Final Project Submission

Please fill out:

- Student name:
- Student pace: self paced / part time / full time
- Scheduled project review date/time:
- Instructor name:
- Blog post URL:

- **id** - unique identified for a house
- **dateDate** - house was sold
- **pricePrice** - is prediction target
- **bedroomsNumber** - of Bedrooms/House
- **bathroomsNumber** - of bathrooms/bedrooms
- **sqft_livingsquare** - footage of the home
- **sqft_lotsquare** - footage of the lot
- **floorsTotal** - floors (levels) in house
- **waterfront** - House which has a view to a waterfront
- **view** - Has been viewed
- **condition** - How good the condition is (Overall)
- **grade** - overall grade given to the housing unit, based on King County grading system
- **sqft_above** - square footage of house apart from basement
- **sqft_basement** - square footage of the basement
- **yr_built** - Built Year
- **yr_renovated** - Year when house was renovated
- **zipcode** - zip
- **lat** - Latitude coordinate
- **long** - Longitude coordinate
- **sqft_living15** - The 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

In [1]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go
import plotly.express as px

from statsmodels.formula.api import ols
from statsmodels.stats.outliers_influence import variance_inflation_factor
import statsmodels.api as sm
import scipy.stats as stats
from sklearn.model_selection import train_test_split
```

Overview of Data

In [2]:
```
house_data = pd.read_csv('data/kc_house_data.csv')
house_data
```

Out[2]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_ba
0	7129300520	10/13/2014	221900.0	3	1.00	1180	5650	1.0	NaN	0.0	...	7	1180	
1	6414100192	12/9/2014	538000.0	3	2.25	2570	7242	2.0	0.0	0.0	...	7	2170	
2	5631500400	2/25/2015	180000.0	2	1.00	770	10000	1.0	0.0	0.0	...	6	770	
3	2487200875	12/9/2014	604000.0	4	3.00	1960	5000	1.0	0.0	0.0	...	7	1050	
4	1954400510	2/18/2015	510000.0	3	2.00	1680	8080	1.0	0.0	0.0	...	8	1680	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
21592	263000018	5/21/2014	360000.0	3	2.50	1530	1131	3.0	0.0	0.0	...	8	1530	
21593	6600060120	2/23/2015	400000.0	4	2.50	2310	5813	2.0	0.0	0.0	...	8	2310	
21594	1523300141	6/23/2014	402101.0	2	0.75	1020	1350	2.0	0.0	0.0	...	7	1020	
21595	291310100	1/16/2015	400000.0	3	2.50	1600	2388	2.0	NaN	0.0	...	8	1600	
21596	1523300157	10/15/2014	325000.0	2	0.75	1020	1076	2.0	0.0	0.0	...	7	1020	

21597 rows × 21 columns

In [3]:
```
house_data.info()
house_data.shape
```

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

Out[3]: `(21597, 21)`

In [4]:
```
# columns to take a closer look at
# look at the unique values for price
house_data.price.unique()[:5]
```

Out[4]: `array([221900., 538000., 180000., 604000., 510000.])`

In [5]:
```
# sanity check missing values
house_data.isna().sum()
```

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

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

Out[6]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	cond
count	2.159700e+04	2.159700e+04	21597.000000	21597.000000	21597.000000	2.159700e+04	21597.000000	19221.000000	21534.000000	21597.0
mean	4.580474e+09	5.402966e+05	3.373200	2.115826	2080.321850	1.509941e+04	1.494096	0.007596	0.233863	3.4
std	2.876736e+09	3.673681e+05	0.926299	0.768984	918.106125	4.141264e+04	0.539683	0.086825	0.765686	0.6
min	1.000102e+06	7.800000e+04	1.000000	0.500000	370.000000	5.200000e+02	1.000000	0.000000	0.000000	1.0
25%	2.123049e+09	3.220000e+05	3.000000	1.750000	1430.000000	5.040000e+03	1.000000	0.000000	0.000000	3.0
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.500000	0.000000	0.000000	3.0
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068500e+04	2.000000	0.000000	0.000000	4.0
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	1.000000	4.000000	5.0

Cleaning & Exploring

Plan

- Set the columns(waterfront, view, yr_renovated) with missing values to '0'
- Replace column dtypes for necessary columns

In [7]:
```
# check the value counts for each column.
for value in house_data.columns:
    print(value) # column name
    print(house_data[value].value_counts(ascending=False)) # distribution of frequently occuring values
    print('                                    ')
```

```
id
795000620     3
1825069031    2
2019200220    2
7129304540    2
1781500435    2
             ..
7812801125    1
4364700875    1
3021059276    1
880000205     1
1777500160    1
Name: id, Length: 21420, dtype: int64

date
6/23/2014    142
6/26/2014    131
6/25/2014    131
7/8/2014     127
4/27/2015    126
            ...
5/17/2014      1
5/24/2015      1
1/10/2015      1
8/30/2014      1
7/27/2014      1
Name: date, Length: 372, dtype: int64

price
350000.0    172
450000.0    172
550000.0    159
500000.0    152
425000.0    150
           ...
870515.0      1
336950.0      1
386100.0      1
176250.0      1
884744.0      1
Name: price, Length: 3622, dtype: int64

bedrooms
3     9824
4     6882
2     2760
5     1601
6      272
1      196
7       38
8       13
9        6
10       3
11       1
33       1
Name: bedrooms, dtype: int64

bathrooms
2.50    5377
1.00    3851
1.75    3048
2.25    2047
2.00    1930
1.50    1445
2.75    1185
3.00     753
3.50     731
3.25     589
3.75     155
4.00     136
4.50     100
4.25      79
0.75      71
4.75      23
5.00      21
5.25      13
5.50      10
1.25       9
6.00       6
5.75       4
0.50       4
8.00       2
6.25       2
6.75       2
6.50       2
7.50       1
7.75       1
Name: bathrooms, dtype: int64

sqft_living
1300    138
1400    135
1440    133
1660    129
1010    129
       ...
4970      1
2905      1
2793      1
4810      1
1975      1
Name: sqft_living, Length: 1034, dtype: int64

sqft_lot
5000      358
6000      290
4000      251
7200      220
7500      119
         ...
1448        1
38884       1
17313       1
35752       1
315374      1
Name: sqft_lot, Length: 9776, dtype: int64

floors
1.0    10673
2.0     8235
1.5     1910
3.0      611
2.5      161
3.5        7
Name: floors, dtype: int64

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

view
0.0    19422
2.0      957
3.0      508
1.0      330
4.0      317
Name: view, dtype: int64

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

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

sqft_above
1300    212
1010    210
1200    206
1220    192
1140    184
       ...
2601      1
440       1
2473      1
2441      1
1975      1
Name: sqft_above, Length: 942, dtype: int64

sqft_basement
0.0       12826
?           454
600.0       217
500.0       209
700.0       208
          ...
1248.0        1
588.0         1
1275.0        1
2300.0        1
374.0         1
Name: sqft_basement, Length: 304, dtype: int64

yr_built
2014    559
2006    453
2005    450
2004    433
2003    420
       ...
1933     30
1901     29
1902     27
1935     24
1934     21
Name: yr_built, Length: 116, dtype: int64

yr_renovated
0.0       17011
2014.0       73
2003.0       31
2013.0       31
2007.0       30
          ...
1946.0        1
1959.0        1
1971.0        1
1951.0        1
1954.0        1
Name: yr_renovated, Length: 70, dtype: int64

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

lat
47.6624    17
47.5491    17
47.5322    17
47.6846    17
47.6711    16
           ..
47.2785     1
47.4162     1
47.3870     1
47.2313     1
47.2715     1
Name: lat, Length: 5033, dtype: int64

long
-122.290    115
-122.300    111
-122.362    104
-122.291    100
-122.372     99
           ...
-121.403      1
-121.804      1
-121.726      1
-121.895      1
-121.893      1
Name: long, Length: 751, dtype: int64

sqft_living15
1540    197
1440    195
1560    192
1500    180
1460    169
       ...
4890      1
2873      1
952       1
3193      1
2049      1
Name: sqft_living15, Length: 777, dtype: int64

sqft_lot15
5000      427
4000      356
6000      288
7200      210
4800      145
         ...
11036       1
8989        1
871200      1
809         1
6147        1
Name: sqft_lot15, Length: 8682, dtype: int64
```

In [8]:
```
# id has multiple counts that means there are duplicate housing info on here
# is it safe to drop the duplicated data? check the %
house_data.duplicated('id').value_counts()
```

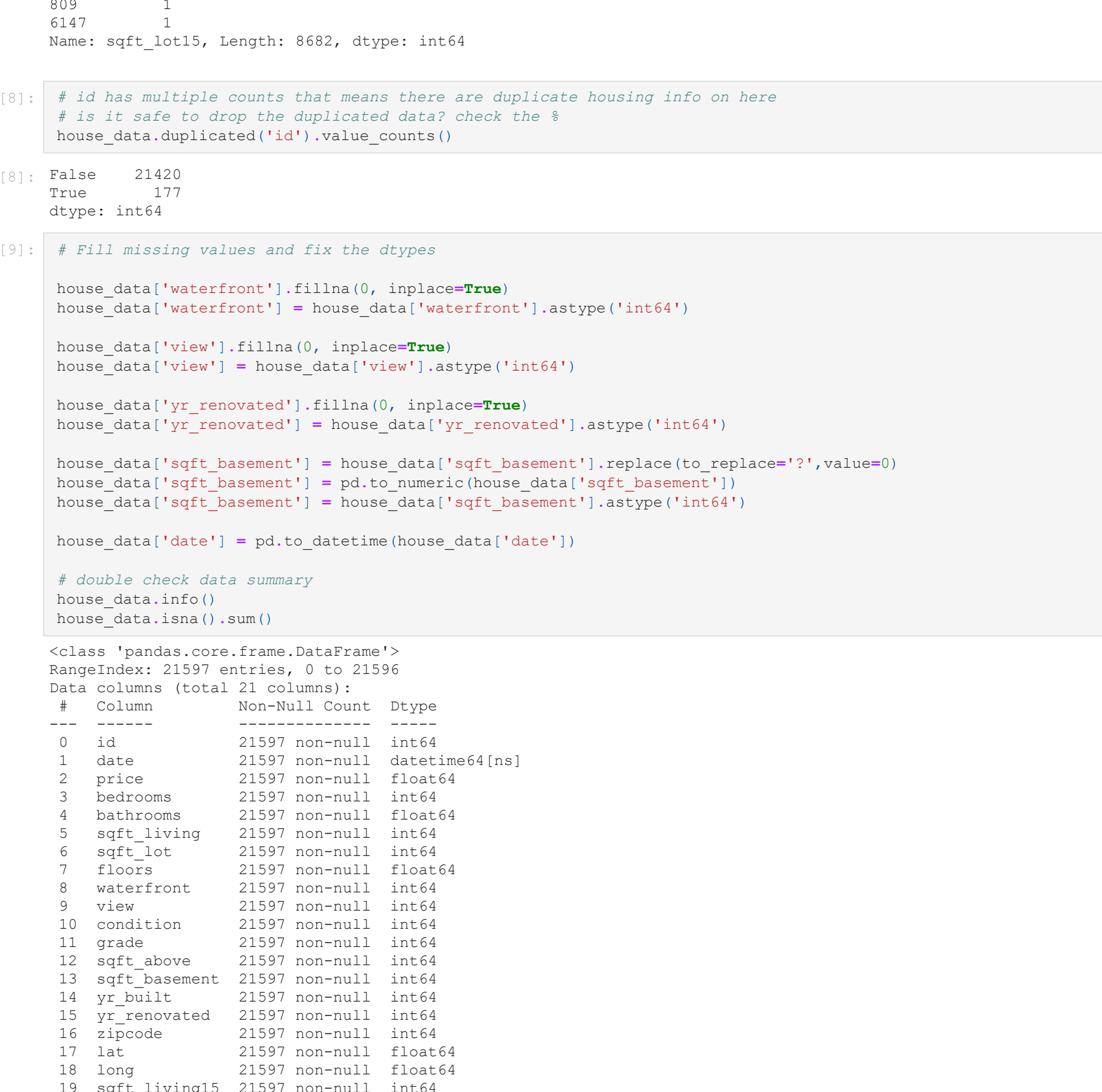
Out[8]:
```
False    21420
True       177
dtype: int64
```

In [9]:
```
# Fill missing values and fix the dtypes

house_data['waterfront'].fillna(0, inplace=True)
house_data['waterfront'] = house_data['waterfront'].astype('int64')

house_data['view'].fillna(0, inplace=True)
house_data['view'] = house_data['view'].astype('int64')

house_data['yr_renovated'].fillna(0, inplace=True)
house_data['yr_renovated'] = house_data['yr_renovated'].astype('int64')

house_data['sqft_basement'] = house_data['sqft_basement'].replace(to_replace='?',value=0)
house_data['sqft_basement'] = pd.to_numeric(house_data['sqft_basement'])
house_data['sqft_basement'] = house_data['sqft_basement'].astype('int64')

house_data['date'] = pd.to_datetime(house_data['date'])

# double check data summary
house_data.info()
house_data.isna().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             21597 non-null  int64
 1   date           21597 non-null  datetime64[ns]
 2   price          21597 non-null  float64
 3   bedrooms       21597 non-null  int64
 4   bathrooms      21597 non-null  float64
 5   sqft_living    21597 non-null  int64
 6   sqft_lot       21597 non-null  int64
 7   floors         21597 non-null  float64
 8   waterfront     21597 non-null  int64
 9   view           21597 non-null  int64
 10  condition      21597 non-null  int64
 11  grade          21597 non-null  int64
 12  sqft_above     21597 non-null  int64
 13  sqft_basement  21597 non-null  int64
 14  yr_built       21597 non-null  int64
 15  yr_renovated   21597 non-null  int64
 16  zipcode        21597 non-null  int64
 17  lat            21597 non-null  float64
 18  long           21597 non-null  float64
 19  sqft_living15  21597 non-null  int64
 20  sqft_lot15     21597 non-null  int64
dtypes: datetime64[ns](1), float64(5), int64(15)
memory usage: 3.5 MB
```

Out[9]:
```
id               0
date             0
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
grade            0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
zipcode          0
lat              0
long             0
sqft_living15    0
sqft_lot15       0
dtype: int64
```

In [10]:
```
# checking the raw distribution for each feature
house_data.hist(figsize=(15,15));
```

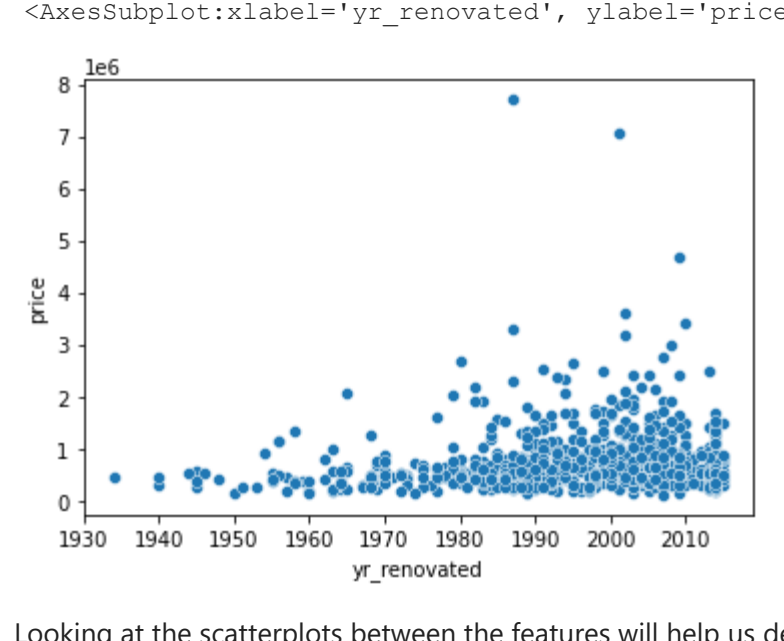


The distribution for each feature is either left or right skewed. This will not pass the normality assumption test. We will work on this when preparing the data.

In [11]:
```
# taking a closer look at our features to closer identify which columns are categorical or numerical
# look out for possible outliers to remove
# picture an imagination regression line
for x in house_data.columns:
    sns.scatterplot(x=x, y='price', data=house_data)
    plt.show()
```

In [12]:
```python
# taking a closer look of the yr_renovated to price
yr = house_data[house_data['yr_renovated'] != 0]
sns.scatterplot(x='yr_renovated', y='price', data=yr)
```

Out[12]: <AxesSubplot:xlabel='yr_renovated', ylabel='price'>

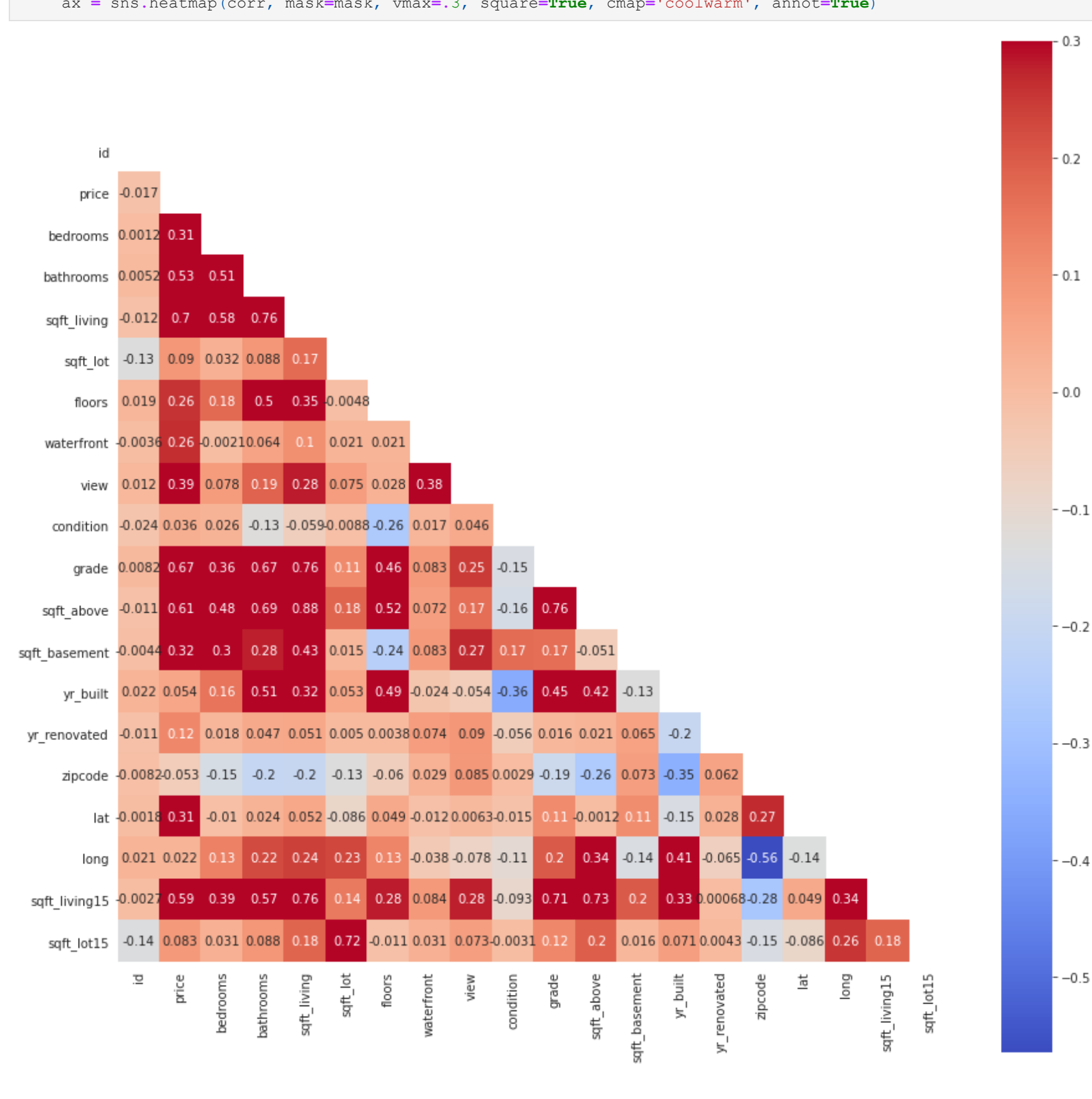


Looking at the scatterplots between the features will help us determine the numerical and categorical variables in regards to price. The scatter plots that show vertical line structrue will be our catergorical variables while the graphs with a cloud structure will be our numerical variables.

In [13]:
```python
num_var = ['sqft_living', 'sqft_lot', ]

sns.lmplot(x = 'sqft_living', y='price', data=house_data)
```

Out[13]: <seaborn.axisgrid.FacetGrid at 0x1910e7958b0>

In [14]:
```python
# visualize the correlation between features
# helps understand the multicollinearity
corr = house_data.corr()
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)] = True
with sns.axes_style("white"):
    f, ax = plt.subplots(figsize=(15, 15))
    ax = sns.heatmap(corr, mask=mask, vmax=.3, square=True, cmap='coolwarm', annot=True)
```



- Dark red represents strong positive linear relationships
- Dark blue presents strong negative linear relationships
- Help explain Multicollinearity
- r > .65 produces multicollinearity in a model
- Multicollinearity is when there is high correlation between three or more variables

In [15]:
```python
target_var = 'price'
pred_features = [x_col for x_col in house_data.columns if x_col != target_var]
target_var in pred_features
pred_features
```

Out[15]:
```
['id',
 'date',
 'bedrooms',
 'bathrooms',
 'sqft_living',
 'sqft_lot',
 'floors',
 'waterfront',
 'view',
 'condition',
 'grade',
 'sqft_above',
 'sqft_basement',
 'yr_built',
 'yr_renovated',
 'zipcode',
 'lat',
 'long',
 'sqft_living15',
 'sqft_lot15']
```

Baseline Model

In [16]:
```python
outcome = 'price'
predictors = house_data.drop(columns = ['price','date', 'id'], axis=1)
# dropping date and id since im not particularly interested in the predictions of these features
pred_sum = '+'.join(predictors.columns)
formula = outcome + '~' + pred_sum
```

In [17]:
```python
formula
```

Out[17]: 'price~bedrooms+bathrooms+sqft_living+sqft_lot+floors+waterfront+view+condition+grade+sqft_above+sqft_basement+yr_built+yr_renovated+zipcode+lat+long+sqft_living15+sqft_lot15'

In [18]:
```python
model_1 = ols(formula=formula, data=house_data).fit()
model_1.summary()
```

Out[18]:

OLS Regression Results

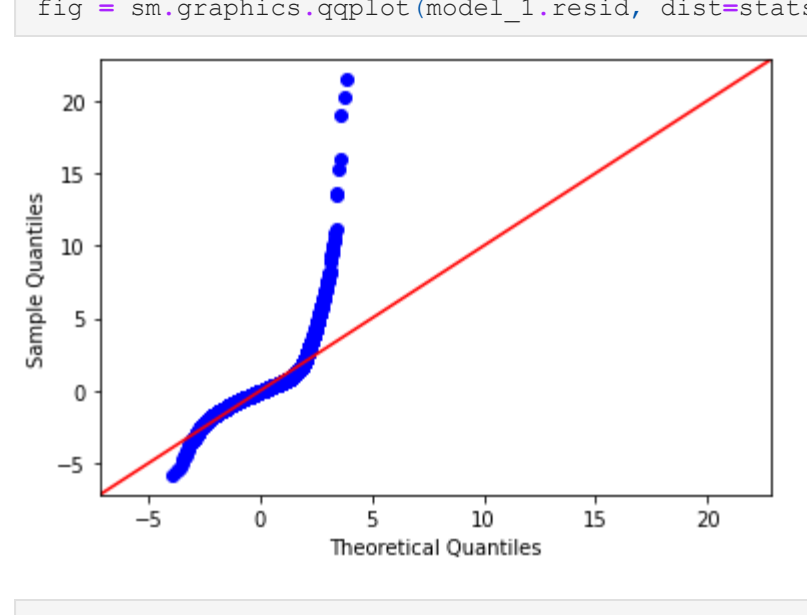
Dep. Variable:	price	**R-squared:**	0.700
Model:	OLS	**Adj. R-squared:**	0.700
Method:	Least Squares	**F-statistic:**	2798.
Date:	Fri, 22 Oct 2021	**Prob (F-statistic):**	0.00
Time:	10:58:32	**Log-Likelihood:**	-2.9439e+05
No. Observations:	21597	**AIC:**	5.888e+05
Df Residuals:	21578	**BIC:**	5.890e+05
Df Model:	18		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	6.656e+06	2.93e+06	2.271	0.023	9.12e+05	1.24e+07
bedrooms	-3.589e+04	1900.636	-18.882	0.000	-3.96e+04	-3.22e+04
bathrooms	4.162e+04	3262.735	12.755	0.000	3.52e+04	4.8e+04
sqft_living	102.8320	18.076	5.689	0.000	67.401	138.263
sqft_lot	0.1281	0.048	2.673	0.008	0.034	0.222
floors	7000.6499	3600.684	1.944	0.052	-56.956	1.41e+04
waterfront	6.181e+05	1.81e+04	34.082	0.000	5.83e+05	6.54e+05
view	5.329e+04	2122.073	25.111	0.000	4.91e+04	5.74e+04
condition	2.685e+04	2346.458	11.443	0.000	2.23e+04	3.15e+04
grade	9.691e+04	2160.604	44.853	0.000	9.27e+04	1.01e+05
sqft_above	78.0057	18.067	4.318	0.000	42.593	113.419
sqft_basement	47.6352	17.907	2.660	0.008	12.537	82.734
yr_built	-2637.5367	71.932	-36.667	0.000	-2778.528	-2496.545
yr_renovated	23.5856	3.977	5.930	0.000	15.790	31.381
zipcode	-584.4582	32.991	-17.716	0.000	-649.122	-519.794
lat	6.007e+05	1.07e+04	55.939	0.000	5.8e+05	6.22e+05
long	-2.177e+05	1.32e+04	-16.546	0.000	-2.43e+05	-1.92e+05
sqft_living15	21.5139	3.451	6.235	0.000	14.751	28.277
sqft_lot15	-0.3877	0.073	-5.285	0.000	-0.531	-0.244

Omnibus:	18343.380	**Durbin-Watson:**	1.989
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1833646.592
Skew:	3.563	**Prob(JB):**	0.00
Kurtosis:	47.575	**Cond. No.**	2.15e+08

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.15e+08. This might indicate that there are
strong multicollinearity or other numerical problems.

In [19]:
```python
# use a qqplot to check for normality
# ha clearly we can't use this model
fig = sm.graphics.qqplot(model_1.resid, dist=stats.norm, line='45', fit=True)
```



In [ ]:

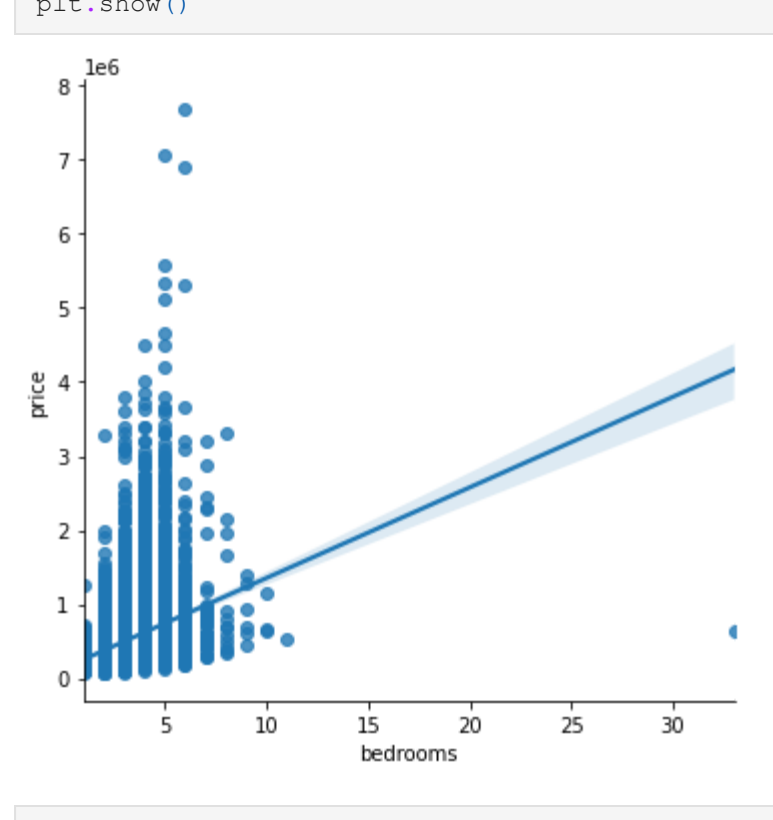
In [20]:

In [ ]:

Dropping Outliers

Bedrooms

In [21]:
```python
sns.lmplot(x='bedrooms', y='price', data=house_data)
plt.show()
```



In [22]:
```python
house_data['bedrooms'].value_counts()
```

Out[22]:
```
3     9824
4     6882
2     2760
5     1601
6      272
1      196
7       38
8       13
9        6
10       3
11       1
33       1
Name: bedrooms, dtype: int64
```

In [23]:
```python
# drop the bedrooms that have the least amount of information
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-23-331ec581ba5f> in <module>
      1 # drop the bedrooms that have the least amount of information
----> 2 house_data[house_data.columns['bedrooms'] == 33]

~\anaconda3\envs\learn-env\lib\site-packages\pandas\core\indexes\base.py in __getitem__(self, key)
   4099         if is_scalar(key):
   4100             key = com.cast_scalar_indexer(key, warn_float=True)
-> 4101             return getitem(key)
   4102 
   4103         if isinstance(key, slice):

IndexError: only integers, slices (`:`), ellipsis (`...`), numpy.newaxis (`None`) and integer or boolean arrays are valid indices
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```python
corr = house_data[pred_features].corr()
```

- Dark red represents strong positive linear relationships
- Dark blue presents strong negative linear relationships
- Help explain Multicollinearity

In [ ]:
```python
# check outliers and remove them
# look at dates maybe add a months colmn
# decide which features to take a closer look at
```

Feature Engineering

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```python
# visualize the correlation between features
# helps understand the multicollinearity
corr = house_data.corr()
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)] = True
with sns.axes_style("white"):
    f, ax = plt.subplots(figsize=(15, 15))
    ax = sns.heatmap(corr, mask=mask, vmax=.3, square=True, cmap='coolwarm', annot=True)
```

- Dark red represents strong positive linear relationships
- Dark blue presents strong negative linear relationships
- Help explain Multicollinearity

In [ ]: