In [15]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler,PolynomialFeatures
from sklearn.linear_model import LinearRegression
%matplotlib inline
```

In [5]:
```
df = pd.read_csv(r'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.
df
```

Out[5]:

	Unnamed: 0	id	date	price	bedrooms	bathrooms	sqft
0	0	7129300520	20141013T000000	221900.0	3.0	1.00	
1	1	6414100192	20141209T000000	538000.0	3.0	2.25	
2	2	5631500400	20150225T000000	180000.0	2.0	1.00	
3	3	2487200875	20141209T000000	604000.0	4.0	3.00	
4	4	1954400510	20150218T000000	510000.0	3.0	2.00	
...	...	...	...	...	...	...	
21608	21608	263000018	20140521T000000	360000.0	3.0	2.50	
21609	21609	6600060120	20150223T000000	400000.0	4.0	2.50	
21610	21610	1523300141	20140623T000000	402101.0	2.0	0.75	
21611	21611	291310100	20150116T000000	400000.0	3.0	2.50	
21612	21612	1523300157	20141015T000000	325000.0	2.0	0.75	

21613 rows × 22 columns

Question 1

In [7]:
```
df.dtypes
```

Out[7]:
```
date              object
price            float64
bedrooms         float64
bathrooms        float64
sqft_living        int64
sqft_lot           int64
floors           float64
waterfront         int64
view               int64
condition          int64
grade              int64
sqft_above         int64
sqft_basement      int64
yr_built           int64
yr_renovated       int64
zipcode            int64
lat              float64
long             float64
sqft_living15      int64
sqft_lot15         int64
dtype: object
```

Question 2

In [6]:
```
df.drop(columns=["id","Unnamed: 0"], axis=1, inplace=True)
df.describe()
```

Out[6]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	fl
count	2.161300e+04	21600.000000	21603.000000	21613.000000	2.161300e+04	21613.00
mean	5.400881e+05	3.372870	2.115736	2079.899736	1.510697e+04	1.49
std	3.671272e+05	0.926657	0.768996	918.440897	4.142051e+04	0.53
min	7.500000e+04	1.000000	0.500000	290.000000	5.200000e+02	1.00
25%	3.219500e+05	3.000000	1.750000	1427.000000	5.040000e+03	1.00
50%	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.50
75%	6.450000e+05	4.000000	2.500000	2550.000000	1.068800e+04	2.00
max	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.50

In [8]:
```
print("number of NaN values for the column bedrooms :", df['bedrooms'].isnull().
print("number of NaN values for the column bathrooms :", df['bathrooms'].isnull(
```

```
number of NaN values for the column bedrooms : 13
number of NaN values for the column bathrooms : 10
```

In [9]:
```
mean=df['bedrooms'].mean()
df['bedrooms'].replace(np.nan,mean, inplace=True)
```

In [10]:
```
mean=df['bathrooms'].mean()
df['bathrooms'].replace(np.nan,mean, inplace=True)
```

In [12]:

```
df
```

Out[12]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors
0	20141013T000000	221900.0	3.0	1.00	1180	5650	1.0
1	20141209T000000	538000.0	3.0	2.25	2570	7242	2.0
2	20150225T000000	180000.0	2.0	1.00	770	10000	1.0
3	20141209T000000	604000.0	4.0	3.00	1960	5000	1.0
4	20150218T000000	510000.0	3.0	2.00	1680	8080	1.0
...	...	...	...	...	...	...	...
21608	20140521T000000	360000.0	3.0	2.50	1530	1131	3.0
21609	20150223T000000	400000.0	4.0	2.50	2310	5813	2.0
21610	20140623T000000	402101.0	2.0	0.75	1020	1350	2.0
21611	20150116T000000	400000.0	3.0	2.50	1600	2388	2.0
21612	20141015T000000	325000.0	2.0	0.75	1020	1076	2.0

21613 rows × 20 columns

In [11]:

```
print("number of NaN values for the column bedrooms :", df['bedrooms'].isnull().
print("number of NaN values for the column bathrooms :", df['bathrooms'].isnull(
```

```
number of NaN values for the column bedrooms : 0
number of NaN values for the column bathrooms : 0
```

Question 3

In [13]:

```
df[["floors"]].value_counts().to_frame()
```

Out[13]:

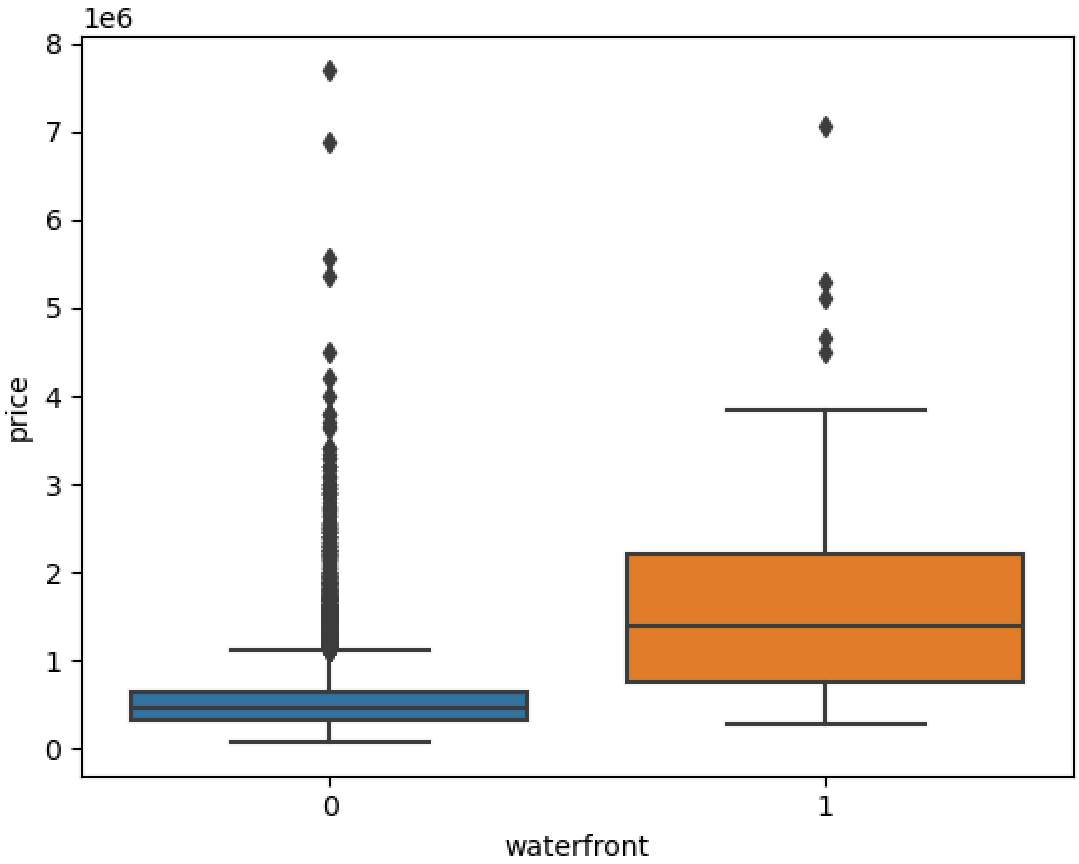
	count
floors	
1.0	10680
2.0	8241
1.5	1910
3.0	613
2.5	161
3.5	8

Question 4

In [16]:

```
sns.boxplot(x="waterfront", y="price", data=df)
```

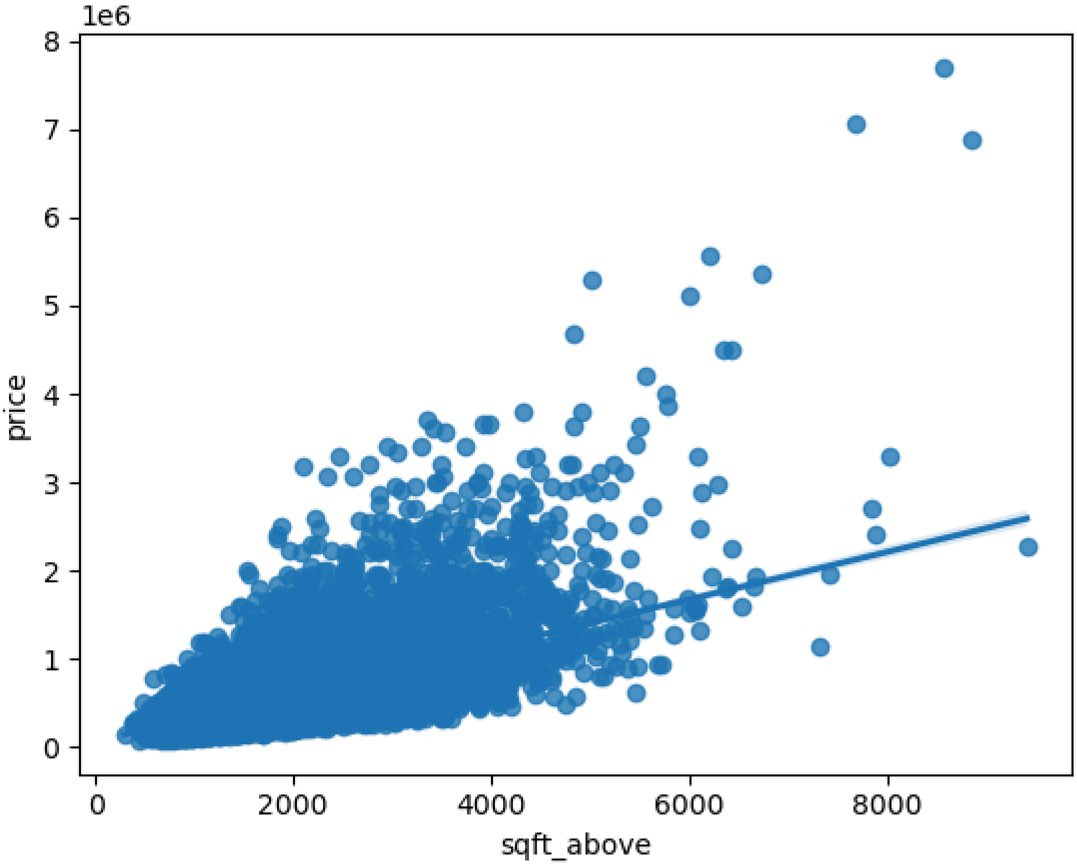
```
Out[16]: <Axes: xlabel='waterfront', ylabel='price'>
```



Question 5

```
In [17]: sns.regplot(x="sqft_above", y="price", data=df)
```

```
Out[17]: <Axes: xlabel='sqft_above', ylabel='price'>
```



In [20]:
```
X = df[['long']]
Y = df['price']
lm = LinearRegression()
lm.fit(X,Y)
lm.score(X, Y)
```

Out[20]:
```
0.00046769430149007363
```

Question 6

In [21]:
```
lm1 = LinearRegression()
lm1.fit(df[['sqft_living']],df[['price']])
yHat1 = lm1.predict(df[['sqft_living']])
lm1.score(df[['sqft_living']],df[['price']])
```

Out[21]:
```
0.4928532179037931
```

Question 7

In [22]:
```
features =["floors", "waterfront","lat" ,"bedrooms" ,"sqft_basement" ,"view" ,"b
lm2 = LinearRegression()
lm2.fit(df[features],df[['price']])
yHat2 = lm2.predict(df[features])
```

In [24]:
```
lm2.score(df[features],df[['price']])
```

Out[24]:
```
0.6576951666037504
```

```
In [25]: Input=[('scale',StandardScaler()),('polynomial', PolynomialFeatures(include_bias
```

Question 8

```
In [26]: Pipe = Pipeline(Input)
         Pipe.fit(df[features],df[["price"]])
         yHat2=Pipe.predict(df[features])
         Pipe.score(df[features],df[["price"]])
```

```
Out[26]: 0.7513402173516526
```

Question 9

```
In [27]: from sklearn.model_selection import cross_val_score
         from sklearn.model_selection import train_test_split
         print("done")
```

```
done
```

```
In [28]: features =["floors", "waterfront","lat" ,"bedrooms" ,"sqft_basement" ,"view" ,"b
         X = df[features]
         Y = df['price']

         x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.15, random


         print("number of test samples:", x_test.shape[0])
         print("number of training samples:",x_train.shape[0])
```

```
number of test samples: 3242
number of training samples: 18371
```

```
In [29]: from sklearn.linear_model import Ridge
```

```
In [30]: RidgeModel = Ridge(alpha = 1)
         RidgeModel.fit(x_train, y_train)
         RidgeModel.score(x_test, y_test)
```

```
Out[30]: 0.6478078664848204
```

Question 10

```
In [31]: pr=PolynomialFeatures(degree=2)
         x_train_pr=pr.fit_transform(x_train)
         x_test_pr=pr.fit_transform(x_test)
         RidgeModel1 = Ridge(alpha = 0.1)
         RidgeModel.fit(x_train_pr,y_train)
         RidgeModel.score(x_test_pr, y_test)
```

```
Out[31]: 0.6996769630569588
```

```
In [ ]:
```