


```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.metrics import mean_absolute_percentage_error, r2_score, mean_squared_error
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

```
df=pd.read_csv('/content/drive/MyDrive/project_files/house_price.csv')
df
```



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


4600 rows × 18 columns

Next steps:

Generate code with df

 View recommended plots

df.head()




	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	s
0	2014-05-02 00:00:00	313000.0	3.0	1.50	1340	7912	1.5	0	0	3	1340	
1	2014-05-02 00:00:00	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5	3370	
2	2014-05-02 00:00:00	342000.0	3.0	2.00	1930	11947	1.0	0	0	4	1930	
3	2014-05-02 00:00:00	420000.0	3.0	2.25	2000	8030	1.0	0	0	4	1000	
4	2014-05-02 00:00:00	550000.0	4.0	2.50	1940	10500	1.0	0	0	4	1140	

Next steps:

[Generate code with df](#)

 [View recommended plots](#)

df.tail()




	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_a
4595	2014-07-09 00:00:00	308166.666667	3.0	1.75	1510	6360	1.0	0	0	4	
4596	2014-07-09 00:00:00	534333.333333	3.0	2.50	1460	7573	2.0	0	0	3	
4597	2014-07-09 00:00:00	416904.166667	3.0	2.50	3010	7014	2.0	0	0	3	
4598	2014-07-10 00:00:00	203400.000000	4.0	2.00	2090	6630	1.0	0	0	3	
4599	2014-07-10 00:00:00	220600.000000	3.0	2.50	1490	8102	2.0	0	0	4	


df.shape

 (4600, 18)

df.columns


 Index(['date', 'price', 'bedrooms', 'bathrooms', 'sqft\_living', 'sqft\_lot', 'floors', 'waterfront', 'view', 'condition', 'sqft\_above', 'sqft\_basement', 'yr\_built', 'yr\_renovated', 'street', 'city', 'statezip', 'country'], dtype='object')

df.describe()




	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition
count	4.600000e+03	4600.000000	4600.000000	4600.000000	4.600000e+03	4600.000000	4600.000000	4600.000000	4600.000000
mean	5.519630e+05	3.400870	2.160815	2139.346957	1.485252e+04	1.512065	0.007174	0.240652	3.451739
std	5.638347e+05	0.908848	0.783781	963.206916	3.588444e+04	0.538288	0.084404	0.778405	0.677230
min	0.000000e+00	0.000000	0.000000	370.000000	6.380000e+02	1.000000	0.000000	0.000000	1.000000
25%	3.228750e+05	3.000000	1.750000	1460.000000	5.000750e+03	1.000000	0.000000	0.000000	3.000000
50%	4.609435e+05	3.000000	2.250000	1980.000000	7.683000e+03	1.500000	0.000000	0.000000	3.000000
75%	6.549625e+05	4.000000	2.500000	2620.000000	1.100125e+04	2.000000	0.000000	0.000000	4.000000
max	2.659000e+07	9.000000	8.000000	13540.000000	1.074218e+06	3.500000	1.000000	4.000000	5.000000

df.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
```

df.dtypes



```
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
```


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

	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	

```
df.duplicated().sum()
```

	0
---	---

```
df.nunique(axis=0)
```

	date	70
	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
	street	4525
	city	44
	statezip	77
	country	1
	dtype: int64	

```
df.dropna(inplace=True)
(df.price == 0).sum()
```


	49
--	----

```
#df['price'].replace(0,np.NaN,inplace=True)
df.loc[df.price==0,'price']=np.NaN
```

```
(df.price == 0).sum()
```

	0
--	---


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



date	0
price	49
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

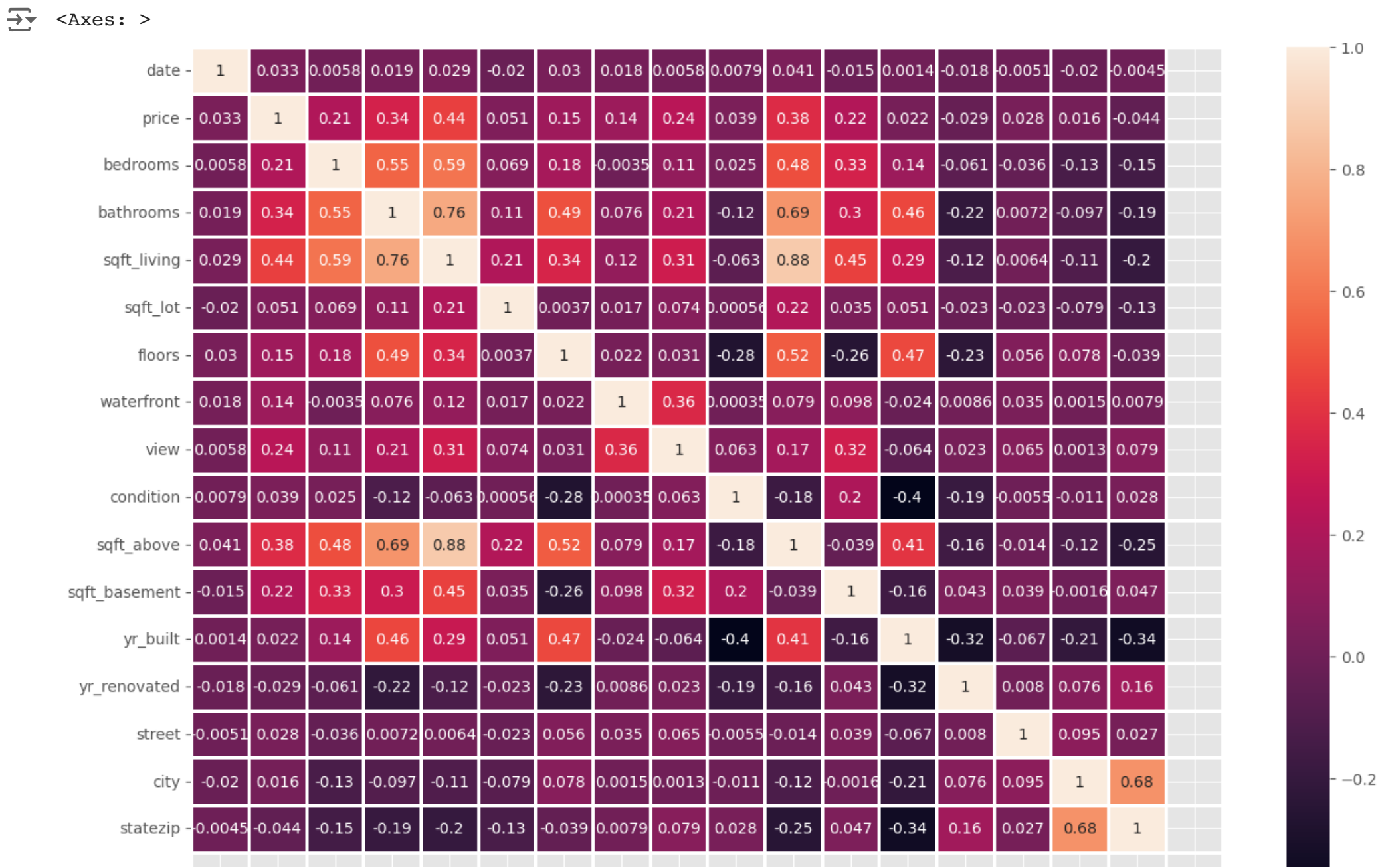
```
df['price']=df['price'].fillna(df['price'].mean())
```

```
lb=LabelEncoder()  
df['street']=lb.fit_transform(df['street'])  
df['city']=lb.fit_transform(df['city'])  
df['statezip']=lb.fit_transform(df['statezip'])  
df['country']=lb.fit_transform(df['country'])  
df['date']=lb.fit_transform(df['date'])  
df.dtypes
```

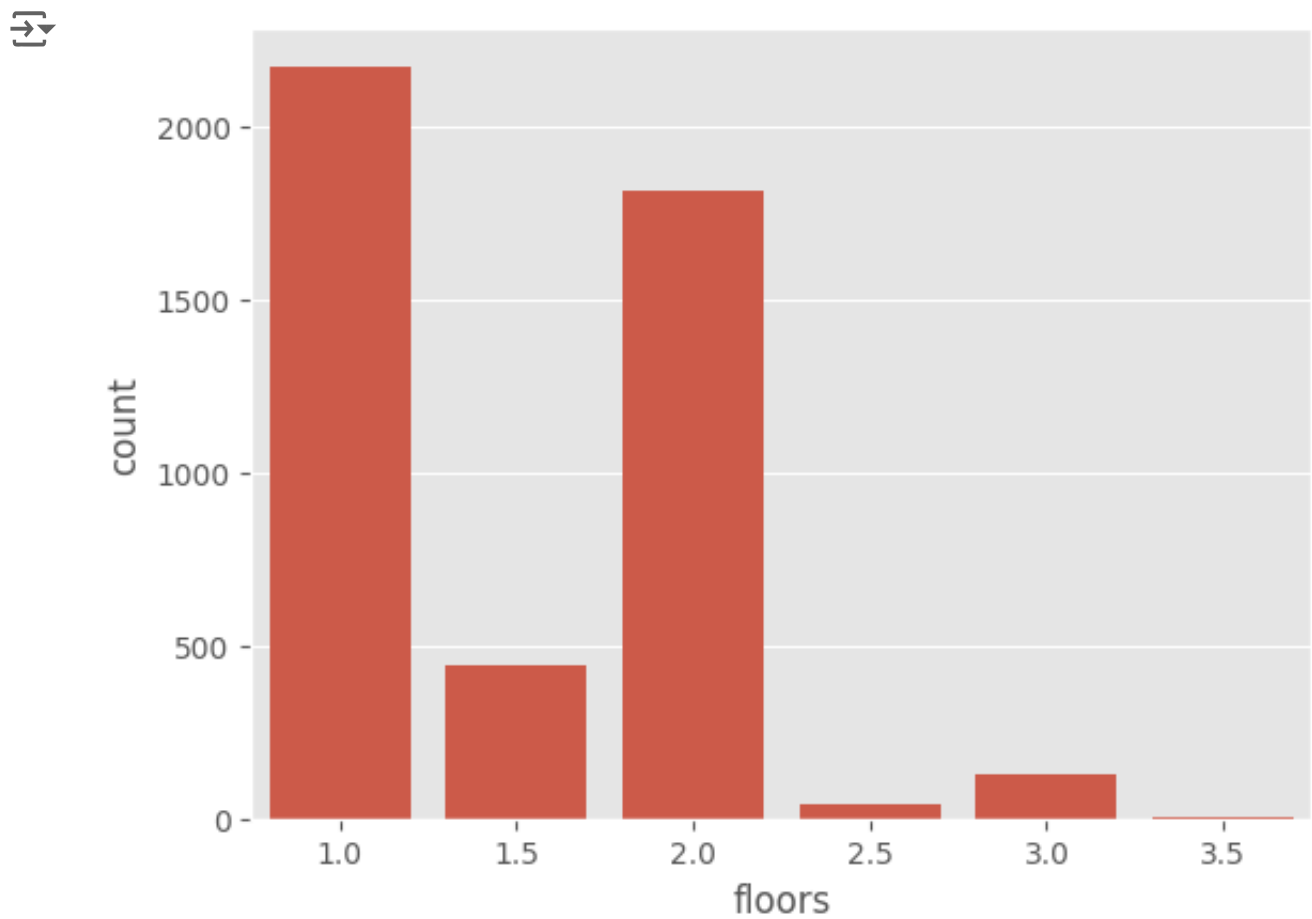


date	int64
price	float64
bedrooms	float64
bathrooms	float64
sqft_living	int64
sqft_lot	int64
floors	float64
waterfront	int64
view	int64
condition	int64
sqft_above	int64
sqft_basement	int64
yr_built	int64
yr_renovated	int64
street	int64
city	int64
statezip	int64
country	int64
dtype:	object

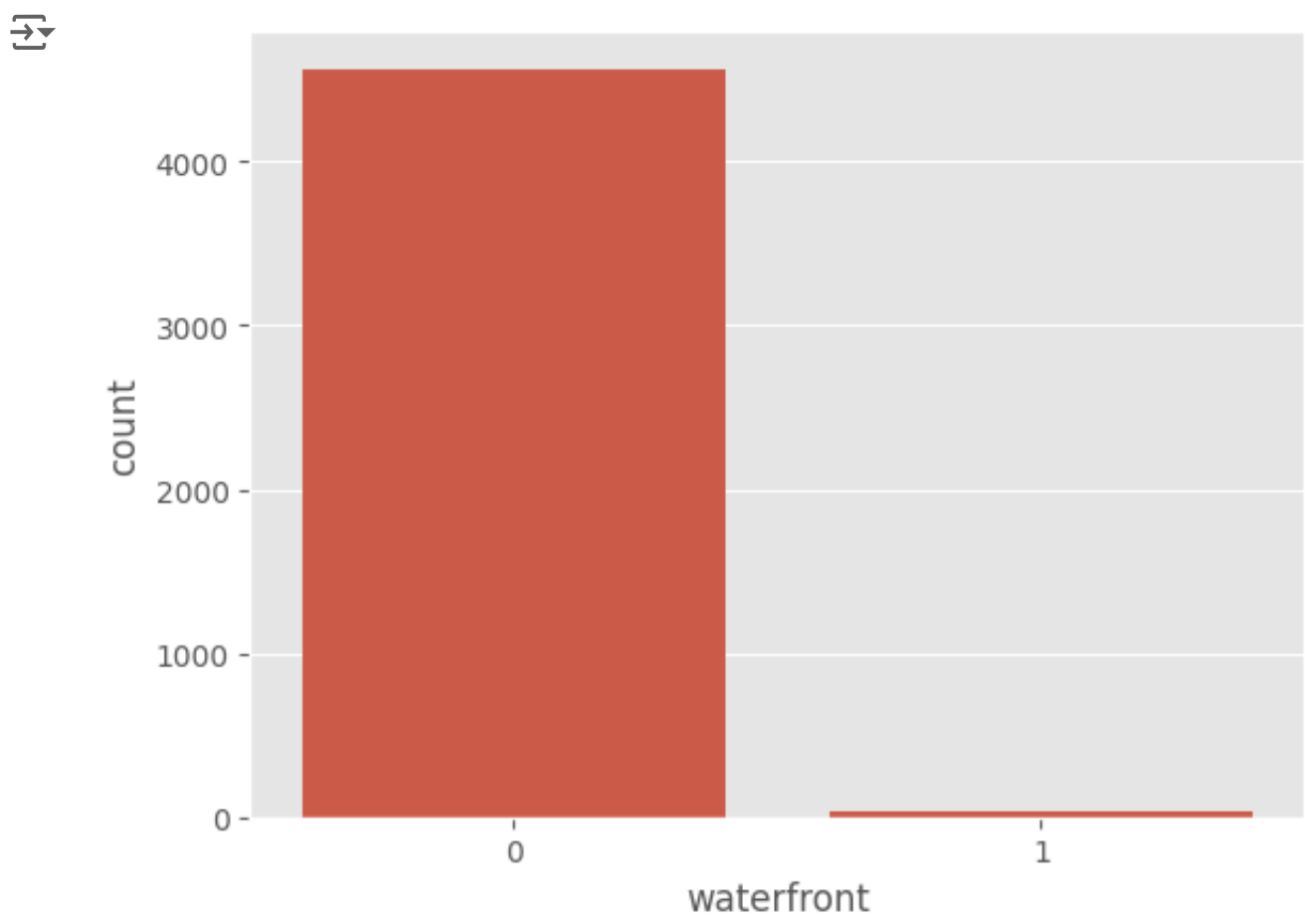
```
plt.figure(figsize=(15,10))
sns.heatmap(df.corr(numeric_only=True),annot=True,linewidth=1)
```



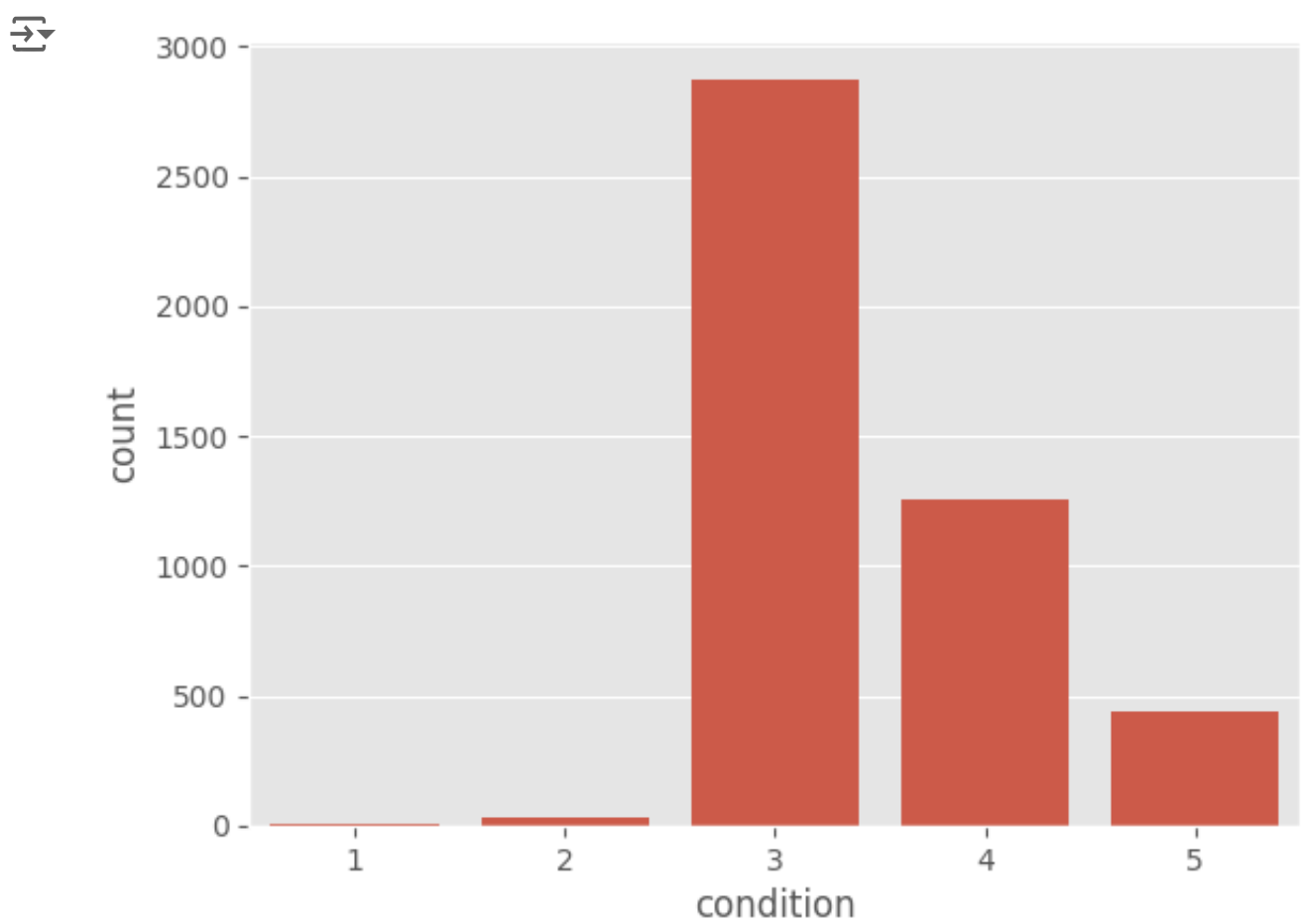
```
sns.countplot(data=df,x=df['floors'])
plt.show()
```



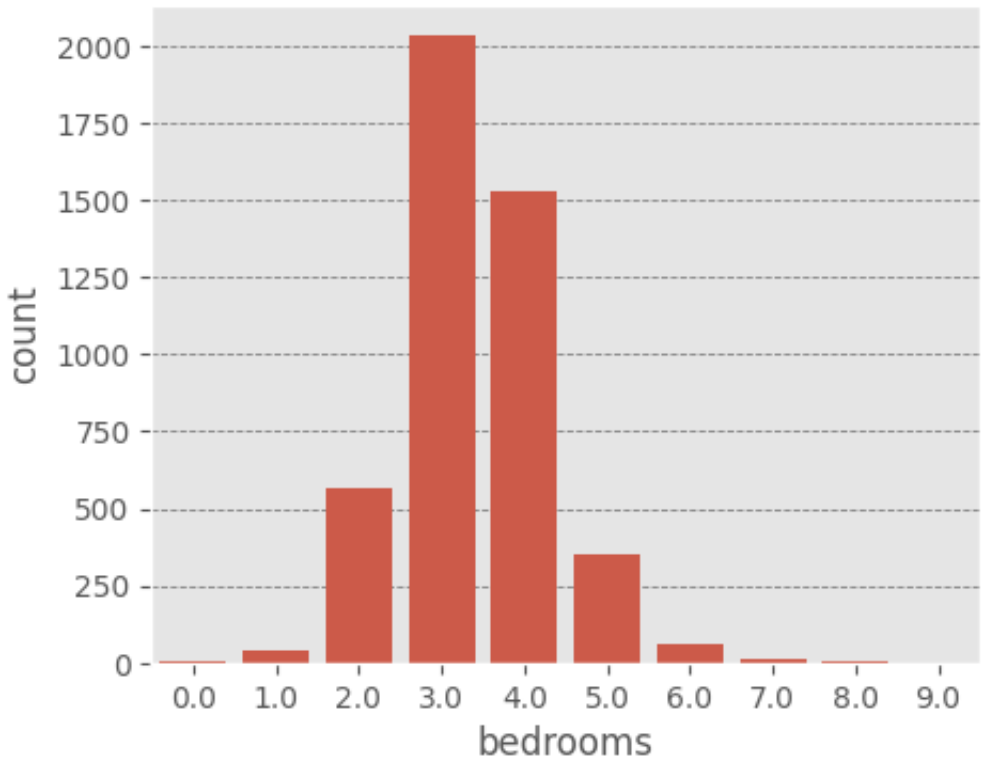
```
sns.countplot(data=df,x=df['waterfront'])
plt.show()
```



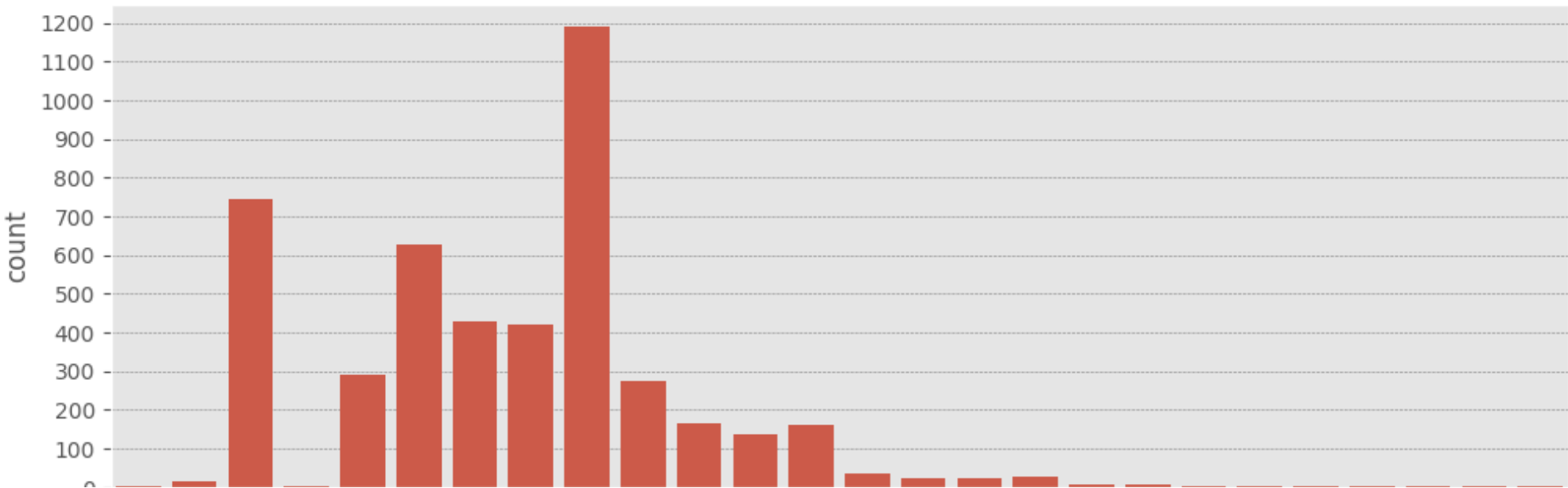
```
sns.countplot(data=df,x=df['condition'])
plt.show()
```



```
#univariate analysis
plt.figure(figsize=(5,4))
sns.countplot(data = df,x = "bedrooms")
plt.grid(axis="y",color="grey",linestyle="--",linewidth=0.6)
plt.style.use("ggplot")
plt.show()
```

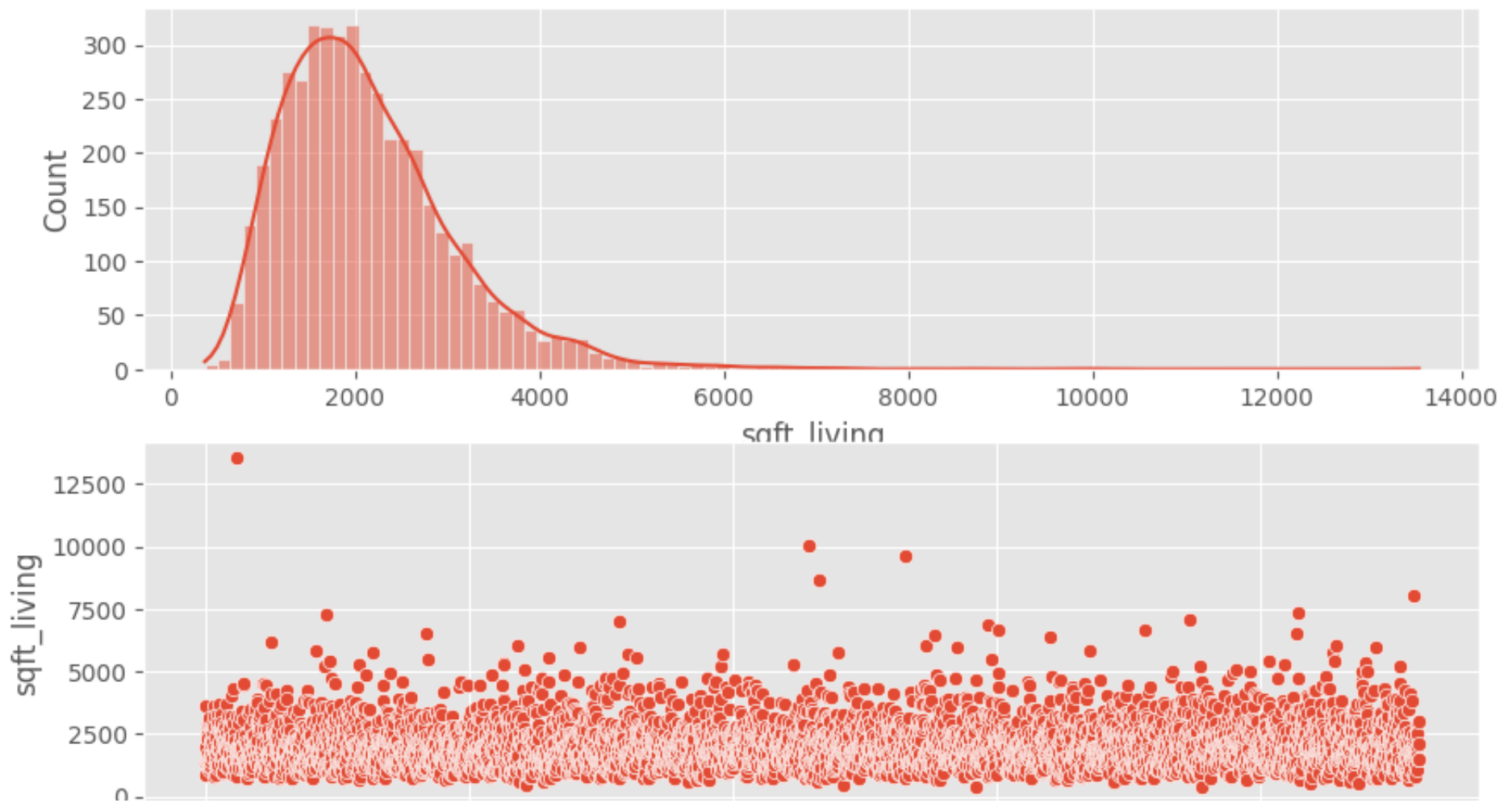


```
plt.figure(figsize=(12,4))
sns.countplot(data = df,x = "bathrooms")
plt.grid(axis="y",color="grey",linestyle="--",linewidth=0.4)
plt.yticks(range(0,1300,100))
plt.style.use("ggplot")
plt.show()
```





```
fig,axs = plt.subplots(2,1,figsize=(10,6))
plt.subplot(2,1,1)
sns.histplot(data=df,x="sqft_living",kde=True)
plt.ticklabel_format(style='plain', axis='x')
plt.subplot(2,1,2)
sns.scatterplot(data=df,x=df.index,y="sqft_living")
plt.ticklabel_format(style='plain', axis='y')
plt.show()
```



# Removing skewness from data

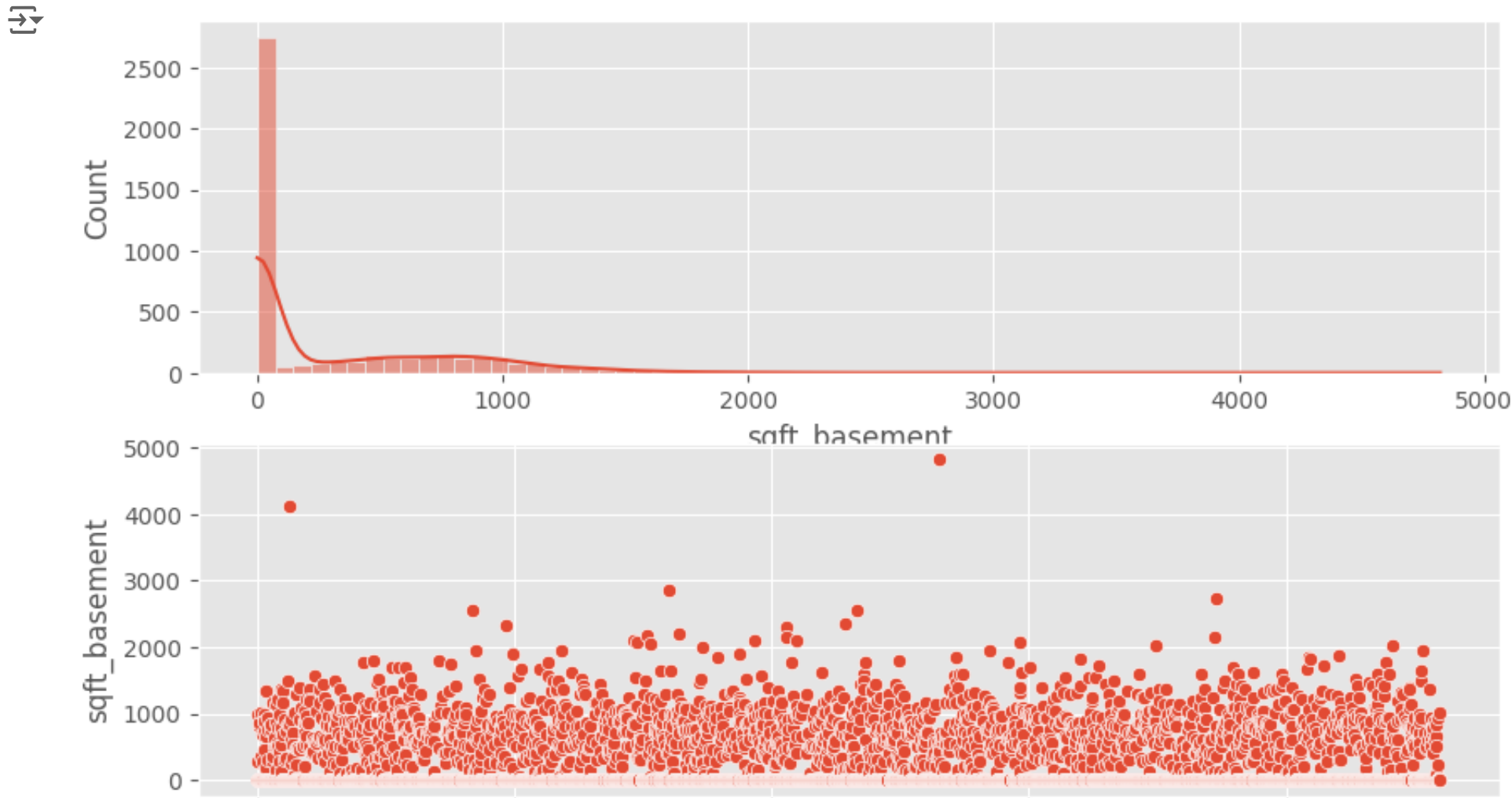
```
print("Before transformation skew : ",df["sqft_living"].skew())
df["sqft_living"] = np.log(df["sqft_living"])
print("After transformation skew : ",df["sqft_living"].skew())
```



```
Before transformation skew : 1.723513270622118
After transformation skew : -0.04949693742696049
```

```
fig,axs = plt.subplots(2,1,figsize=(10,6))
plt.subplot(2,1,1)
sns.histplot(data=df,x="sqft_basement",kde=True)
plt.ticklabel_format(style='plain', axis='x')

plt.subplot(2,1,2)
sns.scatterplot(data=df,x=df.index,y="sqft_basement")
plt.ticklabel_format(style='plain', axis='y')
plt.show()
```



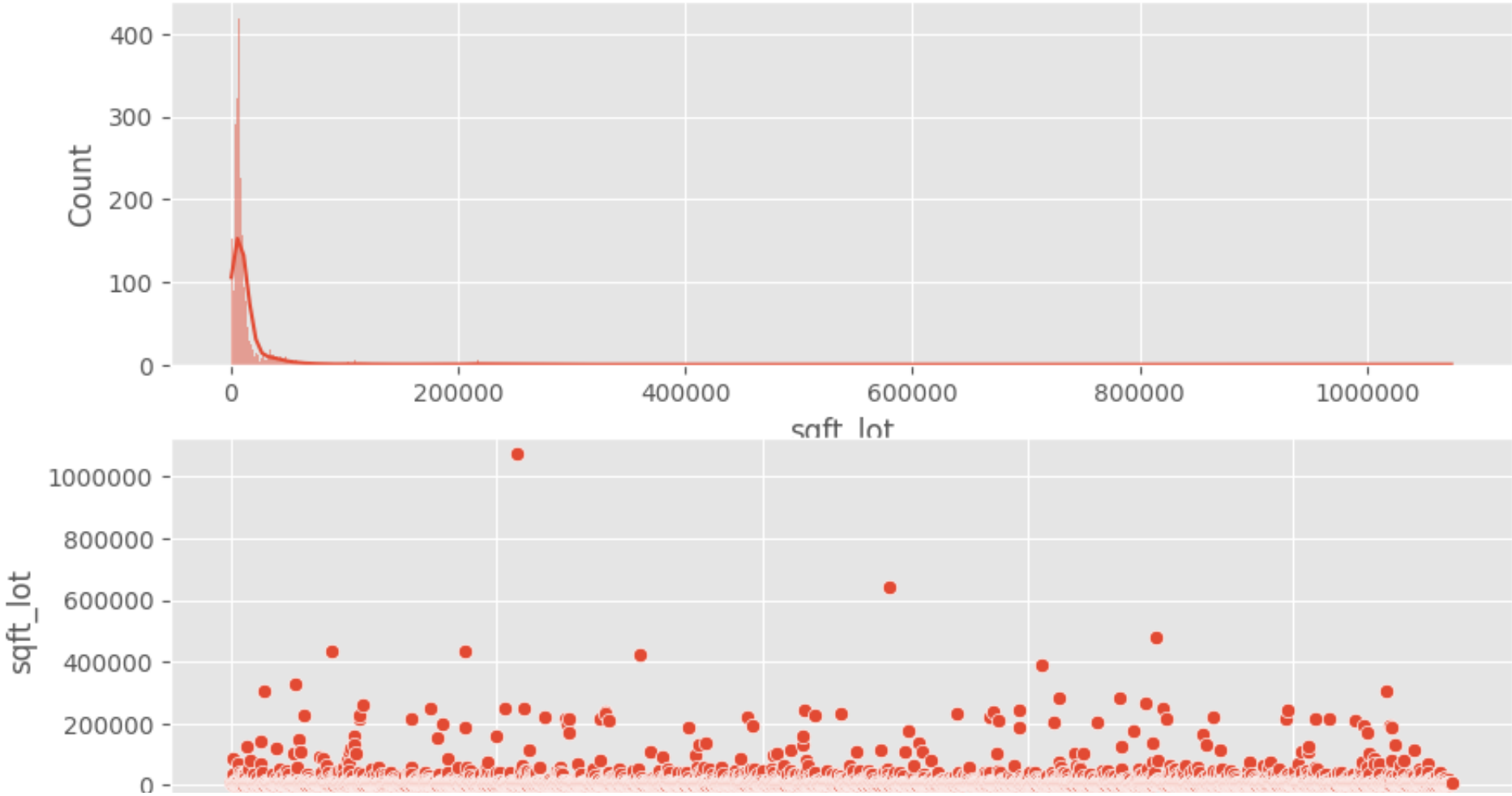
```
# Removing the skeness from data

print("Before transformation skew : ",df["sqft_basement"].skew())
df["sqft_basement"] = np.cbrt(df["sqft_basement"])
print("After transformation skew : ",df["sqft_basement"].skew())
```

Before transformation skew : 1.6427321922167097  
After transformation skew : 0.5627140628962356

```
fig,axs = plt.subplots(2,1,figsize=(10,6))
plt.subplot(2,1,1)
sns.histplot(data=df,x="sqft_lot",kde=True)
plt.ticklabel_format(style='plain', axis='x')

plt.subplot(2,1,2)
sns.scatterplot(data=df,x=df.index,y="sqft_lot")
plt.ticklabel_format(style='plain', axis='y')
plt.show()
```



# Removing the skewness from data

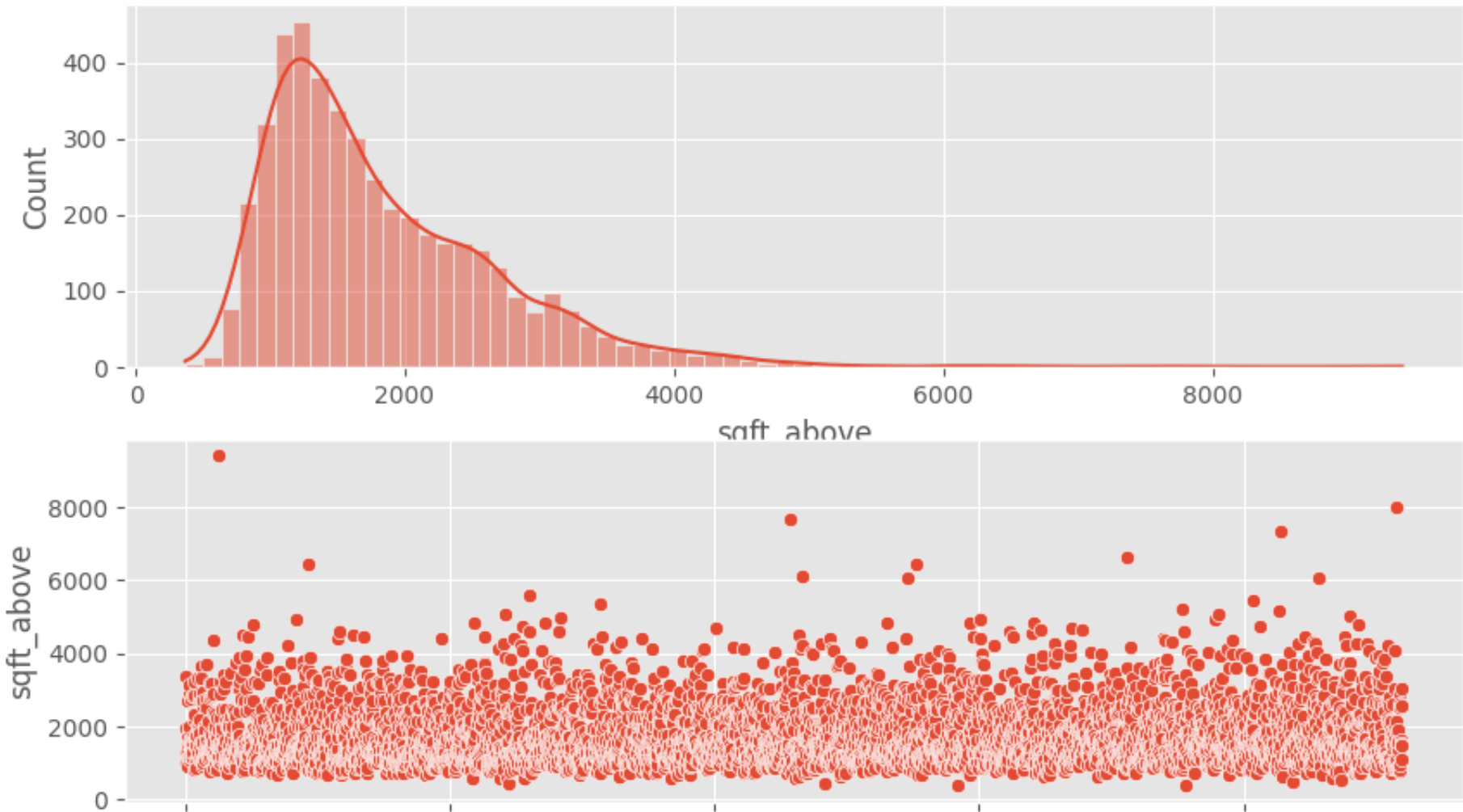
```
print("Before transformation skew : ",df["sqft_lot"].skew())
df["sqft_lot"] = np.log(df["sqft_lot"])
print("After transformation skew : ",df["sqft_lot"].skew())
```



Before transformation skew : 11.307138748782643  
After transformation skew : 0.8416221290325118

```
fig,axs = plt.subplots(2,1,figsize=(10,6))
plt.subplot(2,1,1)
sns.histplot(data=df,x="sqft_above",kde=True)
plt.ticklabel_format(style='plain', axis='x')

plt.subplot(2,1,2)
sns.scatterplot(data=df,x=df.index,y="sqft_above")
plt.ticklabel_format(style='plain', axis='y')
plt.show()
```



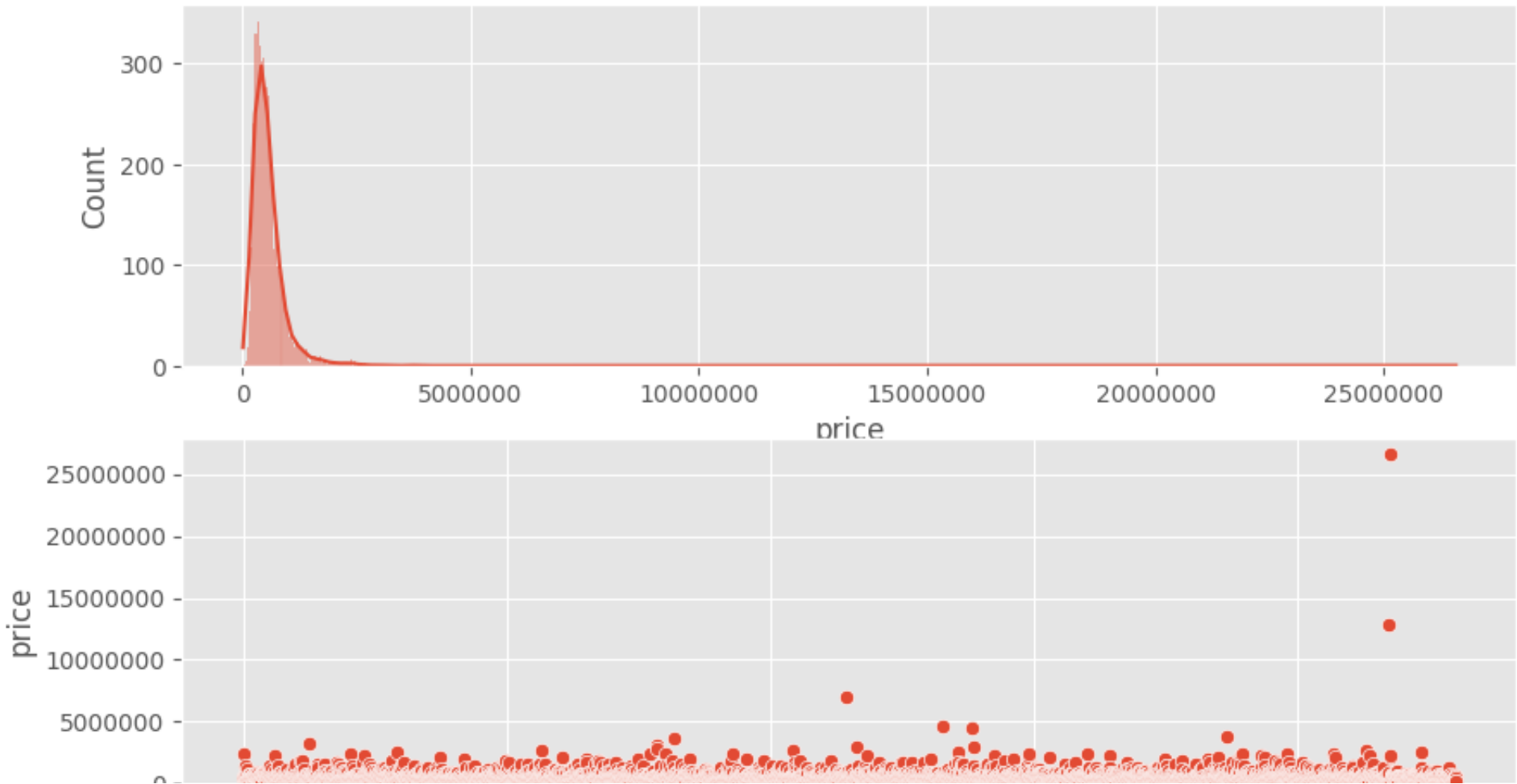
```
# Removing the skewness from data

print("Before transformation skew : ",df["sqft_above"].skew())
df["sqft_above"] = np.log(df["sqft_above"])
print("After transformation skew : ",df["sqft_above"].skew())

Before transformation skew : 1.4942107479829443
After transformation skew : 0.24540379930305353
```

```
fig,axs = plt.subplots(2,1,figsize=(10,6))
plt.subplot(2,1,1)
sns.histplot(data=df,x="price",kde=True)
plt.ticklabel_format(style='plain', axis='x')

plt.subplot(2,1,2)
sns.scatterplot(data=df,x=df.index,y="price")
plt.ticklabel_format(style='plain', axis='y')
plt.show()
```

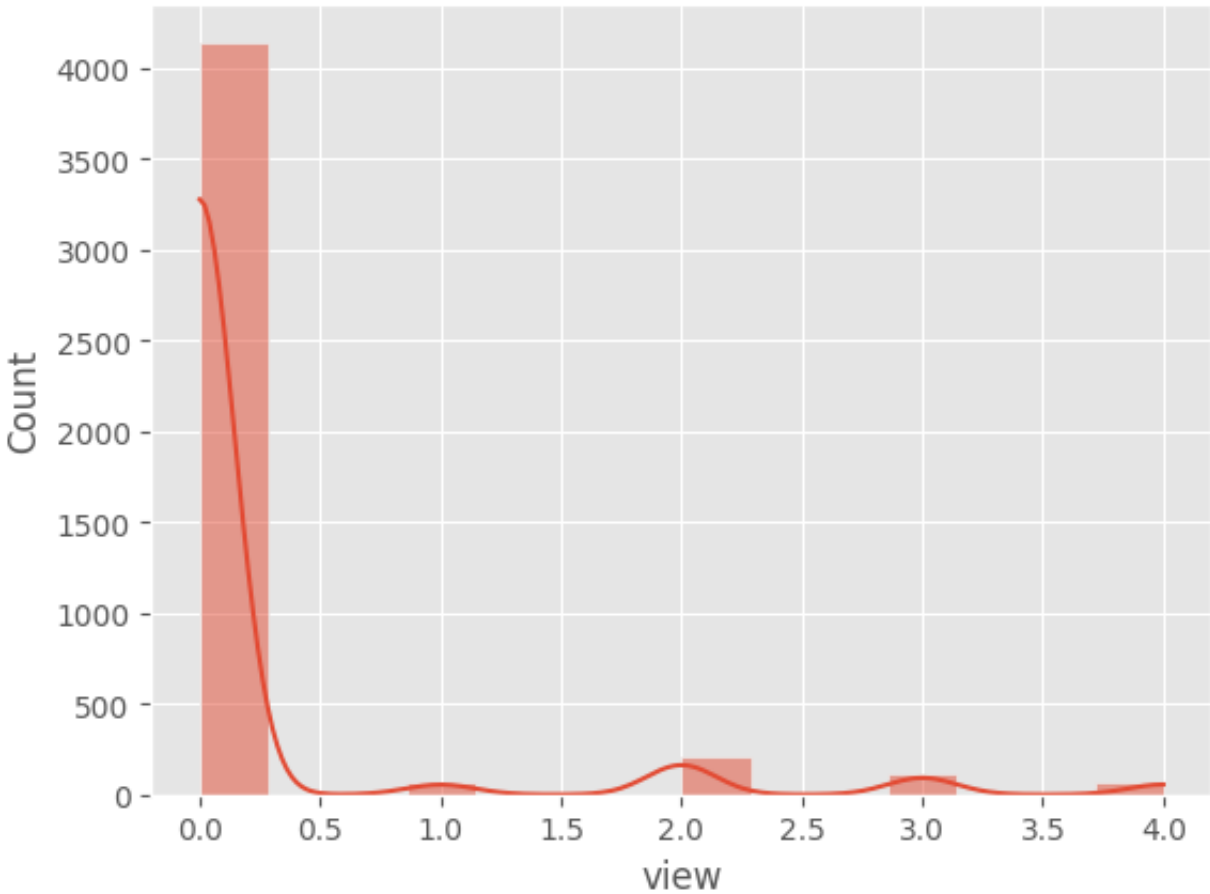


```
print("Before transformation skew : ",df["price"].skew())
df["price"] = np.log(df["price"])
print("After transformation skew : ",df["price"].skew())
```

Before transformation skew : 25.158082239502786  
After transformation skew : 0.3216824447420246

```
sns.histplot(data=df,x="view",kde=True)
```

<Axes: xlabel='view', ylabel='Count'>



```
# Removing the skewness

print("Before transformation skew : ",df["view"].skew())
df["view"] = np.cbrt(df["view"])
print("After transformation skew : ",df["view"].skew())
```

↗ Before transformation skew : 3.341586380673694  
↗ After transformation skew : 2.7633499389523744

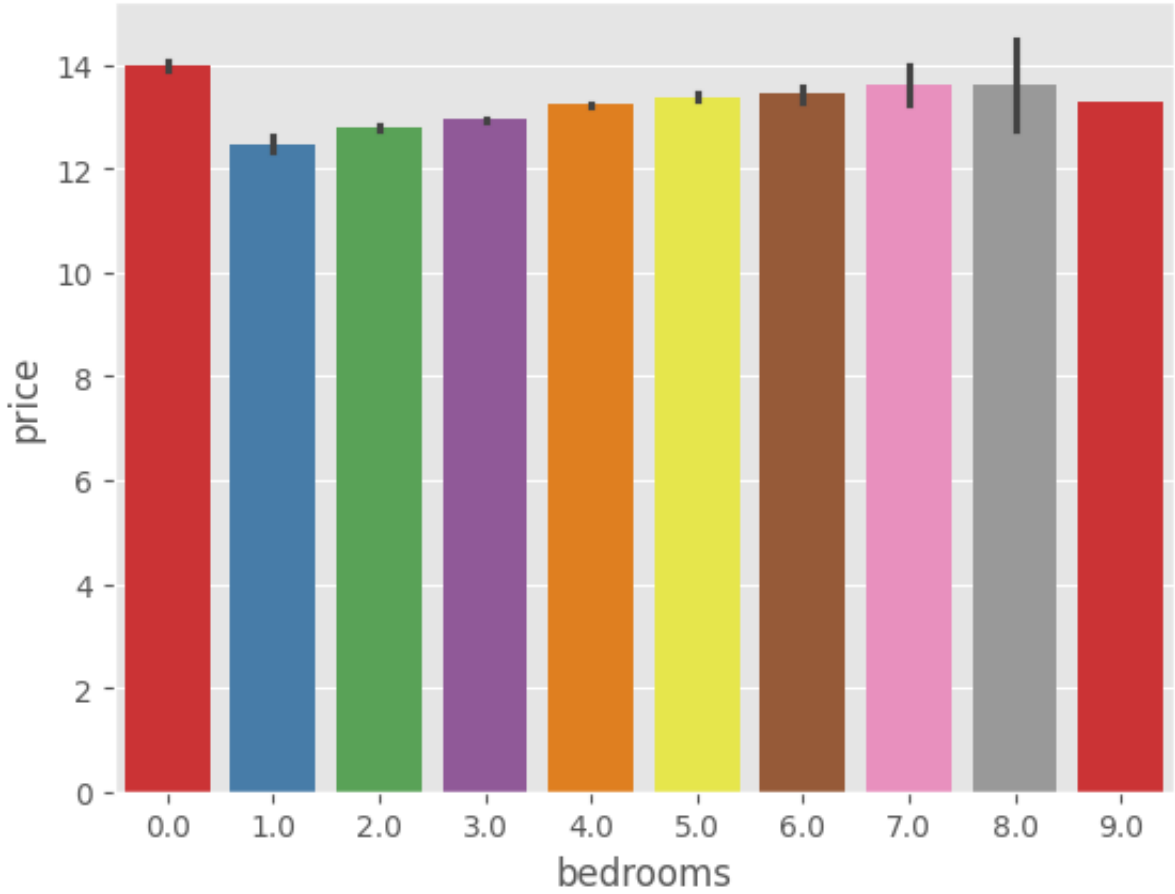
```
sns.barplot(data=df,y="price",x="bedrooms", palette='Set1')
plt.ticklabel_format(style='plain', axis='y')

plt.show()
```

↗ <ipython-input-139-6d700b4f6c53>:1: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable

```
sns.barplot(data=df,y="price",x="bedrooms", palette='Set1')
```

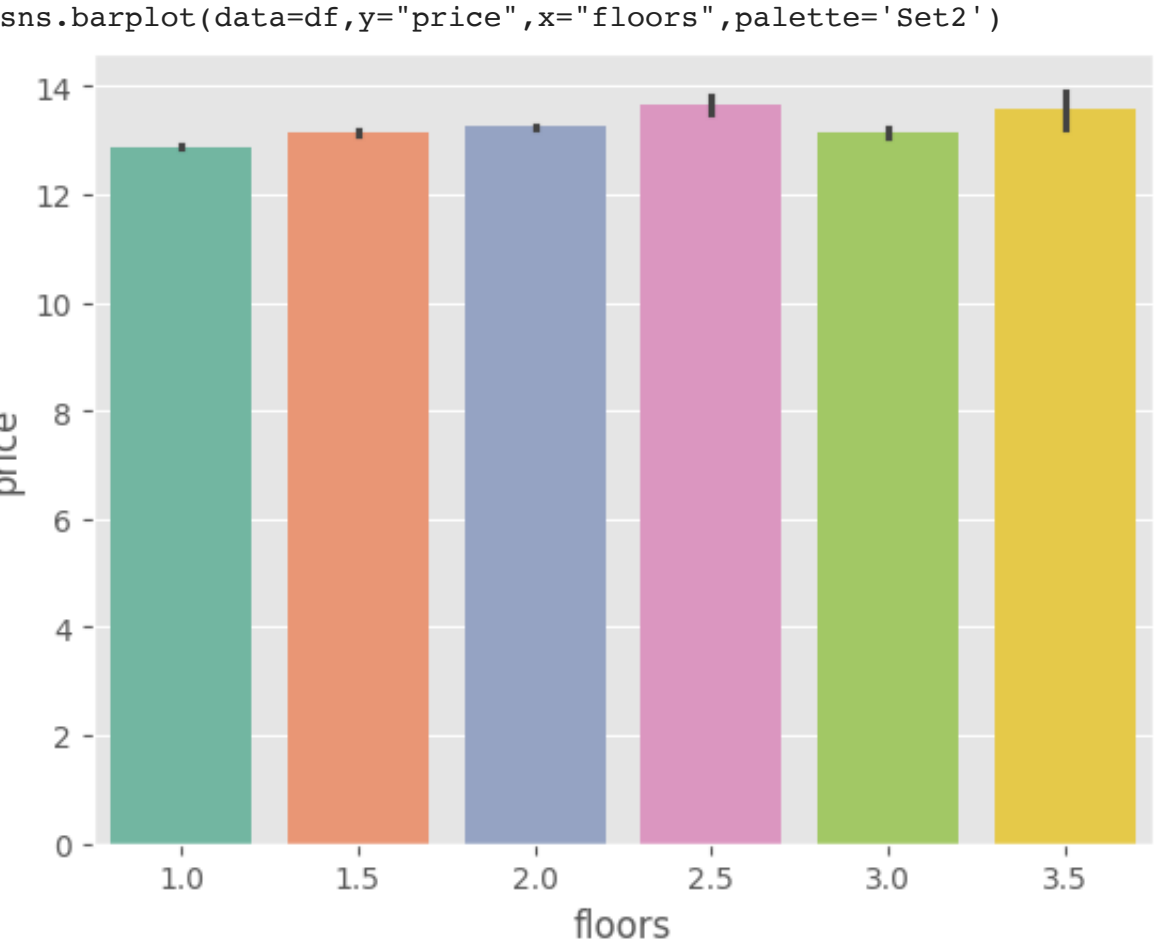


```
sns.barplot(data=df,y="price",x="floors",palette='Set2')
plt.ticklabel_format(style='plain', axis='y')


plt.show()
```

 <ipython-input-140-3165d78db1b3>:1: FutureWarning:

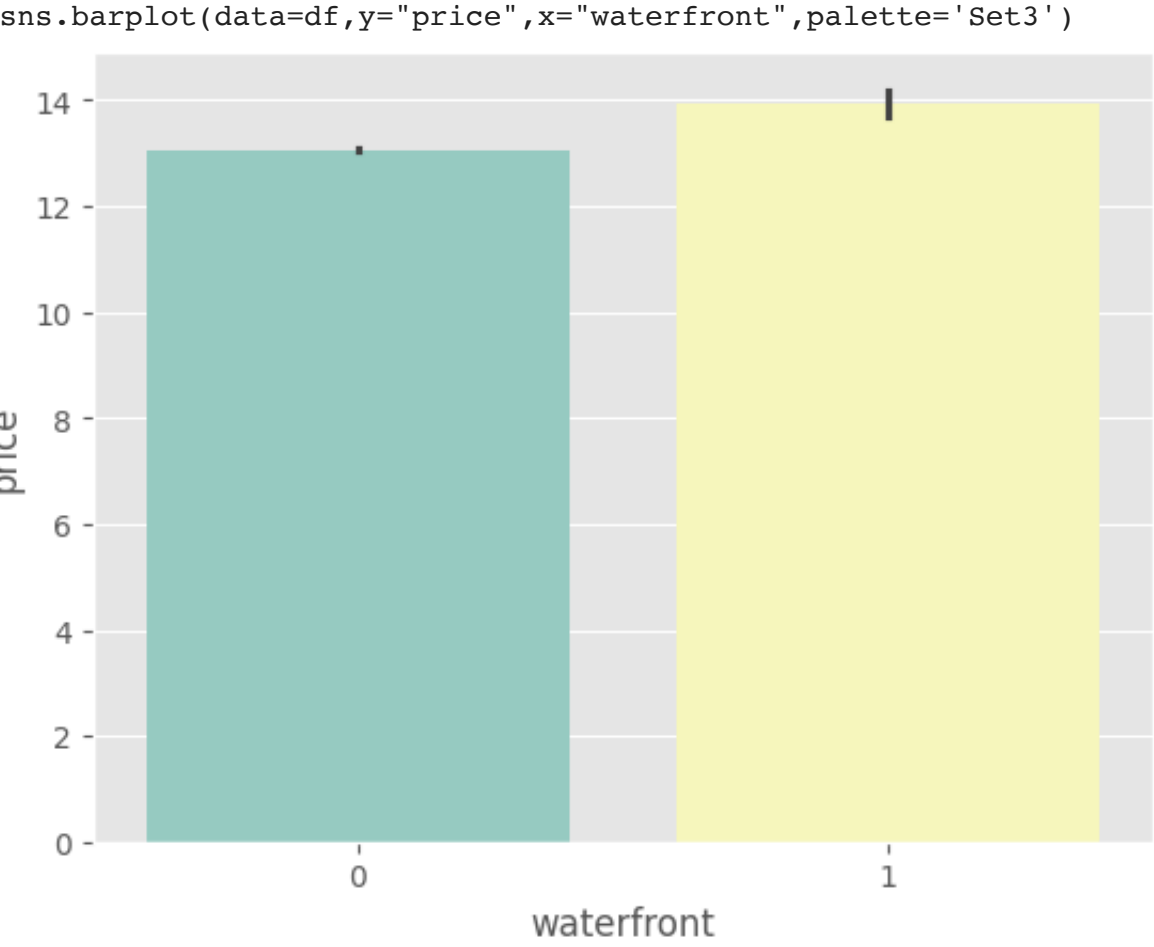
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable



```
sns.barplot(data=df,y="price",x="waterfront",palette='Set3')
plt.ticklabel_format(style='plain', axis='y')
plt.show()
```


 <ipython-input-141-90507285bb24>:1: FutureWarning:

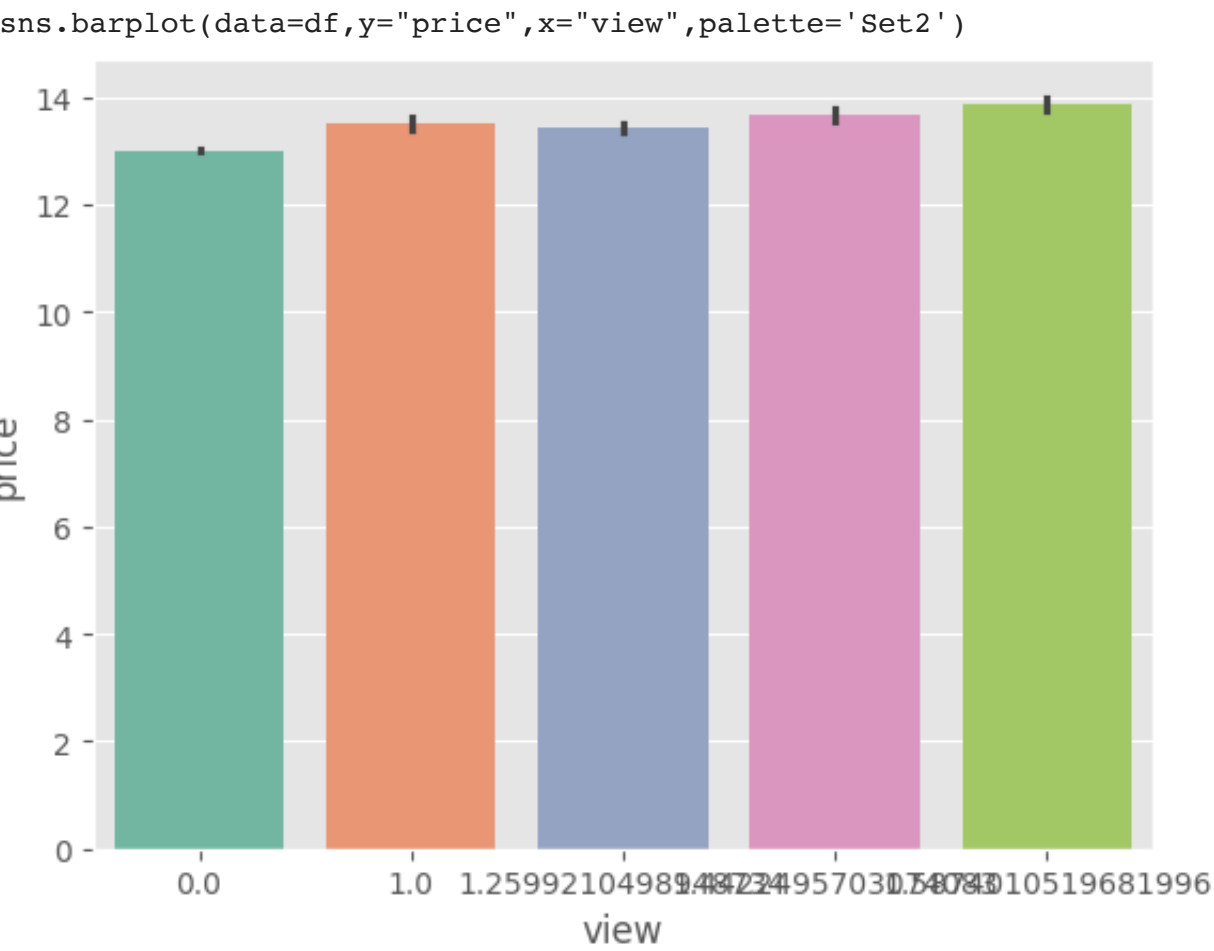
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable




```
sns.barplot(data=df,y="price",x="view",palette='Set2')
plt.ticklabel_format(style='plain', axis='y')

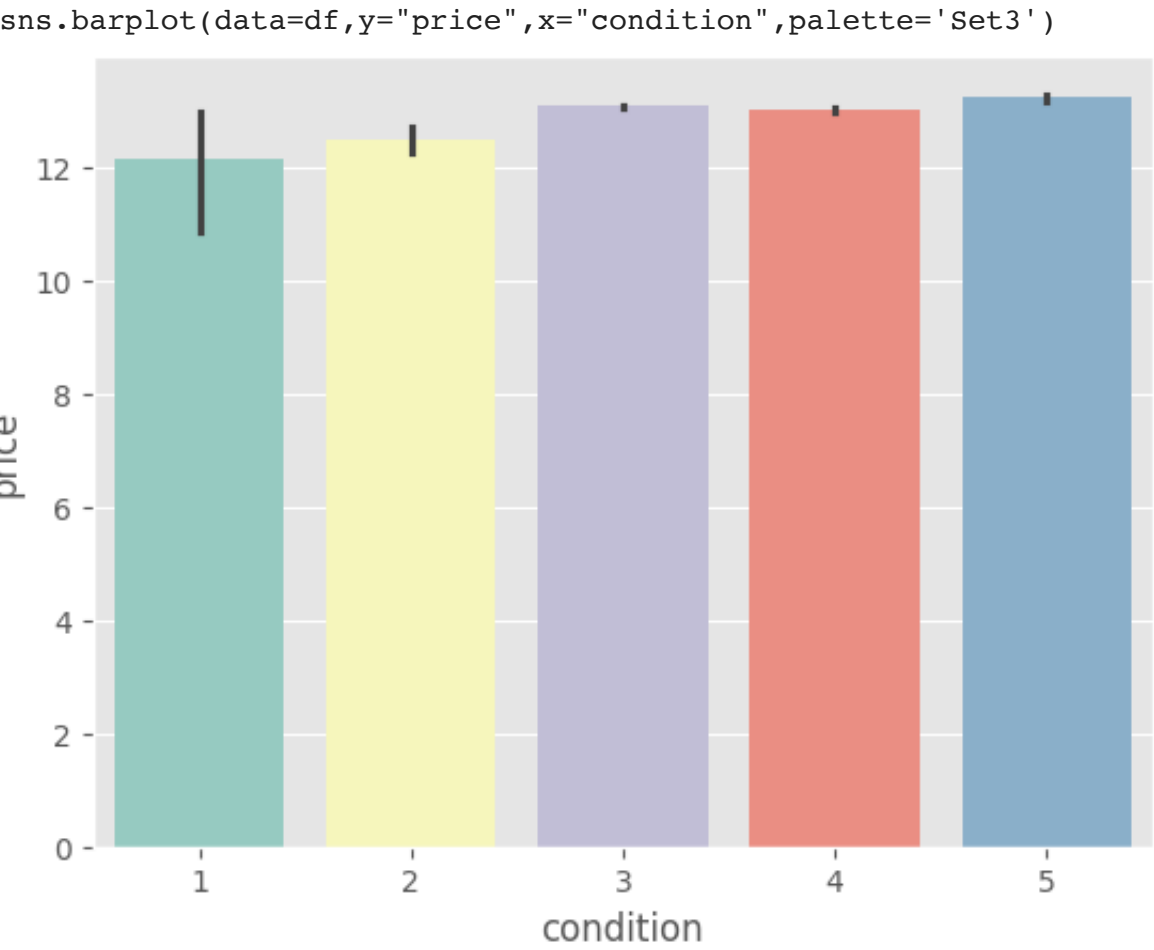
plt.show()
```

 <ipython-input-142-833f7590f7f8>:1: FutureWarning: Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable



```
sns.barplot(data=df,y="price",x="condition",palette='Set3')
plt.ticklabel_format(style='plain', axis='y')
plt.show()
```

 <ipython-input-143-e2287f8bcd8a>:1: FutureWarning: Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable





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

```
x= df.drop(columns='price',axis=1)
x
```

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement
0	3.0	1.50	7.200425	8.976136	1.5	0	0.000000	3	7.200425	0.000000
1	5.0	2.50	8.202482	9.110520	2.0	0	1.587401	5	8.122668	6.542133
2	3.0	2.00	7.565275	9.388235	1.0	0	0.000000	4	7.565275	0.000000
3	3.0	2.25	7.600902	8.990940	1.0	0	0.000000	4	6.907755	10.000000
4	4.0	2.50	7.570443	9.259131	1.0	0	0.000000	4	7.038784	9.283178
...	...	...	...	...	...	...	...	...	...	...
4595	3.0	1.75	7.319865	8.757784	1.0	0	0.000000	4	7.319865	0.000000
4596	3.0	2.50	7.286192	8.932345	2.0	0	0.000000	3	7.286192	0.000000
4597	3.0	2.50	8.009695	8.855663	2.0	0	0.000000	3	8.009695	0.000000
4598	4.0	2.00	7.644919	8.799360	1.0	0	0.000000	3	6.975414	10.066227
4599	3.0	2.50	7.306531	8.999866	2.0	0	0.000000	4	7.306531	0.000000

4600 rows × 13 columns

Next steps:

[Generate code with x](#)



[View recommended plots](#)


```
y=df['price']
y
```

0	12.653958
1	14.684290
2	12.742566
3	12.948010
4	13.217674
...	...
4595	12.638396
4596	13.188775
4597	12.940612
4598	12.222930
4599	12.304106

Name: price, Length: 4600, dtype: float64

✖ Train/Test split

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=0)
x_train
```



	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement
4452	3.0	1.00	6.946976	8.833463	2.0	0	0.00000	3	6.946976	0.000000
2457	5.0	2.75	8.107720	10.075759	2.0	0	0.00000	4	7.691657	10.415804
1390	3.0	2.25	7.770645	12.291332	2.0	0	0.00000	3	7.770645	0.000000
3402	4.0	1.75	7.642524	12.068189	1.0	0	0.00000	3	7.383989	7.802454
3197	4.0	2.50	7.803843	10.922100	2.0	0	0.00000	3	7.803843	0.000000
...	...	...	...	...	...	...	...	...	...	...
1033	3.0	1.50	7.146772	7.274480	3.0	0	0.00000	3	7.146772	0.000000
3264	2.0	1.00	6.877296	8.612503	1.0	0	0.00000	3	6.877296	0.000000
1653	5.0	2.75	7.640123	9.487138	2.0	0	0.00000	3	7.640123	0.000000
2607	4.0	2.50	8.029433	10.446161	1.0	0	1.44225	4	7.635304	10.000000
2732	3.0	1.75	7.438384	9.039789	1.0	0	0.00000	3	7.114769	7.774980


3220 rows × 13 columns

Next steps:

Generate code with x\_train

 View recommended plots

x\_test



	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	y
991	3.0	2.50	7.644919	8.455318	2.0	0	0.0	3	7.644919	0.000000	
2824	4.0	2.50	7.878534	9.062420	2.0	0	0.0	3	7.878534	0.000000	
1906	1.0	1.00	6.476972	9.639782	1.0	0	0.0	4	6.476972	0.000000	
1471	4.0	2.00	7.828038	10.549045	1.0	0	0.0	3	7.828038	0.000000	
1813	4.0	3.50	7.933797	9.222763	1.5	0	0.0	5	7.933797	0.000000	
...	...	...	...	...	...	...	...	...	...	...	...
1523	2.0	2.50	6.956545	7.383368	2.0	0	0.0	3	6.956545	0.000000	
144	3.0	3.00	7.359468	7.609367	3.0	0	0.0	3	7.359468	0.000000	
388	4.0	2.25	8.349957	10.539244	2.0	0	0.0	3	8.349957	0.000000	
4395	4.0	1.00	7.333023	8.881836	1.5	0	0.0	3	7.244228	5.065797	
1669	2.0	1.00	7.098376	8.476371	1.0	0	0.0	3	6.966024	5.313293	

1380 rows × 13 columns

Next steps:

Generate code with x\_test

 View recommended plots

y\_train

```
↵ 4452    11.909098
    2457    13.341499
    1390    13.282780
    3402    13.241923
    3197    13.337475
    ...
    1033    12.994530
    3264    12.254863
    1653    13.197263
    2607    14.467836
    2732    13.071070
    Name: price, Length: 3220, dtype: float64
```

y\_test

```
↵ 991     12.574182
    2824    12.971308
    1906    11.767568
    1471    13.304685
    1813    14.076335
    ...
    1523    12.691580
    144     12.812436
    388     13.800397
    4395    11.815496
    1669    12.847927
    Name: price, Length: 1380, dtype: float64
```

## ✓ Normalization

```
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.fit_transform(x_test)
x_train
```

```
↵ array([[ -0.43585706, -1.46116072, -1.44531812, ..., -0.97908343,
          1.21146099,  0.77814633],
        [ 1.81508077,  0.76024509,  1.24451291, ...,  0.13894917,
        -0.82861952,  1.3632617 ],
        [-0.43585706,  0.12555772,  0.46339815, ...,  0.27446827,
          1.22676924,  0.44379469],
        ...,
        [ 1.81508077,  0.76024509,  0.16093501, ...,  0.54550647,
          1.21248154,  0.61097051],
        [ 0.68961185,  0.44290141,  1.06309532, ..., -0.70804522,
          1.19513218, -0.3084965 ],
        [-0.43585706, -0.50912966, -0.30656301, ...,  0.20670872,
          1.21656374,  0.77814633]])
```

x\_test

```
↵ array([[ -0.45368999,  0.40911564,  0.12981037, ...,  1.04166915,
        -0.81889088, -1.38368358],
        [ 0.59504064,  0.40911564,  0.67863687, ...,  0.54525937,
          1.22616715,  0.52929302],
        [-2.55115124, -1.52990983, -2.61402313, ..., -0.11662032,
        -0.81889088, -0.63512752],
        ...,
        [ 0.59504064,  0.08594473,  1.78614127, ...,  0.61144734,
        -0.81889088,  1.36102198],
        [ 0.59504064, -1.52990983, -0.60292118, ..., -0.74540604,
          1.22003197,  0.77881171],
        [-1.50242061, -1.52990983, -1.15417337, ..., -0.67921807,
          1.23127979,  0.77881171]])
```

## Model creation

Linear Regression

```
lr=LinearRegression()
lr.fit(x_train,y_train)
y_pred=lr.predict(x_test)
y_pred
```

array([12.97344108, 13.29009117, 12.0672723 , ..., 13.61302906,
 12.8589992 , 12.75376323])

```
y_test
```

991 12.574182
2824 12.971308
1906 11.767568
1471 13.304685
1813 14.076335
...
1523 12.691580
144 12.812436
388 13.800397
4395 11.815496
1669 12.847927
Name: price, Length: 1380, dtype: float64

```
mae=mean_absolute_error(y_test,y_pred)
mape=mean_absolute_percentage_error(y_test,y_pred)
mse=mean_squared_error(y_test,y_pred)
rmse=np.sqrt(mse)
r_score=r2_score(y_test,y_pred)
```

```
print("MAE:", mae)
print("MAPE",mape)
print("MSE:", mse)
print("RMSE:", rmse)
print("R2 Score:", r_score)
```

MAE: 0.2846265705186637
MAPE 0.021728867580299297
MSE: 0.14006035188542187
RMSE: 0.37424637858691684
R2 Score: 0.5184802888077391

random forest regression

```
rf=RandomForestRegressor(n_estimators=300, random_state=0)
rf.fit(x_train,y_train)
y_pred=rf.predict(x_test)
y_pred
```

array([12.59002474, 13.05804781, 12.4172821 , ..., 13.56657173,
 12.79312905, 12.86318677])

```
y_test

991      12.574182
2824     12.971308
1906     11.767568
1471     13.304685
1813     14.076335
...
1523     12.691580
144      12.812436
388      13.800397
4395     11.815496
1669     12.847927
Name: price, Length: 1380, dtype: float64
```

```
print("MAE:", mean_absolute_error(y_test,y_pred))
print("MAPE",mean_absolute_percentage_error(y_test,y_pred))
print("MSE:",mean_squared_error(y_test,y_pred))
print("RMSE:", np.sqrt(mse))
print("R2 Score:", r2_score(y_test,y_pred))
```

```
MAE: 0.22647090576813672
MAPE 0.017262669925558055
MSE: 0.1091462311680321
RMSE: 0.37424637858691684
R2 Score: 0.6247613189437886
```

## Decision Tree Regression

```
dt=DecisionTreeRegressor()
dt.fit(x_train,y_train)
y_pred=dt.predict(x_test)
y_pred

array([12.5776362 , 13.07107008, 12.60819885, ..., 12.91195094,
       13.23194559, 13.48422485])
```

```
y_test

991      12.574182
2824     12.971308
1906     11.767568
1471     13.304685
1813     14.076335
...
1523     12.691580
144      12.812436
388      13.800397
4395     11.815496
1669     12.847927
Name: price, Length: 1380, dtype: float64
```

```
print("MAE:", mean_absolute_error(y_test,y_pred))
print("MAPE",mean_absolute_percentage_error(y_test,y_pred))
print("MSE:",mean_squared_error(y_test,y_pred))
print("RMSE:", np.sqrt(mse))
print("R2 Score:", r2_score(y_test,y_pred))
```

```
MAE: 0.32709069418316755
MAPE 0.024925561267985362
MSE: 0.23550563025546045
RMSE: 0.37424637858691684
R2 Score: 0.1903447225555348
```

## Gradient boosting Regression

```
GBoost = GradientBoostingRegressor(n_estimators=5000,random_state =5)
GBoost.fit(x_train, y_train)
pred = GBoost.predict(x_test)
pred
```

```
➞ array([12.51701353, 13.21149603, 12.02828778, ..., 13.51718549,
        13.04713368, 13.21249785])
```

Start coding or [generate](#) with AI.

```
➞ array([12.51701353, 13.21149603, 12.02828778, ..., 13.51718549,
        13.04713368, 13.21249785])
```

```
print("MAE:", mean_absolute_error(y_test,pred))
print("MAPE",mean_absolute_percentage_error(y_test,pred))
print("MSE:",mean_squared_error(y_test,pred))
print("RMSE:", np.sqrt(mse))
print("R2 Score:", r2_score(y_test,pred))
```

```
➞ MAE: 0.23258625989172707
   MAPE 0.017761748561780717
   MSE: 0.12089418475828281
   RMSE: 0.37424637858691684
   R2 Score: 0.5843725069514749
```