1 Scrub

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
pd.set_option('display.max_columns', None)

df = pd.read_csv('../dsc-phase-2-project/data/kc_house_data.csv', thous
df.head()
```

Out[1]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront
0	7129300520	10/13/2014	221900.0	3	1.00	1180	5650	1.0	NaN
1	6414100192	12/9/2014	538000.0	3	2.25	2570	7242	2.0	0.0
2	5631500400	2/25/2015	180000.0	2	1.00	770	10000	1.0	0.0
3	2487200875	12/9/2014	604000.0	4	3.00	1960	5000	1.0	0.0
4	1954400510	2/18/2015	510000.0	3	2.00	1680	8080	1.0	0.0

In [2]:
```
# Date should be a datetime object
# Sftft_basement contains strings

pd.set_option('display.float_format', lambda x: '%.2f' % x)
df.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  float64
 9   view           21534 non-null  float64
 10  condition      21597 non-null  int64  
 11  grade          21597 non-null  int64  
 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(8), int64(11), object(2)
memory usage: 3.5+ MB
```

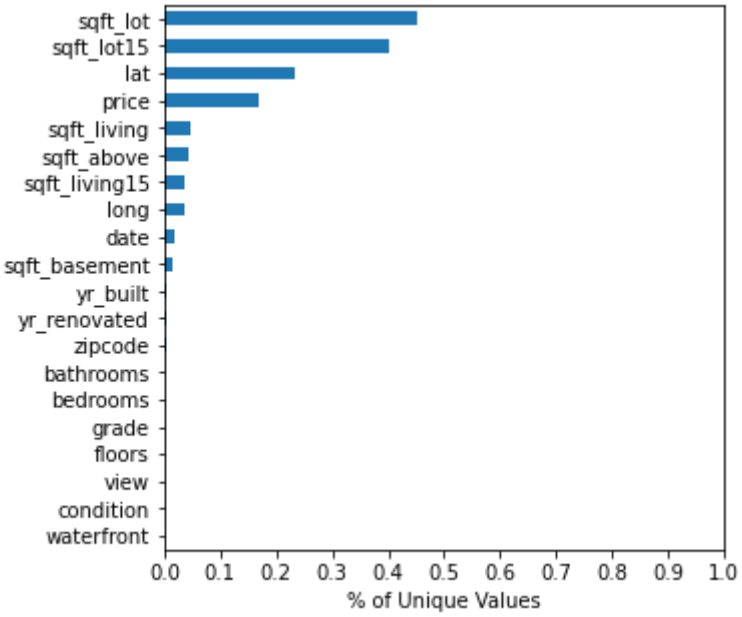
In [3]:
```
df['date'] = pd.to_datetime(df['date'])
```

In [4]:
```
# id isn't truly an int
# price has a very large SD and outliers
# it seems that there may be a couple of outlier values driving up the
# floors, waterfront, view, condition, and grade are ordinal categorica
# zipcode, lat, long isn't truly an int
# waterfront may be a binary variable
# year renovated contains null values, probably due to homes that have

df.describe()
```

Out[4]:

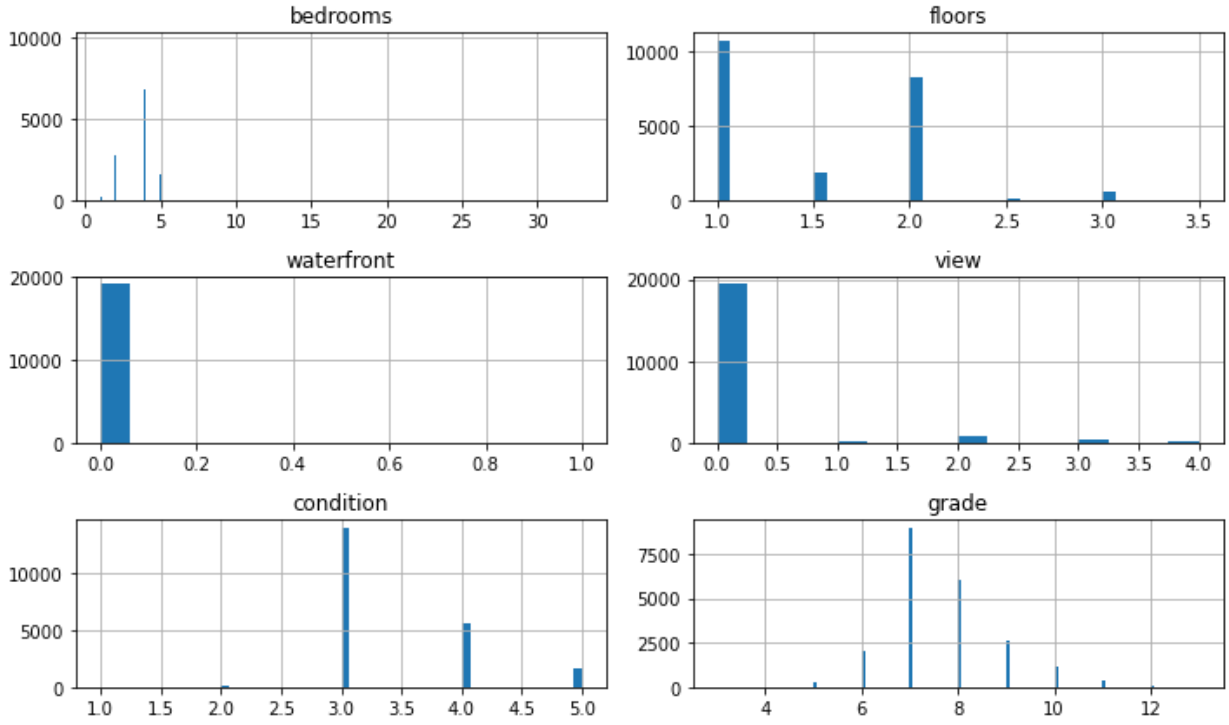
	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfro
count	21597.00	21597.00	21597.00	21597.00	21597.00	21597.00	21597.00	19221.
mean	4580474287.77	540296.57	3.37	2.12	2080.32	15099.41	1.49	0.
std	2876735715.75	367368.14	0.93	0.77	918.11	41412.64	0.54	0.
min	1000102.00	78000.00	1.00	0.50	370.00	520.00	1.00	0.
25%	2123049175.00	322000.00	3.00	1.75	1430.00	5040.00	1.00	0.
50%	3904930410.00	450000.00	3.00	2.25	1910.00	7618.00	1.50	0.
75%	7308900490.00	645000.00	4.00	2.50	2550.00	10685.00	2.00	0.
max	9900000190.00	7700000.00	33.00	8.00	13540.00	1651359.00	3.50	1.

In [5]:
```
# Show the proportion of unique values per column

# Bathrooms, bedrooms, grade, floors, view, condition, and waterfront t
# of unique values. They may be categorical/ordinal
# Zipcode, year renovated, year built, basement, sqft living, sqft abov

nu_dict = dict(df.nunique())
val_list = list(nu_dict.values())

percent = []
for col, val in zip(df.columns, val_list):
    percent.append(val/len(df[col]))
unique_per_column = pd.Series(data=percent, index=df.columns)
unique_per_column.drop('id').sort_values(ascending=True).plot(kind='bar
plt.xlabel('% of Unique Values');
```



In [6]:

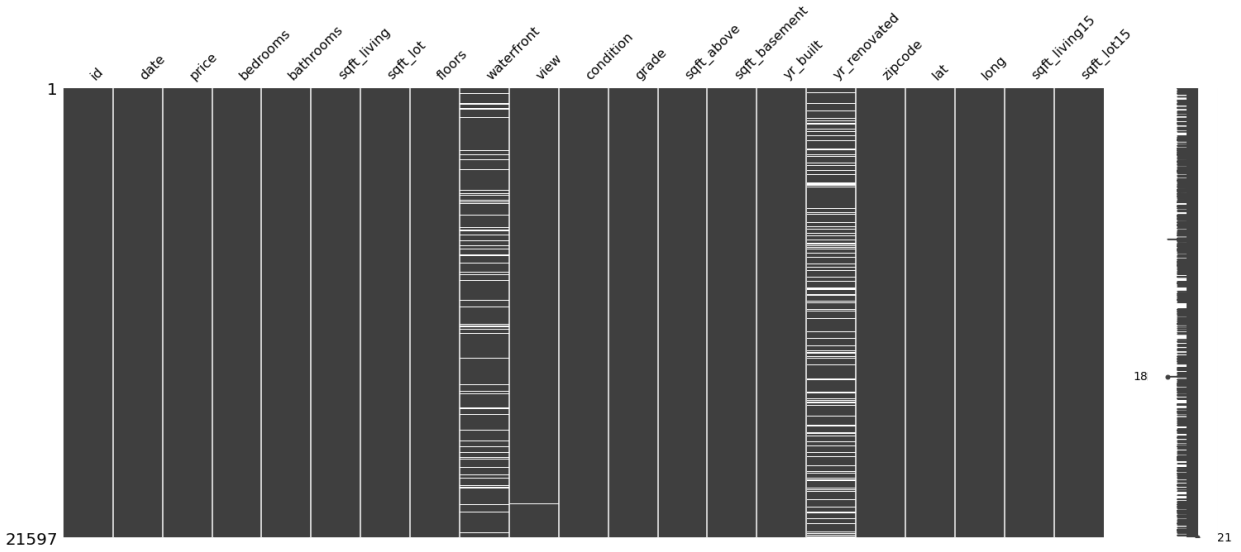
```
# It appears all but waterfront are categorical variables with ordinal
# Waterfront is the only binary variable

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

low_nunique = []
for k,v in nu_dict.items():
    if v < 15:
        low_nunique.append(k)
    else:
        pass
low_nunique

df[low_nunique].hist(figsize=(10,6), bins='auto');
plt.tight_layout();
```



1.1 Fill in missing values

In [7]:
```
# Start checking for null values
# Waterfront, view, and yr_renovated are the only columns with missing

import missingno
missingno.matrix(df)
```

Out[7]: <AxesSubplot:>



In [8]:
```
null = df.isna().sum()
null[null>1]
```

Out[8]:
```
waterfront      2376
view              63
yr_renovated    3842
dtype: int64
```

In [9]:
```
# No recognizable pattern for why certain view values are missing

print(df['view'].value_counts(1, dropna=False))
df[df['view'].isnull()]
```

```
0.00    0.90
2.00    0.04
3.00    0.02
1.00    0.02
4.00    0.01
nan     0.00
Name: view, dtype: float64
```

Out[9]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfron
7	2008000270	2015-01-15	291850.00	3	1.50	1060	9711	1.00	0.0(
114	8961960160	2014-10-28	480000.00	4	2.50	3230	16171	2.00	0.0(
129	7853210060	2015-04-06	430000.00	4	2.50	2070	4310	2.00	0.0(
205	3456000310	2014-08-04	840000.00	4	1.75	2480	11010	1.00	0.0(
487	1895000260	2014-07-21	207950.00	2	2.00	890	5000	1.00	0.0(
...	...	...	...	...	...	...	...	...	..
19989	148000475	2014-05-28	1400000.00	4	3.25	4700	9160	1.00	0.0(
20148	291310170	2014-08-04	384500.00	3	2.50	1600	2610	2.00	0.0(
20380	1196003740	2014-09-24	734000.00	5	4.25	4110	42755	2.00	0.0(
21057	3448900290	2014-08-28	636230.00	4	2.50	2840	6284	2.00	0.0(
21589	3448900210	2014-10-14	610685.00	4	2.50	2520	6023	2.00	0.0(

63 rows × 21 columns

In [10]:
```
# Going to assume null value represents a home that has never been renc

print(df['yr_renovated'].value_counts(1, dropna=False))
df[df['yr_renovated'].isnull()]
```

```
0.00       0.79
nan        0.18
2014.00    0.00
2003.00    0.00
2013.00    0.00
           ...
1944.00    0.00
1948.00    0.00
1976.00    0.00
1934.00    0.00
1953.00    0.00
Name: yr_renovated, Length: 71, dtype: float64
```

Out[10]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfron
2	5631500400	2015-02-25	180000.00	2	1.00	770	10000	1.00	0.0(
12	114101516	2014-05-28	310000.00	3	1.00	1430	19901	1.50	0.0(
23	8091400200	2014-05-16	252700.00	2	1.50	1070	9643	1.00	nar
26	1794500383	2014-06-26	937000.00	3	1.75	2450	2691	2.00	0.0(
28	5101402488	2014-06-24	438000.00	3	1.75	1520	6380	1.00	0.0(
...	...	...	...	...	...	...	...	...	..
21576	1931300412	2015-04-16	475000.00	3	2.25	1190	1200	3.00	0.0(
21577	8672200110	2015-03-17	1090000.00	5	3.75	4170	8142	2.00	0.0(
21579	1972201967	2014-10-31	520000.00	2	2.25	1530	981	3.00	0.0(
21581	191100405	2015-04-21	1580000.00	4	3.25	3410	10125	2.00	0.0(
21583	7202300110	2014-09-15	810000.00	4	3.00	3990	7838	2.00	0.0(

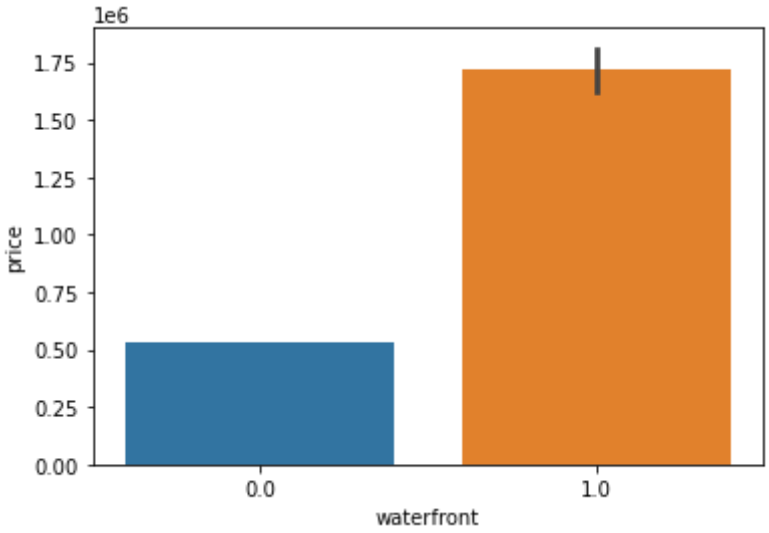
3842 rows × 21 columns

In [11]:
```
# Handle waterfront null values
# Check if it can be explained by location

print('Has view of water (Latitude)')
print(df[df['waterfront']==1.0]['lat'].describe())
print('-----------------------------------------')
print('Does not have view of water (Latitude)')
print(df[df['waterfront']==0.0]['lat'].describe())
print('-----------------------------------------')
print('Has view of water (Price)')
print(df[df['waterfront']==1.0]['price'].describe())
print('-----------------------------------------')
print('Does not have view of water')
print(df[df['waterfront']==0.0]['price'].describe())
```

```
Has view of water (Latitude)
count   146.00
mean     47.54
std       0.11
min      47.33
25%      47.45
50%      47.55
75%      47.61
max      47.77
Name: lat, dtype: float64
-----------------------------------------
Does not have view of water (Latitude)
count   19075.00
mean       47.56
std         0.14
min        47.16
25%        47.47
50%        47.57
75%        47.68
max        47.78
Name: lat, dtype: float64
-----------------------------------------
Has view of water (Price)
count       146.00
mean    1717214.73
std     1145384.86
min      285000.00
25%      827500.00
50%     1510000.00
75%     2282500.00
max     7060000.00
Name: price, dtype: float64
-----------------------------------------
Does not have view of water
count     19075.00
mean     532641.99
std      344959.18
min       78000.00
25%      320000.00
50%      450000.00
75%      638600.00
max     7700000.00
```

```
Name: price, dtype: float64
```

In [12]:
```
# Clearly waterfront homes are much more expensive

sns.barplot(data=df, x='waterfront', y='price', ci=68);
```



In [13]:
```
mplus_water =len(df[(df['price']>1000000) & df['waterfront']==1.0])
print(f'Number of houses over $1,000,000 with waterfront view:\t{mplus_
mminus_water = len(df[(df['price']<1000000) & df['waterfront']==1.0])
print(f'Number of houses under $1,000,000 with waterfront view:\t{mminu
```

```
Number of houses over $1,000,000 with waterfront view:  96
Number of houses under $1,000,000 with waterfront view: 49
```

In [14]:
```
# Going to impute categorical variables the probability that they appea
# Begin with waterfront and functionize

# Prior count below:
# 0.00    19075
# 1.00      146
pd.set_option('display.float_format', lambda x: '%.5f' % x)
# Prob of having waterfront view for homes over $1,000,000
print('$1M+ with waterfront');
print(df.loc[df['price']>1000000]['waterfront'].value_counts(1));
print('-----------------------------------------')
# Prob of having waterfront view for homes under $1,000,000
print('$1M- with waterfront');
print(df.loc[df['price']<1000000]['waterfront'].value_counts(1))
```

```
$1M+ with waterfront
0.00000   0.92666
1.00000   0.07334
Name: waterfront, dtype: float64
-----------------------------------------
$1M- with waterfront
0.00000   0.99726
1.00000   0.00274
Name: waterfront, dtype: float64
```

In [15]:
```
def impute_cat(df, col):
    '''
    Impute null value with value based on liklihood
    of occurring in the original column
    '''
    val_prob = dict(df[col].value_counts(1))
    prob = list(val_prob.values())
    val = list(val_prob.keys())
    np.random.choice(val, p=prob)
    df[col].fillna(np.random.choice(val, p=prob), inplace=True)
    return df
```

In [16]:
```
# Handling question mark as null value

df['sqft_basement'].replace(to_replace='?', value='0.0',inplace=True)
s = df['sqft_basement']
pd.to_numeric(s, downcast='integer')
```

Out[16]:
```
0          0
1        400
2          0
3        910
4          0
        ...
21592      0
21593      0
21594      0
21595      0
21596      0
Name: sqft_basement, Length: 21597, dtype: int16
```

In [17]:
```
df['sqft_basement'].value_counts()
```

Out[17]:
```
0.0       13280
600.0       217
500.0       209
700.0       208
800.0       201
           ...
2240.0        1
3480.0        1
1798.0        1
243.0         1
417.0         1
Name: sqft_basement, Length: 303, dtype: int64
```

In [18]:
```
df['basementyes'] = (df['sqft_basement']!='0.0').map({True:1,
                                                      False: 0})
```

In [19]:
```
df['basementyes'].value_counts(1)
```

Out[19]:
```
0   0.61490
1   0.38510
Name: basementyes, dtype: float64
```

In [20]:
```
df_1mplus=df.loc[df['price']>1000000]
df_1mminus=df.loc[df['price']<1000000]
```

In [21]:
```
df_1mplus =impute_cat(df_1mplus, 'waterfront')
```

```
/Users/ethankunin/opt/anaconda3/envs/learn-env/lib/python3.8/site-package
s/pandas/core/series.py:4517: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  return super().fillna(
```

In [22]:
```
df_1mminus['waterfront'] =df_1mminus['waterfront'].fillna(0)
```

```
<ipython-input-22-d246c886c934>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  df_1mminus['waterfront'] =df_1mminus['waterfront'].fillna(0)
```

In [23]:
```
df=pd.concat([df_1mminus, df_1mplus])
```

In [24]:
```
# Quality of the view from the home based on scale of 1-4

impute_cat(df, 'view')
```

Out[24]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	wate
0	7129300520	2014-10-13	221900.00000	3	1.00000	1180	5650	1.00000	0.
1	6414100192	2014-12-09	538000.00000	3	2.25000	2570	7242	2.00000	0.
2	5631500400	2015-02-25	180000.00000	2	1.00000	770	10000	1.00000	0.
3	2487200875	2014-12-09	604000.00000	4	3.00000	1960	5000	1.00000	0.
4	1954400510	2015-02-18	510000.00000	3	2.00000	1680	8080	1.00000	0.
...	...	...	...	...	...	...	...	...	
21574	7430200100	2014-05-14	1220000.00000	4	3.50000	4910	9444	1.50000	0.
21577	8672200110	2015-03-17	1090000.00000	5	3.75000	4170	8142	2.00000	0.
21581	191100405	2015-04-21	1580000.00000	4	3.25000	3410	10125	2.00000	0.
21584	249000205	2014-10-15	1540000.00000	5	3.75000	4470	8088	2.00000	0.
21590	7936000429	2015-03-26	1010000.00000	4	3.50000	3510	7200	2.00000	0.

21565 rows × 22 columns

In [25]:
```
# Consider 0 to mean the home has not been renovated
# Conservativley determined null values should be considered non-renova

df['yr_renovated'].value_counts().sort_values(ascending=False).nlargest
```

Out[25]:
```
0.00000       16988
2014.00000       73
2003.00000       31
2013.00000       31
2007.00000       30
2000.00000       29
2005.00000       29
1990.00000       22
2004.00000       22
2009.00000       21
1989.00000       20
2006.00000       20
2002.00000       17
1998.00000       16
1984.00000       16
2010.00000       15
1983.00000       15
2001.00000       15
1999.00000       15
2008.00000       15
1991.00000       15
2015.00000       14
1985.00000       14
1986.00000       14
1987.00000       14
1994.00000       14
1992.00000       13
1993.00000       12
1997.00000       12
1995.00000       12
Name: yr_renovated, dtype: int64
```

In [26]:
```
# No null values remaining

df['yr_renovated'].fillna(0, inplace=True)
```

1.2 Handle Duplicates

In [27]:
```
# I found duplicates in the id column. I interpreted them as changes in
# were the same outside of the date. I decided to keep 'last' because t

pd.set_option('display.max_rows', 500)

df[df.duplicated(subset=['id'],keep=False)]
df=df.drop_duplicates(subset=['id'], keep='last')
```

In [28]:
```
# df.sort_values(by='total_rooms', ascending=False)
## 33 bedrooms, believed to be human error because it did not correspon
df=df.drop(15856)
```

2 Baseline Model

In [29]:
```
import statsmodels.api as sm
import statsmodels.stats.api as sms
import statsmodels.formula.api as smf

from scipy import stats
from sklearn.preprocessing import StandardScaler
from statsmodels.formula.api import ols
```

In [30]:
```
def model_summary(df, X_targets, y, qq=True):
    '''
    Produces OLS Linear Regression summary. True/False toggles if the Q
    plot is displayed below the summary
    '''
    outcome = y
    x_cols = X_targets
    predictors = '+'.join(x_cols)
    formula = outcome + '~' + predictors
    model = ols(formula=formula, data=df).fit()
    resid1 = model.resid
    display(model.summary())
    if qq==True:

        sm.graphics.qqplot(resid1, dist=stats.norm, line='45', fit=True

    return model
```

In [31]:
```

x_baseline = ['bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade'
       'sqft_above', 'yr_built', 'yr_renovated', 'zipcode',
       'lat', 'long', 'sqft_living15', 'sqft_lot15', 'basementyes']

```

In [32]:
```
from sklearn.datasets import make_regression
from sklearn.linear_model import LinearRegression
import sklearn.metrics as metrics


def sked_show(df, X_cols, lr=None, val='price'):
    '''
    Produces scatter plot showing measue of homoskedacicity
    '''
    if lr is None:
        lr = LinearRegression()
        lr.fit(df[X_cols], df[val])

        y_hat = lr.predict(df[X_cols])
    else:
        y_hat = lr.predict(df)


    resid = (df[val] - y_hat)
    fig, ax= plt.subplots(figsize=(5,5))
    ax.scatter(x=y_hat,y=resid, alpha=0.1)
    ax.axhline(0, color='red')
    ax.set_xlabel('Price')
    ax.set_ylabel('Residual')
    return fig,ax
```

In [33]:
```
from sklearn.datasets import make_regression
from sklearn.linear_model import LinearRegression
import sklearn.metrics as metrics

```

In [34]:
```
model_base = model_summary(df, x_baseline, 'price')
sked_show(df, x_baseline, model_base)
```

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.700
Model:	OLS	**Adj. R-squared:**	0.700
Method:	Least Squares	**F-statistic:**	2773.
Date:	Mon, 19 Apr 2021	**Prob (F-statistic):**	0.00
Time:	09:29:39	**Log-Likelihood:**	-2.9152e+05
No. Observations:	21387	**AIC:**	5.831e+05
Df Residuals:	21368	**BIC:**	5.832e+05
Df Model:	18		
Covariance Type:	nonrobust		

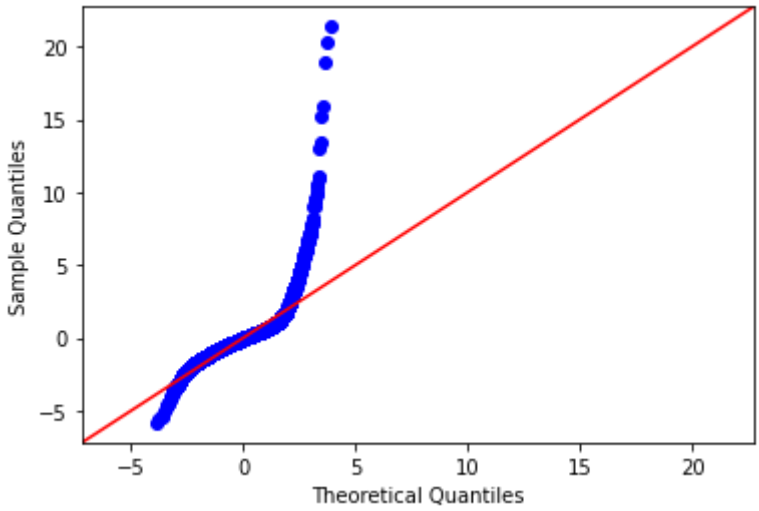
	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	6.706e+06	2.96e+06	2.265	0.024	9.02e+05	1.25e+07
bedrooms	-3.948e+04	1995.040	-19.790	0.000	-4.34e+04	-3.56e+04
bathrooms	4.396e+04	3311.533	13.276	0.000	3.75e+04	5.05e+04
sqft_living	153.9370	6.047	25.455	0.000	142.084	165.790
sqft_lot	0.1241	0.048	2.587	0.010	0.030	0.218
floors	6198.9228	3615.370	1.715	0.086	-887.474	1.33e+04
waterfront	6.214e+05	1.82e+04	34.159	0.000	5.86e+05	6.57e+05
view	5.271e+04	2117.878	24.888	0.000	4.86e+04	5.69e+04
condition	2.633e+04	2366.643	11.127	0.000	2.17e+04	3.1e+04
grade	9.643e+04	2179.377	44.245	0.000	9.22e+04	1.01e+05
sqft_above	28.3925	6.608	4.296	0.000	15.440	41.345
yr_built	-2657.5678	72.361	-36.727	0.000	-2799.401	-2515.735
yr_renovated	23.8560	3.995	5.971	0.000	16.025	31.687
zipcode	-584.6527	33.218	-17.601	0.000	-649.762	-519.544
lat	6.008e+05	1.08e+04	55.684	0.000	5.8e+05	6.22e+05
long	-2.178e+05	1.32e+04	-16.442	0.000	-2.44e+05	-1.92e+05
sqft_living15	21.2177	3.464	6.125	0.000	14.428	28.007
sqft_lot15	-0.3956	0.073	-5.388	0.000	-0.540	-0.252
basementyes	-2959.0642	5095.821	-0.581	0.561	-1.29e+04	7029.128

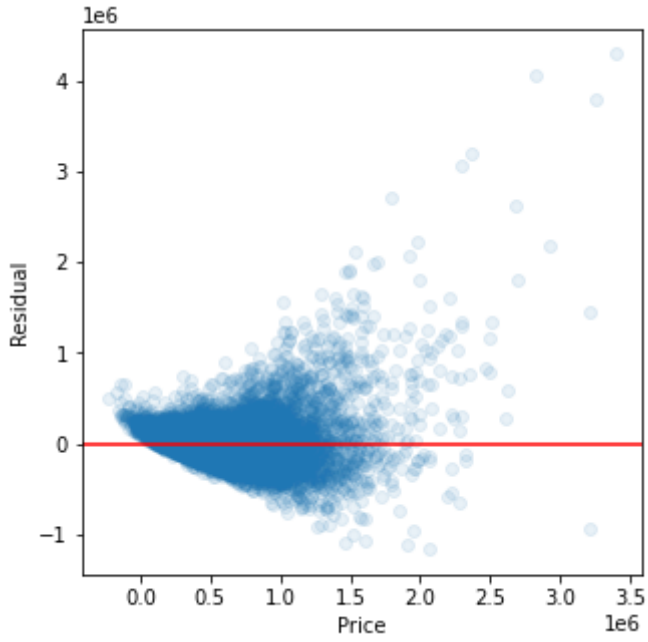
Omnibus:	18085.326	**Durbin-Watson:**	1.620
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1787804.900

Skew:	3.540	**Prob(JB):**	0.00
Kurtosis:	47.228	**Cond. No.**	2.17e+08

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.17e+08. This might indicate that there are
strong multicollinearity or other numerical problems.

Out[34]:
```
(<Figure size 360x360 with 1 Axes>,
 <AxesSubplot:xlabel='Price', ylabel='Residual'>)
```





2.1 Baseline Model Conclusion

- R^2 is 0.70
- QQ Plot deviates significantly around the 3rd quantile
- Floors and basement are not statistically significant
- Residuals trail off around $1.5 million, begins to become cone shaped

- Model not meet assumption of homoscedasticity

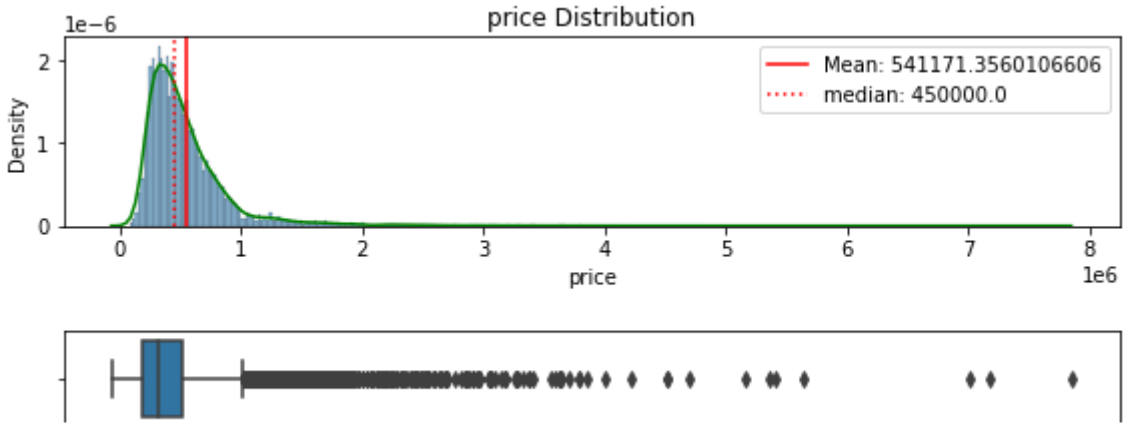
3 EDA/New Feature Model

In [35]:

```
def distr_(df, col):
    fig, ax = plt.subplots(figsize=(8,7), nrows=3, gridspec_kw={'height
    mean=df[col].mean()
    median=df[col].median()
    max_=df[col].max()
    min_=df[col].min()
    std_=df[col].std()
    sns.histplot(df[col],alpha=0.5,stat='density',ax=ax[0]);
    sns.kdeplot(df[col],color='green',ax=ax[0]);
    ax[0].set_xlabel(col)
    ax[0].set_title(f'{col} Distribution')
    ax[0].axvline(mean, label=f'Mean: {mean}', c='red')
    ax[0].axvline(median, label=f'median: {median}', c='red', linestyle
    ax[0].legend()

    sns.boxplot(data=df, x=col, ax=ax[1]);

    sns.scatterplot(data=df, x=df[col], y=df['price']);

    fig.tight_layout();
    print(f'{col.capitalize()} Summary')
    print(f'Median: {median}')
    print(f'Mean: {mean:.4}')
    print(f'Max: {max_}')
    print(f'Min: {min_}')
    print(f'Std: {std_:.4}')
    plt.show()

```

In [36]:
```
eda_check = ['price', 'bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade'
       'sqft_above', 'yr_built', 'yr_renovated', 'zipcode',
       'sqft_living15', 'sqft_lot15', 'basementyes']
for col in eda_check:
    print(distr_(df, col))
    print('-----------------------------------------------------------
```

```
Price Summary
Median: 450000.0
Mean: 5.412e+05
Max: 7700000.0
Min: 78000.0
Std: 3.674e+05
```



EDA Results

- Price is right skewed
- Bedrooms increases up to 5/6 and then decreases
- Bathrooms increases with price
- Sqft living is right skewed and has a positive relationship with price
- Floors doesn't seem to have a relationship with price
- View doesn't seem to have a relationship with price
- Condition increases as price increases
- Grade increases as price increases
- Sqft above increases with price
- Year built doesn't have a relationship with price
- Year renovated looks equivalent
- Zipcode varies
- Sqft lot does not look linear

In [37]:
```
# Total rooms
# Addition, multiplication would create too large of a SD. ie (2 beds 1
df['total_rooms'] = df['bedrooms']+df['bathrooms']
```

In [38]:
```
# Had erroniously stated 33 bedrooms and 1.5 bathrooms
# df=df.drop(15856)
df.sort_values(by='total_rooms', ascending=False)
```

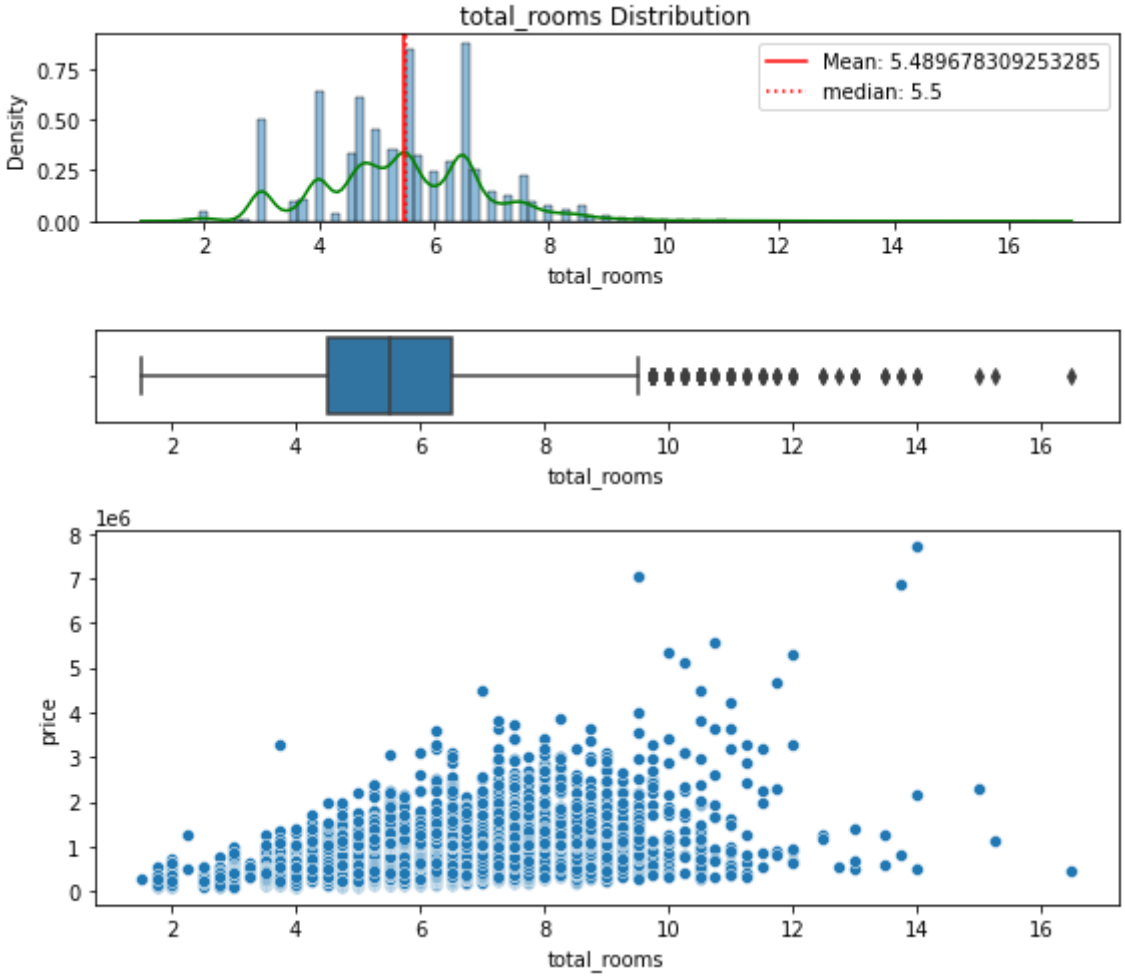
Out[38]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	wate
8537	424049043	2014-08-11	450000.00000	9	7.50000	4050	6504	2.00000	0.
13301	627300145	2014-08-14	1150000.00000	10	5.25000	4590	10920	1.00000	0.
12764	1225069038	2014-05-05	2280000.00000	7	8.00000	13540	307752	3.00000	0.
8748	1773100755	2014-08-21	520000.00000	11	3.00000	3000	4960	2.00000	0.
7245	6762700020	2014-10-13	7700000.00000	6	8.00000	12050	27600	2.50000	0.
...	...	...	...	...	...	...	...	...	
1973	5101404170	2014-11-13	200000.00000	1	0.75000	680	9600	1.00000	0.
9811	3598600049	2015-04-24	224000.00000	1	0.75000	840	7203	1.50000	0.
8614	6303400395	2015-01-30	325000.00000	1	0.75000	410	8636	1.00000	0.
10469	7129304375	2014-07-14	202000.00000	1	0.75000	590	5650	1.00000	0.
11662	7987400316	2014-08-14	255000.00000	1	0.50000	880	1642	1.00000	0.

21387 rows × 23 columns

In [39]:
```
1 distr_(df, 'total_rooms')
```

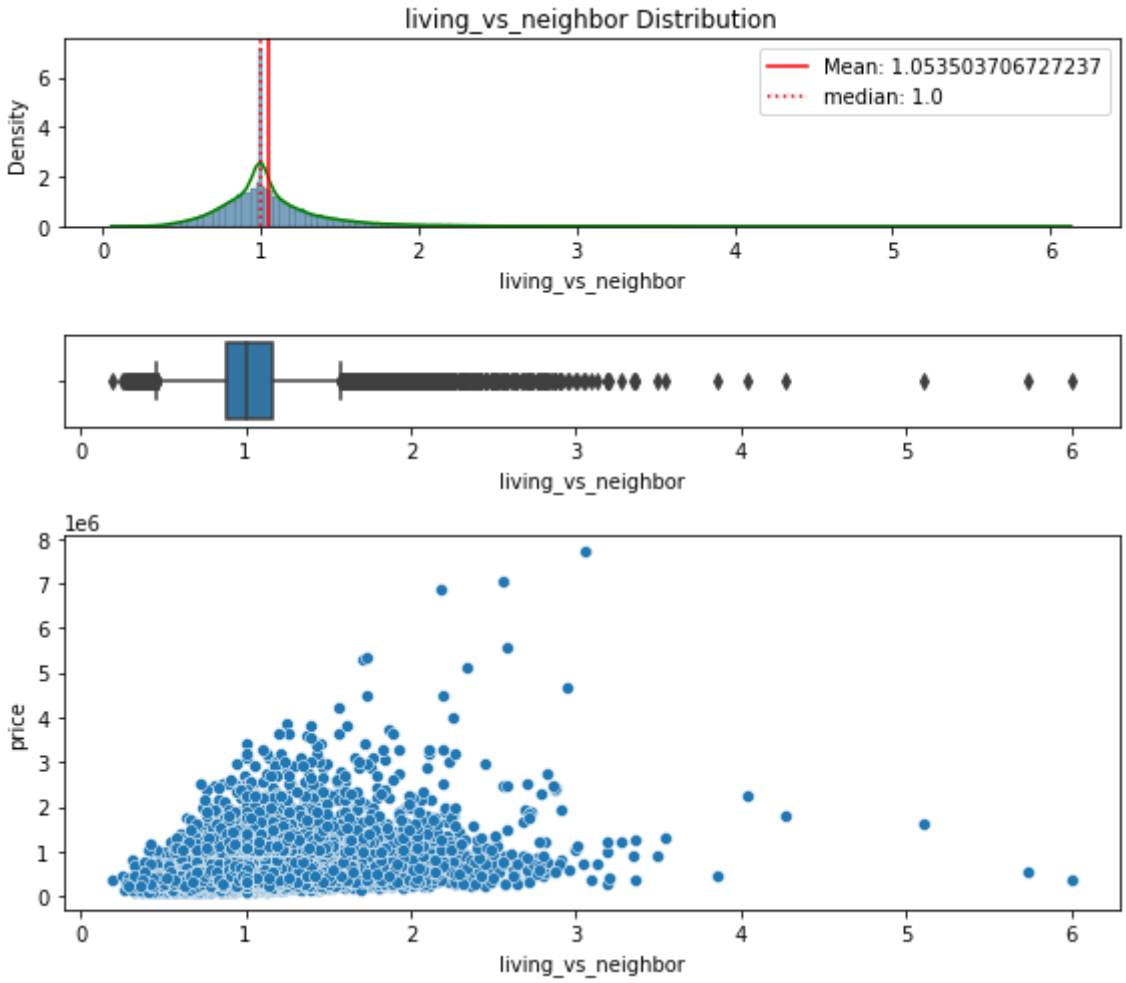
```
Total_rooms Summary
Median: 5.5
Mean: 5.49
Max: 16.5
Min: 1.5
Std: 1.463
```



In [40]:
```
# SQF vs. neighborhood average Living
df['living_vs_neighbor'] = df['sqft_living']/df['sqft_living15']
```

In [41]:
```
distr_(df, 'living_vs_neighbor')
```

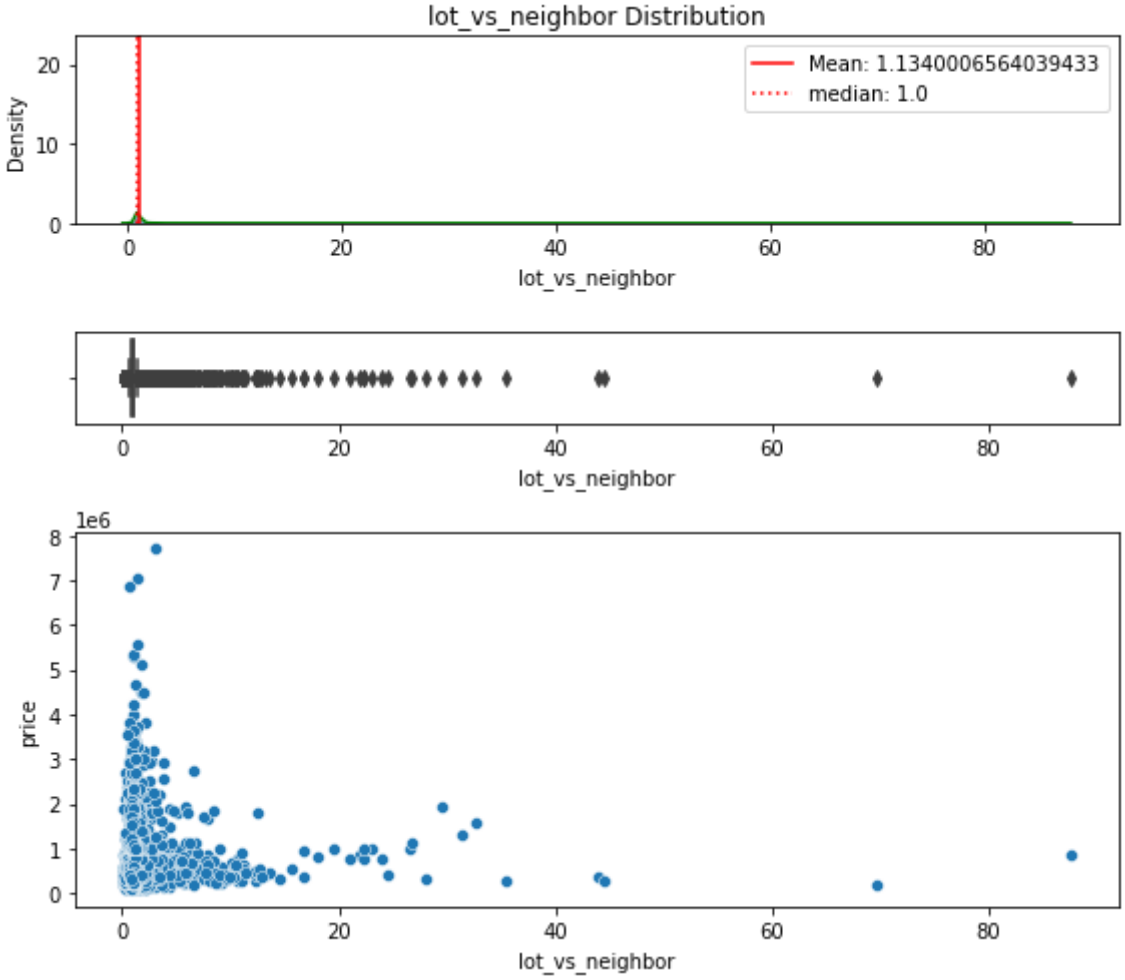
```
Living_vs_neighbor Summary
Median: 1.0
Mean: 1.054
Max: 6.0
Min: 0.1872791519434629
Std: 0.3203
```



In [42]:
```
df['lot_vs_neighbor'] = df['sqft_lot']/df['sqft_lot15']
```

In [43]:
```
1 distr_(df, 'lot_vs_neighbor')
```

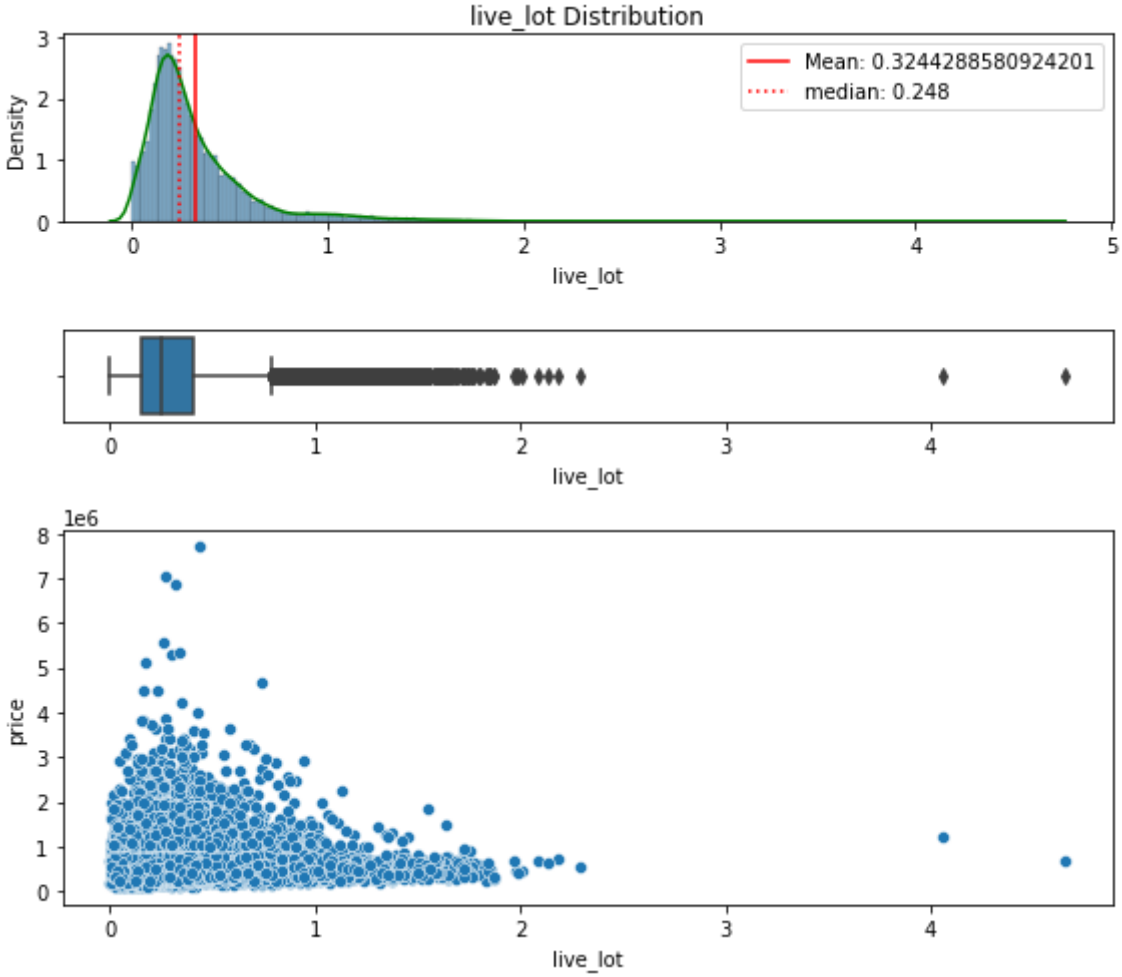
```
Lot_vs_neighbor Summary
Median: 1.0
Mean: 1.134
Max: 87.52717948717948
Min: 0.054971997700810314
Std: 1.286
```



In [44]:
```
1 # SQF vs. lot size
2 df['live_lot'] = df['sqft_living']/df['sqft_lot']
```

In [45]:
```
1 distr_(df, 'live_lot')
```

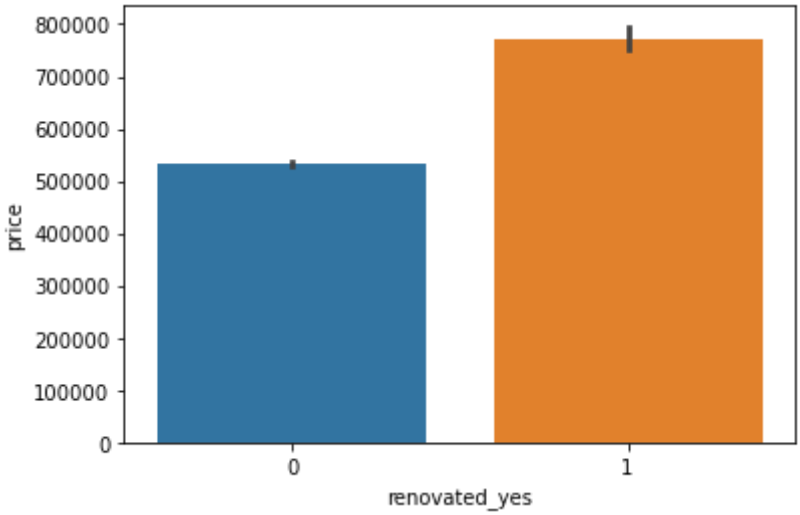
```
Live_lot Summary
Median: 0.248
Mean: 0.3244
Max: 4.653846153846154
Min: 0.0006095498431482305
Std: 0.2692
```



In [46]:
```
1 df['renovated_yes'] = (df['yr_renovated']!=0).map({True:1,
2                                                    False: 0})
3 df['renovated_yes'].value_counts(1)
```

Out[46]:
```
0   0.96549
1   0.03451
Name: renovated_yes, dtype: float64
```

In [47]:
```
# Homes that have been renovated have greater mean price

sns.barplot(data=df, x='renovated_yes', y='price', ci=68);
plt.show()
print(f"Has had renovation mean price: {round(df[df['renovated_yes']==1
print(f"Has NOT had renovation mean price: {round(df[df['renovated_yes'
```




```
Has had renovation mean price: 770466.14
Has NOT had renovation mean price: 532976.31
```

In [48]:
```
X_feats = ['bedrooms', 'bathrooms', 'sqft_living',
           'sqft_lot', 'floors', 'waterfront', 'view',
           'condition', 'grade','sqft_above', 'yr_built',
           'sqft_living15', 'sqft_lot15', 'basementyes', 'zipcode',
           'lat', 'long', 'total_rooms', 'living_vs_neighbor',
           'lot_vs_neighbor','live_lot']

new_feat_model = model_summary(df, X_feats, 'price')
sked_show(df, X_feats, new_feat_model)
```

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.710
Model:	OLS	**Adj. R-squared:**	0.710
Method:	Least Squares	**F-statistic:**	2615.
Date:	Mon, 19 Apr 2021	**Prob (F-statistic):**	0.00
Time:	09:30:04	**Log-Likelihood:**	-2.9117e+05
No. Observations:	21387	**AIC:**	5.824e+05
Df Residuals:	21366	**BIC:**	5.825e+05
Df Model:	20		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.449e+07	2.85e+06	4.919	0.000	8.7e+06	2.03e+07

3.1 New Feature Model Conclusion

- R^2 imporved to 0.71
- Basementyes and lot_vs_neighbor are statistically insignificant
- Does not meet assumption of homoscedasticity
- Next step should be to remove outliers and run tests again

4 Z-Score Outlier Removal

- Removing Z-Scores may address my issue of heteroskedacicity
- It will also reduce my right skew because values on the tails will be removed, especially on the right

In [49]:
```
scaler = StandardScaler()
scaler
```

Out[49]:
```
StandardScaler()
```

In [50]:
```
df_scaled = df.copy()
```

In [51]:
```
# Don't need to scale locatation or binary variables

num_cols = ['bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors','view', 'condition', 'grade',
       'sqft_above', 'yr_built','sqft_living15', 'sqft_lot15',
       'total_rooms', 'living_vs_neighbor', 'lot_vs_neighbor', 'live_lo
```

In [52]:
```
df_scaled[num_cols] = scaler.fit_transform(df_scaled[num_cols])
df_scaled
```

Out[52]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	wa
0	7129300520	2014-10-13	221900.00000	-0.41185	-1.45459	-0.98204	-0.22816	-0.91774	
1	6414100192	2014-12-09	538000.00000	-0.41185	0.17194	0.53098	-0.18985	0.93402	
2	5631500400	2015-02-25	180000.00000	-1.51952	-1.45459	-1.42833	-0.12349	-0.91774	
3	2487200875	2014-12-09	604000.00000	0.69582	1.14785	-0.13301	-0.24380	-0.91774	
4	1954400510	2015-02-18	510000.00000	-0.41185	-0.15337	-0.43779	-0.16969	-0.91774	
...	...	...	...	...	...	...	...	...	...
21574	7430200100	2014-05-14	1220000.00000	0.69582	1.79846	3.07807	-0.13687	0.00814	
21577	8672200110	2015-03-17	1090000.00000	1.80349	2.12377	2.27258	-0.16819	0.93402	
21581	191100405	2015-04-21	1580000.00000	0.69582	1.47316	1.44532	-0.12048	0.93402	
21584	249000205	2014-10-15	1540000.00000	1.80349	2.12377	2.59913	-0.16949	0.93402	
21590	7936000429	2015-03-26	1010000.00000	0.69582	1.79846	1.55417	-0.19086	0.93402	

21387 rows × 27 columns

Since everything is either continuous or binary, we don't need to one hot encode any variables at the moment

In [53]:
```
# Run regression with standardized variables. See if there is any diffe
model_summary(df_scaled, X_feats, 'price')
```

OLS Regression Results

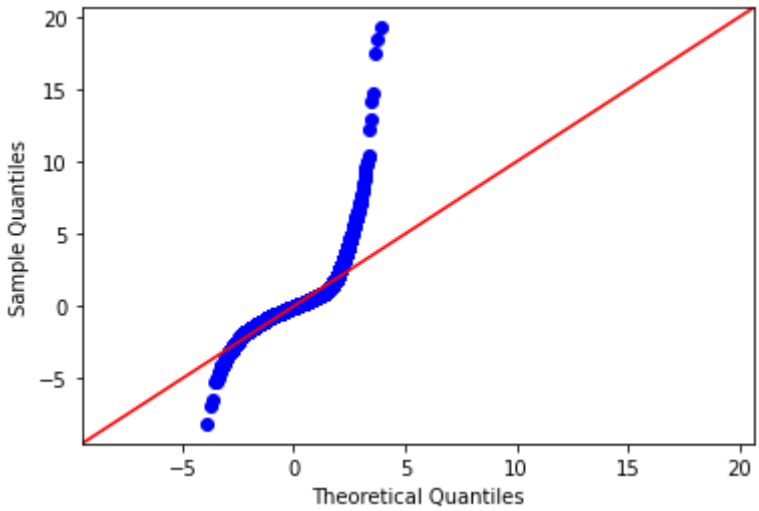
Dep. Variable:	price	**R-squared:**	0.710
Model:	OLS	**Adj. R-squared:**	0.710
Method:	Least Squares	**F-statistic:**	2615.
Date:	Mon, 19 Apr 2021	**Prob (F-statistic):**	0.00
Time:	09:30:04	**Log-Likelihood:**	-2.9117e+05
No. Observations:	21387	**AIC:**	5.824e+05
Df Residuals:	21366	**BIC:**	5.825e+05
Df Model:	20		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	9.616e+06	2.9e+06	3.313	0.001	3.93e+06	1.53e+07
bedrooms	-2.804e+04	1553.990	-18.042	0.000	-3.11e+04	-2.5e+04
bathrooms	3.477e+04	2173.352	15.998	0.000	3.05e+04	3.9e+04
sqft_living	2.568e+05	7330.884	35.024	0.000	2.42e+05	2.71e+05
sqft_lot	7260.9781	2516.633	2.885	0.004	2328.189	1.22e+04
floors	-1.188e+04	2307.488	-5.150	0.000	-1.64e+04	-7361.024
waterfront	6.246e+05	1.79e+04	34.879	0.000	5.89e+05	6.6e+05
view	4.077e+04	1603.600	25.423	0.000	3.76e+04	4.39e+04
condition	1.678e+04	1495.008	11.224	0.000	1.38e+04	1.97e+04
grade	1.133e+05	2514.564	45.038	0.000	1.08e+05	1.18e+05
sqft_above	3.993e+04	5431.143	7.352	0.000	2.93e+04	5.06e+04
yr_built	-8.651e+04	2029.190	-42.634	0.000	-9.05e+04	-8.25e+04
sqft_living15	-8.902e+04	5020.166	-17.733	0.000	-9.89e+04	-7.92e+04
sqft_lot15	-5833.8390	2309.332	-2.526	0.012	-1.04e+04	-1307.376
basementyes	-1708.1717	5109.098	-0.334	0.738	-1.17e+04	8306.044
zipcode	-607.0607	32.742	-18.541	0.000	-671.238	-542.883
lat	5.837e+05	1.07e+04	54.760	0.000	5.63e+05	6.05e+05
long	-1.857e+05	1.31e+04	-14.131	0.000	-2.12e+05	-1.6e+05
total_rooms	962.5428	934.590	1.030	0.303	-869.324	2794.409
living_vs_neighbor	-9.443e+04	3926.571	-24.050	0.000	-1.02e+05	-8.67e+04
lot_vs_neighbor	-934.4996	1767.371	-0.529	0.597	-4398.679	2529.680
live_lot	2.751e+04	2059.569	13.358	0.000	2.35e+04	3.15e+04

Omnibus:	16189.098	**Durbin-Watson:**	1.629
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1091062.914
Skew:	3.067	**Prob(JB):**	0.00
Kurtosis:	37.449	**Cond. No.**	1.40e+18

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 1.05e-22. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

Out[53]: `<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7fd851034700>`



Confirmed that standardized yields the same results as non-standardized

In [54]:
```
# Scale Price and add onto outlier df
df_scaled2 = df_scaled.copy()
```

In [55]:
```
df_scaled2['price'] = scaler.fit_transform(df_scaled2[['price']])
df_scaled2
```

Out[55]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfro
0	7129300520	2014-10-13	-0.86899	-0.41185	-1.45459	-0.98204	-0.22816	-0.91774	0.0000
1	6414100192	2014-12-09	-0.00863	-0.41185	0.17194	0.53098	-0.18985	0.93402	0.0000
2	5631500400	2015-02-25	-0.98304	-1.51952	-1.45459	-1.42833	-0.12349	-0.91774	0.0000
3	2487200875	2014-12-09	0.17101	0.69582	1.14785	-0.13301	-0.24380	-0.91774	0.0000
4	1954400510	2015-02-18	-0.08484	-0.41185	-0.15337	-0.43779	-0.16969	-0.91774	0.0000
...	...	...	...	...	...	...	...	...	
21574	7430200100	2014-05-14	1.84764	0.69582	1.79846	3.07807	-0.13687	0.00814	0.0000
21577	8672200110	2015-03-17	1.49380	1.80349	2.12377	2.27258	-0.16819	0.93402	0.0000
21581	191100405	2015-04-21	2.82749	0.69582	1.47316	1.44532	-0.12048	0.93402	0.0000
21584	249000205	2014-10-15	2.71862	1.80349	2.12377	2.59913	-0.16949	0.93402	0.0000
21590	7936000429	2015-03-26	1.27606	0.69582	1.79846	1.55417	-0.19086	0.93402	0.0000

21387 rows × 27 columns

In [56]:
```
cols_to_check = ['price', 'bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors', 'view', 'condition', 'grade',
       'sqft_above', 'yr_built', 'sqft_living15', 'sqft_lot15',
       'total_rooms', 'living_vs_neighbor', 'lot_vs_neighbor', 'live_lo
```

In [57]:
```
outliers_z = pd.DataFrame()

for col in cols_to_check:
    outliers_z[col] = df_scaled2[col].abs()>3

outliers_z['total'] = outliers_z.any(axis=1)
df_scaledz_orem = df_scaled[~outliers_z['total']].copy()
```

In [58]:
```
df_scaledz_orem.describe()
```

Out[58]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	
count	18739.00000	18739.00000	18739.00000	18739.00000	18739.00000	18739.00000	1873
mean	4642413539.57815	483952.00406	-0.04684	-0.11802	-0.13537	-0.12247	-
std	2864427156.31319	234698.26149	0.93754	0.89297	0.80788	0.26986	
min	1000102.00000	82500.00000	-2.62718	-2.10519	-1.86373	-0.34553	-
25%	2202500202.50000	312000.00000	-0.41185	-0.80398	-0.74257	-0.24249	-
50%	4019301386.00000	434975.00000	-0.41185	-0.15337	-0.25275	-0.18364	-
75%	7348200155.00000	600000.00000	0.69582	0.49724	0.36770	-0.12246	
max	9900000190.00000	1640000.00000	2.91116	2.77438	2.94745	2.87196	

In [59]:
```
print(f'Num observations before dropping with Z-score: {len(df_scaled2)
print('----Dropping all rows where an outlier occurrs accross columns--
print(f'Num observations after dropping with Z-score: {len(df_scaledz_o
print(f'Num observations removed: {len(df_scaled2)-len(df_scaledz_orem)
print(f'Num observations removed as percent of original DF: {round(100*
```

```
Num observations before dropping with Z-score: 21387
----Dropping all rows where an outlier occurrs accross columns----
Num observations after dropping with Z-score: 18739
Num observations removed: 2648
Num observations removed as percent of original DF: 12.38%
```

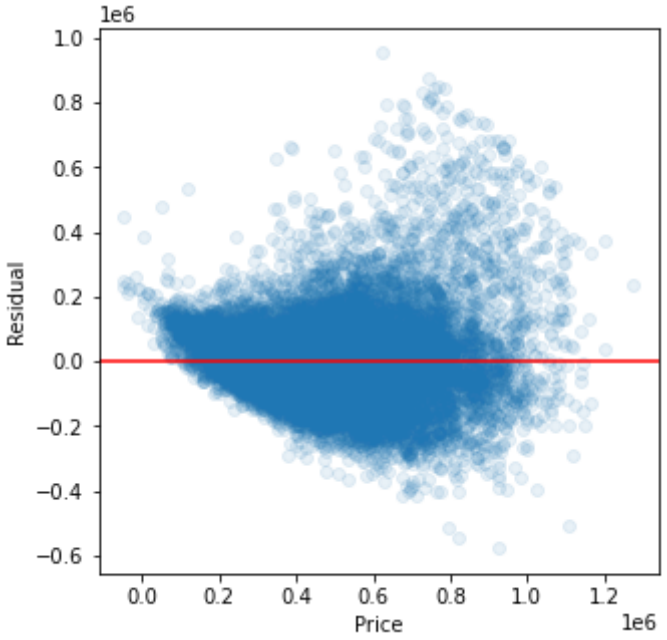
In [60]:
```
print(f"Max price observation: {df_scaledz_orem['price'].max()}")
print(f"Min price observation: {df_scaledz_orem['price'].min()}")
```

```
Max price observation: 1640000.0
Min price observation: 82500.0
```

In [61]:
```
df_scaledz_orem.drop('sqft_basement', axis=1, inplace=True)
```

In [62]:
```
# Run regression to check QQ Plot

X_ztarg = ['bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade'
       'sqft_above', 'yr_built', 'yr_renovated', 'zipcode', 'lat', 'lon
       'sqft_living15', 'sqft_lot15', 'basementyes', 'total_rooms',
       'living_vs_neighbor', 'lot_vs_neighbor', 'live_lot']
```

In [63]:
```
model_scaled = model_summary(df_scaledz_orem, X_ztarg, 'price')
sked_show(df_scaledz_orem, X_ztarg, model_scaled)
```



4.1 Z-Score Outlier Removal Conclusion

- Our R^2 dropped to 0.691
- QQ plot is more normal
- Sqft_lot, floors, sqft_lot15 are insignificant
- Does not achieve homoscedasticity

5 IQR Outlier Removal

- Checking if IQR outlier removal does a better job at normaling QQ plot and achieving homoscedasticity

In [64]:
```
def find_outliers_IQR(data):
    """Detects outliers using the 1.5*IQR thresholds.
    Returns a boolean Series where True=outlier"""
    res = data.describe()
    q1 = res['25%']
    q3 = res['75%']
    thresh = 1.5*(q3-q1)
    idx_outliers =(data < (q1-thresh)) | (data > (q3+thresh))
    return idx_outliers
```

In [65]:
```
df_iqr = df.copy()
```

In [66]:
```
df_iqr.drop(['sqft_basement'], axis=1, inplace=True)
```

In [67]:
```
# Only checking IQR outliers for columns that are continuous/ordinal

iqr_check = ['price', 'bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot','sqft_above', 'yr_built',
       'sqft_living15', 'sqft_lot15', 'total_rooms',
       'living_vs_neighbor', 'lot_vs_neighbor', 'live_lot']
```

In [68]:
```
iqr_outliers = pd.DataFrame()
for col in iqr_check:
    iqr_outliers[col]=find_outliers_IQR(df_iqr[col])
iqr_outliers['total'] = iqr_outliers.any(axis=1)
df_iqr2 = df_iqr[~iqr_outliers['total']].copy()
```

In [69]:
```
print(f'Num observations before dropping with IQR: {len(df_iqr)}')
print(f'Num observations after dropping with IQR: {len(df_iqr2)}')
print(f'Num observations removed: {len(df_iqr)-len(df_iqr2)}')
print(f'Num observations removed as percent of original DF: {(len(df_iq
```

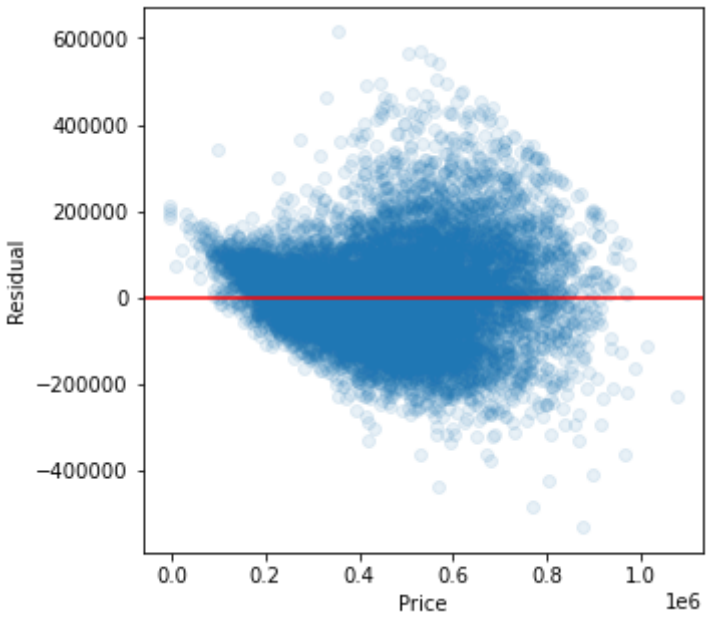
```
Num observations before dropping with IQR: 21387
Num observations after dropping with IQR: 13389
Num observations removed: 7998
Num observations removed as percent of original DF: 0.37%
```

In [70]:
```
print(f"Max price outliers removed: {df_iqr2['price'].max()}")
print(f"Min price outliers removed: {df_iqr2['price'].min()}")
```

```
Max price outliers removed: 1120000.0
Min price outliers removed: 81000.0
```

In [71]:
```
df_iqr2.columns
iqr_pred = ['bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade'
       'sqft_above', 'yr_built', 'yr_renovated', 'zipcode', 'lat', 'lon
       'sqft_living15', 'sqft_lot15', 'basementyes', 'total_rooms',
       'living_vs_neighbor', 'lot_vs_neighbor', 'live_lot', 'renovated_
```

In [72]:
```
model_initiqr = model_summary(df_iqr2, iqr_pred, 'price')
sked_show(df_iqr2, iqr_pred, model_initiqr)
```



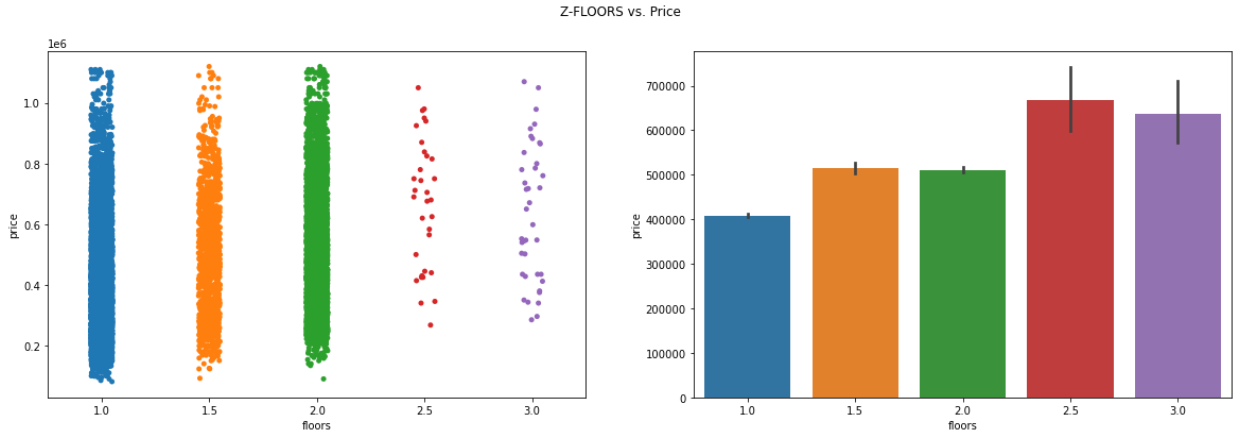
5.1 IQR Outlier Removal Conclusion

- R^2 has increased to 0.71
- sqft_lot15 and living_vs_neighbor are insignificant
- QQ plot is significantly more normal. Small peak towards 2nd quantile
- homoscedasticity has improved because we have removed more outlier variables
- This model is working better but concerned that too much data has been removed (37%)

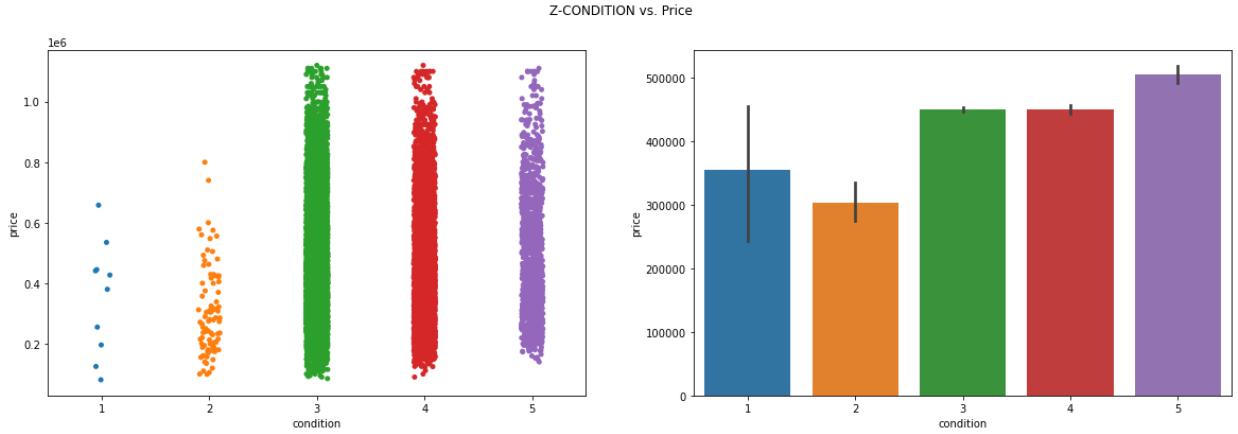
6 Explore OHE Orindal Variables

In [73]:
```
# Does not look like clear linear relationship between ordinal variable
# yr_built does not have a linear relationship
# lot_vs_neighbor does not have a linear relationship

def ordinal_check(df, col, val='price'):
    fig, axes = plt.subplots(ncols=2, figsize=(20,6))
    sns.stripplot(data=df, x=col, y=val, ax=axes[0])
    sns.barplot(data=df, x=col, y=val, ax=axes[1])

    fig.suptitle(f'Z-{col.upper()} vs. Price')
    plt.show()
    print('----------------------------------------------------------------
    print(df[col].value_counts(1))
```

In [74]:
```
for col in ['floors', 'condition']:
    ordinal_check(df_iqr2, col)
```




```
------------------------------------------------------------
1.00000   0.56853
2.00000   0.34155
1.50000   0.08447
3.00000   0.00299
2.50000   0.00246
Name: floors, dtype: float64
```




```
------------------------------------------------------------
3   0.62753
4   0.28494
5   0.07999
2   0.00680
1   0.00075
Name: condition, dtype: float64
```

In [75]:
```
# OHE: Floors/condition
from sklearn.preprocessing import OneHotEncoder
encoder = OneHotEncoder(sparse=False, drop='first')
encoder
```

Out[75]:
```
OneHotEncoder(drop='first', sparse=False)
```

In [76]:
```
cat_cols=['floors', 'condition']
```

In [77]:
```
encoder.fit(df_iqr2[cat_cols])

ohe_vars = encoder.transform(df_iqr2[cat_cols])
encoder.get_feature_names(cat_cols)
cat_vars = pd.DataFrame(ohe_vars,columns=encoder.get_feature_names(cat_
```

In [78]:
```
# Do this for formula OLS

name_dict = {}
for col in cat_vars.columns:
    name_dict[col]=col.replace('.','_')
name_dict
```

Out[78]:
```
{'floors_1.5': 'floors_1_5',
 'floors_2.0': 'floors_2_0',
 'floors_2.5': 'floors_2_5',
 'floors_3.0': 'floors_3_0',
 'condition_2': 'condition_2',
 'condition_3': 'condition_3',
 'condition_4': 'condition_4',
 'condition_5': 'condition_5'}
```

In [79]:
```
cat_vars.rename(columns=name_dict, inplace=True)
cat_vars
```

Out[79]:

	floors_1_5	floors_2_0	floors_2_5	floors_3_0	condition_2	condition_3	condition_4	conditio
0	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	0.00000	0.00(
1	0.00000	1.00000	0.00000	0.00000	0.00000	1.00000	0.00000	0.00(
2	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.00(
3	0.00000	0.00000	0.00000	0.00000	0.00000	1.00000	0.00000	0.00(
4	0.00000	1.00000	0.00000	0.00000	0.00000	1.00000	0.00000	0.00(
...	...	...	...	...	...	...	...	
13384	0.00000	1.00000	0.00000	0.00000	0.00000	1.00000	0.00000	0.00(
13385	0.00000	1.00000	0.00000	0.00000	0.00000	1.00000	0.00000	0.00(
13386	0.00000	1.00000	0.00000	0.00000	0.00000	1.00000	0.00000	0.00(
13387	0.00000	1.00000	0.00000	0.00000	0.00000	1.00000	0.00000	0.00(
13388	0.00000	1.00000	0.00000	0.00000	0.00000	1.00000	0.00000	0.00(

13389 rows × 8 columns

In [80]:
```
df_iqr2 = df_iqr2.reset_index()
```

In [81]:
```
df_iqr2.drop('index', axis=1)
```

Out[81]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	wate
0	7129300520	2014-10-13	221900.00000	3	1.00000	1180	5650	1.00000	0.
1	6414100192	2014-12-09	538000.00000	3	2.25000	2570	7242	2.00000	0.
2	2487200875	2014-12-09	604000.00000	4	3.00000	1960	5000	1.00000	0.
3	1954400510	2015-02-18	510000.00000	3	2.00000	1680	8080	1.00000	0.
4	1321400060	2014-06-27	257500.00000	3	2.25000	1715	6819	2.00000	0.
...	...	...	...	...	...	...	...	...	
13384	5556300109	2014-11-21	1080000.00000	5	3.50000	3230	7560	2.00000	0.
13385	1121000357	2014-08-27	1090000.00000	4	3.00000	3410	6541	2.00000	0.
13386	2428100080	2014-10-01	1060000.00000	4	3.00000	2990	6695	2.00000	0.
13387	8924100308	2015-02-03	1050000.00000	4	2.50000	3260	5974	2.00000	0.
13388	1070000180	2014-10-15	1110000.00000	4	3.50000	3660	4760	2.00000	0.

13389 rows × 26 columns

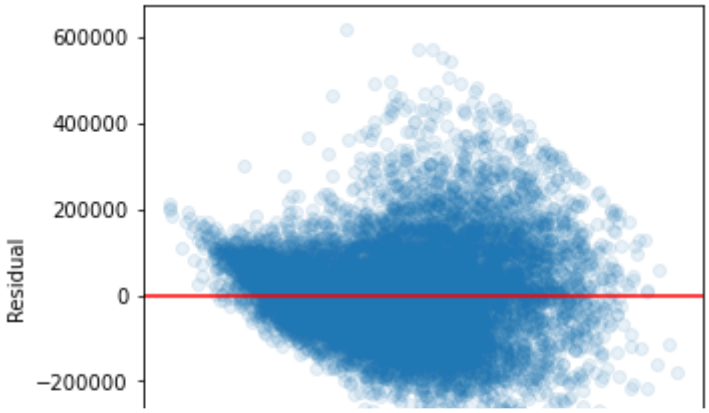
In [82]:
```
df_iqr3 = pd.concat([df_iqr2, cat_vars], axis=1)
```

In [83]:
```
df_iqr3.drop('index', axis=1, inplace=True)
```

In [84]:
```
df_iqr3.drop(['floors', 'condition'], axis=1, inplace=True)
```

In [85]:
```
df_iqr3.columns
X_iqr_ohetargs = ['bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'waterfront', 'view', 'grade', 'sqft_above', 'yr_bui
       'sqft_living15', 'sqft_lot15', 'basementyes', 'lat', 'long', 'zi
       'total_rooms', 'living_vs_neighbor', 'lot_vs_neighbor', 'live_lo
       'renovated_yes', 'floors_1_5', 'floors_2_0', 'floors_2_5', 'floo
       'condition_2', 'condition_3', 'condition_4', 'condition_5']
```

In [86]:
```
model_iqr_ohe = model_summary(df_iqr3, X_iqr_ohetargs, 'price')
sked_show(df_iqr3, X_iqr_ohetargs, model_iqr_ohe)
```



6.1 OHE Ordinal Conclusion

- R^2 of 0.71
- Majority of the OHE variables are not statistically significant so may drop them
- QQ plot looks normal
- Homoscedasticity looks passable

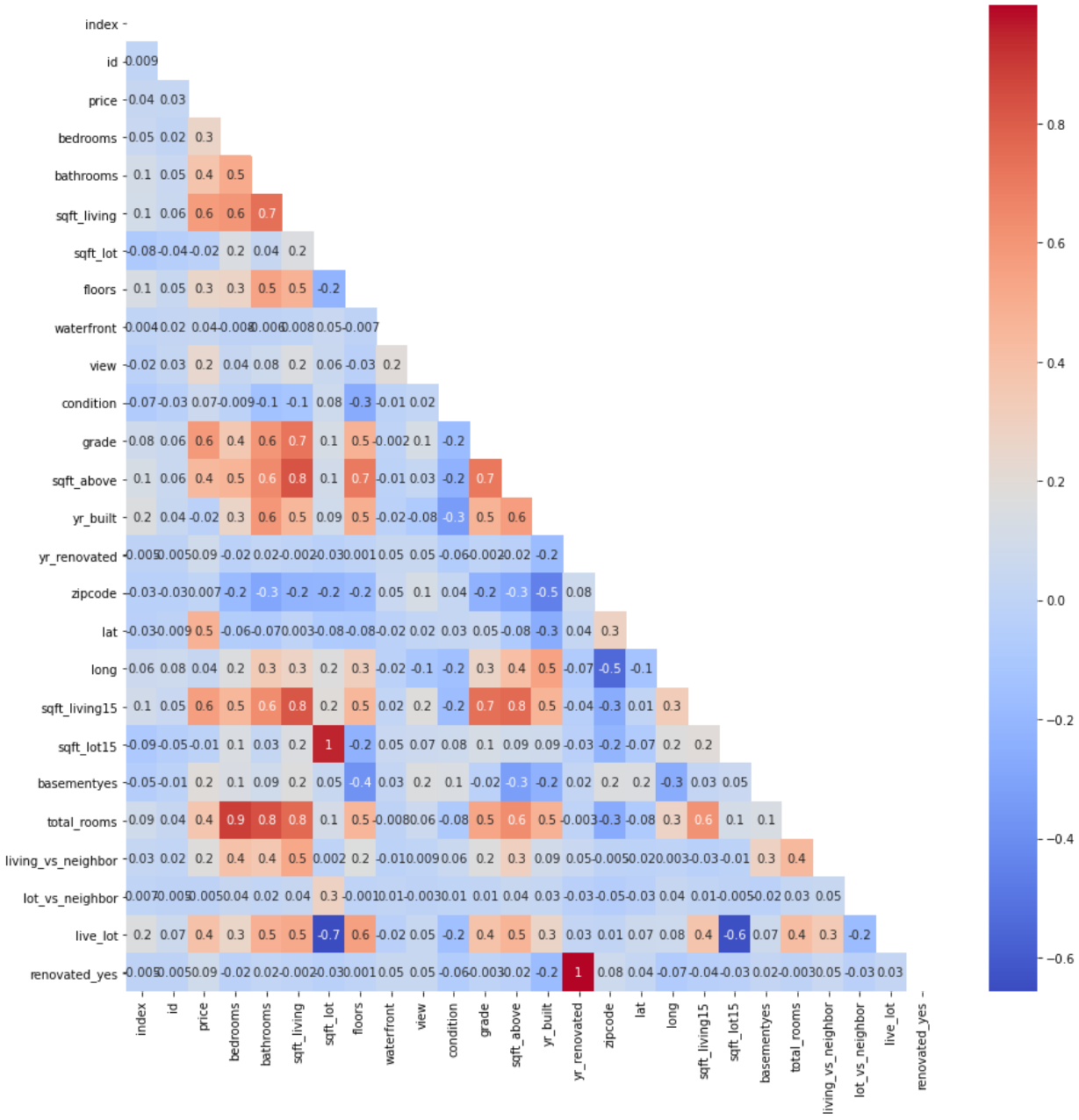
7 Check assumptions of multicollinearity and correlation

In [87]:
```
#https://nbviewer.jupyter.org/github/flatiron-school/Online-DS-FT-02222

df_iqr2.corr()['price'].round(2).sort_values(ascending=False)
```

Out[87]:
```
price                 1.00000
grade                 0.58000
sqft_living           0.57000
sqft_living15         0.56000
lat                   0.48000
sqft_above            0.44000
live_lot              0.40000
bathrooms             0.38000
total_rooms           0.37000
bedrooms              0.27000
floors                0.26000
view                  0.24000
basementyes           0.19000
living_vs_neighbor    0.18000
renovated_yes         0.09000
yr_renovated          0.09000
condition             0.07000
waterfront            0.04000
long                  0.04000
index                 0.04000
id                    0.03000
zipcode               0.01000
sqft_lot15           -0.01000
lot_vs_neighbor      -0.01000
yr_built             -0.02000
sqft_lot             -0.02000
Name: price, dtype: float64
```

In [88]:
```
# https://heartbeat.fritz.ai/seaborn-heatmaps-13-ways-to-customize-corr
corr2 = df_iqr2.corr()

fig, ax = plt.subplots(figsize=(15,15))
matrix = np.triu(corr2)
sns.heatmap(corr2,cmap="coolwarm", annot=True, fmt='.1g', mask=matrix)
```

Out[88]: <AxesSubplot:>



In [89]:
```
def corr_finder(df):
    df_corr = df.corr().abs().stack().reset_index().sort_values(0, asce
    df_corr['pairs'] = list(zip(df_corr.level_0, df_corr.level_1))
    df_corr.set_index(['pairs'], inplace = True)
    df_corr.drop(columns=['level_1', 'level_0'], inplace = True)

    # # cc for correlation coefficient
    df_corr.columns = ['cc']
    df_corr.drop_duplicates(inplace=True)

    return df_corr[(df_corr.cc>.75) & (df_corr.cc<1)]
```

In [90]:
```
#https://github.com/learn-co-curriculum/dsc-multicollinearity-of-featur
corr_finder(df_iqr2)
```

Out[90]:

	cc
pairs	
(yr_renovated, renovated_yes)	0.99997
(sqft_lot15, sqft_lot)	0.95056
(total_rooms, bedrooms)	0.89193
(total_rooms, bathrooms)	0.84303
(sqft_living, sqft_above)	0.82733
(sqft_living15, sqft_living)	0.82134
(sqft_above, sqft_living15)	0.76633
(sqft_living, total_rooms)	0.76241

Methodology will be to check each pair for correlation with price and drop the feature that has a lower correlation with price

- yr_renovated, renovated_yes: DROP - yr_renovated (should be dropped anyway)
- sqft_lot, sqft_lot15: DROP - sqft_lot15
- total_rooms, bedrooms: DROP - total rooms because it takes away from nuance of bath/bed
- bathrooms, total_rooms: DROP - total rooms because it takes away from nuance of bath/bed
- sqft_living, sqft_above: DROP - sqft_above

- sqft_living, sqft_living15: DROP - sqft_living_15
- sqft_above, sqft_living15: DROP - sqft_living_15
- sqft_living, total_rooms: DROP - total_rooms

In [91]:
```
df_iqr_nocolin = df_iqr2.copy()
```

In [92]:
```
cols_to_drop = ['index','yr_renovated', 'sqft_lot15', 'total_rooms', 's
```

In [93]:
```
df_iqr_nocolin.drop(columns=cols_to_drop, axis=1, inplace=True)
```

In [94]:
```
df_iqr_nocolin.columns
X_nocolin_targs = ['bedrooms', 'bathrooms', 'sqft_living', 'zipcode', '
       'sqft_lot', 'waterfront', 'floors','view', 'condition', 'grade',
       'yr_built', 'basementyes', 'living_vs_neighbor', 'lot_vs_neighbo
       'live_lot', 'renovated_yes']
```

In [95]:
```
df_iqr_mult = model_summary(df_iqr_nocolin, X_nocolin_targs, 'price')
sked_show(df_iqr_nocolin, X_nocolin_targs, df_iqr_mult)
```

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.707
Model:	OLS	**Adj. R-squared:**	0.707
Method:	Least Squares	**F-statistic:**	1796.
Date:	Mon, 19 Apr 2021	**Prob (F-statistic):**	0.00
Time:	09:30:08	**Log-Likelihood:**	-1.7390e+05
No. Observations:	13389	**AIC:**	3.478e+05
Df Residuals:	13370	**BIC:**	3.480e+05
Df Model:	18		
Covariance Type:	nonrobust		

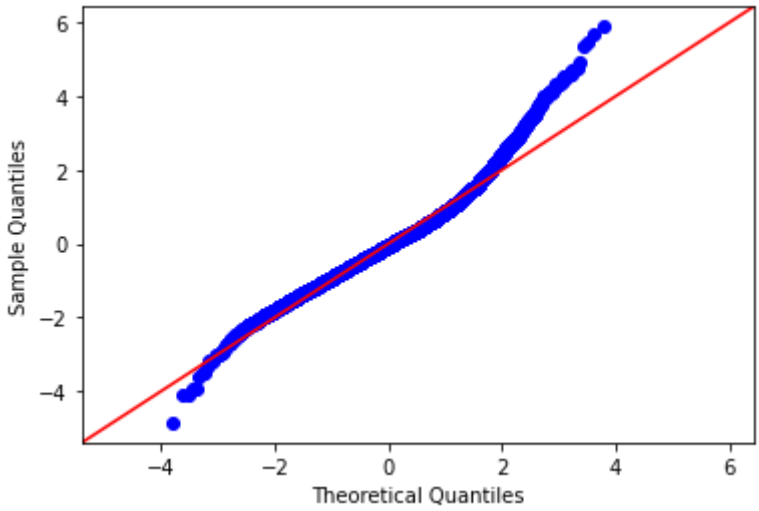
	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.188e+07	2.02e+06	5.895	0.000	7.93e+06	1.58e+07
bedrooms	-7811.2041	1544.522	-5.057	0.000	-1.08e+04	-4783.722
bathrooms	1.61e+04	2529.434	6.366	0.000	1.11e+04	2.11e+04
sqft_living	128.7250	3.609	35.663	0.000	121.650	135.800
zipcode	-300.3712	21.674	-13.859	0.000	-342.855	-257.887
lat	5.292e+05	6992.670	75.679	0.000	5.15e+05	5.43e+05
long	2.932e+04	9237.006	3.174	0.002	1.12e+04	4.74e+04
sqft_lot	-2.5624	0.687	-3.732	0.000	-3.908	-1.216
waterfront	1.724e+05	3.27e+04	5.270	0.000	1.08e+05	2.36e+05
floors	1.334e+04	3129.564	4.262	0.000	7204.789	1.95e+04
view	3.198e+04	1733.906	18.444	0.000	2.86e+04	3.54e+04
condition	2.976e+04	1557.488	19.105	0.000	2.67e+04	3.28e+04
grade	7.191e+04	1617.424	44.460	0.000	6.87e+04	7.51e+04
yr_built	-2232.9777	54.301	-41.122	0.000	-2339.415	-2126.541
basementyes	6756.7809	2560.305	2.639	0.008	1738.220	1.18e+04
living_vs_neighbor	-1.235e+05	5898.948	-20.930	0.000	-1.35e+05	-1.12e+05
lot_vs_neighbor	4.252e+04	8756.572	4.855	0.000	2.54e+04	5.97e+04
live_lot	1.286e+05	1.59e+04	8.080	0.000	9.74e+04	1.6e+05
renovated_yes	3.985e+04	6146.511	6.483	0.000	2.78e+04	5.19e+04

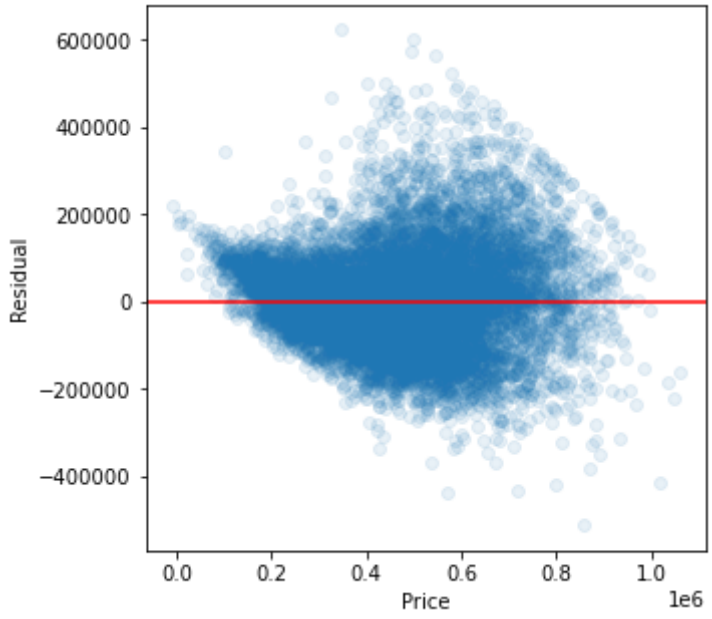
Omnibus:	1366.556	**Durbin-Watson:**	1.870
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2847.472

Skew:	0.653	**Prob(JB):**	0.00
Kurtosis:	4.843	**Cond. No.**	2.17e+08

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.17e+08. This might indicate that there are
strong multicollinearity or other numerical problems.

Out[95]:
```
(<Figure size 360x360 with 1 Axes>,
 <AxesSubplot:xlabel='Price', ylabel='Residual'>)
```



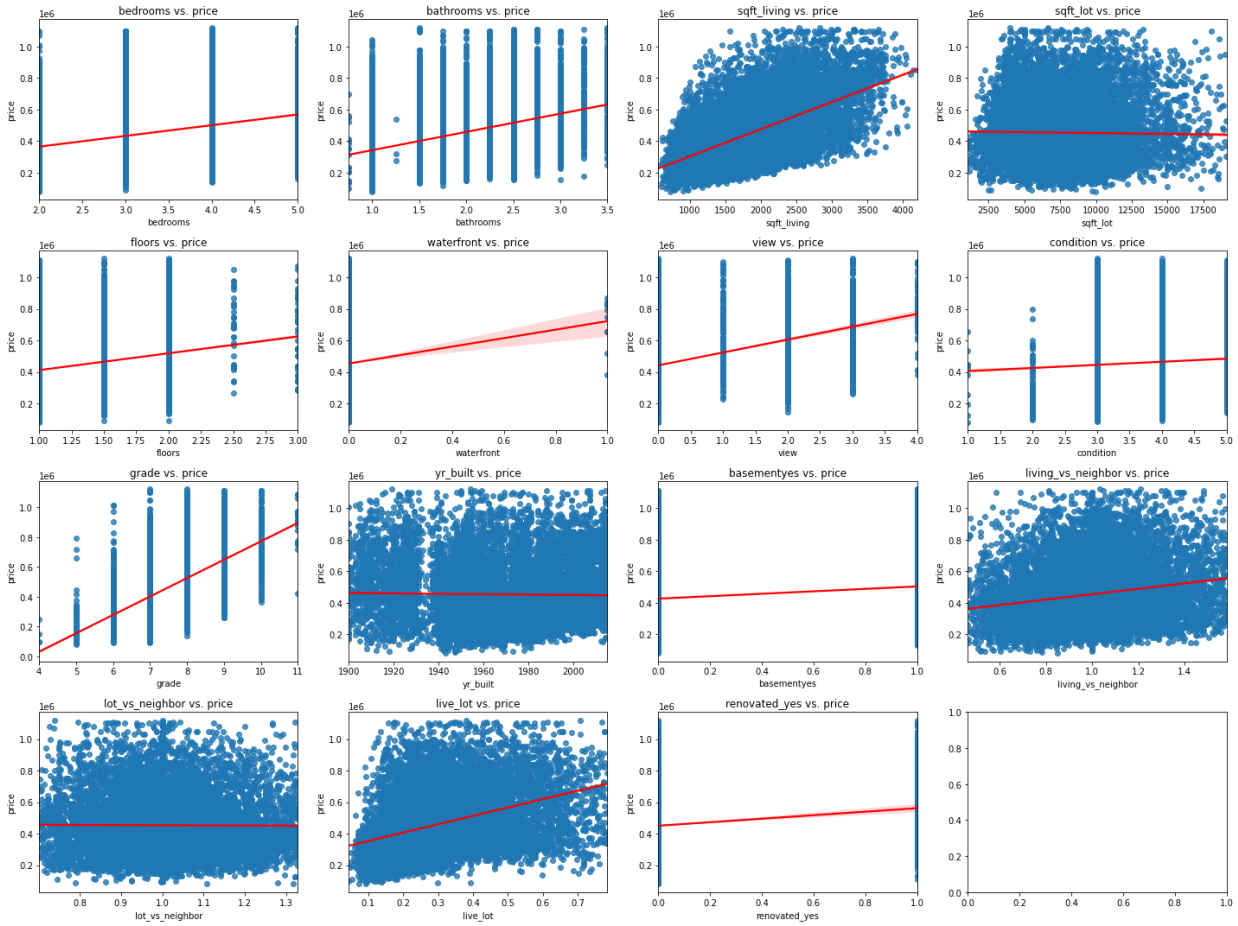


- R^2 of 0.708
- QQ plot looks normal
- No features with insignificant p-values
- homoscedasticity looks passable

Check Assumption of Linearity

In [96]:
```
# Figure out how to delete empty axis

def lin_check(df, cols, ncols=4, figsize=(20,15)):
    fig, axes = plt.subplots(nrows=(len(cols)//ncols) +1, ncols=ncols,
    for ax, col in zip(axes.flatten(), cols):
        sns.regplot(data=df, x=col, y='price', ax=ax, line_kws={'color'
        ax.set_title(f'{col} vs. price')
    fig.tight_layout()
```

In [97]:
```
nocolin_cols = ['bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade'
       'yr_built', 'basementyes', 'living_vs_neighbor', 'lot_vs_neighbo
       'live_lot', 'renovated_yes']
## commenting out for speed

lin_check(df_iqr_nocolin, nocolin_cols)
```



- sqft_lot does not have a linear relationship

- yr_built does not have a linear relationship
- lot_vs_neighbor does not have a linear relationship

In [98]:
```
# comming out for speed
X_iqr_nolin = ['bedrooms', 'bathrooms', 'sqft_living', 'lat', 'long',
               'floors', 'waterfront', 'view', 'condition', 'grade', 'zip
               'basementyes', 'living_vs_neighbor',
               'live_lot', 'renovated_yes']
df_iqr_nocolin = model_summary(df_iqr_nocolin, X_iqr_nolin, 'price')
df_iqr_nocolin
```

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.669
Model:	OLS	**Adj. R-squared:**	0.669
Method:	Least Squares	**F-statistic:**	1805.
Date:	Mon, 19 Apr 2021	**Prob (F-statistic):**	0.00
Time:	09:30:41	**Log-Likelihood:**	-1.7472e+05
No. Observations:	13389	**AIC:**	3.495e+05
Df Residuals:	13373	**BIC:**	3.496e+05
Df Model:	15		
Covariance Type:	nonrobust		

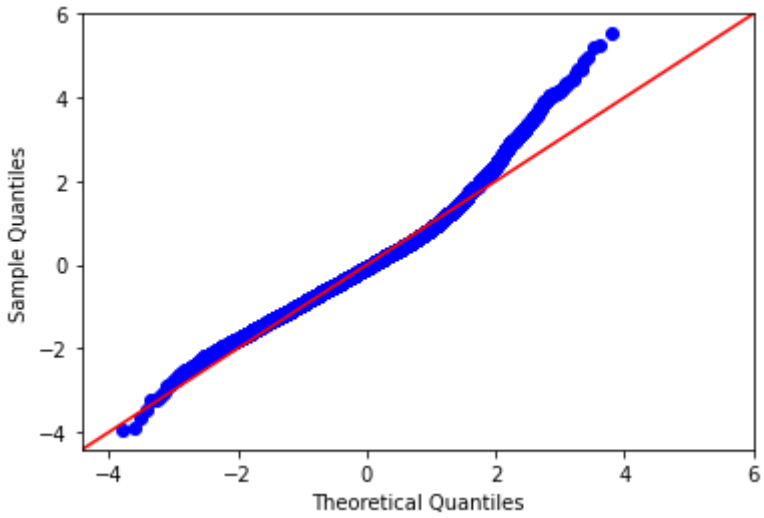
	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-2.072e+07	1.97e+06	-10.521	0.000	-2.46e+07	-1.69e+07
bedrooms	-5231.0322	1639.709	-3.190	0.001	-8445.093	-2016.971
bathrooms	-2.173e+04	2505.852	-8.671	0.000	-2.66e+04	-1.68e+04
sqft_living	124.3866	3.357	37.055	0.000	117.807	130.966
lat	5.949e+05	7234.574	82.237	0.000	5.81e+05	6.09e+05
long	-8.34e+04	9368.570	-8.902	0.000	-1.02e+05	-6.5e+04
floors	1.073e+04	3316.795	3.236	0.001	4232.965	1.72e+04
waterfront	1.523e+05	3.47e+04	4.388	0.000	8.43e+04	2.2e+05
view	3.819e+04	1835.516	20.808	0.000	3.46e+04	4.18e+04
condition	4.881e+04	1577.244	30.945	0.000	4.57e+04	5.19e+04
grade	6.145e+04	1695.715	36.237	0.000	5.81e+04	6.48e+04
zipcode	-184.2113	22.833	-8.068	0.000	-228.966	-139.456
basementyes	1.993e+04	2695.145	7.396	0.000	1.47e+04	2.52e+04
living_vs_neighbor	-1.19e+05	6258.039	-19.016	0.000	-1.31e+05	-1.07e+05
live_lot	1.682e+05	9114.665	18.453	0.000	1.5e+05	1.86e+05
renovated_yes	1.027e+05	6319.250	16.256	0.000	9.03e+04	1.15e+05

Omnibus:	1298.062	**Durbin-Watson:**	1.867
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2325.768

Skew:	0.673	**Prob(JB):**	0.00
Kurtosis:	4.535	**Cond. No.**	1.99e+08

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.99e+08. This might indicate that there are
strong multicollinearity or other numerical problems.

Out[98]: <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7fd873da3a30>



7.1 Linearity Conclusion

- R^2 has dropped to 0.67
- No insignificant p-values
- QQ plot looks normal
- Sked looks normal
- One of the features we dropped must have had a significant impact on our target variable

8 Pivoting Back to Z-Score

- R^2 was reduced
- Testing if I can achieve similar results and drop less data

In [99]:
```
df_z = df_scaledz_orem.copy()
df_z.drop('yr_renovated', axis=1, inplace=True)
```

9 Check for multicolinearity

Not going to OHE the variables that were unsuccesful in IQR method

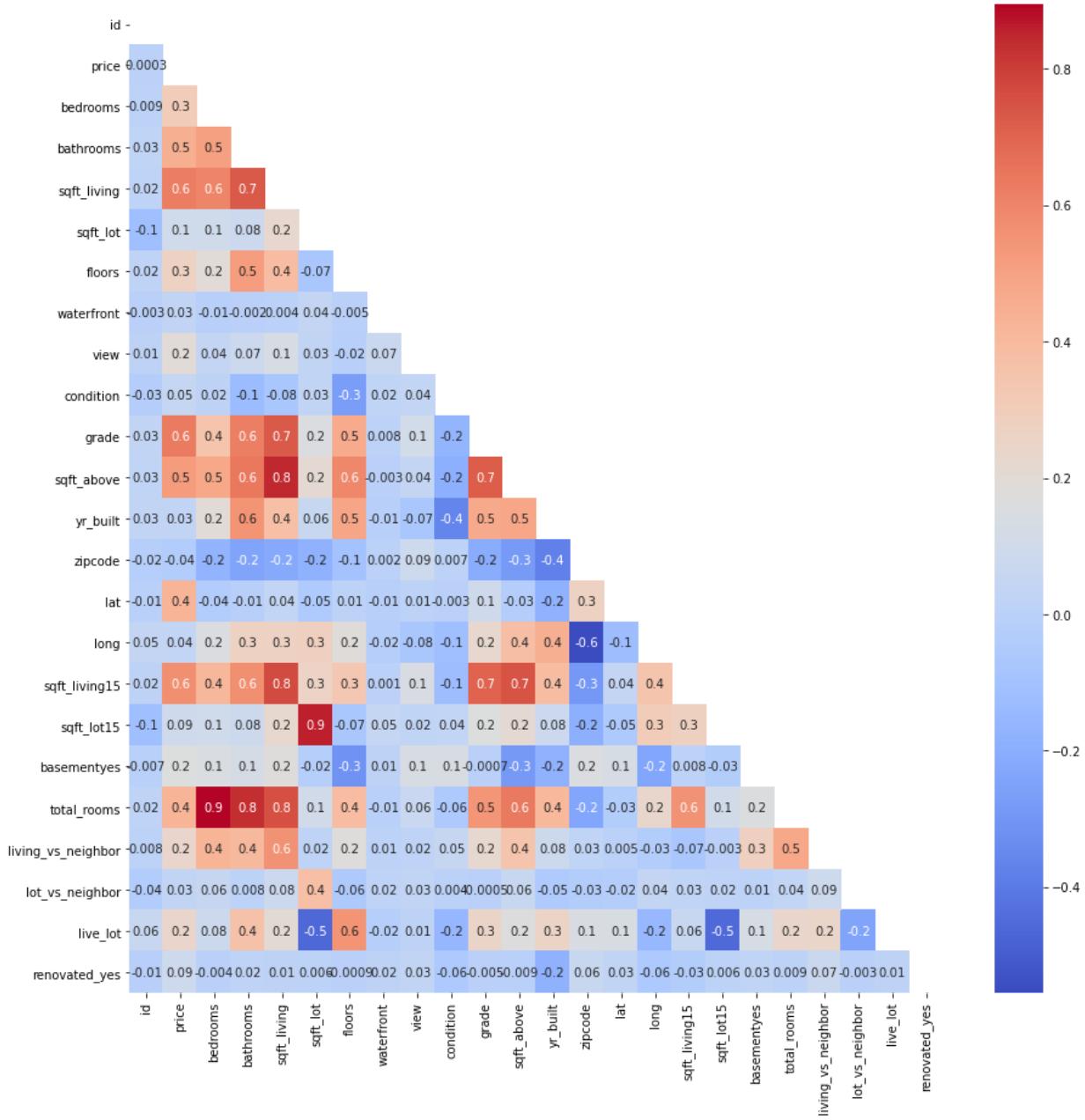
In [100]:
```
# Check correlation
def initial_corr_check(df, col='price'):
    return df_z.corr()['price'].round(2).sort_values(ascending=False)
initial_corr_check(df_z)
```

Out[100]:
```
price                 1.00000
grade                 0.63000
sqft_living           0.62000
sqft_living15         0.57000
sqft_above            0.52000
bathrooms             0.45000
total_rooms           0.43000
lat                   0.43000
bedrooms              0.31000
floors                0.28000
living_vs_neighbor    0.24000
live_lot              0.24000
view                  0.20000
basementyes           0.17000
sqft_lot              0.10000
renovated_yes         0.09000
sqft_lot15            0.09000
condition             0.05000
long                  0.04000
waterfront            0.03000
yr_built              0.03000
lot_vs_neighbor       0.03000
id                    0.00000
zipcode              -0.04000
Name: price, dtype: float64
```

In [101]:
```
def corr_triangle(df):
    corr2 = df.corr()

    fig, ax = plt.subplots(figsize=(15,15))
    matrix = np.triu(corr2)
    return sns.heatmap(corr2,cmap="coolwarm", annot=True, fmt='.1g', ma
```

In [102]:
```
1 corr_triangle(df_z)
```

Out[102]: <AxesSubplot:>



In [103]:
```
# To Drop based on mulitcolinearity:
# total_rooms, bedrooms -DROP total_rooms (nuance)
# sqft_lot, sqft_lot15 -DROP sqft_lot15
# sqft_living, sqft_above -DROP sqft_above
# bathrooms, total_rooms -DROP total_rooms(nuance)
# sqft_living, sqft_living15 -DROP sqft_living15
# total_rooms, sqft_living -DROP total_rooms

corr_finder(df_z)
```

Out[103]:

	cc
pairs	
(bedrooms, total_rooms)	0.89598
(sqft_lot, sqft_lot15)	0.85946
(sqft_above, sqft_living)	0.84780
(total_rooms, bathrooms)	0.83665
(sqft_living15, sqft_living)	0.77120
(total_rooms, sqft_living)	0.75943

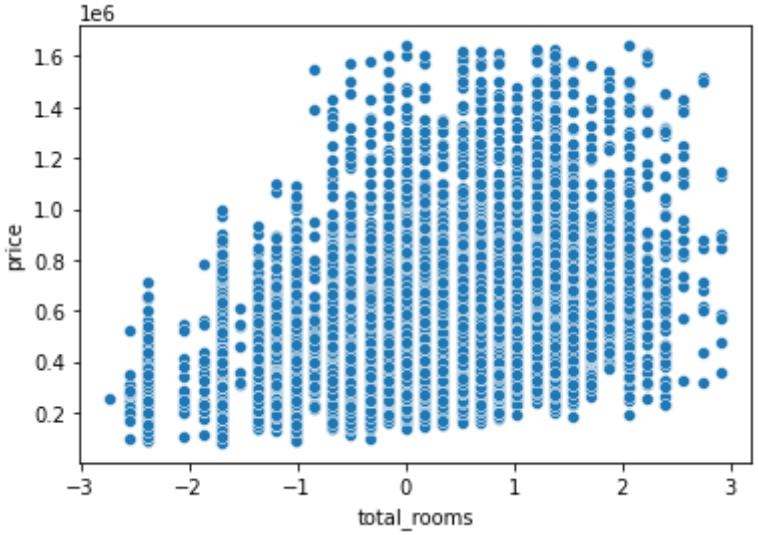
In [104]:
```
# Confirm no more multicolinearity issues

corr_finder(df_z.drop(['total_rooms', 'sqft_lot15', 'sqft_above', 'sqft
```

Out[104]:

cc
pairs

In [105]:
```
# It really doesn't look like there's a linear relationship between tot

sns.scatterplot(data=df_z, x='total_rooms', y='price', ci=68)
```

Out[105]: <AxesSubplot:xlabel='total_rooms', ylabel='price'>



In [106]:
```
df_z_multirem = df_z.copy()
```

In [107]:
```
df_z_multirem.drop(['total_rooms', 'sqft_lot15', 'sqft_above', 'sqft_li
```

In [108]:
```
df_z_multirem.columns
```

Out[108]:
```
Index(['id', 'date', 'price', 'bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade',
       'yr_built', 'zipcode', 'lat', 'long', 'basementyes',
       'living_vs_neighbor', 'lot_vs_neighbor', 'live_lot', 'renovated_ye
s'],
      dtype='object')
```

In [109]:
```
X_multirem_targ = ['bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade'
       'yr_built', 'zipcode', 'lat', 'long', 'basementyes',
       'living_vs_neighbor', 'lot_vs_neighbor', 'live_lot', 'renovated_
```

In [110]:
```
model_z_multirem = model_summary(df_z_multirem, X_multirem_targ, 'price
sked_show(df_z_multirem, X_multirem_targ, model_z_multirem)
```

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.689
Model:	OLS	**Adj. R-squared:**	0.689
Method:	Least Squares	**F-statistic:**	2304.
Date:	Mon, 19 Apr 2021	**Prob (F-statistic):**	0.00
Time:	09:30:42	**Log-Likelihood:**	-2.4737e+05
No. Observations:	18739	**AIC:**	4.948e+05
Df Residuals:	18720	**BIC:**	4.949e+05
Df Model:	18		
Covariance Type:	nonrobust		

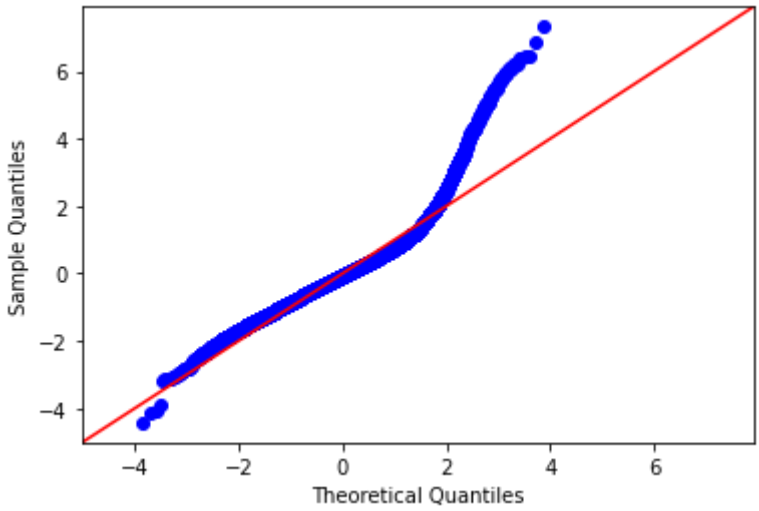
	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	4.836e+06	2.05e+06	2.355	0.019	8.11e+05	8.86e+06
bedrooms	-9365.8393	1340.420	-6.987	0.000	-1.2e+04	-6738.494
bathrooms	1.917e+04	1900.244	10.086	0.000	1.54e+04	2.29e+04
sqft_living	1.285e+05	2639.585	48.673	0.000	1.23e+05	1.34e+05
sqft_lot	1.074e+04	4646.933	2.310	0.021	1628.089	1.98e+04
floors	3089.4827	1657.212	1.864	0.062	-158.803	6337.768
waterfront	2.912e+05	5.88e+04	4.949	0.000	1.76e+05	4.06e+05
view	2.937e+04	1788.635	16.421	0.000	2.59e+04	3.29e+04
condition	1.8e+04	1070.270	16.820	0.000	1.59e+04	2.01e+04
grade	9.959e+04	1841.924	54.068	0.000	9.6e+04	1.03e+05
yr_built	-6.979e+04	1531.873	-45.561	0.000	-7.28e+04	-6.68e+04
zipcode	-379.9658	23.054	-16.481	0.000	-425.154	-334.778
lat	5.451e+05	7431.977	73.346	0.000	5.31e+05	5.6e+05
long	-5.748e+04	9416.926	-6.104	0.000	-7.59e+04	-3.9e+04
basementyes	-5196.2910	2588.381	-2.008	0.045	-1.03e+04	-122.829
living_vs_neighbor	-3.325e+04	1611.279	-20.633	0.000	-3.64e+04	-3.01e+04
lot_vs_neighbor	1.983e+04	3851.877	5.149	0.000	1.23e+04	2.74e+04
live_lot	3.582e+04	1998.492	17.923	0.000	3.19e+04	3.97e+04
renovated_yes	4.118e+04	5907.203	6.970	0.000	2.96e+04	5.28e+04

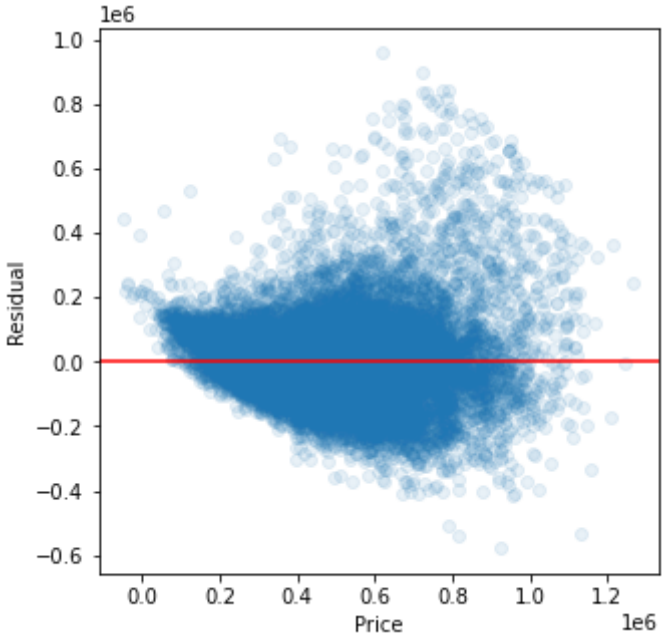
Omnibus:	5313.566	**Durbin-Watson:**	1.469
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	25259.797

Skew:	1.302	**Prob(JB):**	0.00
Kurtosis:	8.057	**Cond. No.**	2.11e+08

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.11e+08. This might indicate that there are
strong multicollinearity or other numerical problems.

Out[110]:
```
(<Figure size 360x360 with 1 Axes>,
 <AxesSubplot:xlabel='Price', ylabel='Residual'>)
```





9.1 Multicolinearity Assumptions Conclusion

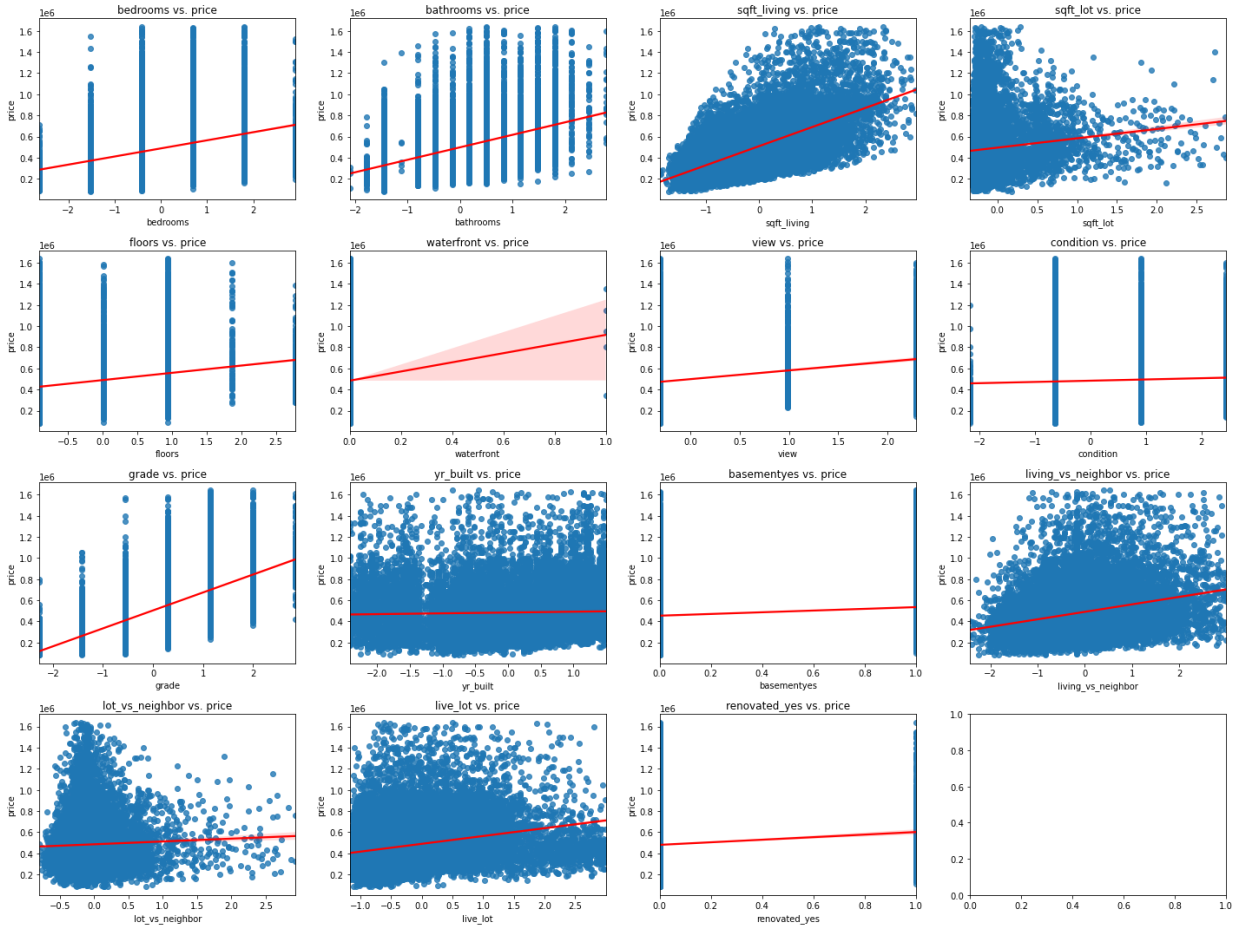
- R^2 is 68.8
- No statistically insignificant features but floors, basement, and sqft loss are close to threshold
- QQ plot seems passable
- Skedacity vears upward as price increases

10 Check for Linearity and possible OHE

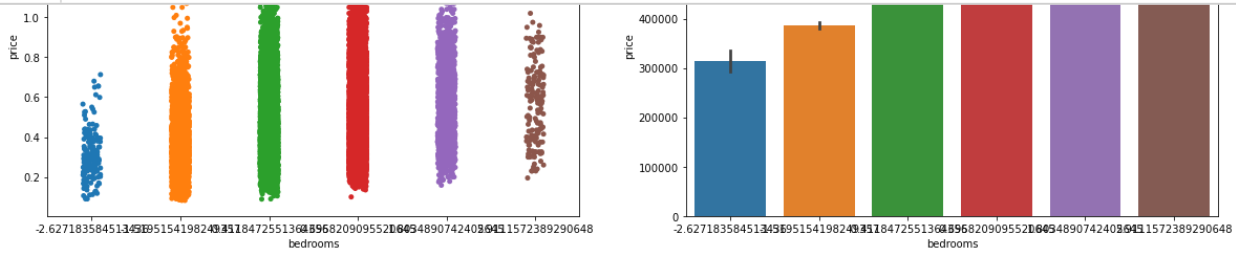
In [111]:
```
1 df_z_multirem.columns
```

Out[111]:
```
Index(['id', 'date', 'price', 'bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade',
       'yr_built', 'zipcode', 'lat', 'long', 'basementyes',
       'living_vs_neighbor', 'lot_vs_neighbor', 'live_lot', 'renovated_ye
s'],
      dtype='object')
```

In [122]:
```
## commenting out for speed
z_multi_check = ['bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade'
       'yr_built', 'basementyes',
       'living_vs_neighbor', 'lot_vs_neighbor', 'live_lot', 'renovated_
lin_check(df_z_multirem, z_multi_check)
```



In [113]:
```
oridnal_cats = ['bedrooms', 'bathrooms', 'floors', 'condition', 'grade'
for col in oridnal_cats:
    print(ordinal_check(df_z_multirem, col))
    print('-----------------------------------------------------------------
```




```
-----------------------------------------------------------------
-0.41185    0.46753
0.69582     0.31901
-1.51952    0.13064
1.80349     0.06569
2.91116     0.00881
-2.62718    0.00832
Name: bedrooms, dtype: float64
None
--------------------------------------------------------------------------------
--------------
```

Most of the data apppears ordinal, floor is in between

In [114]:
```
df_z_loc = df_z_multirem.copy()
```

In [115]:
```
from sklearn.preprocessing import OneHotEncoder
encoder = OneHotEncoder(sparse=False, drop='first')
encoder
loc_col=['zipcode']
encoder.fit(df_z_loc[loc_col])

ohe_vars2 = encoder.transform(df_z_loc[loc_col])
encoder.get_feature_names(loc_col)
cat_vars2 = pd.DataFrame(ohe_vars2,columns=encoder.get_feature_names(lo
```

In [116]:
```
df_z_loc = df_z_loc.reset_index()
```

In [117]:
```
df_z_loc2 = pd.concat([df_z_loc, cat_vars2], axis=1)
```

In [118]:
```
df_z_loc2.drop('zipcode', axis=1, inplace=True)
```

In [119]:
```
X_z_zip = ['bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'waterfront', 'view', 'condition', 'grade',
       'basementyes', 'living_vs_neighbor',
       'live_lot', 'renovated_yes', 'zipcode_98002',
       'zipcode_98003', 'zipcode_98004', 'zipcode_98005', 'zipcode_9800
       'zipcode_98007', 'zipcode_98008', 'zipcode_98010', 'zipcode_9801
       'zipcode_98014', 'zipcode_98019', 'zipcode_98022', 'zipcode_9802
       'zipcode_98024', 'zipcode_98027', 'zipcode_98028', 'zipcode_9802
       'zipcode_98030', 'zipcode_98031', 'zipcode_98032', 'zipcode_9803
       'zipcode_98034', 'zipcode_98038', 'zipcode_98039', 'zipcode_9804
       'zipcode_98042', 'zipcode_98045', 'zipcode_98052', 'zipcode_9805
       'zipcode_98055', 'zipcode_98056', 'zipcode_98058', 'zipcode_9805
       'zipcode_98065', 'zipcode_98070', 'zipcode_98072', 'zipcode_9807
       'zipcode_98075', 'zipcode_98077', 'zipcode_98092', 'zipcode_9810
       'zipcode_98103', 'zipcode_98105', 'zipcode_98106', 'zipcode_9810
       'zipcode_98108', 'zipcode_98109', 'zipcode_98112', 'zipcode_9811
       'zipcode_98116', 'zipcode_98117', 'zipcode_98118', 'zipcode_9811
       'zipcode_98122', 'zipcode_98125', 'zipcode_98126', 'zipcode_9813
       'zipcode_98136', 'zipcode_98144', 'zipcode_98146', 'zipcode_9814
       'zipcode_98155', 'zipcode_98166', 'zipcode_98168', 'zipcode_9817
       'zipcode_98178', 'zipcode_98188', 'zipcode_98198', 'zipcode_9819
```

In [120]:
```
model_zip = model_summary(df_z_loc2, X_z_zip, 'price')
sked_show(df_z_loc2, X_z_zip, model_zip)
```

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.834
Model:	OLS	**Adj. R-squared:**	0.833
Method:	Least Squares	**F-statistic:**	1154.
Date:	Mon, 19 Apr 2021	**Prob (F-statistic):**	0.00
Time:	09:30:46	**Log-Likelihood:**	-2.4151e+05
No. Observations:	18739	**AIC:**	4.832e+05
Df Residuals:	18657	**BIC:**	4.838e+05
Df Model:	81		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	3.33e+05	5232.930	63.635	0.000	3.23e+05	3.43e+05

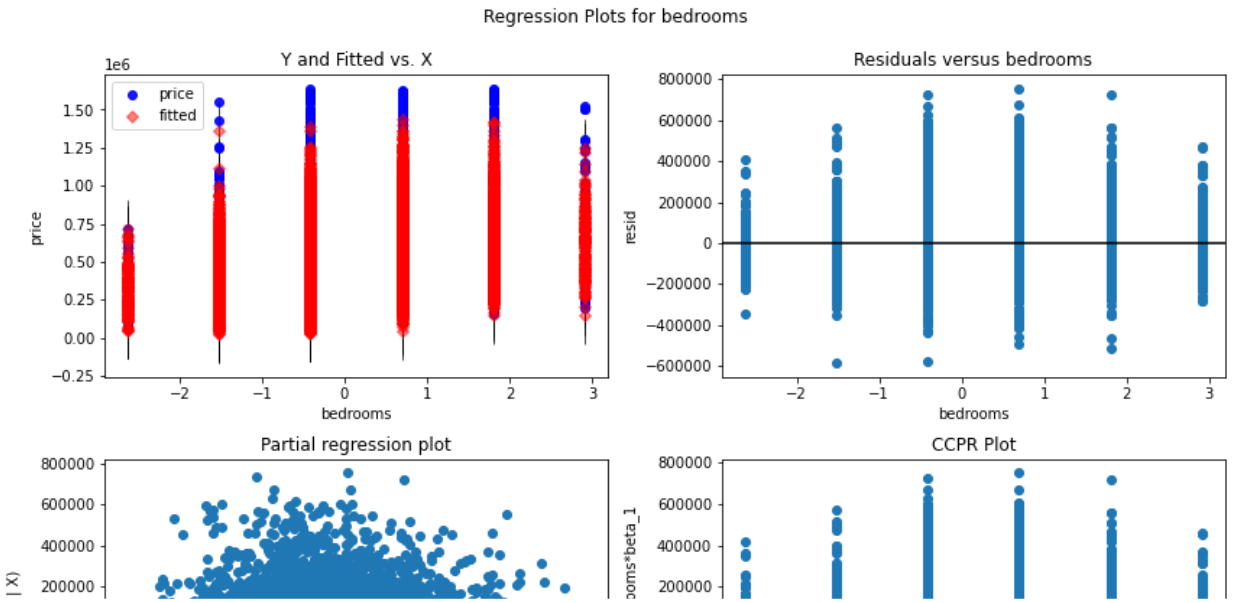
10.1 Checking for Linearity Conclusion

- When adding zipcode, R^2 moves up to 0.834
- Floors are statistically insignificant
- Tried OHE and they came back as majority insignificant
- Dropped floors because no linear relationship
- Some zipcodes were statistically insignificant but enough to drop the variables
- QQ plot trails off 2 quantile
- Homoscedasticity breaks down around $1,000,000

11 Invididually check for homoskedacicity per feature

In [121]:
```
X_z_zip2 = ['bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade'
       'basementyes', 'living_vs_neighbor',
       'live_lot', 'renovated_yes']
for col in X_z_zip2:
    print(f'{col}-----------------')
    sm.graphics.plot_regress_exog(model_zip, col,plt.figure(figsize=(12
    plt.show()
    print('----------------------')
```

```
bedrooms-----------------
```



In [ ]:
```
# Sqft Lot, # live_lot are heteroscedasticstic
```

In [123]:
```
X_z_zip_ho = ['bedrooms', 'bathrooms', 'sqft_living',
       'floors', 'view', 'condition', 'grade',
       'yr_built', 'basementyes', 'living_vs_neighbor',
       'renovated_yes', 'zipcode_98002',
       'zipcode_98003', 'zipcode_98004', 'zipcode_98005', 'zipcode_9800
       'zipcode_98007', 'zipcode_98008', 'zipcode_98010', 'zipcode_9801
       'zipcode_98014', 'zipcode_98019', 'zipcode_98022', 'zipcode_9802
       'zipcode_98024', 'zipcode_98027', 'zipcode_98028', 'zipcode_9802
       'zipcode_98030', 'zipcode_98031', 'zipcode_98032', 'zipcode_9803
       'zipcode_98034', 'zipcode_98038', 'zipcode_98039', 'zipcode_9804
       'zipcode_98042', 'zipcode_98045', 'zipcode_98052', 'zipcode_9805
       'zipcode_98055', 'zipcode_98056', 'zipcode_98058', 'zipcode_9805
       'zipcode_98065', 'zipcode_98070', 'zipcode_98072', 'zipcode_9807
       'zipcode_98075', 'zipcode_98077', 'zipcode_98092', 'zipcode_9810
       'zipcode_98103', 'zipcode_98105', 'zipcode_98106', 'zipcode_9810
       'zipcode_98108', 'zipcode_98109', 'zipcode_98112', 'zipcode_9811
       'zipcode_98116', 'zipcode_98117', 'zipcode_98118', 'zipcode_9811
       'zipcode_98122', 'zipcode_98125', 'zipcode_98126', 'zipcode_9813
       'zipcode_98136', 'zipcode_98144', 'zipcode_98146', 'zipcode_9814
       'zipcode_98155', 'zipcode_98166', 'zipcode_98168', 'zipcode_9817
       'zipcode_98178', 'zipcode_98188', 'zipcode_98198', 'zipcode_9819
```

In [124]:
```
model_ind = model_summary(df_z_loc2, X_z_zip_ho, 'price', True)
sked_show(df_z_loc2, X_z_zip_ho, model_ind)
```

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.832
Model:	OLS	**Adj. R-squared:**	0.831
Method:	Least Squares	**F-statistic:**	1155.
Date:	Mon, 19 Apr 2021	**Prob (F-statistic):**	0.00
Time:	09:35:33	**Log-Likelihood:**	-2.4160e+05
No. Observations:	18739	**AIC:**	4.834e+05
Df Residuals:	18658	**BIC:**	4.840e+05
Df Model:	80		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	3.498e+05	5258.706	66.520	0.000	3.4e+05	3.6e+05

11.1 Skedacicity Individual Conclusion:

- It has not changed much, concern is that too many high outlier values are left in the dataset

12 Going back to IQR because of heteroscedasticity

In [125]:
```
# IQR method is more strict on removing outliers

df_iqr4 = df_iqr.copy()
```

In [126]:
```
df_iqr4.drop('yr_renovated', axis=1, inplace=True)
```

In [127]:
```
# Lower bound is negative

res=df_iqr4['price'].describe()
thresh = res['75%'] -res['25%']
u_bound=res['75%']+1.5*thresh
u_bound

```

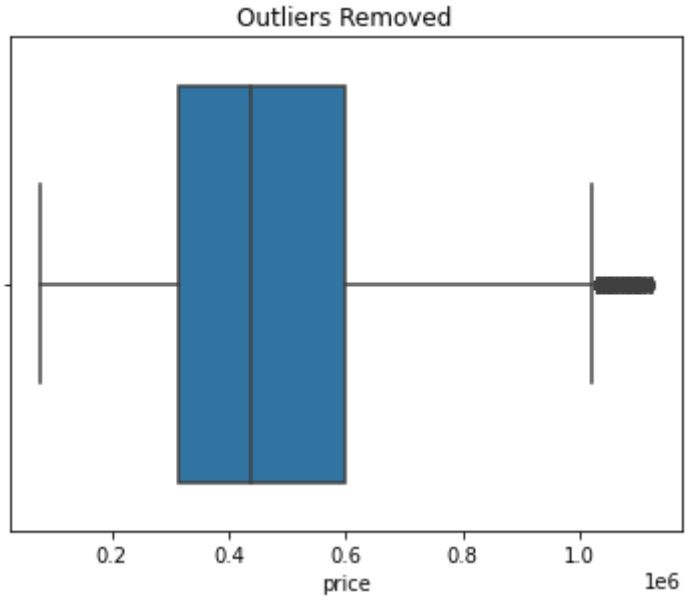
Out[127]: 1125564.75

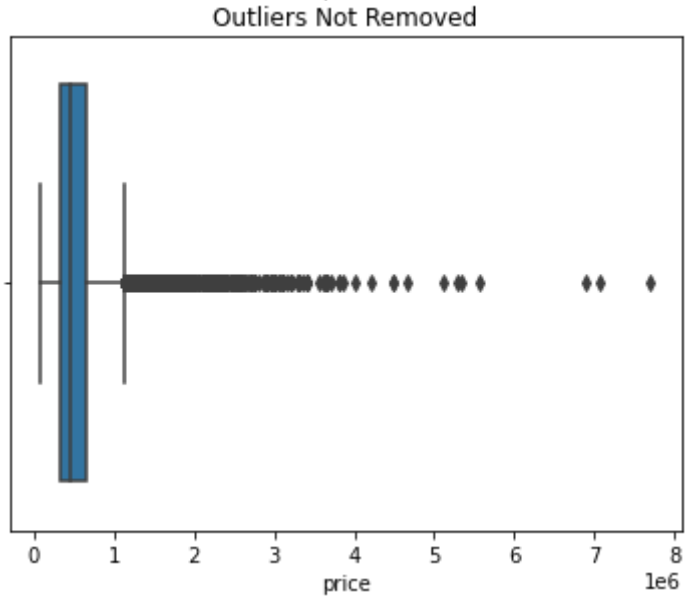
In [128]:
```
df_iqr4['outlier'] = (df_iqr4['price']>u_bound).map({True:True,
                                    False:False})
```

In [129]:
```
df_iqror=df_iqr4.loc[df_iqr4['outlier']==False]
```

In [130]:
```
# Add number of removed values
print(f'Num observations before removal: {len(df)}')
print(f'Num observations after removal:  {len(df_iqror)}')
print(f'Num observations removed:  {len(df) - len(df_iqror)}')
print(f'Percent observations removed:  {round(100*((len(df) - len(df_iq
print('-----------------------------------------')
fig, axes = plt.subplots(nrows=2, figsize=(6,10))
sns.boxplot(data=df_iqror, x='price', ax=axes[0])
axes[0].set_title('Outliers Removed')
sns.boxplot(data=df_iqr4, x='price', ax=axes[1])
axes[1].set_title('Outliers Not Removed')
plt.show();
print('-----------------------------------------')
print(f"Max Home Price: {df_iqror['price'].max()}")
print(f"Min Home Price: {df_iqror['price'].min()}")
```

```
Num observations before removal: 21387
Num observations after removal:  20235
Num observations removed:  1152
Percent observations removed:  5.39%
-----------------------------------------
```






```
-----------------------------------------------
Max Home Price: 1120000.0
Min Home Price: 78000.0
```

13 Baseline Model Price Removal

In [131]:
```
X_targs3= ['bedrooms', 'bathrooms', 'sqft_living', 'lat', 'long', 'zipc
       'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade'
       'sqft_above', 'yr_built', 'sqft_living15',
       'sqft_lot15', 'basementyes', 'total_rooms', 'living_vs_neighbor'
       'lot_vs_neighbor', 'live_lot', 'renovated_yes']
```

In [132]:
```
model_iqr_price = model_summary(df_iqror, X_targs3, 'price', True)
sked_show(df_iqror, X_targs3,model_iqr_price)
```

lat	5.299e+05	6130.485	86.438	0.000	5.18e+05	5.42e+05
long	-2.338e+04	7617.832	-3.069	0.002	-3.83e+04	-8447.572
zipcode	-248.6443	19.146	-12.987	0.000	-286.172	-211.116
sqft_lot	0.2418	0.036	6.685	0.000	0.171	0.313
floors	4169.2359	2541.176	1.641	0.101	-811.676	9150.148
waterfront	1.447e+05	1.66e+04	8.719	0.000	1.12e+05	1.77e+05
view	3.038e+04	1348.375	22.527	0.000	2.77e+04	3.3e+04
condition	2.672e+04	1363.410	19.598	0.000	2.4e+04	2.94e+04
grade	6.898e+04	1292.879	53.355	0.000	6.64e+04	7.15e+04
sqft_above	38.1994	4.180	9.139	0.000	30.006	46.393
yr_built	-1918.8454	43.251	-44.365	0.000	-2003.621	-1834.069
sqft_living15	81.5610	5.075	16.071	0.000	71.613	91.509
sqft_lot15	0.0025	0.050	0.050	0.960	-0.095	0.100

13.1 Baseline IQR Model Conclusion

- While R^2 0.71
- We have homoskedacicity
- QQ model shows very few values tailing off, only past 2.5 quantiles
- Assumptions are better met

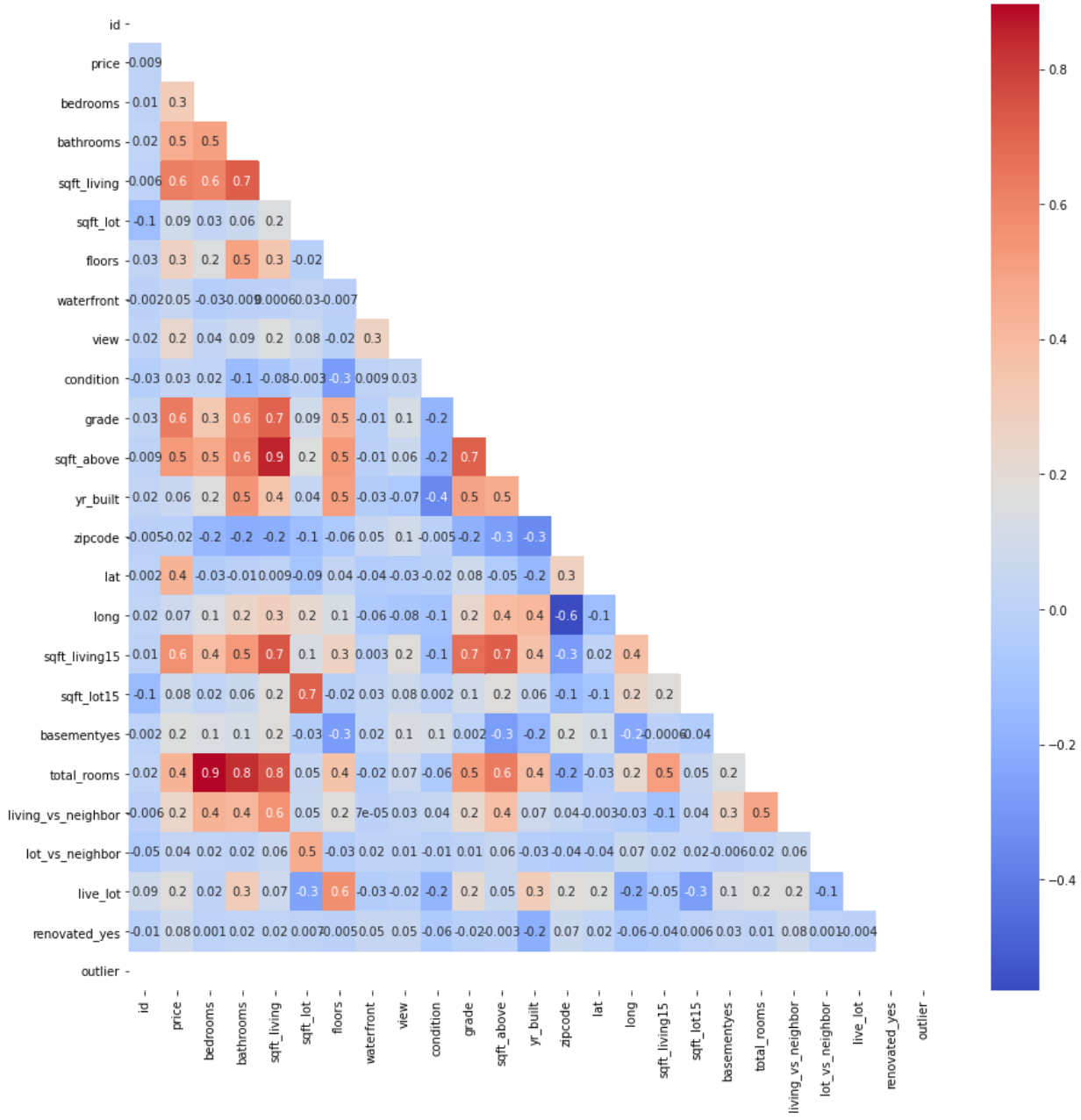
14 Check for Multicolinearity

In [133]:
```
1 df_iqror.corr()['price'].round(2).sort_values(ascending=False)
```

Out[133]:
```
price                 1.00000
grade                 0.63000
sqft_living           0.62000
sqft_living15         0.56000
sqft_above            0.53000
bathrooms             0.45000
lat                   0.43000
total_rooms           0.42000
bedrooms              0.30000
floors                0.27000
living_vs_neighbor    0.24000
view                  0.23000
live_lot              0.17000
basementyes           0.16000
sqft_lot              0.09000
renovated_yes         0.08000
sqft_lot15            0.08000
long                  0.07000
yr_built              0.06000
waterfront            0.05000
lot_vs_neighbor       0.04000
condition             0.03000
id                    0.01000
zipcode              -0.02000
outlier                   nan
Name: price, dtype: float64
```

In [134]:
```
corr2 = df_iqror.corr()

fig, ax = plt.subplots(figsize=(15,15))
matrix = np.triu(corr2)
sns.heatmap(corr2,cmap="coolwarm", annot=True, fmt='.1g', mask=matrix)
```

Out[134]: <AxesSubplot:>



In [135]:
```
# Remove total_rooms, sqft_above

corr_finder(df_iqror)
```

Out[135]:

pairs	cc
(bedrooms, total_rooms)	0.89690
(sqft_living, sqft_above)	0.85335
(bathrooms, total_rooms)	0.83498
(sqft_living, total_rooms)	0.75134

In [136]:
```
# Confirm we don't have multicolinearity

corr_finder(df_iqror.drop(['total_rooms', 'sqft_above'], axis=1))
```

Out[136]:

pairs	cc

In [137]:
```
df_iqror_mc = df_iqror.drop(['total_rooms', 'sqft_above'], axis=1)
```

14.1 Multicolinearity Conclusion

- Had to drop total_rooms and sqft_above because they did not meet assumption of no multicolinearity

In [138]:

```
df_iqror_mc.columns
x_targs = ['bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade'
       'yr_built', 'zipcode', 'lat', 'long', 'sqft_living15', 'sqft_lot
       'basementyes', 'living_vs_neighbor', 'lot_vs_neighbor', 'live_lo
       'renovated_yes']
```

In [139]:
```
model_iqr_co = model_summary(df_iqror_mc, x_targs, 'price', True)
sked_show(df_iqror_mc, x_targs, model_iqr_co)
```

OLS Regression Results

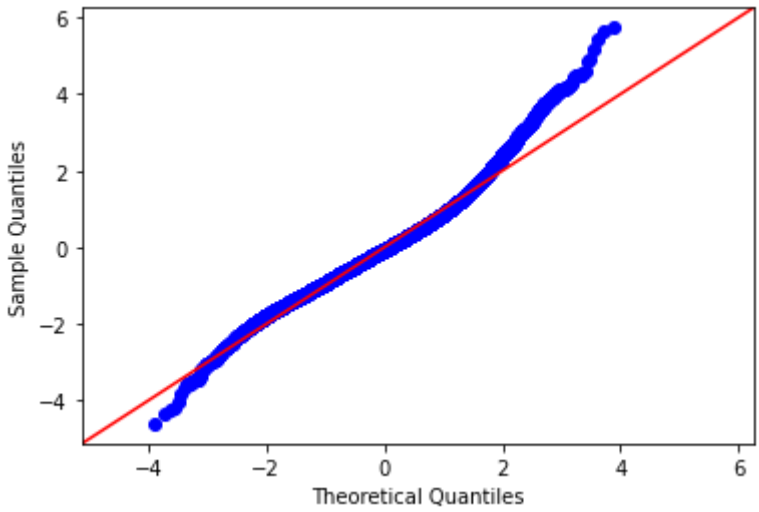
Dep. Variable:	price	**R-squared:**	0.704
Model:	OLS	**Adj. R-squared:**	0.703
Method:	Least Squares	**F-statistic:**	2398.
Date:	Mon, 19 Apr 2021	**Prob (F-statistic):**	0.00
Time:	09:35:59	**Log-Likelihood:**	-2.6405e+05
No. Observations:	20235	**AIC:**	5.282e+05
Df Residuals:	20214	**BIC:**	5.283e+05
Df Model:	20		
Covariance Type:	nonrobust		

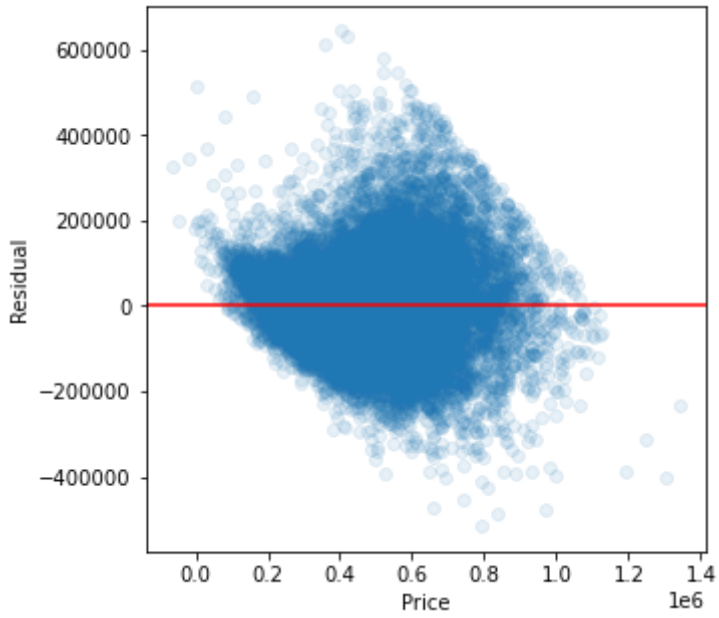
	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-2.307e+05	1.73e+06	-0.134	0.894	-3.61e+06	3.15e+06
bedrooms	-9977.5812	1186.564	-8.409	0.000	-1.23e+04	-7651.819
bathrooms	1.807e+04	1964.441	9.200	0.000	1.42e+04	2.19e+04
sqft_living	44.4406	4.810	9.239	0.000	35.013	53.869
sqft_lot	0.2452	0.036	6.765	0.000	0.174	0.316
floors	1.077e+04	2441.239	4.413	0.000	5987.788	1.56e+04
waterfront	1.481e+05	1.66e+04	8.906	0.000	1.15e+05	1.81e+05
view	2.893e+04	1341.761	21.559	0.000	2.63e+04	3.16e+04
condition	2.561e+04	1360.739	18.820	0.000	2.29e+04	2.83e+04
grade	7.074e+04	1281.161	55.212	0.000	6.82e+04	7.32e+04
yr_built	-1918.4405	43.339	-44.265	0.000	-2003.389	-1833.492
zipcode	-244.4968	19.180	-12.748	0.000	-282.091	-206.903
lat	5.274e+05	6136.610	85.936	0.000	5.15e+05	5.39e+05
long	-1.963e+04	7622.272	-2.575	0.010	-3.46e+04	-4687.387
sqft_living15	86.4843	5.057	17.103	0.000	76.573	96.396
sqft_lot15	-0.0039	0.050	-0.078	0.938	-0.102	0.094
basementyes	-2729.4166	2097.679	-1.301	0.193	-6841.039	1382.205
living_vs_neighbor	5.622e+04	8225.946	6.834	0.000	4.01e+04	7.23e+04
lot_vs_neighbor	1265.3766	832.319	1.520	0.128	-366.037	2896.790
live_lot	8.705e+04	4492.958	19.374	0.000	7.82e+04	9.59e+04
renovated_yes	2.954e+04	4837.659	6.106	0.000	2.01e+04	3.9e+04

Omnibus:	1661.442	**Durbin-Watson:**	1.828
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	3093.886
Skew:	0.578	**Prob(JB):**	0.00
Kurtosis:	4.527	**Cond. No.**	2.19e+08

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.19e+08. This might indicate that there are
strong multicollinearity or other numerical problems.

Out[139]:
```
(<Figure size 360x360 with 1 Axes>,
 <AxesSubplot:xlabel='Price', ylabel='Residual'>)
```





15 Check for assumptions of Linearity

In [140]:
```
df_iqror_mc.columns
```

Out[140]:
```
Index(['id', 'date', 'price', 'bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade',
       'yr_built', 'zipcode', 'lat', 'long', 'sqft_living15', 'sqft_lot1
5',
       'basementyes', 'living_vs_neighbor', 'lot_vs_neighbor', 'live_lo
t',
       'renovated_yes', 'outlier'],
      dtype='object')
```

In [141]:
```
# yr_built, sqft_lot do not have linear relationships

# lin_check(df_iqror_mc, x_targs)
```

In [142]:
```
df_iqr_nocl = df_iqror_mc.drop(['yr_built', 'sqft_lot'], axis=1)
```

In [143]:
```
df_iqr_nocl.columns
x_targs = ['bedrooms', 'bathrooms', 'sqft_living', 'floors',
       'waterfront', 'view', 'condition', 'grade', 'zipcode', 'lat', 'l
       'sqft_living15', 'sqft_lot15', 'basementyes', 'living_vs_neighbo
       'lot_vs_neighbor', 'live_lot', 'renovated_yes']
```

In [144]:
```
model_iqr5 = model_summary(df_iqr_nocl, x_targs, 'price', True)
sked_show(df_iqr_nocl, x_targs, model_iqr5)
```

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.674
Model:	OLS	**Adj. R-squared:**	0.674
Method:	Least Squares	**F-statistic:**	2323.
Date:	Mon, 19 Apr 2021	**Prob (F-statistic):**	0.00
Time:	09:36:07	**Log-Likelihood:**	-2.6501e+05
No. Observations:	20235	**AIC:**	5.301e+05
Df Residuals:	20216	**BIC:**	5.302e+05
Df Model:	18		
Covariance Type:	nonrobust		

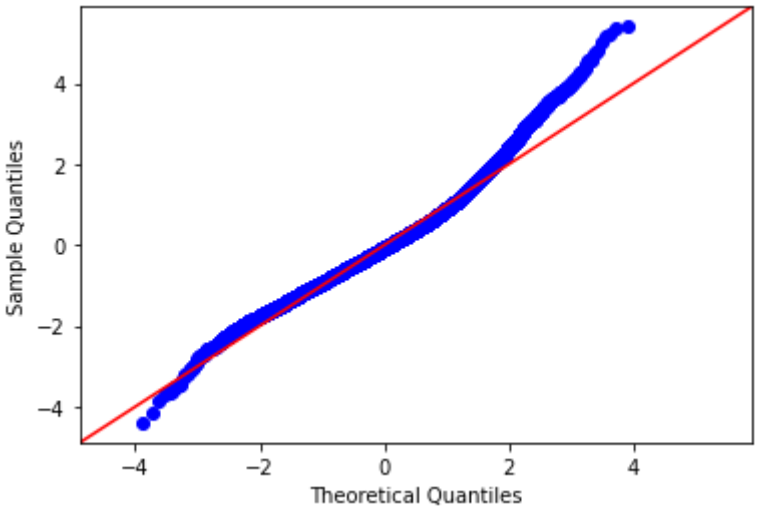
	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-2.695e+07	1.69e+06	-15.912	0.000	-3.03e+07	-2.36e+07
bedrooms	-6927.6524	1241.773	-5.579	0.000	-9361.629	-4493.676
bathrooms	-9820.8440	1950.432	-5.035	0.000	-1.36e+04	-5997.839
sqft_living	52.3617	5.038	10.394	0.000	42.487	62.236
floors	706.8676	2548.024	0.277	0.781	-4287.466	5701.202
waterfront	1.301e+05	1.74e+04	7.469	0.000	9.6e+04	1.64e+05
view	3.423e+04	1400.739	24.437	0.000	3.15e+04	3.7e+04
condition	4.351e+04	1362.956	31.924	0.000	4.08e+04	4.62e+04
grade	5.82e+04	1308.619	44.477	0.000	5.56e+04	6.08e+04
zipcode	-136.6741	19.948	-6.852	0.000	-175.773	-97.575
lat	5.812e+05	6300.897	92.236	0.000	5.69e+05	5.94e+05
long	-1.004e+05	7745.367	-12.965	0.000	-1.16e+05	-8.52e+04
sqft_living15	89.1416	5.297	16.828	0.000	78.759	99.524
sqft_lot15	0.2336	0.034	6.844	0.000	0.167	0.301
basementyes	3002.2427	2195.064	1.368	0.171	-1300.261	7304.747
living_vs_neighbor	6.338e+04	8617.361	7.355	0.000	4.65e+04	8.03e+04
lot_vs_neighbor	6069.4419	678.135	8.950	0.000	4740.242	7398.641
live_lot	5.535e+04	4650.142	11.902	0.000	4.62e+04	6.45e+04
renovated_yes	8.884e+04	4875.206	18.223	0.000	7.93e+04	9.84e+04

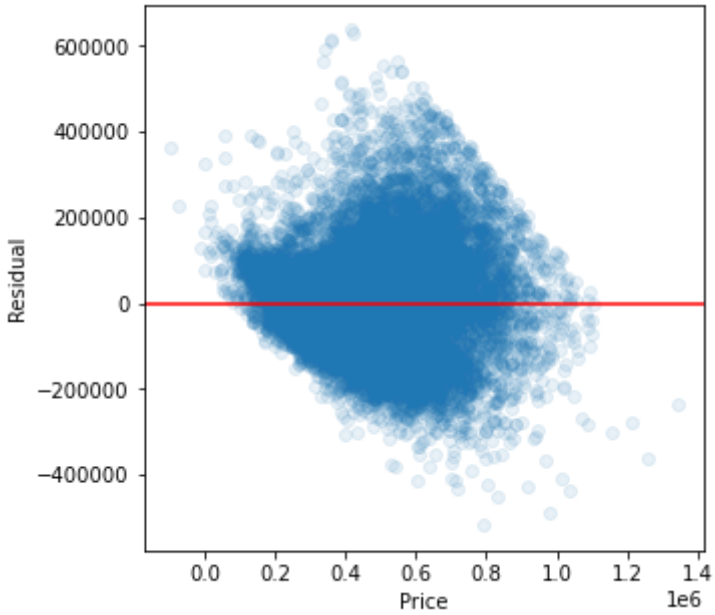
Omnibus:	1713.263	**Durbin-Watson:**	1.831
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2836.776

Skew:	0.631	**Prob(JB):**	0.00
Kurtosis:	4.332	**Cond. No.**	2.02e+08

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.02e+08. This might indicate that there are
strong multicollinearity or other numerical problems.

Out[144]:
```
(<Figure size 360x360 with 1 Axes>,
 <AxesSubplot:xlabel='Price', ylabel='Residual'>)
```





15.1 Checking for assumptions conclusion

- R^2 dropped to 0.674
- All Assumptions have been met

16 Final Model

In [145]:
```
from sklearn.preprocessing import OneHotEncoder
encoder = OneHotEncoder(sparse=False, drop='first')
encoder

cat_cols=['zipcode']

encoder.fit(df_iqr_nocl[cat_cols])

ohe_vars = encoder.transform(df_iqr_nocl[cat_cols])
encoder.get_feature_names(cat_cols)
cat_vars = pd.DataFrame(ohe_vars,columns=encoder.get_feature_names(cat_
```

In [146]:
```
df_iqr_nocl = df_iqr_nocl.reset_index()
```

In [147]:
```
df_iqr_zip = pd.concat([df_iqr_nocl,cat_vars], axis=1)
```

In [148]:
```
df_iqr_zip.columns
x_targs =['bedrooms', 'bathrooms', 'sqft_living',
        'waterfront', 'view', 'condition', 'grade','lat',
       'long', 'sqft_living15', 'sqft_lot15', 'basementyes',
       'living_vs_neighbor', 'lot_vs_neighbor', 'live_lot', 'renovated_
       'zipcode_98002', 'zipcode_98003', 'zipcode_98004',
       'zipcode_98005', 'zipcode_98006', 'zipcode_98007', 'zipcode_9800
       'zipcode_98010', 'zipcode_98011', 'zipcode_98014', 'zipcode_9801
       'zipcode_98022', 'zipcode_98023', 'zipcode_98024', 'zipcode_9802
       'zipcode_98028', 'zipcode_98029', 'zipcode_98030', 'zipcode_9803
       'zipcode_98032', 'zipcode_98033', 'zipcode_98034', 'zipcode_9803
       'zipcode_98039', 'zipcode_98040', 'zipcode_98042', 'zipcode_9804
       'zipcode_98052', 'zipcode_98053', 'zipcode_98055', 'zipcode_9805
       'zipcode_98058', 'zipcode_98059', 'zipcode_98065', 'zipcode_9807
       'zipcode_98072', 'zipcode_98074', 'zipcode_98075', 'zipcode_9807
       'zipcode_98092', 'zipcode_98102', 'zipcode_98103', 'zipcode_9810
       'zipcode_98106', 'zipcode_98107', 'zipcode_98108', 'zipcode_9810
       'zipcode_98112', 'zipcode_98115', 'zipcode_98116', 'zipcode_9811
       'zipcode_98118', 'zipcode_98119', 'zipcode_98122', 'zipcode_9812
       'zipcode_98126', 'zipcode_98133', 'zipcode_98136', 'zipcode_9814
       'zipcode_98146', 'zipcode_98148', 'zipcode_98155', 'zipcode_9816
       'zipcode_98168', 'zipcode_98177', 'zipcode_98178', 'zipcode_9818
       'zipcode_98198', 'zipcode_98199']
```

In [149]:
```
model_iqr6 = model_summary(df_iqr_zip, x_targs, 'price')
sked_show(df_iqr_zip, x_targs, model_iqr6)
```

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.832
Model:	OLS	**Adj. R-squared:**	0.831
Method:	Least Squares	**F-statistic:**	1173.
Date:	Mon, 19 Apr 2021	**Prob (F-statistic):**	0.00
Time:	09:36:12	**Log-Likelihood:**	-2.5832e+05
No. Observations:	20235	**AIC:**	5.168e+05
Df Residuals:	20149	**BIC:**	5.175e+05
Df Model:	85		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-1.861e+07	3.33e+06	-5.587	0.000	-2.51e+07	-1.21e+07
bedrooms	-5191.3639	905.501	-5.733	0.000	-6966.219	-3416.509
bathrooms	9731.4080	1382.333	7.040	0.000	7021.923	1.24e+04
sqft_living	77.5384	3.643	21.286	0.000	70.398	84.678
waterfront	1.661e+05	1.29e+04	12.887	0.000	1.41e+05	1.91e+05
view	2.909e+04	1033.653	28.141	0.000	2.71e+04	3.11e+04
condition	2.526e+04	1003.408	25.174	0.000	2.33e+04	2.72e+04
grade	4.249e+04	967.602	43.912	0.000	4.06e+04	4.44e+04
lat	1.657e+05	3.45e+04	4.803	0.000	9.81e+04	2.33e+05
long	-8.465e+04	2.46e+04	-3.437	0.001	-1.33e+05	-3.64e+04
sqft_living15	62.3583	3.865	16.133	0.000	54.782	69.935
sqft_lot15	0.2089	0.026	7.926	0.000	0.157	0.261
basementyes	-1.634e+04	1406.534	-11.618	0.000	-1.91e+04	-1.36e+04
living_vs_neighbor	4.554e+04	6229.138	7.310	0.000	3.33e+04	5.77e+04
lot_vs_neighbor	4744.3204	489.925	9.684	0.000	3784.027	5704.614
live_lot	-6.138e+04	3081.297	-19.920	0.000	-6.74e+04	-5.53e+04
renovated_yes	4.515e+04	3534.876	12.774	0.000	3.82e+04	5.21e+04
zipcode_98002	1.853e+04	7692.823	2.409	0.016	3450.765	3.36e+04
zipcode_98003	-7230.2698	6875.863	-1.052	0.293	-2.07e+04	6246.983
zipcode_98004	4.655e+05	1.35e+04	34.381	0.000	4.39e+05	4.92e+05
zipcode_98005	2.848e+05	1.36e+04	20.895	0.000	2.58e+05	3.12e+05
zipcode_98006	2.377e+05	1.12e+04	21.189	0.000	2.16e+05	2.6e+05

zipcode_98007	2.221e+05	1.4e+04	15.855	0.000	1.95e+05	2.5e+05
zipcode_98008	2.092e+05	1.33e+04	15.673	0.000	1.83e+05	2.35e+05
zipcode_98010	1.013e+05	1.18e+04	8.604	0.000	7.82e+04	1.24e+05
zipcode_98011	7.759e+04	1.73e+04	4.479	0.000	4.36e+04	1.12e+05
zipcode_98014	8.195e+04	1.91e+04	4.299	0.000	4.46e+04	1.19e+05
zipcode_98019	5.313e+04	1.87e+04	2.837	0.005	1.64e+04	8.98e+04
zipcode_98022	4.39e+04	1.03e+04	4.271	0.000	2.37e+04	6.4e+04
zipcode_98023	-2.93e+04	6343.023	-4.619	0.000	-4.17e+04	-1.69e+04
zipcode_98024	1.432e+05	1.69e+04	8.478	0.000	1.1e+05	1.76e+05
zipcode_98027	1.802e+05	1.14e+04	15.852	0.000	1.58e+05	2.02e+05
zipcode_98028	6.292e+04	1.69e+04	3.734	0.000	2.99e+04	9.6e+04
zipcode_98029	2.175e+05	1.3e+04	16.736	0.000	1.92e+05	2.43e+05
zipcode_98030	2385.9215	7576.150	0.315	0.753	-1.25e+04	1.72e+04
zipcode_98031	1573.9315	7899.391	0.199	0.842	-1.39e+04	1.71e+04
zipcode_98032	-1.357e+04	9159.000	-1.482	0.138	-3.15e+04	4378.045
zipcode_98033	2.691e+05	1.46e+04	18.462	0.000	2.41e+05	2.98e+05
zipcode_98034	1.275e+05	1.55e+04	8.206	0.000	9.7e+04	1.58e+05
zipcode_98038	4.836e+04	8561.822	5.648	0.000	3.16e+04	6.51e+04
zipcode_98039	5.829e+05	4.41e+04	13.206	0.000	4.96e+05	6.69e+05
zipcode_98040	3.745e+05	1.2e+04	31.248	0.000	3.51e+05	3.98e+05
zipcode_98042	1.1e+04	7267.257	1.514	0.130	-3242.256	2.52e+04
zipcode_98045	1.222e+05	1.59e+04	7.688	0.000	9.1e+04	1.53e+05
zipcode_98052	2.083e+05	1.48e+04	14.116	0.000	1.79e+05	2.37e+05
zipcode_98053	1.984e+05	1.58e+04	12.519	0.000	1.67e+05	2.29e+05
zipcode_98055	3.079e+04	8854.784	3.477	0.001	1.34e+04	4.81e+04
zipcode_98056	7.972e+04	9641.386	8.269	0.000	6.08e+04	9.86e+04
zipcode_98058	2.699e+04	8357.216	3.229	0.001	1.06e+04	4.34e+04
zipcode_98059	8.624e+04	9451.794	9.125	0.000	6.77e+04	1.05e+05
zipcode_98065	1.281e+05	1.47e+04	8.724	0.000	9.93e+04	1.57e+05
zipcode_98070	7.312e+04	1.13e+04	6.488	0.000	5.1e+04	9.52e+04
zipcode_98072	1.124e+05	1.73e+04	6.507	0.000	7.85e+04	1.46e+05
zipcode_98074	1.886e+05	1.4e+04	13.508	0.000	1.61e+05	2.16e+05
zipcode_98075	2.08e+05	1.35e+04	15.453	0.000	1.82e+05	2.34e+05
zipcode_98077	1.166e+05	1.8e+04	6.475	0.000	8.13e+04	1.52e+05
zipcode_98092	-7030.2118	6872.977	-1.023	0.306	-2.05e+04	6441.385
zipcode_98102	3.84e+05	1.52e+04	25.310	0.000	3.54e+05	4.14e+05

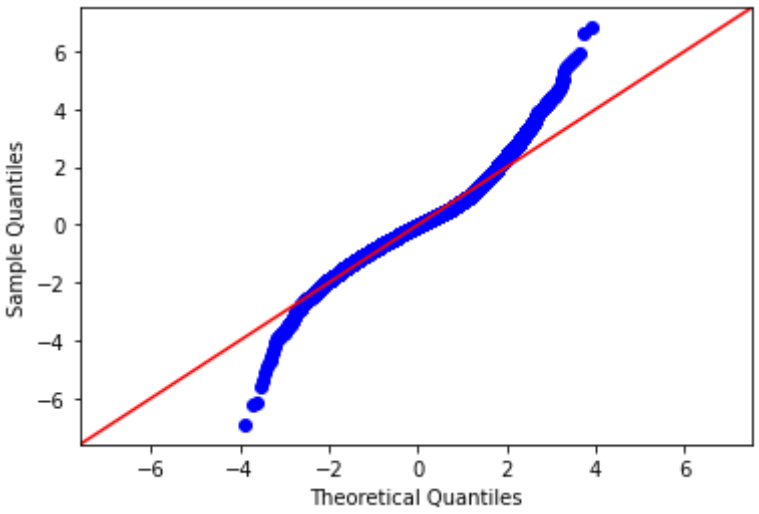
zipcode_98103	2.769e+05	1.39e+04	19.856	0.000	2.5e+05	3.04e+05
zipcode_98105	3.406e+05	1.46e+04	23.363	0.000	3.12e+05	3.69e+05
zipcode_98106	9.241e+04	1.03e+04	8.960	0.000	7.22e+04	1.13e+05
zipcode_98107	2.773e+05	1.44e+04	19.301	0.000	2.49e+05	3.05e+05
zipcode_98108	8.946e+04	1.13e+04	7.899	0.000	6.73e+04	1.12e+05
zipcode_98109	3.81e+05	1.53e+04	24.911	0.000	3.51e+05	4.11e+05
zipcode_98112	3.966e+05	1.37e+04	29.041	0.000	3.7e+05	4.23e+05
zipcode_98115	2.737e+05	1.42e+04	19.276	0.000	2.46e+05	3.02e+05
zipcode_98116	2.617e+05	1.15e+04	22.707	0.000	2.39e+05	2.84e+05
zipcode_98117	2.637e+05	1.44e+04	18.361	0.000	2.36e+05	2.92e+05
zipcode_98118	1.432e+05	1e+04	14.271	0.000	1.24e+05	1.63e+05
zipcode_98119	3.696e+05	1.42e+04	25.999	0.000	3.42e+05	3.97e+05
zipcode_98122	2.849e+05	1.25e+04	22.844	0.000	2.6e+05	3.09e+05
zipcode_98125	1.339e+05	1.54e+04	8.720	0.000	1.04e+05	1.64e+05
zipcode_98126	1.627e+05	1.05e+04	15.424	0.000	1.42e+05	1.83e+05
zipcode_98133	8.63e+04	1.59e+04	5.444	0.000	5.52e+04	1.17e+05
zipcode_98136	2.263e+05	1.08e+04	20.917	0.000	2.05e+05	2.47e+05
zipcode_98144	2.199e+05	1.16e+04	18.888	0.000	1.97e+05	2.43e+05
zipcode_98146	8.429e+04	9684.186	8.704	0.000	6.53e+04	1.03e+05
zipcode_98148	3.963e+04	1.3e+04	3.043	0.002	1.41e+04	6.52e+04
zipcode_98155	6.844e+04	1.65e+04	4.147	0.000	3.61e+04	1.01e+05
zipcode_98166	7.88e+04	8856.313	8.897	0.000	6.14e+04	9.62e+04
zipcode_98168	2.358e+04	9312.752	2.532	0.011	5329.196	4.18e+04
zipcode_98177	1.423e+05	1.66e+04	8.552	0.000	1.1e+05	1.75e+05
zipcode_98178	3.171e+04	9628.280	3.293	0.001	1.28e+04	5.06e+04
zipcode_98188	1.383e+04	9810.860	1.410	0.159	-5401.499	3.31e+04
zipcode_98198	1.002e+04	7462.495	1.343	0.179	-4604.450	2.46e+04
zipcode_98199	3.118e+05	1.38e+04	22.668	0.000	2.85e+05	3.39e+05

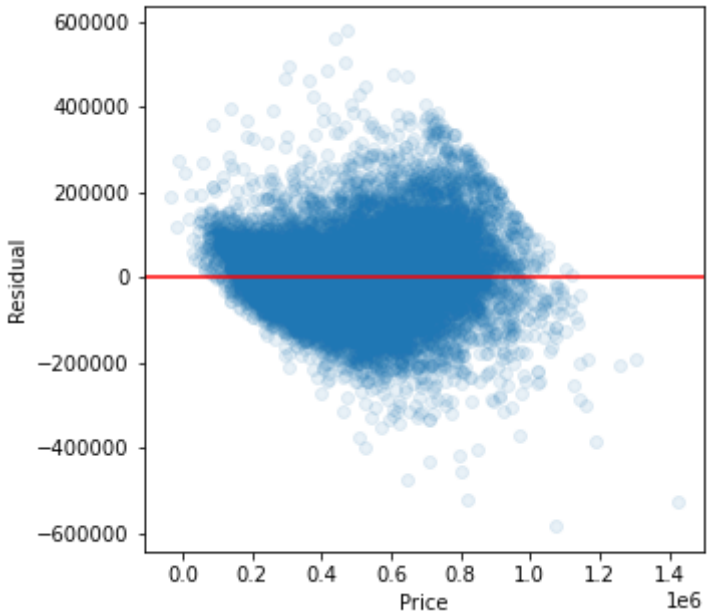
Omnibus:	1938.018	**Durbin-Watson:**	1.807
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	6906.951
Skew:	0.458	**Prob(JB):**	0.00
Kurtosis:	5.712	**Cond. No.**	1.64e+08

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

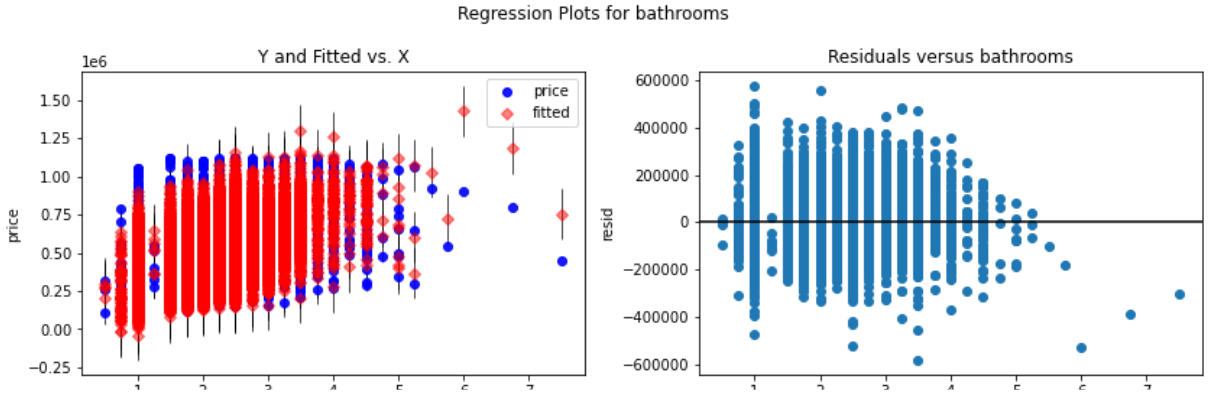
[2] The condition number is large, 1.64e+08. This might indicate that there are strong multicollinearity or other numerical problems.

Out[149]: (<Figure size 360x360 with 1 Axes>,
<AxesSubplot:xlabel='Price', ylabel='Residual'>)





In [150]:
```
exog_check =['bedrooms', 'bathrooms', 'sqft_living',
        'waterfront', 'view', 'condition', 'grade','lat',
       'long', 'sqft_living15', 'sqft_lot15', 'basementyes',
       'living_vs_neighbor', 'lot_vs_neighbor', 'live_lot', 'renovated_

for col in exog_check:
    print(f'{col}------------------')
    sm.graphics.plot_regress_exog(model_iqr6, col,plt.figure(figsize=(1
    plt.show()
    print('----------------------')
```



16.1 Final Model Conclusion

- R^2 0.832
- All assumptions have been met
- Using 20,235 observations

17 Final Visuals

In [151]:
```
df_iqr_zip2 = df_iqr_zip.copy()
cols_to_scale = ['bedrooms', 'bathrooms', 'sqft_living',
       'floors', 'waterfront', 'view', 'condition', 'grade', 'zipcode',
       'long', 'sqft_living15', 'sqft_lot15', 'basementyes',
       'living_vs_neighbor', 'lot_vs_neighbor', 'live_lot', 'renovated_
```

In [152]:
```
df_iqr_zip2[cols_to_scale] = scaler.fit_transform(df_iqr_zip2[cols_to_s

```

In [153]:
```
x_targs2 = ['bedrooms', 'bathrooms', 'sqft_living',
       'waterfront', 'view', 'condition', 'grade',
       'sqft_living15', 'sqft_lot15', 'basementyes',
       'living_vs_neighbor', 'lot_vs_neighbor', 'live_lot', 'renovated_
        'zipcode_98002', 'zipcode_98003', 'zipcode_98004',
       'zipcode_98005', 'zipcode_98006', 'zipcode_98007', 'zipcode_9800
       'zipcode_98010', 'zipcode_98011', 'zipcode_98014', 'zipcode_9801
       'zipcode_98022', 'zipcode_98023', 'zipcode_98024', 'zipcode_9802
       'zipcode_98028', 'zipcode_98029', 'zipcode_98030', 'zipcode_9803
       'zipcode_98032', 'zipcode_98033', 'zipcode_98034', 'zipcode_9803
       'zipcode_98039', 'zipcode_98040', 'zipcode_98042', 'zipcode_9804
       'zipcode_98052', 'zipcode_98053', 'zipcode_98055', 'zipcode_9805
       'zipcode_98058', 'zipcode_98059', 'zipcode_98065', 'zipcode_9807
       'zipcode_98072', 'zipcode_98074', 'zipcode_98075', 'zipcode_9807
       'zipcode_98092', 'zipcode_98102', 'zipcode_98103', 'zipcode_9810
       'zipcode_98106', 'zipcode_98107', 'zipcode_98108', 'zipcode_9810
       'zipcode_98112', 'zipcode_98115', 'zipcode_98116', 'zipcode_9811
       'zipcode_98118', 'zipcode_98119', 'zipcode_98122', 'zipcode_9812
       'zipcode_98126', 'zipcode_98133', 'zipcode_98136', 'zipcode_9814
       'zipcode_98146', 'zipcode_98148', 'zipcode_98155', 'zipcode_9816
       'zipcode_98168', 'zipcode_98177', 'zipcode_98178', 'zipcode_9818
       'zipcode_98198', 'zipcode_98199']
```

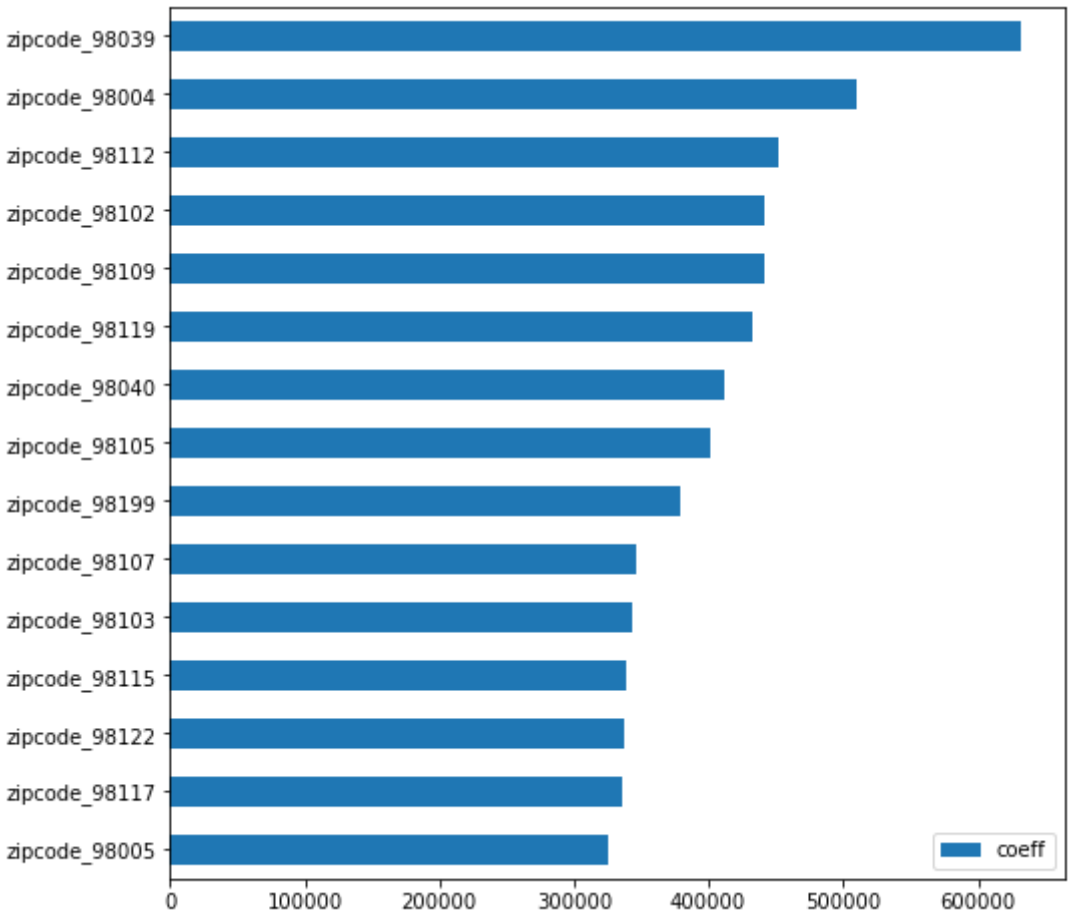
In [154]:
```
model_iqr7 = model_summary(df_iqr_zip, x_targs2, 'price')
```

zipcode_98011	1.454e+05	7595.278	19.150	0.000	1.31e+05	1.6e+05
zipcode_98014	1.069e+05	9107.159	11.734	0.000	8.9e+04	1.25e+05
zipcode_98019	9.751e+04	7668.803	12.715	0.000	8.25e+04	1.13e+05
zipcode_98022	4831.3082	7259.685	0.665	0.506	-9398.268	1.91e+04
zipcode_98023	-2.154e+04	5918.246	-3.640	0.000	-3.31e+04	-9940.739
zipcode_98024	1.537e+05	1.11e+04	13.866	0.000	1.32e+05	1.75e+05
zipcode_98027	1.959e+05	6283.250	31.177	0.000	1.84e+05	2.08e+05
zipcode_98028	1.346e+05	6788.111	19.831	0.000	1.21e+05	1.48e+05
zipcode_98029	2.363e+05	6650.370	35.539	0.000	2.23e+05	2.49e+05
zipcode_98030	5444.5847	6981.721	0.780	0.435	-8240.159	1.91e+04
zipcode_98031	1.034e+04	6848.540	1.510	0.131	-3082.276	2.38e+04
zipcode_98032	-2235.4165	8894.295	-0.251	0.802	-1.97e+04	1.52e+04

In [155]:
```
df_coeff=pd.DataFrame({'coeff': model_iqr7.params, 'abs_coeff': abs(mod
```

In [156]:
```
1 df_coeff.drop('Intercept', axis=0).sort_values(by='abs_coeff').tail(15)
```

Out[156]: <AxesSubplot:>



In [157]:
```
1 df_coeff2 = df_coeff.copy()
```

In [158]:
```
df_coeff2.index
indices = ['Intercept', 'bedrooms', 'bathrooms', 'sqft_living', 'waterf
       'view', 'condition', 'grade', 'sqft_living15', 'sqft_lot15',
       'basementyes', 'living_vs_neighbor', 'lot_vs_neighbor', 'live_lo
       'renovated_yes', 'zipcode_98002', 'zipcode_98003', 'zipcode_9800
       'zipcode_98005', 'zipcode_98006', 'zipcode_98007', 'zipcode_9800
       'zipcode_98010', 'zipcode_98011', 'zipcode_98014', 'zipcode_9801
       'zipcode_98022', 'zipcode_98023', 'zipcode_98024', 'zipcode_9802
       'zipcode_98028', 'zipcode_98029', 'zipcode_98030', 'zipcode_9803
       'zipcode_98032', 'zipcode_98033', 'zipcode_98034', 'zipcode_9803
       'zipcode_98039', 'zipcode_98040', 'zipcode_98042', 'zipcode_9804
       'zipcode_98052', 'zipcode_98053', 'zipcode_98055', 'zipcode_9805
       'zipcode_98058', 'zipcode_98059', 'zipcode_98065', 'zipcode_9807
       'zipcode_98072', 'zipcode_98074', 'zipcode_98075', 'zipcode_9807
       'zipcode_98092', 'zipcode_98102', 'zipcode_98103', 'zipcode_9810
       'zipcode_98106', 'zipcode_98107', 'zipcode_98108', 'zipcode_9810
       'zipcode_98112', 'zipcode_98115', 'zipcode_98116', 'zipcode_9811
       'zipcode_98118', 'zipcode_98119', 'zipcode_98122', 'zipcode_9812
       'zipcode_98126', 'zipcode_98133', 'zipcode_98136', 'zipcode_9814
       'zipcode_98146', 'zipcode_98148', 'zipcode_98155', 'zipcode_9816
       'zipcode_98168', 'zipcode_98177', 'zipcode_98178', 'zipcode_9818
       'zipcode_98198', 'zipcode_98199']
```

In [159]:
```
non_zip = []
for ind in indices:
    if ind.startswith('zip') or ind.startswith('Int'):
        pass
    else:
        non_zip.append(ind)
```

In [160]:
```
df_coeff2.drop('Intercept', axis=0)
```

Out[160]:

	coeff	abs_coeff
bedrooms	-5183.10492	5183.10492
bathrooms	9577.20714	9577.20714
sqft_living	77.41501	77.41501
waterfront	165803.09342	165803.09342
view	29086.11252	29086.11252
condition	25170.41549	25170.41549
grade	42599.51744	42599.51744
sqft_living15	62.52671	62.52671
sqft_lot15	0.20025	0.20025
basementyes	-16094.58374	16094.58374
living_vs_neighbor	45381.65079	45381.65079

In [161]:
```
zips = []
for ind in indices:
    if ind.startswith('zip'):
        zips.append(ind)
    else:
        pass
zips
```

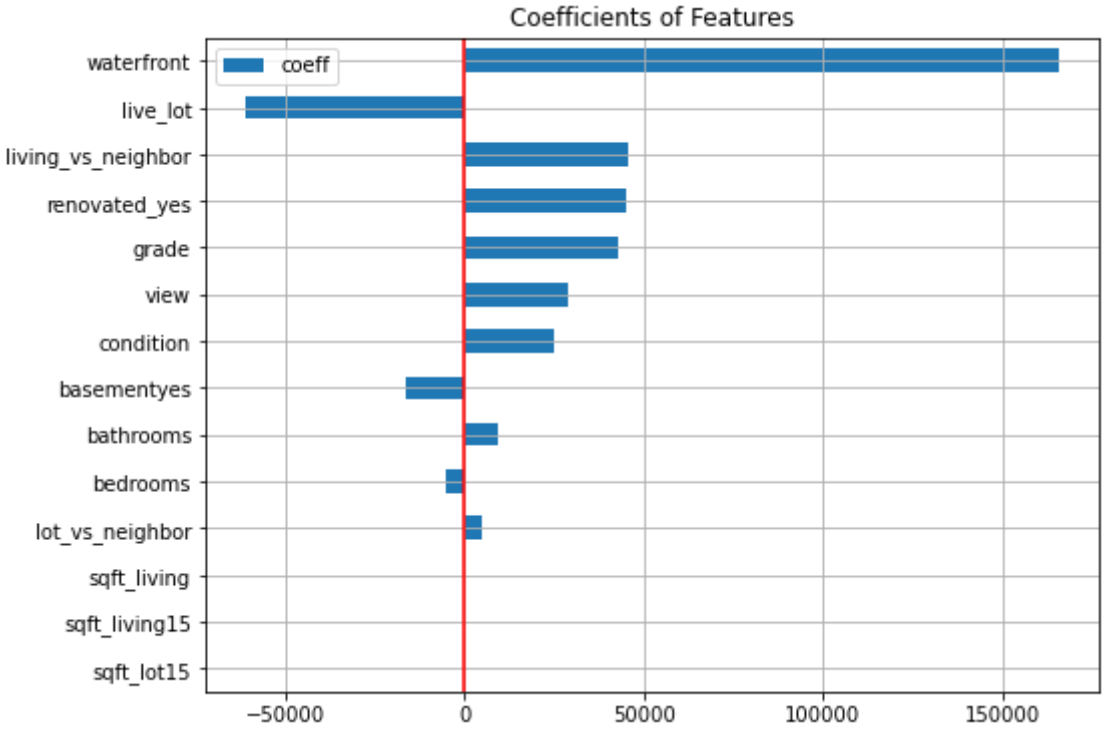
Out[161]:
```
['zipcode_98002',
 'zipcode_98003',
 'zipcode_98004',
 'zipcode_98005',
 'zipcode_98006',
 'zipcode_98007',
 'zipcode_98008',
 'zipcode_98010',
 'zipcode_98011',
 'zipcode_98014',
 'zipcode_98019',
 'zipcode_98022',
 'zipcode_98023',
 'zipcode_98024',
 'zipcode_98027',
 'zipcode_98028',
 'zipcode_98029',
 'zipcode_98030',
 'zipcode_98031',
 'zipcode_98032',
 'zipcode_98033',
 'zipcode_98034',
 'zipcode_98038',
 'zipcode_98039',
 'zipcode_98040',
 'zipcode_98042',
 'zipcode_98045',
 'zipcode_98052',
 'zipcode_98053',
 'zipcode_98055',
 'zipcode_98056',
 'zipcode_98058',
 'zipcode_98059',
 'zipcode_98065',
 'zipcode_98070',
 'zipcode_98072',
 'zipcode_98074',
 'zipcode_98075',
 'zipcode_98077',
 'zipcode_98092',
 'zipcode_98102',
 'zipcode_98103',
 'zipcode_98105',
 'zipcode_98106',
 'zipcode_98107',
 'zipcode_98108',
 'zipcode_98109',
 'zipcode_98112',
 'zipcode_98115',
```

```
'zipcode_98116',
'zipcode_98117',
'zipcode_98118',
'zipcode_98119',
'zipcode_98122',
'zipcode_98125',
'zipcode_98126',
'zipcode_98133',
'zipcode_98136',
'zipcode_98144',
'zipcode_98146',
'zipcode_98148',
'zipcode_98155',
'zipcode_98166',
'zipcode_98168',
'zipcode_98177',
'zipcode_98178',
'zipcode_98188',
'zipcode_98198',
'zipcode_98199']
```

In [162]:
```
df_coeff3 = df_coeff2.drop((zips), axis=0)
```

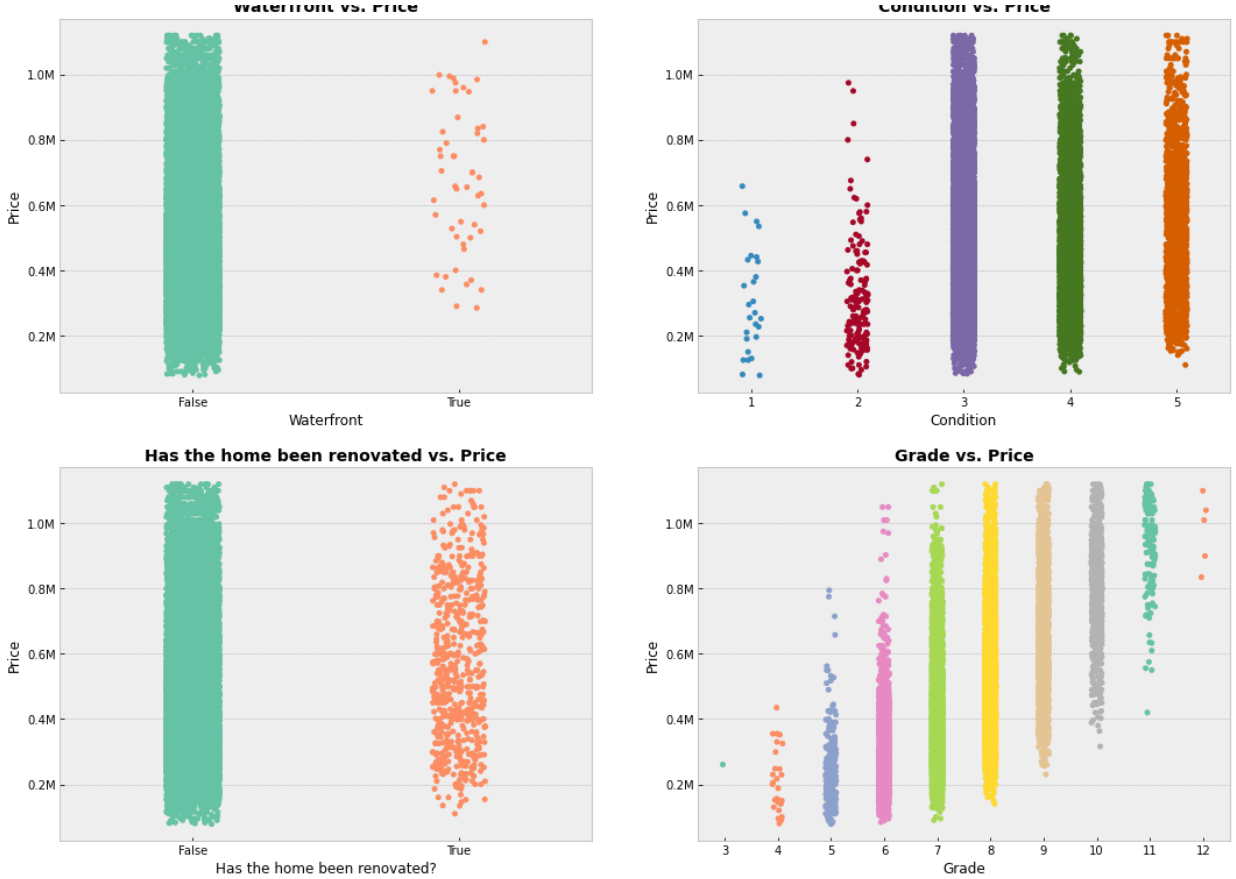
In [163]:
```
df_coeff3.drop('Intercept', axis=0, inplace=True)
```

In [164]:
```
df_coeff3.sort_values(by='abs_coeff').tail(30).plot(kind='barh', y='coe
plt.axvline(0, c='red')
plt.title('Coefficients of Features')
plt.grid()
```



In [165]:

```
# Variables in the owners control

from matplotlib.ticker import FuncFormatter

def millions(x, pos):
    return '%1.1fM' % (x * 1e-6)

with plt.style.context('bmh'):

    fig, axs=plt.subplots(nrows=2, ncols=2, figsize=(18,13))

    sns.stripplot(data=df_iqr_zip, x='waterfront', y='price', ax=axs[0,
    sns.stripplot(data=df_iqr_zip, x='condition', y='price', ax=axs[0,1
    sns.stripplot(data=df_iqr_zip, x='renovated_yes', y='price', ax=axs
    sns.stripplot(data=df_iqr_zip, x='grade', y='price',ax=axs[1,1], pa

    tf_labs = ['False', 'True']

    axs[0,0].set_title('Waterfront vs. Price', fontsize=14, fontweight=
    axs[0,0].set_xlabel('Waterfront')
    axs[0,0].set_ylabel('Price')
    axs[0,0].set_xticklabels(tf_labs)

    axs[0,1].set_title('Condition vs. Price',fontsize=14, fontweight='b
    axs[0,1].set_xlabel('Condition')
    axs[0,1].set_ylabel('Price')

    axs[1,0].set_title('Has the home been renovated vs. Price', fontsiz
    axs[1,0].set_xlabel('Has the home been renovated?')
    axs[1,0].set_ylabel('Price')
    axs[1,0].set_xticklabels(tf_labs)

    axs[1,1].set_title('Grade vs. Price', fontsize=14, fontweight='bold
    axs[1,1].set_xlabel('Grade')
    axs[1,1].set_ylabel('Price')



    formatter = FuncFormatter(millions)
    axs[0,0].yaxis.set_major_formatter(formatter)
    axs[0,1].yaxis.set_major_formatter(formatter)
    axs[1,0].yaxis.set_major_formatter(formatter)
    axs[1,1].yaxis.set_major_formatter(formatter)


```

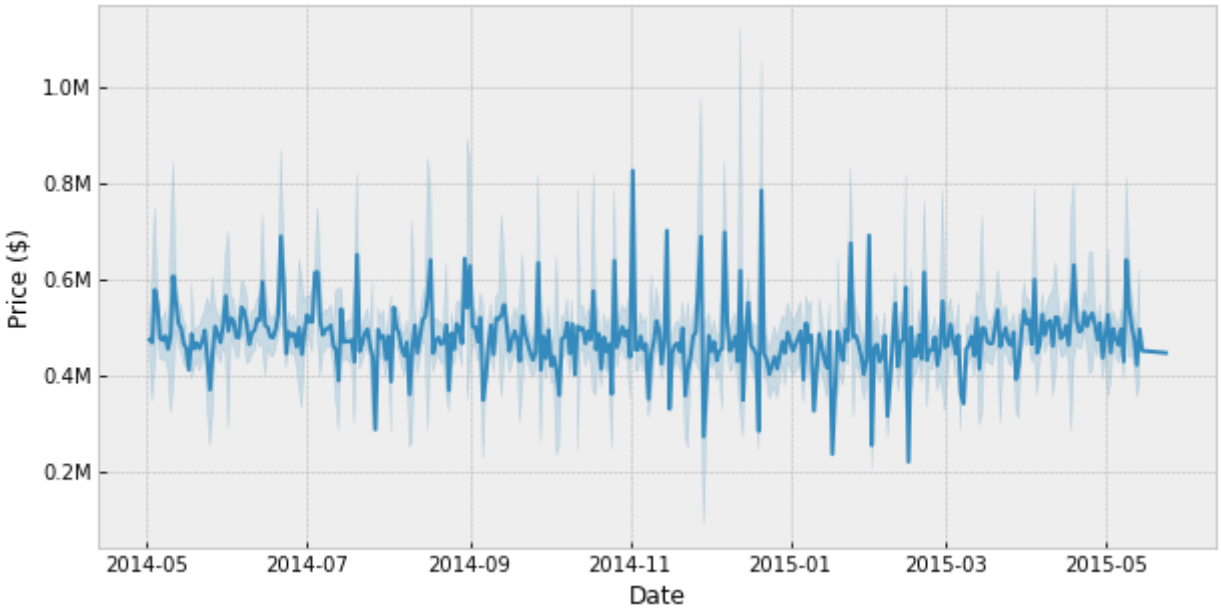


18 Additional Visualizations for Presentation

In [166]:
```
# Plot Home prices over time
df_price = df_iqr_zip2.copy()
df_price = df_price.set_index('date')
df_price = df_price.reset_index()

```

In [167]:

```
# Not using title because this is only going in PPT

with plt.style.context('bmh'):
    fig, ax = plt.subplots(figsize=(10,5))
    sns.lineplot(data=df_price, x='date', y='price')

    formatter = FuncFormatter(millions)
    ax.yaxis.set_major_formatter(formatter)
    ax.set_ylabel('Price ($)')
    ax.set_xlabel('Date')

```



In [ ]:

```

```