problem statement

To predict the best house price using dataset

In [39]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing,svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [40]:
```
df=pd.read_csv(r"C:\Users\LENOVO\Downloads\data.csv")
df
```

Out[40]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sc
0	2014-05-02 00:00:00	3.130000e+05	3.0	1.50	1340	7912	1.5	0	0	3	
1	2014-05-02 00:00:00	2.384000e+06	5.0	2.50	3650	9050	2.0	0	4	5	
2	2014-05-02 00:00:00	3.420000e+05	3.0	2.00	1930	11947	1.0	0	0	4	
3	2014-05-02 00:00:00	4.200000e+05	3.0	2.25	2000	8030	1.0	0	0	4	
4	2014-05-02 00:00:00	5.500000e+05	4.0	2.50	1940	10500	1.0	0	0	4	
...	...	...	...	...	...	...	...	...	...	...	...
4595	2014-07-09 00:00:00	3.081667e+05	3.0	1.75	1510	6360	1.0	0	0	4	
4596	2014-07-09 00:00:00	5.343333e+05	3.0	2.50	1460	7573	2.0	0	0	3	
4597	2014-07-09 00:00:00	4.169042e+05	3.0	2.50	3010	7014	2.0	0	0	3	
4598	2014-07-10 00:00:00	2.034000e+05	4.0	2.00	2090	6630	1.0	0	0	3	
4599	2014-07-10 00:00:00	2.206000e+05	3.0	2.50	1490	8102	2.0	0	0	4	

4600 rows × 18 columns

In [3]:
```
df.shape
```

Out[3]: (4600, 18)

In [4]:
```
df.describe()
```

Out[4]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	v
count	4.600000e+03	4600.000000	4600.000000	4600.000000	4.600000e+03	4600.000000	4600.000000	4600.000
mean	5.519630e+05	3.400870	2.160815	2139.346957	1.485252e+04	1.512065	0.007174	0.240
std	5.638347e+05	0.908848	0.783781	963.206916	3.588444e+04	0.538288	0.084404	0.778
min	0.000000e+00	0.000000	0.000000	370.000000	6.380000e+02	1.000000	0.000000	0.000
25%	3.228750e+05	3.000000	1.750000	1460.000000	5.000750e+03	1.000000	0.000000	0.000
50%	4.609435e+05	3.000000	2.250000	1980.000000	7.683000e+03	1.500000	0.000000	0.000
75%	6.549625e+05	4.000000	2.500000	2620.000000	1.100125e+04	2.000000	0.000000	0.000
max	2.659000e+07	9.000000	8.000000	13540.000000	1.074218e+06	3.500000	1.000000	4.000

In [41]:
```
df.head(10)
```

Out[41]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_abo
0	2014-05-02 00:00:00	313000.0	3.0	1.50	1340	7912	1.5	0	0	3	13
1	2014-05-02 00:00:00	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5	33
2	2014-05-02 00:00:00	342000.0	3.0	2.00	1930	11947	1.0	0	0	4	19
3	2014-05-02 00:00:00	420000.0	3.0	2.25	2000	8030	1.0	0	0	4	10
4	2014-05-02 00:00:00	550000.0	4.0	2.50	1940	10500	1.0	0	0	4	11
5	2014-05-02 00:00:00	490000.0	2.0	1.00	880	6380	1.0	0	0	3	8
6	2014-05-02 00:00:00	335000.0	2.0	2.00	1350	2560	1.0	0	0	3	13
7	2014-05-02 00:00:00	482000.0	4.0	2.50	2710	35868	2.0	0	0	3	27
8	2014-05-02 00:00:00	452500.0	3.0	2.50	2430	88426	1.0	0	0	4	15
9	2014-05-02 00:00:00	640000.0	4.0	2.00	1520	6200	1.5	0	0	3	15

In [42]:
```
df=df[['price','yr_built']]
df
```

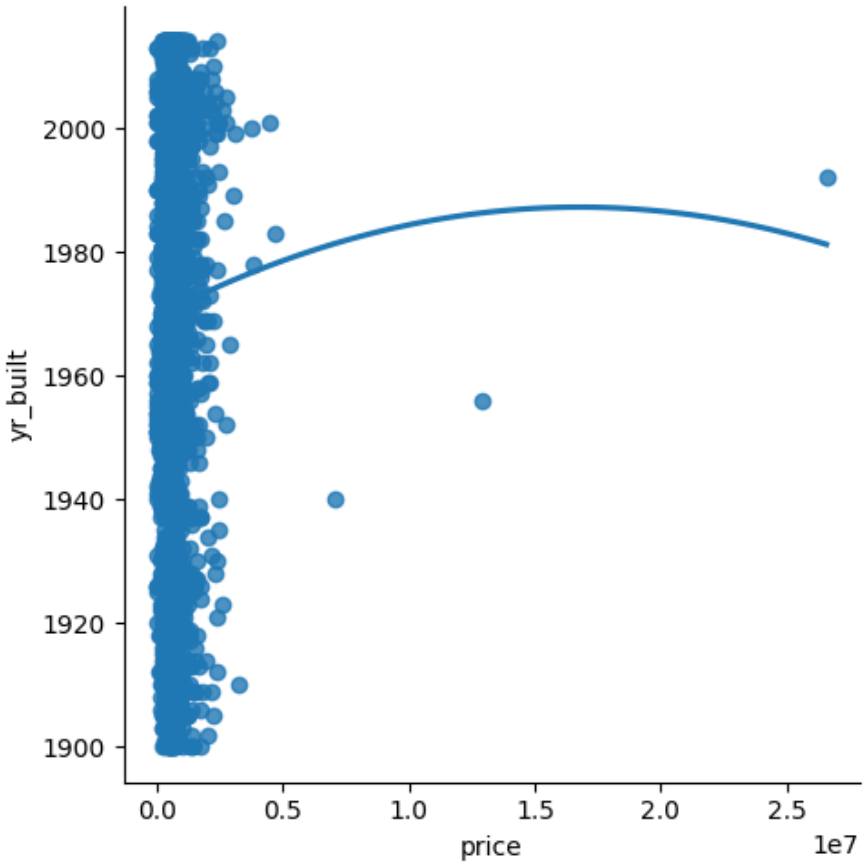
Out[42]:

	price	yr_built
0	3.130000e+05	1955
1	2.384000e+06	1921
2	3.420000e+05	1966
3	4.200000e+05	1963
4	5.500000e+05	1976
...	...	...
4595	3.081667e+05	1954
4596	5.343333e+05	1983
4597	4.169042e+05	2009
4598	2.034000e+05	1974
4599	2.206000e+05	1990

4600 rows × 2 columns

In [43]:
```
sns.lmplot(x="price",y="yr_built",data=df,order=2,ci=None)
```

Out[43]:
```
<seaborn.axisgrid.FacetGrid at 0x1e525979650>
```



In [44]:
```
df.fillna(method='ffill',inplace=True)
```

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_4248\3337295870.py:1: SettingWithCopyWarnin
g:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df.fillna(method='ffill',inplace=True)
```

In [45]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4600 entries, 0 to 4599
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   price     4600 non-null   float64
 1   yr_built  4600 non-null   int64
dtypes: float64(1), int64(1)
memory usage: 72.0 KB
```

In [46]:
```
x=np.array(df['price']).reshape(-1,1)
y=np.array(df['yr_built']).reshape(-1,1)
```

In [47]:
```
df.dropna(inplace=True)
```

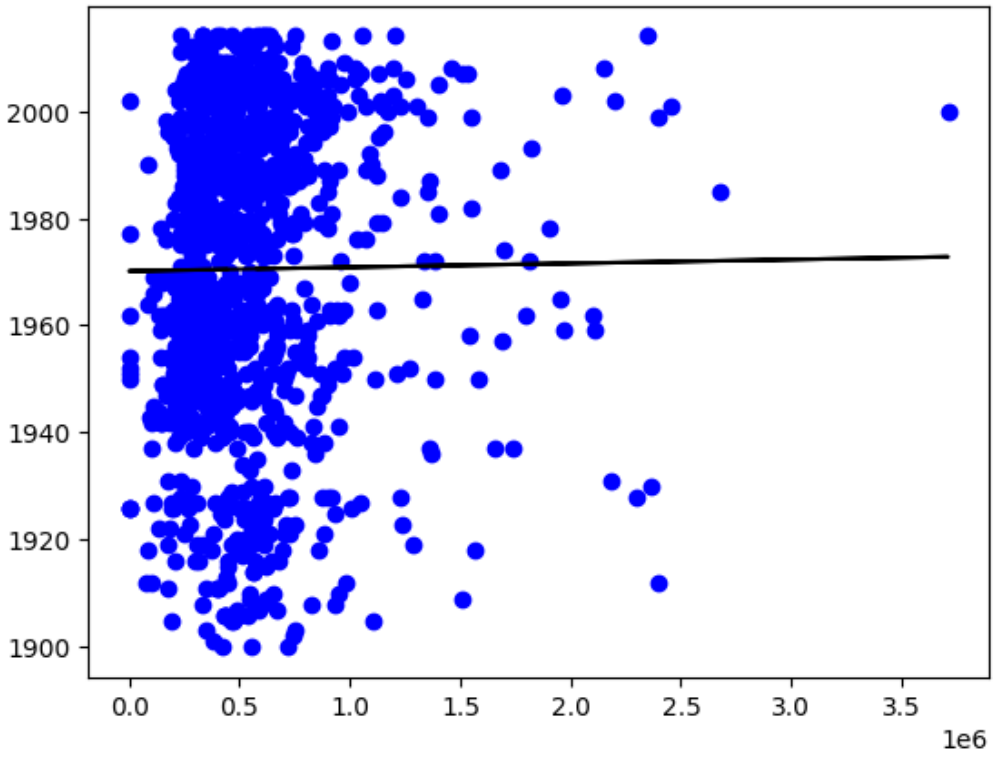
```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_4248\1379821321.py:1: SettingWithCopyWarnin
g:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df.dropna(inplace=True)
```

In [48]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

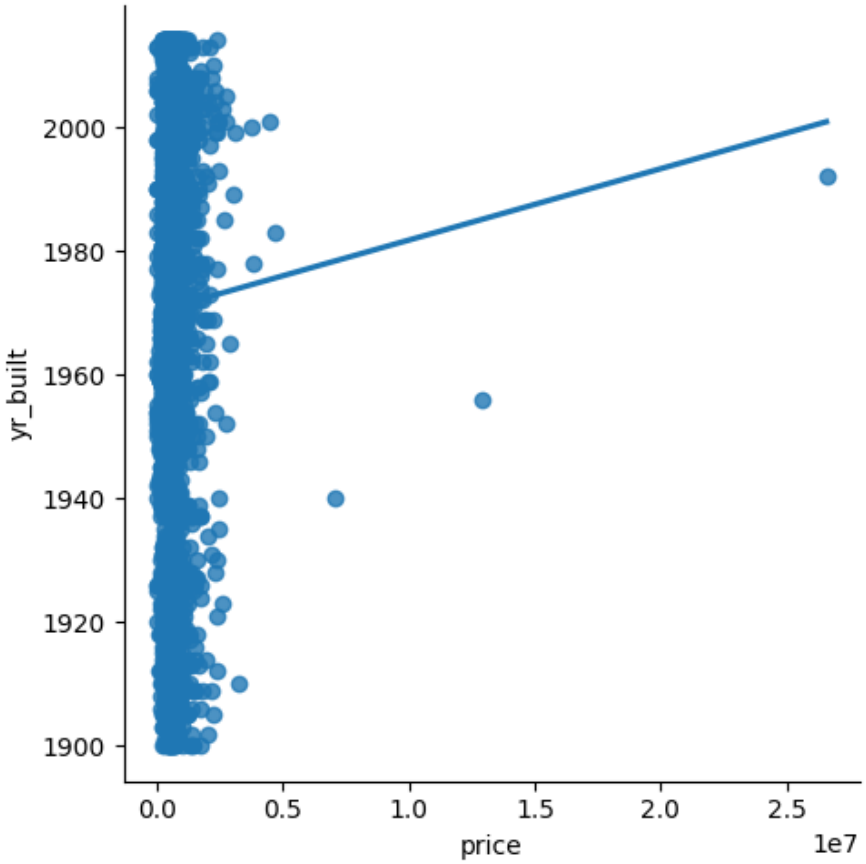
```
-0.00023855891480950575
```

In [49]:
```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



In [34]:
```
df4500=df[:][:4500]
sns.lmplot(x="price",y="yr_built",data=df4500,order=1,ci=None)
```

Out[34]:
```
<seaborn.axisgrid.FacetGrid at 0x1e52578b090>
```



In [53]:
```
x=np.array(df4500['price']).reshape(-1,1)
y=np.array(df4500['yr_built']).reshape(-1,1)
```

In [54]:
```
df4500.dropna(inplace=True)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
regr=LinearRegression()
regr.fit(x_train,y_train)
print("Regression:",regr.score(x_test,y_test))
```

```
Regression: -0.004517424217823418
```

In [55]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
model = LinearRegression()
model.fit(x_train,y_train)
```

Out[55]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook. On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [38]:
```
y_pred=model.predict(x_test)
r2=r2_score(y_test,y_pred)
print("r2 score:",r2)
```

```
r2 score: -0.005510630108097603
```

conclusion:

This data is not best fit for house house prediction since it accuracy value is very less.