```
from google.colab import drive
```

```
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

```
import numpy as np

import pandas as pd
```

```
df = pd.read_csv('/content/gdrive/MyDrive/data.csv')
```

```
#task 1--
df.head()
```

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basemen
0	2014-05-02 00:00:00	313000.0	3.0	1.50	1340	7912	1.5	0	0	3	1340	
1	2014-05-02 00:00:00	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5	3370	28
2	2014-05-02 00:00:00	342000.0	3.0	2.00	1930	11947	1.0	0	0	4	1930	
3	2014-05-02 00:00:00	420000.0	3.0	2.25	2000	8030	1.0	0	0	4	1000	100
4	2014-05-02 00:00:00	550000.0	4.0	2.50	1940	10500	1.0	0	0	4	1140	80

```
#task2--
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4600 entries, 0 to 4599
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   date           4600 non-null   object
 1   price          4600 non-null   float64
 2   bedrooms       4600 non-null   float64
 3   bathrooms      4600 non-null   float64
 4   sqft_living    4600 non-null   int64
 5   sqft_lot       4600 non-null   int64
 6   floors         4600 non-null   float64
 7   waterfront     4600 non-null   int64
 8   view           4600 non-null   int64
 9   condition      4600 non-null   int64
 10  sqft_above     4600 non-null   int64
 11  sqft_basement  4600 non-null   int64
 12  yr_built       4600 non-null   int64
 13  yr_renovated   4600 non-null   int64
 14  street         4600 non-null   object
 15  city           4600 non-null   object
 16  statezip       4600 non-null   object
 17  country        4600 non-null   object
dtypes: float64(4), int64(9), object(5)
memory usage: 647.0+ KB
```

```
#task 3--
df.describe(include='all')
```

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	s
count	4600	4.600000e+03	4600.000000	4600.000000	4600.000000	4.600000e+03	4600.000000	4600.000000	4600.000000	4600.000000	46
unique	70	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
top	2014-06-23 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
freq	142	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
mean	NaN	5.519630e+05	3.400870	2.160815	2139.346957	1.485252e+04	1.512065	0.007174	0.240652	3.451739	18
std	NaN	5.638347e+05	0.908848	0.783781	963.206916	3.588444e+04	0.538288	0.084404	0.778405	0.677230	8
min	NaN	0.000000e+00	0.000000	0.000000	370.000000	6.380000e+02	1.000000	0.000000	0.000000	1.000000	3
25%	NaN	3.228750e+05	3.000000	1.750000	1460.000000	5.000750e+03	1.000000	0.000000	0.000000	3.000000	11
50%	NaN	4.609435e+05	3.000000	2.250000	1980.000000	7.683000e+03	1.500000	0.000000	0.000000	3.000000	15
75%	NaN	6.549625e+05	4.000000	2.500000	2620.000000	1.100125e+04	2.000000	0.000000	0.000000	4.000000	23
max	NaN	2.659000e+07	9.000000	8.000000	13540.000000	1.074218e+06	3.500000	1.000000	4.000000	5.000000	94

```
#task 4--(Find the Null values)
df.isnull()
```

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built
0	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
4595	False	False	False	False	False	False	False	False	False	False	False	False	False
4596	False	False	False	False	False	False	False	False	False	False	False	False	False
4597	False	False	False	False	False	False	False	False	False	False	False	False	False
4598	False	False	False	False	False	False	False	False	False	False	False	False	False
4599	False	False	False	False	False	False	False	False	False	False	False	False	False

4600 rows × 18 columns

```
#finding null values
df.isnull().any()
```

```
date             False
price            False
bedrooms         False
bathrooms        False
sqft_living      False
sqft_lot         False
floors           False
waterfront       False
view             False
condition        False
sqft_above       False
sqft_basement    False
yr_built         False
yr_renovated     False
street           False
city             False
statezip         False
country          False
dtype: bool
```

```
#finding null valus in numerical
df.isnull().sum()
```

```
date            0
price           0
bedrooms        0
bathrooms       0
sqft_living     0
sqft_lot        0
floors          0
waterfront      0
view            0
condition       0
sqft_above      0
sqft_basement   0
yr_built        0
yr_renovated    0
street          0
city            0
statezip        0
country         0
dtype: int64
```

```
#selecting a specific data type
df.select_dtypes(exclude='object')
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built
0	3.130000e+05	3.0	1.50	1340	7912	1.5	0	0	3	1340	0	1955
1	2.384000e+06	5.0	2.50	3650	9050	2.0	0	4	5	3370	280	1921
2	3.420000e+05	3.0	2.00	1930	11947	1.0	0	0	4	1930	0	1966
3	4.200000e+05	3.0	2.25	2000	8030	1.0	0	0	4	1000	1000	1963
4	5.500000e+05	4.0	2.50	1940	10500	1.0	0	0	4	1140	800	1976
...	...	...	...	...	...	...	...	...	...	...	...	...
4595	3.081667e+05	3.0	1.75	1510	6360	1.0	0	0	4	1510	0	1954
4596	5.343333e+05	3.0	2.50	1460	7573	2.0	0	0	3	1460	0	1983
4597	4.169042e+05	3.0	2.50	3010	7014	2.0	0	0	3	3010	0	2009
4598	2.034000e+05	4.0	2.00	2090	6630	1.0	0	0	3	1070	1020	1974
4599	2.206000e+05	3.0	2.50	1490	8102	2.0	0	0	4	1490	0	1990

4600 rows × 13 columns

```
#selecting a specific data types
df.select_dtypes(include='object')
```

	date	street	city	statezip	country
0	2014-05-02 00:00:00	18810 Densmore Ave N	Shoreline	WA 98133	USA
1	2014-05-02 00:00:00	709 W Blaine St	Seattle	WA 98119	USA
2	2014-05-02 00:00:00	26206-26214 143rd Ave SE	Kent	WA 98042	USA
3	2014-05-02 00:00:00	857 170th Pl NE	Bellevue	WA 98008	USA
4	2014-05-02 00:00:00	9105 170th Ave NE	Redmond	WA 98052	USA
...	...	...	...	...	...
4595	2014-07-09 00:00:00	501 N 143rd St	Seattle	WA 98133	USA
4596	2014-07-09 00:00:00	14855 SE 10th Pl	Bellevue	WA 98007	USA
4597	2014-07-09 00:00:00	759 Ilwaco Pl NE	Renton	WA 98059	USA
4598	2014-07-10 00:00:00	5148 S Creston St	Seattle	WA 98178	USA
4599	2014-07-10 00:00:00	18717 SE 258th St	Covington	WA 98042	USA

4600 rows × 5 columns

```
df['city'].unique()
```

```
array(['Shoreline', 'Seattle', 'Kent', 'Bellevue', 'Redmond',
       'Maple Valley', 'North Bend', 'Lake Forest Park', 'Sammamish',
```

```
           'Auburn', 'Des Moines', 'Bothell', 'Federal Way', 'Kirkland',
           'Issaquah', 'Woodinville', 'Normandy Park', 'Fall City', 'Renton',
           'Carnation', 'Snoqualmie', 'Duvall', 'Burien', 'Covington',
           'Inglewood-Finn Hill', 'Kenmore', 'Newcastle', 'Mercer Island',
           'Black Diamond', 'Ravensdale', 'Clyde Hill', 'Algona', 'Skykomish',
           'Tukwila', 'Vashon', 'Yarrow Point', 'SeaTac', 'Medina',
           'Enumclaw', 'Snoqualmie Pass', 'Pacific', 'Beaux Arts Village',
           'Preston', 'Milton'], dtype=object)
```

```
df['city'].value_counts()
```

```
Seattle                1573
Renton                  293
Bellevue                286
Redmond                 235
Issaquah                187
Kirkland                187
Kent                    185
Auburn                  176
Sammamish               175
Federal Way             148
Shoreline               123
Woodinville             115
Maple Valley             96
Mercer Island            86
Burien                   74
Snoqualmie               71
Kenmore                  66
Des Moines               58
North Bend               50
Covington                43
Duvall                   42
Lake Forest Park         36
Bothell                  33
Newcastle                33
SeaTac                   29
Tukwila                  29
Vashon                   29
Enumclaw                 28
Carnation                22
Normandy Park            18
Clyde Hill               11
Medina                   11
Fall City                11
Black Diamond             9
Ravensdale                7
Pacific                   6
Algona                    5
Yarrow Point              4
Skykomish                 3
Preston                   2
Milton                    2
Inglewood-Finn Hill       1
Snoqualmie Pass           1
Beaux Arts Village        1
Name: city, dtype: int64
```

```
df['statezip'].value_counts()
```

```
WA 98103    148
WA 98052    135
WA 98117    132
WA 98115    130
WA 98006    110
           ...
WA 98047      6
WA 98288      3
WA 98050      2
WA 98354      2
WA 98068      1
Name: statezip, Length: 77, dtype: int64
```

```
df.head()
```

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basemen
0	2014-05-02 00:00:00	313000.0	3.0	1.50	1340	7912	1.5	0	0	3	1340	
1	2014-05-02 00:00:00	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5	3370	28
2	2014-05-02 00:00:00	342000.0	3.0	2.00	1930	11947	1.0	0	0	4	1930	
3	2014-05-02 00:00:00	420000.0	3.0	2.25	2000	8030	1.0	0	0	4	1000	100
4	2014-05-02 00:00:00	550000.0	4.0	2.50	1940	10500	1.0	0	0	4	1140	80

```
# Removing the unwanted columns
df = df.drop('date',axis=1)
df.head()
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_ren
0	313000.0	3.0	1.50	1340	7912	1.5	0	0	3	1340	0	1955	
1	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5	3370	280	1921	
2	342000.0	3.0	2.00	1930	11947	1.0	0	0	4	1930	0	1966	
3	420000.0	3.0	2.25	2000	8030	1.0	0	0	4	1000	1000	1963	
4	550000.0	4.0	2.50	1940	10500	1.0	0	0	4	1140	800	1976	

```
pd.get_dummies(df['city'])
```

	Algona	Auburn	Beaux Arts Village	Bellevue	Black Diamond	Bothell	Burien	Carnation	Clyde Hill	Covington	...	SeaTac	Seattle	Sh
0	0	0	0	0	0	0	0	0	0	0	...	0	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	1	
2	0	0	0	0	0	0	0	0	0	0	...	0	0	
3	0	0	0	1	0	0	0	0	0	0	...	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4595	0	0	0	0	0	0	0	0	0	0	...	0	1	
4596	0	0	0	1	0	0	0	0	0	0	...	0	0	
4597	0	0	0	0	0	0	0	0	0	0	...	0	0	
4598	0	0	0	0	0	0	0	0	0	0	...	0	1	
4599	0	0	0	0	0	0	0	0	0	1	...	0	0	

4600 rows × 44 columns

```
from sklearn.preprocessing import LabelEncoder
```

```
le=LabelEncoder()
```

```
df['city']=le.fit_transform(df['city'])
```

```
le2=LabelEncoder()
```

```
df['street']=le2.fit_transform(df['street'])
```

```
le3=LabelEncoder()
```

```
df['statezip']=le3.fit_transform(df['statezip'])
```

```
df.head()
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_ren
0	313000.0	3.0	1.50	1340	7912	1.5	0	0	3	1340	0	1955	
1	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5	3370	280	1921	
2	342000.0	3.0	2.00	1930	11947	1.0	0	0	4	1930	0	1966	
3	420000.0	3.0	2.25	2000	8030	1.0	0	0	4	1000	1000	1963	
4	550000.0	4.0	2.50	1940	10500	1.0	0	0	4	1140	800	1976	

```
df['country'].unique()
```

```
array(['USA'], dtype=object)
```

```
df['country']=df['country'].replace({'USA':1})
df.head()
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_ren
0	313000.0	3.0	1.50	1340	7912	1.5	0	0	3	1340	0	1955	
1	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5	3370	280	1921	
2	342000.0	3.0	2.00	1930	11947	1.0	0	0	4	1930	0	1966	
3	420000.0	3.0	2.25	2000	8030	1.0	0	0	4	1000	1000	1963	
4	550000.0	4.0	2.50	1940	10500	1.0	0	0	4	1140	800	1976	

```
x= df.drop('price',axis=1)
y=df['price']
```

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test=train_test_split( x,y,test_size=0.2)
```

```
x_train.shape,x_test.shape
```

```
((3680, 16), (920, 16))
```

```
x_train.head()
```

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated
2143	4.0	1.75	1300	21000	1.0	0	0	4	1300	0	1969	0
4138	5.0	4.00	7320	217800	2.0	0	0	3	7320	0	1992	0
1753	4.0	3.75	2930	3200	1.5	0	0	5	2130	800	1925	0
435	5.0	1.75	2320	8100	1.0	0	0	4	1160	1160	1956	0
3772	2.0	1.00	910	2002	1.5	0	0	3	910	0	1900	2005

```
from sklearn.preprocessing import MinMaxScaler, StandardScaler
```

```
S = StandardScaler()
```

```
xtrainscaled = S.fit_transform(x_train)
```

```
xtrainscaled
```

```
array([[ 0.6639069 , -0.51343648, -0.86674453, ..., -0.96851074,
        -1.8537657 ,  0.        ],
       [ 1.76741972,  2.36097463,  5.37226467, ...,  0.53297898,
        -0.13204052,  0.        ],
       [ 0.6639069 ,  2.04159561,  0.8225553 , ...,  0.78322727,
         0.92012487,  0.        ],
       ...,
       [-0.43960592,  1.7222166 ,  2.61549316, ...,  0.78322727,
         0.34621648,  0.        ],
       [ 0.6639069 ,  2.36097463,  1.5998405 , ...,  0.61639508,
         0.15491368,  0.        ],
       [-0.43960592,  0.12532155, -0.56619425, ..., -0.96851074,
        -1.75811431,  0.        ]])
```

```
xtestscaled = S.transform(x_test)
```

```
xtestscaled
```

```
array([[ 0.6639069 ,  0.76407957,  0.58418784, ...,  0.78322727,
         0.77664777,  0.        ],
       [ 0.6639069 ,  0.44470056,  0.96764854, ...,  0.53297898,
        -0.27551762,  0.        ],
       [-1.54311874, -1.47157351, -1.03256537, ...,  0.78322727,
         1.11142767,  0.        ],
       ...,
       [-1.54311874, -1.47157351, -1.42638987, ...,  0.78322727,
         1.44620757,  0.        ],
       [-1.54311874,  0.44470056, -0.3641001 , ..., -1.88608779,
        -1.71028861,  0.        ],
       [-0.43960592,  0.12532155,  0.05563391, ...,  0.53297898,
        -0.22769192,  0.        ]])
```

```
#Task5-Build ML model with linear regression(Target column is price)
```

```
df=df[['price','sqft_lot']]
```

```
df
```

	price	sqft_lot
0	3.130000e+05	7912
1	2.384000e+06	9050
2	3.420000e+05	11947
3	4.200000e+05	8030
4	5.500000e+05	10500
...	...	...
4595	3.081667e+05	6360
4596	5.343333e+05	7573
4597	4.169042e+05	7014
4598	2.034000e+05	6630
4599	2.206000e+05	8102

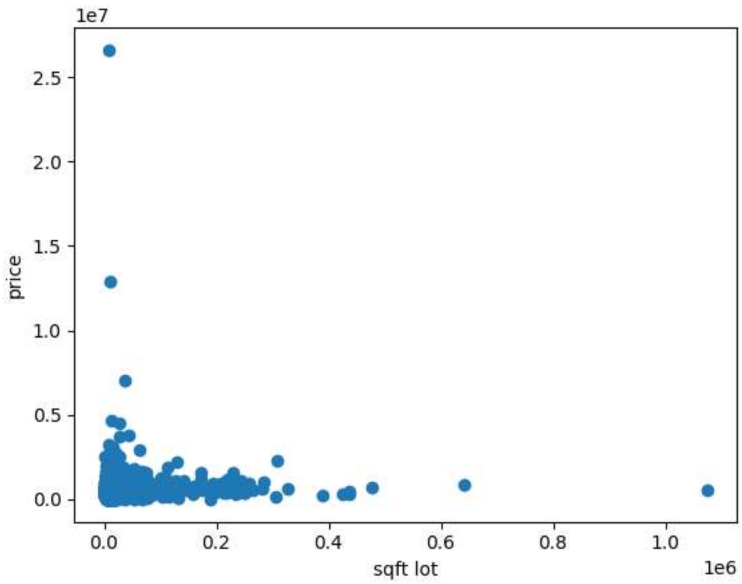
4600 rows × 2 columns

```
x=df['sqft_lot']
y=df['price']
```

```
import matplotlib.pyplot as plt
```

```
plt.scatter(x,y)
plt.xlabel('sqft lot')
plt.ylabel('price')
```

```
Text(0, 0.5, 'price')
```




```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.4,random_state=23)
```

```
x_train=np.array(x_train).reshape(-1,1)
```

```
x_train
```

```
array([[50994],
       [ 5611],
       [54450],
       ...,
       [10650],
       [10362],
       [ 9600]])
```

```
x_test=np.array(x_test).reshape(-1,1)
```

```
x_test
```

```
array([[12686],
       [ 6176],
       [ 5000],
       ...,
       [18200],
       [ 6178],
       [ 1282]])
```

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

```
lr=LinearRegression()
```

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

```
▾ LinearRegression
LinearRegression()
```

```
c=lr.intercept_
```

```
c
```

```
536155.0620619762
```

```
m=lr.coef_
```

```
m
```

```
array([1.14128005])
```

```
y_pred_train=m*x_train+c
y_pred_train.flatten()
```

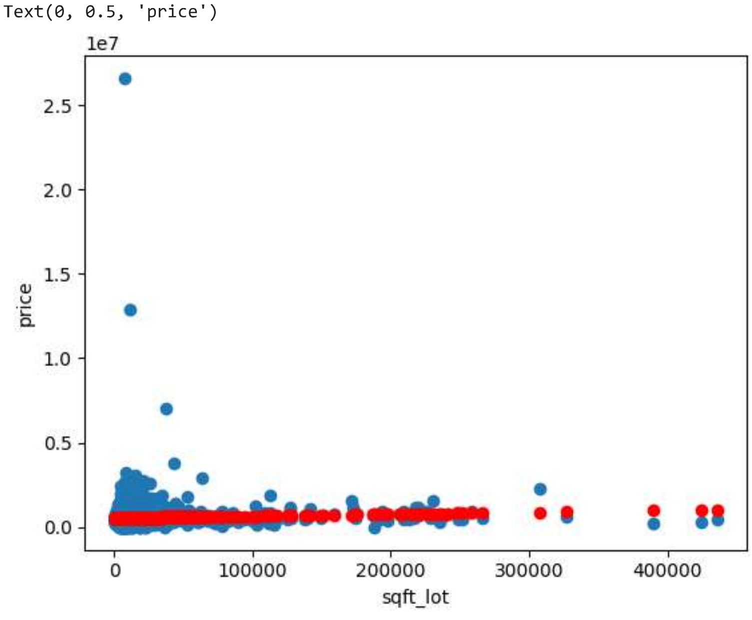
```
array([594353.49699467, 542558.78442946, 598297.76085174, ...,
       548309.69460763, 547981.00595288, 547111.35055383])
```

```
y_pred_train1=lr.predict(x_train)
y_pred_train1
```

```
array([594353.49699467, 542558.78442946, 598297.76085174, ...,
       548309.69460763, 547981.00595288, 547111.35055383])
```

```
plt.scatter(x_train,y_train)
plt.scatter(x_train,y_pred_train1,color='red')
plt.xlabel('sqft_lot')
plt.ylabel('price')
```

```
Text(0, 0.5, 'price')
```




```
plt.scatter(x_train,y_train)
plt.scatter(x_train,y_pred_train1,color='red')
plt.xlabel('sqft_lot')
plt.ylabel('price')
```

```
Text(0, 0.5, 'price')
```


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

```
▾ LinearRegression
LinearRegression()
```

```
y_pred_test=m*x_test+c
y_pred_test.flatten()
```

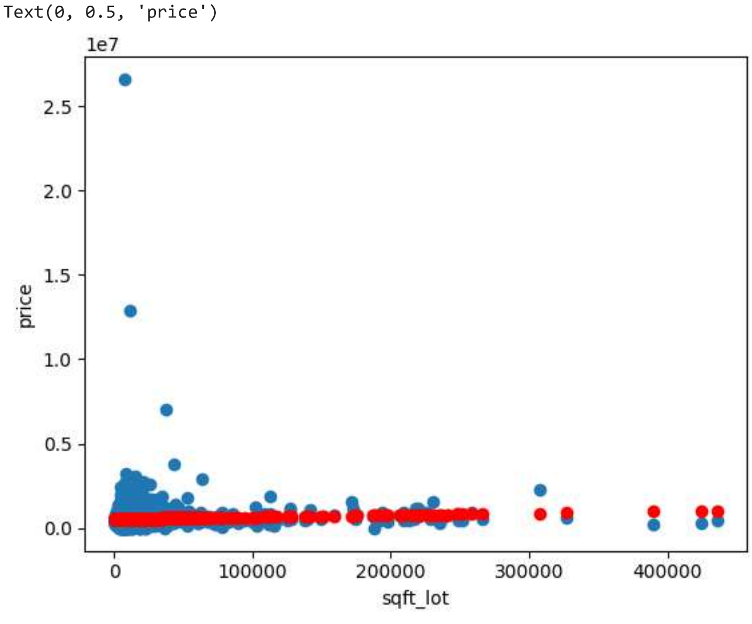
```
array([550633.34079195, 543203.60765841, 541861.46231815, ...,
       556926.35899446, 543205.89021851, 537618.18308766])
```

```
y_pred_test1=lr.predict(x_test)
y_pred_test
```

```
array([[550633.34079195],
       [543203.60765841],
       [541861.46231815],
       ...,
       [556926.35899446],
       [543205.89021851],
       [537618.18308766]])
```

```
plt.scatter(x_train,y_train)
plt.scatter(x_train,y_pred_train1,color='red')
plt.xlabel('sqft_lot')
plt.ylabel('price')
```

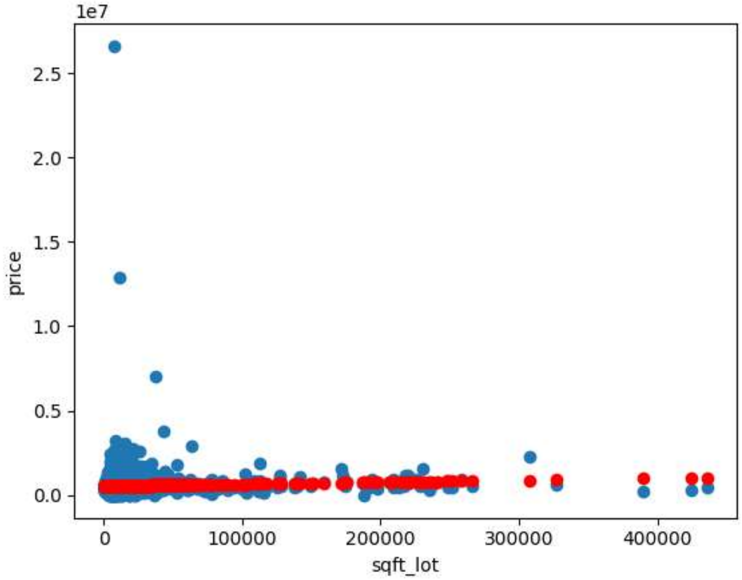
```
Text(0, 0.5, 'price')
```




```
plt.scatter(x_train,y_train)
plt.scatter(x_train,y_pred_train1,color='red')
```

```
plt.xlabel('sqft_lot')
plt.ylabel('price')
```

```
Text(0, 0.5, 'price')
```




```
"Therefore the linear model is satisfied for both train and test date"
```

```
'Therefore the linear model is satisfied for both train and test date'
```