This is for the assignment

What you will do Now you are ready! We are going do three tasks in this assignment. There are 3 results you need to gather along the way to enter into the quiz after this reading.

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 [2]:

```
import graphlab
```

Load some house sales data ¶

Dataset is from house sales in King County, the region where the city of Seattle, WA is located.

In [3]:

```
sales = graphlab.SFrame('home_data.gl/')
```

```
This non-commercial license of GraphLab Create for academic use is a
ssigned to sujingw@hotmail.com and will expire on May 07, 2020.
```

```
[INFO] graphlab.cython.cy_server: GraphLab Create v2.1 started. Logg
ing: /tmp/graphlab_server_1557851486.log
```

In [3]:

```
sales
```

Out[3]:

id	date	price	bedrooms	bathrooms	sqft_liv
7129300520	2014-10-13 00:00:00+00:00	221900	3	1	1180
6414100192	2014-12-09 00:00:00+00:00	538000	3	2.25	2570
5631500400	2015-02-25 00:00:00+00:00	180000	2	1	770
2487200875	2014-12-09 00:00:00+00:00	604000	4	3	1960
1954400510	2015-02-18 00:00:00+00:00	510000	3	2	1680
7237550310	2014-05-12 00:00:00+00:00	1225000	4	4.5	5420
1321400060	2014-06-27 00:00:00+00:00	257500	3	2.25	1715
2008000270	2015-01-15 00:00:00+00:00	291850	3	1.5	1060
2414600126	2015-04-15 00:00:00+00:00	229500	3	1	1780
3793500160	2015-03-12 00:00:00+00:00	323000	3	2.5	1890

view	condition	grade	sqft_above	sqft_basement	yr_built	yr_re
0	3	7	1180	0	1955	
0	3	7	2170	400	1951	1
0	3	6	770	0	1933	
0	5	7	1050	910	1965	
0	3	8	1680	0	1987	
0	3	11	3890	1530	2001	
0	3	7	1715	0	1995	
0	3	7	1060	0	1963	
0	3	7	1050	730	1960	
0	3	7	1890	0	2003	

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

In [4]:

```
sales.show()
```

```
Canvas is accessible via web browser at the URL: http://localhost:50
642/index.html
Opening Canvas in default web browser.
```

So we know that the zipcode is 98039.

In [6]:

```
zipcodes = sales[sales['zipcode']=='98039']
```

In [7]:

```
zipcodes
```

Out[7]:

id	date	price	bedrooms	bathrooms	sqft_liv
3625049014	2014-08-29 00:00:00+00:00	2950000	4	3.5	4860
2540700110	2015-02-12 00:00:00+00:00	1905000	4	3.5	4210
3262300940	2014-11-07 00:00:00+00:00	875000	3	1	1220
3262300940	2015-02-10 00:00:00+00:00	940000	3	1	1220
6447300265	2014-10-14 00:00:00+00:00	4000000	4	5.5	7080
2470100110	2014-08-04 00:00:00+00:00	5570000	5	5.75	9200
2210500019	2015-03-24 00:00:00+00:00	937500	3	1	1320
6447300345	2015-04-06 00:00:00+00:00	1160000	4	3	2680
6447300225	2014-11-06 00:00:00+00:00	1880000	3	2.75	2620
2525049148	2014-10-07 00:00:00+00:00	3418800	5	5	5450

view	condition	grade	sqft_above	sqft_basement	yr_built	yr_re
0	3	12	4860	0	1996	
0	3	11	4210	0	2001	
0	4	7	1220	0	1955	
0	4	7	1220	0	1955	
0	3	12	5760	1320	2008	
0	3	13	6200	3000	2001	
0	4	7	1320	0	1954	
2	3	8	2680	0	1902	1
1	4	9	2620	0	1949	
0	3	11	5450	0	2014	

long	sqft_living15	sqft_lot15
-122.23040939	3580.0	16054.0
-122.2245047	3520.0	18564.0
-122.23554392	1910.0	8119.0
-122.23554392	1910.0	8119.0

In [8]:

```
zipcodes['price'].mean()
```

Out[8]:

```
2160606.5999999996
```

So the above will be the answer!

2. Filtering data

One of the key features we used in our model was the number of square feet of living space (‘sqft_living’) in the house. For this part, we are going to use the idea of filtering (selecting) data.

In particular, we are going to use logical filters to select rows of an SFrame. You can find more info in the Logical Filter section of this documentation (https://turi.com/products/create/docs/generated/graphlab.SFrame.html).

Using such filters, first select the houses that have ‘sqft_living’ higher than 2000 sqft but no larger than 4000 sqft.

What fraction of the all houses have ‘sqft_living’ in this range? Save this result to answer the quiz at the end.

In [9]:

```
houses = sales[(sales['sqft_living'] > 2000) & (sales['sqft_living'] <= 4000)]
```

In [10]:

```
houses
```

Out[10]:

id	date	price	bedrooms	bathrooms	sqft_liv
6414100192	2014-12-09 00:00:00+00:00	538000	3	2.25	2570
1736800520	2015-04-03 00:00:00+00:00	662500	3	2.5	3560
9297300055	2015-01-24 00:00:00+00:00	650000	4	3	2950
2524049179	2014-08-26 00:00:00+00:00	2000000	3	2.75	3050
7137970340	2014-07-03 00:00:00+00:00	285000	5	2.5	2270
3814700200	2014-11-20 00:00:00+00:00	329000	3	2.25	2450
1794500383	2014-06-26 00:00:00+00:00	937000	3	1.75	2450
1873100390	2015-03-02 00:00:00+00:00	719000	4	2.5	2570
8562750320	2014-11-10 00:00:00+00:00	580500	3	2.5	2320
0461000390	2014-06-24 00:00:00+00:00	687500	4	1.75	2330

view	condition	grade	sqft_above	sqft_basement	yr_built	yr_re
0	3	7	2170	400	1951	1
0	3	8	1860	1700	1965	
3	3	9	1980	970	1979	
4	3	9	2330	720	1968	
0	3	8	2270	0	1995	
0	4	8	2450	0	1985	
0	3	8	1750	700	1915	
0	3	8	2570	0	2005	
0	3	8	2320	0	2003	
0	4	7	1510	820	1929	

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

and this is it.

In [20]:

```
len(houses)*1.0/len(sales)
```

Out[20]:

0.42187572294452413

3. Building a regression model with several more features

In the sample notebook, we built two regression models to predict house prices, one using just 'sqft_living' and the other one using a few more features, we called this set

In [12]:

```
my_features = ['bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors', 'zi
pcode']
```

Now, going back to the original dataset, you will build a model using the following features:

In [11]:

```
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
]
```

Compute the RMSE (root mean squared error) on the test_data for the model using just my_features, and for the one using advanced_features.

Note 1: both models must be trained on the original sales dataset, not the filtered one.

Note 2: when doing the train-test split, make sure you use seed=0, so you get the same training and test sets, and thus results, as we do.

Note 3: in the module we discussed residual sum of squares (RSS) as an error metric for regression, but GraphLab Create uses root mean squared error (RMSE). These are two common measures of error regression, and RMSE is simply the square root of the mean RSS:

(Important note: when answering the question below using GraphLab Create, when you call the linear_regression.create() function, make sure you use the parameter validation_set=None, as done in the notebook you download above. When you use regression GraphLab Create, it sets aside a small random subset of the data to validate some parameters. This process can cause fluctuations in the final RMSE, so we will avoid it to make sure everyone gets the same answer.)

What is the difference in RMSE between the model trained with my_features and the one trained with advanced_features? Save this result to answer the quiz at the end.

In [13]:

```
train_data,test_data = sales.random_split(.8,seed=0)
```

In [15]:

```
sqft_model = graphlab.linear_regression.create(train_data, target='price', featu
res = my_features,validation_set=None)
```

```
Linear regression:

--------------------------------------------------------

Number of examples          : 17384

Number of features          : 6

Number of unpacked features : 6

Number of coefficients    : 115

Starting Newton Method

--------------------------------------------------------

+-----------+----------+--------------+--------------------+--------
-------+

| Iteration | Passes   | Elapsed Time | Training-max_error | Trainin
g-rmse |

+-----------+----------+--------------+--------------------+--------
-------+

| 1         | 2        | 1.051274     | 3763208.270523     | 181908.
848367 |

+-----------+----------+--------------+--------------------+--------
-------+

SUCCESS: Optimal solution found.
```

In [16]:

```
print sqft_model.evaluate(test_data)
```

```
{'max_error': 3486584.509381705, 'rmse': 179542.4333126903}
```

That's the answer one.

In [17]:

```
advan_model = graphlab.linear_regression.create(train_data, target='price', feat
ures = advanced_features,validation_set=None)
```

```
Linear regression:

--------------------------------------------------------

Number of examples          : 17384

Number of features          : 18

Number of unpacked features : 18

Number of coefficients    : 127

Starting Newton Method

--------------------------------------------------------

+-----------+----------+--------------+--------------------+--------
-------+

| Iteration | Passes   | Elapsed Time | Training-max_error | Trainin
g-rmse |

+-----------+----------+--------------+--------------------+--------
-------+

| 1         | 2        | 0.122482     | 3469012.450686     | 154580.
940736 |

+-----------+----------+--------------+--------------------+--------
-------+

SUCCESS: Optimal solution found.
```

In [18]:

```
print advan_model.evaluate(test_data)
```

```
{'max_error': 3556849.413858208, 'rmse': 156831.1168021901}
```

Answer two.