King County House Price Predictions.

Background.

King County is one of three Washington counties that are included in the Seattle–Tacoma–Bellevue metropolitan statistical area. It covers an area of $5980km^2$ with a total of 39 towns and cities. According to wikipedia, the population as at 2020 was 2,269,675.

Business Understanding

There are many households who would like to purchase houses around King County, but due to the information asymmetry in the market they go into it blindly. Therefore to mitigate that gap in the market, we are going to study some data on the sale of houses that took place between the year 2014 to 2015, within King County.

Our project aims at providing consultation to a real estate agency that helps households purchase houses. And through studying the data we will provide a way in which one can predict the prices of the houses.

Data Understanding.

In [177]:

```
# importing libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import statsmodels.api as sm
from scipy import stats
from statsmodels.formula.api import ols
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_absolute_error, mean_squared_error,r2_score
from sklearn.preprocessing import PolynomialFeatures, StandardScaler, OneHotEncoder
```

In [80]:

```
# Loading the data
data = pd.read_csv("data/kc_house_data.csv")
data
```

Out[80]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	w
0	7129300520	10/13/2014	221900.0	3	1.00	1180	5650	1.0	
1	6414100192	12/9/2014	538000.0	3	2.25	2570	7242	2.0	
2	5631500400	2/25/2015	180000.0	2	1.00	770	10000	1.0	
3	2487200875	12/9/2014	604000.0	4	3.00	1960	5000	1.0	
4	1954400510	2/18/2015	510000.0	3	2.00	1680	8080	1.0	
...	...	...	...	...	...	...	...	...	
21592	263000018	5/21/2014	360000.0	3	2.50	1530	1131	3.0	
21593	6600060120	2/23/2015	400000.0	4	2.50	2310	5813	2.0	
21594	1523300141	6/23/2014	402101.0	2	0.75	1020	1350	2.0	
21595	291310100	1/16/2015	400000.0	3	2.50	1600	2388	2.0	
21596	1523300157	10/15/2014	325000.0	2	0.75	1020	1076	2.0	

21597 rows × 21 columns

In [81]:

```
data.tail()
```

Out[81]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	w
21592	263000018	5/21/2014	360000.0	3	2.50	1530	1131	3.0	
21593	6600060120	2/23/2015	400000.0	4	2.50	2310	5813	2.0	
21594	1523300141	6/23/2014	402101.0	2	0.75	1020	1350	2.0	
21595	291310100	1/16/2015	400000.0	3	2.50	1600	2388	2.0	
21596	1523300157	10/15/2014	325000.0	2	0.75	1020	1076	2.0	

5 rows × 21 columns

In [82]:

```python
# Check on the information of the daraframe
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             21597 non-null  int64  
 1   date           21597 non-null  object 
 2   price          21597 non-null  float64
 3   bedrooms       21597 non-null  int64  
 4   bathrooms      21597 non-null  float64
 5   sqft_living    21597 non-null  int64  
 6   sqft_lot       21597 non-null  int64  
 7   floors         21597 non-null  float64
 8   waterfront     19221 non-null  object 
 9   view           21534 non-null  object 
 10  condition      21597 non-null  object 
 11  grade          21597 non-null  object 
 12  sqft_above     21597 non-null  int64  
 13  sqft_basement  21597 non-null  object 
 14  yr_built       21597 non-null  int64  
 15  yr_renovated   17755 non-null  float64
 16  zipcode        21597 non-null  int64  
 17  lat            21597 non-null  float64
 18  long           21597 non-null  float64
 19  sqft_living15  21597 non-null  int64  
 20  sqft_lot15     21597 non-null  int64  
dtypes: float64(6), int64(9), object(6)
memory usage: 3.5+ MB
```

In [83]:

```python
# Obtain a statistical summary of the dataframe
data.describe()
```

Out[83]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot
count	2.159700e+04	2.159700e+04	21597.000000	21597.000000	21597.000000	2.159700e+04
mean	4.580474e+09	5.402966e+05	3.373200	2.115826	2080.321850	1.509941e+04
std	2.876736e+09	3.673681e+05	0.926299	0.768984	918.106125	4.141264e+04
min	1.000102e+06	7.800000e+04	1.000000	0.500000	370.000000	5.200000e+02
25%	2.123049e+09	3.220000e+05	3.000000	1.750000	1430.000000	5.040000e+03
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068500e+04
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06

In [84]:

```
# shape of the dataframe
data.shape
```

Out[84]:

```
(21597, 21)
```

In [85]:

```
# check for null values
data.isnull().sum()
```

Out[85]:

```
id                  0
date                0
price               0
bedrooms            0
bathrooms           0
sqft_living         0
sqft_lot            0
floors              0
waterfront       2376
view               63
condition           0
grade               0
sqft_above          0
sqft_basement       0
yr_built            0
yr_renovated     3842
zipcode             0
lat                 0
long                0
sqft_living15       0
sqft_lot15          0
dtype: int64
```

Data Preparation.

Data preparation is the process of cleaning and transforming raw data prior to processing and analysis.

In [86]:

```
# drop the rows with null values
df = data.dropna(axis=0, how='any')
df.isnull().sum()
```

Out[86]:

```
id               0
date             0
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
grade            0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
zipcode          0
lat              0
long             0
sqft_living15    0
sqft_lot15       0
dtype: int64
```

Drop columnns that will not be useful in our analysis.

In [87]:

```
# drop the id column
df = df.drop(columns = ["id",'lat','long','date','sqft_living15', 'sqft_lot15'])
df
```

Out[87]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	conditi
1	538000.0	3	2.25	2570	7242	2.0	NO	NONE	Avera
3	604000.0	4	3.00	1960	5000	1.0	NO	NONE	V Go
4	510000.0	3	2.00	1680	8080	1.0	NO	NONE	Avera
5	1230000.0	4	4.50	5420	101930	1.0	NO	NONE	Avera
6	257500.0	3	2.25	1715	6819	2.0	NO	NONE	Avera
...	...	...	...	...	...	...	...	...	
21591	475000.0	3	2.50	1310	1294	2.0	NO	NONE	Avera
21592	360000.0	3	2.50	1530	1131	3.0	NO	NONE	Avera
21593	400000.0	4	2.50	2310	5813	2.0	NO	NONE	Avera
21594	402101.0	2	0.75	1020	1350	2.0	NO	NONE	Avera
21596	325000.0	2	0.75	1020	1076	2.0	NO	NONE	Avera

15762 rows × 15 columns

In [88]:

```
# drop the rows in sqft_basement with a '?'
df = df.drop(df[df.sqft_basement == '?'].index)
df
```

Out[88]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	conditi
1	538000.0	3	2.25	2570	7242	2.0	NO	NONE	Avera
3	604000.0	4	3.00	1960	5000	1.0	NO	NONE	V Go
4	510000.0	3	2.00	1680	8080	1.0	NO	NONE	Avera
5	1230000.0	4	4.50	5420	101930	1.0	NO	NONE	Avera
8	229500.0	3	1.00	1780	7470	1.0	NO	NONE	Avera
...	...	...	...	...	...	...	...	...	
21591	475000.0	3	2.50	1310	1294	2.0	NO	NONE	Avera
21592	360000.0	3	2.50	1530	1131	3.0	NO	NONE	Avera
21593	400000.0	4	2.50	2310	5813	2.0	NO	NONE	Avera
21594	402101.0	2	0.75	1020	1350	2.0	NO	NONE	Avera
21596	325000.0	2	0.75	1020	1076	2.0	NO	NONE	Avera

15429 rows × 15 columns

In [89]:

```
# Check if there are any duplicates.
df.drop_duplicates(subset = ['sqft_above'], keep = 'first', inplace = True)
```

In [90]:

```
# Confirm that all the duplicates have been dropped.
df.duplicated().sum()
```

Out[90]:

0

Replace the strings with integers depending on the intensities of the strings.

In [91]:

```
df['view1'] = df['view'].replace({'NONE': 0,'FAIR':1,'AVERAGE': 2,'GOOD':3, 'EXCELLENT':4})
```

In [92]:

```
df['waterfront1'] = df['waterfront'].replace({'YES': 0, 'NO':1})
```

In [93]:

```
df['condition1'] = df['condition'].replace({'Poor': 0, 'Fair':1,'Average':2,'Good':3,'Very
```

Split the grade column to a new column which only has the grade value in numbers. This will now make it easier when carrying out statistical measurements.

In [94]:

```
df["Grade1"] = df["grade"].str.split().apply(lambda x: x[0])
# Convert the Grade1 column to an integer.
df["Grade1"] = pd.to_numeric(df["Grade1"])
```

Drop the colums that will not be quite useful.

In [96]:

```
df = df.drop(columns = ['waterfront', 'view', 'grade'])
df[:10]
```

Data Modeling

We'll conduct a test to check on the level of correlation of the features with the 'price'

In [70]:

```
df.corr()['price'].sort_values(ascending = False)
```

Out[70]:

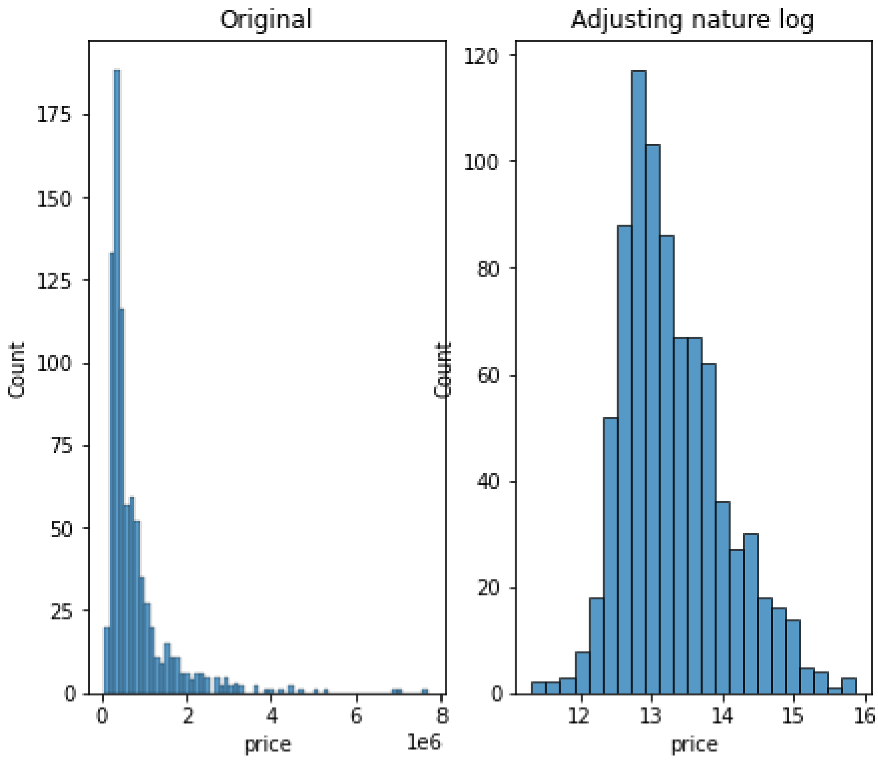
```
price           1.000000
sqft_living     0.792749
sqft_above      0.724267
Grade1          0.682691
bathrooms       0.671431
view1           0.494763
bedrooms        0.416373
floors          0.217050
yr_renovated    0.168015
sqft_lot        0.127459
condition1      0.053972
zipcode        -0.048096
yr_built       -0.048597
waterfront1    -0.331173
Name: price, dtype: float64
```

We find out that sqft_living,sqft_above,Grade1 and bathrooms have the highest correlation with price.

In [151]:

```
fig, ax = plt.subplots(1, 2,figsize=(7,6))

sns.histplot(df['price'], ax=ax[0])
ax[0].set_title('Original')
sns.histplot(np.log(df['price']), ax=ax[1])
ax[1].set_title('Adjusting log')
plt.show()
```



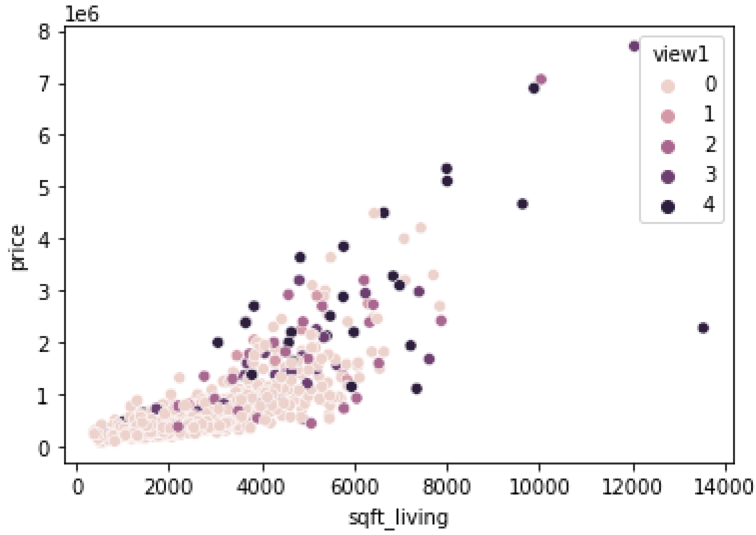
The original price is skewed to the left, and after the adjustment we can see how the normal distribution is to the price.

Correlation between sqft_living and views to price.

We will evaluate a liner regression model using sqft_living and views.

In [154]:

```
sns.scatterplot(data =df,x = 'sqft_living',y= 'price',hue ='view1');
```



The graph shows us that the sqft_living is positively correlated to price and it also helps us to conclude that the better the view, the more expensive it is.

In [181]:

```
a = df[['sqft_living','view1']]
b= df[['price']]
model = sm.OLS(b, a)

result = model.fit()
result.summary()
```

Out[181]:

OLS Regression Results

Dep. Variable:	price	**R-squared (uncentered):**	0.812
Model:	OLS	**Adj. R-squared (uncentered):**	0.812
Method:	Least Squares	**F-statistic:**	1790.
Date:	Fri, 30 Sep 2022	**Prob (F-statistic):**	3.46e-301
Time:	11:13:31	**Log-Likelihood:**	-12038.
No. Observations:	829	**AIC:**	2.408e+04
Df Residuals:	827	**BIC:**	2.409e+04
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
sqft_living	269.2437	5.856	45.974	0.000	257.749	280.739
view1	2.009e+05	1.81e+04	11.114	0.000	1.65e+05	2.36e+05

Omnibus:	593.820	**Durbin-Watson:**	1.912
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	13141.967
Skew:	2.969	**Prob(JB):**	0.00
Kurtosis:	21.580	**Cond. No.**	3.53e+03

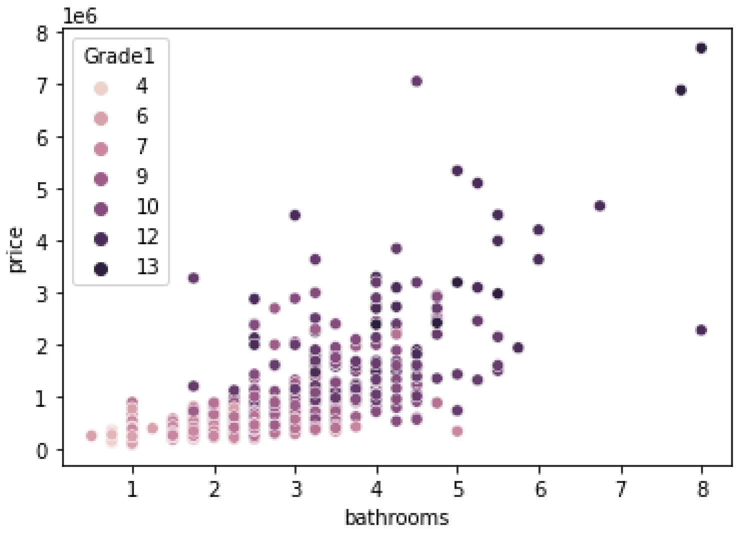
Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[3] The condition number is large, 3.53e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

The P-value is less than 0.05, so we reject the null hypothisis. That means, the model is statistically significant.

Correlation between bathrooms and grade to price.

In [179]:

```
sns.scatterplot(data =df,x = 'bathrooms',y= 'price',hue ='Grade1');
```



The graph shows us that the number of bathrooms is positively correlated to price and it also helps us to conclude that the better the grade of a house, the more expensive it is.

In [187]:

```
X = df[['sqft_living','sqft_above','bathrooms','bedrooms','Grade1','view1']]
y = df['price']
```

In [188]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

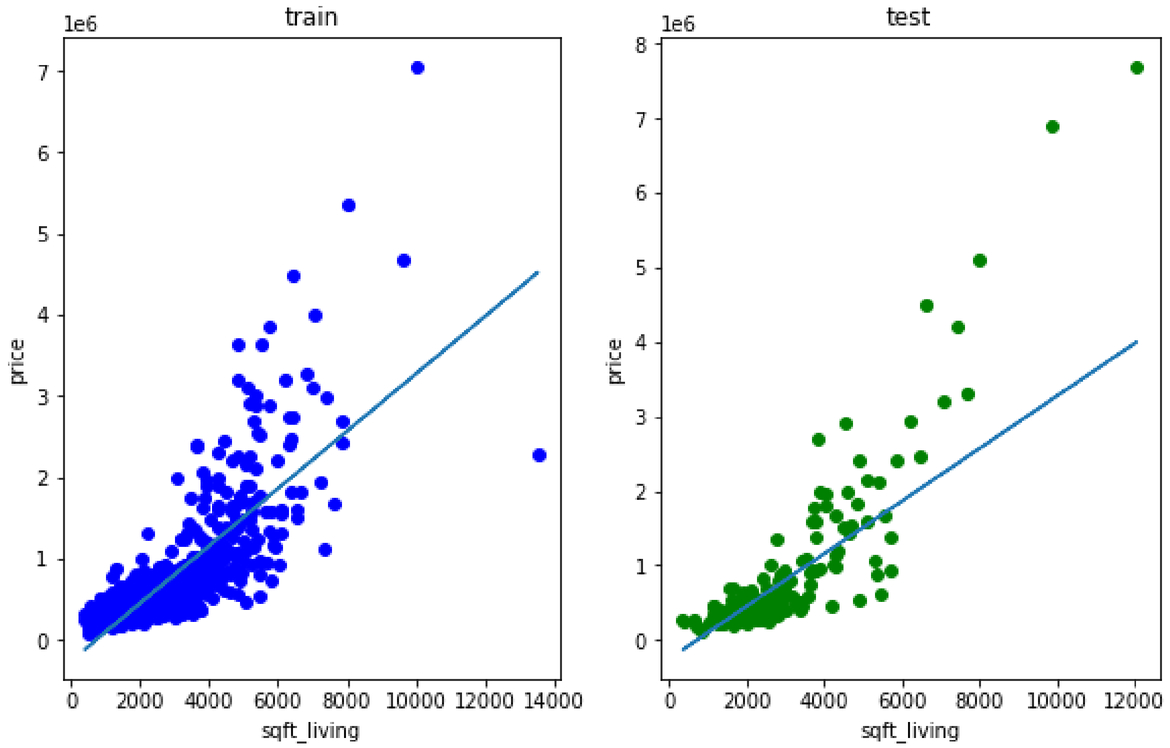
In [189]:

```
sqft = LinearRegression()
sqft.fit(X_train[['sqft_living']], y_train)
sqft.score(X_train[['sqft_living']], y_train)
y_hat_train = sqft.predict(X_train[['sqft_living']])
y_hat_test = sqft.predict(X_test[['sqft_living']])
```

In [190]:

```
plt.figure(figsize=(10,6))
plt.subplot(1,2,1)
plt.scatter(X_train[['sqft_living']], y_train, color = "blue")
plt.plot(X_train[['sqft_living']] ,y_hat_train)
plt.xlabel('sqft_living')
plt.ylabel('price')
plt.title('train')

plt.subplot(1,2,2)
plt.scatter(X_test[['sqft_living']], y_test, color = "green")
plt.plot(X_test[['sqft_living']] ,y_hat_test)
plt.xlabel('sqft_living')
plt.ylabel('price')
plt.title('test');
```

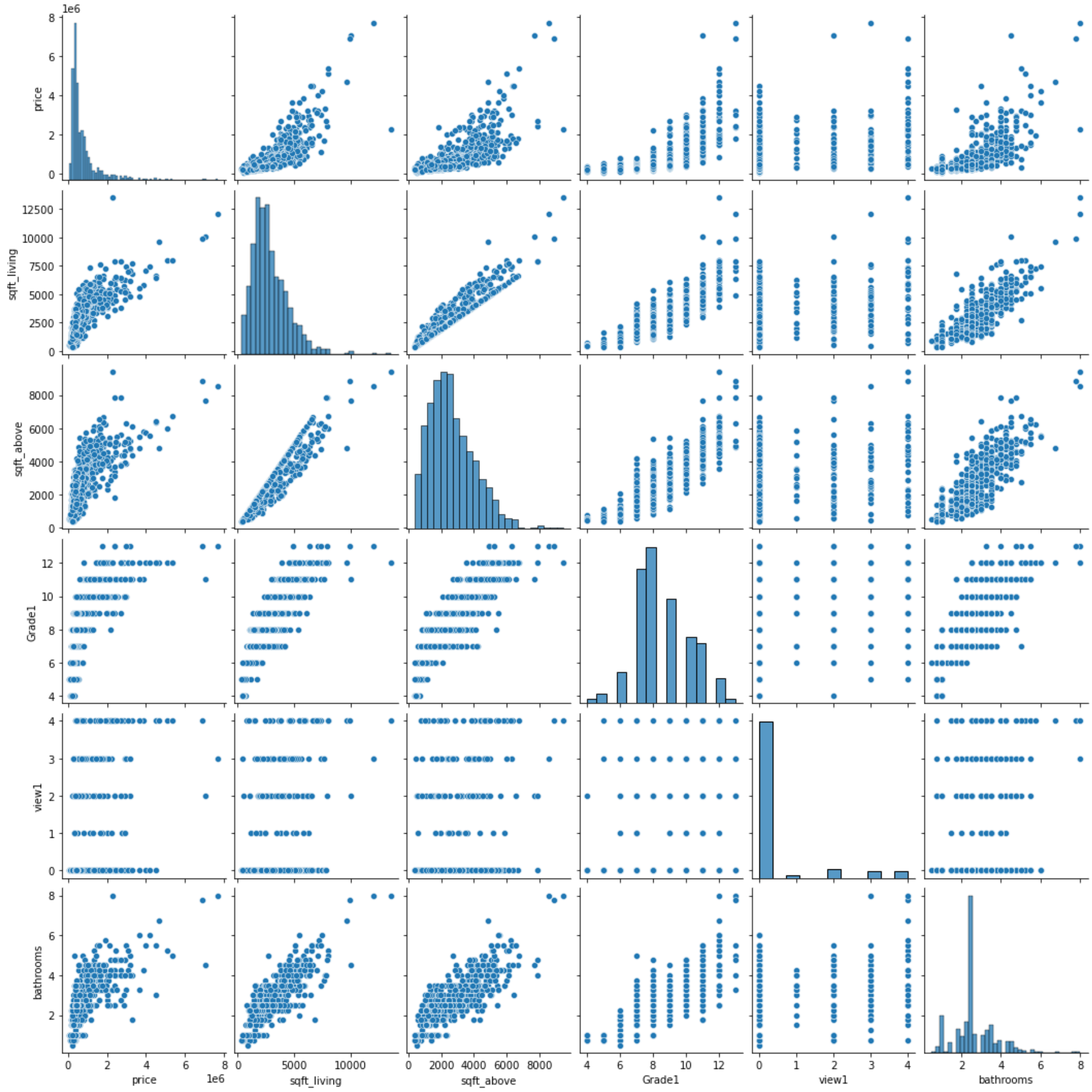


The train and the test sets both have a linear relationship to price.

Check the correlation of the other features and price.

In [191]:

```
df_pairplot = df[['price','sqft_living','sqft_above','Grade1', 'view1','bathrooms']]
sns.pairplot(df_pairplot)
plt.show()
```



We can conclude the the features which have the highest correlation with price are sqft_living, sqft_above, grade1, and bathrooms.

Multiple Linear Regression.

In [192]:

```
reg = sm.add_constant(X, has_constant='add')
model = sm.OLS(, X)
result1 = model.fit()
result1.summary()
```

Out[192]:

OLS Regression Results

Dep. Variable:	price	**R-squared (uncentered):**	0.833
Model:	OLS	**Adj. R-squared (uncentered):**	0.832
Method:	Least Squares	**F-statistic:**	683.8
Date:	Fri, 30 Sep 2022	**Prob (F-statistic):**	9.89e-316
Time:	12:17:04	**Log-Likelihood:**	-11990.
No. Observations:	829	**AIC:**	2.399e+04
Df Residuals:	823	**BIC:**	2.402e+04
Df Model:	6		
Covariance Type:	nonrobust		

The p-value is less than 0.05, so we reject the null hypothesus. That means that the model is statistically significant. The R-Squared is 0.833, meaning almost 83% can be explained by the model.

In [197]:

```
h = np.log(df['price'])
reg = sm.add_constant(X, has_constant='add')
model = sm.OLS(h, X)
result2 = model.fit()
result2.summary()
```

Out[197]:

OLS Regression Results

Dep. Variable:	price	**R-squared (uncentered):**	0.987
Model:	OLS	**Adj. R-squared (uncentered):**	0.987
Method:	Least Squares	**F-statistic:**	1.064e+04
Date:	Fri, 30 Sep 2022	**Prob (F-statistic):**	0.00
Time:	12:21:27	**Log-Likelihood:**	-1512.6
No. Observations:	829	**AIC:**	3037.
Df Residuals:	823	**BIC:**	3066.
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
sqft_living	-0.0004	0.000	-3.717	0.000	-0.001	-0.000
sqft_above	-0.0010	0.000	-7.975	0.000	-0.001	-0.001
bathrooms	0.0626	0.098	0.641	0.521	-0.129	0.254
bedrooms	0.7466	0.065	11.486	0.000	0.619	0.874
Grade1	1.6434	0.033	50.348	0.000	1.579	1.707
view1	0.1021	0.060	1.706	0.088	-0.015	0.220

Omnibus:	53.032	**Durbin-Watson:**	1.894
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	70.507
Skew:	0.550	**Prob(JB):**	4.89e-16
Kurtosis:	3.912	**Cond. No.**	8.58e+03

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[3] The condition number is large, 8.58e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

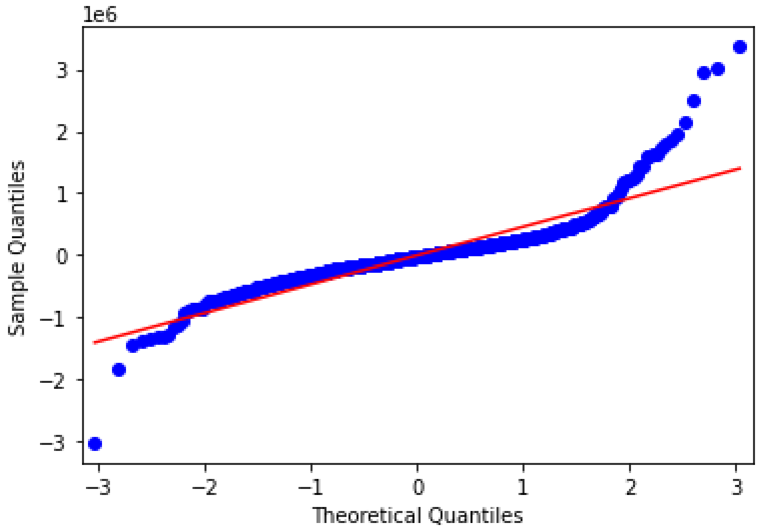
When the data is normally distributed, the model explains 98.7% of the price.

Evaluation.

Check normality and residual pattern.

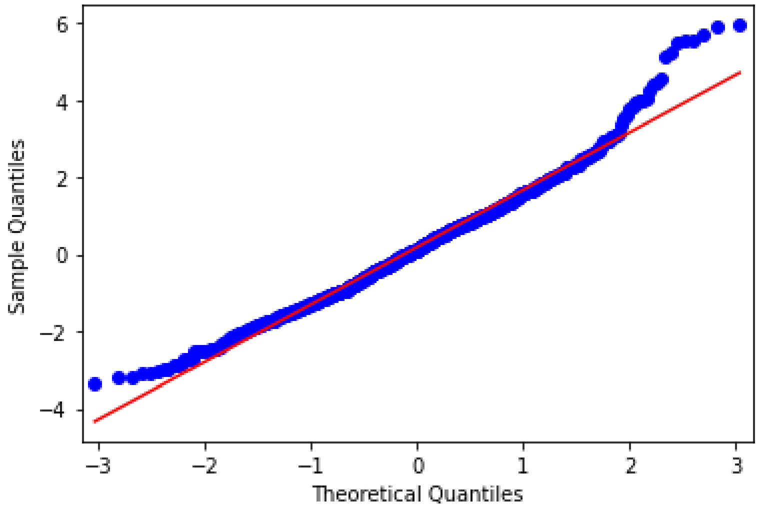
In [195]:

```
qqplot = sm.qqplot(result1.resid,line ='s',dist=stats.norm)
```



In [198]:

```
qqplot = sm.qqplot(result2.resid,line ='s',dist=stats.norm)
```



In [ ]:

```
# Check residual pattern
fitted = result1.predict()

resid = result1.resid
pred = result1.predict(X)
fig = plt.scatter(pred, resid, s=3)

plt.xlabel('Fitted values')
plt.ylabel('Residual')

plt.show()
```

The data satisfies normality.

In [ ]:

```
result1.pvalues
```

The p-values are less than 0.05, so we reject the null hypothesis. Therefore we can say that the model is statistically significant.

Summary on evaluation.

The models that we have constructed has given us a more in depth understanding on the association of the various house features and the prices.The models explain around 83.3% of the sales prices. Most of the households that would love to purchase houses in King County can now have a general idea of the criteria of house pricing.

Conclusion.

The biggest contributors to pricing in a house is the square footage of the living space, square footage of the house excluding the basement and the grade of the house.