Linear Regression Model for HGTV Show

1 Buiness Problem

One of HGTV Network's couples is moving to KingS Country and HGTV does not want to lose them. Instead of letting them go HGTV will be using the King county's data set to figure out what type of show would fit with the couple's new areas. The most important indicator is cost since that will be use to determine the budget.

2 Data Mining - Uploading and Reviewing Data

2.1 Importing

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.formula.api import ols
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import statsmodels.api as sm
import scipy.stats as stats
import folium

%matplotlib inline
```

executed in 3.27s, finished 20:54:17 2021-06-12

In [2]:
```
df = pd.read_csv('./Cloned-folder/dsc-phase-2-project/data/kc_house_data
df.head()
```
executed in 113ms, finished 20:54:17 2021-06-12

Out[2]:

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

5 rows × 21 columns

3 Data Cleaning/ Exploration

3.1 Functions Used

In [3]:
```
#Convert datatype to INT
def convert_int(df, col):
    df[col] = df[col].astype(str).astype(int)
    return df.info()
```
executed in 25ms, finished 20:54:17 2021-06-12

In [4]:
```
#Convert datatype to FLOAT
def convert_float(df, col):
    df[col] = df[col].astype(str).astype(float)
```
executed in 16ms, finished 20:54:17 2021-06-12

In [5]:
```
#Convert datatype to DATE
def convert_date(df, col):
    df[col] = pd.to_datetime(df[col])
    return df.info()
```
executed in 26ms, finished 20:54:17 2021-06-12

In [6]:
```
#converting Null falues to 0
def convert_nan_to_0(col):
    #df['view'].fillna(0.0)

    df[col].fillna(0.0, inplace=True)
    return df.info()
```

executed in 16ms, finished 20:54:17 2021-06-12

3.2 Column Names and Descriptions for Kings County Data Set

In [7]:
```
df.info()
```

executed in 41ms, finished 20:54:17 2021-06-12

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

Removing date from day from date during the converting because filming for any show would be continous and trying to fit a show around a few days in a single month would not be fesiable with productivity. As shown below it is having the day does not contribute addition infomation, the dip in houses sold would be just as evident if only month and year was used. Putting the data in to a histogram is an option it would also not beheltful becuase the information would not be seprated by month(which has unequal days moth to month.

In [8]:
```
1 convert_date(df, 'date')
```
executed in 98ms, finished 20:54:17 2021-06-12

```
<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  datetime64[ns]
 2   price          21597 non-null  float64
 3   bedrooms       21597 non-null  int64
 4   bathrooms      21597 non-null  float64
 5   sqft_living    21597 non-null  int64
 6   sqft_lot       21597 non-null  int64
 7   floors         21597 non-null  float64
 8   waterfront     19221 non-null  float64
 9   view           21534 non-null  float64
 10  condition      21597 non-null  int64
 11  grade          21597 non-null  int64
 12  sqft_above     21597 non-null  int64
 13  sqft_basement  21597 non-null  object
 14  yr_built       21597 non-null  int64
 15  yr_renovated   17755 non-null  float64
 16  zipcode        21597 non-null  int64
 17  lat            21597 non-null  float64
 18  long           21597 non-null  float64
 19  sqft_living15  21597 non-null  int64
 20  sqft_lot15     21597 non-null  int64
dtypes: datetime64[ns](1), float64(8), int64(11), object(1)
memory usage: 3.5+ MB
```

In [9]:
```
1 df['sqft_basement']
```
executed in 16ms, finished 20:54:17 2021-06-12

Out[9]:
```
0          0.0
1        400.0
2          0.0
3        910.0
4          0.0
         ...
21592      0.0
21593      0.0
21594      0.0
21595      0.0
21596      0.0
Name: sqft_basement, Length: 21597, dtype: object
```

In [10]:
```
1 df_date = df.sort_values(['date'])
```
executed in 24ms, finished 20:54:17 2021-06-12

In [11]:
```
fig, ax = plt.subplots(figsize=(100,10))
sns.countplot(x ='date', ax = ax, data = df_date)
plt.xticks(rotation=65, horizontalalignment='right')

```

executed in 24.7s, finished 20:54:42 2021-06-12

```
Text(94, 0, '2014-08-04T00:00:00.000000000'),
Text(95, 0, '2014-08-05T00:00:00.000000000'),
Text(96, 0, '2014-08-06T00:00:00.000000000'),
Text(97, 0, '2014-08-07T00:00:00.000000000'),
Text(98, 0, '2014-08-08T00:00:00.000000000'),
Text(99, 0, '2014-08-09T00:00:00.000000000'),
Text(100, 0, '2014-08-10T00:00:00.000000000'),
Text(101, 0, '2014-08-11T00:00:00.000000000'),
Text(102, 0, '2014-08-12T00:00:00.000000000'),
Text(103, 0, '2014-08-13T00:00:00.000000000'),
Text(104, 0, '2014-08-14T00:00:00.000000000'),
Text(105, 0, '2014-08-15T00:00:00.000000000'),
Text(106, 0, '2014-08-16T00:00:00.000000000'),
Text(107, 0, '2014-08-17T00:00:00.000000000'),
Text(108, 0, '2014-08-18T00:00:00.000000000'),
Text(109, 0, '2014-08-19T00:00:00.000000000'),
Text(110, 0, '2014-08-20T00:00:00.000000000'),
Text(111, 0, '2014-08-21T00:00:00.000000000'),
Text(112, 0, '2014-08-22T00:00:00.000000000'),
```

Interesting findings

- Very few houses get closed on the weekend, at the same time closing date is arbitary since there is so much that goes on in order to close a house.
- There is a fluxuation of number of homes sold theought out these 2 years. shouls group by month to see if there is any futher trends
- The range og the data is from 05-2014 to 05-2015

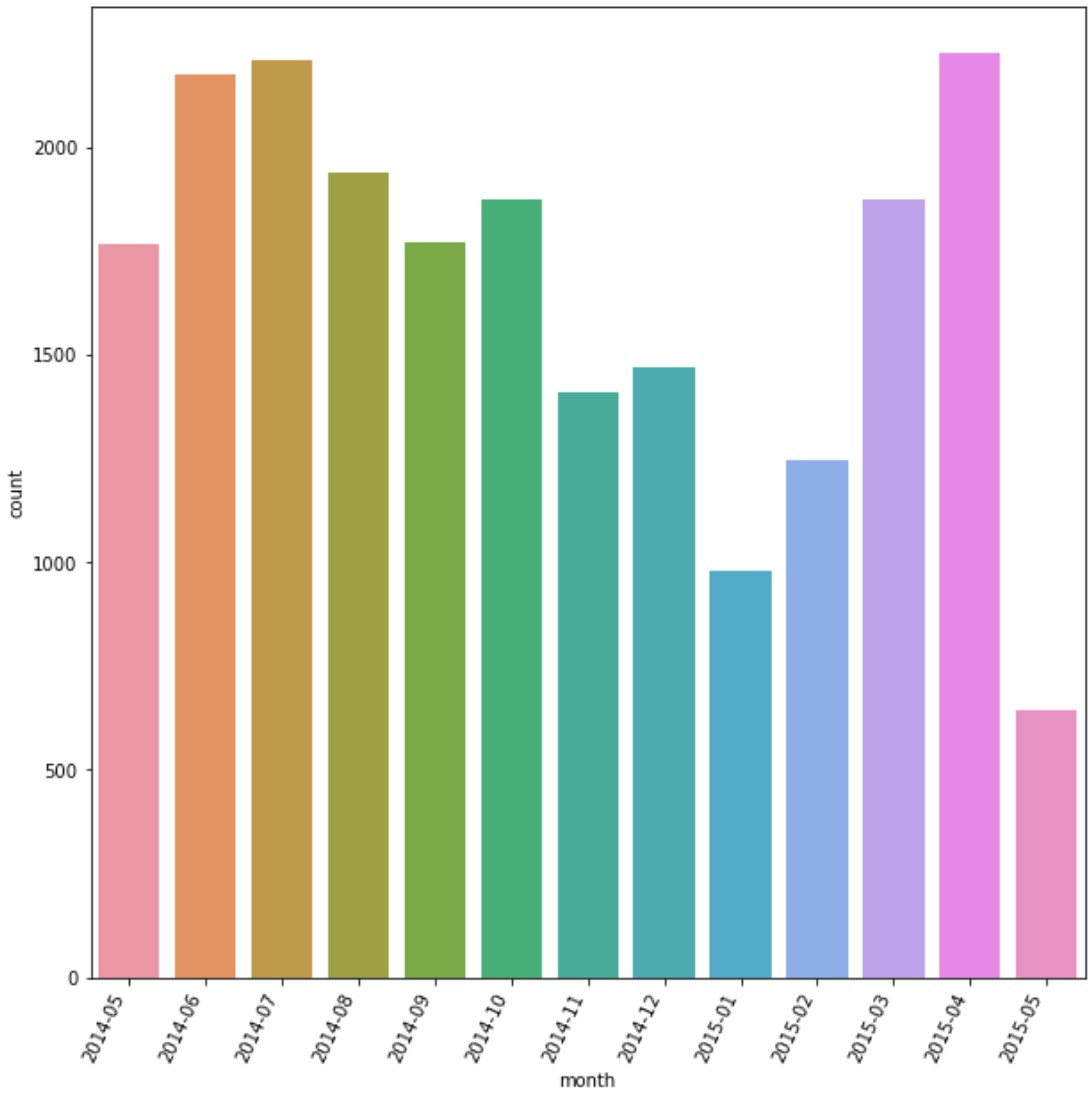
In [12]:
```
df['month_year'] = pd.to_datetime(df_date['date']).dt.strftime('%Y%m')
df_date['month'] = pd.to_datetime(df_date['date']).dt.strftime('%Y-%m')
df['date'] = pd.to_datetime(df['date']).dt.strftime('%Y-%m')

```

executed in 1.36s, finished 20:54:43 2021-06-12

In [13]:
```
fig, ax = plt.subplots(figsize=(10,10))
sns.countplot(x ='month', ax = ax, data = df_date)

plt.xticks(rotation=65, horizontalalignment='right')
```

executed in 594ms, finished 20:54:44 2021-06-12

Out[13]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12]),
 [Text(0, 0, '2014-05'),
  Text(1, 0, '2014-06'),
  Text(2, 0, '2014-07'),
  Text(3, 0, '2014-08'),
  Text(4, 0, '2014-09'),
  Text(5, 0, '2014-10'),
  Text(6, 0, '2014-11'),
  Text(7, 0, '2014-12'),
  Text(8, 0, '2015-01'),
  Text(9, 0, '2015-02'),
  Text(10, 0, '2015-03'),
  Text(11, 0, '2015-04'),
  Text(12, 0, '2015-05')])
```

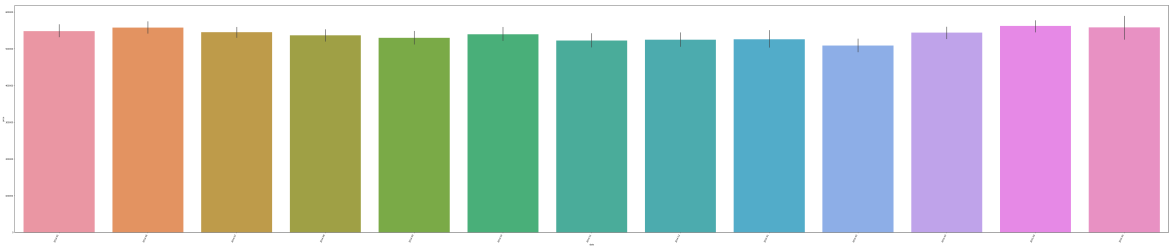


- It look like as the months move closer to the summer time there are more homes being sold.
- It looke like the lost month had a surprising low amount of houses sold for it to be the summer but that could be the result of incomplete data, since the time between when people close on there home and when that information is public has a time gap.
- should in investigate next if time of year had an effect on price.

In [14]:
```
1 dff = df[['price', 'date']].sort_values('date')
```
executed in 64ms, finished 20:54:44 2021-06-12

In [15]:
```
1 figure, ax= plt.subplots(figsize = (100,20))
2 sns.barplot(x = 'date', y = 'price', data = dff)
3 plt.xticks(rotation=65, horizontalalignment='right')
```
executed in 2.70s, finished 20:54:47 2021-06-12

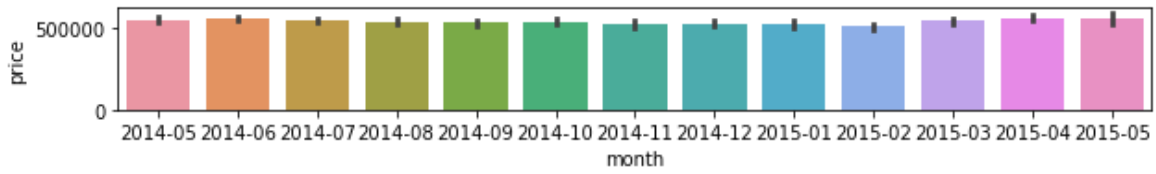
Out[15]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12]),
 [Text(0, 0, '2014-05'),
  Text(1, 0, '2014-06'),
  Text(2, 0, '2014-07'),
  Text(3, 0, '2014-08'),
  Text(4, 0, '2014-09'),
  Text(5, 0, '2014-10'),
  Text(6, 0, '2014-11'),
  Text(7, 0, '2014-12'),
  Text(8, 0, '2015-01'),
  Text(9, 0, '2015-02'),
  Text(10, 0, '2015-03'),
  Text(11, 0, '2015-04'),
  Text(12, 0, '2015-05')])
```



In [16]:
```
figure, ax= plt.subplots(figsize = (10,1))
sns.barplot(x = 'month', y = 'price', data = df_date)
```

executed in 1.81s, finished 20:54:49 2021-06-12

Out[16]: <AxesSubplot:xlabel='month', ylabel='price'>



- The above graph shows that homes sell for slightly more in the summer months. Note there was a peek in October which is a result of the home(s) sold on October 11, 2014 which totaled about 2 million dollars.
- Since the month_year column is a good representiation of homes sold over time the date column is redundant and will be droped.

In [17]:
```
df = df.drop('date', axis = 1)
```

executed in 24ms, finished 20:54:49 2021-06-12

In [18]:
```
1 convert_int(df, 'month_year')
```
executed in 64ms, finished 20:54:49 2021-06-12

```
<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   price          21597 non-null  float64
 2   bedrooms       21597 non-null  int64  
 3   bathrooms      21597 non-null  float64
 4   sqft_living    21597 non-null  int64  
 5   sqft_lot       21597 non-null  int64  
 6   floors         21597 non-null  float64
 7   waterfront     19221 non-null  float64
 8   view           21534 non-null  float64
 9   condition      21597 non-null  int64  
 10  grade          21597 non-null  int64  
 11  sqft_above     21597 non-null  int64  
 12  sqft_basement  21597 non-null  object 
 13  yr_built       21597 non-null  int64  
 14  yr_renovated   17755 non-null  float64
 15  zipcode        21597 non-null  int64  
 16  lat            21597 non-null  float64
 17  long           21597 non-null  float64
 18  sqft_living15  21597 non-null  int64  
 19  sqft_lot15     21597 non-null  int64  
 20  month_year     21597 non-null  int32  
dtypes: float64(8), int32(1), int64(11), object(1)
memory usage: 3.4+ MB
```

In [19]:
```
1 df['sqft_basement'].replace(to_replace =["?"], value ="0", inplace=True)
2 #df[['sqft_basement']].value_counts(ascending=False)
```
executed in 18ms, finished 20:54:49 2021-06-12

In [20]:
```
df[['sqft_basement']].value_counts(ascending=False)
```

executed in 37ms, finished 20:54:49 2021-06-12

Out[20]:
```
sqft_basement
0.0              12826
0                  454
600.0              217
500.0              209
700.0              208
                 ...
225.0                1
2240.0               1
2196.0               1
2190.0               1
2180.0               1
Length: 304, dtype: int64
```

In [21]:
```
convert_float(df, 'sqft_basement')
df.info()
```

executed in 73ms, finished 20:54:49 2021-06-12

```
<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   price          21597 non-null  float64
 2   bedrooms       21597 non-null  int64  
 3   bathrooms      21597 non-null  float64
 4   sqft_living    21597 non-null  int64  
 5   sqft_lot       21597 non-null  int64  
 6   floors         21597 non-null  float64
 7   waterfront     19221 non-null  float64
 8   view           21534 non-null  float64
 9   condition      21597 non-null  int64  
 10  grade          21597 non-null  int64  
 11  sqft_above     21597 non-null  int64  
 12  sqft_basement  21597 non-null  float64
 13  yr_built       21597 non-null  int64  
 14  yr_renovated   17755 non-null  float64
 15  zipcode        21597 non-null  int64  
 16  lat            21597 non-null  float64
 17  long           21597 non-null  float64
 18  sqft_living15  21597 non-null  int64  
 19  sqft_lot15     21597 non-null  int64  
 20  month_year     21597 non-null  int32  
dtypes: float64(9), int32(1), int64(11)
memory usage: 3.4 MB
```

In [22]:

```
df_sqft = df[['sqft_living', 'sqft_lot', 'sqft_above', 'sqft_basement',
              'sqft_living15', 'sqft_lot15']]
df_sqft.info()
```

executed in 46ms, finished 20:54:49 2021-06-12

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   sqft_living    21597 non-null  int64  
 1   sqft_lot       21597 non-null  int64  
 2   sqft_above     21597 non-null  int64  
 3   sqft_basement  21597 non-null  float64
 4   sqft_living15  21597 non-null  int64  
 5   sqft_lot15     21597 non-null  int64  
dtypes: float64(1), int64(5)
memory usage: 1012.5 KB
```

In [23]:

```
df_sqft['base_above'] = df_sqft['sqft_above'] + df_sqft['sqft_basement']
df_sqft[['sqft_living', 'base_above']]
```

executed in 66ms, finished 20:54:49 2021-06-12

```
<ipython-input-23-701d7f16f4a9>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df_sqft['base_above'] = df_sqft['sqft_above'] + df_sqft['sqft_basement']
```

Out[23]:

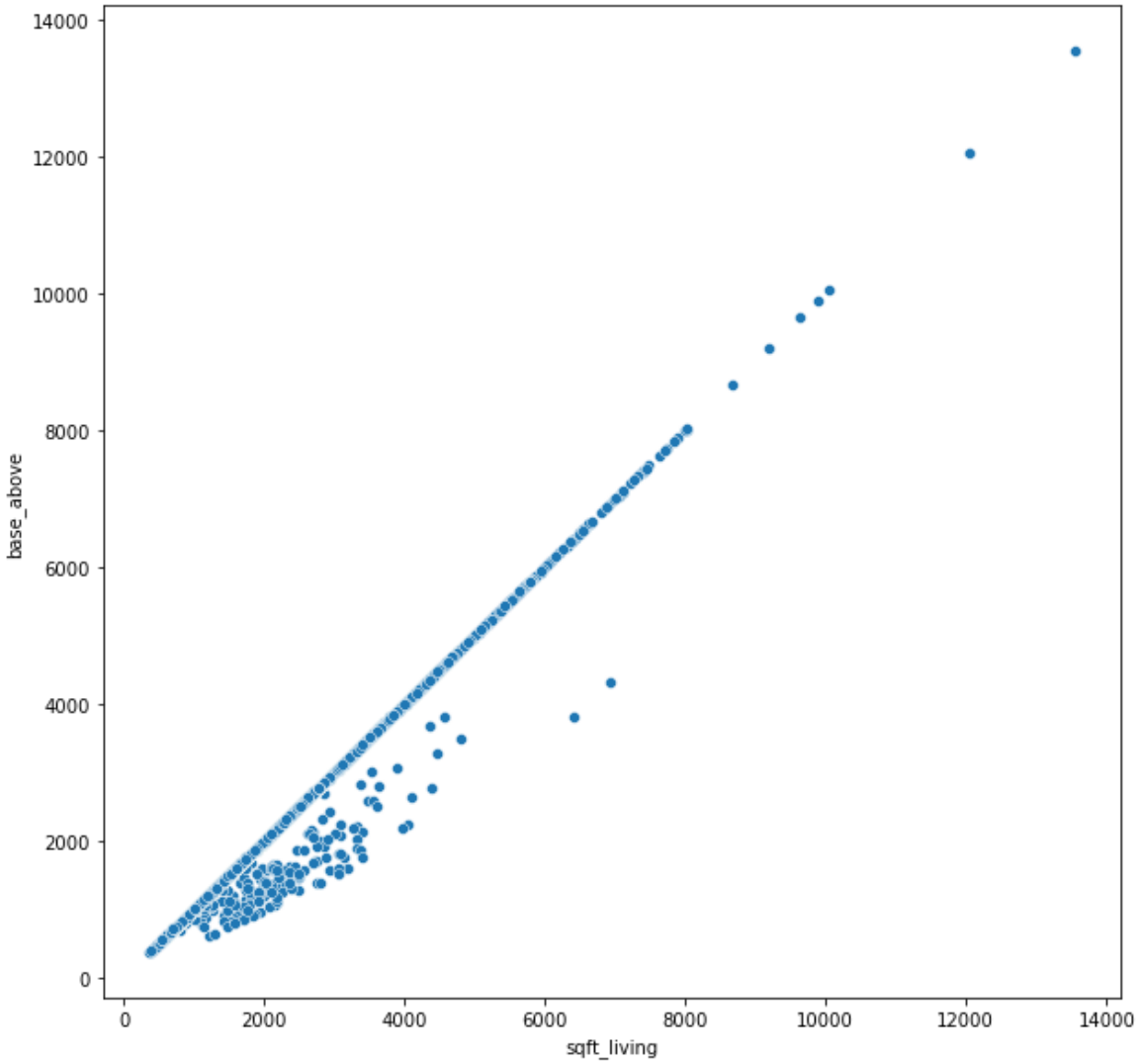
	sqft_living	base_above
0	1180	1180.0
1	2570	2570.0
2	770	770.0
3	1960	1960.0
4	1680	1680.0
...	...	...
21592	1530	1530.0
21593	2310	2310.0
21594	1020	1020.0
21595	1600	1600.0
21596	1020	1020.0

21597 rows × 2 columns

In [24]:
```
figure, ax= plt.subplots(figsize = (10,10))
sns.scatterplot( x = 'sqft_living', y = 'base_above' , data = df_sqft)
```

executed in 625ms, finished 20:54:50 2021-06-12

Out[24]: <AxesSubplot:xlabel='sqft_living', ylabel='base_above'>



from the chart and from the graph it shows that sqft_above and sqft_basement add together to equal sqft living. The ones that are below the line are the homes that didnt have sqft_basment listed. From here sqft_basement can be recalculated to the correct values by subtracting sqft_above from sdft_living.

In [25]:
```
df['new_sqft_basement'] = df['sqft_living'] - df['sqft_above']
df.info()
```

executed in 49ms, finished 20:54:50 2021-06-12

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 22 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   id                 21597 non-null  int64  
 1   price              21597 non-null  float64
 2   bedrooms           21597 non-null  int64  
 3   bathrooms          21597 non-null  float64
 4   sqft_living        21597 non-null  int64  
 5   sqft_lot           21597 non-null  int64  
 6   floors             21597 non-null  float64
 7   waterfront         19221 non-null  float64
 8   view               21534 non-null  float64
 9   condition          21597 non-null  int64  
 10  grade              21597 non-null  int64  
 11  sqft_above         21597 non-null  int64  
 12  sqft_basement      21597 non-null  float64
 13  yr_built           21597 non-null  int64  
 14  yr_renovated       17755 non-null  float64
 15  zipcode            21597 non-null  int64  
 16  lat                21597 non-null  float64
 17  long               21597 non-null  float64
 18  sqft_living15      21597 non-null  int64  
 19  sqft_lot15         21597 non-null  int64  
 20  month_year         21597 non-null  int32  
 21  new_sqft_basement  21597 non-null  int64  
dtypes: float64(9), int32(1), int64(12)
memory usage: 3.5 MB
```

In [26]:
```
df_sqft = df[['sqft_living', 'sqft_lot', 'sqft_above', 'sqft_basement',
              'sqft_living15', 'sqft_lot15', 'new_sqft_basement']]
```

executed in 17ms, finished 20:54:50 2021-06-12

In [27]:

```
df_sqft['new_base_above'] = df_sqft['sqft_above']
+ df_sqft['new_sqft_basement']
```

executed in 38ms, finished 20:54:50 2021-06-12

```
<ipython-input-27-0736299d0b35>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df_sqft['new_base_above'] = df_sqft['sqft_above']
```

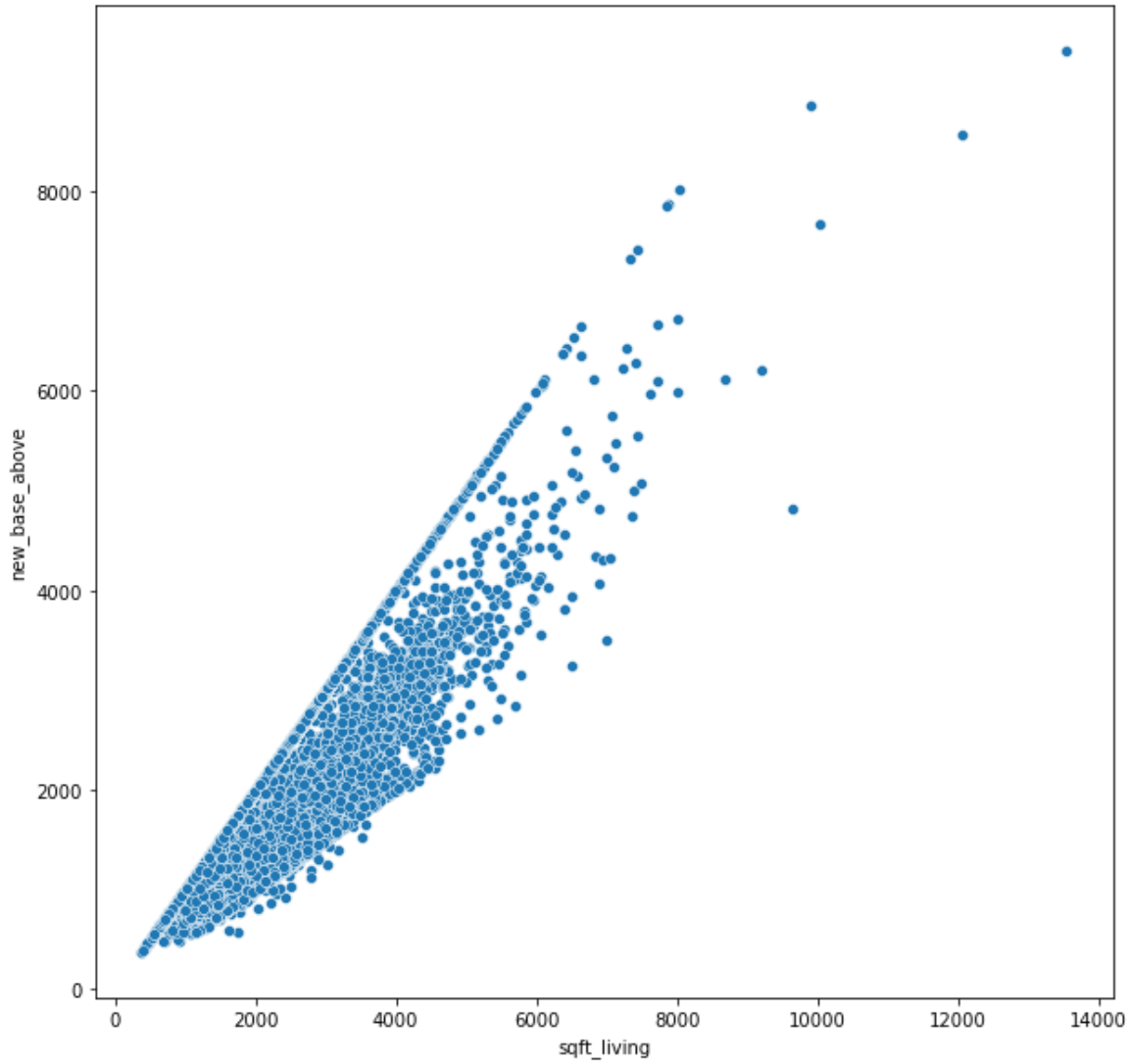
Out[27]:

```
0          0
1        400
2          0
3        910
4          0
        ...
21592      0
21593      0
21594      0
21595      0
21596      0
Name: new_sqft_basement, Length: 21597, dtype: int64
```

In [28]:
```
figure, ax= plt.subplots(figsize = (10,10))
sns.scatterplot( x = 'sqft_living' , y ='new_base_above' , data = df_sqft
```

executed in 728ms, finished 20:54:50 2021-06-12

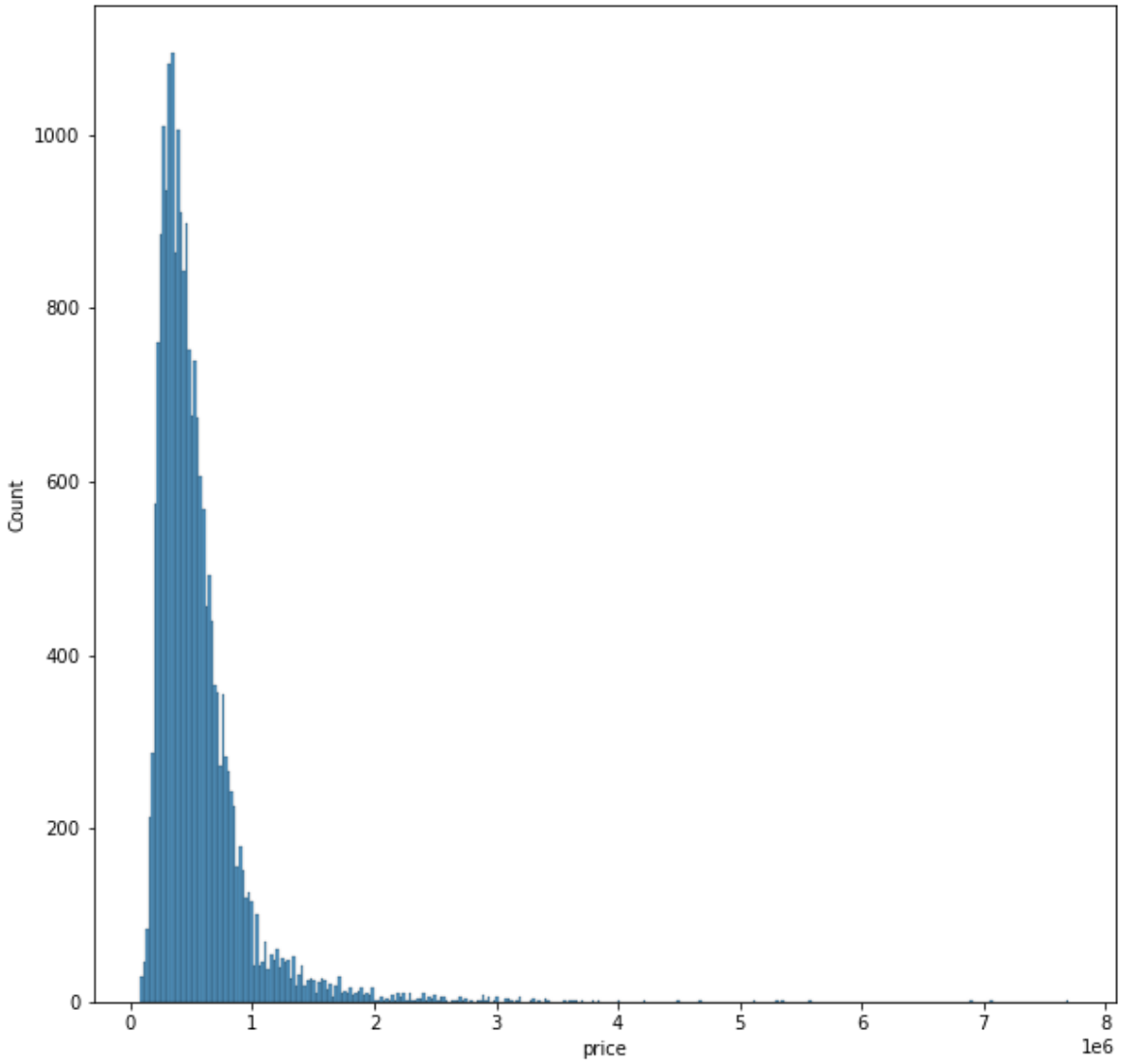
Out[28]: <AxesSubplot:xlabel='sqft_living', ylabel='new_base_above'>



In [29]:
```
1 abs = df[['id', 'price']]
```
executed in 24ms, finished 20:54:51 2021-06-12

In [30]:
```
1 figure, ax= plt.subplots(figsize = (10,10))
2 sns.histplot(data = df, x = df['price'])
```
executed in 1.82s, finished 20:54:52 2021-06-12

Out[30]: <AxesSubplot:xlabel='price', ylabel='Count'>



sqft living can now be droped since is already represent with sqft_above and sqft_basemant.

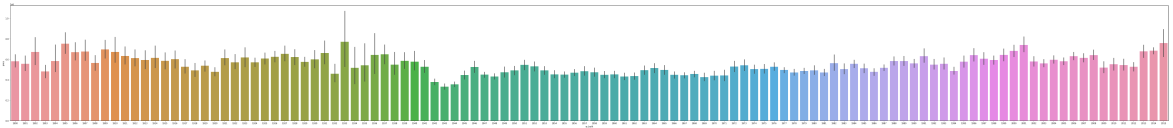
In [31]:
```
df = df.drop('sqft_living', axis = 1)
```
executed in 20ms, finished 20:54:52 2021-06-12

In [32]:
```
df = df.drop('sqft_basement', axis = 1)
df.info()
```
executed in 52ms, finished 20:54:52 2021-06-12

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 20 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   id                 21597 non-null  int64  
 1   price              21597 non-null  float64
 2   bedrooms           21597 non-null  int64  
 3   bathrooms          21597 non-null  float64
 4   sqft_lot           21597 non-null  int64  
 5   floors             21597 non-null  float64
 6   waterfront         19221 non-null  float64
 7   view               21534 non-null  float64
 8   condition          21597 non-null  int64  
 9   grade              21597 non-null  int64  
 10  sqft_above         21597 non-null  int64  
 11  yr_built           21597 non-null  int64  
 12  yr_renovated       17755 non-null  float64
 13  zipcode            21597 non-null  int64  
 14  lat                21597 non-null  float64
 15  long               21597 non-null  float64
 16  sqft_living15      21597 non-null  int64  
 17  sqft_lot15         21597 non-null  int64  
 18  month_year         21597 non-null  int32  
 19  new_sqft_basement  21597 non-null  int64  
dtypes: float64(8), int32(1), int64(11)
memory usage: 3.2 MB
```

In [33]:
```

figure, ax= plt.subplots(figsize = (100,10))
sns.barplot( x = df['yr_built'] , y ='price' , data = df )


```
executed in 14.0s, finished 20:55:06 2021-06-12

Out[33]: <AxesSubplot:xlabel='yr_built', ylabel='price'>



In [34]:
```
1 df.describe()
```
executed in 343ms, finished 20:55:07 2021-06-12

Out[34]:

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

In [35]:
```
1 257/21597
```
executed in 23ms, finished 20:55:07 2021-06-12

Out[35]: 0.011899800898272908

In [36]:

```
df_count = df.loc[df['yr_renovated'] > 2004]
df_count.info()
```

executed in 30ms, finished 20:55:07 2021-06-12

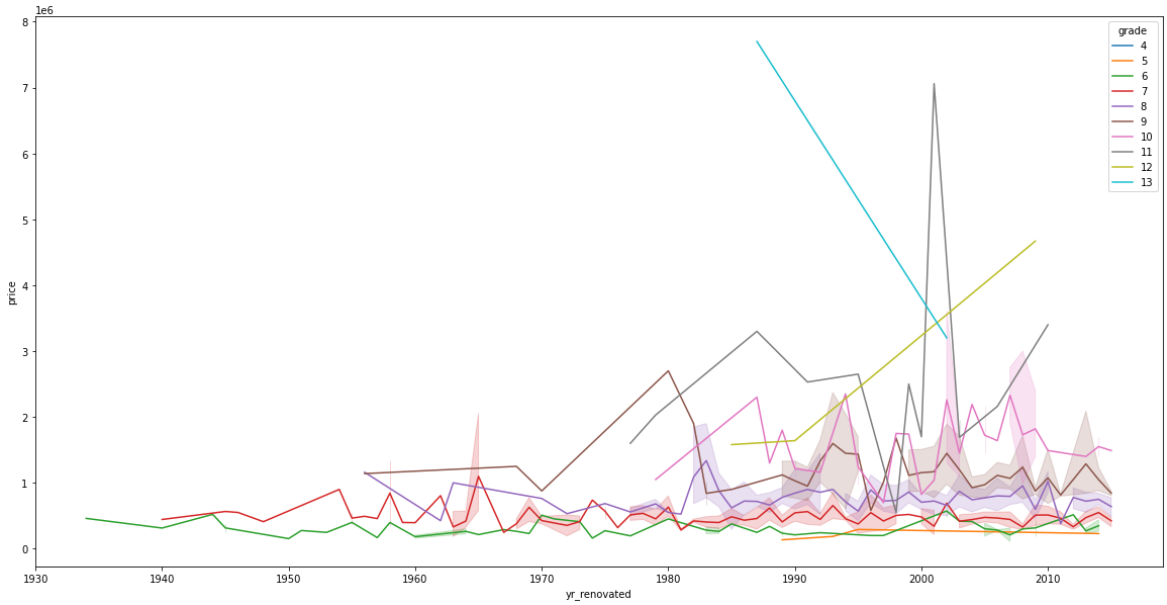
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 265 entries, 103 to 20946
Data columns (total 20 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   id                 265 non-null    int64  
 1   price              265 non-null    float64
 2   bedrooms           265 non-null    int64  
 3   bathrooms          265 non-null    float64
 4   sqft_lot           265 non-null    int64  
 5   floors             265 non-null    float64
 6   waterfront         237 non-null    float64
 7   view               265 non-null    float64
 8   condition          265 non-null    int64  
 9   grade              265 non-null    int64  
 10  sqft_above         265 non-null    int64  
 11  yr_built           265 non-null    int64  
 12  yr_renovated       265 non-null    float64
 13  zipcode            265 non-null    int64  
 14  lat                265 non-null    float64
 15  long               265 non-null    float64
 16  sqft_living15      265 non-null    int64  
 17  sqft_lot15         265 non-null    int64  
 18  month_year         265 non-null    int32  
 19  new_sqft_basement  265 non-null    int64  
dtypes: float64(8), int32(1), int64(11)
memory usage: 42.4 KB
```

In [37]:

```
df1 = df.loc[df['yr_renovated'] > 1]
figure, ax= plt.subplots(figsize = (20,10))
sns.lineplot( x = 'yr_renovated', y ='price' , hue='grade', legend=True,
                palette="tab10",  data = df1 )
```

executed in 7.77s, finished 20:55:15 2021-06-12

Out[37]: <AxesSubplot:xlabel='yr_renovated', ylabel='price'>

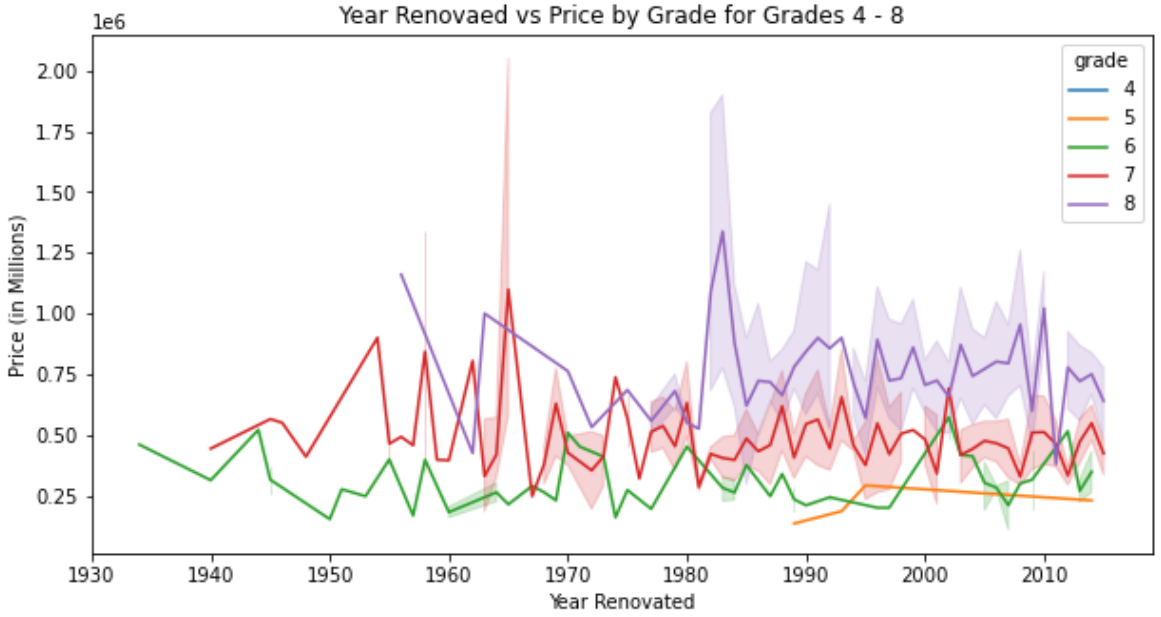


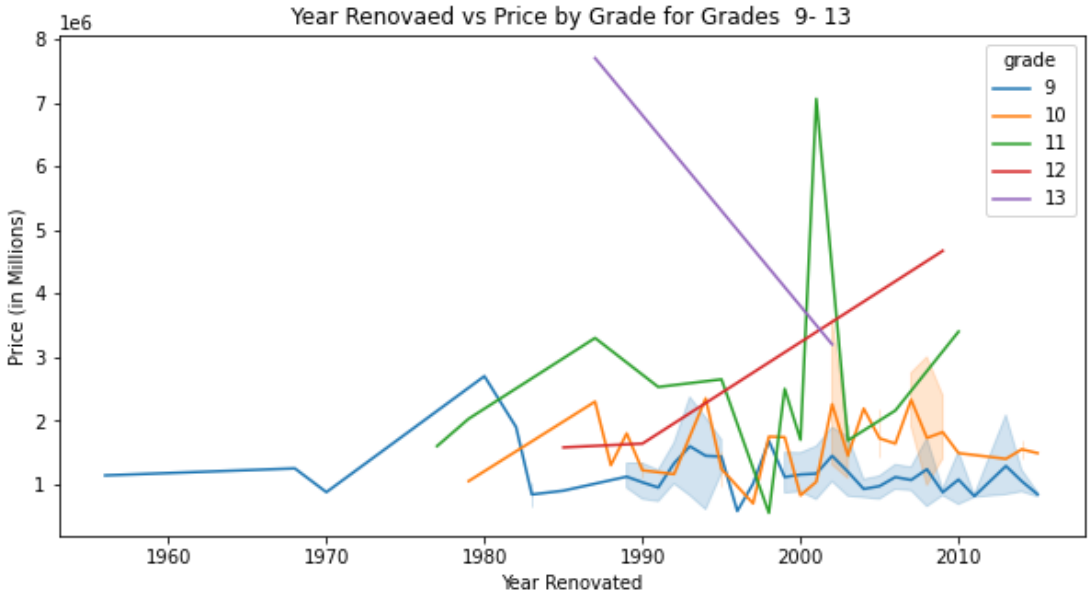
In [38]:

```
df1 = df.loc[df['yr_renovated'] > 1]
df_grade_under_nine = df1.loc[df1['grade'] < 9]
figure, ax= plt.subplots(figsize = (10, 5))
sns.lineplot( x = 'yr_renovated', y ='price' , hue='grade', legend=True,
                palette="tab10",  data = df_grade_under_nine )
plt.title('Year Renovaed vs Price by Grade for Grades 4 - 8')
plt.xlabel('Year Renovated')
plt.ylabel('Price (in Millions)')

df1_2 = df.loc[df['yr_renovated'] > 1]
df_grade_under_nine_2 = df1_2.loc[df1_2['grade'] > 8]
figure, ax= plt.subplots(figsize = (10,5))
sns.lineplot( x = 'yr_renovated', y ='price' , hue='grade', legend=True,
                palette="tab10",  data = df_grade_under_nine_2 )
plt.title('Year Renovaed vs Price by Grade for Grades  9- 13')
plt.xlabel('Year Renovated')
plt.ylabel('Price (in Millions)')

```

executed in 8.19s, finished 20:55:23 2021-06-12

Out[38]: Text(0, 0.5, 'Price (in Millions)')




Year Renovaed vs Price by Grade for Grades 9- 13
1e6
8
7
6
5
4
3
2
1
Price (in Millions)
1960
1970
1980
1990
2000
2010
Year Renovated
grade
9
10
11
12
13

In [39]:

```
ddd = df.loc[df['grade'] == 9]
print(ddd)
```

executed in 47ms, finished 20:55:23 2021-06-12

```
                id      price  bedrooms  bathrooms  sqft_lot  floors  \
15      9297300055   650000.0         4       3.00      5000     2.0
21      2524049179  2000000.0         3       2.75     44867     1.0
40      5547700270   625000.0         4       2.50      5520     2.0
42      7203220400   861990.0         5       2.75      5639     2.0
47      4178300310   785000.0         4       2.50     13416     2.0
...            ...        ...       ...        ...       ...     ...
21580   7502800100   679950.0         5       2.75      9437     2.0
21582   8956200760   541800.0         4       2.50      7866     2.0
21583   7202300110   810000.0         4       3.00      7838     2.0
21589   3448900210   610685.0         4       2.50      6023     2.0
21590   7936000429  1010000.0         4       3.50      7200     2.0

       waterfront  view  condition  grade  sqft_above  yr_built  yr_renovat
ed  \
15            0.0   3.0          3      9        1980      1979
0.0
21            0.0   4.0          3      9        2330      1968
0.0
40            NaN   0.0          3      9        2570      2000           N
aN
42            0.0   0.0          3      9        3595      2014
0.0
47            0.0   0.0          4      9        2290      1981
0.0
...           ...   ...        ...    ...         ...       ...
...
21580         0.0   0.0          3      9        3600      2014
0.0
21582         NaN   2.0          3      9        3118      2014
0.0
21583         0.0   0.0          3      9        3990      2003           N
aN
21589         0.0   NaN          3      9        2520      2014
0.0
21590         0.0   0.0          3      9        2600      2009
0.0

       zipcode      lat     long  sqft_living15  sqft_lot15  month_year  \
15       98126  47.5714 -122.375           2140        4000      201501
21       98040  47.5316 -122.233           4110       20336      201408
40       98074  47.6145 -122.027           2470        5669      201407
42       98053  47.6848 -122.016           3625        5639      201407
47       98007  47.6194 -122.151           2680       13685      201407
...        ...      ...      ...            ...         ...         ...
21580    98059  47.4822 -122.131           3550        9421      201408
21582    98001  47.2931 -122.264           2673        6500      201410
21583    98053  47.6857 -122.046           3370        6814      201409
21589    98056  47.5137 -122.167           2520        6023      201410
21590    98136  47.5537 -122.398           2050        6200      201503

       new_sqft_basement
```

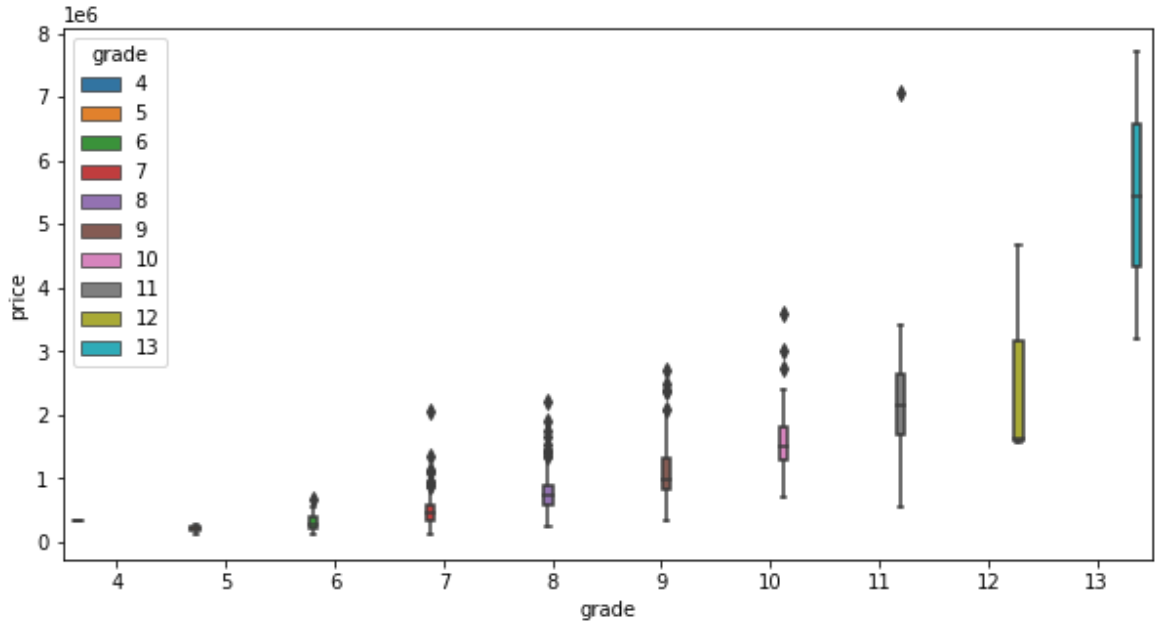
```
15                   970
21                   720
40                     0
42                     0
47                     0
...                  ...
21580                  0
21582                  0
21583                  0
21589                  0
21590                910

[2615 rows x 20 columns]
```

In [40]:

```
df1 = df.loc[df['yr_renovated'] > 1]
df_grade_under_nine = df1.loc[df1['grade'] < 14]
figure, ax= plt.subplots(figsize = (10, 5))
sns.boxplot( x = 'grade', y ='price' , hue='grade',
                palette="tab10",  data = df_grade_under_nine )

```

executed in 1.36s, finished 20:55:24 2021-06-12

Out[40]: <AxesSubplot:xlabel='grade', ylabel='price'>



In [41]:
```
df_columns = ddd.columns
for i in df_columns:
    print('\033[1m' + i.upper() + '\033[0m')
    print(ddd[[i]].value_counts(ascending=False))
    print('_____________________________________________________')
```

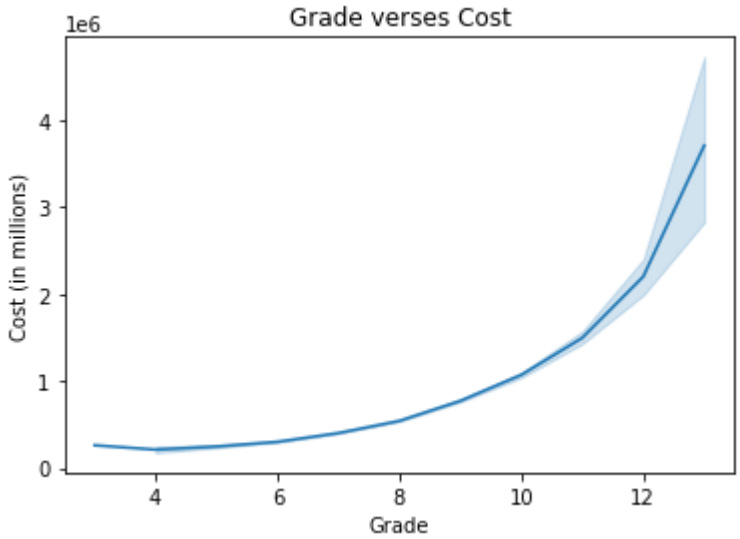
executed in 229ms, finished 20:55:24 2021-06-12

```
ID
id
7504021310    2
9809000020    2
2619920170    2
7853420110    2
7853400250    2
             ..
6635000110    1
6632300567    1
6613000750    1
6613000715    1
5200087       1
Length: 2606, dtype: int64

_____________________________________________________
PRICE
price
650000.0    25
850000.0    25
```

In [42]:
```
sns.lineplot( x = 'grade', y ='price' , legend=True,
                palette="deep",  data = df )
plt.title('Grade verses Cost')
plt.xlabel('Grade')
plt.ylabel('Cost (in millions)')
```

executed in 1.61s, finished 20:55:26 2021-06-12

Out[42]: Text(0, 0.5, 'Cost (in millions)')



from the above line graph it looks like rennovating homes to a certain level will have result in a

higher sell price. you can also see the graph's price takes off at around the grade 8 mark

In [43]:
```
grade1 = df.loc[(df["grade"] <= 8)]
grade1[['price']].describe()
```

executed in 49ms, finished 20:55:26 2021-06-12

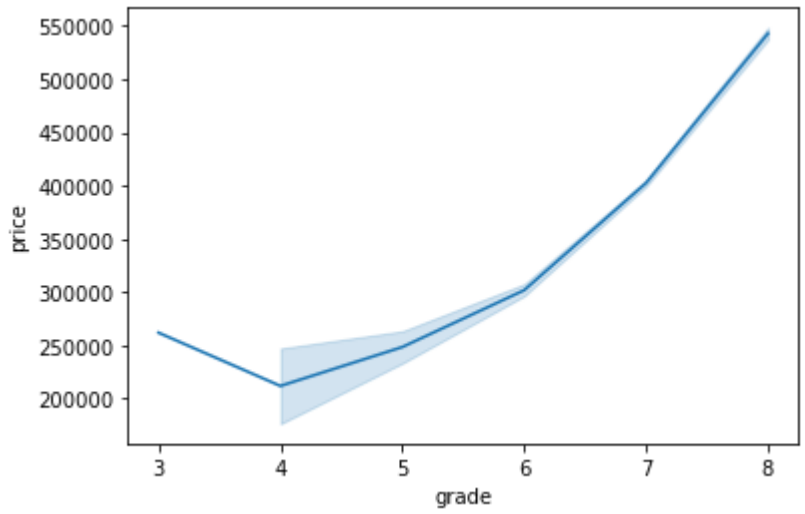
Out[43]:

	price
count	1.734700e+04
mean	4.373974e+05
std	1.958205e+05
min	7.800000e+04
25%	2.990000e+05
50%	4.015000e+05
75%	5.390000e+05
max	3.070000e+06

In [44]:
```
sns.lineplot( x = 'grade', y ='price' , legend=True,
                palette="deep",  data = grade1 )
```

executed in 1.12s, finished 20:55:27 2021-06-12

Out[44]: <AxesSubplot:xlabel='grade', ylabel='price'>



In [45]:
```
grade2 = df.loc[(df["grade"] > 8)]
grade2[['price']].describe()
```

executed in 37ms, finished 20:55:27 2021-06-12

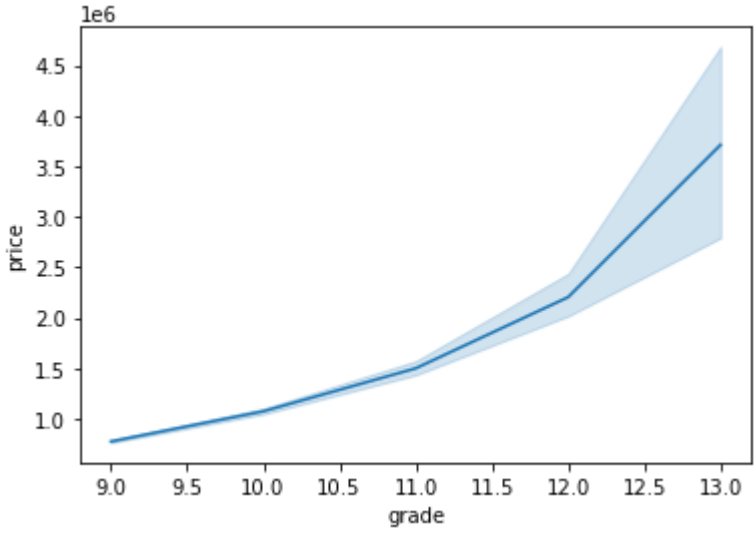
Out[45]:

	price
count	4.250000e+03
mean	9.602945e+05
std	5.565400e+05
min	2.300000e+05
25%	6.423508e+05
50%	8.100000e+05
75%	1.100000e+06
max	7.700000e+06

In [46]:
```
sns.lineplot( x = 'grade', y ='price' , legend=True,
                palette="deep",  data = grade2 )
```

executed in 892ms, finished 20:55:28 2021-06-12

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



In [47]:
```
1 df['bedrooms'].describe()
```
executed in 38ms, finished 20:55:28 2021-06-12

Out[47]:
```
count    21597.000000
mean         3.373200
std          0.926299
min          1.000000
25%          3.000000
50%          3.000000
75%          4.000000
max         33.000000
Name: bedrooms, dtype: float64
```

The mean of bed rooms is between 3 and 4 room with a STD of less than one. Also a home have 33 roomes seems very unlikely

In [48]:
```
1 df.loc[df['bedrooms'] == 33]
```
executed in 48ms, finished 20:55:28 2021-06-12

Out[48]:

	id	price	bedrooms	bathrooms	sqft_lot	floors	waterfront	view	conditi
15856	2402100895	640000.0	33	1.75	6000	1.0	0.0	0.0	

For a home to have 33 rooms with 1620 sqsf of living space gives about 49 sqsf of space per room and 1 shower does not make since as a home wo that value will bedroped

In [49]:

```
df = df.loc[(df["bedrooms"] < 33)]
#df = df.drop(['bedrooms'] == 33, axis = 0)
df.info()
```

executed in 40ms, finished 20:55:28 2021-06-12

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21596 entries, 0 to 21596
Data columns (total 20 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   id                 21596 non-null  int64  
 1   price              21596 non-null  float64
 2   bedrooms           21596 non-null  int64  
 3   bathrooms          21596 non-null  float64
 4   sqft_lot           21596 non-null  int64  
 5   floors             21596 non-null  float64
 6   waterfront         19220 non-null  float64
 7   view               21533 non-null  float64
 8   condition          21596 non-null  int64  
 9   grade              21596 non-null  int64  
 10  sqft_above         21596 non-null  int64  
 11  yr_built           21596 non-null  int64  
 12  yr_renovated       17754 non-null  float64
 13  zipcode            21596 non-null  int64  
 14  lat                21596 non-null  float64
 15  long               21596 non-null  float64
 16  sqft_living15      21596 non-null  int64  
 17  sqft_lot15         21596 non-null  int64  
 18  month_year         21596 non-null  int32  
 19  new_sqft_basement  21596 non-null  int64  
dtypes: float64(8), int32(1), int64(11)
memory usage: 3.4 MB
```

we are not looking for mansion style homes how homes with 6 or more homes will be droped

In [50]:

```
df = df.loc[(df["bedrooms"] < 6)]
df.info()
```

executed in 37ms, finished 20:55:28 2021-06-12

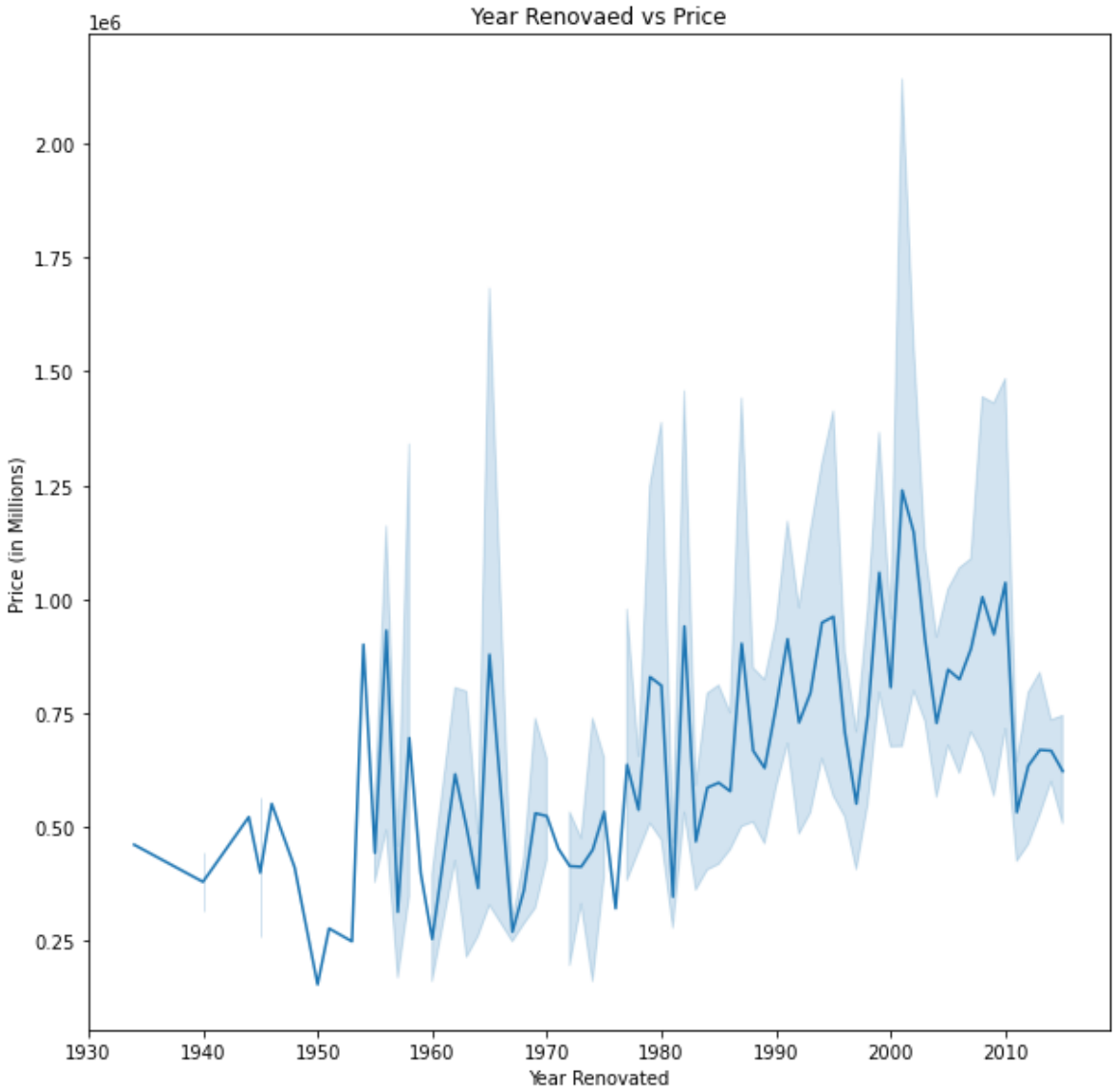
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21263 entries, 0 to 21596
Data columns (total 20 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   id                 21263 non-null  int64  
 1   price              21263 non-null  float64
 2   bedrooms           21263 non-null  int64  
 3   bathrooms          21263 non-null  float64
 4   sqft_lot           21263 non-null  int64  
 5   floors             21263 non-null  float64
 6   waterfront         18937 non-null  float64
 7   view               21202 non-null  float64
 8   condition          21263 non-null  int64  
 9   grade              21263 non-null  int64  
 10  sqft_above         21263 non-null  int64  
 11  yr_built           21263 non-null  int64  
 12  yr_renovated       17473 non-null  float64
 13  zipcode            21263 non-null  int64  
 14  lat                21263 non-null  float64
 15  long               21263 non-null  float64
 16  sqft_living15      21263 non-null  int64  
 17  sqft_lot15         21263 non-null  int64  
 18  month_year         21263 non-null  int32  
 19  new_sqft_basement  21263 non-null  int64  
dtypes: float64(8), int32(1), int64(11)
memory usage: 3.3 MB
```

In [51]:

```
df123 = df.loc[(df["yr_renovated"] > 1)]
figure, ax= plt.subplots(figsize = (10,10))
sns.lineplot( x = df123['yr_renovated'] , y ='price' , data = df123 )
plt.title('Year Renovaed vs Price')
plt.xlabel('Year Renovated')
plt.ylabel('Price (in Millions)')


```

executed in 3.74s, finished 20:55:32 2021-06-12

Out[51]: Text(0, 0.5, 'Price (in Millions)')

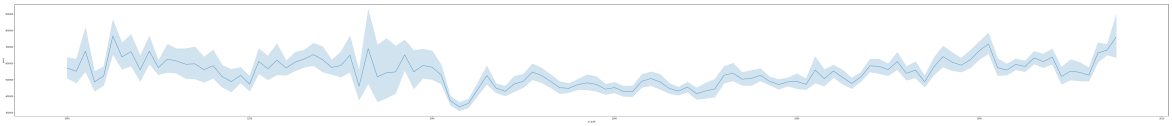


In [52]:

```
figure, ax= plt.subplots(figsize = (100,10))
sns.lineplot( x = df['yr_built'] , y ='price' , data = df )

```

executed in 7.50s, finished 20:55:40 2021-06-12

Out[52]: <AxesSubplot:xlabel='yr_built', ylabel='price'>



In [53]:
```
1 df.loc[(df["price"] >3000000)]
```

executed in 94ms, finished 20:55:40 2021-06-12

Out[53]:

	id	price	bedrooms	bathrooms	sqft_lot	floors	waterfront	view	condi
300	3225069065	3080000.0	4	5.00	18641	1.0	1.0	4.0	
656	3760500116	3070000.0	3	2.50	55867	1.0	1.0	4.0	
1162	1247600105	5110000.0	5	5.25	45517	2.0	1.0	4.0	
1446	8907500070	5350000.0	5	5.00	23985	2.0	0.0	4.0	
2083	8106100105	3850000.0	4	4.25	21300	2.0	1.0	4.0	
2442	7524900003	3280000.0	2	1.75	10000	2.5	1.0	4.0	
2624	7738500731	4500000.0	5	5.50	40014	2.0	1.0	4.0	
2862	4114601570	3600000.0	3	3.25	12431	2.0	1.0	4.0	
3910	9808700762	7060000.0	5	4.50	37325	2.0	1.0	2.0	
4145	6447300265	4000000.0	4	5.50	16573	2.0	0.0	0.0	
4407	2470100110	5570000.0	5	5.75	35069	2.0	0.0	0.0	
5874	2525049148	3420000.0	5	5.00	20412	2.0	0.0	0.0	
7028	853200010	3800000.0	5	5.50	42840	1.0	0.0	2.0	
7306	4131900066	3100000.0	3	3.00	13085	2.0	1.0	4.0	
7982	9362000040	3400000.0	3	4.50	17826	2.0	1.0	4.0	
8085	1924059029	4670000.0	5	6.75	13068	1.0	1.0	4.0	
8629	3835500195	4490000.0	4	3.00	27517	2.0	0.0	0.0	
10435	1118000320	3400000.0	4	4.00	11765	2.0	0.0	0.0	
10454	333100295	3120000.0	3	3.50	56609	2.0	1.0	4.0	
11523	8964800890	3200000.0	3	3.25	13363	1.0	NaN	4.0	
12358	6065300370	4210000.0	5	6.00	21540	2.0	0.0	0.0	
13515	3025059124	3170000.0	5	3.50	11979	1.0	0.0	4.0	
13954	3126059023	3400000.0	4	3.50	47870	1.0	1.0	4.0	
14070	3625059043	3300000.0	5	4.75	13873	2.0	1.0	4.0	
15241	2425049063	3640000.0	4	3.25	22257	2.0	1.0	4.0	
15244	1732800780	3070000.0	5	3.00	7500	2.5	0.0	4.0	
15468	624069108	3200000.0	4	3.25	28206	1.0	1.0	4.0	
16288	7397300170	3710000.0	4	3.50	28078	2.0	0.0	2.0	
16955	3025059093	3100000.0	5	5.25	23669	2.0	0.0	0.0	
18185	3625059152	3300000.0	3	3.25	41300	1.0	1.0	4.0	
18288	6072800246	3300000.0	5	6.25	21738	2.0	0.0	0.0	
18467	4389201095	3650000.0	5	3.75	8694	2.0	0.0	1.0	
19002	2303900100	3800000.0	3	4.25	35000	2.0	0.0	4.0	

	id	price	bedrooms	bathrooms	sqft_lot	floors	waterfront	view	condi
19133	3625049042	3640000.0	5	6.00	19897	2.0	0.0	0.0	
20279	251500080	3200000.0	4	4.00	18851	2.0	0.0	3.0	
20444	9808100150	3350000.0	5	3.75	15360	1.0	0.0	1.0	
21560	9253900271	3570000.0	5	4.50	10584	2.0	1.0	4.0	

In [54]:
```
1 df.columns
```
executed in 23ms, finished 20:55:40 2021-06-12

Out[54]:
```
Index(['id', 'price', 'bedrooms', 'bathrooms', 'sqft_lot', 'floors',
       'waterfront', 'view', 'condition', 'grade', 'sqft_above', 'yr_buil
t',
       'yr_renovated', 'zipcode', 'lat', 'long', 'sqft_living15', 'sqft_lot
15',
       'month_year', 'new_sqft_basement'],
      dtype='object')
```

In [55]:
```
1 df_columns = df.columns
2 for i in df_columns:
3     print('\033[1m' + i.upper() + '\033[0m')
4     print(df[[i]].value_counts(ascending=False))
5     print('_____________________________________________________')
```
executed in 269ms, finished 20:55:40 2021-06-12

```
ID
id
795000620     3
726049190     2
1901600090    2
9809000020    2
3598600049    2
             ..
6306810110    1
6308000020    1
6321000045    1
6324000090    1
1200019       1
Length: 21093, dtype: int64
_____________________________________________________
PRICE
price
350000.0    170
450000.0    168
```

Below is a table detailing each column in the above data table represents. The "X" in the categorical column represents weather the data is catorgorical. If there is no X the data is continious/ numerical.

Ref #	Column	Description	Categorical
1	id	unique identified for a house	
2	dateDate	house was sold	
3	pricePrice	is prediction target	
4	bedroomsNumber	of Bedrooms/House	
5	bathroomsNumber	of bathrooms/bedrooms	
6	sqft_livingsquare	footage of the home	
7	sqft_lotsquare	footage of the lot	
8	floorsTotal	floors (levels) in house	
9	waterfront	House which has a view to a waterfront	X
10	view	Has a view	X
11	condition	How good the condition is (Overall)	X
12	grade	index from 1 to 13, where 1-3 falls short of building construction and design, 7 has an average level of construction and design, and 11-13 have a high quality level of construction and design.	X
13	sqft_above	square footage of house apart from basement	
14	sqft_basement	square footage of the basement	
15	yr_built	Built Year	
16	yr_renovated	Year when house was renovated	X
17	zipcode	zip	X
18	lat	Latitude coordinate	
19	long	Longitude coordinate	
20	sqft_living15	The square footage of interior housing living space for the nearest 15 neighbors	
21	sqft_lot15	The square footage of the land lots of the nearest 15 neighbors	

3.3 Investigating Missing Data

In [56]:
```
df.info()
```
executed in 41ms, finished 20:55:40 2021-06-12

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21263 entries, 0 to 21596
Data columns (total 20 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   id                 21263 non-null  int64  
 1   price              21263 non-null  float64
 2   bedrooms           21263 non-null  int64  
 3   bathrooms          21263 non-null  float64
 4   sqft_lot           21263 non-null  int64  
 5   floors             21263 non-null  float64
 6   waterfront         18937 non-null  float64
 7   view               21202 non-null  float64
 8   condition          21263 non-null  int64  
 9   grade              21263 non-null  int64  
 10  sqft_above         21263 non-null  int64  
 11  yr_built           21263 non-null  int64  
 12  yr_renovated       17473 non-null  float64
 13  zipcode            21263 non-null  int64  
 14  lat                21263 non-null  float64
 15  long               21263 non-null  float64
 16  sqft_living15      21263 non-null  int64  
 17  sqft_lot15         21263 non-null  int64  
 18  month_year         21263 non-null  int32  
 19  new_sqft_basement  21263 non-null  int64  
dtypes: float64(8), int32(1), int64(11)
memory usage: 3.3 MB
```

3.3.1 Missing Data

In [57]:
```
print(str(round(((((df['waterfront'].isna().sum()))/len(df))*100),2))
+ '% Null in waterfront')
print(str(round(((((df['view'].isna().sum()))/len(df))*100),2))
+ '% Null in view')
print(str(round(((((df['yr_renovated'].isna().sum()))/len(df))*100),2))
+ '% Null in yr_renovated')
```
executed in 28ms, finished 20:55:40 2021-06-12

```
10.94% Null in waterfront
0.29% Null in view
17.82% Null in yr_renovated
```

In [58]:
```
1 df[['waterfront','view', 'yr_renovated']].describe()
```
executed in 57ms, finished 20:55:40 2021-06-12

Out[58]:

	waterfront	view	yr_renovated
count	18937.000000	21202.000000	17473.000000
mean	0.007499	0.229601	81.902707
std	0.086271	0.758155	395.959042
min	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000
max	1.000000	4.000000	2015.000000

In [59]:
```
1 df[['waterfront']].value_counts(ascending=False)
```
executed in 41ms, finished 20:55:40 2021-06-12

Out[59]:
```
waterfront
0.0           18795
1.0             142
dtype: int64
```

In [60]:
```
1 df[['view']].value_counts(ascending=False)
```
executed in 24ms, finished 20:55:40 2021-06-12

Out[60]:
```
view
0.0     19154
2.0       930
3.0       492
1.0       324
4.0       302
dtype: int64
```

In [61]:
```
1 df[['yr_renovated']].value_counts(ascending=False)
```

executed in 41ms, finished 20:55:40 2021-06-12

Out[61]:
```
yr_renovated
0.0             16756
2014.0             73
2007.0             30
2013.0             30
2000.0             29
                ...
1976.0              1
1944.0              1
1971.0              1
1934.0              1
1951.0              1
Length: 70, dtype: int64
```

The water front column, view column, yr_renovated have missing data.

- Having a waterfront home is a uncommonand very desireable feature within the housing market, more than 75% of the data that is not waterfront based on the above .describe(). It is hard to imagin some one missing that informations which will result in a much higher competition and sells price. Based on that the assumtion will be that NULL values for waterfront are 0.0.
- Having a home with a view is more common than a water front home since a veiw can included but is not limited to the city, a park, water, art, ect. similar to the water front more than 75% of the data that is not waterfront based on the above .describe(). A home with a veiw is also a desireable feature within the housing market and creates higher competition which will result in a higher sell price. Based on that the assumtion will be that NULL values for veiw are 0.0.
- yr_renovated similar to both the waterfront column and the view column it is a very desireable feature that will drive price and creat competition. with more than 75% of the dats being 0, ie, not renovated it is assumed that the NULL values will be 0.0.

In [62]:

```
1 convert_nan_to_0('waterfront')
```

executed in 44ms, finished 20:55:40 2021-06-12

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21263 entries, 0 to 21596
Data columns (total 20 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   id                 21263 non-null  int64  
 1   price              21263 non-null  float64
 2   bedrooms           21263 non-null  int64  
 3   bathrooms          21263 non-null  float64
 4   sqft_lot           21263 non-null  int64  
 5   floors             21263 non-null  float64
 6   waterfront         21263 non-null  float64
 7   view               21202 non-null  float64
 8   condition          21263 non-null  int64  
 9   grade              21263 non-null  int64  
 10  sqft_above         21263 non-null  int64  
 11  yr_built           21263 non-null  int64  
 12  yr_renovated       17473 non-null  float64
 13  zipcode            21263 non-null  int64  
 14  lat                21263 non-null  float64
 15  long               21263 non-null  float64
 16  sqft_living15      21263 non-null  int64  
 17  sqft_lot15         21263 non-null  int64  
 18  month_year         21263 non-null  int32  
 19  new_sqft_basement  21263 non-null  int64  
dtypes: float64(8), int32(1), int64(11)
memory usage: 3.3 MB
```

In [63]:
```
1 convert_nan_to_0('view')
```

executed in 34ms, finished 20:55:40 2021-06-12

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21263 entries, 0 to 21596
Data columns (total 20 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   id                 21263 non-null  int64  
 1   price              21263 non-null  float64
 2   bedrooms           21263 non-null  int64  
 3   bathrooms          21263 non-null  float64
 4   sqft_lot           21263 non-null  int64  
 5   floors             21263 non-null  float64
 6   waterfront         21263 non-null  float64
 7   view               21263 non-null  float64
 8   condition          21263 non-null  int64  
 9   grade              21263 non-null  int64  
 10  sqft_above         21263 non-null  int64  
 11  yr_built           21263 non-null  int64  
 12  yr_renovated       17473 non-null  float64
 13  zipcode            21263 non-null  int64  
 14  lat                21263 non-null  float64
 15  long               21263 non-null  float64
 16  sqft_living15      21263 non-null  int64  
 17  sqft_lot15         21263 non-null  int64  
 18  month_year         21263 non-null  int32  
 19  new_sqft_basement  21263 non-null  int64  
dtypes: float64(8), int32(1), int64(11)
memory usage: 3.3 MB
```

In [64]:
```
convert_nan_to_0('yr_renovated')
```
executed in 41ms, finished 20:55:40 2021-06-12

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21263 entries, 0 to 21596
Data columns (total 20 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   id                 21263 non-null  int64  
 1   price              21263 non-null  float64
 2   bedrooms           21263 non-null  int64  
 3   bathrooms          21263 non-null  float64
 4   sqft_lot           21263 non-null  int64  
 5   floors             21263 non-null  float64
 6   waterfront         21263 non-null  float64
 7   view               21263 non-null  float64
 8   condition          21263 non-null  int64  
 9   grade              21263 non-null  int64  
 10  sqft_above         21263 non-null  int64  
 11  yr_built           21263 non-null  int64  
 12  yr_renovated       21263 non-null  float64
 13  zipcode            21263 non-null  int64  
 14  lat                21263 non-null  float64
 15  long               21263 non-null  float64
 16  sqft_living15      21263 non-null  int64  
 17  sqft_lot15         21263 non-null  int64  
 18  month_year         21263 non-null  int32  
 19  new_sqft_basement  21263 non-null  int64  
dtypes: float64(8), int32(1), int64(11)
memory usage: 3.3 MB
```

In [65]:
```
print(str(round(((((df['waterfront'].isna().sum()))/len(df))*100),2))
+ '% Null in waterfront')
print(str(round(((((df['view'].isna().sum()))/len(df))*100),2))
+ '% Null in view')
print(str(round(((((df['yr_renovated'].isna().sum()))/len(df))*100),2))
+ '% Null in yr_renovated')
```
executed in 37ms, finished 20:55:40 2021-06-12

```
0.0% Null in waterfront
0.0% Null in view
0.0% Null in yr_renovated
```

4 Feature Engineering

From the data there are quite a few of homes with the flowwing trates that can be turn in to a rennovation tv show.

- With 5 bedroom or less
- Less than a million
- Grade lower than 8

In [66]:

```
1 df.info()
```

executed in 50ms, finished 20:55:40 2021-06-12

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21263 entries, 0 to 21596
Data columns (total 20 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   id                 21263 non-null  int64  
 1   price              21263 non-null  float64
 2   bedrooms           21263 non-null  int64  
 3   bathrooms          21263 non-null  float64
 4   sqft_lot           21263 non-null  int64  
 5   floors             21263 non-null  float64
 6   waterfront         21263 non-null  float64
 7   view               21263 non-null  float64
 8   condition          21263 non-null  int64  
 9   grade              21263 non-null  int64  
 10  sqft_above         21263 non-null  int64  
 11  yr_built           21263 non-null  int64  
 12  yr_renovated       21263 non-null  float64
 13  zipcode            21263 non-null  int64  
 14  lat                21263 non-null  float64
 15  long               21263 non-null  float64
 16  sqft_living15      21263 non-null  int64  
 17  sqft_lot15         21263 non-null  int64  
 18  month_year         21263 non-null  int32  
 19  new_sqft_basement  21263 non-null  int64  
dtypes: float64(8), int32(1), int64(11)
memory usage: 3.3 MB
```

In [67]:
```
1 df.corr()
```
executed in 145ms, finished 20:55:40 2021-06-12

Out[67]:

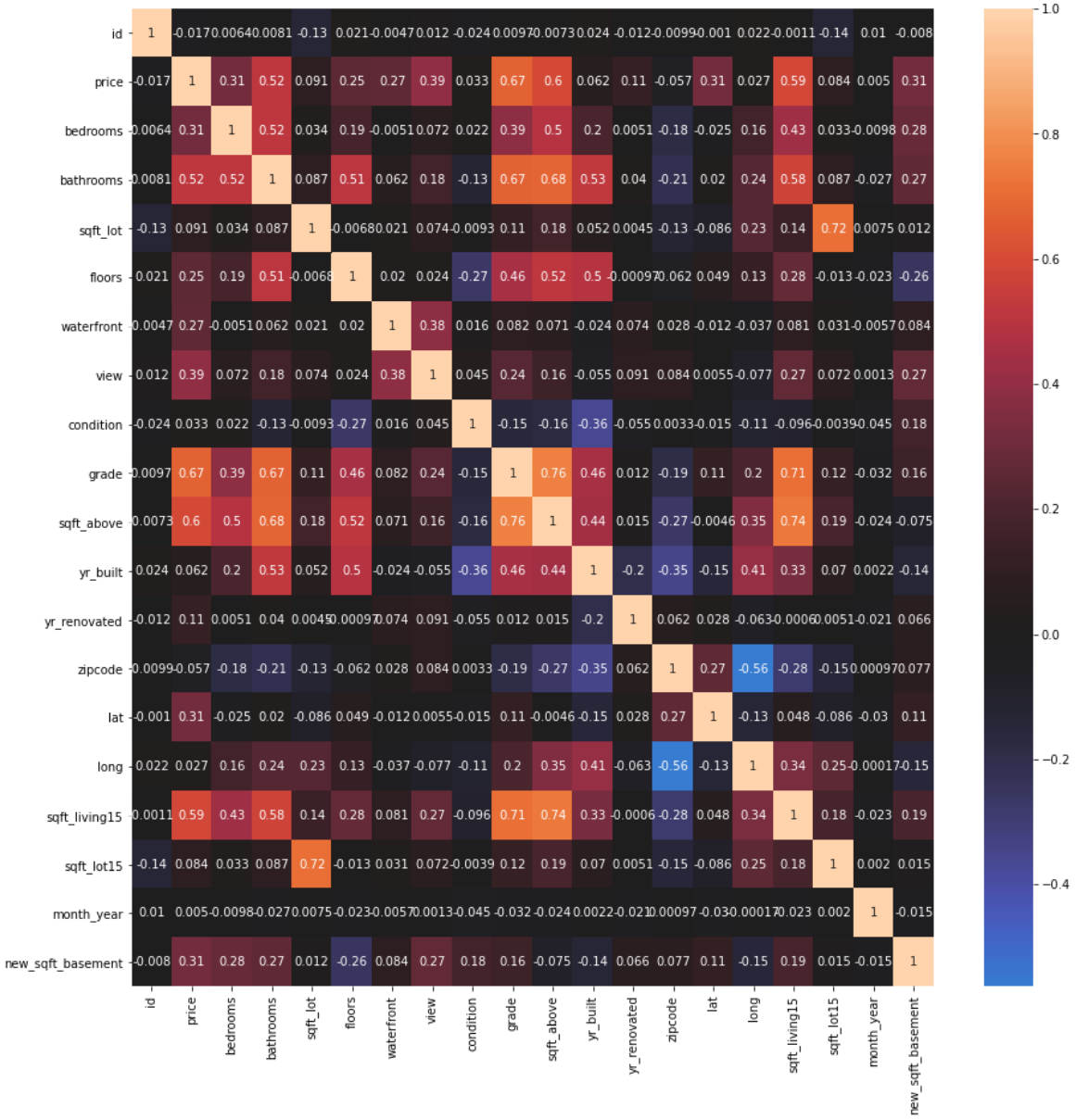
	id	price	bedrooms	bathrooms	sqft_lot	floors	waterfro
id	1.000000	-0.017381	0.006358	0.008093	-0.131518	0.020885	-0.0047
price	-0.017381	1.000000	0.312185	0.520621	0.090662	0.253822	0.2650
bedrooms	0.006358	0.312185	1.000000	0.516771	0.033895	0.187446	-0.0050
bathrooms	0.008093	0.520621	0.516771	1.000000	0.086682	0.507936	0.0615
sqft_lot	-0.131518	0.090662	0.033895	0.086682	1.000000	-0.006754	0.0212
floors	0.020885	0.253822	0.187446	0.507936	-0.006754	1.000000	0.0199
waterfront	-0.004709	0.265070	-0.005088	0.061548	0.021281	0.019964	1.0000
view	0.011758	0.391610	0.071583	0.178344	0.074404	0.024400	0.3847
condition	-0.024033	0.032663	0.022055	-0.129502	-0.009339	-0.266767	0.0159
grade	0.009678	0.670336	0.386365	0.672854	0.113313	0.458708	0.0818
sqft_above	-0.007324	0.598889	0.498238	0.680879	0.181091	0.522309	0.0707
yr_built	0.023793	0.062405	0.197471	0.526513	0.052362	0.498932	-0.0242
yr_renovated	-0.011600	0.114621	0.005094	0.039858	0.004464	-0.000971	0.0739
zipcode	-0.009853	-0.056615	-0.179090	-0.213739	-0.128692	-0.061938	0.0282
lat	-0.001034	0.313484	-0.025082	0.020034	-0.085651	0.048952	-0.0124
long	0.021608	0.027181	0.163422	0.236493	0.228574	0.128728	-0.0374
sqft_living15	-0.001067	0.589025	0.427325	0.576086	0.142869	0.278950	0.0812
sqft_lot15	-0.138964	0.083584	0.033155	0.087040	0.716199	-0.013179	0.0309
month_year	0.010187	0.004967	-0.009779	-0.026832	0.007462	-0.022854	-0.0056
new_sqft_basement	-0.007982	0.313396	0.281930	0.265911	0.012067	-0.255169	0.0841

In [68]:

```
1 plt.figure(figsize = (15, 15))
2 sns.heatmap(data = df.corr(), center=0, annot = True)
```

executed in 9.69s, finished 20:55:50 2021-06-12

Out[68]: <AxesSubplot:>



5 Predictive Modeling - Building Model

In [69]:
```
#df = df.drop('sqft_basement', axis = 1)
#df = df.drop('month', axis = 1)
#df = df.drop('date', axis = 1)
df.info()
```

executed in 37ms, finished 20:55:50 2021-06-12

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21263 entries, 0 to 21596
Data columns (total 20 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   id                 21263 non-null  int64  
 1   price              21263 non-null  float64
 2   bedrooms           21263 non-null  int64  
 3   bathrooms          21263 non-null  float64
 4   sqft_lot           21263 non-null  int64  
 5   floors             21263 non-null  float64
 6   waterfront         21263 non-null  float64
 7   view               21263 non-null  float64
 8   condition          21263 non-null  int64  
 9   grade              21263 non-null  int64  
 10  sqft_above         21263 non-null  int64  
 11  yr_built           21263 non-null  int64  
 12  yr_renovated       21263 non-null  float64
 13  zipcode            21263 non-null  int64  
 14  lat                21263 non-null  float64
 15  long               21263 non-null  float64
 16  sqft_living15      21263 non-null  int64  
 17  sqft_lot15         21263 non-null  int64  
 18  month_year         21263 non-null  int32  
 19  new_sqft_basement  21263 non-null  int64  
dtypes: float64(8), int32(1), int64(11)
memory usage: 3.3 MB
```

5.1 Baseline Model

In [70]:
```
#Dependent variable
outcome = 'price'
#Independent variables
predictors = df.drop('price', axis = 1)
predictor_variables = '+'.join(predictors.columns)
formula = outcome + '~' + predictor_variables
```

executed in 24ms, finished 20:55:50 2021-06-12

In [71]:

```
1 formula
```

executed in 16ms, finished 20:55:50 2021-06-12

Out[71]:

```
'price~id+bedrooms+bathrooms+sqft_lot+floors+waterfront+view+condition+grad
e+sqft_above+yr_built+yr_renovated+zipcode+lat+long+sqft_living15+sqft_lot1
5+month_year+new_sqft_basement'
```

In [72]:
```
baseline_model = ols(formula = formula, data = df).fit()
baseline_model.summary()
```

executed in 205ms, finished 20:55:50 2021-06-12

Out[72]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.704
Model:	OLS	**Adj. R-squared:**	0.704
Method:	Least Squares	**F-statistic:**	2660.
Date:	Sat, 12 Jun 2021	**Prob (F-statistic):**	0.00
Time:	20:55:50	**Log-Likelihood:**	-2.8895e+05
No. Observations:	21263	**AIC:**	5.779e+05
Df Residuals:	21243	**BIC:**	5.781e+05
Df Model:	19		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-5.972e+07	6.67e+06	-8.953	0.000	-7.28e+07	-4.66e+07
id	-1.467e-06	4.67e-07	-3.145	0.002	-2.38e-06	-5.53e-07
bedrooms	-3.825e+04	2069.073	-18.486	0.000	-4.23e+04	-3.42e+04
bathrooms	4.111e+04	3224.274	12.750	0.000	3.48e+04	4.74e+04
sqft_lot	0.1423	0.046	3.079	0.002	0.052	0.233
floors	8414.8521	3498.399	2.405	0.016	1557.725	1.53e+04
waterfront	6.008e+05	1.77e+04	33.978	0.000	5.66e+05	6.35e+05
view	5.276e+04	2073.717	25.442	0.000	4.87e+04	5.68e+04
condition	2.733e+04	2278.019	11.999	0.000	2.29e+04	3.18e+04
grade	9.659e+04	2104.651	45.895	0.000	9.25e+04	1.01e+05
sqft_above	172.6441	3.634	47.514	0.000	165.522	179.766
yr_built	-2552.5411	70.131	-36.397	0.000	-2690.002	-2415.080
yr_renovated	23.7810	3.884	6.123	0.000	16.168	31.394
zipcode	-575.2778	31.879	-18.045	0.000	-637.764	-512.792
lat	6.004e+05	1.04e+04	57.867	0.000	5.8e+05	6.21e+05
long	-2.065e+05	1.27e+04	-16.266	0.000	-2.31e+05	-1.82e+05
sqft_living15	25.9311	3.371	7.693	0.000	19.325	32.538
sqft_lot15	-0.3925	0.071	-5.516	0.000	-0.532	-0.253
month_year	331.1382	29.934	11.062	0.000	272.465	389.812
new_sqft_basement	146.0804	4.348	33.601	0.000	137.559	154.602

Omnibus:	15933.177	**Durbin-Watson:**	1.990

Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	944645.423
Skew:	3.062	**Prob(JB):**	0.00
Kurtosis:	35.074	**Cond. No.**	2.73e+13

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.73e+13. This might indicate that there are strong multicollinearity or other numerical problems.

Make Observatiosn!!! about baseline model

In [73]:
```
X = df.drop('price', axis =1)
y = df['price']
```
executed in 24ms, finished 20:55:51 2021-06-12

In [74]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                    test_size = 0.25, random_state = 25)
```
executed in 31ms, finished 20:55:51 2021-06-12

In [75]:
```
len(X_train)
```
executed in 23ms, finished 20:55:51 2021-06-12

Out[75]: 15947

In [76]:
```
len(X_test)
```
executed in 16ms, finished 20:55:51 2021-06-12

Out[76]: 5316

In [77]:
```
baseline_linreg = LinearRegression()
```
executed in 22ms, finished 20:55:51 2021-06-12

In [78]:
```
baseline_linreg.fit(X_train, y_train)
```
executed in 33ms, finished 20:55:51 2021-06-12

Out[78]: LinearRegression()

In [79]:
```
y_pred = baseline_linreg.predict(X_test)
```
executed in 26ms, finished 20:55:51 2021-06-12

In [80]:
```
residuals = y_pred - y_test
```
executed in 7ms, finished 20:55:51 2021-06-12

In [81]:
```
X.columns
```
executed in 19ms, finished 20:55:51 2021-06-12

Out[81]:
```
Index(['id', 'bedrooms', 'bathrooms', 'sqft_lot', 'floors', 'waterfront',
       'view', 'condition', 'grade', 'sqft_above', 'yr_built', 'yr_renovate
d',
       'zipcode', 'lat', 'long', 'sqft_living15', 'sqft_lot15', 'month_yea
r',
       'new_sqft_basement'],
      dtype='object')
```

In [82]:
```
baseline_linreg.coef_
```
executed in 23ms, finished 20:55:51 2021-06-12

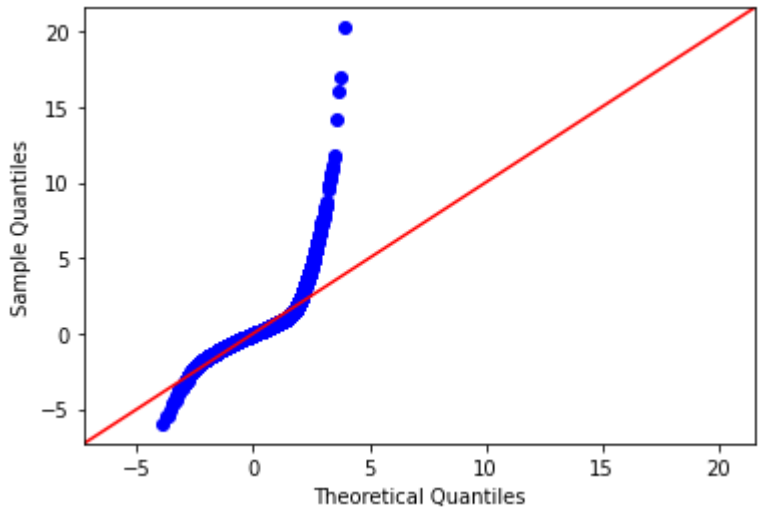
Out[82]:
```
array([-1.44243989e-06, -3.88005202e+04,  4.09480394e+04,  1.12156493e-01,
        7.83681434e+03,  6.32722371e+05,  5.01154981e+04,  2.70576753e+04,
        9.96543842e+04,  1.72189225e+02, -2.54455635e+03,  2.83918588e+01,
       -5.71048370e+02,  5.93557035e+05, -2.05885418e+05,  2.47798559e+01,
       -3.91153104e-01,  3.46622757e+02,  1.45948461e+02])
```

In [83]:
```
mse_train = mean_squared_error (y_train, baseline_linreg.predict(X_train
mse_test = mean_squared_error (y_test, y_pred)

print('Train RMSE:', np.sqrt(mse_train))
print('Train RMSE:', np.sqrt(mse_test))
```
executed in 41ms, finished 20:55:51 2021-06-12

```
Train RMSE: 193851.71413590503
Train RMSE: 190589.97613898848
```

In [84]:
```
fig = sm.graphics.qqplot(baseline_model.resid, dist = stats.norm,
                         line = '45', fit = True)
```

executed in 484ms, finished 20:55:51 2021-06-12



In [85]:
```
df.price.mean()
```

executed in 23ms, finished 20:55:51 2021-06-12

Out[85]: 535420.5864177209

5.2 Model 2 - Dealing with outliers

dropping condition because it is similar information to grade.

In [86]:
```
df = df.drop('condition', axis =1)
```

executed in 23ms, finished 20:55:51 2021-06-12

In [87]:
```
1 df.describe()
```
executed in 178ms, finished 20:55:52 2021-06-12

Out[87]:

	id	price	bedrooms	bathrooms	sqft_lot	floors	
count	2.126300e+04	2.126300e+04	21263.000000	21263.000000	2.126300e+04	21263.000000	2
mean	4.583740e+09	5.354206e+05	3.326012	2.099163	1.505278e+04	1.491911	
std	2.877828e+09	3.547799e+05	0.827807	0.749797	4.146123e+04	0.539395	
min	1.200019e+06	7.800000e+04	1.000000	0.500000	5.200000e+02	1.000000	
25%	2.123700e+09	3.200000e+05	3.000000	1.500000	5.040000e+03	1.000000	
50%	3.904940e+09	4.500000e+05	3.000000	2.250000	7.600000e+03	1.500000	
75%	7.325900e+09	6.399665e+05	4.000000	2.500000	1.062300e+04	2.000000	
max	9.900000e+09	7.060000e+06	5.000000	6.750000	1.651359e+06	3.500000	

In [88]:
```
1 df_no_outliers = df[df.bathrooms >= 1 ]
```
executed in 46ms, finished 20:55:52 2021-06-12

In [89]:
```
1 df_no_outliers = df_no_outliers[df_no_outliers.price <= 1500000 ]
```
executed in 36ms, finished 20:55:52 2021-06-12

In [90]:
```
1 df_no_outliers.describe()
```
executed in 177ms, finished 20:55:52 2021-06-12

Out[90]:

	id	price	bedrooms	bathrooms	sqft_lot	floors	
count	2.070400e+04	2.070400e+04	20704.000000	20704.000000	2.070400e+04	20704.000000	2
mean	4.596222e+09	4.998898e+05	3.313659	2.072317	1.470004e+04	1.485148	
std	2.880678e+09	2.469503e+05	0.817711	0.713538	4.007154e+04	0.538306	
min	1.200019e+06	7.800000e+04	1.000000	1.000000	5.200000e+02	1.000000	
25%	2.125059e+09	3.199375e+05	3.000000	1.500000	5.000000e+03	1.000000	
50%	3.905046e+09	4.450000e+05	3.000000	2.250000	7.536000e+03	1.000000	
75%	7.338200e+09	6.225000e+05	4.000000	2.500000	1.041450e+04	2.000000	
max	9.900000e+09	1.500000e+06	5.000000	6.000000	1.651359e+06	3.500000	

In [91]:

```
#Dependent variable
outcome = 'price'
#Independent variables
predictors = df_no_outliers.drop(['price', 'id'], axis = 1)
predictor_variables = '+'.join(predictors.columns)
formula = outcome + '~' + predictor_variables


```

executed in 29ms, finished 20:55:52 2021-06-12

In [92]:

```
formula
```

executed in 27ms, finished 20:55:52 2021-06-12

Out[92]:

```
'price~bedrooms+bathrooms+sqft_lot+floors+waterfront+view+grade+sqft_above+
yr_built+yr_renovated+zipcode+lat+long+sqft_living15+sqft_lot15+month_year+
new_sqft_basement'
```

In [93]:
```
model_2 = ols(formula = formula, data = df_no_outliers).fit()
model_2 .summary()
```

executed in 201ms, finished 20:55:52 2021-06-12

Out[93]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.708
Model:	OLS	**Adj. R-squared:**	0.708
Method:	Least Squares	**F-statistic:**	2955.
Date:	Sat, 12 Jun 2021	**Prob (F-statistic):**	0.00
Time:	20:55:52	**Log-Likelihood:**	-2.7370e+05
No. Observations:	20704	**AIC:**	5.474e+05
Df Residuals:	20686	**BIC:**	5.476e+05
Df Model:	17		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-4.585e+07	4.65e+06	-9.867	0.000	-5.5e+07	-3.67e+07
bedrooms	-1.305e+04	1468.742	-8.887	0.000	-1.59e+04	-1.02e+04
bathrooms	3.209e+04	2292.533	13.996	0.000	2.76e+04	3.66e+04
sqft_lot	0.1792	0.033	5.407	0.000	0.114	0.244
floors	2.883e+04	2471.906	11.663	0.000	2.4e+04	3.37e+04
waterfront	1.999e+05	1.71e+04	11.661	0.000	1.66e+05	2.33e+05
view	4.011e+04	1516.004	26.458	0.000	3.71e+04	4.31e+04
grade	8.623e+04	1498.485	57.546	0.000	8.33e+04	8.92e+04
sqft_above	87.6780	2.715	32.297	0.000	82.357	92.999
yr_built	-2361.8901	47.208	-50.032	0.000	-2454.421	-2269.359
yr_renovated	12.1787	2.767	4.402	0.000	6.756	17.602
zipcode	-352.5432	22.267	-15.833	0.000	-396.188	-308.898
lat	5.579e+05	7217.272	77.305	0.000	5.44e+05	5.72e+05
long	-9.595e+04	8906.140	-10.773	0.000	-1.13e+05	-7.85e+04
sqft_living15	44.6738	2.476	18.044	0.000	39.821	49.527
sqft_lot15	-0.1569	0.051	-3.092	0.002	-0.256	-0.057
month_year	229.9665	20.953	10.975	0.000	188.896	271.037
new_sqft_basement	90.3913	3.163	28.574	0.000	84.191	96.592

Omnibus:	3748.694	**Durbin-Watson:**	1.978
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	11965.503
Skew:	0.926	**Prob(JB):**	0.00

Kurtosis:	6.231	**Cond. No.**	1.13e+09

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.13e+09. This might indicate that there are
strong multicollinearity or other numerical problems.

Make Observatiosn!!! about baseline model

In [94]:
```
X = df_no_outliers.drop('price', axis =1)
y = df_no_outliers['price']
```
executed in 18ms, finished 20:55:52 2021-06-12

In [95]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.
                                                    random_state = 25)
```
executed in 45ms, finished 20:55:52 2021-06-12

In [96]:
```
len(X_train)
```
executed in 25ms, finished 20:55:52 2021-06-12

Out[96]: 15528

In [97]:
```
len(X_test)
```
executed in 29ms, finished 20:55:52 2021-06-12

Out[97]: 5176

In [98]:
```
model_2 = LinearRegression()
```
executed in 8ms, finished 20:55:52 2021-06-12

In [99]:
```
model_2.fit(X_train, y_train)
```
executed in 30ms, finished 20:55:52 2021-06-12

Out[99]: LinearRegression()

In [100]:
```
y_pred = model_2.predict(X_test)
```
executed in 24ms, finished 20:55:52 2021-06-12

In [101]:
```
residuals = y_pred - y_test
```
executed in 9ms, finished 20:55:52 2021-06-12

In [102]:
```
1 X.columns
```
executed in 19ms, finished 20:55:52 2021-06-12

Out[102]:
```
Index(['id', 'bedrooms', 'bathrooms', 'sqft_lot', 'floors', 'waterfront',
       'view', 'grade', 'sqft_above', 'yr_built', 'yr_renovated', 'zipcod
e',
       'lat', 'long', 'sqft_living15', 'sqft_lot15', 'month_year',
       'new_sqft_basement'],
      dtype='object')
```

In [103]:
```
1 model_2.coef_
```
executed in 22ms, finished 20:55:52 2021-06-12

Out[103]:
```
array([-6.80544437e-07, -1.37653240e+04,  3.17587211e+04,  2.29746501e-01,
        2.79916764e+04,  2.01389841e+05,  3.92694793e+04,  8.50759344e+04,
        8.74583953e+01, -2.30722540e+03,  1.61596694e+01, -3.72014372e+02,
        5.63273279e+05, -9.96259135e+04,  4.62589676e+01, -2.18415699e-01,
        2.17820455e+02,  9.04693169e+01])
```

In [104]:
```
model_2 = ols(formula = formula, data = df_no_outliers).fit()
model_2.summary()
```

executed in 185ms, finished 20:55:52 2021-06-12

Out[104]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.708
Model:	OLS	**Adj. R-squared:**	0.708
Method:	Least Squares	**F-statistic:**	2955.
Date:	Sat, 12 Jun 2021	**Prob (F-statistic):**	0.00
Time:	20:55:52	**Log-Likelihood:**	-2.7370e+05
No. Observations:	20704	**AIC:**	5.474e+05
Df Residuals:	20686	**BIC:**	5.476e+05
Df Model:	17		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-4.585e+07	4.65e+06	-9.867	0.000	-5.5e+07	-3.67e+07
bedrooms	-1.305e+04	1468.742	-8.887	0.000	-1.59e+04	-1.02e+04
bathrooms	3.209e+04	2292.533	13.996	0.000	2.76e+04	3.66e+04
sqft_lot	0.1792	0.033	5.407	0.000	0.114	0.244
floors	2.883e+04	2471.906	11.663	0.000	2.4e+04	3.37e+04
waterfront	1.999e+05	1.71e+04	11.661	0.000	1.66e+05	2.33e+05
view	4.011e+04	1516.004	26.458	0.000	3.71e+04	4.31e+04
grade	8.623e+04	1498.485	57.546	0.000	8.33e+04	8.92e+04
sqft_above	87.6780	2.715	32.297	0.000	82.357	92.999
yr_built	-2361.8901	47.208	-50.032	0.000	-2454.421	-2269.359
yr_renovated	12.1787	2.767	4.402	0.000	6.756	17.602
zipcode	-352.5432	22.267	-15.833	0.000	-396.188	-308.898
lat	5.579e+05	7217.272	77.305	0.000	5.44e+05	5.72e+05
long	-9.595e+04	8906.140	-10.773	0.000	-1.13e+05	-7.85e+04
sqft_living15	44.6738	2.476	18.044	0.000	39.821	49.527
sqft_lot15	-0.1569	0.051	-3.092	0.002	-0.256	-0.057
month_year	229.9665	20.953	10.975	0.000	188.896	271.037
new_sqft_basement	90.3913	3.163	28.574	0.000	84.191	96.592

Omnibus:	3748.694	**Durbin-Watson:**	1.978
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	11965.503
Skew:	0.926	**Prob(JB):**	0.00

Kurtosis:	6.231	**Cond. No.**	1.13e+09

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.13e+09. This might indicate that there are
strong multicollinearity or other numerical problems.

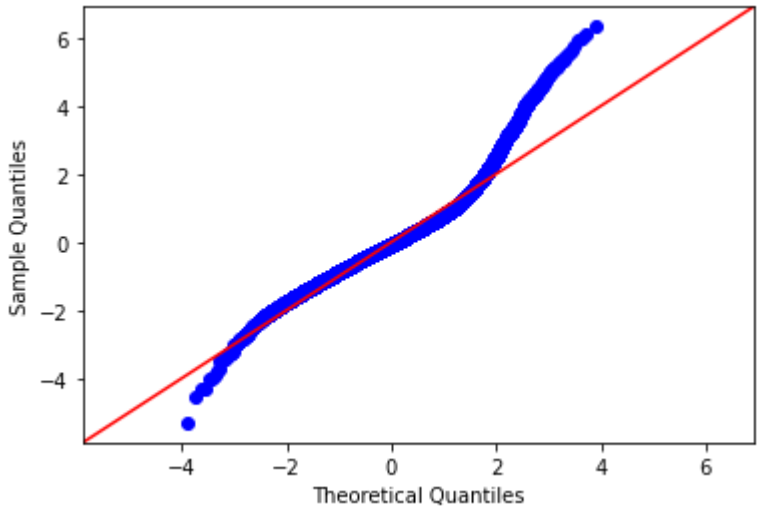
In [105]:
```
mse_train = mean_squared_error (y_train, model_2.predict(X_train))
mse_test = mean_squared_error (y_test, y_pred)

print('Train RMSE:', np.sqrt(mse_train))
print('Test RMSE:', np.sqrt(mse_test))
```

executed in 87ms, finished 20:55:53 2021-06-12

```
Train RMSE: 134053.9830253697
Test RMSE: 131463.33680332475
```

In [106]:
```
fig = sm.graphics.qqplot(model_2.resid, dist = stats.norm,
                         line = '45', fit = True)
```

executed in 447ms, finished 20:55:53 2021-06-12



In [107]:
```
df_no_outliers.price.mean()
```

executed in 24ms, finished 20:55:53 2021-06-12

Out[107]: 499889.77622681606

5.3 Model 3 Dealing with dummies

In [108]:
```
1 df.info()
```
executed in 40ms, finished 20:55:53 2021-06-12

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21263 entries, 0 to 21596
Data columns (total 19 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   id                 21263 non-null  int64  
 1   price              21263 non-null  float64
 2   bedrooms           21263 non-null  int64  
 3   bathrooms          21263 non-null  float64
 4   sqft_lot           21263 non-null  int64  
 5   floors             21263 non-null  float64
 6   waterfront         21263 non-null  float64
 7   view               21263 non-null  float64
 8   grade              21263 non-null  int64  
 9   sqft_above         21263 non-null  int64  
 10  yr_built           21263 non-null  int64  
 11  yr_renovated       21263 non-null  float64
 12  zipcode            21263 non-null  int64  
 13  lat                21263 non-null  float64
 14  long               21263 non-null  float64
 15  sqft_living15      21263 non-null  int64  
 16  sqft_lot15         21263 non-null  int64  
 17  month_year         21263 non-null  int32  
 18  new_sqft_basement  21263 non-null  int64  
dtypes: float64(8), int32(1), int64(10)
memory usage: 3.2 MB
```

In [109]:
```
1 waterfront_dummies = pd.get_dummies(df_no_outliers['waterfront'],
2                                     prefix = 'waterfront', drop_first = T
```
executed in 23ms, finished 20:55:53 2021-06-12

In [110]:
```
1 view_dummies = pd.get_dummies(df_no_outliers['view'],
2                               prefix = 'view', drop_first = True)
```
executed in 17ms, finished 20:55:53 2021-06-12

In [111]:
```
1 grade_dummies = pd.get_dummies(df_no_outliers['grade'],
2                                prefix = 'grade', drop_first = True)
```
executed in 25ms, finished 20:55:53 2021-06-12

In [112]:
```
1 zip_dummies = pd.get_dummies(df_no_outliers['zipcode'],
2                              prefix = 'zip', drop_first = True)
```
executed in 23ms, finished 20:55:53 2021-06-12

In [113]:
```
df_cat = pd.concat([df_no_outliers, waterfront_dummies, view_dummies,
                    grade_dummies, zip_dummies], axis = 1 )
```
executed in 35ms, finished 20:55:53 2021-06-12

In [114]:
```
df_cat.head()
```
executed in 73ms, finished 20:55:53 2021-06-12

Out[114]:

	id	price	bedrooms	bathrooms	sqft_lot	floors	waterfront	view	grade	sqft
0	7129300520	221900.0	3	1.00	5650	1.0	0.0	0.0	7	
1	6414100192	538000.0	3	2.25	7242	2.0	0.0	0.0	7	
2	5631500400	180000.0	2	1.00	10000	1.0	0.0	0.0	6	
3	2487200875	604000.0	4	3.00	5000	1.0	0.0	0.0	7	
4	1954400510	510000.0	3	2.00	8080	1.0	0.0	0.0	8	

5 rows × 101 columns

In [115]:
```
df_cat.drop(columns = ['id', 'waterfront', 'view',
                       'zipcode', 'grade'], inplace = True)
```
executed in 48ms, finished 20:55:53 2021-06-12

In [116]:
```
1 df_cat.info()
```

executed in 67ms, finished 20:55:53 2021-06-12

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20704 entries, 0 to 21595
Data columns (total 96 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   price              20704 non-null  float64
 1   bedrooms           20704 non-null  int64  
 2   bathrooms          20704 non-null  float64
 3   sqft_lot           20704 non-null  int64  
 4   floors             20704 non-null  float64
 5   sqft_above         20704 non-null  int64  
 6   yr_built           20704 non-null  int64  
 7   yr_renovated       20704 non-null  float64
 8   lat                20704 non-null  float64
 9   long               20704 non-null  float64
 10  sqft_living15      20704 non-null  int64  
 11  sqft_lot15         20704 non-null  int64  
 12  month_year         20704 non-null  int32  
 13  new_sqft_basement  20704 non-null  int64  
 14  waterfront_1.0     20704 non-null  uint8  
 15  view_1.0           20704 non-null  uint8  
 16  view_2.0           20704 non-null  uint8  
 17  view_3.0           20704 non-null  uint8  
 18  view_4.0           20704 non-null  uint8  
 19  grade_5            20704 non-null  uint8  
 20  grade_6            20704 non-null  uint8  
 21  grade_7            20704 non-null  uint8  
 22  grade_8            20704 non-null  uint8  
 23  grade_9            20704 non-null  uint8  
 24  grade_10           20704 non-null  uint8  
 25  grade_11           20704 non-null  uint8  
 26  grade_12           20704 non-null  uint8  
 27  zip_98002          20704 non-null  uint8  
 28  zip_98003          20704 non-null  uint8  
 29  zip_98004          20704 non-null  uint8  
 30  zip_98005          20704 non-null  uint8  
 31  zip_98006          20704 non-null  uint8  
 32  zip_98007          20704 non-null  uint8  
 33  zip_98008          20704 non-null  uint8  
 34  zip_98010          20704 non-null  uint8  
 35  zip_98011          20704 non-null  uint8  
 36  zip_98014          20704 non-null  uint8  
 37  zip_98019          20704 non-null  uint8  
 38  zip_98022          20704 non-null  uint8  
 39  zip_98023          20704 non-null  uint8  
 40  zip_98024          20704 non-null  uint8  
 41  zip_98027          20704 non-null  uint8  
 42  zip_98028          20704 non-null  uint8  
 43  zip_98029          20704 non-null  uint8  
 44  zip_98030          20704 non-null  uint8  
 45  zip_98031          20704 non-null  uint8  
 46  zip_98032          20704 non-null  uint8  
 47  zip_98033          20704 non-null  uint8
```

```
 48  zip_98034          20704 non-null  uint8
 49  zip_98038          20704 non-null  uint8
 50  zip_98039          20704 non-null  uint8
 51  zip_98040          20704 non-null  uint8
 52  zip_98042          20704 non-null  uint8
 53  zip_98045          20704 non-null  uint8
 54  zip_98052          20704 non-null  uint8
 55  zip_98053          20704 non-null  uint8
 56  zip_98055          20704 non-null  uint8
 57  zip_98056          20704 non-null  uint8
 58  zip_98058          20704 non-null  uint8
 59  zip_98059          20704 non-null  uint8
 60  zip_98065          20704 non-null  uint8
 61  zip_98070          20704 non-null  uint8
 62  zip_98072          20704 non-null  uint8
 63  zip_98074          20704 non-null  uint8
 64  zip_98075          20704 non-null  uint8
 65  zip_98077          20704 non-null  uint8
 66  zip_98092          20704 non-null  uint8
 67  zip_98102          20704 non-null  uint8
 68  zip_98103          20704 non-null  uint8
 69  zip_98105          20704 non-null  uint8
 70  zip_98106          20704 non-null  uint8
 71  zip_98107          20704 non-null  uint8
 72  zip_98108          20704 non-null  uint8
 73  zip_98109          20704 non-null  uint8
 74  zip_98112          20704 non-null  uint8
 75  zip_98115          20704 non-null  uint8
 76  zip_98116          20704 non-null  uint8
 77  zip_98117          20704 non-null  uint8
 78  zip_98118          20704 non-null  uint8
 79  zip_98119          20704 non-null  uint8
 80  zip_98122          20704 non-null  uint8
 81  zip_98125          20704 non-null  uint8
 82  zip_98126          20704 non-null  uint8
 83  zip_98133          20704 non-null  uint8
 84  zip_98136          20704 non-null  uint8
 85  zip_98144          20704 non-null  uint8
 86  zip_98146          20704 non-null  uint8
 87  zip_98148          20704 non-null  uint8
 88  zip_98155          20704 non-null  uint8
 89  zip_98166          20704 non-null  uint8
 90  zip_98168          20704 non-null  uint8
 91  zip_98177          20704 non-null  uint8
 92  zip_98178          20704 non-null  uint8
 93  zip_98188          20704 non-null  uint8
 94  zip_98198          20704 non-null  uint8
 95  zip_98199          20704 non-null  uint8
dtypes: float64(6), int32(1), int64(7), uint8(82)
memory usage: 3.9 MB
```

In [117]:

```
df_cat.columns = ['price', 'bedrooms', 'bathrooms', 'sqft_lot', 'floors',
       'sqft_above','yr_built','yr_renovated', 'lat', 'long', 'sqft_livi
       'sqft_lot15', 'month_year', 'new_sqft_basement', 'waterfront_1',
       'view_1','view_2', 'view_3', 'view_4', 'grade_5', 'grade_6', 'gra
       'grade_8', 'grade_9', 'grade_10', 'grade_11', 'grade_12', 'zip_98
       'zip_98003', 'zip_98004', 'zip_98005', 'zip_98006', 'zip_98007',
       'zip_98008', 'zip_98010', 'zip_98011', 'zip_98014', 'zip_98019',
       'zip_98022', 'zip_98023', 'zip_98024', 'zip_98027', 'zip_98028',
       'zip_98029', 'zip_98030', 'zip_98031', 'zip_98032', 'zip_98033',
       'zip_98034', 'zip_98038', 'zip_98039', 'zip_98040', 'zip_98042',
       'zip_98045', 'zip_98052', 'zip_98053', 'zip_98055', 'zip_98056',
       'zip_98058', 'zip_98059', 'zip_98065', 'zip_98070', 'zip_98072',
       'zip_98074', 'zip_98075', 'zip_98077', 'zip_98092', 'zip_98102',
       'zip_98103', 'zip_98105', 'zip_98106', 'zip_98107', 'zip_98108',
       'zip_98109', 'zip_98112', 'zip_98115', 'zip_98116', 'zip_98117',
       'zip_98118', 'zip_98119', 'zip_98122', 'zip_98125', 'zip_98126',
       'zip_98133', 'zip_98136', 'zip_98144', 'zip_98146', 'zip_98148',
       'zip_98155', 'zip_98166', 'zip_98168', 'zip_98177', 'zip_98178',
       'zip_98188', 'zip_98198', 'zip_98199']
```

executed in 25ms, finished 20:55:53 2021-06-12

In [118]:

```
df_cat.columns
```

executed in 36ms, finished 20:55:53 2021-06-12

Out[118]:

```
Index(['price', 'bedrooms', 'bathrooms', 'sqft_lot', 'floors', 'sqft_abov
e',
       'yr_built', 'yr_renovated', 'lat', 'long', 'sqft_living15',
       'sqft_lot15', 'month_year', 'new_sqft_basement', 'waterfront_1',
       'view_1', 'view_2', 'view_3', 'view_4', 'grade_5', 'grade_6', 'grade
_7',
       'grade_8', 'grade_9', 'grade_10', 'grade_11', 'grade_12', 'zip_9800
2',
       'zip_98003', 'zip_98004', 'zip_98005', 'zip_98006', 'zip_98007',
       'zip_98008', 'zip_98010', 'zip_98011', 'zip_98014', 'zip_98019',
       'zip_98022', 'zip_98023', 'zip_98024', 'zip_98027', 'zip_98028',
       'zip_98029', 'zip_98030', 'zip_98031', 'zip_98032', 'zip_98033',
       'zip_98034', 'zip_98038', 'zip_98039', 'zip_98040', 'zip_98042',
       'zip_98045', 'zip_98052', 'zip_98053', 'zip_98055', 'zip_98056',
       'zip_98058', 'zip_98059', 'zip_98065', 'zip_98070', 'zip_98072',
       'zip_98074', 'zip_98075', 'zip_98077', 'zip_98092', 'zip_98102',
       'zip_98103', 'zip_98105', 'zip_98106', 'zip_98107', 'zip_98108',
       'zip_98109', 'zip_98112', 'zip_98115', 'zip_98116', 'zip_98117',
       'zip_98118', 'zip_98119', 'zip_98122', 'zip_98125', 'zip_98126',
       'zip_98133', 'zip_98136', 'zip_98144', 'zip_98146', 'zip_98148',
       'zip_98155', 'zip_98166', 'zip_98168', 'zip_98177', 'zip_98178',
       'zip_98188', 'zip_98198', 'zip_98199'],
      dtype='object')
```

In [119]:
```
#Dependent variable
outcome = 'price'
#Independent variables
predictors = df_cat.drop(['price'], axis = 1)
predictor_variables = '+'.join(predictors.columns)
formula = outcome + '~' + predictor_variables


```

executed in 40ms, finished 20:55:54 2021-06-12

In [120]:
```
formula
```

executed in 18ms, finished 20:55:54 2021-06-12

Out[120]:
```
'price~bedrooms+bathrooms+sqft_lot+floors+sqft_above+yr_built+yr_renovated+
lat+long+sqft_living15+sqft_lot15+month_year+new_sqft_basement+waterfront_1
+view_1+view_2+view_3+view_4+grade_5+grade_6+grade_7+grade_8+grade_9+grade_
10+grade_11+grade_12+zip_98002+zip_98003+zip_98004+zip_98005+zip_98006+zip_
98007+zip_98008+zip_98010+zip_98011+zip_98014+zip_98019+zip_98022+zip_98023
+zip_98024+zip_98027+zip_98028+zip_98029+zip_98030+zip_98031+zip_98032+zip_
98033+zip_98034+zip_98038+zip_98039+zip_98040+zip_98042+zip_98045+zip_98052
+zip_98053+zip_98055+zip_98056+zip_98058+zip_98059+zip_98065+zip_98070+zip_
98072+zip_98074+zip_98075+zip_98077+zip_98092+zip_98102+zip_98103+zip_98105
+zip_98106+zip_98107+zip_98108+zip_98109+zip_98112+zip_98115+zip_98116+zip_
98117+zip_98118+zip_98119+zip_98122+zip_98125+zip_98126+zip_98133+zip_98136
+zip_98144+zip_98146+zip_98148+zip_98155+zip_98166+zip_98168+zip_98177+zip_
98178+zip_98188+zip_98198+zip_98199'
```

In [121]:
```
model_3 = ols(formula = formula, data = df_cat).fit()
model_3 .summary()
```

executed in 807ms, finished 20:55:54 2021-06-12

yr_renovated	15.3771	2.064	7.450	0.000	11.331	19.423
lat	1.024e+05	3.98e+04	2.572	0.010	2.44e+04	1.8e+05
long	-9.631e+04	2.92e+04	-3.303	0.001	-1.53e+05	-3.92e+04
sqft_living15	25.0284	1.936	12.926	0.000	21.233	28.824
sqft_lot15	0.0176	0.039	0.454	0.650	-0.059	0.094
month_year	244.9422	15.568	15.734	0.000	214.428	275.456
new_sqft_basement	79.4837	2.375	33.473	0.000	74.829	84.138
waterfront_1	2.34e+05	1.4e+04	16.659	0.000	2.06e+05	2.61e+05
view_1	6.737e+04	5953.164	11.316	0.000	5.57e+04	7.9e+04
view_2	6.733e+04	3601.982	18.693	0.000	6.03e+04	7.44e+04
view_3	1.212e+05	5083.455	23.842	0.000	1.11e+05	1.31e+05
view_4	1.939e+05	8378.098	23.148	0.000	1.78e+05	2.1e+05
grade_5	1.246e+04	2.73e+04	0.456	0.648	-4.11e+04	6.6e+04

Make Observatiosn!!! about baseline model

In [122]:
```
X = df_cat.drop('price', axis =1)
y = df_cat['price']
```
executed in 33ms, finished 20:55:54 2021-06-12

In [123]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                    test_size = 0.25, random_state = 25)
```
executed in 57ms, finished 20:55:54 2021-06-12

In [124]:
```
len(X_train)
```
executed in 17ms, finished 20:55:54 2021-06-12

Out[124]: 15528

In [125]:
```
len(X_test)
```
executed in 40ms, finished 20:55:54 2021-06-12

Out[125]: 5176

In [126]:
```
model_3 = LinearRegression()
```
executed in 32ms, finished 20:55:55 2021-06-12

In [127]:
```
model_3.fit(X_train, y_train)
```
executed in 89ms, finished 20:55:55 2021-06-12

Out[127]: LinearRegression()

In [128]:
```
y_pred = model_3.predict(X_test)
```
executed in 33ms, finished 20:55:55 2021-06-12

In [129]:
```
residuals = y_pred - y_test
```
executed in 16ms, finished 20:55:55 2021-06-12

In [130]:
```
1 X.columns
```

executed in 24ms, finished 20:55:55 2021-06-12

Out[130]:
```
Index(['bedrooms', 'bathrooms', 'sqft_lot', 'floors', 'sqft_above', 'yr_bui
lt',
       'yr_renovated', 'lat', 'long', 'sqft_living15', 'sqft_lot15',
       'month_year', 'new_sqft_basement', 'waterfront_1', 'view_1', 'view_
2',
       'view_3', 'view_4', 'grade_5', 'grade_6', 'grade_7', 'grade_8',
       'grade_9', 'grade_10', 'grade_11', 'grade_12', 'zip_98002', 'zip_980
03',
       'zip_98004', 'zip_98005', 'zip_98006', 'zip_98007', 'zip_98008',
       'zip_98010', 'zip_98011', 'zip_98014', 'zip_98019', 'zip_98022',
       'zip_98023', 'zip_98024', 'zip_98027', 'zip_98028', 'zip_98029',
       'zip_98030', 'zip_98031', 'zip_98032', 'zip_98033', 'zip_98034',
       'zip_98038', 'zip_98039', 'zip_98040', 'zip_98042', 'zip_98045',
       'zip_98052', 'zip_98053', 'zip_98055', 'zip_98056', 'zip_98058',
       'zip_98059', 'zip_98065', 'zip_98070', 'zip_98072', 'zip_98074',
       'zip_98075', 'zip_98077', 'zip_98092', 'zip_98102', 'zip_98103',
       'zip_98105', 'zip_98106', 'zip_98107', 'zip_98108', 'zip_98109',
       'zip_98112', 'zip_98115', 'zip_98116', 'zip_98117', 'zip_98118',
       'zip_98119', 'zip_98122', 'zip_98125', 'zip_98126', 'zip_98133',
       'zip_98136', 'zip_98144', 'zip_98146', 'zip_98148', 'zip_98155',
       'zip_98166', 'zip_98168', 'zip_98177', 'zip_98178', 'zip_98188',
       'zip_98198', 'zip_98199'],
      dtype='object')
```

In [131]:
```
model_3.coef_
```
executed in 19ms, finished 20:55:55 2021-06-12

Out[131]:
```
array([-2.02366806e+03,  2.35311166e+04,  2.99844375e-01, -2.21343210e+04,
        1.19981630e+02, -6.60571437e+02,  1.61210088e+01,  8.78743349e+04,
       -7.62891721e+04,  2.45725511e+01, -6.23863976e-02,  2.33089586e+02,
        8.13396923e+01,  2.33869316e+05,  6.19078323e+04,  6.44966079e+04,
        1.18518866e+05,  2.07950238e+05,  2.82344062e+04,  2.90853837e+04,
        5.21076686e+04,  9.35903063e+04,  1.73661641e+05,  2.44807576e+05,
        3.27775689e+05,  4.18148169e+05,  2.48954572e+04, -5.16936903e+03,
        5.81437386e+05,  3.04497471e+05,  2.65327812e+05,  2.45189667e+05,
        2.36988862e+05,  9.59447830e+04,  1.03348787e+05,  1.05278694e+05,
        8.54981689e+04,  3.52965001e+04, -2.49820546e+04,  1.56439296e+05,
        1.85684075e+05,  9.30657444e+04,  2.27643327e+05,  1.44976297e+04,
        2.03275676e+04, -3.85827673e+03,  3.33734962e+05,  1.64306688e+05,
        5.46822579e+04,  7.25983923e+05,  4.61343850e+05,  2.54786451e+04,
        1.33499915e+05,  2.37744842e+05,  2.24070846e+05,  3.29878084e+04,
        9.62112977e+04,  3.83725465e+04,  1.04210479e+05,  1.23825932e+05,
        8.93938439e+04,  1.49291453e+05,  1.89394620e+05,  2.08014272e+05,
        1.43252785e+05, -8.49765984e+03,  3.88041990e+05,  2.98583238e+05,
        3.89070350e+05,  9.84580216e+04,  2.89347154e+05,  9.65295572e+04,
        4.27503979e+05,  4.60121916e+05,  2.94448073e+05,  2.64085201e+05,
        2.76353422e+05,  1.39201737e+05,  4.07177318e+05,  3.03078462e+05,
        1.64780963e+05,  1.60363960e+05,  1.17396934e+05,  2.21121357e+05,
        2.32485156e+05,  8.33560200e+04,  3.52172785e+04,  1.04319343e+05,
        7.72376463e+04,  2.81468408e+04,  1.79772581e+05,  2.88776844e+04,
        1.74479530e+04,  1.02674117e+04,  3.38075827e+05])
```

In [132]:
```
model_3 = ols(formula = formula, data = df_cat).fit()
model_3.summary()
```
executed in 824ms, finished 20:55:56 2021-06-12

Out[132]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.840
Model:	OLS	**Adj. R-squared:**	0.840
Method:	Least Squares	**F-statistic:**	1142.
Date:	Sat, 12 Jun 2021	**Prob (F-statistic):**	0.00
Time:	20:55:55	**Log-Likelihood:**	-2.6746e+05
No. Observations:	20704	**AIC:**	5.351e+05
Df Residuals:	20608	**BIC:**	5.359e+05
Df Model:	95		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	-6.463e+07	5e+06	-12.917	0.000	-7.44e+07	-5.48e+07

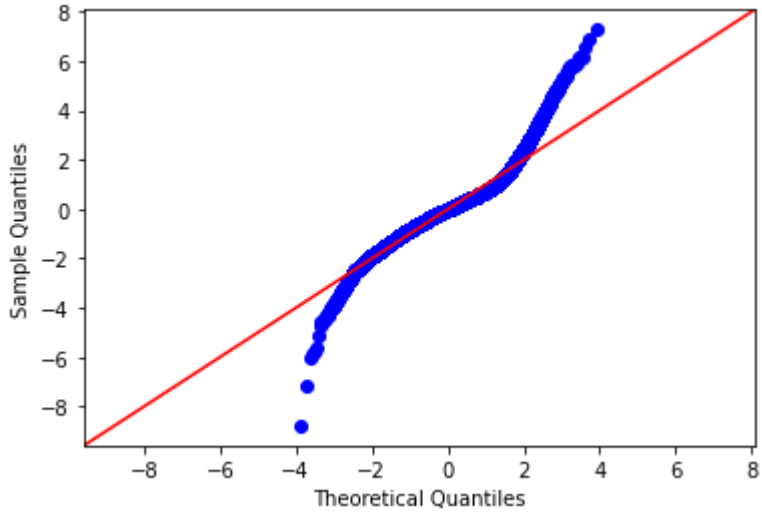
In [133]:
```
mse_train = mean_squared_error (y_train, model_3.predict(X_train))
mse_test = mean_squared_error (y_test, y_pred)

print('Train RMSE:', np.sqrt(mse_train))
print('Test RMSE:', np.sqrt(mse_test))
```

executed in 327ms, finished 20:55:56 2021-06-12

```
Train RMSE: 98185.76728445895
Test RMSE: 100899.38245182061
```

In [134]:
```
fig = sm.graphics.qqplot(model_3.resid, dist = stats.norm,
                         line = '45', fit = True)
```

executed in 558ms, finished 20:55:56 2021-06-12



In [135]:
```
df_no_outliers.price.mean()
```

executed in 16ms, finished 20:55:56 2021-06-12

Out[135]: 499889.77622681606

In [136]:
```
df_cat.price.mean()
```

executed in 33ms, finished 20:55:56 2021-06-12

Out[136]: 499889.77622681606

5.4 Model 5 - Log transformation

In [137]:
```
df_columns = df.columns
df_columns = ['price', 'bedrooms', 'bathrooms', 'sqft_lot', 'floors',
       'waterfront', 'view', 'grade', 'sqft_above', 'yr_built',
       'yr_renovated','zipcode', 'lat', 'long', 'sqft_living15',
       'sqft_lot15', 'month_year',
       'new_sqft_basement']
```

executed in 18ms, finished 20:55:56 2021-06-12

In [138]:
```
for i in df_columns:
    figure, ax= plt.subplots(figsize = (10,10))
    sns.distplot(df, x = df[i])
    plt.title(i)

```

executed in 16.7s, finished 20:56:13 2021-06-12

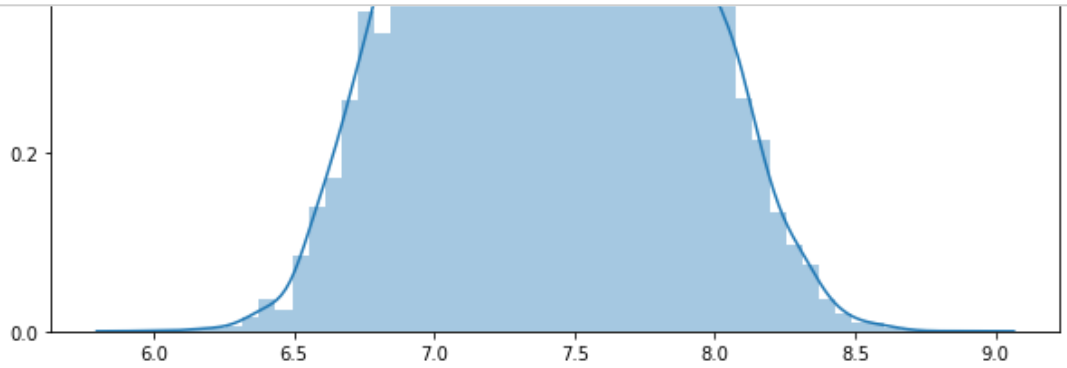
from the above graphs the following columns will have log transformations: sqft_lot, sqft_above, sqft_living15, sqft_lot15

In [139]:
```
df_no_outliers['sqft_lot_log'] = np.log(df_no_outliers['sqft_lot'])
df_no_outliers['sqft_above_log'] = np.log(df_no_outliers['sqft_above'])
df_no_outliers['sqft_living15_log'] = np.log(df_no_outliers['sqft_living1
df_no_outliers['sqft_lot15_log'] = np.log(df_no_outliers['sqft_lot15'])
```

executed in 28ms, finished 20:56:13 2021-06-12

In [140]:
```
log_columns = ['sqft_lot_log', 'sqft_above_log',
               'sqft_living15_log', 'sqft_lot15_log']
for i in log_columns:
    figure, ax= plt.subplots(figsize = (10,10))
    sns.distplot(df, x = df_no_outliers[i])
    plt.title(i)
```

executed in 3.48s, finished 20:56:17 2021-06-12



In [ ]:
```

```

In [141]:
```
#Dependent variable
outcome = 'price'
#Independent variables
predictors = df_no_outliers.drop(['price', 'id'], axis = 1)
predictor_variables = '+'.join(predictors.columns)
formula = outcome + '~' + predictor_variables


```

executed in 24ms, finished 20:56:17 2021-06-12

In [142]:
```
formula
```

executed in 26ms, finished 20:56:17 2021-06-12

Out[142]:
```
'price~bedrooms+bathrooms+sqft_lot+floors+waterfront+view+grade+sqft_above+
yr_built+yr_renovated+zipcode+lat+long+sqft_living15+sqft_lot15+month_year+
new_sqft_basement+sqft_lot_log+sqft_above_log+sqft_living15_log+sqft_lot15_
log'
```

In [143]:
```
model_5 = ols(formula = formula, data = df_no_outliers).fit()
model_5 .summary()
```

executed in 227ms, finished 20:56:17 2021-06-12

Out[143]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.713
Model:	OLS	**Adj. R-squared:**	0.713
Method:	Least Squares	**F-statistic:**	2444.
Date:	Sat, 12 Jun 2021	**Prob (F-statistic):**	0.00
Time:	20:56:17	**Log-Likelihood:**	-2.7354e+05
No. Observations:	20704	**AIC:**	5.471e+05
Df Residuals:	20682	**BIC:**	5.473e+05
Df Model:	21		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-3.71e+07	4.65e+06	-7.984	0.000	-4.62e+07	-2.8e+07
bedrooms	-1.002e+04	1501.000	-6.673	0.000	-1.3e+04	-7073.604
bathrooms	2.966e+04	2301.600	12.885	0.000	2.51e+04	3.42e+04
sqft_lot	0.2937	0.039	7.443	0.000	0.216	0.371
floors	8222.3394	2807.416	2.929	0.003	2719.583	1.37e+04
waterfront	2.256e+05	1.71e+04	13.215	0.000	1.92e+05	2.59e+05
view	4.014e+04	1509.004	26.602	0.000	3.72e+04	4.31e+04
grade	8.521e+04	1493.737	57.046	0.000	8.23e+04	8.81e+04
sqft_above	99.3024	6.129	16.203	0.000	87.290	111.315
yr_built	-2411.7094	47.123	-51.179	0.000	-2504.075	-2319.344
yr_renovated	13.2034	2.749	4.804	0.000	7.816	18.591
zipcode	-384.4474	22.271	-17.262	0.000	-428.100	-340.795
lat	5.471e+05	7200.628	75.974	0.000	5.33e+05	5.61e+05
long	-7.01e+04	8994.424	-7.794	0.000	-8.77e+04	-5.25e+04
sqft_living15	87.6733	8.410	10.425	0.000	71.189	104.158
sqft_lot15	0.3625	0.065	5.546	0.000	0.234	0.491
month_year	224.9249	20.799	10.814	0.000	184.157	265.693
new_sqft_basement	90.5068	3.147	28.759	0.000	84.338	96.675
sqft_lot_log	-1.021e+04	3274.865	-3.118	0.002	-1.66e+04	-3791.837
sqft_above_log	-3633.0193	1.12e+04	-0.325	0.745	-2.55e+04	1.83e+04
sqft_living15_log	-7.462e+04	1.63e+04	-4.566	0.000	-1.07e+05	-4.26e+04
sqft_lot15_log	-2.571e+04	3663.653	-7.017	0.000	-3.29e+04	-1.85e+04

Omnibus:	4006.615	**Durbin-Watson:**	1.980
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	13551.892
Skew:	0.969	**Prob(JB):**	0.00
Kurtosis:	6.457	**Cond. No.**	1.14e+09

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.14e+09. This might indicate that there are strong multicollinearity or other numerical problems.

Make Observatiosn!!! about baseline model

In [144]:
```
X = df_no_outliers.drop('price', axis =1)
y = df_no_outliers['price']
```
executed in 30ms, finished 20:56:17 2021-06-12

In [145]:
```
X_train, X_test, y_train, y_test = train_test_split(X,
                                    y, test_size = 0.25, random_state =
```
executed in 28ms, finished 20:56:17 2021-06-12

In [146]:
```
len(X_train)
```
executed in 20ms, finished 20:56:17 2021-06-12

Out[146]: 15528

In [147]:
```
len(X_test)
```
executed in 23ms, finished 20:56:17 2021-06-12

Out[147]: 5176

In [148]:
```
model_5 = LinearRegression()
```
executed in 24ms, finished 20:56:17 2021-06-12

In [149]:
```
model_5.fit(X_train, y_train)
```
executed in 41ms, finished 20:56:17 2021-06-12

Out[149]: LinearRegression()

In [150]:
```
y_pred = model_5.predict(X_test)
```
executed in 23ms, finished 20:56:17 2021-06-12

In [151]:
```
1 residuals = y_pred - y_test
```
executed in 20ms, finished 20:56:17 2021-06-12

In [152]:
```
1 X.columns
```
executed in 25ms, finished 20:56:17 2021-06-12

Out[152]:
```
Index(['id', 'bedrooms', 'bathrooms', 'sqft_lot', 'floors', 'waterfront',
       'view', 'grade', 'sqft_above', 'yr_built', 'yr_renovated', 'zipcod
e',
       'lat', 'long', 'sqft_living15', 'sqft_lot15', 'month_year',
       'new_sqft_basement', 'sqft_lot_log', 'sqft_above_log',
       'sqft_living15_log', 'sqft_lot15_log'],
      dtype='object')
```

In [153]:
```
1 model_5.coef_
```
executed in 25ms, finished 20:56:17 2021-06-12

Out[153]:
```
array([-1.29713817e-06, -1.08354402e+04,  2.91168670e+04,  3.72802881e-01,
        7.29587019e+03,  2.28787949e+05,  3.94568878e+04,  8.41600635e+04,
        9.85572747e+01, -2.35713955e+03,  1.70691264e+01, -4.01583487e+02,
        5.51343232e+05, -7.16376498e+04,  8.89638158e+01,  2.33054760e-01,
        2.15505748e+02,  9.09599652e+01, -1.40943126e+04, -1.97579405e+03,
       -7.40448630e+04, -2.14401882e+04])
```

In [154]:
```
model_5 = ols(formula = formula, data = df_no_outliers).fit()
model_5.summary()
```

executed in 239ms, finished 20:56:17 2021-06-12

Out[154]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.713
Model:	OLS	**Adj. R-squared:**	0.713
Method:	Least Squares	**F-statistic:**	2444.
Date:	Sat, 12 Jun 2021	**Prob (F-statistic):**	0.00
Time:	20:56:17	**Log-Likelihood:**	-2.7354e+05
No. Observations:	20704	**AIC:**	5.471e+05
Df Residuals:	20682	**BIC:**	5.473e+05
Df Model:	21		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-3.71e+07	4.65e+06	-7.984	0.000	-4.62e+07	-2.8e+07
bedrooms	-1.002e+04	1501.000	-6.673	0.000	-1.3e+04	-7073.604
bathrooms	2.966e+04	2301.600	12.885	0.000	2.51e+04	3.42e+04
sqft_lot	0.2937	0.039	7.443	0.000	0.216	0.371
floors	8222.3394	2807.416	2.929	0.003	2719.583	1.37e+04
waterfront	2.256e+05	1.71e+04	13.215	0.000	1.92e+05	2.59e+05
view	4.014e+04	1509.004	26.602	0.000	3.72e+04	4.31e+04
grade	8.521e+04	1493.737	57.046	0.000	8.23e+04	8.81e+04
sqft_above	99.3024	6.129	16.203	0.000	87.290	111.315
yr_built	-2411.7094	47.123	-51.179	0.000	-2504.075	-2319.344
yr_renovated	13.2034	2.749	4.804	0.000	7.816	18.591
zipcode	-384.4474	22.271	-17.262	0.000	-428.100	-340.795
lat	5.471e+05	7200.628	75.974	0.000	5.33e+05	5.61e+05
long	-7.01e+04	8994.424	-7.794	0.000	-8.77e+04	-5.25e+04
sqft_living15	87.6733	8.410	10.425	0.000	71.189	104.158
sqft_lot15	0.3625	0.065	5.546	0.000	0.234	0.491
month_year	224.9249	20.799	10.814	0.000	184.157	265.693
new_sqft_basement	90.5068	3.147	28.759	0.000	84.338	96.675
sqft_lot_log	-1.021e+04	3274.865	-3.118	0.002	-1.66e+04	-3791.837
sqft_above_log	-3633.0193	1.12e+04	-0.325	0.745	-2.55e+04	1.83e+04
sqft_living15_log	-7.462e+04	1.63e+04	-4.566	0.000	-1.07e+05	-4.26e+04
sqft_lot15_log	-2.571e+04	3663.653	-7.017	0.000	-3.29e+04	-1.85e+04

Omnibus:	4006.615	**Durbin-Watson:**	1.980
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	13551.892
Skew:	0.969	**Prob(JB):**	0.00
Kurtosis:	6.457	**Cond. No.**	1.14e+09

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.14e+09. This might indicate that there are strong multicollinearity or other numerical problems.

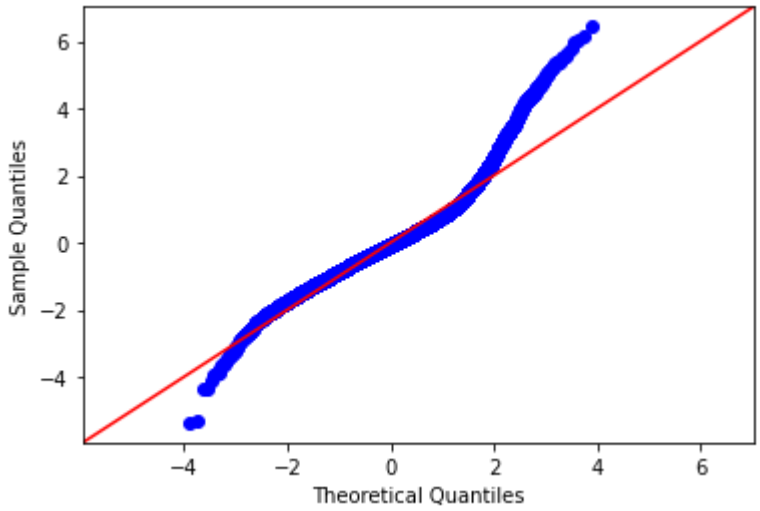
In [155]:
```
mse_train = mean_squared_error (y_train, model_5.predict(X_train))
mse_test = mean_squared_error (y_test, y_pred)

print('Train RMSE:', np.sqrt(mse_train))
print('Test RMSE:', np.sqrt(mse_test))
```

executed in 117ms, finished 20:56:18 2021-06-12

```
Train RMSE: 133098.94416690725
Test RMSE: 130243.7358576574
```

In [156]:
```
fig = sm.graphics.qqplot(model_5.resid, dist = stats.norm, line = '45',
```

executed in 461ms, finished 20:56:18 2021-06-12



In [157]:
```
df_no_outliers.price.mean()
```

executed in 38ms, finished 20:56:18 2021-06-12

Out[157]: 499889.77622681606

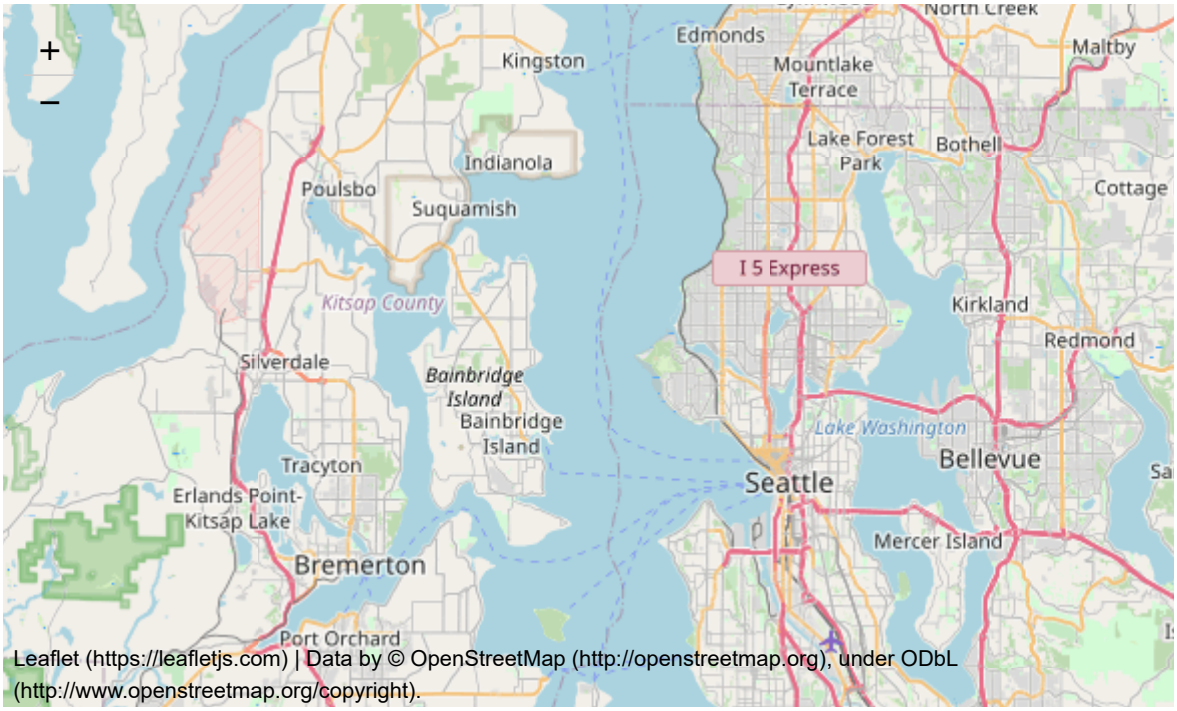
5.5 Model 6 - Zip code revised

In [158]:
```
1 kc_coord = (df.lat.mean(), df.long.mean())
```
executed in 11ms, finished 20:56:18 2021-06-12

In [159]:
```
1 kc_map = folium.Map(location = kc_coord, )
```
executed in 49ms, finished 20:56:18 2021-06-12

In [160]:
```
1 kc_map
```
executed in 40ms, finished 20:56:18 2021-06-12

Out[160]:



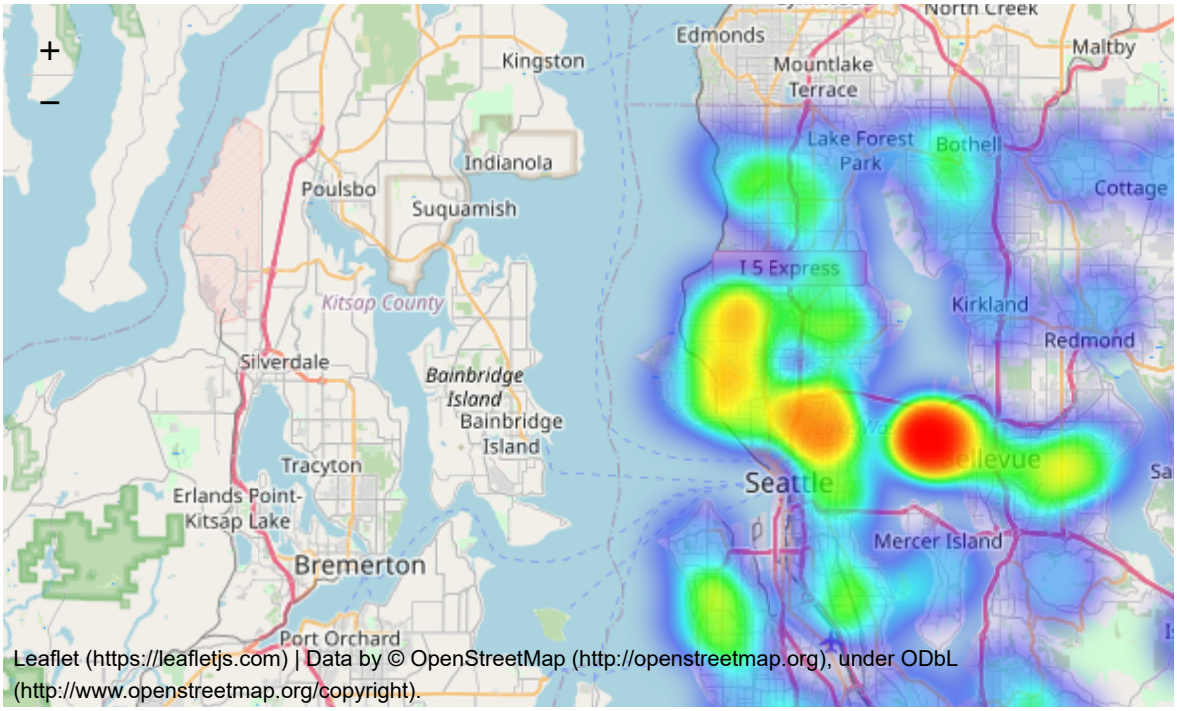
In [161]:
```
1 data = df.groupby('zipcode').median()[['lat', 'long',
2                                        'price']].values.tolist()
```
executed in 59ms, finished 20:56:18 2021-06-12

In [162]:
```
1 from folium.plugins import HeatMap
2
3 HeatMap(data = data).add_to(kc_map)
```
executed in 316ms, finished 20:56:19 2021-06-12

Out[162]:
```
<folium.plugins.heat_map.HeatMap at 0x22d5c0366d0>
```

In [163]:
```
1 kc_map
```
executed in 31ms, finished 20:56:19 2021-06-12

Out[163]:



In [164]:
```
1 df.lat.min()
```
executed in 21ms, finished 20:56:19 2021-06-12

Out[164]: 47.1559

In [165]:
```
1 df.long.max()
```
executed in 15ms, finished 20:56:19 2021-06-12

Out[165]: -121.315

In [166]:
```
1 df.long.min()
```
executed in 30ms, finished 20:56:19 2021-06-12

Out[166]: -122.51899999999999

In [167]:
```
1 df.lat.max()
```
executed in 33ms, finished 20:56:19 2021-06-12

Out[167]: 47.7776

In [168]:
```
1 list(range(-122, 48,9))
```
executed in 15ms, finished 20:56:19 2021-06-12

Out[168]:
```
[-122,
 -113,
 -104,
 -95,
 -86,
 -77,
 -68,
 -59,
 -50,
 -41,
 -32,
 -23,
 -14,
 -5,
 4,
 13,
 22,
 31,
 40]
```

In [169]:
```
1 ((-122+48)/2)/4
2
```
executed in 12ms, finished 20:56:19 2021-06-12

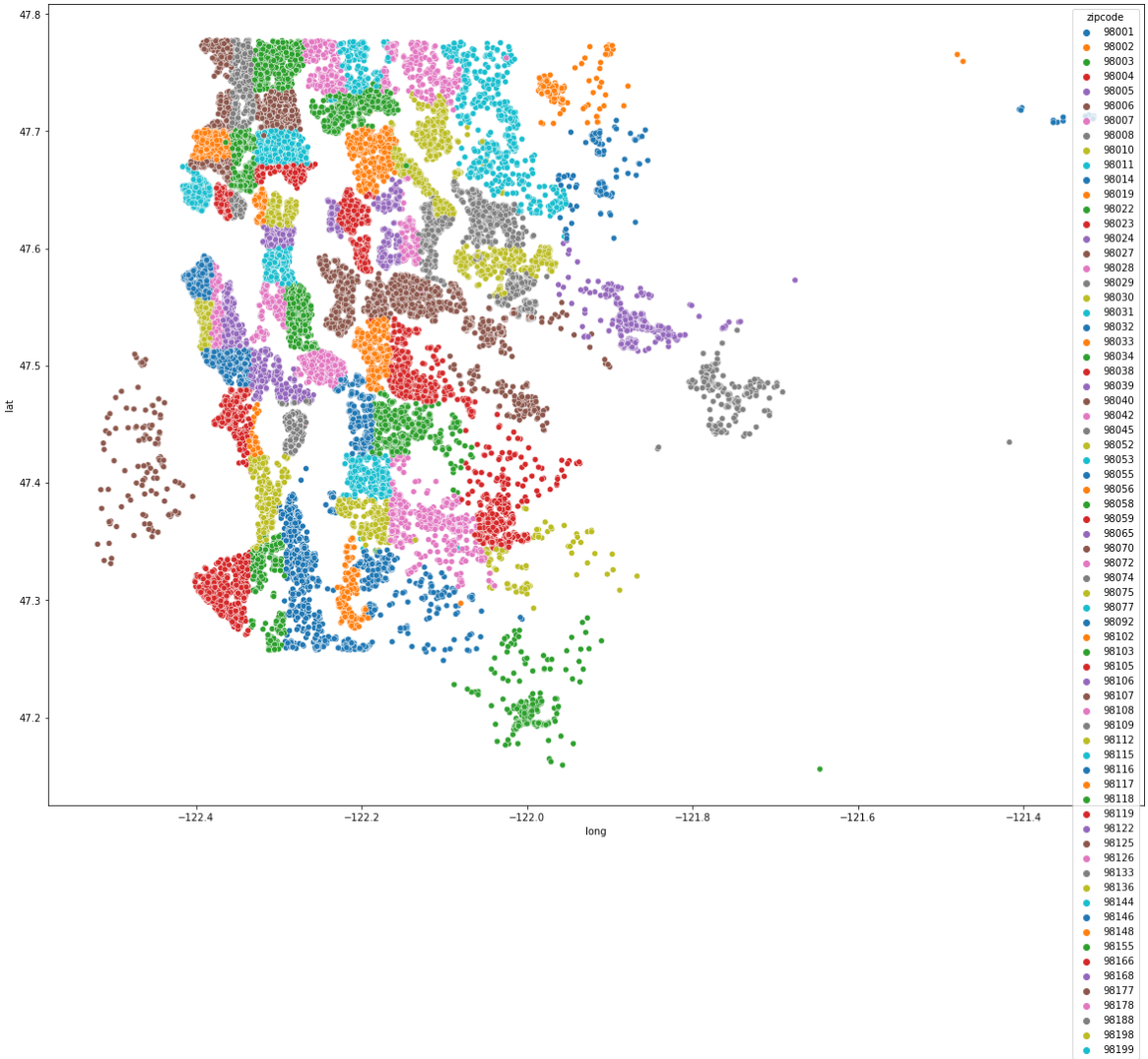
Out[169]:
```
-9.25
```

In [170]:

```
figure, ax= plt.subplots(figsize = (20,15))
sns.scatterplot('long', 'lat', data = df, hue = "zipcode", palette = 'tal
plt.show()
```

executed in 11.2s, finished 20:56:30 2021-06-12

```
C:\Users\laure\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futur
eWarning: 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 misinterpr
etation.
  warnings.warn(
```



In [171]:

```
# create a list of our conditions
conditions = [
    (df['long'] < -122.4) & (df['lat'] < 47.3),
    (df['long'] < -122.4) & (df['lat'] >= 47.3) & (df['lat'] < 47.4),
    (df['long'] < -122.4) & (df['lat'] >= 47.4) & (df['lat'] < 47.5),
    (df['long'] < -122.4) & (df['lat'] >= 47.5) & (df['lat'] < 47.6),
    (df['long'] < -122.4) & (df['lat'] >= 47.6) & (df['lat'] < 47.7),
    (df['long'] < -122.4) & (df['lat'] >= 47.7 ),
    (df['long'] >= -122.4) & (df['long'] < -122.232)
    & (df['lat'] < 47.3),
    (df['long'] >= -122.4) & (df['long'] < -122.232)
    & (df['lat'] >= 47.3) & (df['lat'] <= 47.4),
    (df['long'] >= -122.4) & (df['long'] < -122.232)
    & (df['lat'] >= 47.4) & (df['lat'] <= 47.5),
    (df['long'] >= -122.4) & (df['long'] < -122.232)
    & (df['lat'] >= 47.5) & (df['lat'] <= 47.6),
    (df['long'] >= -122.4) & (df['long'] < -122.232)
    & (df['lat'] >= 47.6) & (df['lat'] <= 47.7),
    (df['long'] >= -122.4) & (df['long'] < -122.232)
    & (df['lat'] >= 47.7),
    (df['long'] >= -122.232) & (df['long'] < -122.066)
    & (df['lat'] < 47.3),
    (df['long'] >= -122.232) & (df['long'] < -122.066)
    & (df['lat'] >= 47.3) & (df['lat'] <= 47.4),
    (df['long'] >= -122.232) & (df['long'] < -122.066)
    & (df['lat'] >= 47.4) & (df['lat'] <= 47.5),
    (df['long'] >= -122.232) & (df['long'] < -122.066)
    & (df['lat'] >= 47.5) & (df['lat'] <= 47.6),
    (df['long'] >= -122.232) & (df['long'] < -122.066)
    & (df['lat'] >= 47.6) & (df['lat'] <= 47.7),
    (df['long'] >= -122.232) & (df['long'] < -122.066)
    & (df['lat'] >= 47.7),
    (df['long'] >= -122.066) & (df['long'] < -121.9) & (df['lat'] < 47.3
    (df['long'] >= -122.066) & (df['long'] < -121.9) & (df['lat'] >= 47.
    & (df['lat'] <= 47.4),
    (df['long'] >= -122.066) & (df['long'] < -121.9) & (df['lat'] >= 47.4
    & (df['lat'] <= 47.5),
    (df['long'] >= -122.066) & (df['long'] < -121.9) & (df['lat'] >= 47.5
    & (df['lat'] <= 47.6),
    (df['long'] >= -122.066) & (df['long'] < -121.9) & (df['lat'] >= 47.6
    & (df['lat'] <= 47.7),
    (df['long'] >= -122.066) & (df['long'] < -121.9) & (df['lat'] >= 47.7
    (df['long'] >= -121.9) & (df['lat'] < 47.3),
    (df['long'] >= -121.9) & (df['lat'] >= 47.3) & (df['lat'] < 47.4),
    (df['long'] >= -121.9) & (df['lat'] >= 47.4) & (df['lat'] < 47.5),
    (df['long'] >= -121.9) & (df['lat'] >= 47.5) & (df['lat'] < 47.6),
    (df['long'] >= -121.9) & (df['lat'] >= 47.6) & (df['lat'] < 47.7),
    (df['long'] >= -121.9) & (df['lat'] >= 47.7 )]

# create a list of the values we want to assign for each condition
values = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
          19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]

# create a new column and use np.select to assign
#values to it using our lists as arguments
df['zone'] = np.select(conditions, values)
```

```
57
58 # display updated DataFrame
59 df.head()
60
```

executed in 273ms, finished 20:56:30 2021-06-12

Out[171]:

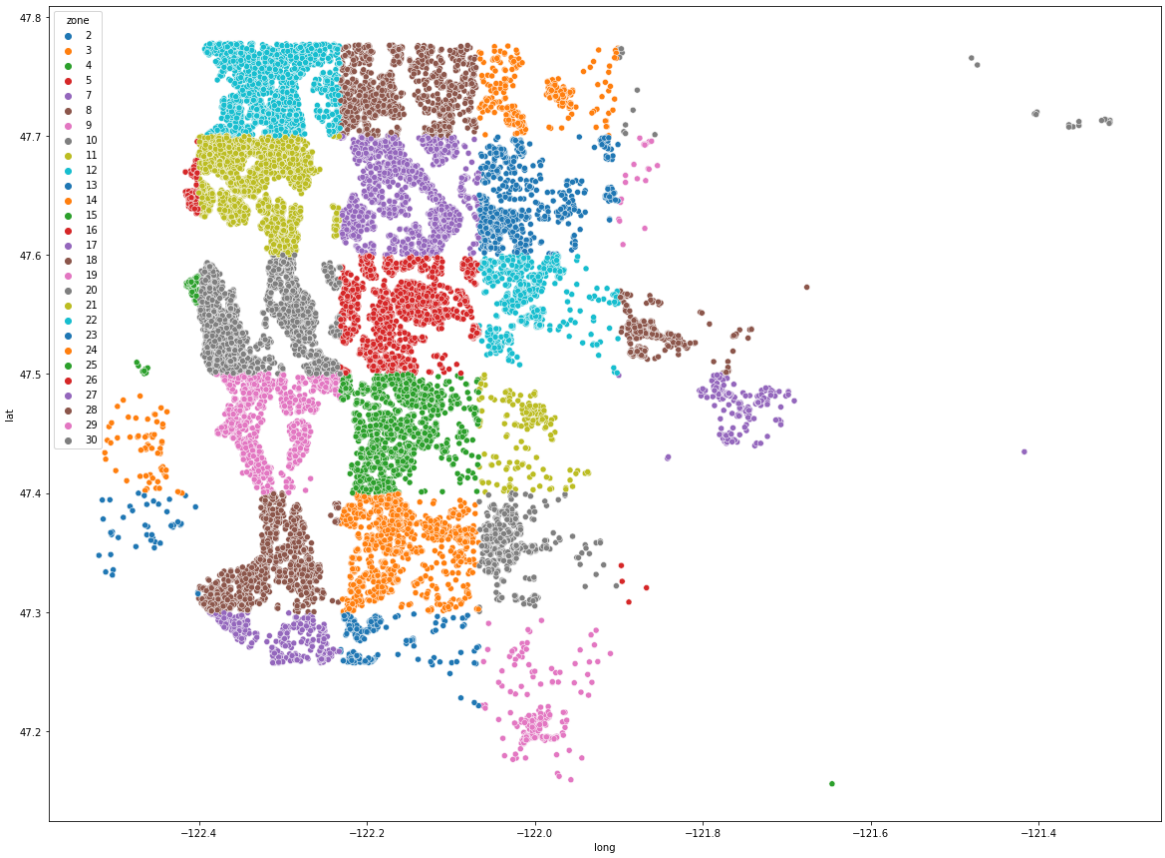
	id	price	bedrooms	bathrooms	sqft_lot	floors	waterfront	view	grade	sqft
0	7129300520	221900.0	3	1.00	5650	1.0	0.0	0.0	7	
1	6414100192	538000.0	3	2.25	7242	2.0	0.0	0.0	7	
2	5631500400	180000.0	2	1.00	10000	1.0	0.0	0.0	6	
3	2487200875	604000.0	4	3.00	5000	1.0	0.0	0.0	7	
4	1954400510	510000.0	3	2.00	8080	1.0	0.0	0.0	8	

In [172]:

```
figure, ax= plt.subplots(figsize = (20,15))
sns.scatterplot('long', 'lat', data = df, hue = "zone", palette = 'tab10
plt.show()
```

executed in 4.14s, finished 20:56:34 2021-06-12

```
C:\Users\laure\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futur
eWarning: 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 misinterpr
etation.
  warnings.warn(
```



In [173]:
```
df_zone = df[df.bathrooms >= 1]
df_zone
```

executed in 110ms, finished 20:56:34 2021-06-12

Out[173]:

	id	price	bedrooms	bathrooms	sqft_lot	floors	waterfront	view	grade
0	7129300520	221900.0	3	1.00	5650	1.0	0.0	0.0	7
1	6414100192	538000.0	3	2.25	7242	2.0	0.0	0.0	7
2	5631500400	180000.0	2	1.00	10000	1.0	0.0	0.0	6
3	2487200875	604000.0	4	3.00	5000	1.0	0.0	0.0	7
4	1954400510	510000.0	3	2.00	8080	1.0	0.0	0.0	8
...	...	...	...	...	...	...	...	...	...
21590	7936000429	1010000.0	4	3.50	7200	2.0	0.0	0.0	9
21591	2997800021	475000.0	3	2.50	1294	2.0	0.0	0.0	8
21592	263000018	360000.0	3	2.50	1131	3.0	0.0	0.0	8
21593	6600060120	400000.0	4	2.50	5813	2.0	0.0	0.0	8
21595	291310100	400000.0	3	2.50	2388	2.0	0.0	0.0	8

21188 rows × 20 columns

In [174]:
```
df_zone = df[df.price <= 1500000 ]
```

executed in 19ms, finished 20:56:35 2021-06-12

In [175]:
```
1 df_zone.info()
```
executed in 41ms, finished 20:56:35 2021-06-12

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20779 entries, 0 to 21596
Data columns (total 20 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   id                 20779 non-null  int64  
 1   price              20779 non-null  float64
 2   bedrooms           20779 non-null  int64  
 3   bathrooms          20779 non-null  float64
 4   sqft_lot           20779 non-null  int64  
 5   floors             20779 non-null  float64
 6   waterfront         20779 non-null  float64
 7   view               20779 non-null  float64
 8   grade              20779 non-null  int64  
 9   sqft_above         20779 non-null  int64  
 10  yr_built           20779 non-null  int64  
 11  yr_renovated       20779 non-null  float64
 12  zipcode            20779 non-null  int64  
 13  lat                20779 non-null  float64
 14  long               20779 non-null  float64
 15  sqft_living15      20779 non-null  int64  
 16  sqft_lot15         20779 non-null  int64  
 17  month_year         20779 non-null  int32  
 18  new_sqft_basement  20779 non-null  int64  
 19  zone               20779 non-null  int32  
dtypes: float64(8), int32(2), int64(10)
memory usage: 3.2 MB
```

In [176]:
```
1 df_zone_columns = df_zone.columns
2
3 df_zone_columns
```
executed in 19ms, finished 20:56:35 2021-06-12

Out[176]:
```
Index(['id', 'price', 'bedrooms', 'bathrooms', 'sqft_lot', 'floors',
       'waterfront', 'view', 'grade', 'sqft_above', 'yr_built', 'yr_renovat
ed',
       'zipcode', 'lat', 'long', 'sqft_living15', 'sqft_lot15', 'month_yea
r',
       'new_sqft_basement', 'zone'],
      dtype='object')
```

In [177]:

```
for i in df_zone_columns:
    figure, ax= plt.subplots(figsize = (10,10))
    sns.distplot(df, x = df[i])
    plt.title(i)
```

executed in 19.0s, finished 20:56:54 2021-06-12

```
C:\Users\laure\anaconda3\lib\site-packages\seaborn\distributions.py:255
1: FutureWarning: `distplot` is a deprecated function and will be remov
ed 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)
C:\Users\laure\anaconda3\lib\site-packages\seaborn\distributions.py:255
1: FutureWarning: `distplot` is a deprecated function and will be remov
ed 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)
C:\Users\laure\anaconda3\lib\site-packages\seaborn\distributions.py:255
1: FutureWarning: `distplot` is a deprecated function and will be remov
ed 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)
C:\Users\laure\anaconda3\lib\site-packages\seaborn\distributions.py:255
```

In [178]:
```
df_zone['sqft_lot_log'] = np.log(df_zone['sqft_lot'])
df_zone['sqft_above_log'] = np.log(df_zone['sqft_above'])
df_zone['sqft_living15_log'] = np.log(df_zone['sqft_living15'])
df_zone['sqft_lot15_log'] = np.log(df_zone['sqft_lot15'])
```

executed in 34ms, finished 20:56:54 2021-06-12

```
<ipython-input-178-f9011d827b30>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df_zone['sqft_lot_log'] = np.log(df_zone['sqft_lot'])
<ipython-input-178-f9011d827b30>:2: 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.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df_zone['sqft_above_log'] = np.log(df_zone['sqft_above'])
<ipython-input-178-f9011d827b30>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df_zone['sqft_living15_log'] = np.log(df_zone['sqft_living15'])
<ipython-input-178-f9011d827b30>:4: 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.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df_zone['sqft_lot15_log'] = np.log(df_zone['sqft_lot15'])
```

In [179]:
```
waterfront_dummies = pd.get_dummies(df_zone['waterfront'],
                                    prefix = 'waterfront', drop_first = T
```

executed in 26ms, finished 20:56:54 2021-06-12

In [180]:
```
view_dummies = pd.get_dummies(df_zone['view'],
                              prefix = 'view', drop_first = True)
```

executed in 21ms, finished 20:56:54 2021-06-12

In [181]:
```
grade_dummies = pd.get_dummies(df_zone['grade'],
                               prefix = 'grade', drop_first = True)
```
executed in 27ms, finished 20:56:54 2021-06-12

In [182]:
```
zone_dummies = pd.get_dummies(df_zone['zone'],
                              prefix = 'zone', drop_first = True)
```
executed in 16ms, finished 20:56:54 2021-06-12

In [183]:
```
df_zone = pd.concat([df_zone, waterfront_dummies, view_dummies,
                     grade_dummies, zone_dummies], axis = 1 )
```
executed in 40ms, finished 20:56:54 2021-06-12

In [184]:
```
df_zone.head()
```
executed in 64ms, finished 20:56:54 2021-06-12

Out[184]:

	id	price	bedrooms	bathrooms	sqft_lot	floors	waterfront	view	grade	sqft
0	7129300520	221900.0	3	1.00	5650	1.0	0.0	0.0	7	
1	6414100192	538000.0	3	2.25	7242	2.0	0.0	0.0	7	
2	5631500400	180000.0	2	1.00	10000	1.0	0.0	0.0	6	
3	2487200875	604000.0	4	3.00	5000	1.0	0.0	0.0	7	
4	1954400510	510000.0	3	2.00	8080	1.0	0.0	0.0	8	

5 rows × 65 columns

In [185]:
```
df_zone.columns = ['id', 'price', 'bedrooms', 'bathrooms', 'sqft_lot', '
       'waterfront', 'view', 'grade', 'sqft_above', 'yr_built', 'yr_reno
       'zipcode', 'lat', 'long', 'sqft_living15', 'sqft_lot15', 'month_y
       'new_sqft_basement', 'zone', 'sqft_lot_log', 'sqft_above_log',
       'sqft_living15_log', 'sqft_lot15_log', 'waterfront_1', 'view_1',
       'view_2', 'view_3', 'view_4', 'grade_4', 'grade_5', 'grade_6',
       'grade_7', 'grade_8', 'grade_9', 'grade_10', 'grade_11', 'grade_1
       'zone_3', 'zone_4', 'zone_5', 'zone_7', 'zone_8', 'zone_9', 'zone_
       'zone_11', 'zone_12', 'zone_13', 'zone_14', 'zone_15', 'zone_16',
       'zone_17', 'zone_18', 'zone_19', 'zone_20', 'zone_21', 'zone_22',
       'zone_23', 'zone_24', 'zone_25', 'zone_26', 'zone_27', 'zone_28',
       'zone_29', 'zone_30']
```
executed in 25ms, finished 20:56:54 2021-06-12

In [186]:

```
1 df_zone = df_zone.drop(columns = ['sqft_lot', 'sqft_above', 'sqft_living
2                                   'sqft_lot15', 'zipcode', 'zone', 'grad
```

executed in 34ms, finished 20:56:54 2021-06-12

In [187]:

```
1 df_zone = df_zone.drop(columns = ['zone_3', 'zone_7', 'zone_9', 'zone_13
```

executed in 16ms, finished 20:56:54 2021-06-12

In [188]:
```
1 df_zone.info()
```

executed in 55ms, finished 20:56:54 2021-06-12

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20779 entries, 0 to 21596
Data columns (total 48 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 20779 non-null  int64
 1   price              20779 non-null  float64
 2   bedrooms           20779 non-null  int64
 3   bathrooms          20779 non-null  float64
 4   floors             20779 non-null  float64
 5   yr_built           20779 non-null  int64
 6   yr_renovated       20779 non-null  float64
 7   month_year         20779 non-null  int32
 8   new_sqft_basement  20779 non-null  int64
 9   sqft_lot_log       20779 non-null  float64
 10  sqft_above_log     20779 non-null  float64
 11  sqft_living15_log  20779 non-null  float64
 12  sqft_lot15_log     20779 non-null  float64
 13  waterfront_1       20779 non-null  uint8
 14  view_1             20779 non-null  uint8
 15  view_2             20779 non-null  uint8
 16  view_3             20779 non-null  uint8
 17  view_4             20779 non-null  uint8
 18  grade_4            20779 non-null  uint8
 19  grade_5            20779 non-null  uint8
 20  grade_6            20779 non-null  uint8
 21  grade_7            20779 non-null  uint8
 22  grade_8            20779 non-null  uint8
 23  grade_9            20779 non-null  uint8
 24  grade_10           20779 non-null  uint8
 25  grade_11           20779 non-null  uint8
 26  grade_12           20779 non-null  uint8
 27  zone_4             20779 non-null  uint8
 28  zone_5             20779 non-null  uint8
 29  zone_8             20779 non-null  uint8
 30  zone_10            20779 non-null  uint8
 31  zone_11            20779 non-null  uint8
 32  zone_12            20779 non-null  uint8
 33  zone_16            20779 non-null  uint8
 34  zone_17            20779 non-null  uint8
 35  zone_18            20779 non-null  uint8
 36  zone_19            20779 non-null  uint8
 37  zone_20            20779 non-null  uint8
 38  zone_21            20779 non-null  uint8
 39  zone_22            20779 non-null  uint8
 40  zone_23            20779 non-null  uint8
 41  zone_24            20779 non-null  uint8
 42  zone_25            20779 non-null  uint8
 43  zone_26            20779 non-null  uint8
 44  zone_27            20779 non-null  uint8
 45  zone_28            20779 non-null  uint8
 46  zone_29            20779 non-null  uint8
 47  zone_30            20779 non-null  uint8
```

```
dtypes: float64(8), int32(1), int64(4), uint8(35)
memory usage: 2.8 MB
```

In [ ]:

```
1
```

executed in 9ms, finished 19:34:21 2021-06-11

In [189]:

```
#Dependent variable
outcome = 'price'
#Independent variables
predictors = df_zone.drop(['price', 'id'], axis = 1)
predictor_variables = '+'.join(predictors.columns)
formula = outcome + '~' + predictor_variables


```

executed in 34ms, finished 20:56:54 2021-06-12

In [190]:

```
formula
```

executed in 18ms, finished 20:56:54 2021-06-12

Out[190]:

```
'price~bedrooms+bathrooms+floors+yr_built+yr_renovated+month_year+new_sqft_
basement+sqft_lot_log+sqft_above_log+sqft_living15_log+sqft_lot15_log+water
front_1+view_1+view_2+view_3+view_4+grade_4+grade_5+grade_6+grade_7+grade_8
+grade_9+grade_10+grade_11+grade_12+zone_4+zone_5+zone_8+zone_10+zone_11+zo
ne_12+zone_16+zone_17+zone_18+zone_19+zone_20+zone_21+zone_22+zone_23+zone_
24+zone_25+zone_26+zone_27+zone_28+zone_29+zone_30'
```

In [191]:

```
model_6 = ols(formula = formula, data = df_zone).fit()
model_6 .summary()
```

executed in 406ms, finished 20:56:54 2021-06-12

Out[191]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.784
Model:	OLS	**Adj. R-squared:**	0.784
Method:	Least Squares	**F-statistic:**	1637.
Date:	Sat, 12 Jun 2021	**Prob (F-statistic):**	0.00
Time:	20:56:54	**Log-Likelihood:**	-2.7157e+05
No. Observations:	20779	**AIC:**	5.432e+05
Df Residuals:	20732	**BIC:**	5.436e+05
Df Model:	46		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-4.788e+07	3.63e+06	-13.177	0.000	-5.5e+07	-4.08e+07

In [192]:
```
df_zone.columns
```
executed in 25ms, finished 20:56:54 2021-06-12

Out[192]:
```
Index(['id', 'price', 'bedrooms', 'bathrooms', 'floors', 'yr_built',
       'yr_renovated', 'month_year', 'new_sqft_basement', 'sqft_lot_log',
       'sqft_above_log', 'sqft_living15_log', 'sqft_lot15_log', 'waterfront
_1',
       'view_1', 'view_2', 'view_3', 'view_4', 'grade_4', 'grade_5', 'grade
_6',
       'grade_7', 'grade_8', 'grade_9', 'grade_10', 'grade_11', 'grade_12',
       'zone_4', 'zone_5', 'zone_8', 'zone_10', 'zone_11', 'zone_12',
       'zone_16', 'zone_17', 'zone_18', 'zone_19', 'zone_20', 'zone_21',
       'zone_22', 'zone_23', 'zone_24', 'zone_25', 'zone_26', 'zone_27',
       'zone_28', 'zone_29', 'zone_30'],
      dtype='object')
```

Make Observatiosn!!! about baseline model

In [193]:
```
X = df_zone.drop('price', axis =1)
y = df_zone['price']
```
executed in 29ms, finished 20:56:54 2021-06-12

In [194]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.
                                                    random_state = 25)
```
executed in 40ms, finished 20:56:54 2021-06-12

In [195]:
```
len(X_train)
```
executed in 23ms, finished 20:56:55 2021-06-12

Out[195]:
```
15584
```

In [196]:
```
len(X_test)
```
executed in 16ms, finished 20:56:55 2021-06-12

Out[196]:
```
5195
```

In [197]:
```
model_6 = LinearRegression()
```
executed in 21ms, finished 20:56:55 2021-06-12

In [198]:
```
model_6.fit(X_train, y_train)
```
executed in 60ms, finished 20:56:55 2021-06-12

Out[198]:
```
LinearRegression()
```

In [199]:
```
1 y_pred = model_6.predict(X_test)
```
executed in 33ms, finished 20:56:55 2021-06-12

In [200]:
```
1 residuals = y_pred - y_test
```
executed in 32ms, finished 20:56:55 2021-06-12

In [201]:
```
1 X.columns
```
executed in 23ms, finished 20:56:55 2021-06-12

Out[201]:
```
Index(['id', 'bedrooms', 'bathrooms', 'floors', 'yr_built', 'yr_renovated',
       'month_year', 'new_sqft_basement', 'sqft_lot_log', 'sqft_above_log',
       'sqft_living15_log', 'sqft_lot15_log', 'waterfront_1', 'view_1',
       'view_2', 'view_3', 'view_4', 'grade_4', 'grade_5', 'grade_6',
       'grade_7', 'grade_8', 'grade_9', 'grade_10', 'grade_11', 'grade_12',
       'zone_4', 'zone_5', 'zone_8', 'zone_10', 'zone_11', 'zone_12',
       'zone_16', 'zone_17', 'zone_18', 'zone_19', 'zone_20', 'zone_21',
       'zone_22', 'zone_23', 'zone_24', 'zone_25', 'zone_26', 'zone_27',
       'zone_28', 'zone_29', 'zone_30'],
      dtype='object')
```

In [202]:
```
1 model_6.coef_
```
executed in 18ms, finished 20:56:55 2021-06-12

Out[202]:
```
array([-1.96749671e-07, -5.93537159e+03,  2.66575662e+04, -1.70025624e+04,
       -9.77250501e+02,  1.78814579e+01,  2.49039307e+02,  7.64256075e+01,
        1.83623664e+04,  1.93364383e+05,  6.65906037e+04, -4.62461953e+03,
        2.07378969e+05,  7.95950717e+04,  6.10728600e+04,  1.08916669e+05,
        1.89381502e+05, -1.84943686e+05, -2.18483837e+05, -2.05245523e+05,
       -1.88057514e+05, -1.37578689e+05, -2.85192379e+04,  6.49256256e+04,
        1.83356539e+05,  3.32262389e+05,  2.58502628e+05,  3.74885650e+05,
       -4.73903451e+04,  1.59284589e+05,  3.27667227e+05,  1.35639783e+05,
        2.14832294e+05,  2.77566225e+05,  1.39616921e+05, -2.68217526e+04,
        6.98583902e+03,  4.43271231e+04,  1.75636619e+05,  1.60528519e+05,
        1.25592385e+05,  9.79475124e+04,  4.32435465e+04,  6.34059702e+04,
        9.40086109e+04,  7.27741603e+04,  4.68703146e+04])
```

In [203]:
```
model_6 = ols(formula = formula, data = df_zone).fit()
model_6 .summary()
```

executed in 364ms, finished 20:56:55 2021-06-12

Out[203]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.784
Model:	OLS	**Adj. R-squared:**	0.784
Method:	Least Squares	**F-statistic:**	1637.
Date:	Sat, 12 Jun 2021	**Prob (F-statistic):**	0.00
Time:	20:56:55	**Log-Likelihood:**	-2.7157e+05
No. Observations:	20779	**AIC:**	5.432e+05
Df Residuals:	20732	**BIC:**	5.436e+05
Df Model:	46		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-4.788e+07	3.63e+06	-13.177	0.000	-5.5e+07	-4.08e+07

In [204]:
```
mse_train = mean_squared_error (y_train, model_6.predict(X_train))
mse_test = mean_squared_error (y_test, y_pred)

print('Train RMSE:', np.sqrt(mse_train))
print('Test RMSE:', np.sqrt(mse_test))
```

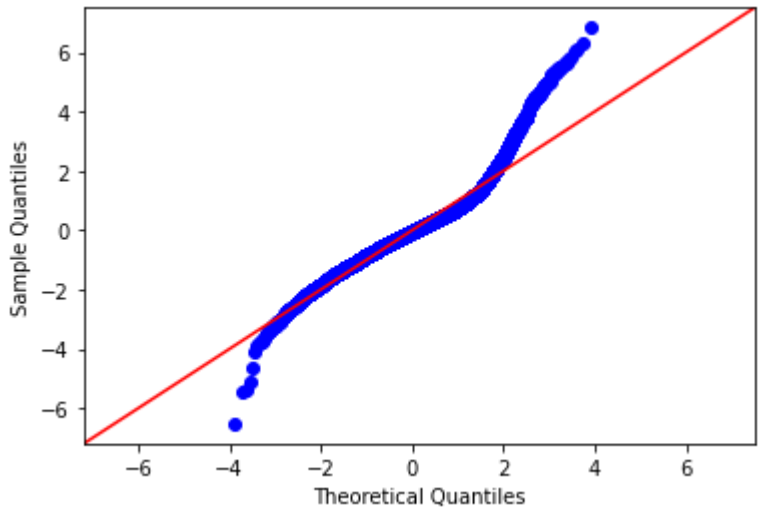
executed in 183ms, finished 20:56:55 2021-06-12

```
Train RMSE: 114709.77278945922
Test RMSE: 115242.927144988
```

In [205]:

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

executed in 500ms, finished 20:56:56 2021-06-12



Even thought this model has the best r value i will not be using it as the final model because the coeficents don't reflect the importance of factors out side of zone.