dsc-phase-2-project (/github/kcoop610/dsc-phase-2-project/tree/main)
/ report.ipynb (/github/kcoop610/dsc-phase-2-project/tree/main/report.ipynb)

Table of Contents

King County Housing Market

Kristin Cooper | DTSC-FT-022221 | Instructor: James Irving

Image Source (https://www.seattletimes.com/business/real-estate/king-county-home-prices-dropped-as-coronavirus-squelched-activity-but-now-the-market-may-be-picking-up/)

Overview

This project intends to model the value of houses in King County, WA, USA in order to help homeowners decide which renovations - if any - would increase the resale potential of their home.

Source Data

The model has been created based on King County House Sales dataset describing houses sold between May 2014 and May 2015.

Import packages & style notebook

In [1]:
```
# basic packages
import pandas as pd
pd.set_option('display.max_columns', 0)
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('bmh') #try also seaborn-darkgrid, fivethirtyeight
%matplotlib inline
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# stats packages
import statsmodels
import statsmodels.api as sm
import statsmodels.stats.api as sms
import statsmodels.formula.api as smf
from statsmodels.formula.api import ols
import scipy.stats as stats
from sklearn.linear_model import LinearRegression

# styling notebook
from IPython.core.display import HTML
def css_styling():
    styles = open("./styles/custom.css", "r").read()
    return HTML(styles)
css_styling()
```

Out[1]:

Load and understand dataset

In [2]:
```
df = pd.read_csv('data/kc_house_data.csv')
display(df.info())
display(df.head())
display(round(df.describe(),3))
```

```
<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

None
```

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

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors
count	2.159700e+04	21597.000	21597.000	21597.000	21597.000	21597.000	21597.000
mean	4.580474e+09	540296.574	3.373	2.116	2080.322	15099.409	1.494
std	2.876736e+09	367368.140	0.926	0.769	918.106	41412.637	0.540
min	1.000102e+06	78000.000	1.000	0.500	370.000	520.000	1.000
25%	2.123049e+09	322000.000	3.000	1.750	1430.000	5040.000	1.000

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors
50%	3.904930e+09	450000.000	3.000	2.250	1910.000	7618.000	1.500
75%	7.308900e+09	645000.000	4.000	2.500	2550.000	10685.000	2.000
max	9.900000e+09	7700000.000	33.000	8.000	13540.000	1651359.000	3.500

Data Preprocessing

To begin, the raw data needs a bit of manipulation to prepare it for visualization and modeling.

Feature Engineering

Some columns I find to have confusing names, so I renamed them to be more intuitive to me. This involved a bit of research from kingcounty.gov (https://www.kingcounty.gov/services/data.aspx).

Based on the columns available in the source data, I was able to also calculate a few features such as the home's age at sale, a boolean value for whether the house had renovations done, and how many years had passed since renovation, which I thought may be interesting to explore.

In [3]:
```
# Set target feature
target = 'price'
```

In [4]:
```
# Rename columns to be more intuitive
df.rename(mapper={'date': 'date_sold',
                  'sqft_living15': 'neighboring_sqft_living',
                  'sqft_lot15': 'neighboring_sqft_lot'}, axis=1, inplac
```

In [5]:
```
# Create a column for Year Sold in order to calculate Age At Sale
df['year_sold'] = df['date_sold'].apply(lambda x: int(x[-4:]))
df['age_at_sale'] = df['year_sold'] - df['yr_built']

# Create a categorical column to indicate if a house was renovated or n
df['renovated?'] = np.where(df['yr_renovated']>0, 'Yes', 'No')
df.head()

df.head()
```

Out[5]:

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

In [6]:
```
# Create a column for Years Since Renovated for houses that have indeed
df_renovated = df.loc[df['renovated?'] == 'Yes']
df_renovated['yrs_since_renovated'] = df_renovated['year_sold'] - df_re
df['yrs_since_renovated'] = df_renovated['yrs_since_renovated']
df
```

```
<ipython-input-6-6d07e190143d>:3: 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-
  df_renovated['yrs_since_renovated'] = df_renovated['year_sold'] - df_
```

Out[6]:

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

21597 rows × 25 columns

Data Types

Storing data as the appropriate type is important for mathematical calculations, modeling, and sorting.

In [7]:
```
# Found "?" value in sqft_basement column, keeping it from being stored
# reviewed the rows with "?" value, decided to replace the ? with nan i
df.loc[df['sqft_basement']=='?'].drop(columns=['id', 'waterfront', 'nei
                                               'neighboring_sqft_lot',
                                               'age_at_sale','yrs_since
df['sqft_basement'].replace(to_replace='?', value=0, inplace=True)
df['sqft_basement'] = df['sqft_basement'].astype(float)
```

In [8]:
```
# Update data types
df['id'] = df['id'].astype(object)
df['date_sold'] = pd.to_datetime(df['date_sold'])
df['price'] = df['price'].astype(int)
df['sqft_basement'] = df['sqft_basement'].astype(int)
df['zipcode'] = df['zipcode'].astype(object)

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 25 columns):
 #   Column                  Non-Null Count  Dtype         
---  ------                  --------------  -----         
 0   id                      21597 non-null  object        
 1   date_sold               21597 non-null  datetime64[ns]
 2   price                   21597 non-null  int64         
 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  int64         
 14  yr_built                21597 non-null  int64         
 15  yr_renovated            17755 non-null  float64       
 16  zipcode                 21597 non-null  object        
 17  lat                     21597 non-null  float64       
 18  long                    21597 non-null  float64       
 19  neighboring_sqft_living 21597 non-null  int64         
 20  neighboring_sqft_lot    21597 non-null  int64         
 21  year_sold               21597 non-null  int64         
 22  age_at_sale             21597 non-null  int64         
 23  renovated?              21597 non-null  object        
 24  yrs_since_renovated     744 non-null    float64       
dtypes: datetime64[ns](1), float64(8), int64(13), object(3)
memory usage: 4.1+ MB
```

Missing Data

Missing data elements may sway calculations and confuse models, so I either filled or removed these from the raw set.

In [9]:
```
# Look for null values, found in columns: waterfront, view, sqft_baseme
df.isna().any()
```

Out[9]:
```
id                          False
date_sold                   False
price                       False
bedrooms                    False
bathrooms                   False
sqft_living                 False
sqft_lot                    False
floors                      False
waterfront                   True
view                         True
condition                   False
grade                       False
sqft_above                  False
sqft_basement               False
yr_built                    False
yr_renovated                 True
zipcode                     False
lat                         False
long                        False
neighboring_sqft_living     False
neighboring_sqft_lot        False
year_sold                   False
age_at_sale                 False
renovated?                  False
yrs_since_renovated          True
dtype: bool
```

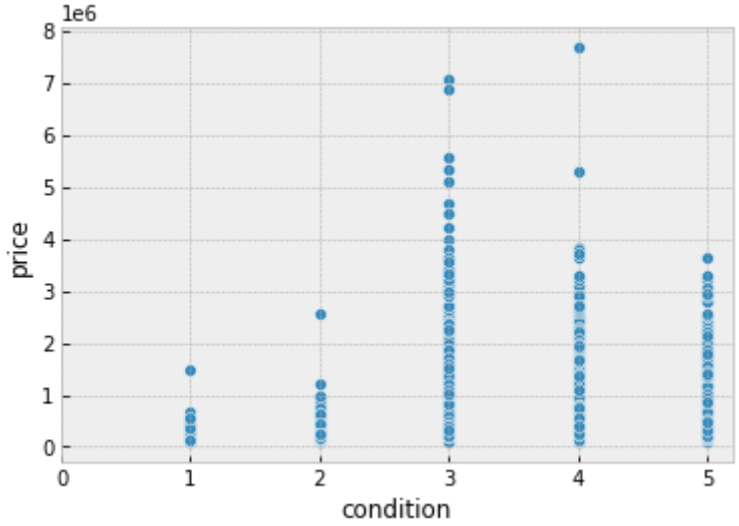
In [10]:
```
# Fix waterfront null values - vast majority of houses are not waterfr
df['waterfront'].fillna(0, inplace=True)


# Fix view null values - discrete categorical column, so created an "u
df['view'].fillna(0, inplace=True)

# Fix yr_renovated null values - continuous numeric column, so selecte
df['yr_renovated'].unique()
df['yr_renovated'].fillna(0, inplace=True)
df['yr_renovated'] = df['yr_renovated'].astype(int)

# Fix yrs_since_renovated null values - continuous numeric column, so s
df['yrs_since_renovated'].fillna(999, inplace=True)
df['yrs_since_renovated'] = df['yrs_since_renovated'].astype(int)

# Fix sqft_basement null values - after some exploration, decided to f
df['sqft_basement'].fillna(0, inplace=True)
```

Feature Exploration

Two features that jumped out as interesting and needing some more detail were "Condition" and "Grade." I found some additional information about the King County grade and condition scales here (https://info.kingcounty.gov/assessor/esales/Glossary.aspx?type=r).

In [11]:
```
# Explore condition values
display(df['condition'].value_counts())
sns.scatterplot(data=df, x='condition', y='price')
plt.xticks(ticks=range(6))
plt.show();
```

```
3    14020
4     5677
5     1701
2      170
1       29
Name: condition, dtype: int64
```

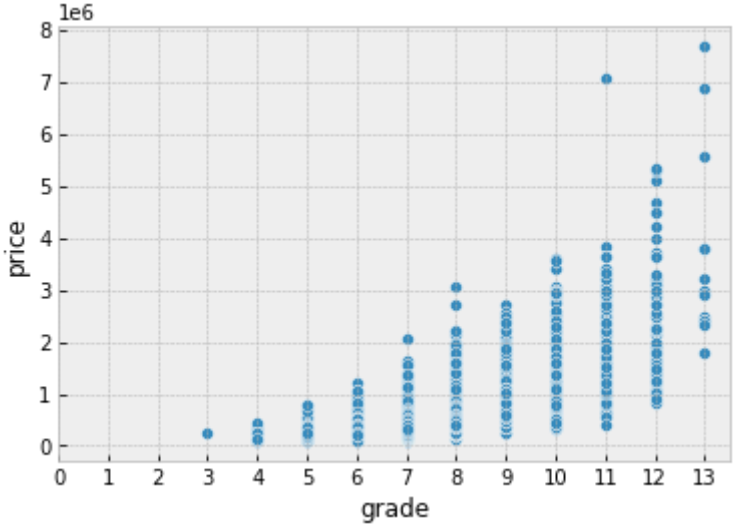


In [12]:
```
# Change condition category to 0 (poor), 1 (average), and 2 (excellent,
df['condition'].replace(to_replace=[1, 2], value=0, inplace=True)
df['condition'].replace(to_replace=[3, 4], value=1, inplace=True)
df['condition'].replace(to_replace=5, value=2, inplace=True)
df['condition']
```

Out[12]:
```
0        1
1        1
2        1
3        2
4        1
        ..
21592    1
21593    1
21594    1
21595    1
21596    1
Name: condition, Length: 21597, dtype: int64
```

In [13]:
```
# Explore grade values
# Appears to have a somewhat linear relationship with price, so I left
display(df['grade'].value_counts())
sns.scatterplot(data=df, x='grade', y='price', )
plt.xticks(ticks=range(14))
plt.show();
```

```
7     8974
8     6065
9     2615
6     2038
10    1134
11     399
5      242
12      89
4       27
13      13
3        1
Name: grade, dtype: int64
```



In [14]:
```
df.head(2)
```

Out[14]:

	id	date_sold	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	water
0	7129300520	2014-10-13	221900	3	1.00	1180	5650	1.0	
1	6414100192	2014-12-09	538000	3	2.25	2570	7242	2.0	

Final Feature Set

At this point, we have a set of cleaned-up features and are ready to look for linear relationships with our target - sale price.

Summarized below are the features in our dataset:

1. **id** - unique identifier for a house (not a predictor)
2. **date_sold** - day when house was sold
3. **price** - sale price of home
4. **bedrooms** - number of bedrooms (1-11 in this sample)
5. **bathrooms** - number of bathrooms (0.50-8.00 in this sample)
6. **sqft_living** - square footage of the home
7. **sqft_lot** - square footage of the lot
8. **floors** - floors (levels) in house
9. **waterfront** - whether or not the house has a view to a waterfront (1=True, 0=False)
10. **view** - an index from 0 to 4 of how good the view of the property was
11. **condition** - originally coded 1-5 (additional details below). For this model, I recoded to 0 (poor), 1 (average), and 2 (excellent)
12. **grade** - represents the construction quality of improvements. Grades run from grade 1 (below minimum builting standards) to 13 (custom-designed mansion). 1-5 do not meet building code. Additional detail below.
13. **sqft_above** - square footage of house apart from basement
14. **sqft_basement** - square footage of the basement
15. **yr_built** - year when the house was first built
16. **yr_renovated** - year when house was renovated
17. **zipcode** - zipcode where home is located
18. **lat** - latitude location of the house
19. **long** - longitude location of the house
20. **neighboring_sqft_living** - square footage of interior housing living space for the nearest 15 neighbors
21. **neighboring_sqft_lot** - square footage of the land lots of the nearest 15 neighbors
22. **year_sold** - extrapolated from date_sold column
23. **age_at_sale** - calculated by the yr_sold minus yr_built
24. **renovated?** - Yes/No if the house has been renovated, as determined by yr_renovated > 0
25. **yrs_since_renovated** - calculated by yr_sold minus yr_renovated

Grade & Condition Category Details

click here to expand for details

Deal with Outliers

Outliers in the dataset are particularly messy when modeling, since they don't follow the same patterns as the vast majority of cases.

One very helpful statistic when looking for outliers is the Interquartile Range, which is a measure of dispersion between the 25th and 75th percentiles of the data. The box-and-whisker plot also calculates the "minimum" and "maximum" whiskers of the data. Any points that fall outside the whiskers are considered outliers.

I created a function to visualize outliers on multiple features, remove them, then revisualize the dispersion.

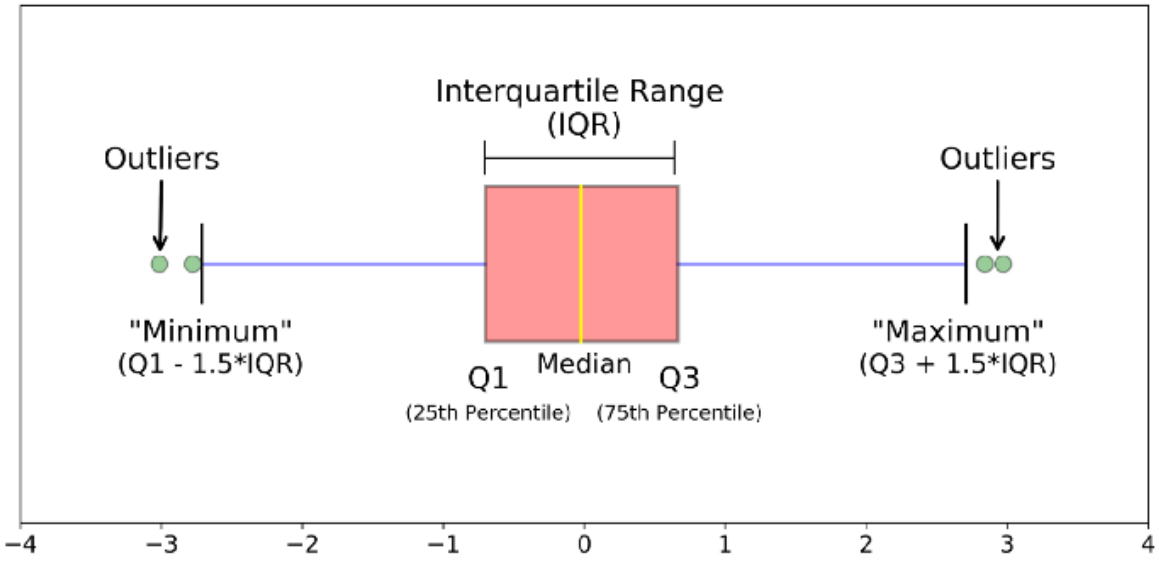
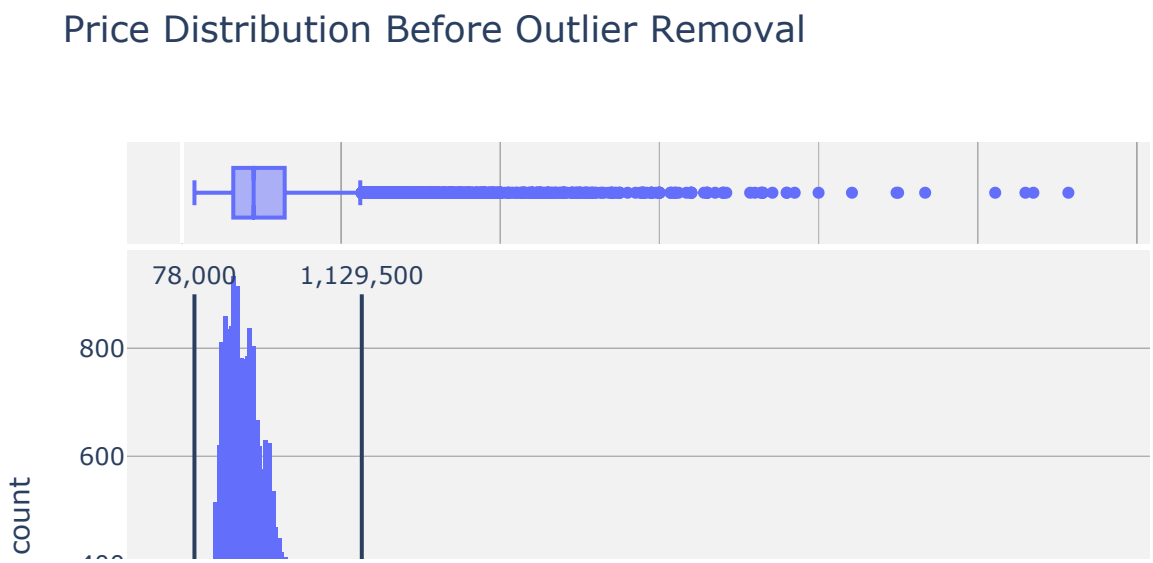


Image Source (https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51)

In [15]:
```
def plot_iqr(df, col, title=None, marker_height=930, line_height=900):
    '''
    Parameters:
    df = pd.dataframe, main data source
    col = column name to plot (x axis)
    title = title for graph
    marker_height = int, where to place marker based on highest y-value
    line_height = int, where to end vertical line markers (default=900)

    Returns:
    Histogram with marginal box plot, markers and labels at max fence a
    to show where there are outliers in your dataset

    '''
    # Calculating min, max, Q1, and Q3 interquartile ranges on price co
    iqr = stats.iqr(df[col])
    q1 = np.quantile(a=df[col], q=.25)
    minimum = q1-1.5*iqr
    q3 = np.quantile(a=df[col], q=.75)
    maximum = q3+1.5*iqr

    if minimum < df[col].min():
        min_fence = df[col].min()
    else:
        min_fence = minimum

    if maximum > df[col].max():
        max_fence = df[col].max()
    else:
        max_fence = maximum

    # Create a variable formatted with commas
    min_fence_str = '{:,}'.format(int(min_fence))
    max_fence_str = '{:,}'.format(int(max_fence))

    # Visualize distribution of target, showing outliers on the high en
    fig = px.histogram(df, x=df[col], marginal='box',
                       title=title)
    fig.update_layout(plot_bgcolor='#f2f2f2', bargap=0.1, height=500,
                      title_font_size=18, font_size=12)

    # max marker
    fig.add_trace(go.Scatter(x=np.array(max_fence), y=np.array(marker_h
                             text=max_fence_str, mode='text',
                             name=('Max Fence: {}'.format(max_fence_str
    fig.add_shape(type='line', x0=max_fence, y0=0, x1=max_fence, y1=lin

    # min marker
    fig.add_trace(go.Scatter(x=np.array(min_fence), y=np.array(marker_h
                             text=(min_fence_str), mode='text',
                             name=('Min Fence: {}'.format(min_fence_str
    fig.add_shape(type='line', x0=min_fence, y0=0, x1=min_fence, y1=lin

    fig.show()
    print('Min: ', df[col].min(), 'Max: ', df[col].max())
```

In [16]:
```
# Remove bedroom outlier
df = df[df['bedrooms'] != 33]
```

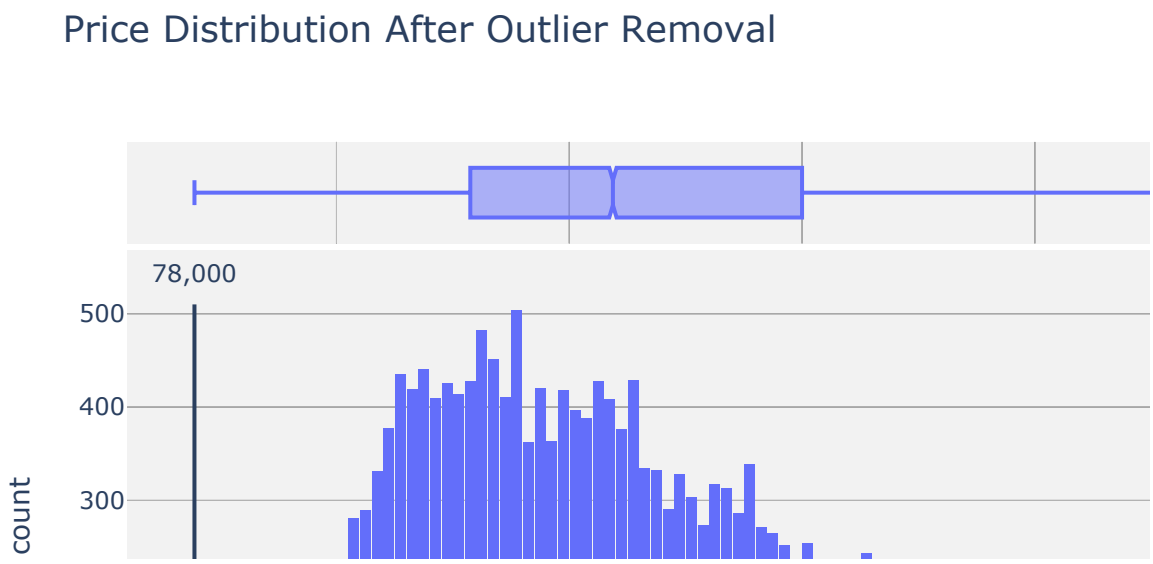
In [17]:
```
plot_iqr(df, target, title='Price Distribution Before Outlier Removal')
```




```
Min:  78000 Max:  7700000
```

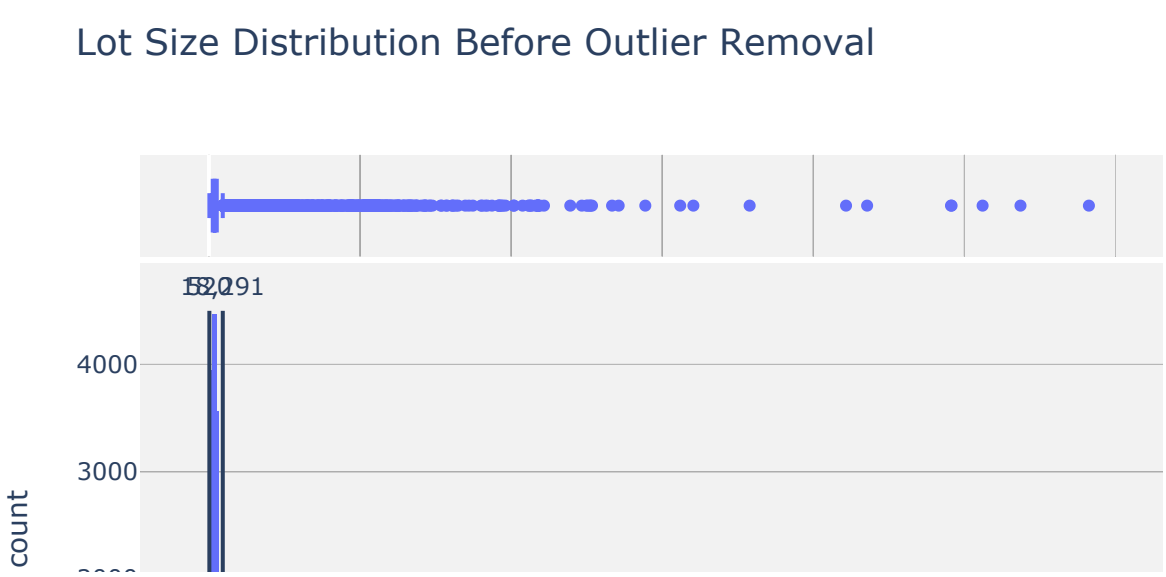
In [18]:
```
# Removed outliers above max fence
df_outliers_removed = df[df.price < 1129500]
```

In [19]:
```
# Checking new distribution
plot_iqr(df_outliers_removed, target, title='Price Distribution After O
         marker_height=540, line_height=510)
```




```
Min:  78000 Max:  1120000
```

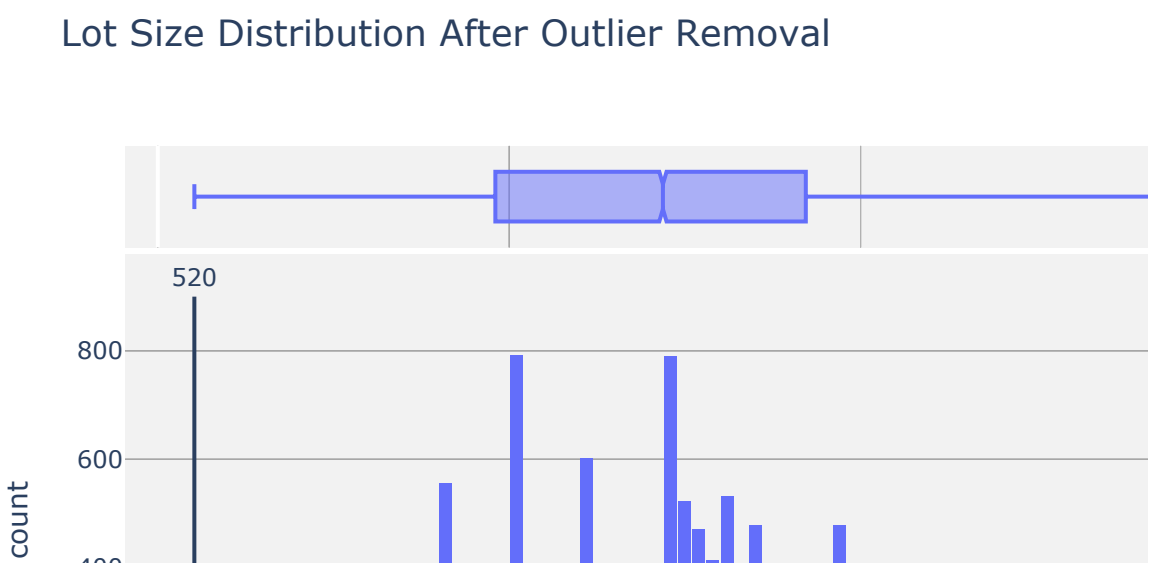
In [20]:
```
plot_iqr(df_outliers_removed, col='sqft_lot', title='Lot Size Distribut
        marker_height=4700, line_height=4500)
```




```
Min:  520 Max:  1651359
```

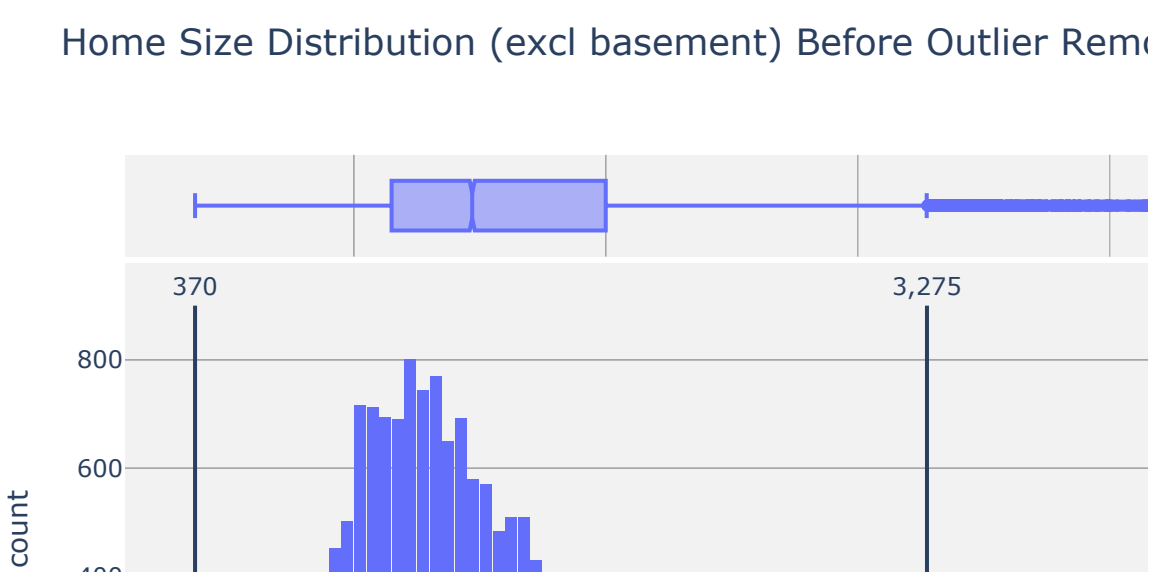
In [21]:
```
# Removed outliers above max fence
df_outliers_removed = df_outliers_removed[df_outliers_removed['sqft_lot
```

In [22]:
```
# Checking new distribution
plot_iqr(df_outliers_removed, col='sqft_lot', title='Lot Size Distribut
```



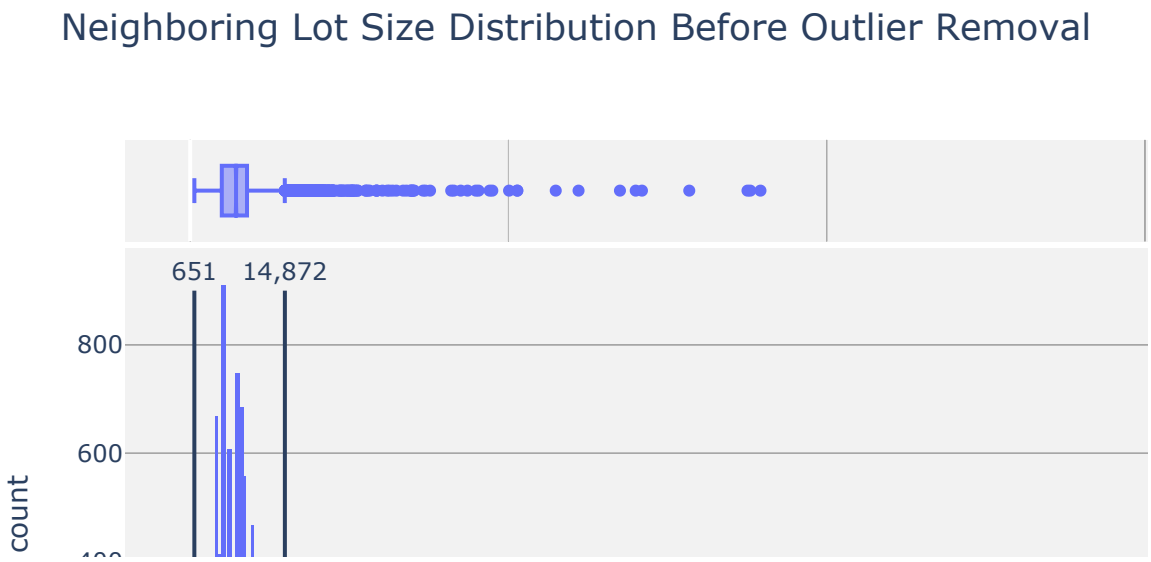

```
Min:  520 Max:  18265
```

In [23]:
```
plot_iqr(df_outliers_removed, col='sqft_above',
         title='Home Size Distribution (excl basement) Before Outlier R
```




```
Min:  370 Max:  5370
```

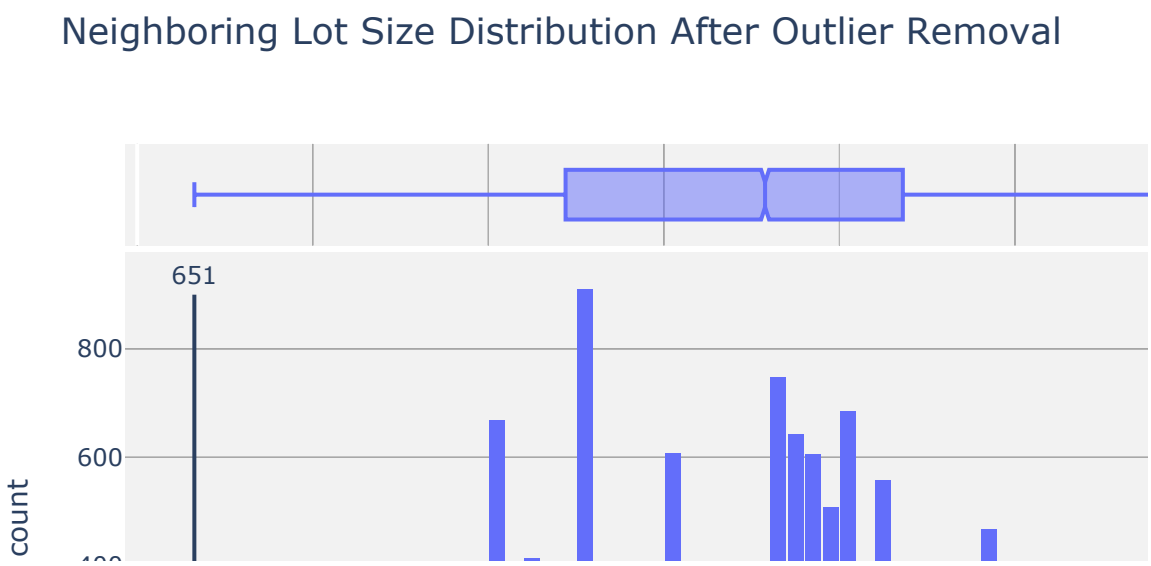
In [24]:
```
plot_iqr(df_outliers_removed, col='neighboring_sqft_lot',
         title='Neighboring Lot Size Distribution Before Outlier Remova
```




```
Min:  651 Max:  196591
```

In [25]:
```
# Remove outliers above max fence
df_outliers_removed = df_outliers_removed[df_outliers_removed['neighbo
```

In [26]:
```
# Check distribution after outlier removal
plot_iqr(df_outliers_removed, col='neighboring_sqft_lot',
         title='Neighboring Lot Size Distribution After Outlier Removal
```




```
Min:  651 Max:  14859
```

Explore

Now that the source data has been cleaned up, we use more descriptive statistics to better understand our data and draw initial ideas of what features might impact our target, price.

Zipcode

Since King County includes the metro area of Seattle, WA, I had a hunch that proximity to the city center would influence house price, so I wanted to visualize this.

In [27]:
```
zipcodes_df = df_outliers_removed.groupby('zipcode')\
                                 .mean().reset_index()\
                                 .drop(columns=['neighboring_sqft_livi
                                                'neighboring_sqft_lot',
                                                'age_at_sale', 'yrs_si
                                                'yr_renovated'])
int_cols = ['price', 'sqft_living', 'sqft_lot', 'sqft_above', 'sqft_ba
zipcodes_df[int_cols] = zipcodes_df[int_cols].astype(int)
zipcodes_df.head()
```

Out[27]:

	zipcode	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	vie
0	98001	268933	3.465704	2.069495	1877	8544	1.480144	0.0	0.0469
1	98002	232821	3.309278	1.819588	1619	7254	1.335052	0.0	0.0000
2	98003	283161	3.339844	2.008789	1864	9098	1.308594	0.0	0.1992
3	98004	833037	3.381679	1.944656	1965	9503	1.270992	0.0	0.0305
4	98005	704517	3.660194	2.223301	2284	10271	1.174757	0.0	0.0097

In [28]:
```
# Took a public data file containing the shapes of zipcodes in King Co
# geojson file source: https://data.seattle.gov/dataset/Zip-Codes/dk58-

# Parsed just the zipcodes in our source dataset from the public geojs
# into an updated json with just the necessary data for mapping
# inspiration from: https://towardsdatascience.com/visualizing-data-at-

import json
with open('data/kc_zipcode_map.geojson', 'r') as jsonFile:
    kczipcodes = json.load(jsonFile)

geozips = []
for i in range(len(kczipcodes['features'])):
    if kczipcodes['features'][i]['properties']['ZIP'] in list(zipcodes_
        geozips.append(kczipcodes['features'][i])

new_json = dict.fromkeys(['type', 'features'])
new_json['type'] = 'FeatureCollection'
new_json['features'] = geozips

open('data/updated_kc_zipcode_map.json', 'w').write(
        json.dumps(new_json, sort_keys=True, indent=4, separators=(',',
```

In [29]:
```
# Checking the format of the new json file to find what I can map to tl
with open('data/updated_kc_zipcode_map.json', 'r') as f:
    kc_geojson = json.load(f)
kc_geojson['features'][0]
```

Out[29]:

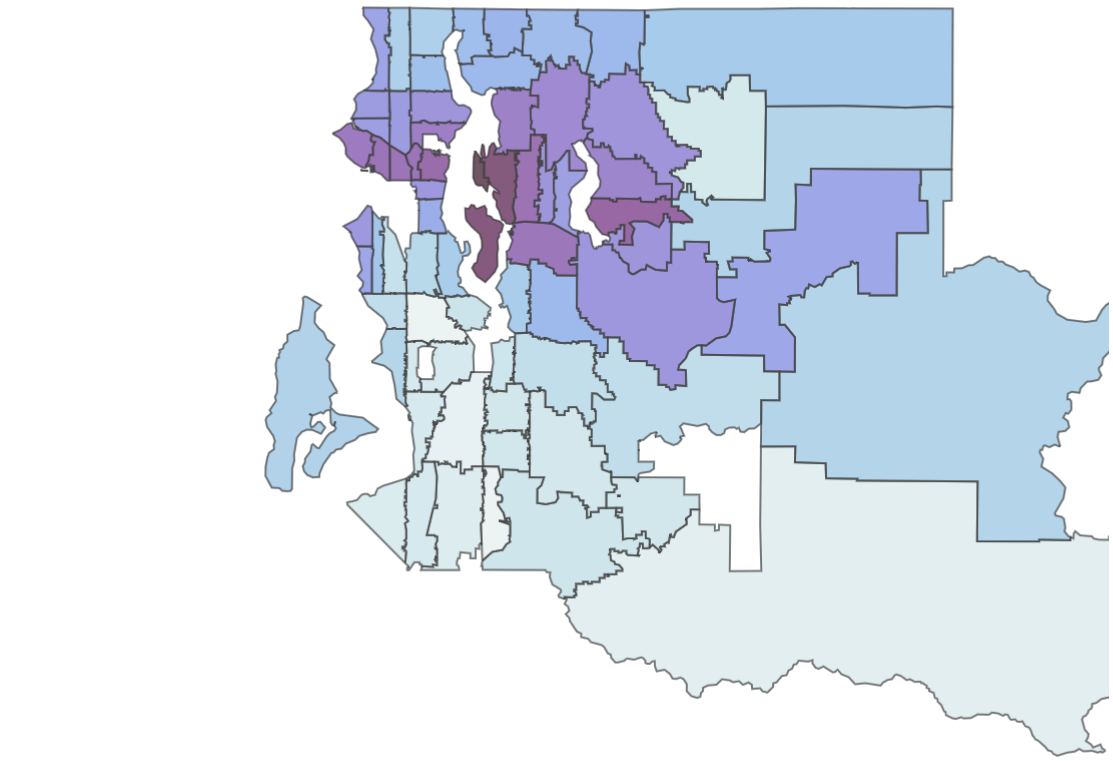
```
{'geometry': {'coordinates': [[[-122.21842289674088, 47.43750364859679
    [-122.21896085798723, 47.43750323467437],
    [-122.21957512310345, 47.43750274754203],
    [-122.22159770707769, 47.43750114891263],
    [-122.22159948023497, 47.43495688786757],
    [-122.22170516151417, 47.43495715523315],
    [-122.22360892800278, 47.434961994316446],
    [-122.22373457127979, 47.434962099304876],
    [-122.22587769127573, 47.434963881291935],
    [-122.22612385037388, 47.43508413516542],
    [-122.22618863127923, 47.43511744801654],
    [-122.22628847403791, 47.43516579242339],
    [-122.2262886113923, 47.43516566719561],
    [-122.22648870399131, 47.43497890626116],
    [-122.22661342094158, 47.43486319441269],
    [-122.22671888296699, 47.434764385764986],
    [-122.2267982615053, 47.4346914367485],
    [-122.22690814577896, 47.43458841811856],
    [-122.22695898576944, 47.43454345348303],
    [-122.22701850789504, 47.434483939093646],
    [-122.2270844426768, 47.43441818005568],
    [-122.22714578795132, 47.43435055127265],
    [-122.22718410846879, 47.434310491103155],
    [-122.22726053540013, 47.43422142788365],
    [-122.22732069522594, 47.43414785273184],
    [-122.2273813395292, 47.434072901411746],
    [-122.22739692376065, 47.43405137660248],
    [-122.22743197734336, 47.434001801752046],
    [-122.22748358336494, 47.43392915432964],
    [-122.22751222433438, 47.43388583251501],
    [-122.2275442872729, 47.433838012733155],
    [-122.22758834146777, 47.43376677212654],
    [-122.22764006809757, 47.43367959868589],
    [-122.22768243768654, 47.43360153347239],
    [-122.22772417248696, 47.43351876184484],
    [-122.22776248474769, 47.433440316901844],
    [-122.22780573224404, 47.43333858560033],
    [-122.22787700091527, 47.43315130309473],
    [-122.22787886231515, 47.43306932215706],
    [-122.22790029497183, 47.43302107143945],
    [-122.22791802063026, 47.43296571778834],
    [-122.22793994389957, 47.43287929255673],
    [-122.22796339899458, 47.43277400009597],
    [-122.22797936792988, 47.43269068325332],
    [-122.2279888524406, 47.43262981826024],
    [-122.22799990716845, 47.43255183198607],
    [-122.22802949768368, 47.431544010204405],
    [-122.22802920794217, 47.431539979039805],
    [-122.22787176095208, 47.43139888046443],
    [-122.22787424969577, 47.431333263770256],
    [-122.2276848179994, 47.43124110416986],
    [-122.2275208963031, 47.43112941443359],
```

```
      [-122.17834708191135, 47.43034756875706],
      [-122.1783505528658, 47.430347547242015],
      [-122.17846427833116, 47.43034680194246],
      [-122.17866119065798, 47.430345532922416],
      [-122.17910664418226, 47.43034263384648],
      [-122.17914344659857, 47.43034238800509],
      [-122.17933789301365, 47.43034112461348],
      [-122.17973890672259, 47.430338500647004],
      [-122.1804548393713, 47.430333810796824],
      [-122.1807175207457, 47.430332088187065],
      [-122.18101421656355, 47.43033015165284],
      [-122.18126834061248, 47.430328726006174],
      [-122.18186936708582, 47.430324564752304],
      [-122.1824760528623, 47.43032036386599],
      [-122.18381065635054, 47.43031108703317],
      [-122.18513637357684, 47.43030186953662],
      [-122.18562808104285, 47.43029844959363],
      [-122.18633995609471, 47.430293501425865],
      [-122.18634367936257, 47.43029347665566],
      [-122.18634252206351, 47.43033136937003],
      [-122.18634213332096, 47.430343991887234],
      [-122.1864030535048, 47.43034403431523],
      [-122.18646391577916, 47.43034407740155],
      [-122.18673470709918, 47.4303442656362],
      [-122.18695109938896, 47.430344417509275],
      [-122.18697938093798, 47.430344436492256],
      [-122.18699149625252, 47.43034444348621],
      [-122.1873026307398, 47.43034465977605],
      [-122.18750464016938, 47.430344799821604],
      [-122.18765666969955, 47.430344907504974],
      [-122.18779194533518, 47.43034499890506],
      [-122.1877923040555, 47.43032740426942],
      [-122.18783250830195, 47.430327342400304],
      [-122.18791685559383, 47.430327208069166],
      [-122.1880158456088, 47.43032705317466],
      [-122.18845758689822, 47.43032666619356],
      ...]],
    'type': 'Polygon'},
   'properties': {'COUNTY': '033',
    'OBJECTID': 1,
    'SHAPE_Area': 228012909.32990217,
    'SHAPE_Length': 117508.21171773598,
    'ZIP': 98031,
    'ZIPCODE': '98031'},
   'type': 'Feature'}
```

In [30]:
```
# Create a string column with formatted price to put in visualization
zipcodes_df['$price'] = zipcodes_df['price'].apply(lambda x: '${:,}'.fo
```

In [31]:
```
def zipcodes_map(df, color, hoverdata=None, title=None, labels=None):
    '''
    Parameters:
    df - pd.Dataframe, main data to read
    color - str, column name in df to visualize
    hoverdata - list, column names to display on map
    labels - optional dict, {hoverdata: display name} to display more i
    title - optional str, title for map

    Returns:
    Choropleth map
    '''
    fig = px.choropleth_mapbox(data_frame=df, locations='zipcode', cent
                               color=color, color_continuous_scale='der
                               geojson=kc_geojson, featureidkey='proper
                               hover_data=hoverdata, labels=labels,
                               title=title, height=700, width=900, opac
    fig.update_layout(mapbox_style='carto-positron')
    fig.update_geos(fitbounds='locations', visible=False)
    fig.write_image((f'images/zipmap_{color}.png'))
    fig.show()
```

In [32]:
```
hoverdata = ['$price', 'sqft_living', 'sqft_lot', 'yr_built']
labels = {'$price': 'Sale Price', 'sqft_living': 'Living Space (sqft)',
          'sqft_lot': 'Lot Size (sqft)', 'yr_built': 'Year Built',
         'price': 'Sale Price ($)'}
mean_title = 'Average House Sale Price($), King County'
zipcodes_map(zipcodes_df, 'price', hoverdata=hoverdata, title=mean_tit
```



Looking for Linear Relationships

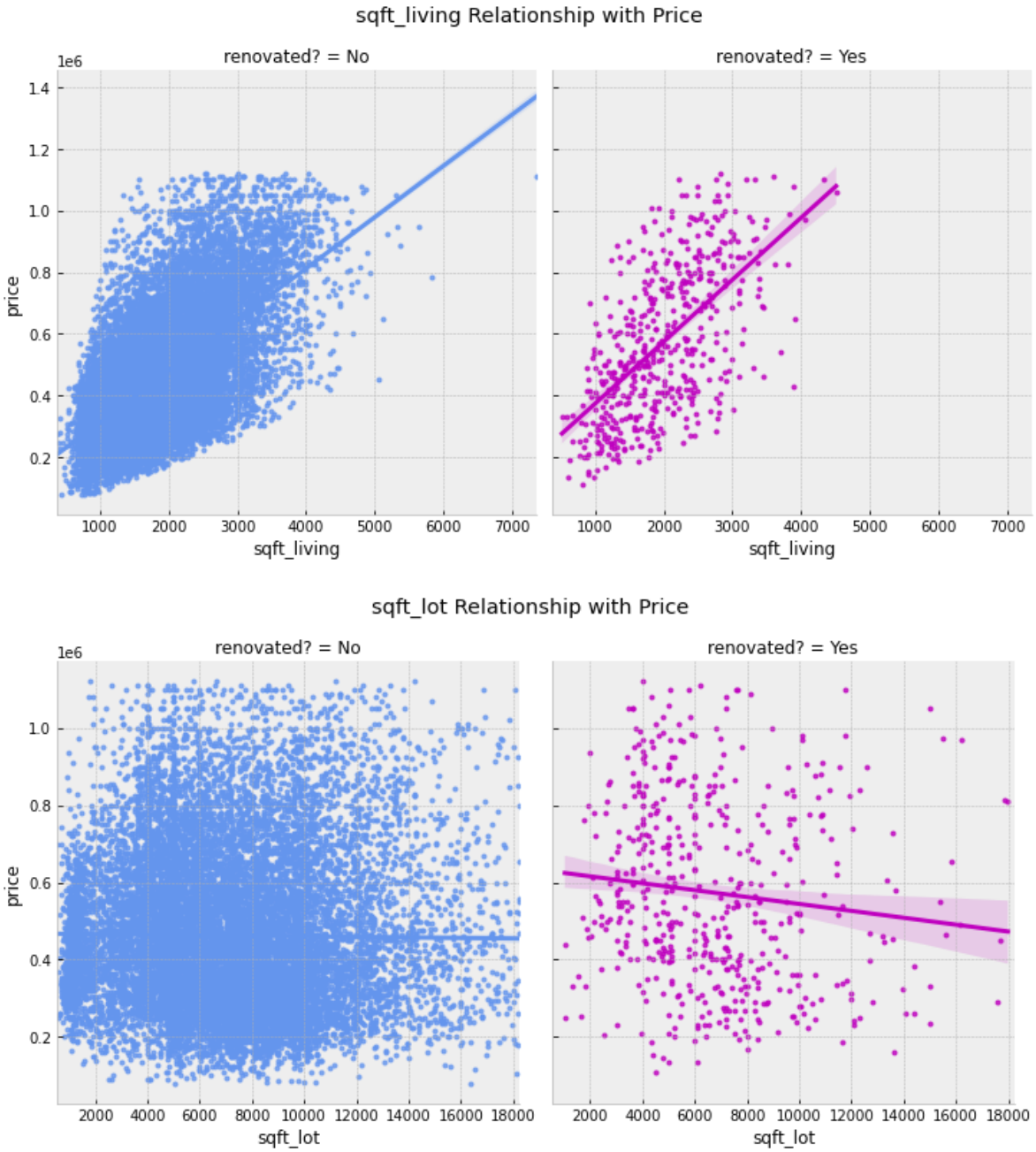
Going into the modeling step, it helps to have a sense of what individual features appear to have a linear relationship with sale price. In this section, we visualize the relationship between each feature and sale price. I also looked for any significant difference in these relationships between houses that had been renovated vs. houses that had not.

In [33]:
```
df_outliers_removed.head(2)
```

Out[33]:

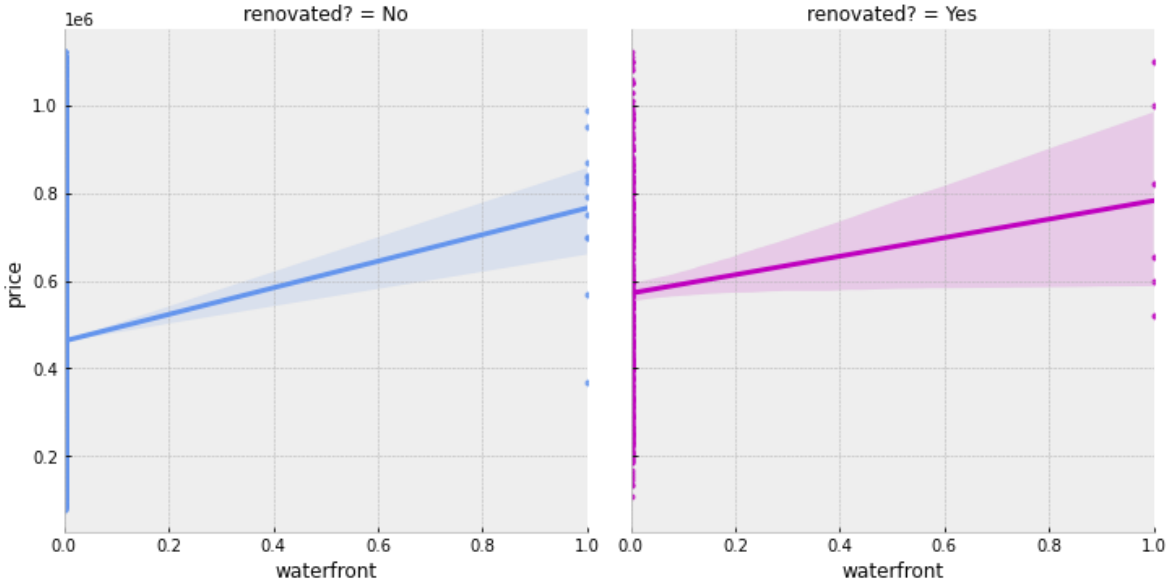
	id	date_sold	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	water
0	7129300520	2014-10-13	221900	3	1.00	1180	5650	1.0	
1	6414100192	2014-12-09	538000	3	2.25	2570	7242	2.0	

In [34]:
```
target='price'
plot_columns = ['bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'fl
                'waterfront', 'view', 'condition', 'grade', 'sqft_above
                'yr_built', 'lat', 'long', 'neighboring_sqft_living',
                'neighboring_sqft_lot', 'year_sold', 'age_at_sale']
for column in plot_columns:
    sns.lmplot(data=df_outliers_removed, x=column, y='price', col='renc
        palette=dict({'No':'cornflowerblue', 'Yes':'m'}), markers='.')
    plt.suptitle(t=f'{column} Relationship with Price', fontsize='x-lar
    plt.savefig(fname=(f'images/lmplot_{column}'), facecolor='w', bbox_
    plt.show()
```

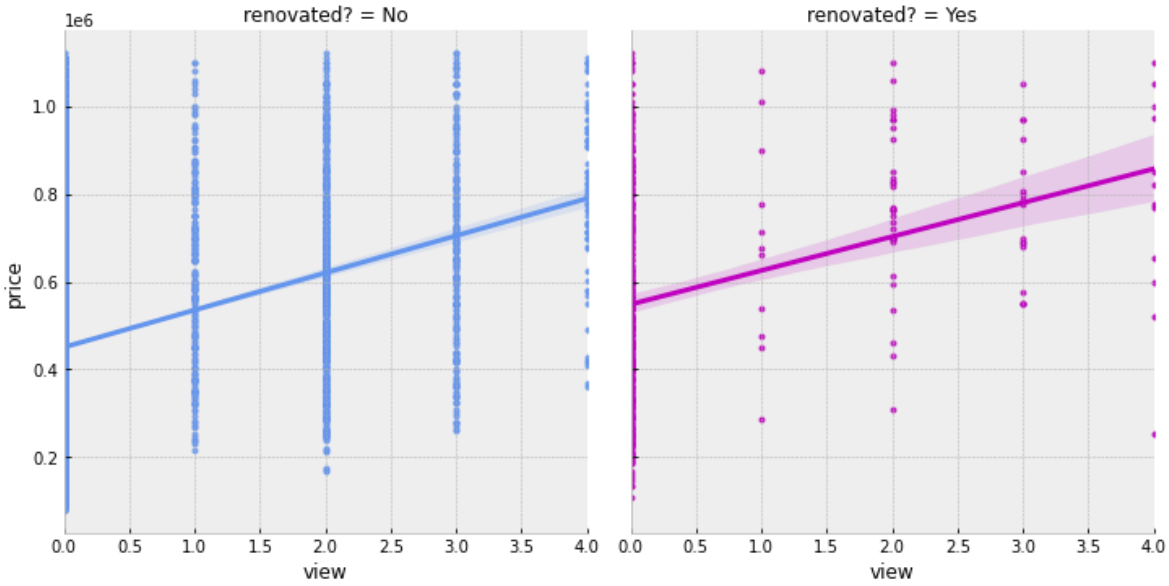

sqft_living Relationship with Price
renovated? = No
renovated? = Yes
1e6
price
sqft_living
sqft_lot Relationship with Price
sqft_lot

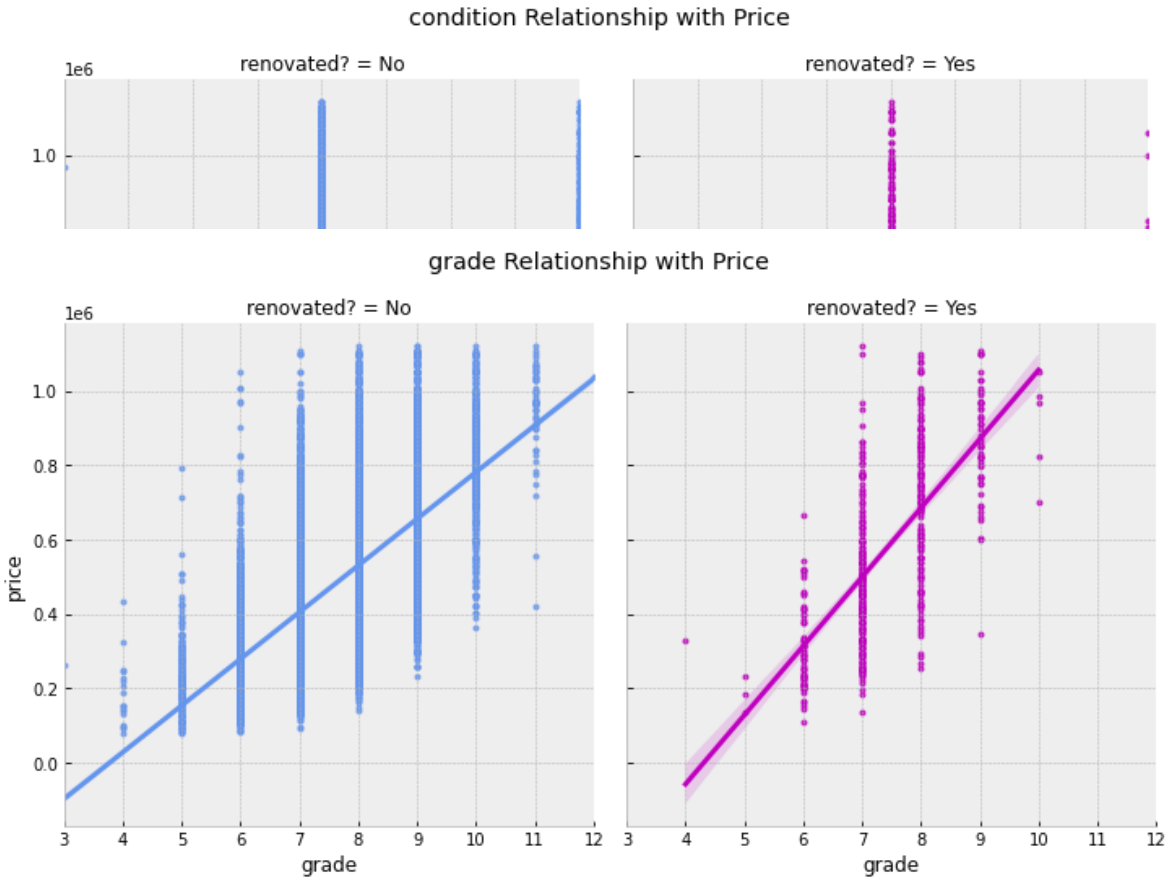
floors Relationship with Price

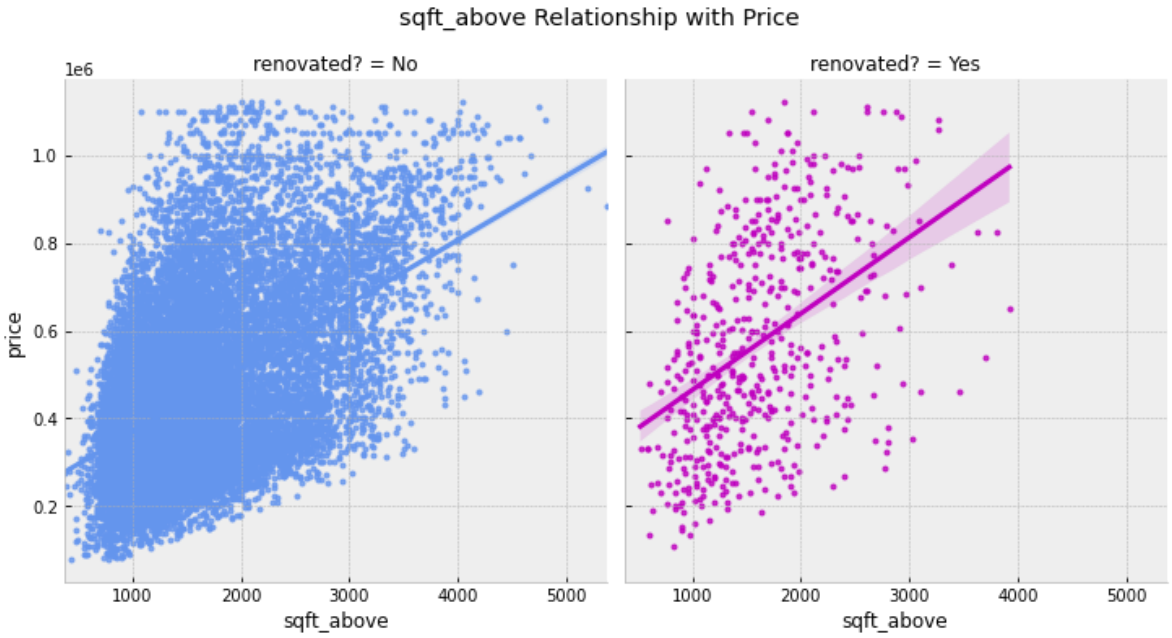
waterfront Relationship with Price

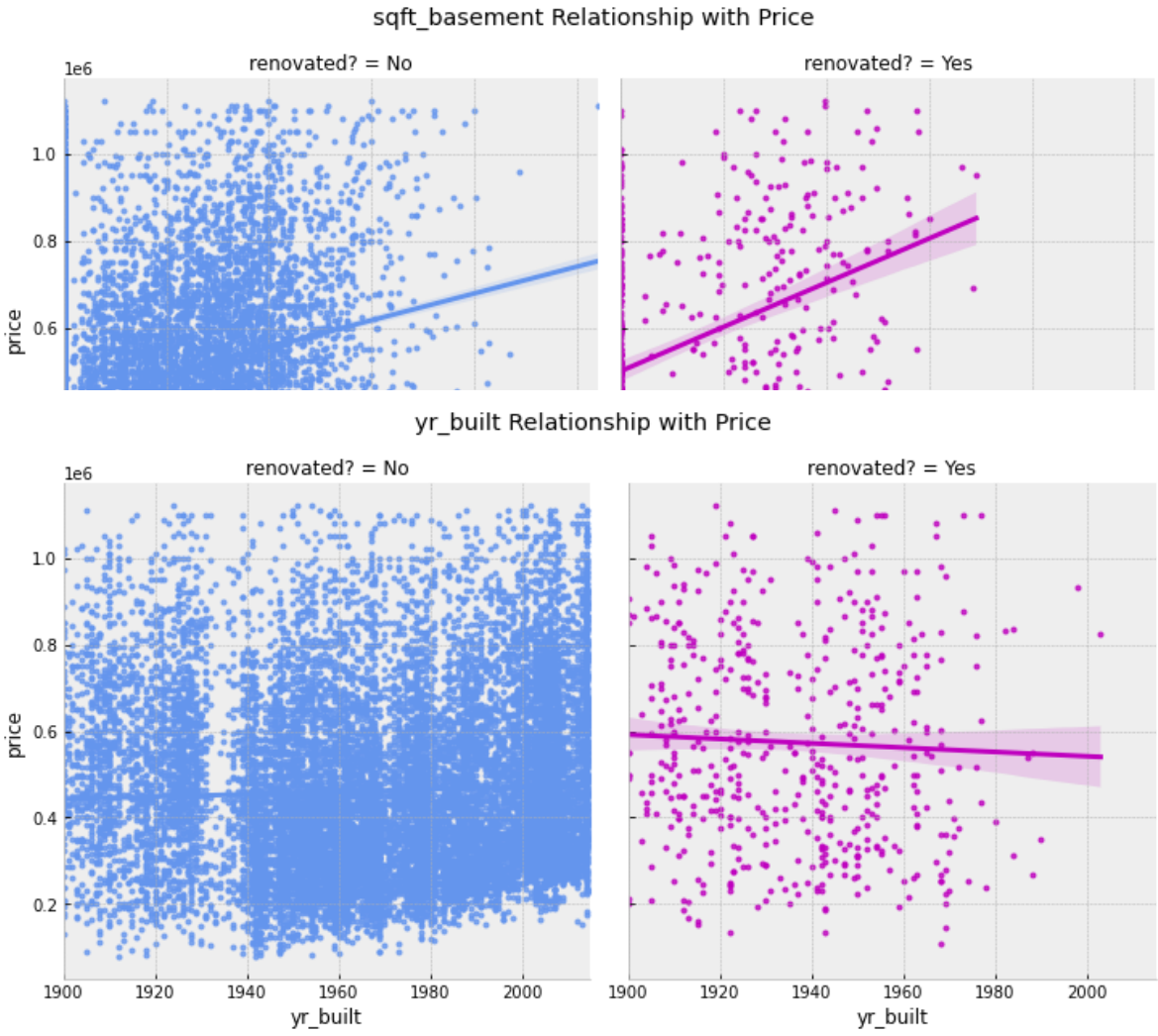


view Relationship with Price

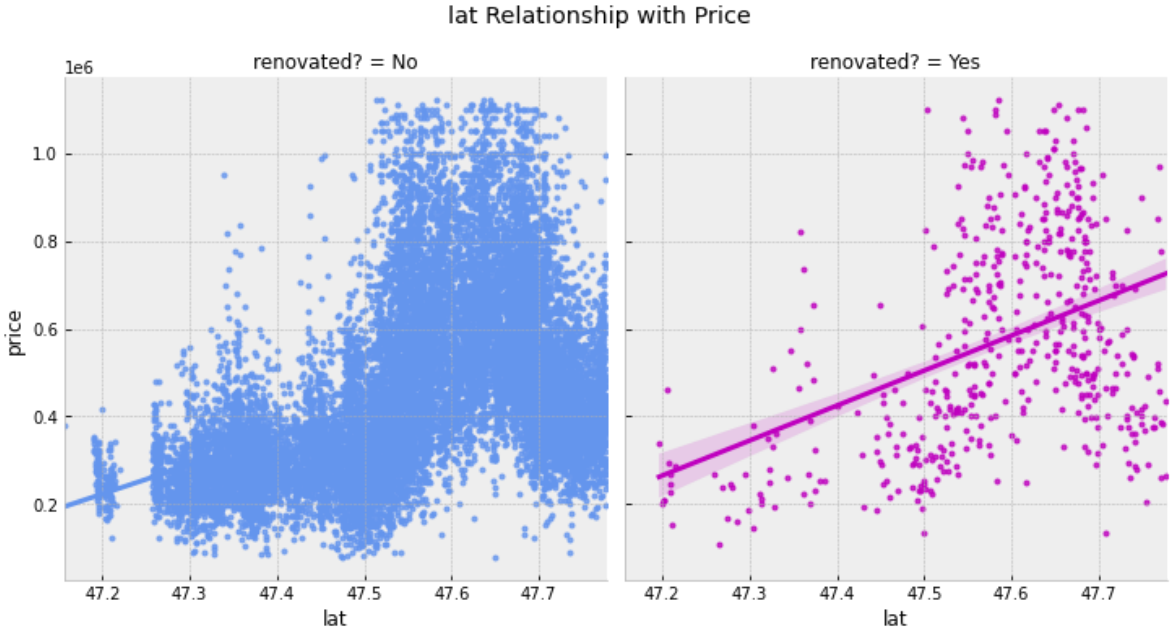


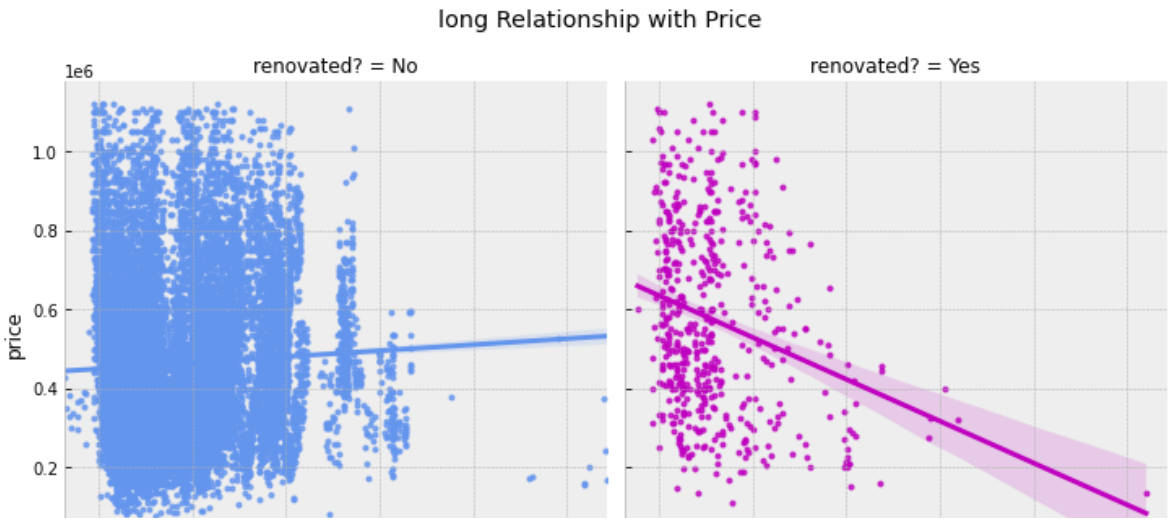

condition Relationship with Price
renovated? = No
renovated? = Yes
1e6
1.0
grade Relationship with Price
renovated? = No
renovated? = Yes
1e6
price
grade
3 4 5 6 7 8 9 10 11 12
0.0 0.2 0.4 0.6 0.8 1.0

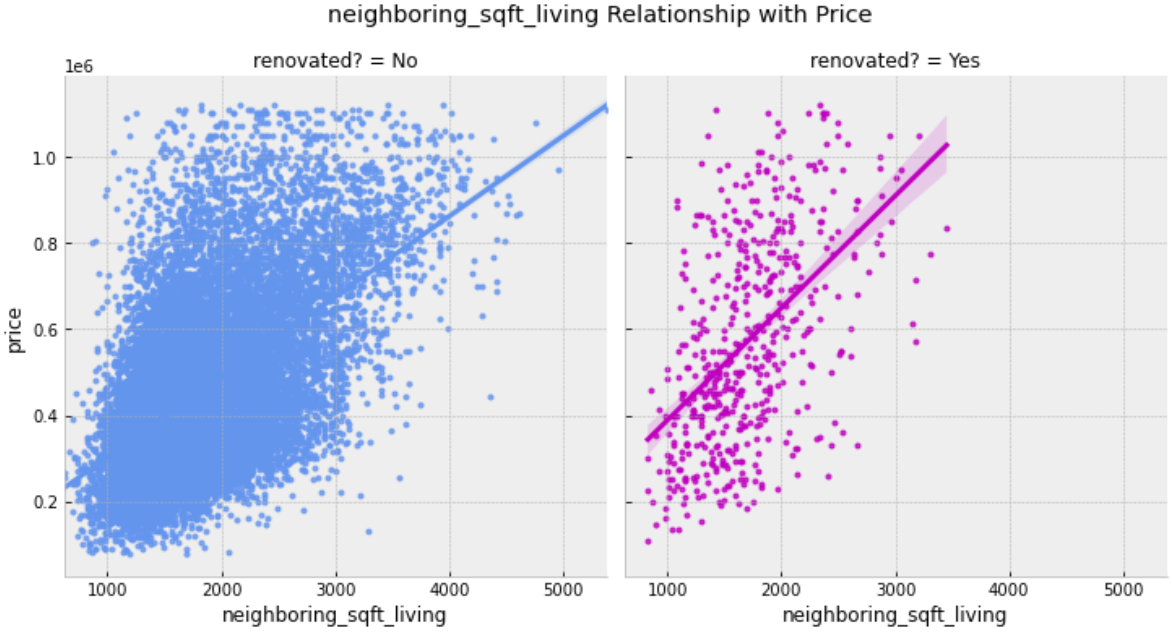

sqft_above Relationship with Price
renovated? = No
renovated? = Yes
1e6
price
sqft_above
1000 2000 3000 4000 5000
0.2 0.4 0.6 0.8 1.0

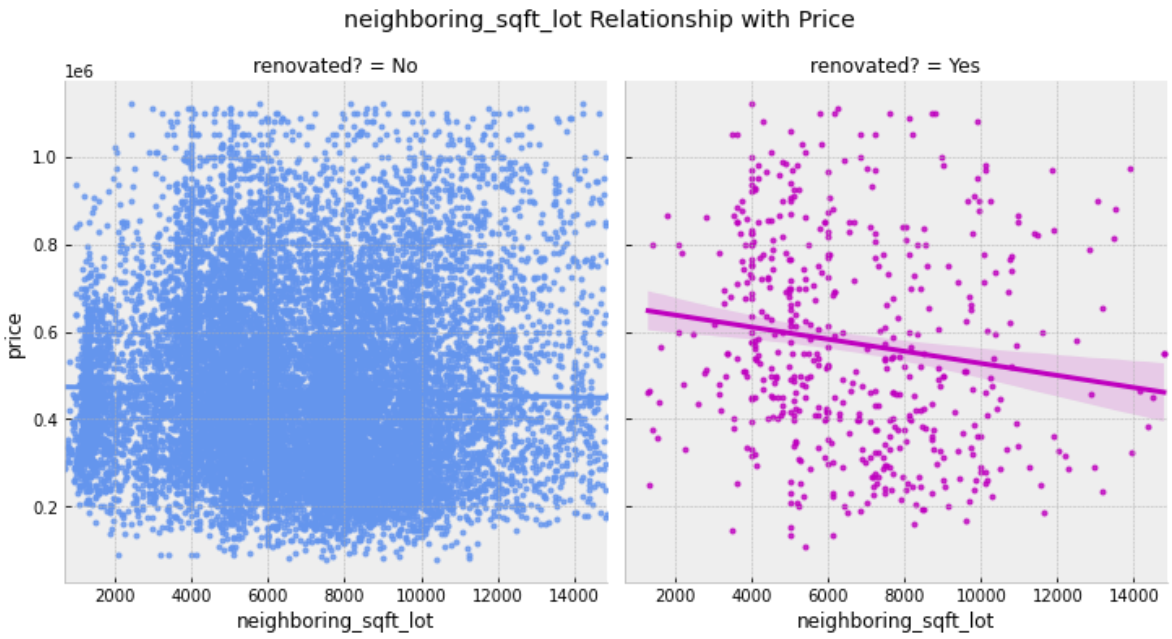

sqft_basement Relationship with Price
renovated? = No
renovated? = Yes
1e6
price
1.0
0.8
0.6

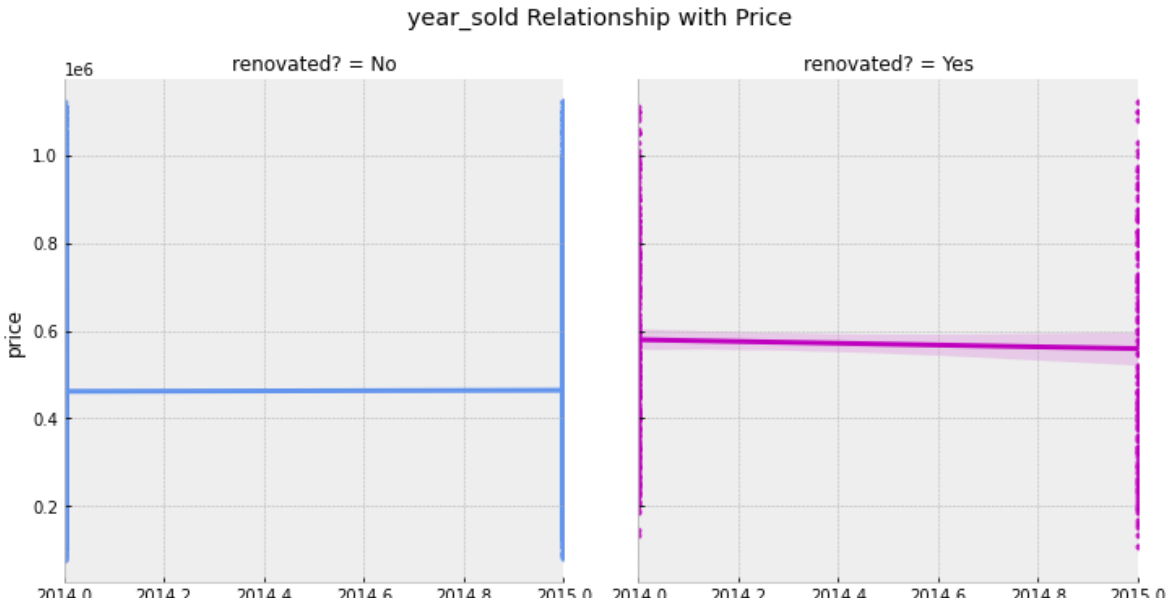
yr_built Relationship with Price
renovated? = No
renovated? = Yes
1e6
price
1.0
0.8
0.6
0.4
0.2
1900
1920
1940
1960
1980
2000
yr_built

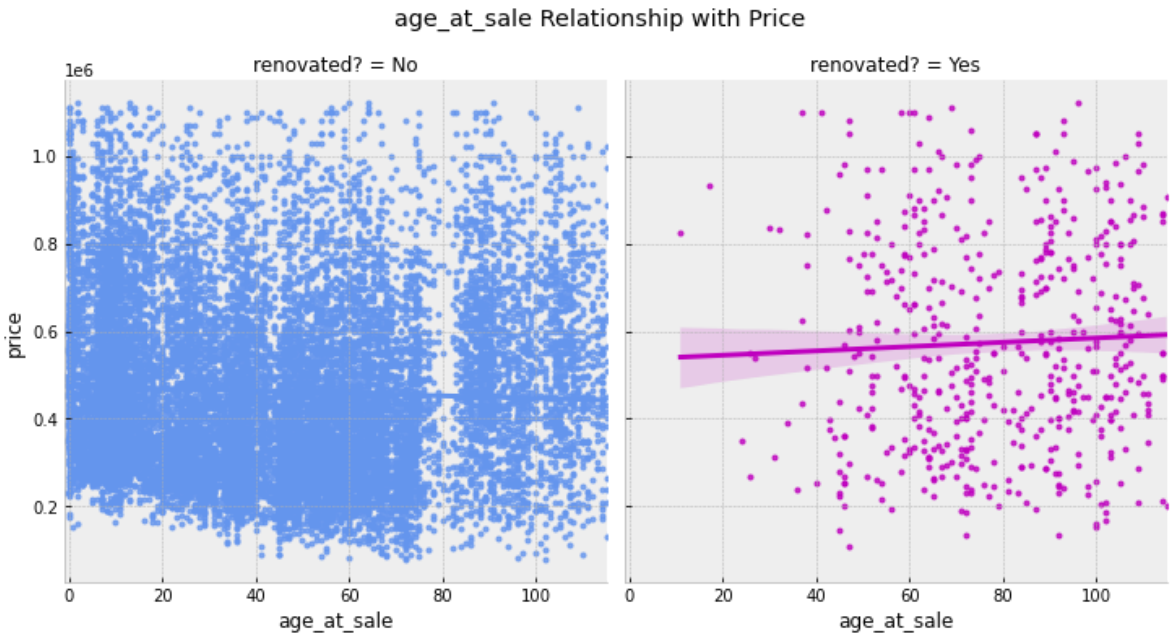

lat Relationship with Price
renovated? = No
renovated? = Yes
1e6
price
1.0
0.8
0.6
0.4
0.2
47.2
47.3
47.4
47.5
47.6
47.7
lat


long Relationship with Price
renovated? = No
renovated? = Yes
1e6
price

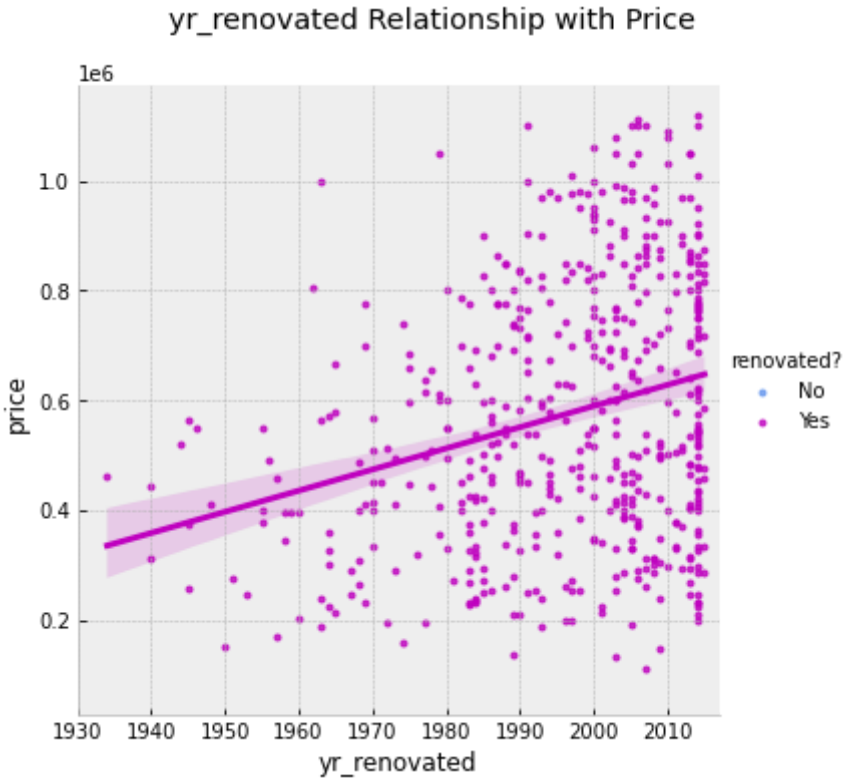

neighboring_sqft_living Relationship with Price
renovated? = No
renovated? = Yes
1e6
price
neighboring_sqft_living


neighboring_sqft_lot Relationship with Price
renovated? = No
renovated? = Yes
1e6
price
neighboring_sqft_lot

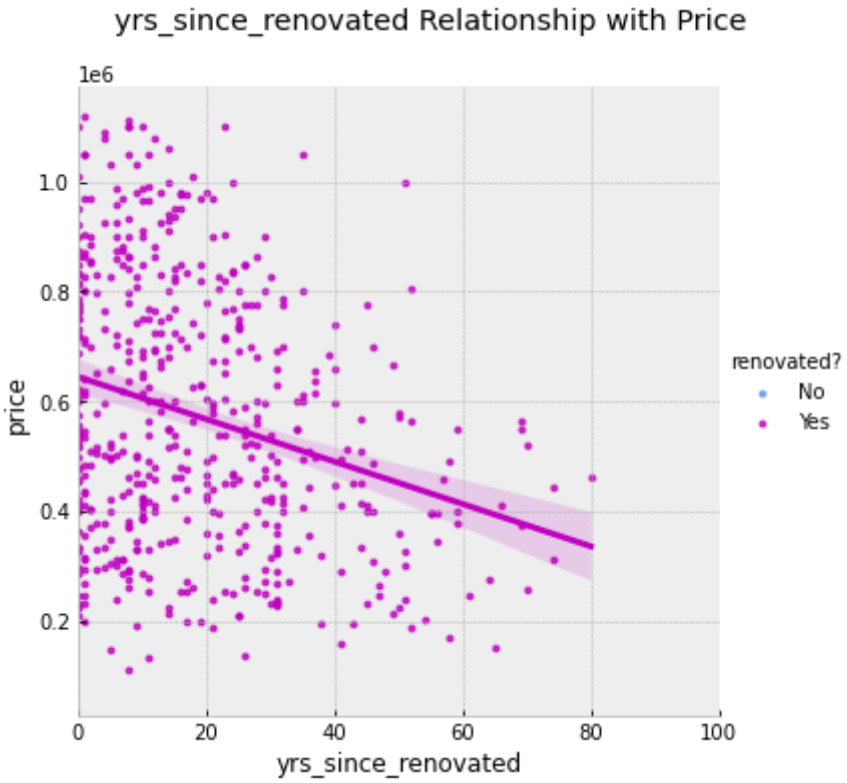

year_sold Relationship with Price
renovated? = No
renovated? = Yes
1e6
price
1.0
0.8
0.6
0.4
0.2
2014.0
2014.2
2014.4
2014.6
2014.8
2015.0


age_at_sale Relationship with Price
renovated? = No
renovated? = Yes
1e6
price
1.0
0.8
0.6
0.4
0.2
0
20
40
60
80
100
age_at_sale

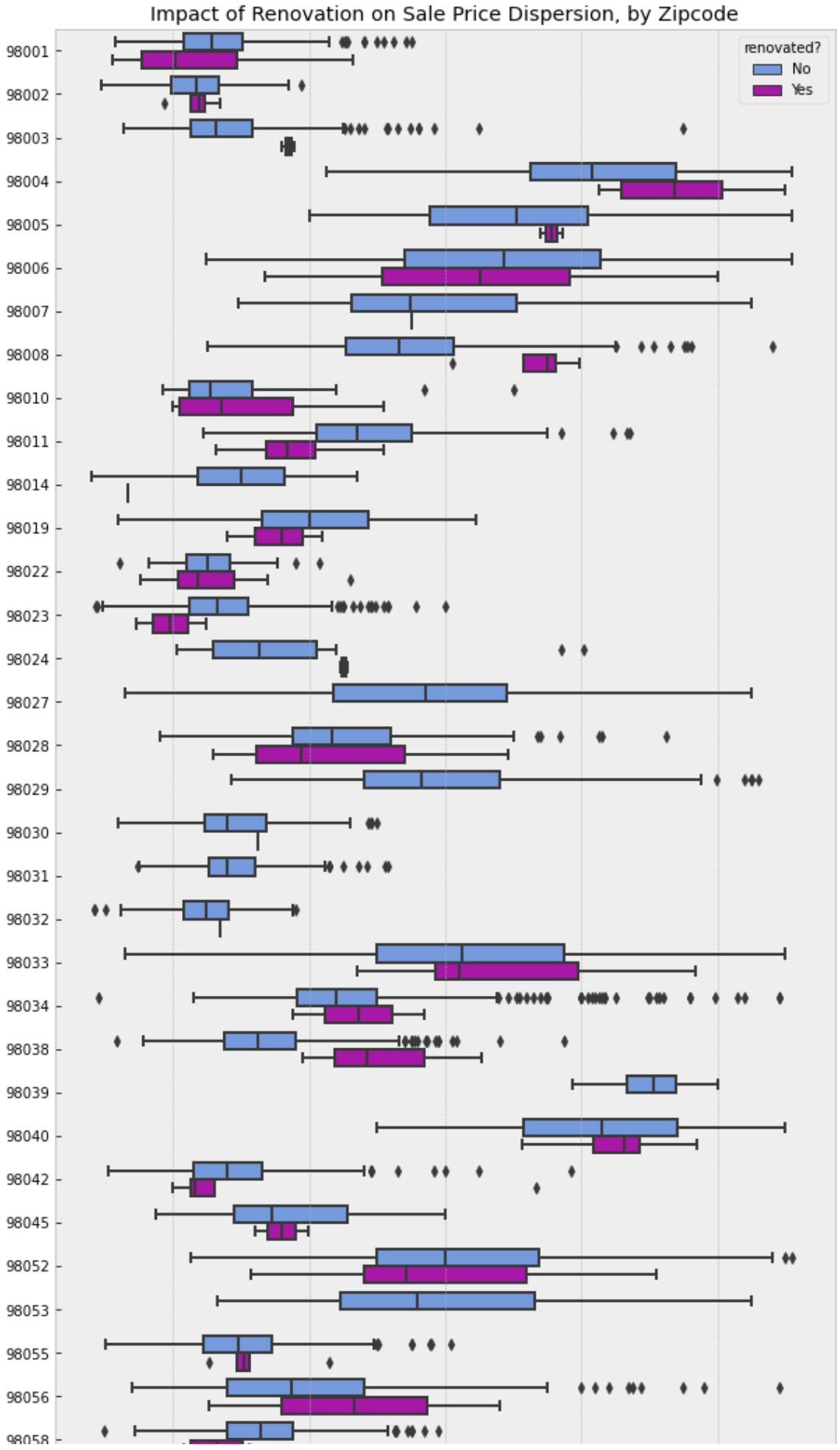
In [35]:
```
column = 'yr_renovated'
sns.lmplot(data=df_outliers_removed, x=column, y='price', hue='renovate
        palette=dict({'No':'cornflowerblue', 'Yes':'m'}), markers='.')
plt.xlim(left=1930, right=2017)
plt.suptitle(t=f'{column} Relationship with Price', fontsize='x-large',
plt.savefig(fname=(f'images/lmplot_{column}'), facecolor='w', bbox_incl
plt.show()
```

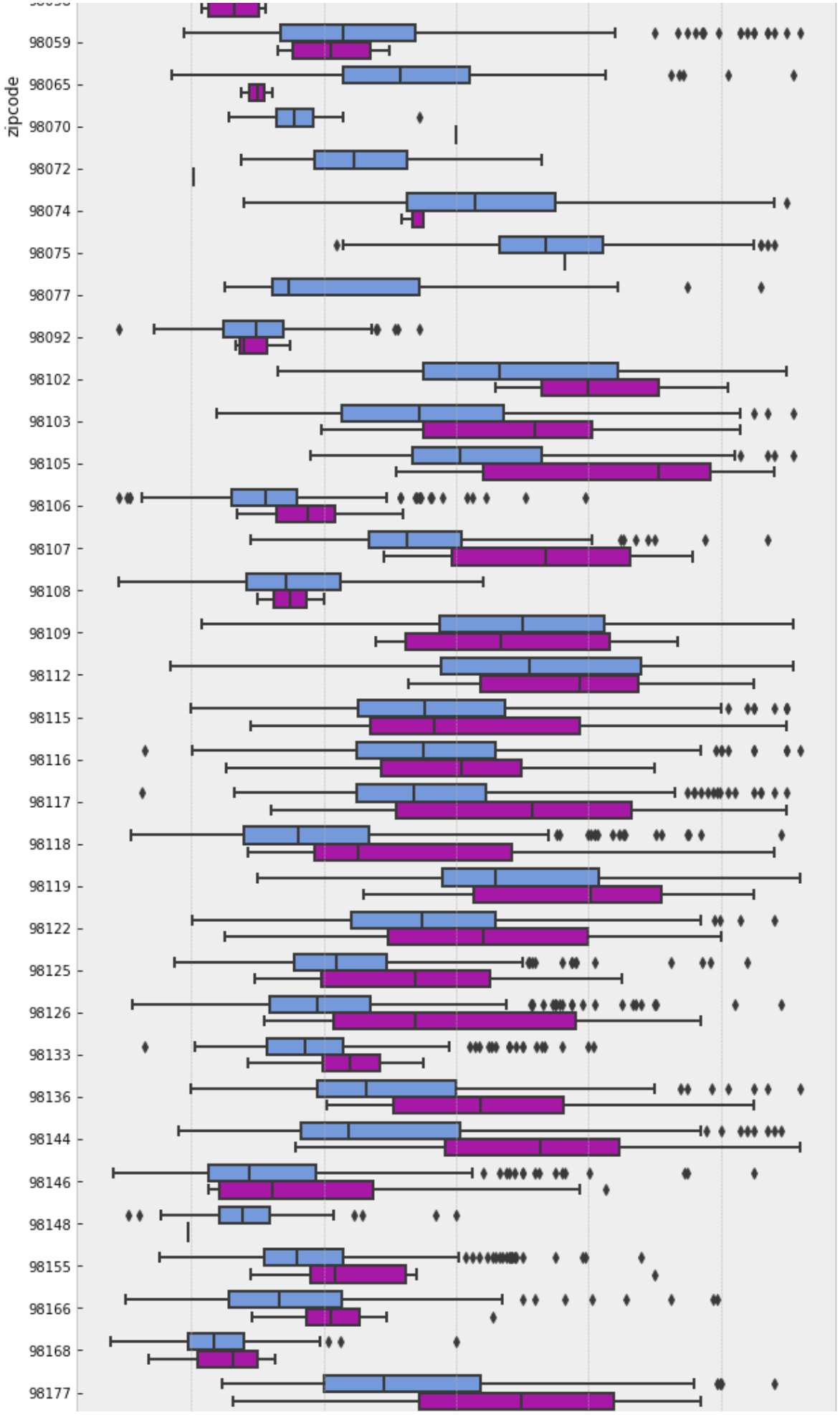


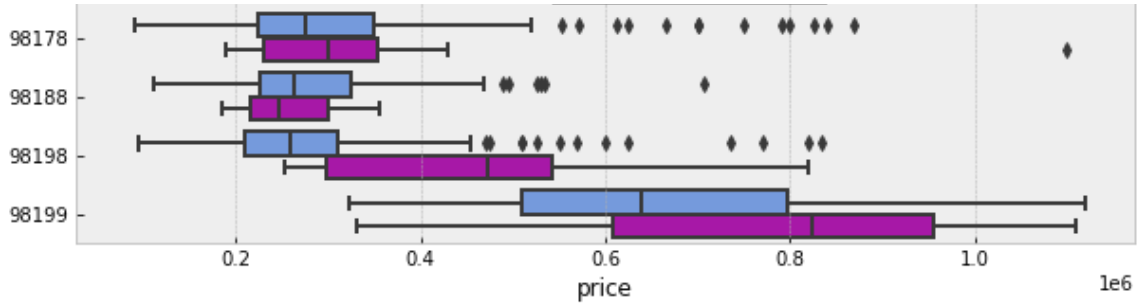
In [36]:
```
column = 'yrs_since_renovated'
sns.lmplot(data=df_outliers_removed, x=column, y='price', hue='renovate
        palette=dict({'No':'cornflowerblue', 'Yes':'m'}), markers='.')
plt.xlim(left=0, right=100)
plt.suptitle(t=f'{column} Relationship with Price', fontsize='x-large',
plt.savefig(fname=(f'images/lmplot_{column}'), facecolor='w', bbox_inch
plt.show()
```



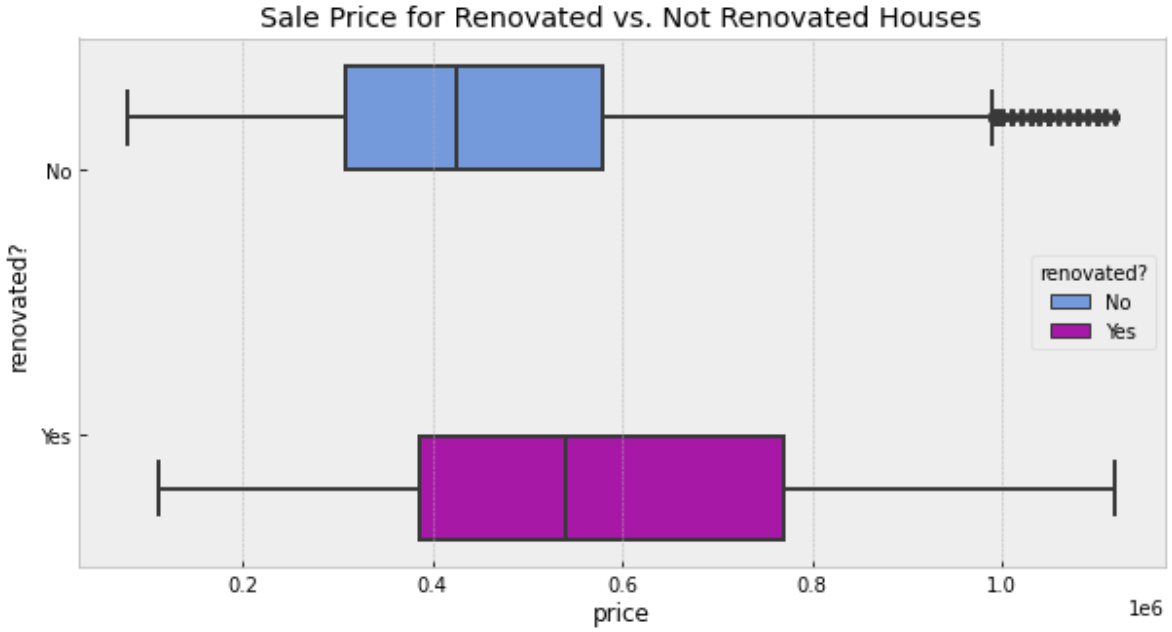
In [37]:
```
plt.figure(figsize=(10,40))
sns.boxplot(orient='h', data=df_outliers_removed, y='zipcode', x='price
          palette=dict({'No':'cornflowerblue', 'Yes':'m'}))
plt.title(f'Impact of Renovation on Sale Price Dispersion, by Zipcode')
plt.savefig(fname=(f'images/boxplot_zipcode_without_outliers'), facecol
plt.show()
```


Impact of Renovation on Sale Price Dispersion, by Zipcode
renovated?
No
Yes
98001
98002
98003
98004
98005
98006
98007
98008
98010
98011
98014
98019
98022
98023
98024
98027
98028
98029
98030
98031
98032
98033
98034
98038
98039
98040
98042
98045
98052
98053
98055
98056
98058


zipcode
98059
98065
98070
98072
98074
98075
98077
98092
98102
98103
98105
98106
98107
98108
98109
98112
98115
98116
98117
98118
98119
98122
98125
98126
98133
98136
98144
98146
98148
98155
98166
98168
98177



In [38]:
```
plt.figure(figsize=(10,5))
sns.boxplot(data=df_outliers_removed, x='price', y='renovated?', hue='r
            palette=dict({'No':'cornflowerblue', 'Yes':'m'}))
plt.title('Sale Price for Renovated vs. Not Renovated Houses')
plt.savefig(fname=(f'images/boxplot_renovated_without_outliers'), facec
              bbox_inches='tight', pad_inches=.5)
plt.show()
```



In [39]:
```
apparent_features = ['bedrooms', 'bathrooms', 'sqft_living', 'floors',
                     'grade', 'sqft_above', 'sqft_basement', 'lat',
                     'neighboring_sqft_living', 'yr_renovated', 'yrs_si
```

The following features appear to have a relatively significant linear relationship with Price:

- bedrooms
- bathrooms
- sqft_living
- grade
- sqft_above
- sqft_basement
- neighboring_sqft_living

Looking in overall King County and by individual zipcode, houses that have had renovations have a higher median price, which supports the idea that renovating improves the value of a house.

Multicollinearity

When constructing a linear regression model with multiple independent variables, it's important to look for multicollinearity between the independent variables. If a model includes multiple features that are actually related to each other, their impact on the target may be overstated in the model.

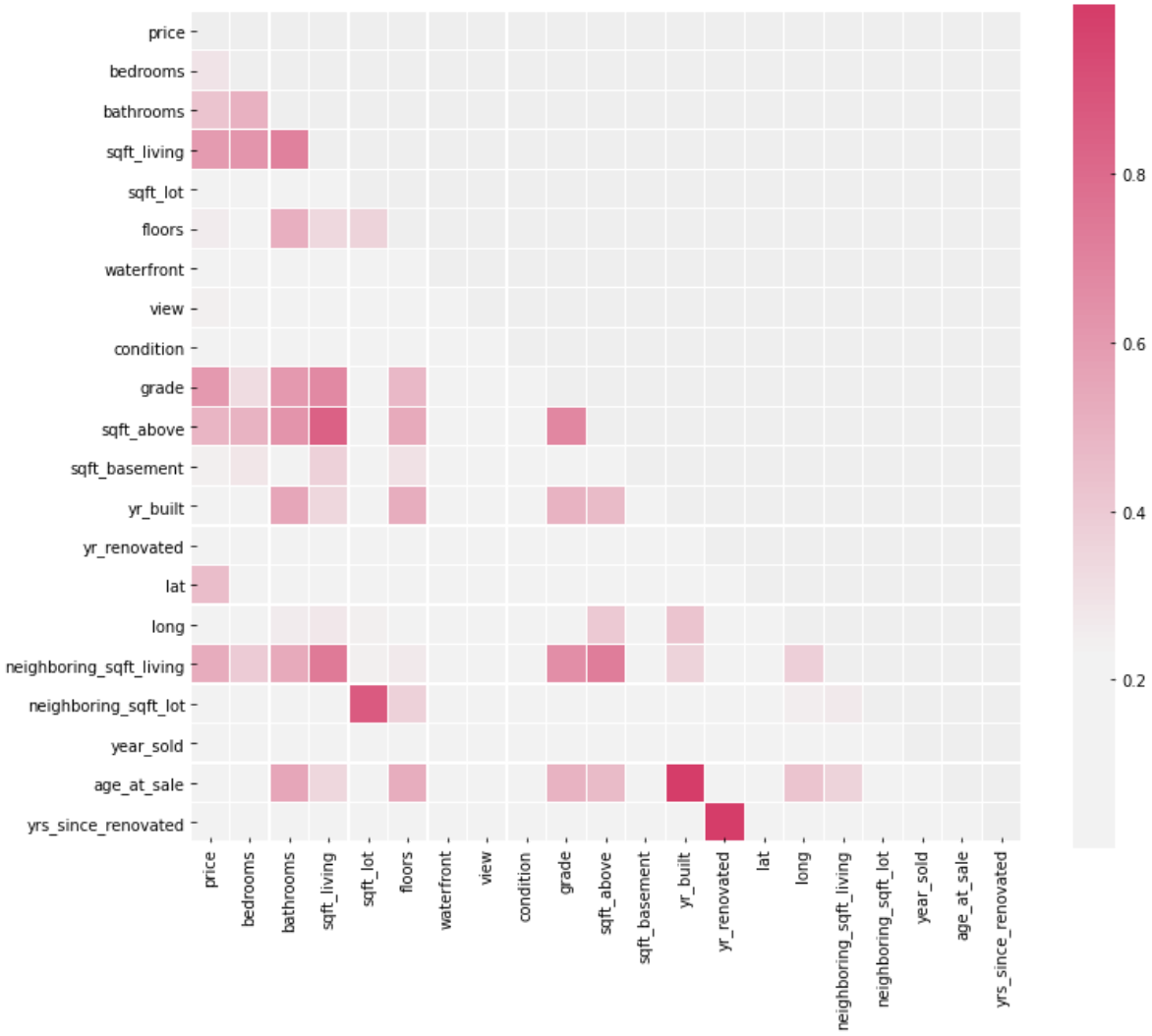
In this section, I look for multicollinearity within my preprocessed dataset.

In [40]:
```
def multicollinearity(data, min_correlation=0.75, figsize=(12,10), anno
    '''
    Parameters:
    data - pd.DataFrame, primary dataset to read from
    min_correlation - float (0-1), the smallest correlation between two
    figsize - tuple, width and height of the resulting figure (default=
    annot - bool, overlay figure with the correlation values for each c

    Returns:
    Dataframe of feature pairs with correlation >= min_correlation
    Heatmap of feature correlations
    '''
    import seaborn as sns

    df = data.corr().abs().stack().reset_index().sort_values(0, ascendi
    df['pairs'] = list(zip(df.level_0, df.level_1))
    df = df.set_index('pairs').drop(columns=['level_0', 'level_1'])
    df.columns = ['correlation']
    df.drop_duplicates(inplace=True)

    mask = np.triu(np.ones_like(data.corr(), dtype=bool))
    cmap = sns.diverging_palette(240, 0, sep=60, s=75, as_cmap=True)
    f, ax = plt.subplots(figsize=figsize)
    sns.heatmap(data.corr().abs(), center=0, linewidths=.3, mask=mask,
                cmap=cmap, square=True)
    return df[df.correlation > min_correlation]
```

In [41]:
```
multicollinearity(df_outliers_removed, annot=False)
```

Out[41]:

pairs	correlation
(price, price)	1.000000
(yr_renovated, yrs_since_renovated)	0.999962
(yr_built, age_at_sale)	0.999879
(sqft_lot, neighboring_sqft_lot)	0.871084
(sqft_above, sqft_living)	0.843652



In [42]:
```
# Remove one out of each multicollinear pair from the apparent features
apparent_features.remove('yr_renovated')
apparent_features.remove('sqft_above')
apparent_features
```

Out[42]:
```
['bedrooms',
 'bathrooms',
 'sqft_living',
 'floors',
 'view',
 'grade',
 'sqft_basement',
 'lat',
 'neighboring_sqft_living',
 'yrs_since_renovated']
```

In [43]:
```
# Create a new df with one out of each multicollinear pair removed
df_simplified = df_outliers_removed.drop(columns=['date_sold', 'yr_renc
                                                  'neighboring_sqft_lot'
df_simplified.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 17744 entries, 0 to 21596
Data columns (total 20 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       17744 non-null  object
 1   price                    17744 non-null  int64
 2   bedrooms                 17744 non-null  int64
 3   bathrooms                17744 non-null  float64
 4   sqft_living              17744 non-null  int64
 5   sqft_lot                 17744 non-null  int64
 6   floors                   17744 non-null  float64
 7   waterfront               17744 non-null  float64
 8   view                     17744 non-null  float64
 9   condition                17744 non-null  int64
 10  grade                    17744 non-null  int64
 11  sqft_basement            17744 non-null  int64
 12  zipcode                  17744 non-null  object
 13  lat                      17744 non-null  float64
 14  long                     17744 non-null  float64
 15  neighboring_sqft_living  17744 non-null  int64
 16  year_sold                17744 non-null  int64
 17  age_at_sale              17744 non-null  int64
 18  renovated?               17744 non-null  object
 19  yrs_since_renovated      17744 non-null  int64
dtypes: float64(6), int64(11), object(3)
memory usage: 3.5+ MB
```

Model

Now that data has been fully preprocessed and explored, it's time to begin modeling!

Workflow & Defining Functions

Workflow:

1. Create baseline dataframe "df"
2. Identify categorical columns and columns to drop from model, put both into lists "columns_cat", "drop_cols"
3. One-hot encode categorical columns to create "df_ohe"
4. Model using "df_ohe" and "target"
5. Interpret results:
    A. R2 should be adequate (looking for ~0.65 and above)
    B. P-values for each feature should be below 0.05 alpha
    C. Residuals should be normally distributed (qq plot)
    D. Residuals should be homoskedastic (scatterplot of residuals)

Repeat until model is maximized.

In [44]:
```
# Function corresponding to workflow step 3
def onehotencode(df, columns_cat, drop_cols=[]):
    '''
    Parameters:
    df = pd.dataframe to model
    columns_cat = list of categorical columns to one-hot encode
    drop_cols = (optional) any columns from overall df to drop before c

    Returns:
    Dataframe with categorical columns one-hot encoded, original columı
    and column names with periods replaced with underscores
    '''
    df_ohe = pd.get_dummies(df.drop(columns=drop_cols), columns=columns
    new_cols = []
    for col in df_ohe.columns:
        new_cols.append(col.replace('.', '_'))
    df_ohe.columns = new_cols
    return df_ohe
```

In [45]:
```
# Function corresponding to workflow step 4
def model(df, target):
    '''
    Parameters:
    df = pd.dataframe, with categorical variables one-hot encoded
    target = string, column name of dependent variable
    exclude_cols = list, columns to exclude from model, due to multico

    Returns:
    Model summary
    QQ plot (to check that residuals are normally distributed)
    Scatter plot of residuals (to check that residuals are homoskedast
    '''
    formula_predictors = ' + '.join(df.drop(columns=target).columns)
    formula = target + ' ~ ' + formula_predictors
    model = ols(formula=formula, data=df).fit()
    display(model.summary())

    fig, ax = plt.subplots(ncols=2, figsize=(10,5))
    sm.graphics.qqplot(data=model.resid, fit=True, line='45', ax=ax[0],
    ax[0].set_title('QQ Plot -\n check for normally distributed residua
    ax[1].scatter(model.predict(df), model.resid)
    ax[1].axhline(y=0, color='black')
    ax[1].set_title('Scatterplot of Errors -\n check for homoskedastici
    plt.tight_layout()
    plt.show();

    return model
```

Model 1

The first model I ran used the raw data before outlier removal. While the R2 value was quite high, indicating that the model was describing ~80% of the variance in sale price, the residuals were quite abnormally distributed. This is when I investigated outliers and realized that there were quite a few.

In [46]:
```
df.head(2)
```

Out[46]:

	id	date_sold	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	water
0	7129300520	2014-10-13	221900	3	1.00	1180	5650	1.0	
1	6414100192	2014-12-09	538000	3	2.25	2570	7242	2.0	

In [47]:
```
# identify df
m1_df = df.copy()

# Identify categorical and drop columns
m1_columns_cat = ['floors', 'waterfront', 'condition', 'zipcode']
m1_drop_cols = ['id', 'renovated?', 'year_sold', 'date_sold']

# one-hot encode categorical columns
m1_df_ohe = onehotencode(df=m1_df, columns_cat=m1_columns_cat, drop_col
m1_df_ohe.head()

# model
model1 = model(df=m1_df_ohe, target=target)
model1
```

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.811
Model:	OLS	**Adj. R-squared:**	0.810
Method:	Least Squares	**F-statistic:**	988.9
Date:	Mon, 19 Apr 2021	**Prob (F-statistic):**	0.00
Time:	16:40:17	**Log-Likelihood:**	-2.8942e+05
No. Observations:	21596	**AIC:**	5.790e+05
Df Residuals:	21502	**BIC:**	5.798e+05
Df Model:	93		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-7.957e+07	7.77e+06	-10.247	0.000	-9.48e+07	-6.44e+07
bedrooms	-2.965e+04	1611.781	-18.395	0.000	-3.28e+04	-2.65e+04
bathrooms	2.485e+04	2642.547	9.405	0.000	1.97e+04	3e+04
sqft_living	104.6241	14.430	7.250	0.000	76.340	132.908
sqft_lot	0.2356	0.038	6.140	0.000	0.160	0.311
view	5.617e+04	1737.455	32.331	0.000	5.28e+04	5.96e+04
grade	5.721e+04	1823.525	31.372	0.000	5.36e+04	6.08e+04
sqft_above	105.6866	14.468	7.305	0.000	77.329	134.044
sqft_basement	26.3853	14.282	1.847	0.065	-1.609	54.380
yr_built	2.944e+04	2344.800	12.556	0.000	2.48e+04	3.4e+04
yr_renovated	-2603.6528	366.441	-7.105	0.000	-3321.904	-1885.402
lat	1.678e+05	6.32e+04	2.654	0.008	4.39e+04	2.92e+05
long	-1.402e+05	4.55e+04	-3.081	0.002	-2.29e+05	-5.1e+04
neighboring_sqft_living	11.3473	2.888	3.929	0.000	5.686	17.009
neighboring_sqft_lot	-0.1401	0.060	-2.320	0.020	-0.258	-0.022

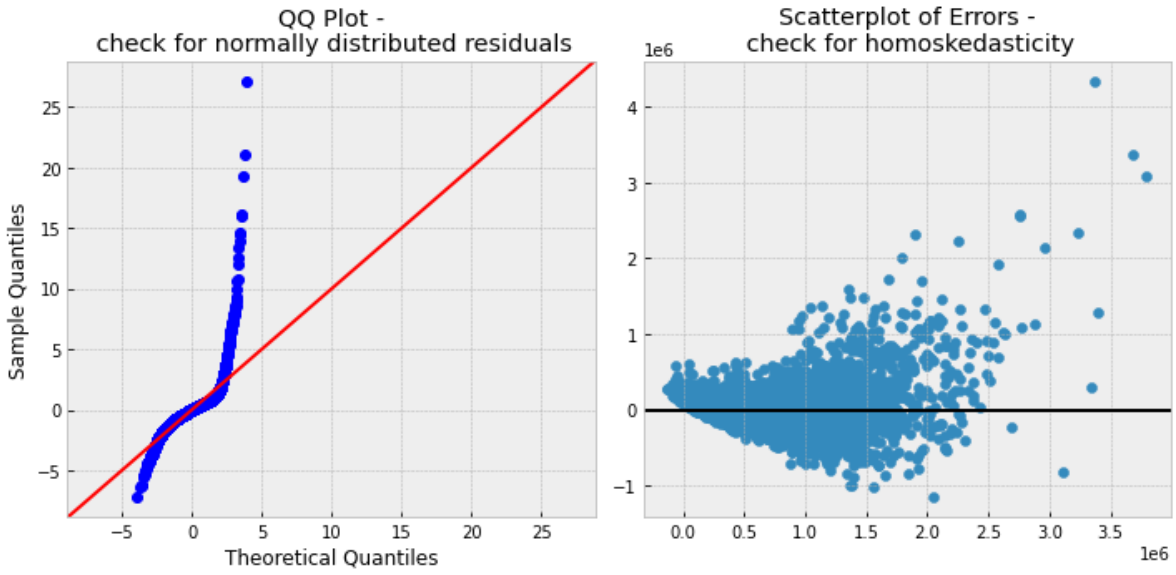
age_at_sale	3.026e+04	2343.988	12.910	0.000	2.57e+04	3.49e+04
yrs_since_renovated	-5332.7640	745.714	-7.151	0.000	-6794.418	-3871.110
floors_1_5	-2.412e+04	4421.401	-5.455	0.000	-3.28e+04	-1.55e+04
floors_2_0	-5.478e+04	3683.704	-14.872	0.000	-6.2e+04	-4.76e+04
floors_2_5	4.085e+04	1.33e+04	3.062	0.002	1.47e+04	6.7e+04
floors_3_0	-8.399e+04	8236.229	-10.197	0.000	-1e+05	-6.78e+04
floors_3_5	2.526e+04	6.09e+04	0.415	0.678	-9.41e+04	1.45e+05
waterfront_1_0	6.888e+05	1.47e+04	46.924	0.000	6.6e+05	7.18e+05
condition_1	6325.3138	1.16e+04	0.547	0.584	-1.63e+04	2.9e+04
condition_2	6.136e+04	1.22e+04	5.044	0.000	3.75e+04	8.52e+04
zipcode_98002	4.025e+04	1.44e+04	2.790	0.005	1.2e+04	6.85e+04
zipcode_98003	-2.28e+04	1.29e+04	-1.767	0.077	-4.81e+04	2492.235
zipcode_98004	7.324e+05	2.35e+04	31.231	0.000	6.86e+05	7.78e+05
zipcode_98005	2.655e+05	2.51e+04	10.595	0.000	2.16e+05	3.15e+05
zipcode_98006	2.429e+05	2.05e+04	11.858	0.000	2.03e+05	2.83e+05
zipcode_98007	2.161e+05	2.59e+04	8.355	0.000	1.65e+05	2.67e+05
zipcode_98008	2.2e+05	2.46e+04	8.955	0.000	1.72e+05	2.68e+05
zipcode_98010	1.026e+05	2.2e+04	4.663	0.000	5.95e+04	1.46e+05
zipcode_98011	5.644e+04	3.2e+04	1.766	0.077	-6203.843	1.19e+05
zipcode_98014	9.732e+04	3.51e+04	2.771	0.006	2.85e+04	1.66e+05
zipcode_98019	6.111e+04	3.46e+04	1.764	0.078	-6774.196	1.29e+05
zipcode_98022	3.94e+04	1.91e+04	2.060	0.039	1903.619	7.69e+04
zipcode_98023	-4.499e+04	1.19e+04	-3.789	0.000	-6.83e+04	-2.17e+04
zipcode_98024	1.594e+05	3.09e+04	5.156	0.000	9.88e+04	2.2e+05
zipcode_98027	1.623e+05	2.1e+04	7.717	0.000	1.21e+05	2.04e+05
zipcode_98028	4.605e+04	3.1e+04	1.483	0.138	-1.48e+04	1.07e+05
zipcode_98029	2.063e+05	2.4e+04	8.581	0.000	1.59e+05	2.53e+05
zipcode_98030	5746.2275	1.42e+04	0.405	0.686	-2.21e+04	3.36e+04
zipcode_98031	1.3e+04	1.48e+04	0.880	0.379	-1.6e+04	4.2e+04
zipcode_98032	-2193.5419	1.72e+04	-0.128	0.898	-3.58e+04	3.14e+04
zipcode_98033	3.139e+05	2.66e+04	11.792	0.000	2.62e+05	3.66e+05
zipcode_98034	1.41e+05	2.85e+04	4.938	0.000	8.5e+04	1.97e+05
zipcode_98038	5.462e+04	1.59e+04	3.425	0.001	2.34e+04	8.59e+04
zipcode_98039	1.272e+06	3.17e+04	40.127	0.000	1.21e+06	1.33e+06
zipcode_98040	4.768e+05	2.07e+04	23.012	0.000	4.36e+05	5.17e+05
zipcode_98042	1.794e+04	1.36e+04	1.321	0.187	-8681.716	4.46e+04
zipcode_98045	1.334e+05	2.95e+04	4.528	0.000	7.57e+04	1.91e+05

zipcode_98052	1.866e+05	2.72e+04	6.864	0.000	1.33e+05	2.4e+05
zipcode_98053	1.581e+05	2.91e+04	5.427	0.000	1.01e+05	2.15e+05
zipcode_98055	2.775e+04	1.65e+04	1.686	0.092	-4517.456	6e+04
zipcode_98056	7.378e+04	1.79e+04	4.128	0.000	3.87e+04	1.09e+05
zipcode_98058	2.231e+04	1.56e+04	1.435	0.151	-8168.084	5.28e+04
zipcode_98059	7.172e+04	1.75e+04	4.090	0.000	3.74e+04	1.06e+05
zipcode_98065	1.008e+05	2.72e+04	3.710	0.000	4.75e+04	1.54e+05
zipcode_98070	-4.504e+04	2.07e+04	-2.177	0.030	-8.56e+04	-4480.383
zipcode_98072	9.449e+04	3.18e+04	2.973	0.003	3.22e+04	1.57e+05
zipcode_98074	1.469e+05	2.58e+04	5.705	0.000	9.64e+04	1.97e+05
zipcode_98075	1.527e+05	2.48e+04	6.169	0.000	1.04e+05	2.01e+05
zipcode_98077	7.344e+04	3.31e+04	2.220	0.026	8612.514	1.38e+05
zipcode_98092	-2.176e+04	1.29e+04	-1.686	0.092	-4.71e+04	3543.375
zipcode_98102	4.427e+05	2.75e+04	16.101	0.000	3.89e+05	4.97e+05
zipcode_98103	2.624e+05	2.58e+04	10.175	0.000	2.12e+05	3.13e+05
zipcode_98105	4.005e+05	2.64e+04	15.150	0.000	3.49e+05	4.52e+05
zipcode_98106	9.627e+04	1.91e+04	5.046	0.000	5.89e+04	1.34e+05
zipcode_98107	2.684e+05	2.66e+04	10.100	0.000	2.16e+05	3.2e+05
zipcode_98108	8.213e+04	2.11e+04	3.899	0.000	4.08e+04	1.23e+05
zipcode_98109	4.199e+05	2.74e+04	15.348	0.000	3.66e+05	4.73e+05
zipcode_98112	5.516e+05	2.43e+04	22.722	0.000	5.04e+05	5.99e+05
zipcode_98115	2.61e+05	2.62e+04	9.971	0.000	2.1e+05	3.12e+05
zipcode_98116	2.259e+05	2.13e+04	10.606	0.000	1.84e+05	2.68e+05
zipcode_98117	2.386e+05	2.65e+04	9.001	0.000	1.87e+05	2.91e+05
zipcode_98118	1.272e+05	1.86e+04	6.835	0.000	9.07e+04	1.64e+05
zipcode_98119	4.038e+05	2.58e+04	15.626	0.000	3.53e+05	4.54e+05
zipcode_98122	2.76e+05	2.3e+04	11.977	0.000	2.31e+05	3.21e+05
zipcode_98125	1.261e+05	2.83e+04	4.461	0.000	7.07e+04	1.82e+05
zipcode_98126	1.384e+05	1.96e+04	7.074	0.000	1e+05	1.77e+05
zipcode_98133	8.565e+04	2.92e+04	2.935	0.003	2.85e+04	1.43e+05
zipcode_98136	1.888e+05	2.01e+04	9.415	0.000	1.49e+05	2.28e+05
zipcode_98144	2.254e+05	2.14e+04	10.519	0.000	1.83e+05	2.67e+05
zipcode_98146	6.529e+04	1.79e+04	3.647	0.000	3.02e+04	1e+05
zipcode_98148	3.445e+04	2.44e+04	1.415	0.157	-1.33e+04	8.22e+04
zipcode_98155	6.544e+04	3.03e+04	2.156	0.031	5959.083	1.25e+05
zipcode_98166	1.872e+04	1.64e+04	1.143	0.253	-1.34e+04	5.08e+04
zipcode_98168	3.924e+04	1.73e+04	2.266	0.023	5301.345	7.32e+04

zipcode_98177	1.293e+05	3.05e+04	4.244	0.000	6.96e+04	1.89e+05
zipcode_98178	5185.1794	1.79e+04	0.290	0.772	-2.99e+04	4.03e+04
zipcode_98188	7816.5705	1.84e+04	0.426	0.670	-2.82e+04	4.38e+04
zipcode_98198	-1.89e+04	1.39e+04	-1.359	0.174	-4.62e+04	8364.745
zipcode_98199	3.05e+05	2.52e+04	12.121	0.000	2.56e+05	3.54e+05

Omnibus:	20759.003	**Durbin-Watson:**	1.995
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	4237764.288
Skew:	4.144	**Prob(JB):**	0.00
Kurtosis:	71.123	**Cond. No.**	3.60e+08

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.6e+08. This might indicate that there are strong multicollinearity or other numerical problems.



Out[47]: <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7fe

Model 2

This model runs with the same features as model 1, but trained on a dataset without outliers.

This model's R2 score improved to .84, and the residuals appear much more normally distributed. Now I notice that some features have p-values greater than 0.05 and decide to deal with them and rerun the model.

In [48]:
```
df_simplified.head(2)
```

Out[48]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view
0	7129300520	221900	3	1.00	1180	5650	1.0	0.0	0.0
1	6414100192	538000	3	2.25	2570	7242	2.0	0.0	0.0

In [49]:
```
# identify df
m2_df = df_simplified.copy()

# Identify categorical and drop columns
m2_columns_cat = ['floors', 'waterfront', 'condition', 'zipcode']
m2_drop_cols = ['id', 'renovated?', 'year_sold']

# one-hot encode categorical columns
m2_df_ohe = onehotencode(df=m2_df, columns_cat=m2_columns_cat, drop_col
m2_df_ohe.head()

# model
model2 = model(df=m2_df_ohe, target=target)
model2
```

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.839
Model:	OLS	**Adj. R-squared:**	0.838
Method:	Least Squares	**F-statistic:**	1033.
Date:	Mon, 19 Apr 2021	**Prob (F-statistic):**	0.00
Time:	16:40:18	**Log-Likelihood:**	-2.2586e+05
No. Observations:	17744	**AIC:**	4.519e+05
Df Residuals:	17654	**BIC:**	4.526e+05
Df Model:	89		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-7.375e+06	4.21e+06	-1.753	0.080	-1.56e+07	8.71e+05
bedrooms	-5100.7303	953.008	-5.352	0.000	-6968.720	-3232.740
bathrooms	1.23e+04	1548.894	7.942	0.000	9265.770	1.53e+04
sqft_living	112.5430	2.078	54.153	0.000	108.469	116.617
sqft_lot	1.7058	0.276	6.187	0.000	1.165	2.246
view	3.349e+04	1167.618	28.679	0.000	3.12e+04	3.58e+04
grade	4.407e+04	1109.626	39.712	0.000	4.19e+04	4.62e+04
sqft_basement	-37.4964	2.346	-15.986	0.000	-42.094	-32.899
lat	-2.346e+04	3.98e+04	-0.590	0.555	-1.01e+05	5.45e+04
long	-6.642e+04	3.16e+04	-2.100	0.036	-1.28e+05	-4427.512
neighboring_sqft_living	31.3687	1.880	16.686	0.000	27.684	35.054
age_at_sale	546.3296	39.126	13.963	0.000	469.638	623.021
yrs_since_renovated	-31.0585	3.924	-7.916	0.000	-38.749	-23.368
floors_1_5	5054.4094	2558.085	1.976	0.048	40.311	1.01e+04
floors_2_0	-8577.4754	2276.565	-3.768	0.000	-1.3e+04	-4115.184

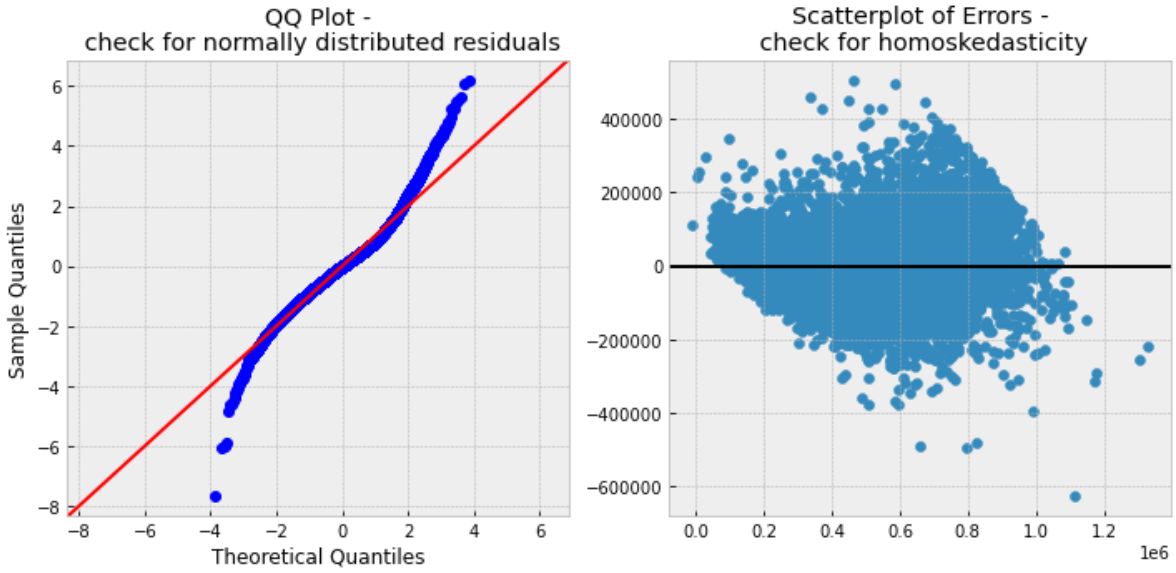
floors_2_5	-1.542e+04	8772.960	-1.758	0.079	-3.26e+04	1776.715
floors_3_0	-4.245e+04	4615.163	-9.198	0.000	-5.15e+04	-3.34e+04
floors_3_5	-5.505e+04	3.36e+04	-1.636	0.102	-1.21e+05	1.09e+04
waterfront_1_0	2.559e+05	2.01e+04	12.754	0.000	2.17e+05	2.95e+05
condition_1	4.203e+04	6822.888	6.161	0.000	2.87e+04	5.54e+04
condition_2	8.531e+04	7141.146	11.946	0.000	7.13e+04	9.93e+04
zipcode_98002	2.251e+04	7904.809	2.848	0.004	7018.178	3.8e+04
zipcode_98003	-2482.9584	7226.940	-0.344	0.731	-1.66e+04	1.17e+04
zipcode_98004	5.295e+05	1.51e+04	35.168	0.000	5e+05	5.59e+05
zipcode_98005	3.526e+05	1.55e+04	22.806	0.000	3.22e+05	3.83e+05
zipcode_98006	2.899e+05	1.26e+04	22.932	0.000	2.65e+05	3.15e+05
zipcode_98007	2.724e+05	1.55e+04	17.613	0.000	2.42e+05	3.03e+05
zipcode_98008	2.62e+05	1.51e+04	17.370	0.000	2.32e+05	2.92e+05
zipcode_98010	9.574e+04	1.61e+04	5.949	0.000	6.42e+04	1.27e+05
zipcode_98011	1.579e+05	1.96e+04	8.077	0.000	1.2e+05	1.96e+05
zipcode_98014	1.57e+05	2.82e+04	5.573	0.000	1.02e+05	2.12e+05
zipcode_98019	1.206e+05	2.17e+04	5.555	0.000	7.81e+04	1.63e+05
zipcode_98022	2.228e+04	1.27e+04	1.752	0.080	-2642.143	4.72e+04
zipcode_98023	-2.092e+04	6958.390	-3.007	0.003	-3.46e+04	-7284.324
zipcode_98024	1.769e+05	2.4e+04	7.383	0.000	1.3e+05	2.24e+05
zipcode_98027	2.504e+05	1.37e+04	18.299	0.000	2.24e+05	2.77e+05
zipcode_98028	1.425e+05	1.91e+04	7.454	0.000	1.05e+05	1.8e+05
zipcode_98029	2.55e+05	1.5e+04	16.979	0.000	2.26e+05	2.84e+05
zipcode_98030	1.456e+04	8039.296	1.811	0.070	-1198.775	3.03e+04
zipcode_98031	2.372e+04	8514.899	2.785	0.005	7025.391	4.04e+04
zipcode_98032	5703.5679	9545.055	0.598	0.550	-1.3e+04	2.44e+04
zipcode_98033	3.361e+05	1.65e+04	20.354	0.000	3.04e+05	3.68e+05
zipcode_98034	2.009e+05	1.77e+04	11.358	0.000	1.66e+05	2.36e+05
zipcode_98038	5.416e+04	1e+04	5.411	0.000	3.45e+04	7.38e+04
zipcode_98039	6.616e+05	3.62e+04	18.297	0.000	5.91e+05	7.32e+05
zipcode_98040	4.355e+05	1.34e+04	32.554	0.000	4.09e+05	4.62e+05
zipcode_98042	2.451e+04	8348.048	2.935	0.003	8142.304	4.09e+04
zipcode_98045	1.354e+05	1.96e+04	6.914	0.000	9.7e+04	1.74e+05
zipcode_98052	2.767e+05	1.68e+04	16.495	0.000	2.44e+05	3.1e+05
zipcode_98053	2.831e+05	1.9e+04	14.925	0.000	2.46e+05	3.2e+05
zipcode_98055	5.03e+04	9627.224	5.225	0.000	3.14e+04	6.92e+04
zipcode_98056	1.159e+05	1.07e+04	10.834	0.000	9.49e+04	1.37e+05

zipcode_98058	4.813e+04	9397.542	5.122	0.000	2.97e+04	6.66e+04
zipcode_98059	1.176e+05	1.07e+04	11.008	0.000	9.67e+04	1.39e+05
zipcode_98065	1.591e+05	1.74e+04	9.168	0.000	1.25e+05	1.93e+05
zipcode_98070	5.27e+04	2.22e+04	2.370	0.018	9116.386	9.63e+04
zipcode_98072	1.733e+05	2.02e+04	8.581	0.000	1.34e+05	2.13e+05
zipcode_98074	2.382e+05	1.61e+04	14.774	0.000	2.07e+05	2.7e+05
zipcode_98075	2.487e+05	1.59e+04	15.603	0.000	2.17e+05	2.8e+05
zipcode_98077	1.868e+05	2.42e+04	7.732	0.000	1.39e+05	2.34e+05
zipcode_98092	-1.468e+04	7654.292	-1.918	0.055	-2.97e+04	320.060
zipcode_98102	4.128e+05	1.65e+04	24.964	0.000	3.8e+05	4.45e+05
zipcode_98103	3.296e+05	1.6e+04	20.658	0.000	2.98e+05	3.61e+05
zipcode_98105	3.843e+05	1.63e+04	23.546	0.000	3.52e+05	4.16e+05
zipcode_98106	1.246e+05	1.15e+04	10.874	0.000	1.02e+05	1.47e+05
zipcode_98107	3.286e+05	1.63e+04	20.192	0.000	2.97e+05	3.6e+05
zipcode_98108	1.239e+05	1.24e+04	10.013	0.000	9.96e+04	1.48e+05
zipcode_98109	4.113e+05	1.67e+04	24.696	0.000	3.79e+05	4.44e+05
zipcode_98112	4.304e+05	1.51e+04	28.418	0.000	4.01e+05	4.6e+05
zipcode_98115	3.281e+05	1.62e+04	20.300	0.000	2.96e+05	3.6e+05
zipcode_98116	2.941e+05	1.3e+04	22.635	0.000	2.69e+05	3.2e+05
zipcode_98117	3.198e+05	1.64e+04	19.457	0.000	2.88e+05	3.52e+05
zipcode_98118	1.711e+05	1.12e+04	15.239	0.000	1.49e+05	1.93e+05
zipcode_98119	4.006e+05	1.58e+04	25.278	0.000	3.7e+05	4.32e+05
zipcode_98122	3.115e+05	1.4e+04	22.281	0.000	2.84e+05	3.39e+05
zipcode_98125	2.051e+05	1.74e+04	11.784	0.000	1.71e+05	2.39e+05
zipcode_98126	1.935e+05	1.18e+04	16.357	0.000	1.7e+05	2.17e+05
zipcode_98133	1.641e+05	1.81e+04	9.089	0.000	1.29e+05	1.99e+05
zipcode_98136	2.577e+05	1.21e+04	21.336	0.000	2.34e+05	2.81e+05
zipcode_98144	2.49e+05	1.3e+04	19.147	0.000	2.23e+05	2.74e+05
zipcode_98146	1.109e+05	1.06e+04	10.428	0.000	9.01e+04	1.32e+05
zipcode_98148	5.639e+04	1.31e+04	4.299	0.000	3.07e+04	8.21e+04
zipcode_98155	1.476e+05	1.88e+04	7.868	0.000	1.11e+05	1.84e+05
zipcode_98166	9.814e+04	1.01e+04	9.717	0.000	7.83e+04	1.18e+05
zipcode_98168	5.276e+04	1.03e+04	5.132	0.000	3.26e+04	7.29e+04
zipcode_98177	2.137e+05	1.88e+04	11.341	0.000	1.77e+05	2.51e+05
zipcode_98178	5.381e+04	1.05e+04	5.134	0.000	3.33e+04	7.44e+04
zipcode_98188	3.473e+04	1.03e+04	3.362	0.001	1.45e+04	5.5e+04
zipcode_98198	2.681e+04	7989.049	3.356	0.001	1.11e+04	4.25e+04

zipcode_98199	3.602e+05	1.56e+04	23.142	0.000	3.3e+05	3.91e+05

Omnibus:	1440.015	**Durbin-Watson:**	1.999
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	5034.073
Skew:	0.378	**Prob(JB):**	0.00
Kurtosis:	5.498	**Cond. No.**	5.62e+07

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.62e+07. This might indicate that there are strong multicollinearity or other numerical problems.



Out[49]: <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7fe

Model 3

For model 3, I decided to make the following changes:

- Remove the following features from the model all together because of their high p-values: sqft_basement, lat and long, neighboring_sqft_lot
- Revise the "floor" feature to consider 0.5 story NOT part of the floor count. In other words, I will create an integer column for floors with the following options: 1 (1- or 1.5-story), 2 (2- or 2.5-story), 3 (3- or 3.5-story). I made this decision because I noticed that the half-floor options had high p-values.

This model still has a high R2 at ~0.836. It turns out the "floors" feature still is not a good predictor, so I'll remove it in model 4. I also notice that "bedrooms" has a negative coefficient. I wonder how my model might be influenced by a ratio of bathrooms to bedrooms, rather than the raw number of either?

In [50]:
```
df_simplified['floors'].value_counts()
```

Out[50]:
```
1.0    9083
2.0    6412
1.5    1583
3.0     565
2.5      95
3.5       6
Name: floors, dtype: int64
```

In [51]:
```
m3_df = df_simplified.copy()

m3_df['floors'].replace(to_replace=1.5, value=1.0)
m3_df['floors'].replace(to_replace=2.5, value=2.0)
m3_df['floors'].replace(to_replace=3.5, value=3.0)
m3_df['floors'] = m3_df['floors'].astype(int)
```

In [52]:
```
# Identify categorical columns
m3_columns_cat = ['waterfront', 'condition', 'zipcode']
m3_drop_cols = ['id', 'renovated?', 'year_sold', 'sqft_basement',
                'lat', 'long']

# one-hot encode categorical columns
m3_df_ohe = onehotencode(df=m3_df, columns_cat=m3_columns_cat, drop_col
m3_df_ohe.head()

# model
model3 = model(df=m3_df_ohe, target=target)
model3
```

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.836
Model:	OLS	**Adj. R-squared:**	0.835
Method:	Least Squares	**F-statistic:**	1094.
Date:	Mon, 19 Apr 2021	**Prob (F-statistic):**	0.00
Time:	16:40:19	**Log-Likelihood:**	-2.2605e+05
No. Observations:	17744	**AIC:**	4.523e+05
Df Residuals:	17661	**BIC:**	4.529e+05
Df Model:	82		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-3.782e+05	1.25e+04	-30.340	0.000	-4.03e+05	-3.54e+05
bedrooms	-4447.7715	955.074	-4.657	0.000	-6319.810	-2575.733
bathrooms	8812.7627	1539.750	5.724	0.000	5794.701	1.18e+04
sqft_living	98.7117	1.846	53.469	0.000	95.093	102.330
sqft_lot	1.7455	0.274	6.366	0.000	1.208	2.283
floors	649.7511	1721.665	0.377	0.706	-2724.882	4024.384
view	3.058e+04	1169.017	26.155	0.000	2.83e+04	3.29e+04
grade	4.713e+04	1106.215	42.608	0.000	4.5e+04	4.93e+04
neighboring_sqft_living	37.0663	1.869	19.827	0.000	33.402	40.731
age_at_sale	662.5125	37.996	17.437	0.000	588.038	736.987
yrs_since_renovated	-30.5891	3.955	-7.734	0.000	-38.342	-22.836
waterfront_1_0	2.586e+05	2.03e+04	12.759	0.000	2.19e+05	2.98e+05
condition_1	3.999e+04	6889.853	5.804	0.000	2.65e+04	5.35e+04
condition_2	8.089e+04	7207.739	11.223	0.000	6.68e+04	9.5e+04
zipcode_98002	2.063e+04	7777.515	2.652	0.008	5380.604	3.59e+04
zipcode_98003	-2727.4624	7182.281	-0.380	0.704	-1.68e+04	1.14e+04
zipcode_98004	5.067e+05	8826.363	57.405	0.000	4.89e+05	5.24e+05

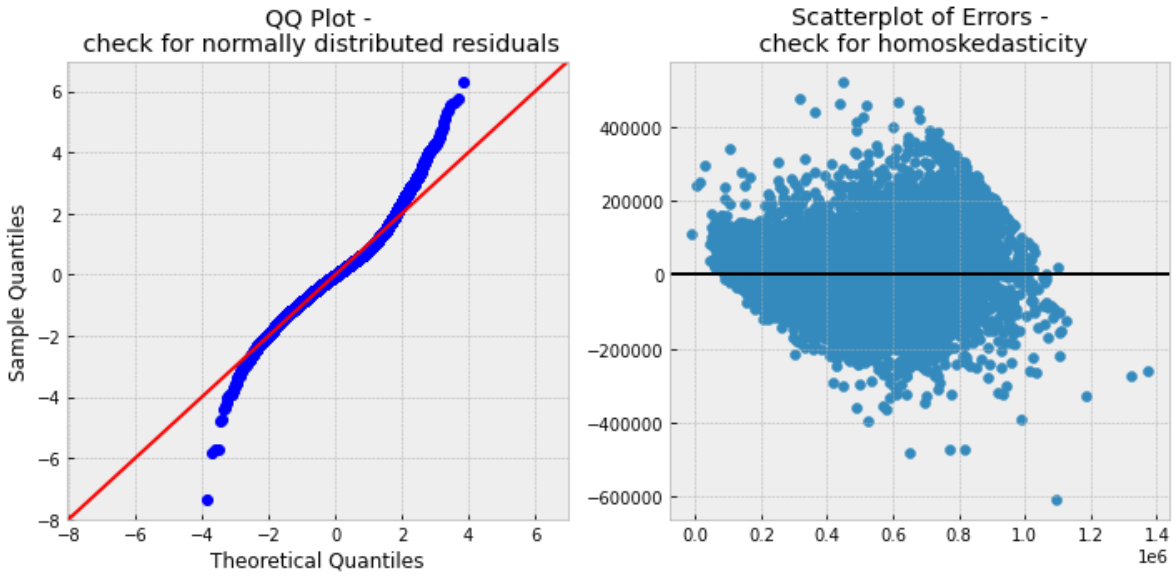
zipcode_98005	3.315e+05	9599.283	34.538	0.000	3.13e+05	3.5e+05
zipcode_98006	2.675e+05	6714.362	39.836	0.000	2.54e+05	2.81e+05
zipcode_98007	2.512e+05	8783.199	28.598	0.000	2.34e+05	2.68e+05
zipcode_98008	2.386e+05	7249.038	32.917	0.000	2.24e+05	2.53e+05
zipcode_98010	7.805e+04	1.4e+04	5.573	0.000	5.06e+04	1.05e+05
zipcode_98011	1.394e+05	7925.218	17.584	0.000	1.24e+05	1.55e+05
zipcode_98014	1.156e+05	1.73e+04	6.688	0.000	8.17e+04	1.5e+05
zipcode_98019	9.346e+04	8699.750	10.742	0.000	7.64e+04	1.11e+05
zipcode_98022	1.082e+04	8519.365	1.270	0.204	-5877.676	2.75e+04
zipcode_98023	-1.835e+04	6345.992	-2.892	0.004	-3.08e+04	-5914.031
zipcode_98024	1.483e+05	1.83e+04	8.088	0.000	1.12e+05	1.84e+05
zipcode_98027	2.253e+05	7281.835	30.940	0.000	2.11e+05	2.4e+05
zipcode_98028	1.257e+05	7257.959	17.321	0.000	1.11e+05	1.4e+05
zipcode_98029	2.326e+05	6995.769	33.253	0.000	2.19e+05	2.46e+05
zipcode_98030	8448.7669	7294.682	1.158	0.247	-5849.527	2.27e+04
zipcode_98031	1.464e+04	7225.151	2.026	0.043	473.882	2.88e+04
zipcode_98032	-911.3142	9308.420	-0.098	0.922	-1.92e+04	1.73e+04
zipcode_98033	3.182e+05	6702.095	47.471	0.000	3.05e+05	3.31e+05
zipcode_98034	1.829e+05	6211.324	29.443	0.000	1.71e+05	1.95e+05
zipcode_98038	4.153e+04	6269.919	6.623	0.000	2.92e+04	5.38e+04
zipcode_98039	6.392e+05	3.41e+04	18.722	0.000	5.72e+05	7.06e+05
zipcode_98040	4.172e+05	8731.703	47.784	0.000	4e+05	4.34e+05
zipcode_98042	1.518e+04	6367.494	2.384	0.017	2696.238	2.77e+04
zipcode_98045	1.005e+05	8944.728	11.236	0.000	8.3e+04	1.18e+05
zipcode_98052	2.549e+05	6220.749	40.973	0.000	2.43e+05	2.67e+05
zipcode_98053	2.674e+05	7434.752	35.961	0.000	2.53e+05	2.82e+05
zipcode_98055	3.954e+04	7259.174	5.447	0.000	2.53e+04	5.38e+04
zipcode_98056	1.048e+05	6552.285	15.992	0.000	9.19e+04	1.18e+05
zipcode_98058	3.545e+04	6577.131	5.389	0.000	2.26e+04	4.83e+04
zipcode_98059	1.087e+05	6619.791	16.419	0.000	9.57e+04	1.22e+05
zipcode_98065	1.335e+05	7133.733	18.720	0.000	1.2e+05	1.48e+05
zipcode_98070	6.021e+04	2.14e+04	2.808	0.005	1.82e+04	1.02e+05
zipcode_98072	1.51e+05	8869.407	17.020	0.000	1.34e+05	1.68e+05
zipcode_98074	2.168e+05	6714.780	32.284	0.000	2.04e+05	2.3e+05
zipcode_98075	2.318e+05	7735.434	29.961	0.000	2.17e+05	2.47e+05
zipcode_98077	1.619e+05	1.55e+04	10.466	0.000	1.32e+05	1.92e+05
zipcode_98092	-1.79e+04	7246.019	-2.470	0.014	-3.21e+04	-3693.711

zipcode_98102	3.917e+05	1.04e+04	37.633	0.000	3.71e+05	4.12e+05
zipcode_98103	3.1e+05	6391.225	48.497	0.000	2.97e+05	3.22e+05
zipcode_98105	3.647e+05	8249.366	44.213	0.000	3.49e+05	3.81e+05
zipcode_98106	1.172e+05	6840.656	17.130	0.000	1.04e+05	1.31e+05
zipcode_98107	3.111e+05	7409.418	41.990	0.000	2.97e+05	3.26e+05
zipcode_98108	1.104e+05	7960.516	13.874	0.000	9.48e+04	1.26e+05
zipcode_98109	3.952e+05	1.05e+04	37.685	0.000	3.75e+05	4.16e+05
zipcode_98112	4.101e+05	8421.307	48.701	0.000	3.94e+05	4.27e+05
zipcode_98115	3.086e+05	6306.889	48.927	0.000	2.96e+05	3.21e+05
zipcode_98116	2.843e+05	7067.456	40.223	0.000	2.7e+05	2.98e+05
zipcode_98117	3.059e+05	6384.528	47.906	0.000	2.93e+05	3.18e+05
zipcode_98118	1.565e+05	6371.678	24.561	0.000	1.44e+05	1.69e+05
zipcode_98119	3.848e+05	8661.537	44.431	0.000	3.68e+05	4.02e+05
zipcode_98122	2.947e+05	7405.029	39.795	0.000	2.8e+05	3.09e+05
zipcode_98125	1.877e+05	6563.079	28.594	0.000	1.75e+05	2.01e+05
zipcode_98126	1.873e+05	6836.435	27.403	0.000	1.74e+05	2.01e+05
zipcode_98133	1.497e+05	6288.979	23.809	0.000	1.37e+05	1.62e+05
zipcode_98136	2.485e+05	7386.072	33.648	0.000	2.34e+05	2.63e+05
zipcode_98144	2.327e+05	7034.715	33.078	0.000	2.19e+05	2.46e+05
zipcode_98146	1.08e+05	7128.502	15.144	0.000	9.4e+04	1.22e+05
zipcode_98148	5.694e+04	1.21e+04	4.692	0.000	3.32e+04	8.07e+04
zipcode_98155	1.341e+05	6513.078	20.596	0.000	1.21e+05	1.47e+05
zipcode_98166	9.514e+04	7987.421	11.911	0.000	7.95e+04	1.11e+05
zipcode_98168	4.589e+04	7432.144	6.174	0.000	3.13e+04	6.05e+04
zipcode_98177	2.003e+05	7752.204	25.840	0.000	1.85e+05	2.16e+05
zipcode_98178	3.968e+04	7279.373	5.452	0.000	2.54e+04	5.4e+04
zipcode_98188	2.776e+04	8879.896	3.126	0.002	1.04e+04	4.52e+04
zipcode_98198	2.511e+04	7277.202	3.451	0.001	1.08e+04	3.94e+04
zipcode_98199	3.457e+05	7268.398	47.565	0.000	3.31e+05	3.6e+05

Omnibus:	1416.774	**Durbin-Watson:**	1.995
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	4711.536
Skew:	0.387	**Prob(JB):**	0.00
Kurtosis:	5.403	**Cond. No.**	5.69e+05

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

[2] The condition number is large, 5.69e+05. This might indicate that there are strong multicollinearity or other numerical problems.



Out[52]: <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7fe

Model 4

Removed floors feature, engineered a bath_to_bed_ratio feature by dividing the number of bedrooms into the number of bathrooms.

This appears to be the best model

In [53]:
```
# Identify df
m4_df = m3_df.copy()
```

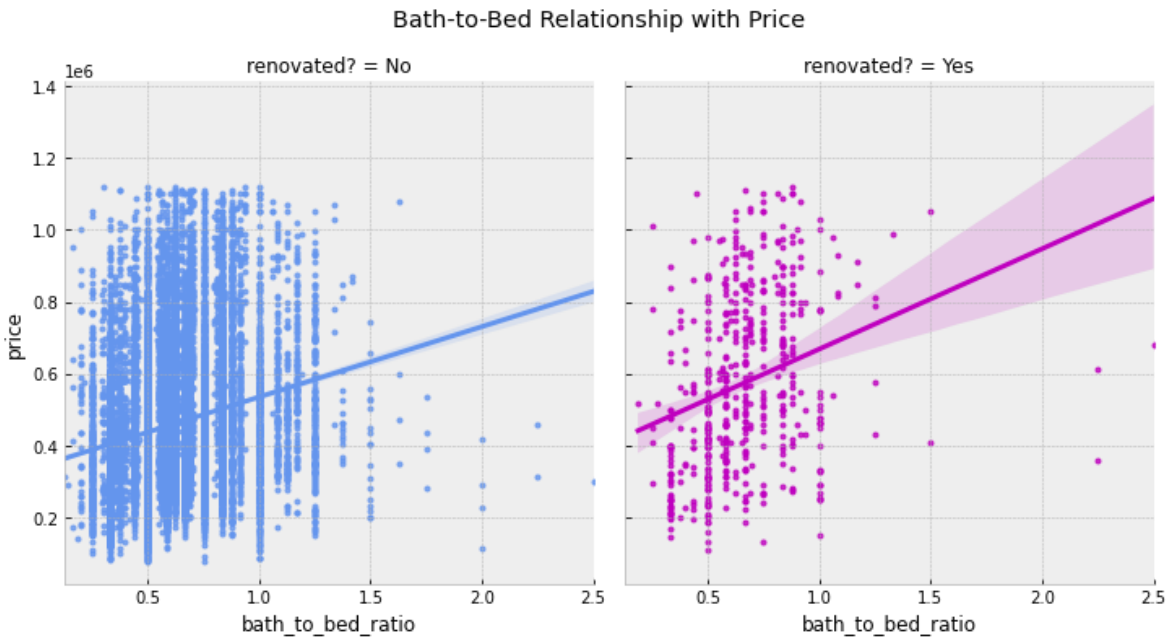
In [54]:
```
# <1 = more bedrooms than bathrooms
m4_df['bath_to_bed_ratio'] = m4_df['bathrooms'] / m4_df['bedrooms']
m4_df.sort_values('bath_to_bed_ratio')
```

Out[54]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront
12029	2991000160	312500	4	0.50	2300	5570	2	0.0
12596	3812400455	291000	7	1.00	2350	8636	1	0.0
3609	809000945	563000	6	1.00	1730	2760	1	0.0
12327	5132000140	415000	6	1.00	1370	5080	1	0.0
13135	3693900155	950000	6	1.00	2330	5000	1	0.0
...	...	...	...	...	...	...	...	...
5364	6450304600	315000	1	2.25	1940	2550	2	0.0
5163	1720800305	611900	1	2.25	1220	2100	2	0.0
5577	537000130	360000	1	2.25	2060	10600	1	0.0
14173	7461400360	299000	1	2.50	1980	7521	1	0.0
4406	4083304190	680000	1	2.50	1820	3008	2	0.0

17744 rows × 21 columns

In [55]:
```
sns.lmplot(data=m4_df, x='bath_to_bed_ratio', y='price', hue='renovatec
        palette=dict({'No':'cornflowerblue', 'Yes':'m'}), markers='.')
plt.suptitle(t=f'Bath-to-Bed Relationship with Price', fontsize='x-larg
plt.savefig(fname=(f'images/lmplot_bath_to_bed_ratio'), facecolor='w',
plt.show()
```



In [56]:
```
# Identify categorical columns
m4_columns_cat = ['waterfront', 'condition', 'zipcode']
m4_drop_cols = ['id', 'renovated?', 'year_sold', 'sqft_basement',
                'lat', 'long', 'floors', 'bedrooms', 'bathrooms']

# one-hot encode categorical columns
m4_df_ohe = onehotencode(df=m4_df, columns_cat=m4_columns_cat, drop_col
m4_df_ohe.head()

# model
model4 = model(df=m4_df_ohe, target=target)
model4
```

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.835
Model:	OLS	**Adj. R-squared:**	0.834
Method:	Least Squares	**F-statistic:**	1119.
Date:	Mon, 19 Apr 2021	**Prob (F-statistic):**	0.00
Time:	16:40:22	**Log-Likelihood:**	-2.2607e+05
No. Observations:	17744	**AIC:**	4.523e+05
Df Residuals:	17663	**BIC:**	4.529e+05
Df Model:	80		
Covariance Type:	nonrobust		

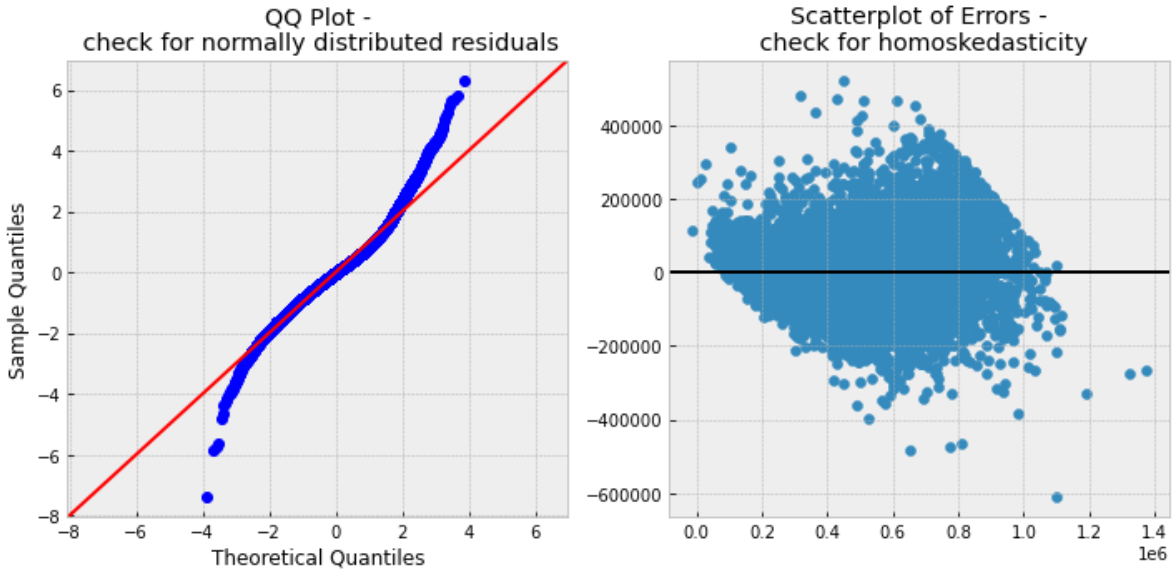
	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-3.816e+05	1.21e+04	-31.668	0.000	-4.05e+05	-3.58e+05
sqft_living	99.6909	1.423	70.036	0.000	96.901	102.481
sqft_lot	1.6118	0.266	6.067	0.000	1.091	2.133
view	3.083e+04	1169.661	26.359	0.000	2.85e+04	3.31e+04
grade	4.758e+04	1096.709	43.383	0.000	4.54e+04	4.97e+04
neighboring_sqft_living	37.3070	1.865	20.006	0.000	33.652	40.962
age_at_sale	606.9057	33.769	17.972	0.000	540.716	673.096
yrs_since_renovated	-32.6901	3.933	-8.313	0.000	-40.398	-24.982
bath_to_bed_ratio	1.116e+04	3587.842	3.110	0.002	4126.911	1.82e+04
waterfront_1_0	2.588e+05	2.03e+04	12.759	0.000	2.19e+05	2.99e+05
condition_1	3.966e+04	6895.415	5.752	0.000	2.61e+04	5.32e+04
condition_2	8.138e+04	7206.443	11.292	0.000	6.73e+04	9.55e+04
zipcode_98002	2.086e+04	7785.311	2.679	0.007	5597.258	3.61e+04
zipcode_98003	-2521.5229	7187.976	-0.351	0.726	-1.66e+04	1.16e+04
zipcode_98004	5.069e+05	8836.059	57.369	0.000	4.9e+05	5.24e+05
zipcode_98005	3.32e+05	9604.672	34.564	0.000	3.13e+05	3.51e+05
zipcode_98006	2.674e+05	6720.724	39.791	0.000	2.54e+05	2.81e+05

zipcode_98007	2.508e+05	8789.835	28.528	0.000	2.34e+05	2.68e+05
zipcode_98008	2.382e+05	7248.756	32.860	0.000	2.24e+05	2.52e+05
zipcode_98010	7.868e+04	1.4e+04	5.614	0.000	5.12e+04	1.06e+05
zipcode_98011	1.397e+05	7933.118	17.614	0.000	1.24e+05	1.55e+05
zipcode_98014	1.156e+05	1.73e+04	6.678	0.000	8.17e+04	1.49e+05
zipcode_98019	9.443e+04	8704.060	10.849	0.000	7.74e+04	1.11e+05
zipcode_98022	1.119e+04	8526.948	1.313	0.189	-5519.668	2.79e+04
zipcode_98023	-1.824e+04	6347.397	-2.874	0.004	-3.07e+04	-5800.865
zipcode_98024	1.491e+05	1.84e+04	8.123	0.000	1.13e+05	1.85e+05
zipcode_98027	2.261e+05	7288.973	31.017	0.000	2.12e+05	2.4e+05
zipcode_98028	1.261e+05	7265.365	17.357	0.000	1.12e+05	1.4e+05
zipcode_98029	2.338e+05	7000.297	33.394	0.000	2.2e+05	2.47e+05
zipcode_98030	8777.5371	7301.698	1.202	0.229	-5534.509	2.31e+04
zipcode_98031	1.509e+04	7231.489	2.086	0.037	911.089	2.93e+04
zipcode_98032	-1586.7657	9317.368	-0.170	0.865	-1.98e+04	1.67e+04
zipcode_98033	3.185e+05	6708.822	47.472	0.000	3.05e+05	3.32e+05
zipcode_98034	1.831e+05	6212.141	29.478	0.000	1.71e+05	1.95e+05
zipcode_98038	4.21e+04	6275.090	6.708	0.000	2.98e+04	5.44e+04
zipcode_98039	6.383e+05	3.42e+04	18.677	0.000	5.71e+05	7.05e+05
zipcode_98040	4.174e+05	8739.701	47.761	0.000	4e+05	4.35e+05
zipcode_98042	1.527e+04	6373.348	2.396	0.017	2778.877	2.78e+04
zipcode_98045	1.009e+05	8952.249	11.272	0.000	8.34e+04	1.18e+05
zipcode_98052	2.551e+05	6226.716	40.976	0.000	2.43e+05	2.67e+05
zipcode_98053	2.677e+05	7412.366	36.109	0.000	2.53e+05	2.82e+05
zipcode_98055	3.999e+04	7267.565	5.503	0.000	2.57e+04	5.42e+04
zipcode_98056	1.048e+05	6557.332	15.976	0.000	9.19e+04	1.18e+05
zipcode_98058	3.532e+04	6582.387	5.366	0.000	2.24e+04	4.82e+04
zipcode_98059	1.084e+05	6626.033	16.362	0.000	9.54e+04	1.21e+05
zipcode_98065	1.343e+05	7133.682	18.820	0.000	1.2e+05	1.48e+05
zipcode_98070	6.139e+04	2.15e+04	2.861	0.004	1.93e+04	1.03e+05
zipcode_98072	1.509e+05	8876.015	17.002	0.000	1.34e+05	1.68e+05
zipcode_98074	2.171e+05	6719.535	32.306	0.000	2.04e+05	2.3e+05
zipcode_98075	2.319e+05	7737.718	29.969	0.000	2.17e+05	2.47e+05
zipcode_98077	1.616e+05	1.55e+04	10.438	0.000	1.31e+05	1.92e+05
zipcode_98092	-1.804e+04	7253.337	-2.487	0.013	-3.23e+04	-3822.278
zipcode_98102	3.934e+05	1.04e+04	37.814	0.000	3.73e+05	4.14e+05
zipcode_98103	3.114e+05	6374.014	48.855	0.000	2.99e+05	3.24e+05

zipcode_98105	3.655e+05	8253.952	44.288	0.000	3.49e+05	3.82e+05
zipcode_98106	1.176e+05	6848.009	17.173	0.000	1.04e+05	1.31e+05
zipcode_98107	3.128e+05	7403.620	42.251	0.000	2.98e+05	3.27e+05
zipcode_98108	1.115e+05	7968.205	13.988	0.000	9.58e+04	1.27e+05
zipcode_98109	3.967e+05	1.05e+04	37.798	0.000	3.76e+05	4.17e+05
zipcode_98112	4.111e+05	8426.073	48.783	0.000	3.95e+05	4.28e+05
zipcode_98115	3.094e+05	6311.806	49.012	0.000	2.97e+05	3.22e+05
zipcode_98116	2.854e+05	7073.440	40.346	0.000	2.72e+05	2.99e+05
zipcode_98117	3.07e+05	6388.542	48.053	0.000	2.94e+05	3.2e+05
zipcode_98118	1.572e+05	6377.261	24.649	0.000	1.45e+05	1.7e+05
zipcode_98119	3.867e+05	8664.253	44.628	0.000	3.7e+05	4.04e+05
zipcode_98122	2.96e+05	7405.831	39.970	0.000	2.81e+05	3.11e+05
zipcode_98125	1.881e+05	6568.448	28.632	0.000	1.75e+05	2.01e+05
zipcode_98126	1.881e+05	6839.037	27.510	0.000	1.75e+05	2.02e+05
zipcode_98133	1.499e+05	6293.221	23.814	0.000	1.38e+05	1.62e+05
zipcode_98136	2.494e+05	7391.070	33.749	0.000	2.35e+05	2.64e+05
zipcode_98144	2.334e+05	7041.412	33.153	0.000	2.2e+05	2.47e+05
zipcode_98146	1.08e+05	7133.991	15.142	0.000	9.4e+04	1.22e+05
zipcode_98148	5.782e+04	1.21e+04	4.759	0.000	3.4e+04	8.16e+04
zipcode_98155	1.343e+05	6519.235	20.601	0.000	1.22e+05	1.47e+05
zipcode_98166	9.514e+04	7994.902	11.900	0.000	7.95e+04	1.11e+05
zipcode_98168	4.603e+04	7436.070	6.190	0.000	3.15e+04	6.06e+04
zipcode_98177	2.006e+05	7757.477	25.859	0.000	1.85e+05	2.16e+05
zipcode_98178	3.971e+04	7286.481	5.450	0.000	2.54e+04	5.4e+04
zipcode_98188	2.77e+04	8888.464	3.116	0.002	1.03e+04	4.51e+04
zipcode_98198	2.504e+04	7281.961	3.439	0.001	1.08e+04	3.93e+04
zipcode_98199	3.467e+05	7274.187	47.665	0.000	3.32e+05	3.61e+05

Omnibus:	1416.204	**Durbin-Watson:**	1.996
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	4690.418
Skew:	0.388	**Prob(JB):**	0.00
Kurtosis:	5.396	**Cond. No.**	5.68e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.68e+05. This might indicate that there are strong multicollinearity or other numerical problems.



Out[56]: <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7fe

Model 5

Just for comparison's sake, I decided to remove a few more features that didn't have a very clear linear relationship with price when they were plotted earlier: sqft_lot, yr_built, waterfront, condition.

The R2 on model 4 was higher, with no other significant changes to the model.

In [57]:
```
# identify df
m5_df = m4_df.copy()

# Identify categorical columns
m5_columns_cat = ['zipcode']
m5_drop_cols = ['id', 'renovated?', 'year_sold', 'floors', 'sqft_baseme
                'lat', 'long', 'sqft_lot', 'waterfront', 'condition']

# one-hot encode categorical columns
m5_df_ohe = onehotencode(df=m5_df, columns_cat=m5_columns_cat, drop_col
m5_df_ohe.head()

# model
model5 = model(df=m5_df_ohe, target=target)
model5
```

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.831
Model:	OLS	**Adj. R-squared:**	0.831
Method:	Least Squares	**F-statistic:**	1116.
Date:	Mon, 19 Apr 2021	**Prob (F-statistic):**	0.00
Time:	16:40:23	**Log-Likelihood:**	-2.2627e+05
No. Observations:	17744	**AIC:**	4.527e+05
Df Residuals:	17665	**BIC:**	4.533e+05
Df Model:	78		
Covariance Type:	nonrobust		

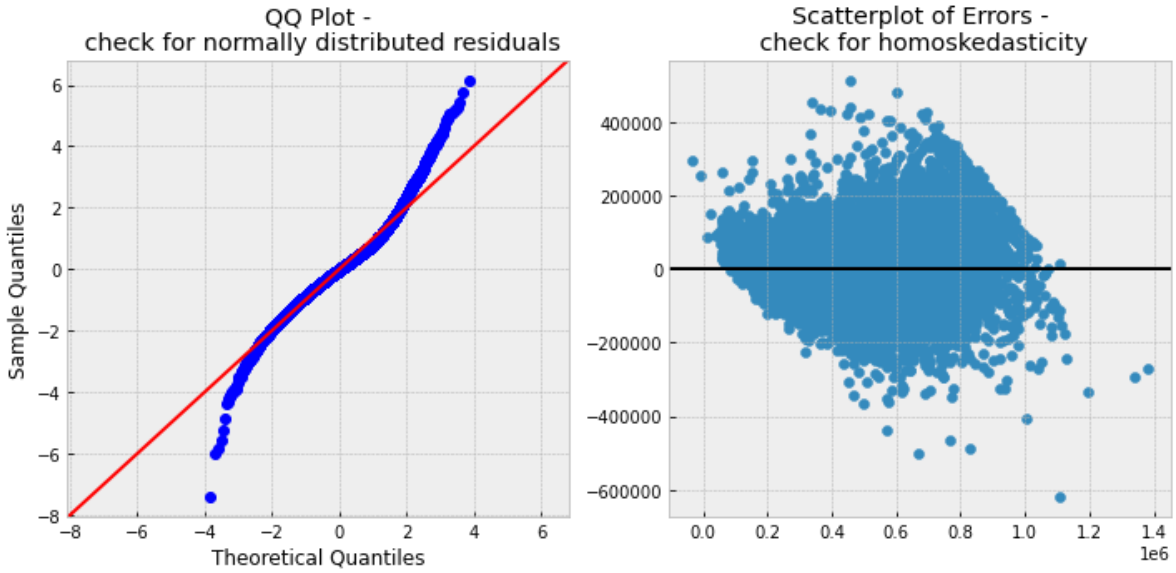
	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-2.874e+05	1.2e+04	-23.884	0.000	-3.11e+05	-2.64e+05
bedrooms	-1.95e+04	2020.947	-9.650	0.000	-2.35e+04	-1.55e+04
bathrooms	3.649e+04	3345.536	10.908	0.000	2.99e+04	4.31e+04
sqft_living	100.1055	1.853	54.015	0.000	96.473	103.738
view	3.393e+04	1158.987	29.279	0.000	3.17e+04	3.62e+04
grade	4.77e+04	1109.237	43.007	0.000	4.55e+04	4.99e+04
neighboring_sqft_living	34.6454	1.891	18.317	0.000	30.938	38.353
age_at_sale	828.0506	33.009	25.085	0.000	763.349	892.752
yrs_since_renovated	-26.3559	3.960	-6.656	0.000	-34.118	-18.594
bath_to_bed_ratio	-8.323e+04	9581.438	-8.687	0.000	-1.02e+05	-6.45e+04
zipcode_98002	1.971e+04	7863.864	2.506	0.012	4294.347	3.51e+04
zipcode_98003	-3676.7764	7270.546	-0.506	0.613	-1.79e+04	1.06e+04
zipcode_98004	5.075e+05	8939.123	56.770	0.000	4.9e+05	5.25e+05
zipcode_98005	3.35e+05	9715.076	34.481	0.000	3.16e+05	3.54e+05
zipcode_98006	2.727e+05	6794.947	40.136	0.000	2.59e+05	2.86e+05

zipcode_98007	2.516e+05	8895.354	28.286	0.000	2.34e+05	2.69e+05
zipcode_98008	2.382e+05	7333.947	32.478	0.000	2.24e+05	2.53e+05
zipcode_98010	8.225e+04	1.42e+04	5.802	0.000	5.45e+04	1.1e+05
zipcode_98011	1.401e+05	8025.348	17.463	0.000	1.24e+05	1.56e+05
zipcode_98014	1.149e+05	1.75e+04	6.560	0.000	8.05e+04	1.49e+05
zipcode_98019	9.41e+04	8805.294	10.687	0.000	7.68e+04	1.11e+05
zipcode_98022	1.505e+04	8620.228	1.746	0.081	-1843.591	3.19e+04
zipcode_98023	-2.083e+04	6420.297	-3.244	0.001	-3.34e+04	-8246.150
zipcode_98024	1.469e+05	1.86e+04	7.909	0.000	1.1e+05	1.83e+05
zipcode_98027	2.226e+05	7348.816	30.290	0.000	2.08e+05	2.37e+05
zipcode_98028	1.262e+05	7350.049	17.176	0.000	1.12e+05	1.41e+05
zipcode_98029	2.273e+05	7035.861	32.310	0.000	2.14e+05	2.41e+05
zipcode_98030	6823.8567	7385.540	0.924	0.356	-7652.527	2.13e+04
zipcode_98031	1.464e+04	7314.976	2.001	0.045	297.807	2.9e+04
zipcode_98032	-1779.1308	9424.040	-0.189	0.850	-2.03e+04	1.67e+04
zipcode_98033	3.196e+05	6784.484	47.110	0.000	3.06e+05	3.33e+05
zipcode_98034	1.818e+05	6282.037	28.937	0.000	1.69e+05	1.94e+05
zipcode_98038	3.965e+04	6333.393	6.260	0.000	2.72e+04	5.21e+04
zipcode_98039	6.377e+05	3.46e+04	18.446	0.000	5.7e+05	7.06e+05
zipcode_98040	4.214e+05	8838.627	47.675	0.000	4.04e+05	4.39e+05
zipcode_98042	1.716e+04	6441.658	2.664	0.008	4535.068	2.98e+04
zipcode_98045	1.011e+05	9050.823	11.165	0.000	8.33e+04	1.19e+05
zipcode_98052	2.539e+05	6296.310	40.323	0.000	2.42e+05	2.66e+05
zipcode_98053	2.692e+05	7508.512	35.858	0.000	2.55e+05	2.84e+05
zipcode_98055	3.698e+04	7335.085	5.042	0.000	2.26e+04	5.14e+04
zipcode_98056	1.086e+05	6617.618	16.407	0.000	9.56e+04	1.22e+05
zipcode_98058	3.533e+04	6656.364	5.308	0.000	2.23e+04	4.84e+04
zipcode_98059	1.1e+05	6701.308	16.420	0.000	9.69e+04	1.23e+05
zipcode_98065	1.295e+05	7198.221	17.994	0.000	1.15e+05	1.44e+05
zipcode_98070	9.773e+04	2.16e+04	4.528	0.000	5.54e+04	1.4e+05
zipcode_98072	1.511e+05	8980.561	16.825	0.000	1.33e+05	1.69e+05
zipcode_98074	2.17e+05	6799.170	31.921	0.000	2.04e+05	2.3e+05
zipcode_98075	2.305e+05	7827.617	29.445	0.000	2.15e+05	2.46e+05
zipcode_98077	1.646e+05	1.57e+04	10.513	0.000	1.34e+05	1.95e+05
zipcode_98092	-1.94e+04	7331.307	-2.646	0.008	-3.38e+04	-5030.483
zipcode_98102	3.8e+05	1.04e+04	36.642	0.000	3.6e+05	4e+05
zipcode_98103	3.016e+05	6245.044	48.300	0.000	2.89e+05	3.14e+05

zipcode_98105	3.574e+05	8196.789	43.604	0.000	3.41e+05	3.73e+05
zipcode_98106	1.107e+05	6867.105	16.122	0.000	9.73e+04	1.24e+05
zipcode_98107	3.022e+05	7323.679	41.270	0.000	2.88e+05	3.17e+05
zipcode_98108	1.031e+05	7986.665	12.911	0.000	8.75e+04	1.19e+05
zipcode_98109	3.833e+05	1.05e+04	36.629	0.000	3.63e+05	4.04e+05
zipcode_98112	4.002e+05	8353.355	47.911	0.000	3.84e+05	4.17e+05
zipcode_98115	3.011e+05	6267.866	48.043	0.000	2.89e+05	3.13e+05
zipcode_98116	2.764e+05	7031.484	39.314	0.000	2.63e+05	2.9e+05
zipcode_98117	2.968e+05	6314.541	46.999	0.000	2.84e+05	3.09e+05
zipcode_98118	1.479e+05	6354.479	23.277	0.000	1.35e+05	1.6e+05
zipcode_98119	3.731e+05	8589.406	43.441	0.000	3.56e+05	3.9e+05
zipcode_98122	2.838e+05	7306.551	38.839	0.000	2.69e+05	2.98e+05
zipcode_98125	1.836e+05	6622.349	27.717	0.000	1.71e+05	1.97e+05
zipcode_98126	1.797e+05	6826.126	26.329	0.000	1.66e+05	1.93e+05
zipcode_98133	1.481e+05	6341.026	23.356	0.000	1.36e+05	1.61e+05
zipcode_98136	2.403e+05	7392.075	32.503	0.000	2.26e+05	2.55e+05
zipcode_98144	2.252e+05	6954.358	32.386	0.000	2.12e+05	2.39e+05
zipcode_98146	1.03e+05	7207.392	14.294	0.000	8.89e+04	1.17e+05
zipcode_98148	5.091e+04	1.23e+04	4.145	0.000	2.68e+04	7.5e+04
zipcode_98155	1.341e+05	6593.809	20.341	0.000	1.21e+05	1.47e+05
zipcode_98166	9.7e+04	8086.112	11.996	0.000	8.11e+04	1.13e+05
zipcode_98168	4.224e+04	7522.949	5.614	0.000	2.75e+04	5.7e+04
zipcode_98177	1.966e+05	7834.797	25.090	0.000	1.81e+05	2.12e+05
zipcode_98178	4.322e+04	7335.313	5.892	0.000	2.88e+04	5.76e+04
zipcode_98188	2.83e+04	8991.184	3.148	0.002	1.07e+04	4.59e+04
zipcode_98198	2.589e+04	7360.906	3.517	0.000	1.15e+04	4.03e+04
zipcode_98199	3.388e+05	7247.354	46.748	0.000	3.25e+05	3.53e+05

Omnibus:	1421.662	**Durbin-Watson:**	1.998
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	4902.836
Skew:	0.376	**Prob(JB):**	0.00
Kurtosis:	5.463	**Cond. No.**	2.03e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.03e+05. This might indicate that there are strong multicollinearity or other numerical problems.



Out[57]: <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7fe

Recursive Feature Elimination

One final model used recursive feature elimination to identify which features to keep. However, results did not provide conclusive evidence for or against any of the features in model 3.

In [58]:
```
# recursive feature elimination
def rfe(df, target, number=4):
    '''
    Parameters:
    df - pd.DataFrame, main dataframe to read from
    target - str, name of target column
    number - int, number of features to select

    '''
    from sklearn.feature_selection import RFE
    from sklearn.linear_model import LinearRegression

    linreg = LinearRegression()
    selector = RFE(linreg, n_features_to_select=number)
    selector = selector.fit(df.drop(columns=[target]), df[target])
    features = dict(zip(df.drop(columns=[target]).columns, selector.sup
    return dict(sorted(features.items(), key=lambda x: x[1], reverse=Tr
```

In [59]:
```
rfe(m3_df_ohe, target, number=45)
```

Out[59]:
```
{'grade': True,
 'waterfront_1_0': True,
 'zipcode_98004': True,
 'zipcode_98005': True,
 'zipcode_98006': True,
 'zipcode_98007': True,
 'zipcode_98008': True,
 'zipcode_98011': True,
 'zipcode_98024': True,
 'zipcode_98027': True,
 'zipcode_98028': True,
 'zipcode_98029': True,
 'zipcode_98033': True,
 'zipcode_98034': True,
 'zipcode_98039': True,
 'zipcode_98040': True,
 'zipcode_98052': True,
 'zipcode_98053': True,
 'zipcode_98056': True,
 'zipcode_98059': True,
 'zipcode_98065': True,
 'zipcode_98072': True,
 'zipcode_98074': True,
 'zipcode_98075': True,
 'zipcode_98077': True,
 'zipcode_98102': True,
 'zipcode_98103': True,
 'zipcode_98105': True,
 'zipcode_98107': True,
 'zipcode_98109': True,
 'zipcode_98112': True,
 'zipcode_98115': True,
 'zipcode_98116': True,
 'zipcode_98117': True,
 'zipcode_98118': True,
 'zipcode_98119': True,
 'zipcode_98122': True,
 'zipcode_98125': True,
 'zipcode_98126': True,
 'zipcode_98133': True,
 'zipcode_98136': True,
 'zipcode_98144': True,
 'zipcode_98155': True,
 'zipcode_98177': True,
 'zipcode_98199': True,
 'bedrooms': False,
 'bathrooms': False,
 'sqft_living': False,
 'sqft_lot': False,
 'floors': False,
 'view': False,
 'neighboring_sqft_living': False,
 'age_at_sale': False,
 'yrs_since_renovated': False,
 'condition_1': False,
```

```
'condition_2': False,
'zipcode_98002': False,
'zipcode_98003': False,
'zipcode_98010': False,
'zipcode_98014': False,
'zipcode_98019': False,
'zipcode_98022': False,
'zipcode_98023': False,
'zipcode_98030': False,
'zipcode_98031': False,
'zipcode_98032': False,
'zipcode_98038': False,
'zipcode_98042': False,
'zipcode_98045': False,
'zipcode_98055': False,
'zipcode_98058': False,
'zipcode_98070': False,
'zipcode_98092': False,
'zipcode_98106': False,
'zipcode_98108': False,
'zipcode_98146': False,
'zipcode_98148': False,
'zipcode_98166': False,
'zipcode_98168': False,
'zipcode_98178': False,
'zipcode_98188': False,
'zipcode_98198': False}
```

Model Interpretation

Based on R2, p-values, and distribution of residuals, Model 4 was the most reliable and predictive linear model of house sale price. The following features together explain 83.6% of the variation in house sale price:

- Ratio of bathrooms to bedrooms
- Living space (sqft)
- Lot size (sqft)
- View
- Grade
- Condition
- Size of neighboring houses
- Age of house
- Years since renovation
- Waterfront
- Zipcode

As one may logically conclude, many of these factors have a positive relationship with price, meaning that as the variable increases, sale price also increases. Also logically, there is a negative relationship between the number of years since the renovation occurred and the sale price, meaning that more recent renovations tend to yield higher sale prices. The zipcode

where the house is located influences sale price, but not consistently. Some zipcodes have positive relationships with price while others have negative relationships, and the amount of influence varies.

Recommendations for King County Homeowners

Several factors that we found to influence house price are probably not in the homeowner's control, while others can be addressed by a renovation. Based on the model, three key recommendations for King County homeowners to consider when renovating their house are:

- **Add living space** - consider finishing an unfinished basement if you have one, or adding on to your house.
- **Add a full or half bath** - bring up your bathroom-to-bedroom ratio
- **Use high-quality materials** - raise your home's grade rating by bringing in marble or quartz countertops, new cabinets, crown molding, wood flooring, luxury finishes, etc.

Appendix

Model 6 - Excluding some zipcodes

I noticed some zipcodes had poor p-values, so I wanted to see if they were throwing off the residuals of model 3. There was no significant change in the model.

In [60]:
```
bad_zips = [98002, 98003, 98022, 98023, 98031, 98032, 98042, 98092, 98(
```

In [61]:
```
m6_df = m3_df[m3_df.zipcode != bad_zips[0]]
for i in range(len(bad_zips)):
    m6_df = m6_df[m6_df.zipcode != bad_zips[i]]
display(m6_df.zipcode.value_counts())
```

```
98103    587
98115    557
98117    543
98052    509
98118    499
        ...
98077     32
98014     25
98024     22
98070     16
98039      6
Name: zipcode, Length: 61, dtype: int64
```

In [62]:
```
# Identify categorical columns
m6_columns_cat = ['waterfront', 'condition', 'zipcode']
m6_drop_cols = ['id', 'renovated?', 'year_sold', 'sqft_basement',
                'lat', 'long']

# one-hot encode categorical columns
m6_df_ohe = onehotencode(df=m6_df, columns_cat=m6_columns_cat, drop_col
m6_df_ohe.head()

# model
model6 = model(df=m6_df_ohe, target=target)
model6
```

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.820
Model:	OLS	**Adj. R-squared:**	0.819
Method:	Least Squares	**F-statistic:**	957.8
Date:	Mon, 19 Apr 2021	**Prob (F-statistic):**	0.00
Time:	16:40:28	**Log-Likelihood:**	-1.9705e+05
No. Observations:	15424	**AIC:**	3.942e+05
Df Residuals:	15350	**BIC:**	3.948e+05
Df Model:	73		
Covariance Type:	nonrobust		

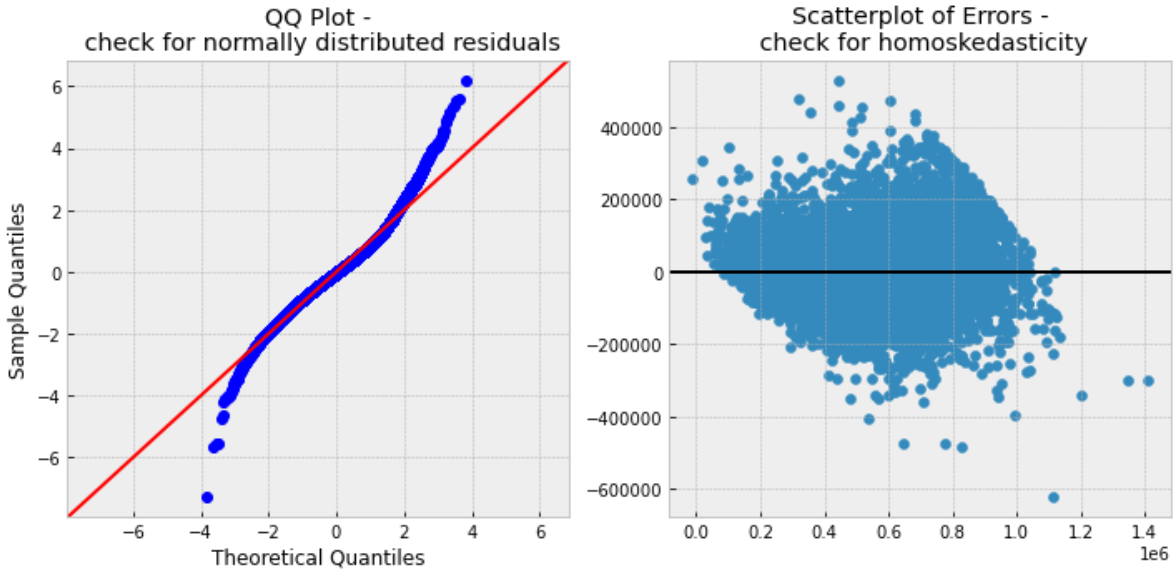
	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-4.278e+05	1.35e+04	-31.608	0.000	-4.54e+05	-4.01e+05
bedrooms	-5071.8249	1041.571	-4.869	0.000	-7113.428	-3030.222
bathrooms	9146.2592	1684.957	5.428	0.000	5843.545	1.24e+04
sqft_living	102.4402	2.023	50.646	0.000	98.476	106.405
sqft_lot	1.7408	0.306	5.687	0.000	1.141	2.341
floors	2525.3954	1903.130	1.327	0.185	-1204.964	6255.755
view	2.951e+04	1246.493	23.678	0.000	2.71e+04	3.2e+04
grade	5.098e+04	1220.293	41.778	0.000	4.86e+04	5.34e+04
neighboring_sqft_living	41.5529	2.047	20.301	0.000	37.541	45.565
age_at_sale	770.1356	41.136	18.722	0.000	689.505	850.766
yrs_since_renovated	-27.5983	4.241	-6.507	0.000	-35.911	-19.285
waterfront_1_0	2.579e+05	2.1e+04	12.261	0.000	2.17e+05	2.99e+05
condition_1	3.895e+04	7457.958	5.222	0.000	2.43e+04	5.36e+04
condition_2	7.918e+04	7807.276	10.142	0.000	6.39e+04	9.45e+04
zipcode_98004	5.013e+05	9155.382	54.759	0.000	4.83e+05	5.19e+05
zipcode_98005	3.244e+05	9957.492	32.583	0.000	3.05e+05	3.44e+05

zipcode_98006	2.584e+05	6972.112	37.066	0.000	2.45e+05	2.72e+05
zipcode_98007	2.46e+05	9109.388	27.008	0.000	2.28e+05	2.64e+05
zipcode_98008	2.36e+05	7518.763	31.387	0.000	2.21e+05	2.51e+05
zipcode_98010	8.192e+04	1.45e+04	5.643	0.000	5.35e+04	1.1e+05
zipcode_98011	1.343e+05	8218.844	16.341	0.000	1.18e+05	1.5e+05
zipcode_98014	1.214e+05	1.79e+04	6.770	0.000	8.62e+04	1.57e+05
zipcode_98019	9.035e+04	9022.221	10.014	0.000	7.27e+04	1.08e+05
zipcode_98024	1.506e+05	1.9e+04	7.922	0.000	1.13e+05	1.88e+05
zipcode_98027	2.192e+05	7558.271	29.004	0.000	2.04e+05	2.34e+05
zipcode_98028	1.221e+05	7525.643	16.224	0.000	1.07e+05	1.37e+05
zipcode_98029	2.262e+05	7267.325	31.119	0.000	2.12e+05	2.4e+05
zipcode_98033	3.141e+05	6950.604	45.194	0.000	3e+05	3.28e+05
zipcode_98034	1.82e+05	6440.662	28.264	0.000	1.69e+05	1.95e+05
zipcode_98038	3.977e+04	6506.041	6.112	0.000	2.7e+04	5.25e+04
zipcode_98039	6.368e+05	3.54e+04	17.992	0.000	5.67e+05	7.06e+05
zipcode_98040	4.077e+05	9062.625	44.992	0.000	3.9e+05	4.26e+05
zipcode_98045	1.008e+05	9273.946	10.865	0.000	8.26e+04	1.19e+05
zipcode_98052	2.483e+05	6456.012	38.460	0.000	2.36e+05	2.61e+05
zipcode_98053	2.642e+05	7720.343	34.222	0.000	2.49e+05	2.79e+05
zipcode_98055	3.914e+04	7528.386	5.198	0.000	2.44e+04	5.39e+04
zipcode_98056	1.036e+05	6795.237	15.248	0.000	9.03e+04	1.17e+05
zipcode_98058	3.347e+04	6819.372	4.908	0.000	2.01e+04	4.68e+04
zipcode_98059	1.031e+05	6868.194	15.011	0.000	8.96e+04	1.17e+05
zipcode_98065	1.26e+05	7415.138	16.997	0.000	1.11e+05	1.41e+05
zipcode_98070	6.296e+04	2.22e+04	2.832	0.005	1.94e+04	1.07e+05
zipcode_98072	1.484e+05	9195.422	16.135	0.000	1.3e+05	1.66e+05
zipcode_98074	2.065e+05	6976.840	29.602	0.000	1.93e+05	2.2e+05
zipcode_98075	2.157e+05	8053.096	26.789	0.000	2e+05	2.32e+05
zipcode_98077	1.593e+05	1.6e+04	9.931	0.000	1.28e+05	1.91e+05
zipcode_98102	3.855e+05	1.08e+04	35.628	0.000	3.64e+05	4.07e+05
zipcode_98103	3.081e+05	6665.134	46.221	0.000	2.95e+05	3.21e+05
zipcode_98105	3.603e+05	8583.300	41.974	0.000	3.43e+05	3.77e+05
zipcode_98106	1.206e+05	7105.230	16.967	0.000	1.07e+05	1.34e+05
zipcode_98107	3.101e+05	7712.470	40.206	0.000	2.95e+05	3.25e+05
zipcode_98108	1.107e+05	8266.386	13.388	0.000	9.45e+04	1.27e+05
zipcode_98109	3.903e+05	1.09e+04	35.806	0.000	3.69e+05	4.12e+05
zipcode_98112	4.037e+05	8764.769	46.062	0.000	3.87e+05	4.21e+05

zipcode_98115	3.063e+05	6562.320	46.673	0.000	2.93e+05	3.19e+05
zipcode_98116	2.821e+05	7352.227	38.369	0.000	2.68e+05	2.97e+05
zipcode_98117	3.045e+05	6648.908	45.797	0.000	2.91e+05	3.18e+05
zipcode_98118	1.566e+05	6624.074	23.640	0.000	1.44e+05	1.7e+05
zipcode_98119	3.796e+05	9014.622	42.112	0.000	3.62e+05	3.97e+05
zipcode_98122	2.911e+05	7713.109	37.734	0.000	2.76e+05	3.06e+05
zipcode_98125	1.876e+05	6810.409	27.542	0.000	1.74e+05	2.01e+05
zipcode_98126	1.886e+05	7110.068	26.523	0.000	1.75e+05	2.03e+05
zipcode_98133	1.505e+05	6527.327	23.062	0.000	1.38e+05	1.63e+05
zipcode_98136	2.473e+05	7675.328	32.222	0.000	2.32e+05	2.62e+05
zipcode_98144	2.31e+05	7321.048	31.551	0.000	2.17e+05	2.45e+05
zipcode_98146	1.103e+05	7396.663	14.914	0.000	9.58e+04	1.25e+05
zipcode_98148	5.827e+04	1.26e+04	4.631	0.000	3.36e+04	8.29e+04
zipcode_98155	1.347e+05	6755.866	19.940	0.000	1.21e+05	1.48e+05
zipcode_98166	9.55e+04	8284.403	11.528	0.000	7.93e+04	1.12e+05
zipcode_98168	4.958e+04	7710.820	6.429	0.000	3.45e+04	6.47e+04
zipcode_98177	1.961e+05	8042.427	24.385	0.000	1.8e+05	2.12e+05
zipcode_98178	4.098e+04	7552.682	5.426	0.000	2.62e+04	5.58e+04
zipcode_98188	2.891e+04	9207.690	3.140	0.002	1.09e+04	4.7e+04
zipcode_98198	2.664e+04	7547.394	3.530	0.000	1.18e+04	4.14e+04
zipcode_98199	3.412e+05	7558.010	45.149	0.000	3.26e+05	3.56e+05

Omnibus:	1083.585	**Durbin-Watson:**	2.006
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	3388.328
Skew:	0.347	**Prob(JB):**	0.00
Kurtosis:	5.189	**Cond. No.**	4.94e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.94e+05. This might indicate that there are strong multicollinearity or other numerical problems.



Out[62]: <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x7fe

Folium

In [63]:
```
import folium

def create_map(data, geojson, zips_col, mapped_feature):
    """
    data = pd.DataFrame, main dataset to read from
    geojson = dict formatted geojson to source the shapes for zip codes
    zips_col = string, column name where zipcodes are
    mapped_feature = string, column name for feature to visualize
    """

    m = folium.Map(location = [47.47, -121.84], zoom_start=9)
    folium.Choropleth(geo_data=geojson, data=data, key_on='feature.prop
                columns=[zips_col, mapped_feature], fill_color='BuPu',
                legend_name=('Average Home ' + mapped_feature.title() +
                     ).add_to(m)
    folium.LayerControl().add_to(m)
    m.add_child(
        folium.features.Tooltip(zipcodes_df['zipcode']))
    return m

# create_map(zipcodes_df, kc_geojson, 'zipcode', 'price')
```

In [64]:
```
# Attempting to get markers and popups on the map; can't get it to wor

# code source: https://www.kaggle.com/saidakbarp/nyc-property-sales-eda

# from folium.plugins import MarkerCluster
# marker_cluster = MarkerCluster().add_to(kc_zipmap)
# for i in zipcodes_df.iterrows():
#     folium.Marker(location=[zipcodes_df['lat'][i], zipcodes_df['long
#                   popup=zipcodes_df['zipcode'][i],
#                   tooltip="""Mean home price: <b>${}</b> <br>
#                            Mean sq footage (living): <b>{}</b> <br>
#                            Mean year built: <b>{}</b>""".format(zipc
#                                                                 zipc
#                                                                 zipc

#     from folium.plugins import MarkerCluster
#     marker_cluster = MarkerCluster()
#     for i in range(len(data)):
#         location = [data['lat'][i],data['long'][i]]
#         tooltip = "Zipcode:{}<br> Click for more".format(data[zips][
#         folium.Marker(location,
#                       popup="""<i>Mean home price: </i> <br> <b>${}<
#                       <i>Mean square footage (living): </i><b><br>{}
#                       <i>Mean year built: </i><b><br>{}</b><br>""".fo
#
#
#                       tooltip=tooltip).add_to(marker_cluster)
#     marker_cluster.add_to(m)
```