rows × 2 columns

In [26]:

```
stats.zscore(df["price"])
```

Out[26]:

```
0       -0.866717
1       -0.005688
2       -0.980849
3        0.174090
4       -0.081958
           ...
21608   -0.490545
21609   -0.381588
21610   -0.375865
21611   -0.381588
21612   -0.585882
Name: price, Length: 21613, dtype: float64
```

#Now we calculate SSR, SSE AND SST AND CALCULATE R square

In [27]:

```
#define the independnt and dependent variable
x=df.iloc[:,:-1]
x
```

Out[27]:

	sqft_living
0	1180
1	2570
2	770
3	1960
4	1680
...	...
21608	1530
21609	2310
21610	1020
21611	1600
21612	1020

21613 rows × 1 columns

In [28]:

```
y=df.iloc[:,1]
y
```

Out[28]:

```
0        221900.0
1        538000.0
2        180000.0
3        604000.0
4        510000.0
           ...
21608    360000.0
21609    400000.0
21610    402101.0
21611    400000.0
21612    325000.0
Name: price, Length: 21613, dtype: float64
```

In [29]:

```
#to find SSR we need to find y mean
ym=y.mean()
ym
```

Out[29]:

```
540088.1417665294
```

In [30]:

```
#now split the data by calling train_test_split ()from sklearn.model_selection
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(x,y,test_size=0.30,random_state=0)
```

In [31]:

```
X_train
```

Out[31]:

	sqft_living
1468	1390
15590	1450
18552	2860
10535	1050
1069	1240
...	...
13123	3960
19648	1400
9845	2360
10799	2370
2732	2380

15129 rows × 1 columns

In [32]:

```
#then call the regressor function from sklearn.Lenrar_model and defnes as LR than tr
from sklearn.linear_model import LinearRegression
LR = LinearRegression()
LR.fit(X_train,y_train)
```

Out[32]:

```
▾ LinearRegression
LinearRegression()
```

In [33]:

```
#then check the model with x_test data , see how they predict
y_pred=LR.predict(X_test)
y_pred
```

Out[33]:

```
array([ 360116.30871034, 1261900.99358095,  362899.59477476, ...,
        560512.90534826,  374032.73903242,  329500.16200177])
```

In [34]:

```
#we knowSSR = np.sum((y_predict-y_mean)**2)
SSR = np.sum((y_pred-ym)**2)
print(SSR)
```

```
411974711737930.75
```

In [35]:

```
# THen calculate SSE
#Lets select the first  8 elements from y then calculate SSE
y1 = y[0:6484]
y1
```

Out[35]:

```
0       221900.0
1       538000.0
2       180000.0
3       604000.0
4       510000.0
          ...
6479    525000.0
6480    217000.0
6481    525000.0
6482    442500.0
6483    525000.0
Name: price, Length: 6484, dtype: float64
```

In [36]:

```
total_count = len(y_pred)
print(total_count)
```

```
6484
```

In [38]:

```
SSE = np.sum((y1-y_pred)**2)
print(SSE)
```

```
1324286718491765.5
```

In [ ]:

```
#SSE
```

In [41]:

```
#SST
#to calculate SST we will convert whole mean dataframe to numpy array
mean_total = np.mean(df.values)
mean_total
```

Out[41]:

```
271084.0207513996
```

In [42]:

```
SST = np.sum((df.values-mean_total)**2)
SST
```

Out[42]:

```
6040907415735318.0
```

In [43]:

```
#r_square
r_square = SSR/SST
r_square
```

Out[43]:

```
0.06819748812319514
```

```
#A low R-squared value suggests that the linear regression model does not provide a
good fit to the data. It indicates that the independent variable(s) included in the
model do not have a strong linear relationship with the dependent variable. Other
factors or variables that are not included in the model may be influencing the
dependent variable to a greater extent.
```