```
# Statistic - Quantitiy & Quality

# How many unique values(Category) do we have?
df['zipcode'].nunique() #unique array
df['zipcode'].unique() #number of unique datas
df['zipcode'].value_counts() #unique() + nunique()

# Mean, median and mode
df['sqft_living'].mean() #average
df['sqft_living'].median() #가운데 수
df['sqft_living'].mode() #가장 많은 수

df.describe() #Shows Descriptive Stats

df[df.bedrooms == 33]

df.sqft_living.plot()
# row: 행 개수
# column: sqft_living 값

print('df[df.sqft_living > 8000]')
df[df.sqft_living > 8000]

print('plt.hist(df.price, bins=200);')
plt.hist(df.price, bins=200);

print('plt.scatter(df.sqft_living, df.price)')
plt.scatter(df.sqft_living, df.price)


# Model
df.describe() #Shows Descriptive Stats

df[df.bedrooms == 33]

df.sqft_living.plot()
# row: 행 개수
# column: sqft_living 값

print('df[df.sqft_living > 8000]')
df[df.sqft_living > 8000]

print('plt.hist(df.price, bins=200);')
plt.hist(df.price, bins=200);

print('plt.scatter(df.sqft_living, df.price)')
plt.scatter(df.sqft_living, df.price)


# Model

from scipy import stats
pred_price = stats.linregress(x=df.sqft_living, y=df.price)
pred_price

def predict(x):
    return pred_price.slope * x + pred_price.intercept
predict(5400)

# How to evaluate predict(5400)?
# Going to go back to Dataset and check the price

df[df.sqft_living == 5400]

# We can ask more questions to customers
# zia@thedevmasters.com
# multi variable regression

# Python > R
```

Out[1]:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 21 columns):
id             21613 non-null int64
date           21613 non-null object
price          21613 non-null float64
bedrooms       21613 non-null int64
bathrooms      21613 non-null float64
sqft_living    21613 non-null int64
sqft_lot       21613 non-null int64
floors         21613 non-null float64
waterfront     21613 non-null int64
grade          21613 non-null int64
sqft_above     21613 non-null int64
sqft_basement  21613 non-null int64
yr_built       21613 non-null int64
```

In [15]:

```
df.head()
```

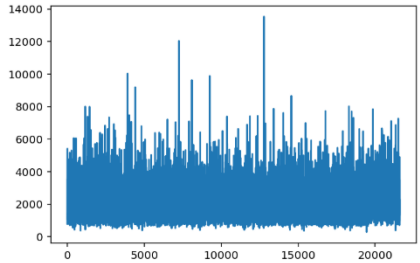
	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode	la
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...	7	1180	0	1955	0	98178	47.511
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	...	7	2170	400	1951	1991	98125	47.721
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...	6	770	0	1933	0	98028	47.737
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	...	7	1050	910	1965	0	98136	47.520
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	...	8	1680	0	1987	0	98074	47.616

5 rows × 21 columns

In [4]:
```
df.sqft_living.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f440ed44160>
```

Out[4]:



In [5]:
```
df[df.sqft_living > 8000]
```

Out[5]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode
1164	1247600105	20141020T000000	5110800.0	5	5.25	8010	45517	2.0	1	4	...	12	5990	2020	1999	0	98033
3914	9808700762	20140611T000000	7062500.0	5	4.50	10040	37325	2.0	1	2	...	11	7680	2360	1940	2001	98004
4411	2470100110	20140804T000000	5570000.0	5	5.75	9200	35069	2.0	0	0	...	13	6200	3000	2001	0	98039
7252	6762700020	20141013T000000	7700000.0	6	8.00	12050	27600	2.5	0	3	...	13	8570	3480	1910	1987	98102
8092	1924059029	20140617T000000	4668000.0	5	6.75	9640	13068	1.0	1	4	...	12	4820	4820	1983	2009	98040
9254	9208900037	20140919T000000	6885000.0	6	7.75	9890	31374	2.0	0	4	...	13	8860	1030	2001	0	98039
12777	1225069038	20140505T000000	2280000.0	7	8.00	13540	307752	3.0	0	4	...	12	9410	4130	1999	0	98053
14556	2303900035	20140611T000000	2888000.0	5	6.25	8670	64033	2.0	0	4	...	13	6120	2550	1965	2003	98177
18302	6072800246	20140702T000000	3300000.0	5	6.25	8020	21738	2.0	0	0	...	11	8020	0	2001	0	98006

9 rows × 21 columns

In [6]:
```
plt.hist(df.price, bins = 200)
```

Out[6]:

```
(array([5.200e+01, 1.500e+02, 3.940e+02, 9.400e+02, 1.485e+03, 1.609e+03,
        1.704e+03, 1.578e+03, 1.514e+03, 1.615e+03, 1.172e+03, 1.201e+03,
        1.165e+03, 9.120e+02, 7.570e+02, 7.130e+02, 6.220e+02, 5.790e+02,
        4.360e+02, 4.230e+02, 3.690e+02, 2.490e+02, 2.530e+02, 1.710e+02,
        1.410e+02, 1.220e+02, 1.000e+02, 6.500e+01, 8.200e+01, 8.700e+01,
        8.700e+01, 5.600e+01, 8.300e+01, 5.000e+01, 6.500e+01, 2.800e+01,
        4.400e+01, 3.500e+01, 2.500e+01, 3.600e+01, 3.600e+01, 2.800e+01,
        4.000e+01, 2.200e+01, 1.400e+01, 2.300e+01, 1.500e+01, 2.600e+01,
        1.100e+01, 1.800e+01, 1.400e+01, 5.000e+00, 3.000e+00, 4.000e+00,
        9.000e+00, 1.400e+01, 6.000e+00, 1.200e+01, 1.100e+01, 5.000e+00,
        1.300e+01, 2.000e+00, 8.000e+00, 9.000e+00, 7.000e+00, 5.000e+00,
        2.000e+00, 3.000e+00, 6.000e+00, 2.000e+00, 3.000e+00, 1.000e+00,
        1.000e+00, 4.000e+00, 5.000e+00, 5.000e+00, 7.000e+00, 0.000e+00,
        3.000e+00, 4.000e+00, 0.000e+00, 5.000e+00, 1.000e+00, 0.000e+00,
        5.000e+00, 1.000e+00, 0.000e+00, 4.000e+00, 0.000e+00, 0.000e+00,
        0.000e+00, 1.000e+00, 1.000e+00, 4.000e+00, 0.000e+00, 1.000e+00,
        0.000e+00, 2.000e+00, 0.000e+00, 1.000e+00, 0.000e+00, 0.000e+00,
        1.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00,
```

In [8]:
```
plt.scatter(df.sqft_living, df.price)
plt.title('plt.scatter', fontsize=15)
plt.show()
```

Out[8]: