House Price Prediction

Importing Libraries and Data Loading

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df=pd.read_csv("/content/House price Prediction.csv")
df
```

Out[2]:

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

4600 rows × 18 columns

In [3]:
```
df.head()
```

Out[3]:

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

In [4]:

```
df.tail()
```

Out[4]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	vie
4595	2014-07-09 00:00:00	308166.666667	3.0	1.75	1510	6360	1.0	0	
4596	2014-07-09 00:00:00	534333.333333	3.0	2.50	1460	7573	2.0	0	
4597	2014-07-09 00:00:00	416904.166667	3.0	2.50	3010	7014	2.0	0	
4598	2014-07-10 00:00:00	203400.000000	4.0	2.00	2090	6630	1.0	0	
4599	2014-07-10 00:00:00	220600.000000	3.0	2.50	1490	8102	2.0	0	

In [5]:

```
df.isna().sum()
```

Out[5]:

	0
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

In [6]:

```
df.dtypes
```

Out[6]:

```
                     0
         date   object
        price  float64
     bedrooms  float64
    bathrooms  float64
  sqft_living    int64
     sqft_lot    int64
       floors  float64
   waterfront    int64
         view    int64
    condition    int64
   sqft_above    int64
sqft_basement    int64
     yr_built    int64
 yr_renovated    int64
       street   object
         city   object
     statezip   object
      country   object

dtype: object
```

In [7]:

```
df.drop(['date','street','city','country'],axis=1,inplace=True)
df
```

Out[7]:

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

4600 rows × 14 columns

In [8]:

```
#Showing no.of unique values in each columns
df.nunique()
```

Out[8]:

	0
price	1741
bedrooms	10
bathrooms	26
sqft_living	566
sqft_lot	3113
floors	6
waterfront	2
view	5
condition	5
sqft_above	511
sqft_basement	207
yr_built	115
yr_renovated	60
statezip	77

dtype: int64

In [9]:

```
for i in df['yr_renovated']:
  if i!=0:
    df['yr_renovated']=df['yr_renovated'].replace(i,2024-i)
df
```

Out[9]:

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

4600 rows × 14 columns

In [10]:

```
df['statezip']=df['statezip'].str.replace('WA','')
df['statezip']=df['statezip'].astype(int)
df
```

Out[10]:

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

4600 rows × 14 columns
```

In [11]:

```python
df.dtypes
```

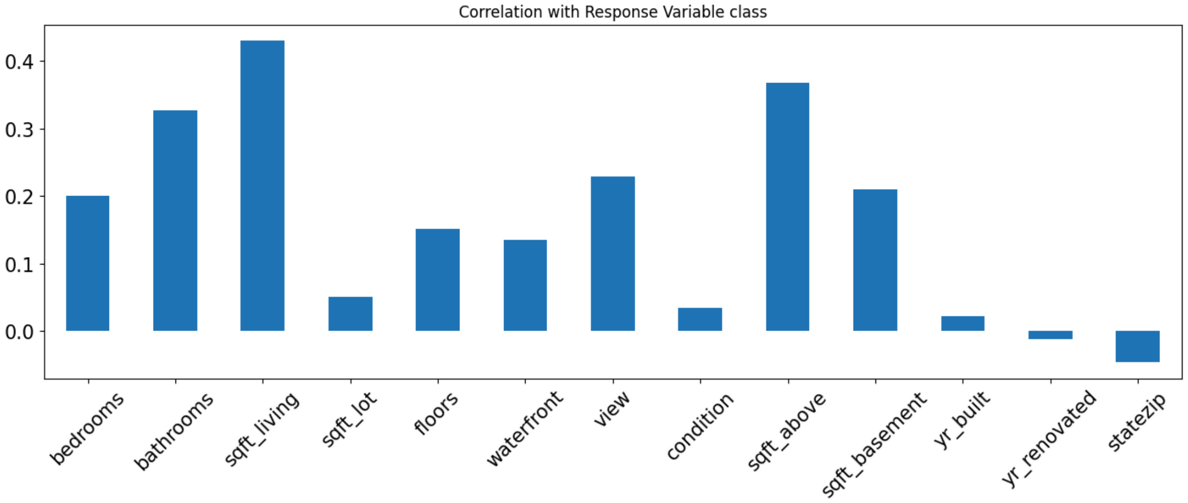
Out[11]:

```
                     0
        price  float64
     bedrooms  float64
    bathrooms  float64
  sqft_living    int64
     sqft_lot    int64
       floors  float64
   waterfront    int64
         view    int64
    condition    int64
   sqft_above    int64
sqft_basement    int64
     yr_built    int64
 yr_renovated    int64
     statezip    int64

dtype: object
```

In [12]:

```python
df1 = df.copy()
 #Correlation with Response Variable class
X = df1.drop(['price'],axis=1)
y = df1['price']
X.corrwith(y).plot.bar(
figsize = (16, 5), title = "Correlation with Response Variable class", fontsize = 1
rot = 45, grid = False)
plt.show()
```



In [13]:

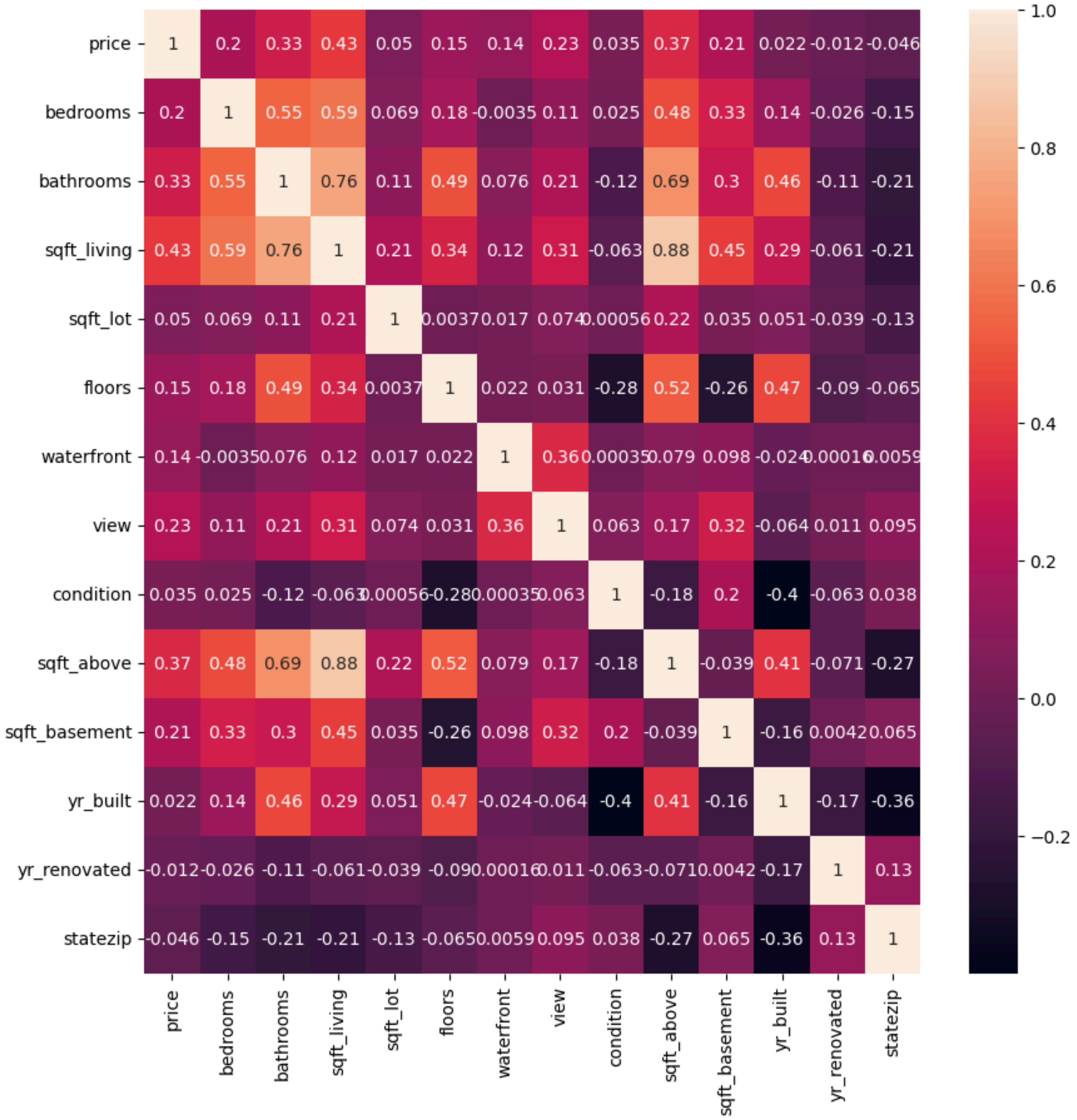
```
df.corr()
```

Out[13]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	
price	1.000000	0.200336	0.327110	0.430410	0.050451	0.151461	0.135648	0.2
bedrooms	0.200336	1.000000	0.545920	0.594884	0.068819	0.177895	-0.003483	0.1
bathrooms	0.327110	0.545920	1.000000	0.761154	0.107837	0.486428	0.076232	0.2
sqft_living	0.430410	0.594884	0.761154	1.000000	0.210538	0.344850	0.117616	0.3
sqft_lot	0.050451	0.068819	0.107837	0.210538	1.000000	0.003750	0.017241	0.0
floors	0.151461	0.177895	0.486428	0.344850	0.003750	1.000000	0.022024	0.0
waterfront	0.135648	-0.003483	0.076232	0.117616	0.017241	0.022024	1.000000	0.3
view	0.228504	0.111028	0.211960	0.311009	0.073907	0.031211	0.360935	1.0
condition	0.034915	0.025080	-0.119994	-0.062826	0.000558	-0.275013	0.000352	0.0
sqft_above	0.367570	0.484705	0.689918	0.876443	0.216455	0.522814	0.078911	0.1
sqft_basement	0.210427	0.334165	0.298020	0.447206	0.034842	-0.255510	0.097501	0.3
yr_built	0.021857	0.142461	0.463498	0.287775	0.050706	0.467481	-0.023563	-0.0
yr_renovated	-0.011928	-0.026228	-0.108796	-0.061498	-0.039293	-0.089938	0.000165	0.0
statezip	-0.046052	-0.153443	-0.206231	-0.210891	-0.133509	-0.064999	0.005937	0.0

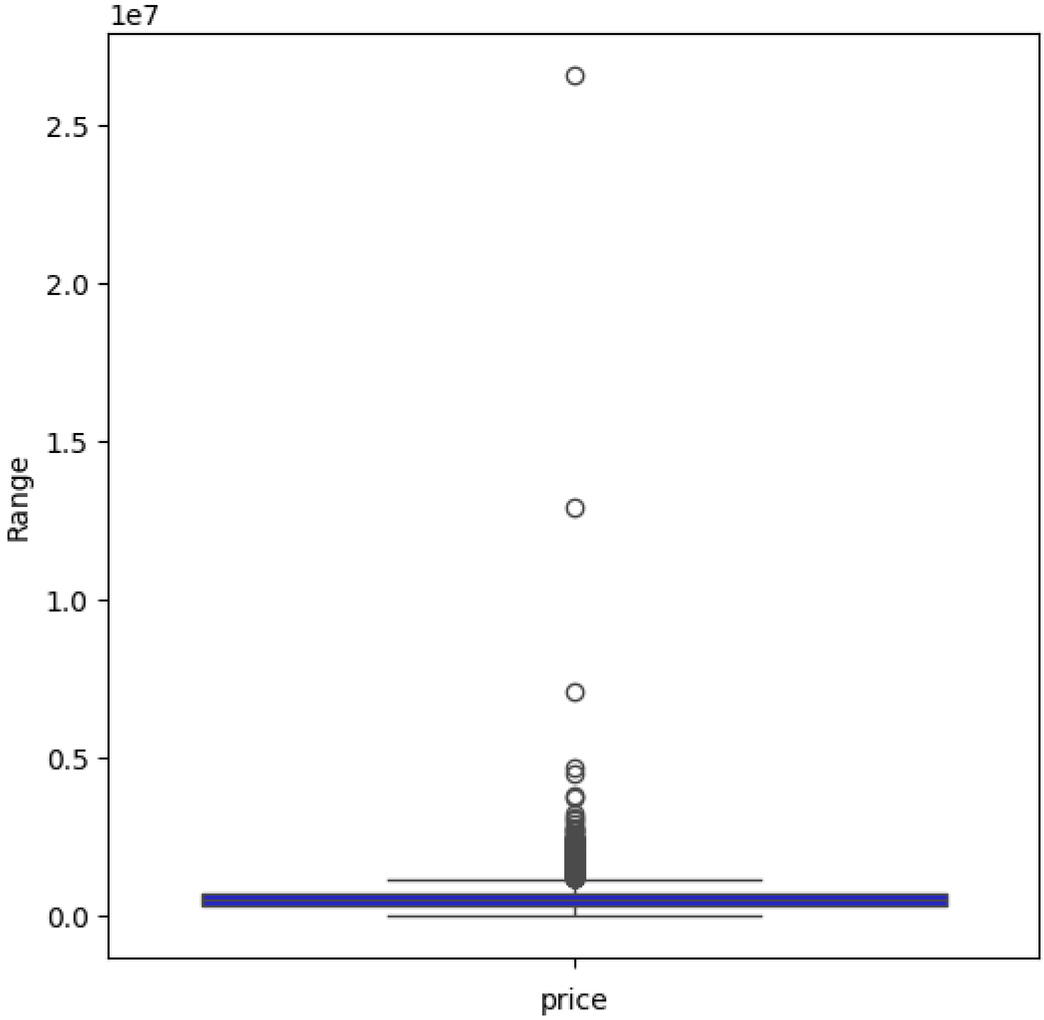
In [14]:

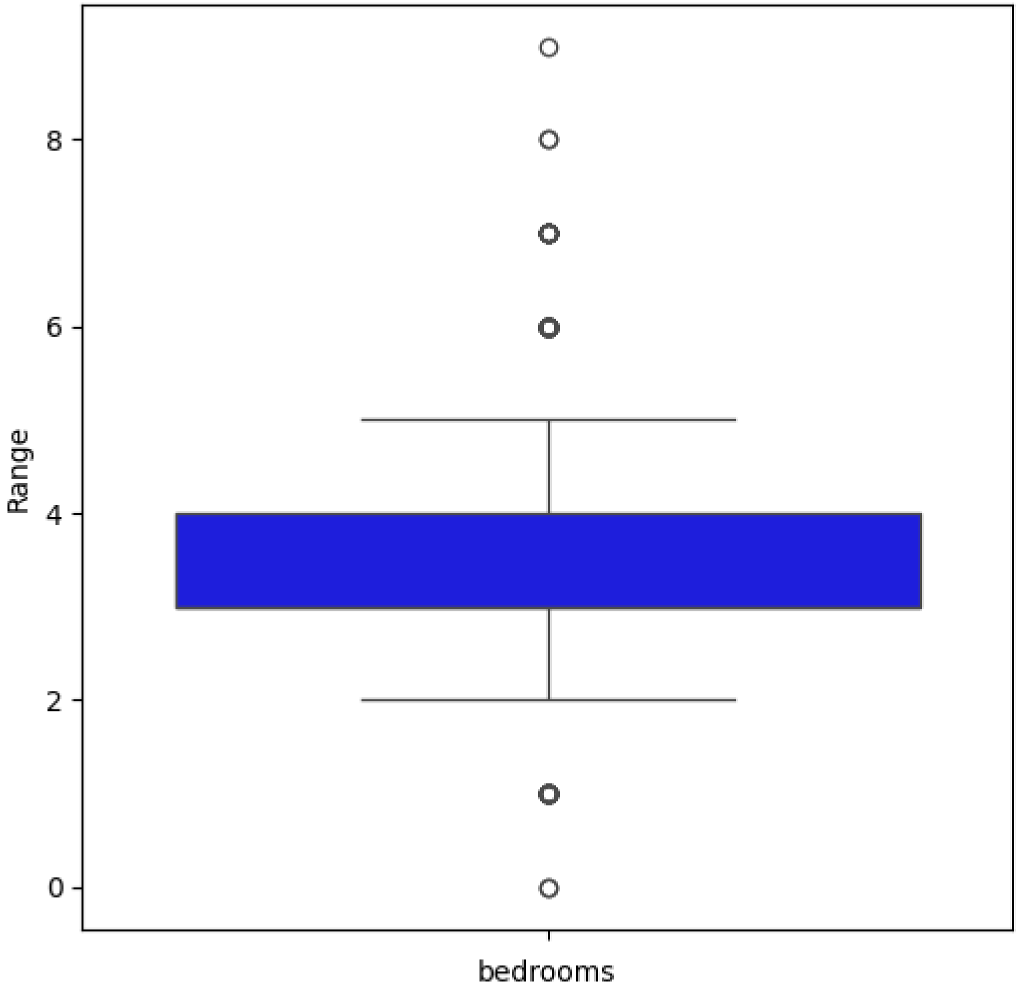
```
plt.figure(figsize=(10,10))
sns.heatmap(df.corr(),annot=True)
plt.show()
```

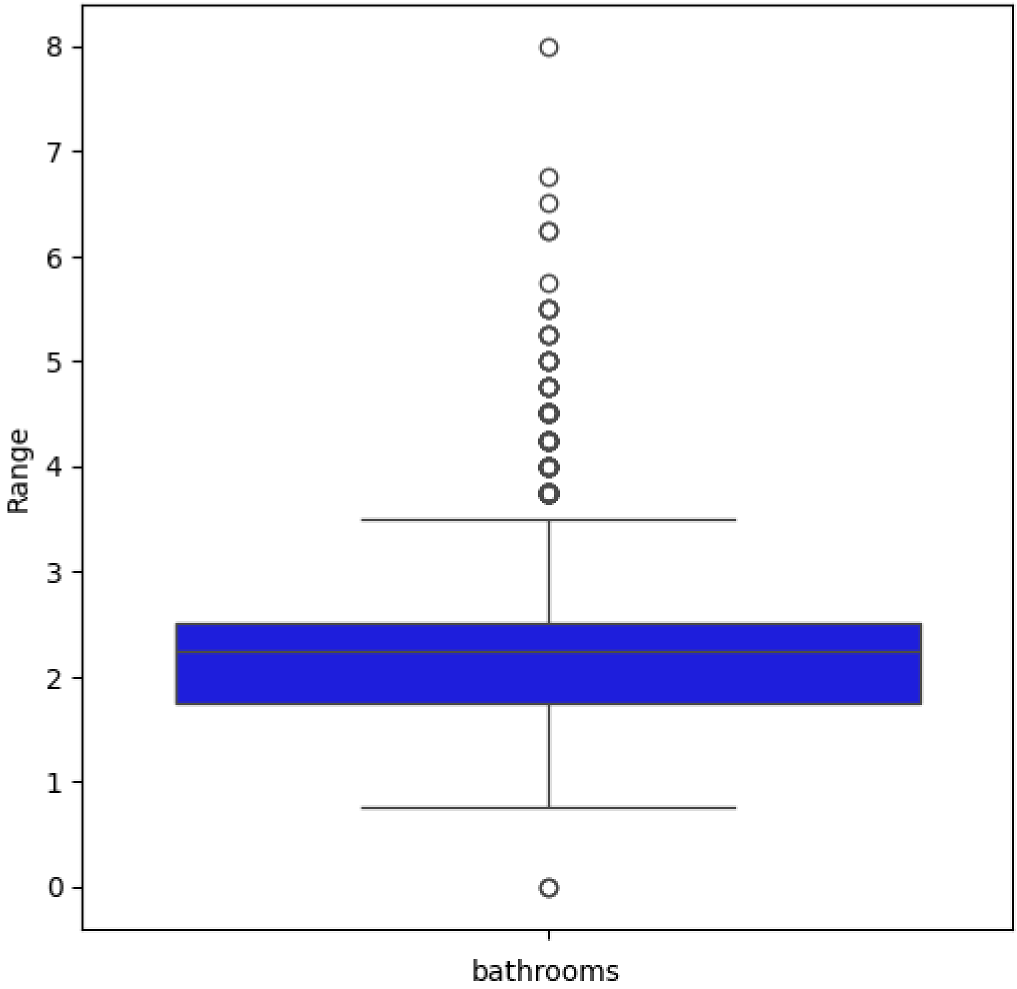


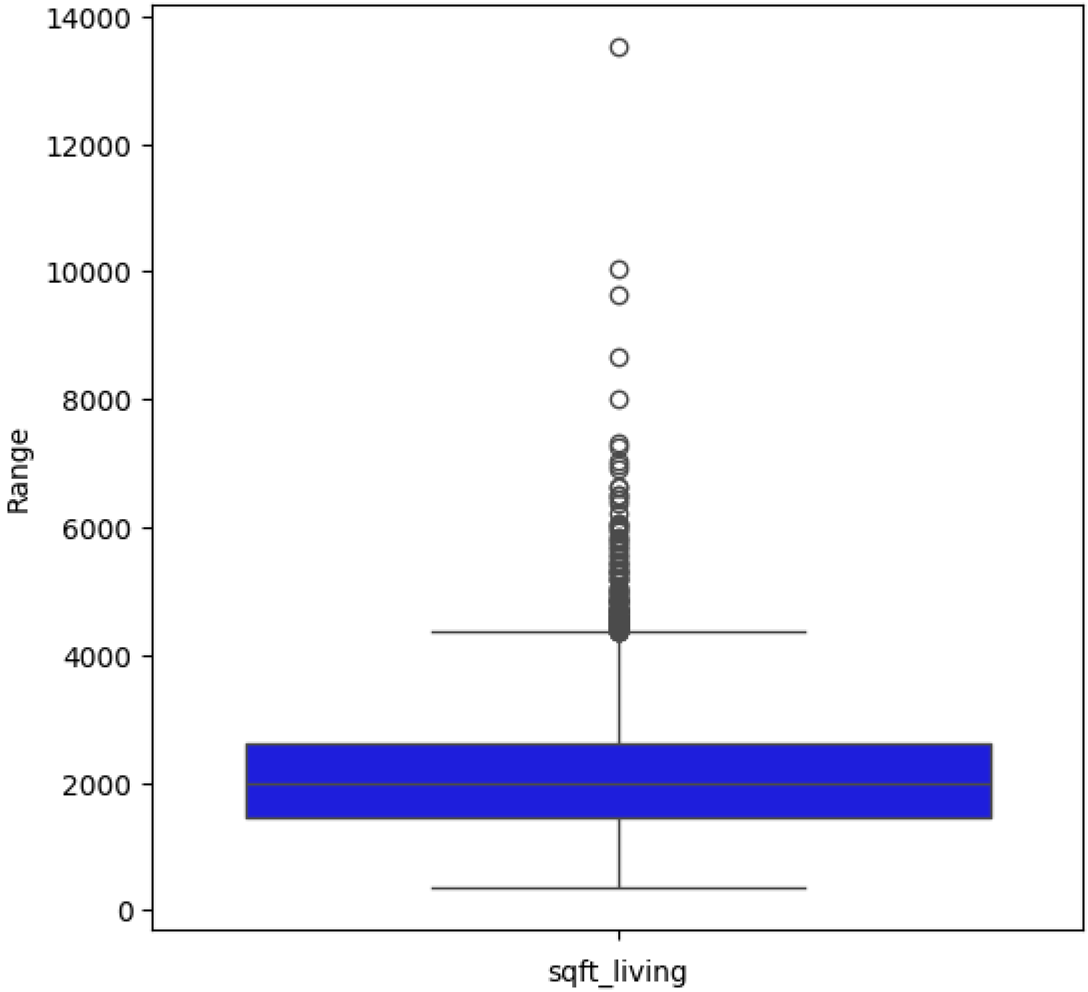
Removal of Outliers

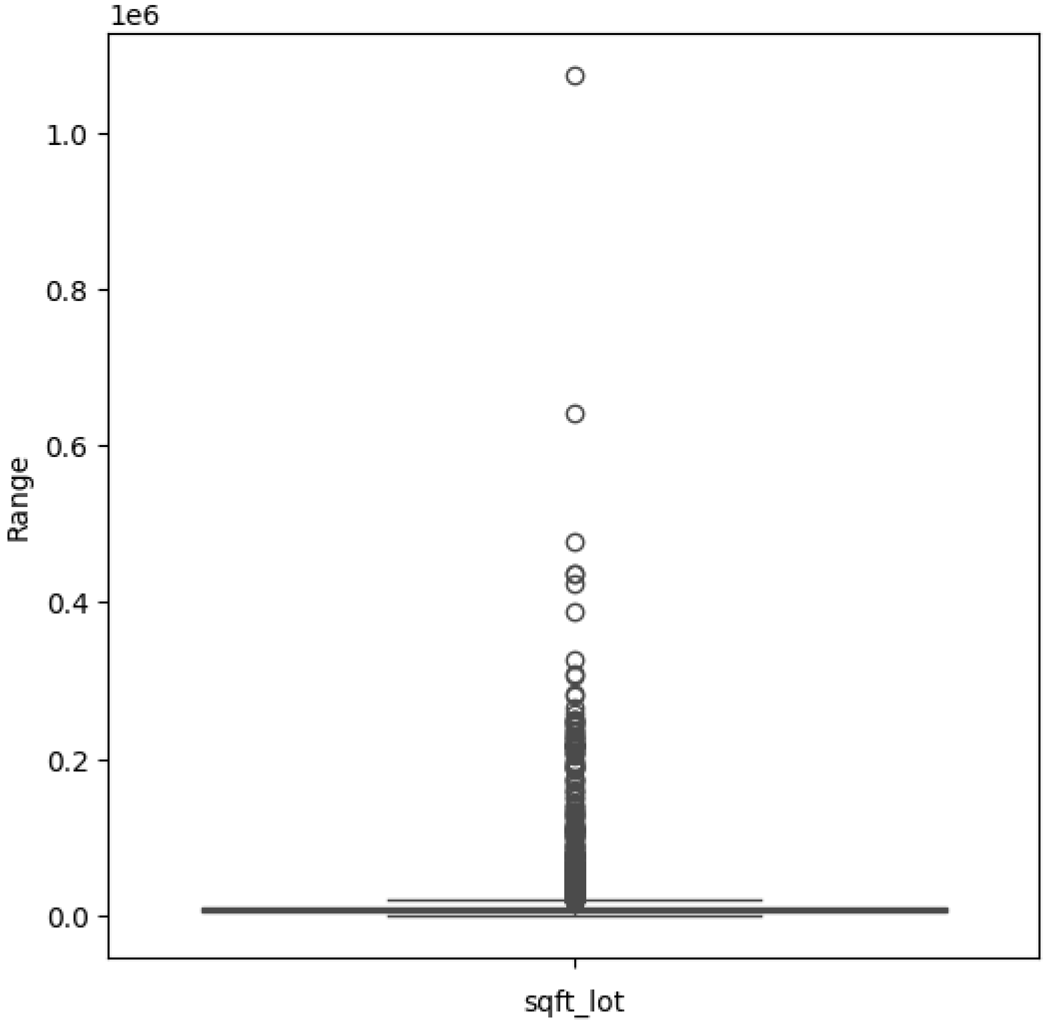
In [15]:
```
for i in df.columns:
 plt.figure(figsize=(6,6))
 sns.boxplot(df[i],color='blue')
 plt.xlabel(i)
 plt.ylabel('Range')
```

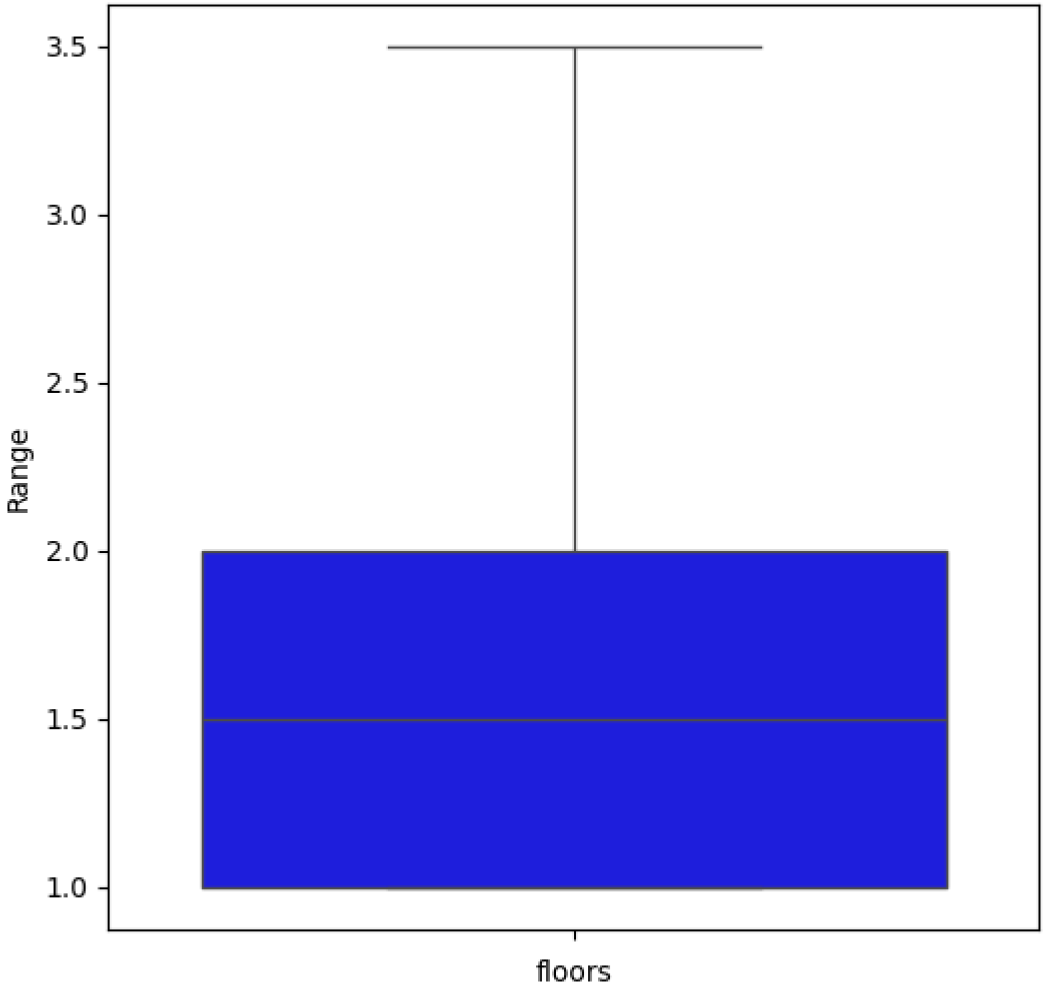

1e7
2.5
2.0
1.5
1.0
0.5
0.0
Range
price

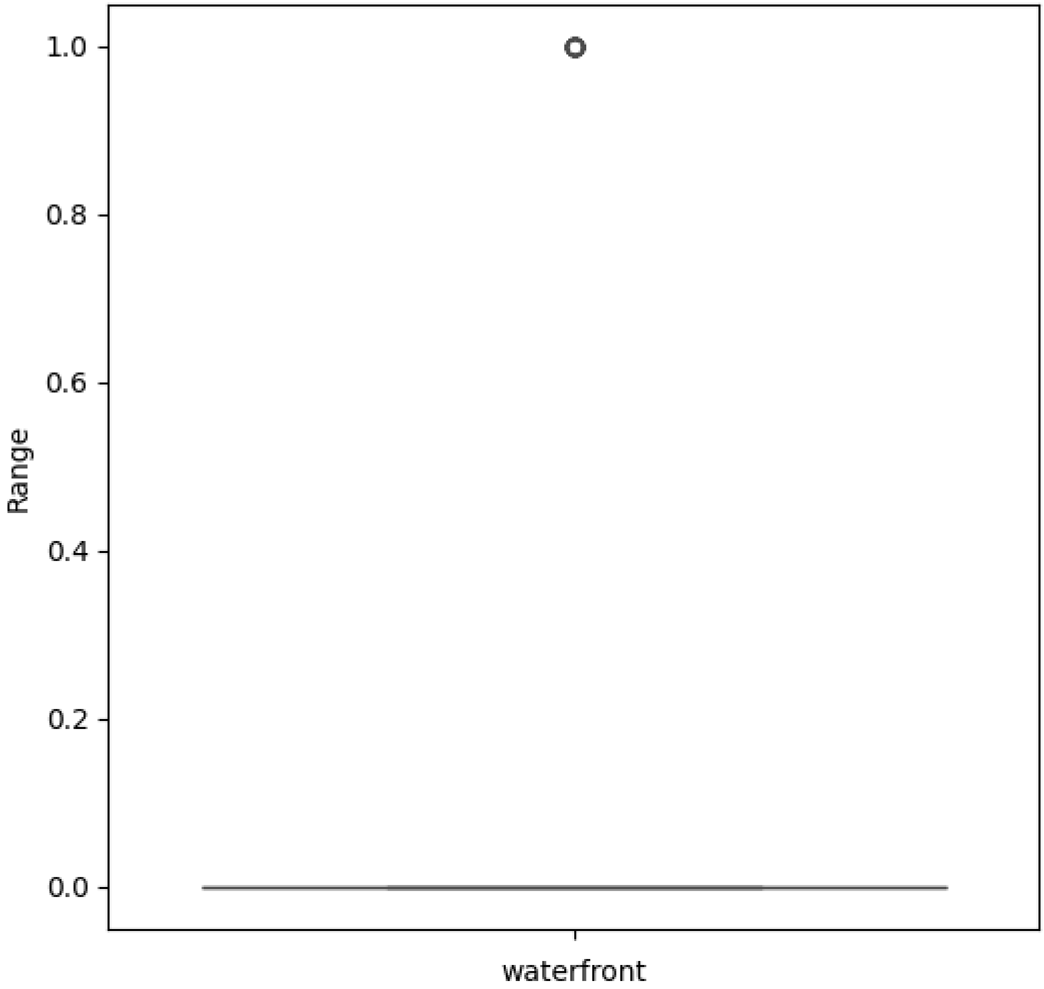

8
6
4
2
0
Range
bedrooms

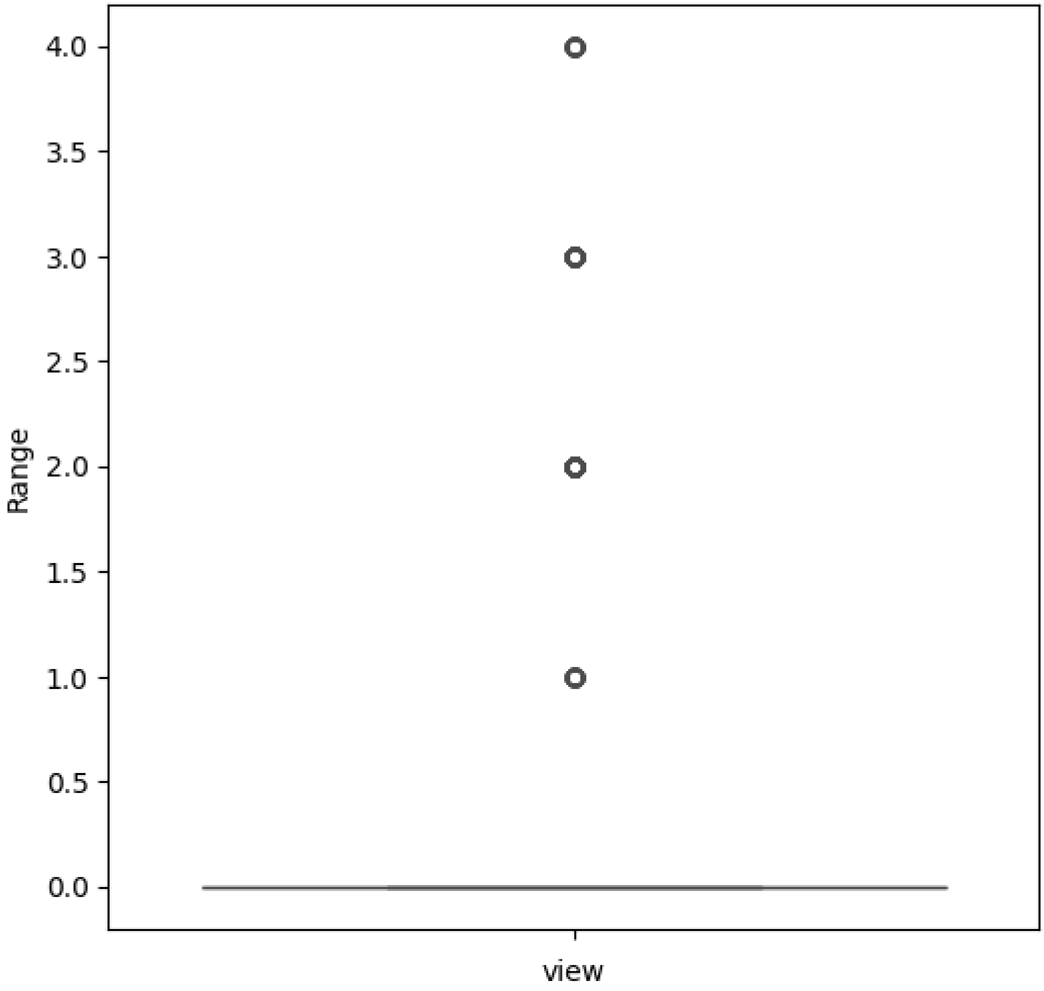

Range
0
1
2
3
4
5
6
7
8
bathrooms

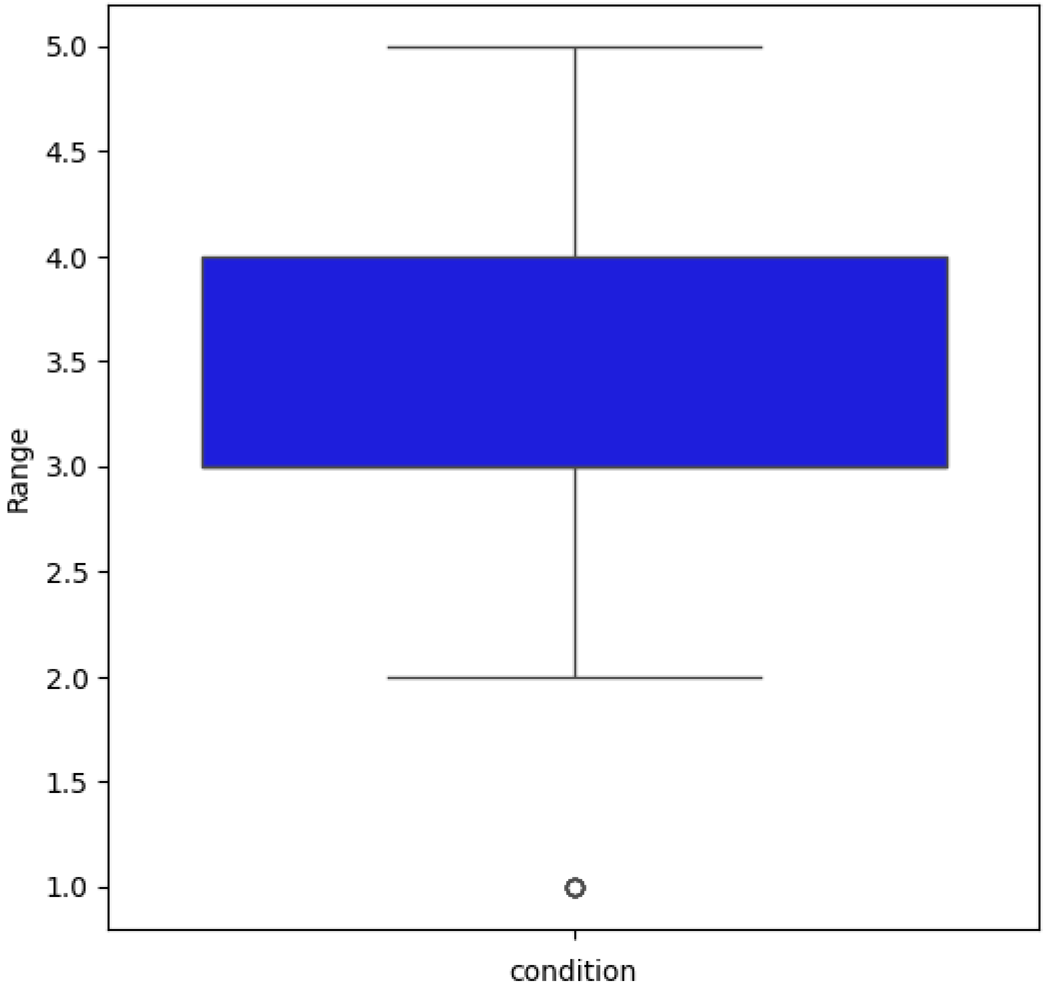

Range
0
2000
4000
6000
8000
10000
12000
14000
sqft_living

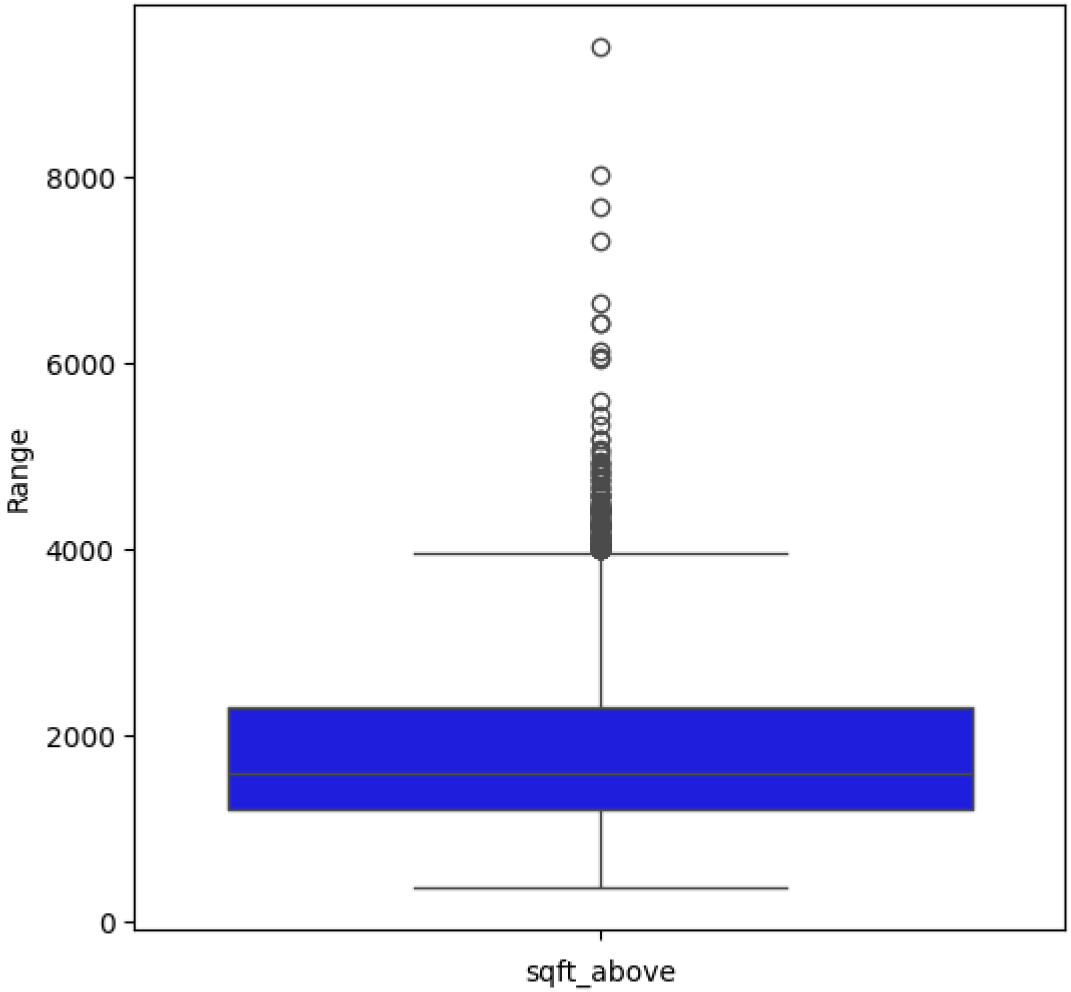

1e6
1.0
0.8
0.6
0.4
0.2
0.0
Range
sqft_lot

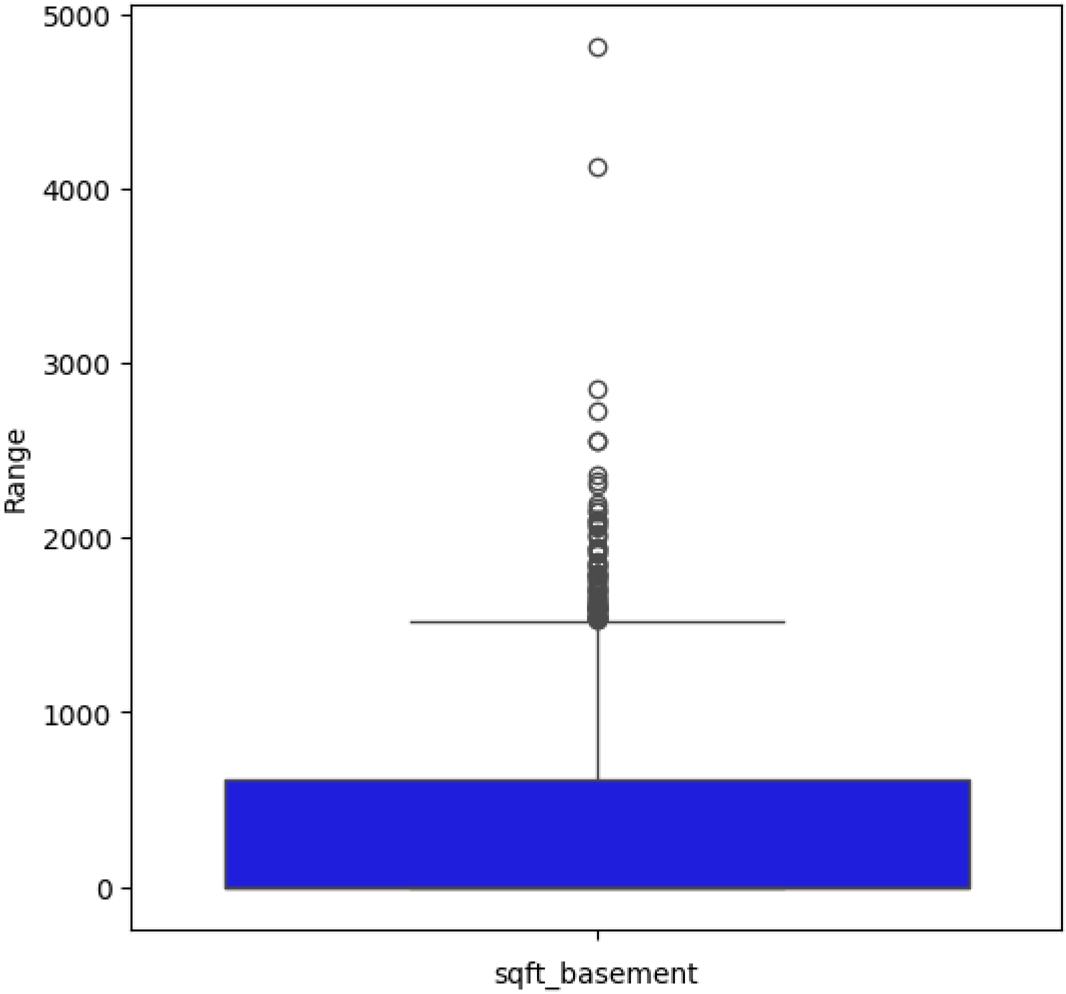

3.5
3.0
2.5
2.0
1.5
1.0
Range
floors

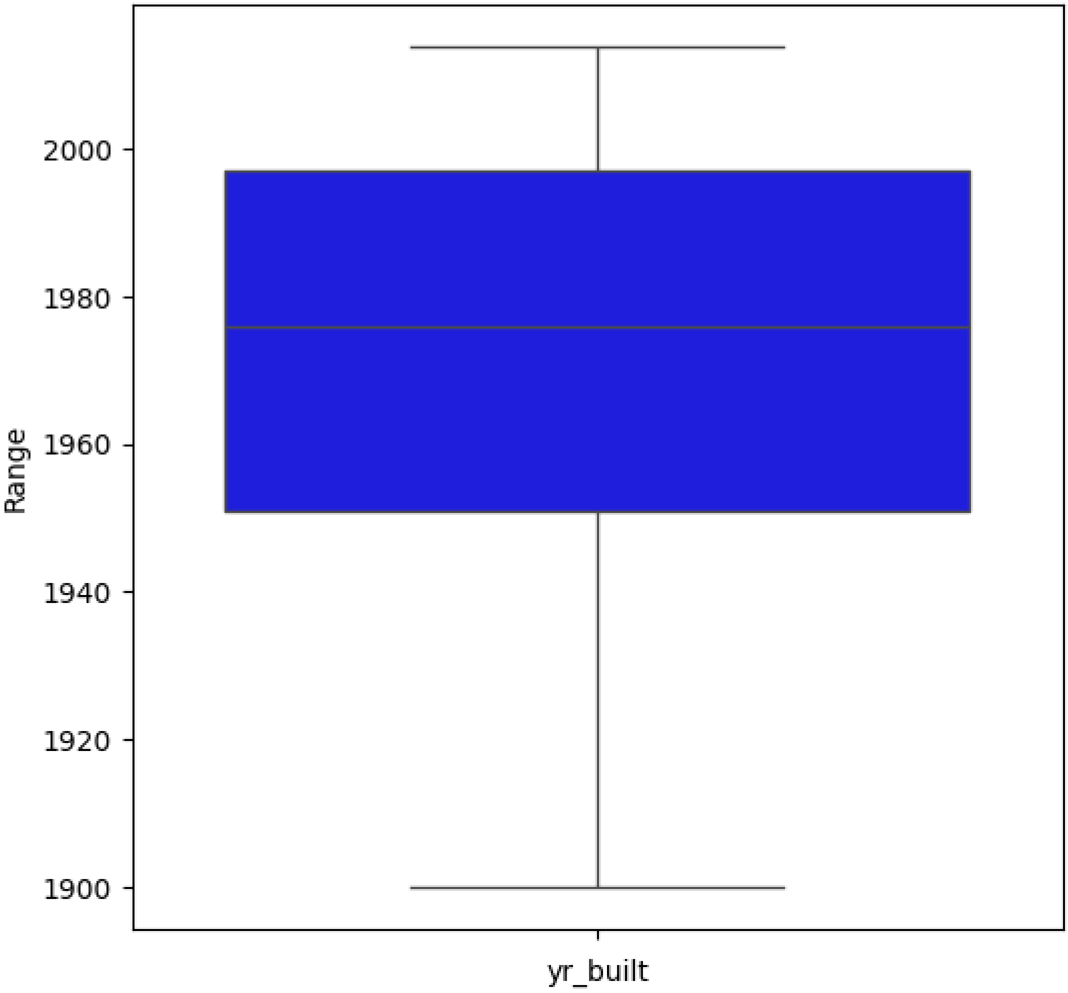

1.0
0.8
0.6
0.4
0.2
0.0
Range
waterfront

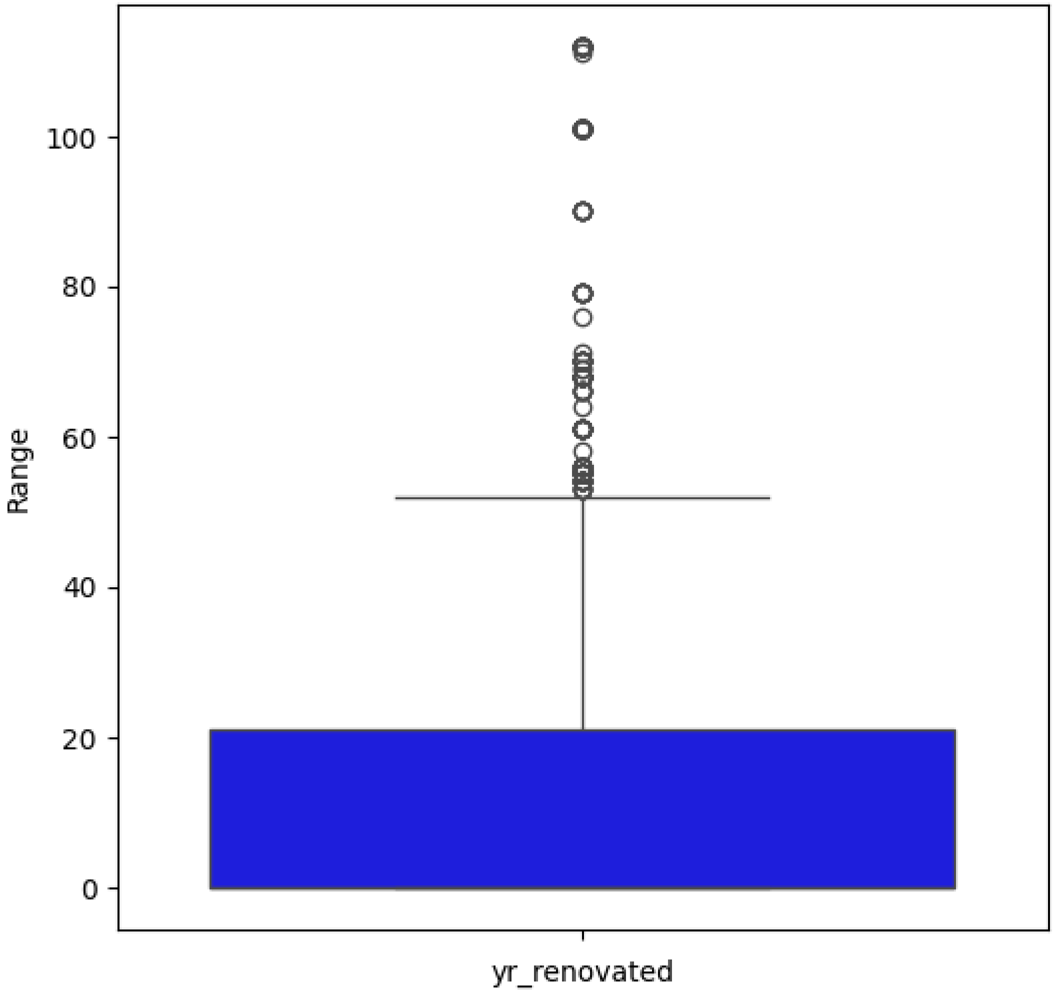

4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Range
view

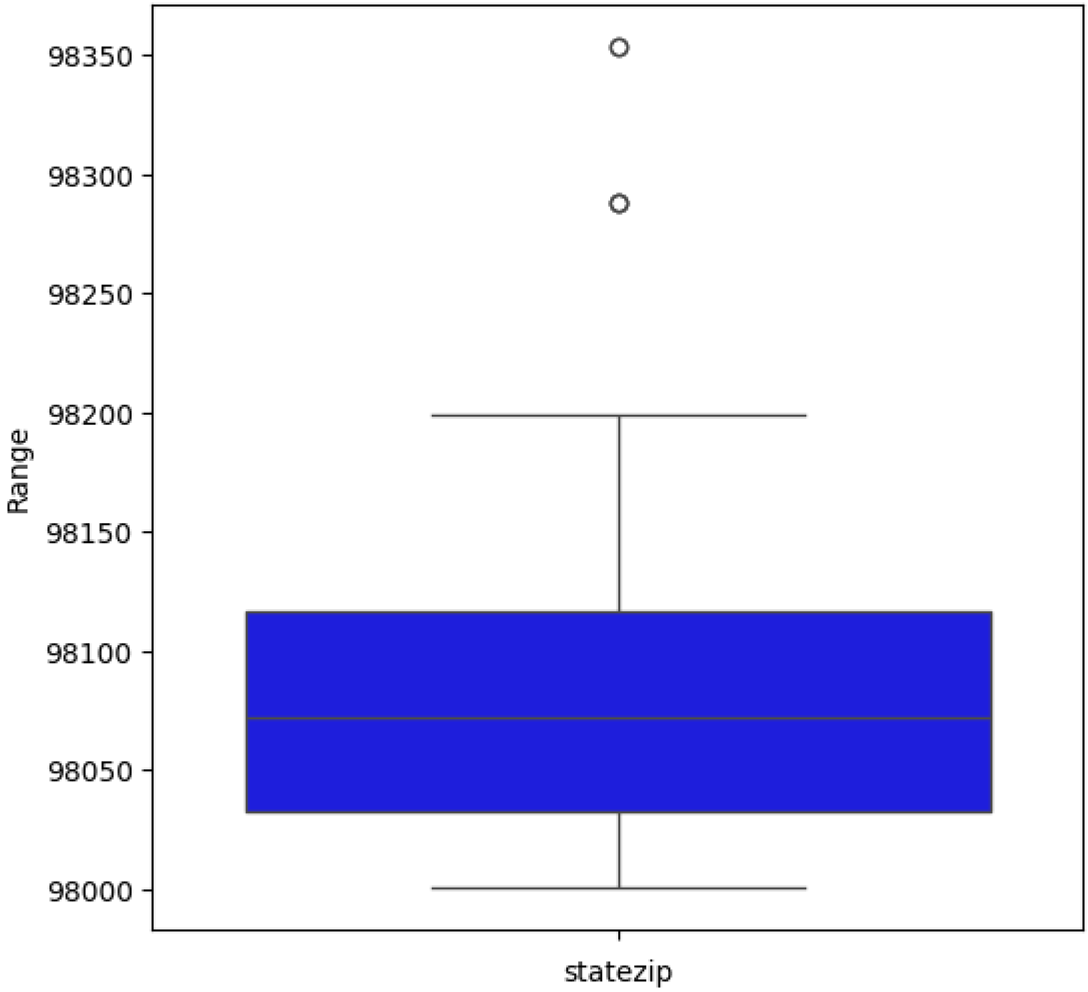

5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
Range
condition


8000
6000
4000
2000
0
Range
sqft_above


5000
4000
3000
2000
1000
0
Range
sqft_basement


2000
1980
1960
1940
1920
1900
Range
yr_built


Range
100
80
60
40
20
0
yr_renovated


Range
98350
98300
98250
98200
98150
98100
98050
98000
statezip

In [16]:

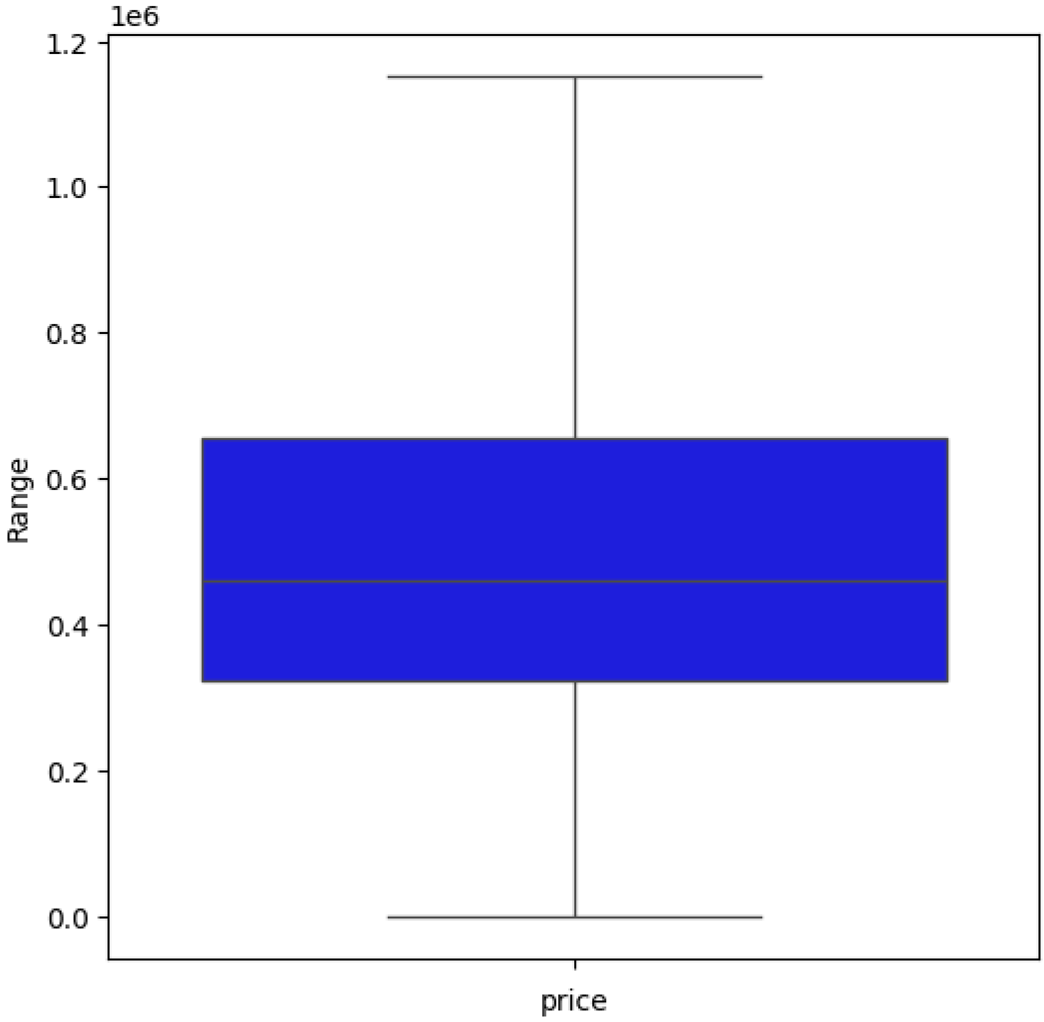
```
df.drop(['waterfront','view'],axis=1,inplace=True)
```

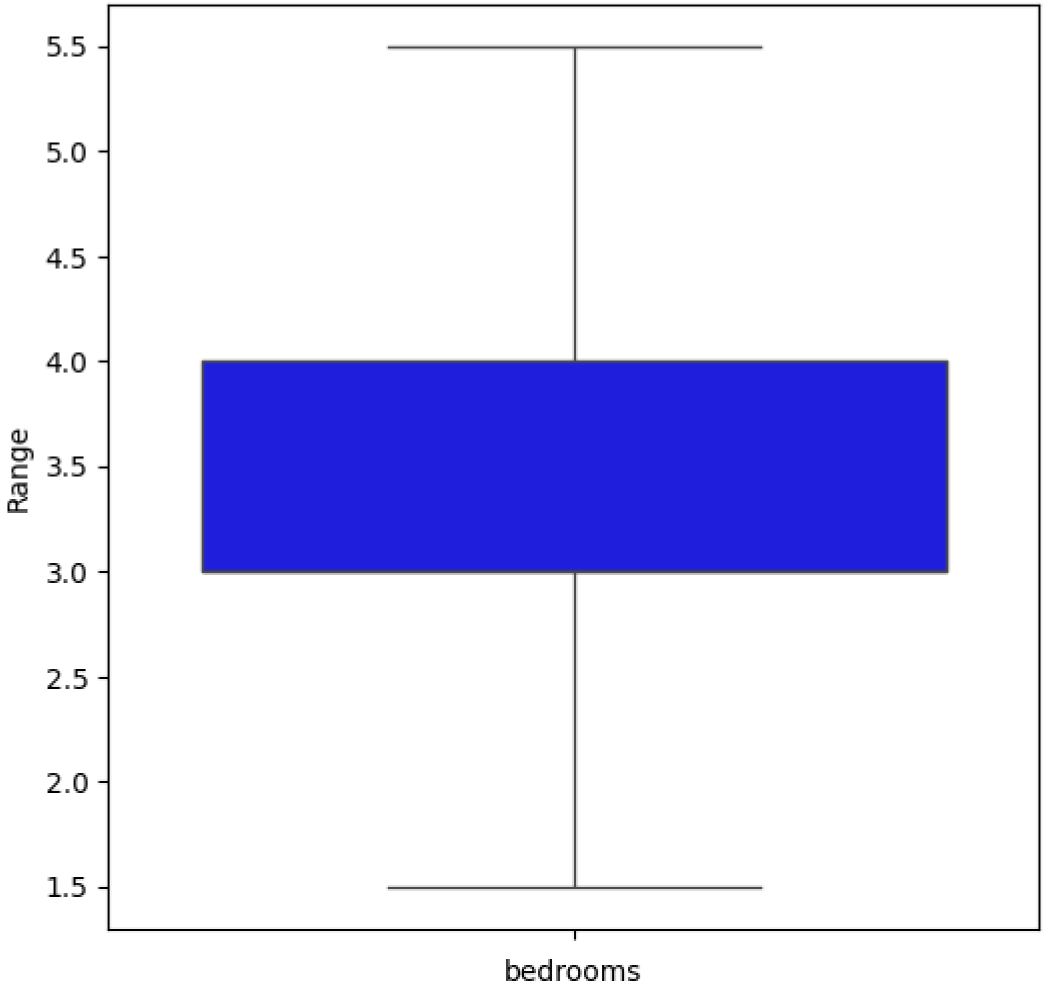
In [17]:

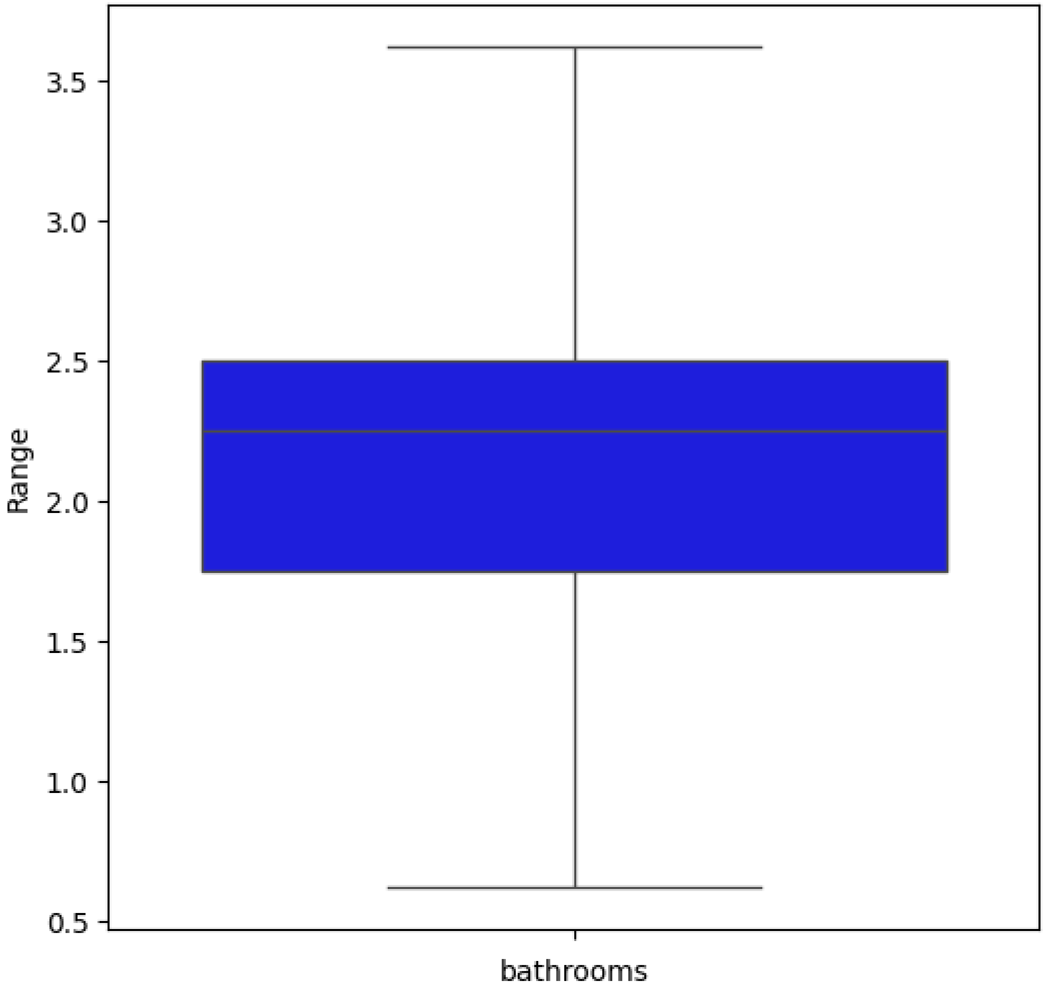
```
for i in df.columns:
  def iqr_method(df,variables):
    q1=df[variables].quantile(0.25)
    q3=df[variables].quantile(0.75)
    iqr=q3-q1
    upper=q3+(1.5*iqr)
    lower=q1-(1.5*iqr)
    return lower,upper
  lower_lim,upper_lim=iqr_method(df,i)
  df[i]=np.where(df[i]>upper_lim,upper_lim,np.where(df[i]<lower_lim,lower_lim,df[i]
```

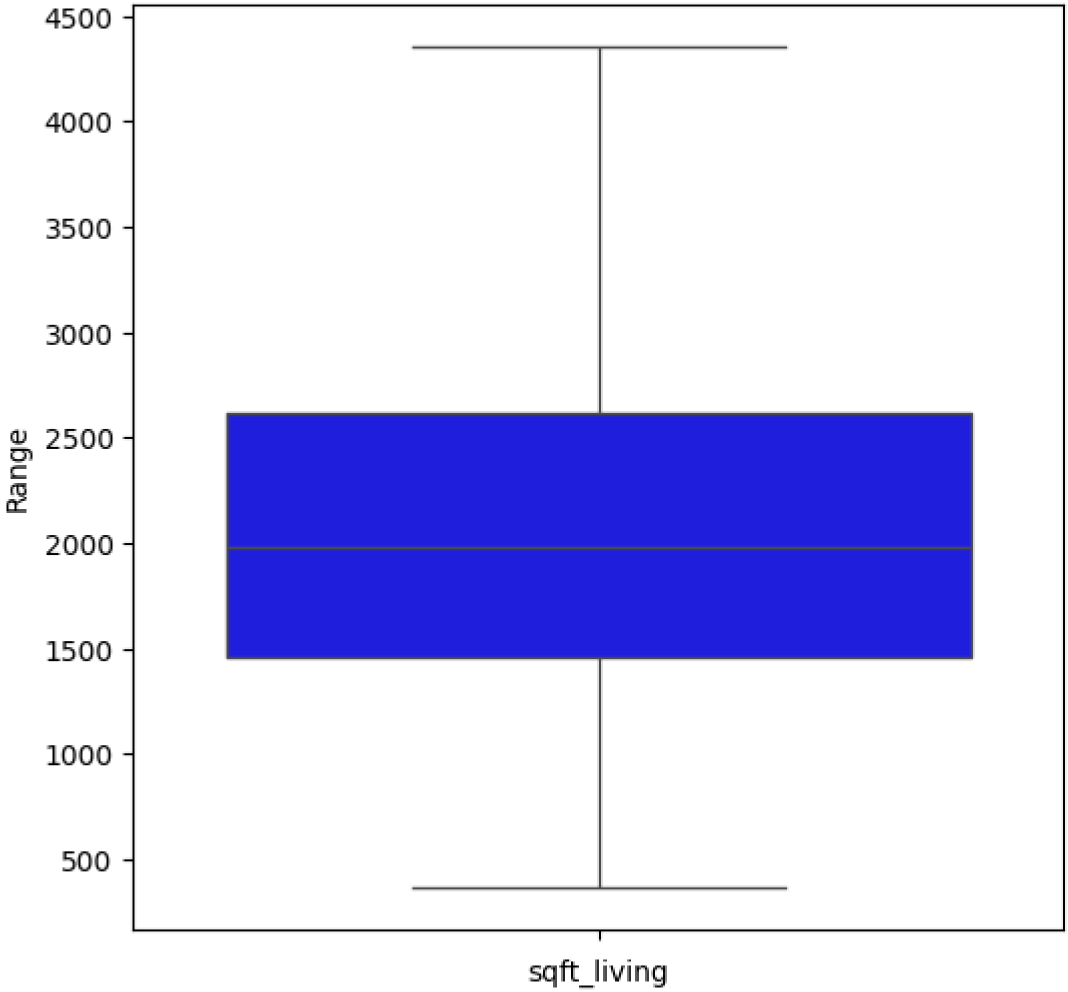
In [18]:

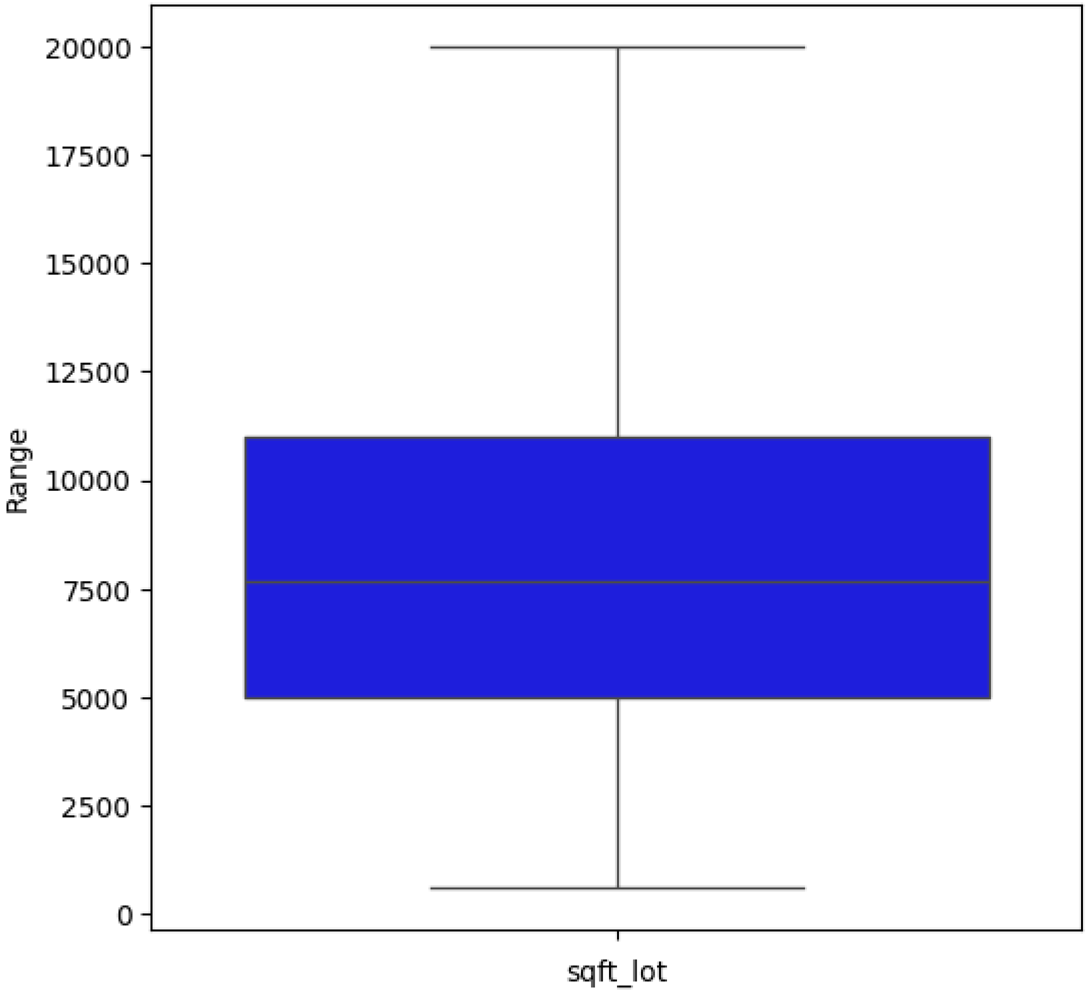
```
for i in df.columns:
 plt.figure(figsize=(6,6))
 sns.boxplot(df[i],color='blue')
 plt.xlabel(i)
 plt.ylabel('Range')
```

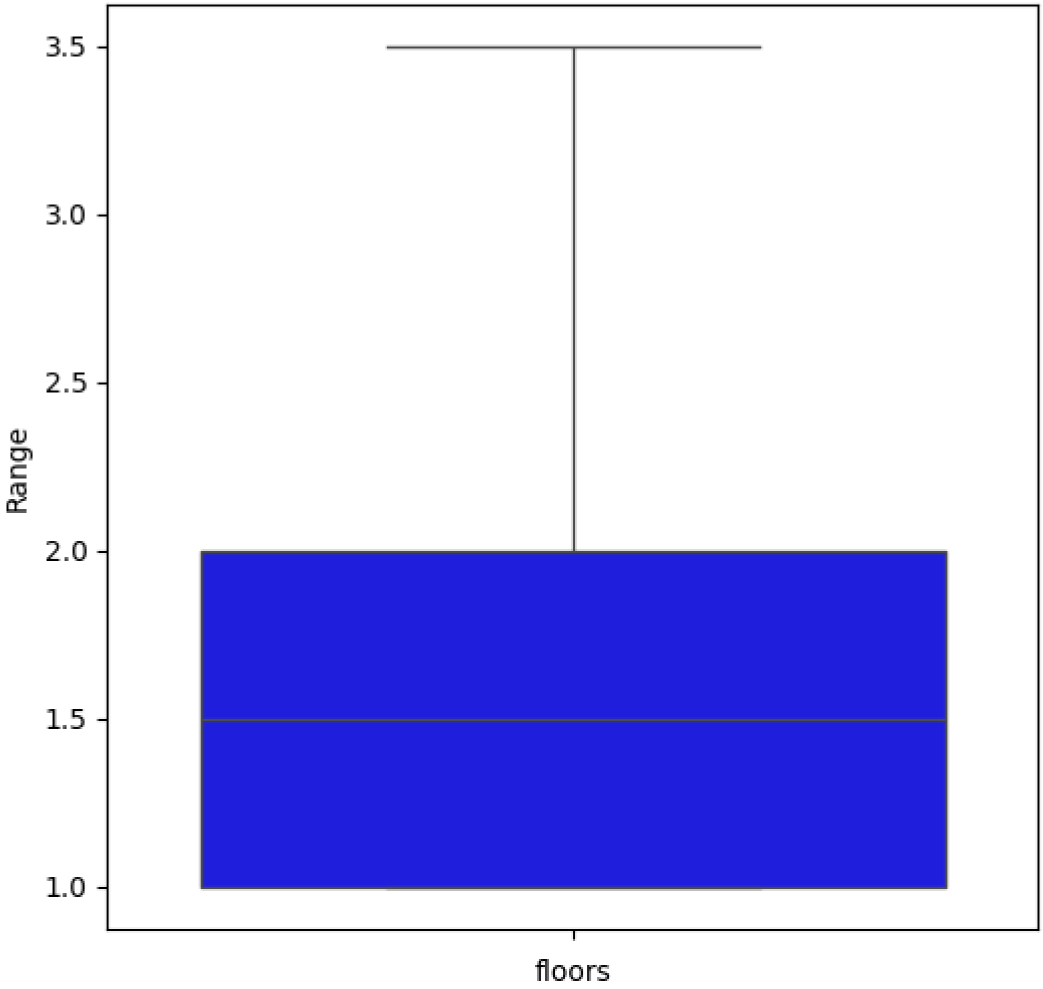


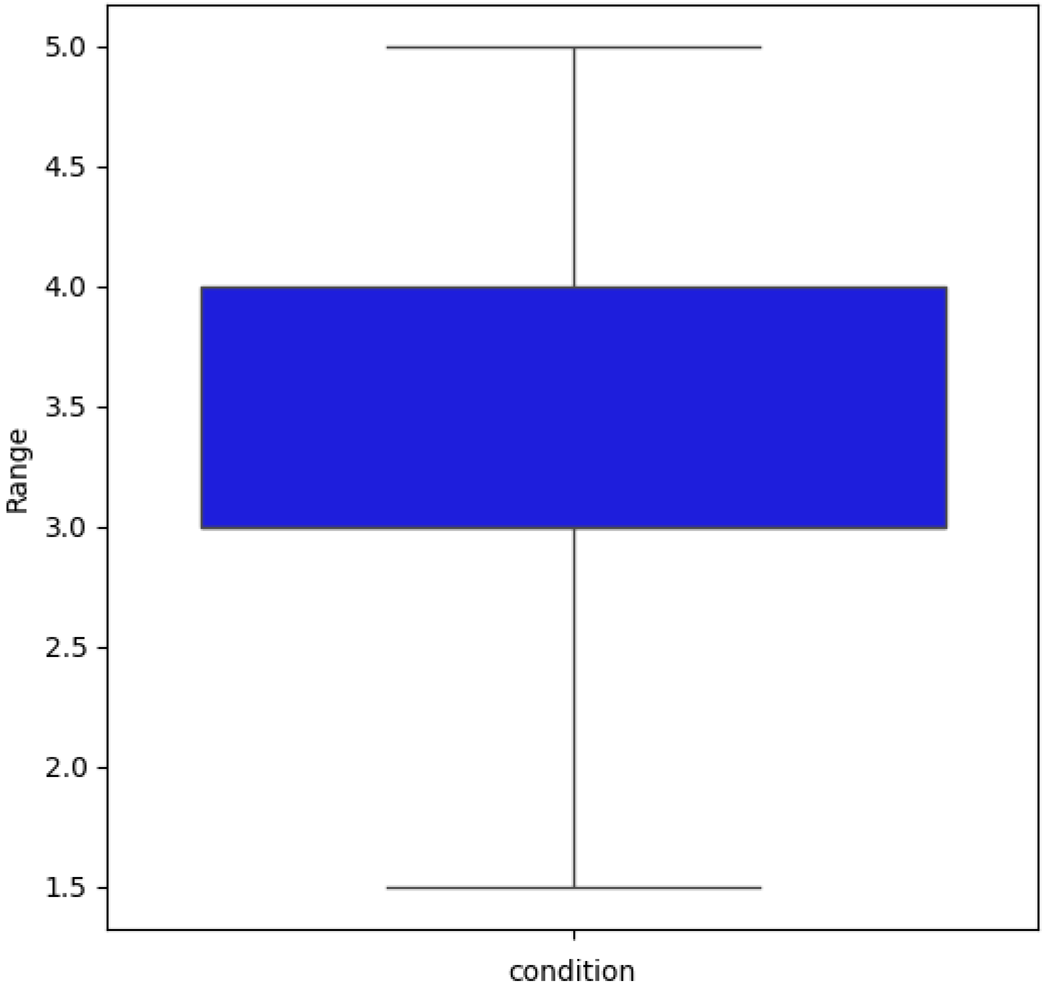

5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
Range
bedrooms

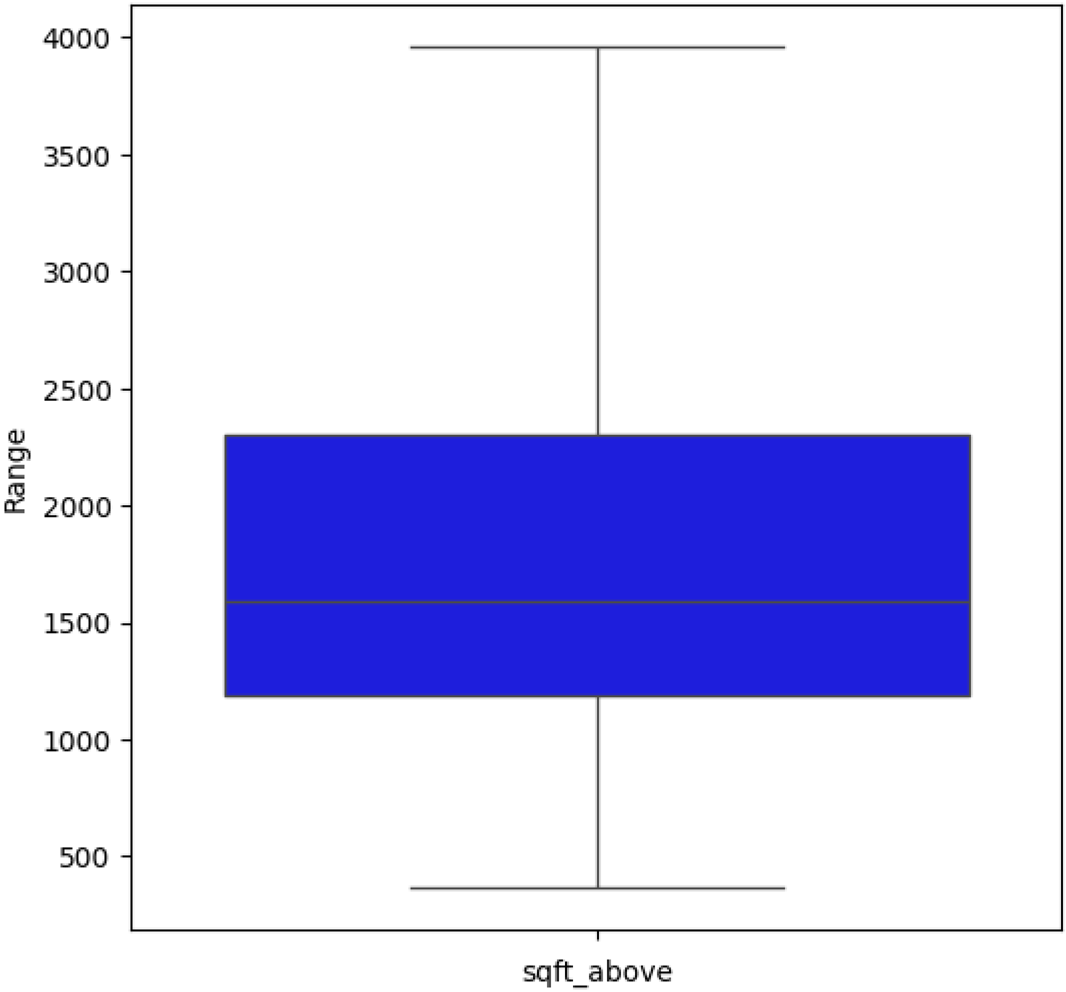

3.5
3.0
2.5
2.0
1.5
1.0
0.5
Range
bathrooms

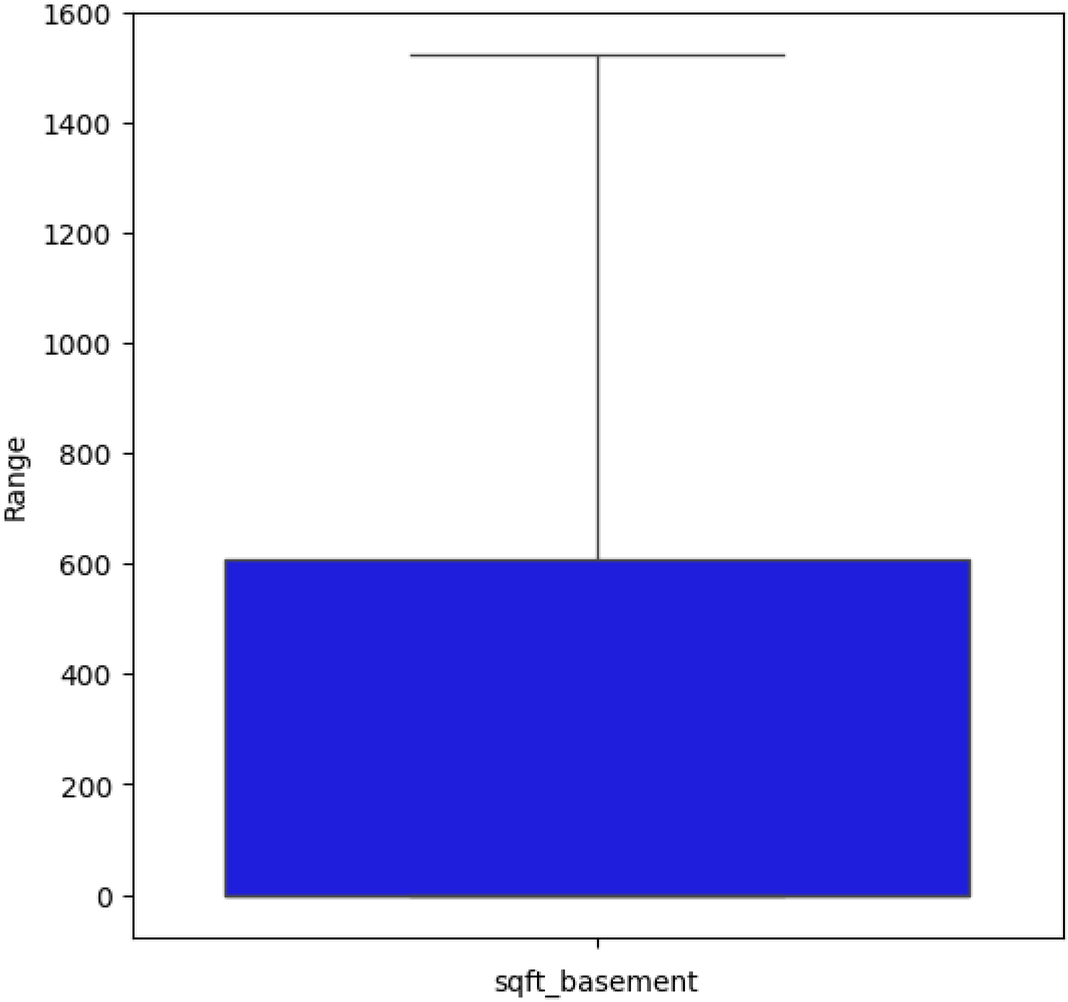

4500
4000
3500
3000
2500
2000
1500
1000
500
Range
sqft_living

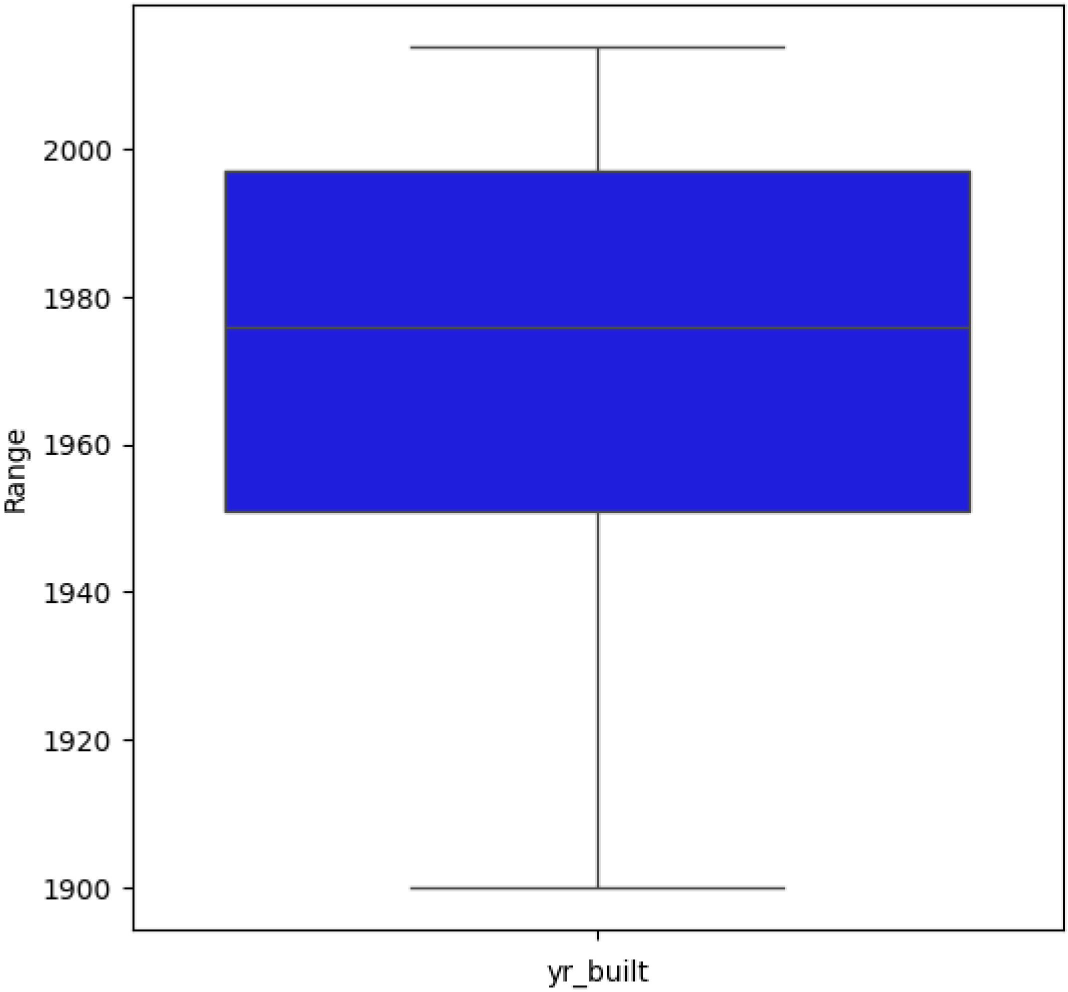

20000
17500
15000
12500
10000
7500
5000
2500
0
Range
sqft_lot

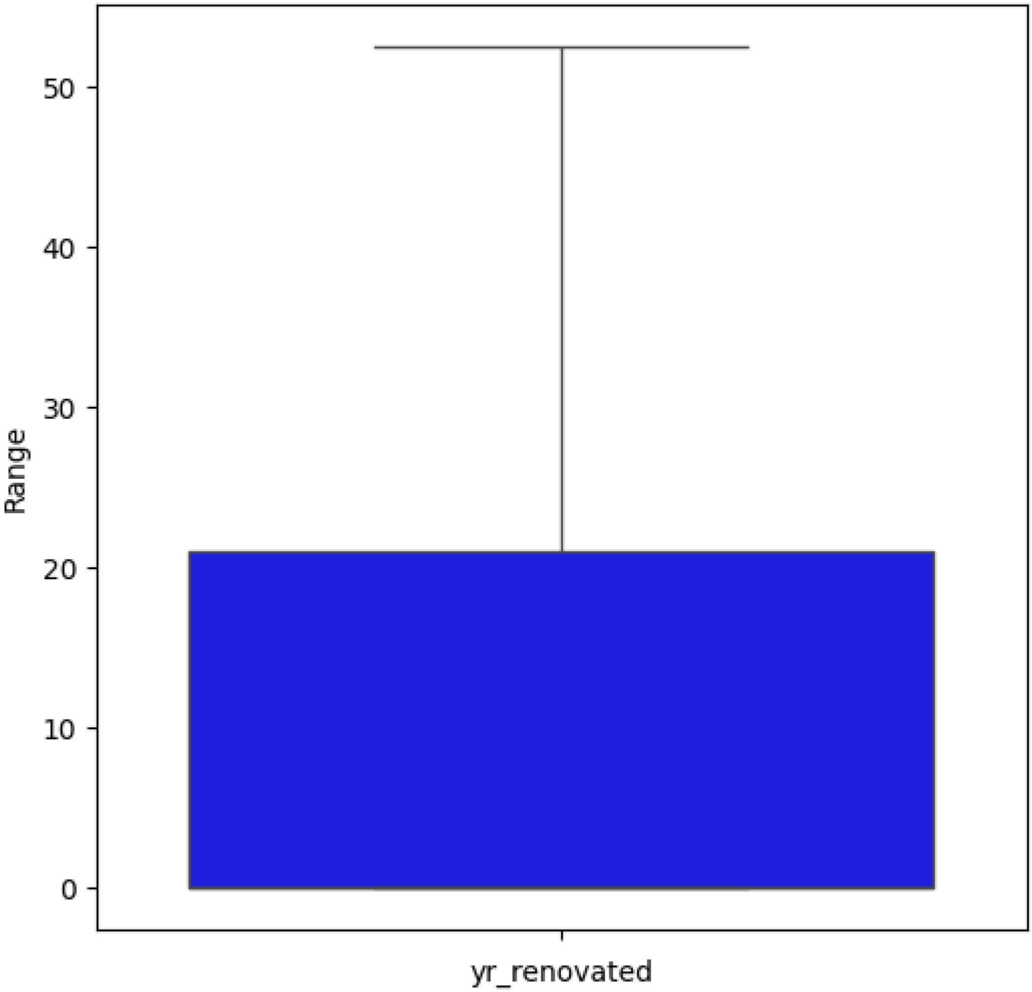

3.5
3.0
2.5
2.0
1.5
1.0
Range
floors

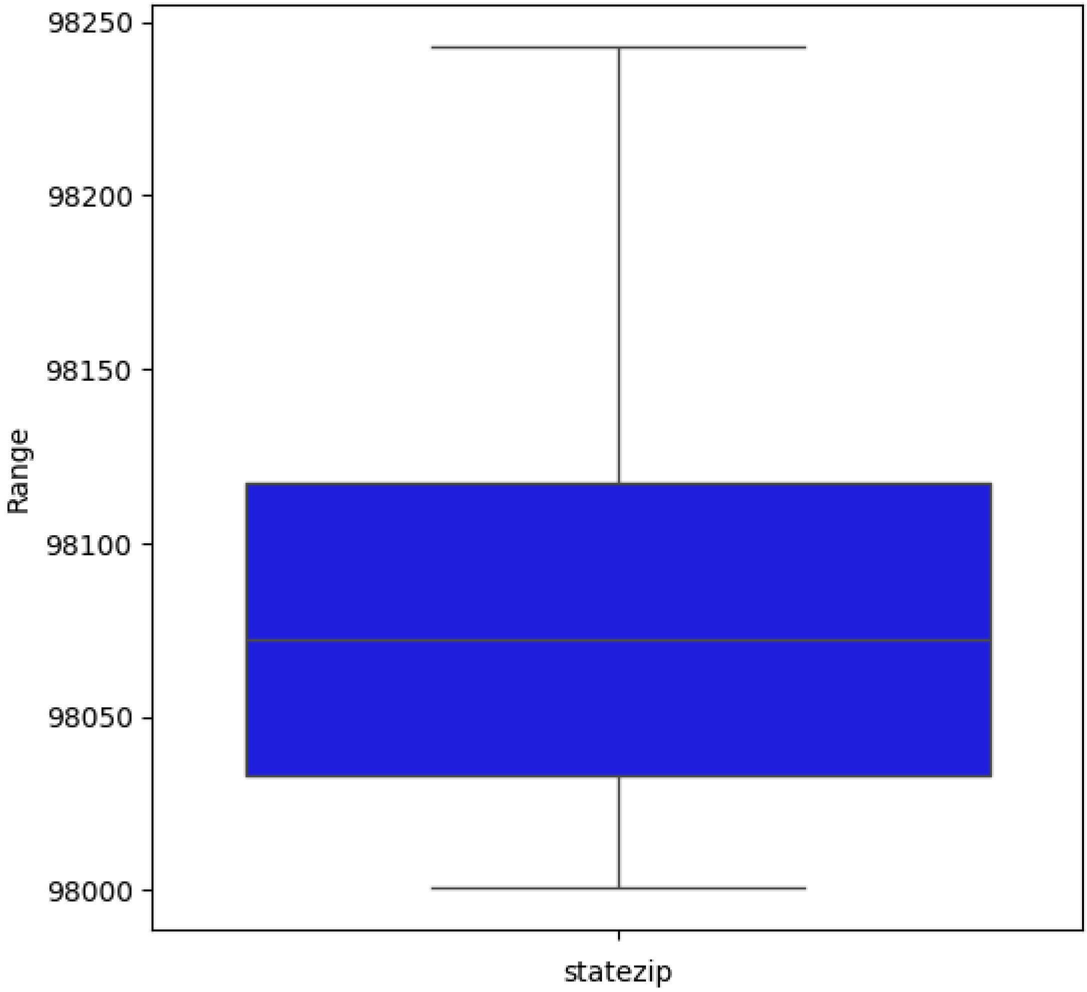

5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
Range
condition


4000
3500
3000
2500
2000
1500
1000
500
Range
sqft_above


1600
1400
1200
1000
800
600
400
200
0
Range
sqft_basement


2000
1980
1960
1940
1920
1900
Range
yr_built


50
40
30
20
10
0
Range
yr_renovated



In [19]:
```
x=df.drop(['price'],axis=1)
y=df['price']
x
```

Out[19]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	condition	sqft_above	sqft_basement	y
0	3.0	1.50	1340.0	7912.0	1.5	3.0	1340.0	0.0	
1	5.0	2.50	3650.0	9050.0	2.0	5.0	3370.0	280.0	
2	3.0	2.00	1930.0	11947.0	1.0	4.0	1930.0	0.0	
3	3.0	2.25	2000.0	8030.0	1.0	4.0	1000.0	1000.0	
4	4.0	2.50	1940.0	10500.0	1.0	4.0	1140.0	800.0	
...	...	...	...	...	...	...	...	...	
4595	3.0	1.75	1510.0	6360.0	1.0	4.0	1510.0	0.0	
4596	3.0	2.50	1460.0	7573.0	2.0	3.0	1460.0	0.0	
4597	3.0	2.50	3010.0	7014.0	2.0	3.0	3010.0	0.0	
4598	4.0	2.00	2090.0	6630.0	1.0	3.0	1070.0	1020.0	
4599	3.0	2.50	1490.0	8102.0	2.0	4.0	1490.0	0.0	

4600 rows × 11 columns

In [20]:
```
y
```

Out[20]:

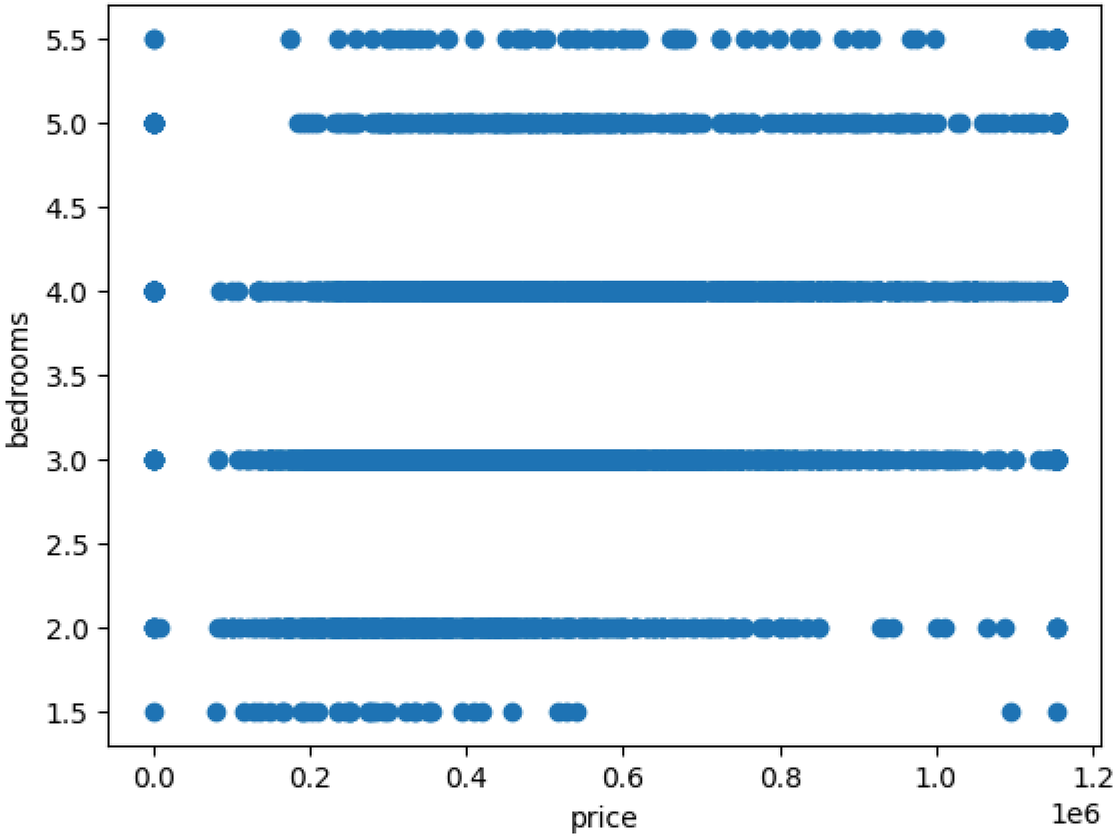
	price
0	3.130000e+05
1	1.153094e+06
2	3.420000e+05
3	4.200000e+05
4	5.500000e+05
...	...
4595	3.081667e+05
4596	5.343333e+05
4597	4.169042e+05
4598	2.034000e+05
4599	2.206000e+05

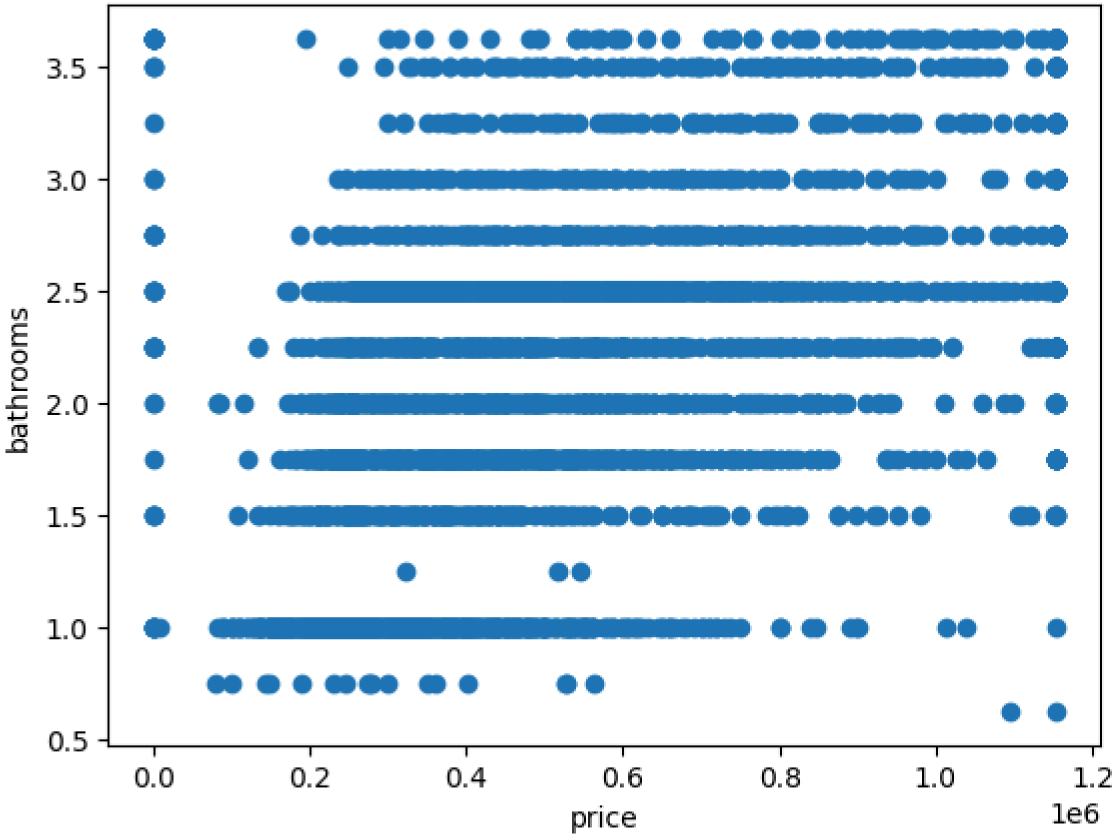
4600 rows × 1 columns

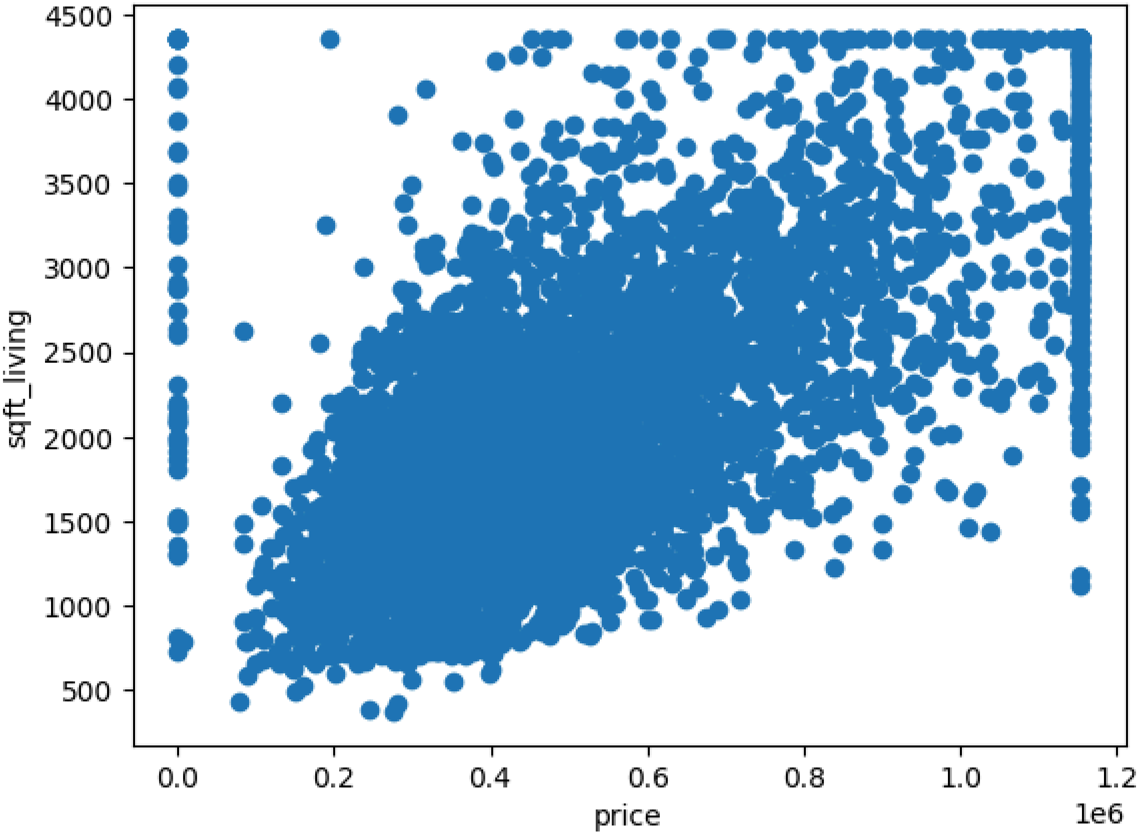
dtype: float64

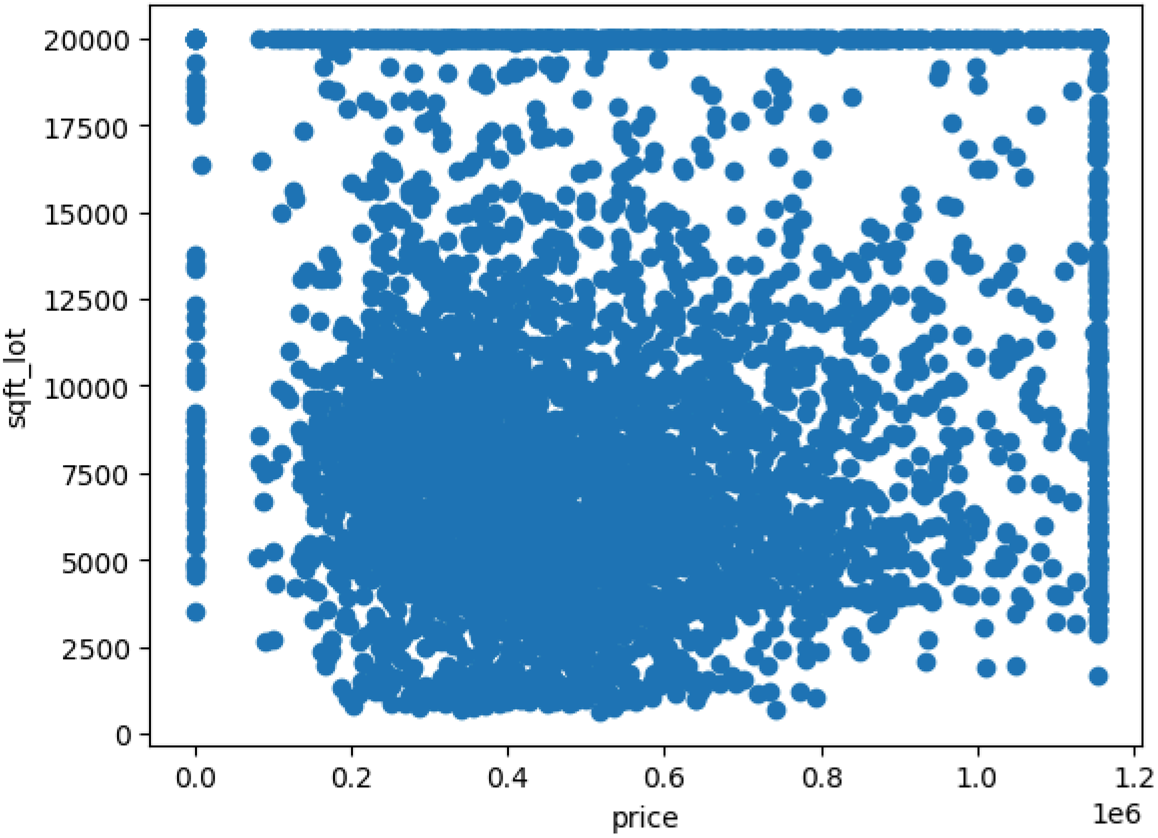
In [21]:

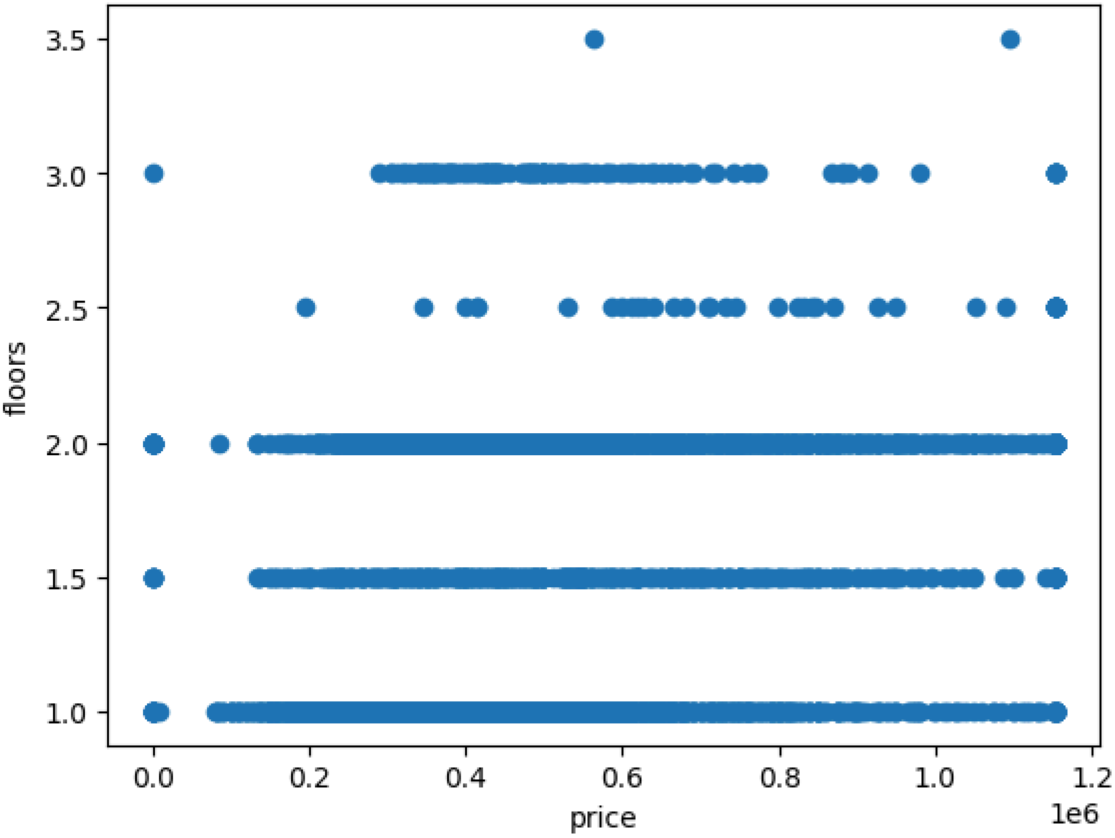
```
for i in x.columns:
  plt.scatter(x=y,y=x[i])
  plt.xlabel('price')
  plt.ylabel(i)
  plt.show()
```

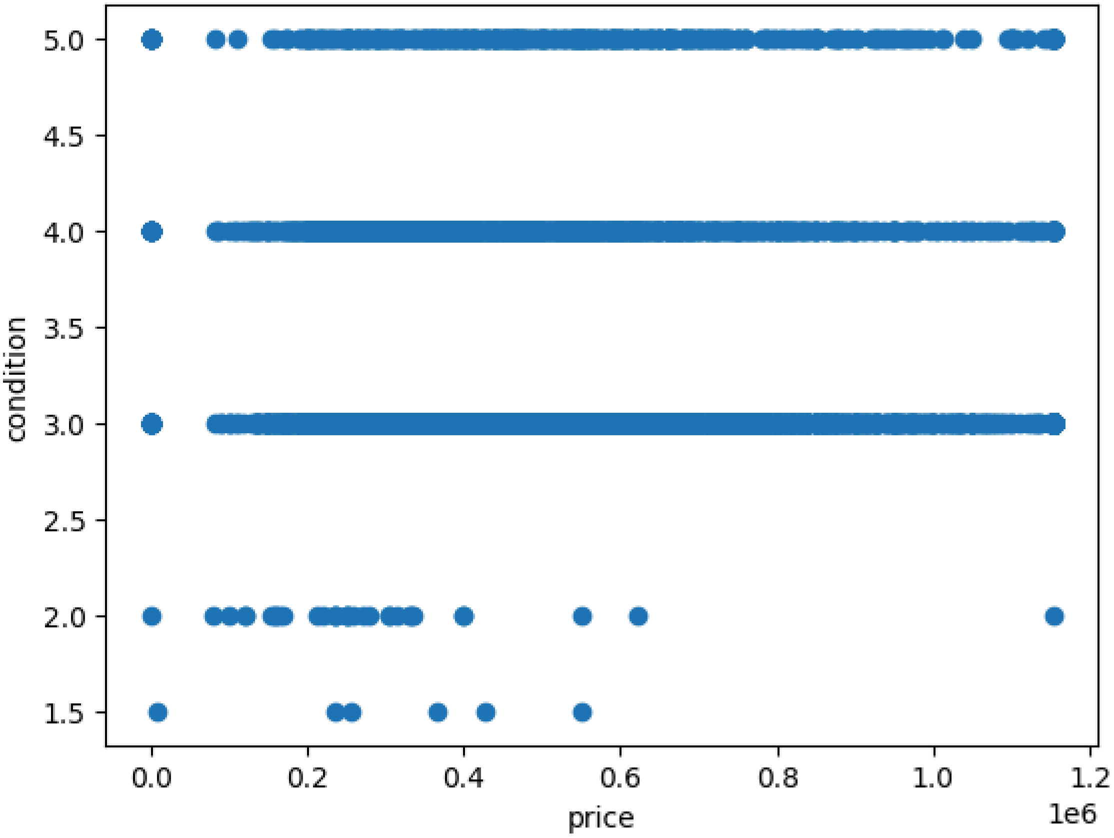


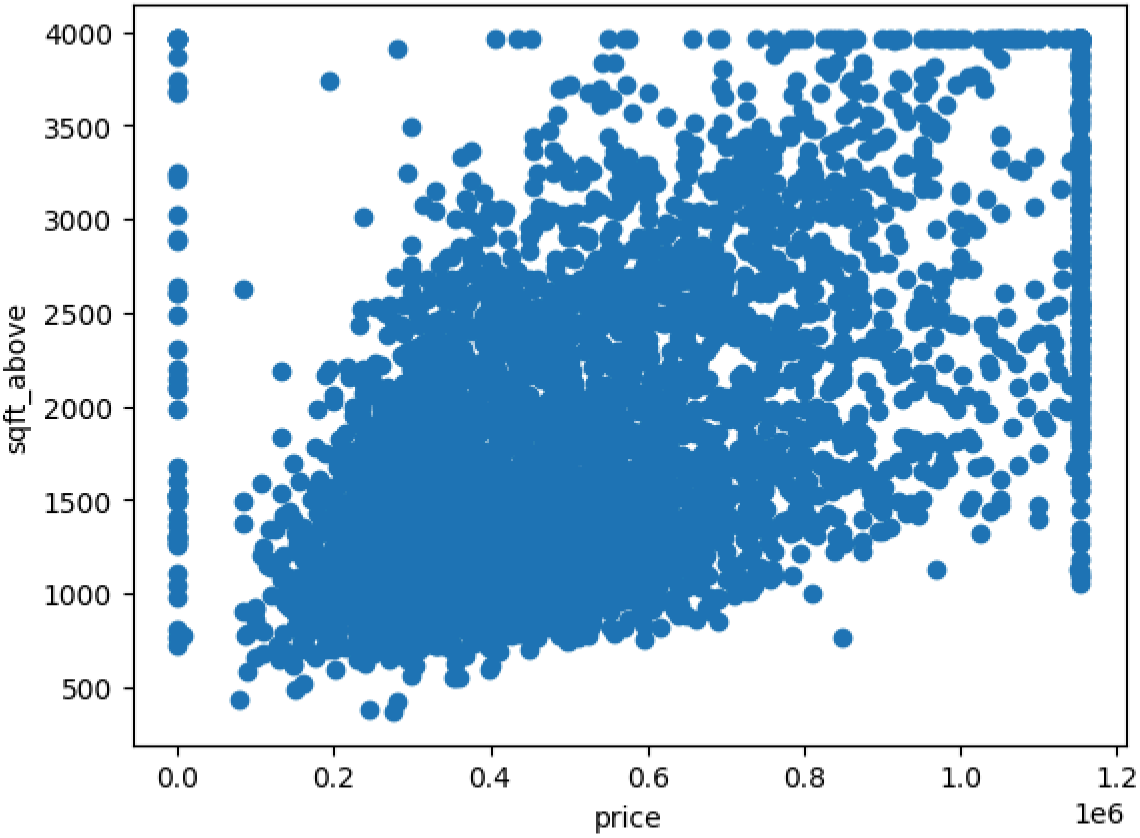

3.5
3.0
2.5
2.0
1.5
1.0
0.5
bathrooms
0.0
0.2
0.4
0.6
0.8
1.0
1.2
price
1e6

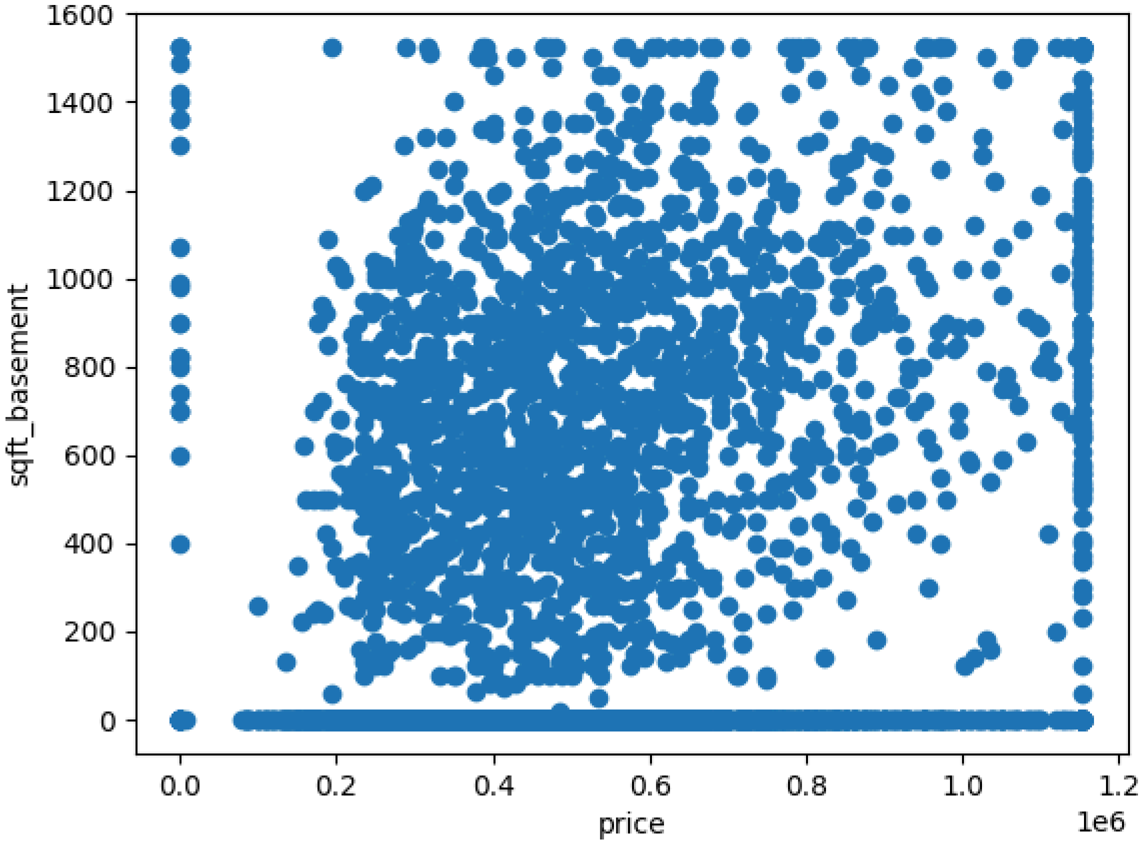

4500
4000
3500
3000
2500
2000
1500
1000
500
sqft_living
0.0
0.2
0.4
0.6
0.8
1.0
1.2
price
1e6

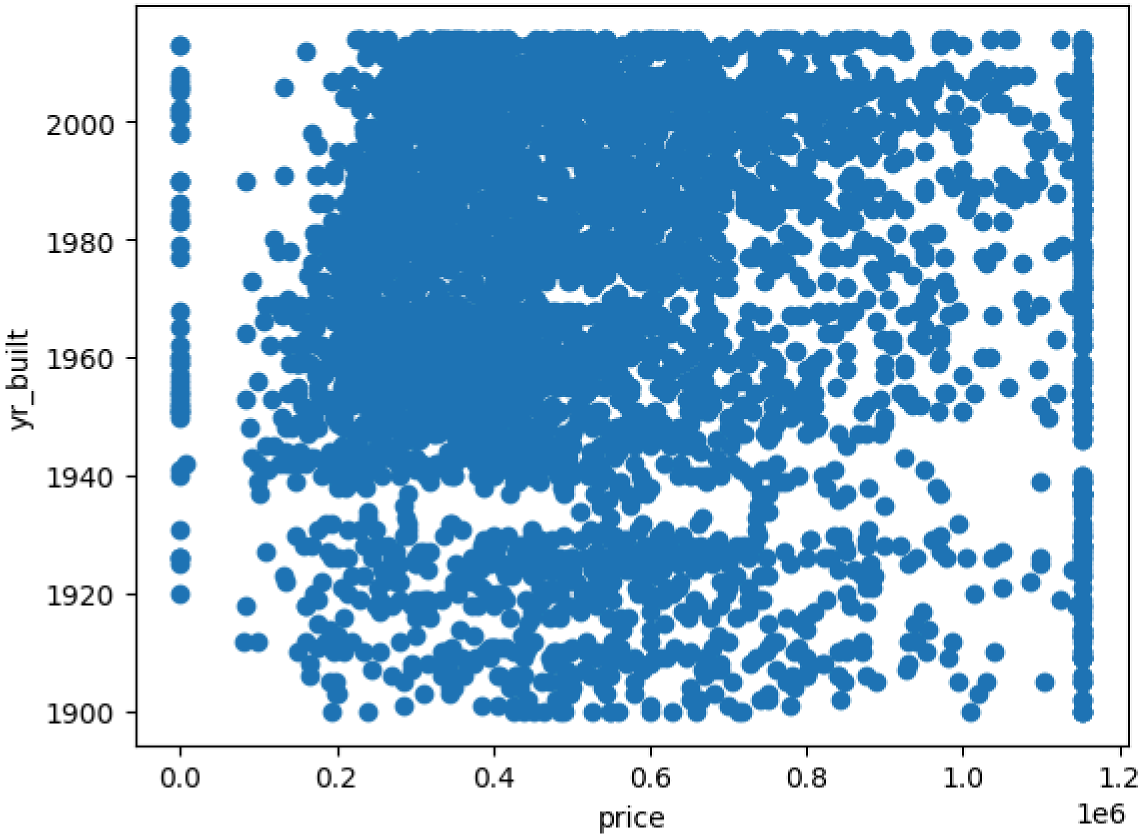

20000
17500
15000
12500
10000
7500
5000
2500
0
sqft_lot
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1e6
price

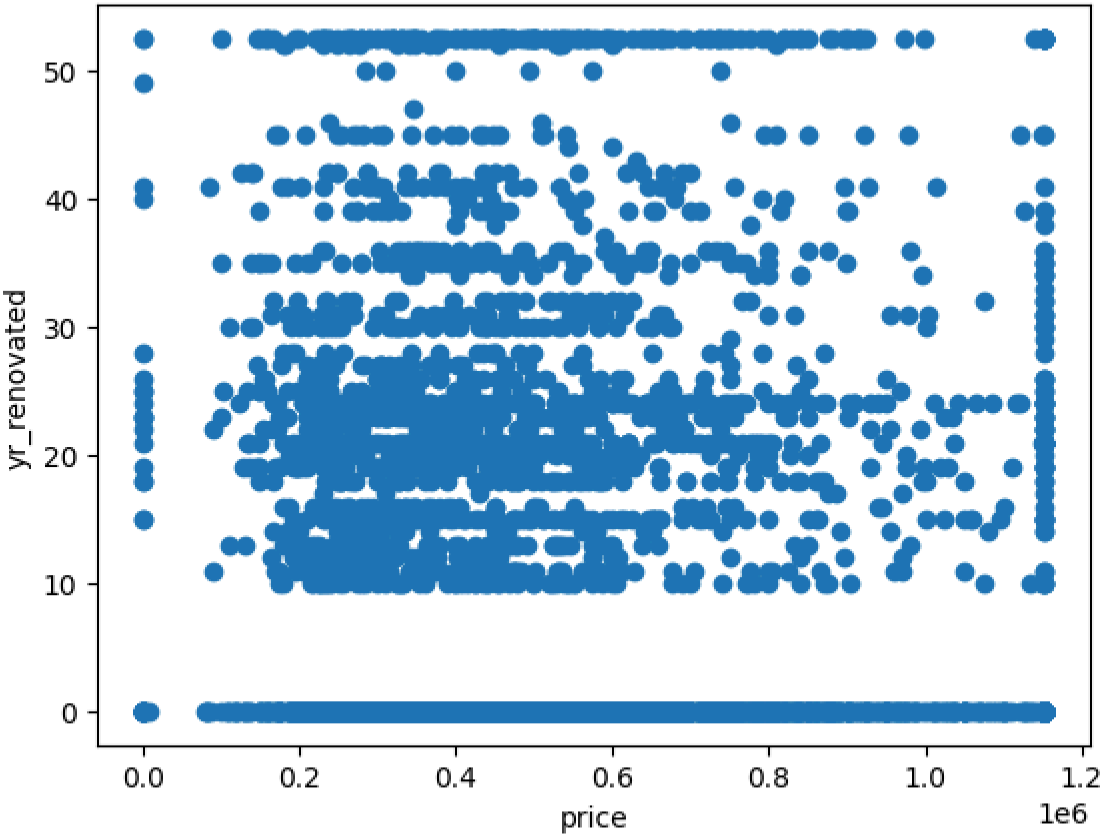

3.5
3.0
2.5
2.0
1.5
1.0
floors
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1e6
price

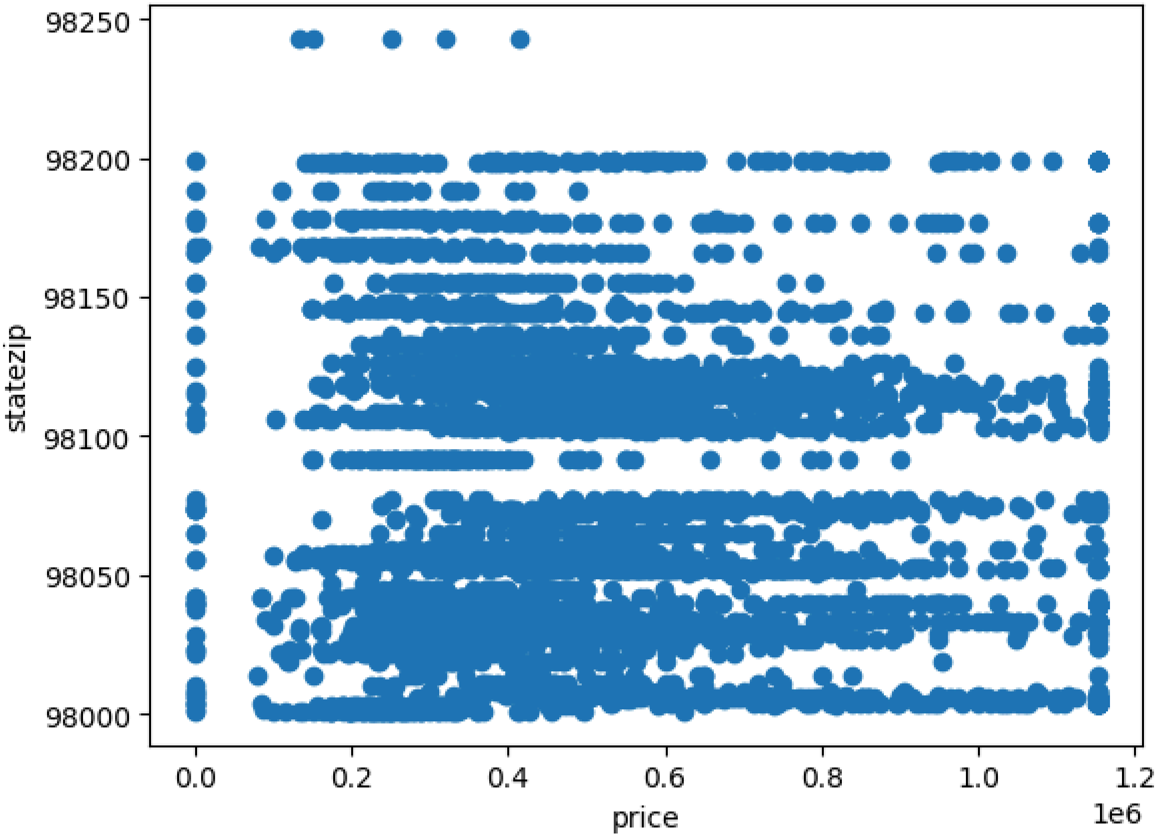

5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
condition
0.0
0.2
0.4
0.6
0.8
1.0
1.2
price
1e6


4000
3500
3000
2500
2000
1500
1000
500
sqft_above
0.0
0.2
0.4
0.6
0.8
1.0
1.2
price
1e6


1600
1400
1200
1000
800
600
400
200
0
sqft_basement
0.0
0.2
0.4
0.6
0.8
1.0
1.2
price
1e6


2000
1980
1960
1940
1920
1900
yr_built
0.0
0.2
0.4
0.6
0.8
1.0
1.2
price
1e6





In [22]:
```
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=1)
x_train
```

Out[22]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	condition	sqft_above	sqft_basement	y
3990	4.0	3.250	3990.0	9786.0	2.0	3.0	3965.0	0.0	
1507	4.0	2.500	2680.0	7178.0	2.0	3.0	2680.0	0.0	
1652	3.0	2.250	2675.0	20002.0	2.0	3.0	2675.0	0.0	
2279	5.5	3.625	3440.0	4500.0	2.0	3.0	3280.0	160.0	
2106	4.0	1.750	2220.0	6500.0	2.0	4.0	2220.0	0.0	
...	...	...	...	...	...	...	...	...	
2895	4.0	1.500	1220.0	9600.0	1.0	3.0	1220.0	0.0	
2763	3.0	1.750	1970.0	8200.0	1.0	5.0	1420.0	550.0	
905	3.0	2.500	1720.0	1916.0	2.0	3.0	1720.0	0.0	
3980	4.0	3.250	4100.0	20002.0	2.0	3.0	2500.0	1525.0	
235	3.0	2.750	1540.0	6760.0	1.0	5.0	1210.0	330.0	

3220 rows × 11 columns

In [23]:

```
x_test
```

Out[23]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	condition	sqft_above	sqft_basement	y
1351	5.0	2.000	2330.0	10750.0	1.0	4.0	1190.0	1140.0	
1687	3.0	2.500	2730.0	5832.0	2.0	3.0	2730.0	0.0	
1297	3.0	2.000	1220.0	1186.0	3.0	3.0	1220.0	0.0	
2101	4.0	1.750	2700.0	7875.0	1.5	4.0	2700.0	0.0	
3920	3.0	3.625	2080.0	2250.0	3.0	3.0	2080.0	0.0	
...	...	...	...	...	...	...	...	...	
3490	4.0	2.500	2250.0	7526.0	2.0	3.0	2250.0	0.0	
3584	2.0	0.750	840.0	20002.0	1.0	4.0	840.0	0.0	
855	2.0	2.000	1360.0	4688.0	1.0	3.0	780.0	580.0	
309	5.0	3.625	4270.0	8076.0	2.0	3.0	3400.0	870.0	
1929	3.0	2.500	2680.0	9750.0	1.0	4.0	1610.0	1070.0	

1380 rows × 11 columns

In [24]:

```
y_train
```

Out[24]:

	price
3990	866000.0
1507	515000.0
1652	660000.0
2279	495000.0
2106	480000.0
...	...
2895	310000.0
2763	540000.0
905	280000.0
3980	775000.0
235	503000.0

3220 rows × 1 columns

dtype: float64

In [25]:

```
y_test
```

Out[25]:

	price
1351	389000.00
1687	1000000.00
1297	355000.00
2101	390000.00
3920	715000.00
...	...
3490	440000.00
3584	528000.00
855	488000.00
309	1153093.75
1929	653000.00

1380 rows × 1 columns

dtype: float64

Model Creation and Performance Evaluation

In [26]:

```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor,GradientBoostingRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score
linreg=LinearRegression()
```

```
ranfor=RandomForestRegressor()
dectree=DecisionTreeRegressor()
grad=GradientBoostingRegressor()
lst=[linreg,ranfor,dectree,grad]
```

In [27]:

```
for i in lst:
  i.fit(x_train,y_train)
  y_pred=i.predict(x_test)
  print("R2score of",i,"is",r2_score(y_test,y_pred))
```

```
R2score of LinearRegression() is 0.483250199383935
R2score of RandomForestRegressor() is 0.6058197511228784
R2score of DecisionTreeRegressor() is 0.33683930554751207
R2score of GradientBoostingRegressor() is 0.6351632169487991
```