King County Housing Data: Multiple Linear Regression Project

Laura Miller

Full time

Column Names and descriptions for Kings County Data Set

- **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** - Quality of view from house
- **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
%matplotlib inline

sns.set_style('darkgrid')

import warnings
warnings.filterwarnings('ignore')

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

Out[1]:

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

5 rows × 21 columns

In [2]:
```
df.info()
```

```
<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
```

In [3]:
```
df.describe()
```

Out[3]:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	2
count	2.159700e+04	2.159700e+04	21597.000000	21597.000000	21597.000000	2.159700e+04	
mean	4.580474e+09	5.402966e+05	3.373200	2.115826	2080.321850	1.509941e+04	
std	2.876736e+09	3.673681e+05	0.926299	0.768984	918.106125	4.141264e+04	
min	1.000102e+06	7.800000e+04	1.000000	0.500000	370.000000	5.200000e+02	
25%	2.123049e+09	3.220000e+05	3.000000	1.750000	1430.000000	5.040000e+03	
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068500e+04	
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	

Scrub data

Initial observations

- Date is dtype string
- Sqft_basement is dtype object
- yr_renovated is float
- Waterfront contains missing values
- yr_renovated contains missing values
- view contains missing values

Examine date column

In [4]:
```
import datetime as dt
df['month'] = pd.to_datetime(df['date']).dt.month
df['yr_sold'] = pd.to_datetime(df['date']).dt.year
#df['date'] = pd.to_datetime(df['date']).map(dt.datetime.toordinal)
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 23 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  
 21  month          21597 non-null  int64  
 22  yr_sold        21597 non-null  int64  
dtypes: float64(8), int64(13), object(2)
memory usage: 3.8+ MB
```

In [6]:
```
df.date.value_counts()
```

Out[6]:
```
6/23/2014    142
6/26/2014    131
6/25/2014    131
7/8/2014     127
4/27/2015    126
            ...
5/24/2015      1
```

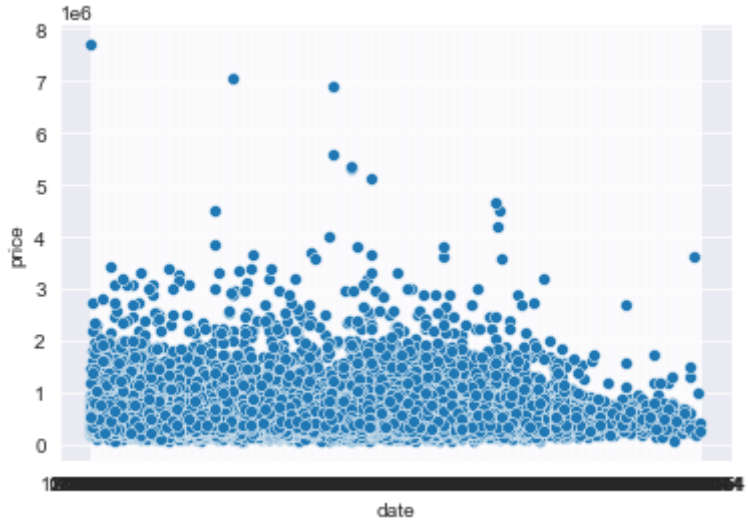
```
5/27/2015      1
7/27/2014      1
8/3/2014       1
8/30/2014      1
Name: date, Length: 372, dtype: int64
```

In [7]:
```
df.date.describe()
```

Out[7]:
```
count          21597
unique           372
top       6/23/2014
freq             142
Name: date, dtype: object
```

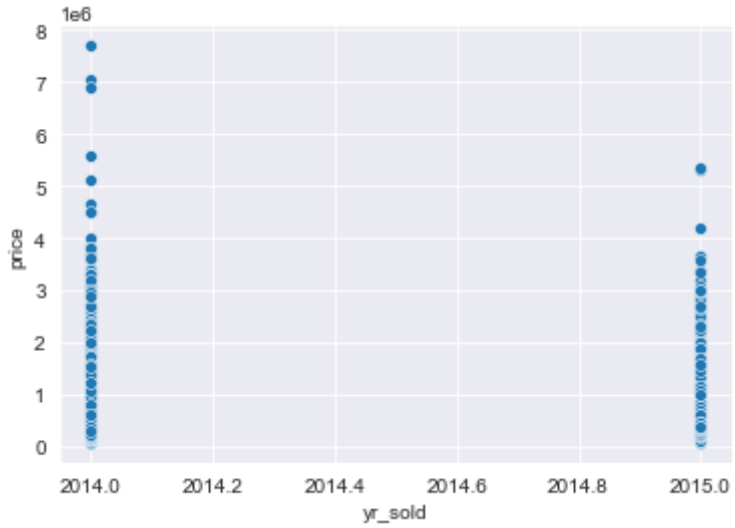
In [8]:
```
sns.scatterplot('date', 'price', data=df)
```

Out[8]:
```
<AxesSubplot:xlabel='date', ylabel='price'>
```



In [9]:
```
sns.scatterplot('yr_sold', 'price', data=df)
```

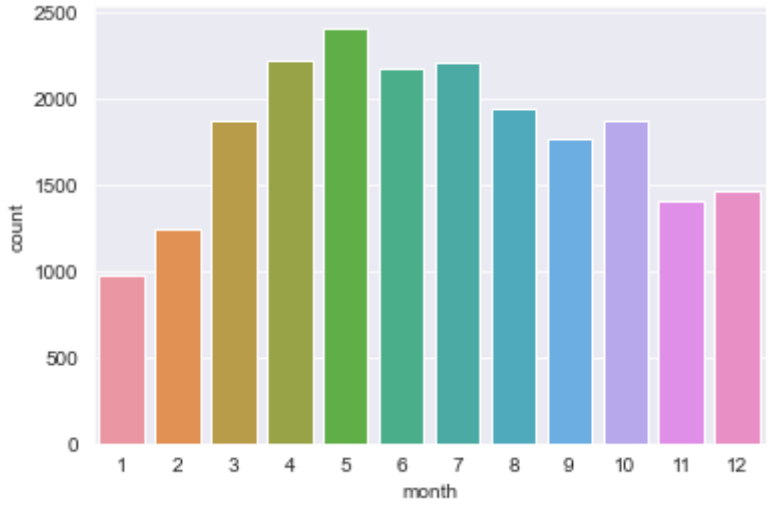
Out[9]:
```
<AxesSubplot:xlabel='yr_sold', ylabel='price'>
```



Price by yr_sold does not look too interesting (perhaps a slight negative trend), as there are only 2 years on record. yr_sold can be used in calculations to create new features.

In [10]:
```
sns.countplot('month', data=df)
```

Out[10]: <AxesSubplot:xlabel='month', ylabel='count'>



In [11]:
```
month_counts = df.month.value_counts().sort_index()
```

In [12]:
```
type(month_counts)
```

Out[12]: pandas.core.series.Series

In [13]:
```
months = list(month_counts.reset_index().iloc[:, 0])
counts = list(month_counts.reset_index().iloc[:, 1])
```

In [14]:
```
counts
```

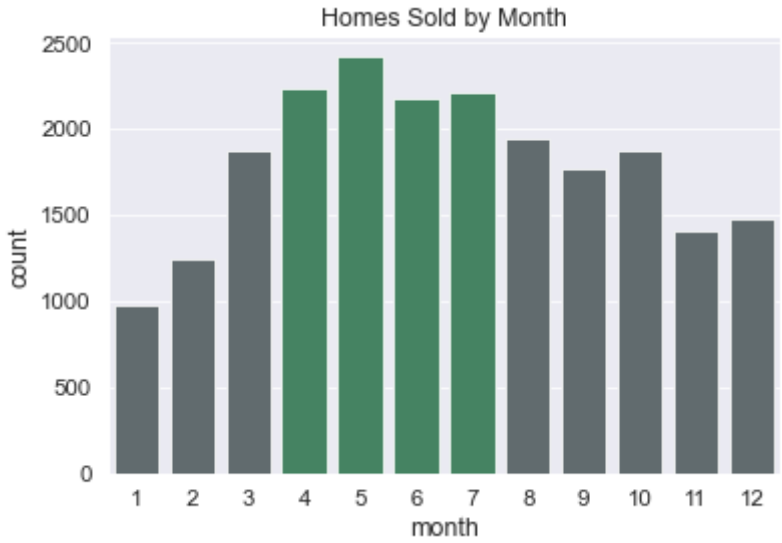
Out[14]: [978, 1247, 1875, 2229, 2414, 2178, 2211, 1939, 1771, 1876, 1409, 1470]

In [15]:
```
sorted_counts = sorted(counts, reverse=True)
sorted_counts
```

Out[15]: [2414, 2229, 2211, 2178, 1939, 1876, 1875, 1771, 1470, 1409, 1247, 978]

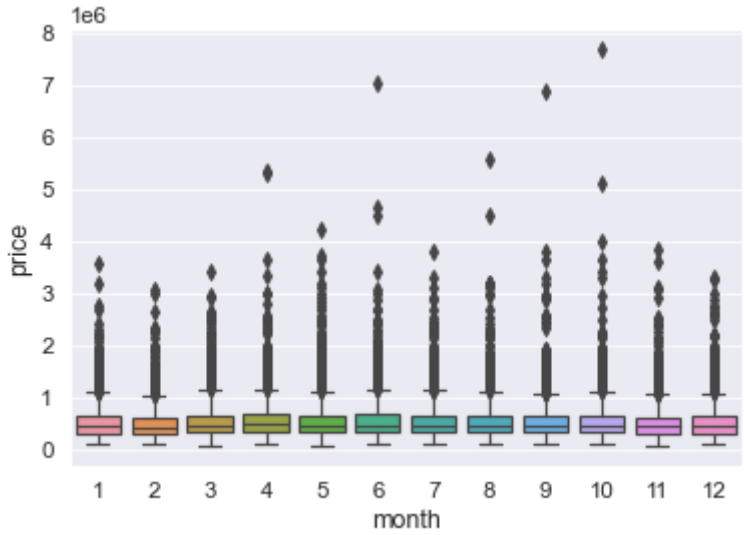
In [16]:
```
sns.set_context('paper', font_scale = 1.3)

top_values = sorted_counts[:4]
clrs = ['#3b8d61' if (x in top_values) else '#5f6c70' for x in counts]
ax = sns.barplot(x=months, y=counts, palette=clrs)
ax.set_title('Homes Sold by Month')
ax.set_xlabel('month')
ax.set_ylabel('count')
plt.savefig('images/count-month.png')
plt.savefig('images/hr-count-month.png', dpi=200);
```



There are certain months where more houses are sold (late spring to early summer). Are there certain months where houses sell for more?

In [17]:
```
sns.boxplot('month', 'price', data=df, showfliers=True);
```

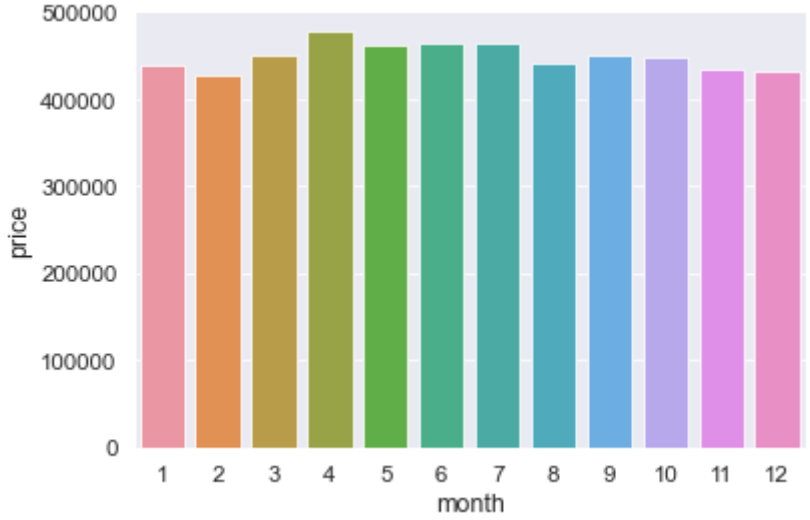


There are lots of outliers for each month. Let's look at the mean price for each month as well as the median price. Median may be more reflective of the data for each month, as the data is right-skewed

In [18]:
```
# Mean price
ax = sns.barplot('month', 'price', data=df, color='#3b8d61')
ax.set_title('Average Price by Month');
```

In [19]:
```
# Median price
sns.barplot('month', 'price', data = df.groupby('month').median().reset_index())
```

Out[19]:
```
<AxesSubplot:xlabel='month', ylabel='price'>
```



There aren't any months where homes seem to sell for significantly more

In [20]:
```
df.groupby('month').median()
```

Out[20]:

month	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront
1	3.905036e+09	438500.0	3.0	2.25	1890.0	7800.0	1.0	0.0
2	3.997500e+09	426500.0	3.0	2.00	1830.0	7667.0	1.0	0.0
3	3.905040e+09	450000.0	3.0	2.25	1870.0	7560.0	1.0	0.0
4	4.022900e+09	477000.0	3.0	2.25	1900.0	7500.0	1.5	0.0
5	3.905005e+09	462000.0	3.0	2.25	1930.0	7498.0	1.0	0.0
6	3.904925e+09	465000.0	3.0	2.25	1980.0	7700.0	1.5	0.0
7	3.885801e+09	465000.0	3.0	2.25	1950.0	7695.0	1.5	0.0

month	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront
8	4.016800e+09	442200.0	3.0	2.25	1940.0	7810.0	1.0	0.0
9	3.826501e+09	450000.0	3.0	2.25	1920.0	7620.0	1.5	0.0
10	3.832180e+09	447000.0	3.0	2.25	1905.0	7413.0	1.5	0.0
11	3.876312e+09	435000.0	3.0	2.00	1870.0	7500.0	1.5	0.0
12	3.972001e+09	432500.0	3.0	2.25	1900.0	7725.0	1.0	0.0

You could potentially get 30K - 40K more for a house by waiting to sell during April - July.

Missing values in waterfront

In [21]:
```
df.waterfront.value_counts(normalize=True)
```

Out[21]:
```
0.0    0.992404
1.0    0.007596
Name: waterfront, dtype: float64
```

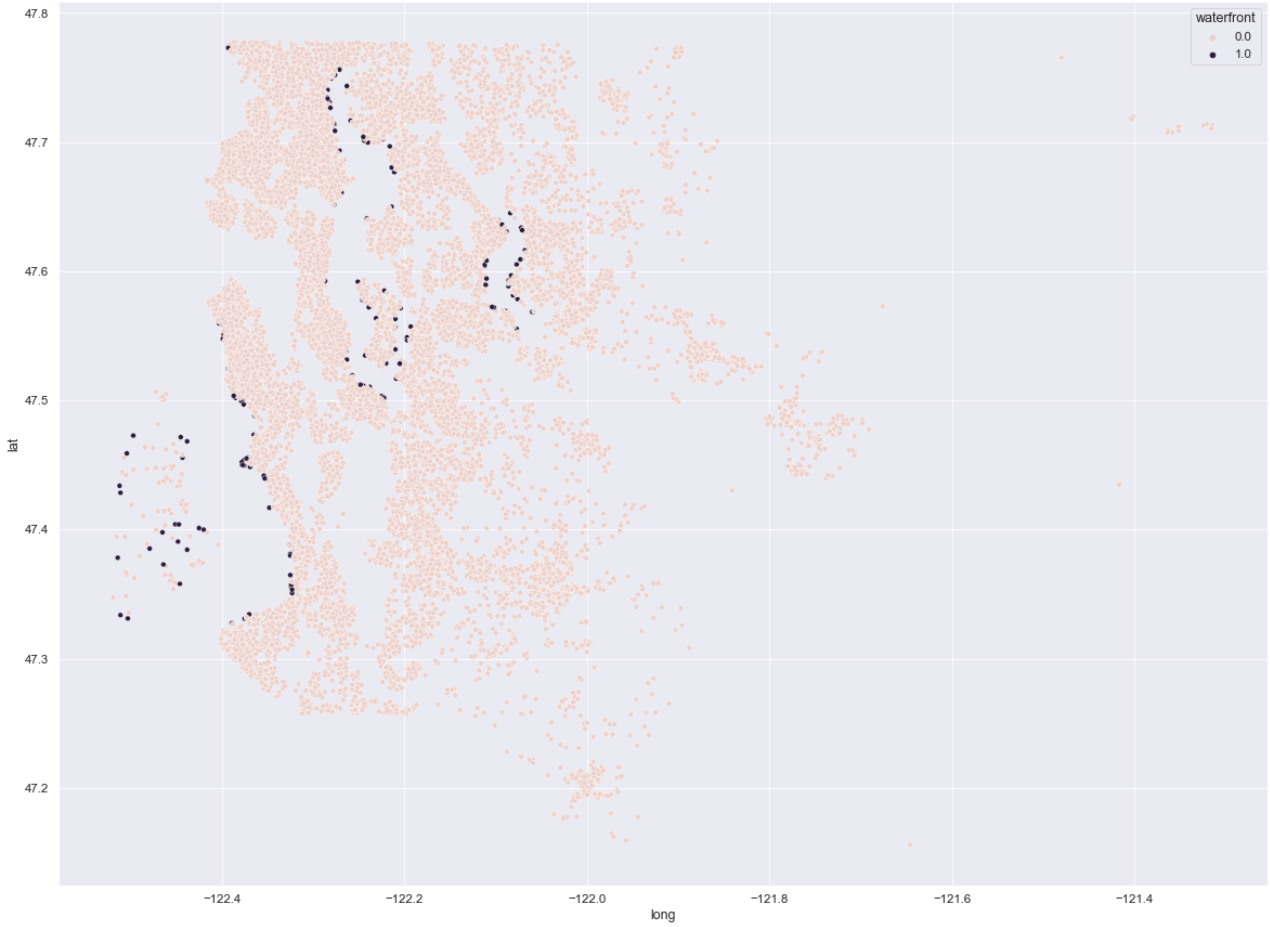
In [22]:
```
df.isna().sum()
```

Out[22]:
```
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
month             0
yr_sold           0
dtype: int64
```

Under 1% of houses have a waterfront. Most do not. There are 2376 missing values

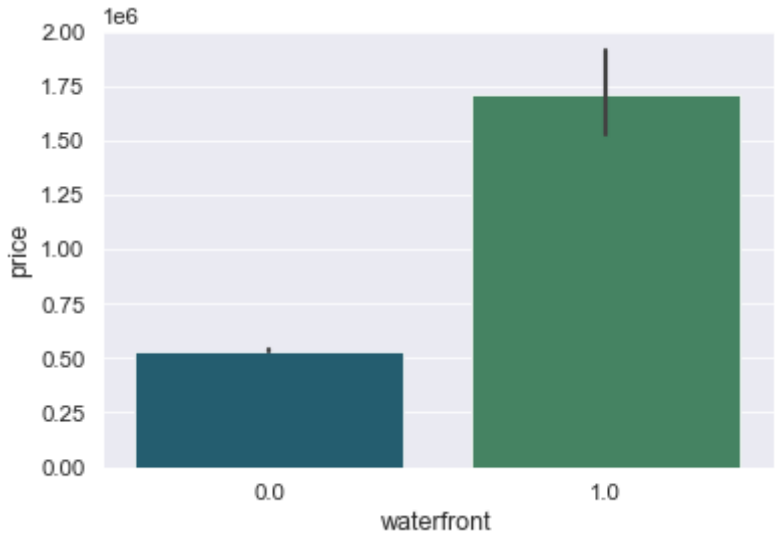
In [23]:
```
plt.figure(figsize=(20, 15))
sns.scatterplot(x = "long", y = "lat", hue = "waterfront",  data = df)
```

Out[23]:
```
<AxesSubplot:xlabel='long', ylabel='lat'>
```



In [24]:
```
clrs = ['#18637b', '#3b8d61']
sns.barplot('waterfront', 'price', data=df, palette=clrs)
```

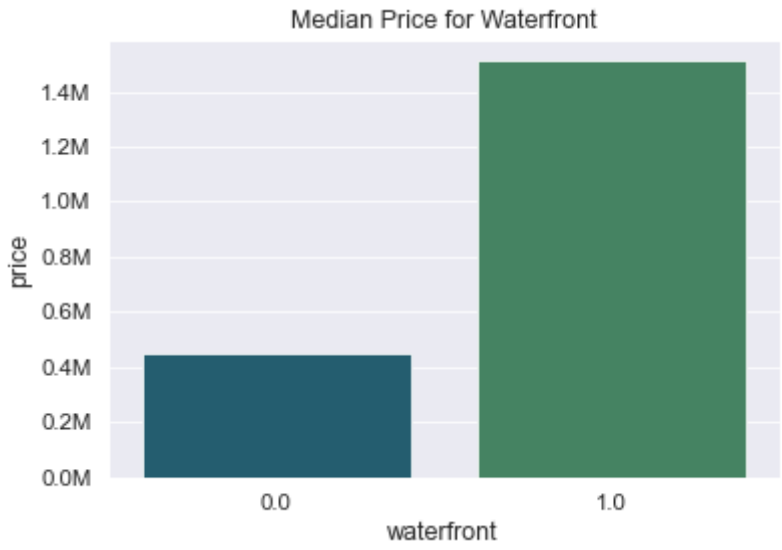
Out[24]: <AxesSubplot:xlabel='waterfront', ylabel='price'>



In [25]:
```
from matplotlib import ticker
```

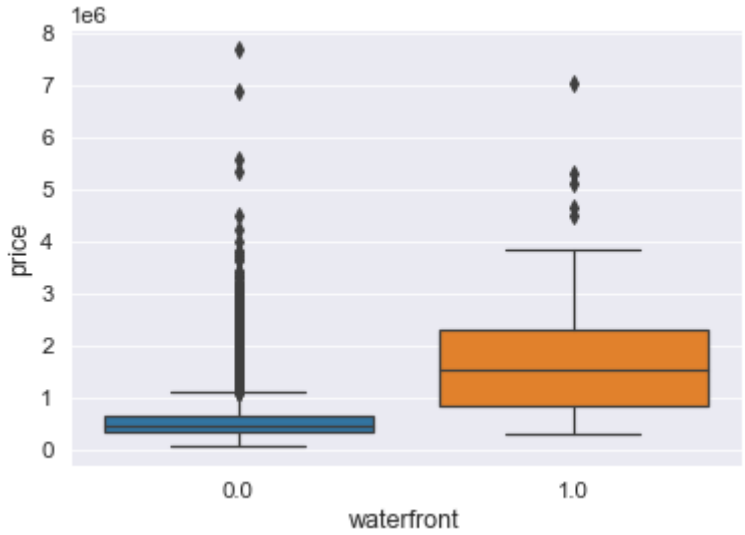
In [26]:
```
clrs = ['#18637b', '#3b8d61']
ax = sns.barplot('waterfront', 'price', data = df.groupby('waterfront').median()
ax.set_title('Median Price for Waterfront')
```

```
ax.yaxis.get_major_formatter().set_scientific(False)
ax.yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '{:,.1f}'.forma
```



In [27]:
```
sns.boxplot('waterfront', 'price', data=df)
```

Out[27]:
```
<AxesSubplot:xlabel='waterfront', ylabel='price'>
```



In [28]:
```
np.random.seed(123)
s = df.waterfront.value_counts(normalize=True)
missing = df.waterfront.isnull()
df.loc[missing, 'waterfront'] = np.random.choice(s.index, size=len(df[missing]),
```

In [29]:
```
df.waterfront.isnull().sum()
```

Out[29]:
```
0
```

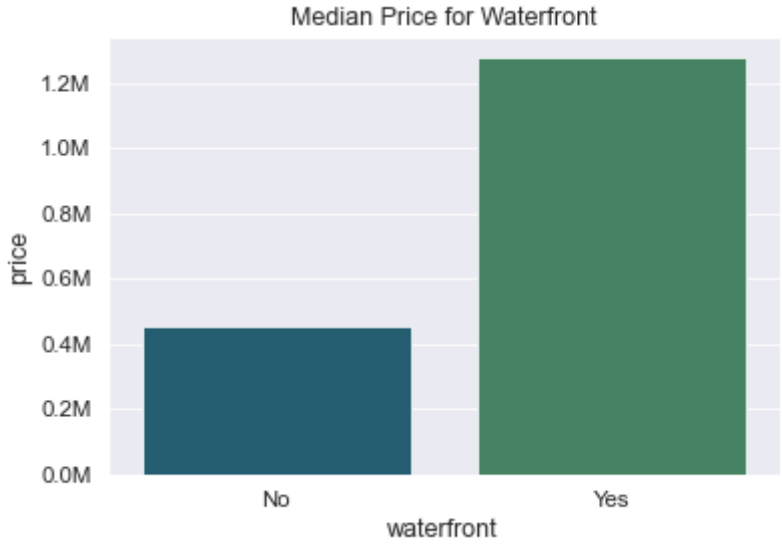
In [30]:
```
df.waterfront.value_counts(normalize=True)
```

Out[30]:
```
0.0    0.992221
1.0    0.007779
Name: waterfront, dtype: float64
```

In [31]:
```
clrs = ['#18637b', '#3b8d61']
```

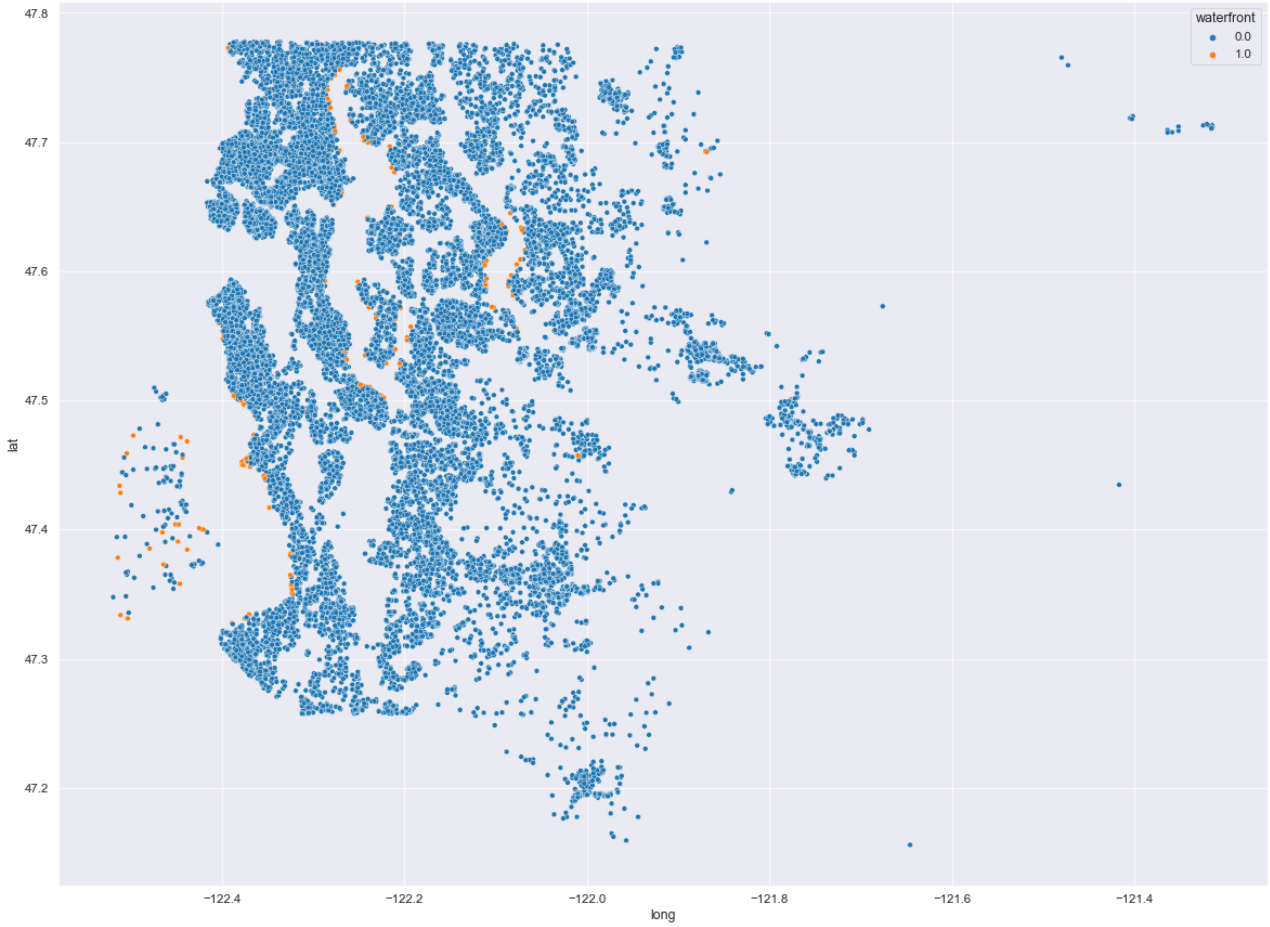
```
ax = sns.barplot('waterfront', 'price', data = df.groupby('waterfront').median()
ax.set_title('Median Price for Waterfront')

ax.yaxis.get_major_formatter().set_scientific(False)
ax.yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '{:,.1f}'.forma
ax.set_xticklabels(['No', 'Yes'])
plt.savefig('images/waterfront.png')
plt.savefig('images/hr-waterfront.png', dpi=200)
```



In [32]:
```
plt.figure(figsize=(20, 15))
sns.scatterplot(x = "long", y = "lat", hue = "waterfront",  data = df)
```

Out[32]:
```
<AxesSubplot:xlabel='long', ylabel='lat'>
```



The reassigned values look pretty good. Most of the orange points lie on the waterfront.

In [33]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 23 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     21597 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  
 21  month          21597 non-null  int64  
 22  yr_sold        21597 non-null  int64  
```

```
dtypes: float64(8), int64(13), object(2)
memory usage: 3.8+ MB
```

sqft_basement

In [34]:
```
df.sqft_basement.value_counts(normalize=True)
```

Out[34]:
```
0.0       0.593879
?         0.021021
600.0     0.010048
500.0     0.009677
700.0     0.009631
            ...
2850.0    0.000046
2610.0    0.000046
2310.0    0.000046
1960.0    0.000046
2190.0    0.000046
Name: sqft_basement, Length: 304, dtype: float64
```

'?' appears to be a placeholder value. The majority of homes (over half) do not have a basement. For the missing data, I will assume that these homes have no basement or the basements are unfinished/unuseable. 0.0 also happens to be the median value.

In [35]:
```
df.sqft_basement.replace('?', np.nan, inplace=True)
df.sqft_basement.fillna(0.0, inplace=True)
df.sqft_basement = df.sqft_basement.astype(float)
```

In [36]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 23 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     21597 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  float64
 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  
 21  month          21597 non-null  int64  
 22  yr_sold        21597 non-null  int64  
dtypes: float64(9), int64(13), object(1)
memory usage: 3.8+ MB
```

In [37]:
```
df.sqft_basement.value_counts(normalize=True)
```

Out[37]:
```
0.0       0.614900
600.0     0.010048
500.0     0.009677
700.0     0.009631
800.0     0.009307
            ...
915.0     0.000046
295.0     0.000046
1281.0    0.000046
2130.0    0.000046
906.0     0.000046
Name: sqft_basement, Length: 303, dtype: float64
```

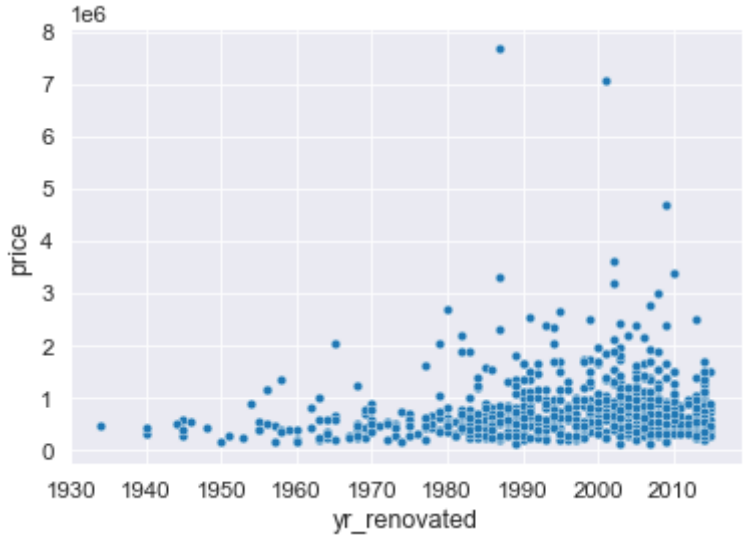
yr_renovated

In [38]:
```
df.yr_renovated.value_counts(normalize=True)
```

Out[38]:
```
0.0       0.958096
2014.0    0.004112
2003.0    0.001746
2013.0    0.001746
2007.0    0.001690
            ...
1946.0    0.000056
1959.0    0.000056
1971.0    0.000056
1951.0    0.000056
1954.0    0.000056
Name: yr_renovated, Length: 70, dtype: float64
```

In [39]:
```
sns.scatterplot('yr_renovated', 'price', data=df[df['yr_renovated'] > 0])
```

Out[39]:
```
<AxesSubplot:xlabel='yr_renovated', ylabel='price'>
```



In [40]:
```
df.yr_renovated.isna().sum()
```

Out[40]:
```
3842
```

Lots of missing values. Let's treat these houses as though they were never renovated, filling with the median which is 0.0. I am assuming that renovated houses are likely to have some record of their renovation.

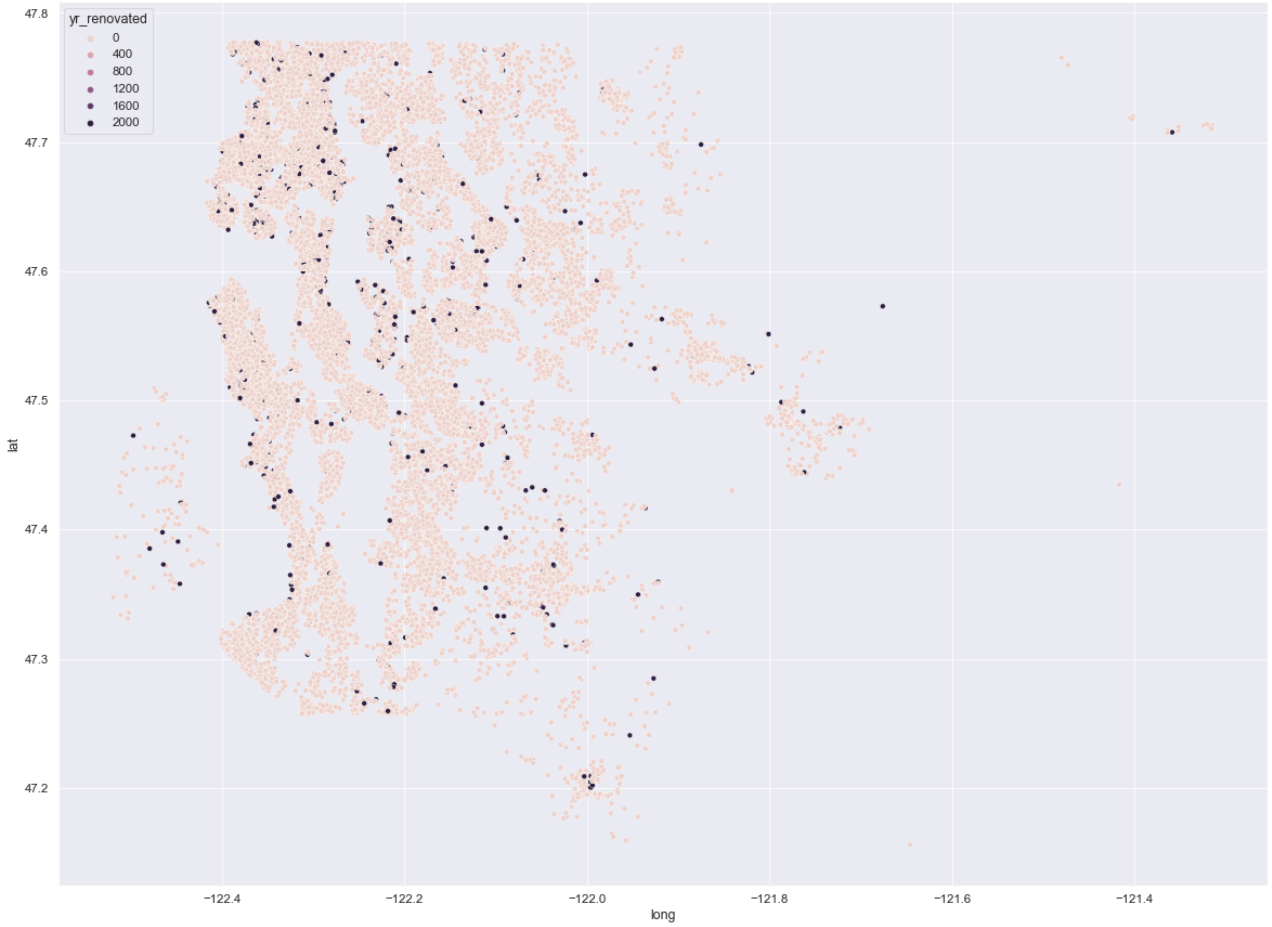
In [41]:
```
df.yr_renovated.fillna(value=0.0, inplace=True)
```

In [42]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 23 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     21597 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  float64
 14  yr_built       21597 non-null  int64  
 15  yr_renovated   21597 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  
 21  month          21597 non-null  int64  
 22  yr_sold        21597 non-null  int64  
dtypes: float64(9), int64(13), object(1)
memory usage: 3.8+ MB
```

In [43]:
```
plt.figure(figsize=(20, 15))
sns.scatterplot(x = "long", y = "lat", hue = "yr_renovated",  data = df)
```

Out[43]:
```
<AxesSubplot:xlabel='long', ylabel='lat'>
```



Many of the renovated houses are close to the waterfront, but there are plenty of homes that are not.

In [44]:
```
df.groupby('yr_renovated').count()
```

Out[44]:

yr_renovated	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfro
0.0	20853	20853	20853	20853	20853	20853	20853	20853	208
1934.0	1	1	1	1	1	1	1	1	
1940.0	2	2	2	2	2	2	2	2	
1944.0	1	1	1	1	1	1	1	1	
1945.0	3	3	3	3	3	3	3	3	
...	...	...	...	...	...	...	...	...	
2011.0	9	9	9	9	9	9	9	9	
2012.0	8	8	8	8	8	8	8	8	
2013.0	31	31	31	31	31	31	31	31	
2014.0	73	73	73	73	73	73	73	73	
2015.0	14	14	14	14	14	14	14	14	

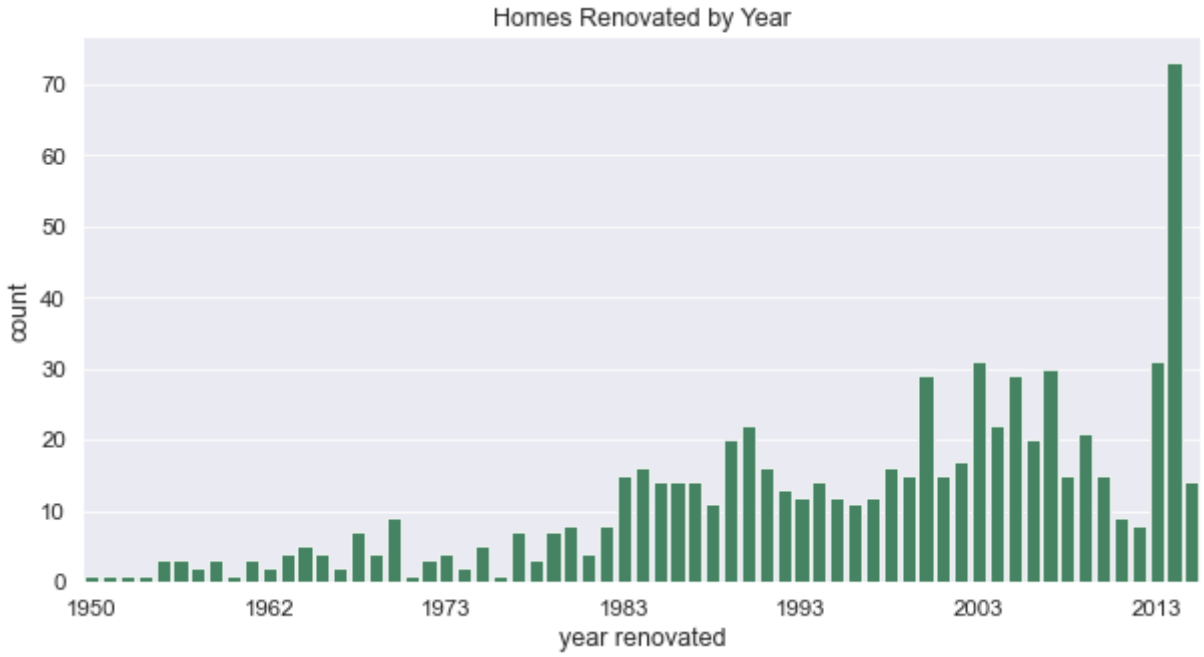
70 rows × 22 columns

In [45]:
```
df['yr_renovated'] = df['yr_renovated'].astype(int)
```

In [46]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 23 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     21597 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  float64
 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  
 21  month          21597 non-null  int64  
 22  yr_sold        21597 non-null  int64  
dtypes: float64(8), int64(14), object(1)
memory usage: 3.8+ MB
```

In [47]:
```
plt.figure(figsize=(10,5))
ax = sns.countplot(x = "yr_renovated", data = df[df['yr_renovated'] >= 1950], co
ax.set_xlabel('year renovated')
ax.set_title('Homes Renovated by Year')
#ax.xaxis.set_ticks(np.arange(1950, 2015, 5))
ax.xaxis.set_major_locator(ticker.MultipleLocator(10))
#ax.xaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '{:,.0f}'.form
#plt.xticks(rotation=90)
plt.savefig('images/count-yr-renovated.png')
plt.savefig('images/hr-count-yr-renovated.png', dpi=200);
```



There have been more renovations in more recent years. Might also have to do with the fact that the number of homes in the county have increased over the years. In any case, renovation seems to be key for improving home value in recent years.

View

In [48]:
```
df.view.value_counts(normalize=True)
```

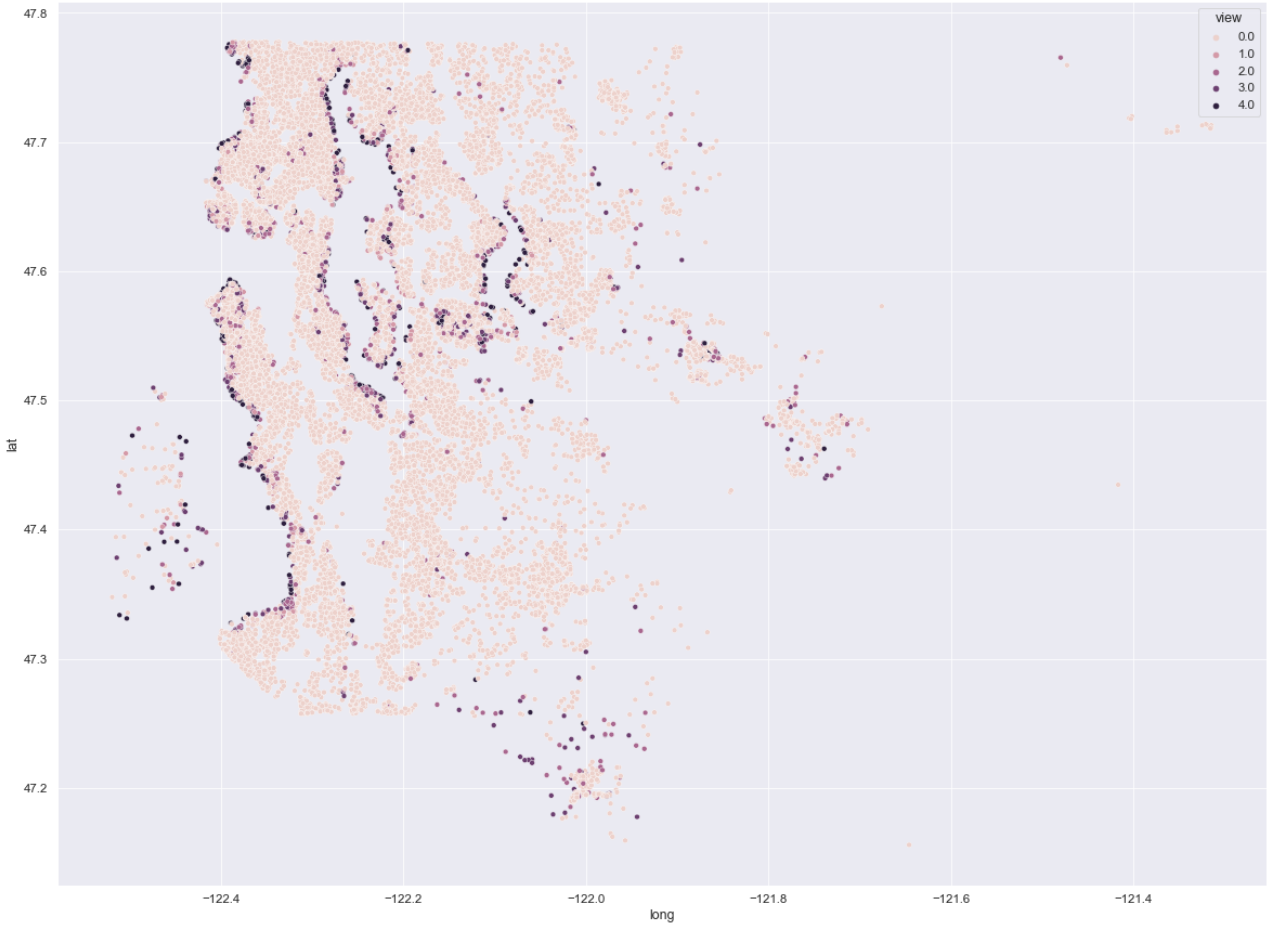
Out[48]:
```
0.0    0.901923
2.0    0.044441
3.0    0.023591
1.0    0.015325
4.0    0.014721
Name: view, dtype: float64
```

In [49]:
```
df.view.isna().sum()
```

Out[49]:
```
63
```

In [50]:
```
plt.figure(figsize=(20, 15))
sns.scatterplot(x = "long", y = "lat", hue = "view",  data = df)
```

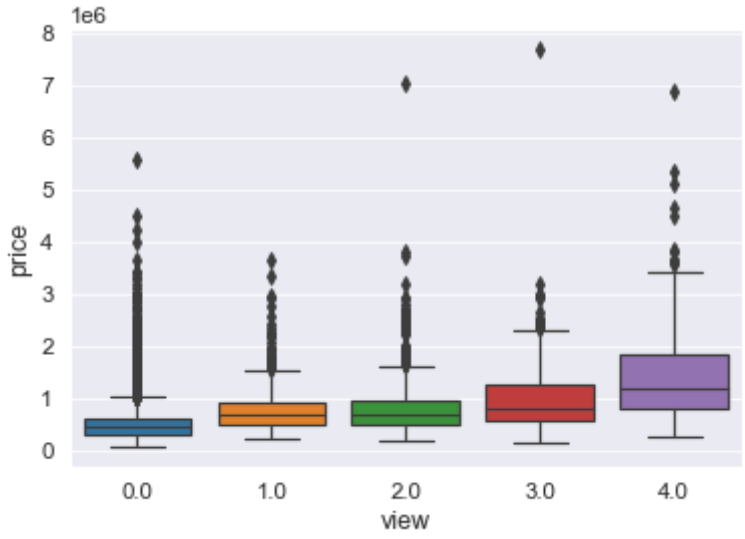
Out[50]:
```
<AxesSubplot:xlabel='long', ylabel='lat'>
```



Many homes with views are also on the waterfront. We can fill in the missing values based on the current distribution of view ratings, or we can just fill them with zero since there aren't that many values. Plus the majority of views are rated at 0.0, which is also the median of the view column.

In [51]:
```
sns.boxplot(x='view', y='price', data=df)
```

Out[51]:
```
<AxesSubplot:xlabel='view', ylabel='price'>
```



In [52]:
```
df.view.isna().sum()
```

Out[52]: 63

In [53]:
```
ax = sns.barplot('view', 'price', data=df, color='#3b8d61')
ax.set_title('Average Price by View Score')
ax.yaxis.get_major_formatter().set_scientific(False)
ax.yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '{:,.1f}'.forma
plt.savefig('images/price-view.png')
plt.savefig('images/hr-price-view.png', dpi=200);
```

Higher median and mean prices for homes with better views. Homes with view scores of 1.0 and 2.0 seem to be comparable in price

In [54]:
```
df.view.fillna(value=0.0, inplace=True)
```

In [55]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 23 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     21597 non-null  float64
 9   view           21597 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  float64
 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  
 21  month          21597 non-null  int64  
```

```
 22  yr_sold        21597 non-null  int64
dtypes: float64(8), int64(14), object(1)
memory usage: 3.8+ MB
```

Check duplicates

id is a unique identifier for a home

In [56]:
```
df[df.duplicated('id', keep=False)].head(30)
```

Out[56]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	wa
93	6021501535	7/25/2014	430000.0	3	1.50	1580	5000	1.0	
94	6021501535	12/23/2014	700000.0	3	1.50	1580	5000	1.0	
313	4139480200	6/18/2014	1380000.0	4	3.25	4290	12103	1.0	
314	4139480200	12/9/2014	1400000.0	4	3.25	4290	12103	1.0	
324	7520000520	9/5/2014	232000.0	2	1.00	1240	12092	1.0	
325	7520000520	3/11/2015	240500.0	2	1.00	1240	12092	1.0	
345	3969300030	7/23/2014	165000.0	4	1.00	1000	7134	1.0	
346	3969300030	12/29/2014	239900.0	4	1.00	1000	7134	1.0	
371	2231500030	10/1/2014	315000.0	4	2.25	2180	10754	1.0	
372	2231500030	3/24/2015	530000.0	4	2.25	2180	10754	1.0	
717	8820903380	7/28/2014	452000.0	6	2.25	2660	13579	2.0	
718	8820903380	1/2/2015	730000.0	6	2.25	2660	13579	2.0	
823	726049190	10/2/2014	287500.0	3	1.00	1810	7200	1.0	
824	726049190	2/18/2015	431000.0	3	1.00	1810	7200	1.0	
836	8682262400	7/18/2014	430000.0	2	1.75	1350	4003	1.0	
837	8682262400	5/13/2015	419950.0	2	1.75	1350	4003	1.0	
1084	9834200885	7/17/2014	360000.0	4	2.50	2080	4080	1.0	
1085	9834200885	4/20/2015	550000.0	4	2.50	2080	4080	1.0	
1127	8062900070	9/9/2014	272000.0	5	1.50	2550	6300	1.0	
1128	8062900070	2/13/2015	369000.0	5	1.50	2550	6300	1.0	
1200	1231000510	9/22/2014	263000.0	3	1.75	1490	3800	1.0	
1201	1231000510	5/4/2015	510000.0	3	1.75	1490	3800	1.0	
1232	6117501820	6/18/2014	250275.0	2	1.00	790	11234	1.0	
1233	6117501820	4/28/2015	435000.0	2	1.00	790	11234	1.0	
1448	2228900270	8/12/2014	215000.0	2	1.00	1010	6000	1.0	
1449	2228900270	2/12/2015	302000.0	2	1.00	1010	6000	1.0	
1462	3630120700	5/13/2014	757000.0	3	3.25	3190	5283	2.0	
1463	3630120700	1/7/2015	765000.0	3	3.25	3190	5283	2.0	
1574	7888000390	6/27/2014	140000.0	3	1.00	1060	7473	1.0	

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	wa
1575	7888000390	4/1/2015	235000.0	3	1.00	1060	7473	1.0	

30 rows × 23 columns

Duplicates indicate a property was sold multiple times. Data looks to be sorted by date (ascending) already, so I don't need to sort values prior to dropping duplicates, keeping the most recent date a home was sold.

In [57]:
```
df = df.drop_duplicates(['id'], keep='last')
```

In [58]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21420 entries, 0 to 21596
Data columns (total 23 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             21420 non-null  int64  
 1   date           21420 non-null  object 
 2   price          21420 non-null  float64
 3   bedrooms       21420 non-null  int64  
 4   bathrooms      21420 non-null  float64
 5   sqft_living    21420 non-null  int64  
 6   sqft_lot       21420 non-null  int64  
 7   floors         21420 non-null  float64
 8   waterfront     21420 non-null  float64
 9   view           21420 non-null  float64
 10  condition      21420 non-null  int64  
 11  grade          21420 non-null  int64  
 12  sqft_above     21420 non-null  int64  
 13  sqft_basement  21420 non-null  float64
 14  yr_built       21420 non-null  int64  
 15  yr_renovated   21420 non-null  int64  
 16  zipcode        21420 non-null  int64  
 17  lat            21420 non-null  float64
 18  long           21420 non-null  float64
 19  sqft_living15  21420 non-null  int64  
 20  sqft_lot15     21420 non-null  int64  
 21  month          21420 non-null  int64  
 22  yr_sold        21420 non-null  int64  
dtypes: float64(8), int64(14), object(1)
memory usage: 3.9+ MB
```

New Features

Years since renovation

The difference between year a house was sold and the year it was renovated/ years since renovation at the time a house was sold.

In [59]:
```
df['yrs_since_reno'] = df['yr_sold'] - df['yr_renovated']
```

In [60]:
```
df.head()
```

Out[60]:

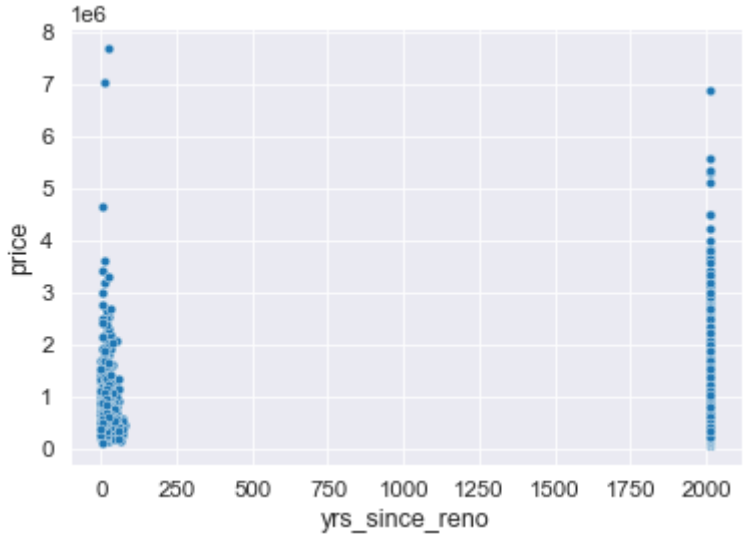
	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfr
0	7129300520	10/13/2014	221900.0	3	1.00	1180	5650	1.0	

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

5 rows × 24 columns

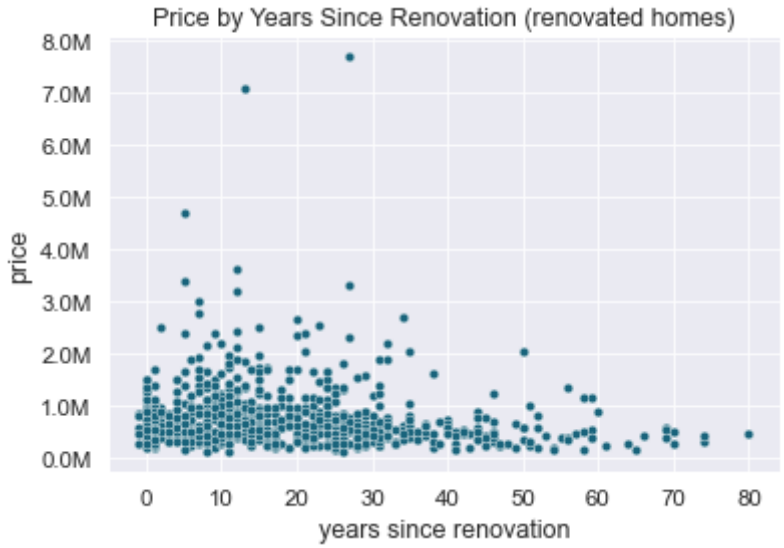
In [61]:
```
sns.scatterplot("yrs_since_reno", "price", data = df)
```

Out[61]: <AxesSubplot:xlabel='yrs_since_reno', ylabel='price'>



Houses that were never renovated have over 2000 values for yrs_since_reno because yr_renovated was equal to 0.0

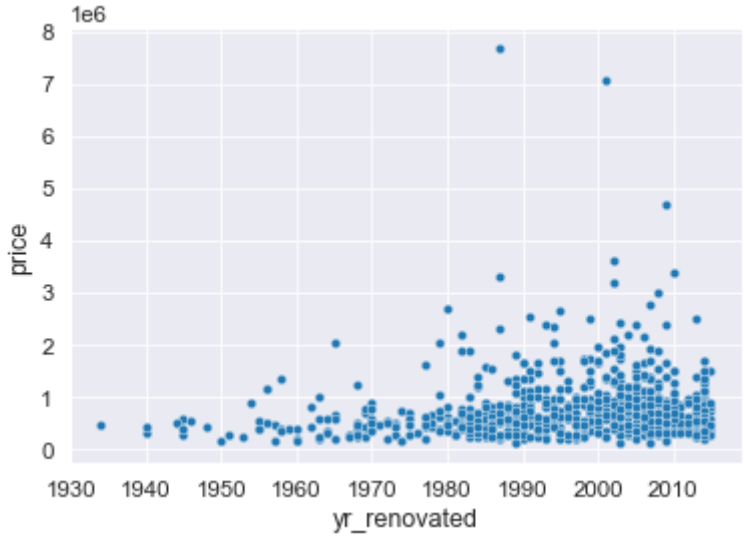
In [62]:
```
ax = sns.scatterplot("yrs_since_reno", "price", data = df[df['yrs_since_reno'] <
ax.set_xlabel('years since renovation')
ax.set_title('Price by Years Since Renovation (renovated homes)')
ax.yaxis.get_major_formatter().set_scientific(False)
ax.yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, pos: '{:,.1f}'.forma
plt.savefig('images/price-yrssincereno.png')
plt.savefig('images/hr-price-yrssincereno.png', dpi=200);
```



Homes with fewer years between their sale date and their last renovation tend to sell for more.

In [63]:
```
sns.scatterplot("yr_renovated", "price", data = df[df['yr_renovated'] > 0])
```

Out[63]:
```
<AxesSubplot:xlabel='yr_renovated', ylabel='price'>
```



Bed-bath ratio

In [64]:
```
df['bed_bath_ratio'] = df['bedrooms']/df['bathrooms']
```

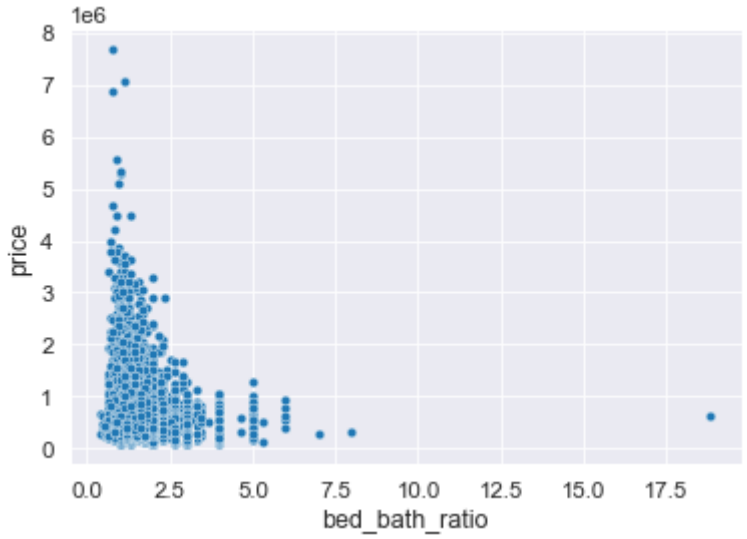
In [65]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21420 entries, 0 to 21596
Data columns (total 25 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              21420 non-null  int64
 1   date            21420 non-null  object
 2   price           21420 non-null  float64
 3   bedrooms        21420 non-null  int64
 4   bathrooms       21420 non-null  float64
 5   sqft_living     21420 non-null  int64
 6   sqft_lot        21420 non-null  int64
 7   floors          21420 non-null  float64
```

```
 8   waterfront      21420 non-null  float64
 9   view            21420 non-null  float64
 10  condition       21420 non-null  int64
 11  grade           21420 non-null  int64
 12  sqft_above      21420 non-null  int64
 13  sqft_basement   21420 non-null  float64
 14  yr_built        21420 non-null  int64
 15  yr_renovated    21420 non-null  int64
 16  zipcode         21420 non-null  int64
 17  lat             21420 non-null  float64
 18  long            21420 non-null  float64
 19  sqft_living15   21420 non-null  int64
 20  sqft_lot15      21420 non-null  int64
 21  month           21420 non-null  int64
 22  yr_sold         21420 non-null  int64
 23  yrs_since_reno  21420 non-null  int64
 24  bed_bath_ratio  21420 non-null  float64
dtypes: float64(9), int64(15), object(1)
memory usage: 4.2+ MB
```

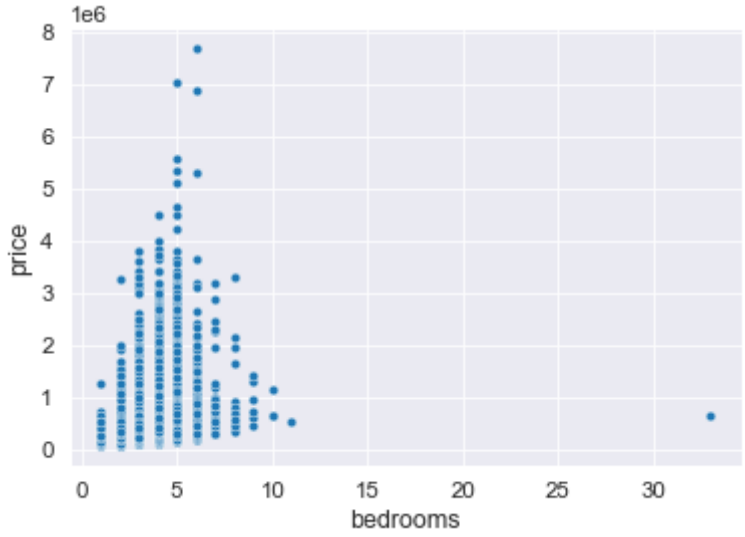
In [66]:
```
sns.scatterplot('bed_bath_ratio', 'price', data=df)
```

Out[66]: <AxesSubplot:xlabel='bed_bath_ratio', ylabel='price'>



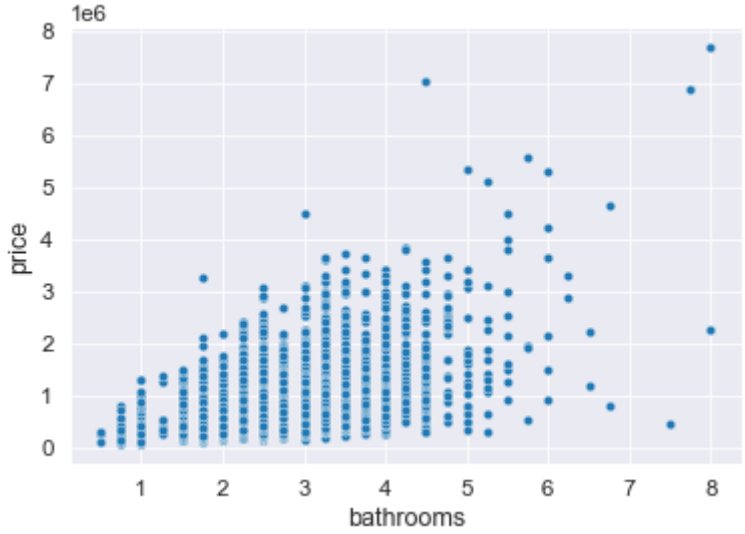
In [67]:
```
sns.scatterplot('bedrooms', 'price', data=df)
```

Out[67]: <AxesSubplot:xlabel='bedrooms', ylabel='price'>



In [68]:
```
sns.scatterplot('bathrooms', 'price', data=df)
```

Out[68]:
```
<AxesSubplot:xlabel='bathrooms', ylabel='price'>
```

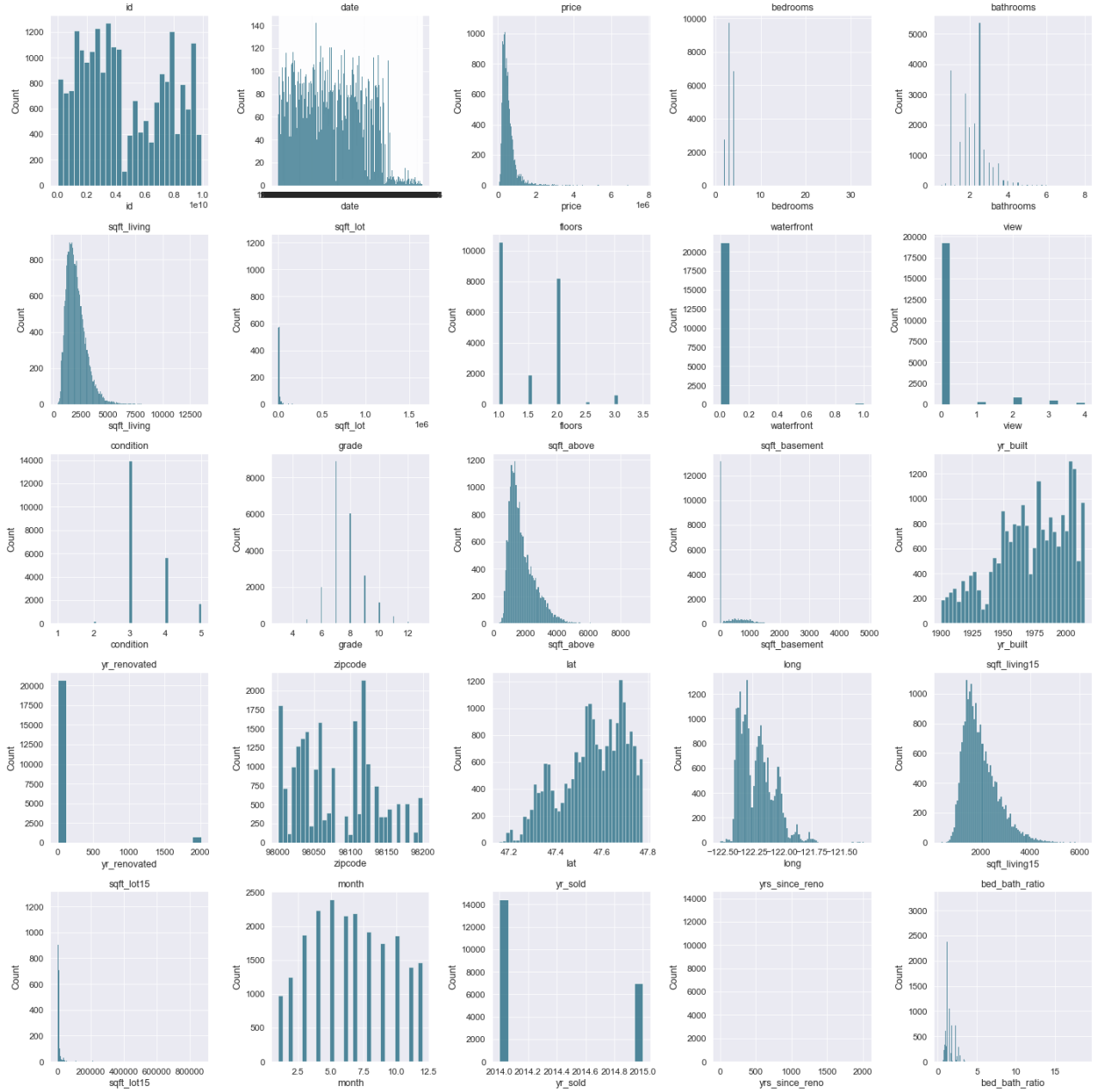


'bedrooms' and 'bed_bath_ratio' seem to have a common outlier, a house that is low priced with over thirty bedrooms. This may be an erroneous entry/ outlier.

Explore Data

In [69]:
```
plt.figure(figsize=(20,20))

for index, col in enumerate(df.columns):
    ax = plt.subplot(5, 5, index+1)
    sns.histplot(x=col, data=df, ax=ax, color='#18637b')
    ax.set_title('{}'.format(col))
plt.tight_layout()
plt.savefig('images/histograms-1.png')
plt.savefig('images/hr-histograms-1.png', dpi=200)
```



Distributions of many columns are skewed ('price', 'sqft_living', 'sqft_lot', 'sqft_above', 'sqft_living15', 'sqft_lot15'). This data will benefit from log transformations down the line after outliers are removed. It may also be beneficial to rescale the data.

In [70]:
```
df.drop(['id'], axis=1, inplace=True)
```

In [71]:
```
df.columns
```

Out[71]:
```
Index(['date', 'price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot',
       'floors', 'waterfront', 'view', 'condition', 'grade', 'sqft_above',
       'sqft_basement', 'yr_built', 'yr_renovated', 'zipcode', 'lat', 'long',
       'sqft_living15', 'sqft_lot15', 'month', 'yr_sold', 'yrs_since_reno',
       'bed_bath_ratio'],
      dtype='object')
```

In [72]:
```
plt.figure(figsize=(18,25))

for index, col in enumerate(df.columns):
    ax = plt.subplot(6, 4, index+1)
```

```
    sns.scatterplot(x=col, y='price', data=df, ax=ax, color='#18637b')
    ax.set_title('{}'.format(col))
plt.tight_layout()
plt.savefig('images/scatterplots-1.png')
plt.savefig('images/hr-scatterplots-1.png', dpi=200)
```

At first glance, 'bathrooms', 'sqft_living', 'grade', 'sqft_above', 'sqft_living15', grade seem to have the most linear relationships with price. There are some outliers in many predictors.

- `bedrooms` : a low pricced property with over 30 bedrooms. I've assumed this was likely a mistake entry.
- `bathrooms` : high and low priced properties with over 5 bathrooms
- `price` : outliers beyond 6M
- outliers in sqft data Some scatter plots have higher pricing in the middle of the plots, where there may be a positive correlation up to a certain value of the predictor variable, followed by a negative correlation. This would mean that the most optimal values for that feature are somewhere in the middle. Features showing this pattern may need to be split into more categories/subfeatures. `bedrooms` is the most obvious example of this

Multicolinearity

In [73]:
```
df.corr()
```

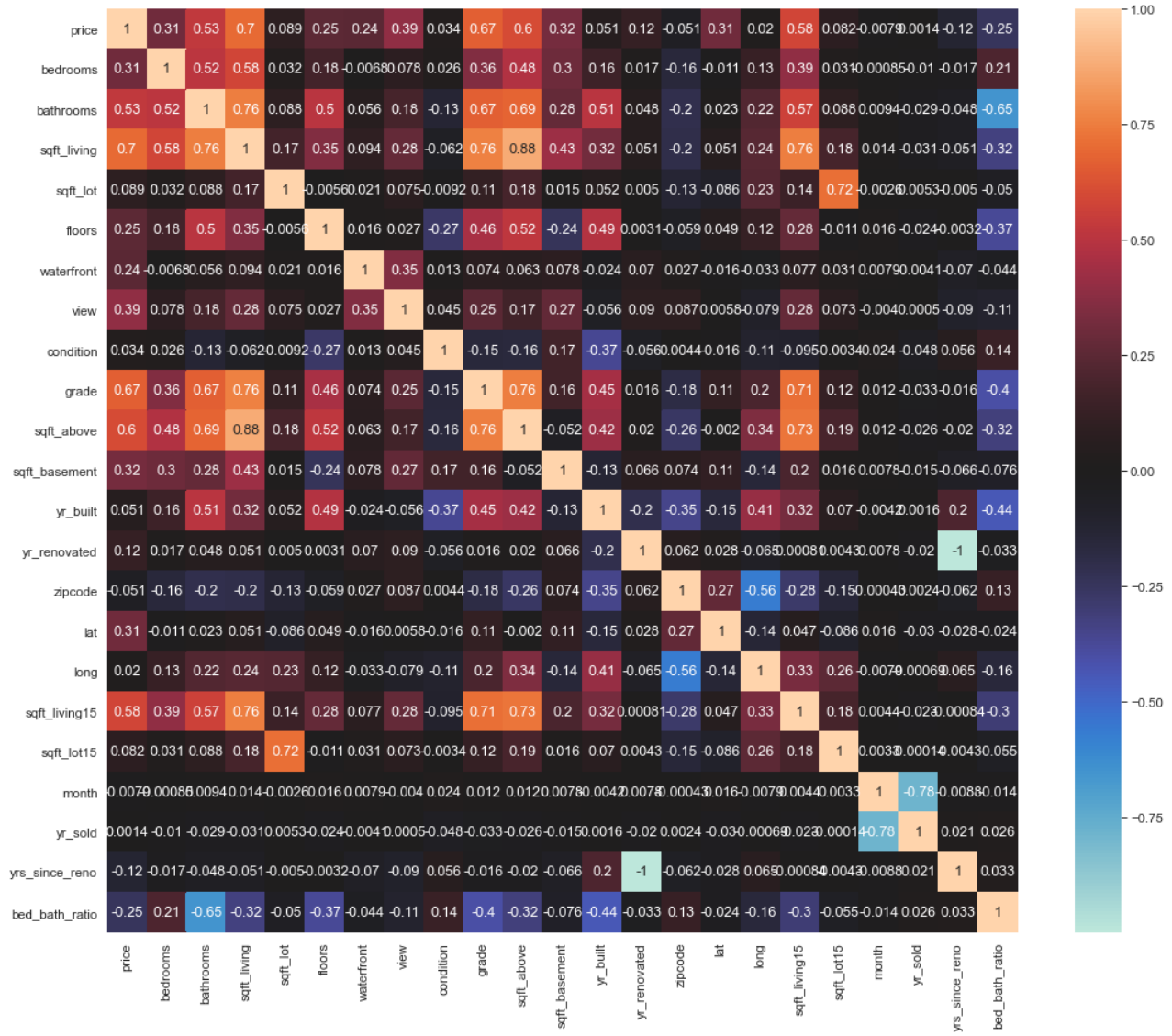
Out[73]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront
price	1.000000	0.309640	0.525215	0.701295	0.088789	0.254972	0.244254
bedrooms	0.309640	1.000000	0.515383	0.579069	0.032490	0.177734	-0.006762
bathrooms	0.525215	0.515383	1.000000	0.755522	0.087779	0.501803	0.056354
sqft_living	0.701295	0.579069	0.755522	1.000000	0.172586	0.352868	0.094080
sqft_lot	0.088789	0.032490	0.087779	0.172586	1.000000	-0.005561	0.021435
floors	0.254972	0.177734	0.501803	0.352868	-0.005561	1.000000	0.016340
waterfront	0.244254	-0.006762	0.056354	0.094080	0.021435	0.016340	1.000000
view	0.392787	0.078353	0.184949	0.280839	0.075360	0.027175	0.353796
condition	0.034219	0.025893	-0.129362	-0.061677	-0.009169	-0.266859	0.013017
grade	0.666835	0.357988	0.665587	0.762477	0.113656	0.458091	0.073902
sqft_above	0.604424	0.480242	0.686328	0.876533	0.183210	0.522751	0.062622
sqft_basement	0.321264	0.297093	0.278140	0.427998	0.014839	-0.242151	0.078154
yr_built	0.051012	0.155241	0.506252	0.316646	0.052312	0.488935	-0.024216
yr_renovated	0.118278	0.017470	0.047686	0.050851	0.004991	0.003141	0.069787
zipcode	-0.051169	-0.155061	-0.203884	-0.198995	-0.129422	-0.058532	0.026801
lat	0.306439	-0.011266	0.023143	0.050739	-0.085822	0.048900	-0.015502
long	0.019826	0.132439	0.223808	0.240212	0.230265	0.124812	-0.032630
sqft_living15	0.583792	0.394949	0.569453	0.756186	0.143805	0.279071	0.076963
sqft_lot15	0.082045	0.030570	0.087760	0.183837	0.717743	-0.011367	0.030857
month	-0.007920	-0.000853	0.009419	0.013848	-0.002576	0.015776	0.007850
yr_sold	0.001385	-0.010479	-0.028630	-0.030793	0.005305	-0.024178	-0.004116
yrs_since_reno	-0.118274	-0.017483	-0.047721	-0.050890	-0.004984	-0.003172	-0.069790
bed_bath_ratio	-0.248366	0.208307	-0.653467	-0.323680	-0.050038	-0.374296	-0.043597

23 rows × 23 columns

In [74]:
```
plt.figure(figsize=(18,15))
sns.heatmap(data = df.corr(), center= 0, annot= True)
```

Out[74]: <AxesSubplot:>



Correlation with absolute value of 0.7-0.8 is considered high.

Sqft_living has the highest correlation with price (0.7), followed by grade (0.6)

Correlations among features (0.75 cutoff value):

- sqft_above and sqft_living (0.88)
- sqft_living and grade (0.76) --> is sqft_living factor into the grade of a home?
- sqft_living15 and sqft_living (0.76) --> space in comps has high-ish correlation with space of home in general
- sqft_living and bathrooms (0.76)
- yr_renovated and yrs_since_reno (-1)
- month and yr_sold (-0.78)

Let's drop the most highly correlated/less useful for now, but will play with dropping yr_renovated v. yrs_since_reno later.

In [75]:
```
df = df.drop(['sqft_above', 'yr_sold'], axis=1)
```

In [76]:
```
df.to_csv('data/data_cleaned.csv', index=False)
```

In [77]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21420 entries, 0 to 21596
Data columns (total 22 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   date            21420 non-null  object 
 1   price           21420 non-null  float64
 2   bedrooms        21420 non-null  int64  
 3   bathrooms       21420 non-null  float64
 4   sqft_living     21420 non-null  int64  
 5   sqft_lot        21420 non-null  int64  
 6   floors          21420 non-null  float64
 7   waterfront      21420 non-null  float64
 8   view            21420 non-null  float64
 9   condition       21420 non-null  int64  
 10  grade           21420 non-null  int64  
 11  sqft_basement   21420 non-null  float64
 12  yr_built        21420 non-null  int64  
 13  yr_renovated    21420 non-null  int64  
 14  zipcode         21420 non-null  int64  
 15  lat             21420 non-null  float64
 16  long            21420 non-null  float64
 17  sqft_living15   21420 non-null  int64  
 18  sqft_lot15      21420 non-null  int64  
 19  month           21420 non-null  int64  
 20  yrs_since_reno  21420 non-null  int64  
 21  bed_bath_ratio  21420 non-null  float64
dtypes: float64(9), int64(12), object(1)
memory usage: 3.8+ MB
```

In [ ]:
```

```