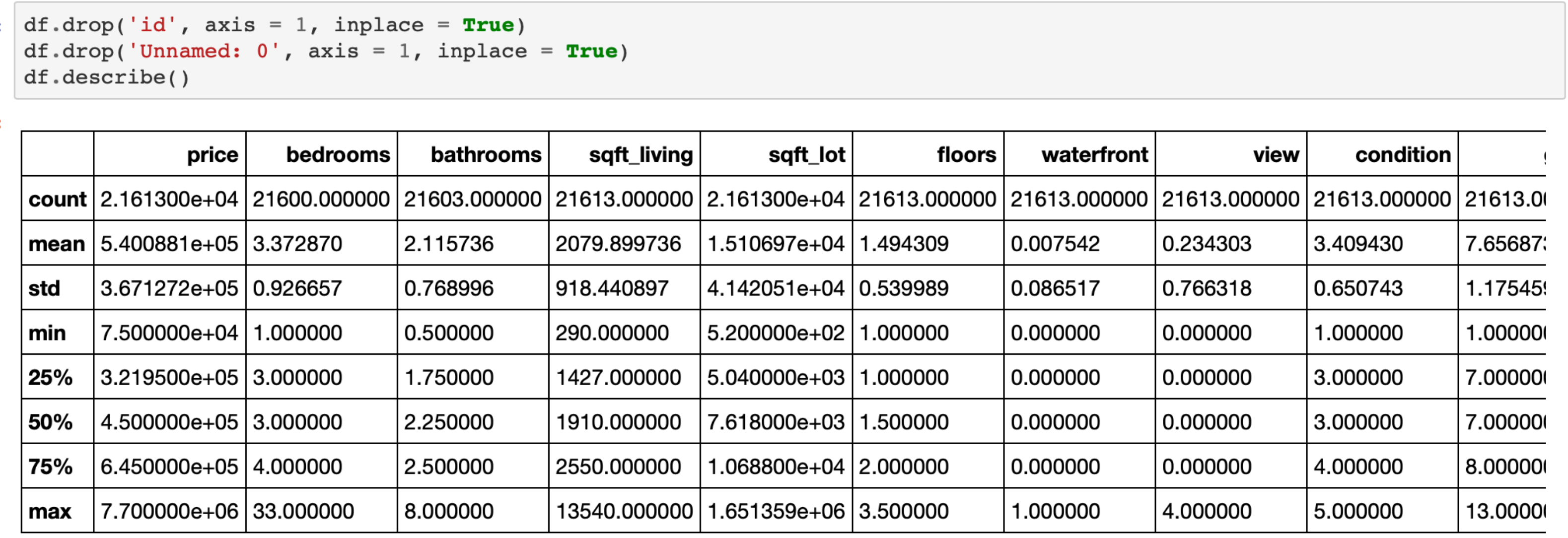
In [9]:
```
print(df.dtypes)
```

```
Unnamed: 0         int64
id                 int64
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
```



```
df.drop('id', axis = 1, inplace = True)
df.drop('Unnamed: 0', axis = 1, inplace = True)
df.describe()
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	
count	2.161300e+04	21600.000000	21603.000000	21613.000000	2.161300e+04	21613.000000	21613.000000	21613.000000	21613.000000	21613.0
mean	5.400881e+05	3.372870	2.115736	2079.899736	1.510697e+04	1.494309	0.007542	0.234303	3.409430	7.65687
std	3.671272e+05	0.926657	0.768996	918.440897	4.142051e+04	0.539989	0.086517	0.766318	0.650743	1.17545
min	7.500000e+04	1.000000	0.500000	290.000000	5.200000e+02	1.000000	0.000000	0.000000	1.000000	1.00000
25%	3.219500e+05	3.000000	1.750000	1427.000000	5.040000e+03	1.000000	0.000000	0.000000	3.000000	7.00000
50%	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.500000	0.000000	0.000000	3.000000	7.00000
75%	6.450000e+05	4.000000	2.500000	2550.000000	1.068800e+04	2.000000	0.000000	0.000000	4.000000	8.00000
max	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	1.000000	4.000000	5.000000	13.0000

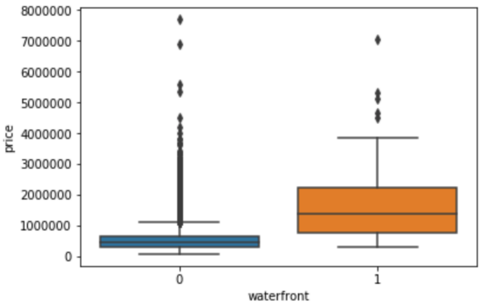
```
In [20]: Unique_floors=df["floors"].value_counts()
         Unique_floors.to_frame()
```

Out[20]:

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

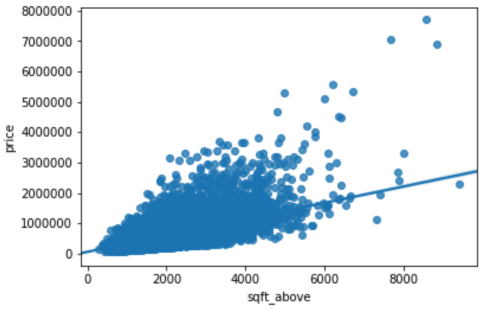
```
In [21]: sns.boxplot(x="waterfront",y="price",data=df)
```

```
Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbc303d1828>
```



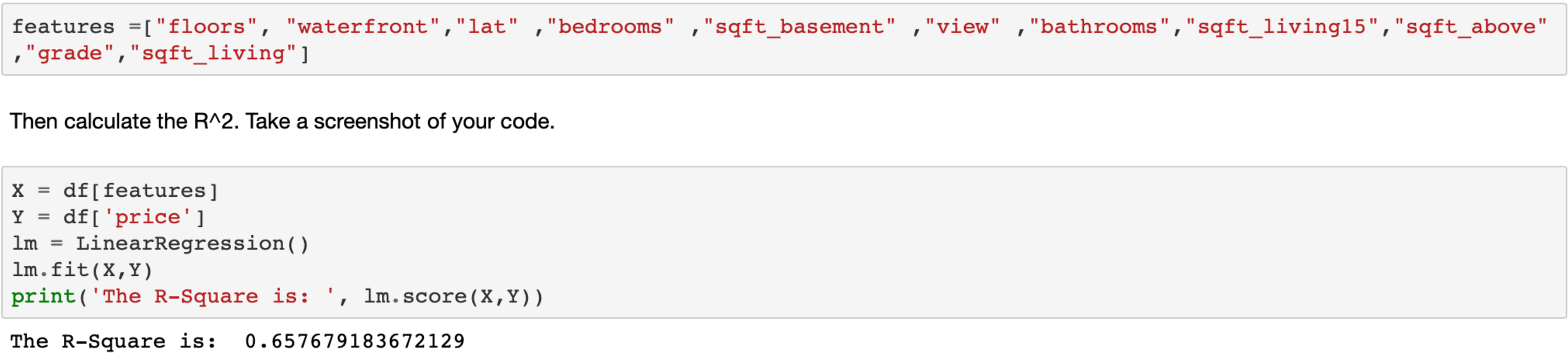
```
In [22]: sns.regplot(x="sqft_above",y="price",data=df)
```

```
Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x7fbc303e76a0>
```



```
In [31]: x=df[['price']]
         y=df[['sqft_living']]
         lm = LinearRegression()
         lm
         lm.fit(x,y)
         lm.score(x,y)

Out[31]: 0.4928532179037931
```



```
features =["floors", "waterfront","lat" ,"bedrooms" ,"sqft_basement" ,"view" ,"bathrooms","sqft_living15","sqft_above"
,"grade","sqft_living"]
```

Then calculate the R^2. Take a screenshot of your code.

```
X = df[features]
Y = df['price']
lm = LinearRegression()
lm.fit(X,Y)
print('The R-Square is: ', lm.score(X,Y))
```

```
The R-Square is:  0.657679183672129
```

```
pipe=Pipeline(Input)
pipe
pipe.fit(X,Y)
Ypipe=pipe.predict(X)
Ypipe[0:4]
```

```
/opt/conda/envs/Python36/lib/python3.6/site-packages/sklearn/preprocessing/data.py:645: DataConversionWarning: Data w
ith input dtype int64, float64 were all converted to float64 by StandardScaler.
  return self.partial_fit(X, y)
/opt/conda/envs/Python36/lib/python3.6/site-packages/sklearn/base.py:467: DataConversionWarning: Data with input dtyp
e int64, float64 were all converted to float64 by StandardScaler.
  return self.fit(X, y, **fit_params).transform(X)
/opt/conda/envs/Python36/lib/python3.6/site-packages/sklearn/pipeline.py:331: DataConversionWarning: Data with input
dtype int64, float64 were all converted to float64 by StandardScaler.
  Xt = transform.transform(Xt)
```

```
array([349649.75, 559166.25, 449506.75, 393246.75])
```

```
from sklearn.linear_model import Ridge
```

```
RidgeModel=Ridge(alpha=0.1)
RidgeModel.fit(x_train, y_train)
RidgeModel.score(x_test, y_test)
```

```
0.6478759163939121
```

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

```
0.7002744279699229
```