Phase 2 Project Home Regression King's County Homesales

- Student name: **Daniel M. Smith**
- Student pace: **Full Time**
- Scheduled project review date/time:
- Instructor name: **Abhineet Kulkarni**
- Blog post URL:

0.1 Stakeholder

We are a small company looking for houses to purchase and remodel and either rent or sell in the Kings County, WA housing market. We will gather a list of target houses to inspect as potential investment properties.

1 Obtain data

In [158]:

```
import pandas as pd
import numpy as np

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

executed in 155ms, finished 00:10:39 2021-06-06

Columns

id - unique identified for a house
date - house was sold
price - is prediction target
bedrooms- of Bedrooms/House
bathrooms- of bathrooms/bedrooms
sqft_living - footage of the home
sqft_lot - footage of the lot
floors - Total floors (levels) in house
waterfront - House which has a view to a waterfront
view - Has been viewed
condition - How good the condition is (Overall)
grade - overall grade given to the housing unit, based on King County grading system
sqft_above - square footage of house apart from basement
sqft_basement - square footage of the basement
yr_built - Built Year
yr_renovated - Year when house was renovated
zipcode - zip
lat - Latitude coordinate

long - Longitude coordinate
sqft_living15 - The avg square footage of interior housing living space for the nearest 15 neighbors
sqft_lot15 - The avg square footage of the land lots of the nearest 15 neighbors

In [159]:

```
data.shape
```

executed in 15ms, finished 00:10:39 2021-06-06

Out[159]:

```
(21597, 21)
```

In [160]:

```
data.describe().T
```

executed in 424ms, finished 00:10:39 2021-06-06

Out[160]:

	count	mean	
id	21597.0	4.580474e+09	2.876
price	21597.0	5.402966e+05	3.673
bedrooms	21597.0	3.373200e+00	9.26
bathrooms	21597.0	2.115826e+00	7.68
sqft_living	21597.0	2.080322e+03	9.181
sqft_lot	21597.0	1.509941e+04	4.141
floors	21597.0	1.494096e+00	5.39
waterfront	19221.0	7.595859e-03	8.68
view	21534.0	2.338627e-01	7.65
condition	21597.0	3.409825e+00	6.50
grade	21597.0	7.657915e+00	1.173
sqft_above	21597.0	1.788597e+03	8.277
yr_built	21597.0	1.971000e+03	2.937
yr_renovated	17755.0	8.363678e+01	3.999

	count	mean	
zipcode	21597.0	9.807795e+04	5.351
lat	21597.0	4.756009e+01	1.38
long	21597.0	-1.222140e+02	1.40
sqft_living15	21597.0	1.986620e+03	6.852
sqft_lot15	21597.0	1.275828e+04	2.727

2 Scrub

In [161]:

```
data.info()
```

executed in 61ms, finished 00:10:40 2021-06-06

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtyp
e
---  ------         --------------  ----
-
 0   id             21597 non-null  int6
4
 1   date           21597 non-null  obje
ct
 2   price          21597 non-null  floa
t64
 3   bedrooms       21597 non-null  int6
4
 4   bathrooms      21597 non-null  floa
t64
 5   sqft_living    21597 non-null  int6
4
 6   sqft_lot       21597 non-null  int6
4
 7   floors         21597 non-null  floa
t64
 8   waterfront     19221 non-null  floa
t64
 9   view           21534 non-null  floa
```

```
t64
 10  condition      21597 non-null  int6
4
 11  grade          21597 non-null  int6
4
 12  sqft_above     21597 non-null  int6
4
 13  sqft_basement  21597 non-null  obje
ct
 14  yr_built       21597 non-null  int6
4
 15  yr_renovated   17755 non-null  floa
t64
 16  zipcode        21597 non-null  int6
4
 17  lat            21597 non-null  floa
t64
 18  long           21597 non-null  floa
t64
 19  sqft_living15  21597 non-null  int6
4
 20  sqft_lot15     21597 non-null  int6
4
dtypes: float64(8), int64(11), object(2)
memory usage: 3.5+ MB
```

In [162]:

```
data['waterfront'].sort_values(ascending=False)
```

executed in 121ms, finished 00:10:40 2021-06-06

Out[162]:

```
14760    1.0
11103    1.0
1166     1.0
17753    1.0
300      1.0
        ...
21578    NaN
21582    NaN
21586    NaN
21587    NaN
21595    NaN
Name: waterfront, Length: 21597, dtype:
float64
```

In [163]:

```
data.head(10)
```

executed in 248ms, finished 00:10:40 2021-06-06

Out[163]:

	id	date	price	bedrooms
0	7129300520	10/13/2014	221900.0	3
1	6414100192	12/9/2014	538000.0	3
2	5631500400	2/25/2015	180000.0	2
3	2487200875	12/9/2014	604000.0	4
4	1954400510	2/18/2015	510000.0	3
5	7237550310	5/12/2014	1230000.0	4
6	1321400060	6/27/2014	257500.0	3
7	2008000270	1/15/2015	291850.0	3
8	2414600126	4/15/2015	229500.0	3
9	3793500160	3/12/2015	323000.0	3

10 rows × 21 columns

In [164]:

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

executed in 116ms, finished 00:10:40 2021-06-06

Out[164]:

```
0.0    19075
1.0      146
Name: waterfront, dtype: int64
```

In [165]:

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

executed in 120ms, finished 00:10:40 2021-06-06

Out[165]:

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

2.0.0.1 Observations:

Observations: 11% of waterfront is NaN
18% of yr_renovated is NaN
.2% of view is NaN
We will set Nan to 0.0 for Nans

Handling Nan Values

In [166]:

```
data['waterfront'] = data['waterfront'].fillna(0.0)
```

executed in 93ms, finished 00:10:40 2021-06-06

In [167]:

```
data['yr_renovated'] = data['yr_renovated'].fillna(0.0
```

executed in 117ms, finished 00:10:40 2021-06-06

In [168]:

```
data['view'] = data['view'].fillna(0.0)
```

executed in 160ms, finished 00:10:41 2021-06-06

Rename date column to date last sold

In [169]:

```
data['month'] = pd.to_datetime(data['date']).dt.month
data.drop(columns=['date'], inplace=True, axis = 1)
```

executed in 275ms, finished 00:10:41 2021-06-06

In [170]:

```
data.info()
```

executed in 112ms, finished 00:10:41 2021-06-06

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtyp
e
---  ------         --------------  ----
-
 0   id             21597 non-null  int6
4
 1   price          21597 non-null  floa
t64
 2   bedrooms       21597 non-null  int6
4
 3   bathrooms      21597 non-null  floa
t64
 4   sqft_living    21597 non-null  int6
4
 5   sqft_lot       21597 non-null  int6
4
 6   floors         21597 non-null  floa
t64
 7   waterfront     21597 non-null  floa
t64
 8   view           21597 non-null  floa
t64
 9   condition      21597 non-null  int6
```

```
4
 10  grade          21597 non-null  int6
4
 11  sqft_above     21597 non-null  int6
4
 12  sqft_basement  21597 non-null  obje
ct
 13  yr_built       21597 non-null  int6
4
 14  yr_renovated   21597 non-null  floa
t64
 15  zipcode        21597 non-null  int6
4
 16  lat            21597 non-null  floa
t64
 17  long           21597 non-null  floa
t64
 18  sqft_living15  21597 non-null  int6
4
 19  sqft_lot15     21597 non-null  int6
4
 20  month          21597 non-null  int6
4
dtypes: float64(8), int64(12), object(1)
memory usage: 3.5+ MB
```

In [171]:

```
#Covers 70 zipcodes
data.zipcode.value_counts()
```

executed in 94ms, finished 00:10:41 2021-06-06

Out[171]:

```
98103    602
98038    589
98115    583
98052    574
98117    553
        ...
98102    104
98010    100
98024     80
98148     57
98039     50
Name: zipcode, Length: 70, dtype: int64
```

Observations:sqft_basement should be converted to a float64 not an object also has ? which should be set to 0.0

In [172]:

```
#replacing '?'
data['sqft_basement'] = [0.0 if value == '?' else valu
```

executed in 142ms, finished 00:10:41 2021-06-06

In [173]:

```
data.loc[data["sqft_basement"] == '?']
```

executed in 195ms, finished 00:10:41 2021-06-06

Out[173]:

id	price	bedrooms	bathrooms	sqft_living	sq

0 rows × 21 columns

In [174]:

```
data["sqft_basement"] = pd.to_numeric(data["sqft_basem
```

executed in 160ms, finished 00:10:42 2021-06-06

Price column seems to be the price since last sold. This could be

an oout of date column if it has been a long while since last sole. We should make a 2 features, 1. age of home and 2. how many years since last sold.

Sqft_living15 seems to be an average of the 15 properties around sqft_lot15 seems to be average of 15 lots around

In [175]:

```
data.tail(5)
```

executed in 93ms, finished 00:10:42 2021-06-06

Out[175]:

	id	price	bedrooms	bathroor
21592	263000018	360000.0	3	2.
21593	6600060120	400000.0	4	2.
21594	1523300141	402101.0	2	0.
21595	291310100	400000.0	3	2.
21596	1523300157	325000.0	2	0.

5 rows × 21 columns

In [176]:

```
data.describe()
```

executed in 417ms, finished 00:10:42 2021-06-06

Out[176]:

	id	price	bedrooms
count	2.159700e+04	2.159700e+04	21597.000000
mean	4.580474e+09	5.402966e+05	3.373200
std	2.876736e+09	3.673681e+05	0.926299
min	1.000102e+06	7.800000e+04	1.000000
25%	2.123049e+09	3.220000e+05	3.000000
50%	3.904930e+09	4.500000e+05	3.000000
75%	7.308900e+09	6.450000e+05	4.000000
max	9.900000e+09	7.700000e+06	33.000000

8 rows × 21 columns

2.0.0.2 Observations

The data is about houses in zipcodes around Seattle, WA.
The dependent or response variable is Price and min value is 78000 and max is 7,700,000 and a mean of 540296 and median of 450000.
sq_ft living ranges from 370 to 13540 sf ft.
bedrooms range from 1 to 33 with a mean of 3.37 and median of 3.
sqft_lot range from 520 to 1651359.0 with mean of 15099 and median of 7618.

categorical data

zipcode-70 zipcodes
grade-grade from kings county grading system 3-13
condition-1 to 5 5 is best
view- has been viewed 0-4
waterfront- view of waterfront 0-1 being yes

In [177]:

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

executed in 23ms, finished 00:10:42 2021-06-06

Out[177]:

```
0.0       20853
2014.0       73
2003.0       31
2013.0       31
2007.0       30
           ...
1946.0        1
1959.0        1
1971.0        1
1951.0        1
1954.0        1
Name: yr_renovated, Length: 70, dtype: i
nt64
```

In [178]:

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

executed in 133ms, finished 00:10:42 2021-06-06

Out[178]:

```
0.0    19485
2.0      957
3.0      508
1.0      330
4.0      317
Name: view, dtype: int64
```

In [179]:

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

executed in 112ms, finished 00:10:42 2021-06-06

Out[179]:

```
0.0    21451
1.0      146
Name: waterfront, dtype: int64
```

In [180]:

```
data.info()
```

executed in 132ms, finished 00:10:43 2021-06-06

```
<class 'pandas.core.frame.DataFram
e'>
RangeIndex: 21597 entries, 0 to 21
596
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
```

In [ ]:

```

```

In [181]:

```
data['age'] = 2015 - data['yr_built']
```

executed in 103ms, finished 00:10:43 2021-06-06

In [182]:

```
data['age']
```

executed in 136ms, finished 00:10:43 2021-06-06

Out[182]:

```
0        60
1        64
2        82
3        50
4        28
        ..
21592     6
21593     1
21594     6
21595    11
21596     7
Name: age, Length: 21597, dtype: int64
```

In [ ]:

In [183]:

```
data.head(10)
```

executed in 224ms, finished 00:10:43 2021-06-06

Out[183]:

	id	price	bedrooms	bathrooms
0	7129300520	221900.0	3	1.00
1	6414100192	538000.0	3	2.25
2	5631500400	180000.0	2	1.00
3	2487200875	604000.0	4	3.00
4	1954400510	510000.0	3	2.00
5	7237550310	1230000.0	4	4.50
6	1321400060	257500.0	3	2.25
7	2008000270	291850.0	3	1.50
8	2414600126	229500.0	3	1.00
9	3793500160	323000.0	3	2.50

10 rows × 22 columns

In [184]:

```
data['yrsince_reno'] = data['age']
```

executed in 98ms, finished 00:10:43 2021-06-06

In [ ]:

```

```

In [185]:

```
data['yrsince_reno'] = [(2015 - val)  if (val !=0.0) e
#expression1 if condition else expression2 for item in
data['yrsince_reno'] = data['yrsince_reno'].fillna(dat
```

executed in 134ms, finished 00:10:43 2021-06-06

In [186]:

```
data.head(10)
```

executed in 110ms, finished 00:10:43 2021-06-06

Out[186]:

	id	price	bedrooms	bathrooms
0	7129300520	221900.0	3	1.00
1	6414100192	538000.0	3	2.25
2	5631500400	180000.0	2	1.00
3	2487200875	604000.0	4	3.00
4	1954400510	510000.0	3	2.00
5	7237550310	1230000.0	4	4.50
6	1321400060	257500.0	3	2.25
7	2008000270	291850.0	3	1.50
8	2414600126	229500.0	3	1.00
9	3793500160	323000.0	3	2.50

10 rows × 23 columns

In [187]:

```
data.describe().T
```

executed in 244ms, finished 00:10:44 2021-06-06

Out[187]:

	count	mean	
id	21597.0	4.580474e+09	2.876736e+
price	21597.0	5.402966e+05	3.673681e+
bedrooms	21597.0	3.373200e+00	9.262989e
bathrooms	21597.0	2.115826e+00	7.689843e
sqft_living	21597.0	2.080322e+03	9.181061e+
sqft_lot	21597.0	1.509941e+04	4.141264e+
floors	21597.0	1.494096e+00	5.396828e
waterfront	21597.0	6.760198e-03	8.194394e
view	21597.0	2.331805e-01	7.646726e
condition	21597.0	3.409825e+00	6.505456e
grade	21597.0	7.657915e+00	1.173200e+
sqft_above	21597.0	1.788597e+03	8.277598e+
sqft_basement	21597.0	2.857166e+02	4.398198e+
yr_built	21597.0	1.971000e+03	2.937523e+

	count	mean	s
yr_renovated	21597.0	6.875821e+01	3.640375e+
zipcode	21597.0	9.807795e+04	5.351307e+
lat	21597.0	4.756009e+01	1.385518e
long	21597.0	-1.222140e+02	1.407235e
sqft_living15	21597.0	1.986620e+03	6.852305e+
sqft_lot15	21597.0	1.275828e+04	2.727444e+
month	21597.0	6.573969e+00	3.115061e+
age	21597.0	4.400032e+01	2.937523e+
yrsince_reno	21597.0	4.205487e+01	2.894539e+

In [188]:

```
data.info()
```

executed in 28ms, finished 00:10:44 2021-06-06

```
<class 'pandas.core.frame.DataFram
e'>
RangeIndex: 21597 entries, 0 to 21
596
Data columns (total 23 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
```

In [189]:

```
data.columns
```

executed in 223ms, finished 00:10:44 2021-06-06

Out[189]:

```
Index(['id', 'price', 'bedrooms', 'bathr
ooms', 'sqft_living', 'sqft_lot',
       'floors', 'waterfront', 'view',
'condition', 'grade', 'sqft_above',
       'sqft_basement', 'yr_built', 'yr_
renovated', 'zipcode', 'lat', 'long',
       'sqft_living15', 'sqft_lot15', 'm
onth', 'age', 'yrsince_reno'],
      dtype='object')
```

In [190]:

```
#rearrange Columns
data = data[['id', 'price', 'bedrooms', 'bathrooms', '
       'sqft_lot', 'floors', 'yr_built','age','conditi
        'yr_renovated','yrsince_reno', 'sqft_above', '
       'lat', 'long', 'sqft_living15', 'sqft_lot15' ]]
```

executed in 141ms, finished 00:10:44 2021-06-06

In [191]:

```
data.info()
```

executed in 131ms, finished 00:10:44 2021-06-06

```
<class 'pandas.core.frame.DataFram
e'>
RangeIndex: 21597 entries, 0 to 21
596
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
```

In [192]:

```
data
```

executed in 260ms, finished 00:10:44 2021-06-06

Out[192]:

	id	price	bedrooms	bathrooms	sqft_li
0	7129300520	221900.0	3	1.00	
1	6414100192	538000.0	3	2.25	
2	5631500400	180000.0	2	1.00	
3	2487200875	604000.0	4	3.00	
4	1954400510	510000.0	3	2.00	
...	...	...	...	...	
21592	263000018	360000.0	3	2.50	
21593	6600060120	400000.0	4	2.50	

In [193]:

```
data.to_csv('./data/kc_house_data_tidy.csv',encoding='
```

executed in 761ms, finished 00:10:45 2021-06-06

2.0.0.3 Summary

3 EDA of data

In [194]:

```
#Load Tidy Data
hddf = pd.read_csv('./data/kc_house_data_tidy.csv')
```

executed in 271ms, finished 00:10:46 2021-06-06

In [195]:

```
hddf.drop(columns='Unnamed: 0',inplace=True)
```

executed in 39ms, finished 00:10:46 2021-06-06

In [196]:

```
hddf.drop(columns=['yr_built', 'yr_renovated'],inplace
```

executed in 131ms, finished 00:10:46 2021-06-06

In [197]:

```
import matplotlib.pyplot as plt
from matplotlib import style
style.use('ggplot')
%matplotlib inline
import seaborn as sns
```

executed in 129ms, finished 00:10:46 2021-06-06

In [198]:

```
for col in hddf:
    print(col.upper())
    print(hddf[col].value_counts())
    print('##################')
```

executed in 360ms, finished 00:10:46 2021-06-06

```
ID
795000620     3
1825069031    2
2019200220    2
7129304540    2
1781500435    2
             ..
7812801125    1
4364700875    1
3021059276    1
880000205     1
1777500160    1
Name: id, Length: 21420, dtype: in
t64
##################
PRICE
350000.0    172
450000.0    172
550000.0    159
```

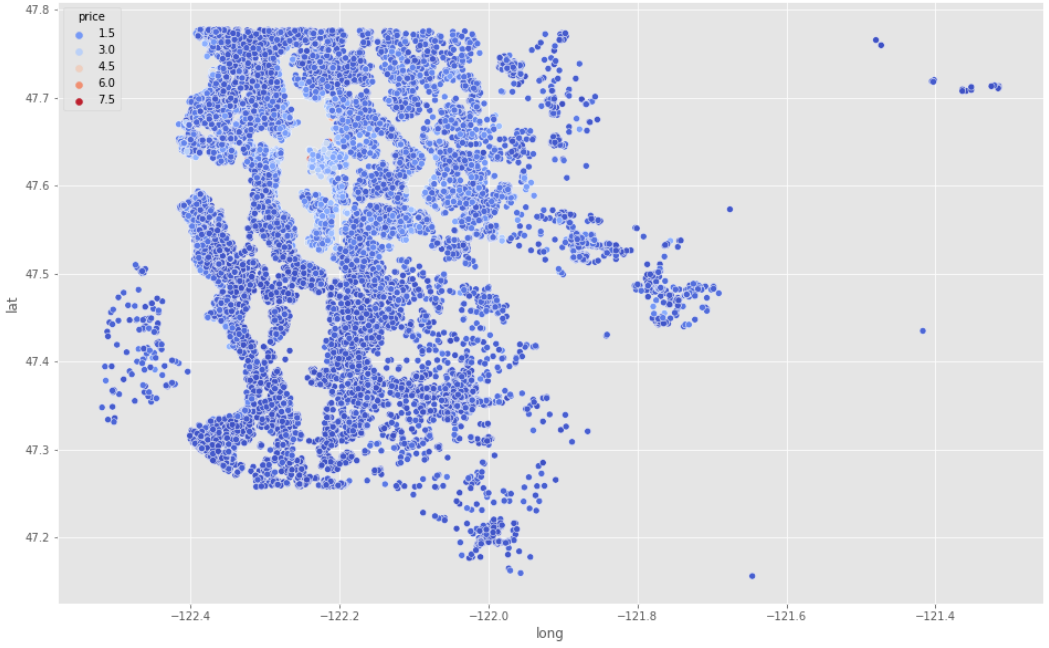
In [199]:

```
plt.figure(figsize=(16,10))
sns.scatterplot(x='long', y='lat', data=hddf, hue='pri
```

executed in 3.04s, finished 00:10:49 2021-06-06

Out[199]:

```
<AxesSubplot:xlabel='long', ylabel='la
t'>
```



In [200]:

```
hddf.columns
```

executed in 30ms, finished 00:10:49 2021-06-06

Out[200]:

```
Index(['id', 'price', 'bedrooms', 'bathr
ooms', 'sqft_living', 'sqft_lot',
       'floors', 'age', 'condition', 'gr
ade', 'waterfront', 'yrsince_reno',
       'sqft_above', 'sqft_basement', 'v
iew', 'zipcode', 'lat', 'long',
       'sqft_living15', 'sqft_lot15'],
      dtype='object')
```

In [201]:

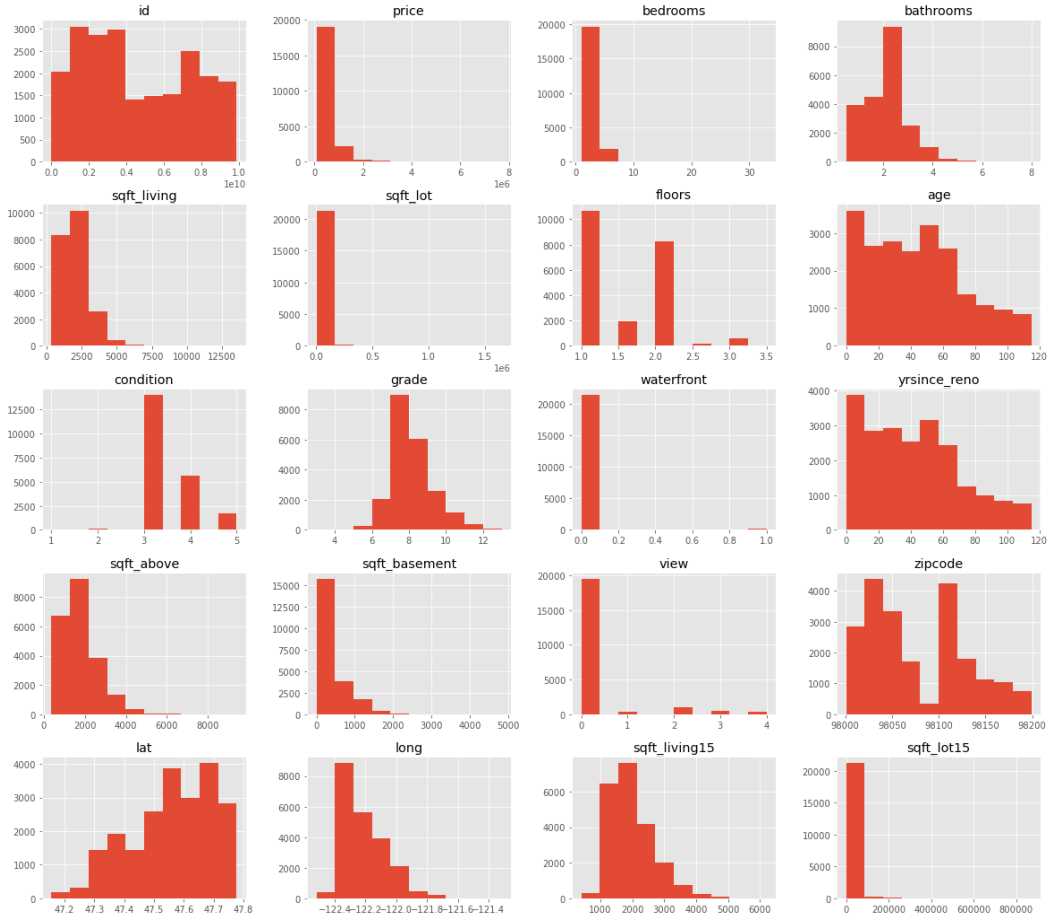
```
#define feature pertinent to model
feats_df = ['bedrooms', 'bathrooms', 'sqft_living', 's
       'floors', 'age', 'condition', 'grade', 'waterfr
       'sqft_above', 'sqft_basement', 'view', 'zipcode
       'sqft_living15', 'sqft_lot15']
```

executed in 114ms, finished 00:10:49 2021-06-06

In [202]:

```
hddf.hist(figsize = (20,18));
```

executed in 11.2s, finished 00:11:01 2021-06-06



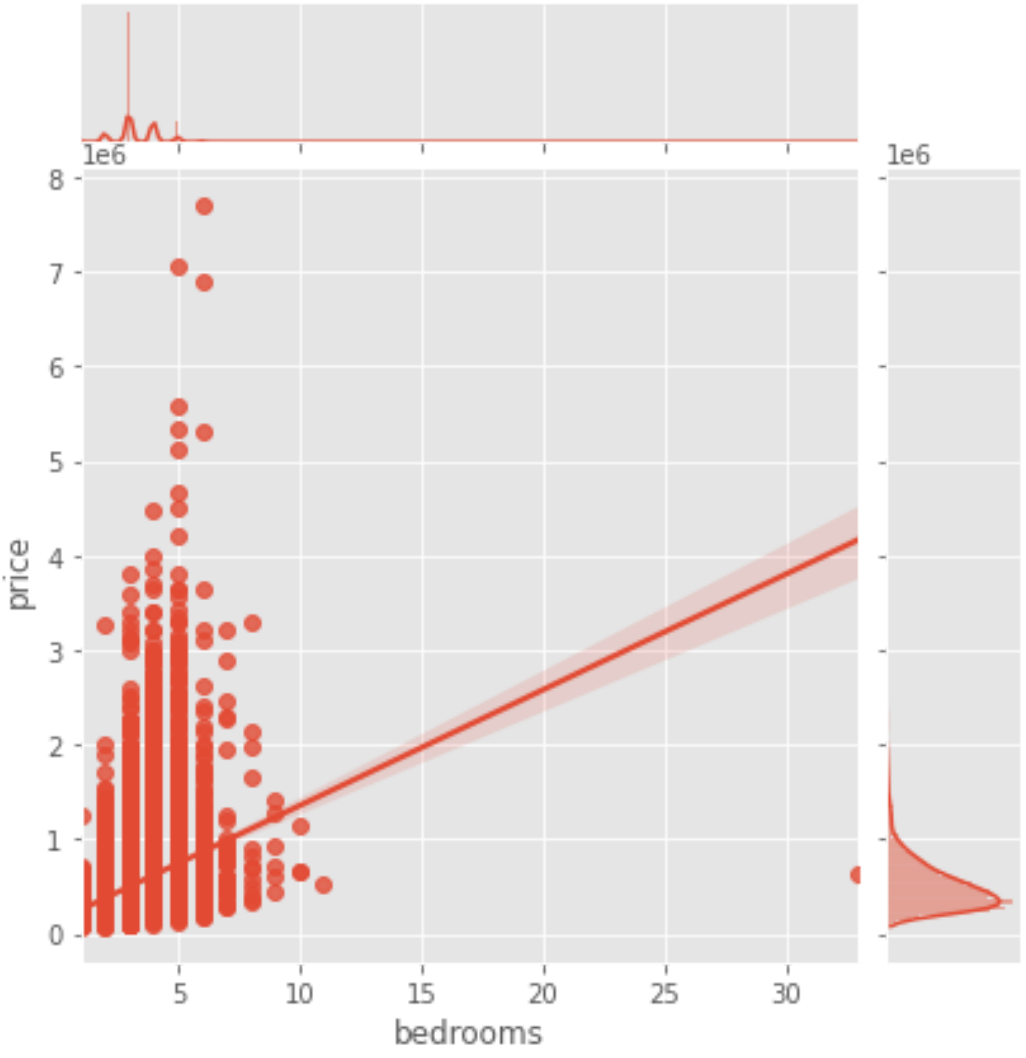
Price looks to be right skewed. We will look for linearity between different pairs of data.

3.1 Check for Linearity

In [203]:

```
sns.jointplot(x='bedrooms',y='price', data=hddf, kind=
```

executed in 6.34s, finished 00:11:07 2021-06-06

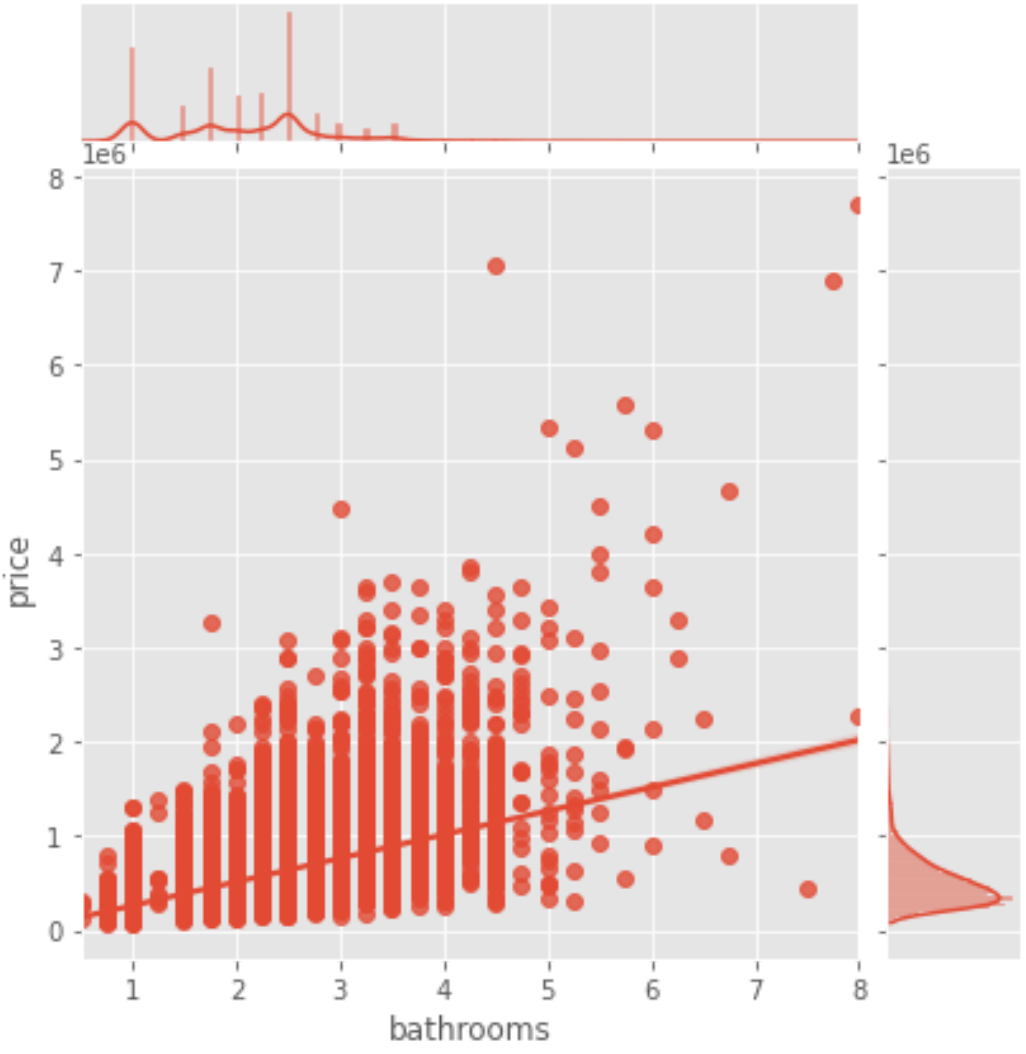


Observation: A Big outlier of 30+ bedrooms. Note: we should remove the outlier.

In [204]:

```
sns.jointplot(x='bathrooms',y='price', data=hddf, kind
```

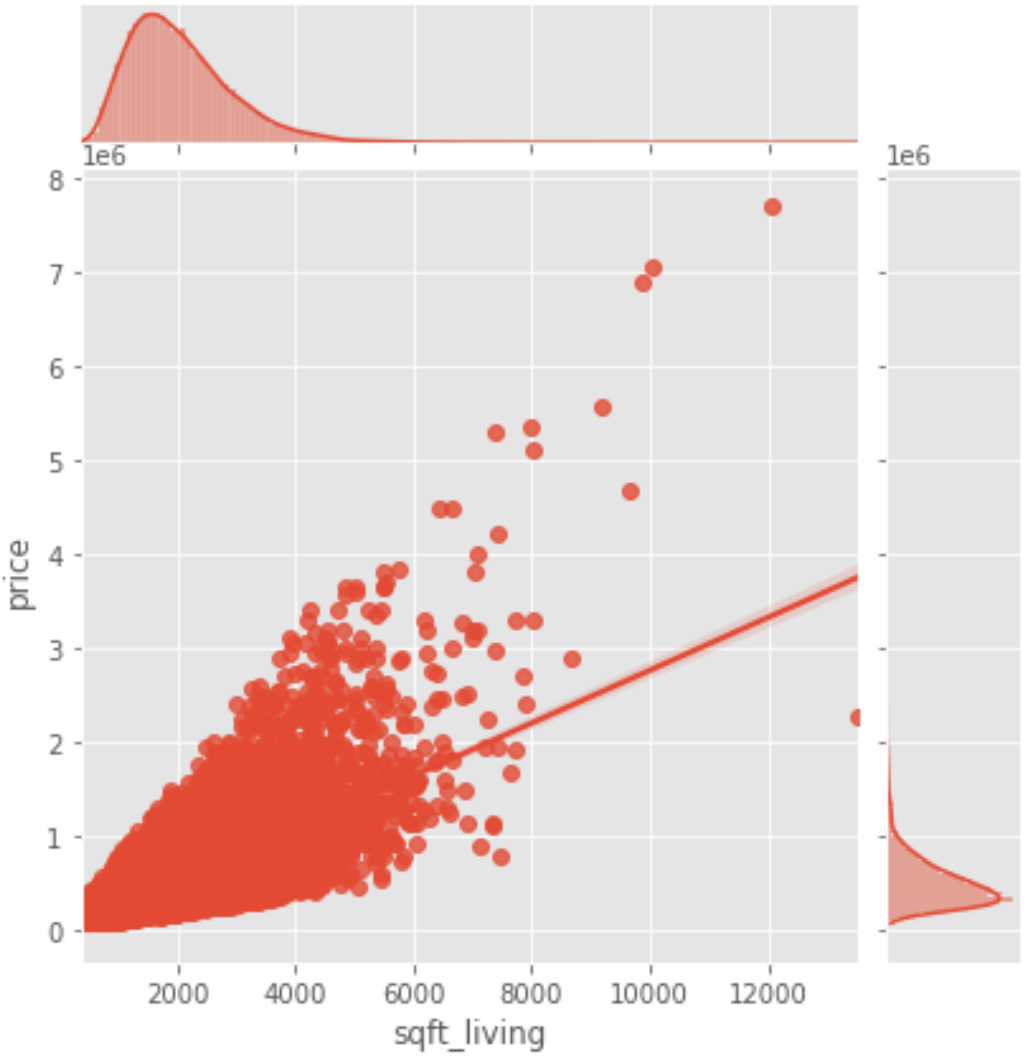
executed in 5.84s, finished 00:11:13 2021-06-06



In [205]:

```
sns.jointplot(x='sqft_living',y='price', data=hddf, ki
```

executed in 5.97s, finished 00:11:19 2021-06-06

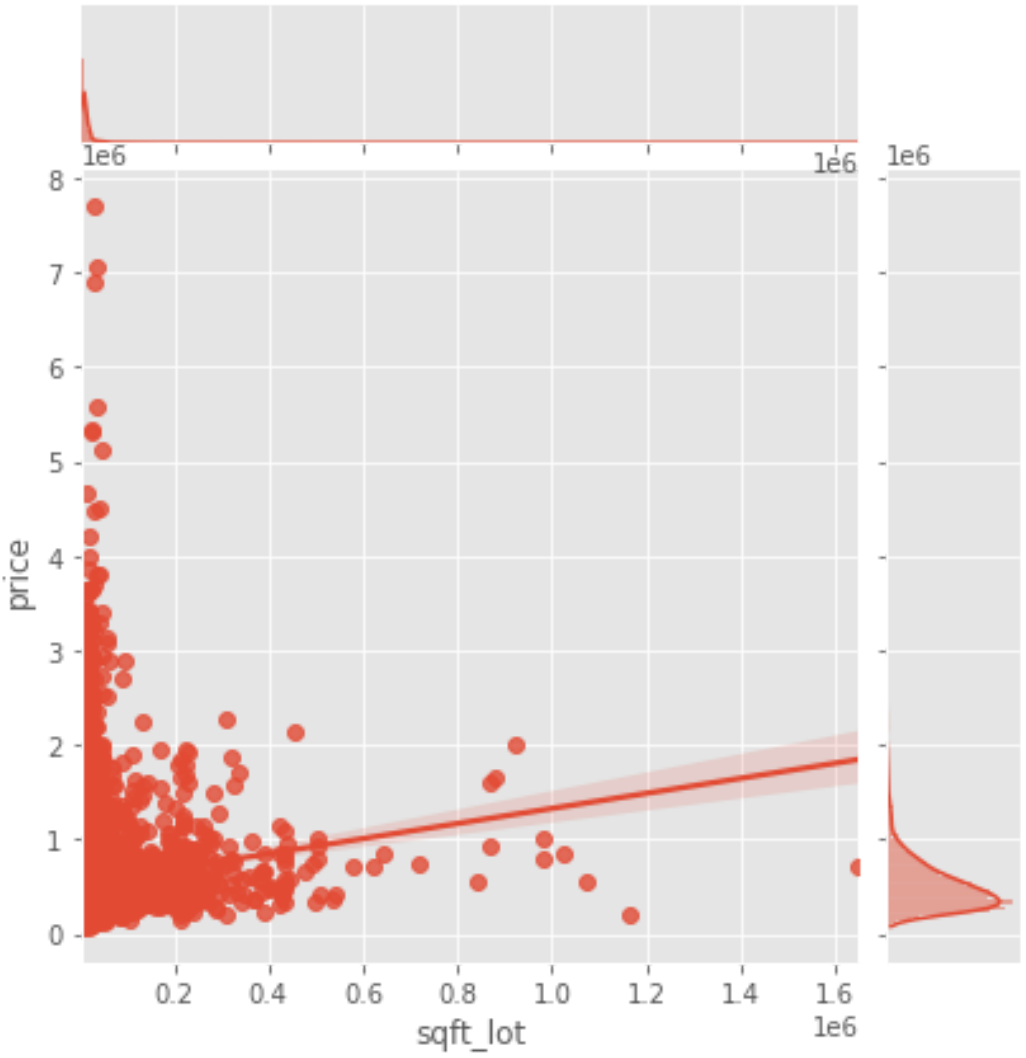


Is linear? not truly but somewhat

In [206]:

```
sns.jointplot(x='sqft_lot',y='price', data=hddf, kind=
```

executed in 17.4s, finished 00:11:36 2021-06-06

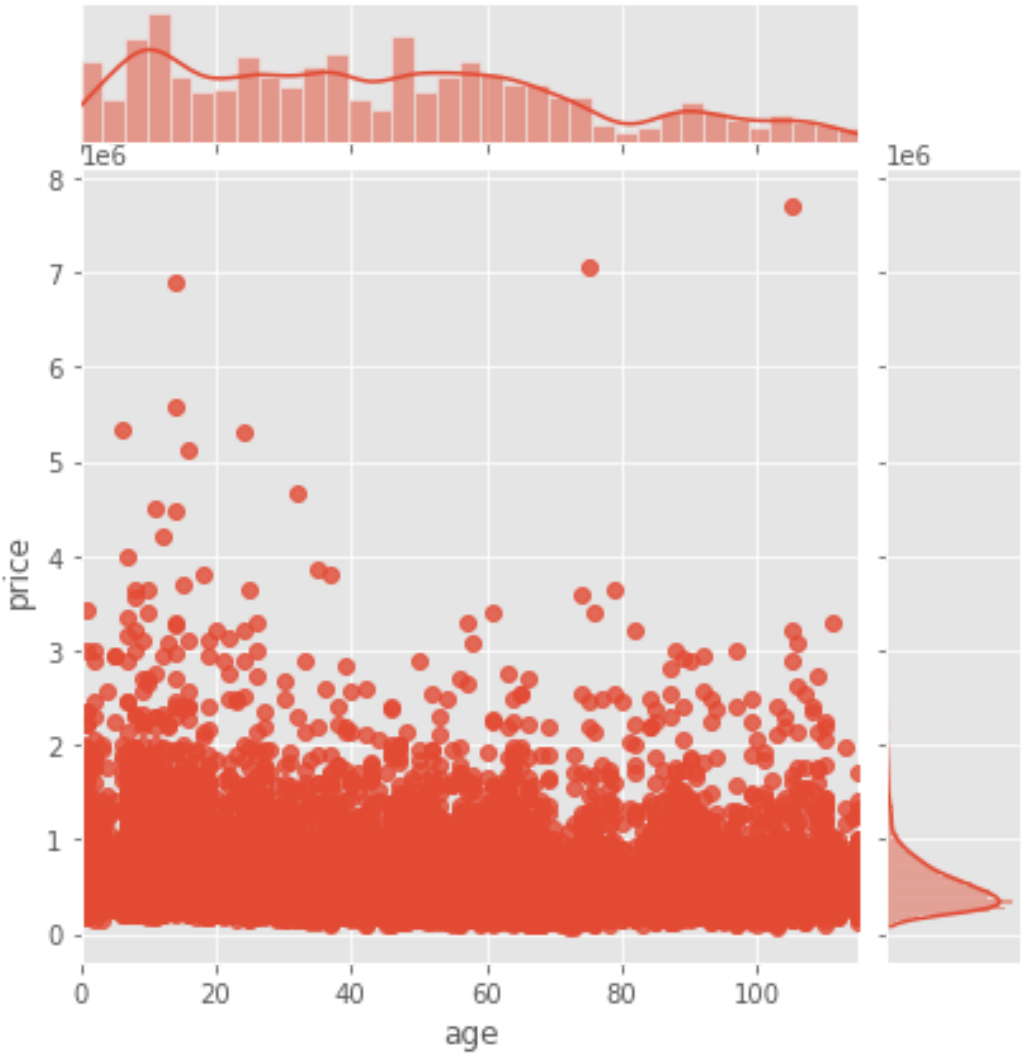


not linear

In [207]:

```
sns.jointplot(x='age',y='price', data=hddf, kind='reg'
```

executed in 4.38s, finished 00:11:41 2021-06-06

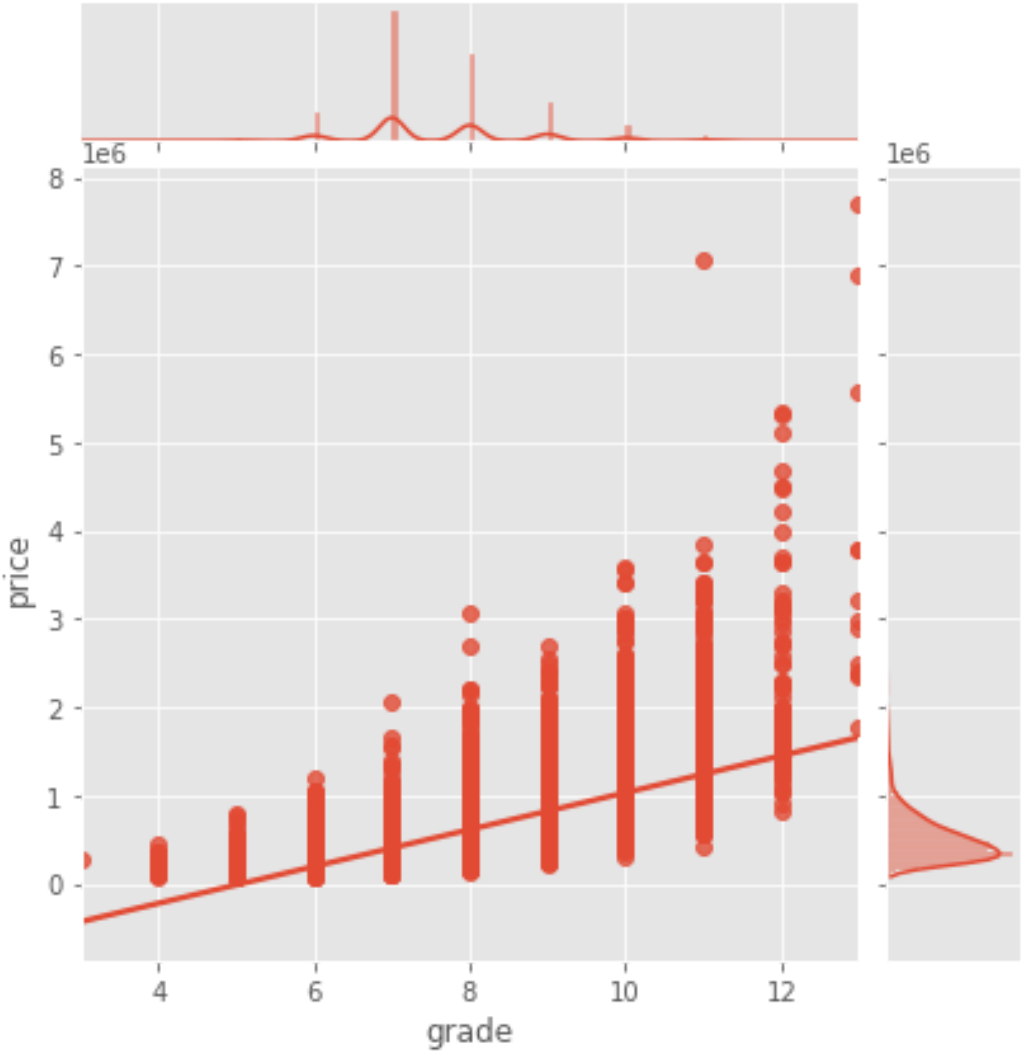


not linear

In [208]:

```
sns.jointplot(x='grade',y='price', data=hddf, kind='re
```

executed in 5.72s, finished 00:11:46 2021-06-06

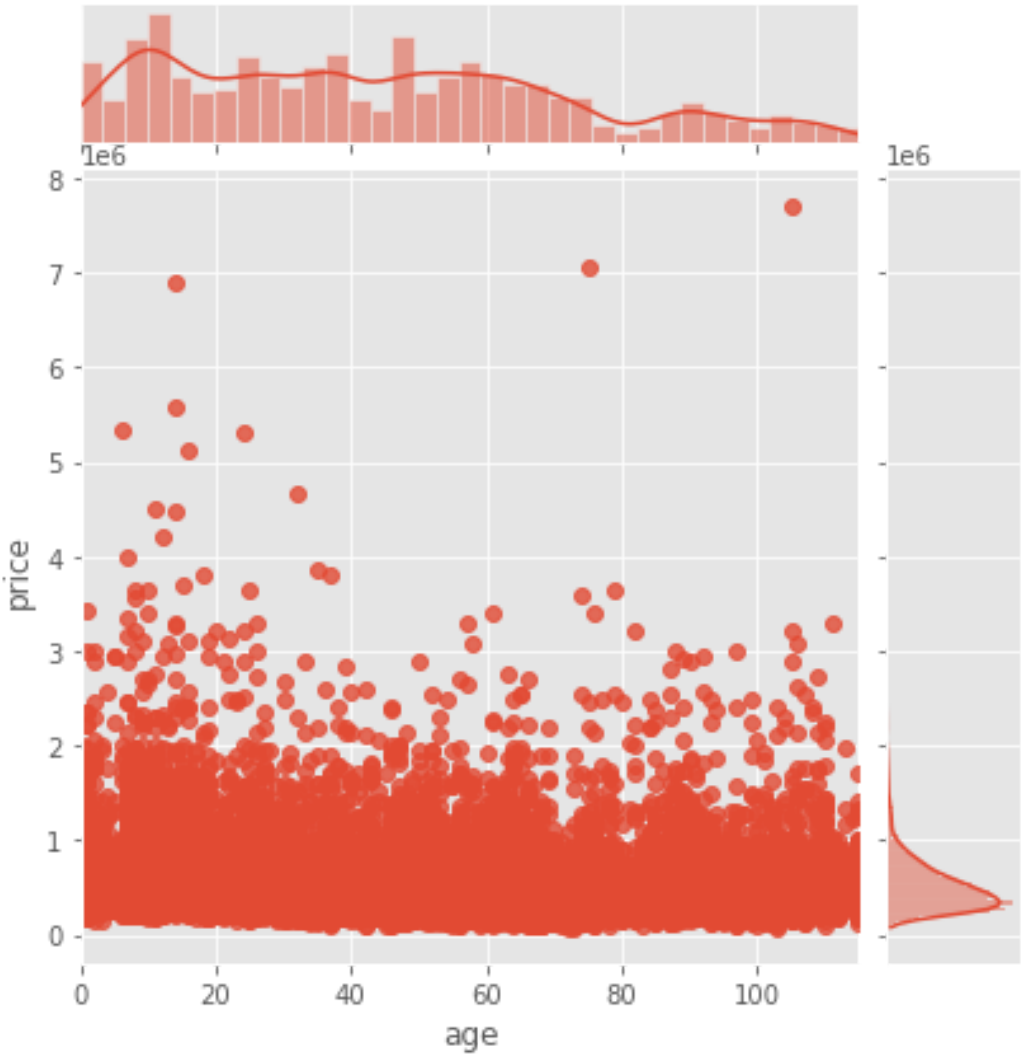


In [ ]:

In [209]:

```
sns.jointplot(x='age',y='price', data=hddf, kind='reg'
```

executed in 4.15s, finished 00:11:51 2021-06-06



In [ ]:

3.2 Check for multicolinearity in features

In [210]:

```
corr = hddf[feats_df].corr()
corr
```

executed in 121ms, finished 00:11:51 2021-06-06

Out[210]:

	bedrooms	bathrooms	sqft_living
bedrooms	1.000000	0.514508	0.578212
bathrooms	0.514508	1.000000	0.755758
sqft_living	0.578212	0.755758	1.000000
sqft_lot	0.032471	0.088373	0.173453
floors	0.177944	0.502582	0.353953
age	-0.155670	-0.507173	-0.318152
condition	0.026496	-0.126479	-0.059445
grade	0.356563	0.665838	0.762779
waterfront	-0.002127	0.063629	0.104637
yrsince_reno	-0.164392	-0.532389	-0.339146
sqft_above	0.479386	0.686668	0.876448
sqft_basement	0.297229	0.278485	0.428660
view	0.078354	0.186016	0.281715
zipcode	-0.154092	-0.204786	-0.199802

	bedrooms	bathrooms	sqft_living
lat	-0.009951	0.024280	0.052155
long	0.132054	0.224903	0.241214
sqft_living15	0.393406	0.569884	0.756402
sqft_lot15	0.030690	0.088303	0.184342

In [211]:

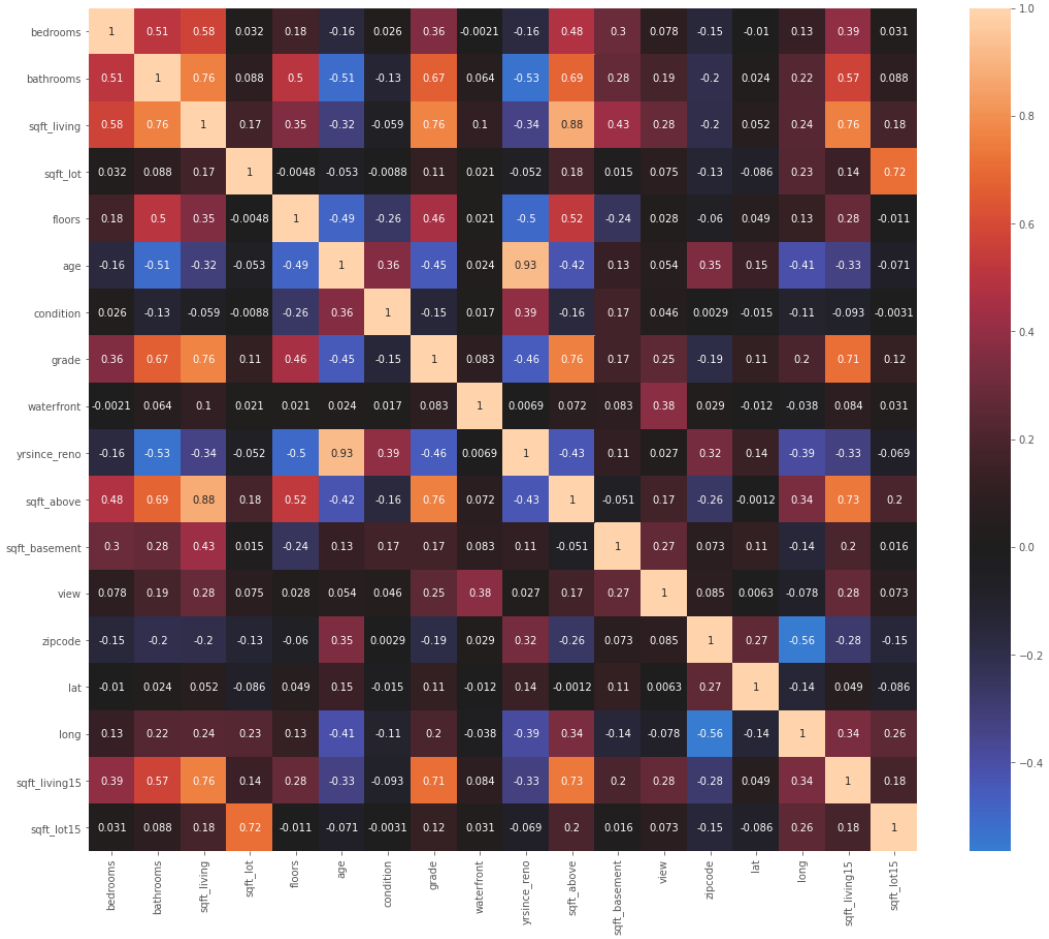
```
#looking for pairwise correlations over .65 is a sign
```

executed in 127ms, finished 00:11:51 2021-06-06

In [212]:

```
plt.figure(figsize=(18,15))
sns.heatmap(corr, center=0, annot=True);
```

executed in 8.90s, finished 00:12:00 2021-06-06



4 Model Building

4.1 Baseline model

In [ ]:

In [213]:

```
hddf.info()
```

executed in 29ms, finished 00:12:00 2021-06-06

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 20 columns):
 #   Column         Non-Null Count  Dtyp
e
---  ------         --------------  ----
-
 0   id             21597 non-null  int6
4
 1   price          21597 non-null  floa
t64
 2   bedrooms       21597 non-null  int6
4
 3   bathrooms      21597 non-null  floa
t64
 4   sqft_living    21597 non-null  int6
4
 5   sqft_lot       21597 non-null  int6
4
 6   floors         21597 non-null  floa
t64
 7   age            21597 non-null  int6
4
 8   condition      21597 non-null  int6
4
 9   grade          21597 non-null  int6
```

```
4
 10  waterfront     21597 non-null  floa
t64
 11  yrsince_reno   21597 non-null  floa
t64
 12  sqft_above     21597 non-null  int6
4
 13  sqft_basement  21597 non-null  floa
t64
 14  view           21597 non-null  floa
t64
 15  zipcode        21597 non-null  int6
4
 16  lat            21597 non-null  floa
t64
 17  long           21597 non-null  floa
t64
 18  sqft_living15  21597 non-null  int6
4
 19  sqft_lot15     21597 non-null  int6
4
dtypes: float64(9), int64(11)
memory usage: 3.3 MB
```

In [214]:

```
from statsmodels.formula.api import ols
```

executed in 99ms, finished 00:12:00 2021-06-06

In [215]:

```
outcome = 'price'
predictors_df = hddf.drop(['id', 'price', ], axis = 1)
predictor_variables = "+".join(predictors_df.columns)
formula = outcome + '~' + predictor_variables
```

executed in 153ms, finished 00:12:00 2021-06-06

In [216]:

```
predictor_variables
```

executed in 168ms, finished 00:12:00 2021-06-06

Out[216]:

```
'bedrooms+bathrooms+sqft_living+sqft_lot
+floors+age+condition+grade+waterfront+y
rsince_reno+sqft_above+sqft_basement+vie
w+zipcode+lat+long+sqft_living15+sqft_lo
t15'
```

In [217]:

```
formula
```

executed in 280ms, finished 00:12:01 2021-06-06

Out[217]:

```
'price~bedrooms+bathrooms+sqft_living+sq
ft_lot+floors+age+condition+grade+waterf
ront+yrsince_reno+sqft_above+sqft_baseme
nt+view+zipcode+lat+long+sqft_living15+s
qft_lot15'
```

In [218]:

```
baseline_model = ols(formula = formula, data = hddf).f
baseline_model.summary()
```

executed in 609ms, finished 00:12:01 2021-06-06

Out[218]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	
Model:	OLS	**Adj. R-squared:**	
Method:	Least Squares	**F-statistic:**	
Date:	Sun, 06 Jun 2021	**Prob (F-statistic):**	
Time:	00:12:01	**Log-Likelihood:**	-2.943
No. Observations:	21597	**AIC:**	5.88
Df Residuals:	21578	**BIC:**	5.89
Df Model:	18		
Covariance Type:	nonrobust		

	coef	**std err**

Intercept	1.329e+06	2.88e+06	0.46
bedrooms	-3.592e+04	1901.055	-18.89
bathrooms	4.167e+04	3269.532	12.74
sqft_living	102.6395	18.081	5.67
sqft_lot	0.1287	0.048	2.68
floors	6726.8993	3605.553	1.86
age	3312.9565	127.530	25.97
condition	2.678e+04	2353.707	11.38
grade	9.676e+04	2162.791	44.73
waterfront	6.21e+05	1.81e+04	34.25
yrsince_reno	-676.0185	131.088	-5.15
sqft_above	78.4711	18.072	4.34
sqft_basement	47.7765	17.911	2.66
view	5.345e+04	2122.012	25.18
zipcode	-584.6090	32.998	-17.71
lat	6.008e+05	1.07e+04	55.92
long	-2.179e+05	1.32e+04	-16.55
sqft_living15	21.5861	3.453	6.25
sqft_lot15	-0.3869	0.073	-5.27

Omnibus:	18331.569	**Durbin-Watson:**	

Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1823
Skew:	3.561	**Prob(JB):**	
Kurtosis:	47.452	**Cond. No.**	2

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.12e+08. This might indicate that there are
strong multicollinearity or other numerical problems

Conclusions from baseline model:

1. Drop **id** as its not a relevant feature (even though it has a good p-value doesn't really tell us anything)
2. **Bedrooms** has a negative coefficient -> As bedrooms increase, price decreases.
3. Price per sqft is about $102
4. **Yr_Built** has a negative coefficient

5. **Zipcode** has a negative coefficent. need to look into how to get location data in the model
6. **sqft_lot15** has a neg coef
7. **yrsince_reno**
8. **floors** .062 not relevant

In [219]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

executed in 7ms, finished 00:12:01 2021-06-06

In [220]:

```
X = predictors_df
y = hddf['price']
X_train, X_test, y_train, y_test = train_test_split(X,
```

executed in 156ms, finished 00:12:01 2021-06-06

In [221]:

```
baseline_linreg = LinearRegression()
baseline_linreg.fit(X_train, y_train)
```

executed in 436ms, finished 00:12:02 2021-06-06

Out[221]:

```
LinearRegression()
```

In [222]:

```
y_pred = baseline_linreg.predict(X_test)
```

executed in 15ms, finished 00:12:02 2021-06-06

In [223]:

```
residuals = y_pred - y_test
```

executed in 128ms, finished 00:12:02 2021-06-06

In [224]:

```
X.columns
```

executed in 174ms, finished 00:12:02 2021-06-06

Out[224]:

```
Index(['bedrooms', 'bathrooms', 'sqft_li
ving', 'sqft_lot', 'floors', 'age',
       'condition', 'grade', 'waterfron
t', 'yrsince_reno', 'sqft_above',
       'sqft_basement', 'view', 'zipcod
e', 'lat', 'long', 'sqft_living15',
       'sqft_lot15'],
      dtype='object')
```

In [225]:

```
baseline_linreg.coef_
```

executed in 105ms, finished 00:12:02 2021-06-06

Out[225]:

```
array([-3.89688786e+04,  4.15896223e+04,
1.24070917e+02,  1.69028472e-01,
        1.08304695e+04,  3.23425813e+03,
2.51784144e+04,  9.82927953e+04,
        6.01867568e+05, -5.70846507e+02,
5.19520944e+01,  2.98032950e+01,
        5.42902143e+04, -5.93251557e+02,
5.98034512e+05, -2.08357883e+05,
        2.43185044e+01, -4.40182678e-0
1])
```

In [226]:

```
from sklearn.metrics import mean_squared_error
```

executed in 169ms, finished 00:12:02 2021-06-06

In [227]:

```
mse_train = mean_squared_error(y_train, baseline_linre
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 181ms, finished 00:12:03 2021-06-06

```
Train RMSE: 199867.60910058717
Test RMSE: 205457.75418752249
```

In [228]:

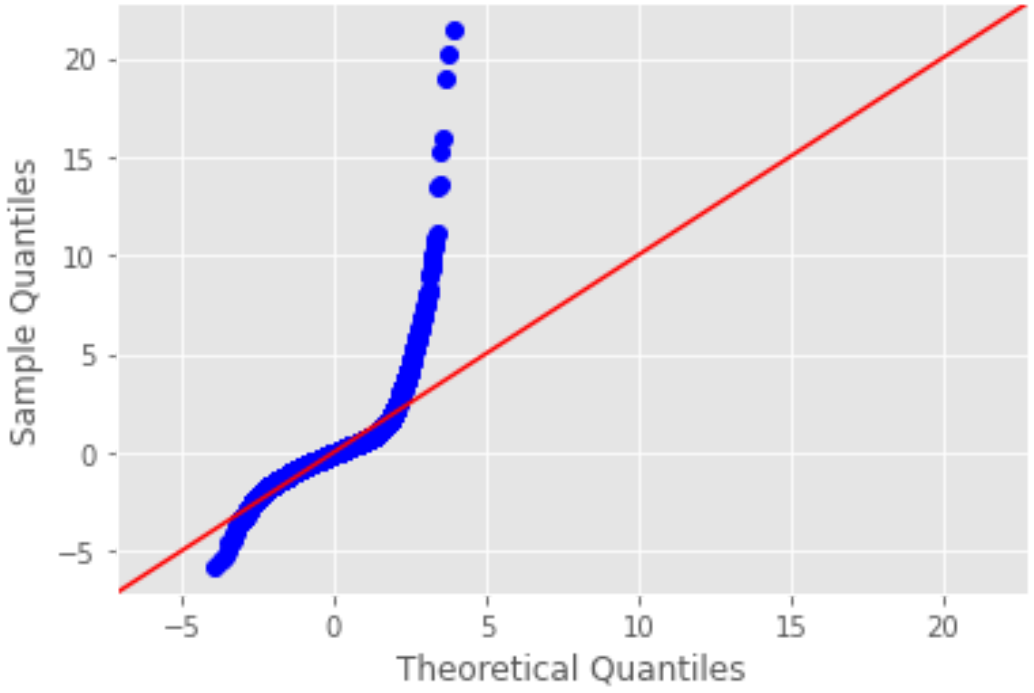
```
import statsmodels.api as sm
import scipy.stats as stats
```

executed in 135ms, finished 00:12:03 2021-06-06

In [229]:

```
fig = sm.graphics.qqplot(baseline_model.resid, dist=st
```

executed in 493ms, finished 00:12:03 2021-06-06



4.2 Model 2 - Dealing with outliers

In [230]:

```
hddf.describe().transpose()
```

executed in 246ms, finished 00:12:04 2021-06-06

Out[230]:

	count	mean	s
id	21597.0	4.580474e+09	2.876736e+
price	21597.0	5.402966e+05	3.673681e+
bedrooms	21597.0	3.373200e+00	9.262989e-
bathrooms	21597.0	2.115826e+00	7.689843e-
sqft_living	21597.0	2.080322e+03	9.181061e+
sqft_lot	21597.0	1.509941e+04	4.141264e+
floors	21597.0	1.494096e+00	5.396828e-
age	21597.0	4.400032e+01	2.937523e+
condition	21597.0	3.409825e+00	6.505456e-
grade	21597.0	7.657915e+00	1.173200e+
waterfront	21597.0	6.760198e-03	8.194394e-
yrsince_reno	21597.0	4.205487e+01	2.894539e+
sqft_above	21597.0	1.788597e+03	8.277598e+
sqft_basement	21597.0	2.857166e+02	4.398198e+

	count	mean	
view	21597.0	2.331805e-01	7.646726e
zipcode	21597.0	9.807795e+04	5.351307e+
lat	21597.0	4.756009e+01	1.385518e
long	21597.0	-1.222140e+02	1.407235e
sqft_living15	21597.0	1.986620e+03	6.852305e+
sqft_lot15	21597.0	1.275828e+04	2.727444e+

In [ ]:

4.2.0.1 hddf_no_out

In [ ]:

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```

executed in 30ms, finished 00:10:22 2021-06-06

In [232]:

```
#removed 75 rows based off bathroom outliers
len(hddf[hddf.bathrooms >= 1])
```

executed in 28ms, finished 00:12:58 2021-06-06

Out[232]:

```
21522
```

In [253]:

```
hddf_no_out = hddf[hddf.bathrooms >= 1]
```

executed in 21ms, finished 00:15:27 2021-06-06

In [254]:

```
#Clean outliers where z score is > 3
#not cleaning zscore outliers where negative as smalle
hddf_no_out.columns
```

executed in 13ms, finished 00:15:30 2021-06-06

Out[254]:

```
Index(['id', 'price', 'bedrooms', 'bathr
ooms', 'sqft_living', 'sqft_lot',
       'floors', 'age', 'condition', 'gr
ade', 'waterfront', 'yrsince_reno',
       'sqft_above', 'sqft_basement', 'v
iew', 'zipcode', 'lat', 'long',
       'sqft_living15', 'sqft_lot15'],
      dtype='object')
```

In [255]:

```
z = stats.zscore(hddf_no_out)
print(z)
```

executed in 54ms, finished 00:15:33 2021-06-06

```
[[ 0.88519717 -0.86846203 -0.4100402
... -0.30658272 -0.94639815
  -0.26006107]
 [ 0.63667269 -0.00860214 -0.4100402
... -0.74828667 -0.43556223
  -0.18715978]
 [ 0.36472761 -0.98243903 -1.49404526
... -0.13560055  1.0677549
  -0.17165589]
 ...
 [-1.50076899 -0.49280037 -0.4100402
... -0.94064161 -0.66908723
  -0.41183796]
 [ 0.70129178 -0.38399178  0.67396486
... -1.05462973 -0.23122787
  -0.20325011]
 [-1.49093154 -0.38399178 -0.4100402
...  1.03277764 -0.84423097
  -0.41997476]]
```

In [256]:

```
threshold = 3.0
print(np.where(z > threshold))
```

executed in 15ms, finished 00:15:43 2021-06-06

```
(array([    5,     5,     5, ..., 21501,
21501, 21511], dtype=int64), array([ 3,
4, 18, ..., 13, 18, 12], dtype=int64))
```

In [257]:

```
hddf_no_out = hddf_no_out[(z < 3).all(axis=1)]
```

executed in 23ms, finished 00:15:50 2021-06-06

In [258]:

```
len(hddf_no_out)
```

executed in 9ms, finished 00:15:54 2021-06-06

Out[258]:

19414

In [ ]:

```
1000/22000
```

executed in 2m 27s, finished 00:04:34 2021-06-06

Removed .4.5% of values with z score > 3

In [259]:

```
hddf_no_out.describe().transpose()
```

executed in 194ms, finished 00:16:01 2021-06-06

Out[259]:

	count	mean	
id	19414.0	4.639691e+09	2.8(
price	19414.0	4.855176e+05	2.3!
bedrooms	19414.0	3.326362e+00	8.5
bathrooms	19414.0	2.043989e+00	6.8
sqft_living	19414.0	1.954245e+03	7.3!
sqft_lot	19414.0	9.838963e+03	1.1(
floors	19414.0	1.481482e+00	5.3
age	19414.0	4.443000e+01	2.9(
condition	19414.0	3.411456e+00	6.4
grade	19414.0	7.533378e+00	1.0!
waterfront	19414.0	0.000000e+00	0.0(
yrsince_reno	19414.0	4.270058e+01	2.9'
sqft_above	19414.0	1.695321e+03	7.0!
sqft_basement	19414.0	2.534919e+02	3.8(

	count	mean	
view	19414.0	9.462244e-02	4.0
zipcode	19414.0	9.807903e+04	5.3
lat	19414.0	4.756110e+01	1.3
long	19414.0	-1.222224e+02	1.2
sqft_living15	19414.0	1.906594e+03	5.9
sqft_lot15	19414.0	9.070747e+03	8.9

In [260]:

```
hddf_no_out
```

executed in 79ms, finished 00:16:10 2021-06-06

Out[260]:

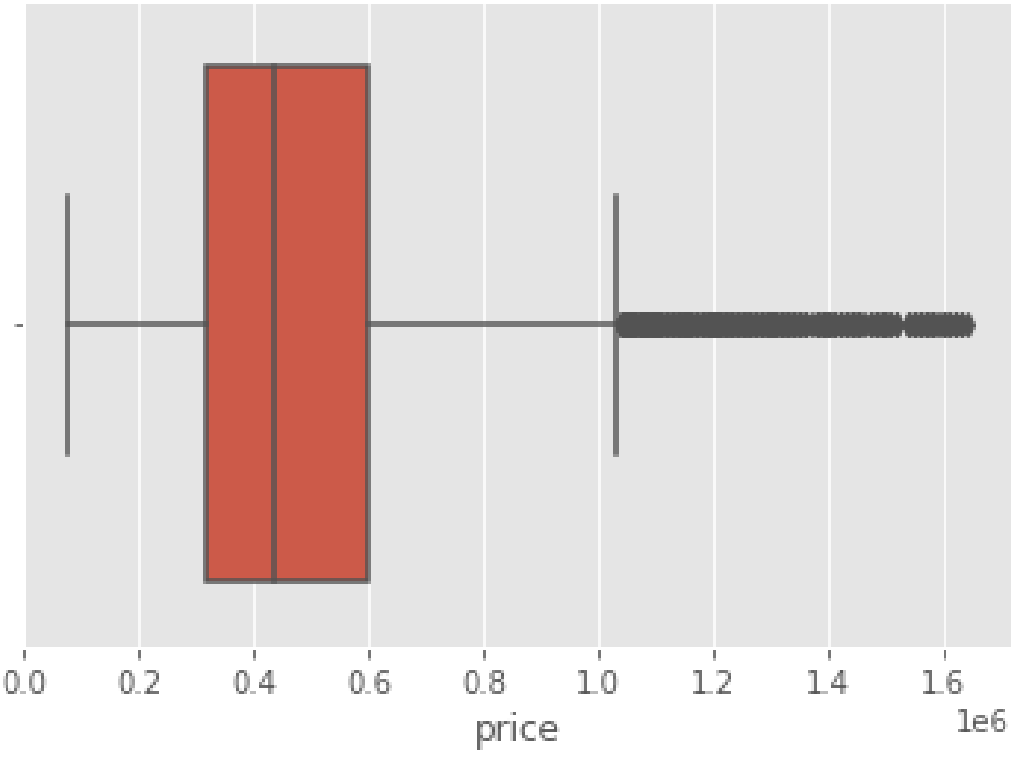
	id	price	bedrooms	bathroc
0	7129300520	221900.0	3	1
1	6414100192	538000.0	3	2
2	5631500400	180000.0	2	1
3	2487200875	604000.0	4	3
4	1954400510	510000.0	3	2
...	...	...	...	
21590	7936000429	1010000.0	4	3
21591	2997800021	475000.0	3	2
21592	263000018	360000.0	3	2
21593	6600060120	400000.0	4	2
21595	291310100	400000.0	3	2

19414 rows × 20 columns

In [261]:

```
sns.boxplot(x=hddf_no_out.price);
```

executed in 346ms, finished 00:16:19 2021-06-06



4.3 Model 2

In [262]:

```
hddf_no_out.shape
```

executed in 8ms, finished 00:16:31 2021-06-06

Out[262]:

```
(19414, 20)
```

In [263]:

```
outcome = 'price'
predictors = hddf_no_out.drop(columns = ['price', 'id'
predictor_variables = "+".join(predictors.columns)
formula = outcome + '~' + predictor_variables
```

executed in 16ms, finished 00:16:35 2021-06-06

In [264]:

```
model_2 = ols(formula = formula, data = hddf_no_out).f
model_2.summary()
```

executed in 238ms, finished 00:16:39 2021-06-06

Out[264]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.679
Model:	OLS	**Adj. R-squared:**	0.679
Method:	Least Squares	**F-statistic:**	2418.
Date:	Sun, 06 Jun 2021	**Prob (F-statistic):**	0.00
Time:	00:16:39	**Log-Likelihood:**	-2.5665e+05
No. Observations:	19414	**AIC:**	5.133e+05
Df Residuals:	19396	**BIC:**	5.135e+05
Df Model:	17		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>**

Intercept	-5.481e+06	2.04e+06	-2.686	0.0(
bedrooms	-1.502e+04	1471.474	-10.210	0.0(
bathrooms	2.933e+04	2435.132	12.044	0.0(
sqft_living	80.4875	13.879	5.799	0.0(
sqft_lot	0.2857	0.145	1.972	0.04
floors	2.849e+04	2619.453	10.875	0.0(
age	2767.6561	93.271	29.673	0.0(
condition	2.762e+04	1650.088	16.738	0.0(
grade	8.601e+04	1575.778	54.581	0.0(
waterfront	1.274e-06	4.74e-07	2.686	0.0(
yrsince_reno	-592.2153	95.526	-6.200	0.0(
sqft_above	17.7745	13.852	1.283	0.19
sqft_basement	13.1119	13.748	0.954	0.34
view	4.03e+04	2445.244	16.480	0.0(
zipcode	-358.4750	23.348	-15.354	0.0(
lat	5.692e+05	7468.260	76.221	0.0(
long	-1.054e+05	1.05e+04	-10.071	0.0(
sqft_living15	46.0971	2.721	16.943	0.0(
sqft_lot15	-1.3384	0.188	-7.116	0.0(

Omnibus:	5219.322	**Durbin-Watson:**	1.966

Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	23220.927
Skew:	1.251	**Prob(JB):**	0.00
Kurtosis:	7.738	**Cond. No.**	8.21e+20

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 2.83e-28. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

Conclusions from model 2:

1. Sqft_Basement no longer relevant feature. Will need to drop it in the next model
2. sqft_lot no longer relevant feature. Will need to drop it in the next model

In [265]:

```
X = hddf_no_out.drop("price", axis = 1)
y = hddf_no_out['price']

X_train, X_test, y_train, y_test = train_test_split(X,
```

executed in 41ms, finished 00:17:06 2021-06-06

In [266]:

```
print(len(X_train))

print(len(X_test))
```

executed in 6ms, finished 00:17:08 2021-06-06

```
14560
4854
```

In [267]:

```
model2_linreg = LinearRegression()
model2_linreg.fit(X_train, y_train)
y_pred = model2_linreg.predict(X_test)
residuals = y_pred - y_test
```

executed in 69ms, finished 00:17:12 2021-06-06

In [268]:

```
model2_linreg.coef_
```

executed in 12ms, finished 00:17:15 2021-06-06

Out[268]:

```
array([-6.89271342e-07, -1.74498048e+04,
3.02656086e+04,  7.93955619e+01,
        1.76508732e-01,  2.91479422e+04,
2.89442033e+03,  2.75479680e+04,
        8.41020068e+04, -1.07320375e-10,
-7.04903888e+02,  2.06022101e+01,
        1.70311848e+01,  4.02302697e+04,
-3.57213641e+02,  5.62383277e+05,
       -9.71159609e+04,  4.63882676e+01,
-1.25659855e+00])
```

In [269]:

```
mse_train = mean_squared_error(y_train, model2_linreg.
mse_test = mean_squared_error(y_test, y_pred)

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

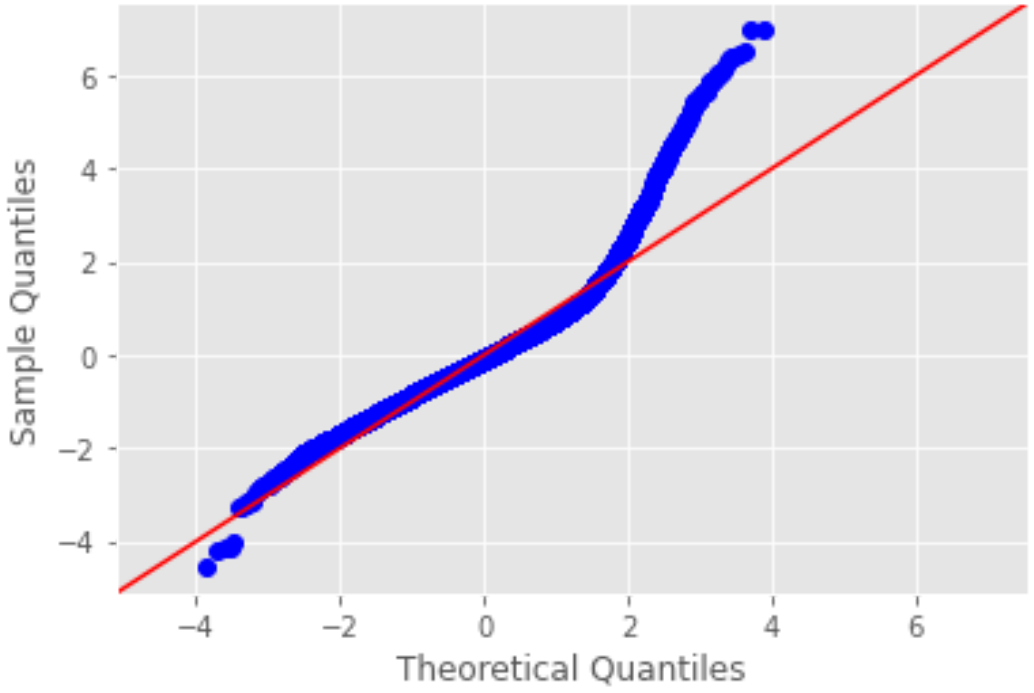
executed in 22ms, finished 00:17:19 2021-06-06

```
Train RMSE: 133527.5873672322
Test RMSE: 132892.7754935861
```

In [270]:

```
fig = sm.graphics.qqplot(model_2.resid, dist=stats.nor
```

executed in 452ms, finished 00:17:24 2021-06-06



4.4 Model 3 Fix Cat Features

In [271]:

```
hddf_no_out.condition.unique()
```

executed in 8ms, finished 00:17:36 2021-06-06

Out[271]:

```
array([3, 5, 4, 1, 2], dtype=int64)
```

In [272]:

```
hddf_no_out.grade.unique()
```

executed in 13ms, finished 00:17:39 2021-06-06

Out[272]:

```
array([ 7,  6,  8,  5,  9, 10, 11,  4],
dtype=int64)
```

In [273]:

```
hddf_no_out.view.unique()
```

executed in 12ms, finished 00:17:42 2021-06-06

Out[273]:

```
array([0., 2., 1.])
```

In [274]:

```
condition_dummies = pd.get_dummies(hddf_no_out['condit
```

executed in 10ms, finished 00:17:45 2021-06-06

In [275]:

```
grade_dummies = pd.get_dummies(hddf_no_out['grade'], p
```

executed in 9ms, finished 00:17:48 2021-06-06

In [276]:

```
view_dummies = pd.get_dummies(hddf_no_out['view'], pre
```

executed in 10ms, finished 00:17:51 2021-06-06

In [277]:

```
grade_dummies.shape
```

executed in 7ms, finished 00:17:54 2021-06-06

Out[277]:

```
(19414, 7)
```

In [278]:

```
view_dummies.shape
```

executed in 8ms, finished 00:17:57 2021-06-06

Out[278]:

```
(19414, 2)
```

4.4.0.1 hddf_no_out_dum

In [279]:

```
hddf_no_out_dum = pd.concat([hddf_no_out,condition_dum
```

executed in 23ms, finished 00:18:06 2021-06-06

In [280]:

```
hddf_no_out_dum.rename(columns={"view_1.0": "view_1",
```

executed in 7ms, finished 00:18:09 2021-06-06

In [281]:

```
hddf_no_out_dum.head()
```

executed in 51ms, finished 00:18:11 2021-06-06

Out[281]:

	id	price	bedrooms	bathrooms
0	7129300520	221900.0	3	1.00
1	6414100192	538000.0	3	2.25
2	5631500400	180000.0	2	1.00
3	2487200875	604000.0	4	3.00
4	1954400510	510000.0	3	2.00

5 rows × 33 columns

In [282]:

```
hddf_no_out_dum.info()
```

executed in 53ms, finished 00:18:15 2021-06-06

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19414 entries, 0 to 21595
Data columns (total 33 columns):
 #   Column          Non-Null Count  Dtyp
e
---  ------          --------------  ----
-
 0   id              19414 non-null  int6
4
 1   price           19414 non-null  floa
t64
 2   bedrooms        19414 non-null  int6
4
 3   bathrooms       19414 non-null  floa
t64
 4   sqft_living     19414 non-null  int6
4
 5   sqft_lot        19414 non-null  int6
4
 6   floors          19414 non-null  floa
t64
 7   age             19414 non-null  int6
4
 8   condition       19414 non-null  int6
4
 9   grade           19414 non-null  int6
```

```
4
 10  waterfront     19414 non-null  floa
t64
 11  yrsince_reno   19414 non-null  floa
t64
 12  sqft_above     19414 non-null  int6
4
 13  sqft_basement  19414 non-null  floa
t64
 14  view           19414 non-null  floa
t64
 15  zipcode        19414 non-null  int6
4
 16  lat            19414 non-null  floa
t64
 17  long           19414 non-null  floa
t64
 18  sqft_living15  19414 non-null  int6
4
 19  sqft_lot15     19414 non-null  int6
4
 20  cond_2         19414 non-null  uint
8
 21  cond_3         19414 non-null  uint
8
 22  cond_4         19414 non-null  uint
8
 23  cond_5         19414 non-null  uint
8
 24  grade_5        19414 non-null  uint
8
 25  grade_6        19414 non-null  uint
8
```

```
 26  grade_7         19414 non-null  uint
8
 27  grade_8         19414 non-null  uint
8
 28  grade_9         19414 non-null  uint
8
 29  grade_10        19414 non-null  uint
8
 30  grade_11        19414 non-null  uint
8
 31  view_1          19414 non-null  uint
8
 32  view_2          19414 non-null  uint
8
dtypes: float64(9), int64(11), uint8(13)
memory usage: 3.4 MB
```

In [283]:

```
hddf_no_out_dum.drop(columns=['id', 'condition', 'view
```

executed in 19ms, finished 00:18:24 2021-06-06

In [284]:

```
hddf_no_out_dum.columns
```

executed in 8ms, finished 00:18:27 2021-06-06

Out[284]:

```
Index(['price', 'bedrooms', 'bathrooms',
'sqft_living', 'sqft_lot', 'floors',
       'age', 'waterfront', 'yrsince_ren
o', 'sqft_above', 'zipcode', 'lat',
       'long', 'sqft_living15', 'sqft_lo
t15', 'cond_2', 'cond_3', 'cond_4',
       'cond_5', 'grade_5', 'grade_6',
'grade_7', 'grade_8', 'grade_9',
       'grade_10', 'grade_11', 'view_1',
'view_2'],
      dtype='object')
```

In [ ]:

In [285]:

```
hddf_no_out_dum.columns
```

executed in 9ms, finished 00:18:31 2021-06-06

Out[285]:

```
Index(['price', 'bedrooms', 'bathrooms',
'sqft_living', 'sqft_lot', 'floors',
       'age', 'waterfront', 'yrsince_ren
o', 'sqft_above', 'zipcode', 'lat',
       'long', 'sqft_living15', 'sqft_lo
t15', 'cond_2', 'cond_3', 'cond_4',
       'cond_5', 'grade_5', 'grade_6',
'grade_7', 'grade_8', 'grade_9',
       'grade_10', 'grade_11', 'view_1',
'view_2'],
      dtype='object')
```

In [286]:

```
outcome = 'price'
predictors = hddf_no_out_dum.drop(columns = ['price'],
predictor_variables = "+".join(predictors.columns)
formula = outcome + '~' + predictor_variables
```

executed in 14ms, finished 00:18:42 2021-06-06

In [287]:

```
model_3 = ols(formula = formula, data = hddf_no_out_du
model_3.summary()
```

executed in 365ms, finished 00:18:49 2021-06-06

Out[287]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.687
Model:	OLS	**Adj. R-squared:**	0.686
Method:	Least Squares	**F-statistic:**	1634.
Date:	Sun, 06 Jun 2021	**Prob (F-statistic):**	0.00
Time:	00:18:48	**Log-Likelihood:**	-2.5642e+05
No. Observations:	19414	**AIC:**	5.129e+05
Df Residuals:	19387	**BIC:**	5.131e+05
Df Model:	26		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**

Intercept	-2.901e+06	2.03e+06	-1.428	0.153
bedrooms	-1.071e+04	1474.692	-7.261	0.000
bathrooms	3.228e+04	2425.129	13.312	0.000
sqft_living	92.8872	3.457	26.868	0.000
sqft_lot	0.3109	0.143	2.167	0.030
floors	2.951e+04	2615.585	11.283	0.000
age	2710.0673	92.292	29.364	0.000
waterfront	5.692e-05	8.88e-06	6.408	0.000
yrsince_reno	-678.5692	94.844	-7.155	0.000
sqft_above	-5.0194	3.394	-1.479	0.139
zipcode	-387.4582	23.258	-16.659	0.000
lat	5.775e+05	7404.747	77.985	0.000
long	-1.089e+05	1.04e+04	-10.494	0.000
sqft_living15	41.2736	2.708	15.244	0.000
sqft_lot15	-1.4249	0.186	-7.652	0.000
cond_2	1.968e+04	2.87e+04	0.687	0.492
cond_3	3.77e+04	2.66e+04	1.415	0.157
cond_4	6.598e+04	2.67e+04	2.476	0.013
cond_5	9.943e+04	2.68e+04	3.710	0.000
grade_5	1.646e+04	3.65e+04	0.451	0.652
grade_6	5.032e+04	3.55e+04	1.418	0.156
grade_7	1.029e+05	3.55e+04	2.902	0.004

grade_8	1.797e+05	3.56e+04	5.052	0.000
grade_9	3.092e+05	3.58e+04	8.639	0.000
grade_10	4.081e+05	3.61e+04	11.296	0.000
grade_11	5.037e+05	3.78e+04	13.311	0.000
view_1	9.295e+04	7973.057	11.658	0.000
view_2	7.299e+04	4986.210	14.639	0.000

Omnibus:	4996.600	**Durbin-Watson:**	1.967
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	22129.820
Skew:	1.196	**Prob(JB):**	0.00
Kurtosis:	7.651	**Cond. No.**	1.00e+16

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 1.9e-18. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

Observations

In [288]:

```
X = hddf_no_out_dum.drop("price", axis = 1)
y = hddf_no_out_dum['price']

X_train, X_test, y_train, y_test = train_test_split(X,
```

executed in 41ms, finished 00:19:07 2021-06-06

In [289]:

```
print(len(X_train))

print(len(X_test))
```

executed in 6ms, finished 00:19:10 2021-06-06

```
14560
4854
```

In [290]:

```
model3_linreg = LinearRegression()
model3_linreg.fit(X_train, y_train)
y_pred = model3_linreg.predict(X_test)
residuals = y_pred - y_test
```

executed in 63ms, finished 00:19:20 2021-06-06

In [291]:

```
model3_linreg.coef_
```

executed in 8ms, finished 00:19:23 2021-06-06

Out[291]:

```
array([-1.32970488e+04,  3.32426783e+04,
9.53766766e+01,  2.24052922e-01,
        3.01113268e+04,  2.82601481e+03,
-6.81270649e-08, -7.74770832e+02,
       -5.50999336e+00, -3.87643315e+02,
5.70464668e+05, -1.02084758e+05,
        4.12752935e+01, -1.32573963e+00,
1.64633212e+02,  1.78006263e+04,
        4.61030079e+04,  7.96383084e+04,
2.13421932e+04,  5.29749206e+04,
        1.06597199e+05,  1.81228018e+05,
3.10594835e+05,  4.01056856e+05,
        4.96798512e+05,  9.34346530e+04,
7.27075858e+04])
```

In [292]:

```
mse_train = mean_squared_error(y_train, model3_linreg.
mse_test = mean_squared_error(y_test, y_pred)

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

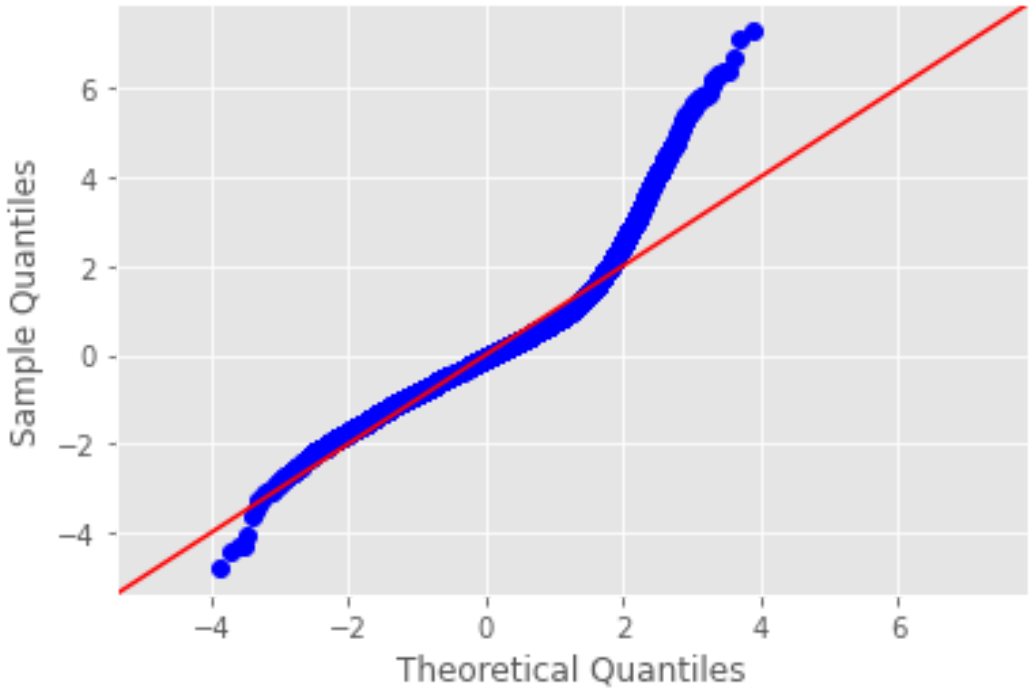
executed in 30ms, finished 00:19:26 2021-06-06

```
Train RMSE: 132072.20998121792
Test RMSE: 131256.31064387024
```

In [293]:

```
fig = sm.graphics.qqplot(model_3.resid, dist=stats.nor
```

executed in 439ms, finished 00:19:30 2021-06-06



In [ ]:

```
##Understanding negative correlation of bedrooms
```

executed in 2m 27s, finished 00:04:34 2021-06-06

In [294]:

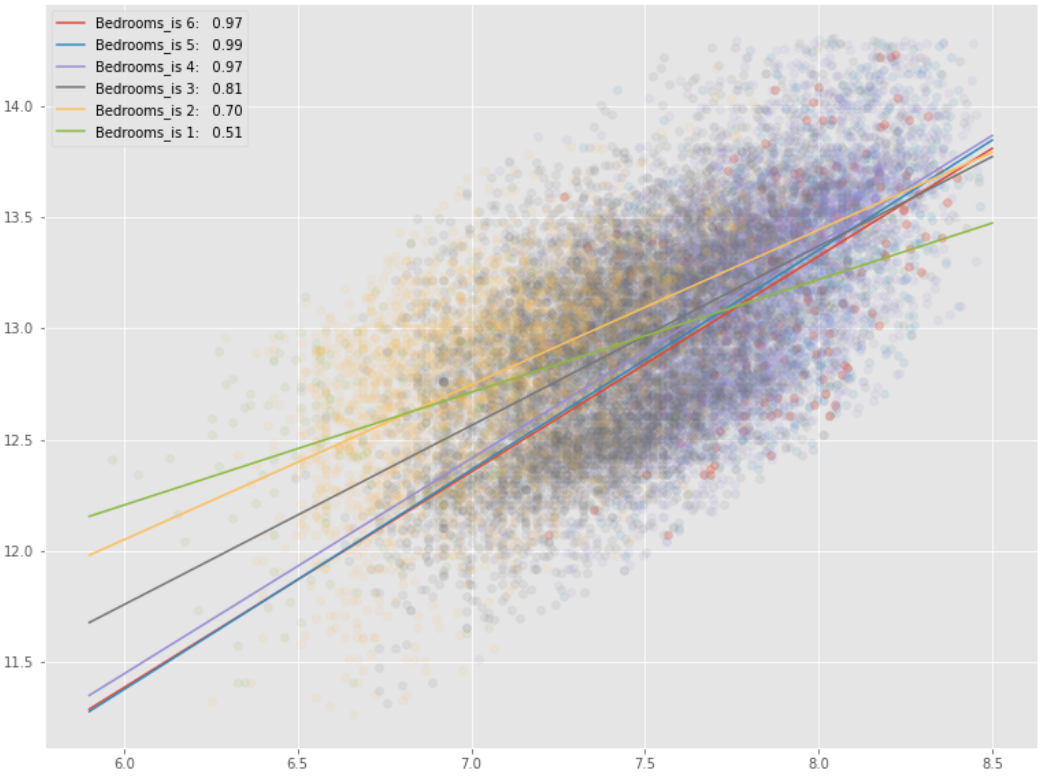
```
# Visualization code here
fig, ax = plt.subplots(figsize=(13, 10))

col = 'sqft_living'
x = np.linspace(5.9, 8.5)


is_in = hddf_no_out_dum.loc[hddf_no_out_dum['bedrooms'
linreg = LinearRegression()
linreg.fit(np.log(is_in[[col]]), np.log(is_in['price']
preds = linreg.predict(np.log(is_in[[col]]))
ax.scatter(np.log(is_in[[col]]), np.log(is_in['price']

ax.plot(x, linreg.predict(x.reshape(-1, 1)), label=f'B

not_in = hddf_no_out_dum.loc[hddf_no_out_dum['bedrooms
linreg = LinearRegression()
linreg.fit(np.log(not_in[[col]]), np.log(not_in['price
preds = linreg.predict(np.log(not_in[[col]]))
ax.scatter(np.log(not_in[[col]]), np.log(not_in['price

ax.plot(x, linreg.predict(x.reshape(-1, 1)), label=f'B


not_in = hddf_no_out_dum.loc[hddf_no_out_dum['bedrooms
linreg = LinearRegression()
linreg.fit(np.log(not_in[[col]]), np.log(not_in['price
preds = linreg.predict(np.log(not_in[[col]]))
ax.scatter(np.log(not_in[[col]]), np.log(not_in['price

ax.plot(x, linreg.predict(x.reshape(-1, 1)), label=f'B
```

```
not_in = hddf_no_out_dum.loc[hddf_no_out_dum['bedrooms
linreg = LinearRegression()
linreg.fit(np.log(not_in[[col]]), np.log(not_in['price
preds = linreg.predict(np.log(not_in[[col]]))
ax.scatter(np.log(not_in[[col]]), np.log(not_in['price

ax.plot(x, linreg.predict(x.reshape(-1, 1)), label=f'B

not_in = hddf_no_out_dum.loc[hddf_no_out_dum['bedrooms
linreg = LinearRegression()
linreg.fit(np.log(not_in[[col]]), np.log(not_in['price
preds = linreg.predict(np.log(not_in[[col]]))
ax.scatter(np.log(not_in[[col]]), np.log(not_in['price

ax.plot(x, linreg.predict(x.reshape(-1, 1)), label=f'B

not_in = hddf_no_out_dum.loc[hddf_no_out_dum['bedrooms
linreg = LinearRegression()
linreg.fit(np.log(not_in[[col]]), np.log(not_in['price
preds = linreg.predict(np.log(not_in[[col]]))
ax.scatter(np.log(not_in[[col]]), np.log(not_in['price

ax.plot(x, linreg.predict(x.reshape(-1, 1)), label=f'B

ax.legend()
```

executed in 1.96s, finished 00:19:45 2021-06-06

Out[294]:

```
<matplotlib.legend.Legend at 0x173
2a2a2190>
```



In [295]:

```
try:
    bedrooms_df=hddf_no_out_dum.drop(columns=['sqft_lo
       'waterfront', 'yrsince_reno', 'sqft_above', 'zi
       'sqft_living15', 'sqft_lot15', 'cond_2', 'cond_
       'grade_5', 'grade_6', 'grade_7', 'grade_8', 'gr
       'grade_11', 'view_1', 'view_2', 'log_price', 'l
       'log_sqft_lot', 'log_sqft_living15', 'log_sqft_
       ])
except:
    bedrooms_df=hddf_no_out_dum
```

executed in 9ms, finished 00:19:54 2021-06-06

In [296]:

```
bedrooms_df['price_per_bedroom'] = bedrooms_df['price'
bedrooms_df['price_per_sqft'] = bedrooms_df['price']/b
```

executed in 14ms, finished 00:19:58 2021-06-06

In [297]:

```
br_group=bedrooms_df.groupby(by='bedrooms').mean()
br_group[['price','price_per_bedroom','sqft_living','p
```

executed in 70ms, finished 00:20:00 2021-06-06

Out[297]:

bedrooms	price	price_per_bedroom	s
1	318095.961039	318095.961039	8
2	388937.274487	194468.637244	12
3	443293.384936	147764.461645	17
4	562155.756885	140538.939221	24
5	616760.016406	123352.003281	27
6	614325.417989	102387.569665	27

In [298]:

```
br_group
```

executed in 47ms, finished 00:20:21 2021-06-06

Out[298]:

bedrooms	price	bathrooms	sqft_living
1	318095.961039	1.137987	891.87013(
2	388937.274487	1.392567	1211.13511
3	443293.384936	1.946832	1755.48506
4	562155.756885	2.355902	2407.63918(
5	616760.016406	2.583008	2714.44218
6	614325.417989	2.649471	2772.27513

6 rows × 29 columns

In [299]:

```
bedrooms_df.bedrooms.value_counts()
```

executed in 11ms, finished 00:20:31 2021-06-06

Out[299]:

```
3    9108
4    6100
2    2583
5    1280
6     189
1     154
Name: bedrooms, dtype: int64
```

4.5 Model 4 - Transformations

In [300]:

```
hddf_no_out_dum.info()
```

executed in 24ms, finished 00:20:41 2021-06-06

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19414 entries, 0 to 21595
Data columns (total 30 columns):
 #   Column             Non-Null Count
Dtype
---  ------             --------------
-----
 0   price              19414 non-null
float64
 1   bedrooms           19414 non-null
int64
 2   bathrooms          19414 non-null
float64
 3   sqft_living        19414 non-null
int64
 4   sqft_lot           19414 non-null
int64
 5   floors             19414 non-null
float64
 6   age                19414 non-null
int64
 7   waterfront         19414 non-null
float64
 8   yrsince_reno       19414 non-null
float64
 9   sqft_above         19414 non-null
```

```
int64
 10  zipcode            19414 non-null
int64
 11  lat                19414 non-null
float64
 12  long               19414 non-null
float64
 13  sqft_living15      19414 non-null
int64
 14  sqft_lot15         19414 non-null
int64
 15  cond_2             19414 non-null
uint8
 16  cond_3             19414 non-null
uint8
 17  cond_4             19414 non-null
uint8
 18  cond_5             19414 non-null
uint8
 19  grade_5            19414 non-null
uint8
 20  grade_6            19414 non-null
uint8
 21  grade_7            19414 non-null
uint8
 22  grade_8            19414 non-null
uint8
 23  grade_9            19414 non-null
uint8
 24  grade_10           19414 non-null
uint8
 25  grade_11           19414 non-null
uint8
```

```
 26  view_1              19414 non-null
uint8
 27  view_2              19414 non-null
uint8
 28  price_per_bedroom   19414 non-null
float64
 29  price_per_sqft      19414 non-null
float64
dtypes: float64(9), int64(8), uint8(13)
memory usage: 2.9 MB
```

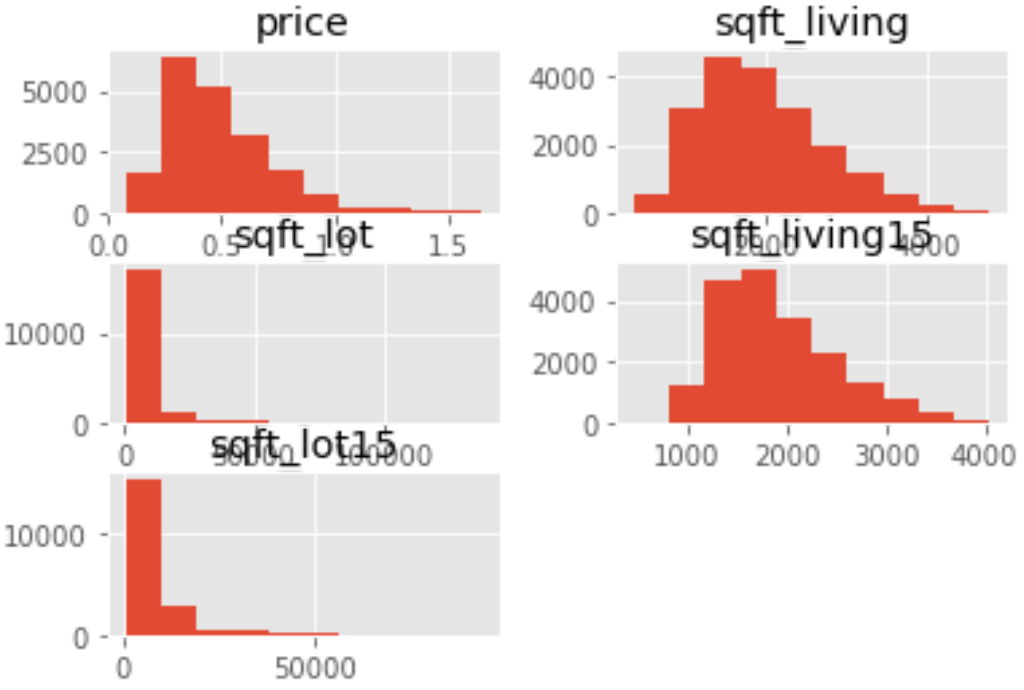
In [301]:

```
hddf_no_out_dum[['price', 'sqft_living', 'sqft_lot', '
```

executed in 1.47s, finished 00:20:53 2021-06-06

Out[301]:

```
array([[<AxesSubplot:title={'center':'pr
ice'}>,
        <AxesSubplot:title={'center':'sq
ft_living'}>],
       [<AxesSubplot:title={'center':'sq
ft_lot'}>,
        <AxesSubplot:title={'center':'sq
ft_living15'}>],
       [<AxesSubplot:title={'center':'sq
ft_lot15'}>, <AxesSubplot:>]],
      dtype=object)
```



In [302]:

```
hddf_no_out_dum['log_price'] = np.log(hddf_no_out_dum[
hddf_no_out_dum['log_sqft_living'] = np.log(hddf_no_ou
hddf_no_out_dum['log_sqft_lot'] = np.log(hddf_no_out_d
hddf_no_out_dum['log_sqft_living15'] = np.log(hddf_no_
hddf_no_out_dum['log_sqft_lot15'] = np.log(hddf_no_out
```

executed in 22ms, finished 00:21:00 2021-06-06

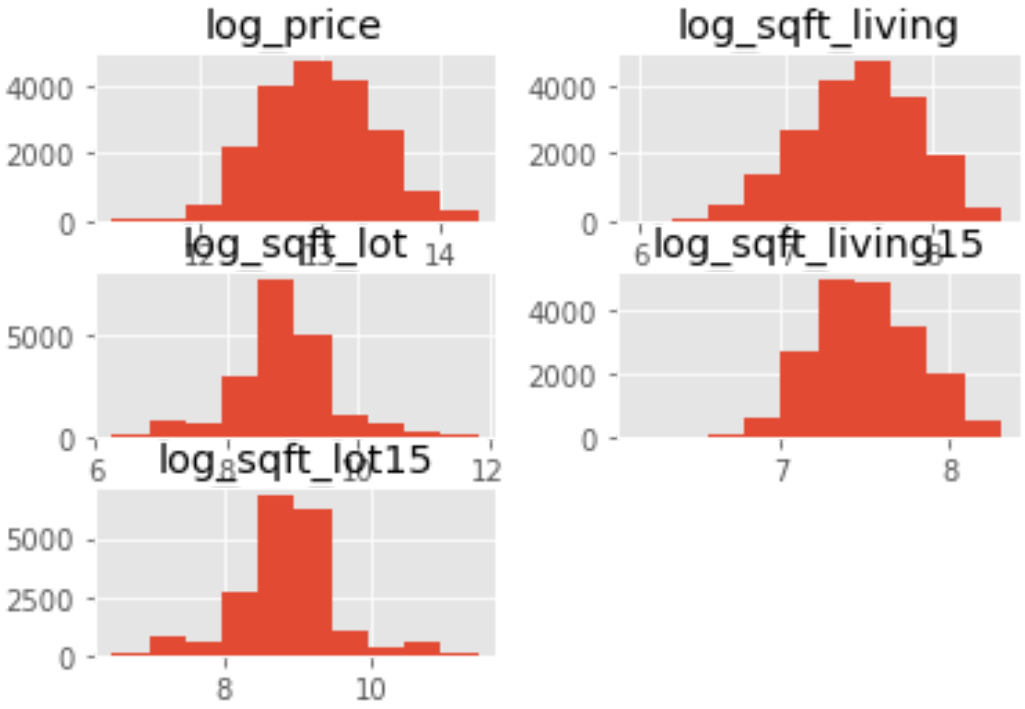
In [303]:

```
hddf_no_out_dum[['log_price','log_sqft_living', 'log_s
```

executed in 1.32s, finished 00:21:07 2021-06-06

Out[303]:

```
array([[<AxesSubplot:title={'center':'lo
g_price'}>,
        <AxesSubplot:title={'center':'lo
g_sqft_living'}>],
       [<AxesSubplot:title={'center':'lo
g_sqft_lot'}>,
        <AxesSubplot:title={'center':'lo
g_sqft_living15'}>],
       [<AxesSubplot:title={'center':'lo
g_sqft_lot15'}>, <AxesSubplot:>]],
      dtype=object)
```



4.5.0.1 hddf_log

In [304]:

```
hddf_log = hddf_no_out_dum.drop(columns = ['price',
                                           'sqft_livin
                                           'sqft_lot',
                                           'sqft_livin
                                           'sqft_lot15
                                           'sqft_above
                                           'grade_5',
                                          ], axis = 1)
```

executed in 15ms, finished 00:21:18 2021-06-06

In [307]:

```
hddf_log = hddf_log.drop(columns = ['price_per_bedroom
                                    'price_per_

                                    ], axis = 1)
```

executed in 11ms, finished 00:24:20 2021-06-06

In [308]:

```
outcome = 'log_price'
predictors = hddf_log.drop(columns = ['log_price'], ax
predictor_variables = "+".join(predictors.columns)
formula = outcome + '~' + predictor_variables
```

executed in 15ms, finished 00:24:23 2021-06-06

In [309]:

```
model_4 = ols(formula = formula, data = hddf_log).fit(
model_4.summary()
```

executed in 308ms, finished 00:24:34 2021-06-06

Out[309]:

OLS Regression Results

Dep. Variable:	log_price	**R-squared:**	0.735
Model:	OLS	**Adj. R-squared:**	0.735
Method:	Least Squares	**F-statistic:**	2244.
Date:	Sun, 06 Jun 2021	**Prob (F-statistic):**	0.00
Time:	00:24:34	**Log-Likelihood:**	250.53
No. Observations:	19414	**AIC:**	-451.1
Df Residuals:	19389	**BIC:**	-254.2
Df Model:	24		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|
Intercept	-3.6664	3.706	-0.989	0.322
bedrooms	-0.0207	0.003	-7.640	0.000
bathrooms	0.0559	0.004	12.747	0.000
floors	0.0363	0.005	8.032	0.000
age	0.0048	0.000	28.812	0.000
waterfront	5.566e-14	1.3e-14	4.289	0.000
yrsince_reno	-0.0010	0.000	-5.741	0.000
zipcode	-0.0006	4.22e-05	-14.073	0.000
lat	1.4029	0.013	104.662	0.000
long	-0.0293	0.019	-1.579	0.114
cond_2	0.1043	0.052	2.009	0.045
cond_3	0.2075	0.048	4.297	0.000
cond_4	0.2660	0.048	5.508	0.000
cond_5	0.3233	0.049	6.655	0.000
grade_6	0.1445	0.017	8.365	0.000
grade_7	0.2814	0.017	16.425	0.000
grade_8	0.4411	0.018	24.598	0.000
grade_9	0.6439	0.019	33.610	0.000
grade_10	0.7698	0.021	36.788	0.000

grade_11	0.8721	0.029	30.112	0.000
view_1	0.1641	0.014	11.382	0.000
view_2	0.1282	0.009	14.239	0.000
log_sqft_living	0.3412	0.010	35.620	0.000
log_sqft_lot	-0.0051	0.006	-0.886	0.376
log_sqft_living15	0.2289	0.009	24.546	0.000
log_sqft_lot15	-0.0644	0.006	-10.257	0.000

Omnibus:	308.800	**Durbin-Watson:**	1.988
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	605.622
Skew:	0.032	**Prob(JB):**	3.10e-132
Kurtosis:	3.863	**Cond. No.**	1.00e+16

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 1.86e-18. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

Observations

1. log_sqft_lot no longer relevant
2. long no longer relevant
3. waterfront no longer relevnant

In [310]:

```
X = hddf_log.drop("log_price", axis = 1)
y = hddf_log['log_price']

X_train, X_test, y_train, y_test = train_test_split(X,
```

executed in 30ms, finished 00:24:51 2021-06-06

In [311]:

```
model4_linreg = LinearRegression()
model4_linreg.fit(X_train, y_train)
y_pred = model4_linreg.predict(X_test)
residuals = y_pred - y_test
```

executed in 55ms, finished 00:24:54 2021-06-06

In [312]:

```
model4_linreg.coef_
```

executed in 8ms, finished 00:24:57 2021-06-06

Out[312]:

```
array([-2.38356433e-02,  5.94897171e-02,
3.87189760e-02,  4.96815877e-03,
        5.55111512e-17, -1.08531942e-03,
-6.26854910e-04,  1.39806343e+00,
       -2.05133916e-02,  8.07216560e-02,
1.84373031e-01,  2.43326025e-01,
        3.01279482e-01,  1.38273864e-01,
2.77718780e-01,  4.32487732e-01,
        6.34760929e-01,  7.54959143e-01,
8.56224860e-01,  1.62453908e-01,
        1.27135106e-01,  3.43797209e-01,
-4.58453697e-03,  2.28188945e-01,
       -6.46019459e-02])
```

In [313]:

```
mse_train = mean_squared_error(y_train, model4_linreg.
mse_test = mean_squared_error(y_test, y_pred)

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

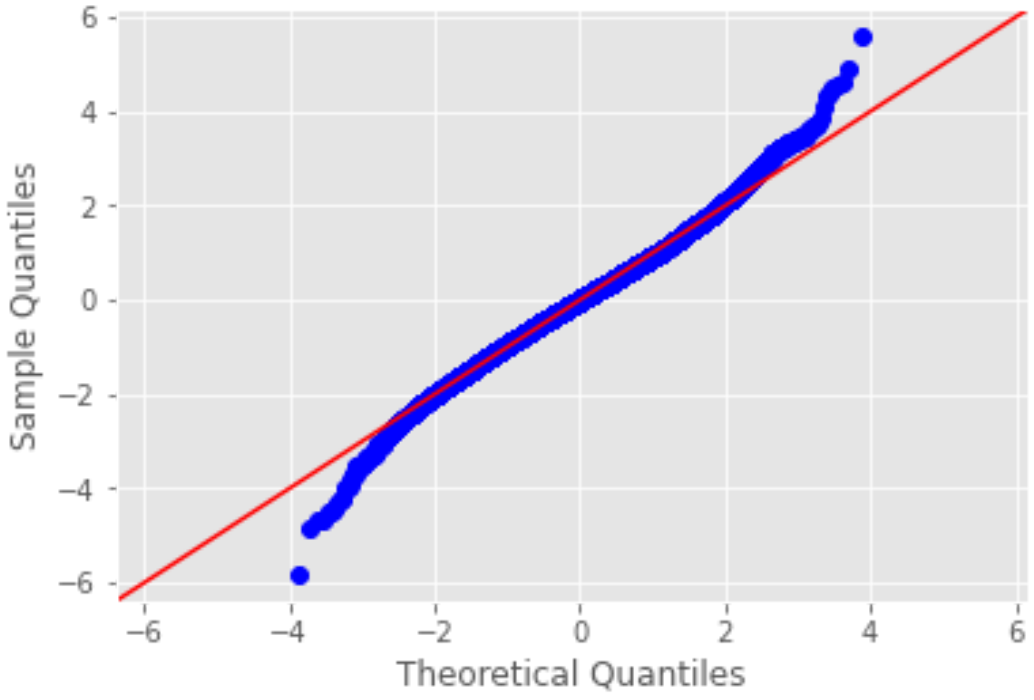
executed in 19ms, finished 00:25:00 2021-06-06

```
Train RMSE: 0.2400775355248702
Test RMSE: 0.23537694857208655
```

In [314]:

```
fig = sm.graphics.qqplot(model_4.resid, dist=stats.nor
```

executed in 448ms, finished 00:25:03 2021-06-06



Observations

4.6 Model 5

1. log_sqft_lot no longer relevant
2. waterfront no longer relevnant

4.6.0.1 hddf_log5

In [315]:

```
hddf_log5 = hddf_log.drop(columns = ['log_sqft_lot', '
hddf_log5.reset_index(inplace=True)
```

executed in 14ms, finished 00:25:17 2021-06-06

In [ ]:

```
hddf_log5.head()
```

executed in 2m 27s, finished 00:04:34 2021-06-06

In [316]:

```
outcome = 'log_price'
predictors = hddf_log5.drop(columns = ['log_price'], a
predictor_variables = "+".join(predictors.columns)
formula = outcome + '~' + predictor_variables
```

executed in 20ms, finished 00:25:21 2021-06-06

In [317]:

```
model_5 = ols(formula = formula, data = hddf_log5).fit
model_5.summary()
```

executed in 409ms, finished 00:25:24 2021-06-06

Out[317]:

OLS Regression Results

Dep. Variable:	log_price	**R-squared:**	0.73
Model:	OLS	**Adj. R-squared:**	0.73
Method:	Least Squares	**F-statistic:**	2248
Date:	Sun, 06 Jun 2021	**Prob (F-statistic):**	0.0
Time:	00:25:24	**Log-Likelihood:**	265.1
No. Observations:	19414	**AIC:**	-480.
Df Residuals:	19389	**BIC:**	-283.
Df Model:	24		
Covariance Type:	nonrobust		

	coef	std err	t
Intercept	-3.3920	3.683	-0.921
index	1.565e-06	2.85e-07	5.483
bedrooms	-0.0207	0.003	-7.644
bathrooms	0.0562	0.004	12.845
floors	0.0359	0.004	8.000
age	0.0049	0.000	29.102
yrsince_reno	-0.0010	0.000	-5.693
zipcode	-0.0006	4.21e-05	-14.136
lat	1.4030	0.013	104.890
long	-0.0283	0.018	-1.529
cond_2	0.1074	0.052	2.071
cond_3	0.2117	0.048	4.390
cond_4	0.2703	0.048	5.604
cond_5	0.3274	0.048	6.752
grade_6	0.1457	0.017	8.443
grade_7	0.2839	0.017	16.575
grade_8	0.4436	0.018	24.753
grade_9	0.6460	0.019	33.747
grade_10	0.7718	0.021	36.906

grade_11	0.8751	0.029	30.231
view_1	0.1646	0.014	11.425
view_2	0.1282	0.009	14.251
log_sqft_living	0.3394	0.009	35.857
log_sqft_living15	0.2290	0.009	24.578
log_sqft_lot15	-0.0674	0.003	-20.199

Omnibus:	311.207	**Durbin-Watson:**	1.99
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	613.26
Skew:	0.029	**Prob(JB):**	6.79 13
Kurtosis:	3.869	**Cond. No.**	2.12e+0

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.12e+08. This might indicate that there are
strong multicollinearity or other numerical problems.

Observations all features look good need to check for multicolinearity between features

In [318]:

```
X = hddf_log5.drop("log_price", axis = 1)
y = hddf_log5['log_price']

X_train, X_test, y_train, y_test = train_test_split(X,
```

executed in 23ms, finished 00:25:33 2021-06-06

In [319]:

```
model5_linreg = LinearRegression()
model5_linreg.fit(X_train, y_train)
y_pred = model5_linreg.predict(X_test)
residuals = y_pred - y_test
```

executed in 57ms, finished 00:25:36 2021-06-06

In [320]:

```
model5_linreg.coef_
```

executed in 10ms, finished 00:25:39 2021-06-06

Out[320]:

```
array([ 1.32788567e-06, -2.37927021e-02,
5.96796149e-02,  3.83716865e-02,
        5.01293350e-03, -1.07672934e-03,
-6.28420995e-04,  1.39821400e+00,
       -1.96524842e-02,  8.53716756e-02,
1.89768474e-01,  2.48652072e-01,
        3.06474179e-01,  1.39417535e-01,
2.79766390e-01,  4.34781703e-01,
        6.36797380e-01,  7.56610980e-01,
8.58915431e-01,  1.62566267e-01,
        1.27014804e-01,  3.42519351e-01,
2.28117944e-01, -6.73167686e-02])
```

In [321]:

```
mse_train = mean_squared_error(y_train, model5_linreg.
mse_test = mean_squared_error(y_test, y_pred)

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

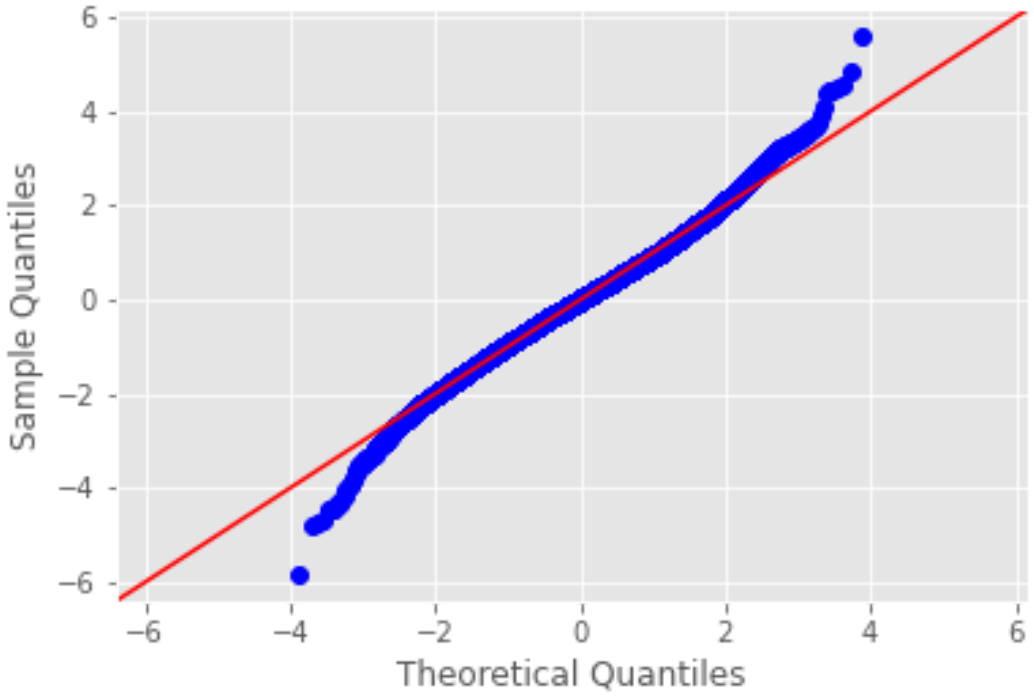
executed in 31ms, finished 00:25:47 2021-06-06

```
Train RMSE: 0.239948550844846
Test RMSE: 0.235054454032162l
```

In [322]:

```
fig = sm.graphics.qqplot(model_5.resid, dist=stats.nor
```

executed in 506ms, finished 00:25:50 2021-06-06



In [ ]:

```
hddf_log5.columns
```

executed in 2m 27s, finished 00:04:34 2021-06-06

In [323]:

```
feat5 =['bedrooms', 'bathrooms', 'floors', 'age', 'yrs
       'lat', 'long', 'cond_2', 'cond_3', 'cond_4', 'c
       'grade_7', 'grade_8', 'grade_9', 'grade_10', 'g
       'view_2', 'log_price', 'log_sqft_living', 'log_
       'log_sqft_lot15']
```

executed in 8ms, finished 00:25:58 2021-06-06

In [324]:

```
corr = hddf_log5[feat5].corr()
corr
```

executed in 165ms, finished 00:26:01 2021-06-06

Out[324]:

	bedrooms	bathrooms	floors	ag
bedrooms	1.000000	0.490961	0.164427	-0.1662
bathrooms	0.490961	1.000000	0.516613	-0.5491
floors	0.164427	0.516613	1.000000	-0.5081
age	-0.166247	-0.549132	-0.508183	1.0000
yrsince_reno	-0.170842	-0.568044	-0.515980	0.9336
zipcode	-0.169266	-0.216688	-0.061218	0.3415
lat	-0.036966	0.001470	0.040225	0.1556
long	0.173683	0.270483	0.143387	-0.4272

In [ ]:

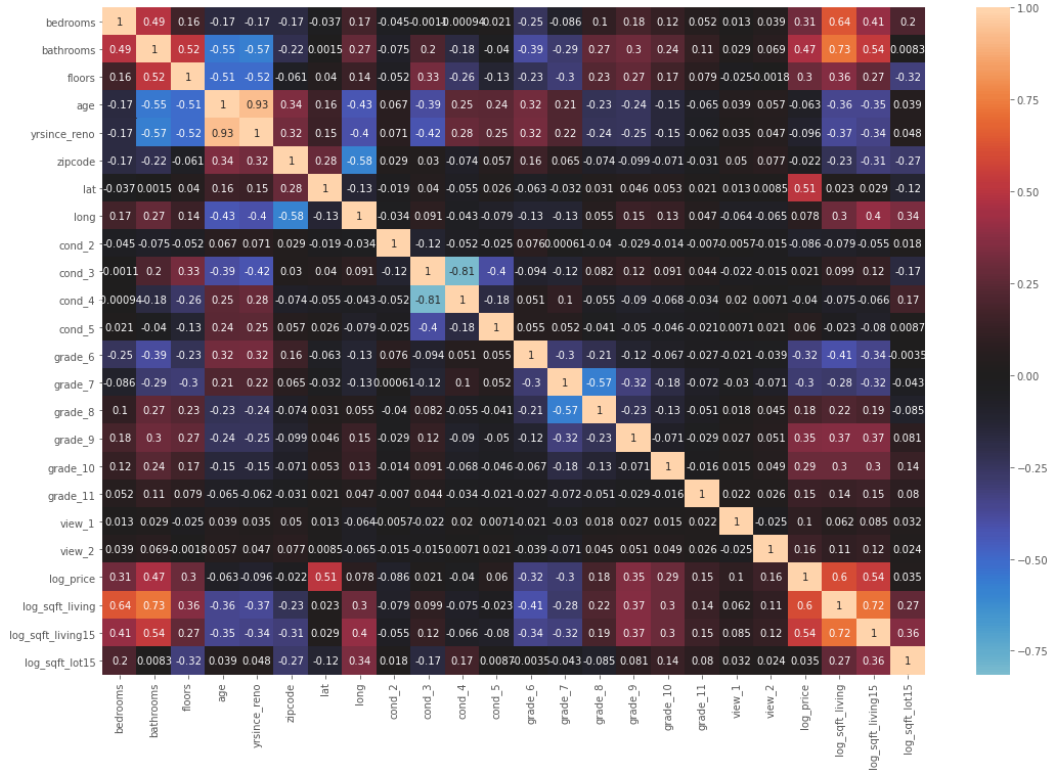
```
#looking for pairwise correlations over .65 is a sign
```

executed in 2m 27s, finished 00:04:34 2021-06-06

In [325]:

```
plt.figure(figsize=(18,12))
sns.heatmap(corr, center=0, annot=True);
```

executed in 14.1s, finished 00:26:23 2021-06-06



Observations: Looking for values greater than .65

Some high correlation between (bedrooms and log_sqft_living)

.73,
(log_sqft_living and logsqftliving15).72,
and (age and age_since_reno) .93.
Can remove logsqftliving15 and either age and age_since_reno

School districts zipcodes Tacoma 98002 fed way 98003 270 Bellevue -98005 Riverview 98019 enumclaw 98022 Isaaquah 98027 Kent 98031 Tahoma 98038 Mercer Island -98040 204 Lake Washington-98052 snoqualmie 98065 268 vashon island 98070 46 seattle pub 98134 shoreline - 98155 424 highline 98166 217 tukwila 98168 264

98001-98199

In [ ]:

```
hddf_log5.info()
```

executed in 2m 27s, finished 00:04:34 2021-06-06

In [326]:

```
len(sorted(hddf_log5['zipcode'].unique()))
```

executed in 10ms, finished 00:26:40 2021-06-06

Out[326]:

```
70
```

In [327]:

```
hddf_log5['zipcode'].min()
```

executed in 10ms, finished 00:26:44 2021-06-06

Out[327]:

```
98001
```

In [328]:

```
!pip install haversine
```

executed in 12.9s, finished 00:27:00 2021-06-06

```
Requirement already satisfied: haversine
in c:\users\dsmith\anaconda3\envs\learn-
env\lib\site-packages (2.3.1)
```

In [329]:

```
import haversine as hs
```

executed in 124ms, finished 00:27:01 2021-06-06

In [330]:

```
amazon=(47.62254,-122.33687)
microsoft=(47.64235,-122.13693)
swedish_iss=(47.63596,-122.02153)
boeing_rent=(47.49226,-122.20616)
loc2=(47.4870,-122.320)
hs.haversine(amazon,loc2)
```

executed in 13ms, finished 00:27:05 2021-06-06

Out[330]:

```
15.124458981947448
```

In [331]:

```
hddf_log5['lat_long'] = tuple(zip(hddf_log5['lat'],hdd
```

executed in 50ms, finished 00:27:09 2021-06-06

In [332]:

```
hddf_log5['lat_long'].tail()
```

executed in 12ms, finished 00:27:12 2021-06-06

Out[332]:

```
19409    (47.5537, -122.398)
19410    (47.5773, -122.409)
19411    (47.6993, -122.346)
19412    (47.5107, -122.362)
19413    (47.5345, -122.069)
Name: lat_long, dtype: object
```

In [333]:

```
#calculate distance from each of the top employers in
distance = []

for i in hddf_log5.lat_long:
    dist_ama = hs.haversine(amazon,i,unit='mi')
    dist_micro = hs.haversine(microsoft,i,unit='mi')
    dist_swed = hs.haversine(swedish_iss,i,unit='mi')
    dist_boeing = hs.haversine(boeing_rent,i,unit='mi'
    distance.append(min(dist_ama,dist_micro,dist_swed,
```

executed in 947ms, finished 00:27:26 2021-06-06

In [334]:

```
hddf_log5['distance'] = pd.Series(distance)
```

executed in 16ms, finished 00:27:29 2021-06-06

In [335]:

```
hddf_log5['distance']
```

executed in 10ms, finished 00:27:32 2021-06-06

Out[335]:

```
0         8.546605
1         6.853555
2         7.971782
3         7.500724
4         1.716678
             ...
19409     5.544209
19410     4.589522
19411     5.320602
19412     7.815713
19413     7.350999
Name: distance, Length: 19414, dtype: fl
oat64
```

In [336]:

```
hddf_log5['lat_long']
```

executed in 22ms, finished 00:27:52 2021-06-06

Out[336]:

```
0          (47.5112, -122.257)
1           (47.721, -122.319)
2          (47.7379, -122.233)
3          (47.5208, -122.393)
4          (47.6168, -122.045)
                  ...
19409      (47.5537, -122.398)
19410      (47.5773, -122.409)
19411      (47.6993, -122.346)
19412      (47.5107, -122.362)
19413      (47.5345, -122.069)
Name: lat_long, Length: 19414, dtype: ob
ject
```

4.7 Model 6 Distance

4.7.0.1 hddf_log6

In [337]:

```
hddf_log6 = hddf_log5.drop(columns = ['lat_long', 'lat
                                      'log_sqft_living
```

executed in 17ms, finished 00:28:10 2021-06-06

In [338]:

```
hddf_log6.head()
```

executed in 46ms, finished 00:28:13 2021-06-06

Out[338]:

	bedrooms	bathrooms	floors	age	zipcode	c
0	3	1.00	1.0	60	98178	
1	3	2.25	2.0	64	98125	
2	2	1.00	1.0	82	98028	
3	4	3.00	1.0	50	98136	
4	3	2.00	1.0	28	98074	

5 rows × 21 columns

In [339]:

```
outcome = 'log_price'
predictors = hddf_log6.drop(columns = ['log_price'], a
predictor_variables = "+".join(predictors.columns)
formula = outcome + '~' + predictor_variables
```

executed in 13ms, finished 00:28:17 2021-06-06

In [340]:

```
model_6 = ols(formula = formula, data = hddf_log6).fit
model_6.summary()
```

executed in 252ms, finished 00:28:23 2021-06-06

cond_4	0.2664	0.045	5.979	0.000	0.179	0.
cond_5	0.3245	0.045	7.243	0.000	0.237	0.
grade_6	0.1256	0.016	7.884	0.000	0.094	0.
grade_7	0.2868	0.016	18.199	0.000	0.256	0.
grade_8	0.4330	0.016	26.328	0.000	0.401	0.
grade_9	0.6064	0.018	34.553	0.000	0.572	0.
grade_10	0.6926	0.019	36.029	0.000	0.655	0.
grade_11	0.8196	0.027	30.828	0.000	0.767	0.
view_1	0.1672	0.013	12.618	0.000	0.141	0.
view_2	0.1363	0.008	16.489	0.000	0.120	0.
log_sqft_living	0.4274	0.008	52.161	0.000	0.411	0.

In [ ]:

In [341]:

```
X = hddf_log6.drop("log_price", axis = 1)
y = hddf_log6['log_price']

X_train, X_test, y_train, y_test = train_test_split(X,
```

executed in 32ms, finished 00:28:42 2021-06-06

In [342]:

```
model6_linreg = LinearRegression()
model6_linreg.fit(X_train, y_train)
y_pred = model6_linreg.predict(X_test)
residuals = y_pred - y_test
```

executed in 47ms, finished 00:28:46 2021-06-06

In [343]:

```
model6_linreg.coef_
```

executed in 9ms, finished 00:28:50 2021-06-06

Out[343]:

```
array([-0.02776458,  0.05089336,  0.0392
9001,  0.00274754, -0.00080889,
        0.04531661,  0.19883733,  0.2455
6404,  0.30455976,  0.12266301,
        0.28516336,  0.42771915,  0.6005
3861,  0.68075352,  0.80672763,
        0.16684553,  0.1362301 ,  0.4285
7337,  0.00742622, -0.03726558])
```

In [344]:

```
mse_train = mean_squared_error(y_train, model6_linreg.
mse_test = mean_squared_error(y_test, y_pred)

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

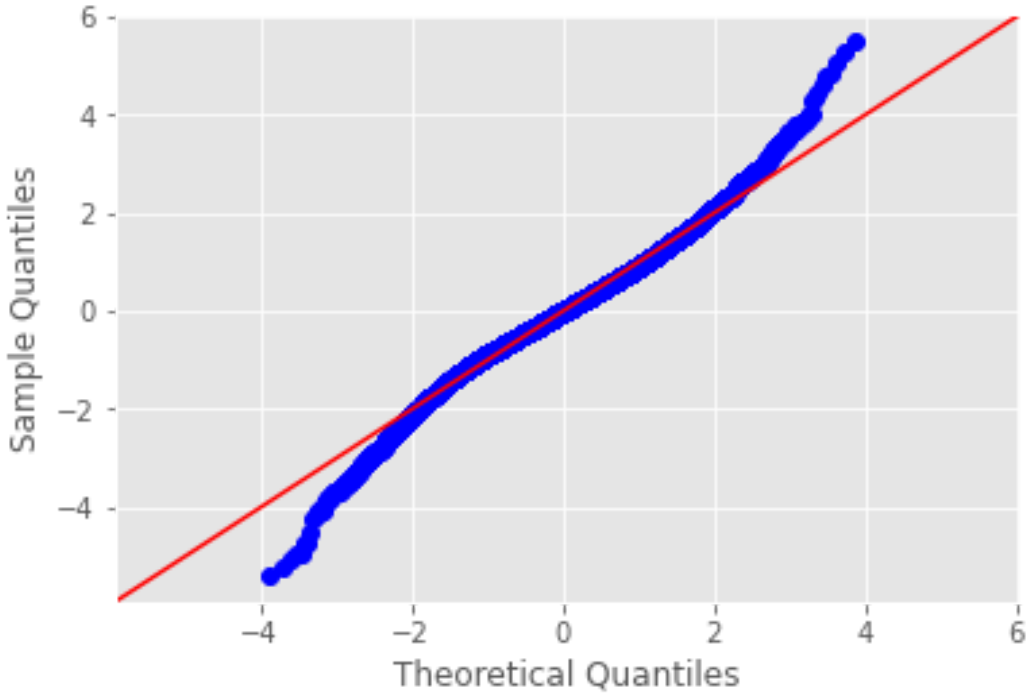
executed in 28ms, finished 00:28:54 2021-06-06

```
Train RMSE: 0.2221194638939187 3
Test RMSE: 0.21659748035627946
```

In [345]:

```
fig = sm.graphics.qqplot(model_6.resid, dist=stats.nor
```

executed in 509ms, finished 00:28:59 2021-06-06



Observation R2 Model a bit less after adding distance.

4.8 Model 7 zipcode dummies

In [346]:

```
zipcode_dummies = pd.get_dummies(hddf_log6['zipcode'],
```

executed in 19ms, finished 00:29:26 2021-06-06

4.8.0.1 hddf_log7

In [347]:

```
hddf_log7 = pd.concat([hddf_log6,zipcode_dummies], axi
```

executed in 15ms, finished 00:29:29 2021-06-06

In [348]:

```
hddf_log7 = hddf_log7.drop(columns = ['zipcode', 'floo
                                      'zip_98003', 'zi
                                      'zip_98092', 'zi
```

executed in 34ms, finished 00:29:32 2021-06-06

In [349]:

```
hddf_log7.info()
```

executed in 48ms, finished 00:29:36 2021-06-06

```
<class 'pandas.core.frame.DataFram
e'>
RangeIndex: 19414 entries, 0 to 19
413
Data columns (total 81 columns):
 #   Column            Non-Null Cou
nt  Dtype
---  ------            ------------
--  -----
 0   bedrooms          19414 non-nu
ll  int64
 1   bathrooms         19414 non-nu
ll  float64
 2   age               19414 non-nu
ll  int64
 3   cond_2            19414 non-nu
ll  uint8
 4   cond_3            19414 non-nu
ll  uint8
```

In [ ]:

```
```

In [350]:

```
outcome = 'log_price'
predictors = hddf_log7.drop(columns = ['log_price'], a
predictor_variables = "+".join(predictors.columns)
formula = outcome + '~' + predictor_variables
```

executed in 20ms, finished 00:29:44 2021-06-06

In [351]:

```
model_7 = ols(formula = formula, data = hddf_log7).fit
model_7.summary()
```

executed in 1.35s, finished 00:29:48 2021-06-06

bathrooms	0.0332	0.003	10.445	0.000	0.027	0.03
age	0.0006	7.27e-05	8.812	0.000	0.000	0.00
cond_2	0.1107	0.038	2.903	0.004	0.036	0.18
cond_3	0.2400	0.035	6.766	0.000	0.170	0.30
cond_4	0.2670	0.035	7.526	0.000	0.197	0.33
cond_5	0.3225	0.036	9.038	0.000	0.253	0.39
grade_6	0.0811	0.013	6.350	0.000	0.056	0.10
grade_7	0.1582	0.013	12.436	0.000	0.133	0.18
grade_8	0.2621	0.013	19.607	0.000	0.236	0.28
grade_9	0.4047	0.014	28.299	0.000	0.377	0.43
grade_10	0.4763	0.016	30.347	0.000	0.446	0.50

In [ ]:

In [352]:

```
X = hddf_log7.drop("log_price", axis = 1)
y = hddf_log7['log_price']

X_train, X_test, y_train, y_test = train_test_split(X,
```

executed in 34ms, finished 00:30:00 2021-06-06

In [353]:

```
model7_linreg = LinearRegression()
model7_linreg.fit(X_train, y_train)
y_pred = model7_linreg.predict(X_test)
residuals = y_pred - y_test
```

executed in 156ms, finished 00:30:03 2021-06-06

In [354]:

```
model7_linreg.coef_
```

executed in 11ms, finished 00:30:05 2021-06-06

Out[354]:

```
array([-1.26067129e-02,  3.6075103
1e-02,  7.01836747e-04,  9.4236741
9e-02,
        2.26955729e-01,  2.5231080
1e-01,  3.08976776e-01,  8.3398489
0e-02,
        1.61823986e-01,  2.6356277
4e-01,  4.05798996e-01,  4.7118100
3e-01,
        6.07186966e-01,  1.1546931
0e-01,  1.18080388e-01,  4.4431938
9e-01,
        5.48233431e-02, -2.1362966
1e-02,  7.28886886e-01,  3.3171060
6e-01,
        2.97564957e-01,  2.2550096
3e-01,  2.09703606e-01,  2.0488082
3e-01,
        1.51819388e-01, -4.3992587
3e-02,  2.48375561e-02,  1.7724835
5e-01,
       -4.11851415e-02,  1.2061803
7e-01,  2.30903418e-01,  1.4257666
2e-01,
```

```
        2.55195788e-01, -3.0523554
2e-02, -4.98755770e-02, -1.5565737
7e-01,
        3.80620540e-01,  1.8855291
4e-01,  9.96308617e-02,  8.8405536
9e-01,
        5.55068764e-01,  1.7478228
1e-01,  2.36289928e-01,  2.2811870
0e-01,
       -6.17115116e-02,  5.4405756
3e-02,  9.69663314e-02,  1.9788268
1e-01,
        1.81796960e-01,  1.8993077
2e-01,  1.27056724e-01,  2.0845671
7e-01,
        1.46766535e-01,  5.0125706
7e-01,  4.22064268e-01,  5.3169066
5e-01,
        4.30556855e-01, -2.2129809
4e-02,  5.25235805e-01,  5.7315798
8e-01,
        4.32039845e-01,  3.5550098
4e-01,  4.24530456e-01,  1.0538114
3e-01,
        5.04558822e-01,  3.6687193
8e-01,  2.10943614e-01,  1.7531992
7e-01,
        1.34049919e-01,  3.2660172
8e-01,  2.30659092e-01, -8.1801358
1e-02,
       -5.40838641e-02,  1.3389137
9e-01, -2.28983306e-01,  2.6407631
4e-01,
```

```
       -1.59268165e-01, -1.3655902
4e-01, -1.04062368e-01,  4.4003686
0e-01])
```

In [355]:

```
mse_train = mean_squared_error(y_train, model7_linreg.
mse_test = mean_squared_error(y_test, y_pred)

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

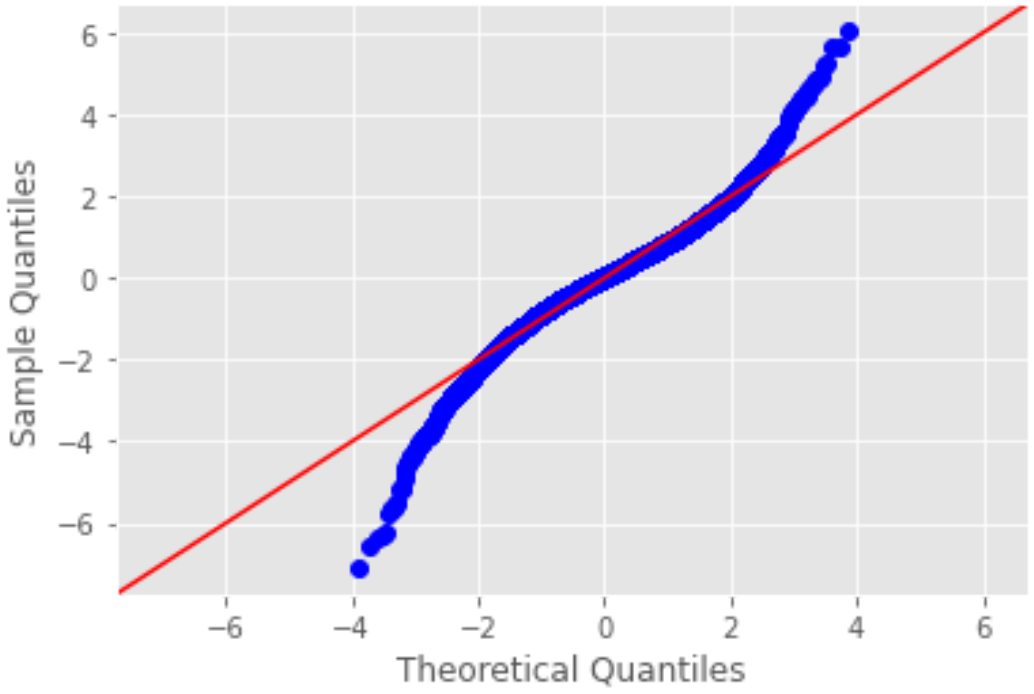
executed in 44ms, finished 00:30:13 2021-06-06

```
Train RMSE: 0.17586824584270275
Test RMSE: 0.17279707456098334
```

In [356]:

```
fig = sm.graphics.qqplot(model_7.resid, dist=stats.nor
```

executed in 487ms, finished 00:30:18 2021-06-06



Observation model 86% variance explained all features relevant

question why grade and condition in model? is view helpful? reducing cat var no more than continous bin condition? <3 bin grade >8

In [357]:

```
try:
    hddf_no_out.reset_index(inplace=True)
except ValueError:
    print("A ValueError exception occurred")
```

executed in 9ms, finished 00:30:30 2021-06-06

In [ ]:

#hddf_no_out want to go bck and create feature where grade is 8+ and remove condition

In [358]:

```
bins = [1,5.99,7.99,13]
hddf_no_out['bin_grade'] = pd.cut(hddf_no_out['grade']
```

executed in 13ms, finished 00:30:35 2021-06-06

In [359]:

```
hddf_no_out['bin_grade'].value_counts()
```

executed in 20ms, finished 00:30:37 2021-06-06

Out[359]:

```
(5.99, 7.99]    10512
(7.99, 13.0]     8683
(1.0, 5.99]       219
Name: bin_grade, dtype: int64
```

In [360]:

```
hddf_no_out[hddf_no_out['grade']>=8]
```

executed in 118ms, finished 00:30:41 2021-06-06

Out[360]:

	index	id	price	bedrooms	bathrooms
4	4	1954400510	510000.0	3	2.0
18	22	7137970340	285000.0	5	2.5
20	24	3814700200	329000.0	3	2.2
22	26	1794500383	937000.0	3	1.7
23	27	3303700376	667000.0	3	1.0

4.8.0.2 hddf_8

In [361]:

```
hddf_8 = hddf_log7
```

executed in 5ms, finished 00:30:52 2021-06-06

In [362]:

```
hddf_8['bin_grade'] = hddf_no_out['bin_grade']
```

executed in 7ms, finished 00:30:55 2021-06-06

In [ ]:

In [ ]:

In [363]:

```
hddf_8.drop(columns = ['grade_6', 'grade_7', 'grade_8'
```

executed in 18ms, finished 00:30:59 2021-06-06

In [364]:

```
hddf_8.drop(columns = ['cond_2', 'cond_3', 'cond_4','c
```

executed in 18ms, finished 00:31:02 2021-06-06

In [ ]:

```
hddf_8.info()
```

executed in 2m 26s, finished 00:04:34 2021-06-06

In [365]:

```
grade_dummies = pd.get_dummies(hddf_8['bin_grade'], pr
```

executed in 8ms, finished 00:31:06 2021-06-06

In [366]:

```
hddf_8 = pd.concat([hddf_8,grade_dummies], axis =1 )
```

executed in 15ms, finished 00:31:15 2021-06-06

In [367]:

```
hddf_8.rename(columns={"grade_(5.99, 7.99]": "grade_6_
```

executed in 7ms, finished 00:31:18 2021-06-06

In [ ]:

```
hddf_8.info()
```

executed in 2m 26s, finished 00:04:34 2021-06-06

In [368]:

```
hddf_8 = pd.concat([hddf_8,hddf_no_out['zipcode']], ax
```

executed in 31ms, finished 00:31:22 2021-06-06

In [ ]:

```
hddf_8
```

executed in 2m 26s, finished 00:04:34 2021-06-06

In [369]:

```
sorted(hddf_8.zipcode.unique())
```

executed in 13ms, finished 00:31:27 2021-06-06

Out[369]:

```
[98001,
 98002,
 98003,
 98004,
 98005,
 98006,
 98007,
 98008,
 98010,
 98011,
 98014,
 98019,
 98022,
 98023,
 98024,
 98027,
 98028,
```

In [370]:

```
#Import districts package got get_districts
from code import districts
```

executed in 49ms, finished 00:31:36 2021-06-06

In [371]:

```
hddf_8['dist'] = pd.Series(districts.get_districts(hdd
```

executed in 55ms, finished 00:31:40 2021-06-06

In [372]:

```
hddf_8['dist'].value_counts()
```

executed in 14ms, finished 00:31:43 2021-06-06

Out[372]:

```
nseattle_9      4492
serural_2       2478
dtseattle_10    2433
anerural_1      2426
fedway_3        2187
redmond_8       2177
renton_5        1815
bellvue_7       1156
mercer_6         204
vi_4              46
Name: dist, dtype: int64
```

In [ ]:

```
hddf_8.columns
```

executed in 2m 26s, finished 00:04:34 2021-06-06

In [ ]:

```
#hddf_8.drop(columns=['d_anerural_1','d_bellvue_7', 'd
```

executed in 2m 26s, finished 00:04:34 2021-06-06

In [ ]:

```

```

In [ ]:

```
hddf_8.columns
```

executed in 2m 26s, finished 00:04:34 2021-06-06

In [373]:

```
dist_dummies = pd.get_dummies(hddf_8['dist'], prefix =
```

executed in 9ms, finished 00:31:53 2021-06-06

In [374]:

```
dist_dummies
```

executed in 29ms, finished 00:31:56 2021-06-06

Out[374]:

	d_anerural_1	d_bellvue_7	d_dtseattle_10
0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	1
4	1	0	0
...	...	...	...
19409	0	0	1
19410	0	0	1
19411	0	0	0
19412	0	0	0
19413	1	0	0

19414 rows × 10 columns

In [ ]:

```
```

In [ ]:

```
```

In [375]:

```
hddf_8 = pd.concat([hddf_8,dist_dummies], axis =1 )
```

executed in 25ms, finished 00:32:07 2021-06-06

In [ ]:

```
hddf_8.info()
```

executed in 2m 26s, finished 00:04:34 2021-06-06

In [ ]:

```
hddf_8.columns
```

executed in 2m 26s, finished 00:04:34 2021-06-06

4.8.0.3 hddf_8_dist

In [376]:

```
hddf_8_dist = hddf_8.drop(columns = ['zip_98004', 'zip
       'zip_98008', 'zip_98010', 'zip_98011', 'zip_980
       'zip_98022', 'zip_98023', 'zip_98024', 'zip_980
       'zip_98029', 'zip_98030', 'zip_98031', 'zip_980
       'zip_98034', 'zip_98038', 'zip_98039', 'zip_980
       'zip_98052', 'zip_98053', 'zip_98055', 'zip_980
       'zip_98065', 'zip_98070', 'zip_98072', 'zip_980
       'zip_98077', 'zip_98102', 'zip_98103', 'zip_981
       'zip_98108', 'zip_98109', 'zip_98112', 'zip_981
       'zip_98117', 'zip_98118', 'zip_98119', 'zip_981
       'zip_98126', 'zip_98133', 'zip_98136', 'zip_981
       'zip_98148', 'zip_98155', 'zip_98168', 'zip_981
       'zip_98188', 'zip_98198', 'zip_98199', 'bin_gra
```

executed in 26ms, finished 00:32:13 2021-06-06

In [ ]:

```
hddf_8_dist.columns
```

executed in 2m 26s, finished 00:04:34 2021-06-06

4.8.0.4 hddf_8_ex

In [377]:

```
hddf_8_ex = hddf_8_dist
```

executed in 6ms, finished 00:32:25 2021-06-06

In [ ]:

```
hddf_8_ex.columns
```

executed in 2m 26s, finished 00:04:34 2021-06-06

In [378]:

```
hddf_8_ex = hddf_8.drop(columns = ['zip_98004', 'zip_9
       'zip_98008', 'zip_98010', 'zip_98011', 'zip_980
       'zip_98022', 'zip_98023', 'zip_98024', 'zip_980
       'zip_98029', 'zip_98030', 'zip_98031', 'zip_980
       'zip_98034', 'zip_98038', 'zip_98039', 'zip_980
       'zip_98052', 'zip_98053', 'zip_98055', 'zip_980
       'zip_98065', 'zip_98070', 'zip_98072', 'zip_980
       'zip_98077', 'zip_98102', 'zip_98103', 'zip_981
       'zip_98108', 'zip_98109', 'zip_98112', 'zip_981
       'zip_98117', 'zip_98118', 'zip_98119', 'zip_981
       'zip_98126', 'zip_98133', 'zip_98136', 'zip_981
       'zip_98148', 'zip_98155', 'zip_98168', 'zip_981
       'zip_98188', 'zip_98198', 'zip_98199', 'bin_gra
```

executed in 15ms, finished 00:32:29 2021-06-06

In [379]:

```
hddf_8_ex['lat'] = hddf_no_out['lat']
```

executed in 11ms, finished 00:32:32 2021-06-06

In [380]:

```
hddf_8_ex['long'] = hddf_no_out['long']
```

executed in 6ms, finished 00:32:36 2021-06-06

In [381]:

```
hddf_8_ex.columns
```

executed in 8ms, finished 00:32:38 2021-06-06

Out[381]:

```
Index(['bedrooms', 'bathrooms', 'age',
'view_1', 'view_2', 'log_price',
       'log_sqft_living', 'log_sqft_lot1
5', 'distance', 'grade_6_8',
       'grade_8_13', 'zipcode', 'dist',
'd_anerural_1', 'd_bellvue_7',
       'd_dtseattle_10', 'd_fedway_3',
'd_mercer_6', 'd_nseattle_9',
       'd_redmond_8', 'd_renton_5', 'd_s
erural_2', 'd_vi_4', 'lat', 'long'],
      dtype='object')
```

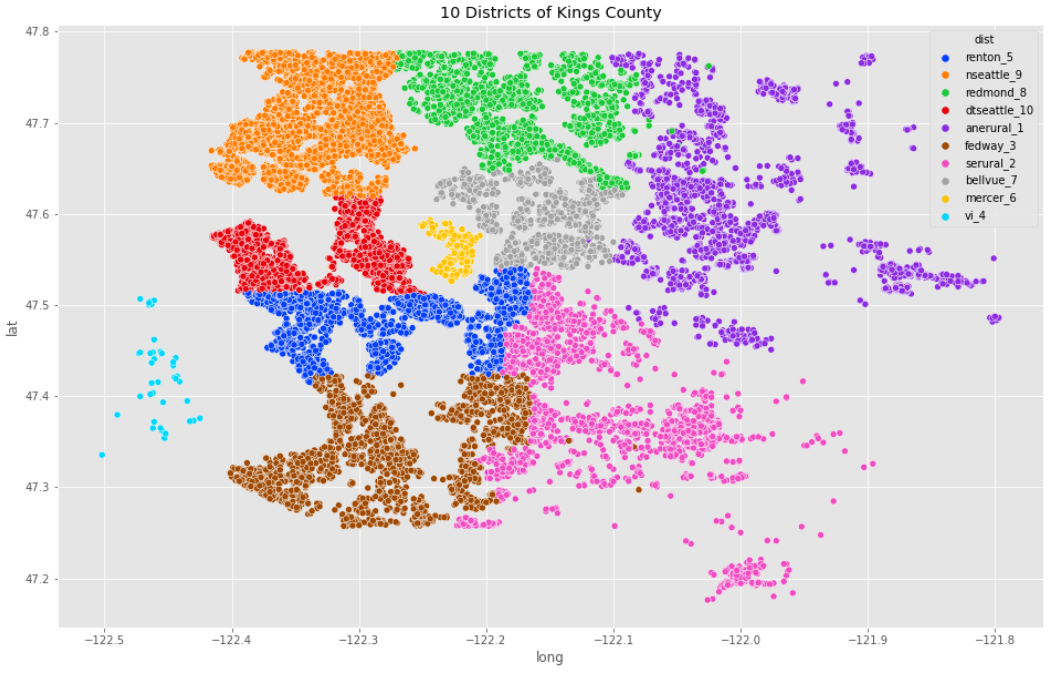
In [480]:

```
fig, ax = plt.subplots(figsize=(16, 10))
ax.set_title('10 Districts of Kings County')
sns.scatterplot(x='long', y='lat', data=hddf_8_ex, hue
```

executed in 3.24s, finished 21:39:42 2021-06-06

Out[480]:

```
<AxesSubplot:title={'center':'10 Districts of Kings County'}, xlabel='long', ylabel='lat'>
```



In [383]:

```
hddf_8_ex['dist'].value_counts()
```

executed in 12ms, finished 00:33:01 2021-06-06

Out[383]:

```
nseattle_9      4492
serural_2       2478
dtseattle_10    2433
anerural_1      2426
fedway_3        2187
redmond_8       2177
renton_5        1815
bellvue_7       1156
mercer_6         204
vi_4              46
Name: dist, dtype: int64
```

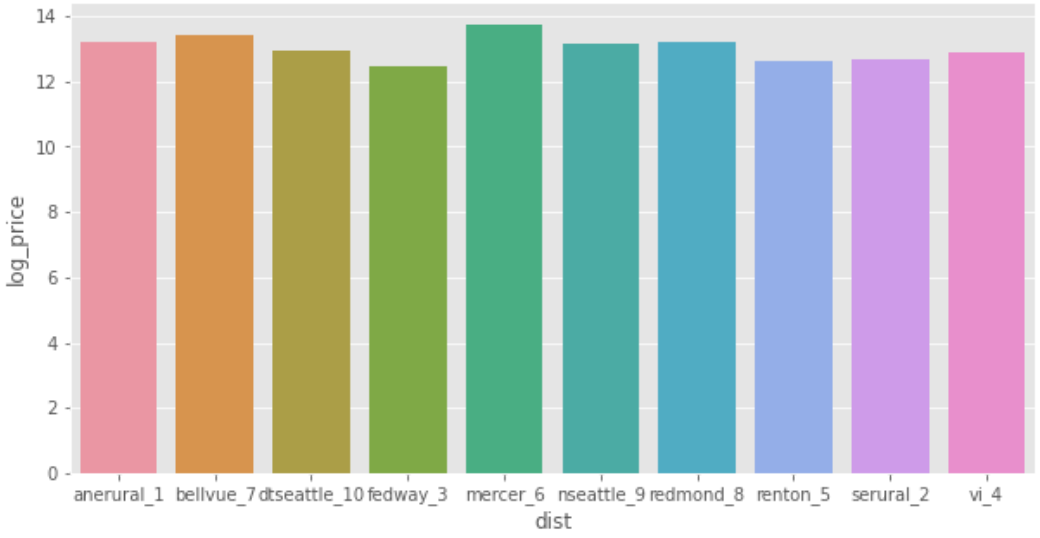
In [384]:

```
plt.figure(figsize=(10,5))
sns.barplot(x = "dist", y = "log_price", data = hddf_8
```

executed in 638ms, finished 00:33:06 2021-06-06

Out[384]:

```
<AxesSubplot:xlabel='dist', ylabel='log_
price'>
```



4.9 Model 8 Districts and Grade Category

In [508]:

```
outcome = 'log_price'
predictors = hddf_8_dist.drop(columns = ['log_price'],
predictor_variables = "+".join(predictors.columns)
formula = outcome + '~' + predictor_variables
```

executed in 671ms, finished 09:44:36 2021-06-07

In [509]:

```
model_8 = ols(formula = formula, data = hddf_8_dist).f
model_8.summary()
```

executed in 8.19s, finished 09:44:44 2021-06-07

Out[509]:

OLS Regression Results

Dep. Variable:	log_price	**R-squared:**	0.78
Model:	OLS	**Adj. R-squared:**	0.78
Method:	Least Squares	**F-statistic:**	381
Date:	Mon, 07 Jun 2021	**Prob (F-statistic):**	0.0
Time:	09:44:43	**Log-Likelihood:**	2457.
No. Observations:	19414	**AIC:**	-487
Df Residuals:	19394	**BIC:**	-471
Df Model:	19		
Covariance Type:	nonrobust		

	coef	std err	t	P
Intercept	7.8532	0.044	177.627	0.(
bedrooms	-0.0313	0.002	-12.953	0.(
bathrooms	0.0552	0.004	14.481	0.(
age	0.0014	7.69e-05	18.429	0.(
view_1	0.1587	0.013	12.382	0.(
view_2	0.1446	0.008	18.098	0.(
log_sqft_living	0.5538	0.007	74.171	0.(
log_sqft_lot15	0.0274	0.003	9.347	0.(
distance	-0.0303	0.001	-56.398	0.(
grade_6_8	0.1043	0.015	6.962	0.(
grade_8_13	0.2588	0.016	16.441	0.(
d_anerural_1	0.7378	0.008	97.524	0.(
d_bellvue_7	0.9091	0.009	105.924	0.(
d_dtseattle_10	0.7006	0.007	104.142	0.(
d_fedway_3	0.6376	0.009	71.467	0.(
d_mercer_6	1.1549	0.015	77.345	0.(
d_nseattle_9	0.8855	0.006	140.341	0.(
d_redmond_8	0.7825	0.007	106.689	0.(
d_renton_5	0.5176	0.007	73.601	0.(
d_serural_2	0.7184	0.008	86.795	0.(

d_vi_4	0.8092	0.029	27.506

Omnibus:	613.954	**Durbin-Watson:**	1.9
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1527.4
Skew:	-0.127	**Prob(JB):**	0.
Kurtosis:	4.351	**Cond. No.**	1.81e+

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 1.78e-27. This might indicate that there are strong multicollinearity problems or that the design matrix is singular.

In [ ]:

In [510]:

```
X = hddf_8_dist.drop("log_price", axis = 1)
y = hddf_8_dist['log_price']

X_train, X_test, y_train, y_test = train_test_split(X,
```

executed in 1.00s, finished 09:44:45 2021-06-07

In [511]:

```
model8_linreg = LinearRegression()
model8_linreg.fit(X_train, y_train)
y_pred = model8_linreg.predict(X_test)
residuals = y_pred - y_test
```

executed in 2.22s, finished 09:44:47 2021-06-07

In [512]:

```
model8_linreg.coef_
```

executed in 182ms, finished 09:44:47 2021-06-07

Out[512]:

```
array([-3.39781200e-02,  5.86644987e-02,
1.46417164e-03,  1.47645970e-01,
        1.43332388e-01,  5.55184277e-01,
2.83500953e-02, -3.00254337e-02,
        1.06889876e-01,  2.58021100e-01,
-4.43063231e-02,  1.30104230e-01,
       -8.58037770e-02, -1.50325131e-01,
3.65229926e-01,  9.89139690e-02,
       -4.34010180e-04, -2.71123358e-01,
-6.73047404e-02,  2.50492145e-02])
```

In [513]:

```
mse_train = mean_squared_error(y_train, model8_linreg.
mse_test = mean_squared_error(y_test, y_pred)

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

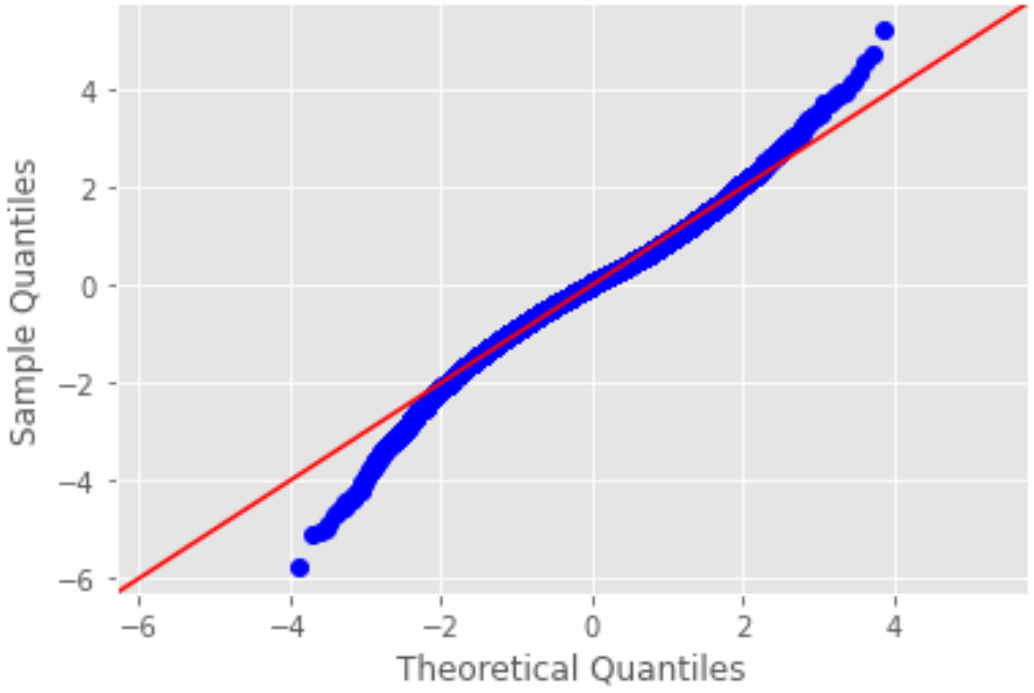
executed in 595ms, finished 09:44:48 2021-06-07

```
Train RMSE: 0.2146664552329853
Test RMSE: 0.2088694793624043
```

In [514]:

```
fig = sm.graphics.qqplot(model_8.resid, dist=stats.nor
```

executed in 5.74s, finished 09:44:53 2021-06-07



Observations: district MercerIsland6 no longer relevant age last renovated coefficient almost insignificant Lets look at one last model with those features removed.

4.10 Model 9 minus fedway3 district removed

4.10.0.1 hddf_9 Drop d_fedway_3 district

In [515]:

```
hddf_9 = hddf_8_dist.drop(columns = ['d_fedway_3', ],
```

executed in 25ms, finished 09:44:54 2021-06-07

In [516]:

```
outcome = 'log_price'
predictors = hddf_9.drop(columns = ['log_price'], axis
predictor_variables = "+".join(predictors.columns)
formula = outcome + '~' + predictor_variables
```

executed in 147ms, finished 09:44:54 2021-06-07

In [517]:

```
model_9 = ols(formula = formula, data = hddf_9).fit()
model_9.summary()
```

executed in 1.69s, finished 09:44:55 2021-06-07

Out[517]:

OLS Regression Results

Dep. Variable:	log_price	**R-squared:**	0.789
Model:	OLS	**Adj. R-squared:**	0.789
Method:	Least Squares	**F-statistic:**	3819.
Date:	Mon, 07 Jun 2021	**Prob (F-statistic):**	0.00
Time:	09:44:55	**Log-Likelihood:**	2457.5
No. Observations:	19414	**AIC:**	-4875.
Df Residuals:	19394	**BIC:**	-4718.
Df Model:	19		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0
Intercept	8.4908	0.049	172.776	0.000	8
bedrooms	-0.0313	0.002	-12.953	0.000	-0
bathrooms	0.0552	0.004	14.481	0.000	0
age	0.0014	7.69e-05	18.429	0.000	0
view_1	0.1587	0.013	12.382	0.000	0
view_2	0.1446	0.008	18.098	0.000	0
log_sqft_living	0.5538	0.007	74.171	0.000	0
log_sqft_lot15	0.0274	0.003	9.347	0.000	0
distance	-0.0303	0.001	-56.398	0.000	-0
grade_6_8	0.1043	0.015	6.962	0.000	0
grade_8_13	0.2588	0.016	16.441	0.000	0
d_anerural_1	0.1002	0.010	9.973	0.000	0
d_bellvue_7	0.2715	0.012	23.521	0.000	0
d_dtseattle_10	0.0630	0.010	6.228	0.000	0
d_mercer_6	0.5173	0.017	30.026	0.000	0
d_nseattle_9	0.2480	0.010	25.927	0.000	0
d_redmond_8	0.1450	0.010	14.553	0.000	0
d_renton_5	-0.1200	0.008	-14.119	0.000	-0
d_serural_2	0.0808	0.006	12.603	0.000	0
d_vi_4	0.1716	0.032	5.339	0.000	0

Omnibus:	613.954	**Durbin-Watson:**	1.984
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1527.478
Skew:	-0.127	**Prob(JB):**	0.00
Kurtosis:	4.351	**Cond. No.**	1.78e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.78e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

In [ ]:

In [518]:

```
X = hddf_9.drop("log_price", axis = 1)
y = hddf_9['log_price']

X_train, X_test, y_train, y_test = train_test_split(X,
```

executed in 1.29s, finished 09:44:57 2021-06-07

In [519]:

```
model9_linreg = LinearRegression()
model9_linreg.fit(X_train, y_train)
y_pred = model9_linreg.predict(X_test)
residuals = y_pred - y_test
```

executed in 1.33s, finished 09:44:58 2021-06-07

In [520]:

```
model9_linreg.coef_
```

executed in 11ms, finished 09:44:58 2021-06-07

Out[520]:

```
array([-0.03397812,  0.0586645 ,  0.0014
6417,  0.14764597,  0.14333239,
        0.55518428,  0.0283501 , -0.0300
2543,  0.10688988,  0.2580211 ,
        0.10601881,  0.28042936,  0.0645
2135,  0.51555506,  0.2492391 ,
        0.14989112, -0.12079823,  0.0830
2039,  0.17537435])
```

In [521]:

```
mse_train = mean_squared_error(y_train, model9_linreg.
mse_test = mean_squared_error(y_test, y_pred)

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

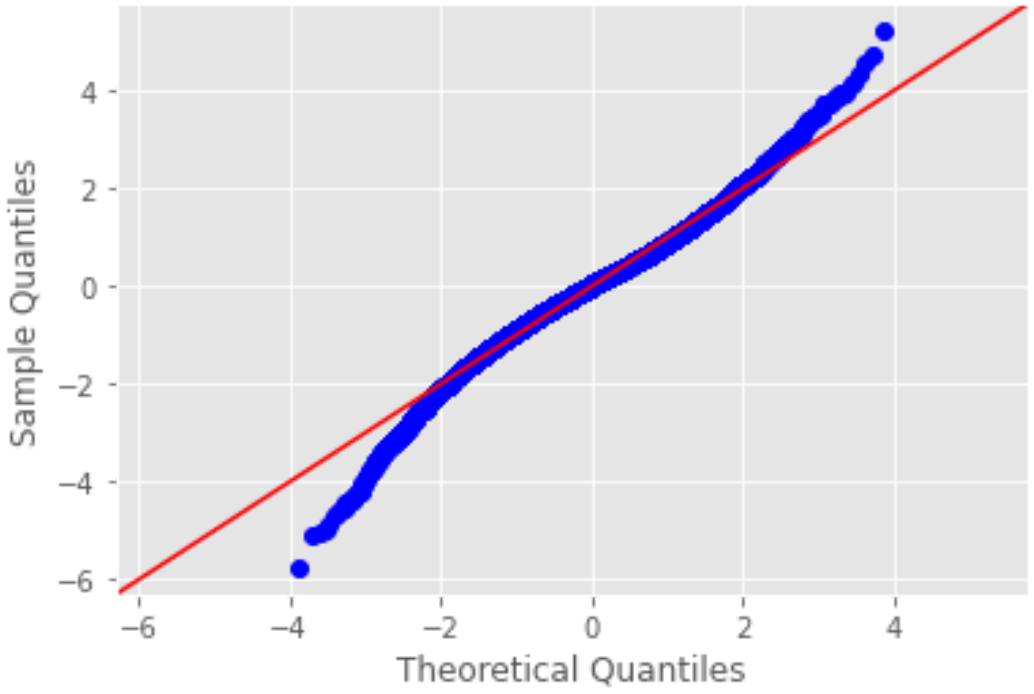
executed in 225ms, finished 09:44:58 2021-06-07

```
Train RMSE: 0.2146664552329852 8
Test RMSE: 0.2088694793624043 5
```

In [522]:

```
fig = sm.graphics.qqplot(model_9.resid, dist=stats.nor
```

executed in 2.18s, finished 09:45:01 2021-06-07



In [ ]:

TODO Bedrooms interaction ...inversede log

4.10.0.2 hddf_8_ex

Folium exploration

In [523]:

```
hddf_8_ex.groupby("dist").median()[["lat", "long", "be
```

executed in 1.88s, finished 09:45:02 2021-06-07

Out[523]:

dist	lat	long	bedrooms
anerural_1	47.59360	-122.020	3
bellvue_7	47.59655	-122.151	4
dtseattle_10	47.55510	-122.321	3
fedway_3	47.33630	-122.286	3
mercer_6	47.56250	-122.224	4
nseattle_9	47.68600	-122.340	3
redmond_8	47.71880	-122.180	3
renton_5	47.49050	-122.271	3
serural_2	47.37010	-122.122	3
vi_4	47.41770	-122.461	3

In [524]:

```
zipcode_g = hddf_8_ex.groupby("dist").mean()[["lat", "
zipcode_g
```

executed in 275ms, finished 09:45:03 2021-06-07

Out[524]:

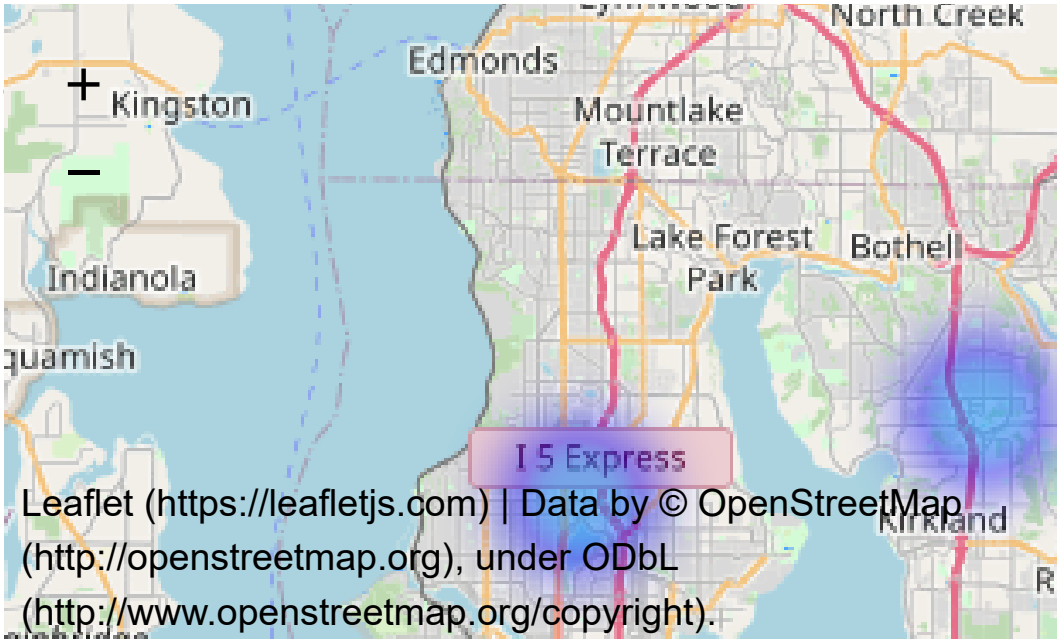
```
[[47.60961673536685, -122.0027093981864
7, 22.744435284418795],
 [47.59511963667822, -122.1550916955018
2, 45.18339100346021],
 [47.55910032881224, -122.3337262638718
7, 59.8734073160707],
 [47.341111659808, -122.27718884316418,
37.90534979423868],
 [47.56048725490193, -122.2250196078431
5, 45.98529411764706],
 [47.69123450578803, -122.3378831255566
2, 61.03517364203027],
 [47.71424092788247, -122.1763514010105
2, 34.98621956821314],
 [47.48384920110196, -122.2682644628097
7, 52.17465564738292],
 [47.38262324455199, -122.1037966101694
8, 28.21630347054076],
 [47.421171739130436, -122.4563043478261
5, 48.913043478260871]]
```

In [525]:

```
import folium
from folium import plugins

m = folium.Map(location=[47.56054, -121.91481],zoom_st
plugins.HeatMap(data = zipcode_g).add_to(m)
m
```

executed in 1.91s, finished 09:45:05 2021-06-07

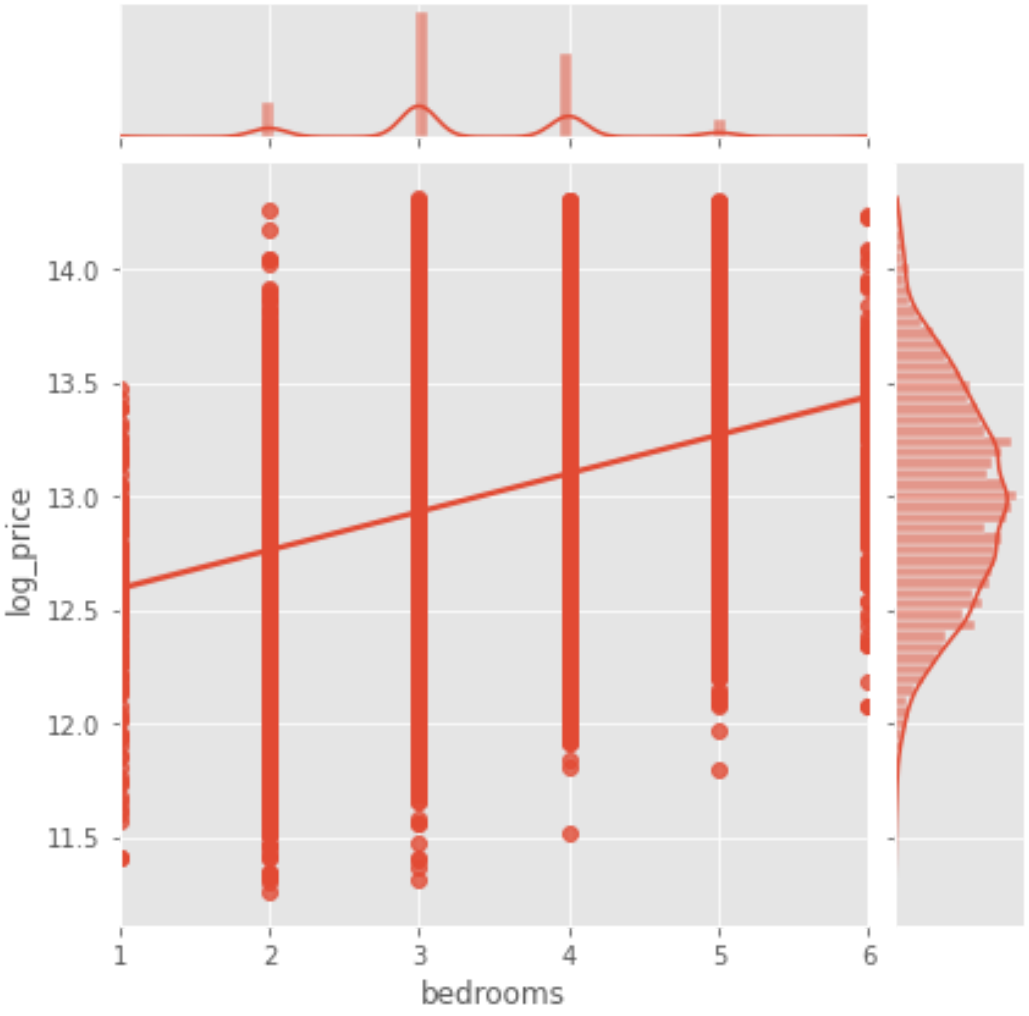
Out[525]:



In [526]:

```
sns.jointplot(x='bedrooms',y='log_price', data=hddf_8_
```

executed in 11.8s, finished 09:45:16 2021-06-07



4.11 Model 10 Check for interactions

In [ ]:

In [527]:

```
hddf_9.info()
```

executed in 653ms, finished 09:45:17 2021-06-07

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19414 entries, 0 to 19413
Data columns (total 20 columns):
 #   Column           Non-Null Count  Dt
ype
---  ------           --------------  --
---
 0   bedrooms         19414 non-null  in
t64
 1   bathrooms        19414 non-null  fl
oat64
 2   age              19414 non-null  in
t64
 3   view_1           19414 non-null  ui
nt8
 4   view_2           19414 non-null  ui
nt8
 5   log_price        19414 non-null  fl
oat64
 6   log_sqft_living  19414 non-null  fl
oat64
 7   log_sqft_lot15   19414 non-null  fl
oat64
 8   distance         19414 non-null  fl
oat64
 9   grade_6_8        19414 non-null  ui
```

```
nt8
 10  grade_8_13       19414 non-null  ui
nt8
 11  d_anerural_1     19414 non-null  ui
nt8
 12  d_bellvue_7      19414 non-null  ui
nt8
 13  d_dtseattle_10   19414 non-null  ui
nt8
 14  d_mercer_6       19414 non-null  ui
nt8
 15  d_nseattle_9     19414 non-null  ui
nt8
 16  d_redmond_8      19414 non-null  ui
nt8
 17  d_renton_5       19414 non-null  ui
nt8
 18  d_serural_2      19414 non-null  ui
nt8
 19  d_vi_4           19414 non-null  ui
nt8
dtypes: float64(5), int64(2), uint8(13)
memory usage: 1.3 MB
```

In [528]:

```
# code to find top interactions by R^2 value here
from itertools import combinations
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
regression = LinearRegression()

X = hddf_9.drop('log_price',  axis=1)
y = hddf_9['log_price']

crossvalidation = KFold(n_splits=10, shuffle=True, ran
baseline = np.mean(cross_val_score(regression, X, y, s
baseline
```

executed in 1.30s, finished 09:45:18 2021-06-07

Out[528]:

```
0.7883988012813241
```

In [ ]:

executed in 1.56s, finished 00:01:47 2021-06-06

In [529]:

```
interactions = []

feat_combinations = combinations(X.columns, 2)

data = X.copy()

for i, (a, b) in enumerate(feat_combinations):
    data['interaction'] = data[a] * data[b]
    score = np.mean(cross_val_score(regression, data,
    if score > baseline:
        interactions.append((a, b, round(score,3)))

    if i % 50 == 0:
        print(i)

print('Top 3 interactions: %s' %sorted(interactions, k
```

executed in 1m 39.9s, finished 09:46:58 2021-06-07

```
0
50
100
150
Top 3 interactions: [('distance',
'd_nseattle_9', 0.803), ('log_sqft
_living', 'grade_6_8', 0.793), ('l
```

```
og sqft_living'  'grade_8_13'  0.7
```

It looks like the top interactions involve the d_nseattle_9 feature so lets add the interaction between distance and d_nseattle_9 to our model.

We can interpret this feature as the relationship between distance and SalePrice when the house is in d_nseattle_9 or not.

4.12 Visualize the Interaction

Separate all houses that are located in d_nseattle_9 and those that are not. Run a linear regression on each population against `logprice`. Visualize the regression line and data points with price on the y axis and distance on the x axis.

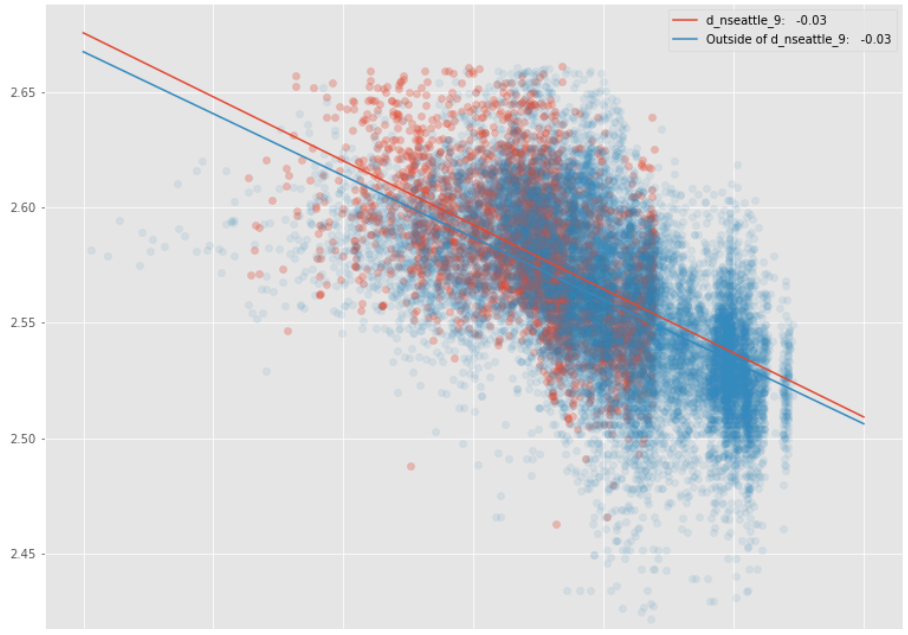
In [530]:

```
# Visualization code here
fig, ax = plt.subplots(figsize=(13, 10))

col = 'distance'
is_in = hddf_9.loc[hddf_9['d_nseattle_9'] == 1, [col,
linreg = LinearRegression()
linreg.fit(np.log(is_in[[col]]), np.log(is_in['log_pri
preds = linreg.predict(np.log(is_in[[col]]))
ax.scatter(np.log(is_in[[col]]), np.log(is_in['log_pri
x = np.linspace(-2, 4)
ax.plot(x, linreg.predict(x.reshape(-1, 1)), label=f'd

not_in = hddf_9.loc[hddf_9['d_nseattle_9'] == 0, [col,
linreg = LinearRegression()
linreg.fit(np.log(not_in[[col]]), np.log(not_in['log_p
preds = linreg.predict(np.log(not_in[[col]]))
ax.scatter(np.log(not_in[[col]]), np.log(not_in['log_p

ax.plot(x, linreg.predict(x.reshape(-1, 1)), label=f'O

ax.legend()
```

executed in 2.56s, finished 09:47:01 2021-06-07

Out[530]:

```
<matplotlib.legend.Legend at 0x1733ba3b3
70>
```



4.13 Build a final model with interactions

Use 10-fold cross-validation to build a model using the above interaction.

In [531]:

```
# code here
regression = LinearRegression()
crossvalidation = KFold(n_splits=10, shuffle=True, ran
final = X.copy()

final['distance*d_nseattle_9'] = final['distance'] * f

final_model = np.mean(cross_val_score(regression, fina

final_model
```

executed in 2.27s, finished 09:47:03 2021-06-07

Out[531]:

```
0.8026853836950311
```

In [532]:

```
final.head()
```

executed in 63ms, finished 09:47:03 2021-06-07

Out[532]:

	bedrooms	bathrooms	age	view_1	view_2	lo
0	3	1.00	60	0	0	
1	3	2.25	64	0	0	
2	2	1.00	82	0	0	
3	4	3.00	50	0	0	
4	3	2.00	28	0	0	

In [533]:

```
# code here
final['log_price'] = hddf_9['log_price']
outcome = 'log_price'
predictors = final.drop(columns = ['log_price'], axis
predictor_variables = "+".join(predictors.columns)
formula = outcome + '~' + predictor_variables
```

executed in 280ms, finished 09:47:04 2021-06-07

In [534]:

```
model_10 = ols(formula = formula, data = final).fit()
model_10.summary()
```

executed in 917ms, finished 09:47:05 2021-06-07

Out[534]:

OLS Regression Results

Dep. Variable:	log_price	**R-squared:**	0.803
Model:	OLS	**Adj. R-squared:**	0.803
Method:	Least Squares	**F-statistic:**	3961.
Date:	Mon, 07 Jun 2021	**Prob (F-statistic):**	0.00
Time:	09:47:04	**Log-Likelihood:**	3137.8
No. Observations:	19414	**AIC:**	-6234.
Df Residuals:	19393	**BIC:**	-6068.
Df Model:	20		
Covariance Type:	nonrobust		

	coef	std err	t	P
Intercept	8.1479	0.048	168.606	0.(
bedrooms	-0.0251	0.002	-10.734	0.(
bathrooms	0.0435	0.004	11.783	0.(
age	0.0007	7.66e-05	9.282	0.(
view_1	0.1613	0.012	13.032	0.(
view_2	0.1381	0.008	17.901	0.(
log_sqft_living	0.5483	0.007	76.040	0.(
log_sqft_lot15	0.0588	0.003	19.897	0.(
distance	-0.0230	0.001	-41.597	0.(
grade_6_8	0.0941	0.014	6.507	0.(
grade_8_13	0.2345	0.015	15.416	0.(
d_anerural_1	0.2012	0.010	19.993	0.(
d_bellvue_7	0.3935	0.012	33.895	0.(
d_dtseattle_10	0.2082	0.011	19.806	0.(
d_mercer_6	0.6207	0.017	36.811	0.(
d_nseattle_9	0.6683	0.015	46.031	0.(
d_redmond_8	0.2488	0.010	24.859	0.(
d_renton_5	-0.0432	0.008	-5.113	0.(
d_serural_2	0.0921	0.006	14.858	0.(
d_vi_4	0.1804	0.031	5.812	0.(

distance:d_nseattle_9	-0.0539	0.001	-37.521	0.(

Omnibus:	799.827	**Durbin-Watson:**	1.990
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2315.372
Skew:	-0.149	**Prob(JB):**	0.00
Kurtosis:	4.665	**Cond. No.**	1.83e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.83e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

In [ ]:

```
```

In [535]:

```
X = final.drop("log_price", axis = 1)
y = final['log_price']

X_train, X_test, y_train, y_test = train_test_split(X,
```

executed in 2.10s, finished 09:47:07 2021-06-07

In [536]:

```
model_10_linreg = LinearRegression()
model_10_linreg.fit(X_train, y_train)
y_pred = model_10_linreg.predict(X_test)
residuals = y_pred - y_test
```

executed in 773ms, finished 09:47:07 2021-06-07

In [537]:

```
model_10_linreg.coef_
```

executed in 71ms, finished 09:47:08 2021-06-07

Out[537]:

```
array([-0.02722363,  0.04637777,  0.0007
4775,  0.14952887,  0.13703141,
        0.54902297,  0.06060631, -0.0225
7943,  0.09696137,  0.2341256 ,
        0.20976453,  0.40560228,  0.2133
5096,  0.62208309,  0.67828833,
        0.25678864, -0.04204509,  0.0942
8725,  0.18774062, -0.05456986])
```

In [538]:

```
mse_train = mean_squared_error(y_train, model_10_linre
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 282ms, finished 09:47:08 2021-06-07

```
Train RMSE: 0.20711188808730613
Test RMSE: 0.2021844535434057 8
```

In [539]:

```
np.sqrt(model_10.scale)
```

executed in 292ms, finished 09:47:08 2021-06-07

Out[539]:

```
0.20597021941758684
```

In [540]:

```
mse_train = mean_squared_error(np.exp(y_train), np.exp
mse_test = mean_squared_error(np.exp(y_test), np.exp(y

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

executed in 375ms, finished 09:47:09 2021-06-07

```
Train RMSE: 112742.0447931623
Test RMSE: 111200.88317938473
```

In [541]:

```
model_10.params
```

executed in 358ms, finished 09:47:09 2021-06-07

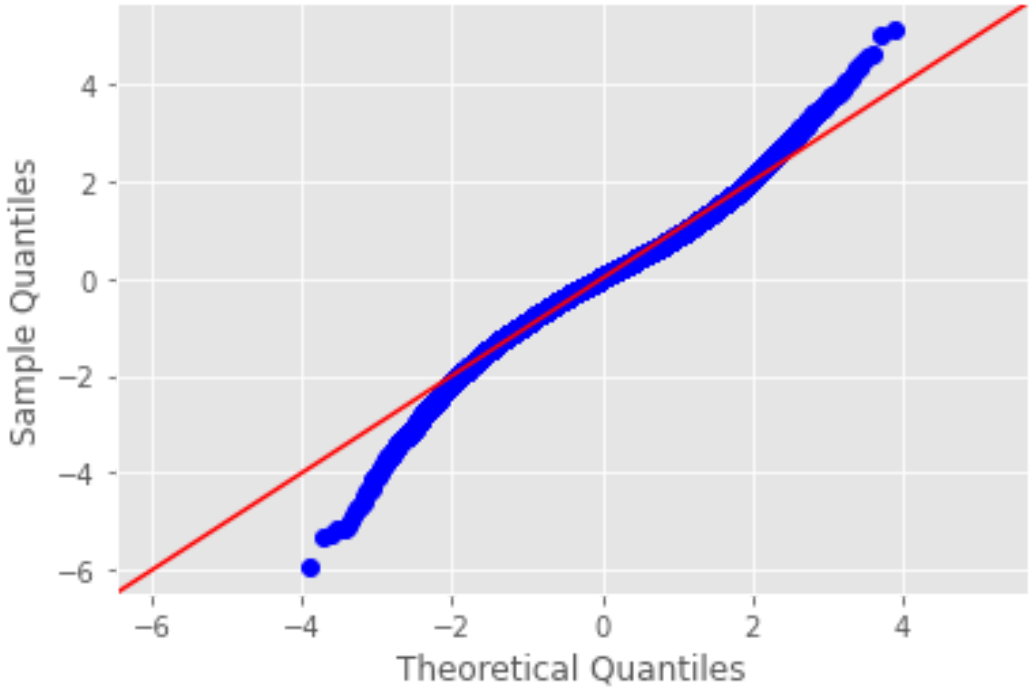
Out[541]:

```
Intercept                 8.147858
bedrooms                 -0.025074
bathrooms                 0.043535
age                       0.000711
view_1                    0.161327
view_2                    0.138131
log_sqft_living           0.548327
log_sqft_lot15            0.058759
distance                 -0.023028
grade_6_8                 0.094107
grade_8_13                0.234535
d_anerural_1              0.201242
d_bellvue_7               0.393494
d_dtseattle_10            0.208234
d_mercer_6                0.620723
d_nseattle_9              0.668327
d_redmond_8               0.248800
d_renton_5               -0.043234
d_serural_2               0.092105
d_vi_4                    0.180416
distance:d_nseattle_9    -0.053882
dtype: float64
```

In [542]:

```
fig = sm.graphics.qqplot(model_10.resid, dist=stats.no
```

executed in 1.25s, finished 09:47:10 2021-06-07



Type *Markdown* and LaTeX: α^2

Investigate why bedrooms is a negative correlation

In [543]:

```
final.head()
```

executed in 89ms, finished 09:47:10 2021-06-07

Out[543]:

	bedrooms	bathrooms	age	view_1	view_2	lo
0	3	1.00	60	0	0	
1	3	2.25	64	0	0	
2	2	1.00	82	0	0	
3	4	3.00	50	0	0	
4	3	2.00	28	0	0	

5 rows × 21 columns

In [544]:

```
final.columns
```

executed in 354ms, finished 09:47:11 2021-06-07

Out[544]:

```
Index(['bedrooms', 'bathrooms', 'age',
'view_1', 'view_2', 'log_sqft_living',
       'log_sqft_lot15', 'distance', 'gr
ade_6_8', 'grade_8_13', 'd_anerural_1',
       'd_bellvue_7', 'd_dtseattle_10',
'd_mercer_6', 'd_nseattle_9',
       'd_redmond_8', 'd_renton_5', 'd_s
erural_2', 'd_vi_4',
       'distance*d_nseattle_9', 'log_pri
ce'],
      dtype='object')
```

In [545]:

```
final['bedrooms'].value_counts()
```

executed in 284ms, finished 09:47:11 2021-06-07

Out[545]:

```
3    9108
4    6100
2    2583
5    1280
6     189
1     154
Name: bedrooms, dtype: int64
```

In [546]:

```
final['distance'].describe()
```

executed in 282ms, finished 09:47:11 2021-06-07

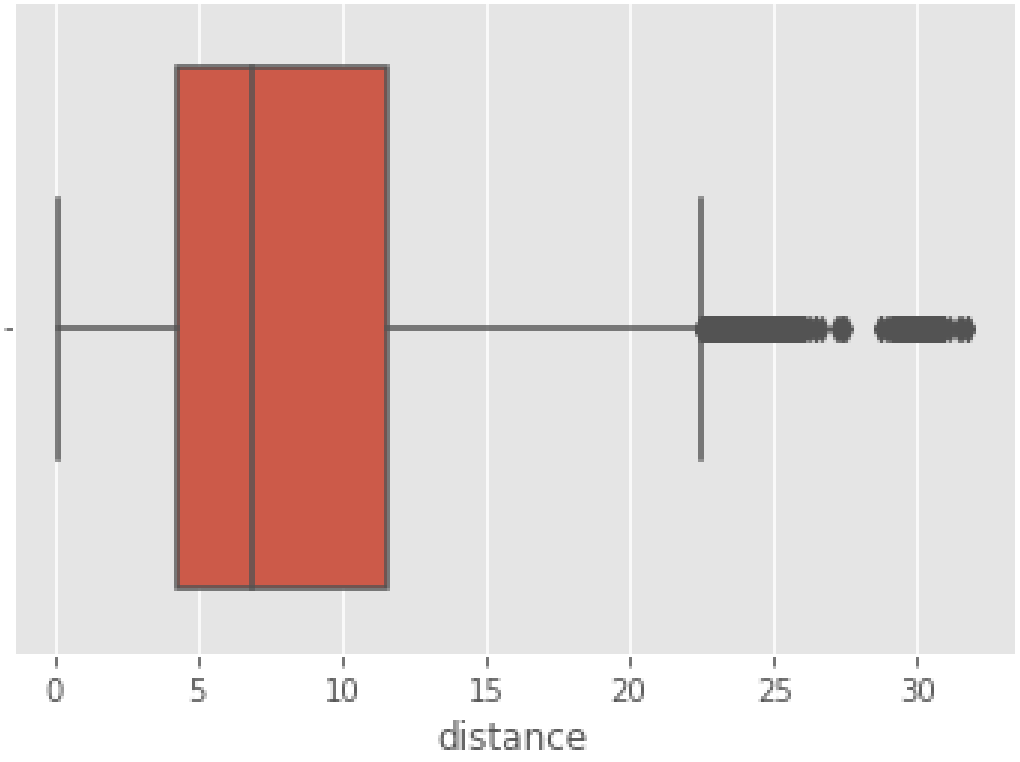
Out[546]:

```
count    19414.000000
mean         8.975723
std          6.468720
min          0.144005
25%          4.256431
50%          6.855100
75%         11.565158
max         31.753260
Name: distance, dtype: float64
```

In [547]:

```
sns.boxplot(x=final['distance']);
```

executed in 1.24s, finished 09:47:12 2021-06-07



In [548]:

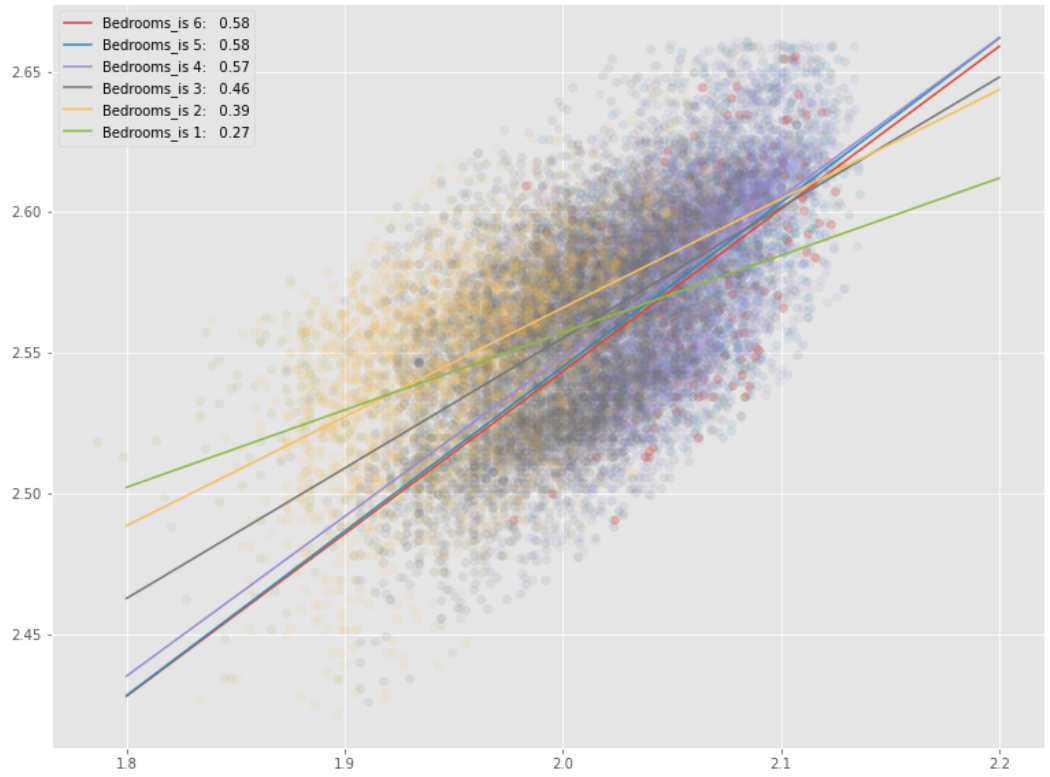
```
# Visualization code here
fig, ax = plt.subplots(figsize=(13, 10))

col = 'log_sqft_living'
x = np.linspace(1.8, 2.2)

is_in = hddf_9.loc[hddf_9['bedrooms'] == 6, [col, 'log
linreg = LinearRegression()
linreg.fit(np.log(is_in[[col]]), np.log(is_in['log_pri
preds = linreg.predict(np.log(is_in[[col]]))
ax.scatter(np.log(is_in[[col]]), np.log(is_in['log_pri

ax.plot(x, linreg.predict(x.reshape(-1, 1)), label=f'B

not_in = hddf_9.loc[hddf_9['bedrooms'] == 5 , [col, 'l
linreg = LinearRegression()
linreg.fit(np.log(not_in[[col]]), np.log(not_in['log_p
preds = linreg.predict(np.log(not_in[[col]]))
ax.scatter(np.log(not_in[[col]]), np.log(not_in['log_p

ax.plot(x, linreg.predict(x.reshape(-1, 1)), label=f'B


not_in = hddf_9.loc[hddf_9['bedrooms'] == 4, [col, 'lo
linreg = LinearRegression()
linreg.fit(np.log(not_in[[col]]), np.log(not_in['log_p
preds = linreg.predict(np.log(not_in[[col]]))
ax.scatter(np.log(not_in[[col]]), np.log(not_in['log_p

ax.plot(x, linreg.predict(x.reshape(-1, 1)), label=f'B
```

```
not_in = hddf_9.loc[hddf_9['bedrooms'] == 3, [col, 'lo
linreg = LinearRegression()
linreg.fit(np.log(not_in[[col]]), np.log(not_in['log_p
preds = linreg.predict(np.log(not_in[[col]]))
ax.scatter(np.log(not_in[[col]]), np.log(not_in['log_p

ax.plot(x, linreg.predict(x.reshape(-1, 1)), label=f'B

not_in = hddf_9.loc[hddf_9['bedrooms'] == 2, [col, 'lo
linreg = LinearRegression()
linreg.fit(np.log(not_in[[col]]), np.log(not_in['log_p
preds = linreg.predict(np.log(not_in[[col]]))
ax.scatter(np.log(not_in[[col]]), np.log(not_in['log_p

ax.plot(x, linreg.predict(x.reshape(-1, 1)), label=f'B

not_in = hddf_9.loc[hddf_9['bedrooms'] == 1, [col, 'lo
linreg = LinearRegression()
linreg.fit(np.log(not_in[[col]]), np.log(not_in['log_p
preds = linreg.predict(np.log(not_in[[col]]))
ax.scatter(np.log(not_in[[col]]), np.log(not_in['log_p

ax.plot(x, linreg.predict(x.reshape(-1, 1)), label=f'B

ax.legend()
```

executed in 2.06s, finished 09:47:14 2021-06-07

Out[548]:

```
<matplotlib.legend.Legend at 0x173
3ba1d6a0>
```



In [ ]:

In [549]:

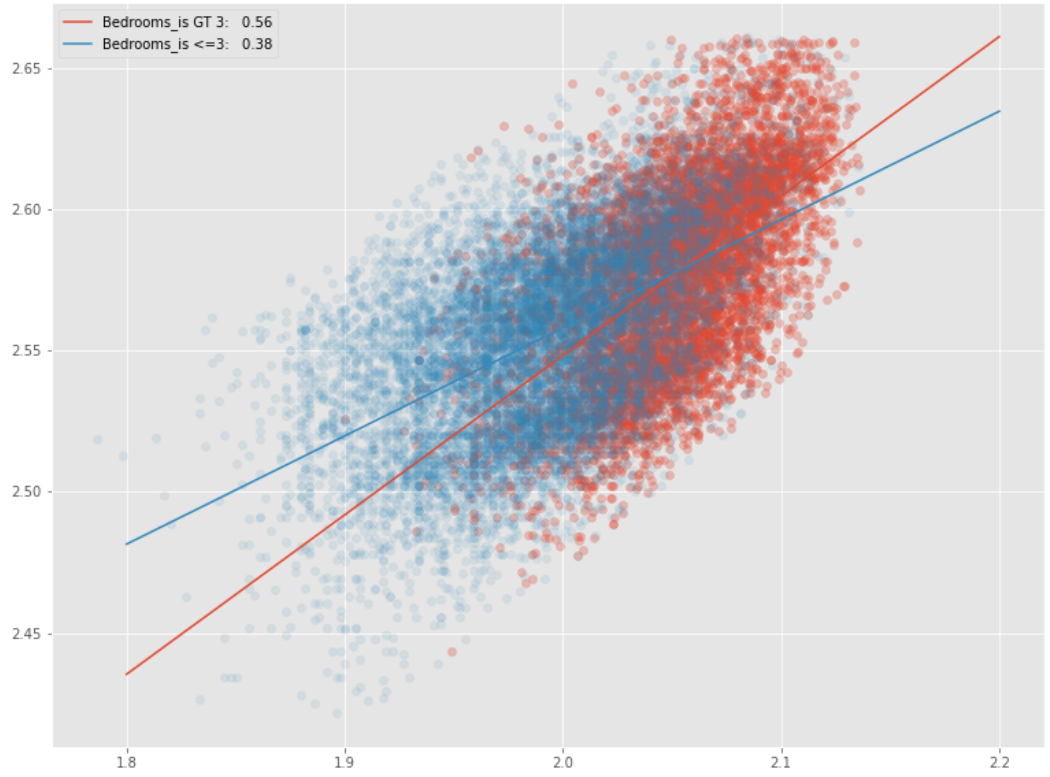
```
# Visualization code here
fig, ax = plt.subplots(figsize=(13, 10))

col = 'log_sqft_living'
x = np.linspace(1.8, 2.2)
linreg = LinearRegression()

is_in = hddf_9.loc[hddf_9['bedrooms'] > 3, [col, 'log_

linreg.fit(np.log(is_in[[col]]), np.log(is_in['log_pri
preds = linreg.predict(np.log(is_in[[col]]))
ax.scatter(np.log(is_in[[col]]), np.log(is_in['log_pri
ax.plot(x, linreg.predict(x.reshape(-1, 1)), label=f'B


not_in = hddf_9.loc[hddf_9['bedrooms'] <= 3 , [col, 'l
linreg.fit(np.log(not_in[[col]]), np.log(not_in['log_p
preds = linreg.predict(np.log(not_in[[col]]))
ax.scatter(np.log(not_in[[col]]), np.log(not_in['log_p
ax.plot(x, linreg.predict(x.reshape(-1, 1)), label=f'B


ax.legend()
```

executed in 2.07s, finished 09:47:17 2021-06-07

Out[549]:

```
<matplotlib.legend.Legend at 0x173
```

```
3cefedc0>
```



In [ ]:

executed in 396ms, finished 15:01:11 2021-05-25

In [ ]:

executed in 1.25s, finished 15:02:25 2021-05-25

5 Interpret

In [550]:

```
#Table of models
```

executed in 5ms, finished 09:47:17 2021-06-07

In [551]:

```
models = ['baseline_model']
for i in range(2,11):
    models.append( 'model_'+str(i))

for myobject in models:
    # add myobject to list called locus
    # e.g. if locus == "A", add to list "A", if locus
    obj = eval(myobject)
    print('Model {} = {}'.format(myobject, obj.rsquare
```

executed in 438ms, finished 09:47:17 2021-06-07

```
Model baseline_model = 0.699977210597367
2
Model model_2 = 0.67938473199336
Model model_3 = 0.6866934704241491
Model model_4 = 0.7352428407389584
Model model_5 = 0.7356419838261794
Model model_6 = 0.7739465509194348
Model model_7 = 0.8579623436702708
Model model_8 = 0.7890862312622597
Model model_9 = 0.7890862312622597
Model model_10 = 0.8033609842873082
```

In [552]:

```
final.columns
```

executed in 62ms, finished 09:47:17 2021-06-07

Out[552]:

```
Index(['bedrooms', 'bathrooms', 'age',
'view_1', 'view_2', 'log_sqft_living',
       'log_sqft_lot15', 'distance', 'gr
ade_6_8', 'grade_8_13', 'd_anerural_1',
       'd_bellvue_7', 'd_dtseattle_10',
'd_mercer_6', 'd_nseattle_9',
       'd_redmond_8', 'd_renton_5', 'd_s
erural_2', 'd_vi_4',
       'distance*d_nseattle_9', 'log_pri
ce'],
      dtype='object')
```

We performed a multiple linear regression in iterative manner in order to determiine significant factors of determining sales price of homes in King's County, WA. A significant model was found to explain 80% of the variance in the data we analzed of the 19000+ rows of home sales data.

The log(**price**) is equal to 8.1479
-0.02722363 * (**bedrooms**) + .04637777 * (**bathrooms**)
+ 0.00074775 * (**age**) + 0.14952887 * (**view_1**)

+ 0.13703141 * (**view_2**) + 0.54902297 * log(**sqft_living**)
+ 0.06060631 * log(**sqft_living15**)
- 0.02257943 * (**distance**) + 0.09696137 * (**grade_6_8**)
+ 0.2341256 * (**grade_8_13**)
+ 0.20976453 * (**d_anerural_1**) + 0.40560228 * (**d_bellvue_7**)
+ 0.21335096 * (**d_dtseattle_10**) + 0.62208309 * (**d_mercer_6**)
+ 0.67828833 * (**d_nseattle_9**) + 0.25678864 * (**d_redmond_8**)
- 0.04204509 * (**d_renton_5**)
+ 0.09428725 * (**d_serural_2**) + 0.18774062 * (**d_vi_4**)
-0.05456986 * (**distance * d_nseattle_9**)

Where view_1 and view_2 is a rating of view from the home
Distance is the distance from minimum distance from one of the 4 major employers
grade_6_8 and grade_8_13 is a condition of the home on the Kings County grading system
and the district information of the home
Interesting to not we found an interaction between the distance of a home
from a major economic center and the existance of a home in the n seattle district.

Example B: Outcome transformed

log(DV) = Intercept + B1 * IV + Error "One unit increase in IV is associated with a (B1 * 100) percent increase in DV."

Example D: Outcome transformed and exposure transformed

log(DV) = Intercept + B1 * log(IV) + Error "One percent increase in IV is associated with a (B1) percent increase in DV."

The most significant factors in the model are the existance of the home
in the **n_seattle_district** where a existance is an increase by 67% increase in price as compared to a home in the fedway district
or existance of home on **mercer island** is a 62% in price
and a house in very good condition,
or improving a home to grade_8_13, increase the outcome by 23% in price or improving a home to grade_6_8, increase the outcome by 9.6% in price.

and a one percent increase in sqft_living is associated with a .55% increase in price. ex.

For example, lets look at all of the sales data for homes in the N Seattle district whose grade level is poor, not in grade 6-8 or not in grade 8-13.
This lets us focus on 21 distinct properties. We can use out model to predict the price outcome for these(stored in bad condition data frame). Now lets set the grade for these homes to grade_8-13, ie

what if we remodeled the property and improved the rating. We can compare the results of price for both. This lets us see an average precidted price of 80712 and std 15785, min 45396, max 125719 of the 21 properties wth an average age of 84 years. This is a business opportunity assuming we could renovate the houses for less than the potential increase in price.

In [553]:

```
bad_condition_df = final.loc[(final["grade_6_8"] == 0)
```

executed in 296ms, finished 09:47:17 2021-06-07

In [554]:

```
bad_condition_df
```

executed in 395ms, finished 09:47:18 2021-06-07

Out[554]:

	bedrooms	bathrooms	age	view_1	view_
1228	1	1.00	76	0	
1825	2	1.50	109	0	
2969	2	1.00	67	0	
4169	2	1.00	91	0	
5012	2	1.00	66	0	
5013	2	1.00	66	0	
5269	1	1.00	97	0	
6290	2	1.00	114	0	
7904	1	1.00	70	0	
8027	3	1.75	88	0	
8575	2	1.00	109	0	
8921	2	1.00	80	0	
8948	1	1.00	61	0	
9864	2	1.00	73	0	

	bedrooms	bathrooms	age	view_1	view_
11223	1	1.00	74	0	
11361	2	1.00	98	0	
11473	2	1.00	110	0	
11545	1	1.00	69	0	
12523	2	1.00	73	0	
12624	1	1.00	99	0	
12984	1	1.00	54	0	
13494	2	1.00	110	0	
14417	2	1.00	95	0	
14928	2	1.00	64	0	
16714	3	2.00	79	0	
17458	2	1.00	90	1	

26 rows × 21 columns

In [555]:

```
goodcond = bad_condition_df.replace({'grade_8_13': {0:
```

executed in 160ms, finished 09:47:18 2021-06-07

In [556]:

```
goodcond
```

executed in 158ms, finished 09:47:18 2021-06-07

Out[556]:

	bedrooms	bathrooms	age	view_1
1228	1	1.00	76	0
1825	2	1.50	109	0
2969	2	1.00	67	0
4169	2	1.00	91	0
5012	2	1.00	66	0
5013	2	1.00	66	0
5269	1	1.00	97	0
6290	2	1.00	114	0
7904	1	1.00	70	0
8027	3	1.75	88	0
8575	2	1.00	109	0
8921	2	1.00	80	0
8948	1	1.00	61	0
9864	2	1.00	73	0

	bedrooms	bathrooms	age	view_1
11223	1	1.00	74	0
11361	2	1.00	98	0
11473	2	1.00	110	0
11545	1	1.00	69	0
12523	2	1.00	73	0
12624	1	1.00	99	0
12984	1	1.00	54	0
13494	2	1.00	110	0
14417	2	1.00	95	0
14928	2	1.00	64	0
16714	3	2.00	79	0
17458	2	1.00	90	1

26 rows × 21 columns

In [557]:

```
np.exp(12.539684)
```

executed in 100ms, finished 09:47:18 2021-06-07

Out[557]:

```
279200.097756214
```

In [558]:

```
y_pred = model_10_linreg.predict(bad_condition_df.drop
y_pred2 = model_10_linreg.predict(goodcond.drop("log_p
```

executed in 14.5s, finished 09:47:33 2021-06-07

In [559]:

```
compare['age'].mean()
```

executed in 140ms, finished 09:47:33 2021-06-07

Out[559]:

```
83.92307692307692
```

In [560]:

```
(np.exp(y_pred2) -np.exp(y_pred)).max()
```

executed in 258ms, finished 09:47:33 2021-06-07

Out[560]:

```
125719.32135627873
```

In [561]:

```
compare = goodcond
```

executed in 193ms, finished 09:47:33 2021-06-07

In [562]:

```
compare['y_pred2']=y_pred2
```

executed in 184ms, finished 09:47:34 2021-06-07

In [563]:

```
compare['y_pred']=y_pred
```

executed in 211ms, finished 09:47:34 2021-06-07

In [564]:

```
compare['pricediff']=(np.exp(y_pred2) -np.exp(y_pred))
```

executed in 220ms, finished 09:47:34 2021-06-07

In [565]:

```
np.exp(compare['y_pred']).mean()
```

executed in 203ms, finished 09:47:34 2021-06-07

Out[565]:

```
305957.7047786023
```

In [566]:

```
np.exp(y_pred2).mean()
```

executed in 247ms, finished 09:47:34 2021-06-07

Out[566]:

```
386670.3306342849
```

In [567]:

```
compare
```

executed in 302ms, finished 09:47:35 2021-06-07

Out[567]:

	bedrooms	bathrooms	age	view_1	view_
1228	1	1.00	76	0	
1825	2	1.50	109	0	
2969	2	1.00	67	0	
4169	2	1.00	91	0	
5012	2	1.00	66	0	
5013	2	1.00	66	0	
5269	1	1.00	97	0	
6290	2	1.00	114	0	
7904	1	1.00	70	0	
8027	3	1.75	88	0	
8575	2	1.00	109	0	
8921	2	1.00	80	0	
8948	1	1.00	61	0	
9864	2	1.00	73	0	

	bedrooms	bathrooms	age	view_1	view_
11223	1	1.00	74	0	
11361	2	1.00	98	0	
11473	2	1.00	110	0	
11545	1	1.00	69	0	
12523	2	1.00	73	0	
12624	1	1.00	99	0	
12984	1	1.00	54	0	
13494	2	1.00	110	0	
14417	2	1.00	95	0	
14928	2	1.00	64	0	
16714	3	2.00	79	0	
17458	2	1.00	90	1	

26 rows × 24 columns

For example a home in the N seattle district which is in poorer condition could be brought up to a grade 8 to 13 condition increase in value has an average price increase of 80712.00 and std 15785 min 45396 max 125719

In [ ]: