casanave / **Seattle_Section_8_Expansion_Project** Public

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

main

Seattle_Section_8_Expansion_Project / **Seattle_Section_8_Project.ipynb**

casanave uploaded jupyter notebook pdf History

1 contributor

3.35 MB

Seattle Section 8 Housing Project

Introduction

Premise

Public/low income housing has become a somewhat controversial issue in Seattle. While many can agree there are two Seattles, or about the existence of a "homelessness issue," ideas about how to solve for this problem range from tax breaks for real estate investors, increased police budgets, to "defensive design" (IE spikes on buildings to stop people from sleeping there) to increasing unemployment budgets and back-to-work programs. In the city's political history, candidates have called the issue of class disparity and homelessness either larger or smaller, depending on their goals and intended audience. In 2015, when this dataset is from, 45 homeless folks died as a direct result of their homelessness, and the mayor declared a state of emergency. (CITATION: https://www.capitolhillseattle.com/2015/11/45-dead-in-2015-mayor-declares-seattle-homelessness-in-a-state-of-emergency/)

Hypothetical Premise: The City of Seattle city government, in partnership with the State of Washington's state senate has a new pilot program called Section 8 Expansion. There is a 100 million dollar budget in this pilot program to buy real estate at market value to convert into section 8 housing. This project has 3 main goals: 1) to increase the availability of low-income housing 2) to increase property values throughout the city 3) to decrease class disparity.

Instead of making parts of the city into the projects, this program aims to inter-splice public housing all over the city. This evidence based practice results in better outcomes for both the tenants and the properties. Socially, it reduces stigma and segregation as well.

Each property bought for this project will undergo a renovation to ensure quality of living, and there is a separate budget set aside specifically for renovating. For this project, I'm not responsible for balancing the cost of renovating these properties.

I am instead tasked with using machine learning to build a model that can be used to recommend where to build more public housing, under the caveat that they will be renovated. I am looking for properties that will have increased property values in the future. Certain considerations will need to be taken into account for this recommendation, such as the spacing these properties.

While other cities such as Dallas, Texas and Gary Indiana have bought properties back from the public with success, this

While other cities such as Dallas, Texas and Gary Indiana have bought properties back from the public with success, this would be on a much larger scale. (CITATION: https://www.fastcompany.com/90618596/the-radical-way-cities-are-tackling-affordable-housing)

(CITATION FOR OVERVIEW ON SECTION 8 HOUSING: https://www.hud.gov/program_offices/housing/mfh/rfp/s8bkinfo)

FURTHER READING: On mixed income housing in NYC, where there is lots of data: https://case.edu/socialwork/nimc/sites/case.edu.nimc/files/2020-05/Schwartz%20NewYork%202020.pdf

Why should low-income housing be interspersed throughout the city? (CITATION)

"Proponents of mixed-income housing see it as a tool to address the difficulties related to what has been termed the *culture of poverty*. This phrase derives from the view that physical concentration of poor households in multifamily projects causes severe problems for the residents, including joblessness, drug abuse, and welfare dependency. According to Brophy and Smith 6 Cityscape this theory, a mixture of income levels will reduce the social pathology caused by concentration. Anticipated results of mixed-income housing include the following:

■ The behavior patterns of some lower income residents will be altered by emulating those of their higher income neighbors. The quality of the living environment, not housing quality alone, leads to upward mobility.

■ Nonworking low-income tenants will find their way into the workplace in greater numbers because of the social norms of their new environment (for example, going to work/school every day) and the informal networking with employed neighbors.

■ The crime rate will fall because the higher income households will demand a stricter and better enforced set of ground rules for the community.

■ Low-income households will have the benefit of better schools, access to jobs, and enhanced safety, enabling them to move themselves and their children beyond their current economic condition.

Many of these anticipated results are subtle and difficult to measure. Unlike such a quantifiable result as the effect of mixed incomes on the project’s financial condition, analysis of the effects of mixed-income housing on the behavior of residents must take into account the subtleties of human behavior. It is also important to differentiate between two reasons for these intended benefits. First, if low-income tenants are subsidized to live in developments that are in locations with good schools, low crime rates, and access to jobs, there is some likelihood that the benefits of the location will result in the anticipated outcomes described above."

https://www.huduser.gov/periodicals/cityscpe/vol3num2/success.pdf

The Stakeholder's Interests:

The city of Seattle has tasked me with ensuring their making good, safe investments with an assured outcome. Since this project is using public funds, it will be scrupulously documented (and discussed in media.) The model, furthermore, needs to be understandable to civic workers.

The city is also especially interested in seeing the range of housing in my analysis. Often in statistics, we focus on the median or mean, however for this project, I'm going to be collecting data on the size of the ranges.

Why Use Machine Learning?

It is coloquial knowledge that there is class disparity in Seattle. However, how large the gap is between neighborhoods specifically, home by home, property by property is what I've been tasked to research. Seattle has decided that the best approach possible is one backed by research, science and evidence.

Using machine learning, we can rely on the data itself to ensure cost-effective and evidence-backed solutions to this problem.

Machine Learning is being implemented here to reduce the political biasing of the solution. However, ML can only be as free of bias as the implementation is thoughtful.

Why I Was Highered to Oversee the Project

In order to counter the mechanisms were ML can propogate racism, classism, etc, (CITATION: https://www.nytimes.com/2021/03/15/technology/artificial-intelligence-google-bias.html) Seattle chose me specifically as a Data Scientist for my professional experience working with homeless folks, at a housing-forward non-profit.

Loading and Inspecting the Data

In [1]:

```
# importing the libraries to use, the basics, the libraries for modeling,
# geopy for distance data collection and selenium for scraping

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

```
import numpy as np

import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
import scipy.stats as stats
from sklearn.preprocessing import StandardScaler

## geopy must be installed first
from geopy import distance

## for scraping distances
from selenium import webdriver
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys

import requests, json

import folium

%matplotlib inline

palette = sns.color_palette("colorblind")
sns.set_style('white')
```

In [2]:

```
# loading the main CSV

data = pd.read_csv('Data/kc_house_data.csv')
```

In [3]:

```
# inspecting the data

data.head()
```

Out[3]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_
0	7129300520	10/13/2014	221900.0	3	1.00	1180	5650	1.0	NaN	NONE	...	7 Average	
1	6414100192	12/9/2014	538000.0	3	2.25	2570	7242	2.0	NO	NONE	...	7 Average	
2	5631500400	2/25/2015	180000.0	2	1.00	770	10000	1.0	NO	NONE	...	6 Low Average	

3	2487200875	12/9/2014	604000.0	4	3.00	1960	5000	1.0	NO	NONE	...	7 Average
4	1954400510	2/18/2015	510000.0	3	2.00	1680	8080	1.0	NO	NONE	...	8 Good

5 rows × 21 columns

In [4]:

```
data.info()

# taking a look at the raw data
```

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

Of note: waterfront, and year renovated have significant data missing, where view is missing only a bit of data and those null values can probably be filled with whatever makes the most sense given the context.

In [5]:

```
data.isna().sum()
```

Out[5]:
```
id                  0
date                0
price               0
bedrooms            0
bathrooms           0
sqft_living         0
sqft_lot            0
floors              0
waterfront       2376
view               63
condition           0
grade               0
sqft_above          0
sqft_basement       0
yr_built            0
yr_renovated     3842
zipcode             0
lat                 0
long                0
sqft_living15       0
sqft_lot15          0
dtype: int64
```

In [6]:
```
data.describe()
```

Out[6]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	sqft_above
count	2.159700e+04	2.159700e+04	21597.000000	21597.000000	21597.000000	2.159700e+04	21597.000000	21597.000000
mean	4.580474e+09	5.402966e+05	3.373200	2.115826	2080.321850	1.509941e+04	1.494096	1788.596842
std	2.876736e+09	3.673681e+05	0.926299	0.768984	918.106125	4.141264e+04	0.539683	827.759761
min	1.000102e+06	7.800000e+04	1.000000	0.500000	370.000000	5.200000e+02	1.000000	370.000000
25%	2.123049e+09	3.220000e+05	3.000000	1.750000	1430.000000	5.040000e+03	1.000000	1190.000000
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.500000	1560.000000
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068500e+04	2.000000	2210.000000
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	9410.000000

Of note--the average house has 3 bedrooms, 2 bathrooms, which is the kind of housing we are already looking to make public. The average house was built in 1970, making the average house on the market 52 years old. The average renovation happened 80 years ago, which is older than the standard property is old, which means the information on renovations must be explored in greater detail.

In [7]:

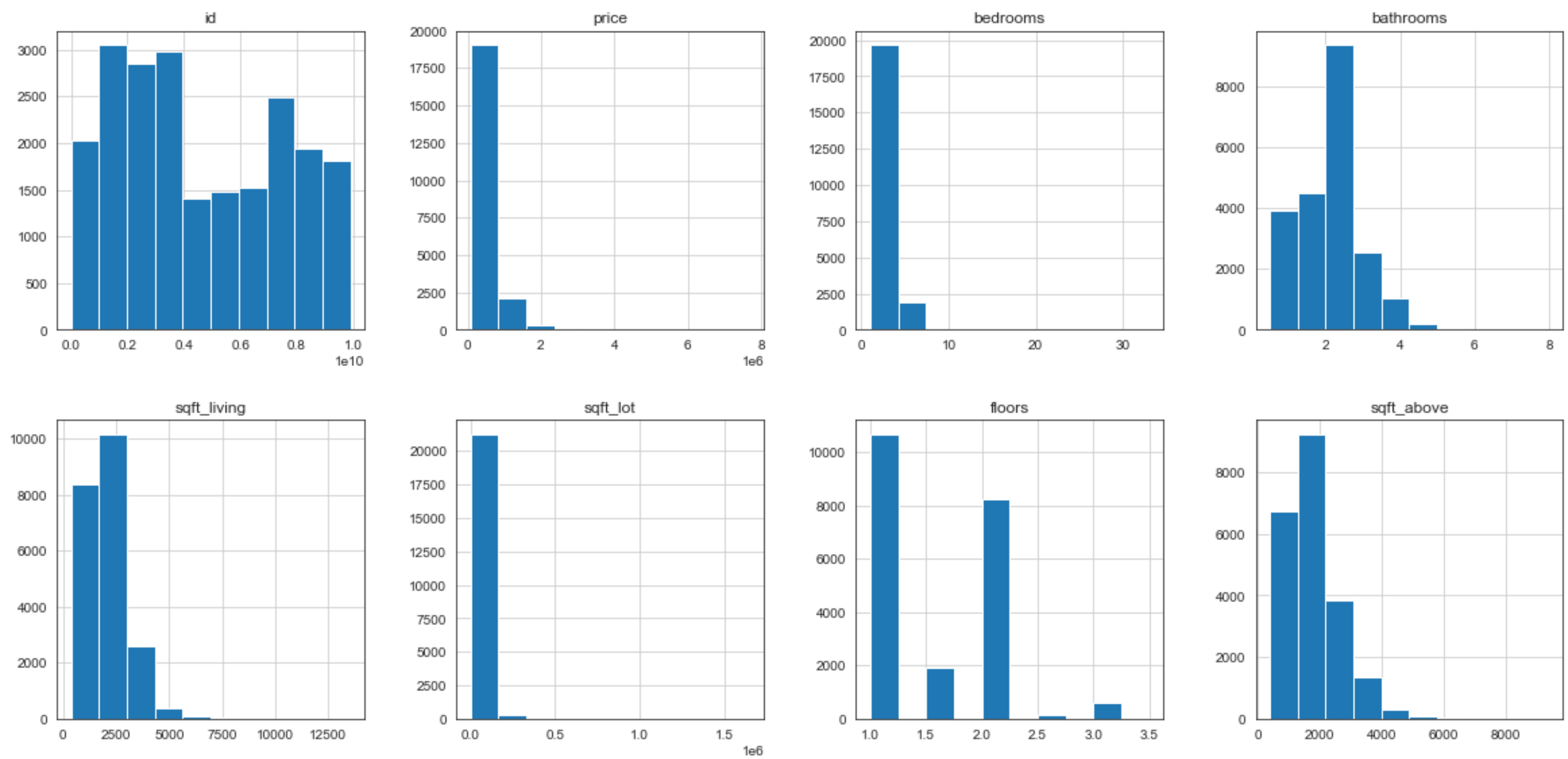
```
data.columns

# looking to see what columns are in the dataset
```

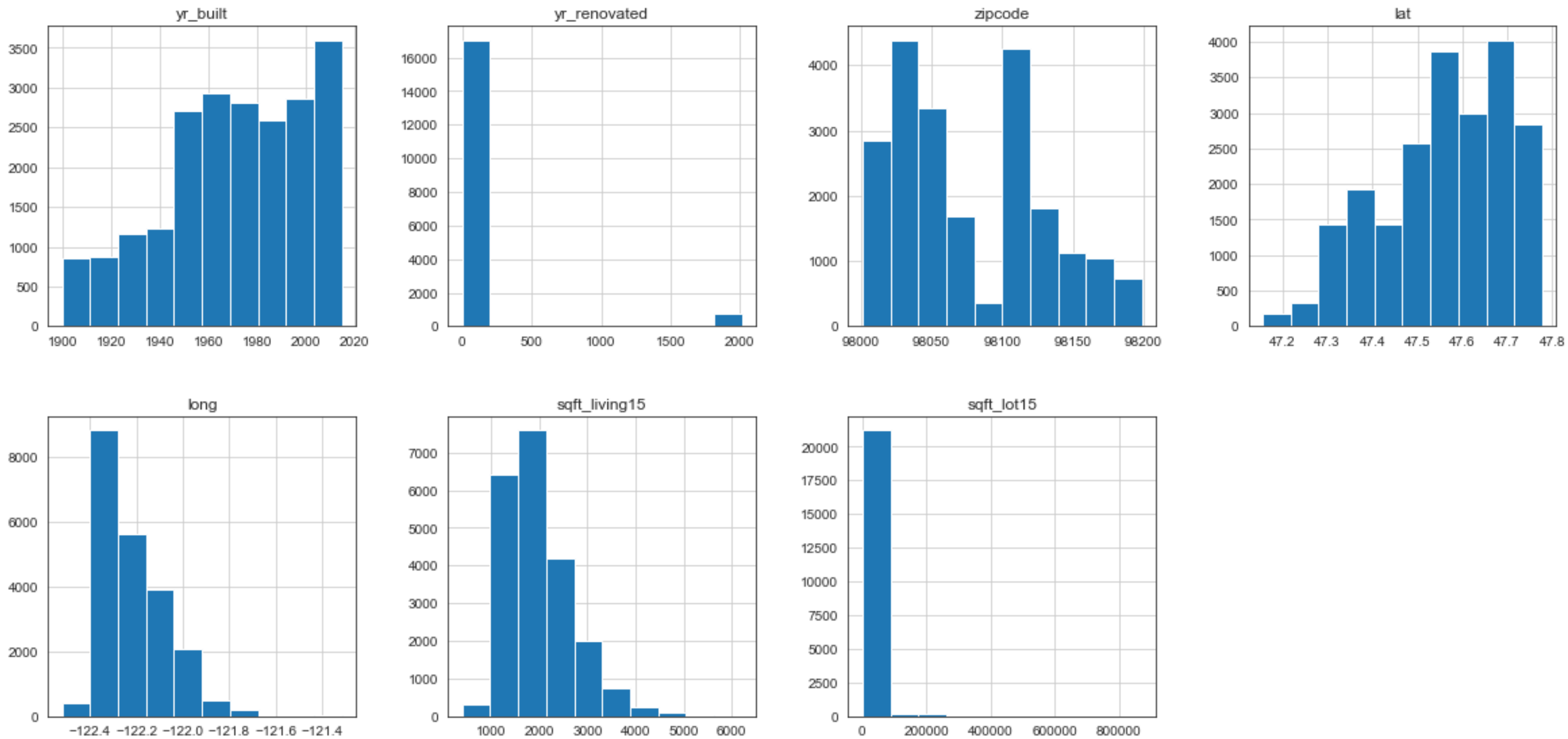
Out[7]:

```
Index(['id', 'date', 'price', 'bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade',
       'sqft_above', 'sqft_basement', 'yr_built', 'yr_renovated', 'zipcode',
       'lat', 'long', 'sqft_living15', 'sqft_lot15'],
      dtype='object')
```

In [8]:

```
sns.set_style('white')

data.hist(figsize=(20,20));

# let's look at the distributions of some of these columns
```





Cleaning and Preparing the Data

Duplicates

In [9]:

```
data['id'].value_counts().head()
# let's take a look to see if there are duplicates
```

Out[9]:

```
795000620     3
1825069031    2
2019200220    2
7129304540    2
1781500435    2
Name: id, dtype: int64
```

Some of these properties are listed three times, that will throw off everything unless we remove the duplicates.

In [10]:

```
data.drop_duplicates(subset='id', inplace=True)
```

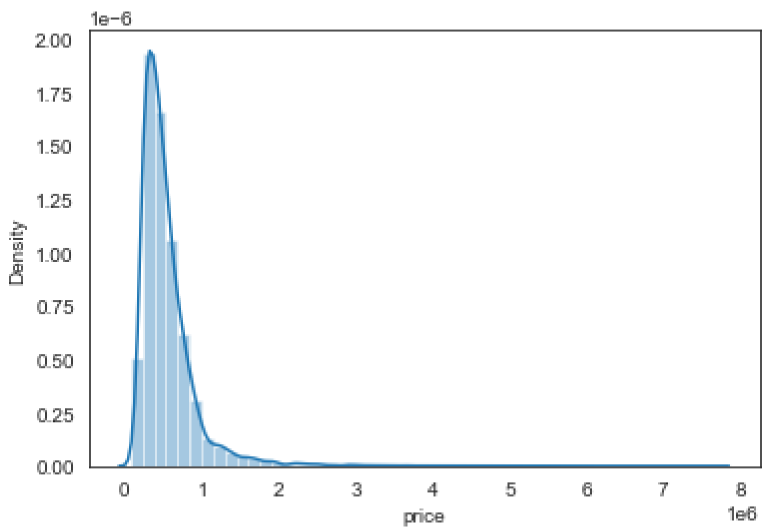
```
# this will get rid of duplicates
```

Price

In [11]:

```
sns.distplot( a=data["price"]);

# this will show a histogram for the price distribution
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



Observations: The distribution of price in this dataset has a very large tail in the more expensive houses, and the data is leptokurtic (taller and skinnier) than a normal distribution.

Since this project is specifically about buying a lot of properties, I'm going to focus only on properties that are priced at 2.5 million and below, which is where a majority of the data is. This will prevent all the calculations from being dragged into more expensive price brackets for the city, and for the tenants of the new housing.

In [12]:

```
data = data.loc[data['price'] < 2500000.0]

# this will get rid of all properties in the data set above 2.5 million
```

```
# this will get rid of all properties in the data set above 2.5 million
```

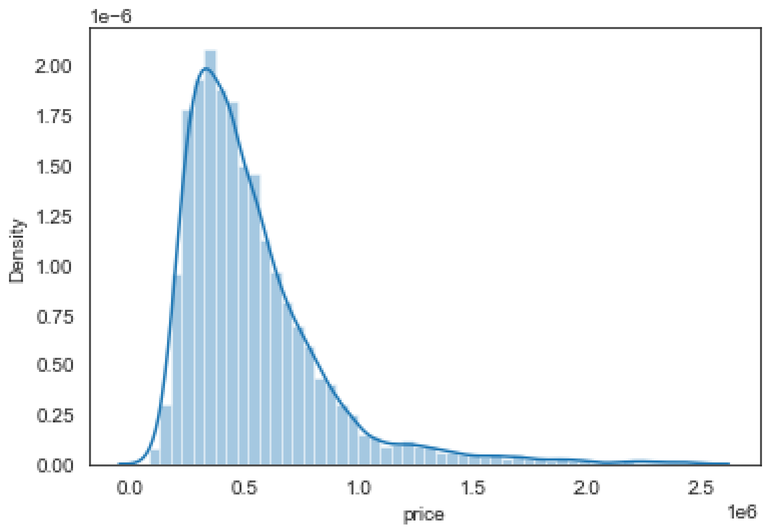
In [13]:

```
sns.distplot( a=data["price"])

# this will show what the distribution of prices looks like in the data
# that remains
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[13]:

```
<AxesSubplot:xlabel='price', ylabel='Density'>
```



Sqft_living

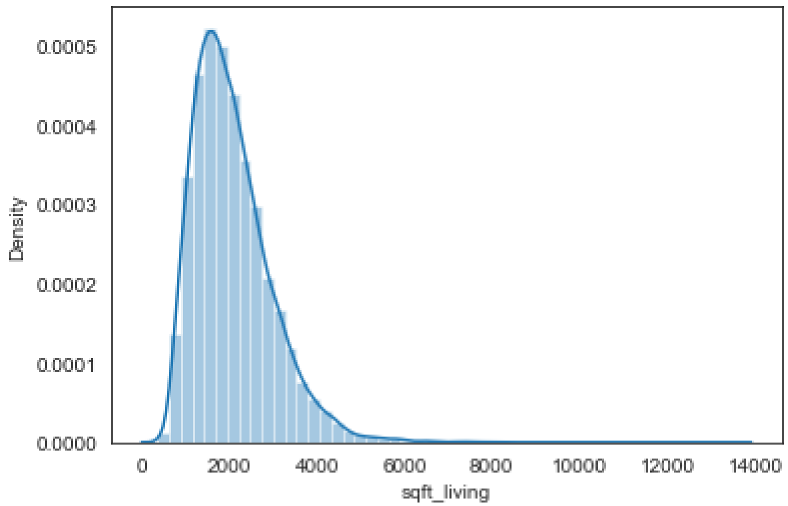
Square footage of living space in the home-from the data.

In [14]:

```
sns.distplot( a=data["sqft_living"]);

# Taking a look at the distribution of this variable
```

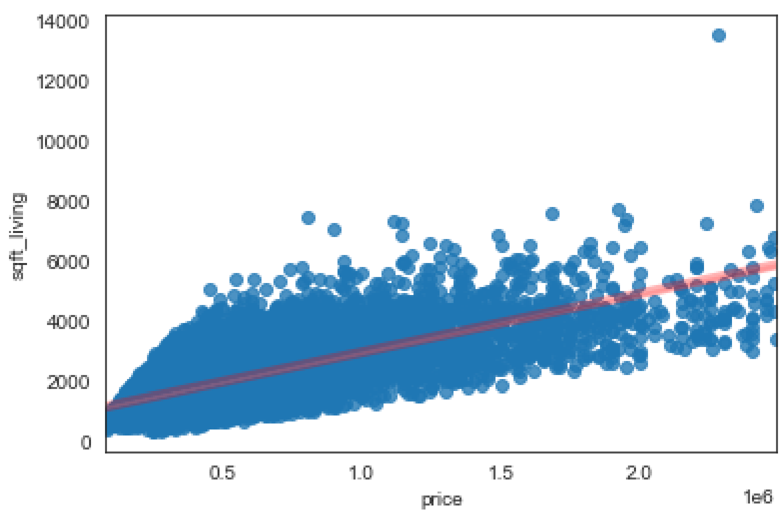
```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
```

```
  warnings.warn(msg, FutureWarning)
```



In [15]:

```
sns.regplot(data = data, x = 'price', y = 'sqft_living',
            line_kws={"color":"r","alpha":0.3,"lw":5})
plt.show()

# Visualizing how sqft_living affects price
```

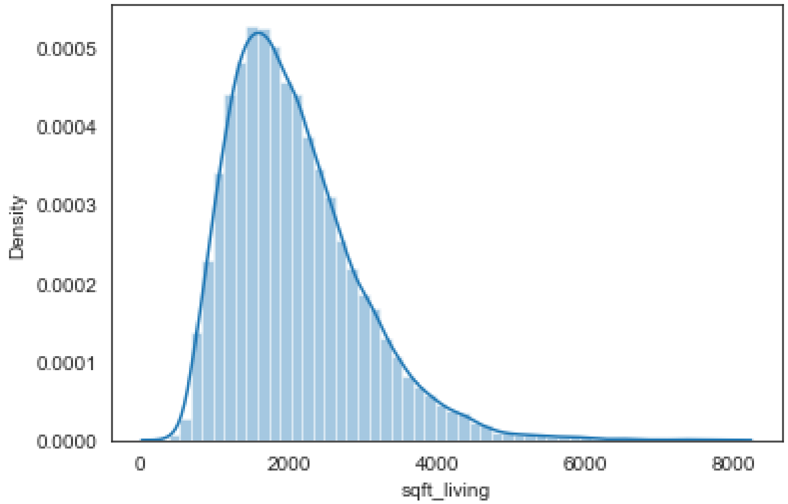


I'm going to get rid of the outliers in the data set, the homes we're looking to make into public housing will not be these outliers.

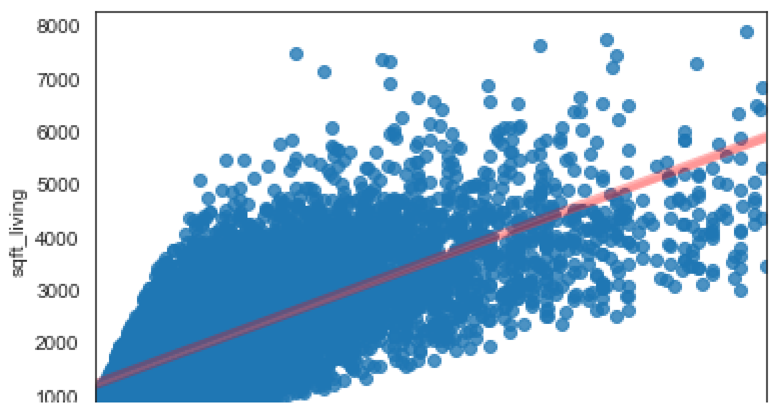
In [16]:
```
data = data.loc[data['sqft_living'] < 8000]
```

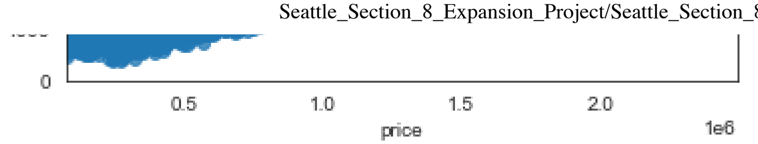
In [17]:
```
sns.distplot( a=data["sqft_living"]);
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [18]:
```
sns.regplot(data = data, x = 'price', y = 'sqft_living',
            line_kws={"color":"r","alpha":0.3,"lw":5})
plt.show()
```





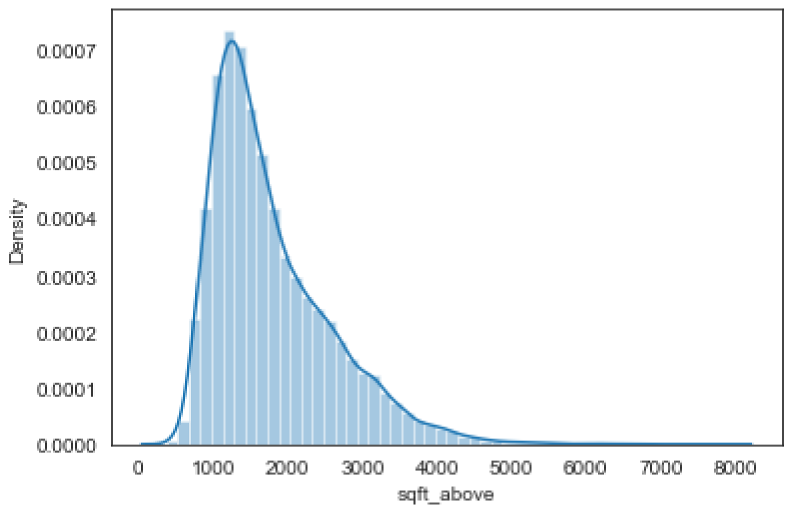
Sqft_above

Square footage of house apart from basement-from the data.

In [19]:

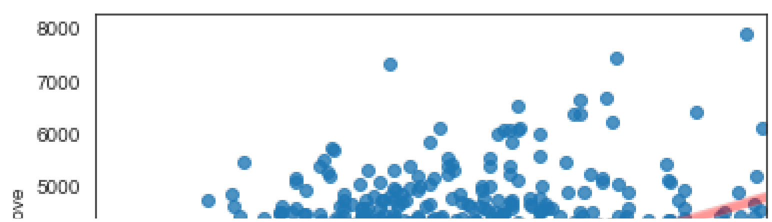
```
sns.distplot( a=data["sqft_above"]);
```

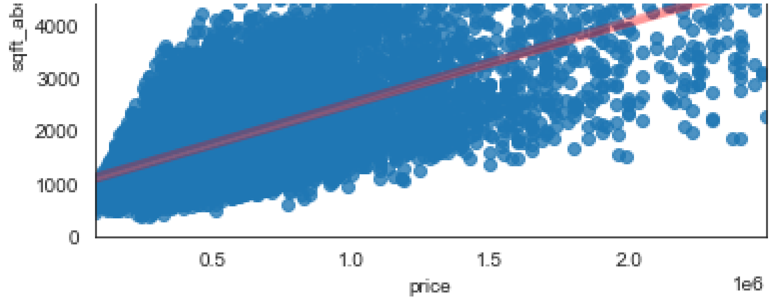
```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/distributions.py:2551:
FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt
your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an ax
es-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [20]:

```
sns.regplot(data = data, x = 'price', y = 'sqft_above',
            line_kws={"color":"r","alpha":0.3,"lw":5})
plt.show()
```





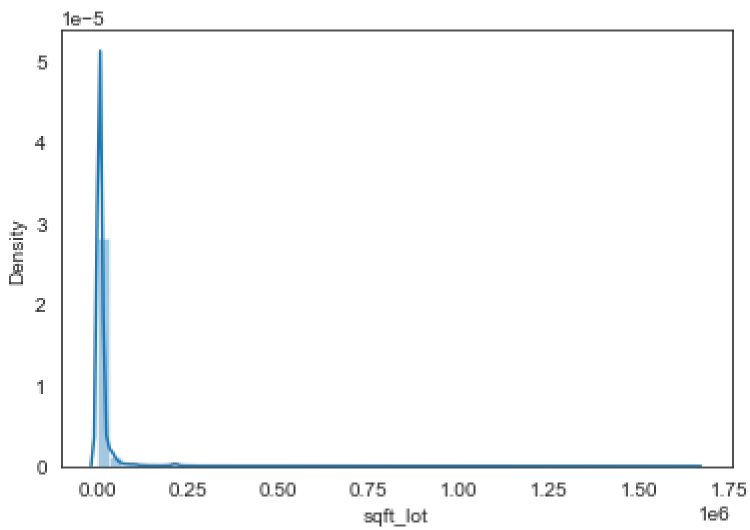
Sqft_lot

Square footage of the lot-from the data.

In [21]:

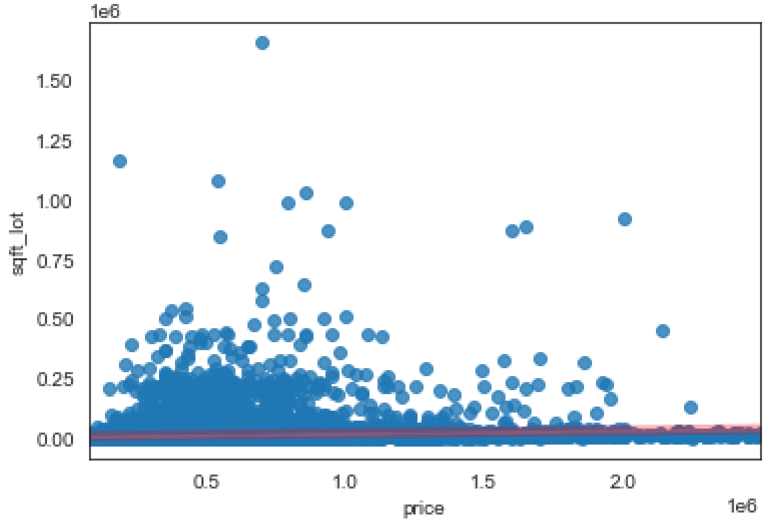
```
sns.distplot( a=data["sqft_lot"]);
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/distributions.py:2551:
FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt
your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an ax
es-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [22]:

```
sns.regplot(data = data, x = 'price', y = 'sqft_lot',
            line_kws={"color":"r","alpha":0.3,"lw":5})
plt.show()
```



In [23]:

```
data['sqft_lot'].describe()
```

Out[23]:

```
count    2.131700e+04
mean     1.508035e+04
std      4.156528e+04
min      5.200000e+02
25%      5.030000e+03
50%      7.590000e+03
75%      1.060000e+04
max      1.651359e+06
Name: sqft_lot, dtype: float64
```

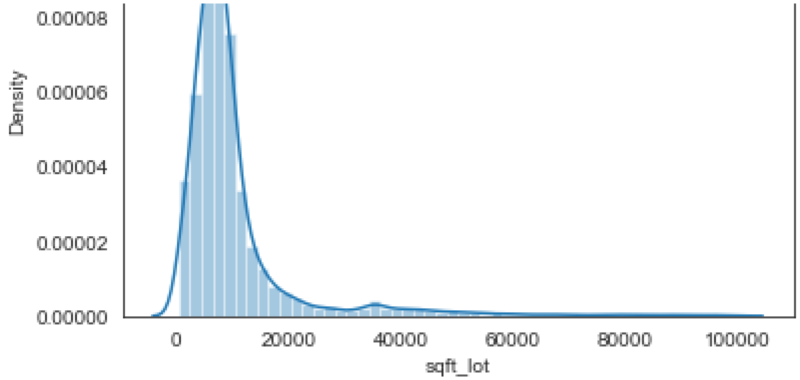
In [24]:

```
data = data.loc[data['sqft_lot'] < 100000]
```

In [25]:

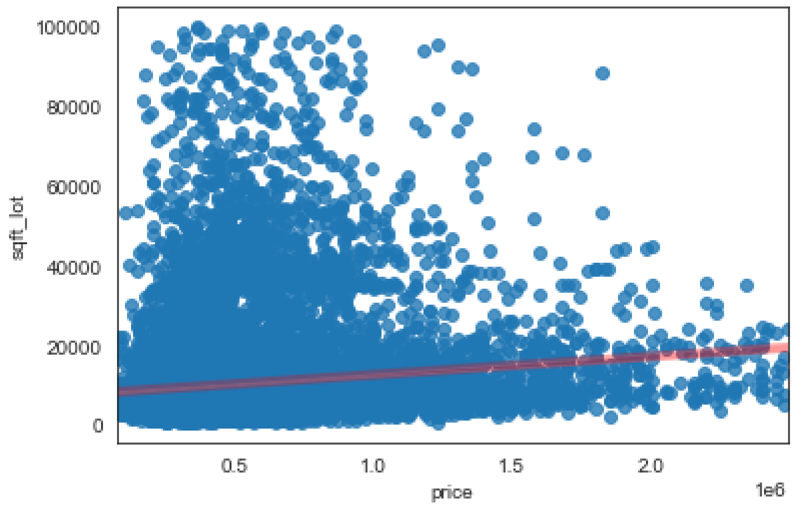
```
sns.distplot( a=data["sqft_lot"]);
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/distributions.py:2551:
FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt
your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an ax
es-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [26]:

```
sns.regplot(data = data, x = 'price', y = 'sqft_lot',
            line_kws={"color":"r","alpha":0.3,"lw":5})
plt.show()
```



Sqft_basement

In [27]:

```
data['sqft_basement'].value_counts().head()
```

Out[27]:

```
0.0       12374
?           437
600.0       214
700.0       204
500.0       204
Name: sqft_basement, dtype: int64
```

```
Name: sqft_basement, dtype: int64
```

In [28]:
```
data['sqft_basement'].describe()
```

Out[28]:
```
count     20855
unique      288
top         0.0
freq      12374
Name: sqft_basement, dtype: object
```

In [29]:
```
data.loc[data['sqft_basement'] == '?','sqft_basement'] = 0.0

#? doesn't make sense as a value. I will replace it with the top value instead.
```

In [30]:
```
data.info()
```

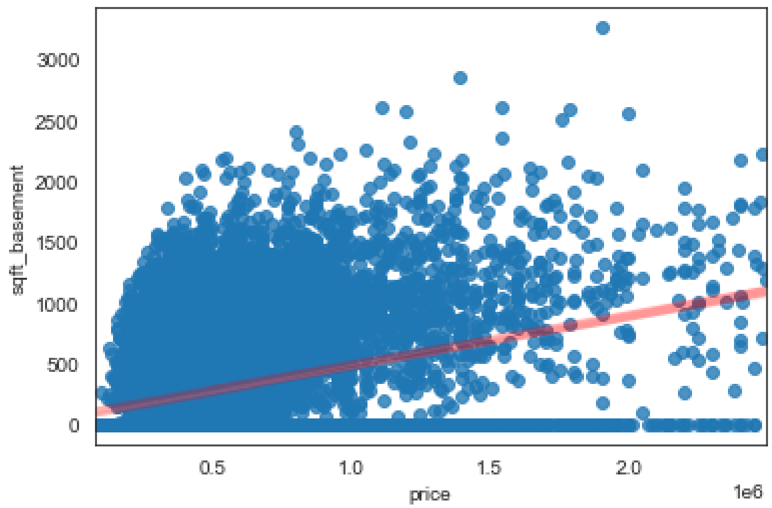
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20855 entries, 0 to 21596
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             20855 non-null  int64  
 1   date           20855 non-null  object 
 2   price          20855 non-null  float64
 3   bedrooms       20855 non-null  int64  
 4   bathrooms      20855 non-null  float64
 5   sqft_living    20855 non-null  int64  
 6   sqft_lot       20855 non-null  int64  
 7   floors         20855 non-null  float64
 8   waterfront     18559 non-null  object 
 9   view           20795 non-null  object 
 10  condition      20855 non-null  object 
 11  grade          20855 non-null  object 
 12  sqft_above     20855 non-null  int64  
 13  sqft_basement  20855 non-null  object 
 14  yr_built       20855 non-null  int64  
 15  yr_renovated   17129 non-null  float64
 16  zipcode        20855 non-null  int64  
 17  lat            20855 non-null  float64
 18  long           20855 non-null  float64
 19  sqft_living15  20855 non-null  int64  
 20  sqft_lot15     20855 non-null  int64  
dtypes: float64(6), int64(9), object(6)
memory usage: 3.5+ MB
```

In [31]:

```
data['sqft_basement'] = data['sqft_basement'].astype(float)
```

In [32]:

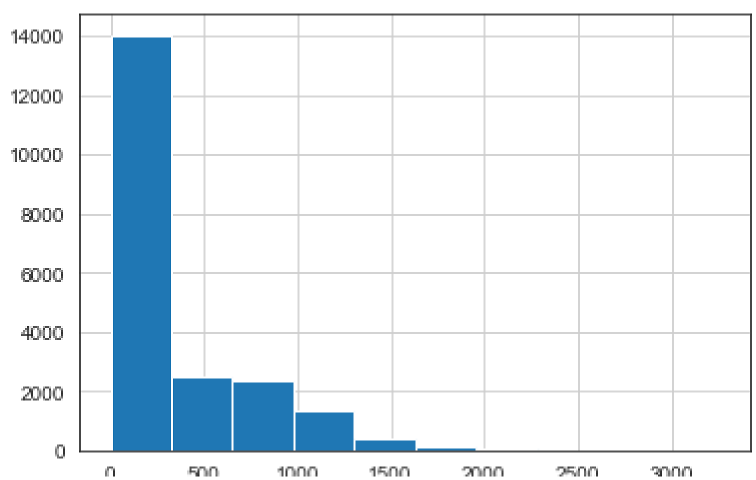
```
sns.regplot(data = data, x = 'price', y = 'sqft_basement',
            line_kws={"color":"r","alpha":0.3,"lw":5})
plt.show()
```



In [33]:

```
data['sqft_basement'].hist()
```

Out[33]: <AxesSubplot:>



I'm going to turn this into a binary variable: 0 if the property has no basement (sqft_basement at 0,) and 1 if the property has a basement. There are too many high priced homes without a basement, and over 50% of homes don't have one.

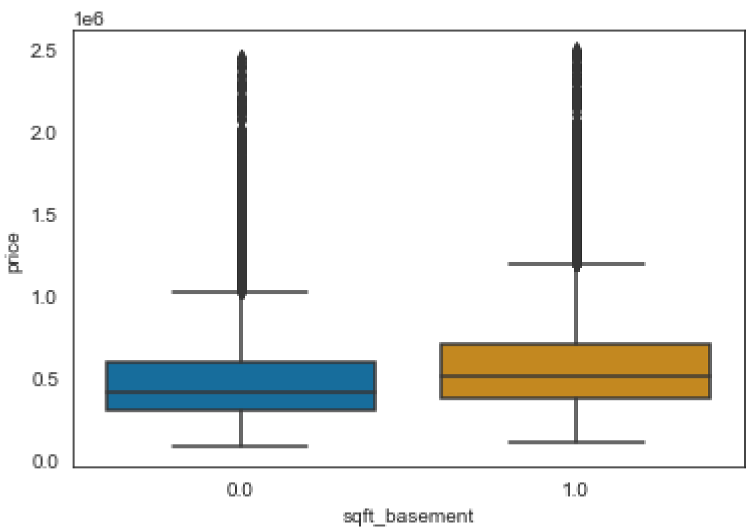
In [34]:
```
data.loc[data['sqft_basement'] != 0,'sqft_basement'] = 1
```

In [35]:
```
data['sqft_basement'].value_counts().head()
```

Out[35]:
```
0.0    12811
1.0     8044
Name: sqft_basement, dtype: int64
```

In [36]:
```
sns.boxplot(data = data, x="sqft_basement", y="price", palette = palette)
```

Out[36]:
```
<AxesSubplot:xlabel='sqft_basement', ylabel='price'>
```



Observations: the minimum quartile price, average price and maximum quartile price are all higher in properties with basements, as to be expected.

Square Foot Living 15 and Square Lot 15

It is noteworthy to have the square footage of the living space and the lot of the closest 15 neighbors. These following two metrics are especially interesting for our purposes. Traditional public housing, in big blocky complexes, would have much lower metrics in the following two variables than in mixed income public housing models.

FURTHER WORK:

None of the properties in this data set are public housing properties. In order to properly compare how the relationships to the closest 15 neighboring properties compares for both public and non-public housing, I'd need to collect more data.

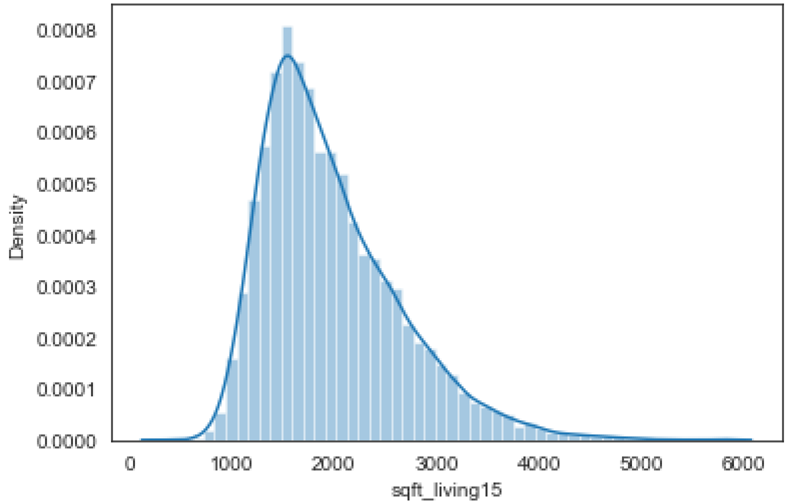
Sqft_living15

The square footage of interior housing living space for the nearest 15 neighbors- from the data.

In [37]:

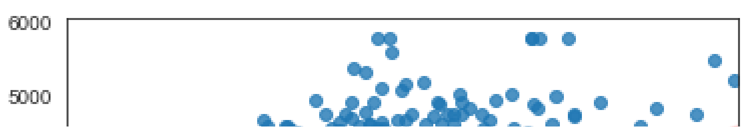
```
sns.distplot( a=data["sqft_living15"]);
```

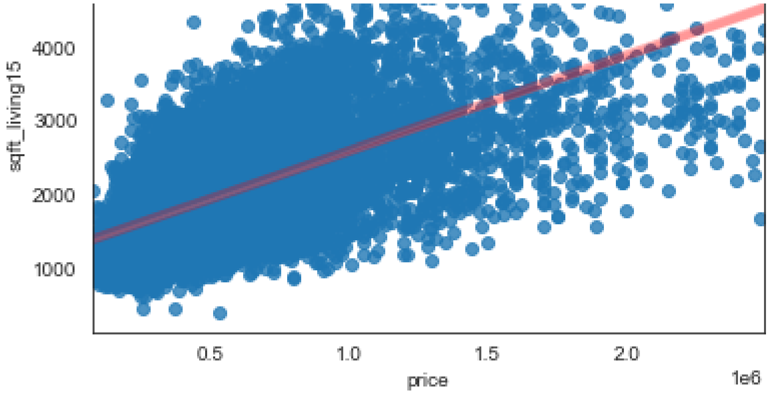
```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/distributions.py:2551:
FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt
your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an ax
es-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [38]:

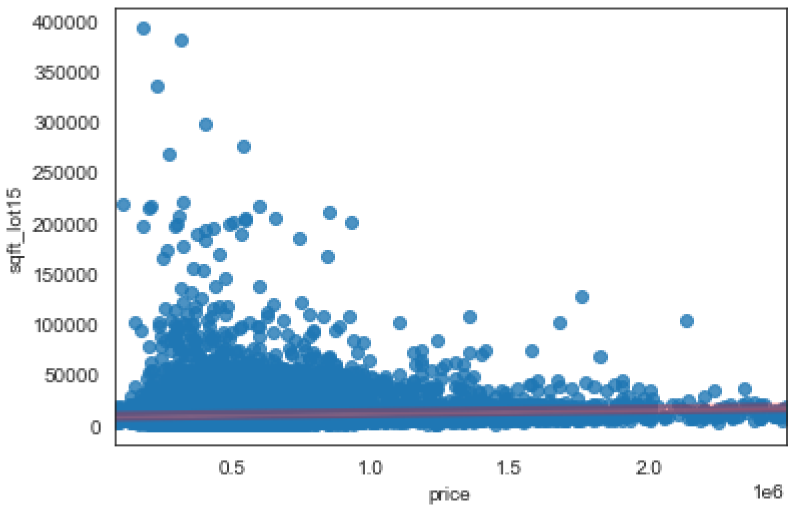
```
sns.regplot(data = data, x = 'price', y = 'sqft_living15',
            line_kws={"color":"r","alpha":0.3,"lw":5})
plt.show()
```





Sqft_lot15

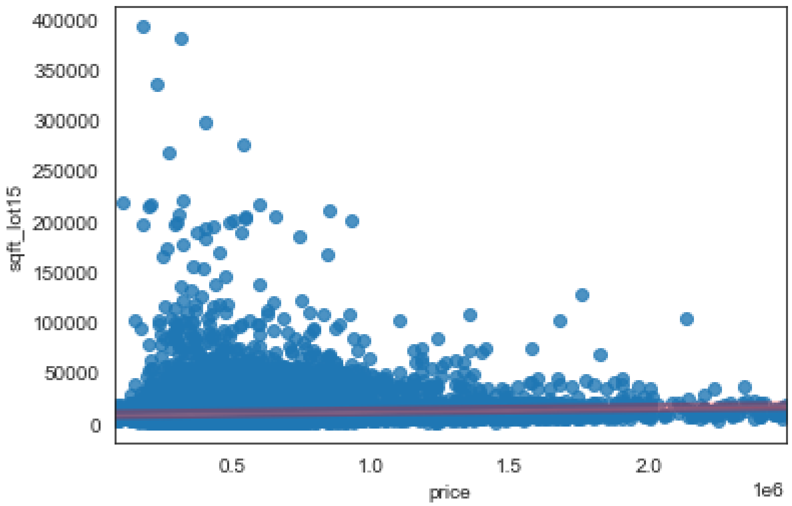
In [39]:
```
sns.regplot(data = data, x = 'price', y = 'sqft_lot15',
            line_kws={"color":"r","alpha":0.3,"lw":5})
plt.show()
```



I'm going to get rid of the upper valued outliers here, which would represent homes where the 15 closest neighbors have a *lot* of square footage. These are mansion neighborhoods and will throw off the data. I want to be conservative however with my cut off, since the goal of this project is, in part, to increase the overall prosperity of the neighbors of properties selected for public housing.

In [40]:
```
data = data.loc[data['sqft_lot15'] < 400000]
```

In [41]:
```
sns.regplot(data = data, x = 'price', y = 'sqft_lot15',
            line_kws={"color":"r","alpha":0.3,"lw":5})
plt.show()
```



Waterfront

In [42]:
```
data['waterfront'].value_counts()
```

Out[42]:
```
NO     18447
YES      112
Name: waterfront, dtype: int64
```

In [43]:
```
data['waterfront'].isna().sum()
```

Out[43]:
```
2296
```

In [44]:
```
data['waterfront'].fillna(value = "NO", inplace = True)
```

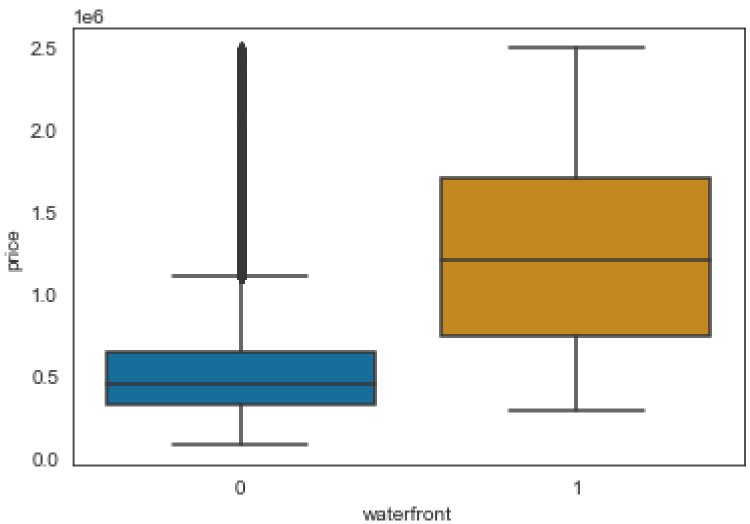
In [45]:
```
data['waterfront'].replace(['NO', 'YES'],
                        [0, 1], inplace=True)
#this will make the value numeric and therefore able to be readily fed to
#the model
```

Because the average value is "no", I'm choosing to fill null values with "no." Because a waterfront view would be a boon to the property, the risk involved of me falsely representing a property with a waterfront to have no waterfront, and that property being selected for public housing has a surprise positive effect for the folks who would live there. By contrast, if I filled these values with "yes" and they didn't have a waterfront, this would falsely inflate the assigned value of that property, and if those properties were selected because of that false inflated value, that would actively be unfair to the folks who lived there.

In [46]:

```
sns.boxplot(data = data, x = 'waterfront', y = 'price', palette = palette)
```

Out[46]: <AxesSubplot:xlabel='waterfront', ylabel='price'>



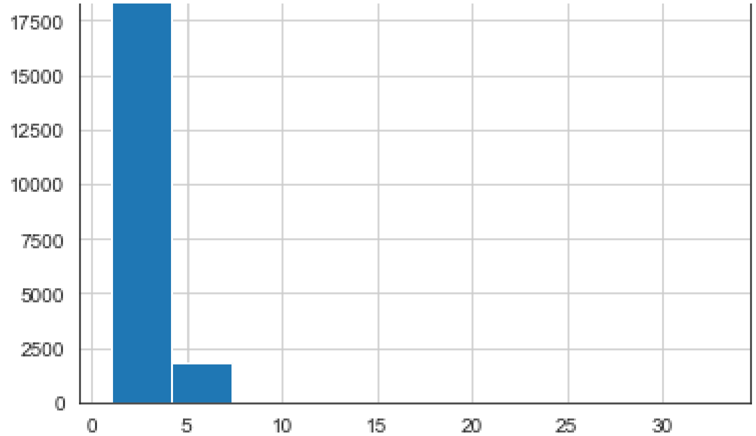
This is a clear visualization that represents: 1) homes with a waterfront have a much wider spread for their average price, especially in the upper limits of how pricey those homes can be. But in general, waterfront homes are more expensive. 2) there are plenty outlying examples of homes with no waterfront that are more expensive.

Bedrooms

In [47]:

```
data['bedrooms'].hist()
# taking a look at the distribution of bedrooms in this data set
```

Out[47]: <AxesSubplot:>



In [48]:
```
data['bedrooms'].value_counts()
```

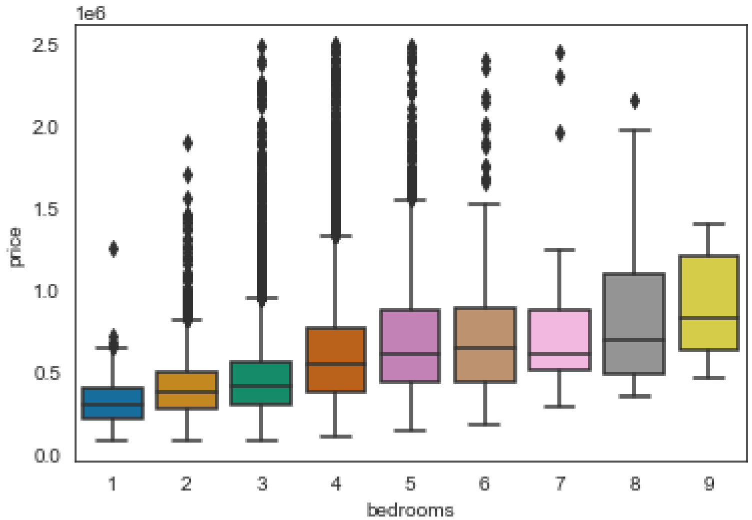
Out[48]:
```
3      9512
4      6649
2      2685
5      1517
6       248
1       186
7        35
8        12
9         6
10        3
11        1
33        1
Name: bedrooms, dtype: int64
```

Since properties where there are more than 9 bedrooms may be difficult to convert into public housing anyway, and because most of the data is outside these values, I'll be dropping properties that have more than 9 bedrooms.

In [49]:
```
data = data.loc[data['bedrooms'] <= 9]
```

In [50]:
```
sns.boxplot(data = data, x="bedrooms", y="price", palette = palette)
# we can see from the figure that as bedrooms increase, price consistently
# increases, and also we can get a better sense for the distribution of price
# outliers by bedroom.
```

Out[50]:
```
<AxesSubplot:xlabel='bedrooms', ylabel='price'>
```



This plot shows 1) that homes show a steady increase in minimum price as bedrooms increase 2) that there are very many outliers in homes that have 1-7 bedrooms in price but even with those outliers, there is also a steady increase in maximum price as bedrooms increase as well.

Transforming Categorical Data

Condition

CITATION: https://info.kingcounty.gov/assessor/esales/Glossary.aspx?type=r#c

Relative to age and grade. Coded 1-5.

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

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

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

expectancy. Effective age and utility is standard for like properties of its class and usage.

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

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

Based on the description as given by the data set, this is a good candidate to be made into continuous data. A house in "very good" condition could get a lower condition score in the future if the property doesn't receive proper maintenance. Likewise a property in "fair" condition could be renovated and made into a property with a higher condition.

Based on the description as given by the data set, this is a good candidate to be made into continuous data. A house in "very good" condition could get a lower condition score in the future if the property doesn't receive proper maintenance. Likewise a property in "fair" condition could be renovated and made into a property with a higher condition.

In [51]:
```
data['condition'].replace(['Poor', 'Fair', 'Average', 'Good', 'Very Good'],
                          [0, 1, 2, 3, 4], inplace=True)

# making these values continuous
```

In [52]:
```
data['condition'].value_counts()

# looking at the spread of the data
```

Out[52]:
```
2    13519
3     5502
4     1653
1      149
0       27
Name: condition, dtype: int64
```

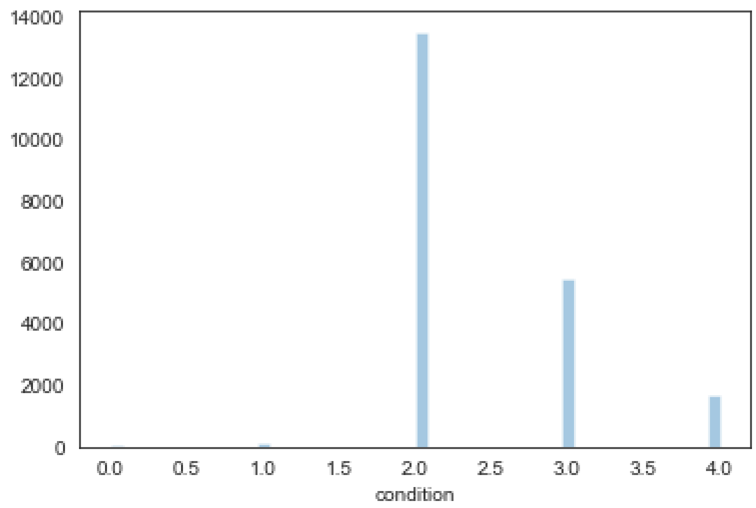
In [53]:
```
sns.distplot( a=data["condition"], hist=True, kde=False, rug=False )

# visualizing the spread of the data
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/distributions.py:2551:
FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt
your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an ax
```

```
es-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[53]:

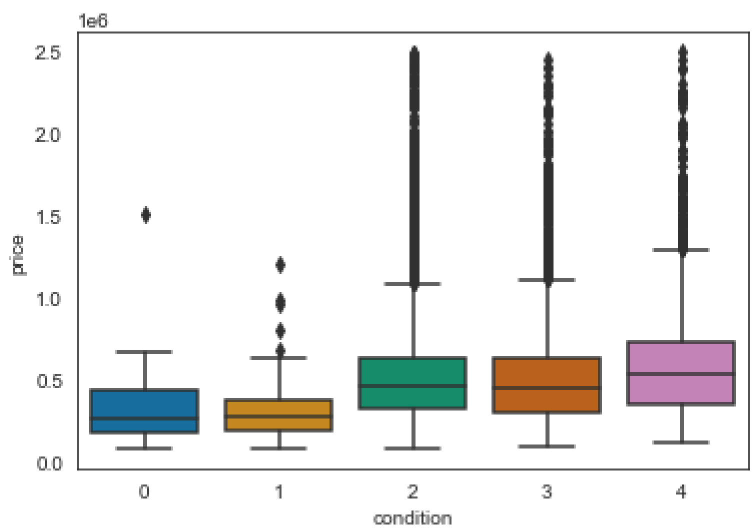
```
<AxesSubplot:xlabel='condition'>
```



Observations: there is relatively no representation of homes with a condition below "fair" in this dataset. A majority of homes are in fair condition, with less in "good" condition and even less in "very good" condition.

In [54]:

```
sns.boxplot(data = data, x = 'condition', y = 'price', palette = palette)
plt.show();
```



Observations: The price of homes in poor condition and fair condition on average have a lower price than homes in better condition. The minimum price for a home goes up as it's condition increases in value, as well as the upper limit price for a home.

Grade

CITATION: https://info.kingcounty.gov/assessor/esales/Glossary.aspx?type=r#c

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

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

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

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

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

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

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

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

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

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

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

In [55]:

```
data['grade'].value_counts()
```

Out[55]:

```
7 Average        8762
8 Good           5928
9 Better         2505
6 Low Average    1967
10 Very Good     1046
```

```
11 Excellent      328
5 Fair            227
12 Luxury          54
4 Low              27
13 Mansion          5
3 Poor              1
Name: grade, dtype: int64
```

In [56]:

```
data['grade'].replace(['3 Poor', '4 Low', '5 Fair', '6 Low Average',
                       '7 Average', '8 Good', '9 Better', '10 Very Good',
                       '11 Excellent', '12 Luxury', '13 Mansion'],
                        [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10], inplace=True)
```

In [57]:

```
data['grade'].value_counts()
```

Out[57]:

```
4     8762
5     5928
6     2505
3     1967
7     1046
8      328
2      227
9       54
1       27
10       5
0        1
Name: grade, dtype: int64
```

For our purposes, it doesn't make sense to keep any Luxury or Mansion properties in the data set, since those kinds of properties would need to be renovated just to make better use of the space, and would effectively be making these properties less valuable. It also doesn't make sense to keep the single property in the poor grade, we can assume that property will be on the list, and take it out of the data for now. So I'm going to drop those rows.

Likewise it doesn't make sense to keep homes in this data set that are below a 3, because these homes would need essential building done, such as installing electricity and water. While we are looking to do renovations on properties, we must draw a distinction between building and renovating. Therefore I'll be dropping properties at a 3 or below.

FURTHER WORK:

I would like to come back to this variable in future works and do further investigation. Homes currently at a grade of 6 or above meet building codes that are currently in place. It may be beneficial to come back and raise the minimum grade of homes in this data set to 6, to lower the cost of renovations needed to make these properties into public housing. All homes not currently up to code would need to be brought up to code in renovations, should they be selected for the Section 8 Expansion project.

In [58]:

```
poor = data.loc[data['grade'] == 0]

# labeling this single property for the city to buy, buldoze and maybe
# make into a small public park
```

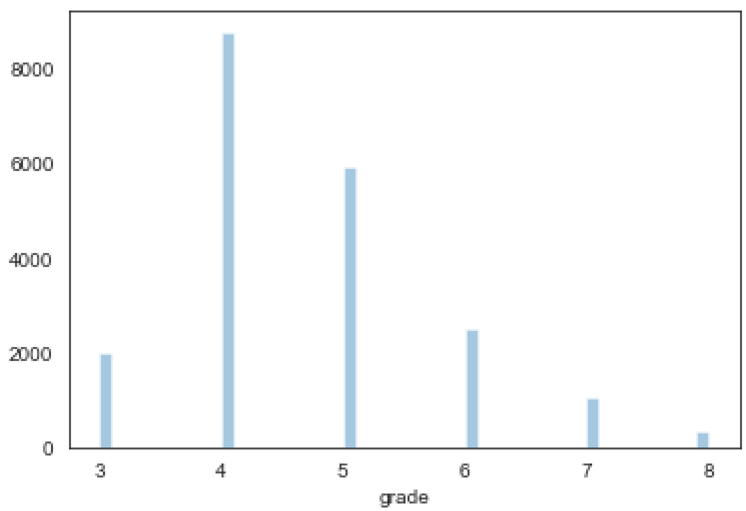
In [59]:

```
data = data.loc[data['grade'] < 9]
data = data.loc[data['grade'] >= 3]
```

In [60]:

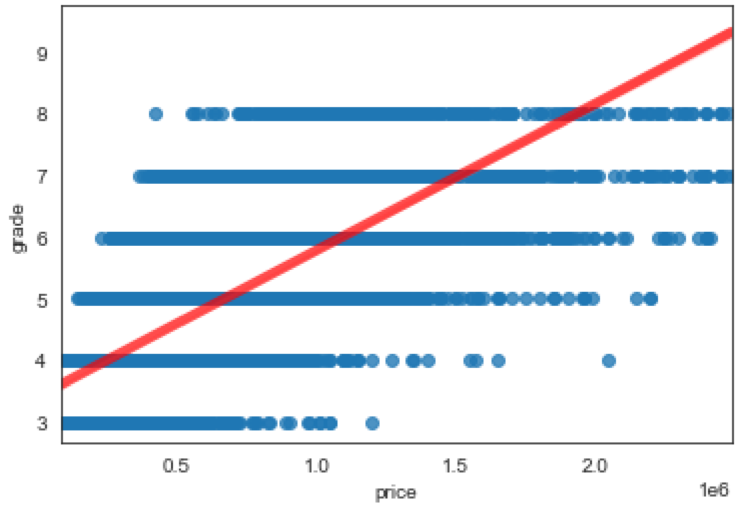
```
sns.distplot( a=data["grade"], hist=True, kde=False, rug=False);
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/distributions.py:2551:
FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt
your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an ax
es-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [61]:

```
sns.regplot(data = data, x = 'price', y = 'grade',
            line_kws={"color":"r","alpha":0.7,"lw":5})
plt.show()
```



Observations: As grade improves, so does price, this is what we'd expect to see.

View

FROM THE DATASET: Includes views of Mt. Rainier, Olympics, Cascades, Territorial, Seattle Skyline, Puget Sound, Lake Washington, Lake Sammamish, small lake / river / creek, and other. Represents quality of view from house.

Since most properties have "none" listed as the view, and this category is subjective to start with, I am going to convert it to a binary column. Homes with "good" or "excellent" views will have a value of 1, all other homes will have a value of 0 for this column.

In [62]:

```
data['view'].value_counts()
```

Out[62]:

```
NONE         18569
AVERAGE        883
GOOD           455
FAIR           316
EXCELLENT      253
Name: view, dtype: int64
```

In [63]:

```
data['view'].replace(['NONE', 'FAIR', 'AVERAGE', 'GOOD', 'EXCELLENT'],
                        [0, 0, 0, 1, 1], inplace=True)

# replacing the values to binary values
```

In [64]:

```
data['view'].fillna(value=0,inplace=True)

# filling in n/a values with 0
```

In [65]:

```
data['view'].value_counts()

# taking a look at the distribution now
```

Out[65]:

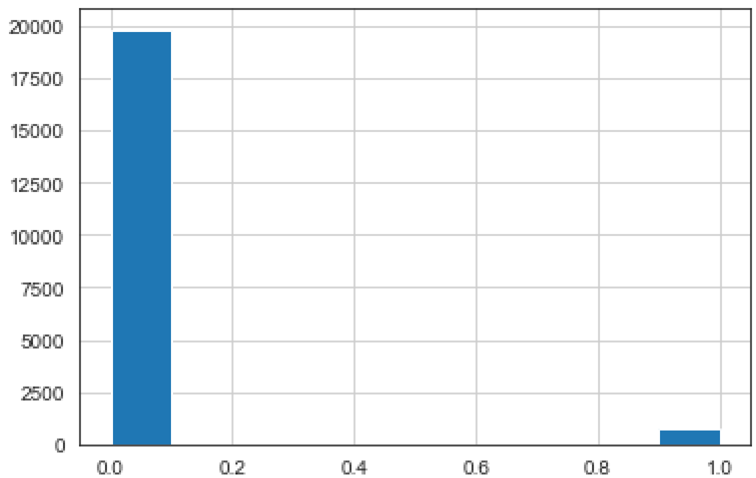
```
0.0    19828
1.0      708
Name: view, dtype: int64
```

In [66]:

```
data['view'].hist()

# it's easier to see visually
```

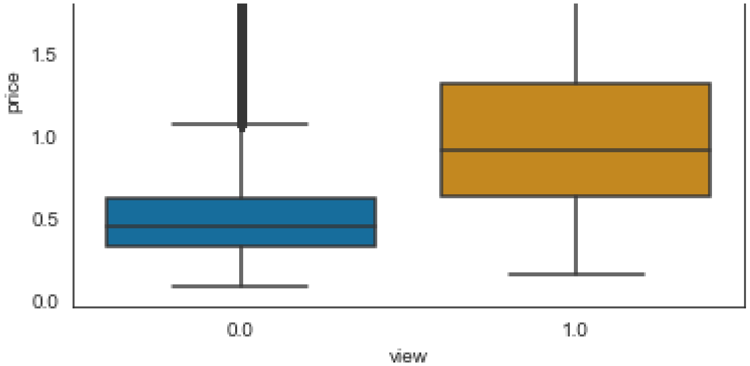
Out[66]:

```
<AxesSubplot:>
```



In [67]:

```
sns.boxplot(data = data, x = 'view', y = 'price', palette = palette)
plt.show();
```



Observations: We can see here that having a view has a clear increase in the value of the home, and there is a much larger spread in homes with a view as to their price. Likewise, we can see that homes without a view are generally less expensive than homes with a view but there are very many outlying cases where that isn't true.

In [68]:

```
data.isna().sum()

#making sure there are no remaining columns with missing values
```

Out[68]:

```
id                  0
date                0
price               0
bedrooms            0
bathrooms           0
sqft_living         0
sqft_lot            0
floors              0
waterfront          0
view                0
condition           0
grade               0
sqft_above          0
sqft_basement       0
yr_built            0
yr_renovated     3658
zipcode             0
lat                 0
long                0
sqft_living15       0
sqft_lot15          0
dtype: int64
```

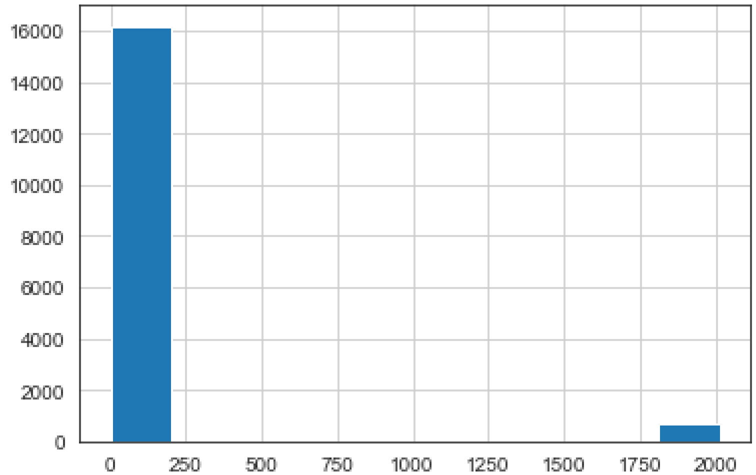
Year Renovated

Because we will be renovating each property selected for this pilot program, it's important to understand the effects renovating a property will have on it's value.

In [69]:

```
data['yr_renovated'].hist()
# taking a look at the distributions of homes renovated vs unrenovated
```

Out[69]: <AxesSubplot:>



This plot doesn't tell us much except that the majority of homes in this data set have never been renovated.

In [70]:

```
data['yr_renovated'].mean()
```

Out[70]: 82.43672236046925

In [71]:

```
data['yr_renovated'].isna().sum()/data.shape[0]

# 17% of the data in this column is missing, and in this column
```

Out[71]: 0.17812621737436696

In [72]:

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

I'm filling the NA values with 0, as the vast majority of homes have never been renovated in this dataset.

Analyzing Property Timelines

Because I'm now finished cleaning and preparing the data, I can start analyzing what data exists.

Comparing Renovated Homes with Unrenovated Homes

In [73]:
```
# the mean price for a home not renovated in this data set is 5.325143e+05

no_reno = data.loc[data['yr_renovated'] == 0]
no_reno_stats = no_reno.mean()
no_reno_stats
```

Out[73]:
```
id                4.648427e+09
price             5.178077e+05
bedrooms          3.373759e+00
bathrooms         2.102563e+00
sqft_living       2.047173e+03
sqft_lot          1.018782e+04
floors            1.496295e+00
waterfront        4.082867e-03
view              3.165482e-02
condition         2.419174e+00
grade             4.651394e+00
sqft_above        1.762171e+03
sqft_basement     3.855537e-01
yr_built          1.972364e+03
yr_renovated      0.000000e+00
zipcode           9.807812e+04
lat               4.756143e+01
long             -1.222163e+02
sqft_living15     1.972658e+03
sqft_lot15        9.898066e+03
dtype: float64
```

In [74]:
```
yes_reno = data.loc[data['yr_renovated'] != 0].copy()
yes_reno_stats = yes_reno.mean()
yes_reno_stats
```

Out[74]:
```
id                4.514145e+09
price             7.200811e+05
```

```
bedrooms          3.440459e+00
bathrooms         2.268651e+00
sqft_living       2.248066e+03
sqft_lot          1.052812e+04
floors            1.502869e+00
waterfront        3.299857e-02
view              1.147776e-01
condition         2.216643e+00
grade             4.714491e+00
sqft_above        1.816657e+03
sqft_basement     5.007174e-01
yr_built          1.939235e+03
yr_renovated      1.996222e+03
zipcode           9.809764e+04
lat               4.758326e+01
long             -1.222680e+02
sqft_living15     1.947396e+03
sqft_lot15        9.874834e+03
dtype: float64
```

FURTHER WORK:

I'd like to come back to these statistics and do a much further analysis on the differences between the average renovated home vs non-renovated home on other variables besides price. IE does the average renovated home have larger living space or neighbors with more living space than non-renovated counterparts. But for now I will only be comparing the prices.

I'd also like to come back to the data and isolate non-renovated homes that have gone up in the last 10 years from all other non-renovated homes and compare the three groups. However, in my initial research, I have only had the resources to do a brief inspection of renovated VS non-renovated.

In [75]:

```
reno_differences = yes_reno_stats - no_reno_stats
reno_difference = reno_differences['price']
reno_difference
```

Out[75]:

```
202273.36808995338
```

Of note: this is the exact calculated difference in average price between homes that have been renovated vs. homes that have not been renovated, in the data as it's been so far cleaned and prepared. While this statistic has not been scaled by amount, such as bedrooms or living space for example, this statistic does indicate indeed that homes that were renovated in this data set where $201,941 more valuable on average.

In [76]:
```
average_yes_reno_price = yes_reno_stats['price']
average_yes_reno_price
```
Out[76]:
```
720081.1162123386
```
In [77]:
```
budget = 100000000
baseline_yes_reno = budget / average_yes_reno_price
baseline_yes_reno
```
Out[77]:
```
138.87324323404678
```
The average non renovated home in the remaining data after cleaning and preparation, is priced at $719,794. If we were to hypothetically spend our entire budget on the average renovated home, we'd be able to buy 138 properties without going over budget.

In [78]:
```
average_no_reno_price = no_reno_stats['price']
average_no_reno_price
```
Out[78]:
```
517807.7481223852
```
In [79]:
```
baseline_no_reno = budget / average_no_reno_price
baseline_no_reno
```
Out[79]:
```
193.12186880673084
```
In [80]:
```
baseline_no_reno - baseline_yes_reno
```
Out[80]:
```
54.24862557268406
```
The average non renovated home in the remaining data after cleaning and preparation, is priced at $517,853. That may sound pretty close to an average renovated home but it fits into the laid out budget for the project about 193 times.

For clarity, if we were only buying average priced non-renovated homes, we'd be able to buy 54 more homes than if we were only buying average priced non renovated homes.

To properly compare visually, I will take each home that has been renovated, and subtract the average price of a non-renovated home from the sale price, this will recenter the visualization's bars to show how above or below each property was compared to the average un-renovated home.

In [81]:

```
# make a function where the input is each price for each property that has
# been renovated and the output is that price minus the average price for a non
# renovated home

def find_price_adjusted(price):
    price = price - average_no_reno_price
    return price

yes_reno['price'] = yes_reno['price'].map(find_price_adjusted)
```

To organize the data, and get a sense of timeline, I'm going to be putting these properties into chronological bins. Every 5 years will show very basic changes by half-decade.

FURTHER WORK:

I'd like to come back to this visualization with more time and resources, and make the bins every four years, in correlation with the mayoral history of Seattle.

In [82]:

```
# aggregating the data by half-decade

bins = [ 1935, 1940, 1945, 1950, 1955, 1960, 1965, 1970, 1975, 1980, 1985,
        1990, 1995, 2000, 2005, 2010, 2015]

labels = ['1940', '1945', '1950', '1955', '1960', '1965', '1970', '1975', '1980',
          '1985', '1990', '1995', '2000', '2005', '2010', '2015']

cut_data, bin_edges = pd.cut(yes_reno['yr_renovated'], bins = bins,
                              labels = labels, retbins = True)
bin_edges
```

Out[82]:

```
array([1935, 1940, 1945, 1950, 1955, 1960, 1965, 1970, 1975, 1980, 1985,
       1990, 1995, 2000, 2005, 2010, 2015])
```

In [83]:

```
yes_reno.head()

#taking a look to make sure aggrigation worked correctly
```

Out[83]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade
1	6414100192	12/9/2014	20192.251878	3	2.25	2570	7242	2.0	0	0.0		7

1	6414100192	12/9/2014	20192.251878	3	2.25	2570	7242	2.0	0	0.0	...	4
35	9547205180	6/13/2014	178192.251878	3	2.50	2300	3060	1.5	0	0.0	...	5
95	1483300570	9/8/2014	387192.251878	4	2.50	3300	10250	1.0	0	0.0	...	4
103	2450000295	10/7/2014	572192.251878	3	2.50	2920	8113	2.0	0	0.0	...	5
125	4389200955	3/2/2015	932192.251878	4	2.75	2750	17789	1.5	0	0.0	...	5

5 rows × 21 columns

In [84]:

```python
yes_reno['yr_renovated'] = cut_data

# replacing the yr_renovated with the binned data
```

Since I have homes grouped by 5 year bins, it's important that for each bin I get an average price instead of a sum value of all the properties in that bin, since I have different numbers of properties in each bin. Getting the average price per bin will show something comparable.

In [85]:

```python
# making sure I get the average per bin, and resetting the index for clarity

yes_reno_averages = yes_reno.groupby(by='yr_renovated').mean()
yes_reno_averages = yes_reno_averages.reset_index()
yes_reno_averages
```

Out[85]:

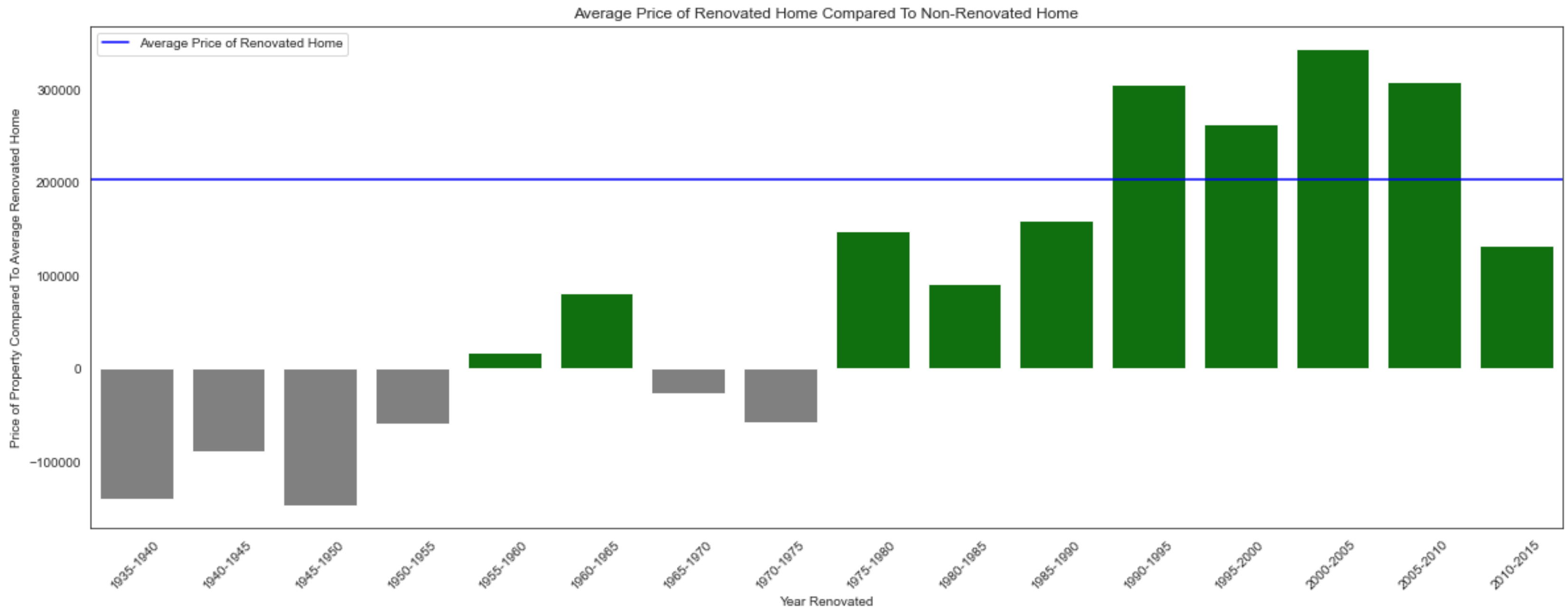
	yr_renovated	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	
0	1940	1.451350e+09	-139407.748122	2.000000	1.000000	1150.000000	4470.000000	1.000000	0.000000	0.(
1	1945	4.448326e+09	-88557.748122	2.750000	1.687500	1607.500000	5750.000000	1.250000	0.000000	0.(
2	1950	3.606523e+09	-146841.081456	1.333333	1.166667	963.333333	4399.333333	1.166667	0.000000	0.(
3	1955	4.577260e+09	-59474.414789	3.166667	1.458333	1586.666667	19070.833333	1.000000	0.000000	0.(
4	1960	5.478101e+09	17870.251878	3.100000	1.500000	1814.500000	9129.700000	1.300000	0.100000	0.
5	1965	3.869926e+09	81296.098031	3.846154	2.038462	2057.692308	9422.538462	1.461538	0.153846	0.
6	1970	4.685418e+09	-26341.081456	3.380952	1.904762	1993.809524	9223.857143	1.309524	0.000000	0.(
7	1975	5.433291e+09	-58104.081456	2.933333	1.683333	1822.666667	14671.600000	1.400000	0.000000	0.(
8	1980	4.851001e+09	147021.797332	2.909091	1.931818	1958.181818	10130.000000	1.636364	0.045455	0
9	1985	4.466804e+09	90343.251878	3.320755	2.037736	2063.584906	13085.981132	1.500000	0.056604	0

10	1990	4.876861e+09	158870.585211	3.291667	2.093750	2166.250000	13860.222222	1.555556	0.097222	0.
11	1995	4.523660e+09	305449.735749	3.451613	2.282258	2379.016129	11229.774194	1.612903	0.064516	0.
12	2000	4.254875e+09	261863.451878	3.400000	2.523333	2478.493333	12374.360000	1.533333	0.026667	0.
13	2005	4.306589e+09	343149.008634	3.630631	2.495495	2466.252252	10483.090090	1.617117	0.018018	0
14	2010	4.364097e+09	308062.113580	3.595745	2.500000	2468.553191	8866.255319	1.537234	0.010638	0.
15	2015	4.627709e+09	132147.432329	3.563910	2.313910	2121.827068	7581.781955	1.375940	0.000000	0.

Visualizing the Comparison

In [86]:

```
# color coding the bars to be green if the average price of home in that bin
# is higher than the average non-renovated home, and grey if below, to help
# visual intuition

values = yes_reno_averages['price']
colors = ['grey' if (val < 0) else 'green' for val in values]

# making the graph
plt.figure(figsize = (20,7))
ax = sns.barplot(data = yes_reno_averages, x = 'yr_renovated',
                 y ='price', palette = colors)

# making the labels for the bins
ax.set_xticklabels(['1935-1940', '1940-1945', '1945-1950', '1950-1955',
                      '1955-1960', '1960-1965', '1965-1970', '1970-1975',
                     '1975-1980', '1980-1985', '1985-1990', '1990-1995',
                     '1995-2000', '2000-2005', '2005-2010', '2010-2015'])

# rotating my labels
plt.xticks(rotation=45);

# making the blue line where the average price for renovated home is with legend
plt.axhline(reno_difference, color = 'blue', label = "Average Price of Renovated Home")
plt.legend()

# making the labels and titles for the axes
ax.set(xlabel = 'Year Renovated',
                 ylabel = 'Price of Property Compared To Average Renovated Home',
                 title = "Average Price of Renovated Home Compared To Non-Renovated Home");
```



KEY

X axis = average price of non-renovated home

Green Bar = above average price of non-renovated home

Blue Line = average price of renovated home

This plot's 0 on the Y axis has been adjusted so that everything above are properties where their price (grouped by every 5 years,) is above the average price for a non-renovated property. The blue line represents the average price of properties that have been renovated. Properties above the blue line represent renovated properties that are valued above the average renovated property.

Observations:

1) all properties renovated after 1975 are more valuable than the average renovated property.

2) properties renovated in the years between 1990 to 2010, are averaging as a higher price compared even to other renovated properties.

Inferences:

Inferences:

Renovating properties increases their value, that's to be expected. However, we now can now infer that if we renovate properties that have already been renovated but their renovation happened before 1975, we can "flip" those properties into having a higher value.

Home History

Age of Home

In [87]:

```
def age_of_home(year_built):
    age = 2014 - year_built
    return age

data['age_of_home'] = data['yr_built'].map(age_of_home)

# showing the age of the home rather than the year it was built, to explore
# a different perspective on the data
```

In [88]:

```
data.head()
```

Out[88]:

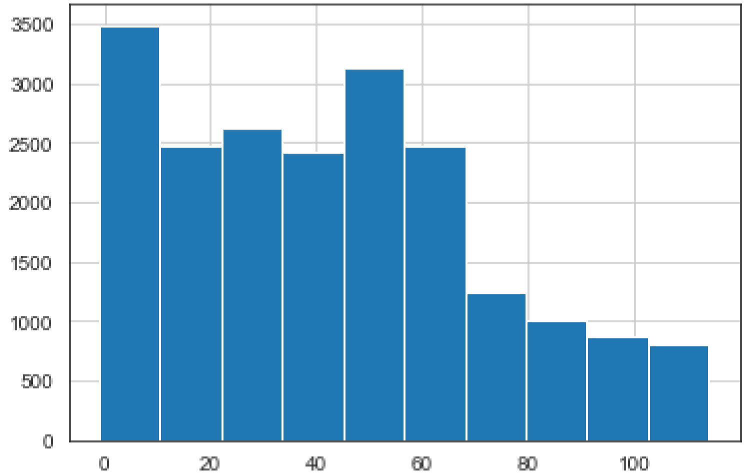
	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	sqft_above	sq
0	7129300520	10/13/2014	221900.0	3	1.00	1180	5650	1.0	0	0.0	...	1180	
1	6414100192	12/9/2014	538000.0	3	2.25	2570	7242	2.0	0	0.0	...	2170	
2	5631500400	2/25/2015	180000.0	2	1.00	770	10000	1.0	0	0.0	...	770	
3	2487200875	12/9/2014	604000.0	4	3.00	1960	5000	1.0	0	0.0	...	1050	
4	1954400510	2/18/2015	510000.0	3	2.00	1680	8080	1.0	0	0.0	...	1680	

5 rows × 22 columns

In [89]:

```
data['age_of_home'].hist()

# seeing the distributions of how old the home is
```

Out[89]: <AxesSubplot:>



Year Modified

I'm going to make a new column for year modified. This column will either list the amount of years since last renovation or the date of construction, whichever one happened most recently. This column helps liken newer non-renovated homes with recently renovated homes, and older non-renovated homes with homes that were renovated a while ago.

In [90]:

```
data['yr_mod'] = data[['yr_built','yr_renovated']].max(axis=1)

# this will select the most recent value between either year built or year renovated
```

In [91]:

```
data.head()
```

Out[91]:

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

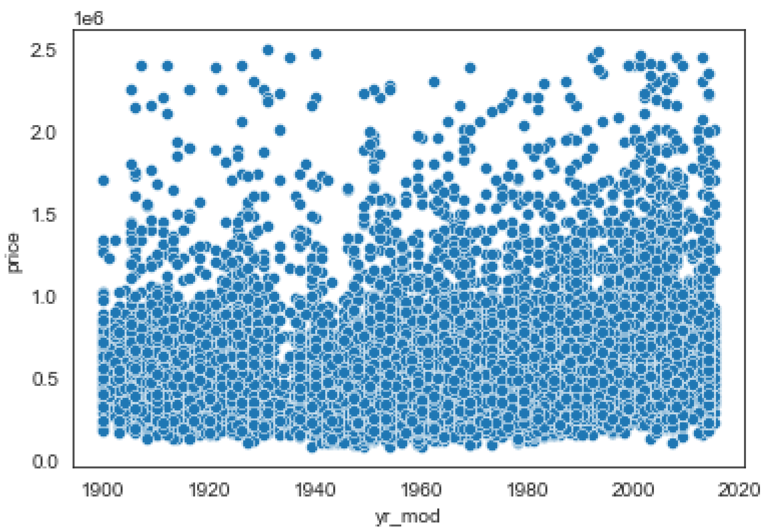
5 rows × 23 columns

In [92]:

In [92]:
```
data['yr_mod'].value_counts().head()
```

Out[92]:
```
2014.0    626
2005.0    465
2006.0    452
2003.0    440
2004.0    435
Name: yr_mod, dtype: int64
```

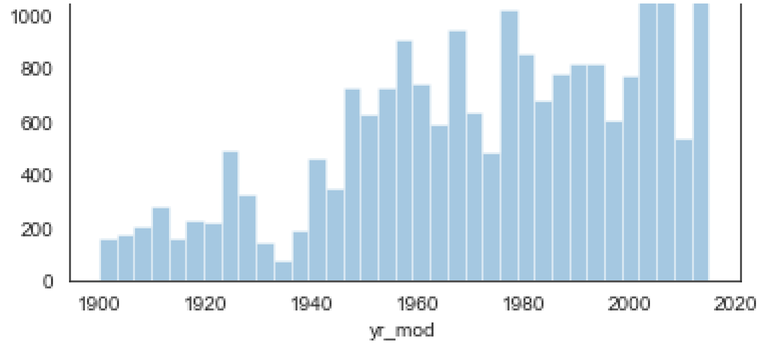
In [93]:
```
sns.scatterplot(data = data, x = 'yr_mod', y = 'price')
plt.show()
```



In [94]:
```
sns.distplot( a=data["yr_mod"], hist=True, kde=False, rug=False )
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/distributions.py:2551:
FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt
your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an ax
es-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[94]:
```
<AxesSubplot:xlabel='yr_mod'>
```



Distance Data

Now that I have limited the amount of properties I'll be looking up additional information for, it's time to gather some more data.

Using Google's Geocoding api and scraping with Selenium to measure distances away from downtown, police stations, hospitals and schools.

ASSUMPTIONS:

I have not had the resources yet to cross reference if the infrastructure points of interest have changed since 2015. I'm assuming in this data that the locations of Hospitals, Police Stations and Schools in 2022 is the same as it was in 2015.

FURTHER WORK:

I'd also like to include data about firehouses, which I have a source for, located in "Data" but didn't have the resources to currently explore. As the scope of this project increases, I'd like to include distances away from public parks, libraries, colleges and universities, and public transportation.

I'd also like to do more research about the history of these infrastructure points to counteract the assumptions stated previously.

Getting Infrastucture Data

In [95]:

```
# for the api
```

```
with open('.secrets/google.json') as f:
    creds = json.load(f)
```

Making a function that intakes a list of addresses and outputs a list of tuples, each tuple containing the coordinates for that address using google's API.

In [96]:
```
# making an empty dictionary to contain the address as the key and a tuple
# of latitude and logitude as the values.

def get_coordinates(addresses):
    for address in addresses:

        google_query = f"https://maps.googleapis.com/maps/api/geocode/json?address={address}&key={creds[

        results = requests.get(google_query)
        results_json = results.json()
        lat = results_json['results'][0]['geometry']['location']['lat']
        long = results_json['results'][0]['geometry']['location']['lng']

        coordinates[f"{address}"]=(lat, long)
```

Downtown

In [97]:
```
downtown = (47.6050, -122.3344)
```

Making a function that intakes the dataframe, and a specific location, as a touple of coordinates and dds a new row to the DF, column name as the pair of coordinates for that location, this column lists the distance each property is away from that location:

Hospitals

CITATION

https://data-seattlecitygis.opendata.arcgis.com/datasets/hospitals/explore?showTable=true

In [98]:
```
# loading the data

hospital_data = pd.read_csv('Data/Hospitals.csv')
```

In [99]:

```
hospital_data.head()

# taking a look at the data in this csv
```

Out[99]:

	X	Y	OBJECTID	FACILITY	ADDRESS	SE_ANNO_CAD_DATA	CITY	ACUTE_CARE	GIS_EDT_DT	
0	-122.336888	47.714248	1	UW Medical Center - Northwest	1550 N 115th ST	NaN	Seattle, WA, 98133-8401	Y	2020/03/17 00:00:00+00	https:/
1	-122.379555	47.667365	2	Swedish Medical Center - Ballard Campus	5300 Tallman Ave NW	NaN	Seattle, WA 98107	Y	2020/03/17 00:00:00+00	https:/
2	-122.341419	47.457800	3	Highline Medical Center	16251 Sylvester Rd SW	NaN	Burien, WA, 98166-3017	Y	2020/03/17 00:00:00+00	https
3	-122.214121	47.442273	4	Valley Medical Center	400 S 43rd St	NaN	Renton, WA, 98055-5714	Y	2020/03/17 00:00:00+00	https:/
4	-122.327279	47.292612	5	St. Francis Hospital	34515 9th Ave S	NaN	Federal Way, WA, 98003-6761	Y	2020/03/17 00:00:00+00	https

In [100…

```
hospital_addresses = hospital_data['ADDRESS'] + hospital_data['CITY']

# getting proper addresses for the Google API
```

In [101…

```
hospital_addresses.head()

# seeing the addresses
```

Out[101…

```
0         1550 N 115th STSeattle, WA, 98133-8401
1           5300 Tallman Ave NWSeattle, WA 98107
2     16251 Sylvester Rd SWBurien, WA, 98166-3017
3            400 S 43rd StRenton, WA, 98055-5714
```

```
4     34515 9th Ave SFederal Way, WA, 98003-6761
dtype: object
```

In [102…

```
hospital_names = hospital_data['FACILITY']

# making a list of hospital names for later
```

Police Stations

CITATION

http://www.seattle.gov/police/about-us/police-locations

In [103…

```
# I didn't bother to scrape for police data, there are only 5 stations
# scraping would be more work than hardcoding at this point

police_stations = {'north_ps' : "10049 College Way N. Seattle, WA 98133",
'west_ps' : "810 Virginia Street, Seattle, WA 98101",
'east_ps' : "1519 12th Avenue Seattle, WA 98122",
'south_ps' : "3001 S. Myrtle Seattle, WA 98108",
'southwest_ps' : "2300 S.W. Webster Seattle, WA 98106"}

ps_addresses = police_stations.values()
ps_names = police_stations.keys()
```

Schools

For the purposes of this project, I'm only including public school data K-12.

Scraping for School Data

In [104…

```
# first telling selenium where the webdriver is on my local machine
driver = webdriver.Chrome('/Users/b0ihazard/Desktop/chromedriver')

# telling selenium where the url is that I want to scrape from and to bring
# that up
school_url = "https://www.seattleschools.org/schools/"
go_article = driver.get(school_url)

# making a list of school names and addresses as selenium objects
grab_school_names = driver.find_elements(By.CSS_SELECTOR, "h4.list-item-title")
```

```
grab_school_addresses = driver.find_elements(By.CSS_SELECTOR, "address")

# making corrasponding lists of addresses and names for text strings
school_addresses = []
school_names = []

# reformatting each raw address to a string and putting it on the address list
for raw_address in grab_school_addresses:
    address = raw_address.text.split("Main Office")[0]
    school_addresses.append(address)

# reformatting each raw name to a string and putting it on the name list
for raw_name in grab_school_names:
    school_name = raw_name.text
    school_names.append(school_name)

# closing the webdriver after use
driver.close()
```

In [105…

```
# taking a look at each school just to makesure formatting worked

number = 0

for address in school_addresses:
    number = number + 1
    print(number, address)
```

```
1 6110 28th Ave. NW
Seattle, WA 98107

2 3928 S Graham St.
Seattle, WA 98118

3 8601 Rainier Ave. S
Seattle, WA 98118

4 3010 59th Ave. SW
Seattle, WA 98116

5 3701 SW 104th St.
Seattle, WA 98146

6 1301 E Yesler Way
Seattle, WA 98122
```

```
7 1418 NW 65th St.
Seattle, WA 98117

8 2025 14th Ave. S
Seattle, WA 98144

9 3921 Linden Ave. N
Seattle, WA 98103

10 13052 Greenwood Ave. N
Seattle, WA 98133

11 3311 NE 60th St.
Seattle, WA 98115

12 520 NE Ravenna Blvd.
Seattle, WA 98115

13 1700 N 90th St.
Seattle, WA 98103

14 2550 34th Ave. W
Seattle, WA 98199

15 3737 NE 135th St.
Seattle, WA 98125

16 2600 SW Thistle St.
Seattle, WA 98126

17 5511 15th Ave. S
Seattle, WA 98108

18 2424 7th Ave. W
Seattle, WA 98119

19 723 S. Concord St.
Seattle, WA 98108

20 7821 Stone Ave. N
Seattle, WA 98103

21 2820 S Orcas St.
Seattle, WA 98108
```

```
22 7711 43rd Ave. NE
Seattle, WA 98115

23 2601 SW Kenyon St.
Seattle, WA 98126

24 4525 S Cloverdale St.
Seattle, WA 98118

25 3003 NE 75th St.
Seattle, WA 98115

26 9709 60th Ave. S
Seattle, WA 98118

27 3800 SW Findlay St.
Seattle, WA 98126

28 3013 S Mt Baker Blvd.
Seattle, WA 98144

29 400 23rd Ave.
Seattle, WA 98122

30 4320 SW Myrtle St.
Seattle, WA 98136

31 5013 SW Dakota St.
Seattle, WA 98116

32 5149 S Graham St.
Seattle, WA 98118

33 2400 N 65th St.
Seattle, WA 98103

34 144 NW 80th St.
Seattle, WA 98117

35 1610 N 41st St.
Seattle, WA 98103

36 4100 39th Ave. S
Seattle, WA 98118

37 11530 12th Ave. NE
```

```
Seattle, WA 98125

38 1012 SW Trenton St.
Seattle, WA 98106

39 1819 N 135th St.
Seattle, WA 98133

40 3528 S Ferdinand St.
Seattle, WA 98118

41 11051 34th Ave. NE
Seattle, WA 98125

42 201 Garfield St.
Seattle, WA 98109

43 3301 S Horton St.
Seattle, WA 98144

44 4030 NE 109th St.
Seattle, WA 98125

45 4057 5th Ave. NE
Seattle, WA 98105

46 7201 Beacon Ave. S
Seattle, WA 98108

47 2645 California Ave. SW
Seattle, WA 98116

48 4530 46th Ave. NE
Seattle, WA 98105

49 4000 27th Ave. W
Seattle, WA 98199

50 135 32nd Ave.
Seattle, WA 98122

51 3015 NW 68th St.
Seattle, WA 98117

52 4400 Interlake Ave. N
```

```
Seattle, WA 98103

53 5950 Delridge Way SW
Seattle, WA 98106

54 1058 E Mercer St.
Seattle, WA 98102

55 7735 25th Ave. NW
Seattle, WA 98117

56 3429 45th Ave. SW
Seattle, WA 98116

57 1121 33rd Ave.
Seattle, WA 98122

58 2418 28th Ave. W
Seattle, WA 98199

59 4925 Corson Ave. S
Seattle, WA 98108

60 6725 45th Ave. S
Seattle, WA 98118

61 1915 1st Ave. W
Seattle, WA 98119

62 144 NE 54th St.
Seattle, WA 98105

63 1617 38th Ave. E
Seattle, WA 98112

64 301 21st Ave. E
Seattle, WA 98112

65 1600 S Columbian Way
Seattle, WA 98108

66 John Marshall: 206-252-9900
Seattle U: 206-720-3078
North Seattle College: 206-934-3957
67 2409 22nd Ave. E
Seattle, WA 98112
```

```
68 10750 30th Ave. NE
Seattle, WA 98125

69 9018 24th Ave. NW
Seattle, WA 98117

70 11725 1st Ave. NE
Seattle, WA 98125

71 2410 E Cherry St.
Seattle, WA 98122

72 13018 20th Ave. NE
Seattle, WA 98125

73 504 NE 95th St.
Seattle, WA 98115

74 5215 46th Ave. S
Seattle, WA 98118

75 1901 SW Genesee St.
Seattle, WA 98106

76 2100 4th Ave. N
Seattle, WA 98109

77 8815 Seward Park Ave. S
Seattle, WA 98118

78 11650 Beacon Ave. S
Seattle, WA 98178

79 8311 Beacon Ave. S
Seattle, WA 98118

80 1330 N 90th St.
Seattle, WA 98103

81 1410 NE 66th St.
Seattle, WA 98115

82 7740 34th Ave. SW
Seattle, WA 98126
```

```
83 9501 20th Ave. NE
Seattle, WA 98115

84 1810 NW 65th St.
Seattle, WA 98117

85 6208 60th Ave. NE
Seattle, WA 98115

86 1812 SW Myrtle St.
Seattle, WA 98106

87 1700 E Union St.
Seattle, WA 98122

88 2445 3rd Ave. S.
Seattle, WA 98134

89 4800 S Henderson St.
Seattle, WA 98118

90 1242 18th Ave. E
Seattle, WA 98112

91 305 Harrison St.
Seattle, WA 98109

92 7712 40th Ave. NE
Seattle, WA 98115

93 2401 S Irving St.
Seattle, WA 98144

94 2500 Franklin Ave. E
Seattle, WA 98102

95 7047 50th Ave. NE
Seattle, WA 98115

96 520 NE Ravenna Blvd (interim location)
Seattle, WA 98115

97 2101 S Jackson St.
Seattle, WA 98144
```

```
98 2720 NE 85th St.
Seattle, WA 98115

99 Schmitz Park Elementary
5000 SW Spokane St.
Seattle WA 98116

100 3000 California Ave. SW
Seattle, WA 98116

101 5601 4th Ave NW
Seattle, WA 98107

102 9201 15th Ave. NW
Seattle, WA 98117

103 1320 NW 75th St.
Seattle, WA 98117

104 3701 S Kenyon St.
Seattle, WA 98118
```

In [106…

```
school_addresses.remove(school_addresses[65])
```

This isn't an address so we must remove it from the data, but I have to remember to take the accompanying name, Middle College from the list as well.

In [107…

```
school_names[65]
```

Out[107…

```
'Middle College'
```

In [108…

```
school_names.remove('Middle College')
```

Finding Distance for Infrastructure

Getting a dictionary of coordinates based on the collected addresses for Hospitals, Police Stations, Schools

In [109…

```
coordinates ={}
```

In [110…
```
get_coordinates(hospital_addresses)
```

In [111…
```
get_coordinates(ps_addresses)
```

In [112…
```
get_coordinates(school_addresses)
```

In [113…
```
coordinates['downtown'] = downtown
```

Getting a list of all the names of these points of interests for the column names.

In [114…
```
all_names = []

for hospital in hospital_names:
    all_names.append(hospital)
for ps_name in ps_names:
    all_names.append(ps_name)
for school_name in school_names:
    all_names.append(school_name)
```

In [115…
```
all_names.append('downtown')
```

In [116…
```
len(all_names) == len(coordinates)

# to make sure that each name has a set of coordinates and vice versa
```

Out[116…
```
True
```

In [117…
```
counter = 0
columns = {}
for name in all_names:
    columns[name] = counter
    counter +=1

# making a dictionary of column names with an index for referencing later
```

Making a function that will calculate how far away each point of interest is by making each point of interest a column, and making the rows how far away that point of interest is from each property. Each column should be labeled properly by name.

In [118…

```
# intake: dataframe, location as a touple of coordinates, list of names
# output: column on dataframe that measures distance from location with column name

def distance_from (data, coordinates, all_names):
    counter = 0
    for coordinate_value in coordinates.values():
        distance_from_coordinates = []
        col_name = all_names[counter]
        for x in data.index:
            property_coordinates = (data.loc[x]['lat'], data.loc[x]['long'])
            dist = distance.distance(property_coordinates, coordinate_value).miles
            distance_from_coordinates.append(dist)
        data[col_name] = distance_from_coordinates
        counter += 1

#     return f"added distance from {coordinate}, {col_name} to each row"
```

In [119…

```
distance_from(data, coordinates, all_names)

# implimenting this function on all of our data does take some time, as a note
```

Finding the closest possible Hospital, Police Station and School for each property.

FURTHER WORK:

This is now a much larger amount of information, I've added over 150 columns to the original dataset. In further work, I would like to isolate each point of interest, to be able to compare how individual points of interest affect property values when compared to other individual points of interest. However, given the parameters of my work so far, I must simplify this data significantly. Therefore, I'll be reducing these 150 columns into only three columns, to show the closest possible hospital, police station and school. This will avoid the multicoliniarity problem with having too much coordinate data as well.

In [120…

```
hospitals = hospital_names
```

In [121…

```
# making a dataframe just for the hospital information for further work
# the last columns in this dataframe will default to the distance to the closest hospital
```

```
data_h = data[hospitals]
data_h.head()
data_h['closest_hospital_distance'] = data[['UW Medical Center - Northwest',
 'Swedish Medical Center - Ballard Campus',
 'Highline Medical Center',
 'Valley Medical Center',
 'St. Francis Hospital',
 'Snoqualmie Valley Hospital',
 'Swedish Medical Center - First Hill',
 'MultiCare  Auburn Medical Center',
 'Overlake Hospital Medical Center',
 "Seattle Children's Hospital",
 'Virginia Mason Medical Center',
 'EvergreenHealth - Kirkland',
 'Swedish Medical Center - Cherry Hill',
 'Seattle VA Medical Center',
 'Swedish Medical Center - Issaquah Campus',
 'Kaiser Permanente Capitol Hill Campus',
 'St. Elizabeth Hospital',
 'Harborview Medical Center',
 'University of Washington Medical Center']].min(axis=1)
```

```
<ipython-input-121-b04b706e5bbc>:6: 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-docs/stable/user_guide/indexing.ht
ml#returning-a-view-versus-a-copy
  data_h['closest_hospital_distance'] = data[['UW Medical Center - Northwest',
```

In [122…

```
# making a dataframe just for school information for future work
# the last columns in this dataframe will default to the distance to the closest school

data_s = data[school_names]
data_s['closest_school_distance'] = data[['Adams Elementary',
 'Aki Kurose Middle School',
 'Alan T. Sugiyama High School',
 'Alki Elementary',
 'Arbor Heights Elementary',
 'Bailey Gatzert Elementary',
 'Ballard High School',
 'Beacon Hill International Elementary',
 'Benjamin Franklin Day Elementary',
 'Broadview-Thomson K-8',
```

```
'Bryant Elementary',
'Cascade Parent Partnership Program',
'Cascadia Elementary',
'Catharine Blaine K-8',
'Cedar Park Elementary',
'Chief Sealth International High School',
'Cleveland High School',
'Coe Elementary',
'Concord International Elementary',
'Daniel Bagley Elementary',
'Dearborn Park International Elementary',
'Decatur Elementary',
'Denny International Middle School',
'Dunlap Elementary',
'Eckstein Middle School',
'Emerson Elementary',
'Fairmount Park Elementary',
'Franklin High School',
'Garfield High School',
'Gatewood Elementary',
'Genesee Hill Elementary',
'Graham Hill Elementary',
'Green Lake Elementary',
'Greenwood Elementary',
'Hamilton International Middle School',
'Hawthorne Elementary',
'Hazel Wolf K-8',
'Highland Park Elementary',
'Ingraham High School',
'Interagency Academy',
'Jane Addams Middle School',
'John Hay Elementary',
'John Muir Elementary',
'John Rogers Elementary',
'John Stanford International Elementary',
'Kimball Elementary',
'Lafayette Elementary',
'Laurelhurst Elementary',
'Lawton Elementary',
'Leschi Elementary',
'Licton Springs K-8',
'Lincoln High School',
'Louisa Boren K-8',
'Lowell Elementary',
'Loyal Heights Elementary',
'Madison Middle School',
```

```
'Madrona Elementary',
'Magnolia Elementary',
'Maple Elementary',
'Martin Luther King, Jr. Elementary',
'McClure Middle School',
'McDonald International Elementary',
'McGilvra Elementary',
'Meany Middle School',
'Mercer International Middle School',
'Montlake Elementary',
'Nathan Hale High School',
'North Beach Elementary',
'Northgate Elementary',
'Nova',
'Olympic Hills Elementary',
'Olympic View Elementary',
'Orca K-8',
'Pathfinder K-8',
'Queen Anne Elementary',
'Rainier Beach High School',
'Rainier View Elementary',
'Rising Star Elementary',
'Robert Eagle Staff Middle School',
'Roosevelt High School',
'Roxhill Elementary',
'Sacajawea Elementary',
'Salmon Bay K-8',
'Sand Point Elementary',
'Sanislo Elementary',
'Seattle World School',
'Skills Center',
'South Shore PK-8',
'Stevens Elementary',
'The Center School',
'Thornton Creek Elementary',
'Thurgood Marshall Elementary',
'TOPS K-8',
'View Ridge Elementary',
'Viewlands Elementary',
'Washington Middle School',
'Wedgwood Elementary',
'West Seattle Elementary',
'West Seattle High School',
'West Woodland Elementary',
'Whitman Middle School',
'Whittier Elementary'
```

```
 'Whittier Elementary',
 'Wing Luke Elementary']].min(axis=1)
```

```
<ipython-input-122-d282df538c6c>:5: 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-docs/stable/user_guide/indexing.ht
ml#returning-a-view-versus-a-copy
  data_s['closest_school_distance'] = data[['Adams Elementary',
```

In [123…

```
# data.to_csv('Housing_Project_Full')
```

In [124…

```
# making a dataframe just for police stations for future work
# the last columns in this dataframe will default to the distance to the closest police station


data_ps = data[ps_names]
data_ps['closest_police_station'] = data[['north_ps', 'west_ps', 'east_ps',
                                          'south_ps', 'southwest_ps']].min(axis=1)
```

```
<ipython-input-124-643df7c19b3e>:6: 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-docs/stable/user_guide/indexing.ht
ml#returning-a-view-versus-a-copy
  data_ps['closest_police_station'] = data[['north_ps', 'west_ps', 'east_ps',
```

Simplifying the Data To Only Show the Closest Distance

In [125…

```
to_drop = hospitals, school_names, ps_names
for columns in to_drop:
    data = data.drop(columns = columns)

# dropping all distance data from main dataframe
```

In [126…

```
data.head()
```

Out[126…

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	yr_built	yr_re

0	7129300520	10/13/2014	221900.0	3	1.00	1180	5650	1.0	0	0.0	...	1955
1	6414100192	12/9/2014	538000.0	3	2.25	2570	7242	2.0	0	0.0	...	1951
2	5631500400	2/25/2015	180000.0	2	1.00	770	10000	1.0	0	0.0	...	1933
3	2487200875	12/9/2014	604000.0	4	3.00	1960	5000	1.0	0	0.0	...	1965
4	1954400510	2/18/2015	510000.0	3	2.00	1680	8080	1.0	0	0.0	...	1987

5 rows × 24 columns

In [127…

```
to_add = data_h['closest_hospital_distance'], data_s['closest_school_distance'], data_ps['closest_police_
for column in to_add:
    data = data.join(column)

# adding simplified distance data back into main dataframe
```

In [128…

```
data.head()
```

Out[128…

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	lat	lo
0	7129300520	10/13/2014	221900.0	3	1.00	1180	5650	1.0	0	0.0	...	47.5112	-122.2
1	6414100192	12/9/2014	538000.0	3	2.25	2570	7242	2.0	0	0.0	...	47.7210	-122.3
2	5631500400	2/25/2015	180000.0	2	1.00	770	10000	1.0	0	0.0	...	47.7379	-122.2
3	2487200875	12/9/2014	604000.0	4	3.00	1960	5000	1.0	0	0.0	...	47.5208	-122.3
4	1954400510	2/18/2015	510000.0	3	2.00	1680	8080	1.0	0	0.0	...	47.6168	-122.0

5 rows × 27 columns

Exploring Distance Data

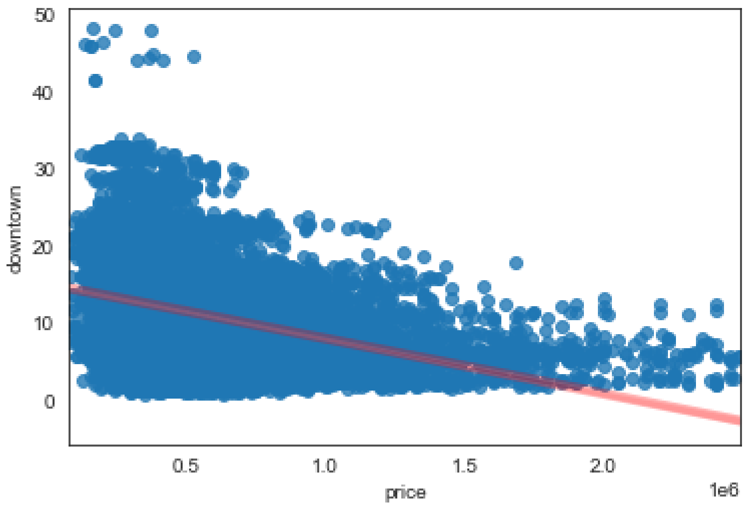
Downtown

I'm going to make a cut-off for houses more than 25 miles away from downtown, the goal of this project is to increase affordable housing in Seattle proper, not the suburbs.

In [129…

```
sns.regplot(data = data, x = 'price', y = 'downtown',
```

```
            line_kws={"color":"r","alpha":0.3,"lw":5})
plt.show()
```



In [130…

```
data = data.loc[data['downtown'] < 25]
```

In [131…

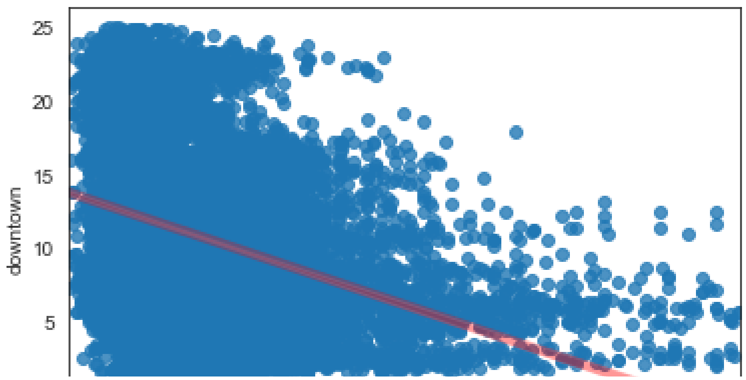
```
data['downtown'].max()
```

Out[131…

```
24.934920914948457
```

In [132…

```
sns.regplot(data = data, x = 'price', y = 'downtown',
            line_kws={"color":"r","alpha":0.3,"lw":5})
plt.show()
```



We can see some clustering in this graph, showing us that prices decrease as distance from downtown increases in the maximum prices for homes.

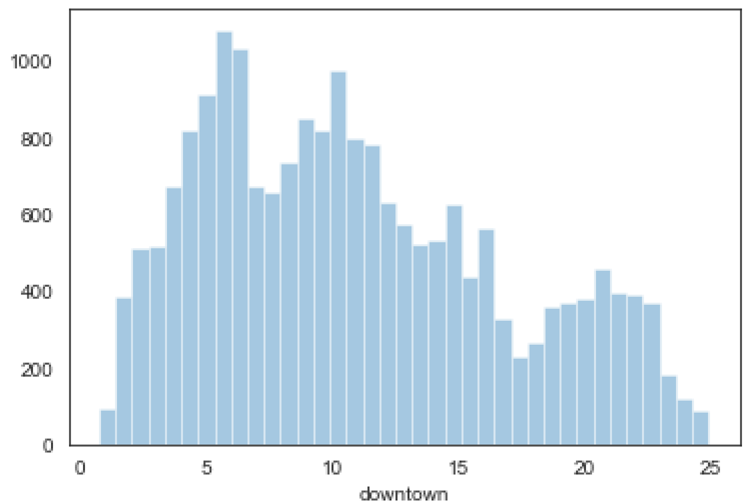
In [133…

```
sns.distplot( a=data["downtown"], hist=True, kde=False, rug=False )
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[133…

```
<AxesSubplot:xlabel='downtown'>
```



In [134…

```
data['downtown'].describe()
```

Out[134…

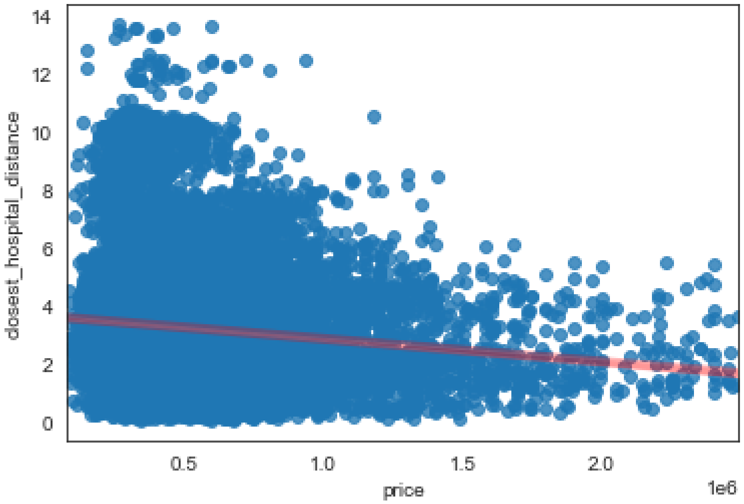
```
count    20109.000000
mean        10.923908
std          5.919272
min          0.719565
25%          5.979404
50%         10.079701
75%         15.018175
```

```
max          24.934921
Name: downtown, dtype: float64
```

Hospitals

In [135…

```
sns.regplot(data = data, x = 'price', y = 'closest_hospital_distance',
            line_kws={"color":"r","alpha":0.3,"lw":5})
plt.show()
```



Observations: while there are plenty of low priced homes close to hospitals, the upper limit of how expensive a home can be does look to go down the farther away from a hospital it is.

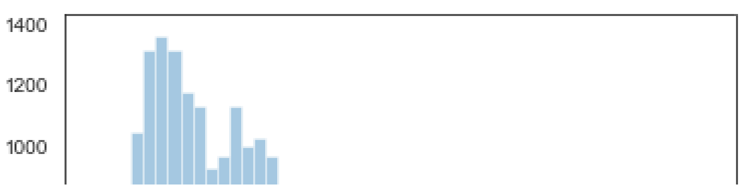
In [136…

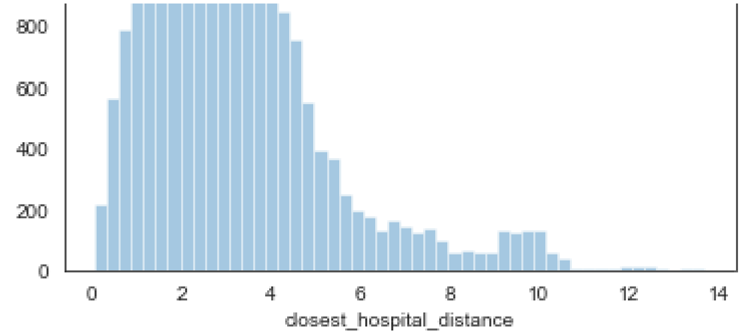
```
sns.distplot( a=data["closest_hospital_distance"], hist=True, kde=False, rug=False )
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/distributions.py:2551:
FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt
your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an ax
es-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[136…

```
<AxesSubplot:xlabel='closest_hospital_distance'>
```

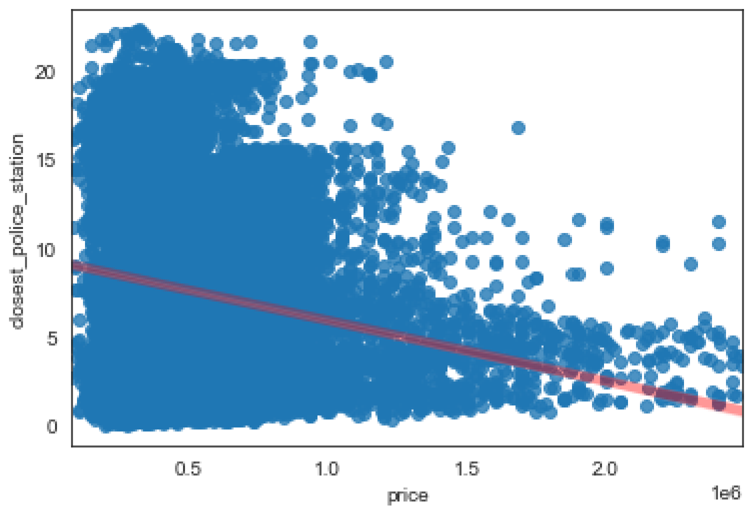




Police Stations

In [137…

```
sns.regplot(data = data, x = 'price', y = 'closest_police_station',
            line_kws={"color":"r","alpha":0.3,"lw":5})
plt.show()
```



Observations: while there are plenty of low priced homes close to police stations, the upper limit of how expensive a home can be does look to go down the farther away from a police station it is, similar to hospitals.

In [138…

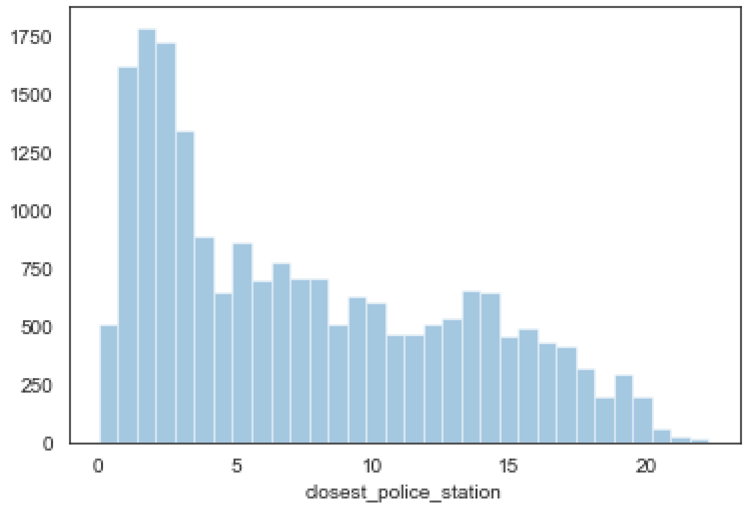
```
sns.distplot( a=data["closest_police_station"], hist=True, kde=False, rug=False )
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/distributions.py:2551:
FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt
your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an ax
es-level function for histograms).
```

```
es-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out [138…

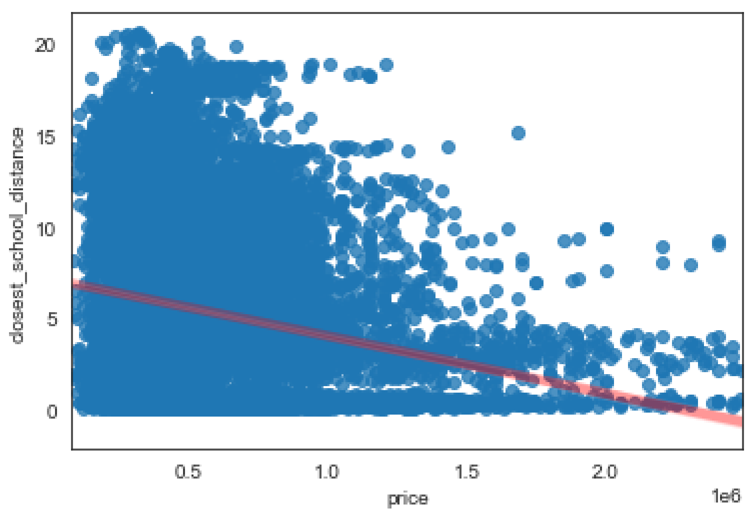
```
<AxesSubplot:xlabel='closest_police_station'>
```



Schools

In [139…

```
sns.regplot(data = data, x = 'price', y = 'closest_school_distance',
            line_kws={"color":"r","alpha":0.3,"lw":5})
plt.show()
```



Observations: while there are plenty of low priced homes close to schools, the upper limit of how expensive a home can be does look to go down the farther away from a school it is, similar to hospitals and police stations.

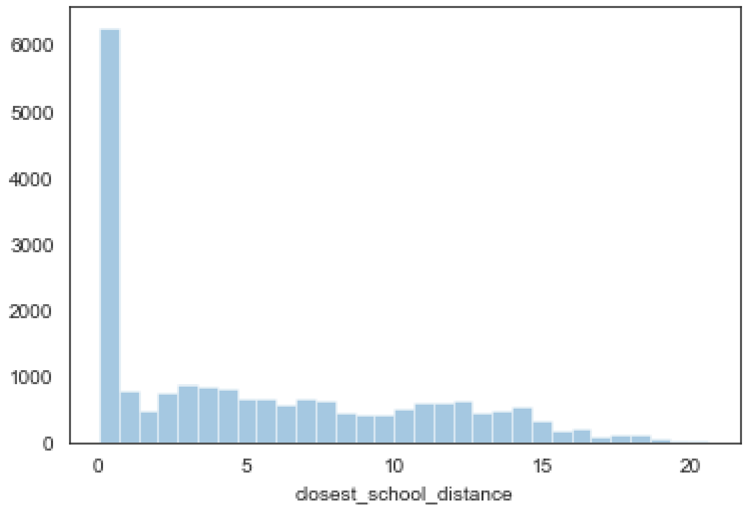
In [140…

```
sns.distplot( a=data["closest_school_distance"], hist=True, kde=False, rug=False )
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/distributions.py:2551:
FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt
your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an ax
es-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[140…

```
<AxesSubplot:xlabel='closest_school_distance'>
```



Preparing for Modeling

Dropping View and Waterfront Properties for Modeling

Since we are using a linear regression model, I must drop values on these binary features.

In [141…

```
data['view'].value_counts()
```

Out[141…

```
0.0    19419
1.0      690
Name: view, dtype: int64
```

In [142…

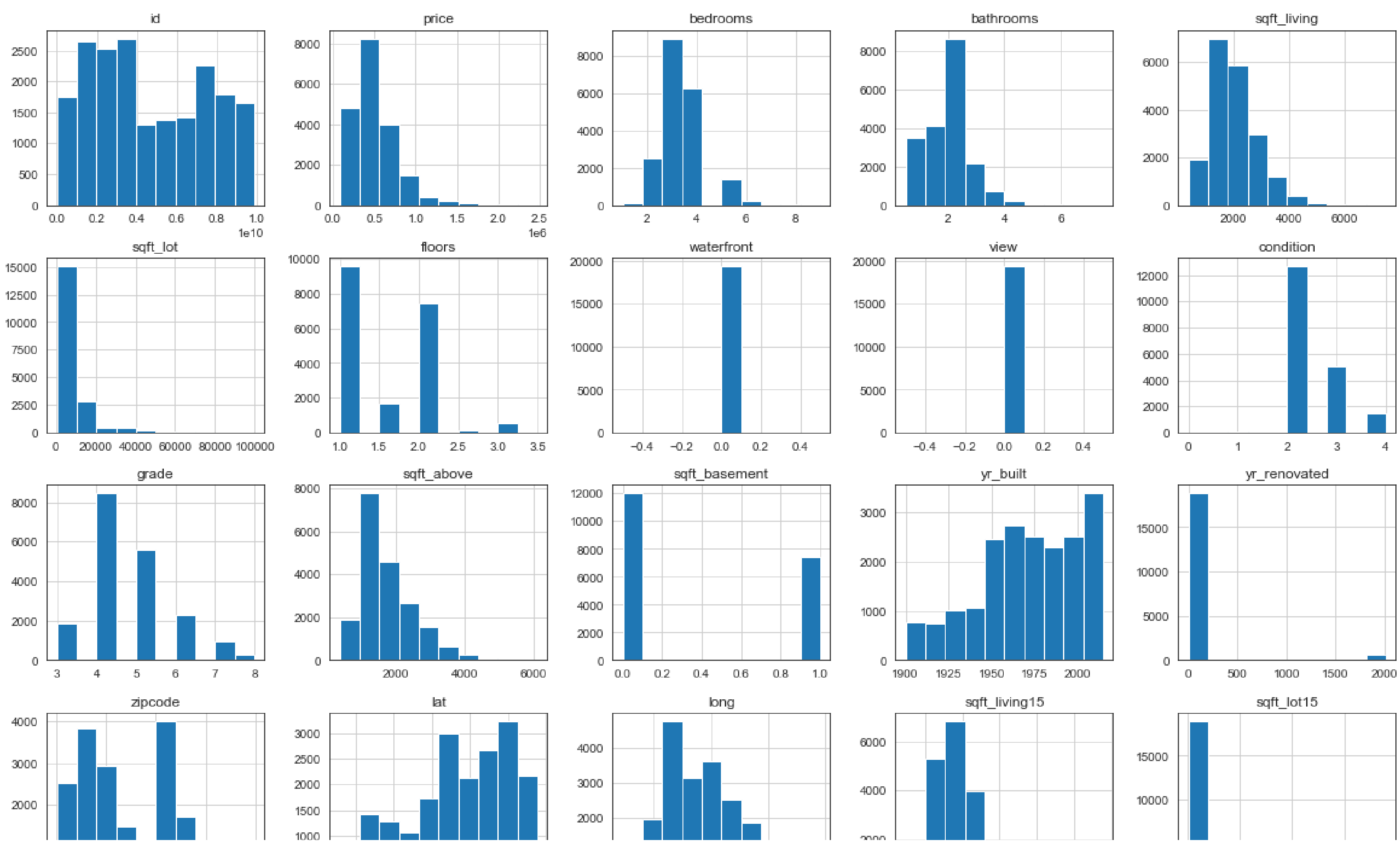
```
data['waterfront'].value_counts()
```

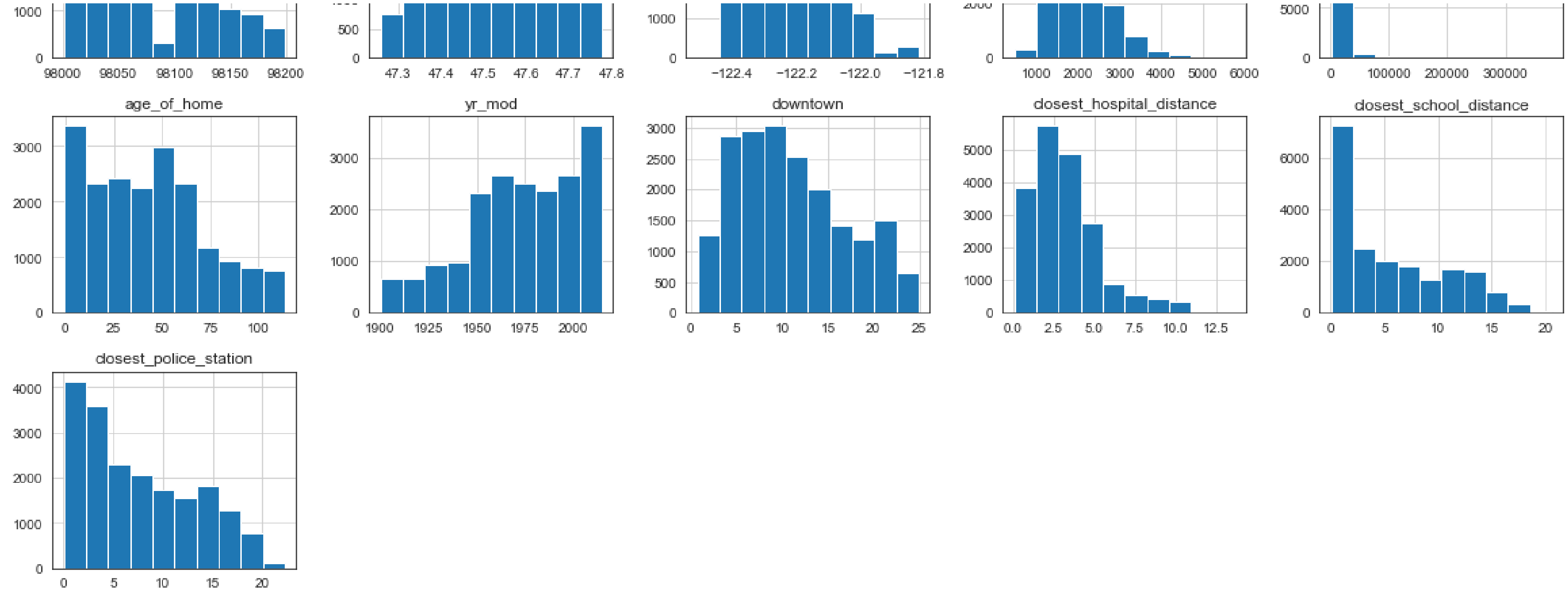
Out[142…
```
0    20005
1      104
Name: waterfront, dtype: int64
```

As a majority of data does not have a view or a waterfront, in order to continue to have enough data to manipulate and have findings I must drop properties where these values are at 1.

In [143…
```python
data = data.loc[data['view'] == 0]
data = data.loc[data['waterfront'] == 0]
```

In [144…
```python
data.hist(figsize=(20,20));
```





Zipcode

Using this kind of model, I'll need to pick one zipcode as the default zipcode so we can compare the price of a property going up or down depending on what zipcode it's in.

Seeing as this project aims to specifically research class disparity, I'll be using 98039 as my default zipcode, it's the wealthiest zipcode in Seattle.

CITATION: https://www.zipdatamaps.com/economics/income/agi/metro/wealthiest-zipcodes-in-metro-seattle-tacoma (Source: US Internal Revenue Service - 2018)

By comparing all other zipcodes to the wealthiest zipcode, we can see the full scope of how much a zipcode can affect property values. I'm assuming 1) that public infrastructure is working well in this zipcode, and is reflected partially in this wealth. I'm also assuming 2) that this zipcode was also the wealthiest zipcode in 2015.

FURTHER WORK: I'd like to, in the future, expand this model to have user INPUT functionality, so the user can select which zipcode they'd like to make the default zipcode. That would allow for civic workers to compare each and every zipcode to each and ever zipcode in relative scale to each other.

In [145…

```
zip_dummies = pd.get_dummies(data.zipcode, drop_first=False, prefix = 'zipcode')
```

```
# Concatenate the dummies to original dataframe
zipcode_data = pd.concat([data, zip_dummies], axis='columns')

## drop zipcode 98039 instead of "first"
```

In [146…

```
zipcode_data= zipcode_data.drop(columns = 'zipcode_98039')
```

In [147…

```
zipcode_data.shape
```

Out[147…

```
(19414, 94)
```

In [148…

```
data.shape
```

Out[148…

```
(19414, 27)
```

I'll be using a combination of data and zipcode_data when appropriate, they are identical except one has each zipcode as a separate column and one does not. Having both of them is useful in making the correlation matrix.

Use csvs when only working on models for speed, and api usage

In [149…

```
# data = pd.read_csv('Data/Housing_Project_No_Dummies',
#                     index_col="Unnamed: 0").reset_index(drop=True)
# zipcode_data = pd.read_csv('Data/Housing_Project_With_Zipcodes',
#                     index_col="Unnamed: 0").reset_index(drop=True)
```

In [150…

```
# data.to_csv('Housing_Project_No_Dummies')
# zipcode_data.to_csv('Housing_Project_With_Zipcodes')
```

Seeing the Correlation

The next step is to avoid multicollinearity by looking at the correlation matrix. When two columns have a correlation of over 70 percent, one will be chosen over the other for reasons described below.

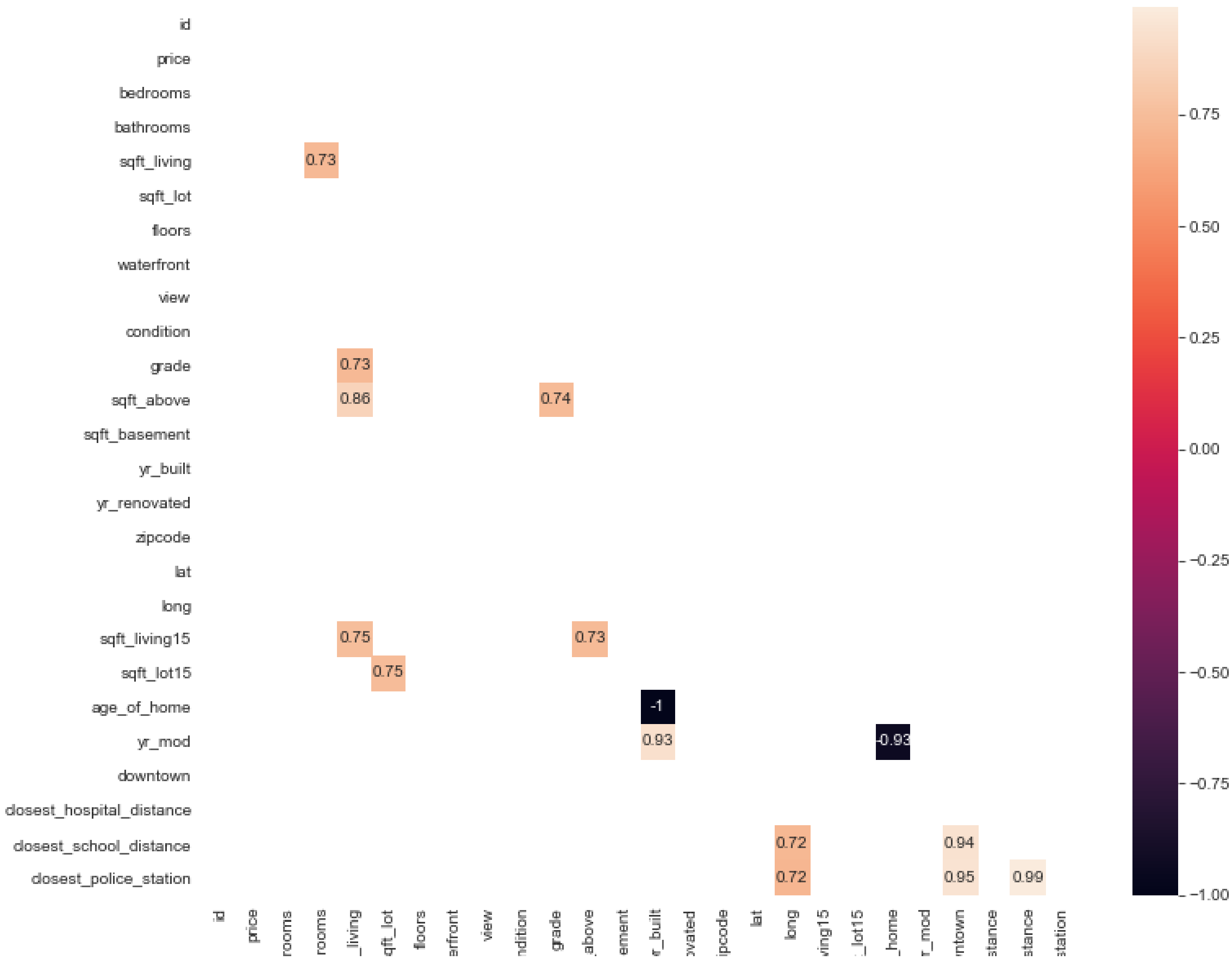
In [151…

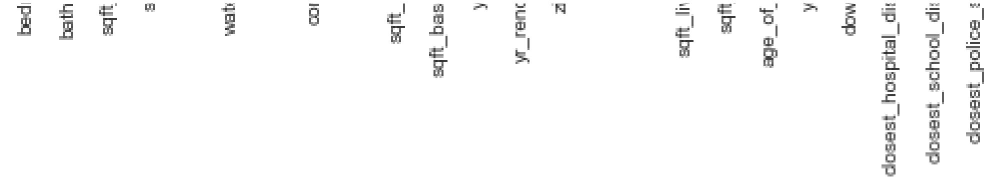
```
corr = data.corr()
# building the matrix
```

```
# building the matrix
trimask = np.triu(np.ones_like(corr, dtype = 'bool'))
# defining the filter to only show half and avoid repeating columns
plt.figure(figsize=(12,10))
sns.heatmap(corr, mask = trimask | (np.abs(corr) <= 0.70), annot = True)
# visualizing the correlation in seaborn and setting it to only show a
# correlation of over 70 percent
```

Out[151…

<AxesSubplot:>





Choosing which features to drop: sqft_living has an over 70% correlation with sqft living15. This makes sense, the closest 15 neighbors to a home having a correlating size of living space. sqft_living also has a correlation with sqft_above, which also makes sense, as these features are very similar. sqft_lot is closely correlated with sqft_lot15 for similar reasons. sqft_living also is interestingly correlated with bathrooms. Because this feature is so correlated to others, I'll be dropping it.

For only keeping one of those variables, Keeping sqft_15 makes sense because it's about the closest 15 neighbors, not about the property itself, which for our needs suits better since this metric is more closely related to the theory of mixed income housing. However, sqft_above is also too closely correlated with grade. And between sqft_above or grade, grade is the feature that we have most control of with renovation of the property. Therefore I'll be getting rid of sqft_above to keep grade as a feature.

yr_built, yr_renovated and yr_mod are all correlating with each other at an over 70% rate which makes sense. Of the three features, I'm keeping yr_renovated since it's most closely related to our interests.

Closest school is almost entirely predicted by closest police station and vice versa, so I'll be dropping closest police station as closest school has a more complex matrix of data associated with it. Downtown is shows multicollinearity with schools and police stations so I'll be dropping downtown also.

Additionally, I'll be dropping zipcode since I've made those into dummies.

In [152…

```
zipcode_data = zipcode_data.drop(columns = ['sqft_living', 'sqft_living15',
                                            'sqft_lot', 'zipcode', 'yr_built',
                                            'yr_mod','downtown', 'sqft_above',
                                            'closest_police_station'])
data = data.drop(columns = ['sqft_living', 'sqft_living15', 'sqft_lot',
                            'zipcode', 'yr_built', 'yr_mod', 'downtown',
                            'sqft_above', 'closest_police_station'])

# dropping these features from both data sets
```

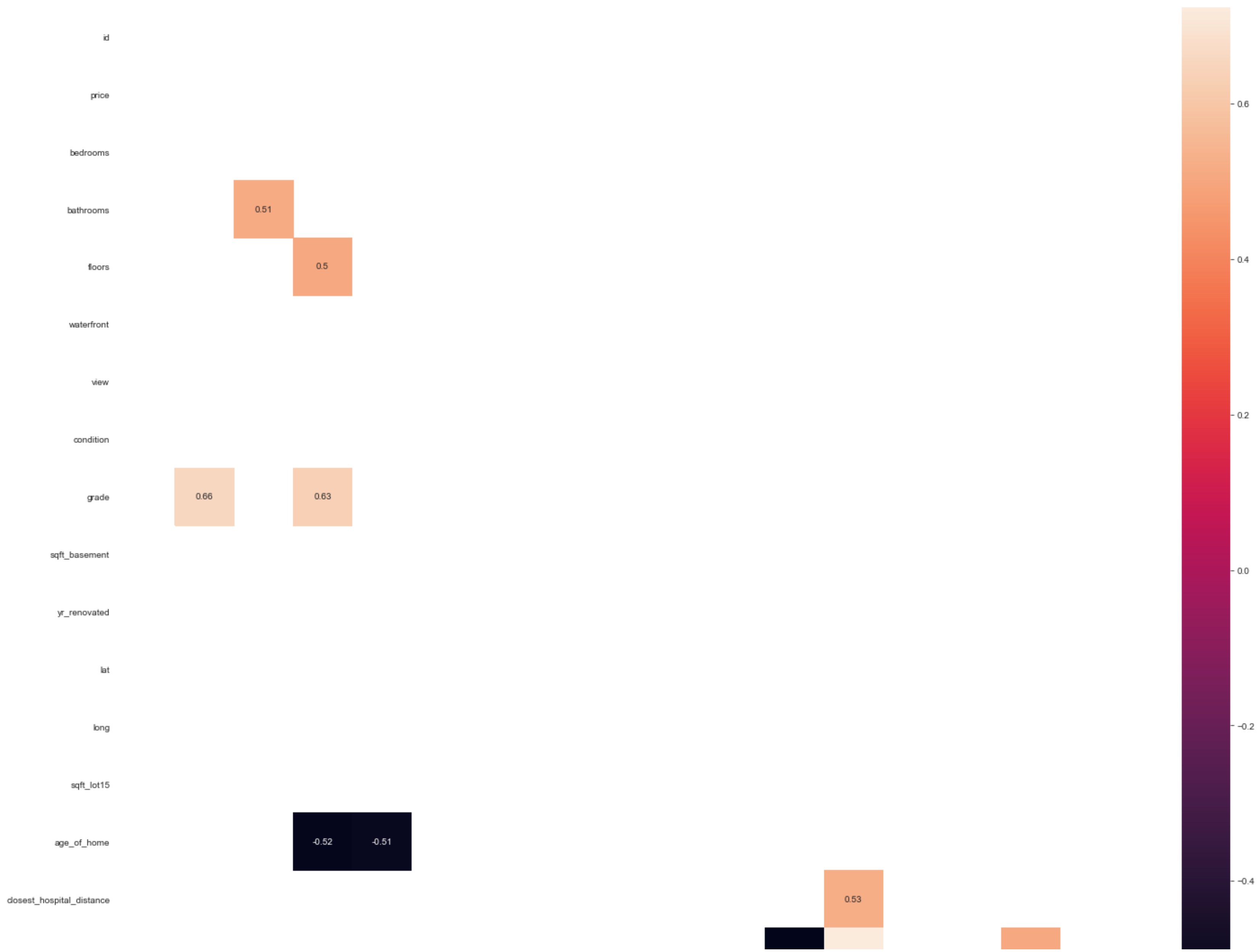
In [153…

```
corr = data.corr()
trimask = np.triu(np.ones_like(corr, dtype = 'bool'))
plt.figure(figsize=(25,20))
```

```
plt.figure(figsize=(25,20))
sns.heatmap(corr, mask = trimask | (np.abs(corr) <= 0.5), annot = True)

# taking a look to see the results and to make sure there's nothing getting
# close to 70 percent
```

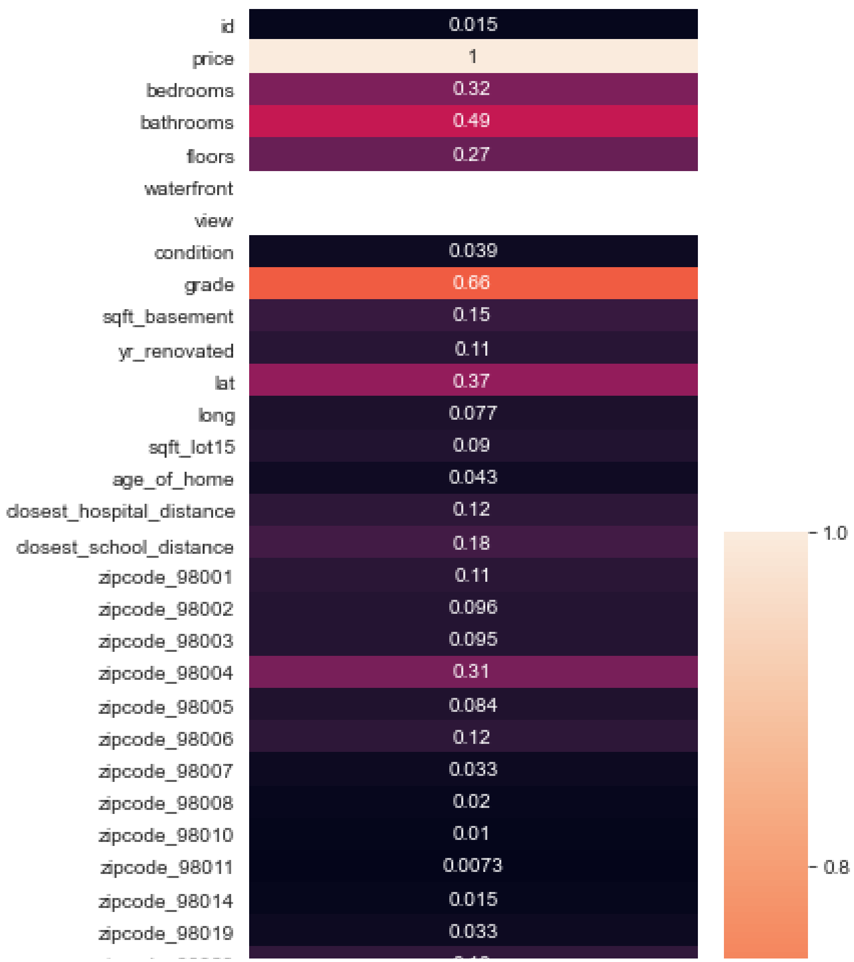
Out[153…

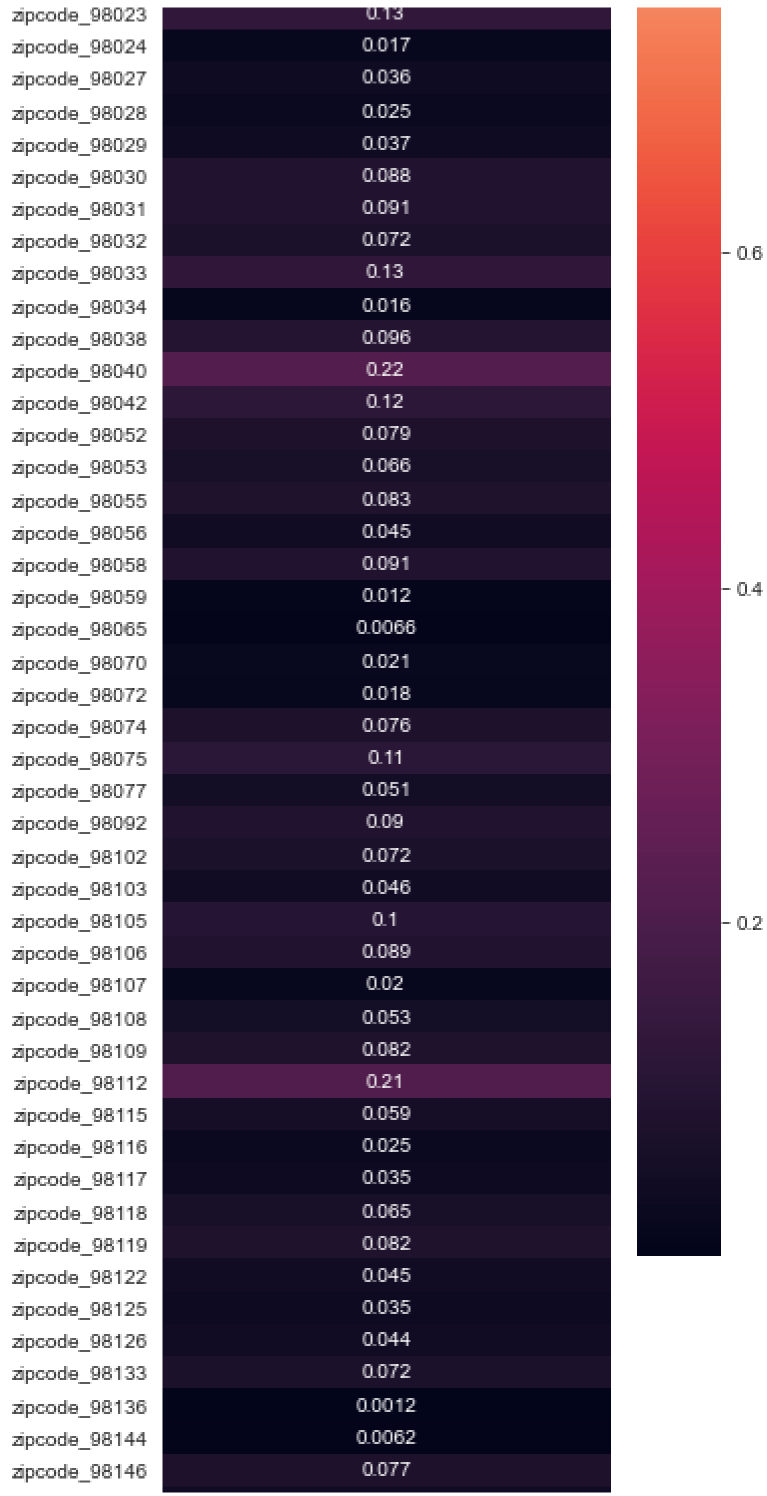
```
<AxesSubplot:>
```

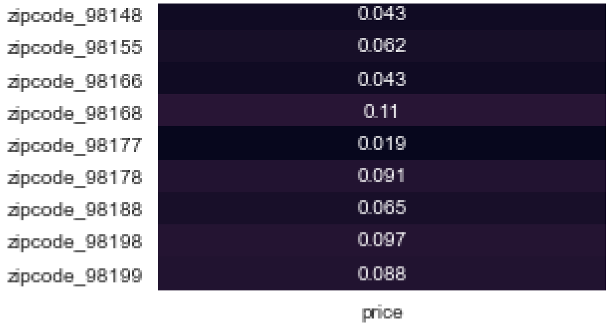


In [154…

```
sns.set_style('white')

plt.figure(figsize=(5, 25))
sns.heatmap(abs(zipcode_data.corr())[['price']], annot=True);

# now we can see the correlation with price and get a sense of how important
# zipcode data will be
```




zipcode_98023 0.13
zipcode_98024 0.017
zipcode_98027 0.036
zipcode_98028 0.025
zipcode_98029 0.037
zipcode_98030 0.088
zipcode_98031 0.091
zipcode_98032 0.072
zipcode_98033 0.13
zipcode_98034 0.016
zipcode_98038 0.096
zipcode_98040 0.22
zipcode_98042 0.12
zipcode_98052 0.079
zipcode_98053 0.066
zipcode_98055 0.083
zipcode_98056 0.045
zipcode_98058 0.091
zipcode_98059 0.012
zipcode_98065 0.0066
zipcode_98070 0.021
zipcode_98072 0.018
zipcode_98074 0.076
zipcode_98075 0.11
zipcode_98077 0.051
zipcode_98092 0.09
zipcode_98102 0.072
zipcode_98103 0.046
zipcode_98105 0.1
zipcode_98106 0.089
zipcode_98107 0.02
zipcode_98108 0.053
zipcode_98109 0.082
zipcode_98112 0.21
zipcode_98115 0.059
zipcode_98116 0.025
zipcode_98117 0.035
zipcode_98118 0.065
zipcode_98119 0.082
zipcode_98122 0.045
zipcode_98125 0.035
zipcode_98126 0.044
zipcode_98133 0.072
zipcode_98136 0.0012
zipcode_98144 0.0062
zipcode_98146 0.077
0.6
0.4
0.2



In [155…

```
order_and_values = np.abs(data.corr()['price']).sort_values(ascending = False)
# saving this series of features and correlations as an object
```

In [156…

```
order = list(order_and_values.index)
# saving only the index of that series
```

In [157…

```
order.remove('price')
order.remove('id')
# removing price and id from order
```

In [158…

```
features_to_remove = []
# making an empty list of features to remove
order = [feature for feature in order if feature not in features_to_remove]
# making sure the order doesn't include any of those features
```

Building a model

I'm going to build to helper functions here, one to tell me which features have P values above 0.05 and are therefore unable to reject the null hypothesis (these features aren't doing a good job of predicting.)

I'm also going to build a simple function to tell me the next three most correlated features in the order of features.

Further work:

Currently these two helper functions are sketched out but with more time and resources, I'd like to make the model building more automated. This will be beneficial for the civic workers tasked with using the model, and also higher quality code. For these preliminary models however, I wanted to keep a track of what features exactly I was including, which is why some of the code is repetitive.

Bugs:

Currently, the list features_to_remove never gets cleared of information. Ideally, it would be emptied before each iteration of the model. As things are right now, I don't end up needing to remove that many features, so this is working as well as it needs to for now. But with more time and resources, this functionality would be improved.

In [159…

```
def high_p_vals(model):
    p_vals = model.pvalues
    labels = list(model.pvalues.index)
    p_dict = dict(zip(labels, p_vals))
    for key, value in p_dict.items():
        if value > 0.05:
            features_to_remove.append(key)
        else:
            print(f'{key} has an acceptable p_value')
    return f'drop {features_to_remove} for the next model due to high p_value(s)'
```

In [160…

```
def next_features(order, features_to_remove):
    print (order[0: 3])
    del order[0: 3]
```

Baseline Model: Grade, Bathrooms, Lat

In [161…

```
next_features(order, features_to_remove)

# finding out what features to include
```

```
['grade', 'bathrooms', 'lat']
```

In [162…

```
df_baseline = zipcode_data.loc[:, ['price', 'grade', 'bathrooms', 'lat']]
df_baseline.info()

# making a new DF with those features
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19414 entries, 0 to 21596
```

```
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   price      19414 non-null  float64
 1   grade      19414 non-null  int64  
 2   bathrooms  19414 non-null  float64
 3   lat        19414 non-null  float64
dtypes: float64(3), int64(1)
memory usage: 758.4 KB
```

In [163…

```
y = zipcode_data['price']
X = df_baseline.drop(['price'], axis = 1)

# assigning my X and y variables
```

In [164…

```
baseline_model = sm.OLS(y, sm.add_constant(X)).fit()
baseline_model.summary()

# building the model and getting a summary
```

Out[164…

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.544
Model:	OLS	**Adj. R-squared:**	0.544
Method:	Least Squares	**F-statistic:**	7709.
Date:	Tue, 12 Apr 2022	**Prob (F-statistic):**	0.00
Time:	17:54:48	**Log-Likelihood:**	-2.6330e+05
No. Observations:	19414	**AIC:**	5.266e+05
Df Residuals:	19410	**BIC:**	5.266e+05
Df Model:	3		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-3.169e+07	4.77e+05	-66.366	0.000	-3.26e+07	-3.08e+07
grade	1.414e+05	1660.306	85.164	0.000	1.38e+05	1.45e+05
bathrooms	5.583e+04	2385.026	23.407	0.000	5.12e+04	6.05e+04

lat	6.607e+05	1e+04	65.753	0.000	6.41e+05	6.8e+05

Omnibus:	9369.097	**Durbin-Watson:**	1.958
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	88342.127
Skew:	2.108	**Prob(JB):**	0.00
Kurtosis:	12.562	**Cond. No.**	1.69e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.69e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

Observations: the model is currently predicting at above 50% accuracy, which is already a turning point, it's getting it right more than it's getting it wrong. The kurtosis however is way too high for only having three features, this must be fixed for the next model to not be broken.

In [165…

```
high_p_vals(baseline_model)
```

```
const has an acceptable p_value
grade has an acceptable p_value
bathrooms has an acceptable p_value
lat has an acceptable p_value
```

Out[165…

```
'drop [] for the next model due to high p_value(s)'
```

All the P values look good to continue.

In [166…

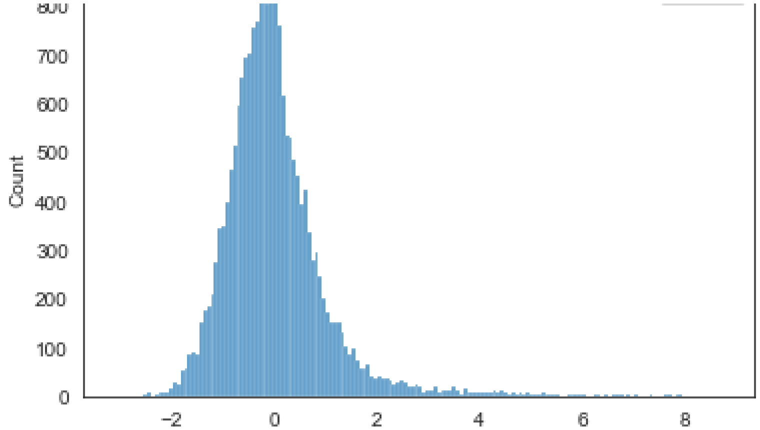
```
scaler = StandardScaler()
```

In [167…

```
scaled_resid_1 = scaler.fit_transform(baseline_model.resid.values.reshape(-1, 1))
sns.histplot(scaled_resid_1)
```

Out[167…

```
<AxesSubplot:ylabel='Count'>
```

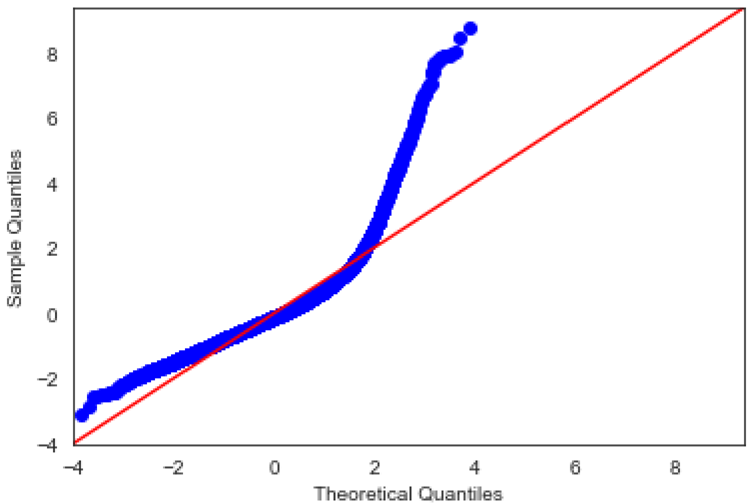


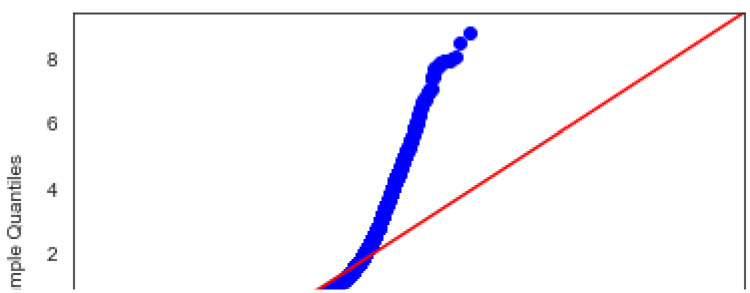
Observations: very long tail on the right side, showing where the residuals are more than 2 standard deviations away from the mean.

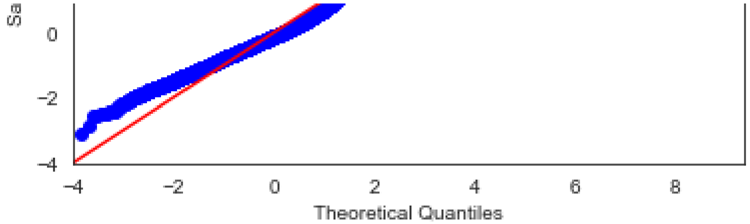
In [168…

```
sm.qqplot(baseline_model.resid, dist = stats.norm, line = "45", fit = True)
```

Out[168…







Observations: the blue line curves up sharply from the red line at about 2 standard deviations from the mean, representing the long tail we saw in the above plot.

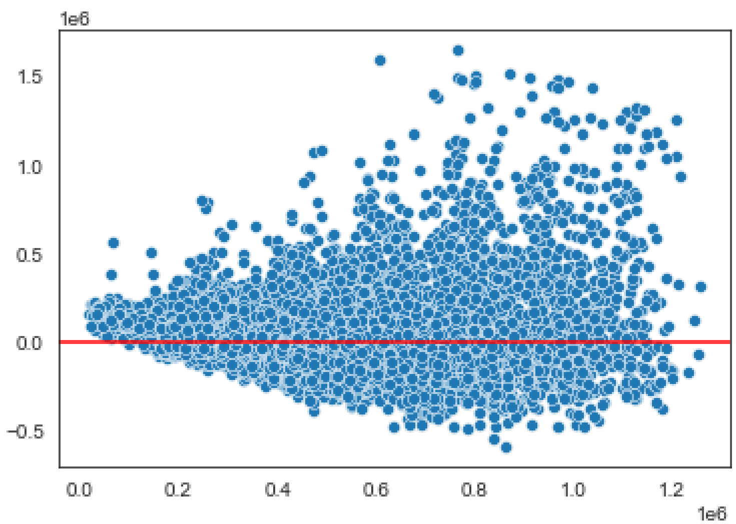
In [169…

```
sns.scatterplot(baseline_model.predict(sm.add_constant(X)), baseline_model.resid)
plt.axhline(0, color = 'red')
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[169…

```
<matplotlib.lines.Line2D at 0x7ff364ab0ac0>
```

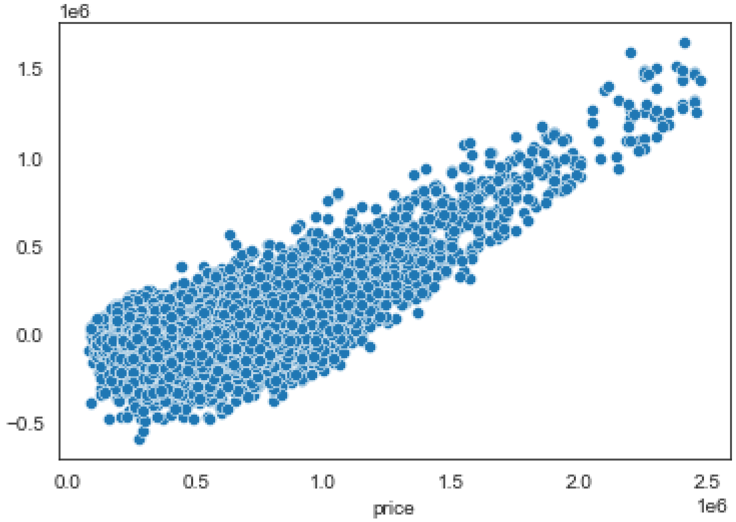


Observations: There is a cone shape on the left side of this plot, representing homoscedasticity. This is to say that there isn't homogeneity of variance, which we must fix for the next model.

In [170…

```
sns.scatterplot(y, baseline_model.resid);
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



Observations: This is a different way to see the homogeneity of the variance. This time, this visual doesn't tell us much.

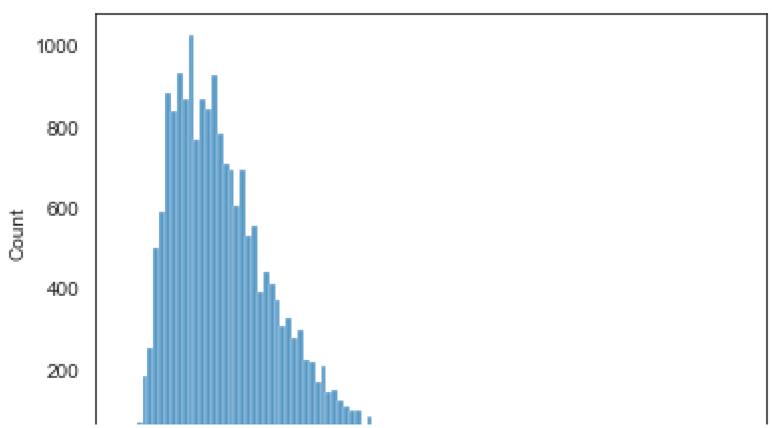
Second Model: Transforming the Price with Log

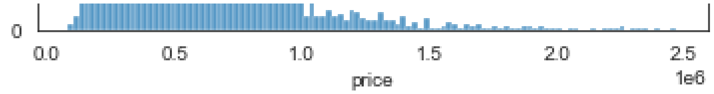
In [171…

```
sns.histplot(zipcode_data['price'])
```

Out[171…

```
<AxesSubplot:xlabel='price', ylabel='Count'>
```





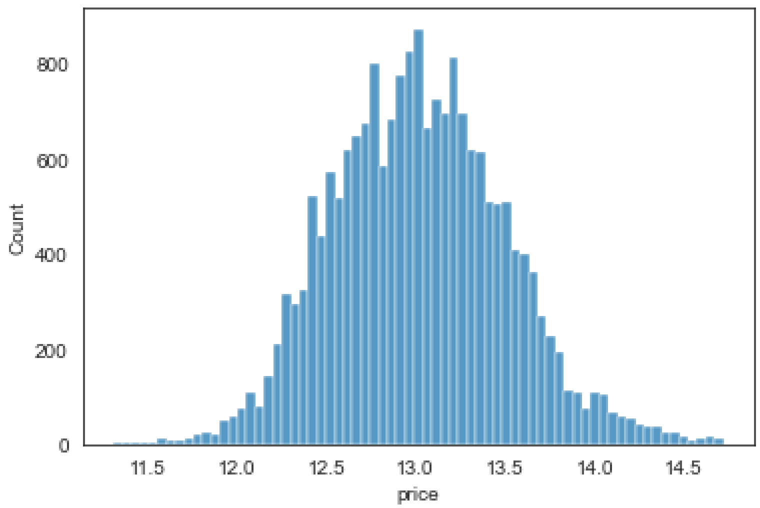
Price of home does not have a normal distribution in this data set. There are many more lower priced homes and a much longer tail on the right hand side, representing the lack of upward limit on home price.

In [172…

```
sns.histplot(np.log(zipcode_data['price']))
```

Out[172…

```
<AxesSubplot:xlabel='price', ylabel='Count'>
```



The log of the price however, does have a much more normal distribution. I'll be using the log of the price for all future models.

NOTE: I'll now be interpreting the price by percentage instead of by dollar amount.

In [173…

```
zipcode_data['price_log'] = np.log(zipcode_data['price'])

# making a new column in the data for the log of the price for each row
```

In [174…

```
price_log = zipcode_data['price_log']
```

In [175…

```
features_2 = zipcode_data.loc[:,['price_log', 'grade', 'bathrooms', 'lat']]
features_2.info()
```

```
# making the dataframe for the second model
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19414 entries, 0 to 21596
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   price_log  19414 non-null  float64
 1   grade      19414 non-null  int64
 2   bathrooms  19414 non-null  float64
 3   lat        19414 non-null  float64
dtypes: float64(3), int64(1)
memory usage: 758.4 KB
```

In [176…

```
y = features_2['price_log']
X = features_2.drop(['price_log'], axis = 1)

# labeling the axes
```

In [177…

```
model_2 = sm.OLS(y, sm.add_constant(X)).fit()
model_2.summary()

# making the model
```

Out[177…

OLS Regression Results

Dep. Variable:	price_log	**R-squared:**	0.631
Model:	OLS	**Adj. R-squared:**	0.630
Method:	Least Squares	**F-statistic:**	1.104e+04
Date:	Tue, 12 Apr 2022	**Prob (F-statistic):**	0.00
Time:	17:54:49	**Log-Likelihood:**	-3753.2
No. Observations:	19414	**AIC:**	7514.
Df Residuals:	19410	**BIC:**	7546.
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]

const	-61.9788	0.746	-83.051	0.000	-63.442	-60.516
grade	0.2313	0.003	89.137	0.000	0.226	0.236
bathrooms	0.1188	0.004	31.870	0.000	0.111	0.126
lat	1.5491	0.016	98.632	0.000	1.518	1.580

Omnibus:	441.832	**Durbin-Watson:**	1.973
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	590.661
Skew:	0.279	**Prob(JB):**	5.49e-129
Kurtosis:	3.647	**Cond. No.**	1.69e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.69e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

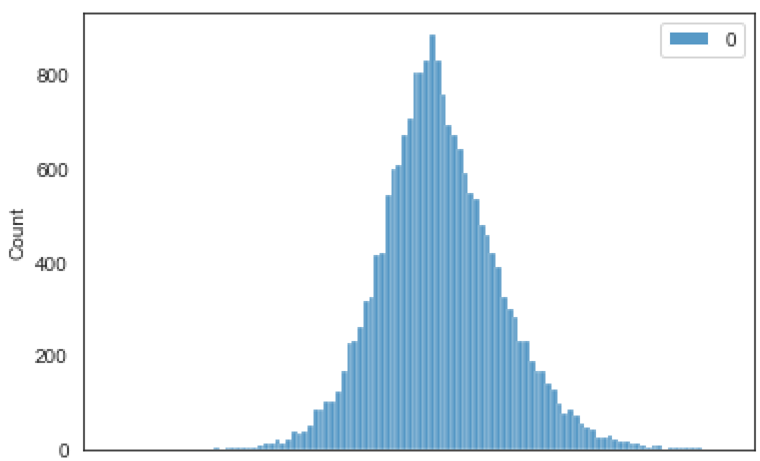
Observations: the kurtosis is much lower now, and we've improved the accuracy of the model by about 10%.

In [178…

```
scaled_resid_2 = scaler.fit_transform(model_2.resid.values.reshape(-1, 1))
sns.histplot(scaled_resid_2)
```

Out[178…

```
<AxesSubplot:ylabel='Count'>
```

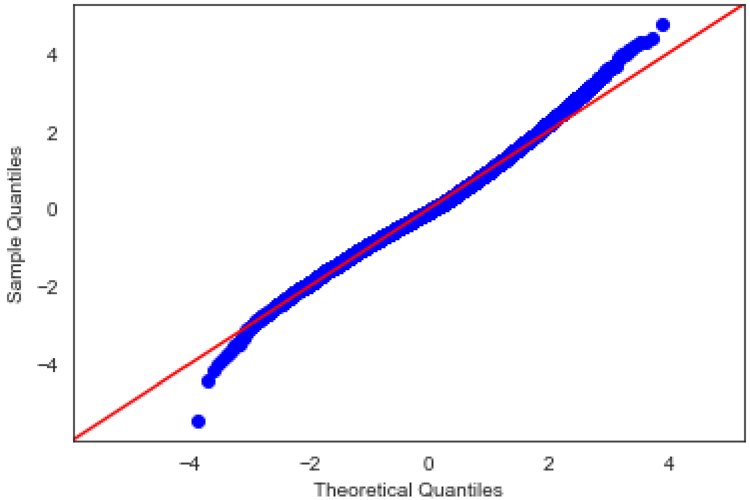


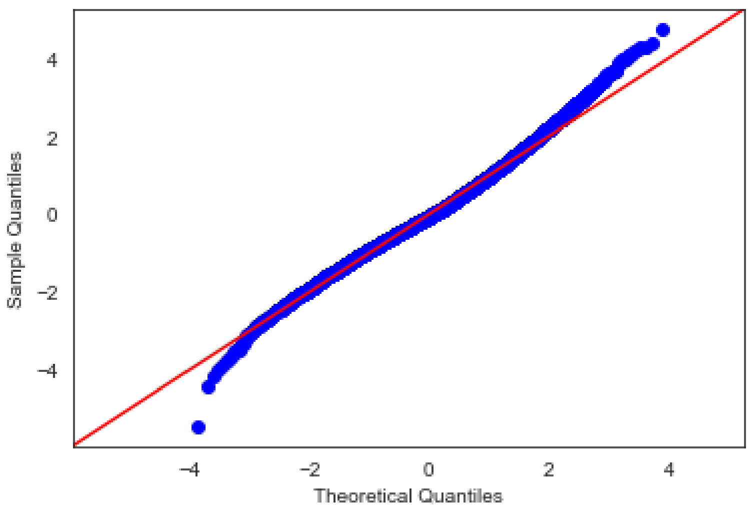
Observations: these residuals fit a normal distribution much better.

In [179…

```
sm.qqplot(model_2.resid, dist = stats.norm, line = "45", fit = True)
```

Out[179…





The blue line is much closer to the red line, representing the more homogeneity of the variance.

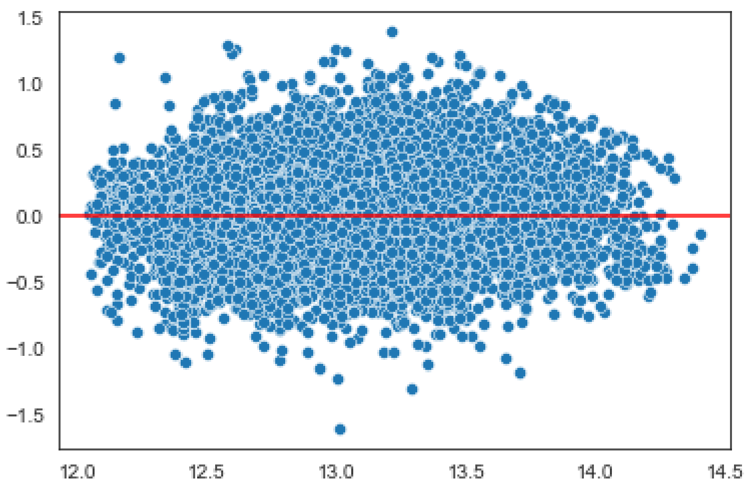
In [180…

```
sns.scatterplot(model_2.predict(sm.add_constant(X)), model_2.resid)
plt.axhline(0, color = 'red')
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[180…

```
<matplotlib.lines.Line2D at 0x7ff364b2ab20>
```

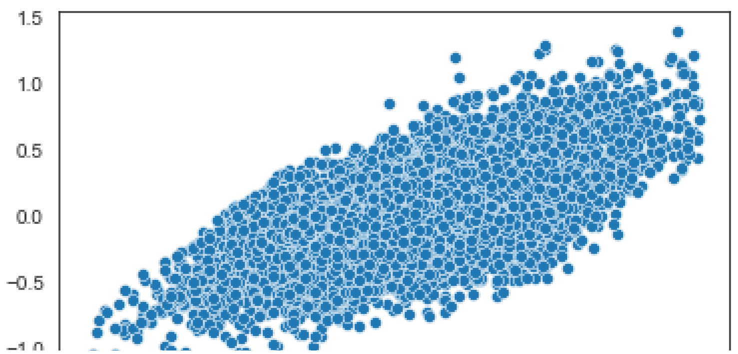


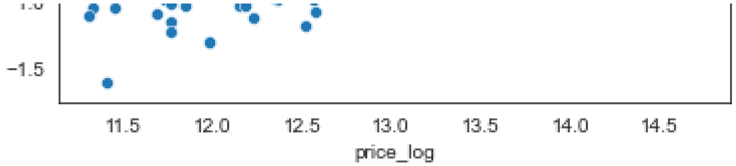
Observations: the cone shape is gone now on the left side, representing that there isn't a pattern that the model isn't picking up.

In [181…

```python
sns.scatterplot(y, model_2.resid);
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```





Observations: the points are now a bit more scattered, which is what we are looking for.

Improving the Model with Adding Features

Third Model: Bedrooms, Floors, Closest School Distance

In [182...
```
next_features(order, features_to_remove)
```

```
['bedrooms', 'floors', 'closest_school_distance']
```

In [183...
```
features_3 = ['bedrooms', 'floors', 'closest_school_distance']
```

In [184...
```
X = pd.concat([X, zipcode_data[features_3]], axis = 1)
```

In [185...
```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19414 entries, 0 to 21596
Data columns (total 6 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   grade                    19414 non-null  int64  
 1   bathrooms                19414 non-null  float64
 2   lat                      19414 non-null  float64
 3   bedrooms                 19414 non-null  int64  
 4   floors                   19414 non-null  float64
 5   closest_school_distance  19414 non-null  float64
dtypes: float64(4), int64(2)
memory usage: 1.0 MB
```

In [186...
```
model_3 = sm.OLS(y, sm.add_constant(X)).fit()
model_3.summary()
```

Out[186…

OLS Regression Results

Dep. Variable:	price_log	**R-squared:**	0.653
Model:	OLS	**Adj. R-squared:**	0.653
Method:	Least Squares	**F-statistic:**	6079.
Date:	Tue, 12 Apr 2022	**Prob (F-statistic):**	0.00
Time:	17:54:50	**Log-Likelihood:**	-3152.5
No. Observations:	19414	**AIC:**	6319.
Df Residuals:	19407	**BIC:**	6374.
Df Model:	6		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	-50.0830	0.872	-57.451	0.000	-51.792	-48.374
grade	0.2437	0.003	93.661	0.000	0.239	0.249
bathrooms	0.1078	0.004	25.545	0.000	0.100	0.116
lat	1.2971	0.018	70.756	0.000	1.261	1.333
bedrooms	0.0554	0.003	20.328	0.000	0.050	0.061
floors	-0.0427	0.005	-9.442	0.000	-0.052	-0.034
closest_school_distance	-0.0127	0.000	-26.354	0.000	-0.014	-0.012

Omnibus:	292.743	**Durbin-Watson:**	1.979
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	433.905
Skew:	0.172	**Prob(JB):**	6.01e-95
Kurtosis:	3.647	**Cond. No.**	2.06e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.06e+04. This might indicate that there are

[2] The condition number is large, 2.06e+04. This might indicate that there are strong multicollinearity or other numerical problems.

Observations: adding these three features didn't make the accuracy jump by more than 3%.

FURTHER WORK:

I have not looked too deeply into floors as I have with some other variables. In the model, it's predicting that as floors decrease, the price rises. This isn't automatically intuitive, and bears further investigation.

In [187…

```
high_p_vals(model_3)
```

```
const has an acceptable p_value
grade has an acceptable p_value
bathrooms has an acceptable p_value
lat has an acceptable p_value
bedrooms has an acceptable p_value
floors has an acceptable p_value
closest_school_distance has an acceptable p_value
```

Out[187…

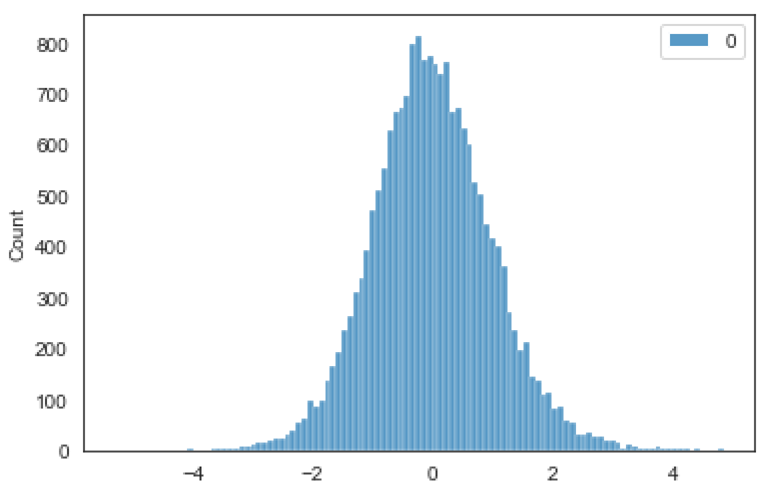
```
'drop [] for the next model due to high p_value(s)'
```

In [188…

```
scaled_resid_3 = scaler.fit_transform(model_3.resid.values.reshape(-1, 1))
sns.histplot(scaled_resid_3)
```

Out[188…

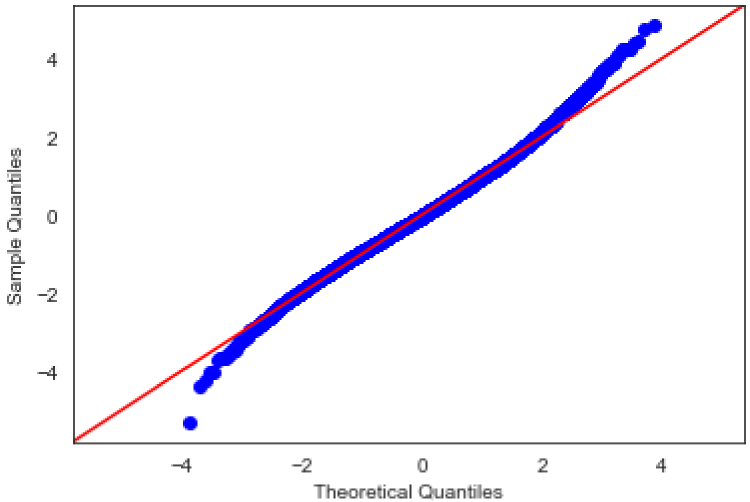
```
<AxesSubplot:ylabel='Count'>
```

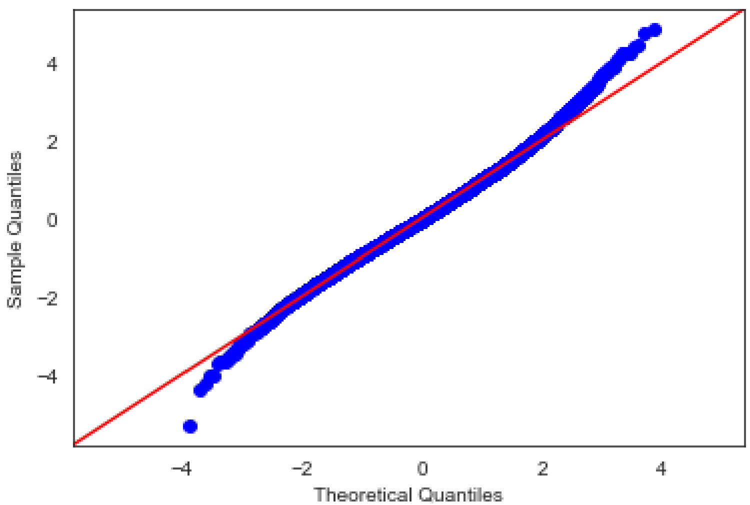


In [189…

```
sm.qqplot(model_3.resid, dist = stats.norm, line = "45", fit = True)
```

Out[189…





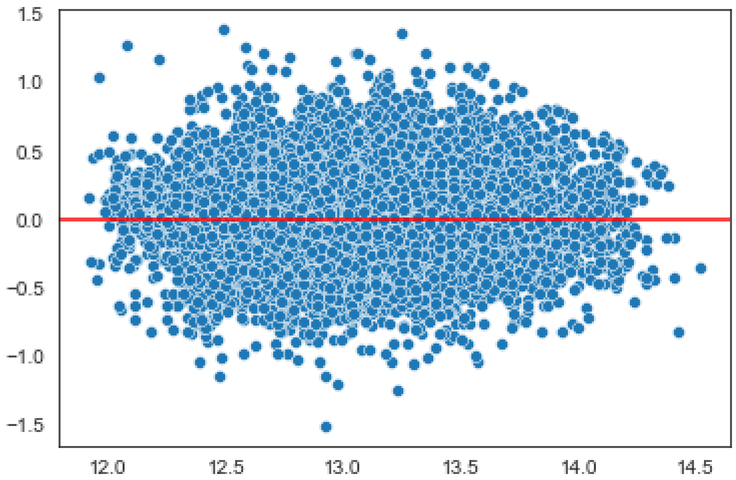
In [190…

```
sns.scatterplot(model_3.predict(sm.add_constant(X)), model_3.resid)
plt.axhline(0, color = 'red')
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[190…

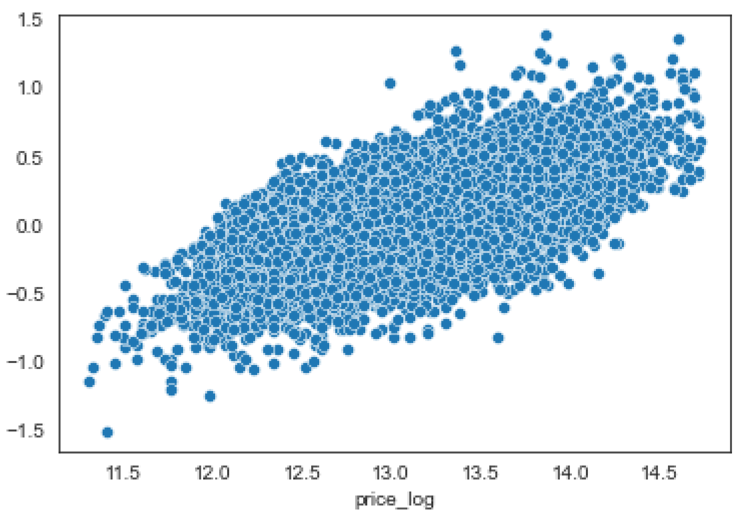
```
<matplotlib.lines.Line2D at 0x7ff36599f490>
```



In [191…

```
sns.scatterplot(y, model_3.resid);
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



Observations: these plots show that the model is functioning properly, represented by the lack of cone shapes in the scatterplots and the blue line being close to the red line on the QQ plot.

Fourth Model: Square Foot Basement, Closest Hospital Distance and Year Renovated

In [192…

```
next_features(order, features_to_remove)
```

```
['sqft_basement', 'closest_hospital_distance', 'yr_renovated']
```

In [193…

```
features_4 = ['sqft_basement', 'closest_hospital_distance', 'yr_renovated']
```

In [194…

```
X = pd.concat([X, zipcode_data[features_4]], axis = 1)
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19414 entries, 0 to 21596
Data columns (total 9 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   grade                      19414 non-null  int64  
 1   bathrooms                  19414 non-null  float64
 2   lat                        19414 non-null  float64
 3   bedrooms                   19414 non-null  int64  
 4   floors                     19414 non-null  float64
 5   closest_school_distance    19414 non-null  float64
 6   sqft_basement              19414 non-null  float64
 7   closest_hospital_distance  19414 non-null  float64
 8   yr_renovated               19414 non-null  float64
dtypes: float64(7), int64(2)
memory usage: 1.5 MB
```

In [195…

```
model_4 = sm.OLS(y, sm.add_constant(X)).fit()
model_4.summary()
```

Out[195… OLS Regression Results

Dep. Variable:	price_log	**R-squared:**	0.661
Model:	OLS	**Adj. R-squared:**	0.661
Method:	Least Squares	**F-statistic:**	4213.
Date:	Tue, 12 Apr 2022	**Prob (F-statistic):**	0.00
Time:	17:54:50	**Log-Likelihood:**	-2903.6

No. Observations:	19414	**AIC:**	5827.
Df Residuals:	19404	**BIC:**	5906.
Df Model:	9		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-51.2004	0.863	-59.347	0.000	-52.891	-49.509
grade	0.2446	0.003	95.185	0.000	0.240	0.250
bathrooms	0.0897	0.004	20.478	0.000	0.081	0.098
lat	1.3204	0.018	72.772	0.000	1.285	1.356
bedrooms	0.0536	0.003	19.868	0.000	0.048	0.059
floors	-0.0196	0.005	-3.997	0.000	-0.029	-0.010
closest_school_distance	-0.0076	0.001	-13.757	0.000	-0.009	-0.007
sqft_basement	0.0569	0.005	11.682	0.000	0.047	0.066
closest_hospital_distance	-0.0122	0.001	-11.112	0.000	-0.014	-0.010
yr_renovated	8.819e-05	5.92e-06	14.901	0.000	7.66e-05	9.98e-05

Omnibus:	285.627	**Durbin-Watson:**	1.978
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	427.533
Skew:	0.164	**Prob(JB):**	1.45e-93
Kurtosis:	3.648	**Cond. No.**	1.49e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.49e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

In [196…

```
high_p_vals(model_4)
```

```
const has an acceptable p_value
grade has an acceptable p_value
bathrooms has an acceptable p_value
lat has an acceptable p_value
bedrooms has an acceptable p_value
floors has an acceptable p_value
closest_school_distance has an acceptable p_value
sqft_basement has an acceptable p_value
closest_hospital_distance has an acceptable p_value
yr_renovated has an acceptable p_value
```

Out[196…

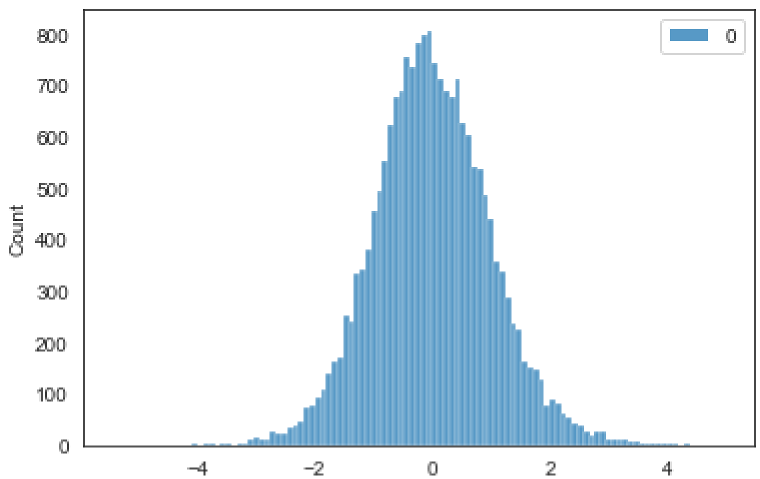
```
'drop [] for the next model due to high p_value(s)'
```

In [197…

```
scaled_resid_4 = scaler.fit_transform(model_4.resid.values.reshape(-1, 1))
sns.histplot(scaled_resid_4)
```

Out[197…

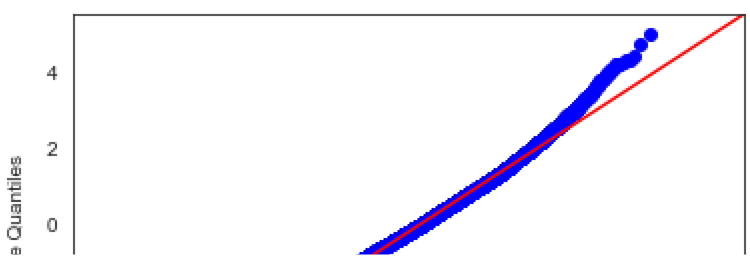
```
<AxesSubplot:ylabel='Count'>
```

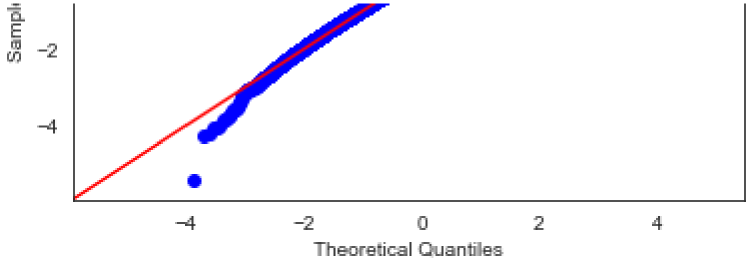


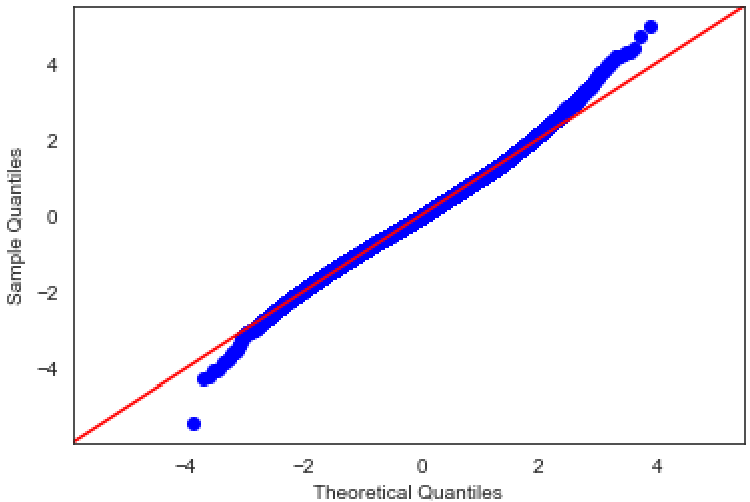
In [198…

```
sm.qqplot(model_4.resid, dist = stats.norm, line = "45", fit = True)
```

Out[198…







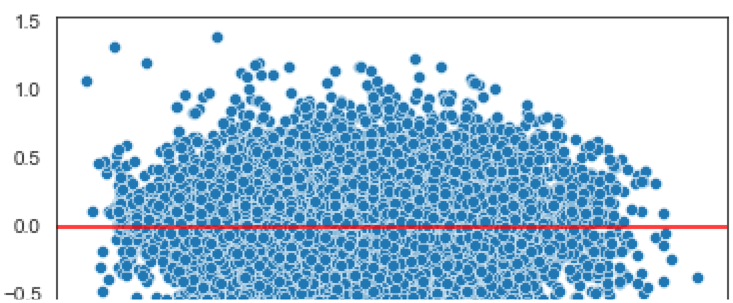
In [199…

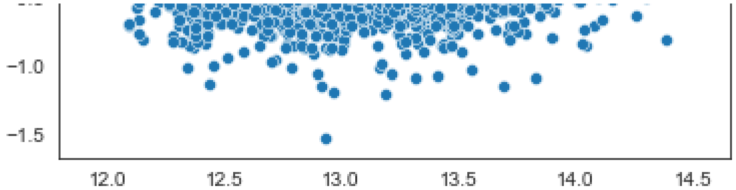
```
sns.scatterplot(model_4.predict(sm.add_constant(X)), model_4.resid)
plt.axhline(0, color = 'red')
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[199…

```
<matplotlib.lines.Line2D at 0x7ff365951f10>
```



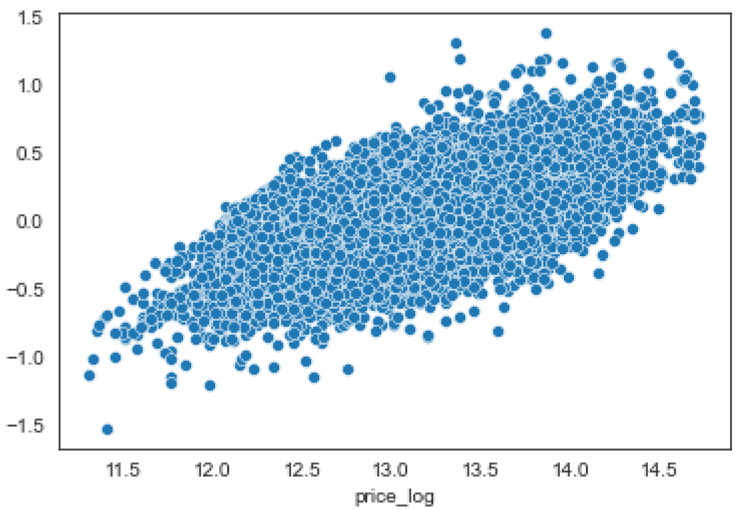


Observations: now there is a slight cone shape on the right side. It isn't dramatic so I will not do anything to fix it for the next model.

In [200…

```
sns.scatterplot(y, model_4.resid);
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



Observations:

Fifth Model: Square Foot Lot of Closest 15 Neighbors, Longitude and Age of Home

In [201…

```
next_features(order, features_to_remove)
```

```
['sqft_lot15', 'long', 'age_of_home']
```

In [202…

```
features_5 = ['sqft_lot15', 'long', 'age_of_home']
```

In [203…

```
X = pd.concat([X, zipcode_data[features_5]], axis = 1)
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19414 entries, 0 to 21596
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   grade                      19414 non-null  int64  
 1   bathrooms                  19414 non-null  float64
 2   lat                        19414 non-null  float64
 3   bedrooms                   19414 non-null  int64  
 4   floors                     19414 non-null  float64
 5   closest_school_distance    19414 non-null  float64
 6   sqft_basement              19414 non-null  float64
 7   closest_hospital_distance  19414 non-null  float64
 8   yr_renovated               19414 non-null  float64
 9   sqft_lot15                 19414 non-null  int64  
 10  long                       19414 non-null  float64
 11  age_of_home                19414 non-null  int64  
dtypes: float64(8), int64(4)
memory usage: 1.9 MB
```

In [204…

```
model_5 = sm.OLS(y, sm.add_constant(X)).fit()
model_5.summary()
```

Out[204… OLS Regression Results

Dep. Variable:	price_log	**R-squared:**	0.706
Model:	OLS	**Adj. R-squared:**	0.706
Method:	Least Squares	**F-statistic:**	3878.
Date:	Tue, 12 Apr 2022	**Prob (F-statistic):**	0.00
Time:	17:54:51	**Log-Likelihood:**	-1542.8
No. Observations:	19414	**AIC:**	3112.
Df Residuals:	19401	**BIC:**	3214.
Df Model:	12		

Df Model:	12
Covariance Type:	nonrobust

	coef	std err	t	P>|t|	[0.025	0.975]
const	46.0883	3.775	12.208	0.000	38.689	53.488
grade	0.2543	0.002	103.909	0.000	0.250	0.259
bathrooms	0.1398	0.004	32.812	0.000	0.131	0.148
lat	1.0569	0.019	54.354	0.000	1.019	1.095
bedrooms	0.0291	0.003	11.393	0.000	0.024	0.034
floors	0.0551	0.005	11.450	0.000	0.046	0.065
closest_school_distance	-0.0140	0.001	-18.848	0.000	-0.015	-0.013
sqft_basement	0.0625	0.005	13.702	0.000	0.054	0.071
closest_hospital_distance	-0.0150	0.001	-13.945	0.000	-0.017	-0.013
yr_renovated	2.387e-05	5.69e-06	4.197	0.000	1.27e-05	3.5e-05
sqft_lot15	1.068e-06	1.66e-07	6.441	0.000	7.43e-07	1.39e-06
long	0.6962	0.026	26.315	0.000	0.644	0.748
age_of_home	0.0045	9.51e-05	47.781	0.000	0.004	0.005

Omnibus:	363.810	**Durbin-Watson:**	1.991
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	726.252
Skew:	0.083	**Prob(JB):**	1.98e-158
Kurtosis:	3.933	**Cond. No.**	3.11e+07

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.11e+07. This might indicate that there are
strong multicollinearity or other numerical problems.

In [205…

```
high_p_vals(model_5)
```

```
const has an acceptable p_value
grade has an acceptable p_value
bathrooms has an acceptable p_value
lat has an acceptable p_value
bedrooms has an acceptable p_value
floors has an acceptable p_value
closest_school_distance has an acceptable p_value
sqft_basement has an acceptable p_value
closest_hospital_distance has an acceptable p_value
yr_renovated has an acceptable p_value
sqft_lot15 has an acceptable p_value
long has an acceptable p_value
age_of_home has an acceptable p_value
```

Out[205…

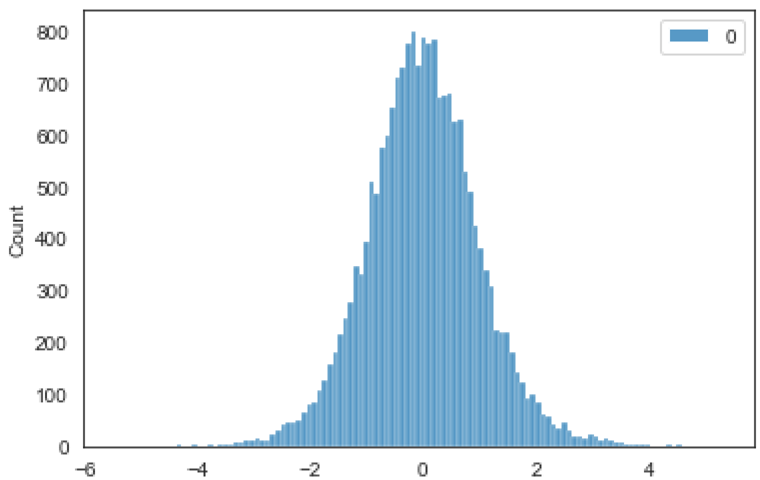
```
'drop [] for the next model due to high p_value(s)'
```

In [206…

```
scaled_resid_5 = scaler.fit_transform(model_5.resid.values.reshape(-1, 1))
sns.histplot(scaled_resid_5)
```

Out[206…

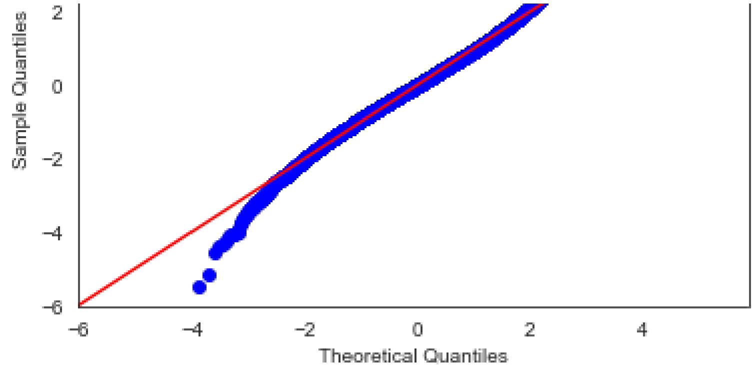
```
&lt;AxesSubplot:ylabel='Count'&gt;
```

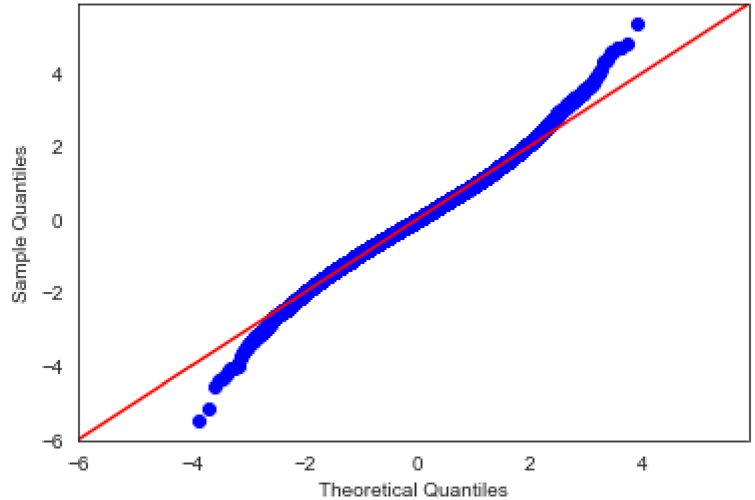


In [207…

```
sm.qqplot(model_5.resid, dist = stats.norm, line = "45", fit = True)
```

Out[207…





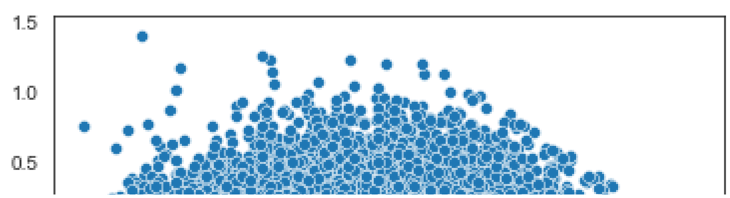
In [208…

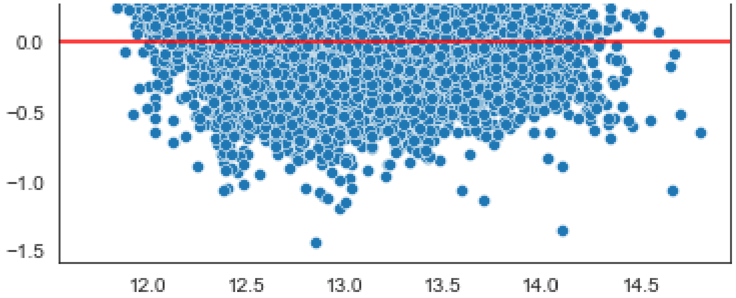
```
sns.scatterplot(model_5.predict(sm.add_constant(X)), model_5.resid)
plt.axhline(0, color = 'red')
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/_decorators.py:36: Futu
reWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid position
al argument will be `data`, and passing other arguments without an explicit keyword will result in an err
or or misinterpretation.
  warnings.warn(
```

Out[208…

```
<matplotlib.lines.Line2D at 0x7ff364b3c6a0>
```





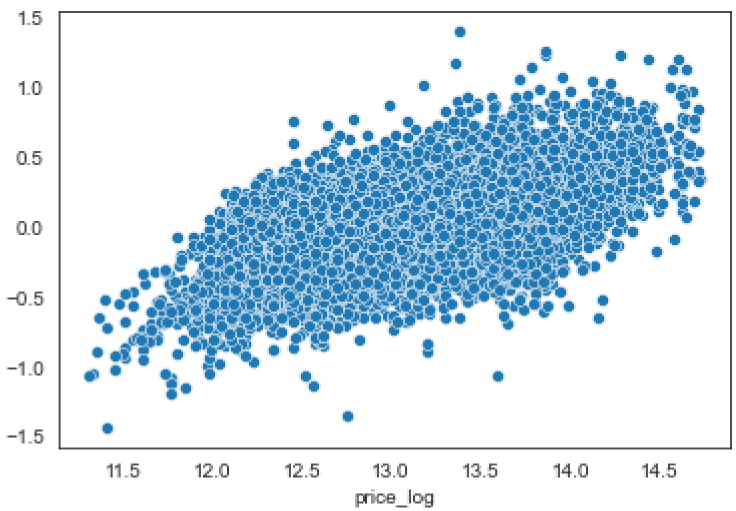
In [209…

```
sns.scatterplot(y, model_5.resid)
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/_decorators.py:36: Futu
reWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid position
al argument will be `data`, and passing other arguments without an explicit keyword will result in an err
or or misinterpretation.
  warnings.warn(
```

Out[209…

```
<AxesSubplot:xlabel='price_log'>
```



Sixth Model: Condition, Waterfront and View

In [210…

```
next_features(order, features_to_remove)
```

```
['condition', 'waterfront', 'view']
```

In [211…

```python
features_6 = ['condition', 'waterfront', 'view']
```

In [212…

```python
X = pd.concat([X, zipcode_data[features_6]], axis = 1)
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19414 entries, 0 to 21596
Data columns (total 15 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   grade                      19414 non-null  int64  
 1   bathrooms                  19414 non-null  float64
 2   lat                        19414 non-null  float64
 3   bedrooms                   19414 non-null  int64  
 4   floors                     19414 non-null  float64
 5   closest_school_distance    19414 non-null  float64
 6   sqft_basement              19414 non-null  float64
 7   closest_hospital_distance  19414 non-null  float64
 8   yr_renovated               19414 non-null  float64
 9   sqft_lot15                 19414 non-null  int64  
 10  long                       19414 non-null  float64
 11  age_of_home                19414 non-null  int64  
 12  condition                  19414 non-null  int64  
 13  waterfront                 19414 non-null  int64  
 14  view                       19414 non-null  float64
dtypes: float64(9), int64(6)
memory usage: 2.4 MB
```

In [213…

```python
model_6 = sm.OLS(y, sm.add_constant(X)).fit()
model_6.summary()
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/statsmodels/regression/linear_m
odel.py:1860: RuntimeWarning: divide by zero encountered in double_scalars
  return np.sqrt(eigvals[0]/eigvals[-1])
```

Out[213…

OLS Regression Results

Dep. Variable:	price_log	**R-squared:**	0.711
Model:	OLS	**Adj. R-squared:**	0.711
Method:	Least Squares	**F-statistic:**	3675.
Date:	Tue, 12 Apr 2022	**Prob (F-statistic):**	0.00
Time:	17:54:51	**Log-Likelihood:**	-1362.2

No. Observations:	19414	**AIC:**	2752.
Df Residuals:	19400	**BIC:**	2863.
Df Model:	13		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	43.4383	3.743	11.606	0.000	36.102	50.775
grade	0.2550	0.002	105.124	0.000	0.250	0.260
bathrooms	0.1329	0.004	31.358	0.000	0.125	0.141
lat	1.0735	0.019	55.665	0.000	1.036	1.111
bedrooms	0.0275	0.003	10.853	0.000	0.023	0.032
floors	0.0643	0.005	13.422	0.000	0.055	0.074
closest_school_distance	-0.0139	0.001	-18.936	0.000	-0.015	-0.012
sqft_basement	0.0605	0.005	13.389	0.000	0.052	0.069
closest_hospital_distance	-0.0151	0.001	-14.177	0.000	-0.017	-0.013
yr_renovated	4.025e-05	5.7e-06	7.062	0.000	2.91e-05	5.14e-05
sqft_lot15	9.948e-07	1.64e-07	6.057	0.000	6.73e-07	1.32e-06
long	0.6819	0.026	26.007	0.000	0.631	0.733
age_of_home	0.0040	9.89e-05	40.225	0.000	0.004	0.004
condition	0.0613	0.003	19.090	0.000	0.055	0.068
waterfront	0	0	nan	nan	0	0
view	0	0	nan	nan	0	0

Omnibus:	365.540	**Durbin-Watson:**	1.992
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	681.974
Skew:	0.124	**Prob(JB):**	8.15e-149
Kurtosis:	3.884	**Cond. No.**	inf

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 0. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [214…

```
high_p_vals(model_6)
```

```
const has an acceptable p_value
grade has an acceptable p_value
bathrooms has an acceptable p_value
lat has an acceptable p_value
bedrooms has an acceptable p_value
floors has an acceptable p_value
closest_school_distance has an acceptable p_value
sqft_basement has an acceptable p_value
closest_hospital_distance has an acceptable p_value
yr_renovated has an acceptable p_value
sqft_lot15 has an acceptable p_value
long has an acceptable p_value
age_of_home has an acceptable p_value
condition has an acceptable p_value
waterfront has an acceptable p_value
view has an acceptable p_value
```

Out[214…

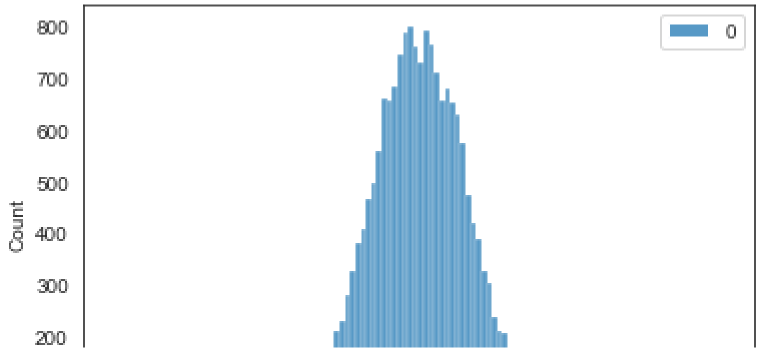
```
'drop [] for the next model due to high p_value(s)'
```

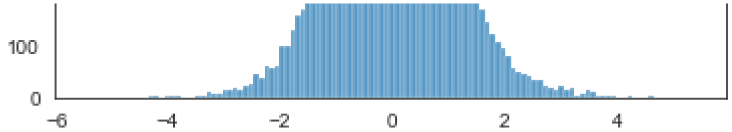
In [215…

```
scaled_resid_6 = scaler.fit_transform(model_6.resid.values.reshape(-1, 1))
sns.histplot(scaled_resid_6)
```

Out[215…

```
<AxesSubplot:ylabel='Count'>
```

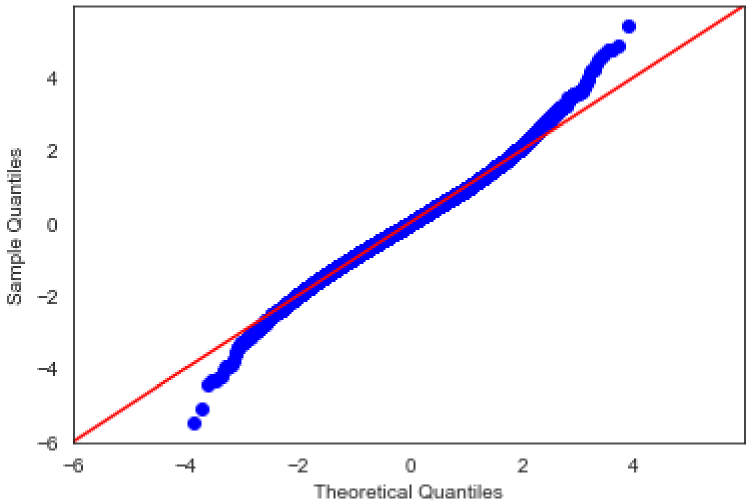


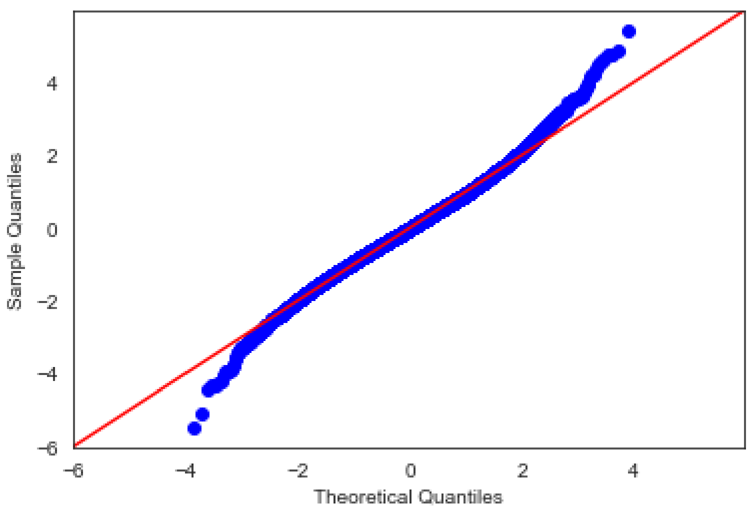


In [216…

```
sm.qqplot(model_6.resid, dist = stats.norm, line = "45", fit = True)
```

Out[216…





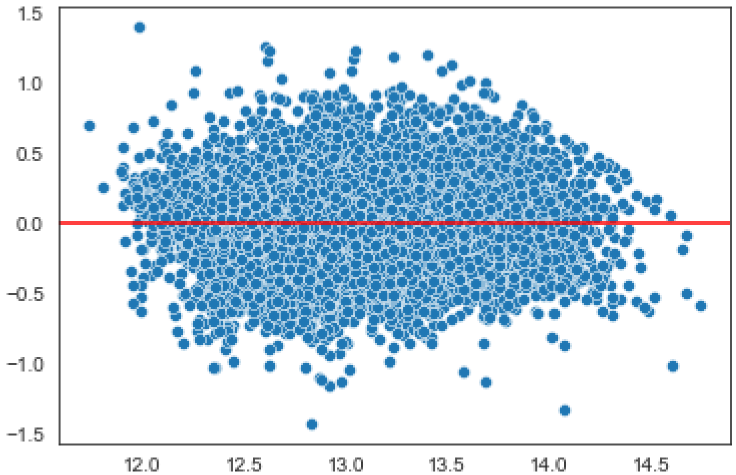
In [217…

```
sns.scatterplot(model_6.predict(sm.add_constant(X)), model_6.resid)
plt.axhline(0, color = 'red')
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[217…

```
<matplotlib.lines.Line2D at 0x7ff34422c0a0>
```



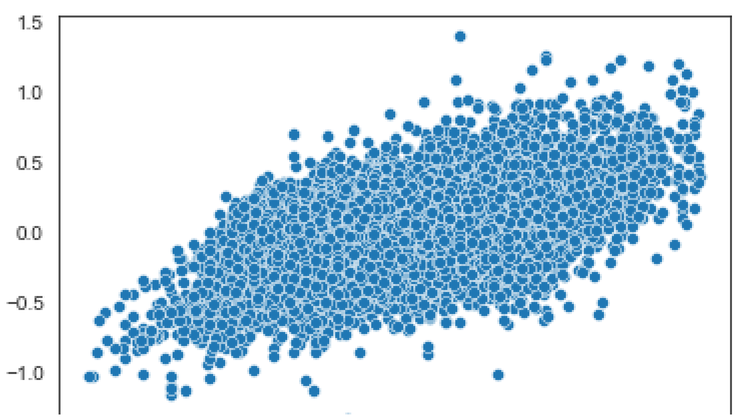
In [218…

```
sns.scatterplot(y, model_6.resid)
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[218…

```
<AxesSubplot:xlabel='price_log'>
```



Seventh Model: Adding Zipcodes

In [219…

```
zipcode_data.columns
```

Out[219…

```
Index(['id', 'date', 'price', 'bedrooms', 'bathrooms', 'floors', 'waterfront',
       'view', 'condition', 'grade', 'sqft_basement', 'yr_renovated', 'lat',
       'long', 'sqft_lot15', 'age_of_home', 'closest_hospital_distance',
       'closest_school_distance', 'zipcode_98001', 'zipcode_98002',
       'zipcode_98003', 'zipcode_98004', 'zipcode_98005', 'zipcode_98006',
       'zipcode_98007', 'zipcode_98008', 'zipcode_98010', 'zipcode_98011',
       'zipcode_98014', 'zipcode_98019', 'zipcode_98023', 'zipcode_98024',
       'zipcode_98027', 'zipcode_98028', 'zipcode_98029', 'zipcode_98030',
       'zipcode_98031', 'zipcode_98032', 'zipcode_98033', 'zipcode_98034',
       'zipcode_98038', 'zipcode_98040', 'zipcode_98042', 'zipcode_98052',
       'zipcode_98053', 'zipcode_98055', 'zipcode_98056', 'zipcode_98058',
       'zipcode_98059', 'zipcode_98065', 'zipcode_98070', 'zipcode_98072',
       'zipcode_98074', 'zipcode_98075', 'zipcode_98077', 'zipcode_98092',
       'zipcode_98102', 'zipcode_98103', 'zipcode_98105', 'zipcode_98106',
       'zipcode_98107', 'zipcode_98108', 'zipcode_98109', 'zipcode_98112',
       'zipcode_98115', 'zipcode_98116', 'zipcode_98117', 'zipcode_98118',
       'zipcode_98119', 'zipcode_98122', 'zipcode_98125', 'zipcode_98126',
       'zipcode_98133', 'zipcode_98136', 'zipcode_98144', 'zipcode_98146',
       'zipcode_98148', 'zipcode_98155', 'zipcode_98166', 'zipcode_98168',
       'zipcode_98177', 'zipcode_98178', 'zipcode_98188', 'zipcode_98198',
       'zipcode_98199', 'price_log'],
      dtype='object')
```

In [220…

```
features_7 = ['zipcode_98001',
       'zipcode_98002', 'zipcode_98003', 'zipcode_98004', 'zipcode_98005',
       'zipcode_98006', 'zipcode_98007', 'zipcode_98008', 'zipcode_98010',
       'zipcode_98011', 'zipcode_98014', 'zipcode_98019', 'zipcode_98023',
       'zipcode_98024', 'zipcode_98027', 'zipcode_98028', 'zipcode_98029',
       'zipcode_98030', 'zipcode_98031', 'zipcode_98032', 'zipcode_98033',
       'zipcode_98034', 'zipcode_98038', 'zipcode_98040', 'zipcode_98042',
       'zipcode_98052', 'zipcode_98053', 'zipcode_98055', 'zipcode_98056',
       'zipcode_98058', 'zipcode_98059', 'zipcode_98065', 'zipcode_98070',
       'zipcode_98072', 'zipcode_98074', 'zipcode_98075', 'zipcode_98077',
       'zipcode_98092', 'zipcode_98102', 'zipcode_98103', 'zipcode_98105',
       'zipcode_98106', 'zipcode_98107', 'zipcode_98108', 'zipcode_98109',
```

```
       'zipcode_98112', 'zipcode_98115', 'zipcode_98116', 'zipcode_98117',
       'zipcode_98118', 'zipcode_98119', 'zipcode_98122', 'zipcode_98125',
       'zipcode_98126', 'zipcode_98133', 'zipcode_98136', 'zipcode_98144',
       'zipcode_98146', 'zipcode_98148', 'zipcode_98155', 'zipcode_98166',
       'zipcode_98168', 'zipcode_98177', 'zipcode_98178', 'zipcode_98188',
       'zipcode_98198', 'zipcode_98199']
```

In [221…

```
X = pd.concat([X, zipcode_data[features_7]], axis = 1)
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19414 entries, 0 to 21596
Data columns (total 82 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   grade                      19414 non-null  int64  
 1   bathrooms                  19414 non-null  float64
 2   lat                        19414 non-null  float64
 3   bedrooms                   19414 non-null  int64  
 4   floors                     19414 non-null  float64
 5   closest_school_distance    19414 non-null  float64
 6   sqft_basement              19414 non-null  float64
 7   closest_hospital_distance  19414 non-null  float64
 8   yr_renovated               19414 non-null  float64
 9   sqft_lot15                 19414 non-null  int64  
 10  long                       19414 non-null  float64
 11  age_of_home                19414 non-null  int64  
 12  condition                  19414 non-null  int64  
 13  waterfront                 19414 non-null  int64  
 14  view                       19414 non-null  float64
 15  zipcode_98001              19414 non-null  uint8  
 16  zipcode_98002              19414 non-null  uint8  
 17  zipcode_98003              19414 non-null  uint8  
 18  zipcode_98004              19414 non-null  uint8  
 19  zipcode_98005              19414 non-null  uint8  
 20  zipcode_98006              19414 non-null  uint8  
 21  zipcode_98007              19414 non-null  uint8  
 22  zipcode_98008              19414 non-null  uint8  
 23  zipcode_98010              19414 non-null  uint8  
 24  zipcode_98011              19414 non-null  uint8  
 25  zipcode_98014              19414 non-null  uint8  
 26  zipcode_98019              19414 non-null  uint8  
 27  zipcode_98023              19414 non-null  uint8  
 28  zipcode_98024              19414 non-null  uint8  
```

```
 29  zipcode_98027              19414 non-null  uint8
 30  zipcode_98028              19414 non-null  uint8
 31  zipcode_98029              19414 non-null  uint8
 32  zipcode_98030              19414 non-null  uint8
 33  zipcode_98031              19414 non-null  uint8
 34  zipcode_98032              19414 non-null  uint8
 35  zipcode_98033              19414 non-null  uint8
 36  zipcode_98034              19414 non-null  uint8
 37  zipcode_98038              19414 non-null  uint8
 38  zipcode_98040              19414 non-null  uint8
 39  zipcode_98042              19414 non-null  uint8
 40  zipcode_98052              19414 non-null  uint8
 41  zipcode_98053              19414 non-null  uint8
 42  zipcode_98055              19414 non-null  uint8
 43  zipcode_98056              19414 non-null  uint8
 44  zipcode_98058              19414 non-null  uint8
 45  zipcode_98059              19414 non-null  uint8
 46  zipcode_98065              19414 non-null  uint8
 47  zipcode_98070              19414 non-null  uint8
 48  zipcode_98072              19414 non-null  uint8
 49  zipcode_98074              19414 non-null  uint8
 50  zipcode_98075              19414 non-null  uint8
 51  zipcode_98077              19414 non-null  uint8
 52  zipcode_98092              19414 non-null  uint8
 53  zipcode_98102              19414 non-null  uint8
 54  zipcode_98103              19414 non-null  uint8
 55  zipcode_98105              19414 non-null  uint8
 56  zipcode_98106              19414 non-null  uint8
 57  zipcode_98107              19414 non-null  uint8
 58  zipcode_98108              19414 non-null  uint8
 59  zipcode_98109              19414 non-null  uint8
 60  zipcode_98112              19414 non-null  uint8
 61  zipcode_98115              19414 non-null  uint8
 62  zipcode_98116              19414 non-null  uint8
 63  zipcode_98117              19414 non-null  uint8
 64  zipcode_98118              19414 non-null  uint8
 65  zipcode_98119              19414 non-null  uint8
 66  zipcode_98122              19414 non-null  uint8
 67  zipcode_98125              19414 non-null  uint8
 68  zipcode_98126              19414 non-null  uint8
 69  zipcode_98133              19414 non-null  uint8
 70  zipcode_98136              19414 non-null  uint8
 71  zipcode_98144              19414 non-null  uint8
 72  zipcode_98146              19414 non-null  uint8
 73  zipcode_98148              19414 non-null  uint8
 74  zipcode_98155              19414 non-null  uint8
```

```
 75  zipcode_98166               19414 non-null  uint8
 76  zipcode_98168               19414 non-null  uint8
 77  zipcode_98177               19414 non-null  uint8
 78  zipcode_98178               19414 non-null  uint8
 79  zipcode_98188               19414 non-null  uint8
 80  zipcode_98198               19414 non-null  uint8
 81  zipcode_98199               19414 non-null  uint8
dtypes: float64(9), int64(6), uint8(67)
memory usage: 3.6 MB
```

In [222…

```
model_7 = sm.OLS(y, sm.add_constant(X)).fit()
model_7.summary()
```

Out[222…

OLS Regression Results

Dep. Variable:	price_log	**R-squared:**	0.824
Model:	OLS	**Adj. R-squared:**	0.824
Method:	Least Squares	**F-statistic:**	1135.
Date:	Tue, 12 Apr 2022	**Prob (F-statistic):**	0.00
Time:	17:54:52	**Log-Likelihood:**	3469.4
No. Observations:	19414	**AIC:**	-6777.
Df Residuals:	19333	**BIC:**	-6139.
Df Model:	80		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	42.7426	14.032	3.046	0.002	15.238	70.247
grade	0.1961	0.002	95.269	0.000	0.192	0.200
bathrooms	0.1203	0.003	36.143	0.000	0.114	0.127
lat	0.1918	0.097	1.979	0.048	0.002	0.382
bedrooms	0.0481	0.002	23.695	0.000	0.044	0.052
floors	0.0089	0.004	2.169	0.030	0.001	0.017
closest_school_distance	-0.0136	0.002	-6.412	0.000	-0.018	-0.009
sqft_basement	0.0230	0.004	6.304	0.000	0.016	0.030

closest_hospital_distance	0.0033	0.002	1.611	0.107	-0.001	0.007
yr_renovated	2.974e-05	4.48e-06	6.644	0.000	2.1e-05	3.85e-05
sqft_lot15	1.502e-06	1.41e-07	10.693	0.000	1.23e-06	1.78e-06
long	0.3236	0.105	3.092	0.002	0.118	0.529
age_of_home	0.0021	8.44e-05	24.817	0.000	0.002	0.002
condition	0.0472	0.003	18.312	0.000	0.042	0.052
waterfront	-3.919e-13	1.68e-14	-23.315	0.000	-4.25e-13	-3.59e-13
view	-1.414e-13	6.08e-15	-23.271	0.000	-1.53e-13	-1.29e-13
zipcode_98001	-1.0592	0.048	-21.971	0.000	-1.154	-0.965
zipcode_98002	-1.1179	0.048	-23.256	0.000	-1.212	-1.024
zipcode_98003	-1.0669	0.049	-21.559	0.000	-1.164	-0.970
zipcode_98004	-0.1629	0.037	-4.410	0.000	-0.235	-0.091
zipcode_98005	-0.5462	0.039	-14.048	0.000	-0.622	-0.470
zipcode_98006	-0.6225	0.037	-16.617	0.000	-0.696	-0.549
zipcode_98007	-0.6542	0.040	-16.458	0.000	-0.732	-0.576
zipcode_98008	-0.6382	0.039	-16.574	0.000	-0.714	-0.563
zipcode_98010	-0.7910	0.055	-14.346	0.000	-0.899	-0.683
zipcode_98011	-0.8249	0.040	-20.404	0.000	-0.904	-0.746
zipcode_98014	-0.8278	0.052	-15.859	0.000	-0.930	-0.726
zipcode_98019	-0.8838	0.048	-18.267	0.000	-0.979	-0.789
zipcode_98023	-1.0743	0.052	-20.654	0.000	-1.176	-0.972
zipcode_98024	-0.7938	0.055	-14.441	0.000	-0.901	-0.686
zipcode_98027	-0.7093	0.040	-17.618	0.000	-0.788	-0.630
zipcode_98028	-0.8813	0.039	-22.413	0.000	-0.958	-0.804
zipcode_98029	-0.6508	0.042	-15.481	0.000	-0.733	-0.568
zipcode_98030	-1.1057	0.044	-25.164	0.000	-1.192	-1.020
zipcode_98031	-1.1357	0.042	-27.020	0.000	-1.218	-1.053

zipcode_98032	-1.2135	0.047	-25.749	0.000	-1.306	-1.121
zipcode_98033	-0.5173	0.037	-13.998	0.000	-0.590	-0.445
zipcode_98034	-0.7861	0.038	-20.927	0.000	-0.860	-0.712
zipcode_98038	-0.9736	0.046	-21.000	0.000	-1.065	-0.883
zipcode_98040	-0.3985	0.038	-10.483	0.000	-0.473	-0.324
zipcode_98042	-1.0955	0.044	-24.951	0.000	-1.182	-1.009
zipcode_98052	-0.6434	0.038	-16.974	0.000	-0.718	-0.569
zipcode_98053	-0.5799	0.042	-13.858	0.000	-0.662	-0.498
zipcode_98055	-1.1251	0.040	-27.829	0.000	-1.204	-1.046
zipcode_98056	-0.9353	0.039	-24.086	0.000	-1.011	-0.859
zipcode_98058	-1.0789	0.040	-26.942	0.000	-1.157	-1.000
zipcode_98059	-0.8766	0.039	-22.431	0.000	-0.953	-0.800
zipcode_98065	-0.6495	0.049	-13.124	0.000	-0.747	-0.552
zipcode_98070	-0.7886	0.057	-13.731	0.000	-0.901	-0.676
zipcode_98072	-0.7727	0.041	-18.862	0.000	-0.853	-0.692
zipcode_98074	-0.6806	0.040	-17.021	0.000	-0.759	-0.602
zipcode_98075	-0.6169	0.041	-15.050	0.000	-0.697	-0.537
zipcode_98077	-0.7882	0.044	-18.035	0.000	-0.874	-0.703
zipcode_98092	-1.0737	0.047	-22.770	0.000	-1.166	-0.981
zipcode_98102	-0.5000	0.041	-12.132	0.000	-0.581	-0.419
zipcode_98103	-0.6035	0.038	-16.089	0.000	-0.677	-0.530
zipcode_98105	-0.4776	0.038	-12.524	0.000	-0.552	-0.403
zipcode_98106	-1.0186	0.040	-25.653	0.000	-1.096	-0.941
zipcode_98107	-0.5947	0.040	-15.042	0.000	-0.672	-0.517
zipcode_98108	-0.9886	0.039	-25.071	0.000	-1.066	-0.911
zipcode_98109	-0.4228	0.042	-10.178	0.000	-0.504	-0.341
zipcode_98112	-0.3683	0.038	-9.799	0.000	-0.442	-0.295
zipcode_98115	-0.5718	0.037	-15.551	0.000	-0.644	-0.500

zipcode_98116	-0.6057	0.040	-14.974	0.000	-0.685	-0.526
zipcode_98117	-0.5782	0.039	-14.929	0.000	-0.654	-0.502
zipcode_98118	-0.8993	0.038	-23.897	0.000	-0.973	-0.826
zipcode_98119	-0.4608	0.040	-11.473	0.000	-0.540	-0.382
zipcode_98122	-0.6405	0.038	-17.076	0.000	-0.714	-0.567
zipcode_98125	-0.8097	0.038	-21.434	0.000	-0.884	-0.736
zipcode_98126	-0.7886	0.040	-19.656	0.000	-0.867	-0.710
zipcode_98133	-0.8988	0.039	-22.990	0.000	-0.975	-0.822
zipcode_98136	-0.6537	0.041	-15.778	0.000	-0.735	-0.572
zipcode_98144	-0.7291	0.038	-19.434	0.000	-0.803	-0.656
zipcode_98146	-1.0106	0.041	-24.594	0.000	-1.091	-0.930
zipcode_98148	-1.0655	0.049	-21.838	0.000	-1.161	-0.970
zipcode_98155	-0.9114	0.039	-23.268	0.000	-0.988	-0.835
zipcode_98166	-0.8932	0.042	-21.156	0.000	-0.976	-0.810
zipcode_98168	-1.1796	0.040	-29.294	0.000	-1.258	-1.101
zipcode_98177	-0.7011	0.042	-16.849	0.000	-0.783	-0.620
zipcode_98178	-1.1747	0.040	-29.102	0.000	-1.254	-1.096
zipcode_98188	-1.1535	0.043	-26.817	0.000	-1.238	-1.069
zipcode_98198	-1.1200	0.045	-24.968	0.000	-1.208	-1.032
zipcode_98199	-0.4936	0.040	-12.369	0.000	-0.572	-0.415

Omnibus:	927.809	**Durbin-Watson:**	1.987
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2842.554
Skew:	-0.186	**Prob(JB):**	0.00
Kurtosis:	4.837	**Cond. No.**	2.31e+20

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 8.75e-29. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [223…

```
high_p_vals(model_7)
```

```
const has an acceptable p_value
grade has an acceptable p_value
bathrooms has an acceptable p_value
lat has an acceptable p_value
bedrooms has an acceptable p_value
floors has an acceptable p_value
closest_school_distance has an acceptable p_value
sqft_basement has an acceptable p_value
yr_renovated has an acceptable p_value
sqft_lot15 has an acceptable p_value
long has an acceptable p_value
age_of_home has an acceptable p_value
condition has an acceptable p_value
waterfront has an acceptable p_value
view has an acceptable p_value
zipcode_98001 has an acceptable p_value
zipcode_98002 has an acceptable p_value
zipcode_98003 has an acceptable p_value
zipcode_98004 has an acceptable p_value
zipcode_98005 has an acceptable p_value
zipcode_98006 has an acceptable p_value
zipcode_98007 has an acceptable p_value
zipcode_98008 has an acceptable p_value
zipcode_98010 has an acceptable p_value
zipcode_98011 has an acceptable p_value
zipcode_98014 has an acceptable p_value
zipcode_98019 has an acceptable p_value
zipcode_98023 has an acceptable p_value
zipcode_98024 has an acceptable p_value
zipcode_98027 has an acceptable p_value
zipcode_98028 has an acceptable p_value
zipcode_98029 has an acceptable p_value
zipcode_98030 has an acceptable p_value
zipcode_98031 has an acceptable p_value
zipcode_98032 has an acceptable p_value
zipcode_98033 has an acceptable p_value
zipcode_98034 has an acceptable p_value
zipcode_98038 has an acceptable p_value
```

```
zipcode_98040 has an acceptable p_value
zipcode_98042 has an acceptable p_value
zipcode_98052 has an acceptable p_value
zipcode_98053 has an acceptable p_value
zipcode_98055 has an acceptable p_value
zipcode_98056 has an acceptable p_value
zipcode_98058 has an acceptable p_value
zipcode_98059 has an acceptable p_value
zipcode_98065 has an acceptable p_value
zipcode_98070 has an acceptable p_value
zipcode_98072 has an acceptable p_value
zipcode_98074 has an acceptable p_value
zipcode_98075 has an acceptable p_value
zipcode_98077 has an acceptable p_value
zipcode_98092 has an acceptable p_value
zipcode_98102 has an acceptable p_value
zipcode_98103 has an acceptable p_value
zipcode_98105 has an acceptable p_value
zipcode_98106 has an acceptable p_value
zipcode_98107 has an acceptable p_value
zipcode_98108 has an acceptable p_value
zipcode_98109 has an acceptable p_value
zipcode_98112 has an acceptable p_value
zipcode_98115 has an acceptable p_value
zipcode_98116 has an acceptable p_value
zipcode_98117 has an acceptable p_value
zipcode_98118 has an acceptable p_value
zipcode_98119 has an acceptable p_value
zipcode_98122 has an acceptable p_value
zipcode_98125 has an acceptable p_value
zipcode_98126 has an acceptable p_value
zipcode_98133 has an acceptable p_value
zipcode_98136 has an acceptable p_value
zipcode_98144 has an acceptable p_value
zipcode_98146 has an acceptable p_value
zipcode_98148 has an acceptable p_value
zipcode_98155 has an acceptable p_value
zipcode_98166 has an acceptable p_value
zipcode_98168 has an acceptable p_value
zipcode_98177 has an acceptable p_value
zipcode_98178 has an acceptable p_value
zipcode_98188 has an acceptable p_value
zipcode_98198 has an acceptable p_value
zipcode_98199 has an acceptable p_value
```

Out[223…

```
"drop ['closest_hospital_distance'] for the next model due to high p_value(s)"
```

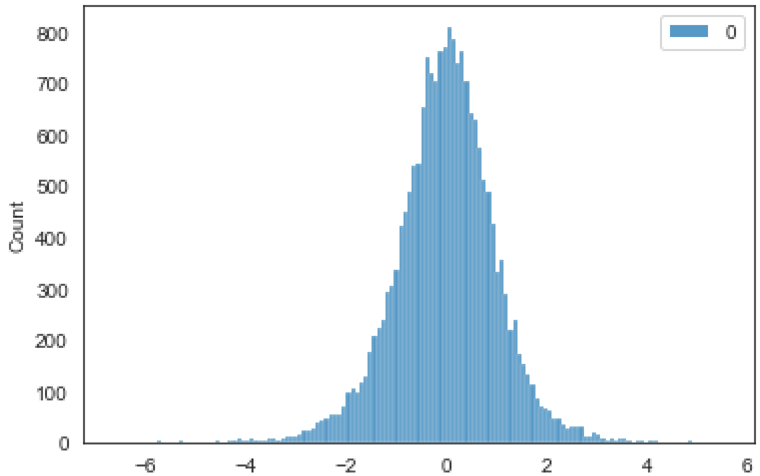
Adding in zipcodes has made the latitude and the closest hospital distance irrelevant as predictors and I'll be removing those features for the next iteration of the model.

In [224…

```
scaled_resid_7 = scaler.fit_transform(model_7.resid.values.reshape(-1, 1))
sns.histplot(scaled_resid_7)
```

Out[224…

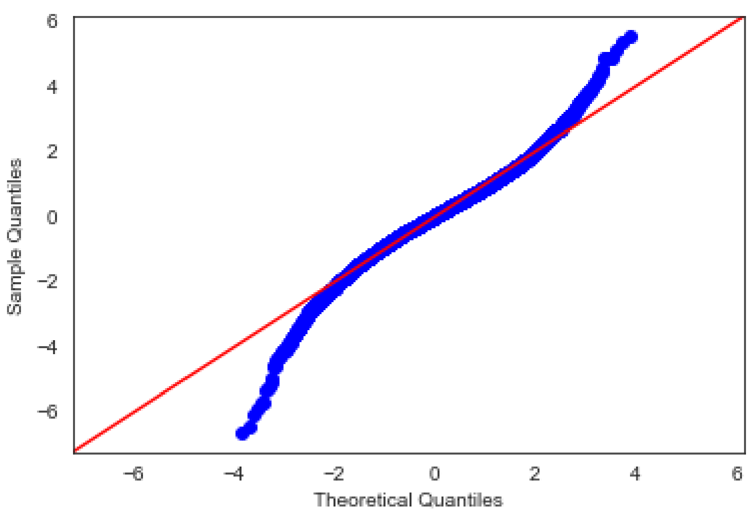
```
<AxesSubplot:ylabel='Count'>
```

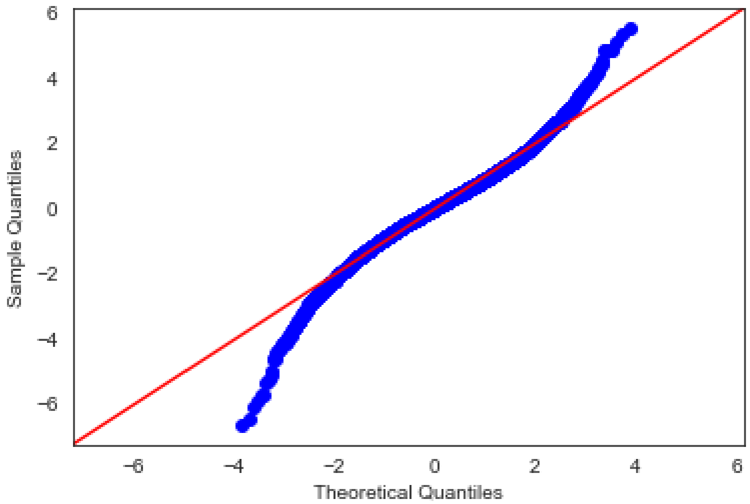


In [225…

```
sm.qqplot(model_7.resid, dist = stats.norm, line = "45", fit = True)
```

Out[225…





Observations: zipcode data has moved the model into performing less well, in that there are now more residuals in the -4 through -2 and 2 through 4 distances away from the mean. Basically, the model is predicting less correctly where the blue line isn't resting on the red line.

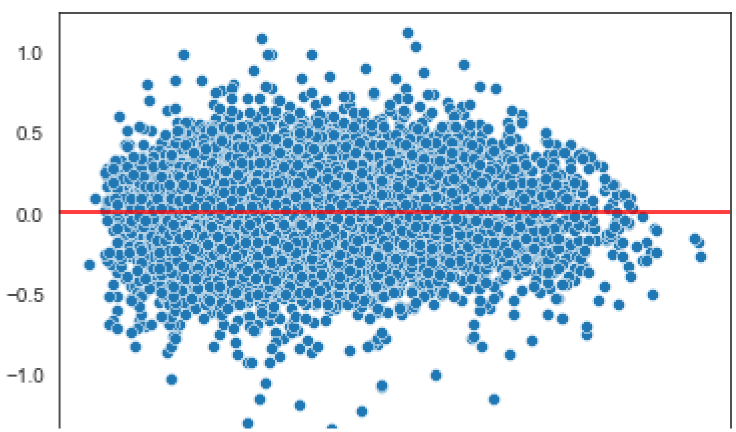
In [226…

```
sns.scatterplot(model_7.predict(sm.add_constant(X)), model_7.resid)
plt.axhline(0, color = 'red')
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[226…

```
<matplotlib.lines.Line2D at 0x7ff343f8cfa0>
```



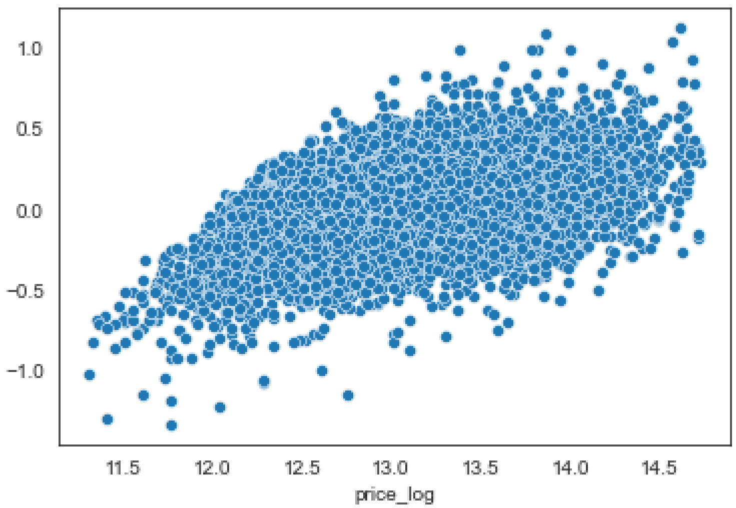
In [227…

```
sns.scatterplot(y, model_7.resid)
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[227…

```
<AxesSubplot:xlabel='price_log'>
```



Eighth Model: Latitude and Closest Hospital Distance

In [228…

```
X = X.drop(columns = ['lat', 'closest_hospital_distance'])
```

In [229…

```
features_8 = [col for col in X.columns]
```

In [230…

```
X = zipcode_data.loc[:, features_8]
```

In [231…

```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19414 entries, 0 to 21596
Data columns (total 80 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   grade                    19414 non-null  int64  
 1   bathrooms                19414 non-null  float64
 2   bedrooms                 19414 non-null  int64  
 3   floors                   19414 non-null  float64
 4   closest_school_distance  19414 non-null  float64
 5   sqft_basement            19414 non-null  float64
 6   yr_renovated             19414 non-null  float64
 7   sqft_lot15               19414 non-null  int64  
 8   long                     19414 non-null  float64
 9   age_of_home              19414 non-null  int64  
 10  condition                19414 non-null  int64  
 11  waterfront               19414 non-null  int64  
 12  view                     19414 non-null  float64
 13  zipcode_98001            19414 non-null  uint8  
 14  zipcode_98002            19414 non-null  uint8  
 15  zipcode_98003            19414 non-null  uint8  
 16  zipcode_98004            19414 non-null  uint8  
 17  zipcode_98005            19414 non-null  uint8  
 18  zipcode_98006            19414 non-null  uint8  
 19  zipcode_98007            19414 non-null  uint8  
 20  zipcode_98008            19414 non-null  uint8  
 21  zipcode_98010            19414 non-null  uint8  
 22  zipcode_98011            19414 non-null  uint8  
 23  zipcode_98014            19414 non-null  uint8  
 24  zipcode_98019            19414 non-null  uint8  
 25  zipcode_98023            19414 non-null  uint8  
 26  zipcode_98024            19414 non-null  uint8  
 27  zipcode_98027            19414 non-null  uint8  
 28  zipcode_98028            19414 non-null  uint8  
 29  zipcode_98029            19414 non-null  uint8  
 30  zipcode_98030            19414 non-null  uint8  
 31  zipcode_98031            19414 non-null  uint8  
 32  zipcode_98032            19414 non-null  uint8  
 33  zipcode_98033            19414 non-null  uint8  
 34  zipcode_98034            19414 non-null  uint8  
 35  zipcode_98038            19414 non-null  uint8  
 36  zipcode_98040            19414 non-null  uint8  
 37  zipcode_98042            19414 non-null  uint8  
 38  zipcode_98052            19414 non-null  uint8  
 39  zipcode_98053            19414 non-null  uint8  
 40  zipcode_98055            19414 non-null  uint8  
```

```
 40  zipcode_98055            19414 non-null  uint8
 41  zipcode_98056            19414 non-null  uint8
 42  zipcode_98058            19414 non-null  uint8
 43  zipcode_98059            19414 non-null  uint8
 44  zipcode_98065            19414 non-null  uint8
 45  zipcode_98070            19414 non-null  uint8
 46  zipcode_98072            19414 non-null  uint8
 47  zipcode_98074            19414 non-null  uint8
 48  zipcode_98075            19414 non-null  uint8
 49  zipcode_98077            19414 non-null  uint8
 50  zipcode_98092            19414 non-null  uint8
 51  zipcode_98102            19414 non-null  uint8
 52  zipcode_98103            19414 non-null  uint8
 53  zipcode_98105            19414 non-null  uint8
 54  zipcode_98106            19414 non-null  uint8
 55  zipcode_98107            19414 non-null  uint8
 56  zipcode_98108            19414 non-null  uint8
 57  zipcode_98109            19414 non-null  uint8
 58  zipcode_98112            19414 non-null  uint8
 59  zipcode_98115            19414 non-null  uint8
 60  zipcode_98116            19414 non-null  uint8
 61  zipcode_98117            19414 non-null  uint8
 62  zipcode_98118            19414 non-null  uint8
 63  zipcode_98119            19414 non-null  uint8
 64  zipcode_98122            19414 non-null  uint8
 65  zipcode_98125            19414 non-null  uint8
 66  zipcode_98126            19414 non-null  uint8
 67  zipcode_98133            19414 non-null  uint8
 68  zipcode_98136            19414 non-null  uint8
 69  zipcode_98144            19414 non-null  uint8
 70  zipcode_98146            19414 non-null  uint8
 71  zipcode_98148            19414 non-null  uint8
 72  zipcode_98155            19414 non-null  uint8
 73  zipcode_98166            19414 non-null  uint8
 74  zipcode_98168            19414 non-null  uint8
 75  zipcode_98177            19414 non-null  uint8
 76  zipcode_98178            19414 non-null  uint8
 77  zipcode_98188            19414 non-null  uint8
 78  zipcode_98198            19414 non-null  uint8
 79  zipcode_98199            19414 non-null  uint8
dtypes: float64(7), int64(6), uint8(67)
memory usage: 3.3 MB
```

In [232…

```
model_8 = sm.OLS(y, sm.add_constant(X)).fit()
model_8.summary()
```

Out[232… OLS Regression Results

Dep. Variable:	price_log	**R-squared:**	0.824
Model:	OLS	**Adj. R-squared:**	0.824
Method:	Least Squares	**F-statistic:**	1163.
Date:	Tue, 12 Apr 2022	**Prob (F-statistic):**	0.00
Time:	17:54:53	**Log-Likelihood:**	3465.5
No. Observations:	19414	**AIC:**	-6773.
Df Residuals:	19335	**BIC:**	-6151.
Df Model:	78		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	57.6242	12.627	4.564	0.000	32.875	82.373
grade	0.1963	0.002	95.575	0.000	0.192	0.200
bathrooms	0.1203	0.003	36.134	0.000	0.114	0.127
bedrooms	0.0481	0.002	23.704	0.000	0.044	0.052
floors	0.0083	0.004	2.031	0.042	0.000	0.016
closest_school_distance	-0.0147	0.002	-7.659	0.000	-0.019	-0.011
sqft_basement	0.0229	0.004	6.284	0.000	0.016	0.030
yr_renovated	2.987e-05	4.48e-06	6.673	0.000	2.11e-05	3.86e-05
sqft_lot15	1.511e-06	1.4e-07	10.786	0.000	1.24e-06	1.79e-06
long	0.3705	0.103	3.588	0.000	0.168	0.573
age_of_home	0.0021	8.44e-05	24.793	0.000	0.002	0.002
condition	0.0471	0.003	18.293	0.000	0.042	0.052
waterfront	-2.983e-13	1.58e-14	-18.860	0.000	-3.29e-13	-2.67e-13
view	-2.622e-13	1.34e-14	-19.621	0.000	-2.88e-13	-2.36e-13
zipcode_98001	-1.1037	0.043	-25.578	0.000	-1.188	-1.019
zipcode_98002	-1.1686	0.043	-27.310	0.000	-1.252	-1.085

zipcode_98003	-1.1091	0.045	-24.670	0.000	-1.197	-1.021
zipcode_98004	-0.1668	0.037	-4.519	0.000	-0.239	-0.094
zipcode_98005	-0.5499	0.039	-14.170	0.000	-0.626	-0.474
zipcode_98006	-0.6280	0.037	-16.993	0.000	-0.700	-0.556
zipcode_98007	-0.6553	0.040	-16.502	0.000	-0.733	-0.577
zipcode_98008	-0.6352	0.038	-16.511	0.000	-0.711	-0.560
zipcode_98010	-0.8193	0.050	-16.257	0.000	-0.918	-0.721
zipcode_98011	-0.7962	0.038	-20.925	0.000	-0.871	-0.722
zipcode_98014	-0.7912	0.051	-15.662	0.000	-0.890	-0.692
zipcode_98019	-0.8346	0.045	-18.605	0.000	-0.923	-0.747
zipcode_98023	-1.1155	0.048	-23.206	0.000	-1.210	-1.021
zipcode_98024	-0.8022	0.055	-14.692	0.000	-0.909	-0.695
zipcode_98027	-0.7272	0.040	-18.304	0.000	-0.805	-0.649
zipcode_98028	-0.8488	0.037	-22.880	0.000	-0.921	-0.776
zipcode_98029	-0.6635	0.042	-15.938	0.000	-0.745	-0.582
zipcode_98030	-1.1413	0.039	-29.240	0.000	-1.218	-1.065
zipcode_98031	-1.1714	0.038	-30.879	0.000	-1.246	-1.097
zipcode_98032	-1.2433	0.043	-29.175	0.000	-1.327	-1.160
zipcode_98033	-0.5071	0.037	-13.892	0.000	-0.579	-0.436
zipcode_98034	-0.7689	0.036	-21.273	0.000	-0.840	-0.698
zipcode_98038	-0.9948	0.041	-24.323	0.000	-1.075	-0.915
zipcode_98040	-0.4060	0.037	-10.910	0.000	-0.479	-0.333
zipcode_98042	-1.1264	0.039	-29.099	0.000	-1.202	-1.051
zipcode_98052	-0.6285	0.037	-16.902	0.000	-0.701	-0.556
zipcode_98053	-0.5522	0.041	-13.613	0.000	-0.632	-0.473
zipcode_98055	-1.1584	0.037	-30.926	0.000	-1.232	-1.085
zipcode_98056	-0.9497	0.037	-25.797	0.000	-1.022	-0.877

zipcode_98058	-1.1087	0.037	-29.926	0.000	-1.181	-1.036
zipcode_98059	-0.8938	0.037	-24.127	0.000	-0.966	-0.821
zipcode_98065	-0.6676	0.049	-13.755	0.000	-0.763	-0.572
zipcode_98070	-0.7960	0.055	-14.527	0.000	-0.903	-0.689
zipcode_98072	-0.7432	0.038	-19.468	0.000	-0.818	-0.668
zipcode_98074	-0.6681	0.040	-16.813	0.000	-0.746	-0.590
zipcode_98075	-0.6176	0.041	-15.090	0.000	-0.698	-0.537
zipcode_98077	-0.7522	0.041	-18.417	0.000	-0.832	-0.672
zipcode_98092	-1.1207	0.042	-26.978	0.000	-1.202	-1.039
zipcode_98102	-0.5006	0.041	-12.148	0.000	-0.581	-0.420
zipcode_98103	-0.5925	0.037	-15.914	0.000	-0.665	-0.520
zipcode_98105	-0.4737	0.038	-12.470	0.000	-0.548	-0.399
zipcode_98106	-1.0289	0.038	-26.841	0.000	-1.104	-0.954
zipcode_98107	-0.5864	0.039	-14.923	0.000	-0.663	-0.509
zipcode_98108	-1.0049	0.039	-26.028	0.000	-1.081	-0.929
zipcode_98109	-0.4183	0.042	-10.078	0.000	-0.500	-0.337
zipcode_98112	-0.3710	0.038	-9.874	0.000	-0.445	-0.297
zipcode_98115	-0.5608	0.036	-15.389	0.000	-0.632	-0.489
zipcode_98116	-0.6051	0.040	-15.287	0.000	-0.683	-0.528
zipcode_98117	-0.5643	0.038	-14.745	0.000	-0.639	-0.489
zipcode_98118	-0.9149	0.036	-25.111	0.000	-0.986	-0.843
zipcode_98119	-0.4546	0.040	-11.335	0.000	-0.533	-0.376
zipcode_98122	-0.6476	0.037	-17.306	0.000	-0.721	-0.574
zipcode_98125	-0.7924	0.037	-21.456	0.000	-0.865	-0.720
zipcode_98126	-0.7950	0.039	-20.488	0.000	-0.871	-0.719
zipcode_98133	-0.8749	0.038	-23.305	0.000	-0.948	-0.801
zipcode_98136	-0.6587	0.040	-16.523	0.000	-0.737	-0.581
zipcode_98144	-0.7391	0.037	-19.848	0.000	-0.812	-0.666

zipcode_98146	-1.0262	0.039	-26.290	0.000	-1.103	-0.950
zipcode_98148	-1.0954	0.047	-23.555	0.000	-1.187	-1.004
zipcode_98155	-0.8799	0.037	-23.820	0.000	-0.952	-0.807
zipcode_98166	-0.9220	0.040	-23.069	0.000	-1.000	-0.844
zipcode_98168	-1.2000	0.038	-31.604	0.000	-1.274	-1.126
zipcode_98177	-0.6714	0.040	-16.888	0.000	-0.749	-0.593
zipcode_98178	-1.1937	0.038	-31.792	0.000	-1.267	-1.120
zipcode_98188	-1.1815	0.040	-29.638	0.000	-1.260	-1.103
zipcode_98198	-1.1473	0.041	-28.089	0.000	-1.227	-1.067
zipcode_98199	-0.4854	0.040	-12.196	0.000	-0.563	-0.407

Omnibus:	928.494	**Durbin-Watson:**	1.988
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2849.400
Skew:	-0.186	**Prob(JB):**	0.00
Kurtosis:	4.840	**Cond. No.**	2.32e+20

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 8.69e-29. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [233…

```
high_p_vals(model_8)
```

```
const has an acceptable p_value
grade has an acceptable p_value
bathrooms has an acceptable p_value
bedrooms has an acceptable p_value
floors has an acceptable p_value
closest_school_distance has an acceptable p_value
sqft_basement has an acceptable p_value
yr_renovated has an acceptable p_value
sqft_lot15 has an acceptable p_value
```

```
long has an acceptable p_value
age_of_home has an acceptable p_value
condition has an acceptable p_value
waterfront has an acceptable p_value
view has an acceptable p_value
zipcode_98001 has an acceptable p_value
zipcode_98002 has an acceptable p_value
zipcode_98003 has an acceptable p_value
zipcode_98004 has an acceptable p_value
zipcode_98005 has an acceptable p_value
zipcode_98006 has an acceptable p_value
zipcode_98007 has an acceptable p_value
zipcode_98008 has an acceptable p_value
zipcode_98010 has an acceptable p_value
zipcode_98011 has an acceptable p_value
zipcode_98014 has an acceptable p_value
zipcode_98019 has an acceptable p_value
zipcode_98023 has an acceptable p_value
zipcode_98024 has an acceptable p_value
zipcode_98027 has an acceptable p_value
zipcode_98028 has an acceptable p_value
zipcode_98029 has an acceptable p_value
zipcode_98030 has an acceptable p_value
zipcode_98031 has an acceptable p_value
zipcode_98032 has an acceptable p_value
zipcode_98033 has an acceptable p_value
zipcode_98034 has an acceptable p_value
zipcode_98038 has an acceptable p_value
zipcode_98040 has an acceptable p_value
zipcode_98042 has an acceptable p_value
zipcode_98052 has an acceptable p_value
zipcode_98053 has an acceptable p_value
zipcode_98055 has an acceptable p_value
zipcode_98056 has an acceptable p_value
zipcode_98058 has an acceptable p_value
zipcode_98059 has an acceptable p_value
zipcode_98065 has an acceptable p_value
zipcode_98070 has an acceptable p_value
zipcode_98072 has an acceptable p_value
zipcode_98074 has an acceptable p_value
zipcode_98075 has an acceptable p_value
zipcode_98077 has an acceptable p_value
zipcode_98092 has an acceptable p_value
zipcode_98102 has an acceptable p_value
zipcode_98103 has an acceptable p_value
```

```
zipcode_98105 has an acceptable p_value
zipcode_98106 has an acceptable p_value
zipcode_98107 has an acceptable p_value
zipcode_98108 has an acceptable p_value
zipcode_98109 has an acceptable p_value
zipcode_98112 has an acceptable p_value
zipcode_98115 has an acceptable p_value
zipcode_98116 has an acceptable p_value
zipcode_98117 has an acceptable p_value
zipcode_98118 has an acceptable p_value
zipcode_98119 has an acceptable p_value
zipcode_98122 has an acceptable p_value
zipcode_98125 has an acceptable p_value
zipcode_98126 has an acceptable p_value
zipcode_98133 has an acceptable p_value
zipcode_98136 has an acceptable p_value
zipcode_98144 has an acceptable p_value
zipcode_98146 has an acceptable p_value
zipcode_98148 has an acceptable p_value
zipcode_98155 has an acceptable p_value
zipcode_98166 has an acceptable p_value
zipcode_98168 has an acceptable p_value
zipcode_98177 has an acceptable p_value
zipcode_98178 has an acceptable p_value
zipcode_98188 has an acceptable p_value
zipcode_98198 has an acceptable p_value
zipcode_98199 has an acceptable p_value
```

Out[233…

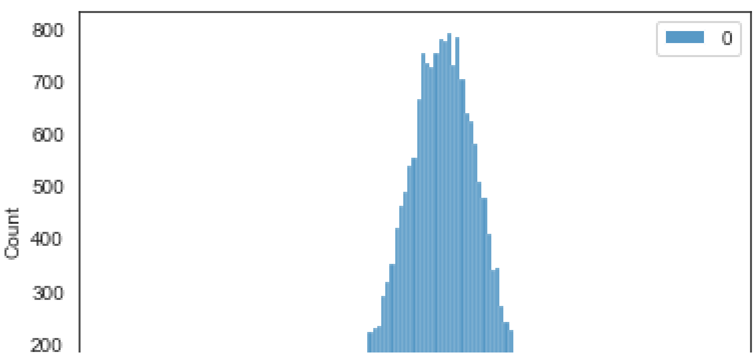
```
"drop ['closest_hospital_distance'] for the next model due to high p_value(s)"
```

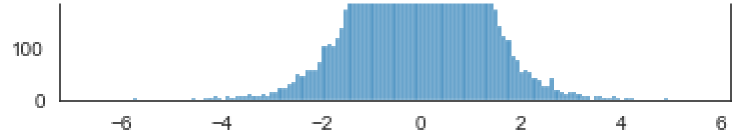
In [234…

```
scaled_resid_8 = scaler.fit_transform(model_8.resid.values.reshape(-1, 1))
sns.histplot(scaled_resid_8)
```

Out[234…

```
<AxesSubplot:ylabel='Count'>
```

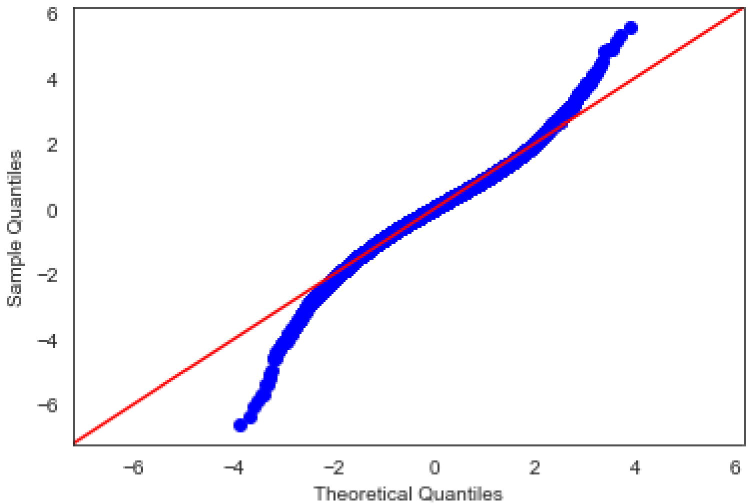


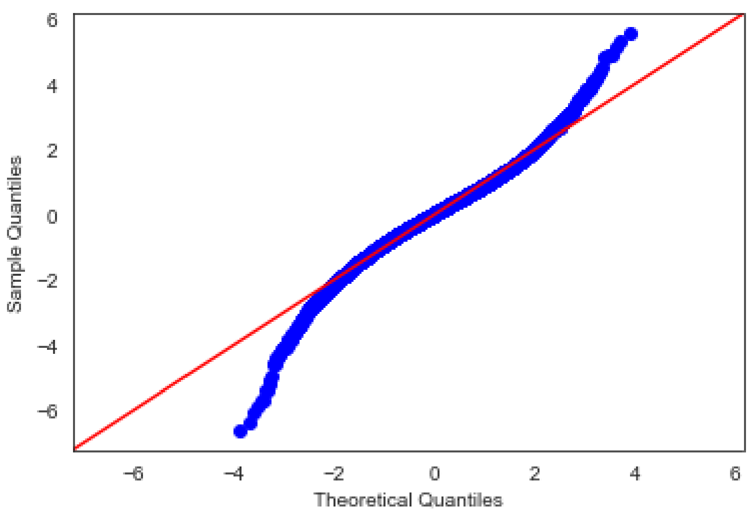


In [235…

```
sm.qqplot(model_8.resid, dist = stats.norm, line = "45", fit = True)
```

Out[235…





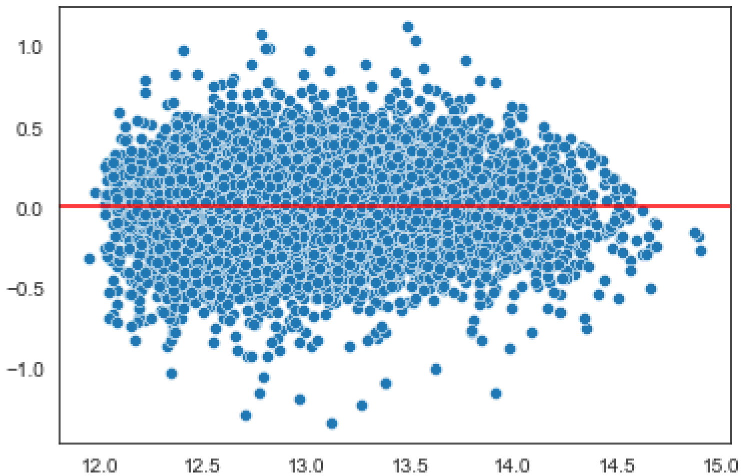
In [236…

```
sns.scatterplot(model_8.predict(sm.add_constant(X)), model_8.resid)
plt.axhline(0, color = 'red')
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[236…

```
<matplotlib.lines.Line2D at 0x7ff344b0aaf0>
```



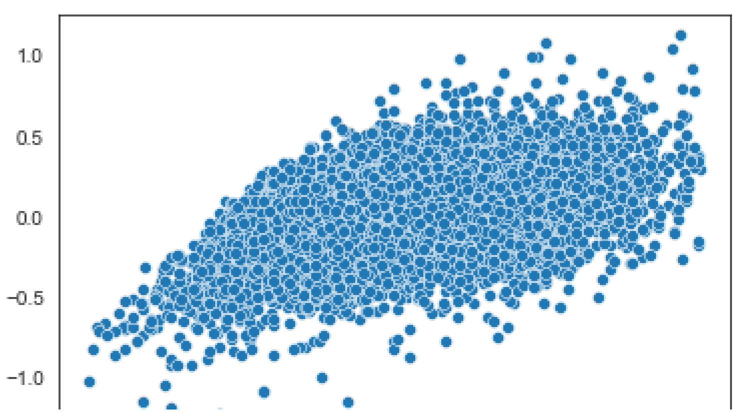
In [237…

```
sns.scatterplot(y, model_8.resid)
```

```
/Users/b0ihazard/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[237…

```
<AxesSubplot:xlabel='price_log'>
```



Conclusion

I'm going to generate a report of the findings of the model's coefficiants.

In [238…

```
pd.set_option('display.max_rows', None)

coeff_s = model_8.params
coeff_s = coeff_s.drop('const')

labels = coeff_s.index
coefficiants = dict(zip(labels, coeff_s))

# grabbing the coefficiants from the stats models reports
```

By generating a report of the findings, should this model be used on other data in the future, understanding the model will become much more approachable and user friendly. This is good news for the civic workers that will be using this model.

In [239…

```
for x, y in coefficiants.items():
    y = y * 100
    if x.startswith('view') or x.startswith('waterfront'):
        print (f' If {x} was a feature of a property, the price would rise by {y:.2f} percent.')
    elif not x.startswith('zipcode') and not x.startswith('const'):
        print (f' As {x} increases by one unit, the price changes by {y:.2f} percent.')
    else:
        x.startswith('zipcode') and not x.startswith('const')
        print (f' If a property was in {x} zipcode instead of 98039, the price would change by {y:.2f}
```

```
As grade increases by one unit, the price changes by 19.63 percent.
As bathrooms increases by one unit, the price changes by 12.03 percent.
As bedrooms increases by one unit, the price changes by 4.81 percent.
As floors increases by one unit, the price changes by 0.83 percent.
As closest_school_distance increases by one unit, the price changes by -1.47 percent.
As sqft_basement increases by one unit, the price changes by 2.29 percent.
As yr_renovated increases by one unit, the price changes by 0.00 percent.
As sqft_lot15 increases by one unit, the price changes by 0.00 percent.
As long increases by one unit, the price changes by 37.05 percent.
As age_of_home increases by one unit, the price changes by 0.21 percent.
```

```
As condition increases by one unit, the price changes by 4.71 percent.
If waterfront was a feature of a property, the price would rise by -0.00 percent.
If view was a feature of a property, the price would rise by -0.00 percent.
If a property was in zipcode_98001 zipcode instead of 98039, the price would change by -110.37  percent.
If a property was in zipcode_98002 zipcode instead of 98039, the price would change by -116.86  percent.
If a property was in zipcode_98003 zipcode instead of 98039, the price would change by -110.91  percent.
If a property was in zipcode_98004 zipcode instead of 98039, the price would change by -16.68  percent.
If a property was in zipcode_98005 zipcode instead of 98039, the price would change by -54.99  percent.
If a property was in zipcode_98006 zipcode instead of 98039, the price would change by -62.80  percent.
If a property was in zipcode_98007 zipcode instead of 98039, the price would change by -65.53  percent.
If a property was in zipcode_98008 zipcode instead of 98039, the price would change by -63.52  percent.
If a property was in zipcode_98010 zipcode instead of 98039, the price would change by -81.93  percent.
If a property was in zipcode_98011 zipcode instead of 98039, the price would change by -79.62  percent.
If a property was in zipcode_98014 zipcode instead of 98039, the price would change by -79.12  percent.
If a property was in zipcode_98019 zipcode instead of 98039, the price would change by -83.46  percent.
If a property was in zipcode_98023 zipcode instead of 98039, the price would change by -111.55  percent.
If a property was in zipcode_98024 zipcode instead of 98039, the price would change by -80.22  percent.
If a property was in zipcode_98027 zipcode instead of 98039, the price would change by -72.72  percent.
If a property was in zipcode_98028 zipcode instead of 98039, the price would change by -84.88  percent.
If a property was in zipcode_98029 zipcode instead of 98039, the price would change by -66.35  percent.
If a property was in zipcode_98030 zipcode instead of 98039, the price would change by -114.13  percent.
If a property was in zipcode_98031 zipcode instead of 98039, the price would change by -117.14  percent.
If a property was in zipcode_98032 zipcode instead of 98039, the price would change by -124.33  percent.
If a property was in zipcode_98033 zipcode instead of 98039, the price would change by -50.71  percent.
If a property was in zipcode_98034 zipcode instead of 98039, the price would change by -76.89  percent.
If a property was in zipcode_98038 zipcode instead of 98039, the price would change by -99.48  percent.
If a property was in zipcode_98040 zipcode instead of 98039, the price would change by -40.60  percent.
If a property was in zipcode_98042 zipcode instead of 98039, the price would change by -112.64  percent.
If a property was in zipcode_98052 zipcode instead of 98039, the price would change by -62.85  percent.
If a property was in zipcode_98053 zipcode instead of 98039, the price would change by -55.22  percent.
If a property was in zipcode_98055 zipcode instead of 98039, the price would change by -115.84  percent.
If a property was in zipcode_98056 zipcode instead of 98039, the price would change by -94.97  percent.
If a property was in zipcode_98058 zipcode instead of 98039, the price would change by -110.87  percent.
If a property was in zipcode_98059 zipcode instead of 98039, the price would change by -89.38  percent.
If a property was in zipcode_98065 zipcode instead of 98039, the price would change by -66.76  percent.
If a property was in zipcode_98070 zipcode instead of 98039, the price would change by -79.60  percent.
If a property was in zipcode_98072 zipcode instead of 98039, the price would change by -74.32  percent.
If a property was in zipcode_98074 zipcode instead of 98039, the price would change by -66.81  percent.
If a property was in zipcode_98075 zipcode instead of 98039, the price would change by -61.76  percent.
If a property was in zipcode_98077 zipcode instead of 98039, the price would change by -75.22  percent.
If a property was in zipcode_98092 zipcode instead of 98039, the price would change by -112.07  percent.
If a property was in zipcode_98102 zipcode instead of 98039, the price would change by -50.06  percent.
If a property was in zipcode_98103 zipcode instead of 98039, the price would change by -59.25  percent.
If a property was in zipcode_98105 zipcode instead of 98039, the price would change by -47.37  percent.
If a property was in zipcode_98106 zipcode instead of 98039, the price would change by -102.89  percent.
If a property was in zipcode_98107 zipcode instead of 98039, the price would change by -58.64  percent.
```

```
If a property was in zipcode_98108 zipcode instead of 98039, the price would change by -100.49  percent.
If a property was in zipcode_98109 zipcode instead of 98039, the price would change by -41.83  percent.
If a property was in zipcode_98112 zipcode instead of 98039, the price would change by -37.10  percent.
If a property was in zipcode_98115 zipcode instead of 98039, the price would change by -56.08  percent.
If a property was in zipcode_98116 zipcode instead of 98039, the price would change by -60.51  percent.
If a property was in zipcode_98117 zipcode instead of 98039, the price would change by -56.43  percent.
If a property was in zipcode_98118 zipcode instead of 98039, the price would change by -91.49  percent.
If a property was in zipcode_98119 zipcode instead of 98039, the price would change by -45.46  percent.
If a property was in zipcode_98122 zipcode instead of 98039, the price would change by -64.76  percent.
If a property was in zipcode_98125 zipcode instead of 98039, the price would change by -79.24  percent.
If a property was in zipcode_98126 zipcode instead of 98039, the price would change by -79.50  percent.
If a property was in zipcode_98133 zipcode instead of 98039, the price would change by -87.49  percent.
If a property was in zipcode_98136 zipcode instead of 98039, the price would change by -65.87  percent.
If a property was in zipcode_98144 zipcode instead of 98039, the price would change by -73.91  percent.
If a property was in zipcode_98146 zipcode instead of 98039, the price would change by -102.62  percent.
If a property was in zipcode_98148 zipcode instead of 98039, the price would change by -109.54  percent.
If a property was in zipcode_98155 zipcode instead of 98039, the price would change by -87.99  percent.
If a property was in zipcode_98166 zipcode instead of 98039, the price would change by -92.20  percent.
If a property was in zipcode_98168 zipcode instead of 98039, the price would change by -120.00  percent.
If a property was in zipcode_98177 zipcode instead of 98039, the price would change by -67.14  percent.
If a property was in zipcode_98178 zipcode instead of 98039, the price would change by -119.37  percent.
If a property was in zipcode_98188 zipcode instead of 98039, the price would change by -118.15  percent.
If a property was in zipcode_98198 zipcode instead of 98039, the price would change by -114.73  percent.
If a property was in zipcode_98199 zipcode instead of 98039, the price would change by -48.54  percent.
```

Recommendations:

1) As we can read in the printed report above, improving the grade and the condition (as is our aim with renovations) would make the value of the properties increase by roughly 1/4th. *Therefore* I'm recommending that the city of Seattle focus on improving both grade and condition of properties during the renovation. In the report, year_renovated by itself had little affect on the price of the home. The conclusion here is that it's not enough to simply renovate a home, but we must improve it's condition and/or grade with that renovation.