

# Data\_Cleaning

```
In [12]: import pandas as pd  
import numpy as np
```

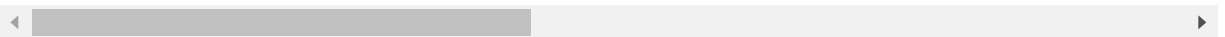
```
In [13]: data = pd.read_csv('data/data.csv')
```

```
In [14]: data.sort_values('id')
```

Out[14]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	wa
<b>2495</b>	1000102	4/22/2015	300000.0	6	3.00	2400	9373	2.0	
<b>2494</b>	1000102	9/16/2014	280000.0	6	3.00	2400	9373	2.0	
<b>6729</b>	1200019	5/8/2014	647500.0	4	1.75	2060	26036	1.0	
<b>8404</b>	1200021	8/11/2014	400000.0	3	1.00	1460	43000	1.0	
<b>8800</b>	2800031	4/1/2015	235000.0	3	1.00	1430	7599	1.5	
...	...	...	...	...	...	...	...	...	
<b>16723</b>	9842300095	7/25/2014	365000.0	5	2.00	1600	4168	1.5	
<b>3257</b>	9842300485	3/11/2015	380000.0	2	1.00	1040	7372	1.0	
<b>7614</b>	9842300540	6/24/2014	339000.0	3	1.00	1100	4128	1.0	
<b>20963</b>	9895000040	7/3/2014	399900.0	2	1.75	1410	1005	1.5	
<b>15937</b>	9900000190	10/30/2014	268950.0	3	1.00	1320	8100	1.0	

21597 rows × 21 columns



**id** -Does not contain any information about the house and not needed for linking with any other tables. Will drop.

```
In [15]: data.drop('id', axis=1, inplace=True)
```

```
In [16]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 20 columns):
date                21597 non-null object
price               21597 non-null float64
bedrooms            21597 non-null int64
bathrooms           21597 non-null float64
sqft_living          21597 non-null int64
sqft_lot             21597 non-null int64
floors               21597 non-null float64
waterfront           19221 non-null float64
view                 21534 non-null float64
condition            21597 non-null int64
grade                21597 non-null int64
sqft_above           21597 non-null int64
sqft_basement        21597 non-null object
yr_built             21597 non-null int64
yr_renovated         17755 non-null float64
zipcode              21597 non-null int64
lat                  21597 non-null float64
long                 21597 non-null float64
sqft_living15        21597 non-null int64
sqft_lot15           21597 non-null int64
dtypes: float64(8), int64(10), object(2)
memory usage: 3.3+ MB
```

## Null Values

```
In [17]: data.waterfront.value_counts()
```

```
Out[17]: 0.0    19075
         1.0     146
         Name: waterfront, dtype: int64
```

**waterfront** -I reckon they'd've mentioned if a property were on the waterfront. Will replace null values with 0.0

```
In [18]: data.waterfront.fillna(0.0, inplace=True)
         data.waterfront.value_counts()
```

```
Out[18]: 0.0    21451
         1.0     146
         Name: waterfront, dtype: int64
```

```
In [19]: data.view.value_counts()
```

```
Out[19]: 0.0    19422
         2.0     957
         3.0     508
         1.0     330
         4.0     317
         Name: view, dtype: int64
```

**view** -similarly I'll replace null with 0.0

```
In [20]: data.view.fillna(0.0, inplace=True)
         data.view.value_counts()
```

```
Out[20]: 0.0    19485
         2.0     957
         3.0     508
         1.0     330
         4.0     317
         Name: view, dtype: int64
```

```
In [21]: data.yr_renovated.value_counts()
```

```
Out[21]: 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
```

**yr\_renovated** - will replace null and 0.0 values with yr\_built. Tempted to drop the column as only 744 entries will be different from yr\_built, but it could have the better relationship with price.

```
In [22]: data.yr_renovated.fillna(data.yr_built, inplace=True)
         data.yr_renovated.replace(0, data.yr_built, inplace=True)
```

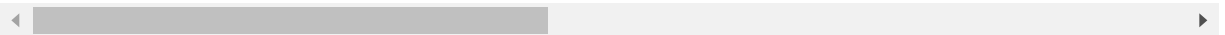
## Unusual / Placeholder Values

```
In [23]: data.date.value_counts()  
data.sort_values('date')
```

Out[23]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view
<b>12076</b>	1/10/2015	325000.0	4	2.50	2240	5105	2.0	0.0	0.0
<b>19359</b>	1/12/2015	330000.0	4	2.50	2240	7589	2.0	0.0	0.0
<b>2548</b>	1/12/2015	265000.0	4	1.50	1740	12728	1.0	0.0	0.0
<b>18398</b>	1/12/2015	359000.0	4	2.50	1820	11325	1.0	0.0	0.0
<b>11086</b>	1/12/2015	435000.0	4	2.50	2060	10125	2.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...
<b>19256</b>	9/9/2014	550000.0	3	1.50	1730	5750	1.0	0.0	0.0
<b>66</b>	9/9/2014	975000.0	4	2.50	2720	11049	2.0	0.0	0.0
<b>20288</b>	9/9/2014	520000.0	2	1.75	1340	1368	2.0	0.0	0.0
<b>12438</b>	9/9/2014	344950.0	3	1.75	1870	7500	1.0	0.0	0.0
<b>15702</b>	9/9/2014	295000.0	2	1.75	1050	6500	1.5	0.0	2.0

21597 rows × 20 columns



**date** -Change to days since first date

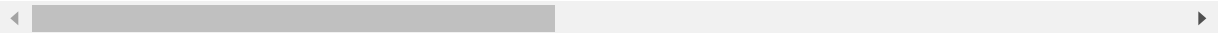
```
In [24]: data.date = pd.to_datetime(data.date)  
data.date = data.date.apply(lambda x: (x - data.date.min()).days)
```

```
In [25]: data.price.value_counts()
data.sort_values('price')
```

Out[25]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	co
<b>15279</b>	4	78000.0	2	1.00	780	16344	1.0	0.0	0.0	
<b>465</b>	21	80000.0	1	0.75	430	5050	1.0	0.0	0.0	
<b>16184</b>	326	81000.0	2	1.00	730	9975	1.0	0.0	0.0	
<b>8267</b>	187	82000.0	3	1.00	860	10426	1.0	0.0	0.0	
<b>2139</b>	6	82500.0	2	1.00	520	22334	1.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	
<b>1446</b>	346	5350000.0	5	5.00	8000	23985	2.0	0.0	4.0	
<b>4407</b>	94	5570000.0	5	5.75	9200	35069	2.0	0.0	0.0	
<b>9245</b>	140	6890000.0	6	7.75	9890	31374	2.0	0.0	4.0	
<b>3910</b>	40	7060000.0	5	4.50	10040	37325	2.0	1.0	2.0	
<b>7245</b>	164	7700000.0	6	8.00	12050	27600	2.5	0.0	3.0	

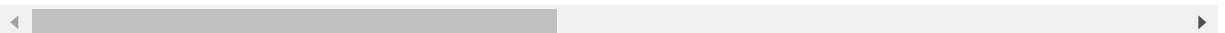
21597 rows × 20 columns



```
In [26]: data.bedrooms.value_counts()
data[data.bedrooms == 33]
```

Out[26]:

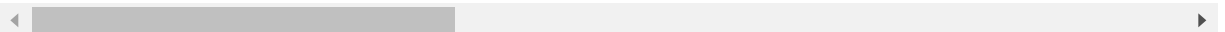
	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	con
<b>15856</b>	54	640000.0	33	1.75	1620	6000	1.0	0.0	0.0	



```
In [27]: data[data.bedrooms == 3].describe()
```

Out[27]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	flo
<b>count</b>	9824.000000	9.824000e+03	9824.0	9824.000000	9824.000000	9824.000000	9824.000000
<b>mean</b>	180.920704	4.662766e+05	3.0	1.968394	1805.837235	14414.790208	1.449
<b>std</b>	112.857681	2.626207e+05	0.0	0.629864	623.118916	35652.545246	0.561
<b>min</b>	0.000000	8.200000e+04	3.0	0.750000	490.000000	572.000000	1.000
<b>25%</b>	82.000000	2.954875e+05	3.0	1.500000	1370.000000	5000.000000	1.000
<b>50%</b>	168.000000	4.130000e+05	3.0	2.000000	1680.000000	7629.500000	1.000
<b>75%</b>	292.000000	5.600000e+05	3.0	2.500000	2110.000000	10364.000000	2.000
<b>max</b>	377.000000	3.800000e+06	3.0	4.500000	6400.000000	843309.000000	3.500



**bedrooms** -The 33 bedroom property is slightly smaller than the average 3 bedroom property (within a standard deviation). I suspect 33 was a typo and will change it to 3

```
In [28]: data.at[15856, 'bedrooms'] = 3
```

```
In [29]: data.bedrooms.value_counts()
```

```
Out[29]: 3      9825
         4      6882
         2      2760
         5      1601
         6       272
         1       196
         7        38
         8        13
         9         6
        10         3
        11         1
        Name: bedrooms, dtype: int64
```

```
In [30]: data.bathrooms.value_counts()
```

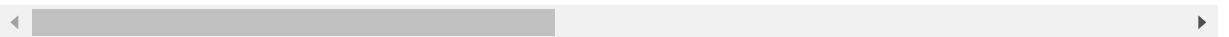
```
Out[30]: 2.50      5377
         1.00      3851
         1.75      3048
         2.25      2047
         2.00      1930
         1.50      1445
         2.75      1185
         3.00       753
         3.50       731
         3.25       589
         3.75       155
         4.00       136
         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
```

```
In [31]: data.sqft_living.value_counts()
data.sort_values('sqft_living')
```

Out[31]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	coi
<b>15367</b>	60	276000.0	1	0.75	370	1801	1.0	0.0	0.0	
<b>860</b>	49	245000.0	1	0.75	380	15000	1.0	0.0	0.0	
<b>21316</b>	374	245000.0	1	1.00	390	2000	1.0	0.0	0.0	
<b>8614</b>	273	325000.0	1	0.75	410	8636	1.0	0.0	0.0	
<b>11488</b>	333	229050.0	1	1.00	420	3298	1.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...
<b>8085</b>	46	4670000.0	5	6.75	9640	13068	1.0	1.0	4.0	
<b>9245</b>	140	6890000.0	6	7.75	9890	31374	2.0	0.0	4.0	
<b>3910</b>	40	7060000.0	5	4.50	10040	37325	2.0	1.0	2.0	
<b>7245</b>	164	7700000.0	6	8.00	12050	27600	2.5	0.0	3.0	
<b>12764</b>	3	2280000.0	7	8.00	13540	307752	3.0	0.0	4.0	

21597 rows × 20 columns

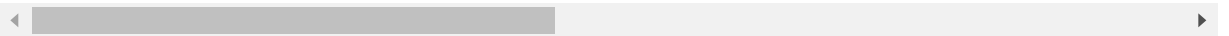


```
In [32]: data.sqft_lot.value_counts()
data.sort_values('sqft_living')
```

Out[32]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	coi
<b>15367</b>	60	276000.0	1	0.75	370	1801	1.0	0.0	0.0	
<b>860</b>	49	245000.0	1	0.75	380	15000	1.0	0.0	0.0	
<b>21316</b>	374	245000.0	1	1.00	390	2000	1.0	0.0	0.0	
<b>8614</b>	273	325000.0	1	0.75	410	8636	1.0	0.0	0.0	
<b>11488</b>	333	229050.0	1	1.00	420	3298	1.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...
<b>8085</b>	46	4670000.0	5	6.75	9640	13068	1.0	1.0	4.0	
<b>9245</b>	140	6890000.0	6	7.75	9890	31374	2.0	0.0	4.0	
<b>3910</b>	40	7060000.0	5	4.50	10040	37325	2.0	1.0	2.0	
<b>7245</b>	164	7700000.0	6	8.00	12050	27600	2.5	0.0	3.0	
<b>12764</b>	3	2280000.0	7	8.00	13540	307752	3.0	0.0	4.0	

21597 rows × 20 columns



```
In [33]: data.floors.value_counts()
```

```
Out[33]: 1.0    10673  
        2.0     8235  
        1.5     1910  
        3.0      611  
        2.5     161  
        3.5        7  
        Name: floors, dtype: int64
```

```
In [34]: data.waterfront.value_counts()
```

```
Out[34]: 0.0    21451  
        1.0     146  
        Name: waterfront, dtype: int64
```

```
In [35]: data.view.value_counts()
```

```
Out[35]: 0.0    19485  
        2.0     957  
        3.0     508  
        1.0     330  
        4.0     317  
        Name: view, dtype: int64
```

```
In [36]: data.condition.value_counts()
```

```
Out[36]: 3    14020  
        4     5677  
        5     1701  
        2      170  
        1       29  
        Name: condition, dtype: int64
```

```
In [37]: data.grade.value_counts()
```

```
Out[37]: 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
```

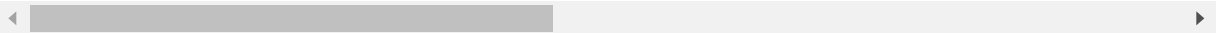


```
In [38]: data.sqft_above.value_counts()
data.sort_values('sqft_above')
```

Out[38]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	coi
<b>15367</b>	60	276000.0	1	0.75	370	1801	1.0	0.0	0.0	
<b>860</b>	49	245000.0	1	0.75	380	15000	1.0	0.0	0.0	
<b>21316</b>	374	245000.0	1	1.00	390	2000	1.0	0.0	0.0	
<b>8614</b>	273	325000.0	1	0.75	410	8636	1.0	0.0	0.0	
<b>14452</b>	41	280000.0	1	0.75	420	6720	1.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...
<b>13398</b>	273	2420000.0	5	4.75	7880	24250	2.0	0.0	2.0	
<b>18288</b>	61	3300000.0	5	6.25	8020	21738	2.0	0.0	0.0	
<b>7245</b>	164	7700000.0	6	8.00	12050	27600	2.5	0.0	3.0	
<b>9245</b>	140	6890000.0	6	7.75	9890	31374	2.0	0.0	4.0	
<b>12764</b>	3	2280000.0	7	8.00	13540	307752	3.0	0.0	4.0	

21597 rows × 20 columns

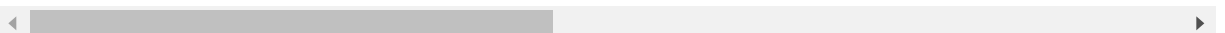


```
In [39]: data.sqft_basement.value_counts()
data.sort_values('sqft_basement')
```

Out[39]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	coi
<b>0</b>	164	221900.0	3	1.00	1180	5650	1.0	0.0	0.0	
<b>12644</b>	335	1270000.0	4	1.75	2040	5000	2.0	0.0	0.0	
<b>12645</b>	179	290000.0	3	1.75	1460	7980	1.0	0.0	0.0	
<b>12646</b>	200	158000.0	3	1.50	990	8925	1.0	0.0	0.0	
<b>12648</b>	7	583000.0	4	2.50	2660	4000	2.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...
<b>2041</b>	335	219950.0	3	1.00	1200	7727	1.0	0.0	0.0	
<b>10910</b>	279	215000.0	3	1.00	1060	7900	1.0	0.0	0.0	
<b>12363</b>	21	549900.0	4	3.00	2830	213879	2.0	0.0	0.0	
<b>6410</b>	185	238950.0	2	1.00	810	4838	1.0	0.0	0.0	
<b>19234</b>	194	129888.0	2	1.00	710	9900	1.0	0.0	0.0	

21597 rows × 20 columns



**sqft\_basement** -replace question marks with zero and change column to type: int

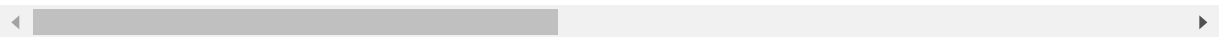
```
In [40]: data.sqft_basement.replace('?', 0, inplace=True)
data.sqft_basement = data.sqft_basement.astype('float64', copy=False).astype(
'int64', copy=False)
```

```
In [41]: data.yr_built.value_counts()
data.sort_values('yr_built')
```

Out[41]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	con
<b>14069</b>	87	255500.0	4	1.00	1370	41194	1.5	0.0	2.0	
<b>14783</b>	174	560000.0	4	1.00	1360	5814	1.5	0.0	0.0	
<b>10973</b>	348	730000.0	3	1.75	1650	5000	1.5	0.0	0.0	
<b>115</b>	203	740500.0	3	3.50	4380	6350	2.0	0.0	0.0	
<b>4693</b>	39	558000.0	4	2.00	2180	3870	1.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	
<b>20235</b>	368	771005.0	5	4.50	4000	6713	2.0	0.0	0.0	
<b>7519</b>	243	614285.0	5	2.75	2730	6401	2.0	0.0	0.0	
<b>14911</b>	354	671000.0	4	2.75	1890	1475	2.0	0.0	0.0	
<b>4150</b>	355	631000.0	3	2.25	1670	1396	2.0	0.0	0.0	
<b>19789</b>	91	455000.0	3	1.75	1320	1014	3.0	0.0	0.0	

21597 rows × 20 columns

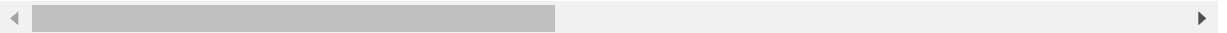


```
In [42]: data.yr_renovated.value_counts()
data.sort_values('yr_renovated')
```

Out[42]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	coi
<b>14069</b>	87	255500.0	4	1.00	1370	41194	1.5	0.0	2.0	
<b>4393</b>	83	551000.0	3	1.00	940	1948	1.0	0.0	0.0	
<b>19370</b>	357	240000.0	3	2.00	1553	6550	1.0	0.0	0.0	
<b>14783</b>	174	560000.0	4	1.00	1360	5814	1.5	0.0	0.0	
<b>4951</b>	186	611000.0	2	1.00	1270	5100	1.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	
<b>21353</b>	334	455950.0	4	2.50	2720	5771	2.0	0.0	0.0	
<b>20917</b>	356	1550000.0	3	3.25	3530	4920	2.0	0.0	0.0	
<b>8032</b>	53	455000.0	2	1.50	1200	1259	2.0	0.0	0.0	
<b>8683</b>	369	1490000.0	6	2.75	4430	6440	2.0	0.0	3.0	
<b>21300</b>	312	1700000.0	4	3.50	3950	6240	2.0	0.0	0.0	

21597 rows × 20 columns

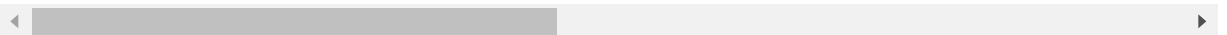


```
In [43]: data.zipcode.value_counts()
data.sort_values('zipcode')
```

Out[43]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	con
<b>8508</b>	355	265000.0	3	1.50	1780	10196	1.0	0.0	0.0	
<b>17199</b>	4	197000.0	3	1.75	1690	7735	1.0	0.0	0.0	
<b>2338</b>	307	230000.0	4	2.00	1440	10800	1.0	0.0	0.0	
<b>8728</b>	150	199129.0	3	1.00	860	33664	1.0	0.0	0.0	
<b>19729</b>	318	355000.0	4	2.75	2050	4000	2.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	
<b>10822</b>	372	812000.0	4	2.00	2380	6122	1.0	0.0	2.0	
<b>10897</b>	124	535000.0	2	2.00	1510	5133	1.5	0.0	0.0	
<b>15491</b>	347	700000.0	4	1.75	1870	6000	1.0	0.0	0.0	
<b>10957</b>	62	554729.0	4	2.50	2020	4350	2.0	0.0	0.0	
<b>3110</b>	90	850000.0	3	1.75	2450	8603	1.0	0.0	0.0	

21597 rows × 20 columns

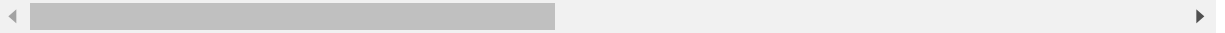


```
In [44]: data.lat.value_counts()  
data.sort_values('lat')
```

Out[44]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	con
<b>3292</b>	53	380000.0	3	2.25	1860	15559	2.0	0.0	0.0	
<b>15585</b>	31	687000.0	4	3.25	4400	186846	2.0	0.0	0.0	
<b>12993</b>	339	750000.0	3	2.50	2350	715690	1.5	0.0	0.0	
<b>12656</b>	48	335000.0	4	2.00	2030	103672	1.0	0.0	0.0	
<b>7712</b>	311	245000.0	3	1.75	1670	24650	1.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	
<b>6808</b>	312	285000.0	4	2.00	2120	6865	1.0	0.0	0.0	
<b>6049</b>	311	270000.0	3	1.00	1480	7374	1.0	0.0	0.0	
<b>306</b>	336	550000.0	4	2.75	1800	7750	1.0	0.0	0.0	
<b>17450</b>	165	389950.0	3	1.75	1580	9049	1.0	0.0	0.0	
<b>15752</b>	345	407500.0	4	2.50	1900	9075	2.0	0.0	0.0	

21597 rows × 20 columns

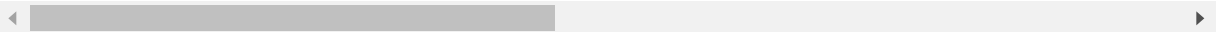


```
In [45]: data.long.value_counts()  
data.sort_values('long')
```

Out[45]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	con
<b>13873</b>	215	575000.0	3	2.00	2690	435600	2.0	0.0	0.0	
<b>9289</b>	123	565000.0	3	2.50	2030	217805	1.0	0.0	0.0	
<b>2962</b>	201	999000.0	3	2.75	2830	505166	1.0	1.0	3.0	
<b>1166</b>	160	290000.0	2	0.75	440	8313	1.0	1.0	3.0	
<b>6096</b>	138	340000.0	2	0.75	1060	48292	1.0	1.0	2.0	
...	...	...	...	...	...	...	...	...	...	
<b>13236</b>	350	375000.0	3	1.75	2140	13598	1.5	0.0	0.0	
<b>10886</b>	173	241000.0	2	1.75	1070	9750	1.5	0.0	0.0	
<b>13059</b>	161	155000.0	2	1.00	1010	43056	1.5	0.0	0.0	
<b>4199</b>	68	150000.0	3	0.75	490	38500	1.5	0.0	0.0	
<b>2925</b>	363	167000.0	1	1.00	780	10235	1.5	0.0	0.0	

21597 rows × 20 columns

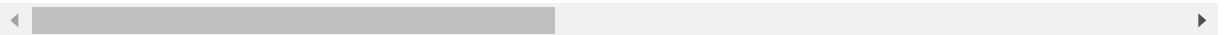


```
In [46]: data.sqft_living15.value_counts()
data.sort_values('sqft_living15')
```

Out[46]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	coi
<b>13428</b>	375	536000.0	3	2.75	2290	34548	2.0	0.0	3.0	
<b>17272</b>	45	378000.0	5	2.50	2760	8015	1.0	0.0	0.0	
<b>12094</b>	53	257500.0	2	2.00	1180	9265	1.0	0.0	0.0	
<b>17873</b>	200	255000.0	2	1.00	620	4760	1.0	0.0	0.0	
<b>1918</b>	292	265000.0	2	1.00	620	4760	1.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...
<b>16416</b>	276	1750000.0	6	4.25	5860	13928	2.0	0.0	3.0	
<b>5446</b>	161	1780000.0	4	3.25	4890	13402	2.0	0.0	0.0	
<b>20814</b>	153	1750000.0	5	3.25	5790	12739	2.0	0.0	3.0	
<b>10362</b>	224	2980000.0	5	5.50	7400	18898	2.0	0.0	3.0	
<b>19842</b>	265	2700000.0	4	4.00	7850	89651	2.0	0.0	0.0	

21597 rows × 20 columns

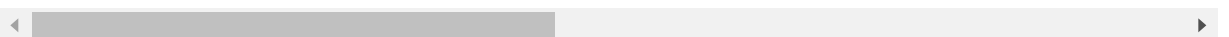


```
In [47]: data.sqft_lot15.value_counts()
data.sort_values('sqft_lot15')
```

Out[47]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	coi
<b>19653</b>	152	299900.0	3	2.50	1210	2046	2.0	0.0	0.0	
<b>20733</b>	86	286308.0	2	1.50	1220	1036	3.0	0.0	0.0	
<b>513</b>	187	290000.0	4	1.00	1330	8184	1.5	0.0	0.0	
<b>20999</b>	330	338500.0	2	2.25	1150	711	2.0	0.0	0.0	
<b>20891</b>	234	325000.0	2	2.25	1150	711	2.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...
<b>3797</b>	116	637000.0	4	3.50	3080	118918	2.0	0.0	0.0	
<b>8655</b>	170	549950.0	3	1.75	2930	266587	2.0	0.0	0.0	
<b>13451</b>	189	790000.0	3	2.50	2640	432036	1.5	0.0	3.0	
<b>20436</b>	348	1600000.0	4	5.50	6530	871200	2.0	0.0	2.0	
<b>9705</b>	250	937500.0	4	4.00	5545	871200	2.0	0.0	0.0	

21597 rows × 20 columns



## Removing Outliers

```
In [48]: data.describe()
```

Out[48]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot
count	21597.000000	2.159700e+04	21597.000000	21597.000000	21597.000000	2.159700e+04
mean	180.180997	5.402966e+05	3.371811	2.115826	2080.321850	1.509941e+04
std	113.059987	3.673681e+05	0.904096	0.768984	918.106125	4.141264e+04
min	0.000000	7.800000e+04	1.000000	0.500000	370.000000	5.200000e+02
25%	81.000000	3.220000e+05	3.000000	1.750000	1430.000000	5.040000e+03
50%	167.000000	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03
75%	291.000000	6.450000e+05	4.000000	2.500000	2550.000000	1.068500e+04
max	390.000000	7.700000e+06	11.000000	8.000000	13540.000000	1.651359e+06

Using mean +/- 3(stddev) for each column except for lat, long, zipcode, date. Don't want to remove outlying rows before finding each columns outliers; otherwise it may affect the calculation. So I'll find the indices of the rows I want to remove for each column, then remove them.

```
In [73]: for col in data.drop(['lat', 'long', 'zipcode'], axis=1):
        try:
            ind = []
            mean = data[col].mean()
            std = data[col].std()

            ind.append(data[col][(data[col] >= mean + 3*std) |
                                (data[col] <= mean - 3*std)].index)

        except:
            print(col, '-----except')
            continue
```

```
In [74]: d_in = data.drop(set(list(ind[0])))
```

In [75]: *#Checking min and max values for the method I used compared to removing each s  
eparately and singly*

```
for col in data.columns:  
    x = data[col][(data[col] < mean + 3*std) | (data[col] > mean - 3*std)]  
    print(col)  
    print('d_in min:',d_in[col].min(),'d_in max:', d_in[col].max())  
    print('data min:',x.min(), 'data max:',x.max())
```

date  
d\_in min: 0 d\_in max: 390  
data min: 0 data max: 390  
price  
d\_in min: 78000.0 d\_in max: 7700000.0  
data min: 78000.0 data max: 7700000.0  
bedrooms  
d\_in min: 1 d\_in max: 11  
data min: 1 data max: 11  
sqft\_living  
d\_in min: 370 d\_in max: 13540  
data min: 370 data max: 13540  
sqft\_lot  
d\_in min: 520 d\_in max: 1651359  
data min: 520 data max: 1651359  
waterfront  
d\_in min: 0 d\_in max: 1  
data min: 0 data max: 1  
view  
d\_in min: 0 d\_in max: 4  
data min: 0 data max: 4  
condition  
d\_in min: 1 d\_in max: 5  
data min: 1 data max: 5  
grade  
d\_in min: 3 d\_in max: 13  
data min: 3 data max: 13  
sqft\_above  
d\_in min: 370 d\_in max: 9410  
data min: 370 data max: 9410  
sqft\_basement  
d\_in min: 0 d\_in max: 4820  
data min: 0 data max: 4820  
yr\_built  
d\_in min: 1900 d\_in max: 2015  
data min: 1900 data max: 2015  
yr\_renovated  
d\_in min: 1900 d\_in max: 2015  
data min: 1900 data max: 2015  
zipcode  
d\_in min: 98001 d\_in max: 98199  
data min: 98001 data max: 98199  
lat  
d\_in min: 47.1559 d\_in max: 47.7776  
data min: 47.1559 data max: 47.7776  
long  
d\_in min: -122.51899999999999 d\_in max: -121.315  
data min: -122.51899999999999 data max: -121.315  
sqft\_living15  
d\_in min: 399 d\_in max: 6210  
data min: 399 data max: 6210  
sqft\_lot15  
d\_in min: 651 d\_in max: 871200  
data min: 651 data max: 871200  
bathroomsx4  
d\_in min: 2 d\_in max: 32  
data min: 2 data max: 32



```
floorsx2
d_in min: 2 d_in max: 6
data min: 2 data max: 7
```

```
In [72]: data[data.floorsx2 ==7]
```

Out[72]:

	date	price	bedrooms	sqft_living	sqft_lot	waterfront	view	condition	grade	sqft_
<b>10066</b>	91	435000.0	3	1440	1350	0	2	3	8	
<b>11582</b>	273	544000.0	3	1760	1755	0	0	3	8	
<b>14871</b>	335	525000.0	3	1730	1074	0	0	3	8	
<b>15410</b>	349	479000.0	2	1730	1037	0	0	3	8	
<b>18462</b>	171	3300000.0	8	7710	11750	0	0	5	12	
<b>20292</b>	139	525000.0	2	1310	1268	0	0	3	8	
<b>20756</b>	54	563500.0	3	1400	1312	0	0	3	8	

Difference in floorsx2 max, 6-7: not many 7s, probably removed as outliers for other features

```
In [53]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 20 columns):
date                21597 non-null int64
price               21597 non-null float64
bedrooms           21597 non-null int64
bathrooms          21597 non-null float64
sqft_living         21597 non-null int64
sqft_lot            21597 non-null int64
floors             21597 non-null float64
waterfront         21597 non-null float64
view               21597 non-null float64
condition           21597 non-null int64
grade              21597 non-null int64
sqft_above          21597 non-null int64
sqft_basement       21597 non-null int64
yr_built            21597 non-null int64
yr_renovated        21597 non-null float64
zipcode             21597 non-null int64
lat                 21597 non-null float64
long                21597 non-null float64
sqft_living15       21597 non-null int64
sqft_lot15          21597 non-null int64
dtypes: float64(8), int64(12)
memory usage: 3.3 MB
```

```
In [ ]:
```

## Extra cleaning for dummies

Multiplying to turn float feature values into ints, so that dummied column names don't have periods

```
In [54]: data['bathroomsx4'] = data.bathrooms*4  
data.bathroomsx4 = data.bathroomsx4.astype('int32')  
data.drop('bathrooms', axis=1, inplace=True)
```

```
In [55]: set(data.floors)
```

```
Out[55]: {1.0, 1.5, 2.0, 2.5, 3.0, 3.5}
```

```
In [56]: data['floorsx2'] = data.floors*2  
data.floorsx2 = data.floorsx2.astype('int32')  
data.drop('floors', axis=1, inplace=True)
```

```
In [57]: data = data.astype({'view': 'int32', 'waterfront': 'int32', 'yr_renovated': 'int32'})
```

```
In [58]: set(data.view)
```

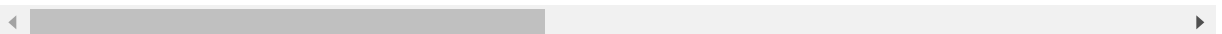
```
Out[58]: {0, 1, 2, 3, 4}
```

```
In [59]: data
```

```
Out[59]:
```

	date	price	bedrooms	sqft_living	sqft_lot	waterfront	view	condition	grade	sqft_i
0	164	221900.0	3	1180	5650	0	0	3	7	
1	221	538000.0	3	2570	7242	0	0	3	7	
2	299	180000.0	2	770	10000	0	0	3	6	
3	221	604000.0	4	1960	5000	0	0	5	7	
4	292	510000.0	3	1680	8080	0	0	3	8	
...	...	...	...	...	...	...	...	...	...	
21592	19	360000.0	3	1530	1131	0	0	3	8	
21593	297	400000.0	4	2310	5813	0	0	3	8	
21594	52	402101.0	2	1020	1350	0	0	3	7	
21595	259	400000.0	3	1600	2388	0	0	3	8	
21596	166	325000.0	2	1020	1076	0	0	3	7	

21597 rows × 20 columns



In [61]: data.describe()

Out[61]:

	date	price	bedrooms	sqft_living	sqft_lot	waterfront	
count	21597.000000	2.159700e+04	21597.000000	21597.000000	2.159700e+04	21597.000000	215
mean	180.180997	5.402966e+05	3.371811	2080.321850	1.509941e+04	0.006760	
std	113.059987	3.673681e+05	0.904096	918.106125	4.141264e+04	0.081944	
min	0.000000	7.800000e+04	1.000000	370.000000	5.200000e+02	0.000000	
25%	81.000000	3.220000e+05	3.000000	1430.000000	5.040000e+03	0.000000	
50%	167.000000	4.500000e+05	3.000000	1910.000000	7.618000e+03	0.000000	
75%	291.000000	6.450000e+05	4.000000	2550.000000	1.068500e+04	0.000000	
max	390.000000	7.700000e+06	11.000000	13540.000000	1.651359e+06	1.000000	

