In [2]:
```
import turicreate
```

In [3]:
```
home_sframe = turicreate.SFrame('home_data.sframe/')
```

In [4]:
```
home_sframe
```

Out[4]:

id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot
7129300520	2014-10-13 00:00:00+00:00	221900.0	3.0	1.0	1180.0	5650.0
6414100192	2014-12-09 00:00:00+00:00	538000.0	3.0	2.25	2570.0	7242.0
5631500400	2015-02-25 00:00:00+00:00	180000.0	2.0	1.0	770.0	10000.0
2487200875	2014-12-09 00:00:00+00:00	604000.0	4.0	3.0	1960.0	5000.0
1954400510	2015-02-18 00:00:00+00:00	510000.0	3.0	2.0	1680.0	8080.0
7237550310	2014-05-12 00:00:00+00:00	1225000.0	4.0	4.5	5420.0	101930.0
1321400060	2014-06-27 00:00:00+00:00	257500.0	3.0	2.25	1715.0	6819.0
2008000270	2015-01-15 00:00:00+00:00	291850.0	3.0	1.5	1060.0	9711.0
2414600126	2015-04-15 00:00:00+00:00	229500.0	3.0	1.0	1780.0	7470.0
3793500160	2015-03-12 00:00:00+00:00	323000.0	3.0	2.5	1890.0	6560.0

view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcod
0	3	7.0	1180.0	0.0	1955.0	0.0	9817
0	3	7.0	2170.0	400.0	1951.0	1991.0	9812
0	3	6.0	770.0	0.0	1933.0	0.0	9802
0	5	7.0	1050.0	910.0	1965.0	0.0	9813
0	3	8.0	1680.0	0.0	1987.0	0.0	9807
0	3	11.0	3890.0	1530.0	2001.0	0.0	9805
0	3	7.0	1715.0	0.0	1995.0	0.0	9800
0	3	7.0	1060.0	0.0	1963.0	0.0	9819
0	3	7.0	1050.0	730.0	1960.0	0.0	9814
0	3	7.0	1890.0	0.0	2003.0	0.0	9803

long	sqft_living15	sqft_lot15
-122.25677536	1340.0	5650.0

-122.3188624	1690.0	7639.0
-122.23319601	2720.0	8062.0
-122.39318505	1360.0	5000.0
-122.04490059	1800.0	7503.0
-122.00528655	4760.0	101930.0
-122.32704857	2238.0	6819.0

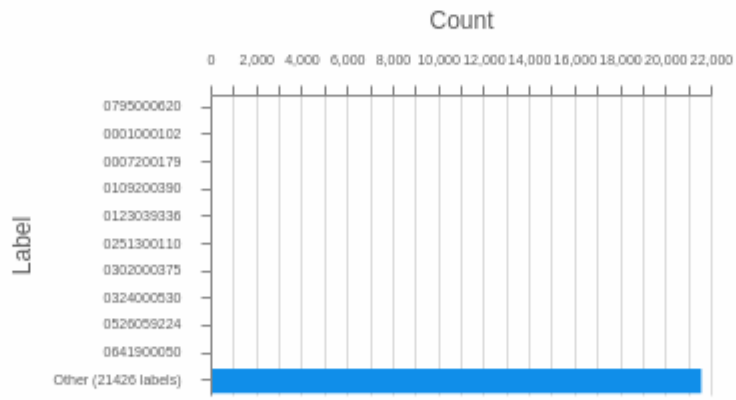
In [5]:
```
home_sframe['price'].max()
```
Out[5]: 7700000.0

In [6]:
```
home_sframe['price'].mean()
```
Out[6]: 540088.1419053345

In [7]:
```
home_sframe['price'].max()
```
Out[7]: 7700000.0

In [8]:
```
home_sframe.show()
```
```
Materializing SFrame
Warning: Skipping column 'date'. Unable to show columns of type 'datet
ime'; only [int, float, str] can be shown.
Further warnings of unsupported type will be suppressed.
```

id



Num. Rows:

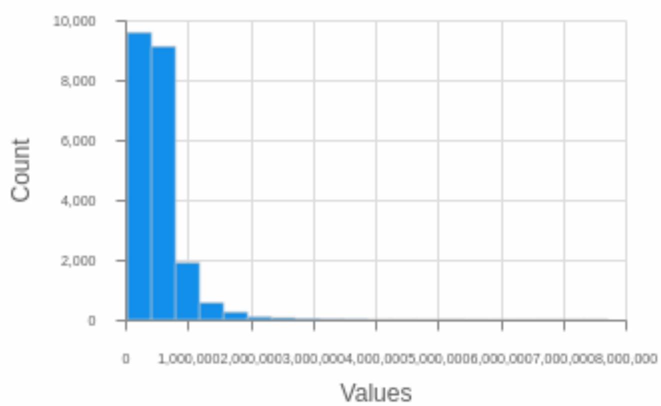
Num. Unique:

Missing:

Frequent Items

0795000620
0001000102
0007200179
0109200390
0123039336

price



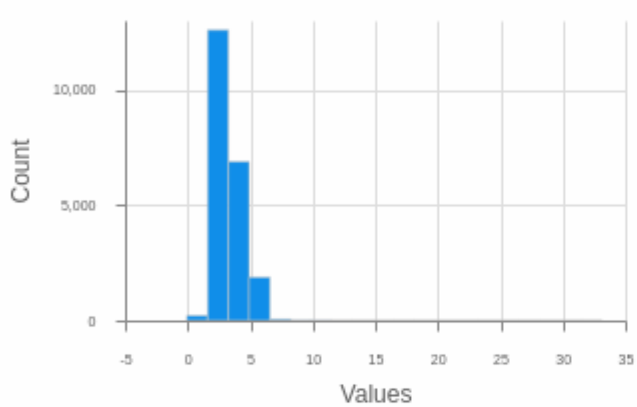
Num. Rows:

Num. Unique:

Missing:

Mean:
Min:
Max:
Median:
St. Dev:

bedrooms



Num. Rows:

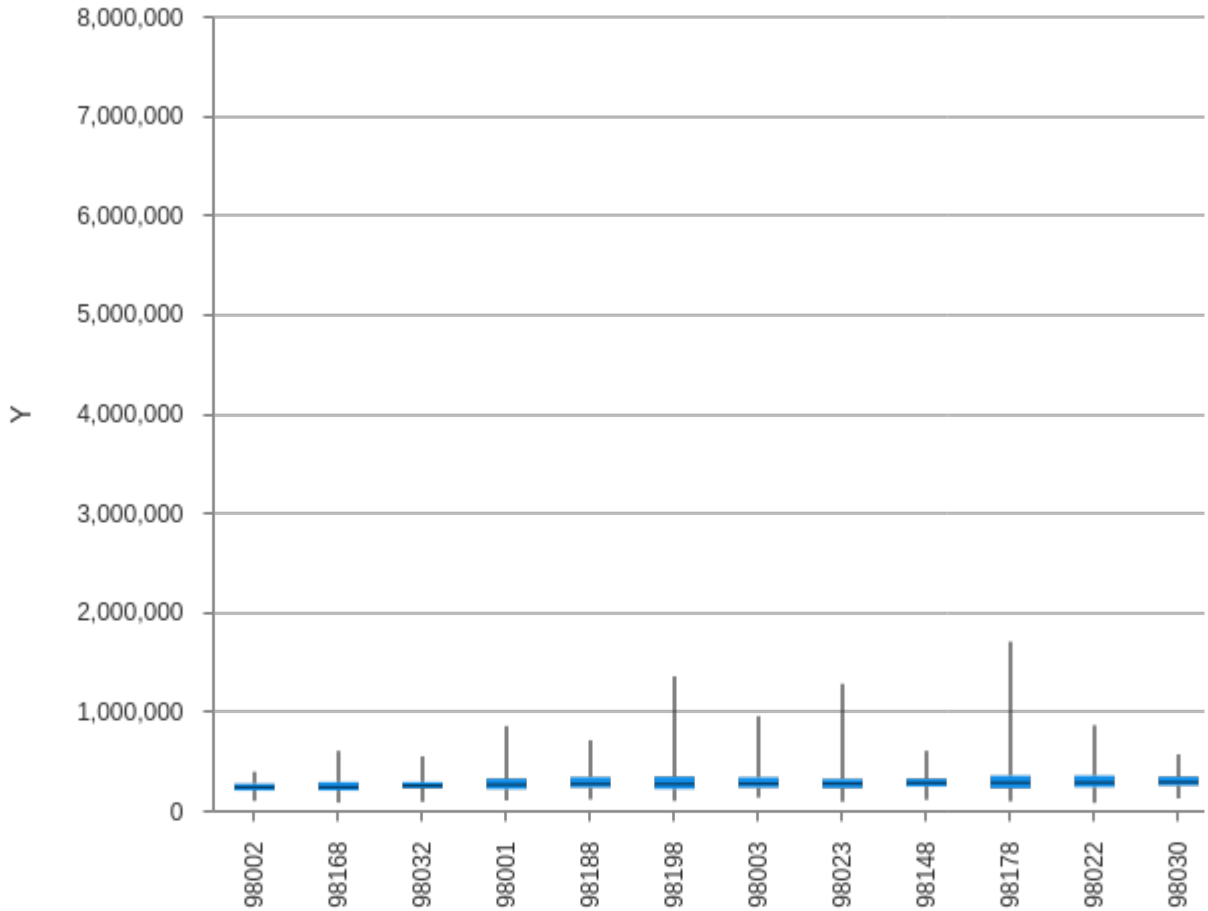
Num. Unique:

Missing:

Mean:
Min:
Max:
Median:
St. Dev:

In [9]:
```
turicreate.show(home_sframe['price'],home_sframe['zipcode'])
```

```
Materializing X axis SArray
Materializing Y axis SArray
```



In [10]:
```
train_set,test_set=home_sframe.random_split(0.8,seed=0)
```

In [11]:
```
simple_features = ['sqft_living']
```

In [12]:
```
simple_features_model = turicreate.linear_regression.create(train_set,target=
```

```
PROGRESS: Creating a validation set from 5 percent of training data. This may
take a while.
          You can set ``validation_set=None`` to disable validation tracking.
```

```
Linear regression:
--------------------------------------------------------
Number of examples          : 16514
Number of features          : 1
Number of unpacked features : 1
Number of coefficients    : 2
Starting Newton Method
--------------------------------------------------------
+-----------+----------+--------------+--------------------+----------
-----------+---------------------------------+----------------------
-----------+
| Iteration | Passes   | Elapsed Time | Training Max Error | Validatio
n Max Error | Training Root-Mean-Square Error | Validation Root-Mean-S
quare Error |
+-----------+----------+--------------+--------------------+----------
-----------+---------------------------------+----------------------
-----------+
| 1         | 2        | 1.004356     | 4345716.638199     | 2190936.3
94326       | 262638.791561                   | 268668.689943
|
+-----------+----------+--------------+--------------------+----------
-----------+---------------------------------+----------------------
-----------+
SUCCESS: Optimal solution found.
```

In [ ]:

In [ ]:

In [13]:
```
simple_features = ['sqft_living']
```

In [14]:
```
simple_features_model.evaluate
```

Out[14]:
```
<bound method LinearRegression.evaluate of Class                          : L
inearRegression

Schema
------
Number of coefficients         : 2
Number of examples             : 16514
Number of feature columns      : 1
Number of unpacked features    : 1
```

```
Hyperparameters
---------------
L1 penalty                     : 0.0
L2 penalty                     : 0.01

Training Summary
----------------
Solver                         : newton
Solver iterations              : 1
Solver status                  : SUCCESS: Optimal solution found.
Training time (sec)            : 1.009

Settings
--------
Residual sum of squares        : 1139121432628916.0
Training RMSE                  : 262638.7916

Highest Positive Coefficients
-----------------------------
sqft_living                    : 282.3415

Lowest Negative Coefficients
----------------------------
(intercept)                    : -47931.7246
>
```

In [15]:
```
simple_features_model.coefficients
```

Out[15]:

name	index	value	stderr
(intercept)	None	-47931.724557466805	5038.004933345714
sqft_living	None	282.3415009426225	2.211666366210711

[2 rows x 4 columns]

In [16]:
```
advanced_features = [
'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors', 'zipcode',
'condition', # condition of house
'grade', # measure of quality of construction
'waterfront', # waterfront property
'view', # type of view
'sqft_above', # square feet above ground
'sqft_basement', # square feet in basement
'yr_built', # the year built
'yr_renovated', # the year renovated
'lat', 'long', # the lat-long of the parcel
'sqft_living15', # average sq.ft. of 15 nearest neighbors
'sqft_lot15', # average lot size of 15 nearest neighbors
]
```

In [17]:
```
advanced_features_model = turicreate.linear_regression.create(train_set,targe
```

```
PROGRESS: Creating a validation set from 5 percent of training data. This may
take a while.
          You can set ``validation_set=None`` to disable validation tracking.

Linear regression:
```

```
--------------------------------------------------------
Number of examples          : 16514
Number of features          : 18
Number of unpacked features : 18
Number of coefficients    : 87
Starting Newton Method
--------------------------------------------------------
+-----------+----------+--------------+--------------------+----------
------------+--------------------------------+----------------------
-----------+
| Iteration | Passes   | Elapsed Time | Training Max Error | Validatio
n Max Error | Training Root-Mean-Square Error | Validation Root-Mean-S
quare Error |
+-----------+----------+--------------+--------------------+----------
------------+--------------------------------+----------------------
-----------+
| 1         | 2        | 0.052140     | 4318201.703010     | 1147156.7
64998       | 162835.970479                  | 154516.133169
|
+-----------+----------+--------------+--------------------+----------
------------+--------------------------------+----------------------
-----------+
SUCCESS: Optimal solution found.
```

In [18]:
```
advanced_features_model.coefficients
```

Out[18]:

name	index	value	stderr
(intercept)	None	-2062717.885990942	7109721.051740051
bedrooms	None	-31139.94433933164	1860.2563371985857
bathrooms	None	23162.014001569172	3062.6673112809967
sqft_living	None	93.50979895832255	4963017.229513706
sqft_lot	None	0.2669786738631242	0.04462691018101468
floors	None	-46884.124919075075	3680.310756077988
zipcode	98125	151248.39036648287	21782.819700989985
zipcode	98028	63702.02450579184	24692.670128817004
zipcode	98136	202766.65695139786	17827.989720683294
zipcode	98074	123141.24333168165	20398.39661379382

[87 rows x 4 columns]

Note: Only the head of the SFrame is printed.

You can use print_rows(num_rows=m, num_columns=n) to print more rows and columns.

In [19]:
```
print(simple_features_model.evaluate(test_set))
print(advanced_features_model.evaluate(test_set))
```

```
{'max_error': 4140574.2802349306, 'rmse': 255200.21790797458}
{'max_error': 3189783.7899840837, 'rmse': 155586.723923284}
```

1. Selection and summary statistics: In the notebook we covered in the module, we discovered which neighborhood (zip code) of Seattle had the highest average house sale price. Now, take the sales data, select only the houses with this zip code, and compute the average price. Save this result to answer the quiz at the end.

In [20]:
```
home_sframe[home_sframe['zipcode']=='98146' ]
```

Out[20]:

id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot
2414600126	2015-04-15 00:00:00+00:00	229500.0	3.0	1.0	1780.0	7470.0
0084000105	2014-05-07 00:00:00+00:00	255000.0	5.0	2.25	2060.0	8632.0
1909600046	2014-07-03 00:00:00+00:00	445838.0	3.0	2.5	2250.0	5692.0
7454001200	2014-06-04 00:00:00+00:00	390000.0	3.0	2.25	1250.0	7500.0
7520000520	2014-09-05 00:00:00+00:00	232000.0	2.0	1.0	1240.0	12092.0
7520000520	2015-03-11 00:00:00+00:00	240500.0	2.0	1.0	1240.0	12092.0
1223039290	2014-09-05 00:00:00+00:00	403950.0	4.0	2.5	2120.0	13780.0
0284000223	2014-09-16 00:00:00+00:00	578000.0	3.0	1.75	2120.0	10875.0
2586800270	2015-04-07 00:00:00+00:00	425000.0	4.0	1.0	1260.0	7645.0
1842000140	2014-07-30 00:00:00+00:00	335000.0	3.0	1.75	1570.0	7500.0

view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcod
0	3	7.0	1050.0	730.0	1960.0	0.0	9814
0	3	7.0	1030.0	1030.0	1962.0	0.0	9814
0	3	8.0	2250.0	0.0	2000.0	0.0	9814
0	5	7.0	1250.0	0.0	1942.0	0.0	9814
0	3	6.0	960.0	280.0	1922.0	1984.0	9814

0	3	6.0	960.0	280.0	1922.0	1984.0	9814€
0	3	8.0	2120.0	0.0	1993.0	0.0	9814€
2	3	8.0	1540.0	580.0	1977.0	0.0	9814€
0	3	6.0	1260.0	0.0	1925.0	0.0	9814€
1	3	7.0	1300.0	270.0	1953.0	0.0	9814€

long	sqft_living15	sqft_lot15
-122.33659507	1780.0	8113.0
-122.33506693	1010.0	11680.0
-122.3785914	1320.0	5390.0
-122.37318137	1280.0	7392.0
-122.35226024	1820.0	7460.0
-122.35226024	1820.0	7460.0
-122.36484286	1880.0	12000.0

In [21]:
```
# Find the zip code which has the highest mean price
```

In [22]:
```
home_sframe['price'].max()
```

Out[22]: 7700000.0

In [23]:
```
home_sframe['price'==7700000.0]
```

Out[23]:
```
{'id': '7129300520',
 'date': datetime.datetime(2014, 10, 13, 0, 0, tzinfo=GMT +0.0),
 'price': 221900.0,
 'bedrooms': 3.0,
 'bathrooms': 1.0,
 'sqft_living': 1180.0,
 'sqft_lot': 5650.0,
 'floors': 1.0,
 'waterfront': 0,
 'view': 0,
 'condition': 3,
 'grade': 7.0,
 'sqft_above': 1180.0,
 'sqft_basement': 0.0,
 'yr_built': 1955.0,
 'yr_renovated': 0.0,
 'zipcode': '98178',
 'lat': 47.51123398,
 'long': -122.25677536,
 'sqft_living15': 1340.0,
 'sqft_lot15': 5650.0}
```

In [24]:
```
# find the zip of the property with the highest home value
home_sframe['price'==home_sframe['price'].max()]
```

Out[24]:
```
{'id': '7129300520',
```

```
'date': datetime.datetime(2014, 10, 13, 0, 0, tzinfo=GMT +0.0),
'price': 221900.0,
'bedrooms': 3.0,
'bathrooms': 1.0,
'sqft_living': 1180.0,
'sqft_lot': 5650.0,
'floors': 1.0,
'waterfront': 0,
'view': 0,
'condition': 3,
'grade': 7.0,
'sqft_above': 1180.0,
'sqft_basement': 0.0,
'yr_built': 1955.0,
'yr_renovated': 0.0,
'zipcode': '98178',
'lat': 47.51123398,
'long': -122.25677536,
'sqft_living15': 1340.0,
```

find avg home value per zipcode

In [27]:
```
zip_code_arr_raw = home_sframe['zipcode'].unique()
```

In [28]:
```
home_sframe.filter_by(zip_code_arr_raw,'zipcode')
```

Out[28]:

id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot
7129300520	2014-10-13 00:00:00+00:00	221900.0	3.0	1.0	1180.0	5650.0
6414100192	2014-12-09 00:00:00+00:00	538000.0	3.0	2.25	2570.0	7242.0
5631500400	2015-02-25 00:00:00+00:00	180000.0	2.0	1.0	770.0	10000.0
2487200875	2014-12-09 00:00:00+00:00	604000.0	4.0	3.0	1960.0	5000.0
1954400510	2015-02-18 00:00:00+00:00	510000.0	3.0	2.0	1680.0	8080.0
7237550310	2014-05-12 00:00:00+00:00	1225000.0	4.0	4.5	5420.0	101930.0
1321400060	2014-06-27 00:00:00+00:00	257500.0	3.0	2.25	1715.0	6819.0
2008000270	2015-01-15 00:00:00+00:00	291850.0	3.0	1.5	1060.0	9711.0
2414600126	2015-04-15 00:00:00+00:00	229500.0	3.0	1.0	1780.0	7470.0
3793500160	2015-03-12 00:00:00+00:00	323000.0	3.0	2.5	1890.0	6560.0

view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcod
0	3	7.0	1180.0	0.0	1955.0	0.0	9817

0	3	7.0	2170.0	400.0	1951.0	1991.0	9812
0	3	6.0	770.0	0.0	1933.0	0.0	9802
0	5	7.0	1050.0	910.0	1965.0	0.0	9813
0	3	8.0	1680.0	0.0	1987.0	0.0	9807
0	3	11.0	3890.0	1530.0	2001.0	0.0	9805
0	3	7.0	1715.0	0.0	1995.0	0.0	9800
0	3	7.0	1060.0	0.0	1963.0	0.0	9819
0	3	7.0	1050.0	730.0	1960.0	0.0	9814
0	3	7.0	1890.0	0.0	2003.0	0.0	9803

long	sqft_living15	sqft_lot15
-122.25677536	1340.0	5650.0
-122.3188624	1690.0	7639.0
-122.23319601	2720.0	8062.0
-122.39318505	1360.0	5000.0
-122.04490059	1800.0	7503.0
-122.00528655	4760.0	101930.0
-122.32704857	2238.0	6819.0
-122.31457273	1650.0	9711.0

In [31]:
```
import turicreate.aggregate as agg
avg_price_per_zipcode = home_sframe.groupby(key_column_names='zipcode',operat
```

In [32]:
```
avg_price_per_zipcode
```

Out[32]:

zipcode	avg_price
98033	803719.5324074076
98032	251296.24
98065	527961.2032258068
98077	682774.8787878787
98144	594547.6413994174
98136	551688.6730038024
98115	619900.5506003429
98075	790576.6685236767
98034	521652.8587155963
98058	353608.63516483514

[70 rows x 2 columns]

Note: Only the head of the SFrame is printed.

You can use print_rows(num_rows=m, num_columns=n) to print more rows and columns

In [34]:
```
avg_price_per_zipcode['avg_price'].max()
```

Out[34]: 2160606.599999999

Filtering data: What fraction of the houses have living space between 2000 sq.ft. and 4000 sq.ft.?

In [41]:
```
home_2k_4k_sframe = home_sframe[(home_sframe['sqft_living'] > 2000) & (home_s
home_2k_4k_sframe
```

Out[41]:

id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot
6414100192	2014-12-09 00:00:00+00:00	538000.0	3.0	2.25	2570.0	7242.0
1736800520	2015-04-03 00:00:00+00:00	662500.0	3.0	2.5	3560.0	9796.0
9297300055	2015-01-24 00:00:00+00:00	650000.0	4.0	3.0	2950.0	5000.0
2524049179	2014-08-26 00:00:00+00:00	2000000.0	3.0	2.75	3050.0	44867.0
7137970340	2014-07-03 00:00:00+00:00	285000.0	5.0	2.5	2270.0	6300.0
3814700200	2014-11-20 00:00:00+00:00	329000.0	3.0	2.25	2450.0	6500.0
1794500383	2014-06-26 00:00:00+00:00	937000.0	3.0	1.75	2450.0	2691.0
1873100390	2015-03-02 00:00:00+00:00	719000.0	4.0	2.5	2570.0	7173.0
8562750320	2014-11-10 00:00:00+00:00	580500.0	3.0	2.5	2320.0	3980.0
0461000390	2014-06-24 00:00:00+00:00	687500.0	4.0	1.75	2330.0	5000.0

view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcod
0	3	7.0	2170.0	400.0	1951.0	1991.0	98125
0	3	8.0	1860.0	1700.0	1965.0	0.0	98007
3	3	9.0	1980.0	970.0	1979.0	0.0	98126
4	3	9.0	2330.0	720.0	1968.0	0.0	98040
0	3	8.0	2270.0	0.0	1995.0	0.0	98092
0	4	8.0	2450.0	0.0	1985.0	0.0	98030
0	3	8.0	1750.0	700.0	1915.0	0.0	98119
0	3	8.0	2570.0	0.0	2005.0	0.0	98052
0	3	8.0	2320.0	0.0	2003.0	0.0	98027
0	4	7.0	1510.0	820.0	1929.0	0.0	98117

long	sqft_living15	sqft_lot15
-122.3188624	1690.0	7639.0
-122.14529566	2210.0	8925.0
-122.37541218	2140.0	4000.0
-122.23345881	4110.0	20336.0
-122.16892624	2240.0	7005.0
-122.17228981	2200.0	6865.0
-122.35985573	1760.0	3573.0
-122.11029785	2630.0	6026.0
-122.06971484	2580.0	3980.0

In [50]:

```
print("Total homes between 2000 and 4000 sq. ft. =",home_2k_4k_sframe.num_row
```

```
Total homes between 2000 and 4000 sq. ft. = 9111
```

In [52]:

```
print("Total homes =",home_sframe.num_rows())
```

```
Total homes = 21613
```

In [53]:

```
home_2k_4k_percent = (home_2k_4k_sframe.num_rows() / home_sframe.num_rows())
print('%age of homes found = ', home_2k_4k_percent )
```

```
%age of homes found =  0.4215518437977143
```

In [55]:

```
print (simple_features_model.evaluate(test_set))
print (advanced_features_model.evaluate(test_set))
print('Difference -> ', simple_features_model.evaluate(test_set)['rmse'] - ad
```

```
{'max_error': 4140574.2802349306, 'rmse': 255200.21790797458}
{'max_error': 3189783.7899840837, 'rmse': 155586.723923284}
Difference ->  99613.49398469058
```

In [56]:

```
# The above is not the right answer. It should be 21569 approx. Still trying
```

In [ ]: