Behind_The_Scene

October 28, 2019

[80]:
```
import pandas as pd
data = pd.read_csv('data.csv')
data.head()
```

[80]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated	street	city	statezip	country
0	2014-05-02 00:00:00	313000.0	3.0	1.5	1340	7912	1.5	0	0	3	1340	0	1955	2005	18810 Densmore Ave N	Shoreline	WA 98133	USA
1	2014-05-02 00:00:00	2384000.0	5.0	2.5	3650	9050	2.0	0	4	5	3370	280	1921	0	709 W Blaine St	Seattle	WA 98119	USA
2	2014-05-02 00:00:00	342000.0	3.0	2.0	1930	11947	1.0	0	0	4	1930	0	1966	0	26206-26214 143rd Ave SE	Kent	WA 98042	USA
3	2014-05-02 00:00:00	420000.0	3.0	2.2	2000	8030	1.0	0	0	4	1000	1000	1963	0	857 170th Pl NE	Bellevue	WA 98008	USA
4	2014-05-02 00:00:00	550000.0	4.0	2.5	1940	10500	1.0	0	0	4	1140	800	1976	1992	9105 170th Ave NE	Redmond	WA 98052	USA

[81]:
```
data.shape
```

[81]:
```
(4600, 18)
```

[82]:
```
data['bedrooms'].value_counts()
```

[82]:
```
3.0    2032
4.0    1531
2.0     566
5.0     353
6.0      61
1.0      38
7.0      14
0.0       2
8.0       2
```

```
9.0       1
Name: bedrooms, dtype: int64
```

[83]:
```
data[data['bedrooms'] == 9]
```

[83]:
```
                    date    price  bedrooms  bathrooms  sqft_living  sqft_lot  \
241  2014-05-07 00:00:00 599999.0       9.0        4.5         3830      6988

     floors  waterfront  view  condition  sqft_above  sqft_basement  yr_built  \
241     2.5           0     0          3        2450           1380      1938

     yr_renovated               street     city  statezip country
241          2003  8809 Densmore Ave N  Seattle  WA 98103     USA
```

[84]:
```
data['price'].max()
```

[84]:
```
26590000.0
```

[85]:
```
data[data['price'] == 26590000.0]
```

[85]:
```
                    date      price  bedrooms  bathrooms  sqft_living  \
4350 2014-07-03 00:00:00 26590000.0       3.0        2.0         1180

      sqft_lot  floors  waterfront  view  condition  sqft_above  \
4350      7793     1.0           0     0          4        1180

      sqft_basement  yr_built  yr_renovated             street  city  \
4350              0      1992             0  12005 SE 219th Ct  Kent

      statezip country
4350  WA 98031     USA
```

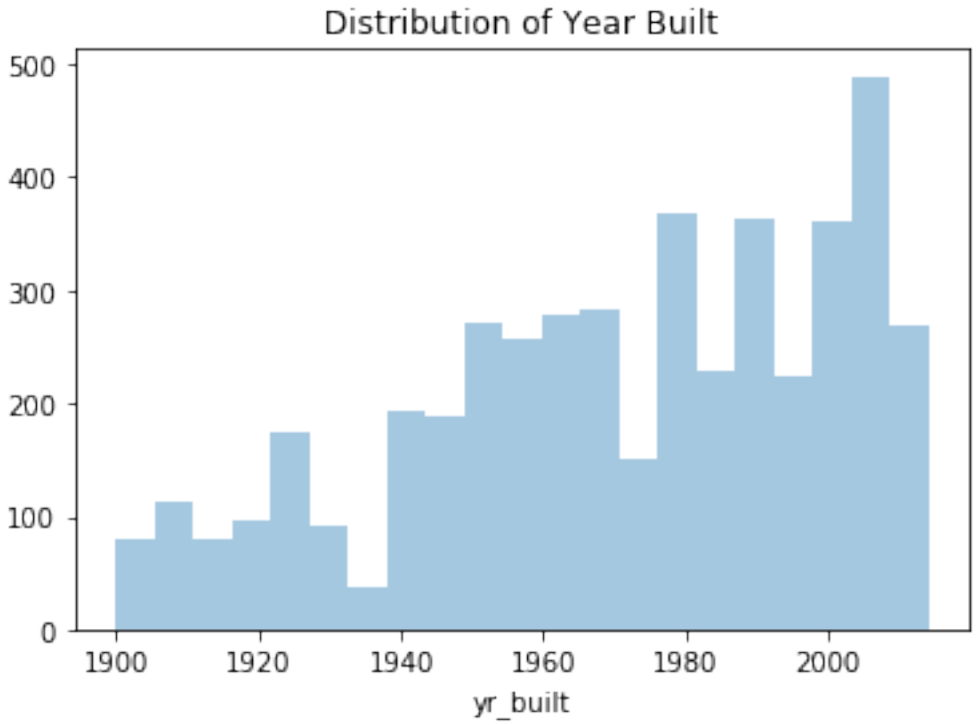
[86]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4600 entries, 0 to 4599
Data columns (total 18 columns):
date             4600 non-null object
price            4600 non-null float64
bedrooms         4600 non-null float64
bathrooms        4600 non-null float64
sqft_living      4600 non-null int64
sqft_lot         4600 non-null int64
floors           4600 non-null float64
waterfront       4600 non-null int64
view             4600 non-null int64
condition        4600 non-null int64
sqft_above       4600 non-null int64
sqft_basement    4600 non-null int64
yr_built         4600 non-null int64
yr_renovated     4600 non-null int64
```

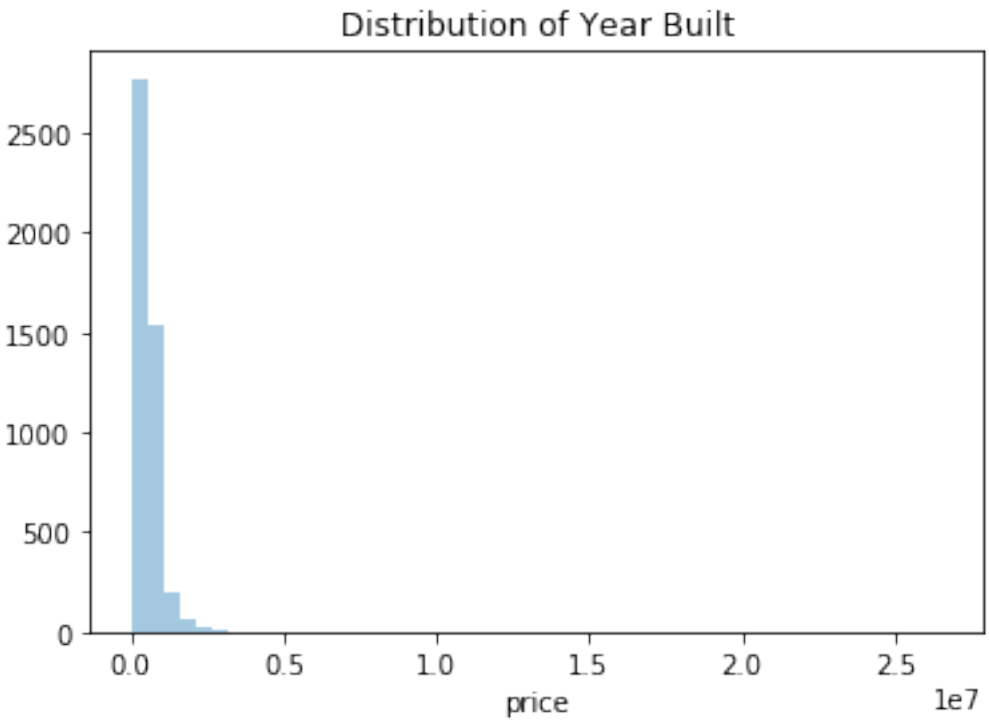
```
street           4600 non-null object
city             4600 non-null object
statezip         4600 non-null object
country          4600 non-null object
dtypes: float64(4), int64(9), object(5)
memory usage: 647.0+ KB
```

[87]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

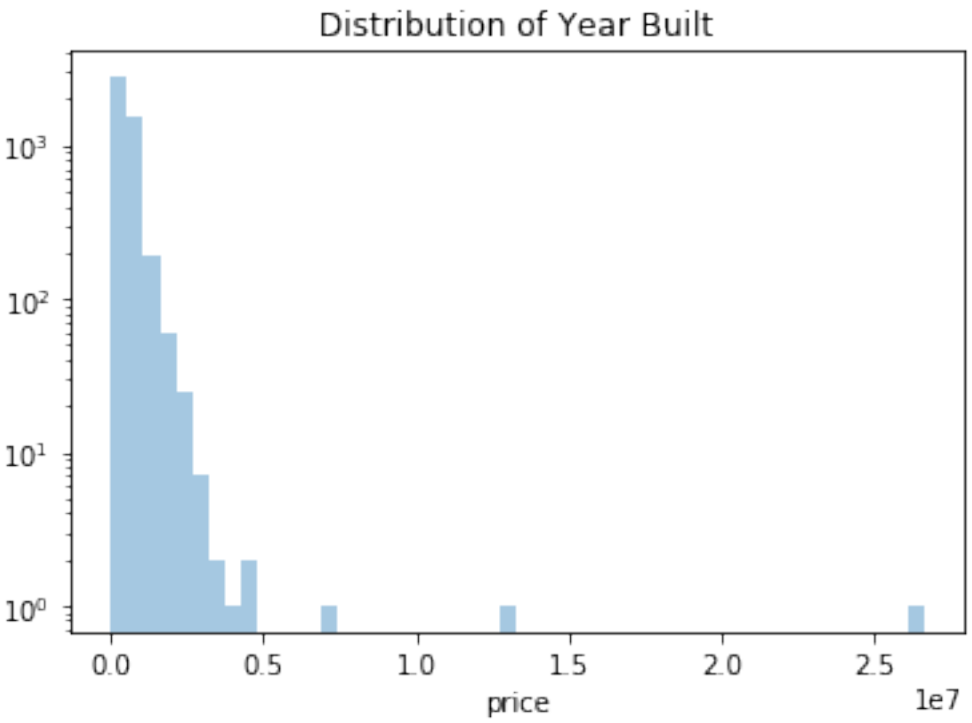
[88]:
```
plt.title("Distribution of Year Built")
sns.distplot(data['yr_built'],kde=False);
```



[89]:
```
plt.title("Distribution of Year Built")
sns.distplot(data['price'],kde=False,);
```

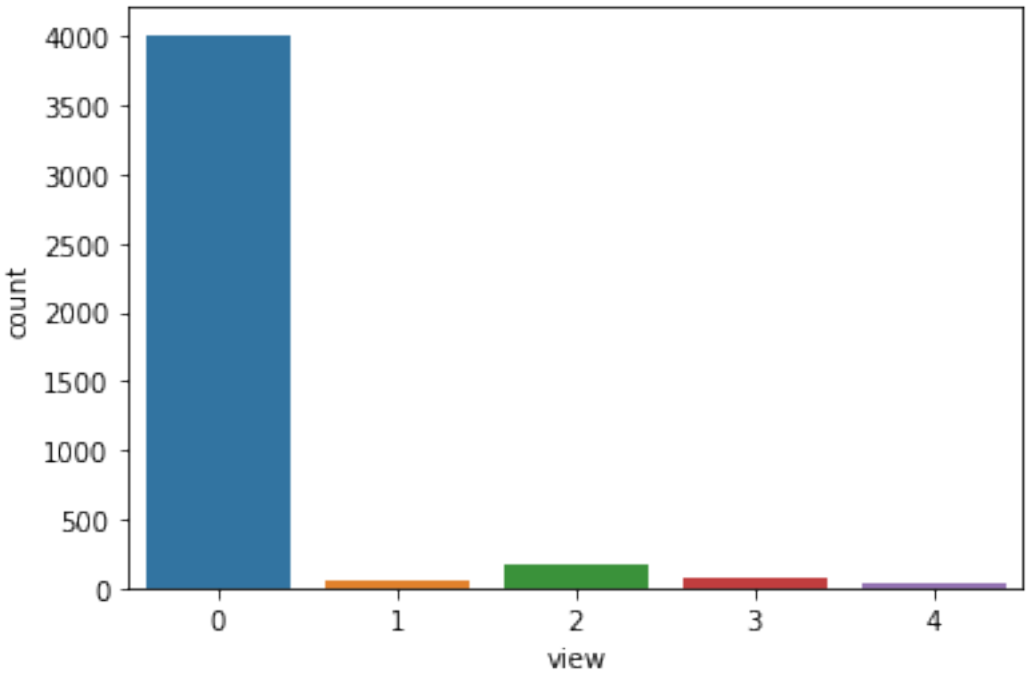


[90]:
```
plt.title("Distribution of Year Built")
ax = sns.distplot(data['price'],kde=False);
ax.set_yscale('log')
#ax.set_xscale('log')
```

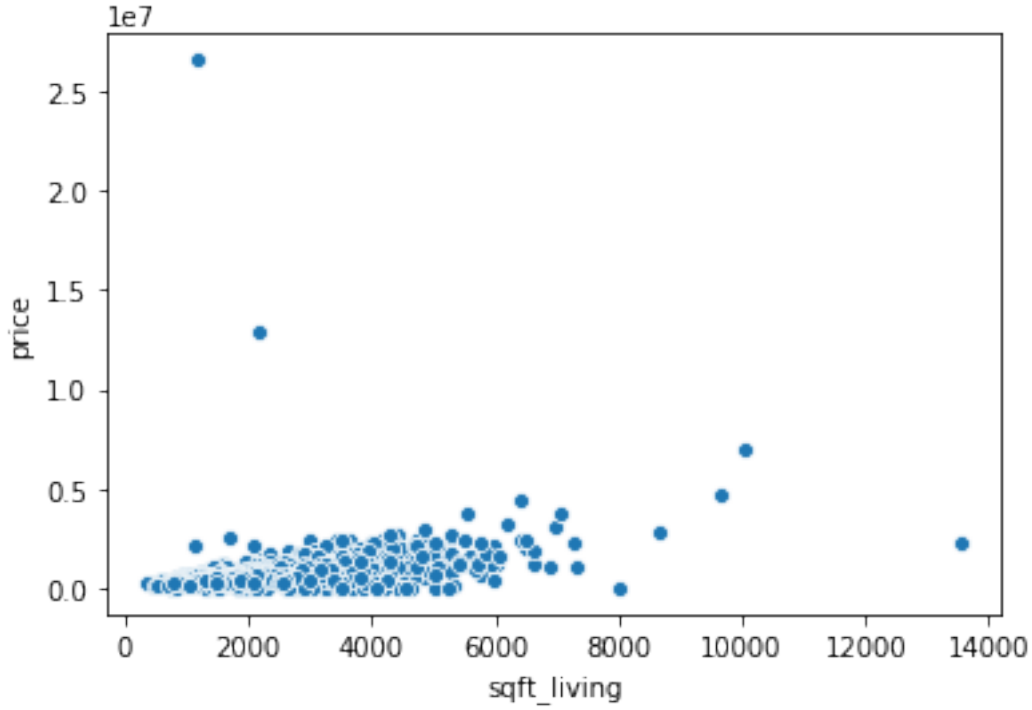


[164]:
```
sns.countplot(data['view'])
```

[164]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x19e2b3da908>
```



[107]:
```
sns.scatterplot('sqft_living', 'price', data=data);
```



[118]:
```
q1 = data['price'].describe()[4]
q3 = data['price'].describe()[6]
```

[119]:
```
data['price'].describe()
```

[119]:
```
count       4600.00
mean      551962.99
std       563834.70
min            0.00
25%       322875.00
50%       460943.46
75%       654962.50
max     26590000.00
Name: price, dtype: float64
```

[120]:
```
data['sqft_living'].describe()
```

[120]:
```
count    4600.00
mean     2139.35
std       963.21
min       370.00
25%      1460.00
```

```
50%      1980.00
75%      2620.00
max     13540.00
Name: sqft_living, dtype: float64
```

[121]:
```
print(q1 - (1.5*(q3-q1)))
print("--------------")
print(q3 + (1.5*(q3-q1)))
```

```
-175256.25
--------------
1153093.75
```

[124]:
```
data = data[data['price'] <= 1153093.75]
```

[125]:
```
list(data.corr(method='spearman')['price'].sort_values().index)
```

[125]:
```
['yr_renovated',
 'condition',
 'sqft_lot',
 'waterfront',
 'yr_built',
 'view',
 'sqft_basement',
 'bedrooms',
 'floors',
 'bathrooms',
 'sqft_above',
 'sqft_living',
 'price']
```

[126]:
```
data.corr(method='spearman')['price']
```

[126]:
```
price            1.00
bedrooms         0.30
bathrooms        0.45
sqft_living      0.58
sqft_lot         0.03
floors           0.30
waterfront       0.04
view             0.19
condition        0.02
sqft_above       0.49
sqft_basement    0.19
yr_built         0.09
yr_renovated    -0.07
Name: price, dtype: float64
```

[140]:
```
data_model = data[[ 'bedrooms',
 'floors',
```

```
 'bathrooms',
 'sqft_above',
 'sqft_living',
 'price']]
```

[141]:
```
pd.options.display.float_format = '{:.2f}'.format
```

[142]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

[ ]:
```

```

[143]:
```
X = data_model.drop('price',axis=1)
y = data_model['price']
```

[144]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2,␣
↪random_state=1)
```

[145]:
```
model = LinearRegression()
```

[146]:
```
model.fit(X_train, y_train)
```

[146]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

[157]:
```
val = model.predict(np.array([[1, 3, 2, 2000 ,2000]]))
print(val[0][0])
```

```
      ␣
↪---------------------------------------------------------------------------

      IndexError                                Traceback (most recent call␣
↪last)

      <ipython-input-157-03becaf8f493> in <module>
        1 val = model.predict(np.array([[1, 3, 2, 2000 ,2000]]))
  ----> 2 print(val[0][0])


      IndexError: invalid index to scalar variable.
```

[159]:
```
val[0]
```

[159]:
```
594209.5224014644
```

[148]:
```
from sklearn.metrics import mean_squared_error
np.sqrt(mean_squared_error(y_test, model.predict(X_test)))
```

[148]:
```
181084.75130786453
```

[149]:
```
y_test
```

[149]:
```
4373    346750.00
4570    318000.00
4008    645000.00
4019    821000.00
3946   1150000.00
1741    850000.00
4318    342500.00
588      90000.00
3010    339990.00
3154    285000.00
4124    375000.00
4115    585000.00
1979    356000.00
4486    229629.50
377     230000.00
93      770000.00
796     248000.00
918     442500.00
3722    478000.00
1038    515000.00
3170    687500.00
1875    453500.00
3472    800000.00
764     839000.00
2914    605000.00
840     377500.00
2556    375000.00
1473    590000.00
4179    569000.00
4045    230000.00
          ...
48      445700.00
4095    235000.00
3548    279000.00
4078    789900.00
4359    439333.33
2445    230000.00
3495    900000.00
3190    225000.00
3342    283000.00
1239   1050000.00
1071    368250.00
2254    700000.00
1255    275000.00
4365    444845.00
1117    318989.00
3777    725000.00
```

```
2044    185000.00
1760    474900.00
2674    198000.00
1114    285000.00
1862    289950.00
3461    580000.00
1187    635700.00
2258    680000.00
2450    810000.00
1816    285000.00
3987    527000.00
3994    320000.00
4048    900000.00
385     494000.00
Name: price, Length: 872, dtype: float64
```

[152]:
```
pd.DataFrame({'a':(model.predict(X_test)).reshape(1,872)[0],
              'b':y_test})
```

[152]:
```
              a          b
4373 421119.83  346750.00
4570 426518.96  318000.00
4008 584421.90  645000.00
4019 605660.57  821000.00
3946 702403.56 1150000.00
1741 372716.57  850000.00
4318 356293.54  342500.00
588  291255.97   90000.00
3010 590457.66  339990.00
3154 443891.62  285000.00
4124 417555.86  375000.00
4115 574758.20  585000.00
1979 312193.97  356000.00
4486 341185.06  229629.50
377  362529.77  230000.00
93   549631.76  770000.00
796  464872.04  248000.00
918  425956.72  442500.00
3722 441539.82  478000.00
1038 581200.67  515000.00
3170 529391.10  687500.00
1875 483317.01  453500.00
3472 471053.97  800000.00
764  336759.91  839000.00
2914 573147.59  605000.00
840  407698.92  377500.00
2556 314868.44  375000.00
1473 444019.86  590000.00
```

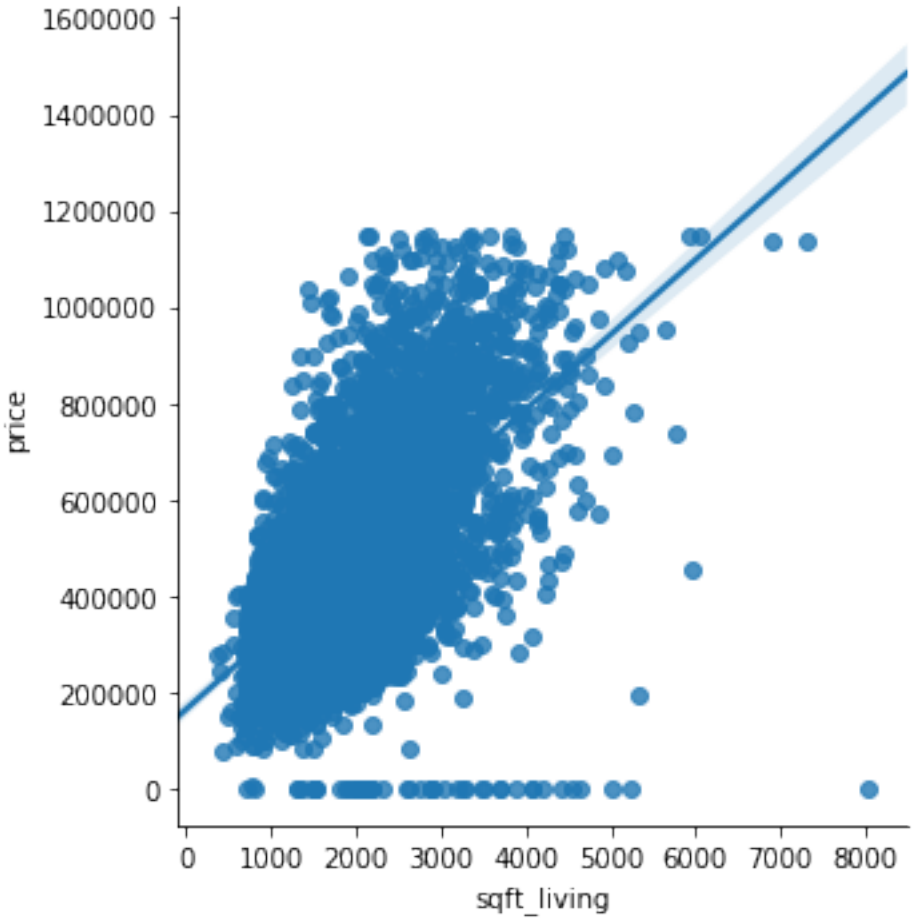
```
4179 521944.16  569000.00
4045 362747.89  230000.00
...         ...        ...
48   424169.41  445700.00
4095 328300.13  235000.00
3548 404243.07  279000.00
4078 728846.98  789900.00
4359 684549.90  439333.33
2445 313804.59  230000.00
3495 714490.55  900000.00
3190 350257.78  225000.00
3342 281998.98  283000.00
1239 876762.54 1050000.00
1071 522811.79  368250.00
2254 624224.57  700000.00
1255 465492.82  275000.00
4365 434227.92  444845.00
1117 431067.03  318989.00
3777 478712.23  725000.00
2044 459565.85  185000.00
1760 424217.86  474900.00
2674 383516.64  198000.00
1114 569213.86  285000.00
1862 404149.29  289950.00
3461 395716.81  580000.00
1187 666751.52  635700.00
2258 427522.43  680000.00
2450 472565.77  810000.00
1816 634605.28  285000.00
3987 577095.98  527000.00
3994 518961.77  320000.00
4048 688705.22  900000.00
385  548581.65  494000.00

[872 rows x 2 columns]
```

[153]:
```
data_model.head()
```

[153]:
```
   bedrooms  floors  bathrooms  sqft_above  sqft_living     price
0      3.00    1.50       1.50        1340         1340 313000.00
2      3.00    1.00       2.00        1930         1930 342000.00
3      3.00    1.00       2.25        1000         2000 420000.00
4      4.00    1.00       2.50        1140         1940 550000.00
5      2.00    1.00       1.00         880          880 490000.00
```

[154]:
```
ax = sns.lmplot(x='sqft_living', y='price', data=data_model, ci=95)
```



[ ]:
```
X = data_model[['sqft_living']]
y = data_model[['price']]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2,␣
↪random_state=1)

model = LinearRegression()

model.fit(X_train, y_train)

model.predict(X_test)
```

[41]:
```
from sklearn.metrics import mean_squared_error
np.sqrt(mean_squared_error(y_test, model.predict(X_test)))
```

[41]:
```
269625.67935840687
```

[160]:
```
def modelling(floors, bed, bath, sqft_above, sqft_living):
    prediction = model.predict([[floors, bed, bath, sqft_above, sqft_living]])
    return prediction[0]
```

[161]:
```
input_floors = float(input(("Masukkan jumlah lantai: ")))
input_bedrooms = float(input(("Masukkan jumlah kamar tidur: ")))
input_bathrooms=float(input(("Masukkan jumlah kamar mandi: ")))
input_sqft_above=float(input(("Masukkan luas ruangan atas: ")))
input_sqft_living=float(input(("Masukkan luas ruangan tamu: ")))

modelling(input_floors, input_bedrooms, input_bathrooms, input_sqft_above,␣
 ↪input_sqft_living)
```

```
Masukkan jumlah lantai: 2
Masukkan jumlah kamar tidur: 2
Masukkan jumlah kamar mandi: 5
Masukkan luas ruangan atas: 2000
Masukkan luas ruangan tamu: 500
```

[161]:
```
233688.21791751985
```

[ ]:
```
175399.69519344968
233688.21791751985
```