King County House Sales Analysis

Author: Lili Beit

Overview

A large real estate firm in the Seattle area is seeking to maximize prices for home sellers. My task is to use data from previous home sales to predict future prices. The firm aims to cast a wide net and attract clients at all price points from throughout the county.

Business Problem

The real estate firm operates throughout King County, which includes the metropolis of Seattle, as well as suburban and rural areas. Home prices vary greatly between these diverse landscapes, as well as between neighborhoods in Seattle. The firm needs to accurately price a home based on data such as its size, location, and number of bedrooms, in order to get the best sale price for its clients. It needs a model that can generate a good estimate of value for homes in every part of the county.

Data Understanding

To build a model to predict prices, I used data from the King County House Sales dataset, which can be found here:

(https://www.kaggle.com/harlfoxem/housesalesprediction)

This dataset contains information on over 21,000 houses sold in King County between May, 2014 and May, 2015. Although the median sale price is $450,000, the dataset also includes multi-million dollar homes. At the top of the market are about 1,000 properties which sold between \$1.2 million and $7.7 million, so the price data are right-skewed with a few very high outliers.

In addition to sale price, the dataset includes details about the homes, including square footage, lot square footage, number of bedrooms, zip code, and the dates when the houses were built, renovated, and sold. Although the data seem mostly accurate, some values are missing, and many columns have outliers.

Definitions of all column names are below:

Column Names and Descriptions for King County Data Set

- **id** - unique identifier for a house
- **date** - house was sold
- **price** - is prediction target
- **bedrooms** - number of bedrooms
- **bathrooms** - number of bathrooms
- **sqft_living** - footage of the home
- **sqft_lot** - footage of the lot
- **floors** - 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

- **yr_renovated - Year when house was renovated**
- **zipcode - zip**
- **lat - Latitude coordinate**
- **long - Longitude coordinate**
- **sqft_living15 - The square footage of interior housing living space for the nearest 15 neighbors**
- **sqft_lot15 - The square footage of the land lots of the nearest 15 neighbors**

Import Data and Split into Training and Test Sets

In [112]:

```
# import packages

import pandas as pd
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.float_format', lambda x: '%.5f' % x)

import numpy as np

from itertools import combinations

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error

import statsmodels.api as sm
from statsmodels.formula.api import ols

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

In [113]:

```
# import data

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

In [114]:

```
# split data into test and training sets

# choose relevant columns:

X=data.drop(columns=['price'])

y=data['price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42
)

print(len(X_train), len(X_test), len(y_train), len(y_test))
```

```
15117 6480 15117 6480
```

In [115]:

```
# concatenate X_train and y_train back together for initial exploration

train_data = pd.concat([X_train, y_train], axis=1)
train_data
```

Out[115]:

	id	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sq
753	8682300890	8/28/2014	2	2.50000	2380	6600	1.00000	nan	0.00000	3	8	
1418	8073000550	4/15/2015	4	3.75000	3190	17186	2.00000	1.00000	4.00000	3	10	
8178	7212680850	9/3/2014	3	2.50000	1730	6930	2.00000	0.00000	0.00000	3	8	
2254	8880600070	11/12/2014	4	2.00000	1870	8750	1.00000	0.00000	2.00000	3	7	
4063	7226500100	2/19/2015	8	3.00000	2850	12714	1.00000	nan	0.00000	3	7	
...	...	...	...	...	...	...	...	...	...	...	...	
11964	7853230570	9/15/2014	3	2.50000	2230	5800	2.00000	0.00000	0.00000	3	7	
21575	4140940150	10/2/2014	4	2.75000	2770	3852	2.00000	0.00000	0.00000	3	8	
5390	8658300480	7/21/2014	4	1.50000	1530	9000	1.00000	0.00000	0.00000	4	6	
860	1723049033	6/20/2014	1	0.75000	380	15000	1.00000	0.00000	0.00000	3	5	
15795	8567450080	3/25/2015	4	2.50000	2755	11612	2.00000	0.00000	0.00000	3	8	

15117 rows × 21 columns

Data Cleaning and Preprocessing

Right-Skewed Columns and Outliers

While exploring the data, I found several columns that are right skewed, including:

- price
- bedrooms
- bathrooms
- sqft_living
- sqft_lot
- sqft_above
- sqft_living15
- sqft_lot15

In this section, I investigated the right-skewed columns. I found that although the data do not appear inaccurate, many columns have high outliers. I removed the highest outliers in price, square footage, and lot square footage from the data to improve the model's accuracy for the remaining homes. Later, I will use log transformation on some columns to reduce the effect of the skewness.

In [116]:

```
# explore training data

train_data.head(500)
train_data.info()
train_data.describe()

# questions and observations:
# high outliers in price, bedrooms, bathrooms, sqft_living, sqft_lot
# null values in waterfront, view, yr_renovated
# waterfront has lots of 0s in addition to null values
# need to turn date (sale date) into a date
# need to turn sqft_basement into a float (but it has some non-number values, like ?)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15117 entries, 753 to 15795
Data columns (total 21 columns):
id               15117 non-null int64
date             15117 non-null object
bedrooms         15117 non-null int64
bathrooms        15117 non-null float64
sqft_living      15117 non-null int64
sqft_lot         15117 non-null int64
floors           15117 non-null float64
waterfront       13467 non-null float64
view             15073 non-null float64
condition        15117 non-null int64
grade            15117 non-null int64
sqft_above       15117 non-null int64
sqft_basement    15117 non-null object
yr_built         15117 non-null int64
yr_renovated     12418 non-null float64
zipcode          15117 non-null int64
lat              15117 non-null float64
long             15117 non-null float64
sqft_living15    15117 non-null int64
sqft_lot15       15117 non-null int64
price            15117 non-null float64
dtypes: float64(8), int64(11), object(2)
memory usage: 2.5+ MB
```

Out[116]:

	id	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	c
count	15117.00000	15117.00000	15117.00000	15117.00000	15117.00000	15117.00000	13467.00000	15073.00000	1511
mean	4595180775.49031	3.37600	2.11995	2087.04062	15169.37832	1.49636	0.00765	0.23287	
std	2889110228.57206	0.90917	0.77023	922.64361	41063.71788	0.54095	0.08712	0.76726	
min	1000102.00000	1.00000	0.50000	370.00000	520.00000	1.00000	0.00000	0.00000	
25%	2115720130.00000	3.00000	1.75000	1430.00000	5070.00000	1.00000	0.00000	0.00000	
50%	3905081500.00000	3.00000	2.25000	1912.00000	7623.00000	1.50000	0.00000	0.00000	
75%	7340500270.00000	4.00000	2.50000	2560.00000	10754.00000	2.00000	0.00000	0.00000	
max	9900000190.00000	11.00000	8.00000	13540.00000	1651359.00000	3.50000	1.00000	4.00000	

In [117]:

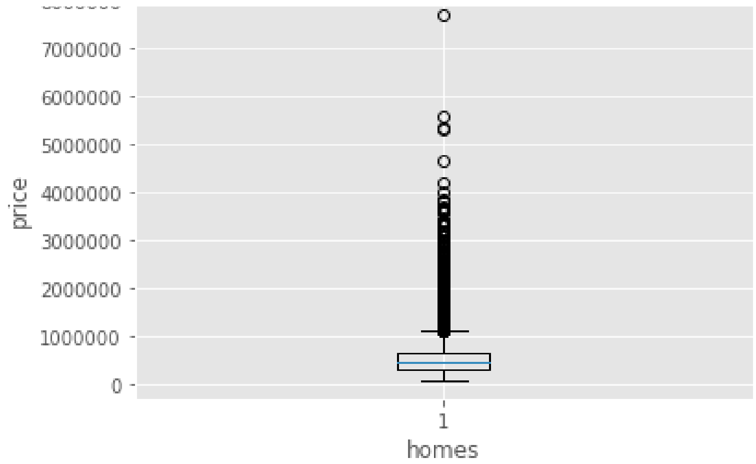
```
# check data set time frame

pd.to_datetime(train_data['date']).describe()

# homes sold between May 2014 and May 2015
```

Out[117]:

```
count                   15117
unique                    365
top       2015-04-27 00:00:00
freq                       97
first     2014-05-02 00:00:00
last      2015-05-27 00:00:00
Name: date, dtype: object
```

In [118]:

```
# investigate price outliers
plt.boxplot(train_data['price'])
plt.xlabel('homes')
plt.ylabel('price'); # looks like outliers are probably accurate, but may decrease the mo
del's efficacy
```



In [119]:

```
# find 95th percentile of price data
train_data['price'].quantile(0.95)
```

Out[119]:

```
1170000.0
```

In [120]:

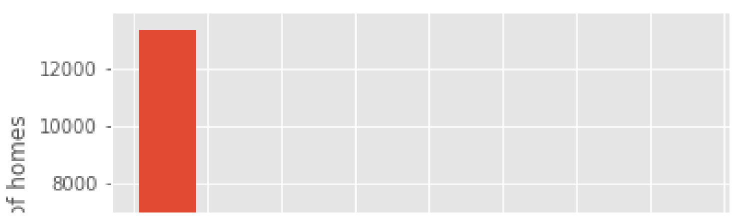
```
# look at highest price outliers

data_price_outliers = train_data.loc[data.price >= 1170000].sort_values(by='price', asce
nding=False)
data_price_outliers.head(500)
data_price_outliers.describe()
```

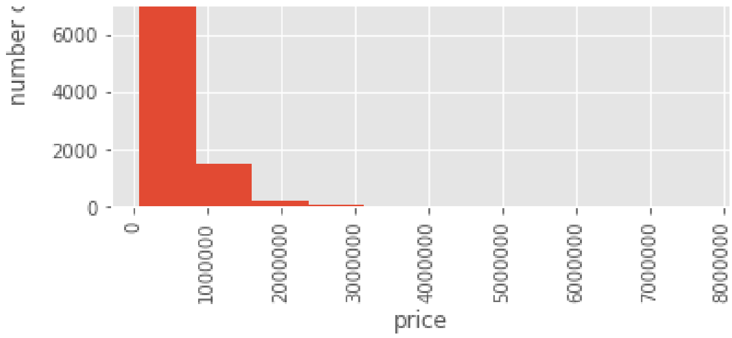
Out[120]:

	id	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	
count	760.00000	760.00000	760.00000	760.00000	760.00000	760.00000	688.00000	757.00000	760.00000	760
mean	4205659171.97500	4.13158	3.28092	3954.68421	22092.81316	1.82961	0.10174	1.39894	3.49211	9
std	2817033846.28464	0.91023	0.88811	1235.60161	50436.82774	0.49793	0.30253	1.58678	0.72409	1
min	46100204.00000	1.00000	1.00000	1560.00000	1620.00000	1.00000	0.00000	0.00000	2.00000	6
25%	1727850347.50000	4.00000	2.50000	3127.50000	7200.00000	1.50000	0.00000	0.00000	3.00000	9
50%	3629915175.00000	4.00000	3.25000	3780.00000	11452.00000	2.00000	0.00000	0.00000	3.00000	10
75%	6602500310.50000	5.00000	3.75000	4560.00000	18902.50000	2.00000	0.00000	3.00000	4.00000	11
max	9831200520.00000	9.00000	8.00000	13540.00000	881654.00000	3.50000	1.00000	4.00000	5.00000	13

In [121]:

```
train_data['price'].hist(bins = 10)
plt.xticks(rotation = 'vertical')
plt.xlabel('price')
plt.ylabel('number of homes');
```





In both the box plot and the histogram above, prices look extremely unusual above $3 million. There are 36 homes at or above this sale price in the training data. I will remove these to improve the model later.

In [122]:

```
# how many homes sold at or above $3 million?

train_data['price'].loc[train_data['price'] >= 3000000].count()  #36 homes
```

Out[122]:

```
36
```

In [123]:

```
# drop rows with price > $3 million for training data

train_data = train_data.loc[train_data['price'] < 3000000]
len(train_data)
```

Out[123]:

```
15081
```

In [124]:

```
# make the same change to the test data

test_data = pd.concat([X_test, y_test], axis=1)
test_data['price'].loc[test_data['price'] >= 3000000].count()  #15 homes
test_data = test_data.loc[test_data['price'] < 3000000]
len(test_data)
```

Out[124]:

```
6465
```

In [125]:

```
# looking at row detail, max values for bedrooms, bathrooms, sqft_living & sqft_above see
m plausible
# but for sqft_living and sqft_above, there is one home with a huge outlier
# same with sqft_lot

# outliers = train_data.sort_values(by='bedrooms', ascending=False).head(500)
# outliers = train_data.sort_values(by='bathrooms', ascending=False).head(500)
# outliers = train_data.sort_values(by='sqft_living', ascending=False).head(500)
# outliers = train_data.sort_values(by='sqft_above', ascending=False).head(500)
outliers = train_data.sort_values(by='sqft_lot', ascending=False).head(500)
outliers
```

Out[125]:

	id	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sc
1717	1020069017	3/27/2015	4	1.00000	1300	1651359	1.00000	0.00000	3.00000	4	6	
7640	2623069031	5/21/2014	5	3.25000	3010	1074218	1.50000	nan	0.00000	5	8	

	id	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sc
7762	2323089009	1/19/2015	4	3.50000	4030	1024068	2.00000	0.00000	0.00000	3	10	
3945	722069232	9/5/2014	4	3.25000	3770	982998	2.00000	0.00000	0.00000	3	10	
4437	3626079040	7/30/2014	2	3.00000	2560	982278	1.00000	0.00000	0.00000	3	8	
7070	2724079090	1/5/2015	4	3.25000	3920	881654	3.00000	nan	3.00000	3	11	
9705	225079036	1/7/2015	4	4.00000	5545	871200	2.00000	0.00000	0.00000	3	11	
4536	2522029039	9/29/2014	3	2.00000	3650	843309	2.00000	0.00000	0.00000	4	7	
7287	1923039022	11/20/2014	2	1.75000	1679	577605	2.00000	0.00000	0.00000	3	9	
2962	2322029048	11/19/2014	3	2.75000	2830	505166	1.00000	1.00000	3.00000	4	8	
20405	1623089165	5/6/2015	4	3.75000	4030	503989	2.00000	0.00000	0.00000	3	10	
17335	2825079001	8/14/2014	5	1.75000	1930	501376	2.00000	0.00000	0.00000	3	7	
8436	125069038	11/25/2014	4	3.75000	5150	453895	2.00000	nan	3.00000	3	11	
14674	621069057	3/23/2015	4	3.50000	2700	443440	1.50000	0.00000	0.00000	3	8	
7243	722039049	10/9/2014	4	3.00000	3230	438213	2.00000	0.00000	0.00000	3	9	
13237	2523089025	2/10/2015	3	3.00000	4020	435600	1.50000	0.00000	2.00000	3	10	
13873	3522029124	12/3/2014	3	2.00000	2690	435600	2.00000	0.00000	0.00000	3	8	
18213	820079101	12/22/2014	3	2.25000	2040	435600	2.00000	0.00000	2.00000	4	7	
18827	3624079067	5/8/2014	2	2.00000	1550	435600	1.50000	0.00000	0.00000	2	7	
15920	1823099056	12/22/2014	3	2.50000	2810	435600	2.00000	nan	0.00000	3	9	
5068	2322059136	3/9/2015	3	2.50000	2920	434728	2.00000	0.00000	3.00000	4	8	
12741	620069061	5/7/2015	3	2.50000	2880	426452	2.00000	0.00000	3.00000	3	7	
13617	1920079103	9/11/2014	2	1.75000	1460	426450	1.00000	0.00000	0.00000	5	7	
2869	1820069019	5/29/2014	2	1.00000	900	423838	1.00000	0.00000	2.00000	5	6	
19141	1020069042	10/1/2014	4	3.50000	4370	422967	1.00000	0.00000	2.00000	4	10	
19878	1422069070	5/7/2015	3	2.50000	1860	415126	2.00000	0.00000	0.00000	3	7	
5223	2124079093	1/12/2015	2	3.25000	3570	392475	1.00000	0.00000	0.00000	3	9	
2755	3520069033	6/23/2014	3	1.00000	1530	389126	1.50000	0.00000	0.00000	4	7	
4107	522079067	4/8/2015	3	2.50000	3310	387684	1.00000	nan	0.00000	3	8	
16908	2026079016	9/4/2014	3	1.75000	1480	383328	1.50000	0.00000	0.00000	3	8	
[illegible]	[illegible]	11/24/2014	[illegible]	1.00000	[illegible]	[illegible]	1.00000	0.00000	0.00000	[illegible]	7	

	id	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sc
13817	123079023	11/24/2014	2	1.00000	1430	365904	1.00000	0.00000	0.00000	3	7	
12375	1226069045	8/27/2014	4	3.75000	4133	361548	2.00000	0.00000	0.00000	3	11	
1701	3121069036	12/8/2014	3	1.75000	3020	360241	2.00000	0.00000	nan	3	8	
14016	1520069052	7/21/2014	3	1.50000	1510	344124	1.00000	0.00000	2.00000	4	7	
9066	2124079010	10/28/2014	3	2.25000	3190	324086	2.00000	0.00000	2.00000	3	9	
10514	624069050	4/7/2015	4	3.50000	5370	323215	2.00000	0.00000	0.00000	3	10	
10307	2323089065	12/17/2014	4	2.75000	4600	322188	1.00000	0.00000	4.00000	3	10	
15751	1622069127	11/18/2014	5	3.25000	3960	321908	2.00000	0.00000	0.00000	4	9	
18273	1524079188	7/29/2014	4	5.25000	5240	320917	2.00000	nan	2.00000	3	10	
145	1526069017	12/3/2014	4	2.50000	3670	315374	2.00000	0.00000	0.00000	4	9	
12764	1225069038	5/5/2014	7	8.00000	13540	307752	3.00000	0.00000	4.00000	3	12	
1772	1549500370	5/5/2014	3	1.00000	1340	306848	1.00000	nan	0.00000	3	5	
8597	2422059015	8/8/2014	2	1.00000	910	295772	1.00000	0.00000	0.00000	3	5	
19372	1223089066	8/14/2014	4	3.00000	3400	292723	2.00000	0.00000	0.00000	3	10	
3452	3026059085	3/17/2015	5	3.50000	4090	290980	1.00000	0.00000	0.00000	3	11	
14938	322059049	10/3/2014	2	1.00000	820	288367	1.00000	nan	0.00000	3	6	
12937	120059044	2/17/2015	3	1.75000	1628	286355	1.00000	0.00000	0.00000	3	7	
8201	521079025	4/17/2015	3	2.50000	3160	286181	2.00000	0.00000	3.00000	3	9	
20483	1623089086	10/15/2014	4	2.75000	3980	285318	2.00000	0.00000	2.00000	3	9	
12448	2025079045	6/23/2014	2	1.75000	2260	280962	2.00000	0.00000	2.00000	3	9	
21335	3421069049	10/21/2014	2	1.75000	1130	276170	1.00000	0.00000	0.00000	3	8	
20958	8835800450	5/4/2015	3	2.50000	2780	275033	1.00000	0.00000	0.00000	3	10	
3520	1322059002	3/19/2015	3	1.75000	1980	273556	1.00000	0.00000	0.00000	3	6	
12112	3123039082	10/6/2014	3	1.75000	2040	273556	1.00000	0.00000	0.00000	3	7	
20251	8835800010	12/23/2014	4	4.50000	4920	270236	2.00000	0.00000	3.00000	3	10	
8619	823069044	3/25/2015	5	4.00000	4460	269345	2.00000	nan	4.00000	3	9	
4389	1221059176	3/11/2015	4	2.75000	2200	268329	1.00000	0.00000	0.00000	3	7	
8655	3226079059	10/19/2014	3	1.75000	2930	266587	2.00000	0.00000	0.00000	3	8	
10635	1025079074	12/2/2014	3	2.00000	2350	266151	1.50000	0.00000	0.00000	3	7	

	id	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sc
9090	1525069058	6/26/2014	4	1.75000	2110	265716	1.00000	0.00000	0.00000	4	8	
3758	2523089097	10/29/2014	3	1.50000	3430	264844	1.00000	0.00000	2.00000	3	7	
7826	3125079013	4/30/2015	3	2.50000	3970	263538	1.50000	0.00000	0.00000	3	9	
4395	524069019	11/20/2014	4	3.25000	4400	262666	2.00000	0.00000	0.00000	3	11	
13670	2024089011	8/26/2014	5	1.00000	2150	262231	1.50000	0.00000	0.00000	3	7	
7334	526069024	5/12/2014	5	3.00000	4530	258746	1.50000	0.00000	0.00000	4	9	
12770	2023069054	3/18/2015	3	1.75000	1160	257875	1.00000	0.00000	0.00000	2	7	
7063	1425069071	3/23/2015	4	2.50000	3230	256132	2.00000	0.00000	0.00000	3	9	
14168	1623069023	7/29/2014	4	2.50000	2920	252648	2.00000	0.00000	0.00000	3	10	
13027	822039025	5/1/2015	3	2.50000	2260	251460	1.50000	nan	0.00000	3	10	
19446	1626079154	5/20/2014	3	2.00000	2010	251341	2.00000	0.00000	0.00000	3	8	
20370	1222029064	6/26/2014	3	1.75000	1444	249126	1.50000	0.00000	0.00000	3	7	
2409	3020079078	10/27/2014	6	3.25000	4750	248600	2.00000	nan	0.00000	4	8	
17400	1825079005	6/9/2014	4	2.50000	2800	246114	2.00000	0.00000	0.00000	3	9	
12117	3022079094	10/6/2014	4	2.50000	3320	244807	2.00000	0.00000	0.00000	3	9	
11673	1126069045	6/20/2014	6	4.25000	6900	244716	2.00000	0.00000	0.00000	4	9	
1044	1825079070	3/13/2015	3	1.75000	1560	242629	1.00000	0.00000	0.00000	3	7	
3123	1926069137	7/7/2014	4	3.25000	4100	241322	2.00000	0.00000	0.00000	3	9	
6122	2623069069	9/11/2014	3	2.50000	2620	241200	1.50000	nan	0.00000	4	9	
6622	3322049005	9/30/2014	4	2.75000	5440	239580	1.00000	0.00000	0.00000	2	9	
9904	3323069045	11/10/2014	3	1.00000	1240	239144	1.00000	0.00000	0.00000	3	6	
4077	3321069006	12/31/2014	3	2.50000	3520	237402	2.50000	0.00000	0.00000	3	9	
3480	925069111	5/7/2015	3	1.75000	1760	235224	1.00000	0.00000	0.00000	3	7	
18787	3621059043	5/27/2014	4	2.50000	3250	235063	1.00000	0.00000	2.00000	3	9	
17663	2724079014	3/31/2015	3	3.25000	2970	234788	2.00000	0.00000	3.00000	3	9	
7052	323069120	8/27/2014	4	2.75000	3640	231739	1.50000	0.00000	0.00000	3	10	
19592	1026069106	4/21/2015	3	2.25000	1790	231303	1.00000	0.00000	0.00000	3	7	
16759	1630700380	1/30/2015	5	5.75000	7730	230868	2.00000	nan	0.00000	3	12	

	id	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sc
8528	326069131	6/11/2014	4	2.50000	2790	230868	2.00000	0.00000	0.00000	3	8	
527	3225079035	6/18/2014	6	5.00000	6050	230652	2.00000	nan	3.00000	3	11	
8910	1120069059	9/18/2014	3	1.50000	1790	229125	2.00000	0.00000	3.00000	3	7	
4549	2126079014	5/12/2014	4	2.25000	2540	228254	1.00000	0.00000	0.00000	3	8	
15940	3223039109	2/20/2015	3	2.50000	2750	226512	2.00000	0.00000	0.00000	3	9	
15876	1525069088	5/4/2015	5	3.25000	4240	226097	2.00000	0.00000	0.00000	3	8	
10785	3223069065	9/17/2014	2	1.75000	1800	224769	1.00000	0.00000	0.00000	3	7	
14295	3528000040	3/26/2015	3	3.25000	5290	224442	2.00000	0.00000	0.00000	4	11	
4865	3421059049	6/10/2014	2	1.75000	1490	224334	1.00000	0.00000	2.00000	3	8	
13313	2120069003	11/24/2014	3	1.00000	1000	223462	1.00000	0.00000	2.00000	4	6	
3771	3022059066	1/30/2015	4	2.50000	2960	223462	2.00000	0.00000	0.00000	3	10	
6495	2626069030	2/9/2015	4	5.75000	7220	223462	2.00000	0.00000	4.00000	3	12	
2570	1626079012	2/25/2015	3	1.75000	1720	223377	1.00000	0.00000	0.00000	3	7	
13203	822069029	2/17/2015	3	2.75000	2660	223027	1.00000	0.00000	0.00000	3	8	
12209	2824089053	1/27/2015	3	2.00000	2250	222156	1.00000	0.00000	0.00000	3	7	
9230	3022079080	7/15/2014	4	2.50000	3420	222156	2.00000	0.00000	0.00000	3	9	
13592	2724079061	10/10/2014	3	1.75000	1650	221720	1.00000	0.00000	0.00000	3	7	
238	326069104	7/1/2014	3	3.50000	3830	221284	2.00000	0.00000	0.00000	3	10	
12724	123059042	4/23/2015	3	2.25000	2190	220414	1.00000	nan	0.00000	4	7	
1322	3323069084	9/9/2014	4	2.50000	1840	220308	2.00000	0.00000	0.00000	3	8	
13155	2723069052	4/20/2015	3	2.25000	2600	220300	1.50000	0.00000	0.00000	5	8	
443	822079033	4/22/2015	3	1.50000	1250	219978	1.00000	0.00000	0.00000	4	6	
6861	525069099	10/22/2014	3	2.50000	2320	219978	2.00000	0.00000	0.00000	4	8	
2511	2024079035	6/5/2014	3	2.75000	3150	219978	2.00000	0.00000	0.00000	4	9	
15208	1424069069	5/22/2014	6	4.50000	6040	219542	2.00000	0.00000	0.00000	3	11	
19262	822069066	2/23/2015	4	2.50000	1620	219542	2.00000	0.00000	0.00000	3	7	
11496	2025079037	10/1/2014	3	2.25000	2750	219542	2.00000	0.00000	0.00000	3	7	
7545	322069020	6/19/2014	3	1.75000	1940	219527	1.00000	0.00000	0.00000	3	7	
18106	2924079044	7/23/2014	3	3.75000	3830	219106	2.00000	nan	0.00000	3	9	

	id	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sc
19578	1321059013	3/19/2015	4	2.50000	3750	218506	2.00000	0.00000	0.00000	3	10	
5717	1526079026	8/13/2014	5	3.50000	3530	218472	2.00000	0.00000	0.00000	3	7	
762	826079094	3/24/2015	3	2.00000	1400	218252	1.00000	0.00000	0.00000	3	7	
2986	825079019	12/3/2014	3	2.50000	3360	218235	1.00000	0.00000	0.00000	3	8	
6479	2324079073	8/15/2014	3	2.75000	2930	218235	2.00000	0.00000	2.00000	3	8	
11979	1330910370	10/20/2014	4	3.00000	4370	217882	2.00000	0.00000	0.00000	3	10	
14335	3421069020	10/20/2014	3	1.75000	1350	217852	1.00000	0.00000	0.00000	3	8	
6587	1125069102	4/27/2015	4	3.00000	3310	217800	1.50000	0.00000	0.00000	3	9	
2353	1425069116	11/7/2014	4	3.50000	4340	217800	2.00000	0.00000	0.00000	3	11	
5845	326069026	1/21/2015	4	3.00000	3810	217800	2.00000	0.00000	0.00000	3	9	
11918	723069128	5/14/2014	2	2.00000	2370	217800	1.50000	0.00000	0.00000	3	7	
7274	123059071	7/8/2014	3	2.00000	1860	217800	2.00000	0.00000	2.00000	3	8	
18579	3023069166	7/8/2014	5	4.00000	7320	217800	2.00000	0.00000	0.00000	3	11	
4937	1121059030	10/13/2014	3	2.50000	3110	217800	2.00000	0.00000	0.00000	3	9	
590	2525069041	9/4/2014	3	1.50000	1830	217800	1.00000	0.00000	nan	3	7	
18187	522079015	3/22/2015	3	2.00000	2400	217800	2.00000	0.00000	0.00000	3	8	
15908	2425069069	5/27/2014	3	2.25000	2370	217800	2.00000	0.00000	0.00000	3	7	
13208	820079081	9/11/2014	4	3.00000	2710	217800	2.50000	0.00000	0.00000	3	9	
1007	1624079104	4/2/2015	3	2.25000	2000	217800	2.00000	0.00000	0.00000	3	8	
14309	2222039011	11/3/2014	5	1.75000	2080	217800	1.00000	0.00000	0.00000	5	7	
15246	3521059134	5/23/2014	3	3.50000	4080	217697	1.50000	0.00000	3.00000	3	10	
3098	622069006	8/20/2014	4	5.50000	6550	217374	1.00000	0.00000	0.00000	3	11	
561	1921069084	7/7/2014	4	2.25000	2340	217014	1.00000	0.00000	0.00000	4	8	
4583	725079058	8/11/2014	3	1.75000	2220	216493	1.00000	0.00000	2.00000	3	8	
12800	2726079098	9/18/2014	3	2.50000	2840	216493	2.00000	0.00000	0.00000	3	9	
17786	3521059124	9/24/2014	2	2.50000	2550	216344	2.50000	nan	0.00000	3	7	
2510	2126079046	4/7/2015	3	1.75000	1220	216332	1.00000	nan	0.00000	3	7	
19166	826079047	8/14/2014	3	2.25000	2990	216057	2.00000	0.00000	0.00000	3	9	

	id	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sc
7804	[illegible]	5/8/2015	3	1.75000	[illegible]	[illegible]	2.00000	0.00000	0.00000	4	7	
20904	2124069115	10/21/2014	4	4.25000	4500	215186	2.00000	0.00000	3.00000	3	11	
6213	2421059090	5/11/2015	4	2.50000	4090	215186	2.00000	0.00000	0.00000	4	8	
18745	1525069021	12/1/2014	3	2.50000	2580	214315	1.50000	0.00000	0.00000	3	8	
9631	2623089002	4/16/2015	3	2.50000	2380	214315	1.50000	nan	0.00000	3	9	
858	623069068	6/27/2014	3	1.00000	1520	213444	1.50000	0.00000	3.00000	5	8	
20187	821079102	10/17/2014	4	3.50000	3720	213073	1.00000	0.00000	2.00000	3	10	
19219	1324079029	3/17/2015	3	1.00000	960	213008	1.00000	0.00000	0.00000	2	6	
9077	1222069133	2/24/2015	4	2.50000	2210	213008	1.00000	0.00000	0.00000	4	7	
16500	1222069136	12/12/2014	4	2.75000	3000	213008	1.00000	0.00000	0.00000	4	8	
17055	225069016	7/22/2014	3	1.75000	1930	213008	1.00000	0.00000	2.00000	3	7	
3759	3026079055	8/26/2014	4	2.75000	3470	212639	2.00000	0.00000	0.00000	3	7	
13476	1426079047	9/11/2014	3	2.25000	2520	212137	2.00000	nan	0.00000	3	9	
18706	2624079010	4/29/2015	5	3.50000	2990	212137	2.00000	0.00000	0.00000	3	8	
3040	2326079039	2/11/2015	1	1.00000	890	211576	1.50000	0.00000	0.00000	3	7	
14876	1720069075	5/8/2015	3	3.00000	2450	211266	1.50000	nan	3.00000	3	8	
12108	1822069041	11/13/2014	6	2.00000	2320	210830	2.00000	nan	0.00000	4	8	
15995	1923099034	1/16/2015	4	3.50000	3970	210830	2.00000	0.00000	0.00000	3	9	
8290	1923099058	10/15/2014	4	2.50000	2980	210395	2.00000	0.00000	0.00000	3	9	
17582	1824079052	4/1/2015	4	3.25000	4200	210394	2.00000	0.00000	0.00000	4	10	
13567	2622029072	10/1/2014	4	3.50000	2734	210201	2.00000	0.00000	0.00000	5	8	
13452	1822069052	7/9/2014	5	2.50000	2850	209523	1.00000	0.00000	0.00000	4	7	
16274	1549500585	4/27/2015	3	2.00000	2220	209523	1.00000	0.00000	0.00000	3	7	
10402	2224079050	7/18/2014	4	3.50000	3980	209523	2.00000	0.00000	2.00000	3	9	
681	3526069070	5/28/2014	4	3.00000	2580	209523	2.00000	nan	0.00000	3	8	
8846	1725079025	9/3/2014	3	2.00000	2350	209088	1.00000	0.00000	0.00000	3	7	
3667	8847400115	7/23/2014	3	2.00000	2420	208652	1.50000	0.00000	0.00000	3	8	
10420	3123039171	8/5/2014	3	2.75000	1830	208216	2.00000	0.00000	0.00000	3	8	
4263	2621069066	4/27/2015	3	2.00000	3190	207346	2.00000	0.00000	0.00000	3	9	

	id	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sc
20409	2526069092	8/8/2014	4	3.75000	4690	207141	2.00000	0.00000	0.00000	3	10	
3490	2023069059	10/30/2014	3	3.00000	2840	206910	2.00000	0.00000	0.00000	3	10	
1622	1223089083	10/28/2014	3	2.75000	3010	206910	2.00000	0.00000	2.00000	3	10	
1812	3125079062	4/26/2015	3	2.50000	2660	206480	1.00000	0.00000	0.00000	3	8	
14900	2622029073	9/5/2014	3	2.25000	2100	205603	2.00000	0.00000	0.00000	3	8	
1651	2025079033	12/10/2014	1	2.00000	3000	204732	2.50000	0.00000	2.00000	3	8	
5729	2921079027	9/24/2014	4	2.50000	2170	204296	1.00000	0.00000	0.00000	4	7	
6458	3226079091	9/12/2014	3	2.50000	3680	203860	1.50000	0.00000	0.00000	3	9	
13558	1026069061	1/29/2015	4	2.50000	3600	203425	2.00000	0.00000	0.00000	3	9	
9324	3324079089	11/21/2014	4	4.00000	5050	202554	2.00000	0.00000	0.00000	3	10	
14793	1725079047	11/4/2014	3	2.25000	2280	200811	1.00000	nan	0.00000	3	7	
3395	1022069058	10/9/2014	4	2.00000	2430	199940	1.00000	0.00000	0.00000	3	8	
13285	7167000040	8/13/2014	4	3.00000	3350	199253	2.00000	nan	0.00000	3	10	
13286	7167000040	3/5/2015	4	3.00000	3350	199253	2.00000	0.00000	0.00000	3	10	
7480	326069027	3/26/2015	3	2.50000	2420	198198	2.00000	nan	0.00000	3	9	
4582	3522029031	5/16/2014	3	1.75000	1726	197326	2.00000	0.00000	0.00000	4	7	
7509	822039111	3/27/2015	3	2.50000	2120	196995	1.00000	nan	1.00000	3	9	
12831	3221059044	5/23/2014	4	3.50000	4220	196817	2.00000	0.00000	0.00000	3	10	
519	1923069078	8/5/2014	4	3.25000	3180	194278	2.00000	0.00000	0.00000	3	10	
13142	2521059042	11/7/2014	5	2.75000	2720	193406	1.00000	0.00000	4.00000	4	7	
19049	2320069014	7/9/2014	3	2.00000	2660	192099	1.00000	0.00000	0.00000	4	9	
6238	5703000050	5/8/2014	3	2.25000	1780	191228	2.00000	0.00000	2.00000	3	8	
4094	3407700047	10/29/2014	3	3.25000	2990	189852	2.00000	0.00000	0.00000	4	10	
2604	1926069035	7/22/2014	2	1.00000	1070	189486	1.00000	0.00000	0.00000	3	6	
4374	321059059	5/19/2014	3	1.00000	1290	189486	1.00000	0.00000	0.00000	4	7	
11819	3221069035	6/20/2014	4	1.75000	2670	189486	2.00000	0.00000	4.00000	3	8	
7273	922059169	12/1/2014	6	4.25000	5480	189050	2.00000	0.00000	0.00000	4	10	
15154	8835800480	2/23/2015	1	2.00000	1780	188465	2.00000	0.00000	0.00000	3	10	

	id	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sc
11696	220069083	5/9/2014	2	2.50000	2200	188200	1.00000	0.00000	3.00000	3	8	
7275	1626069069	10/29/2014	3	2.50000	1350	187313	1.00000	0.00000	0.00000	4	7	
10320	3353404510	1/7/2015	2	1.00000	1960	186872	1.50000	0.00000	0.00000	4	6	
15585	619079016	6/2/2014	4	3.25000	4400	186846	2.00000	0.00000	0.00000	4	9	
9290	2723069146	4/24/2015	4	2.50000	3170	186436	2.00000	0.00000	0.00000	3	9	
8607	1122069006	7/10/2014	3	2.00000	2800	185130	1.00000	0.00000	0.00000	3	8	
8514	2726079103	7/22/2014	3	2.50000	2630	185130	2.00000	0.00000	0.00000	3	9	
18874	621069218	2/19/2015	5	2.50000	2670	184140	1.00000	0.00000	0.00000	3	8	
1806	225069017	7/14/2014	4	3.00000	2720	183823	2.00000	0.00000	0.00000	3	8	
15183	6979900010	9/5/2014	4	2.50000	3420	183387	2.00000	nan	0.00000	3	10	
10627	1921069068	4/29/2015	4	2.50000	3030	180263	2.00000	0.00000	0.00000	3	7	
10447	7167000060	11/24/2014	4	2.50000	3410	179419	2.00000	0.00000	0.00000	3	10	
4538	3124089086	10/2/2014	4	1.00000	1730	177657	1.50000	0.00000	0.00000	3	5	
18319	2820069048	5/4/2015	4	2.50000	2480	176418	1.50000	nan	3.00000	5	8	
2887	2723069147	9/2/2014	3	2.25000	2680	175982	1.00000	nan	0.00000	3	9	
416	824079032	6/26/2014	4	1.75000	2085	174240	1.00000	0.00000	0.00000	3	7	
2845	1723099031	10/20/2014	4	3.50000	3010	174240	2.00000	0.00000	0.00000	3	9	
16848	525069127	5/23/2014	4	3.50000	4740	172497	2.00000	0.00000	0.00000	3	11	
419	8678500060	7/10/2014	5	4.25000	6070	171626	2.00000	0.00000	0.00000	3	12	
8010	3223039089	9/29/2014	3	1.00000	1230	171190	1.00000	0.00000	0.00000	3	7	
11168	3345100030	7/22/2014	3	2.25000	3270	168000	2.00000	0.00000	0.00000	4	10	
11859	8835800350	1/12/2015	4	3.25000	7420	167869	2.00000	0.00000	3.00000	3	12	
6089	9206700190	3/6/2015	3	2.50000	3370	167706	1.00000	0.00000	0.00000	3	10	
19098	722079015	10/17/2014	3	2.50000	2080	167270	1.00000	0.00000	0.00000	3	7	
13564	2623029003	12/16/2014	3	1.75000	1940	167125	1.00000	1.00000	1.00000	4	7	
8646	326069118	6/30/2014	4	2.50000	3300	165528	2.00000	0.00000	0.00000	3	8	
14466	1720069029	3/13/2015	3	1.00000	1350	165092	1.00000	0.00000	3.00000	5	6	
20711	522079068	5/6/2015	3	2.50000	2150	161607	2.00000	0.00000	0.00000	3	7	

	id	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sc
19527	1630700276	1/5/2015	2	1.50000	1370	159865	1.00000	0.00000	0.00000	3	7	
18198	7931000053	12/29/2014	4	1.75000	2140	159865	1.00000	0.00000	0.00000	4	7	
5776	2224079001	1/26/2015	3	2.00000	2570	159865	1.00000	0.00000	0.00000	5	7	
21415	2725079018	5/9/2014	4	3.25000	3540	159430	2.00000	0.00000	0.00000	3	9	
1793	2925079012	11/5/2014	4	2.50000	2940	156988	2.00000	0.00000	2.00000	3	9	
11362	2421059036	4/15/2015	3	2.50000	2577	156816	2.00000	0.00000	0.00000	3	8	
14638	3622069103	1/23/2015	4	2.50000	3600	155509	2.00000	0.00000	0.00000	3	9	
2793	1326069050	5/4/2015	2	2.00000	2370	155130	1.00000	0.00000	0.00000	3	7	
15651	2922069134	8/29/2014	3	1.75000	2170	153767	1.00000	0.00000	0.00000	3	7	
21074	1624079024	5/15/2014	3	2.50000	3150	151588	2.00000	0.00000	0.00000	3	9	
5118	425069102	11/26/2014	4	2.75000	3660	150282	2.00000	0.00000	0.00000	3	10	
5276	1126059007	3/23/2015	3	2.25000	2670	150270	2.00000	nan	0.00000	3	9	
2022	224069084	3/25/2015	3	1.00000	1250	150117	1.00000	0.00000	0.00000	3	7	
7928	2726079061	5/7/2014	3	1.75000	2720	149410	1.50000	0.00000	0.00000	3	9	
18811	3221069054	10/28/2014	3	2.50000	4040	147856	2.00000	0.00000	0.00000	3	9	
9819	425079046	7/29/2014	3	2.50000	1778	147823	2.00000	0.00000	0.00000	3	7	
7657	943100220	9/25/2014	3	1.00000	1100	145490	1.50000	0.00000	0.00000	4	6	
16900	1324079007	11/10/2014	3	1.75000	1610	144619	1.00000	0.00000	0.00000	3	7	
19057	322059210	2/3/2015	3	2.50000	2650	144183	1.00000	0.00000	0.00000	3	8	
10939	853600310	8/28/2014	5	4.50000	6085	142725	3.00000	0.00000	0.00000	3	11	
9393	425069020	5/5/2014	4	2.50000	4340	141570	2.50000	0.00000	0.00000	3	11	
13023	2421059009	2/20/2015	3	1.75000	2280	139392	1.00000	0.00000	0.00000	3	8	
3775	2623069067	3/5/2015	3	2.50000	2460	138085	2.00000	0.00000	0.00000	4	9	
18049	9537200037	4/28/2015	4	1.50000	1310	137214	1.50000	0.00000	0.00000	4	7	
15775	4045900020	4/13/2015	2	1.50000	1440	136778	1.00000	0.00000	0.00000	4	8	
17251	2625079030	10/28/2014	3	2.50000	3550	136343	2.00000	0.00000	0.00000	3	10	
13749	1223089077	4/1/2015	3	1.75000	4060	136290	1.00000	0.00000	0.00000	3	8	
1745	1320069179	11/4/2014	3	2.00000	1710	134489	1.00000	nan	2.00000	5	7	
1327	2723069082	4/24/2015	4	2.25000	2510	133729	2.00000	nan	0.00000	4	8	

	id	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sc
5835	1523069215	6/3/2014	3	1.75000	2220	132858	1.00000	0.00000	0.00000	3	7	
7127	320069049	5/14/2014	4	1.50000	1590	131551	1.00000	0.00000	3.00000	4	7	
4962	8678500020	12/13/2014	4	3.50000	5830	131116	2.00000	0.00000	0.00000	3	11	
21470	98300230	4/28/2015	4	4.00000	4620	130208	2.00000	0.00000	0.00000	3	10	
15966	425079001	4/23/2015	3	2.50000	3230	129578	1.00000	0.00000	0.00000	4	8	
12953	3223059123	6/30/2014	4	1.50000	2750	128502	1.00000	0.00000	0.00000	2	7	
16100	8645900080	8/27/2014	3	2.00000	1720	128066	1.00000	0.00000	0.00000	3	7	
2882	2523069192	7/8/2014	4	3.75000	4740	126759	2.00000	0.00000	0.00000	4	10	
19550	2423600100	5/2/2014	4	1.75000	2190	125452	1.00000	0.00000	2.00000	3	9	
14839	1465400120	3/26/2015	3	2.50000	3110	123710	2.00000	0.00000	0.00000	3	8	
9122	1921069059	12/30/2014	1	1.00000	720	123710	1.00000	0.00000	0.00000	4	6	
7357	8647600020	11/11/2014	4	2.50000	3340	123600	2.00000	0.00000	0.00000	3	10	
8072	2125079054	2/24/2015	4	2.75000	2200	122403	1.50000	0.00000	0.00000	3	7	
1370	3521069051	12/23/2014	4	2.25000	2380	122038	2.00000	0.00000	0.00000	4	8	
6929	2323069022	1/15/2015	2	1.00000	1800	119790	1.00000	0.00000	0.00000	4	7	
3797	1550000463	8/26/2014	4	3.50000	3080	118918	2.00000	0.00000	0.00000	3	9	
18165	3024079096	4/14/2015	4	2.50000	2600	118666	1.00000	0.00000	0.00000	3	7	
8549	2523069172	8/4/2014	3	2.50000	3580	118047	1.00000	0.00000	0.00000	3	9	
4939	1136100006	1/27/2015	2	1.50000	1110	118047	1.00000	nan	0.00000	3	7	
4650	114100745	5/6/2014	6	3.00000	3470	117612	1.50000	0.00000	0.00000	3	7	
17712	1422059039	8/28/2014	4	2.75000	3030	117378	1.00000	0.00000	0.00000	4	8	
8903	2223069112	11/12/2014	3	2.25000	2560	117176	1.00000	0.00000	0.00000	4	9	
8602	6398000191	8/27/2014	2	1.50000	1995	115670	1.50000	nan	1.00000	4	8	
19086	5126210360	10/22/2014	4	2.50000	3420	115434	2.00000	0.00000	0.00000	3	9	
3346	1423089055	6/13/2014	4	2.75000	4070	115434	2.00000	nan	0.00000	3	9	
4014	622059031	6/4/2014	4	1.00000	1540	115434	1.50000	0.00000	0.00000	4	7	
1046	723069135	9/15/2014	2	1.75000	2040	114562	1.00000	0.00000	0.00000	4	7	
9560	722059002	9/9/2014	2	2.50000	2110	114127	1.00000	0.00000	0.00000	4	8	

	id	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sc
15012	1323089056	11/10/2014	2	1.75000	1620	113862	1.50000	0.00000	0.00000	3	7	
14253	1324079054	9/22/2014	4	1.50000	1980	113691	1.00000	0.00000	0.00000	3	7	
5603	3024079063	7/1/2014	4	3.25000	4350	112750	1.00000	0.00000	0.00000	3	9	
18279	114100758	10/22/2014	2	1.00000	960	112384	1.00000	0.00000	0.00000	3	7	
18910	2325069054	5/21/2014	2	1.00000	1396	111949	1.00000	0.00000	0.00000	3	7	
3970	1326059085	7/21/2014	3	2.25000	2080	111513	1.50000	0.00000	0.00000	3	8	
132	1243100136	6/12/2014	3	3.50000	3950	111078	1.50000	0.00000	0.00000	3	9	
5054	1125069153	8/22/2014	4	3.50000	5990	111078	2.00000	nan	0.00000	3	11	
5639	1626069178	9/10/2014	4	2.50000	2200	110642	1.00000	0.00000	0.00000	5	7	
4862	1117200170	9/19/2014	4	3.50000	3260	110579	2.00000	0.00000	0.00000	3	10	
18672	622079089	6/16/2014	4	2.50000	2040	109336	1.50000	0.00000	0.00000	4	8	
9277	1423069095	5/7/2014	3	2.50000	2460	108900	1.00000	0.00000	0.00000	4	9	
4535	3223069118	6/16/2014	3	3.50000	3380	108900	2.00000	0.00000	0.00000	3	9	
560	3624079046	10/28/2014	4	3.00000	2230	108900	1.00000	0.00000	0.00000	3	7	
5389	322069109	5/5/2015	2	2.25000	1910	108900	1.00000	0.00000	0.00000	4	7	
17879	2122039137	4/13/2015	3	2.50000	1656	108900	1.00000	0.00000	0.00000	4	7	
15520	922069169	5/29/2014	3	2.00000	2590	108900	2.00000	nan	0.00000	3	8	
19548	1625069101	7/7/2014	4	3.00000	5430	108900	2.00000	0.00000	0.00000	4	10	
6924	1824079073	3/31/2015	5	4.25000	4650	108464	2.00000	0.00000	0.00000	3	10	
2861	324069015	7/8/2014	4	3.50000	3110	108464	2.00000	0.00000	2.00000	4	8	
11200	2723069129	5/6/2015	3	2.50000	2620	108464	2.00000	nan	0.00000	4	8	
9146	721069087	5/7/2014	3	2.50000	3240	108366	2.00000	0.00000	0.00000	4	10	
19877	2621069017	3/3/2015	3	2.25000	1670	107157	1.00000	0.00000	0.00000	3	7	
15546	1326059182	4/6/2015	5	3.25000	5600	107157	2.00000	0.00000	0.00000	3	10	
7489	1722069052	10/24/2014	5	2.50000	4320	107157	1.00000	nan	0.00000	4	8	
4492	621069039	2/20/2015	4	2.25000	1620	106722	1.00000	0.00000	0.00000	3	8	
13737	220069106	4/1/2015	3	2.50000	1970	106722	1.00000	0.00000	4.00000	3	9	
17645	620079042	3/23/2015	2	1.00000	2360	105850	1.00000	0.00000	2.00000	2	6	
5475	1330910280	4/27/2015	4	2.50000	3720	105850	2.00000	0.00000	0.00000	4	10	

	id	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sc
2369	3623029045	9/25/2014	3	1.75000	2600	105587	1.00000	0.00000	0.00000	4	7	
21345	3123089027	7/21/2014	3	2.50000	3800	104979	2.00000	0.00000	0.00000	3	8	
3718	126059097	10/23/2014	3	3.50000	2690	104544	2.00000	0.00000	0.00000	3	8	
12087	2423069084	3/3/2015	4	2.75000	2400	104108	2.00000	0.00000	0.00000	3	8	
11136	1122059037	4/13/2015	3	1.75000	1560	104108	1.00000	0.00000	0.00000	3	7	
15551	5126210280	9/26/2014	3	2.50000	3440	103672	2.00000	0.00000	0.00000	3	9	
19238	1822059156	1/14/2015	3	3.50000	3650	103672	1.00000	0.00000	0.00000	3	10	
12656	619079061	6/19/2014	4	2.00000	2030	103672	1.00000	0.00000	0.00000	4	7	
1169	1117200390	5/7/2014	4	4.00000	4460	103382	2.00000	0.00000	0.00000	3	11	
14825	522039106	6/6/2014	3	1.00000	1210	103237	1.00000	0.00000	0.00000	2	6	
14688	1423069077	9/15/2014	2	1.75000	2870	102366	2.00000	0.00000	2.00000	4	8	
1253	1624079051	7/10/2014	2	2.50000	2410	102366	1.00000	0.00000	0.00000	4	7	
5	7237550310	5/12/2014	4	4.50000	5420	101930	1.00000	0.00000	0.00000	3	11	
1180	1224069074	8/6/2014	4	2.50000	3300	101930	2.00000	0.00000	0.00000	4	10	
6406	925069071	1/26/2015	5	3.75000	3500	101494	1.50000	0.00000	0.00000	3	8	
4432	1796100015	4/23/2015	4	3.50000	3090	100835	2.00000	0.00000	0.00000	3	9	
5451	3423059153	10/8/2014	4	3.00000	3370	100681	1.00000	0.00000	0.00000	5	8	
5700	2624079022	10/20/2014	3	2.25000	1880	100623	1.50000	0.00000	0.00000	3	8	
13230	1523069072	7/23/2014	3	2.25000	2680	100188	2.00000	0.00000	0.00000	4	8	
7511	1320069223	6/24/2014	3	1.50000	1810	100188	1.00000	0.00000	0.00000	5	7	
18984	1722069097	12/29/2014	3	2.50000	3100	100188	1.00000	0.00000	0.00000	4	7	
2382	621069074	6/3/2014	3	2.50000	1720	99916	2.00000	0.00000	0.00000	4	7	
1271	1822069116	12/17/2014	3	2.50000	2400	99752	1.00000	0.00000	0.00000	3	9	
4240	1926069143	10/16/2014	4	3.25000	3400	99170	1.00000	0.00000	0.00000	4	8	
3988	2023039160	4/23/2015	4	2.25000	2620	98881	1.00000	0.00000	0.00000	3	7	
8179	5229300085	4/11/2015	3	2.25000	2680	98445	1.00000	0.00000	0.00000	5	8	
13839	3422059010	3/27/2015	3	1.75000	2160	98445	2.00000	0.00000	0.00000	3	8	
10745	825069078	3/24/2015	5	2.25000	3100	97661	2.00000	0.00000	0.00000	3	9	

	id	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sc
6514	1722069145	12/23/2014	4	2.50000	3580	97574	2.00000	0.00000	0.00000	3	9	
2027	421079105	3/9/2015	3	2.25000	1480	97138	1.50000	0.00000	0.00000	3	7	
19914	2923039264	9/10/2014	2	1.75000	1728	95950	1.00000	0.00000	3.00000	3	9	
9467	222069057	3/30/2015	3	3.50000	3580	95832	1.50000	0.00000	0.00000	3	9	
17668	1423069076	9/26/2014	3	2.00000	2870	95396	1.00000	0.00000	0.00000	4	9	
3768	622059019	9/19/2014	5	1.50000	1830	94960	1.50000	0.00000	0.00000	3	7	
17738	126059019	3/16/2015	4	2.50000	3170	94855	1.00000	0.00000	0.00000	4	9	
5096	824069193	9/11/2014	4	1.75000	1760	94525	1.50000	nan	0.00000	3	7	
236	4058000060	4/9/2015	3	2.00000	2220	94300	1.00000	0.00000	0.00000	5	7	
13110	2322069010	10/7/2014	5	5.00000	3960	94089	2.00000	0.00000	0.00000	3	10	
10433	1775500310	1/21/2015	4	1.75000	3060	94089	1.00000	0.00000	0.00000	3	8	
12104	520069032	7/16/2014	3	1.75000	1890	93218	1.00000	0.00000	0.00000	4	7	
19433	2023059052	5/4/2015	3	1.00000	1350	92721	1.00000	0.00000	0.00000	2	6	
11910	2591720160	5/1/2015	3	2.75000	3510	92347	2.00000	0.00000	0.00000	3	10	
10994	3024059036	5/30/2014	4	1.75000	2500	92347	1.00000	0.00000	0.00000	4	8	
16085	2523069134	4/6/2015	4	2.50000	2480	91911	1.00000	0.00000	2.00000	4	7	
6396	2624049091	3/13/2015	5	2.50000	3750	91681	2.00000	1.00000	4.00000	3	10	
9554	3023039231	7/14/2014	1	1.00000	920	91476	1.50000	0.00000	0.00000	3	6	
7849	524069101	7/23/2014	4	2.00000	3380	90968	1.00000	0.00000	0.00000	4	9	
19259	821069025	2/13/2015	3	2.50000	3290	90796	2.00000	0.00000	0.00000	4	10	
15784	1525069134	3/12/2015	4	3.50000	3790	90169	2.00000	0.00000	0.00000	3	11	
16784	3124089049	12/8/2014	4	1.75000	2800	90169	2.00000	nan	0.00000	3	7	
19842	2524069078	1/22/2015	4	4.00000	7850	89651	2.00000	0.00000	0.00000	3	12	
9390	926069140	7/21/2014	4	3.00000	3590	89640	2.00000	0.00000	0.00000	3	10	
3206	1125069064	3/31/2015	4	2.50000	2770	89298	2.00000	0.00000	0.00000	3	8	
5396	2423069120	5/8/2014	2	1.75000	2200	89298	1.00000	0.00000	0.00000	3	7	
4887	2424059127	8/20/2014	2	1.75000	3490	88909	1.00000	0.00000	3.00000	3	10	
13387	1524069044	10/9/2014	4	4.50000	6380	88714	2.00000	0.00000	0.00000	3	12	
19412	1423069162	6/4/2014	4	2.25000	2740	88426	2.00000	0.00000	0.00000	3	7	

	id	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sc
3320	1323089184	5/2/2014	3	2.50000	2430	88426	1.00000	0.00000	0.00000	4	7	
4358	1221079058	8/27/2014	2	1.00000	1120	88327	1.50000	nan	0.00000	4	6	
6486	8656800190	10/2/2014	3	1.75000	2080	87991	1.00000	0.00000	0.00000	3	6	
8535	3523069047	8/25/2014	4	2.75000	4010	87555	2.00000	0.00000	0.00000	3	10	
4480	1122069019	8/26/2014	4	3.50000	3490	87497	2.00000	0.00000	0.00000	3	9	
5338	1926069063	3/6/2015	3	1.75000	1790	87213	1.00000	0.00000	0.00000	4	7	
19465	1720069146	7/15/2014	3	2.00000	1590	87120	1.00000	0.00000	3.00000	3	8	
6993	3401700255	7/29/2014	4	2.00000	3090	87120	1.00000	0.00000	0.00000	4	7	
1969	2129700320	5/5/2015	1	0.75000	940	87120	1.00000	0.00000	0.00000	3	6	
18658	1724079048	12/8/2014	3	2.50000	2680	87117	1.00000	0.00000	0.00000	3	7	
5788	1125069134	4/30/2015	3	2.25000	2980	86636	1.00000	nan	0.00000	3	9	
6906	2386000020	10/8/2014	4	2.25000	4470	86225	2.00000	0.00000	0.00000	3	10	
16905	293800680	4/15/2015	4	3.00000	4270	85643	2.00000	0.00000	0.00000	3	11	
11511	1523069128	3/31/2015	5	2.75000	2910	85377	1.00000	0.00000	0.00000	4	8	
4961	4014400190	7/14/2014	4	2.50000	2846	85377	1.50000	0.00000	0.00000	3	8	
14661	2051200506	4/13/2015	3	1.00000	1190	85226	1.50000	0.00000	0.00000	5	5	
577	1526069135	12/11/2014	4	4.00000	6050	84942	2.50000	0.00000	2.00000	3	9	
3716	2526059076	2/25/2015	6	2.75000	3360	84506	1.00000	nan	0.00000	5	7	
1054	5416300240	2/2/2015	4	4.50000	5670	84267	2.00000	0.00000	2.00000	3	11	
9792	5561100431	8/6/2014	5	2.50000	2510	83231	1.00000	0.00000	0.00000	4	7	
8854	1226059161	12/29/2014	4	2.75000	2560	83200	1.00000	0.00000	0.00000	3	8	
8866	522039103	11/13/2014	2	1.50000	1040	83199	1.00000	0.00000	0.00000	4	7	
10086	3224059033	10/7/2014	4	1.50000	3050	82764	1.00000	0.00000	0.00000	3	8	
17171	1523069022	5/6/2015	3	1.50000	1630	82764	1.00000	0.00000	0.00000	4	6	
16863	2725069108	8/5/2014	3	3.25000	4610	81935	2.00000	0.00000	0.00000	4	9	
17929	3223039010	8/4/2014	2	1.00000	570	81893	1.00000	0.00000	1.00000	3	6	
5083	921059132	8/13/2014	3	2.00000	1680	81893	1.00000	0.00000	0.00000	3	7	
2389	323089084	8/27/2014	3	2.50000	2400	81892	2.00000	0.00000	0.00000	3	8	

	id	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sc
13931	3223059141	5/9/2014	2	1.00000	1420	81892	1.00000	0.00000	0.00000	3	7	
1240	226059078	2/27/2015	2	1.00000	1840	81892	1.00000	nan	0.00000	3	6	
1241	1796100140	7/15/2014	3	1.50000	1350	81549	1.00000	0.00000	0.00000	2	7	
17557	3751606514	6/26/2014	2	1.00000	1780	81021	1.00000	nan	3.00000	4	9	
14806	1523069151	7/11/2014	2	1.00000	1470	81021	1.00000	0.00000	0.00000	4	6	
11034	7511200350	9/19/2014	3	1.75000	2040	81021	1.00000	0.00000	0.00000	3	8	
1426	2481630070	1/28/2015	4	3.00000	3180	80837	2.00000	0.00000	0.00000	3	11	
4262	8887001140	7/23/2014	3	3.00000	3290	80471	2.00000	0.00000	2.00000	4	8	
17361	5238800020	12/8/2014	2	2.25000	1600	80400	2.00000	0.00000	0.00000	4	7	
11878	1523089012	11/20/2014	4	1.00000	1520	80150	1.00000	0.00000	0.00000	2	5	
1349	2423059104	10/8/2014	3	2.00000	1970	79714	1.00000	0.00000	0.00000	3	7	
14467	921049141	12/1/2014	3	2.25000	3280	79279	1.00000	0.00000	0.00000	3	10	
14531	2326059080	8/1/2014	3	2.50000	3420	79279	2.00000	0.00000	0.00000	3	11	
15519	2924069132	5/27/2014	3	1.75000	2310	78844	1.00000	0.00000	0.00000	3	8	
13363	524069115	5/9/2014	3	2.25000	2950	78843	1.50000	0.00000	0.00000	3	9	
4798	1922069099	5/23/2014	3	2.00000	1370	78408	1.00000	0.00000	0.00000	5	7	
4866	524069049	4/2/2015	3	1.50000	1460	78408	1.00000	0.00000	0.00000	4	7	
4380	2423029245	6/17/2014	3	1.75000	2240	78225	2.00000	0.00000	0.00000	5	8	
358	325059171	5/5/2014	3	1.00000	1330	77972	1.00000	0.00000	0.00000	3	7	
20858	5416300230	7/17/2014	4	3.50000	4130	77832	2.00000	0.00000	2.00000	3	10	
7629	4008400515	1/20/2015	1	0.75000	780	77603	1.00000	0.00000	0.00000	1	5	
10254	224069114	8/29/2014	4	2.50000	2470	77550	1.00000	0.00000	0.00000	4	7	
10710	226059121	8/13/2014	3	2.75000	1560	77536	1.00000	0.00000	0.00000	3	7	
5024	3244500078	8/22/2014	3	2.50000	4930	77536	2.00000	0.00000	0.00000	3	9	
7151	3622059180	7/3/2014	4	2.00000	1900	76877	1.00000	0.00000	0.00000	3	8	
2123	4379600030	7/29/2014	3	3.75000	6400	76665	1.00000	0.00000	2.00000	4	10	
10787	3751602797	7/2/2014	4	2.00000	2370	76665	2.00000	0.00000	0.00000	4	8	
18792	2721049061	7/9/2014	3	1.75000	3160	76230	1.00000	0.00000	0.00000	4	8	

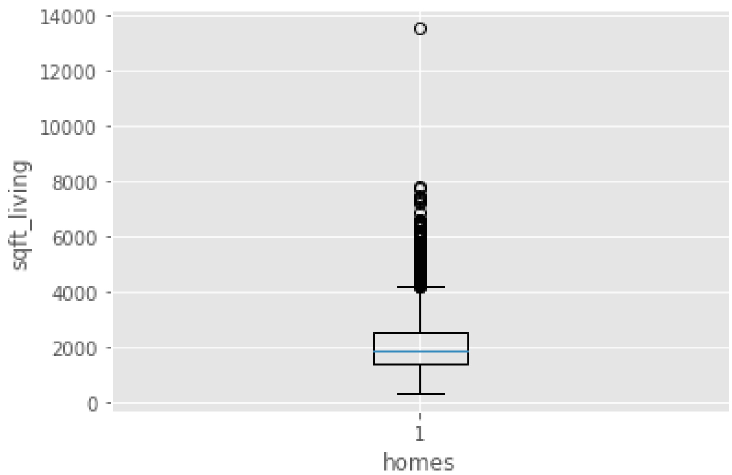
	id	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sc
532	4403600270	2/24/2015	4	3.25000	4740	76230	2.00000	0.00000	0.00000	3	10	
4365	4166600115	11/21/2014	3	2.75000	3230	75889	2.00000	1.00000	4.00000	3	7	
575	222069082	12/17/2014	2	1.00000	1220	75794	1.00000	0.00000	0.00000	4	7	
6139	3522049063	4/2/2015	4	2.50000	3380	75794	2.00000	0.00000	0.00000	3	10	
12467	3226059083	6/26/2014	3	1.75000	2080	75794	1.00000	0.00000	0.00000	3	7	
4644	1726069064	3/24/2015	2	1.00000	1140	75132	1.00000	0.00000	0.00000	3	7	
2121	1085610030	8/1/2014	4	2.50000	2790	74495	2.00000	0.00000	0.00000	3	9	
5027	1117200190	8/4/2014	3	2.50000	3010	74390	2.00000	0.00000	0.00000	3	10	
15685	3521069142	2/24/2015	3	2.50000	2260	74297	2.00000	0.00000	0.00000	3	9	
17569	1920079039	8/15/2014	2	1.00000	1140	74052	1.00000	0.00000	0.00000	4	6	
3160	1226059112	2/20/2015	3	1.00000	1360	73616	1.00000	0.00000	0.00000	3	7	
12481	1921069101	5/8/2015	3	1.75000	2170	73616	1.00000	0.00000	0.00000	3	7	
1424	3751604974	12/4/2014	2	1.50000	1320	73600	1.00000	0.00000	0.00000	3	7	
11660	1326059185	3/20/2015	4	2.50000	2800	72309	2.00000	0.00000	0.00000	3	9	
4897	522069022	7/14/2014	5	2.50000	2950	72309	2.00000	0.00000	0.00000	3	8	
10915	1823069155	5/5/2014	5	1.75000	2550	71874	1.00000	0.00000	0.00000	5	7	
6928	5153200651	3/16/2015	3	1.00000	1220	71191	1.00000	0.00000	0.00000	3	6	
17488	3521069150	10/17/2014	3	2.50000	2440	71002	1.00000	0.00000	0.00000	4	9	
16859	4058000010	5/9/2014	4	1.50000	1470	70800	1.00000	0.00000	0.00000	3	7	
15485	3425079088	8/19/2014	3	2.50000	2210	70567	2.00000	0.00000	3.00000	3	9	
13887	1242700035	11/3/2014	4	2.75000	3470	70131	1.00000	0.00000	0.00000	4	8	
20876	7299810040	4/6/2015	4	3.00000	5370	69848	2.00000	nan	0.00000	3	10	
2191	1117200550	10/14/2014	3	2.75000	3530	69834	2.00000	0.00000	0.00000	3	10	
368	424069250	4/23/2015	4	2.75000	2440	69415	1.00000	0.00000	0.00000	4	8	
15690	7937600010	12/12/2014	4	1.00000	1750	68841	1.00000	0.00000	0.00000	3	7	
7757	7574200210	6/18/2014	4	1.50000	2310	68824	2.00000	0.00000	0.00000	4	7	
16331	7802900224	7/7/2014	5	2.50000	2860	68519	2.00000	0.00000	0.00000	5	8	
3629	425079100	12/31/2014	3	2.75000	1840	68479	1.00000	0.00000	2.00000	3	8	
18274	9262800208	9/19/2014	4	3.50000	4083	68377	2.00000	0.00000	0.00000	3	10	

	id	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sc
9244	853600150	5/24/2014	4	4.25000	5584	68257	2.00000	0.00000	0.00000	3	11	
12449	182000350	3/25/2015	5	2.00000	2020	67953	1.50000	0.00000	0.00000	4	7	
3418	638100015	3/12/2015	3	2.00000	1540	67953	1.00000	nan	0.00000	3	7	
15152	3304700130	1/28/2015	4	4.00000	3860	67953	2.00000	0.00000	2.00000	4	12	
11239	625100004	3/17/2015	3	2.00000	1540	67756	1.00000	0.00000	0.00000	3	7	
5422	3528000545	8/15/2014	4	3.25000	3090	67518	2.00000	0.00000	0.00000	3	10	
13903	3395350210	3/24/2015	5	3.25000	2950	67475	1.00000	0.00000	0.00000	3	8	
14300	1560930450	10/24/2014	3	2.50000	3090	67082	2.00000	0.00000	0.00000	3	9	
6935	1026069120	5/8/2014	2	3.00000	3160	66646	2.00000	nan	0.00000	3	7	
11214	4054530240	4/27/2015	4	3.50000	4380	66613	1.50000	0.00000	0.00000	3	11	
21370	774101755	4/17/2015	3	1.75000	1790	66250	1.50000	0.00000	0.00000	3	7	
18896	822069118	7/29/2014	3	3.25000	3660	66211	2.00000	0.00000	0.00000	3	10	
198	2824079053	1/13/2015	3	2.50000	1910	66211	2.00000	0.00000	0.00000	3	7	
11930	625100181	5/8/2014	4	2.50000	2280	65836	2.00000	nan	0.00000	3	8	
17021	2260800170	7/18/2014	3	2.25000	3130	65775	2.00000	0.00000	0.00000	4	8	
12622	226059106	1/2/2015	3	1.75000	2090	65558	1.00000	0.00000	0.00000	3	8	
12456	2386000240	9/29/2014	5	3.50000	3870	65556	2.00000	nan	0.00000	3	10	
8331	6902000100	9/15/2014	3	1.75000	2420	65501	2.00000	nan	1.00000	3	8	
13859	2423069164	4/10/2015	3	2.00000	1990	65340	2.00000	0.00000	0.00000	3	8	
11669	3622059157	10/9/2014	4	1.75000	1850	65340	1.50000	0.00000	0.00000	4	7	
6524	1240700006	5/11/2015	3	2.00000	2320	65340	1.50000	0.00000	0.00000	3	9	
10959	2623089135	6/16/2014	3	2.50000	1830	65340	1.00000	0.00000	0.00000	3	8	
2658	823069074	12/23/2014	4	2.50000	2660	65340	2.00000	0.00000	0.00000	3	8	
317	3422059208	5/11/2015	3	2.50000	1930	64904	1.00000	0.00000	0.00000	4	8	
12454	1121039105	12/3/2014	4	3.00000	2150	64694	1.00000	0.00000	0.00000	3	8	
19964	774100475	6/27/2014	3	2.75000	2600	64626	1.50000	0.00000	0.00000	3	8	
6099	1775500050	1/29/2015	1	1.00000	1160	64469	1.00000	0.00000	0.00000	3	7	
10830	1526059051	8/28/2014	2	2.00000	1600	64468	1.00000	0.00000	0.00000	3	7	

	id	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sc
20154	6626300095	5/19/2014	4	2.50000	3430	64441	2.00000	0.00000	0.00000	3	8	
2411	3407700046	6/24/2014	3	2.50000	2410	64073	1.00000	0.00000	0.00000	4	8	
11137	5515600163	9/16/2014	5	2.25000	3070	64033	1.00000	0.00000	0.00000	3	9	
4637	4054520100	2/10/2015	4	2.50000	3700	63991	2.00000	0.00000	0.00000	3	10	
15234	1320069249	10/20/2014	1	1.00000	470	63737	1.00000	0.00000	2.00000	5	5	
14470	3523069060	11/7/2014	3	1.75000	1340	63597	1.00000	0.00000	0.00000	4	7	
19474	926069009	6/9/2014	4	2.50000	2350	63162	2.00000	0.00000	0.00000	4	8	
4627	203600590	6/27/2014	4	2.50000	2770	63118	2.00000	0.00000	0.00000	3	9	
5517	826069016	12/12/2014	4	3.00000	3280	62726	1.50000	nan	0.00000	3	7	
5302	3293200190	12/13/2014	4	3.25000	4750	62365	2.00000	0.00000	0.00000	3	11	
4222	522069119	5/12/2015	3	2.50000	2720	62310	1.00000	0.00000	0.00000	3	8	
9223	7214700580	6/8/2014	4	2.25000	2450	62290	2.00000	0.00000	0.00000	3	8	
13157	1428000970	5/21/2014	3	1.75000	1300	62290	1.00000	0.00000	0.00000	3	7	
17606	1593000690	4/8/2015	3	1.00000	1170	62290	2.00000	0.00000	0.00000	3	5	
15306	321059132	4/27/2015	3	1.75000	1450	61419	1.00000	0.00000	0.00000	4	8	

In [126]:

```
# investigate sqft_living outliers

plt.boxplot(train_data.sqft_living)
plt.xlabel('homes')
plt.ylabel('sqft_living');

# the max value may be accurate, but let's drop it to improve model accuracy
```



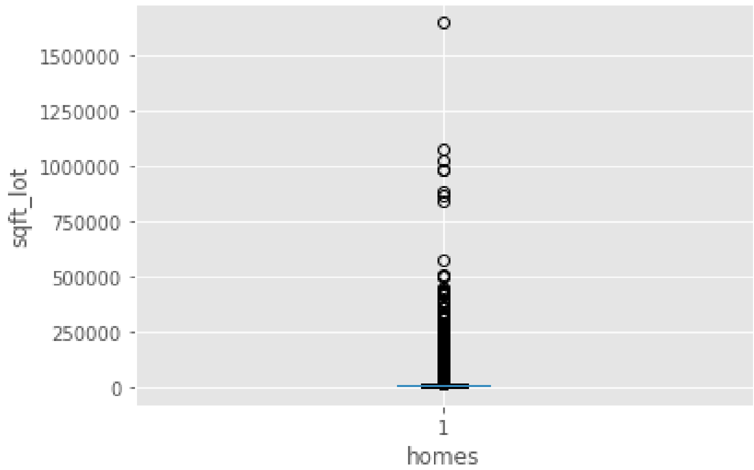
In [127]:

```
# drop sqft_living > 8000 from training and testing data

train_data = train_data.loc[train_data['sqft_living'] <= 8000]
```

```
test_data = test_data.loc[test_data['sqft_living'] <= 8000]
```

In [128]:

```
# investigate sqft_lot outliers

plt.boxplot(train_data.sqft_lot)
plt.xlabel('homes')
plt.ylabel('sqft_lot');

# the max values may be accurate, but let's drop them to improve model accuracy
```



In [129]:

```
# remove sqft_lot > 600,000 from train and test data

train_data = train_data.loc[train_data['sqft_lot'] <= 600000]
test_data = test_data.loc[test_data['sqft_lot'] <= 600000]
```

Null values

Four columns had null values: 'waterfront', 'yr_renovated', 'sqft_basement', and 'view'. Since fewer than 1% of properties were marked as having waterfront views, I dropped this column from the analysis. I replaced 'yr_renovated' with a binary column showing whether or not the home was marked renovated in any year. I replaced the null values in 'sqft_basement' (which appeared as ? in the data) with zeros, since the median of the non-null values in this column was also zero. Since 'view' refers to the number of times a house had been viewed (not whether it has a nice view), I dropped this column from the analysis.

In [130]:

```
# investigate null values in waterfront

train_data.waterfront.value_counts() # binary - 1 or 0
train_data.waterfront.isna().sum() #1647 null values out of 21596
train_data.waterfront.value_counts()

# Only 89 homes are marked as waterfront -- less than 1% of data
# So I'll drop this feature from the analysis
```

Out[130]:

```
0.00000    13336
1.00000       89
Name: waterfront, dtype: int64
```

In [131]:

```
# deal with null values in yr_renovated

train_data['yr_renovated'].value_counts().head(50)
# 11869 values are 0 (meaning no true value)
```

```
nulls = train_data['yr_renovated'].isna().sum()
nulls #2692 values are null

# many of the values with years are old, e.g. 1950's-1990's
```

Out[131]:

```
2692
```

In [132]:

```
# create a new column showing homes renovated or not
train_data['renovated'] = np.where(train_data['yr_renovated'] > 0, 1, 0)
train_data['renovated'].value_counts() # only 511 homes show a year renovated

# make same change to testing data
test_data['renovated'] = np.where(test_data['yr_renovated'] > 0, 1, 0)
```

In [133]:

```
# deal with non-number values in sqft_basement
train_data['sqft_basement'].value_counts() # continuous variable, but has many '?' values
# also many 0 values.  Not sure if these homes truly do not have basements.

# per cent of data that is missing:
missing_sqft_basement = round((len(train_data.loc[train_data['sqft_basement'] == '?'])/l
en(train_data))*100, 2)
print(missing_sqft_basement, "% of basement data is missing")

# per cent of data that is zero:
missing_sqft_basement = round((len(train_data.loc[train_data['sqft_basement'] == '0.0'])
/len(train_data))*100, 2)
print(missing_sqft_basement, "% of basement data is zero")
```

```
2.11 % of basement data is missing
59.37 % of basement data is zero
```

In [134]:

```
# for now, let's fill missing values with zero, since that's the median

# replace all '?' values with '0'
train_data.loc[train_data['sqft_basement'] == '?', 'sqft_basement'] = '0'

# same for test data
test_data.loc[test_data['sqft_basement'] == '?', 'sqft_basement'] = '0'
```

In [135]:

```
# now convert sqft_basement values to integers

train_data['sqft_basement'] = pd.to_numeric(train_data['sqft_basement'])

test_data['sqft_basement'] = pd.to_numeric(test_data['sqft_basement'])

train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15072 entries, 753 to 15795
Data columns (total 22 columns):
id               15072 non-null int64
date             15072 non-null object
bedrooms         15072 non-null int64
bathrooms        15072 non-null float64
sqft_living      15072 non-null int64
sqft_lot         15072 non-null int64
floors           15072 non-null float64
waterfront       13425 non-null float64
view             15029 non-null float64
condition        15072 non-null int64
grade            15072 non-null int64
sqft_above       15072 non-null int64
```

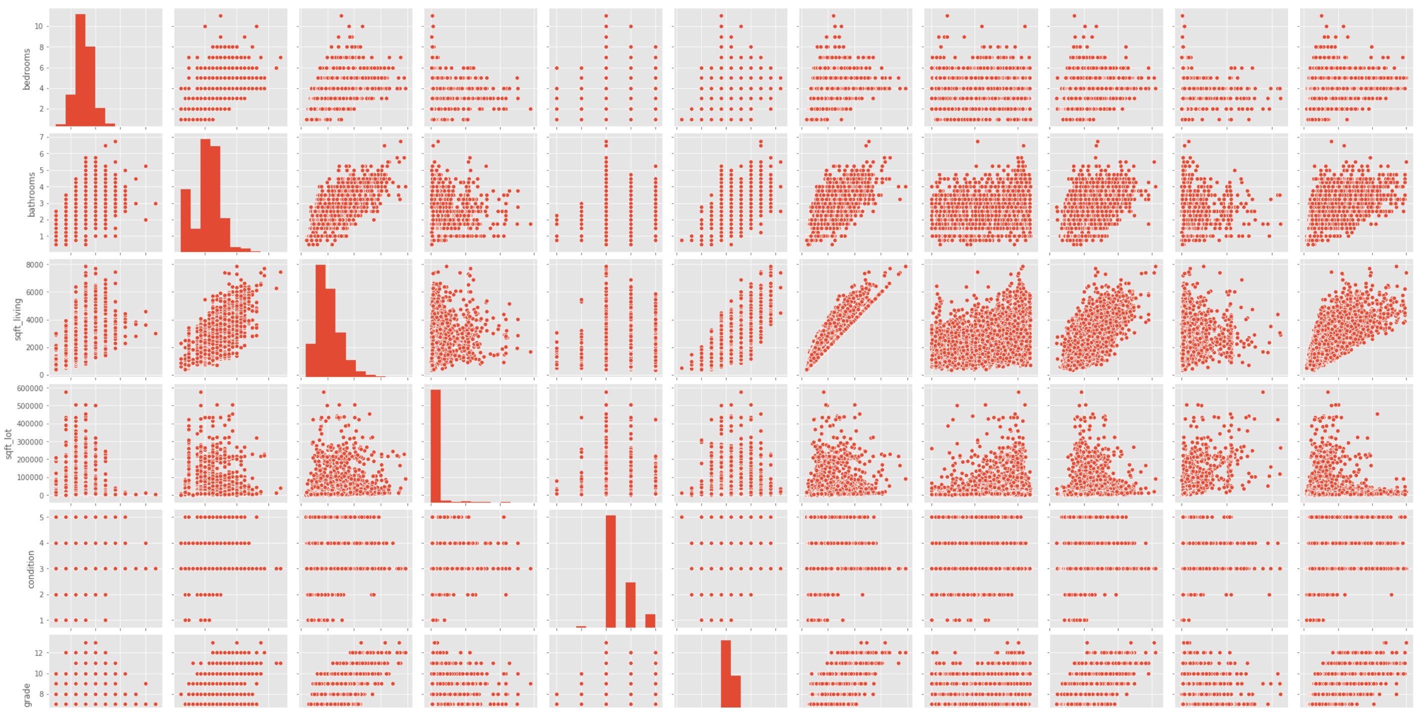
```
sqft_basement    15072 non-null float64
yr_built         15072 non-null int64
yr_renovated     12380 non-null float64
zipcode          15072 non-null int64
lat              15072 non-null float64
long             15072 non-null float64
sqft_living15    15072 non-null int64
sqft_lot15       15072 non-null int64
price            15072 non-null float64
renovated        15072 non-null int64
dtypes: float64(9), int64(12), object(1)
memory usage: 2.6+ MB
```

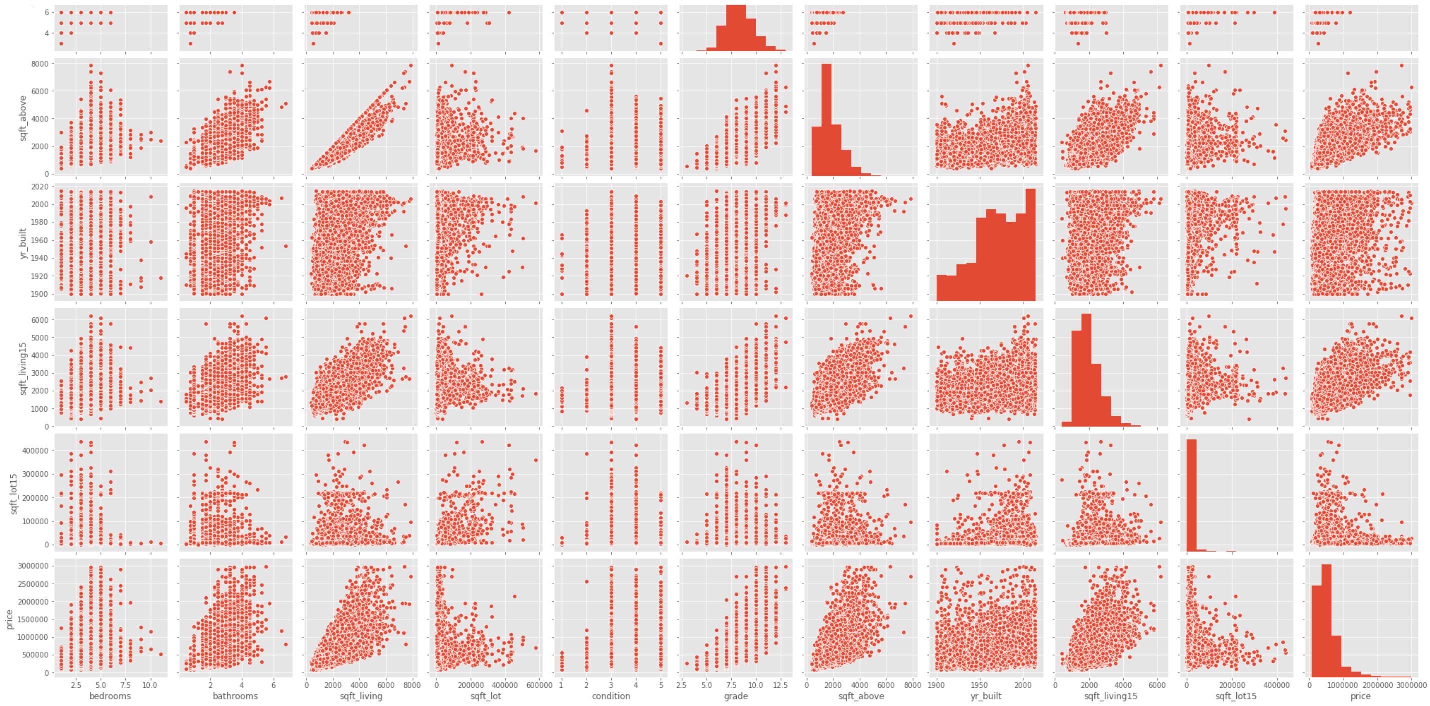
Exploring Correlations

The analysis below shows that the strongest correlations with price (the target variable) are sqft_living, sqft_above, sqft_living15, grade, and bathrooms. The strongest correlations between X variables are among these same columns -- all five are correlated with each other. This multicolinearity could negatively impact a prediction model, so I will experiment with dropping combinations of multicolinear columns later on. Of all the variables, only grade and bedrooms look normally distributed. Most are right-skewed, including price. Later, I will see if log transformations on these variables improve the model.

In [136]:

```
# explore training data
# let's try a pairplot to see if anything stands out

cols_of_interest = [
                    'bedrooms',
                    'bathrooms',
                    'sqft_living',
                    'sqft_lot',
                    'condition',
                    'grade',
                    'sqft_above',
                    'yr_built',
                    'sqft_living15',
                    'sqft_lot15',
                    'price']

sns.pairplot(train_data[cols_of_interest]);

# the strongest correlations with price are sqft_living, sqft_above, sqft_living15, grade
, and bathrooms
# strongest correlations between X variables are among these same columns
# only grade and bedrooms look normally distributed
```



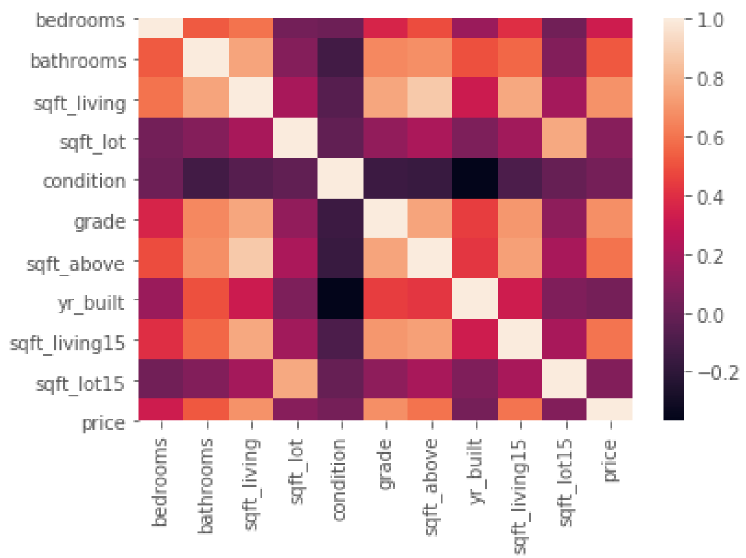


In [137]:

```
# let's make a heatmap to be sure

sns.heatmap(train_data[cols_of_interest].corr())

# yes, confirms the observations above
```

Out[137]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a36321128>
```



In [138]:

```
# let's look at the numbers

train_data[cols_of_interest].corr()

# sqft_living is the best predictor of price so far
```

Out[138]:

	bedrooms	bathrooms	sqft_living	sqft_lot	condition	grade	sqft_above	yr_built	sqft_living15	sqft_lot15	p
bedrooms	1.00000	0.52770	0.59653	0.03883	0.01801	0.36141	0.49047	0.15949	0.40267	0.03151	0.3
bathrooms	0.52770	1.00000	0.75319	0.09375	-0.12716	0.66083	0.68084	0.50416	0.56656	0.08478	0.5
sqft_living	0.59653	0.75319	1.00000	0.20370	-0.06040	0.75806	0.87374	0.32238	0.76134	0.19124	0.6
sqft_lot	0.03883	0.09375	0.20370	1.00000	-0.01926	0.13559	0.21213	0.06891	0.17966	0.77070	0.1

	bedrooms	bathrooms	sqft_living	sqft_lot	condition	grade	sqft_above	yr_built	sqft_living15	sqft_lot15	p
sqft_lot	[illegible]	[illegible]	[illegible]	1.00000	-0.01926	0.13559	0.21213	0.06891	0.17966	0.77070	[illegible]
condition	0.01801	-0.12716	-0.06040	-0.01926	1.00000	-0.14761	-0.16191	-0.36453	-0.09209	-0.00618	0.04
grade	0.36141	0.66083	0.75806	0.13559	-0.14761	1.00000	0.75205	0.44632	0.71063	0.12543	0.68
sqft_above	0.49047	0.68084	0.87374	0.21213	-0.16191	0.75205	1.00000	0.42744	0.73452	0.20124	0.59
yr_built	0.15949	0.50416	0.32238	0.06891	-0.36453	0.44632	0.42744	1.00000	0.32377	0.07746	0.04
sqft_living15	0.40267	0.56656	0.76134	0.17966	-0.09209	0.71063	0.73452	0.32377	1.00000	0.20082	0.60
sqft_lot15	0.03151	0.08478	0.19124	0.77070	-0.00618	0.12543	0.20124	0.07746	0.20082	1.00000	0.08
price	0.32388	0.52305	0.69110	0.10209	0.04500	0.68038	0.59729	0.04564	0.60402	0.08481	1.00

In [139]:

```
# let's look just at the correlations with price

train_data[cols_of_interest].corr()['price'].sort_values(ascending=False)

# interesting, sqft_living and grade are far above the rest
# grade is probably based in part on sqft_living
```

Out[139]:

```
price            1.00000
sqft_living      0.69110
grade            0.68038
sqft_living15    0.60402
sqft_above       0.59729
bathrooms        0.52305
bedrooms         0.32388
sqft_lot         0.10209
sqft_lot15       0.08481
yr_built         0.04564
condition        0.04500
Name: price, dtype: float64
```

Column Exclusion

Based on the analyses above, I decided to exclude the following columns from the model. Justifications are provided below:

- id - the randomly assigned house id
- date - date sold, all are between May 2014 and May 2015. May investigate impact of month later
- waterfront - less than 1% of homes marked as waterfront
- view - number of times the home has been viewed - not relevant for pricing homes newly on the market
- yr_renovated - missing values. Turned into binary column 'renovated'
- 'lat' and 'long' - latitude and longitude of house - easier to pull location info with zipcode

Modeling

In this section, I tested nine models, and selected Model 8 as the most effective. \ \ I first calculated a model-less baseline with an R-squared of 0 and a Mean Absolute Error of $227K. I then tested a baseline linear regression model without transforming any features of the data. This produced an R-squared of 0.64 and 0.62 for the training and test sets respectively, and Mean Absolute Errors of \$135K and $136K respectively. \ \ After experimenting with log-transforming the X variables and the target variable price, I was able to improve the metrics by log-transforming price, sqft_living15, and sqft_lot15. By assigning zip codes to price-based classifications, I improved the R-squared to 0.83 for both the training and test data, with Mean Absolute Errors of \$88K and $87K respectively. \ \ I also tested other strategies, such as reducing multicolinearity by dropping columns, omitting features with high p-values, and assigning the yr_built data to categories. None of these changes improved the model. However, omitting features with high p-values did not reduce the model's effectiveness either, so I kept this change in order to simplify the model.

In [140]:

```
# split the preprocessed training and test sets back into X and y

# choose relevant columns:

X_train=train_data[['bedrooms',
       'bathrooms',
       'sqft_living',
       'sqft_lot',
       'floors',
       'condition',
       'grade',
       'sqft_above',
       'sqft_basement',
       'yr_built',
       'zipcode',
       'sqft_living15',
       'sqft_lot15',
       'renovated']]

y_train=train_data['price']

X_test=test_data[['bedrooms',
       'bathrooms',
       'sqft_living',
       'sqft_lot',
       'floors',
       'condition',
       'grade',
       'sqft_above',
       'sqft_basement',
       'yr_built',
       'zipcode',
       'sqft_living15',
       'sqft_lot15',
       'renovated']]

y_test=test_data['price']

print(len(X_train), len(X_test), len(y_train), len(y_test))
```

```
15072 6458 15072 6458
```

Model 1: Model-less Baseline

In [141]:

```
# for our first model-less baseline, let's use the mean price
# start with training set

mean_price = y_train.mean()
y_pred_train = np.full(shape=(len(X_train), 1), fill_value=mean_price)

# check r2
r2_baseline_train = round(r2_score(y_true=y_train, y_pred=y_pred_train), 6)

# check Mean Absolute Error
mae_baseline_train = round(mean_absolute_error(y_true=y_train, y_pred=y_pred_train), 2)

# check Root Mean Squared Error
rmse_baseline_train = round(np.sqrt(mean_squared_error(y_true=y_train, y_pred=y_pred_tra
in)), 2)

print('Training Data', '\n',
      'Mean Price:', round(mean_price, 2), '\n',
      'R-Squared:', r2_baseline_train, '\n',
      'Mean Absolute Error:', mae_baseline_train, '\n',
      'Root Mean Squared Error:', rmse_baseline_train)
```

```
Training Data
```

```
Mean Price: 534162.34
 R-Squared: 0.0
 Mean Absolute Error: 226613.15
 Root Mean Squared Error: 331365.24
```

In [142]:

```
# now let's calculate baseline r2, MAE, and RMSE for the test set

y_pred_test = np.full(shape=(len(X_test), 1), fill_value=mean_price)

r2_baseline_test = round(r2_score(y_true=y_test, y_pred=y_pred_test), 6)
mae_baseline_test = round(mean_absolute_error(y_true=y_test, y_pred=y_pred_test), 2)
rmse_baseline_test = round(np.sqrt(mean_squared_error(y_true=y_test, y_pred=y_pred_test)
), 2)

print('Testing Data', '\n',
      'Mean Price:', round(mean_price, 2), '\n',
      'R-Squared:', r2_baseline_test, '\n',
      'Mean Absolute Error:', mae_baseline_test, '\n',
      'Root Mean Squared Error:', rmse_baseline_test)
```

```
Testing Data
 Mean Price: 534162.34
 R-Squared: -0.000468
 Mean Absolute Error: 221118.2
 Root Mean Squared Error: 315393.0
```

In [143]:

```
# create a function for evaluating models:

def evaluate_model(y_train, y_train_pred, y_test, y_test_pred):

    """Calculate evaluation metrics for the model: R-Squared, Mean Absolute Error, and Ro
ot Mean Squared Error

    Parameters
    ----------
    y_train: Series of true values from the training set target variable
    y_train_pred: Series of target variable values predicted by the model for the trainin
g set
    y_test: Series of true values from the test set target variable
    y_test_pred: Series of target variable values predicted by the model for the test set

    Returns
    -------
    Print of metrics for training and test sets"""

    # check train r2
    r2_train = round(r2_score(y_true=y_train, y_pred=y_train_pred), 6)

    # check train Mean Absolute Error
    mae_train = round(mean_absolute_error(y_true=y_train, y_pred=y_train_pred), 2)

    # check train Root Mean Squared Error
    rmse_train = round(np.sqrt(mean_squared_error(y_true=y_train, y_pred=y_train_pred)),
2)

    print('Training Data', '\n',
          'R-Squared:', r2_train, '\n',
          'Mean Absolute Error:', mae_train, '\n',
          'Root Mean Squared Error:', rmse_train, '\n')

    # check test r2
    r2_test = round(r2_score(y_true=y_test, y_pred=y_test_pred), 6)

    # check test Mean Absolute Error
    mae_test = round(mean_absolute_error(y_true=y_test, y_pred=y_test_pred), 2)

    # check train Root Mean Squared Error
```

```
    rmse_test = round(np.sqrt(mean_squared_error(y_true=y_test, y_pred=y_test_pred)), 2)

    print('Testing Data', '\n',
          'R-Squared:', r2_test, '\n',
          'Mean Absolute Error:', mae_test, '\n',
          'Root Mean Squared Error:', rmse_test)
```

In [144]:

```
# the mean is not a good predictor of price!

# let's fit a baseline regression model
```

Model 2: Baseline Linear Regression

In [145]:

```
# first, let's scale the data so we can evaluate the coefficients of the baseline model

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [146]:

```
# now do a linear regression

linreg = LinearRegression()
linreg.fit(X_train_scaled, y_train)

y_train_pred2 = linreg.predict(X_train_scaled)
y_test_pred2 = linreg.predict(X_test_scaled)

evaluate_model(y_train, y_train_pred2, y_test, y_test_pred2)
```

```
Training Data
 R-Squared: 0.635164
 Mean Absolute Error: 135333.41
 Root Mean Squared Error: 200150.17

Testing Data
 R-Squared: 0.618171
 Mean Absolute Error: 135514.39
 Root Mean Squared Error: 194842.99
```

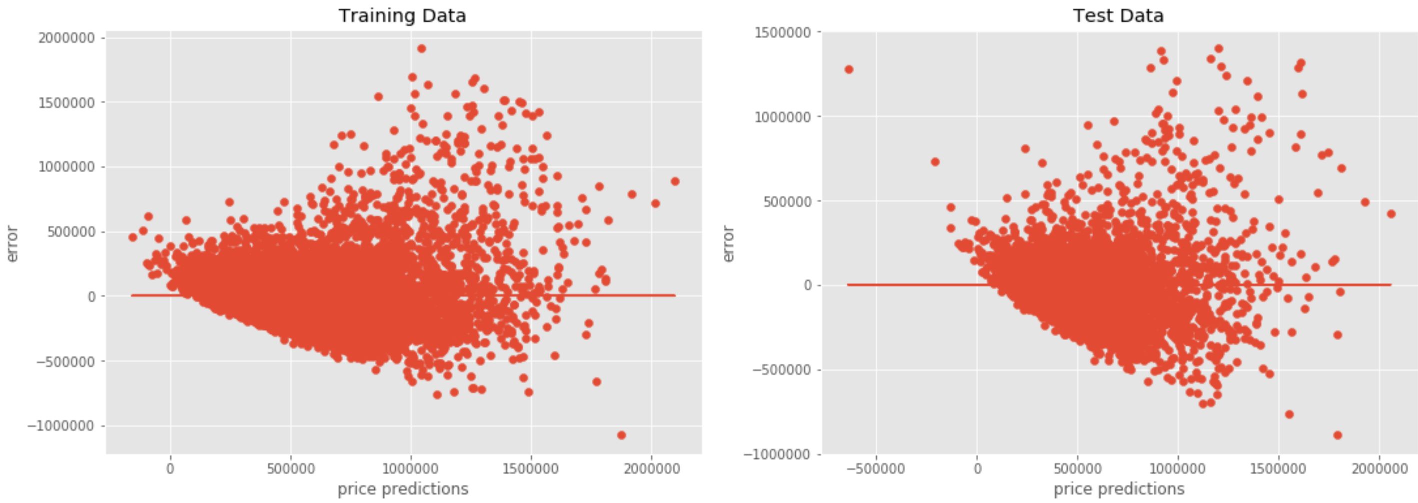
In [147]:

```
# that's better, but the model still only explains about 60% of the variance
# store as 'best_r2' for comparison

best_r2 = {'train': 0.635164, 'test': 0.618171}
```

In [148]:

```
# let's plot training set residuals

fig, ax = plt.subplots(nrows=1, ncols=2, figsize = (18,6))

residuals_train = y_train-y_train_pred2
ax1 = plt.subplot(121)
ax1.scatter(y_train_pred2, residuals_train)
ax1.plot(y_train_pred2, [0 for i in range(len(y_train_pred2))])
plt.title('Training Data')
plt.xlabel('price predictions')
plt.ylabel('error')

residuals_test = y_test-y_test_pred2
ax2 = plt.subplot(122)
```

```
ax2.scatter(y_test_pred2, residuals_test)
ax2.plot(y_test_pred2, [0 for i in range(len(y_test_pred2))])
plt.title('Test Data')
plt.xlabel('price predictions')
plt.ylabel('error');

# cone-shaped residuals indicate heteroskedasticity
# means that as price increases, error increases as well
# will try log transformations to reduce the effect of outliers
```



In [149]:

```
# run it in Statsmodels to check coefficients

model = sm.OLS(y_train, sm.add_constant(pd.DataFrame(X_train_scaled, columns=X_train.col
umns, index=X_train.index)))
results = model.fit()

results.summary()

# same R-squared as above
# sqft_lot and sqft_above have p-values above 0.05.  Experiment with removing these later
```

```
/opt/anaconda3/envs/learn-env/lib/python3.6/site-packages/numpy/core/fromnumeric.py:2580:
FutureWarning: Method .ptp is deprecated and will be removed in a future version. Use num
py.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

Out[149]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.635
Model:	OLS	**Adj. R-squared:**	0.635
Method:	Least Squares	**F-statistic:**	1872.
Date:	Sun, 31 Jan 2021	**Prob (F-statistic):**	0.00
Time:	11:57:17	**Log-Likelihood:**	-2.0537e+05
No. Observations:	15072	**AIC:**	4.108e+05
Df Residuals:	15057	**BIC:**	4.109e+05
Df Model:	14		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	5.342e+05	1631.123	327.481	0.000	5.31e+05	5.37e+05
bedrooms	-3.373e+04	2129.912	-15.834	0.000	-3.79e+04	-2.96e+04
bathrooms	3.44e+04	2978.015	11.551	0.000	2.86e+04	4.02e+04
sqft_living	9.403e+04	1.97e+04	4.765	0.000	5.53e+04	1.33e+05

sqft_lot	675.6564	2584.241	0.261	0.794	-4389.771	5741.084
floors	2.328e+04	2308.084	10.087	0.000	1.88e+04	2.78e+04
condition	1.622e+04	1816.099	8.931	0.000	1.27e+04	1.98e+04
grade	1.448e+05	2908.904	49.788	0.000	1.39e+05	1.51e+05
sqft_above	1.147e+04	1.79e+04	0.639	0.523	-2.37e+04	4.66e+04
sqft_basement	2.053e+04	9471.972	2.168	0.030	1966.229	3.91e+04
yr_built	-1.08e+05	2391.705	-45.136	0.000	-1.13e+05	-1.03e+05
zipcode	7009.9545	1851.959	3.785	0.000	3379.890	1.06e+04
sqft_living15	4.159e+04	2767.834	15.027	0.000	3.62e+04	4.7e+04
sqft_lot15	-1.154e+04	2584.213	-4.466	0.000	-1.66e+04	-6474.869
renovated	7959.0619	1718.238	4.632	0.000	4591.106	1.13e+04

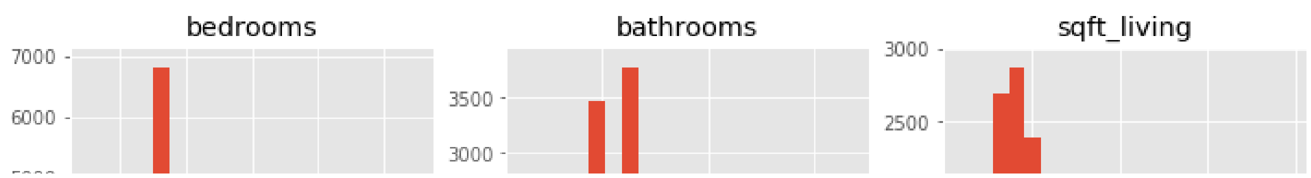
Omnibus:	7051.516	**Durbin-Watson:**	1.986
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	79640.454
Skew:	1.959	**Prob(JB):**	0.00
Kurtosis:	13.557	**Cond. No.**	38.6

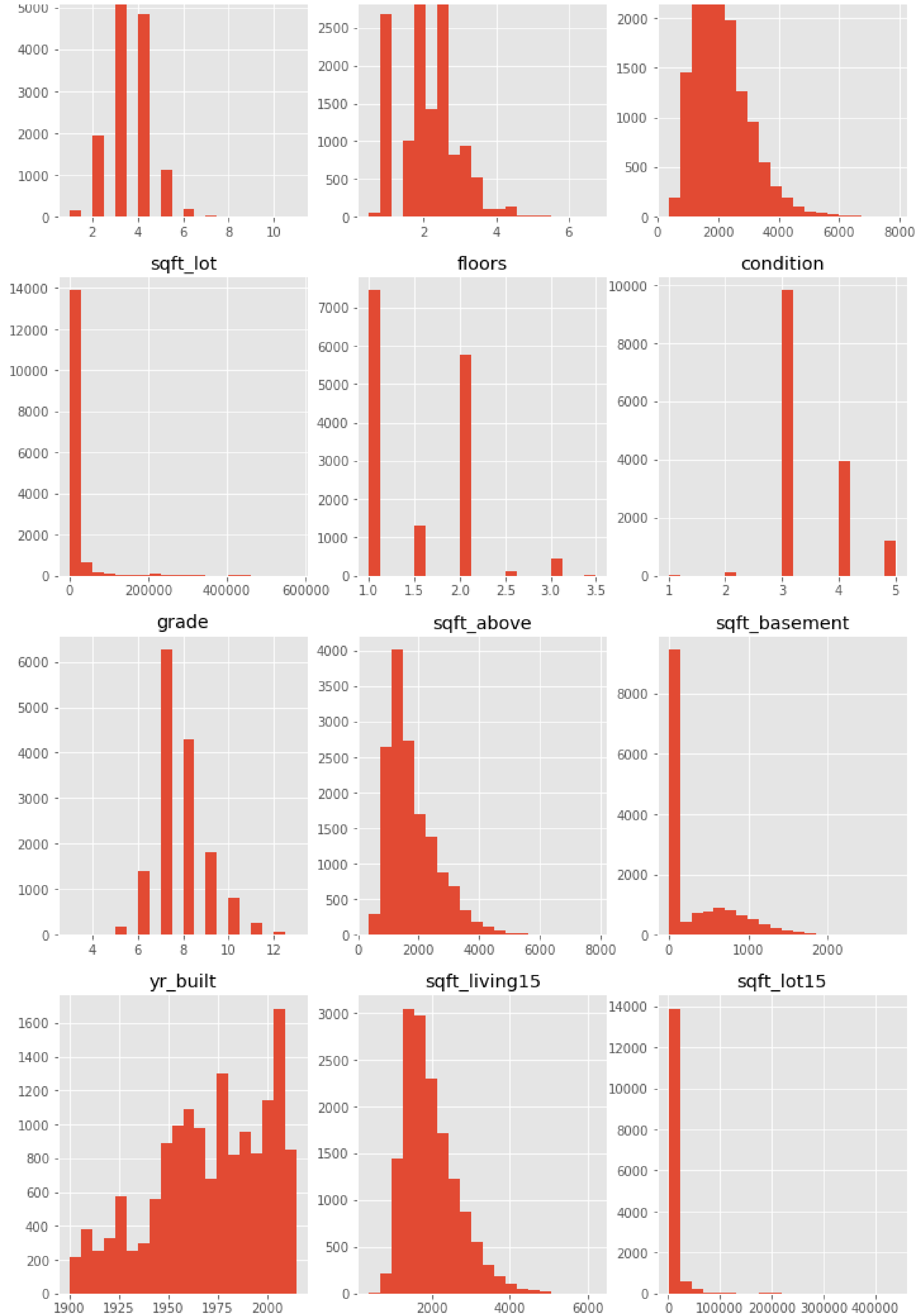
Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Model 3: 7 Log-Transformed X Variables

In [150]:

```
# let's look at the numeric variables.  Are they normally distributed?

numeric = ['bedrooms',
       'bathrooms',
       'sqft_living',
       'sqft_lot',
       'floors',
       'condition',
       'grade',
       'sqft_above',
       'sqft_basement',
       'yr_built',
       'sqft_living15',
       'sqft_lot15'
             ]

num_cols = 3
if len(numeric)%num_cols == 0:
    num_rows = len(numeric)//num_cols
else:
    num_rows =  (len(numeric)//num_cols)+1


fig, axs = plt.subplots(figsize=(12,20), nrows=num_rows, ncols=num_cols)


for feat in numeric:
    axs[numeric.index(feat)//num_cols, numeric.index(feat)%num_cols].hist(X_train[feat],
bins=20)
    axs[numeric.index(feat)//num_cols, numeric.index(feat)%num_cols].set_title(feat)
```





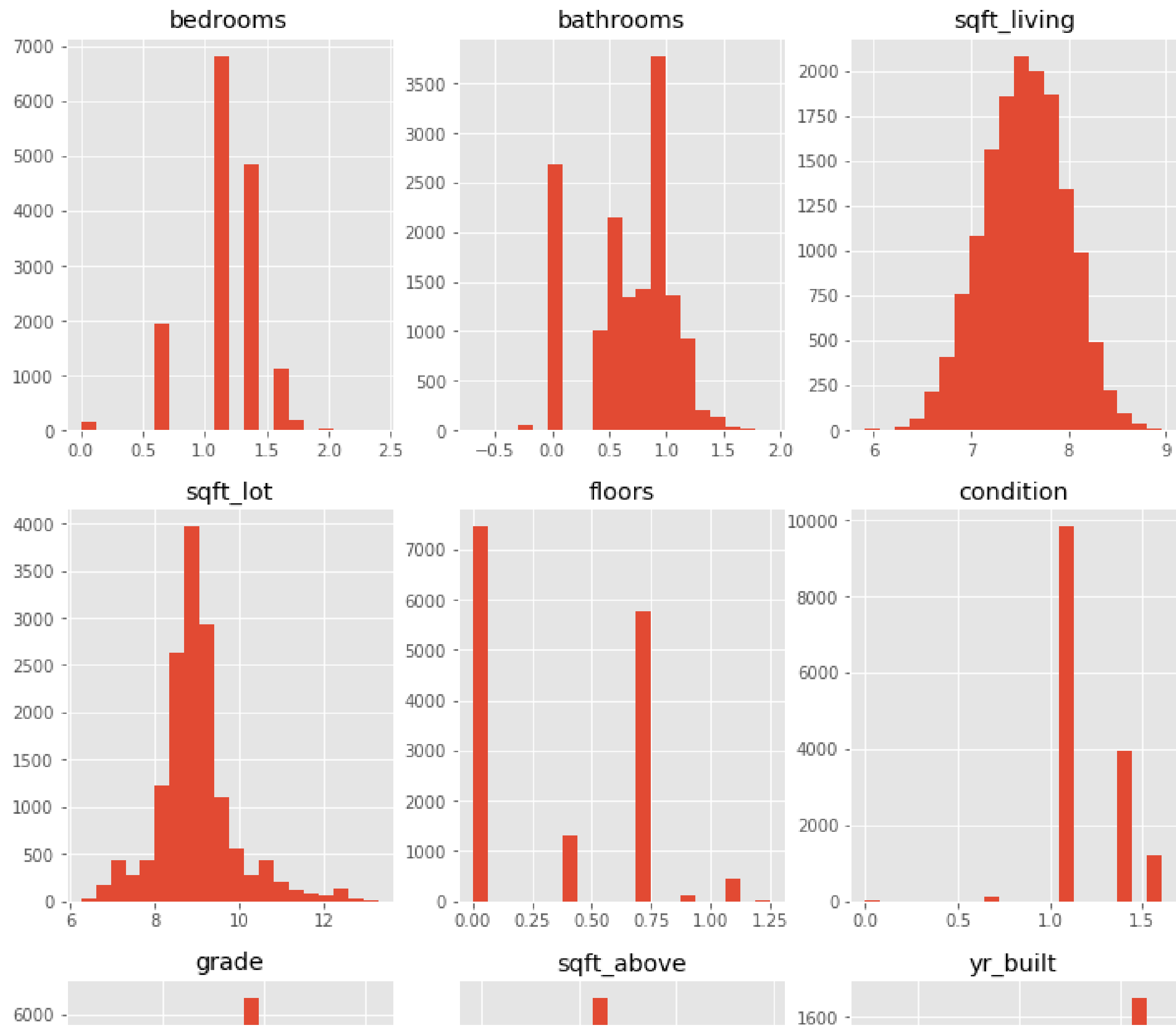
In [151]:

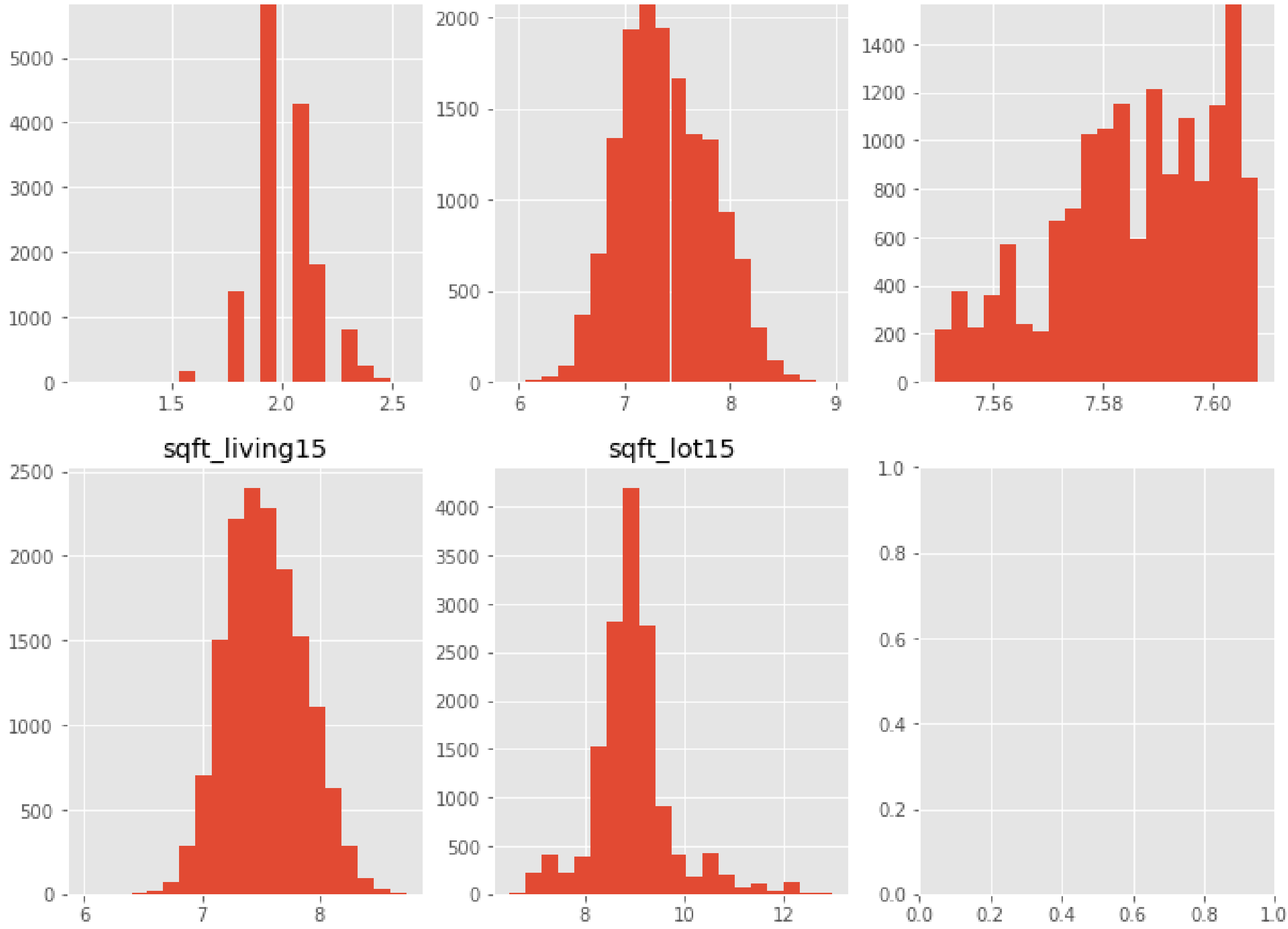
```
# numeric variables are not normally distributed
# let's try to log these and see if they become more normal
# don't include features with zeros, like sqft_basement

non_zero = ['bedrooms',
        'bathrooms',
```

```
        'sqft_living',
        'sqft_lot',
        'floors',
        'condition',
        'grade',
        'sqft_above',
        'yr_built',
        'sqft_living15',
        'sqft_lot15'
              ]

X_train_logged = X_train.copy()

for feat in non_zero:
    X_train_logged[feat] = X_train_logged[feat].map(lambda x: np.log(x))
```

In [152]:

```
# Did it help?  Make more histograms

num_cols = 3
if len(non_zero)%num_cols == 0:
    num_rows = len(non_zero)//num_cols
else:
    num_rows =  (len(non_zero)//num_cols)+1


fig, axs = plt.subplots(figsize=(12,20), nrows=num_rows, ncols=num_cols)


for feat in non_zero:
    axs[non_zero.index(feat)//num_cols, non_zero.index(feat)%num_cols].hist(X_train_logge
d[feat], bins=20)
    axs[non_zero.index(feat)//num_cols, non_zero.index(feat)%num_cols].set_title(feat)
```





In [153]:

```
# it helped - some variables look more normally distributed
# like sqft_living, sqft_lot, grade, sqft_above, sqft_living15, sqft_lot15
# and to a lesser extent, bedrooms and grade too
```

In [154]:

```
# what if we build a model with just the above columns logged

# build a new X_train with just the above features logged

to_log = ['bedrooms',
       'sqft_living',
       'sqft_lot',
       'grade',
       'sqft_above',
       'sqft_living15',
       'sqft_lot15'
             ]

X_train3 = X_train.copy()

for feat in to_log:
    X_train3[feat] = X_train3[feat].map(lambda x: np.log(x))

# log the test data

X_test3 = X_test.copy()

for feat in to_log:
    X_test3[feat] = X_test3[feat].map(lambda x: np.log(x))
```

In [155]:

```
# build a function to scale the X variables, and do a linear regression

def scale_lin_reg(X_train, y_train, X_test):
```

```
    """Perform standard scaling and linear regression given training set and test set X-v
ariables

    Parameters
    ----------
    X_train: DataFrame of training set input variables
    y_train: Array of true values from the training set target variable
    X_test: DataFrame of test set input variables

    Returns
    -------
    y_train_pred: Series of training set target variable predictions
    y_test_pred: Series of test set target variable predictions
    """

    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    linreg = LinearRegression()
    linreg.fit(X_train_scaled, y_train)

    y_train_pred = linreg.predict(X_train_scaled)
    y_test_pred = linreg.predict(X_test_scaled)

    return(y_train_pred, y_test_pred)
```

In [156]:

```
# let's scale and do a linear regression on the transformed data, to return y_train_pred
and y_test_pred
# the inputs X_train3 and X_test3 have 7 features logged

y_train_pred3, y_test_pred3 = scale_lin_reg(X_train=X_train3, y_train=y_train, X_test=X_
test3)
```

In [157]:

```
# now let's evaluate that model
evaluate_model(y_train=y_train, y_train_pred=y_train_pred3, y_test=y_test, y_test_pred=y
_test_pred3)
```

```
Training Data
 R-Squared: 0.604166
 Mean Absolute Error: 140699.83
 Root Mean Squared Error: 208479.59

Testing Data
 R-Squared: 0.584709
 Mean Absolute Error: 140646.01
 Root Mean Squared Error: 203201.36
```

In [158]:

```
best_r2

# oh no!  It didn't help!  R2 got worse
```

Out[158]:

```
{'train': 0.635164, 'test': 0.618171}
```

Model 4: 2 Logged X Variables

In [159]:

```
# these two features have the most improvement in normality after log transformations:
# sqft_living15
# sqft_lot15
```

```
# what if we just log these?

to_log = [
       'sqft_living15',
       'sqft_lot15'
             ]

X_train4 = X_train.copy()

for feat in to_log:
    X_train4[feat] = X_train4[feat].map(lambda x: np.log(x))

# log the test data

X_test4 = X_test.copy()

for feat in to_log:
    X_test4[feat] = X_test4[feat].map(lambda x: np.log(x))
```

In [160]:

```
# let's scale and do a linear regression on the transformed data, to return y_train_pred
and y_test_pred
# the inputs X_train4 and X_test4 have 2 features logged

y_train_pred4, y_test_pred4 = scale_lin_reg(X_train=X_train4, y_train=y_train, X_test=X_
test4)
```

In [161]:

```
# evaluate the model

evaluate_model(y_train=y_train, y_train_pred=y_train_pred4, y_test=y_test, y_test_pred=y
_test_pred4)
```

```
Training Data
 R-Squared: 0.635315
 Mean Absolute Error: 134720.89
 Root Mean Squared Error: 200108.55

Testing Data
 R-Squared: 0.62024
 Mean Absolute Error: 134413.73
 Root Mean Squared Error: 194314.5
```

In [162]:

```
best_r2

# That is a very slight improvement over the baseline model
# Price (target variable) was also right-skewed.  Let's try logging this as well.
```

Out[162]:

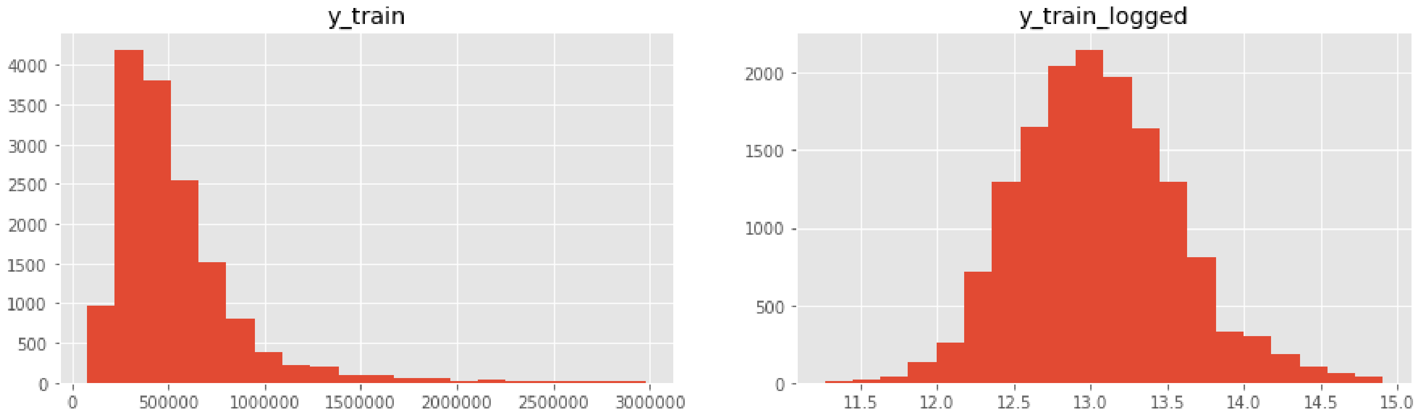
```
{'train': 0.635164, 'test': 0.618171}
```

Model 5: Two Logged X Variables and Logged Target Variable

In [163]:

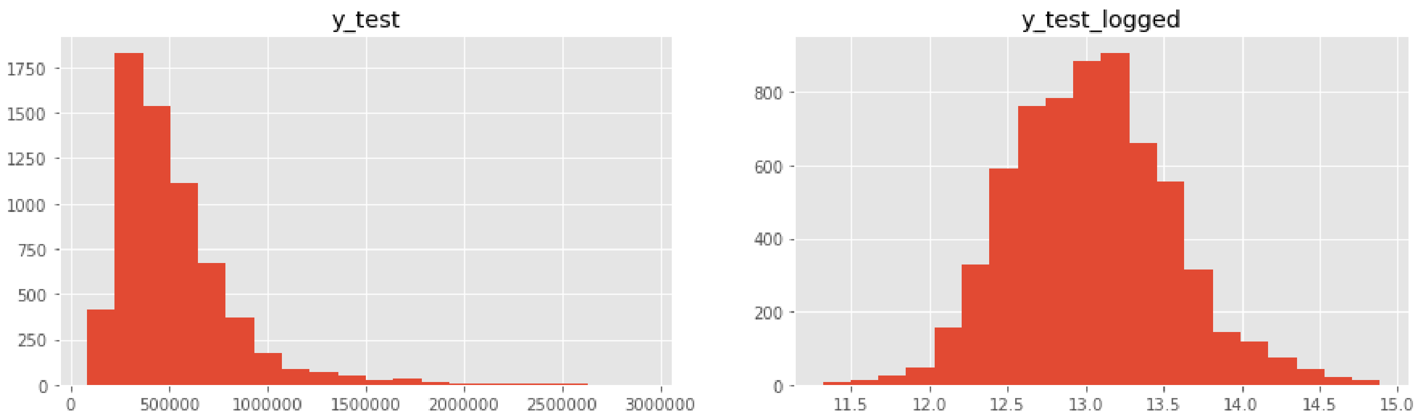
```
y_train_logged = y_train.copy()
y_train_logged = y_train_logged.map(lambda y: np.log1p(y))

fig, ax = plt.subplots(figsize = (15,4), nrows=1, ncols=2)

ax[0].hist(y_train, bins=20)
ax[0].set_title('y_train')

ax[1].hist(y_train_logged, bins=20)
ax[1].set_title('y_train_logged');
```

```
# the logged y_train definitely looks more normally distributed
```



In [164]:

```
# log y_test as well

y_test_logged = y_test.copy()
y_test_logged = y_test_logged.map(lambda y: np.log1p(y))

fig, ax = plt.subplots(figsize = (15,4), nrows=1, ncols=2)

ax[0].hist(y_test, bins=20)
ax[0].set_title('y_test')

ax[1].hist(y_test_logged, bins=20)
ax[1].set_title('y_test_logged');

# the logged y_test looks more normally distributed too
```



In [165]:

```
# now let's run and evaluate the model with X_train4, X_test4, y_train_logged and y_test_
logged

y_train_pred5, y_test_pred5 = scale_lin_reg(X_train=X_train4, y_train=y_train_logged, X_t
est=X_test4)
```

In [166]:

```
# create a function to only evaluate R-squared, since MAE and RMSE must use unlogged pric
e predictions

def eval_r2(y_train, y_train_pred, y_test, y_test_pred):
    """Evalute R-Squared for training and test predictions

    Parameters
    ----------
    y_train: Array of true values from the training set target variable
    y_train_pred: Array of target variable values predicted by the model for the training
set
    y_test: Array of true values from the test set target variable
```

```
    y_test_pred: Array of target variable values predicted by the model for the test set

    Returns
    -------
    Print of R-Squared for training and test sets"""

    # calculate r2 using logged target variable
    r2_train = round(r2_score(y_true=y_train, y_pred=y_train_pred), 6)
    r2_test = round(r2_score(y_true=y_test, y_pred=y_test_pred), 6)

    print('Training Data', '\n',
          'R-Squared:', r2_train, '\n')

    print('Test Data', '\n',
          'R-Squared:', r2_test)
```

In [167]:

```
eval_r2(y_train=y_train_logged, y_train_pred=y_train_pred5, y_test=y_test_logged, y_test
_pred=y_test_pred5)
```

```
Training Data
 R-Squared: 0.656332

Test Data
 R-Squared: 0.62867
```

In [168]:

```
# great, it helped a little, but need to unlog y_train_pred5 and y_test_pred5 to measure
price errors

# create a function to unlog predictions and measure MAE and RMSE
def unlog_MAE_RMSE(y_train, y_train_logged_pred, y_test, y_test_logged_pred):

    """Unlog target variable values, and evaluate Mean Absolute Error and Root Mean Squar
ed Error for training and test predictions

    Parameters
    ----------
    y_train: Series of true values from the training set target variable
    y_train_logged_pred: Series of target variable values predicted using a logged targe
t variable; training set
    y_test: Series of true values from the test set target variable
    y_test_pred: Series of target variable values predicted using a logged target variabl
e; test set

    Returns
    -------
    Print of Mean Absolute Error and Root Mean Squared Error for training and test sets""
"

    # unlog target variable predictions to measure MAE and RMSE
    y_train_pred_exp = np.expm1(y_train_logged_pred)
    y_test_pred_exp = np.expm1(y_test_logged_pred)

    # check Mean Absolute Error
    mae_train = round(mean_absolute_error(y_true=y_train, y_pred=y_train_pred_exp), 2)
    mae_test = round(mean_absolute_error(y_true=y_test, y_pred=y_test_pred_exp), 2)

    # check Root Mean Squared Error
    rmse_train = round(np.sqrt(mean_squared_error(y_true=y_train, y_pred=y_train_pred_ex
p)), 2)
    rmse_test = round(np.sqrt(mean_squared_error(y_true=y_test, y_pred=y_test_pred_exp))
, 2)

    print('Training Data', '\n',
          'Mean Absolute Error:', mae_train, '\n',
          'Root Mean Squared Error:', rmse_train, '\n')

    print('Test Data', '\n',
```

```
          'Mean Absolute Error:', mae_test, '\n',
          'Root Mean Squared Error:', rmse_test, '\n')
```

In [169]:

```
unlog_MAE_RMSE(y_train=y_train, y_train_logged_pred=y_train_pred5, y_test=y_test, y_test
_logged_pred=y_test_pred5)
```

```
Training Data
 Mean Absolute Error: 126748.86
 Root Mean Squared Error: 196464.59

Test Data
 Mean Absolute Error: 128505.75
 Root Mean Squared Error: 195365.54
```

In [170]:

```
# update best_r2

best_r2['train'] = 0.656332
best_r2['test'] = 0.62867
```

Model 6: Two Logged X Variables, Logged Target Variable, and Zip Code Categories

In [171]:

```
# let's try to assign zip codes to price categories

X_train6 = X_train4.copy() # use X_train4, which had two features logged

X_train6.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15072 entries, 753 to 15795
Data columns (total 14 columns):
bedrooms         15072 non-null int64
bathrooms        15072 non-null float64
sqft_living      15072 non-null int64
sqft_lot         15072 non-null int64
floors           15072 non-null float64
condition        15072 non-null int64
grade            15072 non-null int64
sqft_above       15072 non-null int64
sqft_basement    15072 non-null float64
yr_built         15072 non-null int64
zipcode          15072 non-null int64
sqft_living15    15072 non-null float64
sqft_lot15       15072 non-null float64
renovated        15072 non-null int64
dtypes: float64(5), int64(9)
memory usage: 1.7 MB
```

In [172]:

```
X_train6['zipcode'].value_counts().count() # 70 different zips
X_train6['zipcode'].value_counts()
```

Out[172]:

```
98103    441
98052    418
98115    402
98038    401
98034    392
98117    380
98042    372
98133    361
98023    361
98118    334
```

```
98006    329
98059    321
98058    313
98155    306
98027    305
98074    301
98125    294
98033    293
98056    279
98053    271
98075    269
98001    255
98126    241
98092    235
98106    232
98116    229
98144    229
98199    229
98029    220
98065    217
98004    207
98122    205
98055    201
98198    196
98031    195
98112    195
98008    195
98072    194
98003    194
98028    193
98168    190
98040    189
98178    189
98166    187
98146    186
98136    184
98177    180
98107    178
98030    171
98105    162
98045    161
98022    152
98108    142
98077    138
98002    133
98011    133
98019    131
98119    130
98005    119
98188    102
98007     97
98014     92
98032     82
98010     78
98070     77
98102     76
98109     71
98024     59
98148     44
98039     34
Name: zipcode, dtype: int64
```

In [173]:

```
zips=pd.concat([X_train6['zipcode'], pd.DataFrame(y_train)['price']], axis=1)
zips
```

Out[173]:

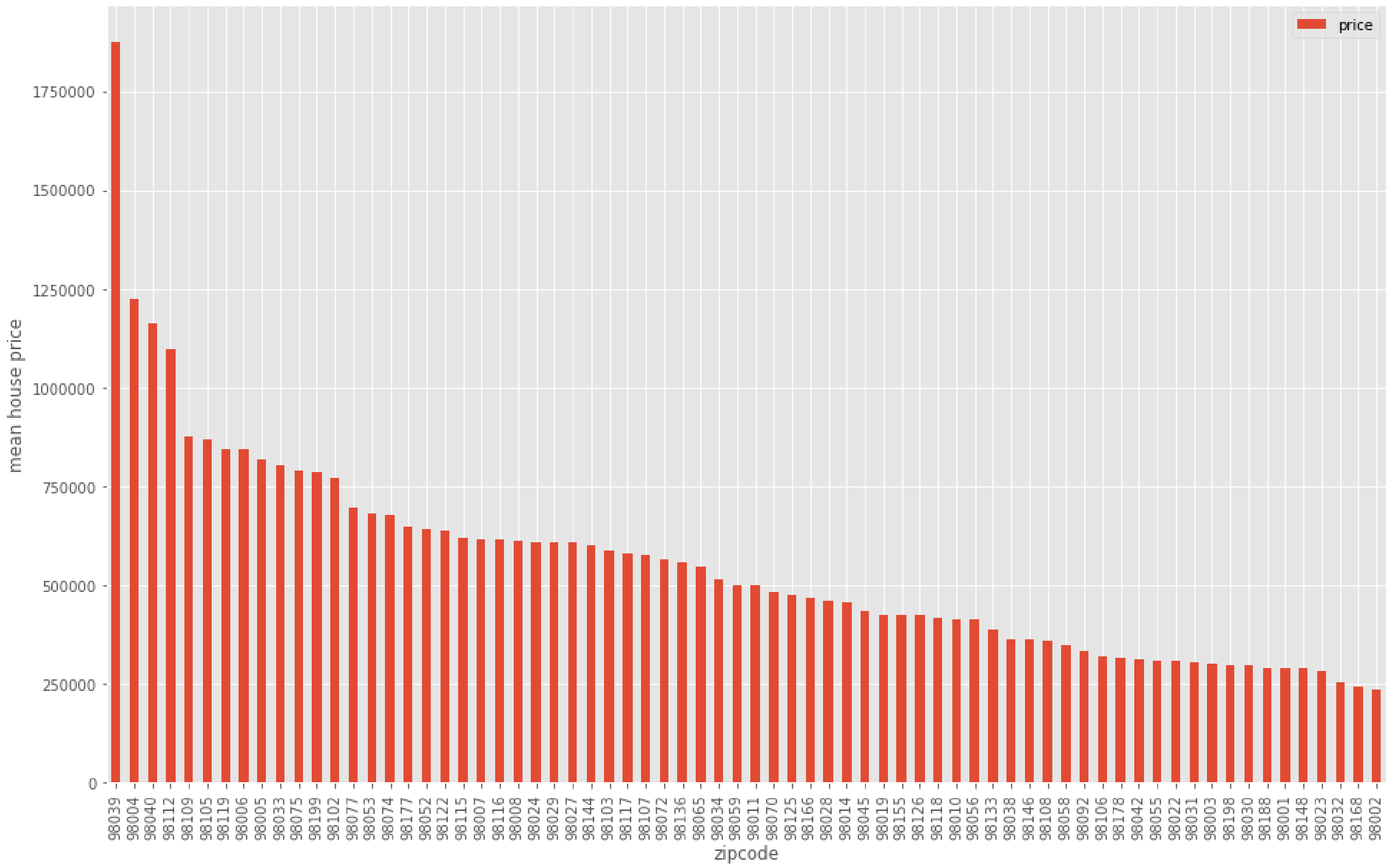
	zipcode	price
753	98053	699800.00000

	zipcode	price
1418	98178	1700000.00000
8178	98003	258000.00000
2254	98022	245000.00000
4063	98055	373000.00000
...	...	...
11964	98065	440000.00000
21575	98178	572000.00000
5390	98014	299800.00000
860	98168	245000.00000
15795	98019	545000.00000

15072 rows × 2 columns

In [174]:

```
# find mean price by zip to see if any stand out

zips_pivot = zips.pivot_table(values='price', index='zipcode', ).sort_values(by='price', ascending=False)
zips_pivot.plot(kind='bar', figsize=(16,10))
plt.ylabel('mean house price')
plt.legend;

plt.savefig('price_by_zip_code')

# yes, some do stand out!  the top four, the bottom three
# what if I classified them based on price? I can make a dictionary
```



In [175]:

```
# create a dictionary of zip codes and classifications

ordered_zip_list = list(zips_pivot.index)
zip_dict = {}
```

```python
# display all zips and index in price-ordered list for eyeballing

for i in ordered_zip_list:
    print(i, ordered_zip_list.index(i))
```

```
98039 0
98004 1
98040 2
98112 3
98109 4
98105 5
98119 6
98006 7
98005 8
98033 9
98075 10
98199 11
98102 12
98077 13
98053 14
98074 15
98177 16
98052 17
98122 18
98115 19
98007 20
98116 21
98008 22
98024 23
98029 24
98027 25
98144 26
98103 27
98117 28
98107 29
98072 30
98136 31
98065 32
98034 33
98059 34
98011 35
98070 36
98125 37
98166 38
98028 39
98014 40
98045 41
98019 42
98155 43
98126 44
98118 45
98010 46
98056 47
98133 48
98038 49
98146 50
98108 51
98058 52
98092 53
98106 54
98178 55
98042 56
98055 57
98022 58
98031 59
98003 60
98198 61
98030 62
98188 63
98001 64
98148 65
98023 66
```

```
98032 67
98168 68
98002 69
```

In [176]:

```python
# classify zips in dict:

zip_dict[ordered_zip_list[0]] = 'Zip Class 1'

# make a function to add entries more easily
def add_to_zip_dict(list_index_start, list_index_stop, category):

    """Add entries to zip_dict based on their index in ordered_zip_list.

    Parameters
    ----------
    list_index_start: index in ordered_zip_list of the first zip code to enter
    list_index_stop: index in ordered_zip_list of the zip code to stop at
    category: zip class to assign to these entries"""

    for i in ordered_zip_list[list_index_start:list_index_stop]:
        zip_dict[i] = category

add_to_zip_dict(1, 4, 'Zip Class 2')
add_to_zip_dict(4, 13, 'Zip Class 3')
add_to_zip_dict(13, 34, 'Zip Class 4')
add_to_zip_dict(34, 49, 'Zip Class 5')
add_to_zip_dict(49, 67, 'Zip Class 6')
add_to_zip_dict(67, 70, 'Zip Class 7')

zip_dict
```

Out[176]:

```
{98039: 'Zip Class 1',
 98004: 'Zip Class 2',
 98040: 'Zip Class 2',
 98112: 'Zip Class 2',
 98109: 'Zip Class 3',
 98105: 'Zip Class 3',
 98119: 'Zip Class 3',
 98006: 'Zip Class 3',
 98005: 'Zip Class 3',
 98033: 'Zip Class 3',
 98075: 'Zip Class 3',
 98199: 'Zip Class 3',
 98102: 'Zip Class 3',
 98077: 'Zip Class 4',
 98053: 'Zip Class 4',
 98074: 'Zip Class 4',
 98177: 'Zip Class 4',
 98052: 'Zip Class 4',
 98122: 'Zip Class 4',
 98115: 'Zip Class 4',
 98007: 'Zip Class 4',
 98116: 'Zip Class 4',
 98008: 'Zip Class 4',
 98024: 'Zip Class 4',
 98029: 'Zip Class 4',
 98027: 'Zip Class 4',
 98144: 'Zip Class 4',
 98103: 'Zip Class 4',
 98117: 'Zip Class 4',
 98107: 'Zip Class 4',
 98072: 'Zip Class 4',
 98136: 'Zip Class 4',
 98065: 'Zip Class 4',
 98034: 'Zip Class 4',
 98059: 'Zip Class 5',
 98011: 'Zip Class 5',
 98070: 'Zip Class 5',
 98125: 'Zip Class 5',
```

```
 98125: 'Zip Class 5',
 98166: 'Zip Class 5',
 98028: 'Zip Class 5',
 98014: 'Zip Class 5',
 98045: 'Zip Class 5',
 98019: 'Zip Class 5',
 98155: 'Zip Class 5',
 98126: 'Zip Class 5',
 98118: 'Zip Class 5',
 98010: 'Zip Class 5',
 98056: 'Zip Class 5',
 98133: 'Zip Class 5',
 98038: 'Zip Class 6',
 98146: 'Zip Class 6',
 98108: 'Zip Class 6',
 98058: 'Zip Class 6',
 98092: 'Zip Class 6',
 98106: 'Zip Class 6',
 98178: 'Zip Class 6',
 98042: 'Zip Class 6',
 98055: 'Zip Class 6',
 98022: 'Zip Class 6',
 98031: 'Zip Class 6',
 98003: 'Zip Class 6',
 98198: 'Zip Class 6',
 98030: 'Zip Class 6',
 98188: 'Zip Class 6',
 98001: 'Zip Class 6',
 98148: 'Zip Class 6',
 98023: 'Zip Class 6',
 98032: 'Zip Class 7',
 98168: 'Zip Class 7',
 98002: 'Zip Class 7'}
```

In [177]:

```
# add classification column to training data

X_train6['zip_class'] = X_train6['zipcode'].map(lambda x: zip_dict[x])
X_train6
```

Out[177]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	condition	grade	sqft_above	sqft_basement	yr_built	zipcode	sq
753	2	2.50000	2380	6600	1.00000	3	8	2380	0.00000	2010	98053	
1418	4	3.75000	3190	17186	2.00000	3	10	3190	0.00000	1999	98178	
8178	3	2.50000	1730	6930	2.00000	3	8	1730	0.00000	1994	98003	
2254	4	2.00000	1870	8750	1.00000	3	7	1870	0.00000	1977	98022	
4063	8	3.00000	2850	12714	1.00000	3	7	2850	0.00000	1959	98055	
...	...	...	...	...	...	...	...	...	...	...	...	
11964	3	2.50000	2230	5800	2.00000	3	7	2230	0.00000	2004	98065	
21575	4	2.75000	2770	3852	2.00000	3	8	2770	0.00000	2014	98178	
5390	4	1.50000	1530	9000	1.00000	4	6	1530	0.00000	1976	98014	
860	1	0.75000	380	15000	1.00000	3	5	380	0.00000	1963	98168	
15795	4	2.50000	2755	11612	2.00000	3	8	2755	0.00000	2001	98019	

15072 rows × 15 columns

In [178]:

```
# one hot encode classification column and drop zipcode and zip_class columns

zip_class_columns = pd.get_dummies(X_train6['zip_class'], drop_first=True)
zip_class_columns

X_train6 = pd.concat([X_train6, zip_class_columns], axis=1)
X_train6.drop(columns=['zipcode','zip_class'], inplace=True)
```

In [179]:

```
# add same features to test set

X_test6 = X_test4
X_test6['zip_class'] = X_test6['zipcode'].map(lambda x: zip_dict[x])

zip_class_columns = pd.get_dummies(X_test6['zip_class'], drop_first=True)
X_test6 = pd.concat([X_test6, zip_class_columns], axis=1)
X_test6.drop(columns=['zipcode','zip_class'], inplace=True)
```

In [180]:

```
X_train6 #looks good
X_test6 #looks good
```

Out[180]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	condition	grade	sqft_above	sqft_basement	yr_built	sqft_living15
3686	3	0.75000	850	8573	1.00000	3	6	600	250.00000	1945	6.74524
10247	3	1.00000	1510	6083	1.00000	4	6	860	650.00000	1940	7.31986
4037	4	2.25000	1790	42000	1.00000	3	7	1170	620.00000	1983	7.63046
3437	2	1.50000	1140	2500	1.00000	3	7	630	510.00000	1988	7.31322
19291	3	1.00000	1500	3920	1.00000	3	7	1000	500.00000	1947	7.40245
...	...	...	...	...	...	...	...	...	...	...	...
9400	4	3.50000	2650	3060	2.00000	3	9	2060	590.00000	2001	7.29302
9092	4	2.75000	2670	6780	2.00000	5	8	1630	1040.00000	1908	7.78322
6650	3	1.75000	1600	10280	1.00000	3	7	1050	550.00000	1977	7.37149
21095	5	3.50000	2760	3865	2.50000	3	8	2760	0.00000	2013	7.85941
3372	2	1.75000	1060	16470	1.00000	3	7	1060	0.00000	1977	7.48997

6458 rows × 19 columns

In [181]:

```
# generate predictions

y_train_pred6, y_test_pred6 = scale_lin_reg(X_train=X_train6, y_train=y_train_logged, X_t
est=X_test6)
```

In [182]:

```
# evaluate model using R-squared

eval_r2(y_train=y_train_logged, y_train_pred=y_train_pred6, y_test=y_test_logged, y_test
_pred=y_test_pred6)
```

Training Data

```
Training Data
 R-Squared: 0.831058

Test Data
 R-Squared: 0.825734
```

In [183]:

```
# to evaluate MAE and RMSE, unlog y_train_pred6 and y_test_pred6

unlog_MAE_RMSE(y_train=y_train, y_train_logged_pred=y_train_pred6, y_test=y_test, y_test
_logged_pred=y_test_pred6)
```

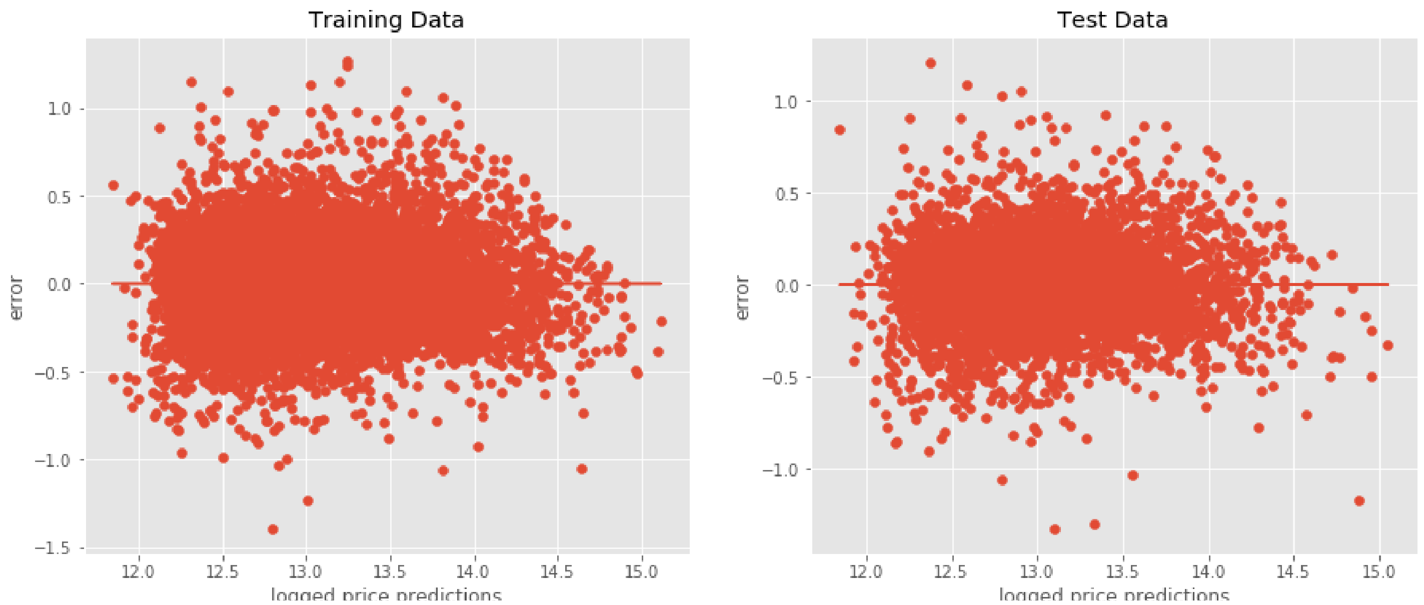
```
Training Data
 Mean Absolute Error: 88248.13
 Root Mean Squared Error: 151546.53

Test Data
 Mean Absolute Error: 86914.78
 Root Mean Squared Error: 144785.38
```

In [184]:

```
# great, this really helped!
# update best_r2

best_r2 = {'train': 0.831058, 'test': 0.825734}
```

In [185]:

```
# let's look at the training set residuals:

fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(15,6))

residuals_train6 = y_train_logged-y_train_pred6
ax1 = plt.subplot(121)
plt.scatter(y_train_pred6, residuals_train6)
plt.plot(y_train_pred6, [0 for i in range(len(y_train_pred6))])
plt.title('Training Data')
plt.xlabel('logged price predictions')
plt.ylabel('error')

residuals_test6 = y_test_logged-y_test_pred6
ax2 = plt.subplot(122)
plt.scatter(y_test_pred6, residuals_test6)
plt.plot(y_test_pred6, [0 for i in range(len(y_test_pred6))])
plt.title('Test Data')
plt.xlabel('logged price predictions')
plt.ylabel('error');

# looks better than the cone shape
```



logged price predictions logged price predictions

In [186]:

```
# look at coefficients

model = sm.OLS(y_train_logged, sm.add_constant(pd.DataFrame(X_train6, columns=X_train6.c
olumns, index=X_train6.index)))
results = model.fit()

results.summary()

# sqft_above and sqft_basement have high p-values
# experiment with removing these later
```

```
/opt/anaconda3/envs/learn-env/lib/python3.6/site-packages/numpy/core/fromnumeric.py:2580:
FutureWarning: Method .ptp is deprecated and will be removed in a future version. Use num
py.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

Out[186]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.831
Model:	OLS	**Adj. R-squared:**	0.831
Method:	Least Squares	**F-statistic:**	3897.
Date:	Sun, 31 Jan 2021	**Prob (F-statistic):**	0.00
Time:	11:57:24	**Log-Likelihood:**	1910.3
No. Observations:	15072	**AIC:**	-3781.
Df Residuals:	15052	**BIC:**	-3628.
Df Model:	19		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	16.9820	0.174	97.653	0.000	16.641	17.323
bedrooms	-0.0149	0.003	-5.956	0.000	-0.020	-0.010
bathrooms	0.0548	0.004	13.012	0.000	0.047	0.063
sqft_living	0.0002	2.35e-05	7.555	0.000	0.000	0.000
sqft_lot	8.706e-07	7.08e-08	12.305	0.000	7.32e-07	1.01e-06
floors	0.0253	0.005	5.090	0.000	0.016	0.035
condition	0.0416	0.003	14.037	0.000	0.036	0.047
grade	0.1188	0.003	43.040	0.000	0.113	0.124
sqft_above	-2.888e-06	2.34e-05	-0.123	0.902	-4.88e-05	4.3e-05
sqft_basement	6.476e-06	2.33e-05	0.278	0.781	-3.91e-05	5.21e-05
yr_built	-0.0028	8.6e-05	-33.017	0.000	-0.003	-0.003
sqft_living15	0.1610	0.009	18.181	0.000	0.144	0.178
sqft_lot15	-0.0351	0.003	-10.077	0.000	-0.042	-0.028
renovated	0.0746	0.010	7.364	0.000	0.055	0.094
Zip Class 2	-0.2860	0.038	-7.580	0.000	-0.360	-0.212
Zip Class 3	-0.4974	0.037	-13.405	0.000	-0.570	-0.425
Zip Class 4	-0.6171	0.037	-16.706	0.000	-0.689	-0.545
Zip Class 5	-0.8223	0.037	-22.183	0.000	-0.895	-0.750
Zip Class 6	-1.1048	0.037	-29.815	0.000	-1.177	-1.032
Zip Class 7	-1.2159	0.038	-31.593	0.000	-1.291	-1.140

Omnibus:	832.303	**Durbin-Watson:**	2.019
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	3064.888
Skew:	0.156	**Prob(JB):**	0.00
Kurtosis:	5.187	**Cond. No.**	3.65e+06

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.65e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

In [187]:

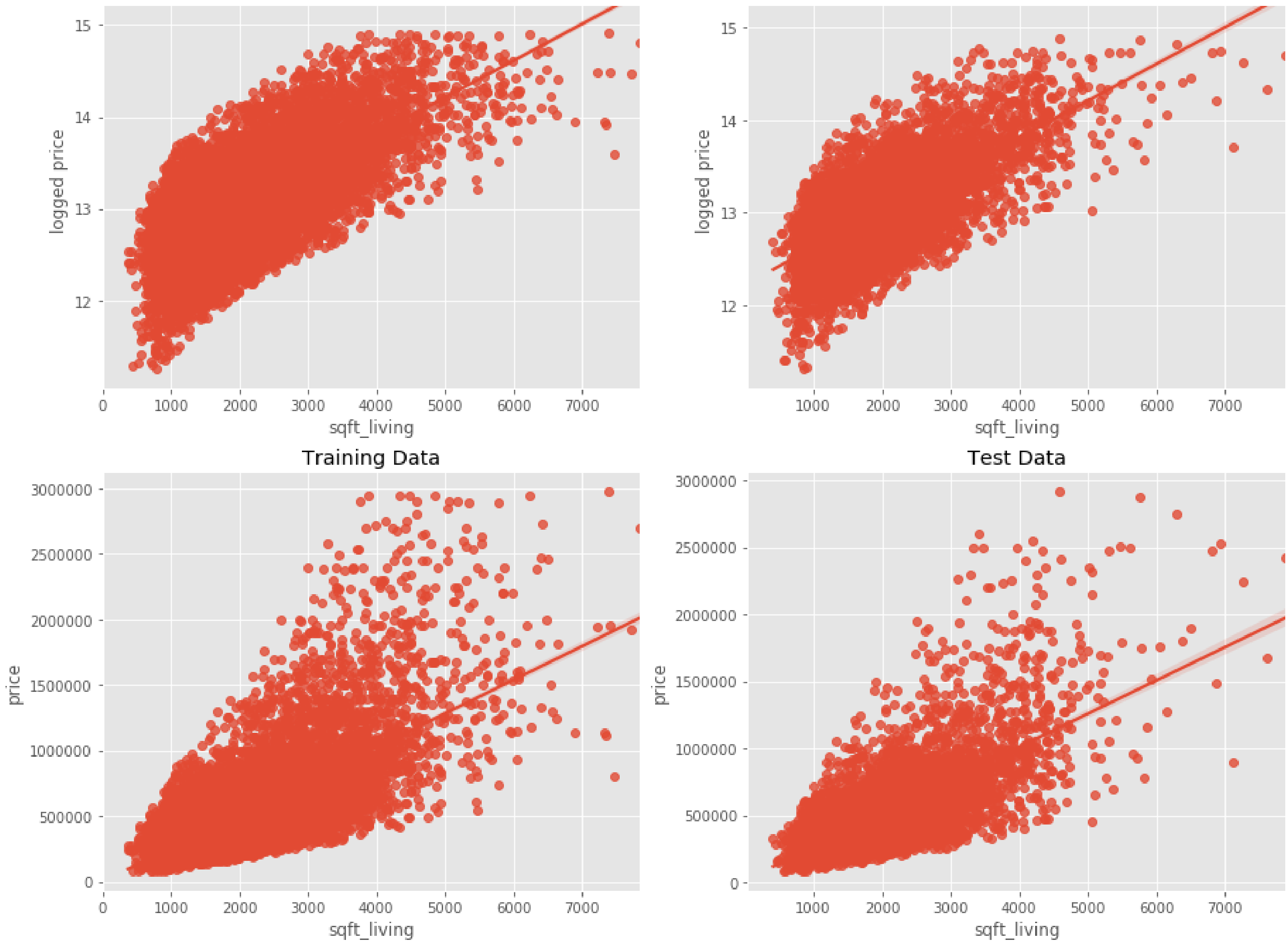
```
# which coefficients are most closely correlated with price?
X_train6.corrwith(y_train_logged).sort_values(ascending=False)

# grade and sqft_living, as in the original data
```

Out[187]:

```
grade             0.69756
sqft_living       0.68763
sqft_living15     0.60771
sqft_above        0.59249
bathrooms         0.54577
bedrooms          0.34674
Zip Class 3       0.33063
Zip Class 2       0.32369
floors            0.30996
sqft_basement     0.29778
Zip Class 4       0.28168
sqft_lot15        0.12260
sqft_lot          0.11223
renovated         0.10675
yr_built          0.07036
condition         0.04354
Zip Class 5      -0.12547
Zip Class 7      -0.21902
Zip Class 6      -0.50094
dtype: float64
```

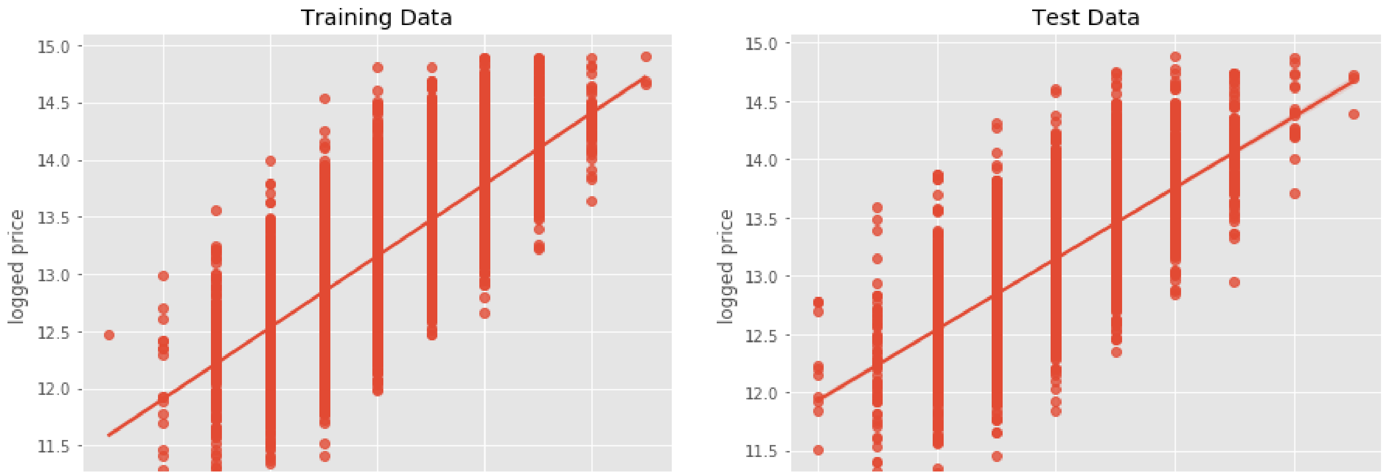
In [188]:

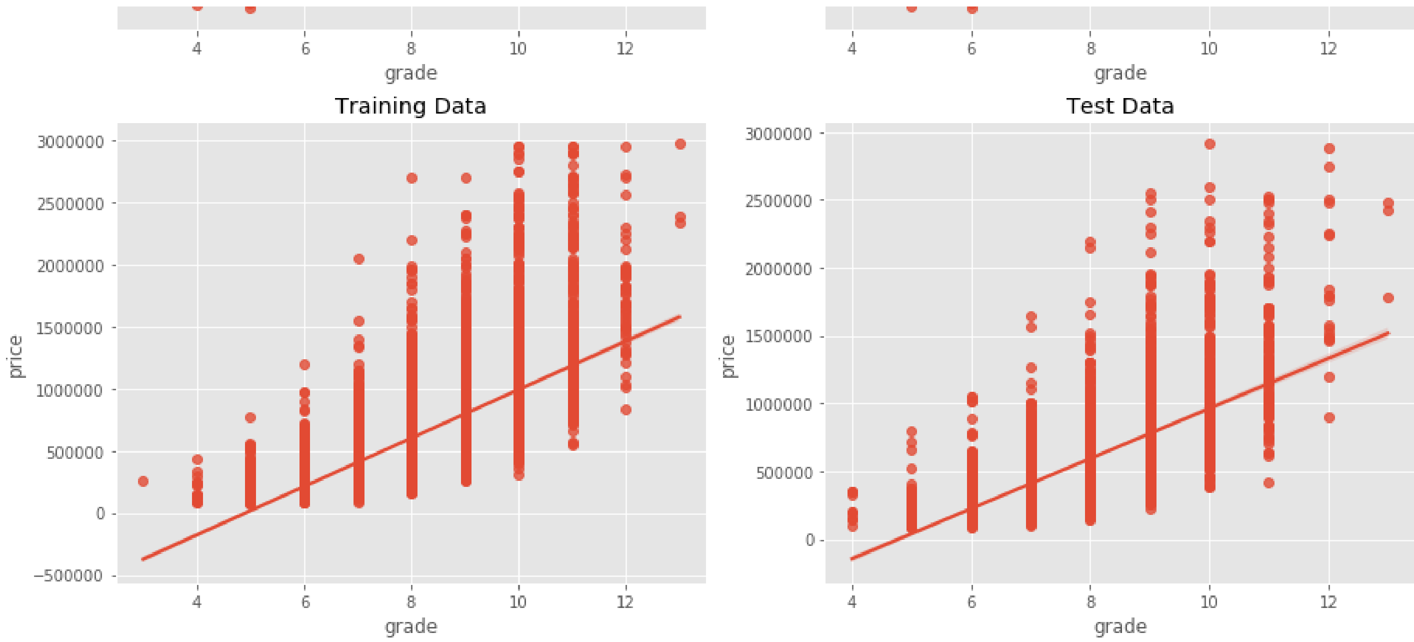
```
# plot coefficients that are closely correlated with price, for presentation to non-techn
ical stakeholders

plt.subplots(figsize=(15,12));

ax1 = plt.subplot(221)
sns.regplot(X_train6['sqft_living'], y_train_logged)
plt.title('Training Data')
plt.ylabel('logged price')

ax2 = plt.subplot(222)
sns.regplot(X_test6['sqft_living'], y_test_logged)
plt.title('Test Data')
plt.ylabel('logged price')

ax3 = plt.subplot(223)
sns.regplot(X_train6['sqft_living'], y_train)
plt.title('Training Data')

ax4 = plt.subplot(224)
sns.regplot(X_test6['sqft_living'], y_test)
plt.title('Test Data');
```



In [189]:

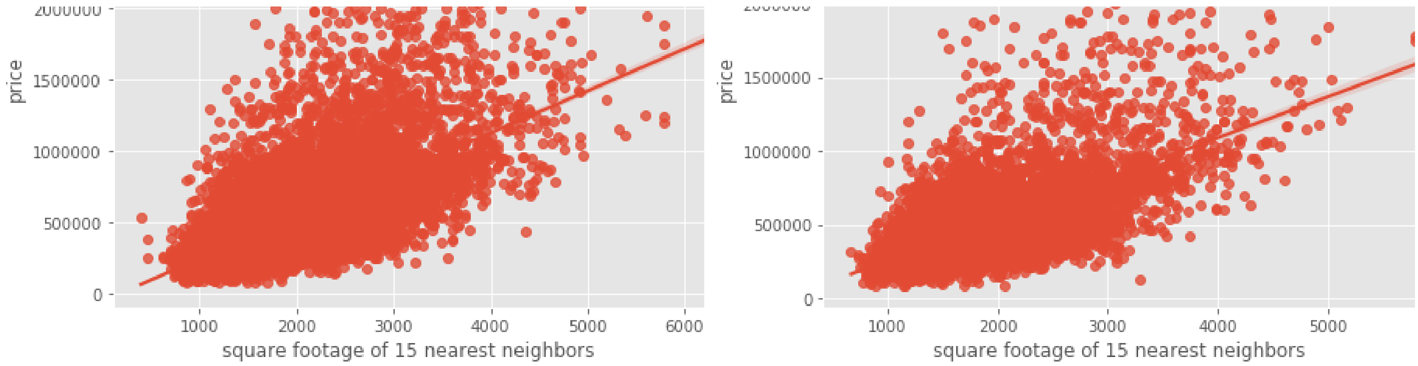
```
plt.subplots(figsize=(15,12));

ax1 = plt.subplot(221)
sns.regplot(X_train6['grade'], y_train_logged)
plt.title('Training Data')
plt.ylabel('logged price')

ax2 = plt.subplot(222)
sns.regplot(X_test6['grade'], y_test_logged)
plt.title('Test Data')
plt.ylabel('logged price')

ax3 = plt.subplot(223)
sns.regplot(X_train6['grade'], y_train)
plt.title('Training Data')

ax4 = plt.subplot(224)
sns.regplot(X_test6['grade'], y_test)
plt.title('Test Data');
```





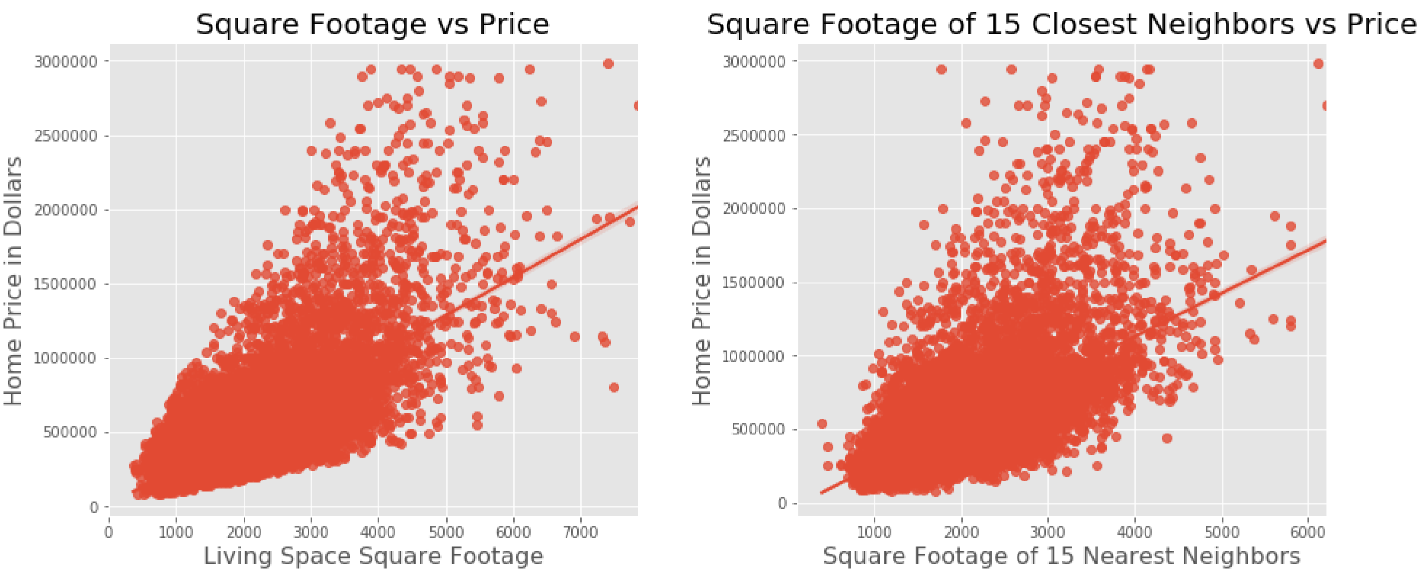
In [190]:

```
plt.subplots(figsize=(15,12));

ax1 = plt.subplot(221)
sns.regplot(X_train6['sqft_living15'], y_train_logged)
plt.title('Training Data')
plt.xlabel('logged square footage of 15 nearest neighbors')
plt.ylabel('logged price')

ax2 = plt.subplot(222)
sns.regplot(X_test6['sqft_living15'], y_test_logged)
plt.title('Test Data')
plt.xlabel('logged square footage of 15 nearest neighbors')
plt.ylabel('logged price')

ax3 = plt.subplot(223)
sns.regplot(np.expm1(X_train6['sqft_living15']), y_train)
plt.title('Training Data')
plt.xlabel('square footage of 15 nearest neighbors')

ax4 = plt.subplot(224)
sns.regplot(np.expm1(X_test6['sqft_living15']), y_test)
plt.title('Test Data')
plt.xlabel('square footage of 15 nearest neighbors');
```



In [191]:

```
# save figures for presentation

plt.subplots(figsize=(15,6));

ax1 = plt.subplot(121)
sns.regplot(X_train6['sqft_living'], y_train)
plt.title('Square Footage vs Price', fontsize=20)
plt.xlabel('Living Space Square Footage', fontsize=16)
plt.ylabel('Home Price in Dollars', fontsize=16)

ax3 = plt.subplot(122)
sns.regplot(np.expm1(X_train6['sqft_living15']), y_train)
plt.title('Square Footage of 15 Closest Neighbors vs Price', fontsize=20)
plt.xlabel('Square Footage of 15 Nearest Neighbors', fontsize=16)
plt.ylabel('Home Price in Dollars', fontsize=16)

plt.subplots_adjust(wspace = 0.3)

plt.savefig('images/regplots')
```



Model 7: Testing Removing Multicolinear Columns

In [192]:

```
# let's see if removing multicolinearity helps
# find top correlations
# code from Flatiron Data Science course's Multicollinearity Lab

df=X_train6.corr().abs().stack().reset_index().sort_values(0, ascending=False)

# zip the variable name columns
df['pairs'] = list(zip(df.level_0, df.level_1))

# set index to pairs
df.set_index(['pairs'], inplace = True)
```

```
#d rop level columns
df.drop(columns=['level_1', 'level_0'], inplace = True)

# rename correlation column as cc rather than 0
df.columns = ['cc']

# drop duplicates
df.drop_duplicates(inplace=True)
```

In [193]:

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

# high correlations among these four variables:
# sqft_living, sqft_above, grade, bathrooms
```

Out[193]:

	cc
pairs	
(sqft_above, sqft_living)	0.87374
(sqft_living, grade)	0.75806
(sqft_living, bathrooms)	0.75319
(grade, sqft_above)	0.75205

In [194]:

```
best_r2
```

Out[194]:

```
{'train': 0.831058, 'test': 0.825734}
```

In [195]:

```
# iterate thru combinations of highly correlated variables to see if dropping them increa
ses r2

correlated = ['sqft_living',
              'sqft_above',
              'grade',
              'bathrooms']

combs_list=[]

for n in range(1,4):

    comb = combinations(correlated,n)
    combs_list = combs_list + list(comb)

combs_list

for c in combs_list:
    print(c)
    X_train7 = X_train6.drop(columns = list(c))
    X_test7 = X_test6.drop(columns = list(c))
    y_train_pred7, y_test_pred7 = scale_lin_reg(X_train=X_train7, y_train=y_train_logged
, X_test=X_test7)
    eval_r2(y_train=y_train_logged, y_train_pred=y_train_pred7, y_test=y_test_logged, y_
test_pred=y_test_pred7)
    print('\n')

# despite the multicolinearity, dropping combinations of these columns does not result in
an improved R2
# dropping sqft_above returns almost exactly the same result
```

```
('sqft_living',)
Training Data
 R-Squared: 0.830418
```

```
 R-Squared: 0.830418

Test Data
 R-Squared: 0.825369


('sqft_above',)
Training Data
 R-Squared: 0.831058

Test Data
 R-Squared: 0.82574


('grade',)
Training Data
 R-Squared: 0.810266

Test Data
 R-Squared: 0.802736


('bathrooms',)
Training Data
 R-Squared: 0.829158

Test Data
 R-Squared: 0.824471


('sqft_living', 'sqft_above')
Training Data
 R-Squared: 0.816

Test Data
 R-Squared: 0.810406


('sqft_living', 'grade')
Training Data
 R-Squared: 0.809057

Test Data
 R-Squared: 0.801954


('sqft_living', 'bathrooms')
Training Data
 R-Squared: 0.828259

Test Data
 R-Squared: 0.823891


('sqft_above', 'grade')
Training Data
 R-Squared: 0.810259

Test Data
 R-Squared: 0.802671


('sqft_above', 'bathrooms')
Training Data
 R-Squared: 0.829154

Test Data
 R-Squared: 0.824497


('grade', 'bathrooms')
Training Data
 R-Squared: 0.807461
```

```
 R-Squared: 0.807461

Test Data
 R-Squared: 0.800854


('sqft_living', 'sqft_above', 'grade')
Training Data
 R-Squared: 0.768717

Test Data
 R-Squared: 0.75855


('sqft_living', 'sqft_above', 'bathrooms')
Training Data
 R-Squared: 0.808163

Test Data
 R-Squared: 0.804131


('sqft_living', 'grade', 'bathrooms')
Training Data
 R-Squared: 0.805802

Test Data
 R-Squared: 0.79969


('sqft_above', 'grade', 'bathrooms')
Training Data
 R-Squared: 0.80746

Test Data
 R-Squared: 0.800837
```

Model 8: Test removing features with high p-values

In [196]:

```
# according to the statsmodels output, p-values for sqft_above and sqft_basement were abo
ve 0.05

X_train8 = X_train6.copy()
X_train8.drop(columns=['sqft_above', 'sqft_basement'], inplace=True)

X_test8 = X_test6.copy()
X_test8.drop(columns=['sqft_above', 'sqft_basement'], inplace=True)
```

In [197]:

```
y_train_pred8, y_test_pred8 = scale_lin_reg(X_train=X_train8, y_train=y_train_logged, X_t
est=X_test8)
```

In [198]:

```
eval_r2(y_train=y_train_logged, y_train_pred=y_train_pred8, y_test=y_test_logged, y_test
_pred=y_test_pred8)
```

```
Training Data
 R-Squared: 0.831025

Test Data
 R-Squared: 0.825705
```

In [199]:

```
unlog_MAE_RMSE(y_train=y_train, y_train_logged_pred=y_train_pred8, y_test=y_test, y_test
```

```
unlog_MAE_RMSE(y_train=y_train, y_train_logged_pred=y_train_pred8, y_test=y_test, y_test
_logged_pred=y_test_pred8)
```

```
Training Data
 Mean Absolute Error: 88274.84
 Root Mean Squared Error: 151654.38

Test Data
 Mean Absolute Error: 86930.64
 Root Mean Squared Error: 144778.39
```

In [200]:

```
best_r2
# no improvement in r2 for Model 8, but let's keep this model since at least it reduces m
ulticolinearity
# and removes coefficients with high p-values
```

Out[200]:

```
{'train': 0.831058, 'test': 0.825734}
```

In [201]:

```
# look at coefficients for Model 8

model = sm.OLS(y_train_logged, sm.add_constant(pd.DataFrame(X_train8, columns=X_train8.c
olumns, index=X_train8.index)))
results = model.fit()

results.summary()
```

```
/opt/anaconda3/envs/learn-env/lib/python3.6/site-packages/numpy/core/fromnumeric.py:2580:
FutureWarning: Method .ptp is deprecated and will be removed in a future version. Use num
py.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

Out[201]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.831
Model:	OLS	**Adj. R-squared:**	0.831
Method:	Least Squares	**F-statistic:**	4355.
Date:	Sun, 31 Jan 2021	**Prob (F-statistic):**	0.00
Time:	11:57:35	**Log-Likelihood:**	1908.8
No. Observations:	15072	**AIC:**	-3782.
Df Residuals:	15054	**BIC:**	-3645.
Df Model:	17		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	17.0226	0.172	98.829	0.000	16.685	17.360
bedrooms	-0.0149	0.003	-5.951	0.000	-0.020	-0.010
bathrooms	0.0558	0.004	13.404	0.000	0.048	0.064
sqft_living	0.0002	4.35e-06	40.889	0.000	0.000	0.000
sqft_lot	8.785e-07	7.06e-08	12.444	0.000	7.4e-07	1.02e-06
floors	0.0213	0.004	4.870	0.000	0.013	0.030
condition	0.0420	0.003	14.218	0.000	0.036	0.048
grade	0.1181	0.003	43.205	0.000	0.113	0.123
yr_built	-0.0028	8.58e-05	-33.197	0.000	-0.003	-0.003

sqft_living15	0.1598	0.009	18.106	0.000	0.142	0.177
sqft_lot15	-0.0364	0.003	-10.727	0.000	-0.043	-0.030
renovated	0.0749	0.010	7.393	0.000	0.055	0.095
Zip Class 2	-0.2832	0.038	-7.513	0.000	-0.357	-0.209
Zip Class 3	-0.4949	0.037	-13.348	0.000	-0.568	-0.422
Zip Class 4	-0.6148	0.037	-16.655	0.000	-0.687	-0.542
Zip Class 5	-0.8203	0.037	-22.140	0.000	-0.893	-0.748
Zip Class 6	-1.1032	0.037	-29.780	0.000	-1.176	-1.031
Zip Class 7	-1.2142	0.038	-31.560	0.000	-1.290	-1.139

Omnibus:	835.184	**Durbin-Watson:**	2.018
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	3079.639
Skew:	0.157	**Prob(JB):**	0.00
Kurtosis:	5.192	**Cond. No.**	3.62e+06

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.62e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

Model 9: Experiment with categorizing year built

In [202]:

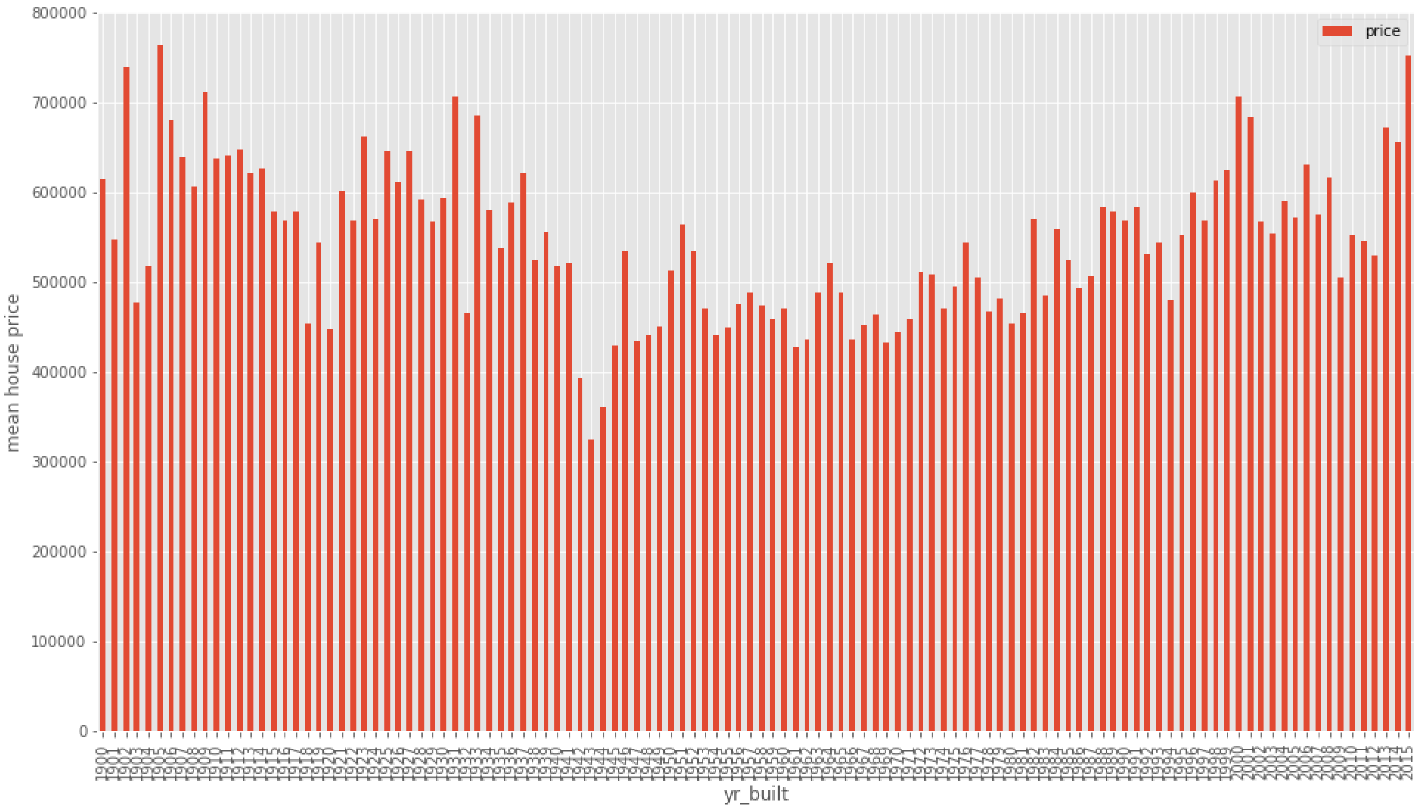
```
# made a df to add price back to X variables table, so we can make a pivot table
built_df = pd.concat([X_train8, y_train], axis=1)
built_pivot = built_df.pivot_table(values='price', index='yr_built', ).sort_values(by='y
r_built')

# plot a bar graph to look at possible categories
built_pivot.plot(kind='bar', figsize=(16,9))
plt.ylabel('mean house price')
plt.legend;

built_df

# hmmm, almost looks like older homes and new homes are highly valued, while homes in the
middle are not
```

Out[202]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	condition	grade	yr_built	sqft_living15	sqft_lot15	renovated	Z Clas
753	2	2.50000	2380	6600	1.00000	3	8	2010	7.53369	8.79482	0	
1418	4	3.75000	3190	17186	2.00000	3	10	1999	7.73631	9.51015	0	
8178	3	2.50000	1730	6930	2.00000	3	8	1994	7.48437	8.84362	0	
2254	4	2.00000	1870	8750	1.00000	3	7	1977	7.47873	9.01274	0	
4063	8	3.00000	2850	12714	1.00000	3	7	1959	7.29980	8.50553	0	
...	...	...	...	...	...	...	...	...	...	...	...	
11964	3	2.50000	2230	5800	2.00000	3	7	2004	7.70976	8.71407	0	
21575	4	2.75000	2770	3852	2.00000	3	8	2014	7.50108	8.63782	0	
5390	4	1.50000	1530	9000	1.00000	4	6	1976	7.32647	9.04782	0	
860	1	0.75000	380	15000	1.00000	3	5	1963	7.06476	9.61581	0	
15795	4	2.50000	2755	11612	2.00000	3	8	2001	[illegible]	[illegible]	0	

15795	4	2.50000	2755	11612	2.00000	3	8	2001	7.94449	9.45962	0	
	bedrooms	bathrooms	sqft_living	sqft_lot	floors	condition	grade	yr_built	sqft_living15	sqft_lot15	renovated	Z Clas

15072 rows × 18 columns



In [203]:

```python
# create a dictionary of years, showing the corresponding categories

years_list = list(built_pivot.index)
years_dict = {}

for i in years_list[0:41]:
    years_dict[i] = 'pre-war'

for i in years_list[41:88]:
    years_dict[i] = 'mid-century'

for i in years_list[88:117]:
    years_dict[i] = 'recent'

years_dict
```

Out[203]:

```
{1900: 'pre-war',
 1901: 'pre-war',
 1902: 'pre-war',
 1903: 'pre-war',
 1904: 'pre-war',
 1905: 'pre-war',
 1906: 'pre-war',
 1907: 'pre-war',
 1908: 'pre-war',
 1909: 'pre-war',
 1910: 'pre-war',
 1911: 'pre-war',
 1912: 'pre-war',
 1913: 'pre-war',
 1914: 'pre-war',
 1915: 'pre-war',
 1916: 'pre-war',
 1917: 'pre-war',
 1918: 'pre-war',
 1919: 'pre-war',
```

```
    1920: 'pre-war',
    1921: 'pre-war',
    1922: 'pre-war',
    1923: 'pre-war',
    1924: 'pre-war',
    1925: 'pre-war',
    1926: 'pre-war',
    1927: 'pre-war',
    1928: 'pre-war',
    1929: 'pre-war',
    1930: 'pre-war',
    1931: 'pre-war',
    1932: 'pre-war',
    1933: 'pre-war',
    1934: 'pre-war',
    1935: 'pre-war',
    1936: 'pre-war',
    1937: 'pre-war',
    1938: 'pre-war',
    1939: 'pre-war',
    1940: 'pre-war',
    1941: 'mid-century',
    1942: 'mid-century',
    1943: 'mid-century',
    1944: 'mid-century',
    1945: 'mid-century',
    1946: 'mid-century',
    1947: 'mid-century',
    1948: 'mid-century',
    1949: 'mid-century',
    1950: 'mid-century',
    1951: 'mid-century',
    1952: 'mid-century',
    1953: 'mid-century',
    1954: 'mid-century',
    1955: 'mid-century',
    1956: 'mid-century',
    1957: 'mid-century',
    1958: 'mid-century',
    1959: 'mid-century',
    1960: 'mid-century',
    1961: 'mid-century',
    1962: 'mid-century',
    1963: 'mid-century',
    1964: 'mid-century',
    1965: 'mid-century',
    1966: 'mid-century',
    1967: 'mid-century',
    1968: 'mid-century',
    1969: 'mid-century',
    1970: 'mid-century',
    1971: 'mid-century',
    1972: 'mid-century',
    1973: 'mid-century',
    1974: 'mid-century',
    1975: 'mid-century',
    1976: 'mid-century',
    1977: 'mid-century',
    1978: 'mid-century',
    1979: 'mid-century',
    1980: 'mid-century',
    1981: 'mid-century',
    1982: 'mid-century',
    1983: 'mid-century',
    1984: 'mid-century',
    1985: 'mid-century',
    1986: 'mid-century',
    1987: 'mid-century',
    1988: 'recent',
    1989: 'recent',
    1990: 'recent',
    1991: 'recent',
```

```
 1992: 'recent',
 1993: 'recent',
 1994: 'recent',
 1995: 'recent',
 1996: 'recent',
 1997: 'recent',
 1998: 'recent',
 1999: 'recent',
 2000: 'recent',
 2001: 'recent',
 2002: 'recent',
 2003: 'recent',
 2004: 'recent',
 2005: 'recent',
 2006: 'recent',
 2007: 'recent',
 2008: 'recent',
 2009: 'recent',
 2010: 'recent',
 2011: 'recent',
 2012: 'recent',
 2013: 'recent',
 2014: 'recent',
 2015: 'recent'}
```

In [204]:

```python
# add a column with yr_built categories

built_df['built_cat'] = built_df['yr_built'].map(lambda x: years_dict[x])
built_df
```

Out[204]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	condition	grade	yr_built	sqft_living15	sqft_lot15	renovated	Z Clas
753	2	2.50000	2380	6600	1.00000	3	8	2010	7.53369	8.79482	0	
1418	4	3.75000	3190	17186	2.00000	3	10	1999	7.73631	9.51015	0	
8178	3	2.50000	1730	6930	2.00000	3	8	1994	7.48437	8.84362	0	
2254	4	2.00000	1870	8750	1.00000	3	7	1977	7.47873	9.01274	0	
4063	8	3.00000	2850	12714	1.00000	3	7	1959	7.29980	8.50553	0	
...	...	...	...	...	...	...	...	...	...	...	...	
11964	3	2.50000	2230	5800	2.00000	3	7	2004	7.70976	8.71407	0	
21575	4	2.75000	2770	3852	2.00000	3	8	2014	7.50108	8.63782	0	
5390	4	1.50000	1530	9000	1.00000	4	6	1976	7.32647	9.04782	0	
860	1	0.75000	380	15000	1.00000	3	5	1963	7.06476	9.61581	0	
15795	4	2.50000	2755	11612	2.00000	3	8	2001	7.94449	9.45962	0	

15072 rows × 19 columns

In [205]:

```python
# one hot encode classification column

built_cat_columns = pd.get_dummies(built_df['built_cat'], drop_first=True)
built_cat_columns

built_df = pd.concat([built_df, built_cat_columns], axis=1)
```

```
built_df.drop(columns=['yr_built','built_cat'], inplace=True)
```

In [206]:

```
built_df
```

Out[206]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	condition	grade	sqft_living15	sqft_lot15	renovated	Zip Class 2	Zip Class 3
753	2	2.50000	2380	6600	1.00000	3	8	7.53369	8.79482	0	0	0
1418	4	3.75000	3190	17186	2.00000	3	10	7.73631	9.51015	0	0	0
8178	3	2.50000	1730	6930	2.00000	3	8	7.48437	8.84362	0	0	0
2254	4	2.00000	1870	8750	1.00000	3	7	7.47873	9.01274	0	0	0
4063	8	3.00000	2850	12714	1.00000	3	7	7.29980	8.50553	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
11964	3	2.50000	2230	5800	2.00000	3	7	7.70976	8.71407	0	0	0
21575	4	2.75000	2770	3852	2.00000	3	8	7.50108	8.63782	0	0	0
5390	4	1.50000	1530	9000	1.00000	4	6	7.32647	9.04782	0	0	0
860	1	0.75000	380	15000	1.00000	3	5	7.06476	9.61581	0	0	0
15795	4	2.50000	2755	11612	2.00000	3	8	7.94449	9.45962	0	0	0

15072 rows × 19 columns

In [207]:

```
# now we just have to stick these columns back onto the training and test sets

#training set first
X_train9 = X_train8.copy()

columns_to_add = built_df[['pre-war', 'recent']]
X_train9 = pd.concat([X_train9, columns_to_add], axis=1)
X_train9.drop(columns='yr_built', inplace=True)
```

In [208]:

```
# now do test set

X_test9 = X_test8.copy()

X_test9['built_cat'] = X_test9['yr_built'].map(lambda x: years_dict[x])
built_cat_columns = pd.get_dummies(X_test9['built_cat'], drop_first=True)

X_test9 = pd.concat([X_test9, built_cat_columns], axis=1)
X_test9.drop(columns=['yr_built','built_cat'], inplace=True)

X_test9
```

Out[208]:

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	condition	grade	sqft_living15	sqft_lot15	renovated	Zip Class 2	Zip Class 3
3686	3	0.75000	850	8573	1.00000	3	6	6.74524	9.03384	0	0	0
10247	3	1.00000	1510	6083	1.00000	4	6	7.31986	8.65032	0	0	0
4037	4	2.25000	1790	42000	1.00000	3	7	7.63046	10.82166	0	0	0
3437	2	1.50000	1140	2500	1.00000	3	7	7.31322	8.51719	0	0	0
19291	3	1.00000	1500	3920	1.00000	3	7	7.40245	8.29829	0	0	0

...	bedrooms	bathrooms	sqft_living	sqft_lot	floors	condition	grade	sqft_living15	sqft_lot15	renovated	Zip Class	Zip Class
9400	4	3.50000	2650	3060	2.00000	3	9	7.29302	8.02617	0	0	0
9092	4	2.75000	2670	6780	2.00000	5	8	7.78322	8.69768	0	0	0
6650	3	1.75000	1600	10280	1.00000	3	7	7.37149	8.99962	0	0	0
21095	5	3.50000	2760	3865	2.50000	3	8	7.85941	8.43098	0	0	0
3372	2	1.75000	1060	16470	1.00000	3	7	7.48997	9.72603	0	0	0

6458 rows × 18 columns

In [209]:

```
# let's test model 9!

y_train_pred9, y_test_pred9 = scale_lin_reg(X_train=X_train9, y_train=y_train_logged, X_t
est=X_test9)
```

In [210]:

```
eval_r2(y_train=y_train_logged, y_train_pred=y_train_pred9, y_test=y_test_logged, y_test
_pred=y_test_pred9)

# R-squared is less than for Model 8
# so, segmenting year_built into categories does not help explain any variance
# perhaps this variance can be explained by square footage and location alone
```

```
Training Data
 R-Squared: 0.828802

Test Data
 R-Squared: 0.824774
```

In [211]:

```
best_r2
```

Out[211]:

```
{'train': 0.831058, 'test': 0.825734}
```

In [213]:

```
model = sm.OLS(y_train_logged, sm.add_constant(pd.DataFrame(X_train9, columns=X_train9.c
olumns, index=X_train9.index)))
results = model.fit()

results.summary()
# interesting, now floors has the highest p-value
```

```
/opt/anaconda3/envs/learn-env/lib/python3.6/site-packages/numpy/core/fromnumeric.py:2580:
FutureWarning: Method .ptp is deprecated and will be removed in a future version. Use num
py.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

Out[213]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.829
Model:	OLS	**Adj. R-squared:**	0.829
Method:	Least Squares	**F-statistic:**	4049.
Date:	Sun, 31 Jan 2021	**Prob (F-statistic):**	0.00
Time:	11:59:27	**Log-Likelihood:**	1810.3
No. Observations:	15072	**AIC:**	-3583.
Df Residuals:	15053	**BIC:**	-3438.

Df Residuals:	[illegible]	BIC:	[illegible]
Df Model:	18		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	11.5503	0.073	158.954	0.000	11.408	11.693
bedrooms	-0.0107	0.003	-4.212	0.000	-0.016	-0.006
bathrooms	0.0372	0.004	9.033	0.000	0.029	0.045
sqft_living	0.0002	4.35e-06	42.537	0.000	0.000	0.000
sqft_lot	7.875e-07	7.17e-08	10.989	0.000	6.47e-07	9.28e-07
floors	-0.0058	0.005	-1.221	0.222	-0.015	0.004
condition	0.0522	0.003	17.462	0.000	0.046	0.058
grade	0.1126	0.003	41.192	0.000	0.107	0.118
sqft_living15	0.1418	0.009	15.915	0.000	0.124	0.159
sqft_lot15	-0.0290	0.004	-8.250	0.000	-0.036	-0.022
renovated	0.1067	0.010	10.562	0.000	0.087	0.126
Zip Class 2	-0.2881	0.038	-7.592	0.000	-0.362	-0.214
Zip Class 3	-0.5113	0.037	-13.699	0.000	-0.584	-0.438
Zip Class 4	-0.6387	0.037	-17.191	0.000	-0.712	-0.566
Zip Class 5	-0.8433	0.037	-22.612	0.000	-0.916	-0.770
Zip Class 6	-1.1356	0.037	-30.461	0.000	-1.209	-1.063
Zip Class 7	-1.2398	0.039	-32.016	0.000	-1.316	-1.164
pre-war	0.1529	0.006	27.387	0.000	0.142	0.164
recent	-0.0343	0.006	-5.900	0.000	-0.046	-0.023

Omnibus:	902.979	Durbin-Watson:	2.020
Prob(Omnibus):	0.000	Jarque-Bera (JB):	3205.348
Skew:	0.221	Prob(JB):	0.00
Kurtosis:	5.215	Cond. No.	2.14e+06

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.14e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

Conclusions and Future Work

To accurately price homes in King County, the real estate firm should use a model that segments zip codes into price-based categories, such as Model 8. This model combines data about house features that are highly correlated with price, such as square footage, with knowledge of the mean house price of each zip code, to produce predictions that explain 83% of the variance from the mean price. Model 8 is a significant improvement over the baseline regression model, which only explains 63% of the variance. In addition, while the baseline regression model's predictions were an average of $136K off from the actual prices of the test data, Model 8's Mean Absolute Error was only $87K for the test data. \ \ Square footage and grade have the strongest positive correlation with price, but the model vastly improved after zip code classifications were included. Unsurprisingly, location seems extremely important to home buyers in the Seattle area, which is a diverse landscape that includes, urban, suburban, and rural neighborhoods. \ \ Much work remains to investigate potential improvements to this model. In particular, including interactions among X variables may increase the model's accuracy. Since square footage and zip code are such powerful predictors of price, perhaps an interaction between these variables would enhance the model. Also, since zip code classification was so effective in improving the model, perhaps including a few more zip classes would help by segmenting the market even

further.

In addition, the month when the house was sold may affect price, and was not tested in these models. Also not tested was a feature that would indicate whether the house was recently renovated, for example in the past 20 years. It may also help to programmatically iterate through the X-variables to select the best features for inclusion in the model.

Finally, a handful of properties (less than half of one per cent) must be excluded from this model. Creating models that can generate predictions for these homes as well would benefit the real estate firm.