In [8]:

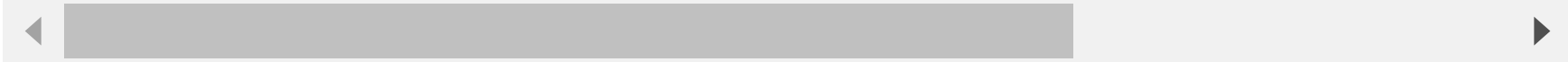
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing,svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [9]:
```
df=pd.read_csv(r"C:\Users\MY HOME\Downloads\data.csv")
df
```

Out[9]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	y
0	2014-05-02 00:00:00	3.130000e+05	3.0	1.50	1340	7912	1.5	0	0	3	1340	0	
1	2014-05-02 00:00:00	2.384000e+06	5.0	2.50	3650	9050	2.0	0	4	5	3370	280	
2	2014-05-02 00:00:00	3.420000e+05	3.0	2.00	1930	11947	1.0	0	0	4	1930	0	
3	2014-05-02 00:00:00	4.200000e+05	3.0	2.25	2000	8030	1.0	0	0	4	1000	1000	
4	2014-05-02 00:00:00	5.500000e+05	4.0	2.50	1940	10500	1.0	0	0	4	1140	800	
...	...	...	...	...	...	...	...	...	...	...	...	...	...
4595	2014-07-09 00:00:00	3.081667e+05	3.0	1.75	1510	6360	1.0	0	0	4	1510	0	
4596	2014-07-09 00:00:00	5.343333e+05	3.0	2.50	1460	7573	2.0	0	0	3	1460	0	
4597	2014-07-09 00:00:00	4.169042e+05	3.0	2.50	3010	7014	2.0	0	0	3	3010	0	
4598	2014-07-10 00:00:00	2.034000e+05	4.0	2.00	2090	6630	1.0	0	0	3	1070	1020	
4599	2014-07-10 00:00:00	2.206000e+05	3.0	2.50	1490	8102	2.0	0	0	4	1490	0	

4600 rows × 18 columns



In [10]:
```
#taking selected columns from dataset
df=df[['sqft_living','yr_built']]
df
```

Out[10]:

	sqft_living	yr_built
0	1340	1955
1	3650	1921
2	1930	1966
3	2000	1963
4	1940	1976
...	...	...
4595	1510	1954
4596	1460	1983
4597	3010	2009
4598	2090	1974
4599	1490	1990

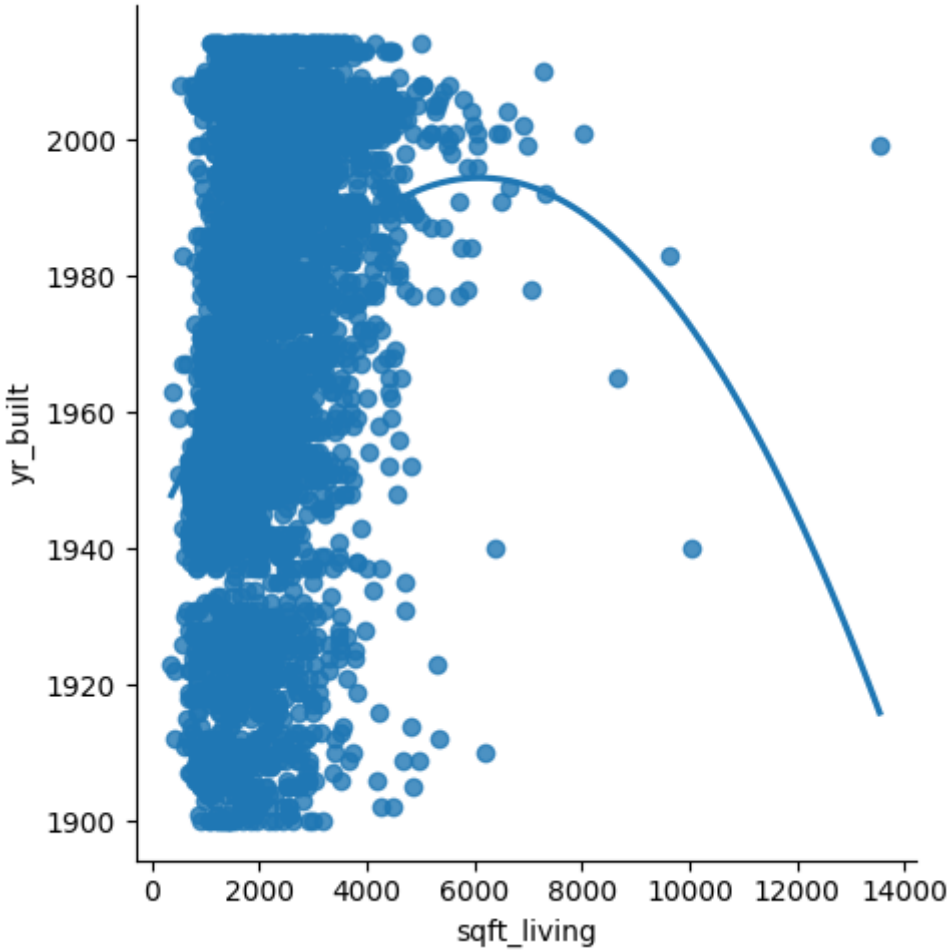
4600 rows × 2 columns

In [11]:
```
df.head(10)
```

Out[11]:

	sqft_living	yr_built
0	1340	1955
1	3650	1921
2	1930	1966
3	2000	1963
4	1940	1976
5	880	1938
6	1350	1976
7	2710	1989
8	2430	1985
9	1520	1945

In [12]:
```
sns.lmplot(x="sqft_living",y="yr_built",order=2,data=df,ci=None)
plt.show()
```



In [13]:
```
df.describe()
```

Out[13]:

	sqft_living	yr_built
count	4600.000000	4600.000000
mean	2139.346957	1970.786304
std	963.206916	29.731848
min	370.000000	1900.000000
25%	1460.000000	1951.000000
50%	1980.000000	1976.000000
75%	2620.000000	1997.000000
max	13540.000000	2014.000000

In [14]:
```
df.info()
```

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

In [15]:
```
df.fillna(method="ffill",inplace=True)
df
```

```
C:\Users\MY HOME\AppData\Local\Temp\ipykernel_14416\2729279820.py:1: SettingWithCopyWarning:
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_guide/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)
```

Out[15]:

	sqft_living	yr_built
0	1340	1955
1	3650	1921
2	1930	1966
3	2000	1963
4	1940	1976
...	...	...
4595	1510	1954
4596	1460	1983
4597	3010	2009
4598	2090	1974
4599	1490	1990

4600 rows × 2 columns

In [ ]:
```
###step 5:Training our model
```

In [16]:
```
#Separating data into independent & dependent variables
#Now each dataframe contains only one coloumn
x=np.array(df['sqft_living']).reshape(-1,1)
y=np.array(df['yr_built']).reshape(-1,1)
#Dropping any rows with Nan values
df.dropna(inplace=True)
df
```

```
C:\Users\MY HOME\AppData\Local\Temp\ipykernel_14416\285053503.py:6: SettingWithCopyWarning:
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_guide/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)
```

Out[16]:

	sqft_living	yr_built
0	1340	1955
1	3650	1921
2	1930	1966
3	2000	1963
4	1940	1976
...	...	...
4595	1510	1954
4596	1460	1983
4597	3010	2009
4598	2090	1974
4599	1490	1990

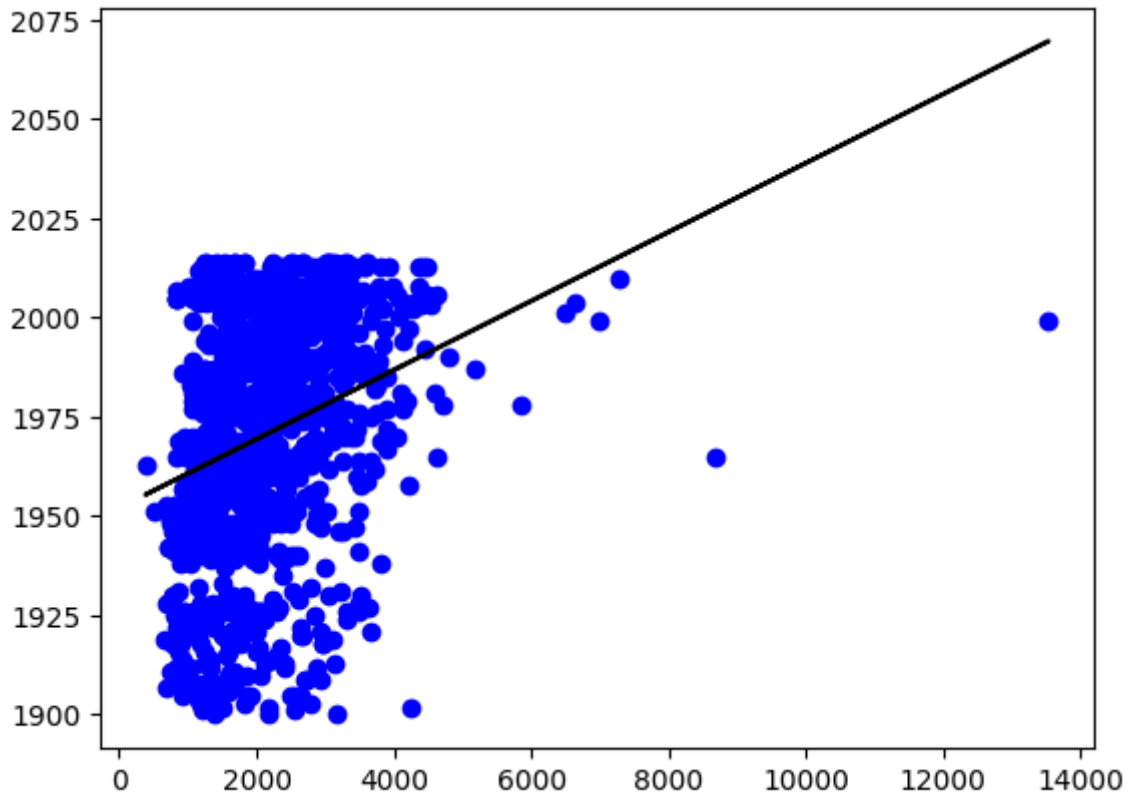
4600 rows × 2 columns

In [17]:
```
#Splitting the data into training and testing data
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))
```

```
0.09303377068504926
```

In [ ]:
```
###step 6:Exploring our results
```

In [18]:
```
#Data scatter to predict the values
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 [19]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
#Train the model
model=LinearRegression()
model.fit(x_train,y_train)
#evaluate the model on the test set
y_pred=model.predict(x_test)
r2=r2_score(y_test,y_pred)
print("r2 Score:",r2)
```

```
r2 Score: 0.09303377068504926
```

conclusion:

The dataset we have taken is acceptable.but,it may cannot be a best fit.p