In [45]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot  as plt
import seaborn as sns
```

In [2]:
```
data = pd.read_csv('house_data.csv')
print(data)
```

```
                     date         price  bedrooms  bathrooms  sqft_living  \
0     2014-05-02 00:00:00  3.130000e+05       3.0       1.50         1340
1     2014-05-02 00:00:00  2.384000e+06       5.0       2.50         3650
2     2014-05-02 00:00:00  3.420000e+05       3.0       2.00         1930
3     2014-05-02 00:00:00  4.200000e+05       3.0       2.25         2000
4     2014-05-02 00:00:00  5.500000e+05       4.0       2.50         1940
...                   ...           ...       ...        ...          ...
4595  2014-07-09 00:00:00  3.081667e+05       3.0       1.75         1510
4596  2014-07-09 00:00:00  5.343333e+05       3.0       2.50         1460
4597  2014-07-09 00:00:00  4.169042e+05       3.0       2.50         3010
4598  2014-07-10 00:00:00  2.034000e+05       4.0       2.00         2090
4599  2014-07-10 00:00:00  2.206000e+05       3.0       2.50         1490

      sqft_lot  floors  waterfront  view  condition  sqft_above  \
0         7912     1.5           0     0          3        1340
1         9050     2.0           0     4          5        3370
2        11947     1.0           0     0          4        1930
3         8030     1.0           0     0          4        1000
4        10500     1.0           0     0          4        1140
...        ...     ...         ...   ...        ...         ...
4595      6360     1.0           0     0          4        1510
4596      7573     2.0           0     0          3        1460
4597      7014     2.0           0     0          3        3010
4598      6630     1.0           0     0          3        1070
4599      8102     2.0           0     0          4        1490

      sqft_basement  yr_built  yr_renovated                    street  \
0                 0      1955          2005       18810 Densmore Ave N
1               280      1921             0           709 W Blaine St
2                 0      1966             0  26206-26214 143rd Ave SE
3              1000      1963             0           857 170th Pl NE
4               800      1976          1992         9105 170th Ave NE
...             ...       ...           ...                       ...
4595              0      1954          1979            501 N 143rd St
4596              0      1983          2009          14855 SE 10th Pl
4597              0      2009             0          759 Ilwaco Pl NE
4598           1020      1974             0         5148 S Creston St
4599              0      1990             0         18717 SE 258th St

           city  statezip country
0     Shoreline  WA 98133     USA
1       Seattle  WA 98119     USA
2          Kent  WA 98042     USA
3      Bellevue  WA 98008     USA
4       Redmond  WA 98052     USA
...         ...       ...     ...
4595    Seattle  WA 98133     USA
4596   Bellevue  WA 98007     USA
4597     Renton  WA 98059     USA
4598    Seattle  WA 98178     USA
4599  Covington  WA 98042     USA

[4600 rows x 18 columns]
```

In [4]:
```
data.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
```

In [5]:
```
data.drop(['date'], axis = 1, inplace = True)
data.head()
```

Out[5]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sc
0	313000.0	3.0	1.50	1340	7912	1.5	0	0	3	
1	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5	
2	342000.0	3.0	2.00	1930	11947	1.0	0	0	4	
3	420000.0	3.0	2.25	2000	8030	1.0	0	0	4	
4	550000.0	4.0	2.50	1940	10500	1.0	0	0	4	

In [6]:
```
data.country.value_counts()
```

Out[6]:
```
USA    4600
Name: country, dtype: int64
```

In [8]:
```
data.drop(['country'], axis = 1, inplace = True)
data.head()
```

Out[8]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sc
0	313000.0	3.0	1.50	1340	7912	1.5	0	0	3	
1	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5	
2	342000.0	3.0	2.00	1930	11947	1.0	0	0	4	
3	420000.0	3.0	2.25	2000	8030	1.0	0	0	4	
4	550000.0	4.0	2.50	1940	10500	1.0	0	0	4	

In [9]:
```
data.drop(['street', 'city'], axis = 1, inplace = True)
data.head()
```

Out[9]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sc
0	313000.0	3.0	1.50	1340	7912	1.5	0	0	3	
1	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5	
2	342000.0	3.0	2.00	1930	11947	1.0	0	0	4	
3	420000.0	3.0	2.25	2000	8030	1.0	0	0	4	
4	550000.0	4.0	2.50	1940	10500	1.0	0	0	4	

In [11]:
```
data.isnull().sum()
```

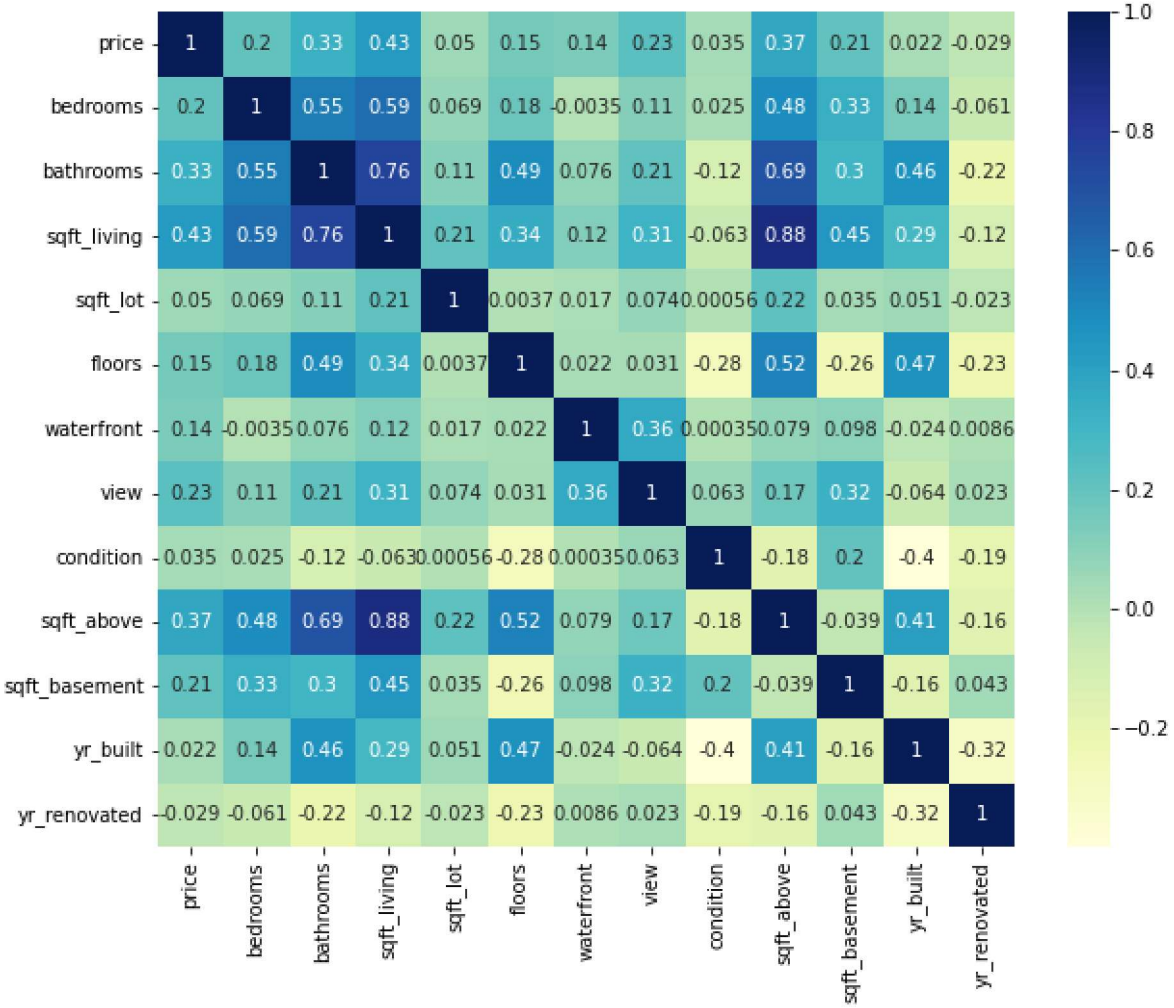
Out[11]:
```
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
statezip         0
dtype: int64
```

In [12]:
```
data.notnull().sum()
```

Out[12]:
```
price            4600
bedrooms         4600
bathrooms        4600
sqft_living      4600
sqft_lot         4600
floors           4600
waterfront       4600
view             4600
condition        4600
sqft_above       4600
sqft_basement    4600
yr_built         4600
yr_renovated     4600
statezip         4600
dtype: int64
```

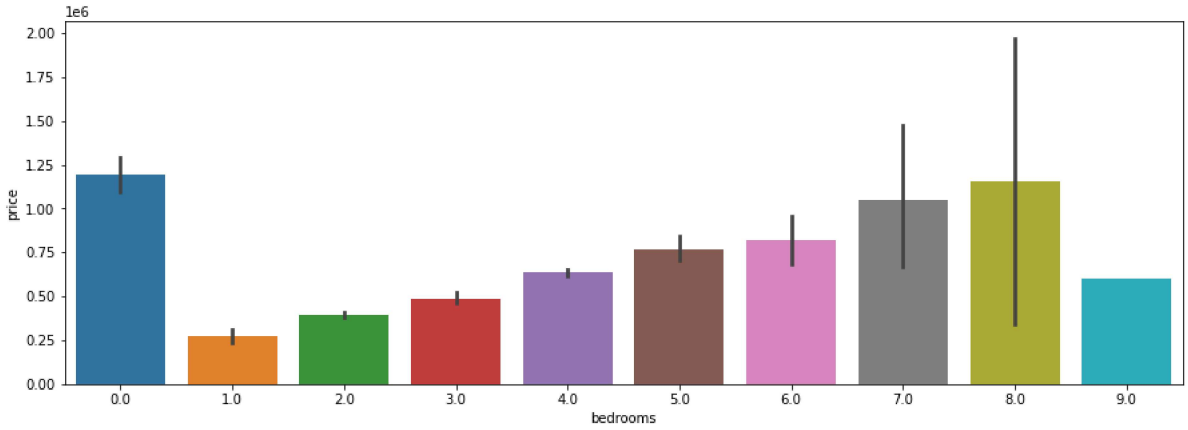
In [15]:
```
a4_dims = (10, 8)
fig, ax = plt.subplots(figsize=a4_dims)
cor = data.corr()
sns.heatmap(cor, annot = True, cmap="YlGnBu")
```

Out[15]:
```
<AxesSubplot:>
```



In [16]:
```
a4_dims = (15, 5)
fig, ax = plt.subplots(figsize=a4_dims)
sns.barplot(x = data.bedrooms, y = data.price)
```

Out[16]:
```
<AxesSubplot:xlabel='bedrooms', ylabel='price'>
```



In [17]:
```
data.groupby('bedrooms').price.agg([len, min, max])
```

Out[17]:

bedrooms	len	min	max
0.0	2	1095000.0	1295648.0
1.0	38	0.0	540000.0
2.0	566	0.0	1695000.0
3.0	2032	0.0	26590000.0
4.0	1531	0.0	4489000.0
5.0	353	0.0	7062500.0
6.0	61	0.0	3100000.0
7.0	14	280000.0	3200000.0
8.0	2	340000.0	1970000.0
9.0	1	599999.0	599999.0

In [19]:
```
data.shape
```

Out[19]:
```
(4600, 14)
```

In [20]:
```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
```

In [22]:
```
data['statezip_encoded'] = le.fit_transform(data.statezip)
data.head()
```

Out[22]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sc
0	313000.0	3.0	1.50	1340	7912	1.5	0	0	3	
1	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5	
2	342000.0	3.0	2.00	1930	11947	1.0	0	0	4	
3	420000.0	3.0	2.25	2000	8030	1.0	0	0	4	
4	550000.0	4.0	2.50	1940	10500	1.0	0	0	4	

In [23]:
```
data.statezip_encoded.value_counts()
```

Out[23]:
```
47    148
31    135
56    132
54    130
5     110
     ...
28      6
75      3
29      2
76      2
39      1
Name: statezip_encoded, Length: 77, dtype: int64
```

In [25]:
```
data.drop(['statezip'], axis = 1, inplace = True)
data.head()
```

Out[25]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sc
0	313000.0	3.0	1.50	1340	7912	1.5	0	0	3	
1	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5	
2	342000.0	3.0	2.00	1930	11947	1.0	0	0	4	
3	420000.0	3.0	2.25	2000	8030	1.0	0	0	4	
4	550000.0	4.0	2.50	1940	10500	1.0	0	0	4	

In [26]:
```
from sklearn.preprocessing import OneHotEncoder
ohc = OneHotEncoder()
```

In [28]:
```
ohc_df = pd.DataFrame(ohc.fit_transform(data[['statezip_encoded']]).toarray())
ohc_df.head()
```

Out[28]:

	0	1	2	3	4	5	6	7	8	9	...	67	68	69	70	71	72	73	74	75	76
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 77 columns

In [29]:
```
data = data.join(ohc_df)
data.head()
```

Out[29]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sc
0	313000.0	3.0	1.50	1340	7912	1.5	0	0	3	
1	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5	
2	342000.0	3.0	2.00	1930	11947	1.0	0	0	4	
3	420000.0	3.0	2.25	2000	8030	1.0	0	0	4	
4	550000.0	4.0	2.50	1940	10500	1.0	0	0	4	

5 rows × 91 columns

In [31]:
```
data.tail()
```

Out[31]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condi
4595	308166.666667	3.0	1.75	1510	6360	1.0	0	0	
4596	534333.333333	3.0	2.50	1460	7573	2.0	0	0	
4597	416904.166667	3.0	2.50	3010	7014	2.0	0	0	
4598	203400.000000	4.0	2.00	2090	6630	1.0	0	0	
4599	220600.000000	3.0	2.50	1490	8102	2.0	0	0	

5 rows × 91 columns

In [32]:
```
data.drop(['statezip_encoded'], axis = 1, inplace = True)
```

In [33]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4600 entries, 0 to 4599
Data columns (total 90 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   price          4600 non-null   float64
 1   bedrooms       4600 non-null   float64
 2   bathrooms      4600 non-null   float64
 3   sqft_living    4600 non-null   int64  
 4   sqft_lot       4600 non-null   int64  
 5   floors         4600 non-null   float64
 6   waterfront     4600 non-null   int64  
 7   view           4600 non-null   int64  
 8   condition      4600 non-null   int64  
 9   sqft_above     4600 non-null   int64  
 10  sqft_basement  4600 non-null   int64  
 11  yr_built       4600 non-null   int64  
 12  yr_renovated   4600 non-null   int64  
 13  0              4600 non-null   float64
 14  1              4600 non-null   float64
 15  2              4600 non-null   float64
 16  3              4600 non-null   float64
 17  4              4600 non-null   float64
 18  5              4600 non-null   float64
 19  6              4600 non-null   float64
 20  7              4600 non-null   float64
 21  8              4600 non-null   float64
 22  9              4600 non-null   float64
 23  10             4600 non-null   float64
 24  11             4600 non-null   float64
 25  12             4600 non-null   float64
 26  13             4600 non-null   float64
 27  14             4600 non-null   float64
 28  15             4600 non-null   float64
 29  16             4600 non-null   float64
 30  17             4600 non-null   float64
 31  18             4600 non-null   float64
 32  19             4600 non-null   float64
 33  20             4600 non-null   float64
 34  21             4600 non-null   float64
 35  22             4600 non-null   float64
 36  23             4600 non-null   float64
 37  24             4600 non-null   float64
 38  25             4600 non-null   float64
 39  26             4600 non-null   float64
 40  27             4600 non-null   float64
 41  28             4600 non-null   float64
 42  29             4600 non-null   float64
 43  30             4600 non-null   float64
 44  31             4600 non-null   float64
 45  32             4600 non-null   float64
 46  33             4600 non-null   float64
 47  34             4600 non-null   float64
 48  35             4600 non-null   float64
 49  36             4600 non-null   float64
 50  37             4600 non-null   float64
 51  38             4600 non-null   float64
 52  39             4600 non-null   float64
 53  40             4600 non-null   float64
 54  41             4600 non-null   float64
 55  42             4600 non-null   float64
 56  43             4600 non-null   float64
 57  44             4600 non-null   float64
 58  45             4600 non-null   float64
```

```
 59  46              4600 non-null   float64
 60  47              4600 non-null   float64
 61  48              4600 non-null   float64
 62  49              4600 non-null   float64
 63  50              4600 non-null   float64
 64  51              4600 non-null   float64
 65  52              4600 non-null   float64
 66  53              4600 non-null   float64
 67  54              4600 non-null   float64
 68  55              4600 non-null   float64
 69  56              4600 non-null   float64
 70  57              4600 non-null   float64
 71  58              4600 non-null   float64
 72  59              4600 non-null   float64
 73  60              4600 non-null   float64
 74  61              4600 non-null   float64
 75  62              4600 non-null   float64
 76  63              4600 non-null   float64
 77  64              4600 non-null   float64
 78  65              4600 non-null   float64
 79  66              4600 non-null   float64
 80  67              4600 non-null   float64
 81  68              4600 non-null   float64
 82  69              4600 non-null   float64
 83  70              4600 non-null   float64
 84  71              4600 non-null   float64
 85  72              4600 non-null   float64
 86  73              4600 non-null   float64
 87  74              4600 non-null   float64
 88  75              4600 non-null   float64
 89  76              4600 non-null   float64
dtypes: float64(81), int64(9)
memory usage: 3.2 MB
```

In [35]:
```
X = data.iloc[:, 1:]
X.shape
```

Out[35]:
```
(4600, 89)
```

In [36]:
```
y = data.price
```

In [37]:
```
from sklearn.model_selection import train_test_split
X_train, X_rem, y_train, y_rem = train_test_split(X, y, test_size=0.1, random_state
```

In [39]:
```
print(len(X_train) / len(data))
```

```
0.9
```

In [40]:
```
X_val, X_test, y_val, y_test = train_test_split(X_rem, y_rem, test_size=0.5, random
print(len(X_test) / len(y_rem))
```

```
0.5
```

In [41]:
```
print(len(X_train))
print(len(X_val))
print(len(X_val))
```

```
4140
230
230
```

In [42]:
```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
```

In [43]:
```
lin_reg.fit(X_train, y_train)
```

```
C:\Users\J.P.Monpara\anaconda3\lib\site-packages\sklearn\utils\validation.py:1688:
FutureWarning: Feature names only support names that are all strings. Got feature
names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

Out[43]:
```
LinearRegression()
```

In [46]:
```
from sklearn.metrics import mean_squared_error
y_pred = lin_reg.predict(X_val)
mse = mean_squared_error(y_pred, y_val)
rmse = np.sqrt(mse)
rmse
```

```
C:\Users\J.P.Monpara\anaconda3\lib\site-packages\sklearn\utils\validation.py:1688:
FutureWarning: Feature names only support names that are all strings. Got feature
names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

Out[46]:
```
171735.7621469671
```

In [47]:
```
y_val.head(10)
```

Out[47]:
```
3214     455000.0
2395     445000.0
132      210000.0
683      349950.0
2475     590000.0
291      540000.0
1522     782000.0
734     1255000.0
4599     220600.0
1833     299000.0
Name: price, dtype: float64
```

In [48]:
```
y_pred
```

Out[48]:
```
array([ 430712.78126359,  432325.10255265,  228233.69947648,
        279993.80048966,  553202.75733781,  841446.67237687,
        938421.26710343, 1102036.68086076,  200714.68971086,
        103553.66236711,  641721.35011125,  605091.18263078,
        314406.57667375,  813644.66798234, 1123798.38966203,
        538390.18433976,  415837.91065812,  761689.14063859,
        425962.95313859,  501155.49757218,  206925.81593156,
        121474.71925187,  149235.95362687,  585535.68360734,
        451890.02491593,  305202.32130265,  328606.73243546,
        485550.59425187,  413599.86256242,  350713.02247453,
        799522.92140031,  522199.59766984,  924306.48609757,
        871793.94093156,  273194.45216203,  840216.02162004,
       1125362.21168351,  223172.36976457,  197481.18702531,
        423576.77149796, 1391828.66645646,  410531.15174699,
        537159.23634171, 1614147.65675187,  731503.88184953,
        442044.53809953,  521229.35493302,  464819.23634171,
        705638.7392714 ,  459110.42725968,  119217.93116593,
        777171.86720109, 1057021.82618546,  388596.41065812,
        776113.78785539,  356600.377455  ,  272498.72181535,
        410329.97706437,  197244.65675187,  484270.36134171,
        387576.76856828,  165235.42237687,  512493.86329484,
        342088.34327531,  735262.38136125,  250780.27516007,
        825828.72657609,  882502.88160539,  755267.47169328,
        168095.05824113,  575859.39796281,  507932.37306046,
        116309.93116593,  815658.99561906,  427916.13087296,
        441829.86720109,  310446.18751359,  469261.27393937,
        490395.75013566,  882574.33595109,  691434.31782365,
       1185276.20289445,  466011.79981828, 1021384.533705  ,
       1237486.14625382,  191603.09986711,  522195.86573625,
        421382.3696425 ,  325231.14283586,  845815.47187638,
        489331.99751115,  284119.28834367,  201520.48170304,
        177989.37647843,  310445.74317765,  612228.08399796,
        608971.21046281,  483821.39405656,  223425.6352675 ,
        432329.92750382,  224399.42237687,  283040.63233781,
        396848.20313859,  253429.54347062,  833020.33643937,
        566360.37452531, 1261915.97633195,  703586.00660539,
        436005.64845109,  415455.79981828,  254465.21119523,
       1113179.14704728,  448003.26417375,  144094.08656144,
         61878.67725968, 1666653.31300187,  471257.47437882,
        241406.58155656,  672997.15382218,  197368.71681046,
        171134.68873429,  504882.34278703,  235351.38905168,
        506321.80323625,  336363.75587296,  278118.06080461,
        760401.85462785,  464801.89893937,  160356.06471086,
       1354140.57185197,  233475.15528703,  841578.34351945,
        446591.28242326,  245876.24757218,  267753.67994523,
        296541.84962296,  589496.66554093, 1037430.23585343,
       1074444.00343156,  486493.06910539,  757716.89649796,
        529994.09126115,  185207.06175065,  303262.01808   ,
       1284214.89649796,  529758.77149796,  504264.52003312,
        221997.77198625,  268329.98194718,  490737.21095109,
       1112023.64063859,  667949.21443009,  267323.81788468,
        790773.02094865,  898556.96925187,  446693.08399796,
        835600.86842179,  498067.96827531,  247338.26240373,
        851913.4568007 ,  215768.01954484,  725296.07539201,
        530573.02094865,  394343.9790175 ,  139836.6455214 ,
        185594.99415421,  790504.97315812,  202051.3128798 ,
       1318765.69801164,  435170.75880265,  403471.68012834,
        298814.64210343,  539611.33595109,  346469.80909562,
        262430.70216203, 1364507.09919572,  970581.51954484,
        850149.45948625,  660644.51783586,  835496.7705214 ,
        652706.57765031,  699036.78126359,  692906.43281388,
         94846.1845839 ,  426938.99903703,  364431.58058   ,
        366313.03468156,  275497.12110734,  582314.50880265,
       1398687.29371476,  563967.92481828,  373149.49827409,
```

```
 499041.67042375,  272930.11134171,  924195.65272355,
 812563.53101945, 1100472.34528947,  847077.08643937,
 750788.7158339 ,  438985.45948625,  564100.06690812,
 883855.47047257,  236753.15455461,  278859.36720109,
 156667.09498429,  253238.44398332,  706707.67091203,
 777849.46827531,  453563.24195695,  715587.16065812,
 534562.63624406,  446361.52149796,  241306.71925187,
 704911.98414445,  764943.17188859,  525876.22511125,
 341111.71192765,  506055.0863173 ,  420551.52003312,
 380705.43849015,  599066.48646379,  618128.93897843,
1436976.76417375,  601541.02931046,  318125.73121476,
 597443.58888078, 1132528.82911515,  125558.19727921,
 667178.72071671,  109296.03907609])
```

In [49]:
```
y_pred_test = lin_reg.predict(X_test)
mse = mean_squared_error(y_pred_test, y_test)
rmse = np.sqrt(mse)
rmse
```

```
C:\Users\J.P.Monpara\anaconda3\lib\site-packages\sklearn\utils\validation.py:1688:
FutureWarning: Feature names only support names that are all strings. Got feature
names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

Out[49]:
```
217675.7753546666
```

In [50]:
```
lin_reg.score(X_test, y_test)
```

```
C:\Users\J.P.Monpara\anaconda3\lib\site-packages\sklearn\utils\validation.py:1688:
FutureWarning: Feature names only support names that are all strings. Got feature
names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

Out[50]:
```
0.6970097195591245
```

In [51]:
```
y_test
```

Out[51]:
```
764      839000.0
252     3200000.0
3528     140000.0
1919     549000.0
2279     495000.0
          ...
3711     445000.0
1534     690000.0
4279     374950.0
999      555000.0
3308     583000.0
Name: price, Length: 230, dtype: float64
```

In [52]:
```
y_pred_test
```

Out[52]:
```
array([ 916719.87354875, 1859420.72559953,   85161.97193742,
         611467.6443007 ,  605476.19068742,  436996.89649796,
        1136947.70954728,  839187.08595109,  207213.81104875,
        1000148.11085343, 1148018.64649796,  275424.29026628,
         298043.89771867,  570562.2705214 ,  442660.76612687,
         752934.596205  ,  622177.20582414,  518651.0102675 ,
         596903.27393937,  470628.32130265, 1477745.32569718,
         199560.85767961,  534959.91126847,  624190.23878312,
         233882.28761125,  559008.52149796,  319492.033705  ,
        1100573.76710343, 1244180.92188859,  817632.34528947,
         496358.95997453,  356816.2602675 ,  820282.54176164,
         265354.00709367,  458828.19972062,  448363.32130265,
         658873.79835343,  453987.13429093,  520752.61231828,
         550306.80299211,  235324.81251359,  244334.40675187,
         600512.00709367,  418172.39942765,  442572.71827531,
         285412.30344987,  895703.1220839 ,  623909.00562882,
         388741.34278703,  800213.13477921,  452022.18067765,
         411124.31104875,  318967.5258925 ,  472650.71485734,
         329986.85499406, 1124780.45857072, 1032135.53321671,
         977519.16016984,  341666.60352921,  558618.43067765,
         762484.4845717 ,  915245.85914445,  442122.71485734,
         663811.82032609,  386809.65772843,  820788.89301896,
         507577.31446671,  745667.33021379, 1277376.5595839 ,
         758915.60597062,  564996.16578507,  241038.58155656,
         652948.20069718,  578327.00977921,  435157.64942765,
          59449.52015519, 1785067.69410539,  388810.12354875,
        1007243.36933732,  310361.79664445,  164561.67188859,
         471778.20606828,  554911.71998429,  521211.45265031,
         262068.58814836,  816655.1040175 ,  615157.33985734,
         231019.72041154,  261172.62061906,  478050.65919328,
         530573.02094865,  751312.21485734,  532513.02854753,
         612577.84376359,  673629.30317521,  430850.43653703,
         280385.95411515,  538619.43089128,  502113.61353898,
         350608.63331437, 1290412.82911515,  194398.25514054,
         277805.3017714 ,  482958.46534562,  288468.8720839 ,
         578143.07130265,  229074.0571425 ,  774033.89698625,
         359133.65431046,  573318.21168351,  564042.25147843,
         445219.09986711,  205921.04493546,  645389.84913468,
         392377.7236464 ,  596717.5415175 ,  334653.6767714 ,
        1102969.19581437,  787211.75099015,  636363.16969132,
         414967.75343156,  576366.01808   , 1020332.23200822,
         939195.39625382,  610160.56251359,  427727.26856828,
        1227960.29316545,  691258.09571671,  942435.51814103,
         670950.38331437,  581877.95997453,  759753.11695695,
         846770.51954484,  245053.18812394,  731002.61622453,
         724477.06129289,  761748.93824601,  193931.20631242,
         525864.56837296,  367288.83643937,  300580.54737687,
         612431.08399796,  798151.16822648,  448858.71485734,
         162974.73170304,  454728.72779679,  292575.23975968,
         793494.26466203,  745207.29176164,  484892.93702531,
         284034.62403703,  889102.12891984,  335627.10987687,
         922214.33106828,  454845.29249406,  464652.66645646,
         422444.49757218,  516767.30470109,  842453.46046281,
         823874.06397843,  927172.21290421,  449292.75880265,
         286560.80030656,  579603.43067765,  776250.93958879,
         448338.79884171,  403280.74903703,  207494.57472062,
          85281.52198625,  859898.82374406,  318722.4477675 ,
        1034995.48689103, 3128922.77247453,  858640.92188859,
         317004.79054093,  288132.29542375,  323592.16065812,
         597417.03419328, 1590232.98683   ,  365519.95704484,
         753434.90040421,  270663.36720109,  487468.07642961,
         623214.63673234, 1357208.40186906,  636781.00562882,
         379483.7290175 ,  821742.42848039,  110485.52784562,
         298642.93653703,  967262.50294328,  586271.18556046,
```

```
 390644.48536515,  450840.71241593,  591600.8173964 ,
 317883.67481828,  460392.02393937,  256485.19111466,
 760511.68433976,  168667.58350968,  612094.44630265,
1277745.79249406,  510203.03907609,  435854.54835343,
 116696.7548964 ,  512984.65138078,  203746.08350968,
 493391.53126359,  762580.60621476,  250005.81715226,
 760422.36836076,  660684.18867326,  763568.89234757,
 234579.26881242,  297636.05799699,  284536.31153703,
 470357.02442765,  714631.80641007,  553531.81349015,
1261447.40138078,  261033.8390944 ,  175755.77638078,
 239698.07105851,  798327.20509171,  479554.23438859,
 338922.80299211,  778877.78565812,  447600.77149796,
 718517.54493546,  377422.60981584])
```