House Sales in King County

Will analyse and predict housing prices using attributes or features such as square footage, number of bedrooms, number of floors and so on.

Import libraries

In [14]:

```
import pandas as pd
```

Create a dataframe

In [15]:

```
HS_df = pd.read_csv('kc_house_data.csv')
HS_df.head()
```

Out[15]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floor
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.

5 rows × 21 columns

Display the data types of each column

In [16]:

```
HS_df.dtypes
```

Out[16]:

```
id                 int64
date              object
price            float64
bedrooms           int64
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
```

Drop the columns "id" and "date"

In [17]:

```
HS_df.drop(['id', 'date'], axis = 1, inplace = True)
HS_df.describe()
```

Out[17]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors
count	2.161300e+04	21613.000000	21613.000000	21613.000000	2.161300e+04	21613.000000
mean	5.400881e+05	3.370842	2.114757	2079.899736	1.510697e+04	1.494309
std	3.671272e+05	0.930062	0.770163	918.440897	4.142051e+04	0.539989
min	7.500000e+04	0.000000	0.000000	290.000000	5.200000e+02	1.000000
25%	3.219500e+05	3.000000	1.750000	1427.000000	5.040000e+03	1.000000
50%	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.500000
75%	6.450000e+05	4.000000	2.500000	2550.000000	1.068800e+04	2.000000
max	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000

Houses with unique floor values

In [18]:

```
HS_df['floors'].value_counts().to_frame()
```

Out[18]:

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

Produce a plot that can be used to determine whether houses with a waterfront view or without a waterfront view have more price outliers

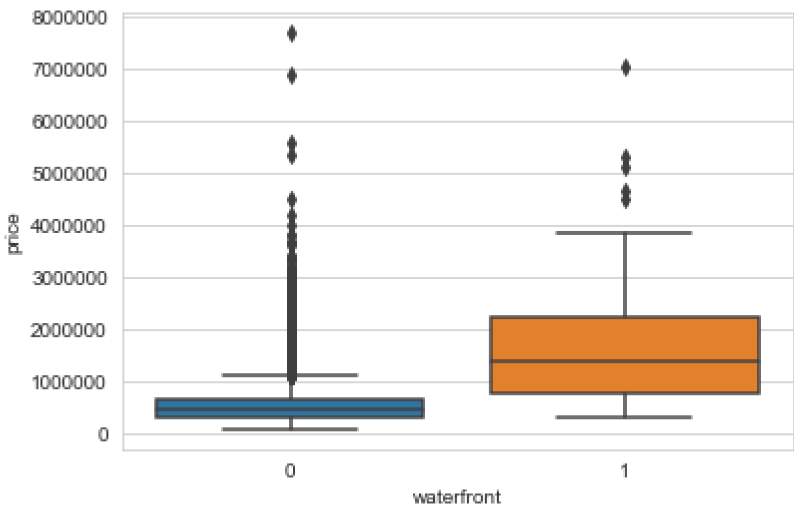
In [25]:

```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline


sns.set_style("whitegrid")
sns.boxplot(x = 'waterfront', y = 'price', data = HS_df)
```

Out[25]:

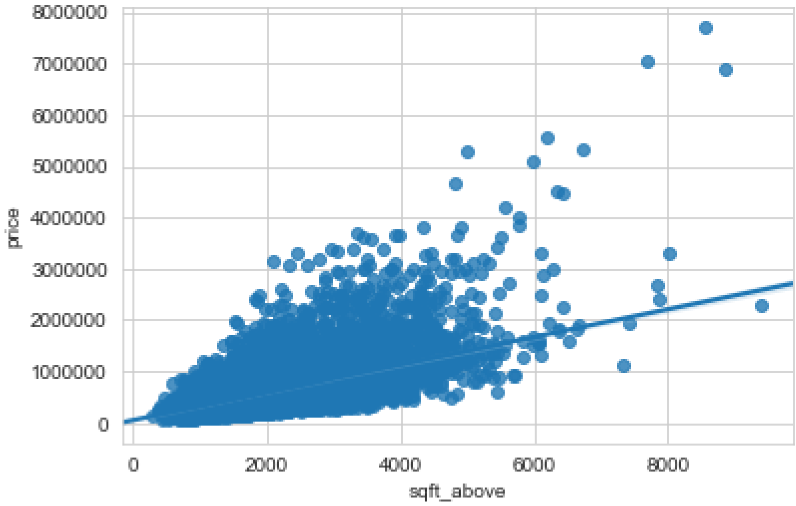
```
<matplotlib.axes._subplots.AxesSubplot at 0x22dad0df128>
```



Determine if the feature sqft_above is negatively or positively correlated with price.

In [28]:

```
sns.regplot(x="sqft_above", y="price", data=HS_df);
```



Fit a linear regression model to predict the price using the feature 'sqft_living' then calculate the **R^2.**

In [29]:

```
import numpy as np
from sklearn.linear_model import LinearRegression
```

In [31]:

```
lm = LinearRegression()
X = HS_df[['sqft_living']]
Y = HS_df['price']
```

In [32]:

```
lm.fit(X,Y)
```

Out[32]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [34]:

```
r_sq = lm.score(X,Y)
print('coefficient of determination:', r_sq)
```

```
coefficient of determination: 0.49285321790379316
```

Fit a linear regression model to predict the 'price' using the list of features: then calculate the **R^2.**

"floors" "waterfront" "lat" "bedrooms" "sqft_basement" "view" "bathrooms" "sqft_living15" "sqft_above" "grade" "sqft_living"

In [38]:

```
Z = HS_df[['floors', 'waterfront', 'lat','bedrooms','sqft_basement','view','bathrooms',
'sqft_living15','sqft_above','grade','sqft_living']]
```

In [39]:

```
lm.fit(Z, HS_df['price'])
```

Out[39]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [40]:

```
r_sq = lm.score(Z, HS_df['price'])
print('coefficient of determination:', r_sq)
```

```
coefficient of determination: 0.6577147048948742
```

Create a **pipeline object** that **scales the data performs a polynomial transform and fits a linear regression model. Fit the object** using the above features above, then fit the model and calculate the R^2.

In [50]:

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
```

In [57]:

```
Input= [('scale',StandardScaler()),('polynomial',PolynomialFeatures(degree=2)),('model'
,LinearRegression())]
```

In [58]:

```
Pipe = Pipeline(Input)
```

In [60]:

```
Pipe.fit(Z, HS_df['price'])
```

```
C:\Users\aksha\Anaconda3\lib\site-packages\sklearn\preprocessing\data.py:6
45: DataConversionWarning: Data with input dtype int64, float64 were all c
onverted to float64 by StandardScaler.
  return self.partial_fit(X, y)
C:\Users\aksha\Anaconda3\lib\site-packages\sklearn\base.py:467: DataConver
sionWarning: Data with input dtype int64, float64 were all converted to fl
oat64 by StandardScaler.
  return self.fit(X, y, **fit_params).transform(X)
```

Out[60]:

```
Pipeline(memory=None,
     steps=[('scale', StandardScaler(copy=True, with_mean=True, with_std=T
rue)), ('polynomial', PolynomialFeatures(degree=2, include_bias=True, inte
raction_only=False)), ('model', LinearRegression(copy_X=True, fit_intercep
t=True, n_jobs=None,
         normalize=False))])
```

In [62]:

```
r_sq = Pipe.score(Z, HS_df['price'])
print('coefficient of determination:', r_sq)
```

```
coefficient of determination: 0.7491783035196988

C:\Users\aksha\Anaconda3\lib\site-packages\sklearn\pipeline.py:511: DataCo
nversionWarning: Data with input dtype int64, float64 were all converted t
o float64 by StandardScaler.
  Xt = transform.transform(Xt)
```

Create and fit a **Ridge regression** object using the training data, setting the regularization parameter to 0.1 and calculate the R^2 using the test data.

In [75]:

```
from sklearn.linear_model import Ridge
rm=Ridge(alpha=0.1)
```

In [76]:

```
rm.fit(Z,Y)
r_sq = rm.score(Z,Y)
print('coefficient of determination:', r_sq)
```

```
coefficient of determination: 0.6577150926380133
```

Perform a **second order polynomial transform** on both the training data and testing data. Create and fit a Ridge regression object using the training data, setting the regularisation parameter to 0.1. Calculate the R^2 utilising the test data provided

In [77]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split( Z, Y, test_size=0.3, random_state=
0)
print ('Train set:', X_train.shape,  y_train.shape)
print ('Test set:', X_test.shape,  y_test.shape)
```

```
Train set: (15129, 11) (15129,)
Test set: (6484, 11) (6484,)
```

In [78]:

```
pr=PolynomialFeatures(degree=2)
X_train_pr=pr.fit_transform(X_train)
```

In [79]:

```
from sklearn.linear_model import Ridge
rm=Ridge(alpha=0.1)
```

In [80]:

```
rm.fit(X_train_pr, y_train)
r_sq = rm.score(X_train_pr, y_train)
print('coefficient of determination:', r_sq)
```

```
coefficient of determination: 0.7401029209453733
```