Final Project Submission

Please fill out:

- Student name: Lauren Petrillo
- Student pace: Flex
- Scheduled project review date/time:
- Instructor name:
- Blog post URL:

Introduction

Objective

I was hired by West Seattle Realty (a local real estate agency) to provide insights to better their speciality in helping buyers and sellers navigate the King County residential home market.

Link to local agency: https://westseattlerealty.com/west-seattle-realty

Questions to Consider for Modeling

1. What features in a house are linearly related to price?
2. What are the most important factors when determining price of a house?
3. Does location (zipcode/lat/long) have an effect on price?

Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
from numpy.random import randn
from pandas import DataFrame, Series
import warnings
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import plotly.express as px
from scipy import stats
import scipy.stats as stats
import statsmodels.api as sm
import statsmodels.formula.api as smf
from statsmodels.formula.api import ols
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, make_scorer
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import MinMaxScaler
```

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_validate
from sklearn.preprocessing import OneHotEncoder, StandardScaler
warnings.filterwarnings("ignore")
%matplotlib inline
pd.set_option('display.max_columns', 0)
plt.style.use('seaborn')
```

Load Data

In [2]:
```
df = pd.read_csv('data/kc_house_data.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	v
0	7129300520	10/13/2014	221900.0	3	1.00	1180	5650	1.0	NaN	N
1	6414100192	12/9/2014	538000.0	3	2.25	2570	7242	2.0	NO	N
2	5631500400	2/25/2015	180000.0	2	1.00	770	10000	1.0	NO	N
3	2487200875	12/9/2014	604000.0	4	3.00	1960	5000	1.0	NO	N
4	1954400510	2/18/2015	510000.0	3	2.00	1680	8080	1.0	NO	N

King County Data Set Info

Column Descriptions

id - unique identified for a house **dateDate** - house was sold pricePrice - is prediction target bedroomsNumber - of Bedrooms/House bathroomsNumber - of bathrooms/bedrooms sqft_livingsquare - footage of the home sqft_lotsquare - footage of the lot floorsTotal - floors (levels) in house waterfront - House which has a view to a waterfront view - Has been viewed condition - How good the condition is (Overall) grade - overall grade given to the housing unit, based on King County grading system sqft_above - square footage of house apart from basement sqft_basement - square footage of the basement yr_built - Built Year yr_renovated - Year when house was renovated zipcode - zip lat - Latitude coordinate long - Longitude coordinate sqft_living15 - The square footage of interior housing living space for the nearest 15 neighbors sqft_lot15 - The square footage of the land lots of the nearest 15 neighbors

Building Condition

Relative to age and grade. Coded 1-5.

1 = Poor- Worn out. Repair and overhaul needed on painted surfaces, roofing, plumbing, heating and numerous functional inadequacies. Excessive deferred maintenance and abuse, limited value-in-use, approaching abandonment or major reconstruction; reuse or change in occupancy is imminent. Effective age is near the end of the scale regardless of the actual chronological age.

2 = Fair- Badly worn. Much repair needed. Many items need refinishing or overhauling, deferred maintenance obvious, inadequate building utility and systems all shortening the life expectancy and increasing the effective age.

3 = Average- Some evidence of deferred maintenance and normal obsolescence with age in that a few minor repairs are needed, along with some refinishing. All major components still functional and contributing toward an extended life expectancy. Effective age and utility is standard for like properties of its class and usage.

4 = Good- No obvious maintenance required but neither is everything new. Appearance and utility are above the standard and the overall effective age will be lower than the typical property.

5= Very Good- All items well maintained, many having been overhauled and repaired as they have shown signs of wear, increasing the life expectancy and lowering the effective age with little deterioration or obsolescence evident with a high degree of utility.

Building Grade

Represents the construction quality of improvements. Grades run from grade 1 to 13. Generally defined as:

1-3 = Falls short of minimum building standards. Normally cabin or inferior structure.

4 = Generally older, low quality construction. Does not meet code.

5 = Low construction costs and workmanship. Small, simple design.

6 = Lowest grade currently meeting building code. Low quality materials and simple designs.

7 = Average grade of construction and design. Commonly seen in plats and older sub-divisions.

8 = Just above average in construction and design. Usually better materials in both the exterior and interior finish work.

9 = Better architectural design with extra interior and exterior design and quality.

10 = Homes of this quality generally have high quality features. Finish work is better and more design quality is seen in the floor plans. Generally have a larger square footage.

11 = Custom design and higher quality finish work with added amenities of solid woods, bathroom fixtures and more luxurious options.

12 = Custom design and excellent builders. All materials are of the highest quality and all conveniences are present.

13 = Generally custom designed and built. Mansion level. Large amount of highest quality cabinet work, wood trim, marble, entry ways etc.

Received information on building grade and condition using this source: https://info.kingcounty.gov/assessor/esales/Glossary.aspx?type=r

More information can be found regarding the demographics/background info of King County here: https://kingcounty.gov/depts/executive/performance-strategy-budget/regional-planning/Demographics.aspx

Data Engineering

Data Cleaning/Exploration

In [4]:
```
df.info()
#need to change sqft_basement and date from objects to ints
#waterfront, view, and yr_renovated have missing values
#look through data to see if any sets need to be dropped
```

```
<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 [5]:
```
#detect any missing values
df.isna().sum()
```

Out[5]:
```
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
```

In [6]:
```
#replace missing values for waterfront, yr_renovated, and view

fill_waterfront = ['waterfront']
fill_yr_renovated = ['yr_renovated']
fill_view = ['view']

for replace in fill_waterfront:
    missing = df[replace].mode()[0]
    df[replace].fillna(missing, inplace = True)

for replace in fill_yr_renovated:
    missing = df[replace].mode()[0]
    df[replace].fillna(missing, inplace = True)

for replace in fill_view:
    missing = df[replace].mode()[0]
    df[replace].fillna(missing, inplace = True)
```

In [7]:
```
df.isna().sum()
```

Out[7]:
```
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
```

In [8]:
```
#split up the month, day, and year from 'date'
df[['month', 'day', 'year']] = df['date'].str.split('/', expand=True)
```

In [9]:
```
df['month']
```

Out[9]:
```
0        10
1        12
2         2
3        12
4         2
         ..
21592     5
21593     2
21594     6
21595     1
21596    10
Name: month, Length: 21597, dtype: object
```

In [10]:
```
#drop the date, day, and year. Only keeping the month
df = df.drop(['date','day', 'year'], axis=1)
```

In [11]:
```
#create new feature for month sold
df['month_sold'] = df['month'].astype(str).astype(int)
```

In [12]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 22 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             21597 non-null  int64  
 1   price          21597 non-null  float64
 2   bedrooms       21597 non-null  int64  
 3   bathrooms      21597 non-null  float64
 4   sqft_living    21597 non-null  int64  
 5   sqft_lot       21597 non-null  int64  
 6   floors         21597 non-null  float64
 7   waterfront     21597 non-null  object 
 8   view           21597 non-null  object 
 9   condition      21597 non-null  object 
 10  grade          21597 non-null  object 
 11  sqft_above     21597 non-null  int64  
 12  sqft_basement  21597 non-null  object 
 13  yr_built       21597 non-null  int64  
 14  yr_renovated   21597 non-null  float64
 15  zipcode        21597 non-null  int64  
 16  lat            21597 non-null  float64
 17  long           21597 non-null  float64
 18  sqft_living15  21597 non-null  int64  
 19  sqft_lot15     21597 non-null  int64  
 20  month          21597 non-null  object 
 21  month_sold     21597 non-null  int32  
dtypes: float64(6), int32(1), int64(9), object(6)
memory usage: 3.5+ MB
```

In [13]:
```
#first fill the '?' value with '0.0' and then convert sqft_basement from a string to a 
df['sqft_basement'] = df['sqft_basement'].replace(to_replace = '?', value = 0.0)
df['sqft_basement'] = df['sqft_basement'].astype(str).astype(float)
```

In [14]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 22 columns):
 #   Column         Non-Null Count  Dtype  
```

```
---  ------         --------------  -----
 0   id             21597 non-null  int64
 1   price          21597 non-null  float64
 2   bedrooms       21597 non-null  int64
 3   bathrooms      21597 non-null  float64
 4   sqft_living    21597 non-null  int64
 5   sqft_lot       21597 non-null  int64
 6   floors         21597 non-null  float64
 7   waterfront     21597 non-null  object
 8   view           21597 non-null  object
 9   condition      21597 non-null  object
 10  grade          21597 non-null  object
 11  sqft_above     21597 non-null  int64
 12  sqft_basement  21597 non-null  float64
 13  yr_built       21597 non-null  int64
 14  yr_renovated   21597 non-null  float64
 15  zipcode        21597 non-null  int64
 16  lat            21597 non-null  float64
 17  long           21597 non-null  float64
 18  sqft_living15  21597 non-null  int64
 19  sqft_lot15     21597 non-null  int64
 20  month          21597 non-null  object
 21  month_sold     21597 non-null  int32
dtypes: float64(7), int32(1), int64(9), object(5)
memory usage: 3.5+ MB
```

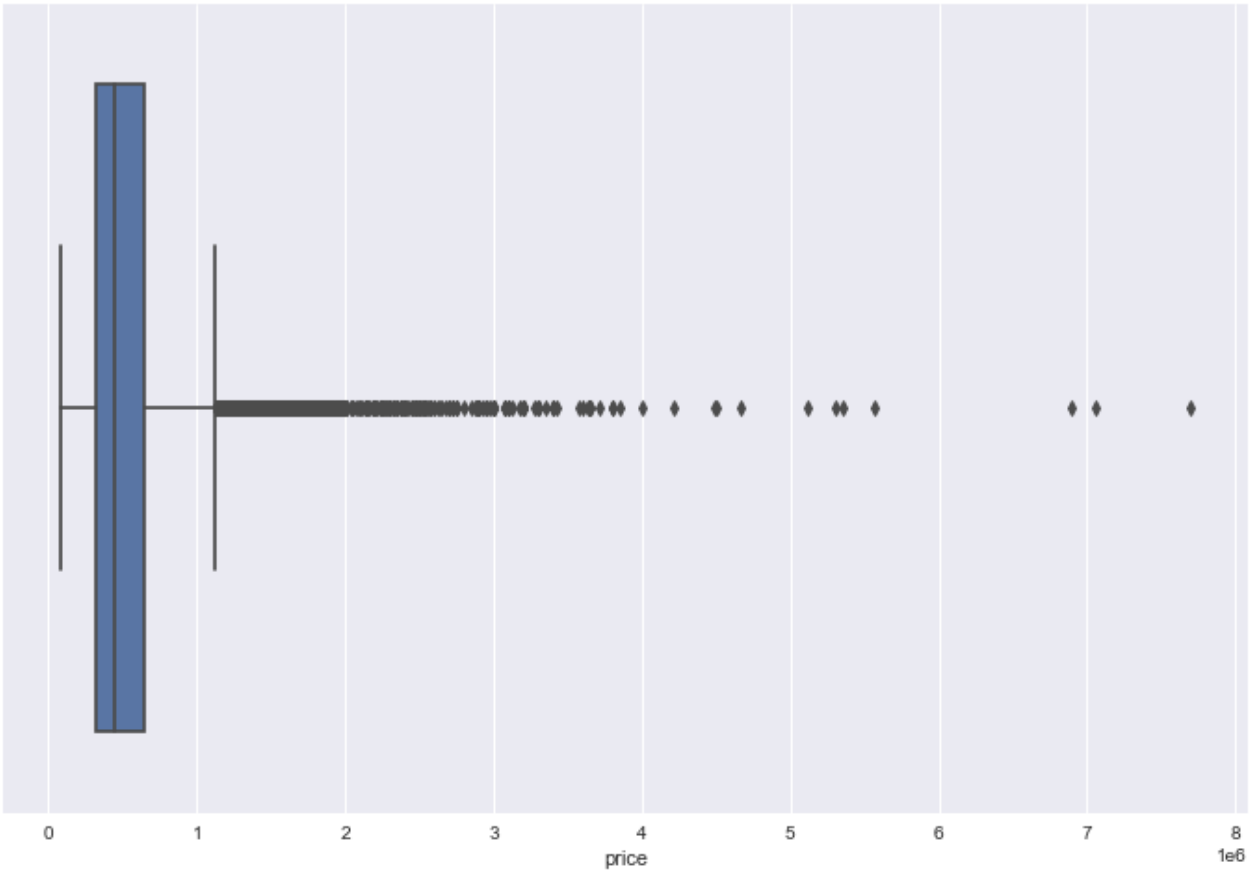
Price

In [15]:
```
df['price'].describe()
```

Out[15]:
```
count    2.159700e+04
mean     5.402966e+05
std      3.673681e+05
min      7.800000e+04
25%      3.220000e+05
50%      4.500000e+05
75%      6.450000e+05
max      7.700000e+06
Name: price, dtype: float64
```

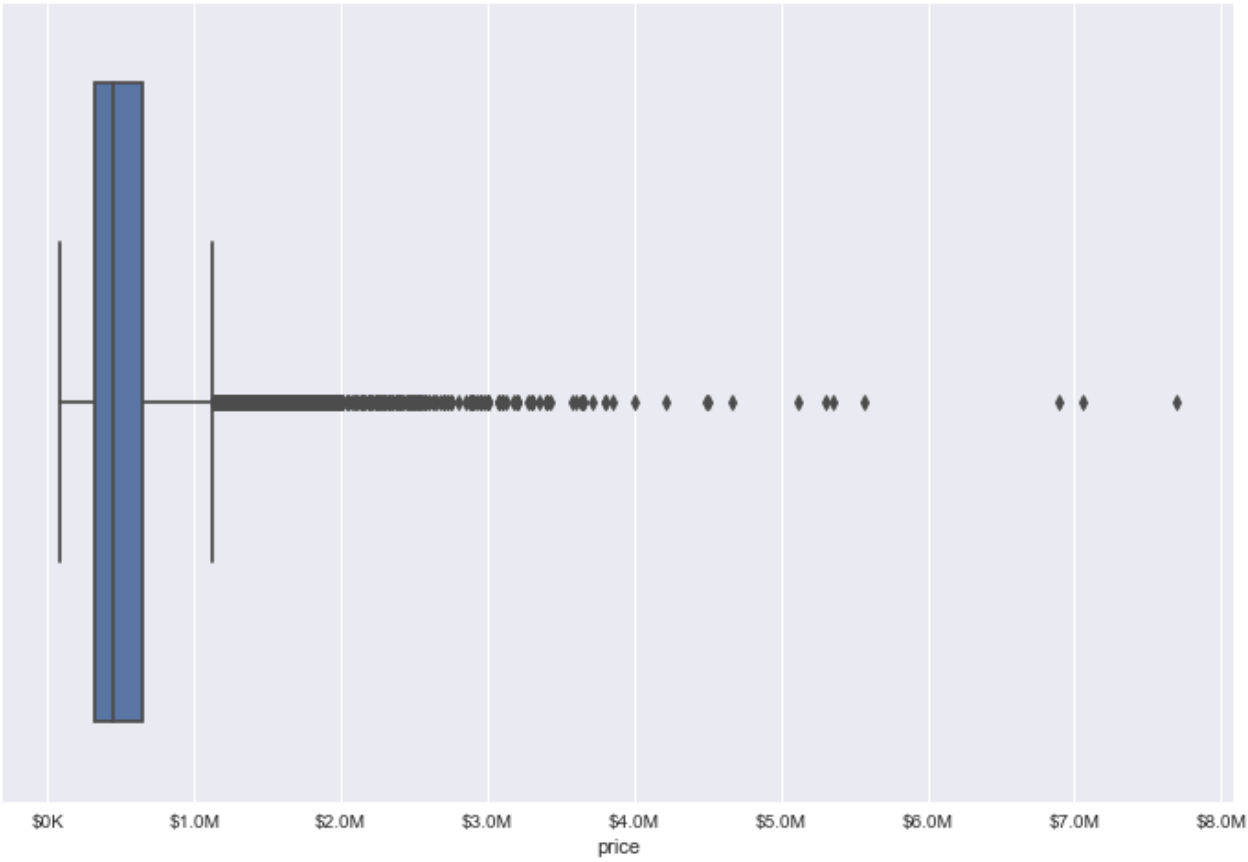
In [16]:
```
fig, ax = plt.subplots(figsize=(12,8))
sns.boxplot(x='price', data=df, ax=ax)
```

Out[16]:
```
<AxesSubplot:xlabel='price'>
```



In [17]:
```
def currency(x, pos):
    """The two args are the value and tick position"""
    if x >= 1e10:
        s = '${:1.1f}B'.format(x*1e-10)
    elif x >= 1e6:
        s = '${:1.1f}M'.format(x*1e-6)
    else:
        s = '${:1.0f}K'.format(x*1e-2)
    return s
```

In [18]:
```
fig, ax = plt.subplots(figsize=(12,8))
sns.boxplot(x='price', data=df, ax=ax)
ax.xaxis.set_major_formatter(currency)
```



In [19]:
```
df = df.loc[df['price'] < 6000000]
```

In [22]:
```
df['price'].min()
```

Out[22]:
```
78000.0
```

Bedrooms

In [23]:
```
df['bedrooms'].value_counts()
```

Out[23]:
```
3     9824
4     6882
2     2760
5     1600
6      270
1      196
7       38
8       13
9        6
10       3
11       1
33       1
Name: bedrooms, dtype: int64
```

In [24]:
```
df = df.loc[df['bedrooms'] < 10]
```

Bathrooms

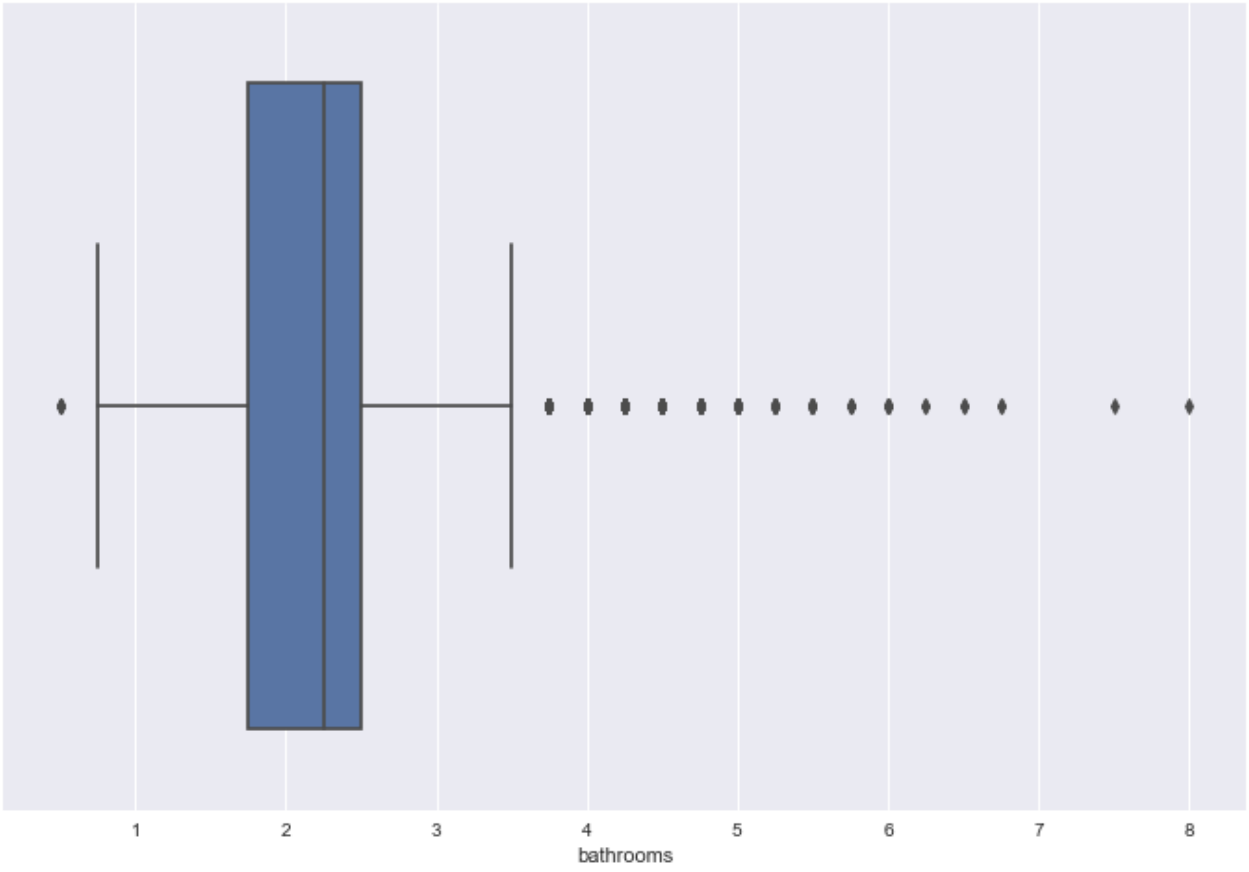
In [25]:
```
df['bathrooms'].value_counts()
```

Out[25]:
```
2.50    5377
1.00    3851
```

```
1.75    3047
2.25    2047
2.00    1929
1.50    1445
2.75    1185
3.00     751
3.50     731
3.25     589
3.75     155
4.00     136
4.50      99
4.25      79
0.75      71
4.75      23
5.00      21
5.25      12
5.50      10
1.25       9
6.00       6
0.50       4
5.75       4
6.25       2
6.50       2
6.75       2
8.00       1
7.50       1
Name: bathrooms, dtype: int64
```

In [26]:
```
fig, ax = plt.subplots(figsize=(12,8))
sns.boxplot(x='bathrooms', data=df, ax=ax)
```

Out[26]: <AxesSubplot:xlabel='bathrooms'>

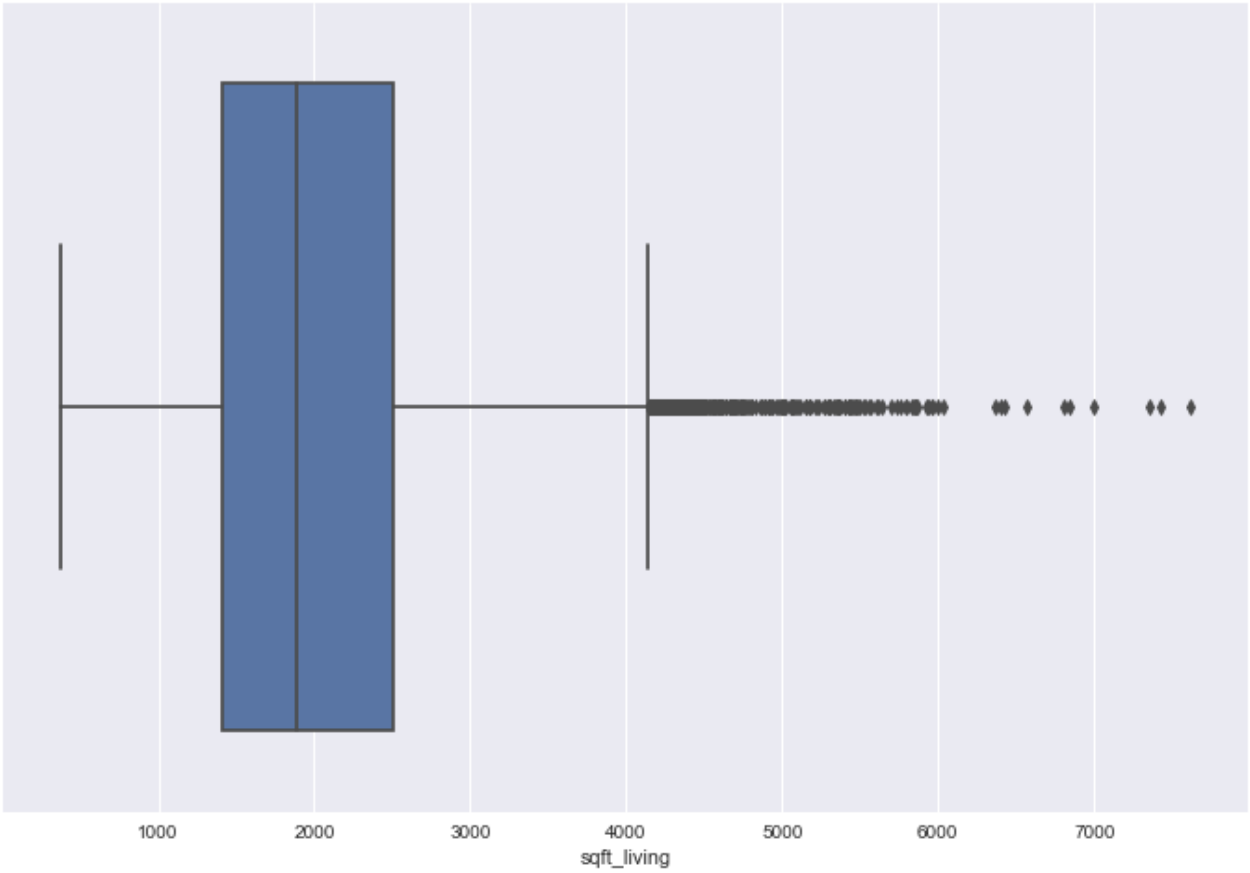


In [27]:
```
df = df.loc[df['bathrooms'] < 4]
```

Sqft_living

In [28]:
```
fig, ax = plt.subplots(figsize=(12,8))
sns.boxplot(x='sqft_living', data=df, ax=ax)
```

Out[28]: <AxesSubplot:xlabel='sqft_living'>

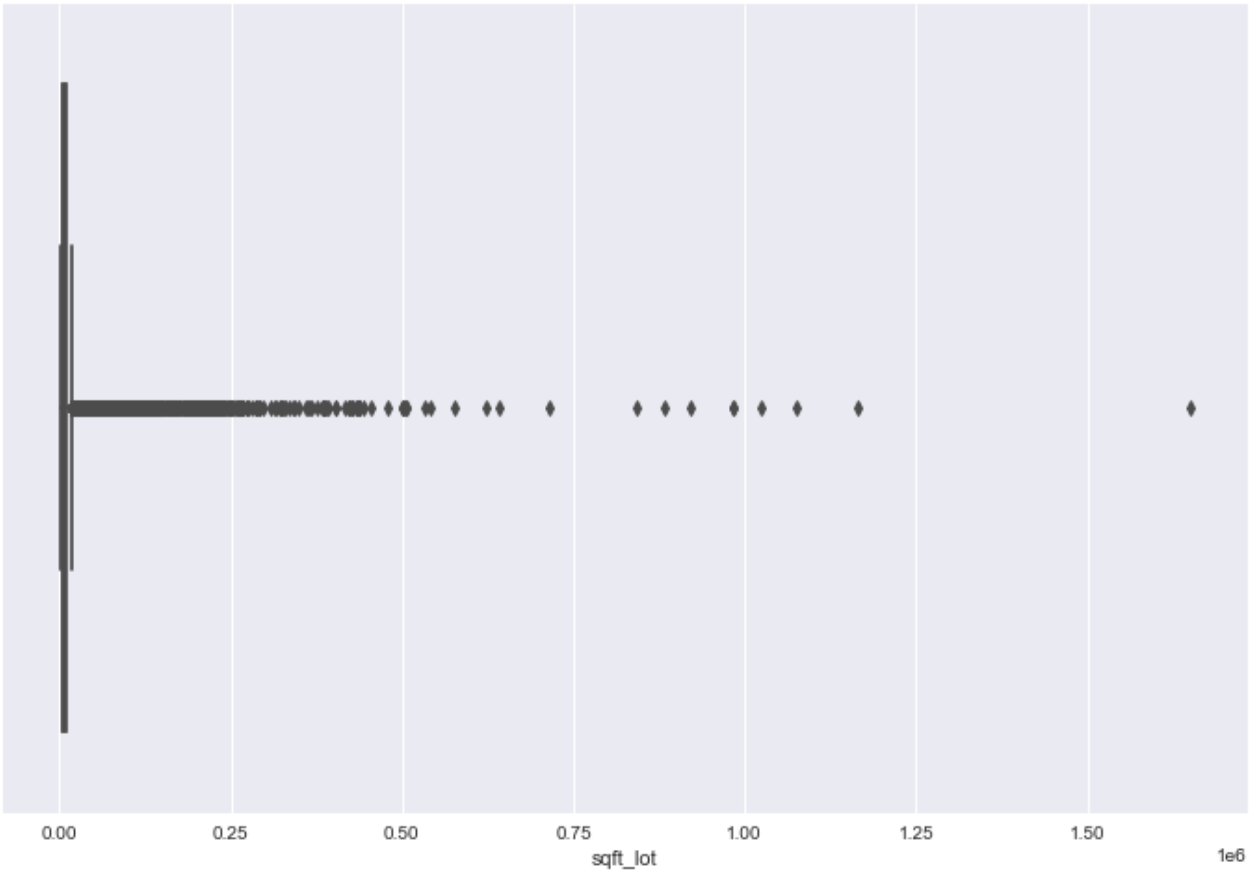


In [32]:
```
df = df.loc[df['sqft_living'] < 6000]
```

Sqft_lot

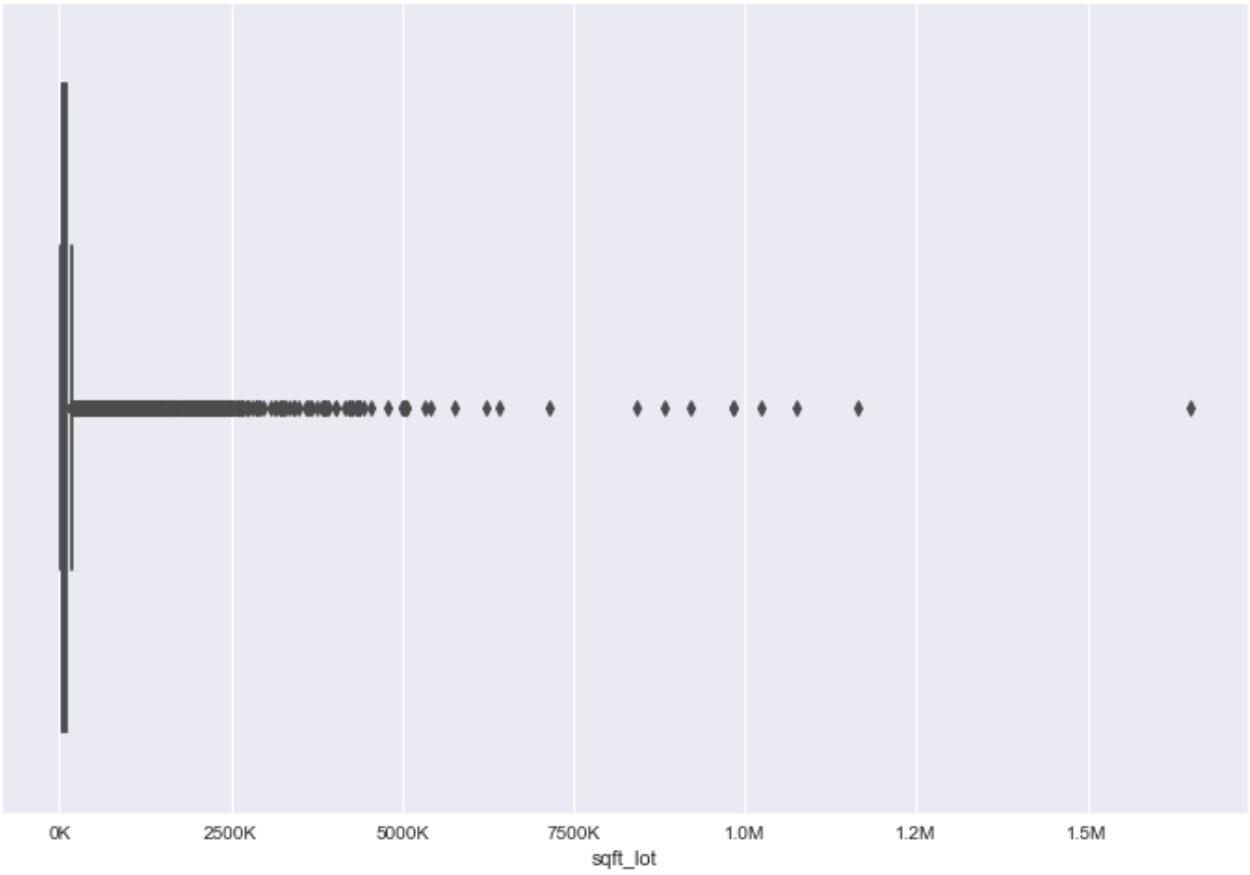
In [33]:
```
fig, ax = plt.subplots(figsize=(12,8))
sns.boxplot(x='sqft_lot', data=df, ax=ax)
```

Out[33]: <AxesSubplot:xlabel='sqft_lot'>



In [34]:
```
def number(x, pos):
    """The two args are the value and tick position"""
    if x >= 1e10:
        s = '{:1.1f}B'.format(x*1e-10)
    elif x >= 1e6:
        s = '{:1.1f}M'.format(x*1e-6)
    else:
        s = '{:1.0f}K'.format(x*1e-2)
    return s
```

In [35]:
```
fig, ax = plt.subplots(figsize=(12,8))
sns.boxplot(x='sqft_lot', data=df, ax=ax)
ax.xaxis.set_major_formatter(number)
```



In [36]:
```
df = df.loc[df['sqft_lot'] < 750000]
```

Floors

In [37]:
```
df['floors'].value_counts()
```

Out[37]:
```
1.0    10629
2.0     7908
1.5     1898
3.0      588
2.5      142
3.5        6
Name: floors, dtype: int64
```

In [38]:
```
df = df.loc[df['floors'] < 3]
```

Waterfront

In [39]:
```
df['waterfront'].value_counts()
```

Out[39]:
```
NO     20460
YES      117
Name: waterfront, dtype: int64
```

In [40]:
```
df['waterfront'] = df['waterfront'].map({'YES': 1, 'NO': 0})
```

View

In [41]:
```
df['view'].value_counts()
```

Out[41]:
```
NONE         18663
```

```
AVERAGE        887
GOOD           448
FAIR           313
EXCELLENT      266
Name: view, dtype: int64
```

Condition

In [42]:
```
df['condition'].value_counts()
```

Out[42]:
```
Average      13103
Good          5607
Very Good     1669
Fair           169
Poor            29
Name: condition, dtype: int64
```

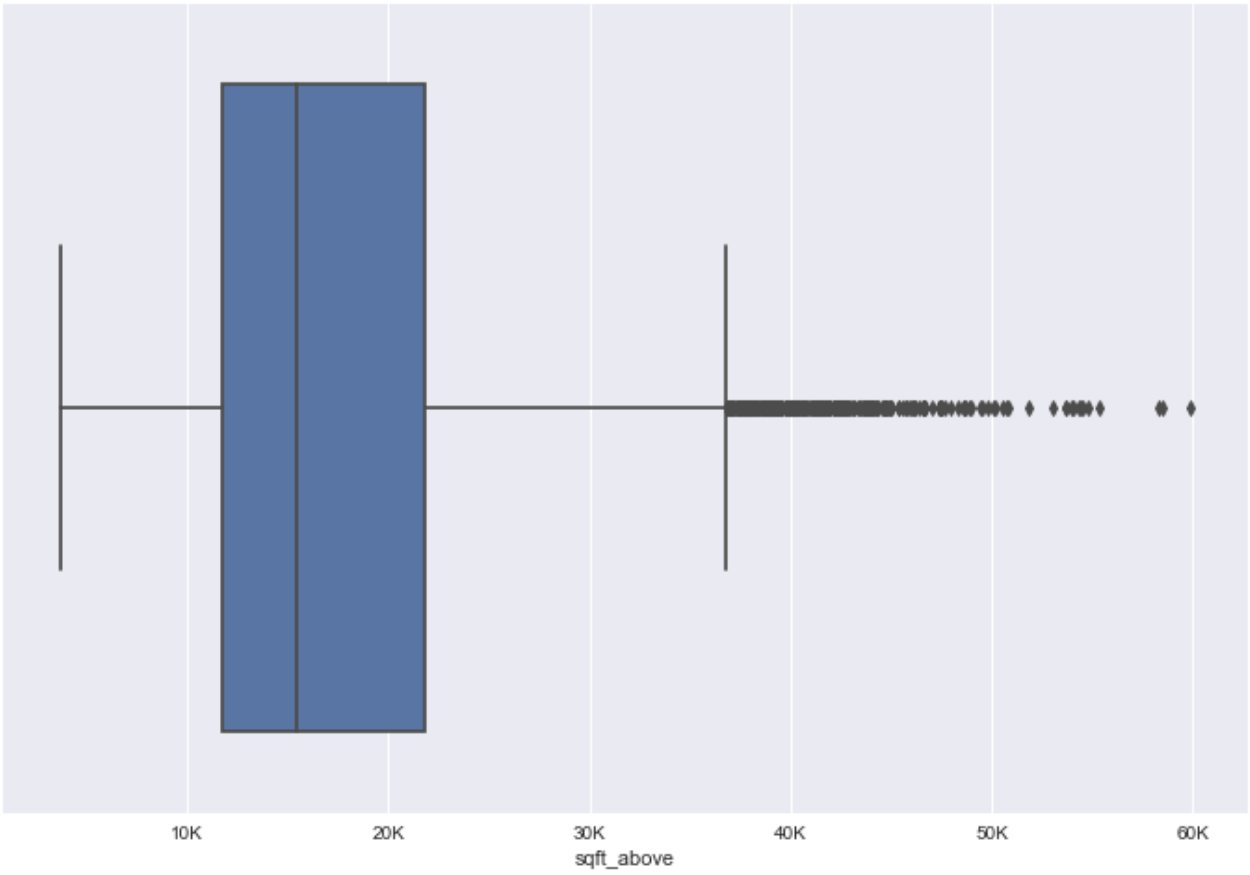
Grade

In [43]:
```
df['grade'].value_counts()
```

Out[43]:
```
7 Average        8840
8 Good           5630
9 Better         2490
6 Low Average    2037
10 Very Good      995
11 Excellent      272
5 Fair            241
12 Luxury          42
4 Low              27
13 Mansion          2
3 Poor              1
Name: grade, dtype: int64
```

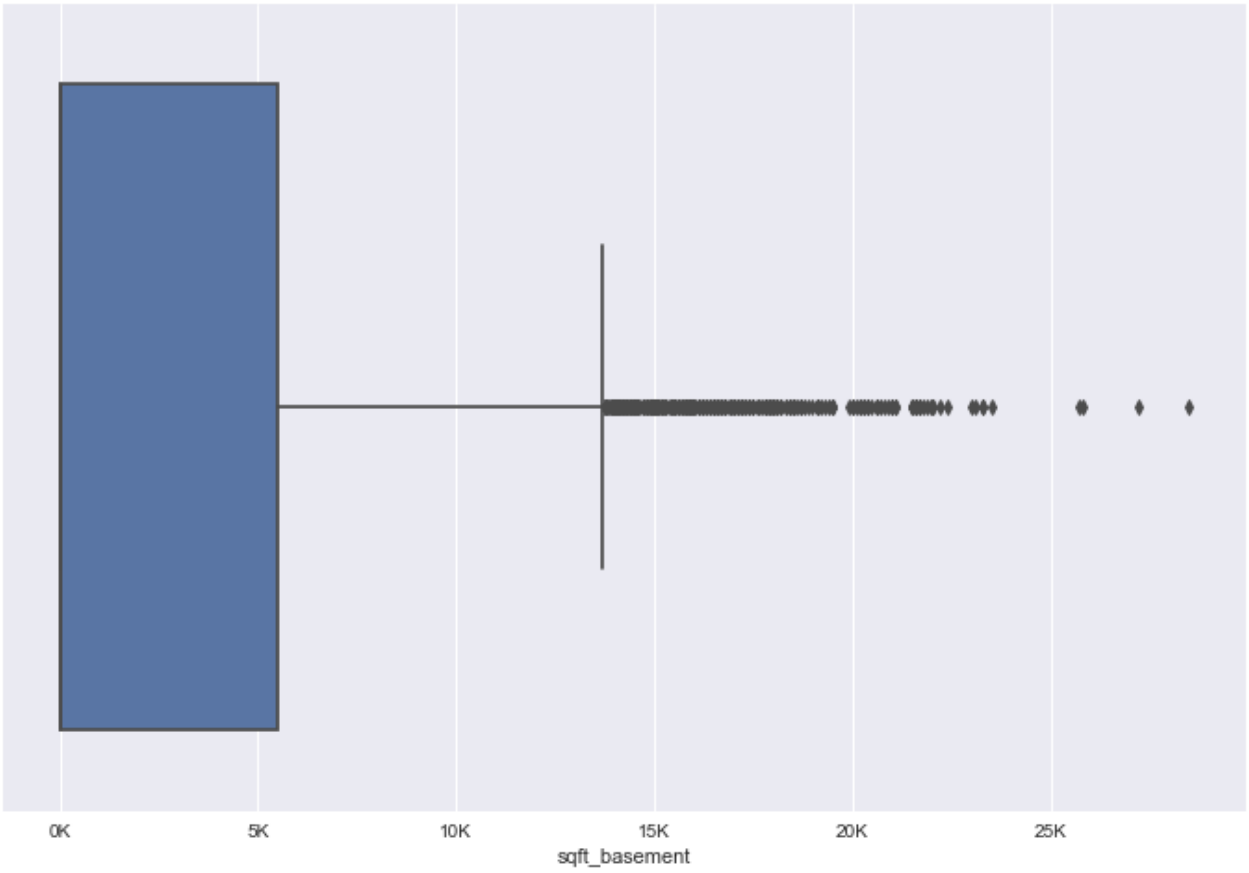
Sqft_above

In [44]:
```
fig, ax = plt.subplots(figsize=(12,8))
sns.boxplot(x='sqft_above', data=df, ax=ax)
ax.xaxis.set_major_formatter(number)
```



Sqft_basement

In [45]:
```
fig, ax = plt.subplots(figsize=(12,8))
sns.boxplot(x='sqft_basement', data=df, ax=ax)
ax.xaxis.set_major_formatter(number)
```



In [46]:
```
#create has basement category
df['has_basement'] = df['sqft_basement'].astype('bool').astype('int')
```

Yr_built

In [47]:
```
df['yr_built'].value_counts()
```

Out[47]:
```
2014    462
1977    414
2005    396
2003    394
2004    388
       ...
1933     27
1902     27
2015     25
1935     24
1934     21
Name: yr_built, Length: 116, dtype: int64
```

In [48]:
```
#create column for age of house
df['house_age'] = 2022 - df['yr_built']
```

Yr_renovated

In [49]:
```
df['yr_renovated'].value_counts()
```

Out[49]:
```
0.0       19869
2014.0       70
2007.0       29
2013.0       29
2003.0       28
```

```
          ...
1950.0       1
1953.0       1
1934.0       1
1948.0       1
1971.0       1
Name: yr_renovated, Length: 70, dtype: int64
```

In [50]:
```
#create renovated category
df['renovated'] = df['yr_renovated'].astype('bool').astype('int')
```

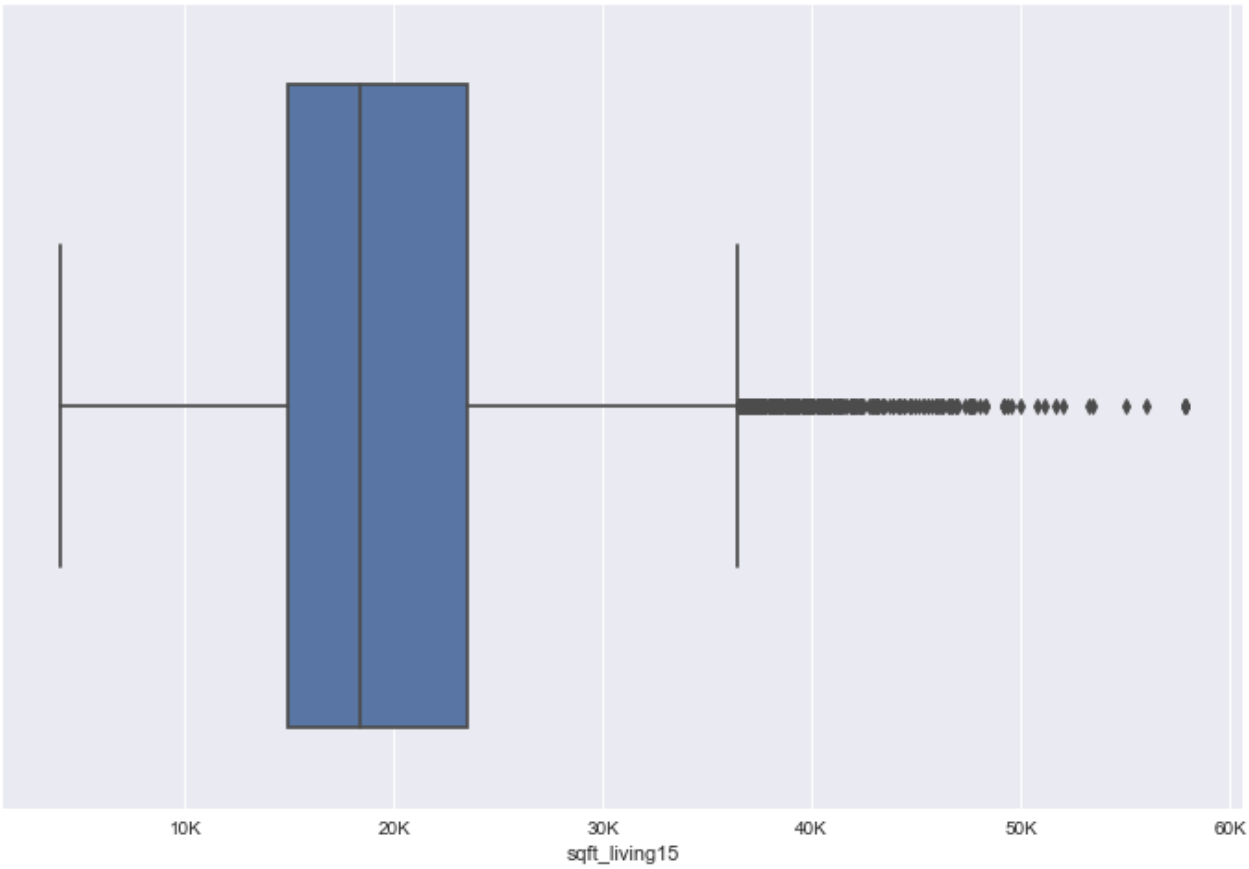
Zipcode

In [51]:
```
df['zipcode'].unique()
```

Out[51]:
```
array([98178, 98125, 98028, 98136, 98074, 98003, 98198, 98146, 98038,
       98007, 98115, 98107, 98126, 98019, 98103, 98002, 98133, 98040,
       98092, 98030, 98119, 98112, 98052, 98027, 98117, 98058, 98001,
       98056, 98166, 98053, 98023, 98070, 98148, 98105, 98042, 98008,
       98059, 98122, 98144, 98004, 98005, 98034, 98116, 98010, 98118,
       98199, 98032, 98045, 98102, 98077, 98108, 98168, 98177, 98065,
       98029, 98006, 98109, 98022, 98075, 98033, 98155, 98024, 98011,
       98031, 98106, 98072, 98188, 98014, 98055, 98039], dtype=int64)
```

Sqft_living15

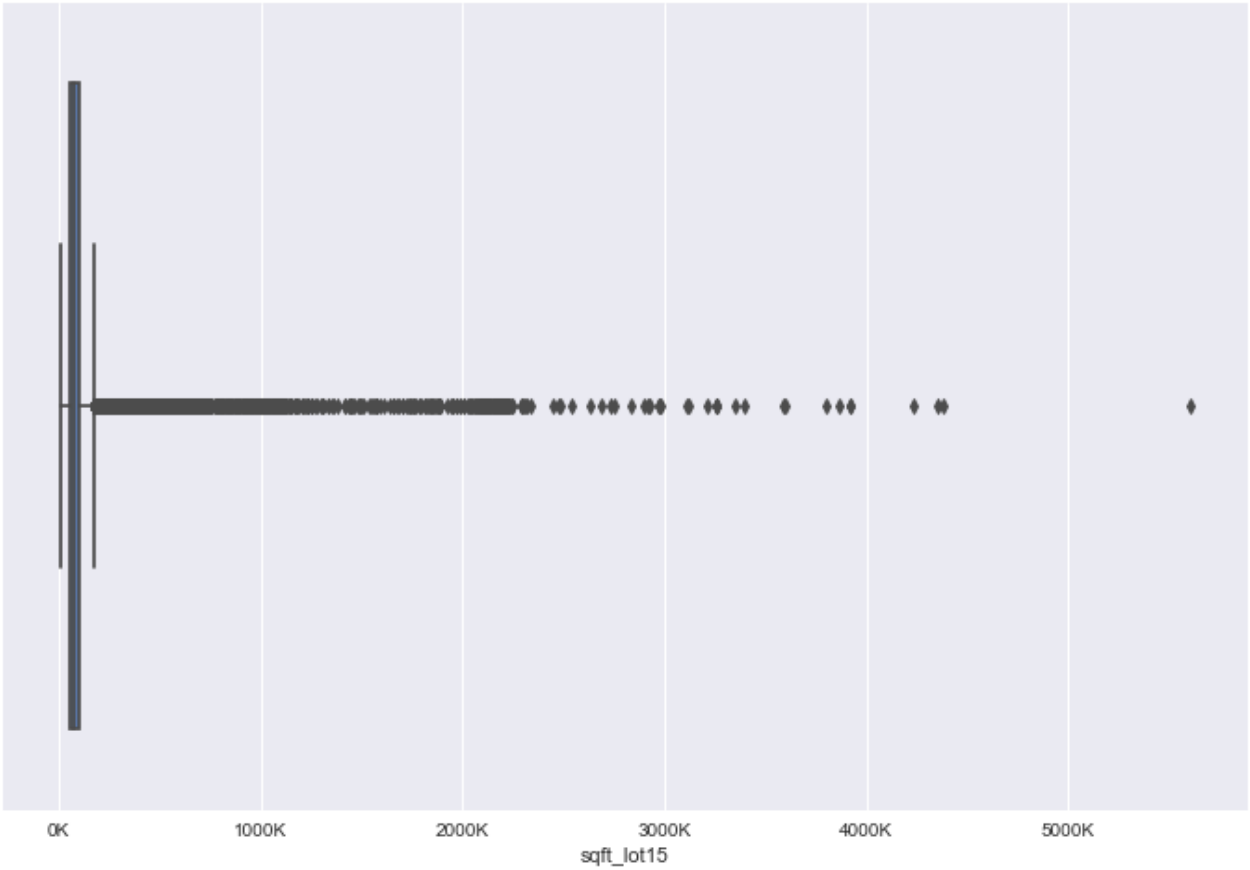
In [52]:
```
fig, ax = plt.subplots(figsize=(12,8))
sns.boxplot(x='sqft_living15', data=df, ax=ax)
ax.xaxis.set_major_formatter(number)
```



Sqft_lot15

In [53]:
```
fig, ax = plt.subplots(figsize=(12,8))
```

```
sns.boxplot(x='sqft_lot15', data=df, ax=ax)
ax.xaxis.set_major_formatter(number)
```



In [54]:
```
df = df.loc[df['sqft_lot15'] < 50000]
```

In [55]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19936 entries, 0 to 21596
Data columns (total 25 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             19936 non-null  int64
 1   price          19936 non-null  float64
 2   bedrooms       19936 non-null  int64
 3   bathrooms      19936 non-null  float64
 4   sqft_living    19936 non-null  int64
 5   sqft_lot       19936 non-null  int64
 6   floors         19936 non-null  float64
 7   waterfront     19936 non-null  int64
 8   view           19936 non-null  object
 9   condition      19936 non-null  object
 10  grade          19936 non-null  object
 11  sqft_above     19936 non-null  int64
 12  sqft_basement  19936 non-null  float64
 13  yr_built       19936 non-null  int64
 14  yr_renovated   19936 non-null  float64
 15  zipcode        19936 non-null  int64
 16  lat            19936 non-null  float64
 17  long           19936 non-null  float64
 18  sqft_living15  19936 non-null  int64
 19  sqft_lot15     19936 non-null  int64
 20  month          19936 non-null  object
```

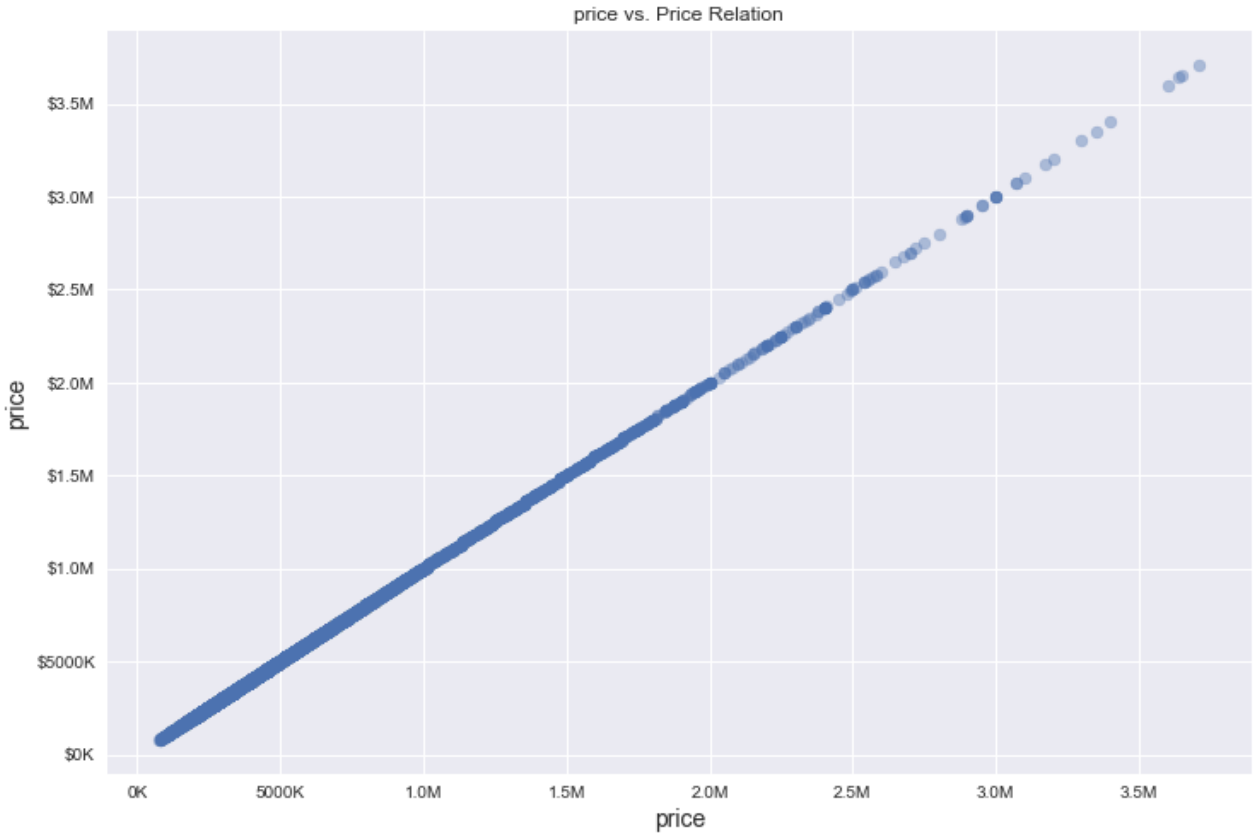
```
 21  month_sold     19936 non-null  int32
 22  has_basement   19936 non-null  int32
 23  house_age      19936 non-null  int64
 24  renovated      19936 non-null  int32
dtypes: float64(7), int32(3), int64(11), object(4)
memory usage: 3.7+ MB
```

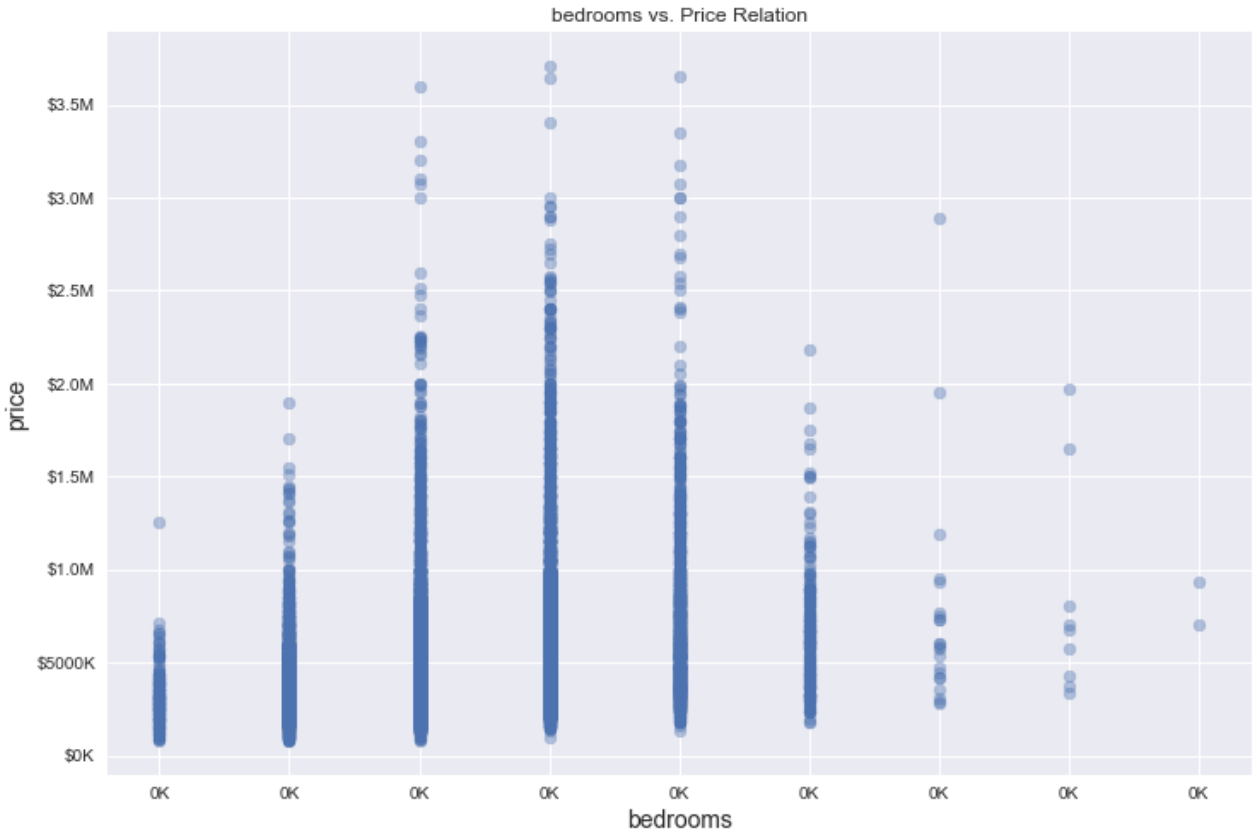
In [56]:
```
#Dropping columns I won't use
df.drop(columns=['id','month','lat','long', 'yr_renovated',
                 'yr_built', 'sqft_basement'],inplace=True)
```

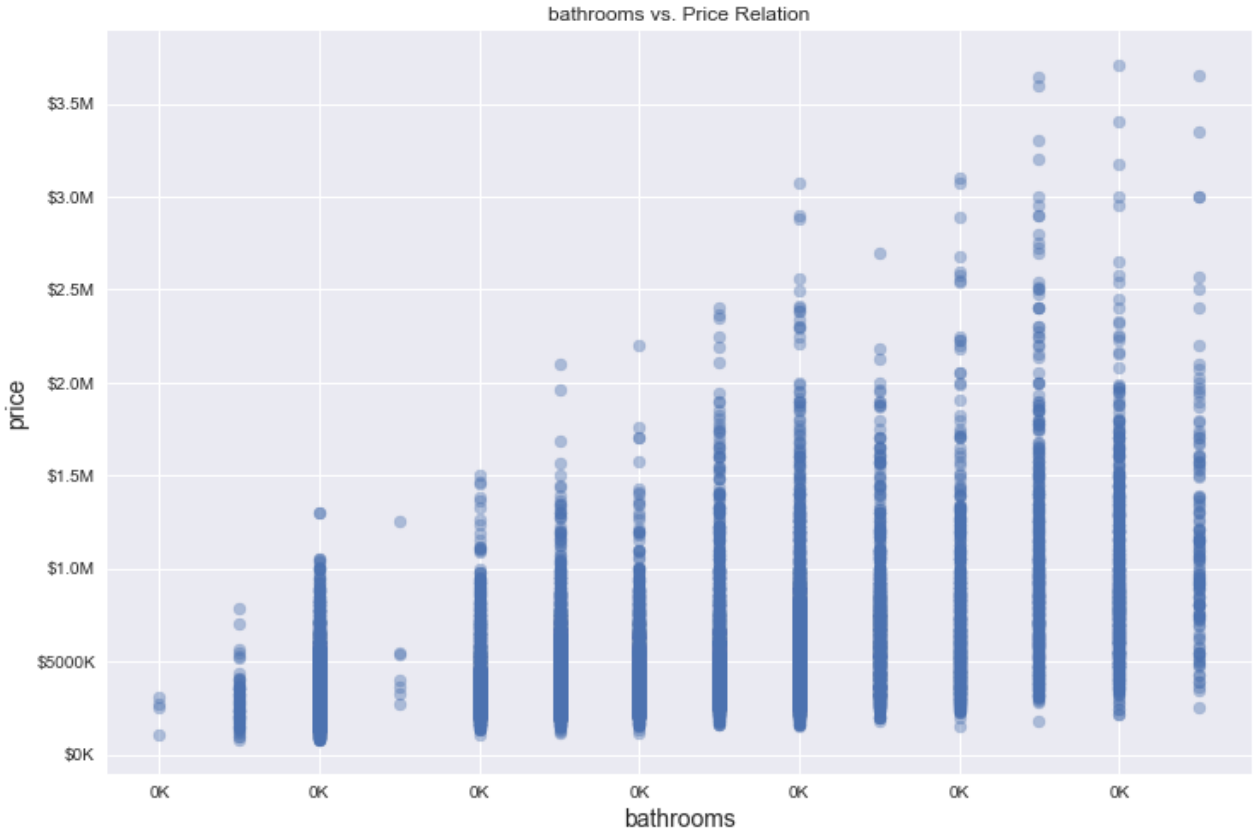
In [57]:
```
df.info()
```

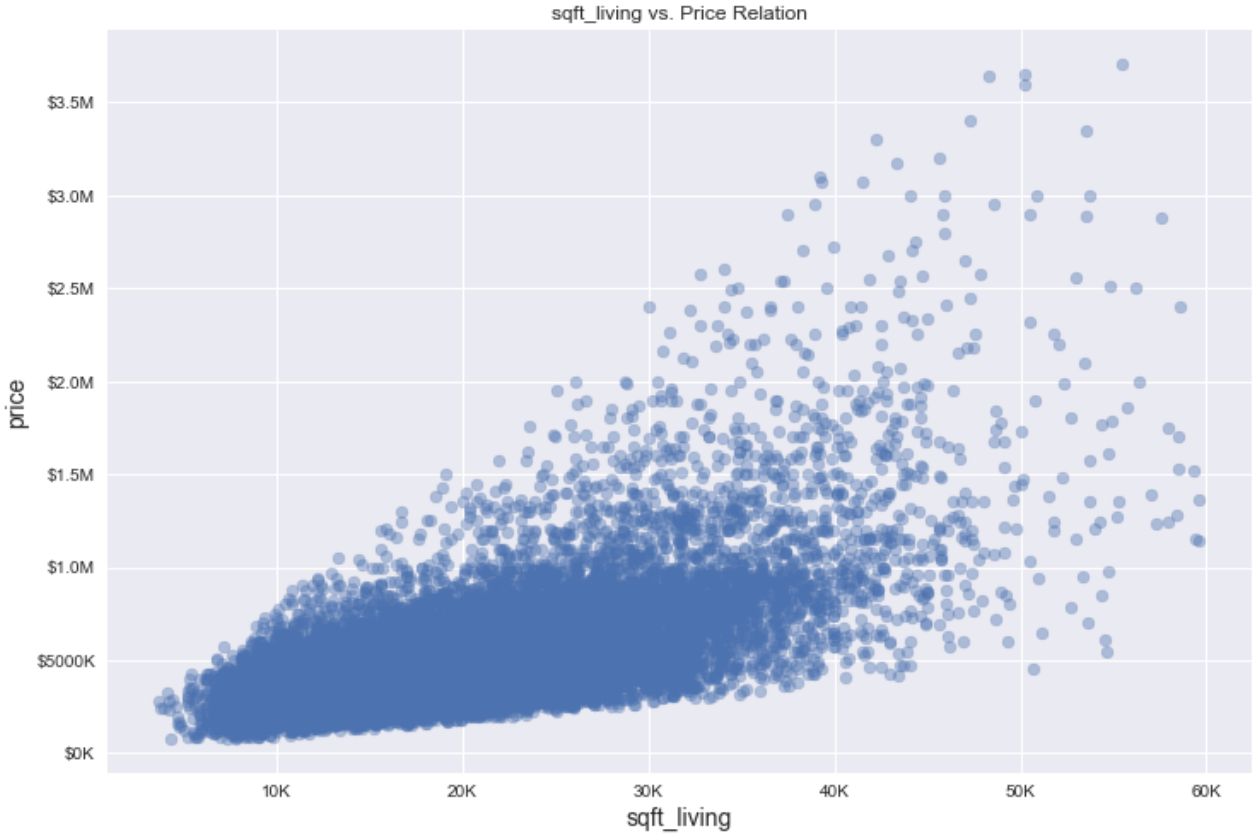
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19936 entries, 0 to 21596
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   price          19936 non-null  float64
 1   bedrooms       19936 non-null  int64
 2   bathrooms      19936 non-null  float64
 3   sqft_living    19936 non-null  int64
 4   sqft_lot       19936 non-null  int64
 5   floors         19936 non-null  float64
 6   waterfront     19936 non-null  int64
 7   view           19936 non-null  object
 8   condition      19936 non-null  object
 9   grade          19936 non-null  object
 10  sqft_above     19936 non-null  int64
 11  zipcode        19936 non-null  int64
 12  sqft_living15  19936 non-null  int64
 13  sqft_lot15     19936 non-null  int64
 14  month_sold     19936 non-null  int32
 15  has_basement   19936 non-null  int32
 16  house_age      19936 non-null  int64
 17  renovated      19936 non-null  int32
dtypes: float64(3), int32(3), int64(9), object(3)
memory usage: 2.7+ MB
```

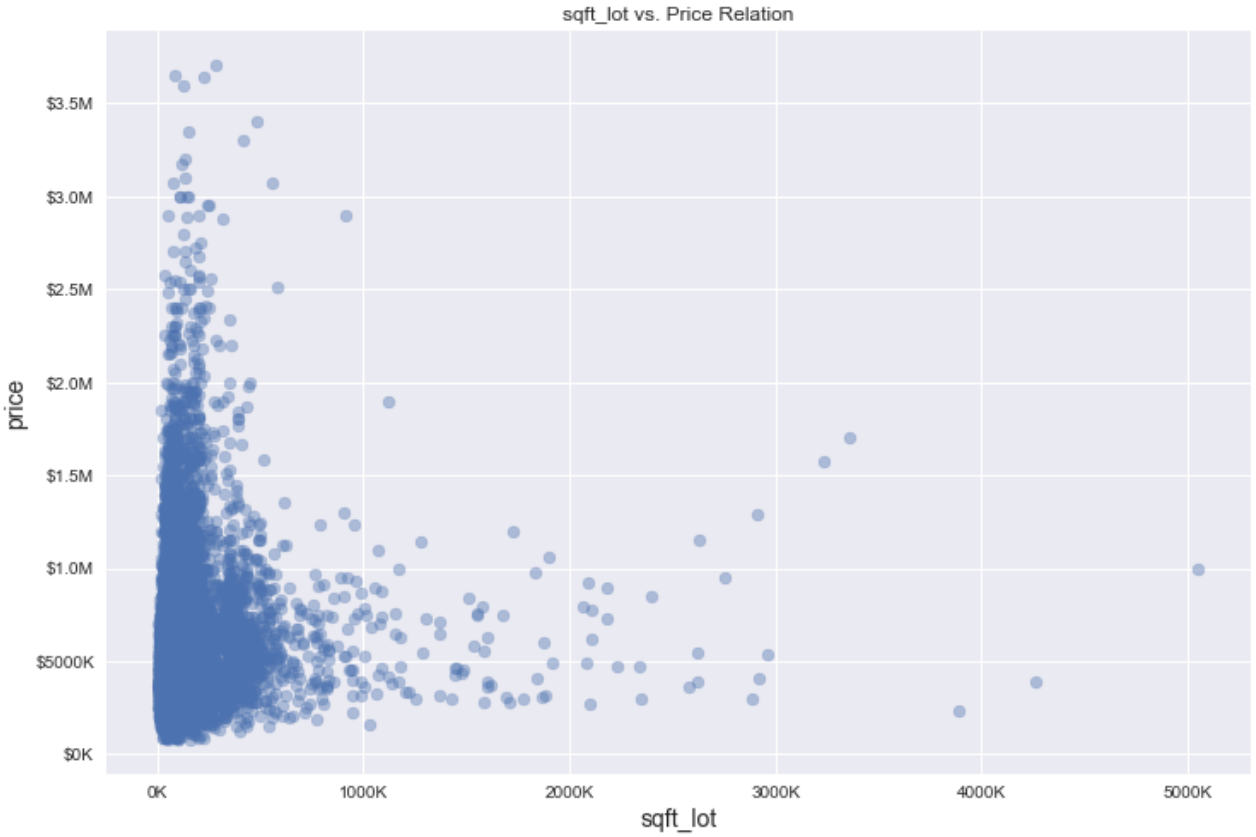
In [58]:
```
#getting all numerical columns in df
for col in df.describe().columns:
    fig , ax = plt.subplots(figsize=(12,8))
    ax.scatter(df[col], df.price ,alpha=.4)
    ax.set_xlabel(col, fontsize=14)
    ax.set_ylabel('price',fontsize=14)
    ax.set_title(f'{col} vs. Price Relation')
    ax.yaxis.set_major_formatter(currency)
    ax.xaxis.set_major_formatter(number)
    plt.show()
```

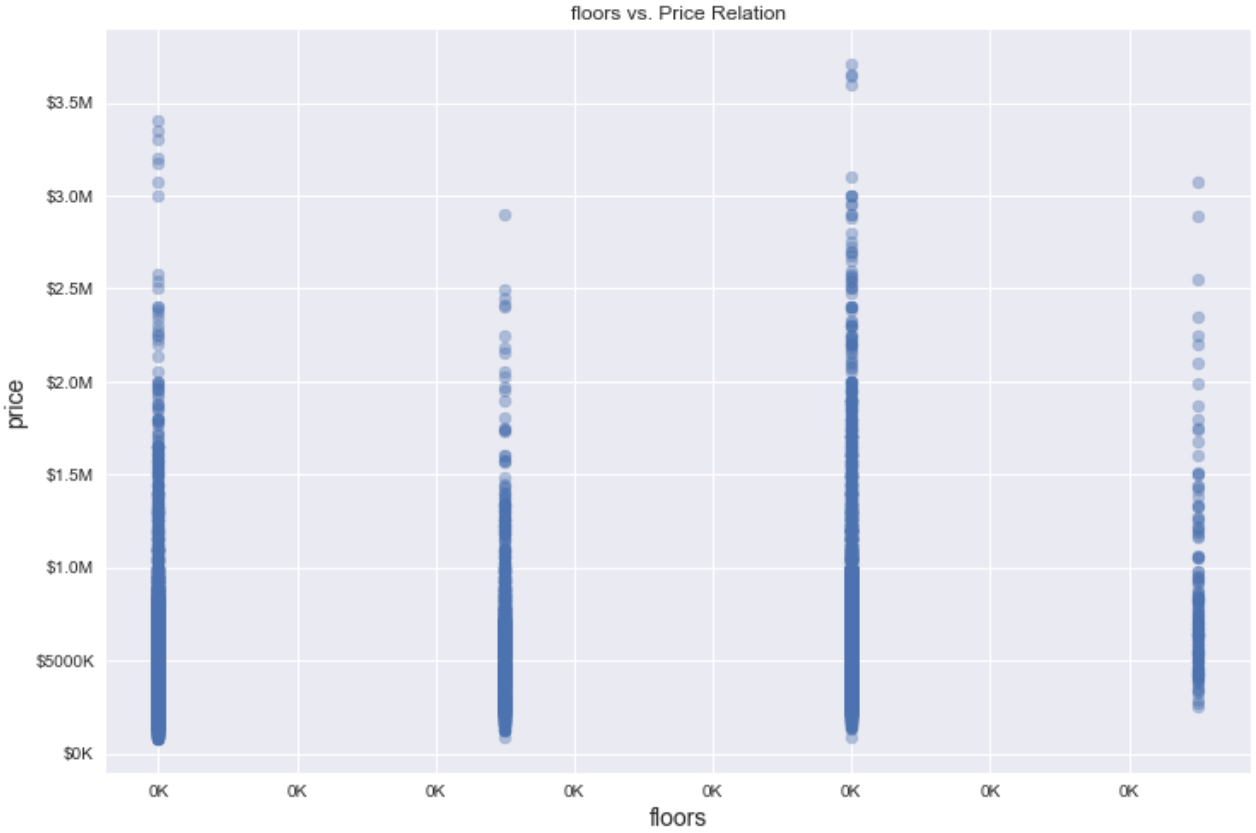

price vs. Price Relation
$3.5M
$3.0M
$2.5M
$2.0M
$1.5M
$1.0M
$5000K
$0K
price
0K
5000K
1.0M
1.5M
2.0M
2.5M
3.0M
3.5M
price

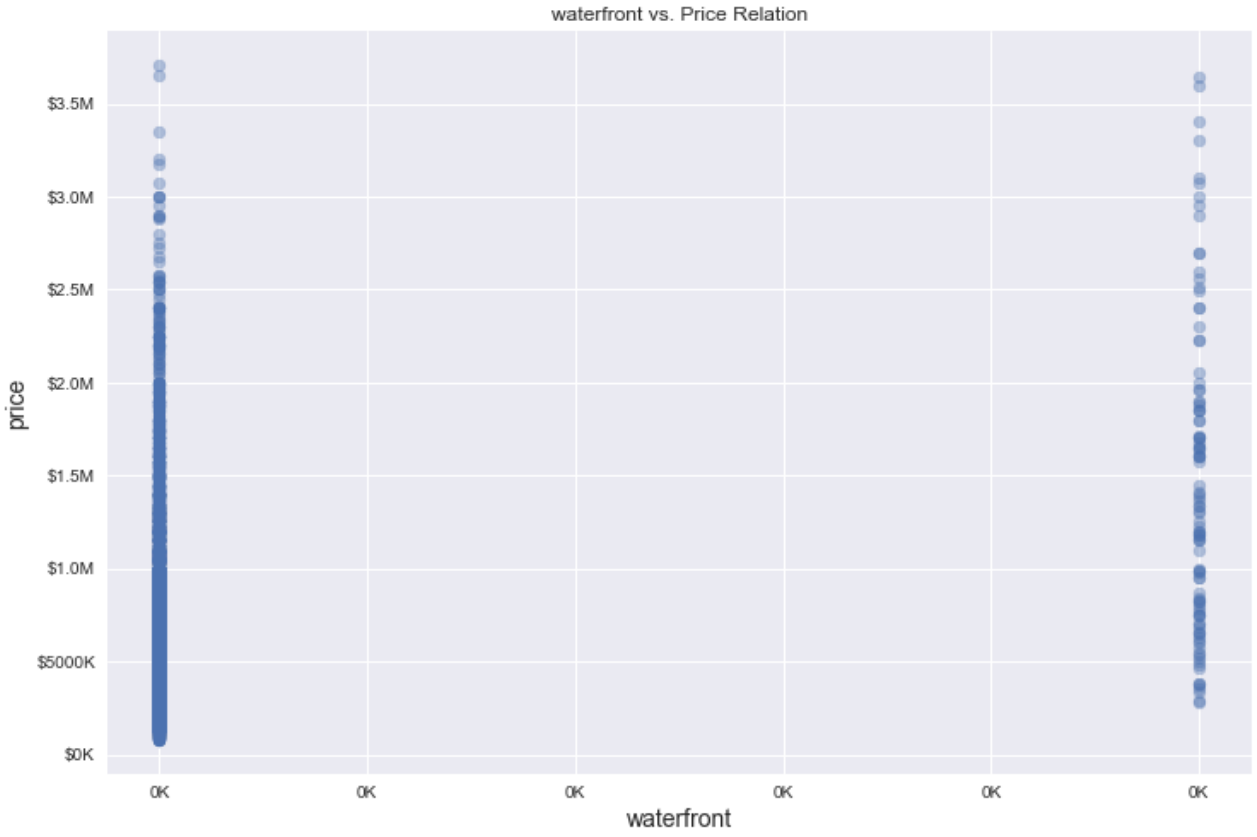

bedrooms vs. Price Relation
$3.5M
$3.0M
$2.5M
$2.0M
$1.5M
$1.0M
$5000K
$0K
price
0K
0K
0K
0K
0K
0K
0K
0K
0K
bedrooms

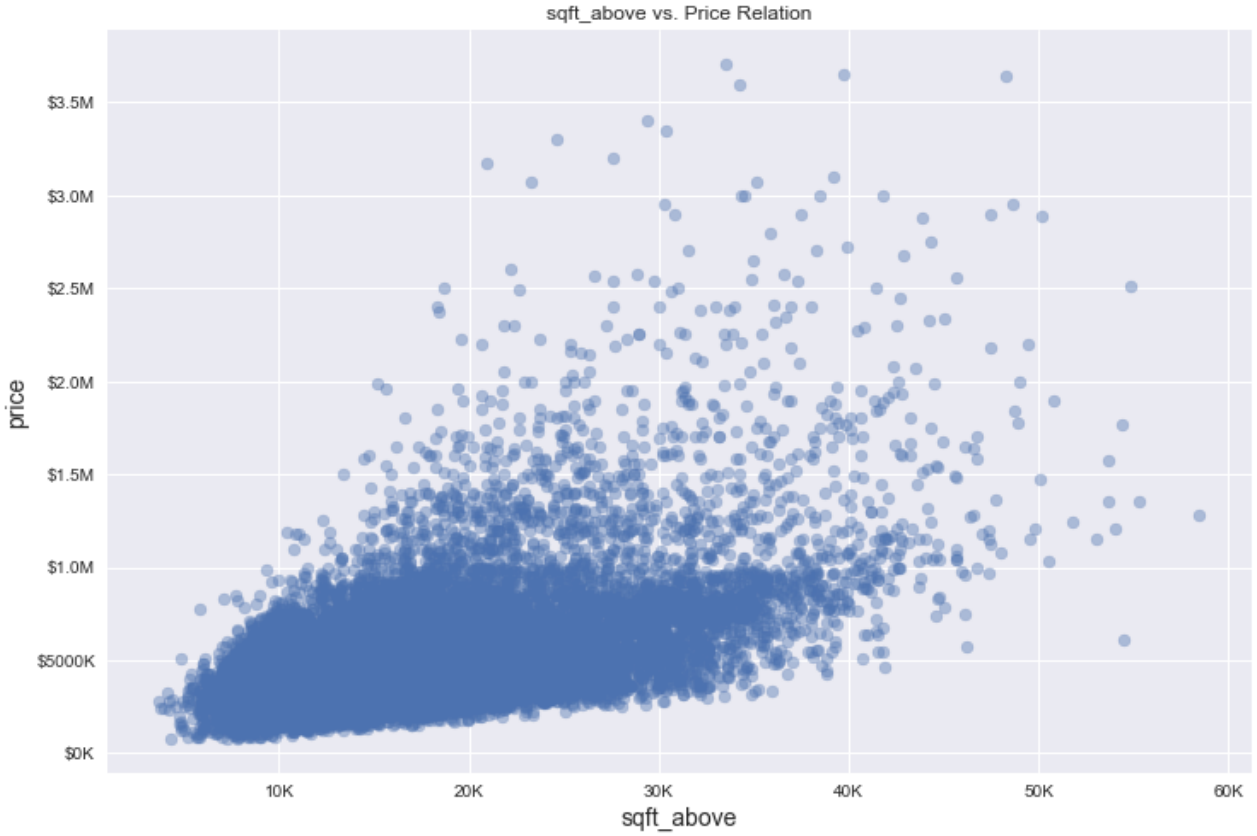

bathrooms vs. Price Relation
price
$3.5M
$3.0M
$2.5M
$2.0M
$1.5M
$1.0M
$5000K
$0K
0K
0K
0K
0K
0K
0K
0K
bathrooms

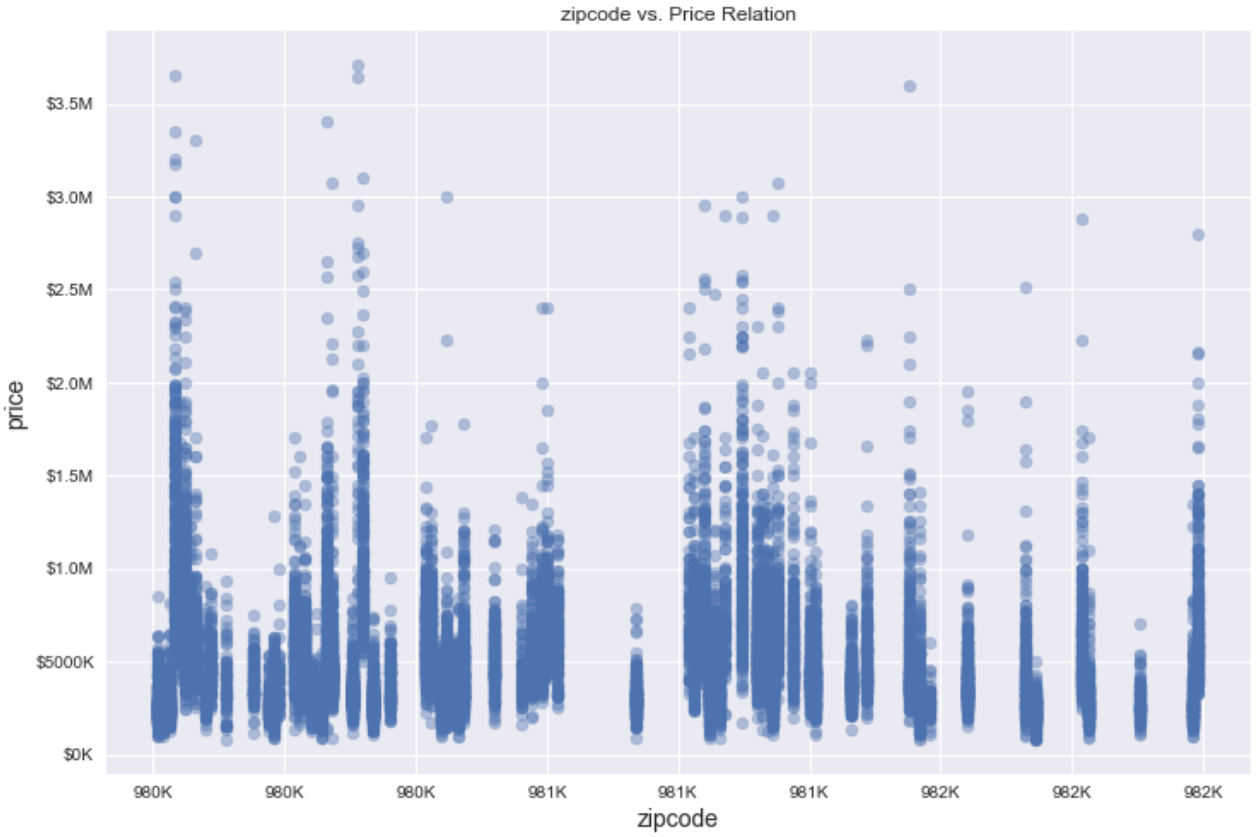

sqft_living vs. Price Relation
price
$3.5M
$3.0M
$2.5M
$2.0M
$1.5M
$1.0M
$5000K
$0K
10K
20K
30K
40K
50K
60K
sqft_living

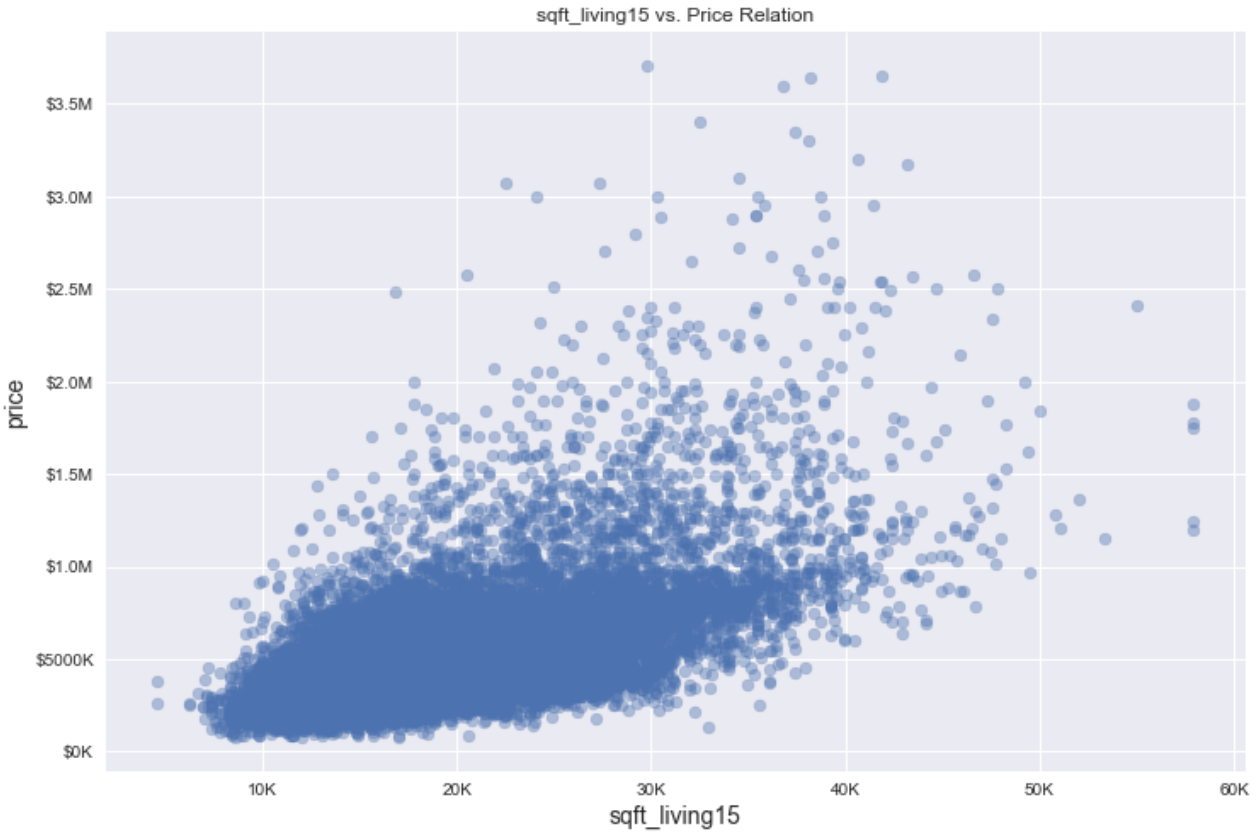

sqft_lot vs. Price Relation
price
$3.5M
$3.0M
$2.5M
$2.0M
$1.5M
$1.0M
$5000K
$0K
0K
1000K
2000K
3000K
4000K
5000K
sqft_lot

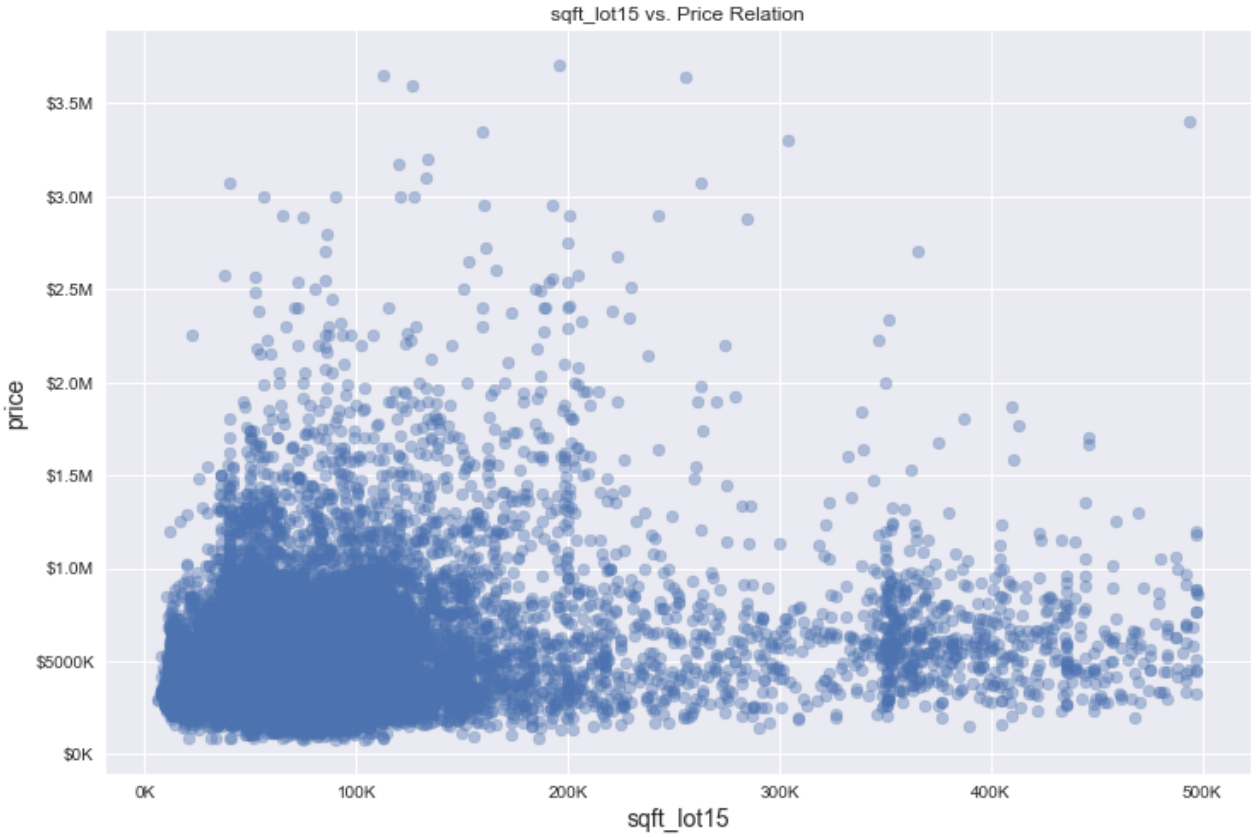

floors vs. Price Relation
price
$3.5M
$3.0M
$2.5M
$2.0M
$1.5M
$1.0M
$5000K
$0K
0K
floors

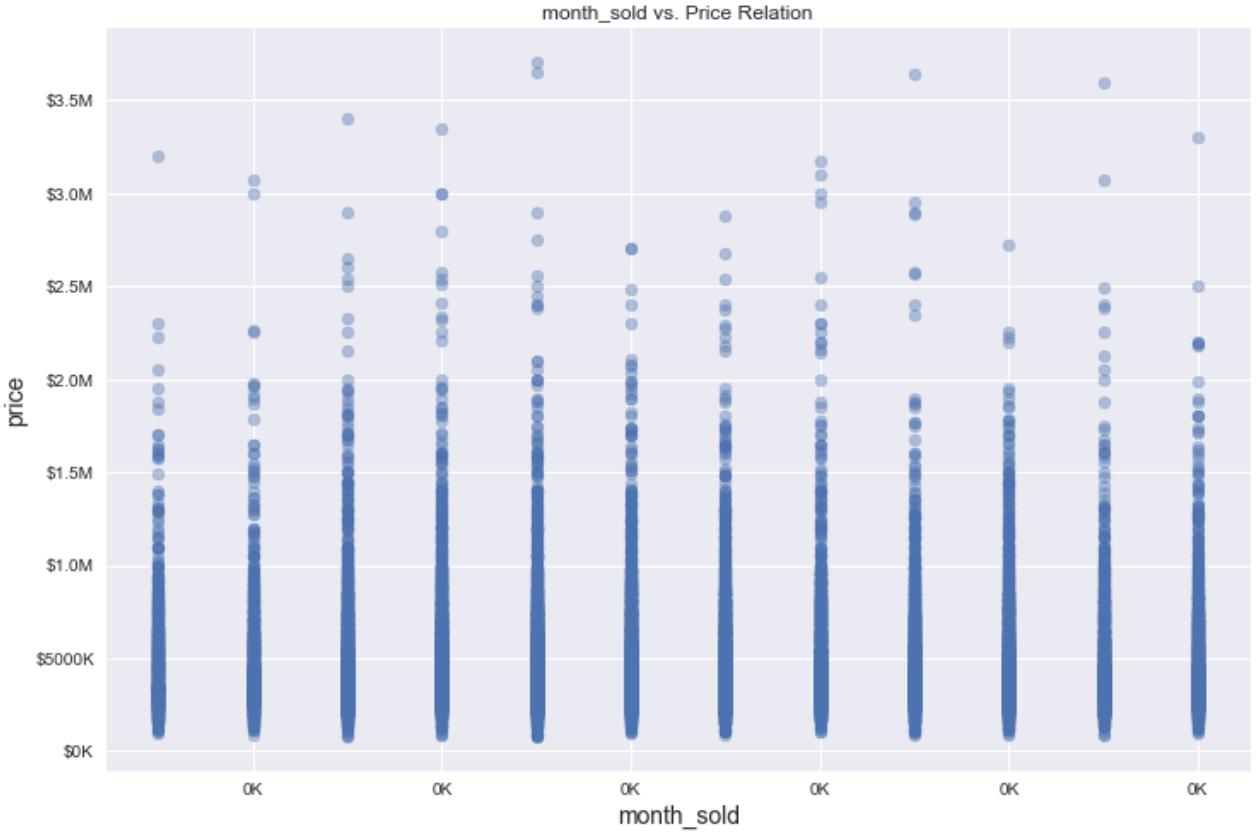

waterfront vs. Price Relation
price
$3.5M
$3.0M
$2.5M
$2.0M
$1.5M
$1.0M
$5000K
$0K
0K
0K
0K
0K
0K
0K
waterfront

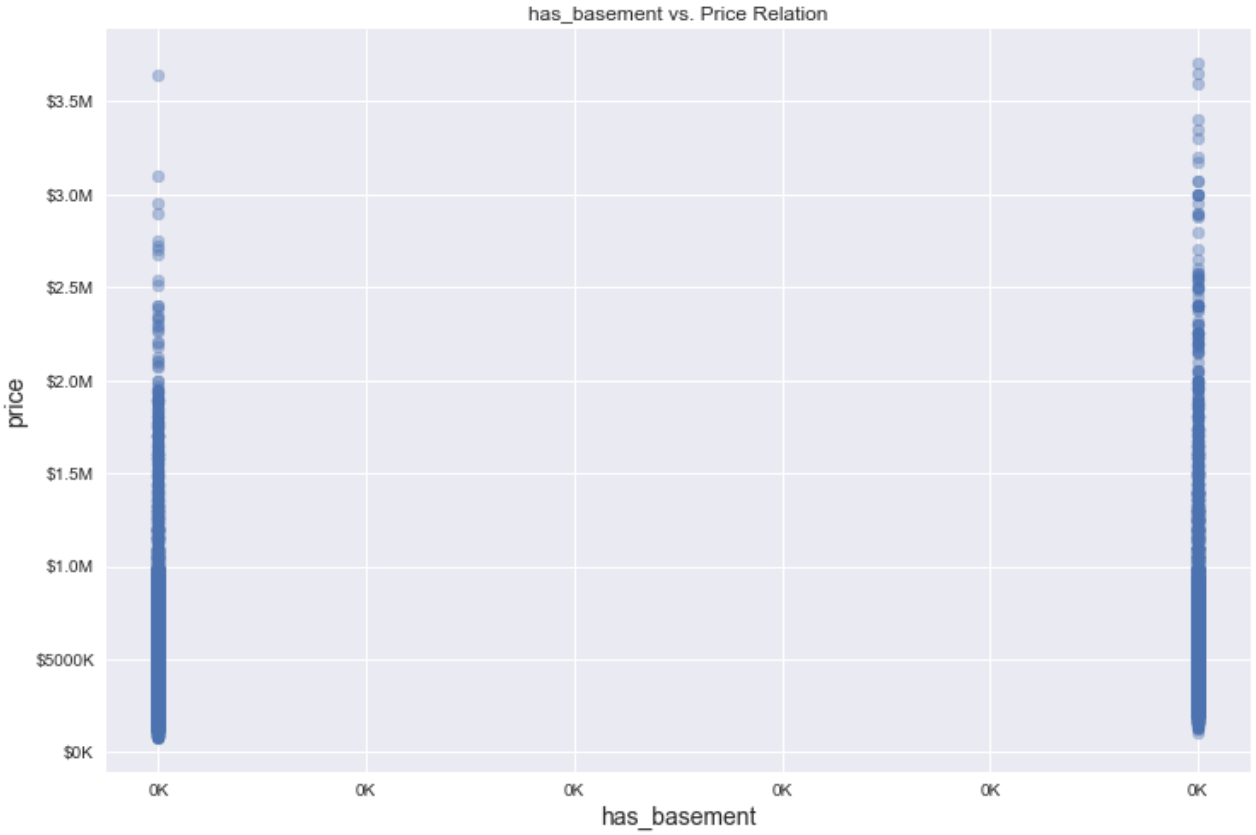

sqft_above vs. Price Relation
price
$3.5M
$3.0M
$2.5M
$2.0M
$1.5M
$1.0M
$5000K
$0K
10K
20K
30K
40K
50K
60K
sqft_above

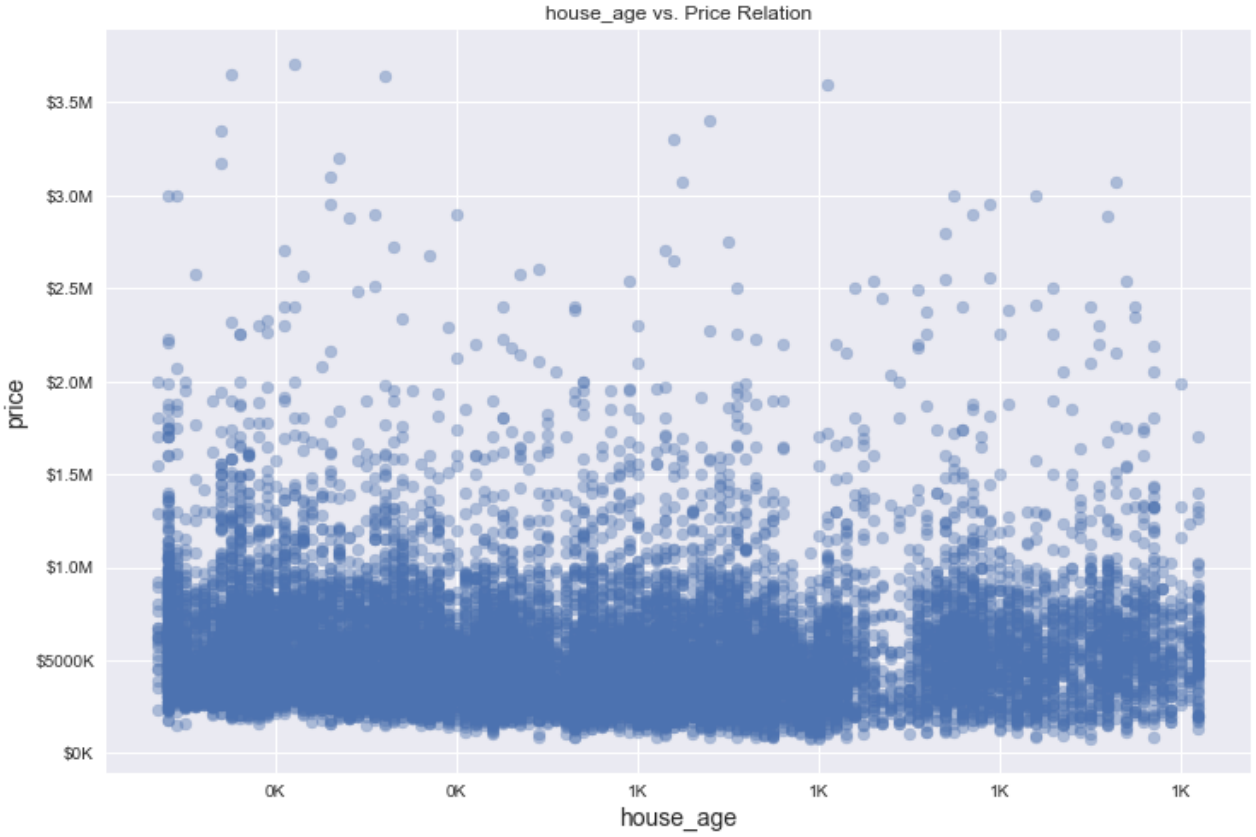

zipcode vs. Price Relation
price
$3.5M
$3.0M
$2.5M
$2.0M
$1.5M
$1.0M
$5000K
$0K
980K
980K
980K
981K
981K
981K
982K
982K
982K
zipcode

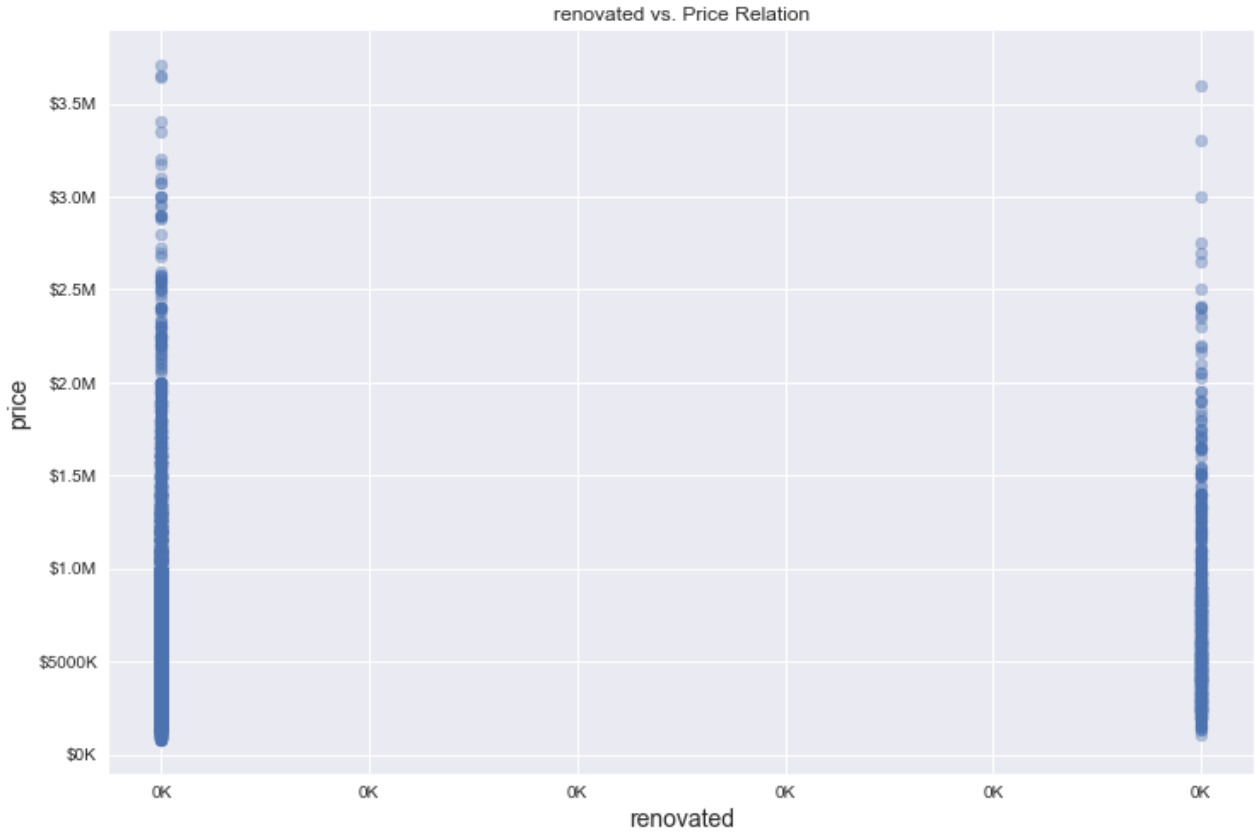

sqft_living15 vs. Price Relation
price
$3.5M
$3.0M
$2.5M
$2.0M
$1.5M
$1.0M
$5000K
$0K
10K
20K
30K
40K
50K
60K
sqft_living15


sqft_lot15 vs. Price Relation
price
$3.5M
$3.0M
$2.5M
$2.0M
$1.5M
$1.0M
$5000K
$0K
0K
100K
200K
300K
400K
500K
sqft_lot15


month_sold vs. Price Relation
price
$3.5M
$3.0M
$2.5M
$2.0M
$1.5M
$1.0M
$5000K
$0K
0K
0K
0K
0K
0K
0K
month_sold


has_basement vs. Price Relation
price
$3.5M
$3.0M
$2.5M
$2.0M
$1.5M
$1.0M
$5000K
$0K
0K
0K
0K
0K
0K
0K
has_basement


house_age vs. Price Relation
price
$3.5M
$3.0M
$2.5M
$2.0M
$1.5M
$1.0M
$5000K
$0K
0K
0K
1K
1K
1K
1K
house_age



In [59]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19936 entries, 0 to 21596
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   price          19936 non-null  float64
 1   bedrooms       19936 non-null  int64  
 2   bathrooms      19936 non-null  float64
 3   sqft_living    19936 non-null  int64  
 4   sqft_lot       19936 non-null  int64  
 5   floors         19936 non-null  float64
 6   waterfront     19936 non-null  int64  
 7   view           19936 non-null  object 
 8   condition      19936 non-null  object 
 9   grade          19936 non-null  object 
 10  sqft_above     19936 non-null  int64  
 11  zipcode        19936 non-null  int64  
 12  sqft_living15  19936 non-null  int64  
 13  sqft_lot15     19936 non-null  int64  
 14  month_sold     19936 non-null  int32  
 15  has_basement   19936 non-null  int32  
 16  house_age      19936 non-null  int64  
 17  renovated      19936 non-null  int32  
dtypes: float64(3), int32(3), int64(9), object(3)
memory usage: 2.7+ MB
```

In [60]:
```
#change zipcode from int to category to OHE
df['zipcode']=df['zipcode'].astype('category')
```

Train Test Split

In [61]:
```
y = df['price']
X= df.drop('price',axis=1)
```

In [62]:
```
X_train , X_test , y_train , y_test = train_test_split(X,y,test_size=0.2,random_state =
X_train.shape , X_test.shape , y_train.shape , y_test.shape
```

Out[62]: ((15948, 17), (3988, 17), (15948,), (3988,))

OHE

In [63]:
```
#splitting up the continous vs. categorical data
df_continuous = ['sqft_living', 'sqft_above' 'sqft_living15', 'sqft_lot', 'sqft_lot15']

df_categorical = ['waterfront', 'bedrooms', 'grade', 'renovated', 'view', 'zipcode', 'h
```

In [64]:
```
X_train_cat=pd.DataFrame()
for i in df_categorical:
    X_train_cat[i]=X_train[i].astype('category')

X_test_cat=pd.DataFrame()
for i in df_categorical:
    X_test_cat[i]=X_test[i].astype('category')
```

In [ ]:
```

```

In [65]:
```
for i in df_categorical:
    dummies=pd.get_dummies(X_train_cat[i],prefix=i, drop_first=True)
    X_train_cat=X_train_cat.join(dummies)
    X_train_cat.drop([i], axis=1, inplace=True)


for i in df_categorical:
    dummies=pd.get_dummies(X_test_cat[i],prefix=i, drop_first=True)
    X_test_cat=X_test_cat.join(dummies)
    X_test_cat.drop([i], axis=1, inplace=True)
```

Multicollinearity

In [66]:
```
x = df.drop('price',axis=1)
```

In [67]:
```
# # Creating VIF Dictionary.
def create_vif_dct(dataframe, const_col_name='const'):

    if const_col_name not in dataframe.columns:
        dataframe = sm.add_constant(dataframe)

    # Dummy-checking.
    df = dataframe.select_dtypes('number')
    if df.shape != dataframe.shape:
        warnings.warn('\n\nThere are non-numerical columns trying to be passed!\nThese
    if df.isna().sum().any():
        raise ValueError('There may not be any missing values in the dataframe!')

    # Creating VIF Dictionary.
    vif_dct = {}

    # Loop through each row and set the variable name to the VIF.
```

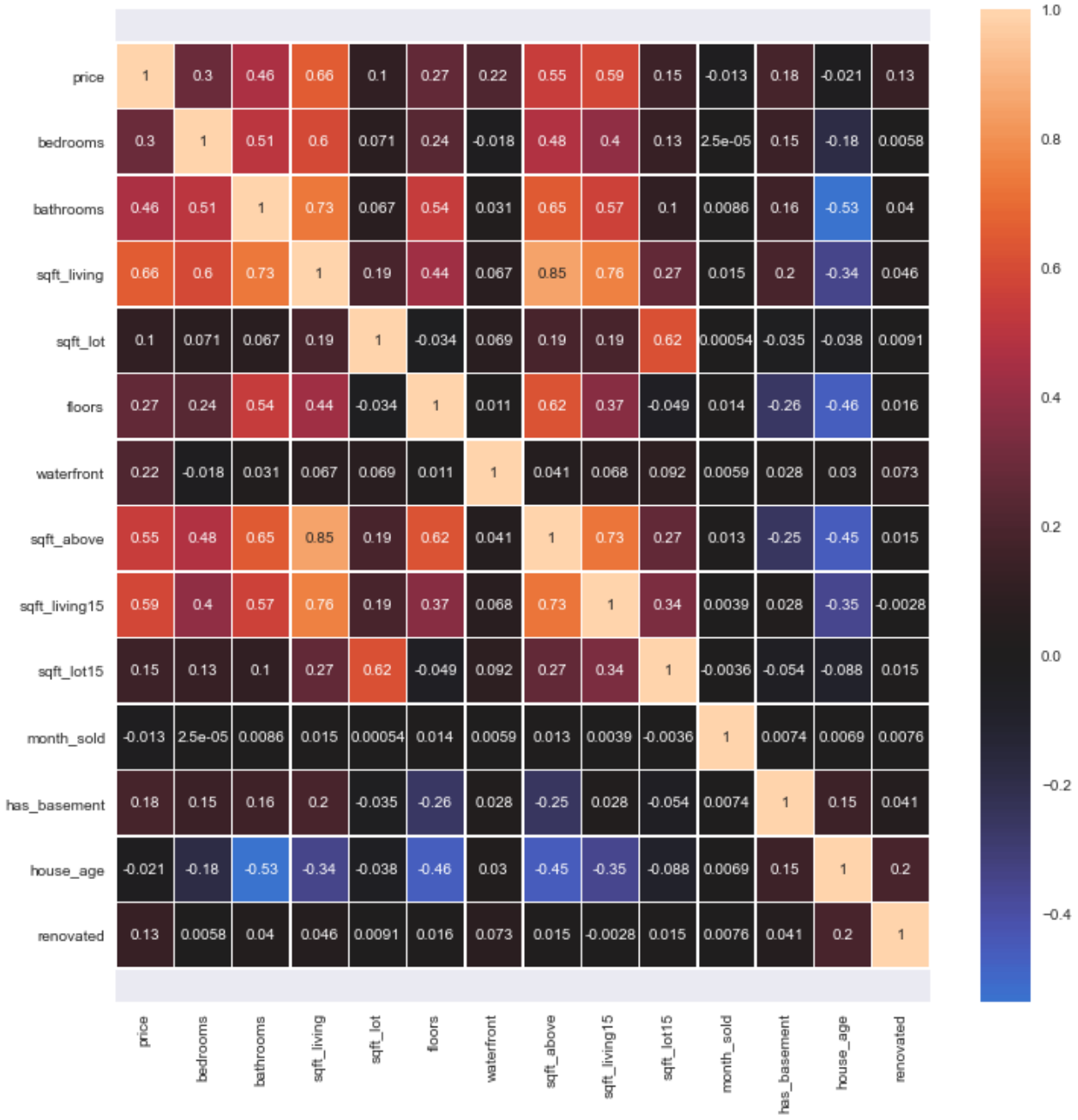
```
    for i in range(len(df.columns)):
        vif = variance_inflation_factor(df.values, i)
        v = df.columns[i]
        vif_dct[v] = vif

    return vif_dct
```

In [68]:
```
create_vif_dct(df)
```

Out[68]:
```
{'const': 55.022724198167936,
 'price': 2.2985408828428797,
 'bedrooms': 1.6802078637693854,
 'bathrooms': 3.2383158695835093,
 'sqft_living': 14.688164021777045,
 'sqft_lot': 1.6277215158469391,
 'floors': 2.230002774313753,
 'waterfront': 1.078430912077137,
 'sqft_above': 14.714453503122858,
 'sqft_living15': 2.8236325399180373,
 'sqft_lot15': 1.8589869591929093,
 'month_sold': 1.0026896051947436,
 'has_basement': 3.3491766847912894,
 'house_age': 1.9451900845210486,
 'renovated': 1.086912571340142}
```

In [69]:
```
#check for multicollinearity visually
plt.figure(figsize=(12, 12))
ax = sns.heatmap(df.corr(), center=0, linewidths=.5, annot=True)
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
plt.show()
```



In [70]:
```
#Dropping most multicollinearity columns.
df.drop(columns=['sqft_above','bathrooms'],inplace=True)
```

In [71]:
```
cont_features =['sqft_living','sqft_lot','sqft_living15','sqft_lot15']
```

In [72]:
```
X_train_cont = X_train[cont_features]
X_test_cont = X_test[cont_features]
```

In [73]:
```
linreg = LinearRegression()
linreg.fit(X_train_cont,y_train)
```

Out[73]:
```
LinearRegression()
```

In [74]:
```
linreg.score(X_train_cont,y_train)
```

Out[74]:
```
0.45462587938469934
```

Scaling Data

In [75]:
```
ss = StandardScaler()
X_train_cont_scaled = ss.fit_transform(X_train_cont)
X_test_cont_scaled = ss.transform(X_test_cont)
```

In [76]:
```
linreg_norm = LinearRegression()
linreg_norm.fit(X_train_cont_scaled,y_train)
```

Out[76]:
```
LinearRegression()
```

In [77]:
```
linreg_norm.score(X_train_cont_scaled,y_train)
```

Out[77]:
```
0.45462587938469934
```

In [78]:
```
print('Train Continuous MSE:',mean_squared_error(y_train, linreg_norm.predict(X_train_c
```

```
Train Continuous MSE: 52600594372.51586
```

In [79]:
```
X_train_cont
```

Out[79]:

	sqft_living	sqft_lot	sqft_living15	sqft_lot15
1153	700	3000	1560	4500
17534	2240	8162	1550	8163
17123	3230	7800	3030	6600
9503	5844	10766	3413	10766
12355	2390	8102	2310	8606
...	...	...	...	...
9693	1210	7140	1150	7376
14988	2970	11985	2990	12049
840	1660	10725	1340	9023
13822	1330	7125	1570	7350
1946	1430	8775	1390	7800

15948 rows × 4 columns

In [80]:
```
X_train_cont_scaled = pd.DataFrame(X_train_cont_scaled , columns=X_train_cont.columns)
X_test_cont_scaled = pd.DataFrame(X_test_cont_scaled , columns=X_test_cont.columns)
```

In [81]:
```
X_train_cat.reset_index(inplace=True)
X_test_cat.reset_index(inplace=True)
```

In [82]:
```
X_train_cat
```

Out[82]:

	index	waterfront_1	bedrooms_2	bedrooms_3	bedrooms_4	bedrooms_5	bedrooms_6	bedroon

	index	waterfront_1	bedrooms_2	bedrooms_3	bedrooms_4	bedrooms_5	bedrooms_6	bedroon
0	1153	0	1	0	0	0	0	
1	17534	0	0	0	1	0	0	
2	17123	0	0	1	0	0	0	
3	9503	0	0	0	1	0	0	
4	12355	0	0	1	0	0	0	
...	...	...	...	...	...	...	...	
15943	9693	0	1	0	0	0	0	
15944	14988	0	0	1	0	0	0	
15945	840	0	0	0	1	0	0	
15946	13822	0	0	1	0	0	0	
15947	1946	0	0	0	1	0	0	

15948 rows × 228 columns

In [83]:
```
X_train_cat.drop(columns=['index'],inplace=True)
X_test_cat.drop(columns=['index'],inplace=True)
```

In [84]:
```
X_train_all = pd.concat([X_train_cont_scaled, X_train_cat], axis=1)
X_test_all = pd.concat([X_test_cont_scaled, X_test_cat], axis=1)
```

In [85]:
```
X_train_cont_scaled
```

Out[85]:

	sqft_living	sqft_lot	sqft_living15	sqft_lot15
0	-1.617424	-0.484255	-0.615062	-0.637948
1	0.272458	-0.149943	-0.630398	-0.126578
2	1.487382	-0.173387	1.639302	-0.344779
3	4.695273	0.018703	2.226664	0.236812
4	0.456538	-0.153828	0.535124	-0.064733
...	...	...	...	...
15943	-0.991554	-0.216131	-1.243830	-0.236446
15944	1.168311	0.097650	1.577959	0.415924
15945	-0.439316	0.016048	-0.952450	-0.006518
15946	-0.844290	-0.217103	-0.599726	-0.240076
15947	-0.721571	-0.110242	-0.875771	-0.177254

15948 rows × 4 columns

In [86]:
```
X_train_cat
```

Out[86]:

	waterfront_1	bedrooms_2	bedrooms_3	bedrooms_4	bedrooms_5	bedrooms_6	bedrooms_7	be
0	0	1	0	0	0	0	0	
1	0	0	0	1	0	0	0	
2	0	0	1	0	0	0	0	
3	0	0	0	1	0	0	0	
4	0	0	1	0	0	0	0	
...	...	...	...	...	...	...	...	
15943	0	1	0	0	0	0	0	
15944	0	0	1	0	0	0	0	
15945	0	0	0	1	0	0	0	
15946	0	0	1	0	0	0	0	
15947	0	0	0	1	0	0	0	

15948 rows × 227 columns

In [87]:
```
lin = LinearRegression()
lin.fit(X_train_all,y_train)
```

Out[87]:
```
LinearRegression()
```

Functions for Modeling

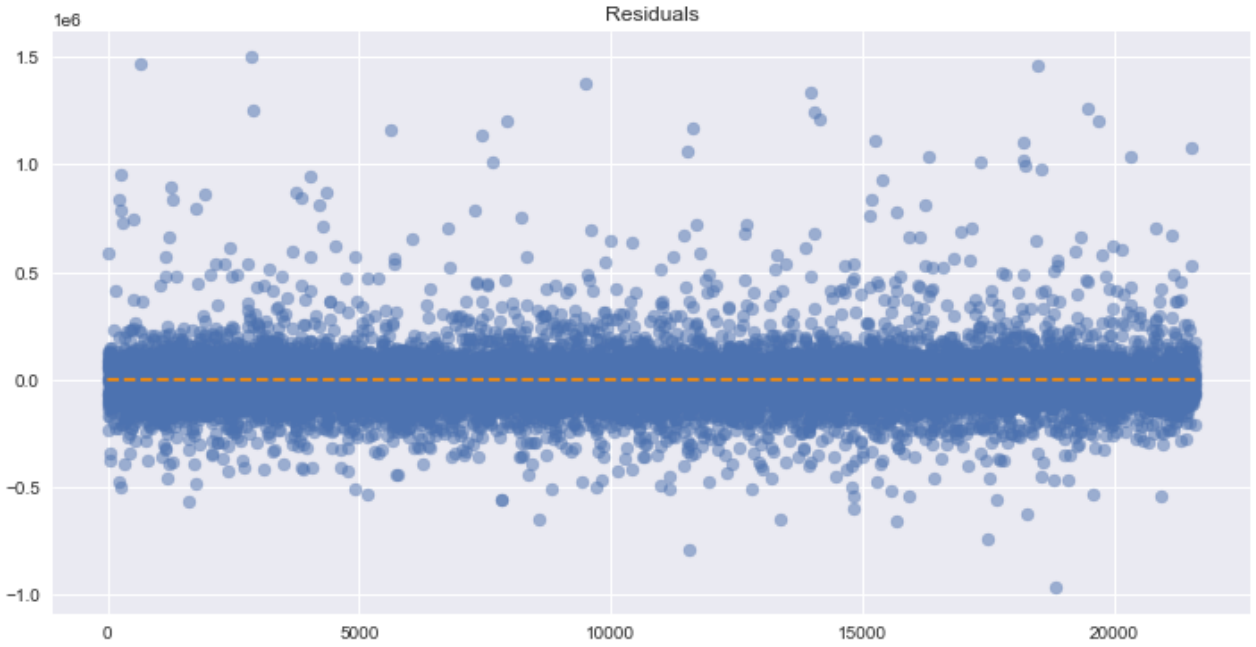
In [88]:
```
def calculate_residuals(model, features, label):
#Caluclates residuals and predict features
    predictions = model.predict(features)
    df_results = pd.DataFrame({'Actual': label, 'Predicted': predictions})
    df_results['Residuals'] = abs(df_results['Actual']) - abs(df_results['Predicted'])

    return df_results
```

In [89]:
```
#Creating function to visualize homoscedasticity.
def visualize_homoscedasticity(model,features, label):
    df_results = calculate_residuals(model, features, label)

    plt.subplots(figsize=(12, 6))
    ax = plt.subplot(111)  # To remove spines
    plt.scatter(x=df_results.index, y=df_results.Residuals, alpha=0.5)
    plt.plot(np.repeat(0, df_results.index.max()), color='darkorange', linestyle='--')
    ax.spines['right'].set_visible(False)  # Removing the right spine
    ax.spines['top'].set_visible(False)  # Removing the top spine
    plt.title('Residuals')
    plt.show()
```

In [90]:
```
#Looking for train data homoscedasticity.
visualize_homoscedasticity(lin, X_train_all, y_train)
```



In [91]:
```
X_train_all
```

Out[91]:

	sqft_living	sqft_lot	sqft_living15	sqft_lot15	waterfront_1	bedrooms_2	bedrooms_3	bedroor
0	-1.617424	-0.484255	-0.615062	-0.637948	0	1	0	
1	0.272458	-0.149943	-0.630398	-0.126578	0	0	0	
2	1.487382	-0.173387	1.639302	-0.344779	0	0	1	
3	4.695273	0.018703	2.226664	0.236812	0	0	0	
4	0.456538	-0.153828	0.535124	-0.064733	0	0	1	
...	...	...	...	...	...	...	...	
15943	-0.991554	-0.216131	-1.243830	-0.236446	0	1	0	
15944	1.168311	0.097650	1.577959	0.415924	0	0	1	
15945	-0.439316	0.016048	-0.952450	-0.006518	0	0	0	
15946	-0.844290	-0.217103	-0.599726	-0.240076	0	0	1	
15947	-0.721571	-0.110242	-0.875771	-0.177254	0	0	0	

15948 rows × 231 columns

Modeling

Model 1

In [92]:
```
y_train = list(y_train)
X_train_all = sm.add_constant(X_train_all)
```

In [93]:
```
model = sm.OLS(y_train,X_train_all).fit()
model.summary()
```

Out[93]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.840
Model:	OLS	**Adj. R-squared:**	0.838
Method:	Least Squares	**F-statistic:**	357.2
Date:	Fri, 14 Jan 2022	**Prob (F-statistic):**	0.00
Time:	10:55:11	**Log-Likelihood:**	-2.0970e+05
No. Observations:	15948	**AIC:**	4.199e+05
Df Residuals:	15716	**BIC:**	4.216e+05
Df Model:	231		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	6.108e+05	3.44e+04	17.766	0.000	5.43e+05	6.78e+05
sqft_living	1.089e+05	2196.204	49.597	0.000	1.05e+05	1.13e+05
sqft_lot	9109.2623	1275.896	7.139	0.000	6608.359	1.16e+04
sqft_living15	1.587e+04	1857.717	8.540	0.000	1.22e+04	1.95e+04
sqft_lot15	-481.1765	1517.909	-0.317	0.751	-3456.452	2494.099
waterfront_1	4.483e+05	1.66e+04	27.002	0.000	4.16e+05	4.81e+05
bedrooms_2	1.807e+04	1.11e+04	1.622	0.105	-3761.428	3.99e+04
bedrooms_3	3.559e+04	1.11e+04	3.193	0.001	1.37e+04	5.74e+04
bedrooms_4	2.279e+04	1.14e+04	1.999	0.046	443.333	4.51e+04
bedrooms_5	1.138e+04	1.21e+04	0.944	0.345	-1.22e+04	3.5e+04
bedrooms_6	-1.765e+04	1.51e+04	-1.167	0.243	-4.73e+04	1.2e+04
bedrooms_7	2.464e+04	3.2e+04	0.770	0.441	-3.8e+04	8.73e+04
bedrooms_8	3912.9363	4.92e+04	0.079	0.937	-9.26e+04	1e+05
bedrooms_9	-4.761e+04	9.03e+04	-0.527	0.598	-2.25e+05	1.29e+05
grade_11 Excellent	1.956e+05	1.02e+04	19.092	0.000	1.76e+05	2.16e+05
grade_12 Luxury	4.63e+05	2.43e+04	19.087	0.000	4.15e+05	5.1e+05
grade_13 Mansion	9.387e+05	8.92e+04	10.528	0.000	7.64e+05	1.11e+06
grade_3 Poor	-1.121e+05	1.27e+05	-0.882	0.378	-3.61e+05	1.37e+05
grade_4 Low	-2.474e+05	2.86e+04	-8.641	0.000	-3.04e+05	-1.91e+05
grade_5 Fair	-2.696e+05	1.25e+04	-21.483	0.000	-2.94e+05	-2.45e+05

grade_6 Low Average	-2.685e+05	8133.094	-33.008	0.000	-2.84e+05	-2.53e+05
grade_7 Average	-2.504e+05	6724.293	-37.237	0.000	-2.64e+05	-2.37e+05
grade_8 Good	-2.141e+05	5956.979	-35.935	0.000	-2.26e+05	-2.02e+05
grade_9 Better	-1.263e+05	5707.198	-22.123	0.000	-1.37e+05	-1.15e+05
renovated_1	6.804e+04	5816.644	11.697	0.000	5.66e+04	7.94e+04
view_EXCELLENT	2.615e+05	1.21e+04	21.617	0.000	2.38e+05	2.85e+05
view_FAIR	8701.4812	9560.311	0.910	0.363	-1e+04	2.74e+04
view_GOOD	8.787e+04	8507.513	10.329	0.000	7.12e+04	1.05e+05
view_NONE	-7.201e+04	5236.301	-13.752	0.000	-8.23e+04	-6.17e+04
zipcode_98002	9121.1786	1.27e+04	0.717	0.473	-1.58e+04	3.41e+04
zipcode_98003	-840.4064	1.14e+04	-0.074	0.941	-2.31e+04	2.15e+04
zipcode_98004	7.149e+05	1.16e+04	61.859	0.000	6.92e+05	7.38e+05
zipcode_98005	3.174e+05	1.36e+04	23.263	0.000	2.91e+05	3.44e+05
zipcode_98006	2.575e+05	1.01e+04	25.371	0.000	2.38e+05	2.77e+05
zipcode_98007	2.543e+05	1.42e+04	17.894	0.000	2.26e+05	2.82e+05
zipcode_98008	2.567e+05	1.14e+04	22.507	0.000	2.34e+05	2.79e+05
zipcode_98010	8.544e+04	1.84e+04	4.648	0.000	4.94e+04	1.21e+05
zipcode_98011	1.421e+05	1.25e+04	11.354	0.000	1.18e+05	1.67e+05
zipcode_98014	1.113e+05	1.72e+04	6.466	0.000	7.76e+04	1.45e+05
zipcode_98019	9.517e+04	1.33e+04	7.141	0.000	6.9e+04	1.21e+05
zipcode_98022	-1.58e+04	1.31e+04	-1.207	0.227	-4.15e+04	9859.917
zipcode_98023	-2.124e+04	9873.336	-2.151	0.032	-4.06e+04	-1882.712
zipcode_98024	1.371e+05	2.23e+04	6.132	0.000	9.32e+04	1.81e+05
zipcode_98027	1.742e+05	1.1e+04	15.783	0.000	1.53e+05	1.96e+05
zipcode_98028	1.334e+05	1.13e+04	11.776	0.000	1.11e+05	1.56e+05
zipcode_98029	2.26e+05	1.13e+04	19.994	0.000	2.04e+05	2.48e+05
zipcode_98030	7555.4395	1.15e+04	0.657	0.511	-1.5e+04	3.01e+04
zipcode_98031	2.259e+04	1.13e+04	1.998	0.046	422.938	4.48e+04
zipcode_98032	1490.1012	1.46e+04	0.102	0.919	-2.71e+04	3.01e+04
zipcode_98033	3.519e+05	1.03e+04	34.199	0.000	3.32e+05	3.72e+05
zipcode_98034	1.993e+05	9766.305	20.406	0.000	1.8e+05	2.18e+05
zipcode_98038	3.332e+04	9871.749	3.375	0.001	1.4e+04	5.27e+04
zipcode_98039	1.111e+06	2.41e+04	46.203	0.000	1.06e+06	1.16e+06
zipcode_98040	4.828e+05	1.2e+04	40.396	0.000	4.59e+05	5.06e+05
zipcode_98042	6867.3434	9724.202	0.706	0.480	-1.22e+04	2.59e+04

zipcode_98045	1.052e+05	1.29e+04	8.145	0.000	7.99e+04	1.3e+05
zipcode_98052	2.423e+05	9724.186	24.917	0.000	2.23e+05	2.61e+05
zipcode_98053	2.28e+05	1.09e+04	20.975	0.000	2.07e+05	2.49e+05
zipcode_98055	3.882e+04	1.15e+04	3.375	0.001	1.63e+04	6.14e+04
zipcode_98056	9.256e+04	1.04e+04	8.937	0.000	7.23e+04	1.13e+05
zipcode_98058	3.423e+04	1.01e+04	3.405	0.001	1.45e+04	5.39e+04
zipcode_98059	8.456e+04	9996.503	8.459	0.000	6.5e+04	1.04e+05
zipcode_98065	9.938e+04	1.14e+04	8.693	0.000	7.7e+04	1.22e+05
zipcode_98070	-2.966e+04	1.94e+04	-1.532	0.126	-6.76e+04	8295.563
zipcode_98072	1.549e+05	1.16e+04	13.311	0.000	1.32e+05	1.78e+05
zipcode_98074	1.726e+05	1.05e+04	16.472	0.000	1.52e+05	1.93e+05
zipcode_98075	1.815e+05	1.1e+04	16.470	0.000	1.6e+05	2.03e+05
zipcode_98077	1.209e+05	1.42e+04	8.491	0.000	9.3e+04	1.49e+05
zipcode_98092	-2.608e+04	1.09e+04	-2.396	0.017	-4.74e+04	-4748.736
zipcode_98102	4.627e+05	1.71e+04	27.066	0.000	4.29e+05	4.96e+05
zipcode_98103	3.551e+05	1.07e+04	33.244	0.000	3.34e+05	3.76e+05
zipcode_98105	4.582e+05	1.28e+04	35.747	0.000	4.33e+05	4.83e+05
zipcode_98106	1.215e+05	1.11e+04	10.978	0.000	9.98e+04	1.43e+05
zipcode_98107	3.342e+05	1.33e+04	25.200	0.000	3.08e+05	3.6e+05
zipcode_98108	1.212e+05	1.31e+04	9.266	0.000	9.56e+04	1.47e+05
zipcode_98109	4.821e+05	1.69e+04	28.515	0.000	4.49e+05	5.15e+05
zipcode_98112	5.786e+05	1.25e+04	46.452	0.000	5.54e+05	6.03e+05
zipcode_98115	3.276e+05	1.01e+04	32.342	0.000	3.08e+05	3.47e+05
zipcode_98116	2.819e+05	1.15e+04	24.565	0.000	2.59e+05	3.04e+05
zipcode_98117	3.136e+05	1.02e+04	30.649	0.000	2.94e+05	3.34e+05
zipcode_98118	1.644e+05	1.01e+04	16.224	0.000	1.45e+05	1.84e+05
zipcode_98119	4.831e+05	1.43e+04	33.721	0.000	4.55e+05	5.11e+05
zipcode_98122	3.412e+05	1.19e+04	28.647	0.000	3.18e+05	3.65e+05
zipcode_98125	2.004e+05	1.06e+04	18.814	0.000	1.79e+05	2.21e+05
zipcode_98126	1.847e+05	1.1e+04	16.844	0.000	1.63e+05	2.06e+05
zipcode_98133	1.598e+05	1.02e+04	15.594	0.000	1.4e+05	1.8e+05
zipcode_98136	2.372e+05	1.19e+04	19.874	0.000	2.14e+05	2.61e+05
zipcode_98144	2.704e+05	1.13e+04	24.033	0.000	2.48e+05	2.92e+05
zipcode_98146	8.94e+04	1.14e+04	7.864	0.000	6.71e+04	1.12e+05
zipcode_98148	7.354e+04	2.05e+04	3.581	0.000	3.33e+04	1.14e+05

zipcode_98155	1.441e+05	1.03e+04	14.033	0.000	1.24e+05	1.64e+05
zipcode_98166	7.387e+04	1.17e+04	6.335	0.000	5.1e+04	9.67e+04
zipcode_98168	5.154e+04	1.17e+04	4.410	0.000	2.86e+04	7.45e+04
zipcode_98177	2.222e+05	1.2e+04	18.449	0.000	1.99e+05	2.46e+05
zipcode_98178	2.924e+04	1.19e+04	2.466	0.014	6003.452	5.25e+04
zipcode_98188	3.93e+04	1.42e+04	2.765	0.006	1.14e+04	6.72e+04
zipcode_98198	1.276e+04	1.14e+04	1.121	0.262	-9544.756	3.51e+04
zipcode_98199	3.722e+05	1.17e+04	31.883	0.000	3.49e+05	3.95e+05
has_basement_1	-2.204e+04	2641.827	-8.342	0.000	-2.72e+04	-1.69e+04
month_sold_2	2304.8183	6277.036	0.367	0.713	-9998.894	1.46e+04
month_sold_3	2.417e+04	5768.971	4.190	0.000	1.29e+04	3.55e+04
month_sold_4	3.041e+04	5606.314	5.425	0.000	1.94e+04	4.14e+04
month_sold_5	4206.6615	5571.967	0.755	0.450	-6715.035	1.51e+04
month_sold_6	-2266.7963	5632.105	-0.402	0.687	-1.33e+04	8772.777
month_sold_7	-7029.2458	5623.804	-1.250	0.211	-1.81e+04	3994.056
month_sold_8	-7405.6376	5740.258	-1.290	0.197	-1.87e+04	3845.928
month_sold_9	-1.308e+04	5818.416	-2.247	0.025	-2.45e+04	-1671.877
month_sold_10	-1.256e+04	5765.820	-2.178	0.029	-2.39e+04	-1256.798
month_sold_11	-9807.9928	6083.893	-1.612	0.107	-2.17e+04	2117.137
month_sold_12	-4030.7368	6043.477	-0.667	0.505	-1.59e+04	7815.173
condition_Fair	-1.724e+04	1.14e+04	-1.516	0.130	-3.95e+04	5053.278
condition_Good	2.391e+04	2623.650	9.114	0.000	1.88e+04	2.91e+04
condition_Poor	-8.611e+04	2.61e+04	-3.303	0.001	-1.37e+05	-3.5e+04
condition_Very Good	7.04e+04	4064.825	17.320	0.000	6.24e+04	7.84e+04
house_age_8	-6013.7168	3.13e+04	-0.192	0.847	-6.73e+04	5.53e+04
house_age_9	751.1242	3.24e+04	0.023	0.982	-6.28e+04	6.43e+04
house_age_10	-2.48e+04	3.26e+04	-0.761	0.447	-8.87e+04	3.91e+04
house_age_11	-1.527e+04	3.33e+04	-0.459	0.646	-8.05e+04	4.99e+04
house_age_12	-3.674e+04	3.3e+04	-1.112	0.266	-1.01e+05	2.8e+04
house_age_13	-3.172e+04	3.23e+04	-0.982	0.326	-9.5e+04	3.16e+04
house_age_14	-3.995e+04	3.18e+04	-1.257	0.209	-1.02e+05	2.24e+04
house_age_15	-5.058e+04	3.15e+04	-1.604	0.109	-1.12e+05	1.12e+04
house_age_16	-6.079e+04	3.14e+04	-1.934	0.053	-1.22e+05	824.738
house_age_17	-5.708e+04	3.14e+04	-1.815	0.069	-1.19e+05	4548.555
house_age_18	-5.692e+04	3.15e+04	-1.808	0.071	-1.19e+05	4775.385

house_age_19	-4.363e+04	3.14e+04	-1.388	0.165	-1.05e+05	1.8e+04
house_age_20	-5.739e+04	3.22e+04	-1.782	0.075	-1.21e+05	5735.069
house_age_21	-5.382e+04	3.18e+04	-1.691	0.091	-1.16e+05	8556.715
house_age_22	-3.752e+04	3.26e+04	-1.152	0.249	-1.01e+05	2.63e+04
house_age_23	-5.044e+04	3.21e+04	-1.571	0.116	-1.13e+05	1.25e+04
house_age_24	-6.511e+04	3.21e+04	-2.029	0.042	-1.28e+05	-2224.255
house_age_25	-6.715e+04	3.26e+04	-2.063	0.039	-1.31e+05	-3335.287
house_age_26	-2.667e+04	3.24e+04	-0.823	0.411	-9.02e+04	3.69e+04
house_age_27	-5.229e+04	3.26e+04	-1.606	0.108	-1.16e+05	1.15e+04
house_age_28	-4.813e+04	3.21e+04	-1.501	0.133	-1.11e+05	1.47e+04
house_age_29	-4.907e+04	3.24e+04	-1.514	0.130	-1.13e+05	1.45e+04
house_age_30	-6.772e+04	3.23e+04	-2.095	0.036	-1.31e+05	-4372.623
house_age_31	-5.8e+04	3.23e+04	-1.796	0.072	-1.21e+05	5290.671
house_age_32	-8.767e+04	3.17e+04	-2.762	0.006	-1.5e+05	-2.54e+04
house_age_33	-5.656e+04	3.19e+04	-1.775	0.076	-1.19e+05	5902.014
house_age_34	-6.804e+04	3.19e+04	-2.130	0.033	-1.31e+05	-5432.411
house_age_35	-6.184e+04	3.17e+04	-1.951	0.051	-1.24e+05	287.703
house_age_36	-5.572e+04	3.22e+04	-1.732	0.083	-1.19e+05	7348.331
house_age_37	-4.182e+04	3.22e+04	-1.300	0.194	-1.05e+05	2.12e+04
house_age_38	-4.412e+04	3.22e+04	-1.370	0.171	-1.07e+05	1.9e+04
house_age_39	-3.557e+04	3.24e+04	-1.100	0.272	-9.9e+04	2.78e+04
house_age_40	-2.519e+04	3.39e+04	-0.743	0.457	-9.16e+04	4.13e+04
house_age_41	-1.783e+04	3.24e+04	-0.550	0.582	-8.13e+04	4.57e+04
house_age_42	-3.318e+04	3.22e+04	-1.031	0.303	-9.63e+04	2.99e+04
house_age_43	-6.323e+04	3.17e+04	-1.997	0.046	-1.25e+05	-1166.074
house_age_44	-5.67e+04	3.16e+04	-1.796	0.072	-1.19e+05	5170.032
house_age_45	-5.73e+04	3.15e+04	-1.818	0.069	-1.19e+05	4477.153
house_age_46	-4.801e+04	3.21e+04	-1.497	0.134	-1.11e+05	1.48e+04
house_age_47	-4.285e+04	3.26e+04	-1.316	0.188	-1.07e+05	2.1e+04
house_age_48	-5.156e+04	3.28e+04	-1.571	0.116	-1.16e+05	1.28e+04
house_age_49	-3.237e+04	3.29e+04	-0.984	0.325	-9.68e+04	3.21e+04
house_age_50	-3.981e+04	3.29e+04	-1.209	0.227	-1.04e+05	2.47e+04
house_age_51	-4.71e+04	3.41e+04	-1.382	0.167	-1.14e+05	1.97e+04
house_age_52	-5.45e+04	3.32e+04	-1.644	0.100	-1.19e+05	1.05e+04
house_age_53	-2.425e+04	3.19e+04	-0.760	0.447	-8.68e+04	3.83e+04

house_age_54	-3.236e+04	3.17e+04	-1.022	0.307	-9.44e+04	2.97e+04
house_age_55	-3.934e+04	3.17e+04	-1.240	0.215	-1.02e+05	2.28e+04
house_age_56	-6.371e+04	3.21e+04	-1.984	0.047	-1.27e+05	-761.987
house_age_57	-5.61e+04	3.24e+04	-1.729	0.084	-1.2e+05	7484.398
house_age_58	-5.403e+04	3.27e+04	-1.654	0.098	-1.18e+05	1e+04
house_age_59	-4.09e+04	3.2e+04	-1.279	0.201	-1.04e+05	2.18e+04
house_age_60	-4.71e+04	3.19e+04	-1.478	0.139	-1.1e+05	1.54e+04
house_age_61	-5.118e+04	3.22e+04	-1.589	0.112	-1.14e+05	1.19e+04
house_age_62	-4.728e+04	3.21e+04	-1.473	0.141	-1.1e+05	1.56e+04
house_age_63	-4.301e+04	3.18e+04	-1.354	0.176	-1.05e+05	1.93e+04
house_age_64	-3.545e+04	3.22e+04	-1.100	0.271	-9.86e+04	2.77e+04
house_age_65	-3.563e+04	3.23e+04	-1.104	0.270	-9.89e+04	2.76e+04
house_age_66	-4.492e+04	3.24e+04	-1.385	0.166	-1.08e+05	1.87e+04
house_age_67	-5.549e+04	3.2e+04	-1.735	0.083	-1.18e+05	7187.807
house_age_68	-4.66e+04	3.18e+04	-1.464	0.143	-1.09e+05	1.58e+04
house_age_69	-6.006e+04	3.23e+04	-1.860	0.063	-1.23e+05	3242.078
house_age_70	-4.175e+04	3.22e+04	-1.295	0.195	-1.05e+05	2.14e+04
house_age_71	-2.8e+04	3.21e+04	-0.872	0.383	-9.09e+04	3.49e+04
house_age_72	-3.644e+04	3.21e+04	-1.136	0.256	-9.93e+04	2.64e+04
house_age_73	-1.488e+04	3.24e+04	-0.459	0.646	-7.84e+04	4.86e+04
house_age_74	-3.038e+04	3.23e+04	-0.941	0.347	-9.36e+04	3.29e+04
house_age_75	-3.263e+04	3.21e+04	-1.018	0.309	-9.55e+04	3.02e+04
house_age_76	-2.465e+04	3.33e+04	-0.740	0.460	-9e+04	4.07e+04
house_age_77	-1.316e+04	3.42e+04	-0.385	0.700	-8.01e+04	5.38e+04
house_age_78	-1.855e+04	3.31e+04	-0.561	0.575	-8.34e+04	4.63e+04
house_age_79	-3.184e+04	3.27e+04	-0.973	0.330	-9.6e+04	3.23e+04
house_age_80	-3.334e+04	3.23e+04	-1.033	0.302	-9.66e+04	2.99e+04
house_age_81	1785.7757	3.26e+04	0.055	0.956	-6.22e+04	6.58e+04
house_age_82	-1.316e+04	3.28e+04	-0.401	0.689	-7.75e+04	5.12e+04
house_age_83	6241.6273	3.37e+04	0.185	0.853	-5.99e+04	7.23e+04
house_age_84	-1.388e+04	3.68e+04	-0.378	0.706	-8.59e+04	5.82e+04
house_age_85	1.162e+04	3.5e+04	0.332	0.740	-5.7e+04	8.03e+04
house_age_86	-2.082e+04	3.86e+04	-0.539	0.590	-9.65e+04	5.49e+04
house_age_87	-5284.2748	4.18e+04	-0.127	0.899	-8.71e+04	7.66e+04
house_age_88	-3.686e+04	4.66e+04	-0.791	0.429	-1.28e+05	5.44e+04

house_age_89	3948.2713	4.09e+04	0.097	0.923	-7.61e+04	8.4e+04
house_age_90	-3033.6336	3.95e+04	-0.077	0.939	-8.05e+04	7.44e+04
house_age_91	-7941.1617	3.58e+04	-0.222	0.824	-7.8e+04	6.21e+04
house_age_92	-2.081e+04	3.44e+04	-0.604	0.546	-8.83e+04	4.67e+04
house_age_93	-2.687e+04	3.36e+04	-0.801	0.423	-9.26e+04	3.89e+04
house_age_94	-4032.2846	3.32e+04	-0.121	0.903	-6.92e+04	6.11e+04
house_age_95	5743.2901	3.36e+04	0.171	0.864	-6.02e+04	7.17e+04
house_age_96	2765.5200	3.26e+04	0.085	0.932	-6.11e+04	6.66e+04
house_age_97	-1.328e+04	3.27e+04	-0.406	0.685	-7.74e+04	5.09e+04
house_age_98	-1.551e+04	3.3e+04	-0.470	0.639	-8.02e+04	4.92e+04
house_age_99	-363.3423	3.45e+04	-0.011	0.992	-6.79e+04	6.72e+04
house_age_100	-3.354e+04	3.4e+04	-0.988	0.323	-1e+05	3.3e+04
house_age_101	-1485.8584	3.49e+04	-0.043	0.966	-6.99e+04	6.69e+04
house_age_102	-7523.9381	3.4e+04	-0.221	0.825	-7.41e+04	5.91e+04
house_age_103	-2.03e+04	3.42e+04	-0.593	0.553	-8.74e+04	4.68e+04
house_age_104	-2.3e+04	3.33e+04	-0.692	0.489	-8.82e+04	4.22e+04
house_age_105	-2.601e+04	3.59e+04	-0.724	0.469	-9.64e+04	4.44e+04
house_age_106	5568.9865	3.44e+04	0.162	0.872	-6.19e+04	7.31e+04
house_age_107	-2.809e+04	3.52e+04	-0.797	0.425	-9.72e+04	4.1e+04
house_age_108	-3.403e+04	3.6e+04	-0.944	0.345	-1.05e+05	3.66e+04
house_age_109	9395.3173	3.61e+04	0.260	0.795	-6.14e+04	8.02e+04
house_age_110	-1.717e+04	3.45e+04	-0.498	0.619	-8.48e+04	5.05e+04
house_age_111	7046.2782	3.48e+04	0.203	0.839	-6.11e+04	7.52e+04
house_age_112	-1.75e+04	3.3e+04	-0.530	0.596	-8.22e+04	4.72e+04
house_age_113	-1069.9838	3.41e+04	-0.031	0.975	-6.78e+04	6.57e+04
house_age_114	-4.723e+04	3.43e+04	-1.379	0.168	-1.14e+05	1.99e+04
house_age_115	2422.6771	3.55e+04	0.068	0.946	-6.71e+04	7.2e+04
house_age_116	-3.213e+04	3.43e+04	-0.938	0.348	-9.93e+04	3.5e+04
house_age_117	3.395e+04	3.54e+04	0.958	0.338	-3.55e+04	1.03e+05
house_age_118	-5.088e+04	3.8e+04	-1.340	0.180	-1.25e+05	2.36e+04
house_age_119	-3.254e+04	3.7e+04	-0.879	0.380	-1.05e+05	4e+04
house_age_120	-8.394e+04	4.09e+04	-2.052	0.040	-1.64e+05	-3745.427
house_age_121	-9.87e+04	4.18e+04	-2.362	0.018	-1.81e+05	-1.68e+04
house_age_122	-2.908e+04	3.45e+04	-0.843	0.399	-9.67e+04	3.85e+04
floors_1.5	-1.149e+04	4237.765	-2.712	0.007	-1.98e+04	-3184.427

floors_2.0	-6796.4669	3539.217	-1.920	0.055	-1.37e+04	140.806
floors_2.5	5.504e+04	1.31e+04	4.203	0.000	2.94e+04	8.07e+04

Omnibus:	8612.243	**Durbin-Watson:**	2.022
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	268182.737
Skew:	2.017	**Prob(JB):**	0.00
Kurtosis:	22.680	**Cond. No.**	579.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Model 2

In [94]:
```
#Removing features with a pvalue < 0.05.
columns = model.pvalues[model.pvalues <= 0.05]
columns.index
model = sm.OLS(y_train, X_train_all[columns.index]).fit()
model.summary()
```

Out[94]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.837
Model:	OLS	**Adj. R-squared:**	0.836
Method:	Least Squares	**F-statistic:**	822.2
Date:	Fri, 14 Jan 2022	**Prob (F-statistic):**	0.00
Time:	10:55:12	**Log-Likelihood:**	-2.0984e+05
No. Observations:	15948	**AIC:**	4.199e+05
Df Residuals:	15848	**BIC:**	4.207e+05
Df Model:	99		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	5.807e+05	8152.898	71.227	0.000	5.65e+05	5.97e+05
sqft_living	1.059e+05	1914.197	55.334	0.000	1.02e+05	1.1e+05
sqft_lot	8361.0940	1071.468	7.803	0.000	6260.895	1.05e+04
sqft_living15	1.715e+04	1813.392	9.457	0.000	1.36e+04	2.07e+04
waterfront_1	4.455e+05	1.65e+04	27.079	0.000	4.13e+05	4.78e+05
bedrooms_3	1.9e+04	2724.018	6.974	0.000	1.37e+04	2.43e+04
bedrooms_4	7964.9669	3009.063	2.647	0.008	2066.862	1.39e+04
grade_11 Excellent	1.938e+05	1.02e+04	18.937	0.000	1.74e+05	2.14e+05

grade_12 Luxury	4.63e+05	2.42e+04	19.101	0.000	4.15e+05	5.11e+05
grade_13 Mansion	9.379e+05	8.95e+04	10.481	0.000	7.62e+05	1.11e+06
grade_4 Low	-2.334e+05	2.81e+04	-8.308	0.000	-2.88e+05	-1.78e+05
grade_5 Fair	-2.558e+05	1.2e+04	-21.344	0.000	-2.79e+05	-2.32e+05
grade_6 Low Average	-2.579e+05	7691.074	-33.532	0.000	-2.73e+05	-2.43e+05
grade_7 Average	-2.487e+05	6456.084	-38.528	0.000	-2.61e+05	-2.36e+05
grade_8 Good	-2.15e+05	5845.410	-36.777	0.000	-2.26e+05	-2.04e+05
grade_9 Better	-1.258e+05	5658.940	-22.238	0.000	-1.37e+05	-1.15e+05
renovated_1	7.726e+04	5568.081	13.876	0.000	6.63e+04	8.82e+04
view_EXCELLENT	2.598e+05	1.18e+04	21.947	0.000	2.37e+05	2.83e+05
view_GOOD	8.57e+04	8123.460	10.550	0.000	6.98e+04	1.02e+05
view_NONE	-7.543e+04	4565.452	-16.522	0.000	-8.44e+04	-6.65e+04
zipcode_98004	7.13e+05	9117.684	78.204	0.000	6.95e+05	7.31e+05
zipcode_98005	3.119e+05	1.17e+04	26.655	0.000	2.89e+05	3.35e+05
zipcode_98006	2.516e+05	7380.351	34.087	0.000	2.37e+05	2.66e+05
zipcode_98007	2.46e+05	1.23e+04	19.985	0.000	2.22e+05	2.7e+05
zipcode_98008	2.523e+05	8914.846	28.304	0.000	2.35e+05	2.7e+05
zipcode_98010	8.033e+04	1.7e+04	4.733	0.000	4.71e+04	1.14e+05
zipcode_98011	1.379e+05	1.05e+04	13.151	0.000	1.17e+05	1.58e+05
zipcode_98014	1.079e+05	1.57e+04	6.859	0.000	7.71e+04	1.39e+05
zipcode_98019	8.981e+04	1.14e+04	7.880	0.000	6.75e+04	1.12e+05
zipcode_98023	-2.591e+04	7043.585	-3.679	0.000	-3.97e+04	-1.21e+04
zipcode_98024	1.38e+05	2.12e+04	6.495	0.000	9.63e+04	1.8e+05
zipcode_98027	1.714e+05	8640.079	19.841	0.000	1.54e+05	1.88e+05
zipcode_98028	1.332e+05	9007.827	14.783	0.000	1.16e+05	1.51e+05
zipcode_98029	2.209e+05	8899.504	24.824	0.000	2.03e+05	2.38e+05
zipcode_98031	1.826e+04	8961.573	2.038	0.042	695.791	3.58e+04
zipcode_98033	3.498e+05	7639.553	45.783	0.000	3.35e+05	3.65e+05
zipcode_98034	1.968e+05	6852.443	28.721	0.000	1.83e+05	2.1e+05
zipcode_98038	2.734e+04	7016.635	3.897	0.000	1.36e+04	4.11e+04
zipcode_98039	1.109e+06	2.3e+04	48.125	0.000	1.06e+06	1.15e+06
zipcode_98040	4.793e+05	9625.740	49.796	0.000	4.6e+05	4.98e+05
zipcode_98045	1.048e+05	1.09e+04	9.594	0.000	8.34e+04	1.26e+05
zipcode_98052	2.39e+05	6844.260	34.922	0.000	2.26e+05	2.52e+05
zipcode_98053	2.25e+05	8307.107	27.087	0.000	2.09e+05	2.41e+05

zipcode_98055	3.736e+04	9195.472	4.063	0.000	1.93e+04	5.54e+04
zipcode_98056	9.095e+04	7700.807	11.810	0.000	7.59e+04	1.06e+05
zipcode_98058	2.944e+04	7330.433	4.016	0.000	1.51e+04	4.38e+04
zipcode_98059	8.302e+04	7280.223	11.404	0.000	6.88e+04	9.73e+04
zipcode_98065	9.453e+04	9070.746	10.421	0.000	7.67e+04	1.12e+05
zipcode_98072	1.518e+05	9310.361	16.300	0.000	1.34e+05	1.7e+05
zipcode_98074	1.695e+05	7823.211	21.662	0.000	1.54e+05	1.85e+05
zipcode_98075	1.767e+05	8621.069	20.500	0.000	1.6e+05	1.94e+05
zipcode_98077	1.156e+05	1.23e+04	9.437	0.000	9.16e+04	1.4e+05
zipcode_98092	-2.726e+04	8443.526	-3.228	0.001	-4.38e+04	-1.07e+04
zipcode_98102	4.716e+05	1.52e+04	31.008	0.000	4.42e+05	5.01e+05
zipcode_98103	3.705e+05	7580.831	48.880	0.000	3.56e+05	3.85e+05
zipcode_98105	4.735e+05	1.03e+04	45.757	0.000	4.53e+05	4.94e+05
zipcode_98106	1.263e+05	8540.434	14.792	0.000	1.1e+05	1.43e+05
zipcode_98107	3.448e+05	1.11e+04	31.174	0.000	3.23e+05	3.66e+05
zipcode_98108	1.268e+05	1.1e+04	11.553	0.000	1.05e+05	1.48e+05
zipcode_98109	4.961e+05	1.51e+04	32.945	0.000	4.67e+05	5.26e+05
zipcode_98112	5.943e+05	9839.451	60.403	0.000	5.75e+05	6.14e+05
zipcode_98115	3.404e+05	6996.438	48.653	0.000	3.27e+05	3.54e+05
zipcode_98116	2.898e+05	8866.769	32.685	0.000	2.72e+05	3.07e+05
zipcode_98117	3.282e+05	7079.149	46.362	0.000	3.14e+05	3.42e+05
zipcode_98118	1.688e+05	7185.319	23.498	0.000	1.55e+05	1.83e+05
zipcode_98119	4.947e+05	1.21e+04	40.853	0.000	4.71e+05	5.18e+05
zipcode_98122	3.473e+05	9242.299	37.576	0.000	3.29e+05	3.65e+05
zipcode_98125	2.053e+05	7849.817	26.150	0.000	1.9e+05	2.21e+05
zipcode_98126	1.967e+05	8195.440	24.000	0.000	1.81e+05	2.13e+05
zipcode_98133	1.621e+05	7391.798	21.924	0.000	1.48e+05	1.77e+05
zipcode_98136	2.458e+05	9495.793	25.885	0.000	2.27e+05	2.64e+05
zipcode_98144	2.771e+05	8561.756	32.365	0.000	2.6e+05	2.94e+05
zipcode_98146	9.42e+04	8858.860	10.634	0.000	7.68e+04	1.12e+05
zipcode_98148	6.979e+04	1.92e+04	3.630	0.000	3.21e+04	1.07e+05
zipcode_98155	1.442e+05	7429.868	19.406	0.000	1.3e+05	1.59e+05
zipcode_98166	7.315e+04	9225.847	7.928	0.000	5.51e+04	9.12e+04
zipcode_98168	5.574e+04	9305.991	5.989	0.000	3.75e+04	7.4e+04
zipcode_98177	2.249e+05	9693.007	23.200	0.000	2.06e+05	2.44e+05

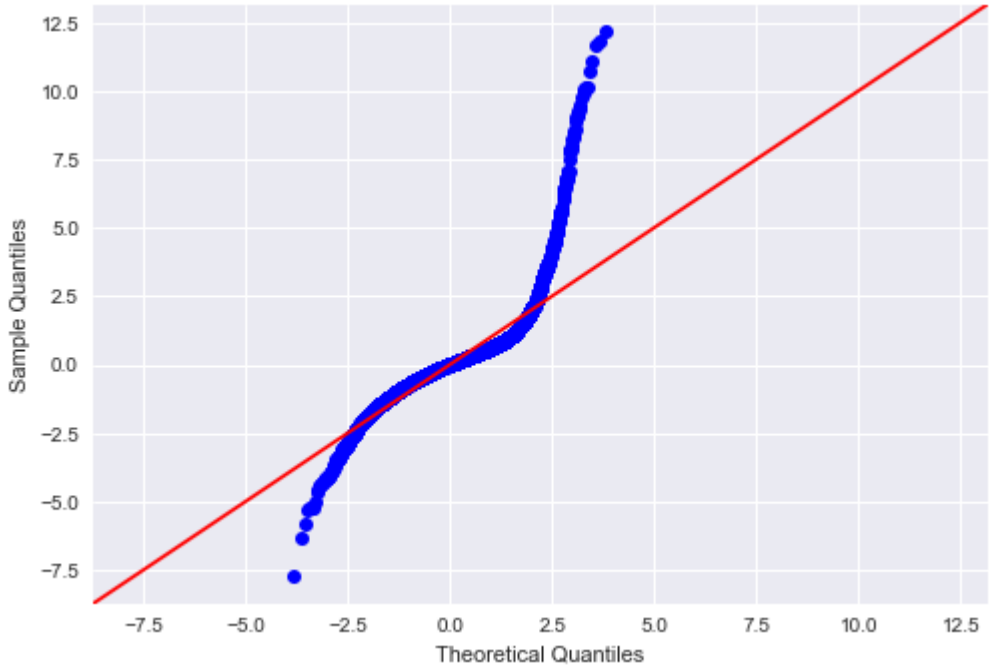
zipcode_98178	3.215e+04	9404.058	3.419	0.001	1.37e+04	5.06e+04
zipcode_98188	3.817e+04	1.24e+04	3.081	0.002	1.39e+04	6.25e+04
zipcode_98199	3.845e+05	8996.877	42.734	0.000	3.67e+05	4.02e+05
has_basement_1	-2.015e+04	2403.686	-8.384	0.000	-2.49e+04	-1.54e+04
month_sold_3	2.765e+04	3614.524	7.650	0.000	2.06e+04	3.47e+04
month_sold_4	3.327e+04	3336.841	9.972	0.000	2.67e+04	3.98e+04
month_sold_9	-9449.7625	3687.610	-2.563	0.010	-1.67e+04	-2221.628
month_sold_10	-1.013e+04	3597.773	-2.814	0.005	-1.72e+04	-3073.515
condition_Good	2.725e+04	2442.754	11.156	0.000	2.25e+04	3.2e+04
condition_Poor	-8.339e+04	2.6e+04	-3.205	0.001	-1.34e+05	-3.24e+04
condition_Very Good	7.506e+04	3875.882	19.367	0.000	6.75e+04	8.27e+04
house_age_24	-2.317e+04	9841.955	-2.355	0.019	-4.25e+04	-3882.686
house_age_25	-2.429e+04	1.13e+04	-2.155	0.031	-4.64e+04	-2200.172
house_age_30	-2.634e+04	1.05e+04	-2.498	0.012	-4.7e+04	-5675.128
house_age_32	-4.504e+04	8592.573	-5.242	0.000	-6.19e+04	-2.82e+04
house_age_34	-2.517e+04	9246.638	-2.723	0.006	-4.33e+04	-7050.143
house_age_43	-2.117e+04	7941.700	-2.666	0.008	-3.67e+04	-5603.878
house_age_56	-2.149e+04	9298.159	-2.311	0.021	-3.97e+04	-3259.946
house_age_120	-5.897e+04	2.72e+04	-2.168	0.030	-1.12e+05	-5661.207
house_age_121	-7.954e+04	2.84e+04	-2.803	0.005	-1.35e+05	-2.39e+04
floors_1.5	659.4987	3724.783	0.177	0.859	-6641.499	7960.496
floors_2.5	6.273e+04	1.28e+04	4.916	0.000	3.77e+04	8.77e+04

Omnibus:	8669.329	**Durbin-Watson:**	2.021
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	274540.586
Skew:	2.032	**Prob(JB):**	0.00
Kurtosis:	22.916	**Cond. No.**	150.

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [95]:
```
#checking normality
sm.graphics.qqplot(model.resid, dist=stats.norm, line='45', fit=True);
```



Not quite the model we want, we'll try logging the values.

Model 3

In [96]:
```
y_train = np.log(y_train)
y_test = np.log(y_test)
```

In [97]:
```
model= sm.OLS(y_train, X_train_all).fit()
model.summary()
```

Out[97]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.882
Model:	OLS	**Adj. R-squared:**	0.880
Method:	Least Squares	**F-statistic:**	508.6
Date:	Fri, 14 Jan 2022	**Prob (F-statistic):**	0.00
Time:	10:55:13	**Log-Likelihood:**	5209.6
No. Observations:	15948	**AIC:**	-9955.
Df Residuals:	15716	**BIC:**	-8174.
Df Model:	231		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	12.7728	0.048	264.399	0.000	12.678	12.868
sqft_living	0.1674	0.003	54.242	0.000	0.161	0.173
sqft_lot	0.0227	0.002	12.666	0.000	0.019	0.026
sqft_living15	0.0481	0.003	18.411	0.000	0.043	0.053

sqft_lot15	0.0048	0.002	2.255	0.024	0.001	0.009
waterfront_1	0.4332	0.023	18.568	0.000	0.387	0.479
bedrooms_2	0.0556	0.016	3.554	0.000	0.025	0.086
bedrooms_3	0.1019	0.016	6.507	0.000	0.071	0.133
bedrooms_4	0.1047	0.016	6.536	0.000	0.073	0.136
bedrooms_5	0.0725	0.017	4.283	0.000	0.039	0.106
bedrooms_6	0.0579	0.021	2.725	0.006	0.016	0.100
bedrooms_7	-0.0058	0.045	-0.130	0.897	-0.094	0.082
bedrooms_8	-0.0136	0.069	-0.197	0.844	-0.149	0.122
bedrooms_9	0.0553	0.127	0.436	0.663	-0.193	0.304
grade_11 Excellent	0.0634	0.014	4.406	0.000	0.035	0.092
grade_12 Luxury	0.0826	0.034	2.423	0.015	0.016	0.149
grade_13 Mansion	0.4721	0.125	3.768	0.000	0.227	0.718
grade_3 Poor	0.0086	0.179	0.048	0.962	-0.342	0.359
grade_4 Low	-0.6009	0.040	-14.937	0.000	-0.680	-0.522
grade_5 Fair	-0.4856	0.018	-27.540	0.000	-0.520	-0.451
grade_6 Low Average	-0.3742	0.011	-32.749	0.000	-0.397	-0.352
grade_7 Average	-0.2477	0.009	-26.221	0.000	-0.266	-0.229
grade_8 Good	-0.1456	0.008	-17.400	0.000	-0.162	-0.129
grade_9 Better	-0.0513	0.008	-6.400	0.000	-0.067	-0.036
renovated_1	0.0839	0.008	10.269	0.000	0.068	0.100
view_EXCELLENT	0.2035	0.017	11.971	0.000	0.170	0.237
view_FAIR	0.0052	0.013	0.384	0.701	-0.021	0.031
view_GOOD	0.0667	0.012	5.581	0.000	0.043	0.090
view_NONE	-0.1132	0.007	-15.386	0.000	-0.128	-0.099
zipcode_98002	-0.0252	0.018	-1.407	0.159	-0.060	0.010
zipcode_98003	0.0177	0.016	1.104	0.270	-0.014	0.049
zipcode_98004	1.1001	0.016	67.748	0.000	1.068	1.132
zipcode_98005	0.7423	0.019	38.727	0.000	0.705	0.780
zipcode_98006	0.6343	0.014	44.481	0.000	0.606	0.662
zipcode_98007	0.6643	0.020	33.262	0.000	0.625	0.703
zipcode_98008	0.6606	0.016	41.220	0.000	0.629	0.692
zipcode_98010	0.2762	0.026	10.693	0.000	0.226	0.327
zipcode_98011	0.4532	0.018	25.765	0.000	0.419	0.488
zipcode_98014	0.3000	0.024	12.403	0.000	0.253	0.347

zipcode_98019	0.3104	0.019	16.580	0.000	0.274	0.347
zipcode_98022	-0.0117	0.018	-0.638	0.523	-0.048	0.024
zipcode_98023	-0.0215	0.014	-1.548	0.122	-0.049	0.006
zipcode_98024	0.4114	0.031	13.102	0.000	0.350	0.473
zipcode_98027	0.5226	0.016	33.704	0.000	0.492	0.553
zipcode_98028	0.4089	0.016	25.685	0.000	0.378	0.440
zipcode_98029	0.5865	0.016	36.925	0.000	0.555	0.618
zipcode_98030	0.0442	0.016	2.734	0.006	0.013	0.076
zipcode_98031	0.0768	0.016	4.833	0.000	0.046	0.108
zipcode_98032	-0.0284	0.021	-1.387	0.165	-0.069	0.012
zipcode_98033	0.7727	0.014	53.452	0.000	0.744	0.801
zipcode_98034	0.5464	0.014	39.815	0.000	0.519	0.573
zipcode_98038	0.1411	0.014	10.174	0.000	0.114	0.168
zipcode_98039	1.2676	0.034	37.505	0.000	1.201	1.334
zipcode_98040	0.8784	0.017	52.303	0.000	0.845	0.911
zipcode_98042	0.0478	0.014	3.500	0.000	0.021	0.075
zipcode_98045	0.3151	0.018	17.372	0.000	0.280	0.351
zipcode_98052	0.6400	0.014	46.838	0.000	0.613	0.667
zipcode_98053	0.5873	0.015	38.456	0.000	0.557	0.617
zipcode_98055	0.1249	0.016	7.724	0.000	0.093	0.157
zipcode_98056	0.3125	0.015	21.477	0.000	0.284	0.341
zipcode_98058	0.1622	0.014	11.481	0.000	0.134	0.190
zipcode_98059	0.3284	0.014	23.380	0.000	0.301	0.356
zipcode_98065	0.3803	0.016	23.677	0.000	0.349	0.412
zipcode_98070	0.2669	0.027	9.807	0.000	0.214	0.320
zipcode_98072	0.4705	0.016	28.774	0.000	0.438	0.503
zipcode_98074	0.5311	0.015	36.079	0.000	0.502	0.560
zipcode_98075	0.5488	0.015	35.448	0.000	0.518	0.579
zipcode_98077	0.4369	0.020	21.828	0.000	0.398	0.476
zipcode_98092	-0.0140	0.015	-0.916	0.360	-0.044	0.016
zipcode_98102	0.9195	0.024	38.280	0.000	0.872	0.967
zipcode_98103	0.8194	0.015	54.600	0.000	0.790	0.849
zipcode_98105	0.8965	0.018	49.772	0.000	0.861	0.932
zipcode_98106	0.3151	0.016	20.256	0.000	0.285	0.346
zipcode_98107	0.8033	0.019	43.102	0.000	0.767	0.840

zipcode_98108	0.3381	0.018	18.398	0.000	0.302	0.374
zipcode_98109	0.9332	0.024	39.284	0.000	0.887	0.980
zipcode_98112	0.9930	0.018	56.737	0.000	0.959	1.027
zipcode_98115	0.7738	0.014	54.373	0.000	0.746	0.802
zipcode_98116	0.7114	0.016	44.120	0.000	0.680	0.743
zipcode_98117	0.7740	0.014	53.835	0.000	0.746	0.802
zipcode_98118	0.4422	0.014	31.063	0.000	0.414	0.470
zipcode_98119	0.9333	0.020	46.367	0.000	0.894	0.973
zipcode_98122	0.7690	0.017	45.953	0.000	0.736	0.802
zipcode_98125	0.5619	0.015	37.551	0.000	0.533	0.591
zipcode_98126	0.5108	0.015	33.156	0.000	0.481	0.541
zipcode_98133	0.4472	0.014	31.050	0.000	0.419	0.475
zipcode_98136	0.6331	0.017	37.751	0.000	0.600	0.666
zipcode_98144	0.6197	0.016	39.199	0.000	0.589	0.651
zipcode_98146	0.2524	0.016	15.804	0.000	0.221	0.284
zipcode_98148	0.1666	0.029	5.776	0.000	0.110	0.223
zipcode_98155	0.4161	0.014	28.833	0.000	0.388	0.444
zipcode_98166	0.2966	0.016	18.102	0.000	0.264	0.329
zipcode_98168	0.0722	0.016	4.394	0.000	0.040	0.104
zipcode_98177	0.5804	0.017	34.296	0.000	0.547	0.614
zipcode_98178	0.1255	0.017	7.533	0.000	0.093	0.158
zipcode_98188	0.1008	0.020	5.048	0.000	0.062	0.140
zipcode_98198	0.0664	0.016	4.151	0.000	0.035	0.098
zipcode_98199	0.8167	0.016	49.798	0.000	0.785	0.849
has_basement_1	-0.0202	0.004	-5.449	0.000	-0.028	-0.013
month_sold_2	0.0197	0.009	2.234	0.025	0.002	0.037
month_sold_3	0.0523	0.008	6.453	0.000	0.036	0.068
month_sold_4	0.0699	0.008	8.875	0.000	0.054	0.085
month_sold_5	0.0123	0.008	1.573	0.116	-0.003	0.028
month_sold_6	0.0044	0.008	0.554	0.580	-0.011	0.020
month_sold_7	-0.0046	0.008	-0.588	0.557	-0.020	0.011
month_sold_8	3.532e-05	0.008	0.004	0.997	-0.016	0.016
month_sold_9	-0.0125	0.008	-1.532	0.126	-0.029	0.004
month_sold_10	-0.0122	0.008	-1.507	0.132	-0.028	0.004
month_sold_11	-0.0124	0.009	-1.453	0.146	-0.029	0.004

month_sold_12	0.0006	0.008	0.068	0.946	-0.016	0.017
condition_Fair	-0.1521	0.016	-9.519	0.000	-0.183	-0.121
condition_Good	0.0520	0.004	14.112	0.000	0.045	0.059
condition_Poor	-0.2444	0.037	-6.672	0.000	-0.316	-0.173
condition_Very Good	0.1152	0.006	20.168	0.000	0.104	0.126
house_age_8	0.0491	0.044	1.117	0.264	-0.037	0.135
house_age_9	0.0626	0.046	1.375	0.169	-0.027	0.152
house_age_10	0.0366	0.046	0.798	0.425	-0.053	0.126
house_age_11	0.0287	0.047	0.614	0.539	-0.063	0.120
house_age_12	0.0128	0.046	0.275	0.783	-0.078	0.104
house_age_13	-0.0051	0.045	-0.113	0.910	-0.094	0.084
house_age_14	-0.0195	0.045	-0.436	0.663	-0.107	0.068
house_age_15	-0.0475	0.044	-1.073	0.283	-0.134	0.039
house_age_16	-0.0337	0.044	-0.764	0.445	-0.120	0.053
house_age_17	-0.0368	0.044	-0.832	0.405	-0.123	0.050
house_age_18	-0.0302	0.044	-0.682	0.495	-0.117	0.057
house_age_19	-0.0131	0.044	-0.296	0.767	-0.100	0.074
house_age_20	-0.0128	0.045	-0.283	0.777	-0.101	0.076
house_age_21	-0.0170	0.045	-0.380	0.704	-0.105	0.071
house_age_22	-0.0031	0.046	-0.067	0.947	-0.093	0.087
house_age_23	-0.0102	0.045	-0.226	0.821	-0.099	0.078
house_age_24	-0.0223	0.045	-0.494	0.622	-0.111	0.066
house_age_25	-0.0361	0.046	-0.790	0.430	-0.126	0.054
house_age_26	0.0079	0.046	0.174	0.862	-0.081	0.097
house_age_27	-0.0143	0.046	-0.313	0.754	-0.104	0.075
house_age_28	-0.0143	0.045	-0.318	0.750	-0.103	0.074
house_age_29	-0.0111	0.046	-0.244	0.807	-0.100	0.078
house_age_30	-0.0366	0.045	-0.807	0.420	-0.126	0.052
house_age_31	-0.0175	0.045	-0.386	0.700	-0.106	0.071
house_age_32	-0.0410	0.045	-0.920	0.358	-0.128	0.046
house_age_33	-0.0204	0.045	-0.456	0.649	-0.108	0.067
house_age_34	-0.0396	0.045	-0.883	0.377	-0.128	0.048
house_age_35	-0.0416	0.045	-0.935	0.350	-0.129	0.046
house_age_36	-0.0397	0.045	-0.877	0.380	-0.128	0.049
house_age_37	-0.0295	0.045	-0.654	0.513	-0.118	0.059

house_age_38	-0.0353	0.045	-0.780	0.435	-0.124	0.053
house_age_39	-0.0309	0.045	-0.679	0.497	-0.120	0.058
house_age_40	-0.0371	0.048	-0.779	0.436	-0.130	0.056
house_age_41	-0.0256	0.046	-0.561	0.575	-0.115	0.064
house_age_42	-0.0591	0.045	-1.308	0.191	-0.148	0.030
house_age_43	-0.0916	0.044	-2.059	0.040	-0.179	-0.004
house_age_44	-0.0829	0.044	-1.868	0.062	-0.170	0.004
house_age_45	-0.0758	0.044	-1.713	0.087	-0.163	0.011
house_age_46	-0.0630	0.045	-1.399	0.162	-0.151	0.025
house_age_47	-0.0606	0.046	-1.325	0.185	-0.150	0.029
house_age_48	-0.0659	0.046	-1.429	0.153	-0.156	0.025
house_age_49	-0.0648	0.046	-1.402	0.161	-0.155	0.026
house_age_50	-0.0699	0.046	-1.510	0.131	-0.161	0.021
house_age_51	-0.0935	0.048	-1.954	0.051	-0.187	0.000
house_age_52	-0.0935	0.047	-2.007	0.045	-0.185	-0.002
house_age_53	-0.0842	0.045	-1.878	0.060	-0.172	0.004
house_age_54	-0.0545	0.044	-1.225	0.220	-0.142	0.033
house_age_55	-0.0564	0.045	-1.265	0.206	-0.144	0.031
house_age_56	-0.0941	0.045	-2.085	0.037	-0.183	-0.006
house_age_57	-0.0823	0.046	-1.805	0.071	-0.172	0.007
house_age_58	-0.0788	0.046	-1.717	0.086	-0.169	0.011
house_age_59	-0.0653	0.045	-1.454	0.146	-0.153	0.023
house_age_60	-0.0770	0.045	-1.720	0.085	-0.165	0.011
house_age_61	-0.0577	0.045	-1.275	0.202	-0.146	0.031
house_age_62	-0.0689	0.045	-1.528	0.127	-0.157	0.019
house_age_63	-0.0757	0.045	-1.696	0.090	-0.163	0.012
house_age_64	-0.0482	0.045	-1.065	0.287	-0.137	0.041
house_age_65	-0.0702	0.045	-1.549	0.121	-0.159	0.019
house_age_66	-0.0415	0.046	-0.911	0.363	-0.131	0.048
house_age_67	-0.0700	0.045	-1.557	0.119	-0.158	0.018
house_age_68	-0.0533	0.045	-1.191	0.234	-0.141	0.034
house_age_69	-0.0589	0.045	-1.298	0.194	-0.148	0.030
house_age_70	-0.0227	0.045	-0.501	0.616	-0.111	0.066
house_age_71	-0.0386	0.045	-0.856	0.392	-0.127	0.050
house_age_72	-0.0264	0.045	-0.586	0.558	-0.115	0.062

house_age_73	-0.0061	0.046	-0.133	0.894	-0.095	0.083
house_age_74	-0.0027	0.045	-0.060	0.952	-0.092	0.086
house_age_75	-0.0222	0.045	-0.493	0.622	-0.110	0.066
house_age_76	0.0325	0.047	0.695	0.487	-0.059	0.124
house_age_77	0.0192	0.048	0.400	0.689	-0.075	0.113
house_age_78	-0.0172	0.046	-0.369	0.712	-0.108	0.074
house_age_79	-0.0314	0.046	-0.683	0.495	-0.121	0.059
house_age_80	-0.0277	0.045	-0.610	0.542	-0.117	0.061
house_age_81	0.0226	0.046	0.493	0.622	-0.067	0.112
house_age_82	0.0073	0.046	0.159	0.874	-0.083	0.098
house_age_83	0.0281	0.047	0.593	0.553	-0.065	0.121
house_age_84	0.0435	0.052	0.843	0.399	-0.058	0.145
house_age_85	0.0116	0.049	0.235	0.814	-0.085	0.108
house_age_86	0.0485	0.054	0.894	0.371	-0.058	0.155
house_age_87	-0.0226	0.059	-0.385	0.700	-0.138	0.092
house_age_88	0.0221	0.065	0.338	0.735	-0.106	0.150
house_age_89	0.0011	0.057	0.020	0.984	-0.111	0.114
house_age_90	0.0569	0.055	1.025	0.305	-0.052	0.166
house_age_91	0.0306	0.050	0.610	0.542	-0.068	0.129
house_age_92	0.0097	0.048	0.201	0.841	-0.085	0.105
house_age_93	0.0142	0.047	0.302	0.763	-0.078	0.107
house_age_94	0.0115	0.047	0.246	0.805	-0.080	0.103
house_age_95	0.0469	0.047	0.993	0.321	-0.046	0.140
house_age_96	0.0446	0.046	0.975	0.330	-0.045	0.134
house_age_97	0.0381	0.046	0.828	0.408	-0.052	0.128
house_age_98	0.0503	0.046	1.085	0.278	-0.041	0.141
house_age_99	0.0383	0.048	0.791	0.429	-0.057	0.133
house_age_100	0.0014	0.048	0.029	0.977	-0.092	0.095
house_age_101	0.0421	0.049	0.859	0.390	-0.054	0.138
house_age_102	0.0324	0.048	0.680	0.497	-0.061	0.126
house_age_103	0.0498	0.048	1.035	0.301	-0.044	0.144
house_age_104	-0.0115	0.047	-0.247	0.805	-0.103	0.080
house_age_105	0.0084	0.050	0.167	0.867	-0.090	0.107
house_age_106	0.0245	0.048	0.507	0.612	-0.070	0.119
house_age_107	0.0166	0.050	0.336	0.737	-0.080	0.114

house_age_108	-0.0064	0.051	-0.127	0.899	-0.106	0.093
house_age_109	0.0200	0.051	0.395	0.693	-0.079	0.120
house_age_110	-0.0014	0.048	-0.028	0.978	-0.096	0.094
house_age_111	0.0613	0.049	1.254	0.210	-0.035	0.157
house_age_112	0.0284	0.046	0.612	0.541	-0.063	0.119
house_age_113	0.0406	0.048	0.848	0.396	-0.053	0.134
house_age_114	-0.0320	0.048	-0.664	0.507	-0.126	0.062
house_age_115	0.0678	0.050	1.360	0.174	-0.030	0.166
house_age_116	0.0330	0.048	0.685	0.493	-0.061	0.127
house_age_117	0.0784	0.050	1.574	0.115	-0.019	0.176
house_age_118	0.0067	0.053	0.125	0.900	-0.098	0.111
house_age_119	-0.0062	0.052	-0.119	0.905	-0.108	0.096
house_age_120	-0.0258	0.057	-0.448	0.654	-0.138	0.087
house_age_121	-0.0438	0.059	-0.745	0.456	-0.159	0.071
house_age_122	0.0093	0.048	0.192	0.848	-0.086	0.104
floors_1.5	-0.0061	0.006	-1.026	0.305	-0.018	0.006
floors_2.0	-0.0151	0.005	-3.027	0.002	-0.025	-0.005
floors_2.5	-0.0051	0.018	-0.275	0.783	-0.041	0.031

Omnibus:	1364.945	**Durbin-Watson:**	2.012
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	6451.735
Skew:	-0.290	**Prob(JB):**	0.00
Kurtosis:	6.061	**Cond. No.**	579.

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Model 4

In [98]:
```
columns = model.pvalues[model.pvalues <= 0.05]
columns.index
model = sm.OLS(y_train, X_train_all[columns.index]).fit()
model.summary()
```

Out[98]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.878
Model:	OLS	**Adj. R-squared:**	0.877
Method:	Least Squares	**F-statistic:**	1164.

Date:	Fri, 14 Jan 2022	**Prob (F-statistic):**	0.00
Time:	10:55:14	**Log-Likelihood:**	4945.2
No. Observations:	15948	**AIC:**	-9692.
Df Residuals:	15849	**BIC:**	-8932.
Df Model:	98		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	12.7216	0.018	705.108	0.000	12.686	12.757
sqft_living	0.1700	0.003	55.925	0.000	0.164	0.176
sqft_lot	0.0234	0.002	12.969	0.000	0.020	0.027
sqft_living15	0.0522	0.003	20.130	0.000	0.047	0.057
sqft_lot15	0.0003	0.002	0.142	0.887	-0.004	0.004
waterfront_1	0.4372	0.023	18.608	0.000	0.391	0.483
bedrooms_2	0.0613	0.014	4.261	0.000	0.033	0.089
bedrooms_3	0.0981	0.014	6.893	0.000	0.070	0.126
bedrooms_4	0.1015	0.015	6.996	0.000	0.073	0.130
bedrooms_5	0.0698	0.015	4.524	0.000	0.040	0.100
bedrooms_6	0.0455	0.020	2.269	0.023	0.006	0.085
grade_11 Excellent	0.0603	0.014	4.156	0.000	0.032	0.089
grade_12 Luxury	0.0805	0.034	2.349	0.019	0.013	0.148
grade_13 Mansion	0.4498	0.127	3.555	0.000	0.202	0.698
grade_4 Low	-0.5538	0.040	-13.821	0.000	-0.632	-0.475
grade_5 Fair	-0.4528	0.017	-26.491	0.000	-0.486	-0.419
grade_6 Low Average	-0.3578	0.011	-32.528	0.000	-0.379	-0.336
grade_7 Average	-0.2474	0.009	-26.680	0.000	-0.266	-0.229
grade_8 Good	-0.1476	0.008	-17.727	0.000	-0.164	-0.131
grade_9 Better	-0.0497	0.008	-6.197	0.000	-0.065	-0.034
renovated_1	0.0944	0.008	11.972	0.000	0.079	0.110
view_EXCELLENT	0.1985	0.017	11.838	0.000	0.166	0.231
view_GOOD	0.0660	0.012	5.725	0.000	0.043	0.089
view_NONE	-0.1120	0.006	-17.273	0.000	-0.125	-0.099
zipcode_98004	1.1041	0.013	85.121	0.000	1.079	1.130
zipcode_98005	0.7329	0.017	44.015	0.000	0.700	0.766
zipcode_98006	0.6340	0.011	60.286	0.000	0.613	0.655

zipcode_98007	0.6561	0.017	37.510	0.000	0.622	0.690
zipcode_98008	0.6515	0.013	51.191	0.000	0.627	0.676
zipcode_98010	0.2962	0.024	12.270	0.000	0.249	0.344
zipcode_98011	0.4579	0.015	30.706	0.000	0.429	0.487
zipcode_98014	0.3210	0.022	14.319	0.000	0.277	0.365
zipcode_98019	0.3306	0.016	20.351	0.000	0.299	0.362
zipcode_98024	0.4332	0.030	14.359	0.000	0.374	0.492
zipcode_98027	0.5354	0.012	43.488	0.000	0.511	0.560
zipcode_98028	0.4215	0.013	32.823	0.000	0.396	0.447
zipcode_98029	0.6011	0.013	47.480	0.000	0.576	0.626
zipcode_98030	0.0631	0.013	4.807	0.000	0.037	0.089
zipcode_98031	0.0842	0.013	6.592	0.000	0.059	0.109
zipcode_98033	0.7837	0.011	71.816	0.000	0.762	0.805
zipcode_98034	0.5446	0.010	55.399	0.000	0.525	0.564
zipcode_98038	0.1583	0.010	15.660	0.000	0.138	0.178
zipcode_98039	1.2710	0.033	38.980	0.000	1.207	1.335
zipcode_98040	0.8700	0.014	63.565	0.000	0.843	0.897
zipcode_98042	0.0556	0.010	5.656	0.000	0.036	0.075
zipcode_98045	0.3348	0.016	21.466	0.000	0.304	0.365
zipcode_98052	0.6435	0.010	65.864	0.000	0.624	0.663
zipcode_98053	0.6054	0.012	50.669	0.000	0.582	0.629
zipcode_98055	0.1388	0.013	10.573	0.000	0.113	0.165
zipcode_98056	0.3230	0.011	29.258	0.000	0.301	0.345
zipcode_98058	0.1652	0.010	15.751	0.000	0.145	0.186
zipcode_98059	0.3407	0.010	32.710	0.000	0.320	0.361
zipcode_98065	0.3980	0.013	30.653	0.000	0.373	0.423
zipcode_98070	0.2884	0.026	11.251	0.000	0.238	0.339
zipcode_98072	0.4769	0.013	35.580	0.000	0.451	0.503
zipcode_98074	0.5405	0.011	48.532	0.000	0.519	0.562
zipcode_98075	0.5569	0.012	45.428	0.000	0.533	0.581
zipcode_98077	0.4438	0.018	24.972	0.000	0.409	0.479
zipcode_98102	0.9638	0.022	44.724	0.000	0.922	1.006
zipcode_98103	0.8793	0.011	81.354	0.000	0.858	0.901
zipcode_98105	0.9506	0.015	64.864	0.000	0.922	0.979
zipcode_98106	0.3459	0.012	28.305	0.000	0.322	0.370

zipcode_98107	0.8526	0.016	54.220	0.000	0.822	0.883
zipcode_98108	0.3751	0.016	23.980	0.000	0.344	0.406
zipcode_98109	0.9917	0.021	46.431	0.000	0.950	1.034
zipcode_98112	1.0437	0.014	75.189	0.000	1.017	1.071
zipcode_98115	0.8205	0.010	81.865	0.000	0.801	0.840
zipcode_98116	0.7522	0.013	59.472	0.000	0.727	0.777
zipcode_98117	0.8256	0.010	81.375	0.000	0.806	0.845
zipcode_98118	0.4752	0.010	46.039	0.000	0.455	0.495
zipcode_98119	0.9900	0.017	57.740	0.000	0.956	1.024
zipcode_98122	0.8090	0.013	61.885	0.000	0.783	0.835
zipcode_98125	0.5845	0.011	51.952	0.000	0.562	0.607
zipcode_98126	0.5558	0.012	47.274	0.000	0.533	0.579
zipcode_98133	0.4667	0.011	44.016	0.000	0.446	0.488
zipcode_98136	0.6665	0.014	49.117	0.000	0.640	0.693
zipcode_98144	0.6596	0.012	53.940	0.000	0.636	0.684
zipcode_98146	0.2764	0.013	21.807	0.000	0.252	0.301
zipcode_98148	0.1741	0.027	6.382	0.000	0.121	0.228
zipcode_98155	0.4303	0.011	40.361	0.000	0.409	0.451
zipcode_98166	0.3128	0.013	23.718	0.000	0.287	0.339
zipcode_98168	0.0924	0.013	6.952	0.000	0.066	0.118
zipcode_98177	0.5933	0.014	42.954	0.000	0.566	0.620
zipcode_98178	0.1416	0.013	10.529	0.000	0.115	0.168
zipcode_98188	0.1084	0.018	6.147	0.000	0.074	0.143
zipcode_98198	0.0720	0.013	5.583	0.000	0.047	0.097
zipcode_98199	0.8538	0.013	66.501	0.000	0.829	0.879
has_basement_1	-0.0259	0.004	-7.295	0.000	-0.033	-0.019
month_sold_2	0.0219	0.006	3.561	0.000	0.010	0.034
month_sold_3	0.0542	0.005	10.679	0.000	0.044	0.064
month_sold_4	0.0704	0.005	15.065	0.000	0.061	0.080
condition_Fair	-0.1520	0.016	-9.474	0.000	-0.183	-0.121
condition_Good	0.0489	0.004	13.850	0.000	0.042	0.056
condition_Poor	-0.2424	0.037	-6.582	0.000	-0.315	-0.170
condition_Very Good	0.1163	0.006	21.011	0.000	0.105	0.127
house_age_43	-0.0498	0.011	-4.436	0.000	-0.072	-0.028
house_age_52	-0.0510	0.018	-2.875	0.004	-0.086	-0.016

house_age_56	-0.0471	0.013	-3.573	0.000	-0.073	-0.021
floors_2.0	0.0004	0.004	0.099	0.921	-0.008	0.008

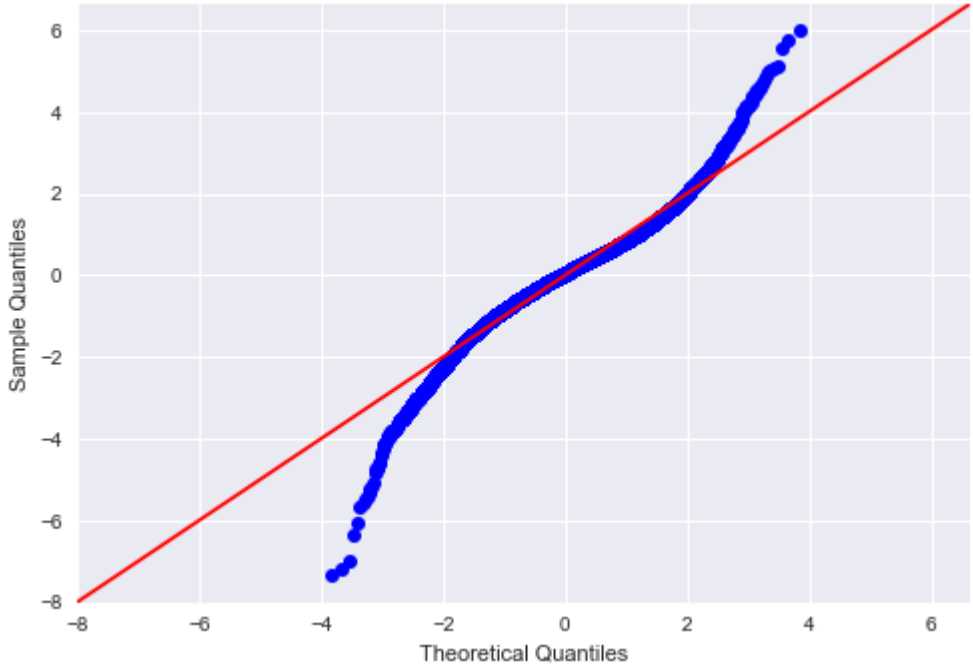
Omnibus:	1345.608	**Durbin-Watson:**	2.008
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	6264.737
Skew:	-0.288	**Prob(JB):**	0.00
Kurtosis:	6.016	**Cond. No.**	154.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Assumptions

Normality

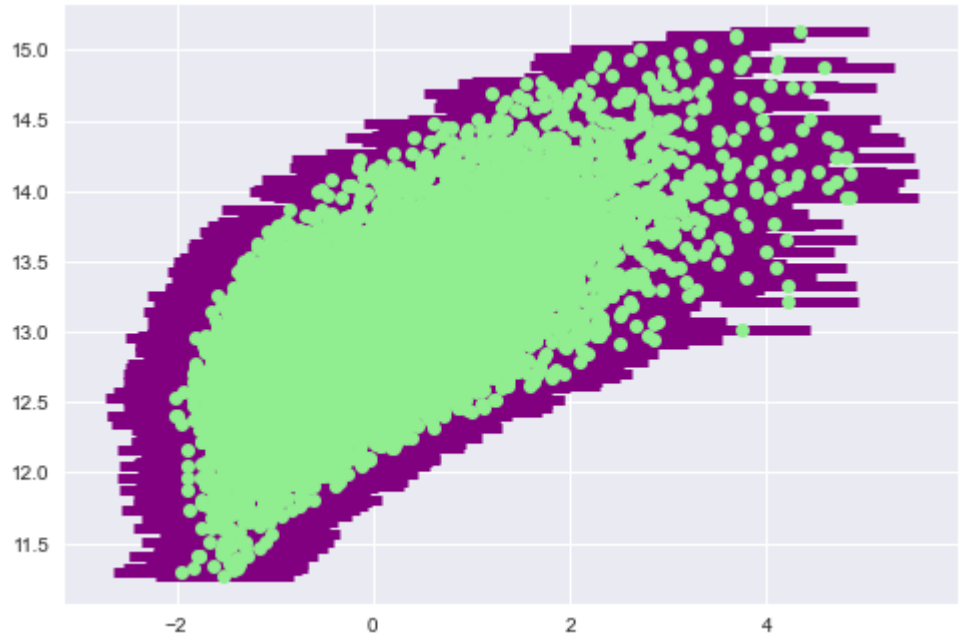
In [99]:
```
sm.graphics.qqplot(model.resid, dist=stats.norm, line='45', fit=True);
```



Homoscedasticity

In [100…
```
plt.errorbar(X_train_all['sqft_living'], y_train, xerr = 0.7, fmt = 'o',color = 'lightg
             ecolor = 'purple', elinewidth = 5, capsize=10)
```

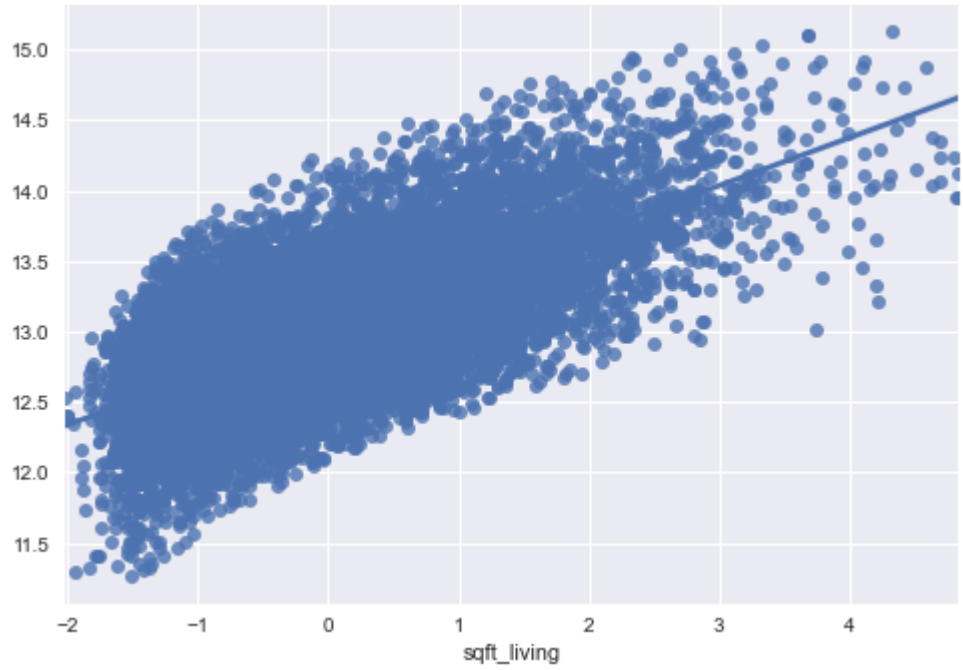
Out[100…
```
<ErrorbarContainer object of 3 artists>
```



Linearity

In [101…
```
sns.regplot(X_train_all['sqft_living'],y_train)
```

Out[101…
```
<AxesSubplot:xlabel='sqft_living'>
```



Overfitting

In [102…
```
lin1 = LinearRegression()
lin1.fit(X_train_all, y_train)
```

Out[102…
```
LinearRegression()
```

In [103…
```
y_pred = lin1.predict(X_train_all)
```

In [104…
```
y_pred
```

Out[104…
```
array([12.88192105, 13.08750703, 14.02850463, ..., 12.40862988,
       12.83408222, 12.21711837])
```

In [105…
```
train_mae = metrics.mean_absolute_error(y_train, y_pred)
train_mse = metrics.mean_squared_error(y_train, y_pred)
train_rmse = np.sqrt(train_mse) # or mse**(0.5)
train_r2 = metrics.r2_score(y_train,y_pred)

print("Results of training data:")
print("MAE:",train_mae)
print("MSE:", train_mse)
print("RMSE:", train_rmse)
print("R-Squared:", train_r2)
```

```
Results of training data:
MAE: 0.1272428546695554
MSE: 0.030464105751575037
RMSE: 0.17453969677862693
R-Squared: 0.8820112599335201
```

In [106…
```
lin2 = LinearRegression()
lin2.fit(X_test_all,y_test)
```

Out[106…
```
LinearRegression()
```

In [107…
```
y_test_pred = lin2.predict(X_test_all)
y_test_pred
```

Out[107…
```
array([12.60201909, 12.35724527, 13.13753382, ..., 13.27976736,
       12.98008558, 12.92393652])
```

In [108…
```
test_mae = metrics.mean_absolute_error(y_test, y_test_pred)
test_mse = metrics.mean_squared_error(y_test, y_test_pred)
test_rmse = np.sqrt(test_mse) # or mse**(0.5)
test_r2 = metrics.r2_score(y_test,y_test_pred)

print("Results of tests data:")
print("MAE:",test_mae)
print("MSE:", test_mse)
print("RMSE:", test_rmse)
print("R-Squared:", test_r2)
```

```
Results of tests data:
MAE: 0.1265533732192119
MSE: 0.028810490980716016
RMSE: 0.16973653401880226
R-Squared: 0.8901668329557788
```

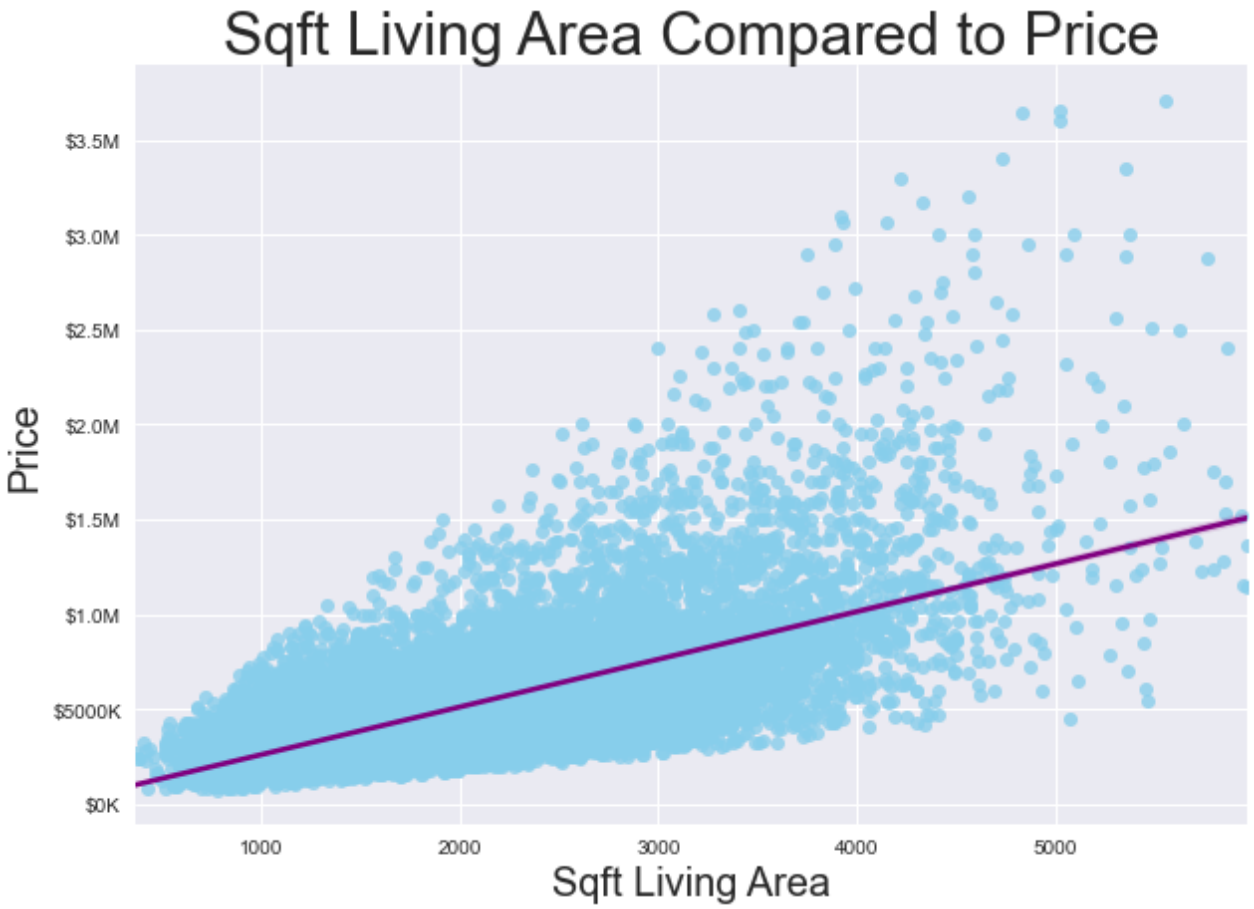
There is no overfitting per the results

Conclusion

Graphs for Presentation

In [109…
```
fig, ax = plt.subplots(figsize=(10,7))
sns.regplot(data=df, x='sqft_living', y='price', ax=ax, color='skyblue', line_kws={'col
```

```
ax.set_title('Sqft Living Area Compared to Price', fontsize=30)
ax.set_xlabel('Sqft Living Area', fontsize=20)
ax.set_ylabel('Price', fontsize=20)
ax.yaxis.set_major_formatter(currency);
```



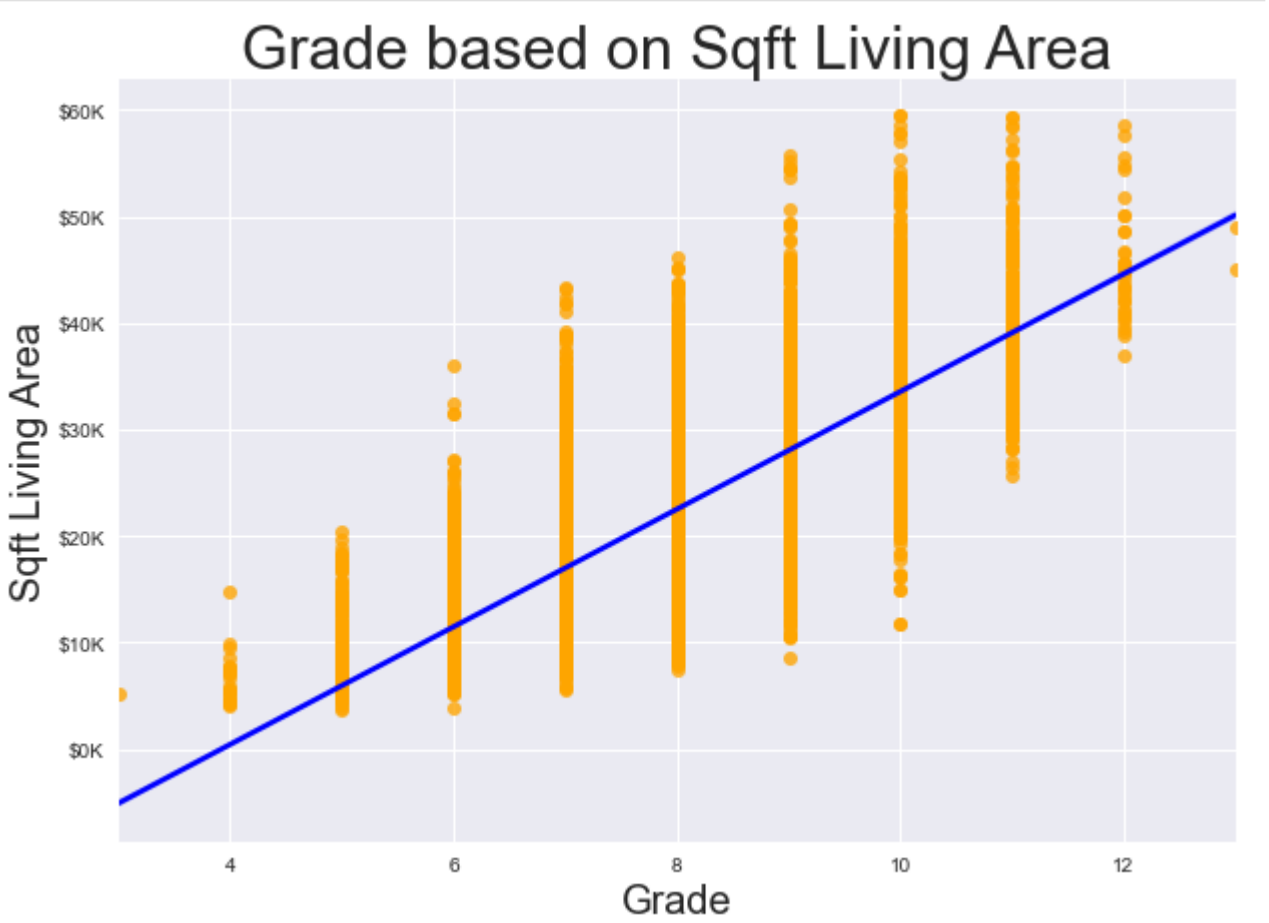
In [110…
```
df['grade'] = df['grade'].str.split(" ", n = 1, expand = True)
df['grade'] = df['grade'].astype(object).astype(int)
```

In [111…
```
df['grade'] = df['grade'].sort_values(axis=0, ascending=False)
```

In [112…
```
df['grade']
```

Out[112…
```
0        7
1        7
2        6
3        7
4        8
       ..
21591    8
21593    8
21594    7
21595    8
21596    7
Name: grade, Length: 19936, dtype: int32
```

In [115…
```
fig, ax = plt.subplots(figsize=(10,7))
sns.regplot(data=df, x='grade', y='sqft_living', ax=ax, color='orange', line_kws={'colo
ax.set_title('Grade based on Sqft Living Area', fontsize=30)
ax.set_xlabel('Grade', fontsize=20)
ax.set_ylabel('Sqft Living Area', fontsize=20)
ax.yaxis.set_major_formatter(currency)
```



In [116…

```
with plt.style.context('fivethirtyeight'):
    fig, ax = plt.subplots(figsize=(10,7))
    sns.regplot(data=df, x='grade', y='price', ax=ax, color='red', line_kws={'color': '
    ax.set_title('Grade based on Price', fontsize=30)
    ax.set_xlabel('Grade', fontsize=20)
    ax.set_ylabel('Price', fontsize=20)
    ax.yaxis.set_major_formatter(currency)
```

Interpretations

Top Features:

- Zipcodes 98112, 98039, & 98004
- Square foot living
- Grade
- Condition
- View

Interpretations

- The top three coefficients for Zipcodes 98112, 98039, and 98004 are 1.04, 1.27, and 1.1 repsectively. This means that price will increase in that location by 10.4%, 12.7%, and 11%.
- The coefficient for sqft_living is .17 meaning for every 100 sqft added, the price will increase by 1.7%
- The coefficient for the highest grade is .44 meaning if the house is mansion grade, the price will increase by 4.4%
- The coefficient for the top condition is .12 meaning if the house is in very good condition, the price will increase by 1.2%
- The coefficient for the top view is .2 meaning if the view is considered excellent, the price will increase by 2%

Recommendations

Recommendations:

- If you are selling your house or increase the value of it, try and increase the square footage of the house (whether it's adding an addition to the property)
- Target homes in specific areas (such as Zipcodes 98112, 98039, or 98004) to buy and resell since it has proven to be the biggest factor effecting price
- If you are selling, improve the quality of your house by investing in upgrades. Grade has proven to be a huge factor correlating with the price

Future Work

- Look at location (lat/long) to see how much of a factor it plays into the price of a house.
- Gather data from different school districts in KC to see if there is a relationship with prices of a house and the quality of schools.
- Get before/after stats on renovated houses to see the frequency houses are being renovated and if it has an effect on prices

In [ ]:

In [ ]:

In [ ]: