In [1]:
```
#step-1:problem statement
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 [2]:
```
#step-2:reading the data set
df=pd.read_csv(r"C:\Users\chinta pavani\Downloads\data.csv")
df
```

Out[2]:

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

4600 rows × 18 columns

In [3]:
```
df=df[['sqft_living','yr_renovated']]
df.columns=['sqft','yr']
```

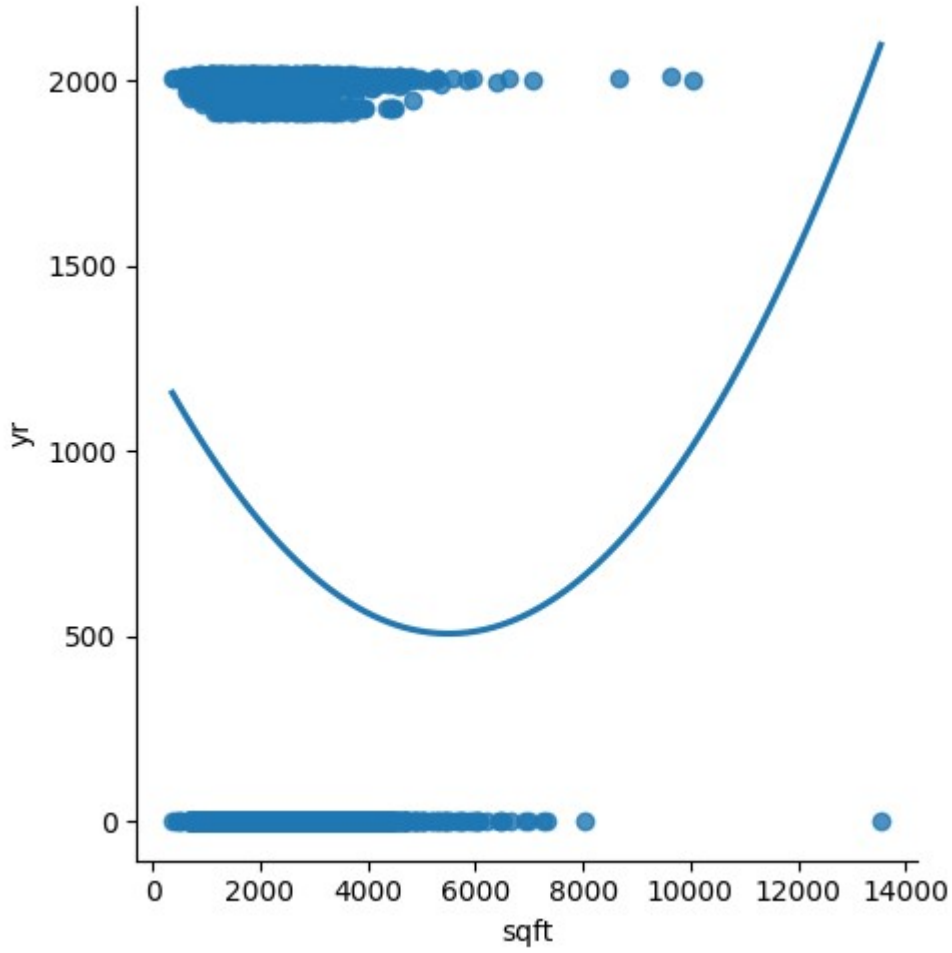
In [4]:
```
df.head(10)
```

Out[4]:

	sqft	yr
0	1340	2005
1	3650	0
2	1930	0
3	2000	0
4	1940	1992
5	880	1994
6	1350	0
7	2710	0
8	2430	0
9	1520	2010

In [5]:
```
#step-3:exploring the data scatter-plotting the data scatter
sns.lmplot(x="sqft",y="yr",data=df,order=2,ci=None)
```

Out[5]: <seaborn.axisgrid.FacetGrid at 0x26149f36010>



In [6]:
```
df.describe()
```

Out[6]:

	sqft	yr
count	4600.000000	4600.000000
mean	2139.346957	808.608261
std	963.206916	979.414536
min	370.000000	0.000000
25%	1460.000000	0.000000
50%	1980.000000	0.000000
75%	2620.000000	1999.000000
max	13540.000000	2014.000000

In [8]:
```
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    4600 non-null   int64
 1   yr      4600 non-null   int64
dtypes: int64(2)
memory usage: 72.0 KB
```

In [9]:
```
#step-4:data cleaning-eliminating nan or missing input
df.fillna(method='ffill',inplace=True)
```

```
C:\Users\chinta pavani\AppData\Local\Temp\ipykernel_19336\2352573223.py:
2: 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-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  df.fillna(method='ffill',inplace=True)
```

In [10]:
```
#step-5:training our model

x=np.array(df['sqft']).reshape(-1,1)

y=np.array(df['yr']).reshape(-1,1)
```

In [11]:
```
df.dropna(inplace=True)
```

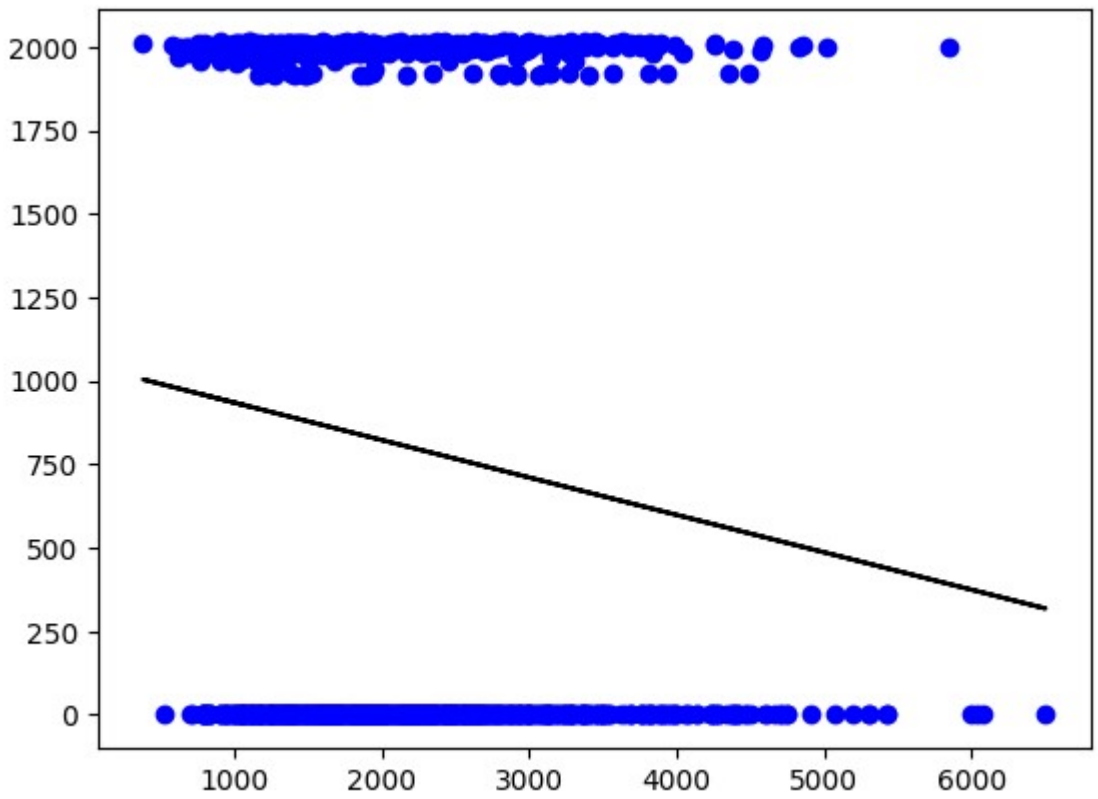
```
C:\Users\chinta pavani\AppData\Local\Temp\ipykernel_19336\1379821321.py:
1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

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

In [12]:
```
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))
```

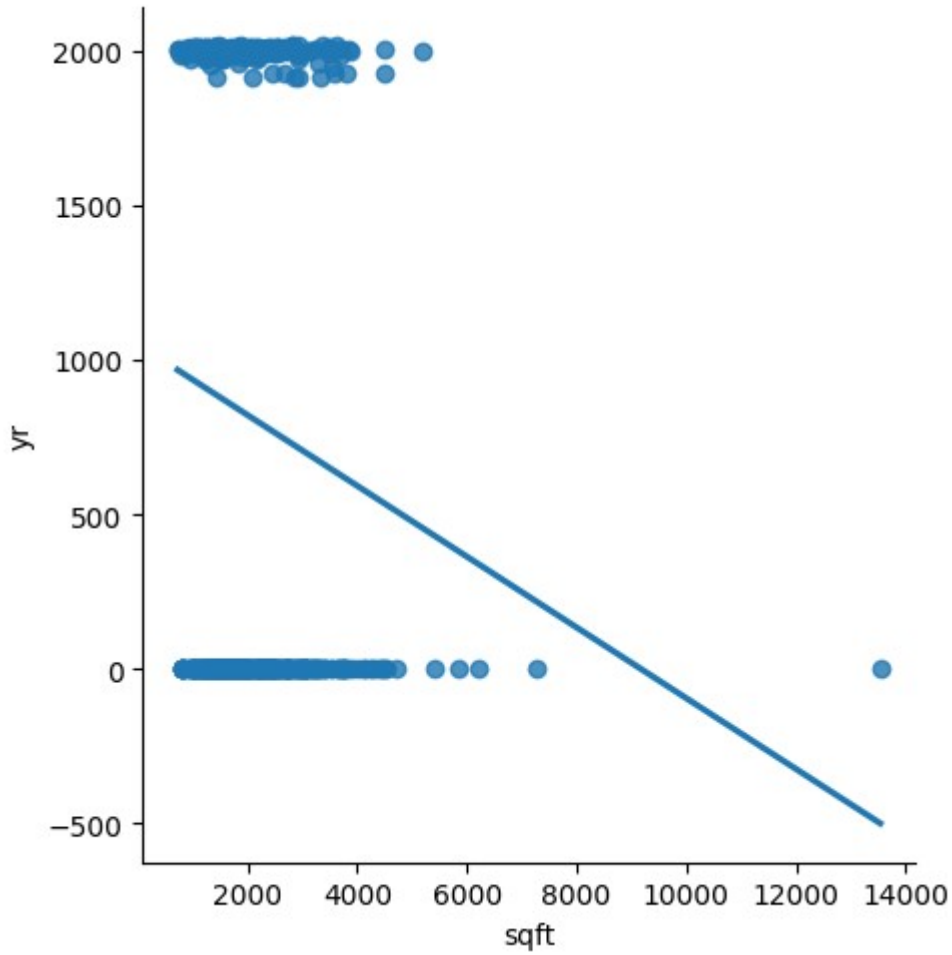
```
0.022971547897271183
```

In [13]:
```
#step-6:exploring our results
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
#data scatter of predicted values
```



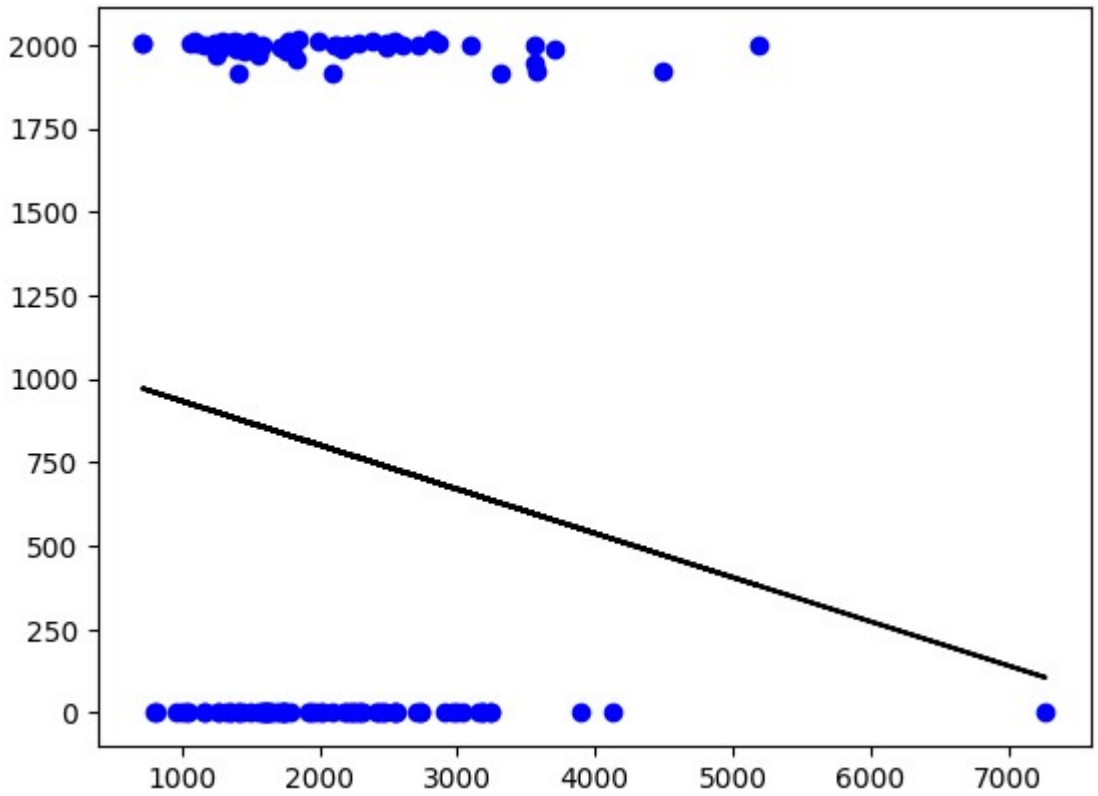
In [14]:
```
#step-7:working with a smallest dataset
df500=df[:][:500]
sns.lmplot(x="sqft",y="yr",data=df500,order=1,ci=None)
```

Out[14]: <seaborn.axisgrid.FacetGrid at 0x26149f71d90>



In [15]:
```
df500.fillna(method='ffill',inplace=True)
x=np.array(df500['sqft']).reshape(-1,1)
y=np.array(df500['yr']).reshape(-1,1)
df500.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))
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```

```
Regression: -0.01354464674424749
```



In [16]:
```
#step-8:evaluation of model
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
model=LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
r2=r2_score(y_test,y_pred)
print("R2 score:",r2)
```

```
R2 score: -0.01354464674424749
```

In [17]:
```
#step-9:conclusion
#dataset we have taken is poor for linear model but with the smaller data
```

In [ ]: