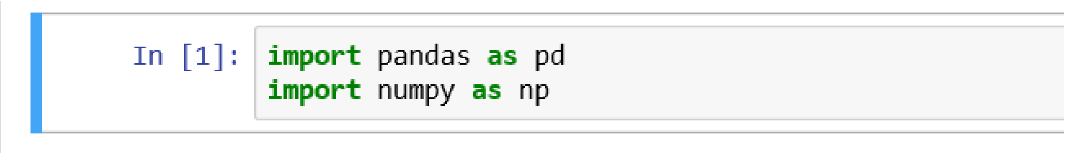
REAL ESTATE PRICE PREDICTION OF PUNE CITY

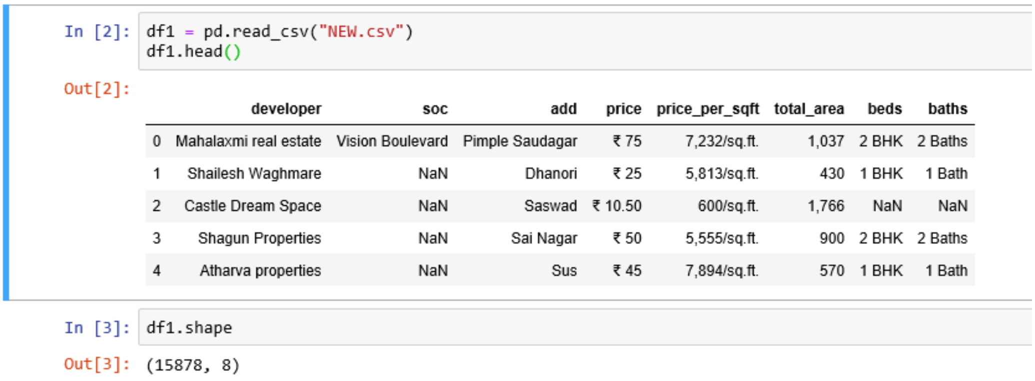
- **Data Extraction:**

We extracted our dataset from 99acres.com and magicbricks.com which are property selling websites by using Data Toolbar software. Firstly, we will study the whole dataset after extracting. And after that we will apply various data analysis tools according to that.

First of all, we import our dataset in our Jupyter Notebook software. And for that, we import various libraries which we going to use.



```
In [1]: import pandas as pd
        import numpy as np
```



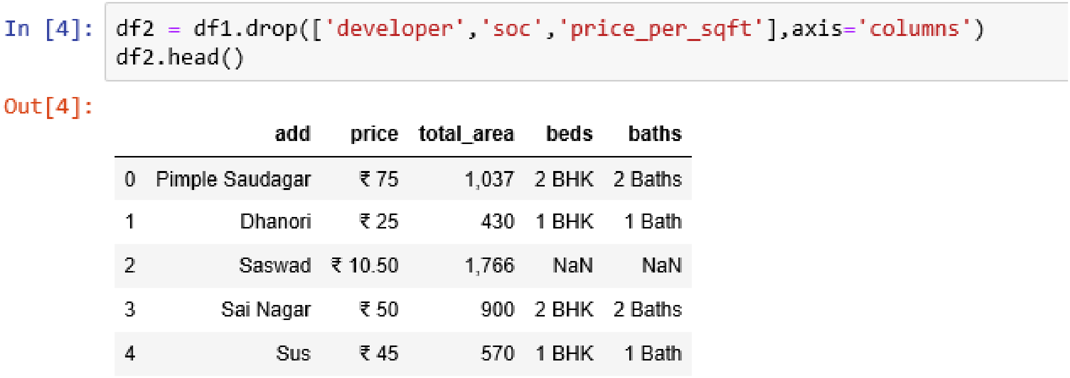
```
In [2]: df1 = pd.read_csv("NEW.csv")
        df1.head()
Out[2]:
```

	developer	soc	add	price	price_per_sqft	total_area	beds	baths
0	Mahalaxmi real estate	Vision Boulevard	Pimple Saudagar	₹ 75	7,232/sq.ft.	1,037	2 BHK	2 Baths
1	Shailesh Waghmare	NaN	Dhanori	₹ 25	5,813/sq.ft.	430	1 BHK	1 Bath
2	Castle Dream Space	NaN	Saswad	₹ 10.50	600/sq.ft.	1,766	NaN	NaN
3	Shagun Properties	NaN	Sai Nagar	₹ 50	5,555/sq.ft.	900	2 BHK	2 Baths
4	Atharva properties	NaN	Sus	₹ 45	7,894/sq.ft.	570	1 BHK	1 Bath

```
In [3]: df1.shape
Out[3]: (15878, 8)
```

We can see that size of the dataset is around 15858 rows and 8 columns.

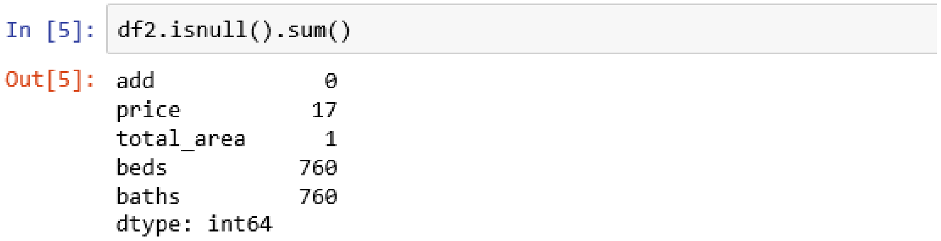
Here we drop columns which are not needful.



```
In [4]: df2 = df1.drop(['developer','soc','price_per_sqft'],axis='columns')
        df2.head()
Out[4]:
```

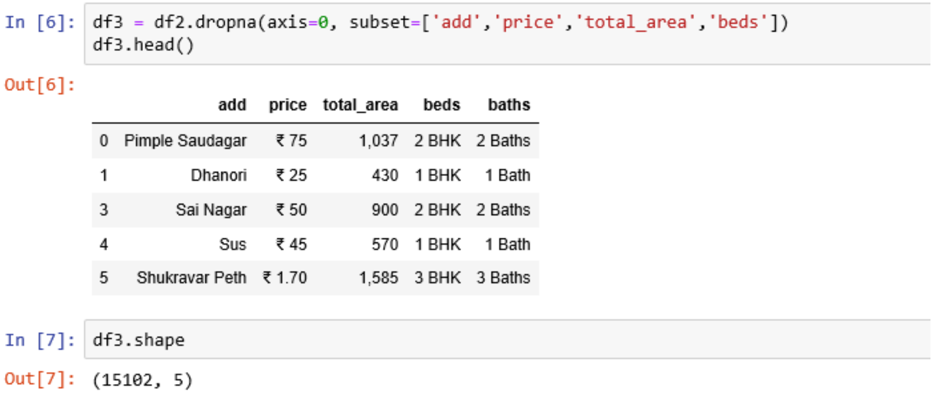
	add	price	total_area	beds	baths
0	Pimple Saudagar	₹ 75	1,037	2 BHK	2 Baths
1	Dhanori	₹ 25	430	1 BHK	1 Bath
2	Saswad	₹ 10.50	1,766	NaN	NaN
3	Sai Nagar	₹ 50	900	2 BHK	2 Baths
4	Sus	₹ 45	570	1 BHK	1 Bath

- **Data Cleaning:**



```
In [5]: df2.isnull().sum()

Out[5]: add             0
        price          17
        total_area      1
        beds          760
        baths         760
        dtype: int64
```

Here we will remove rows which are null.



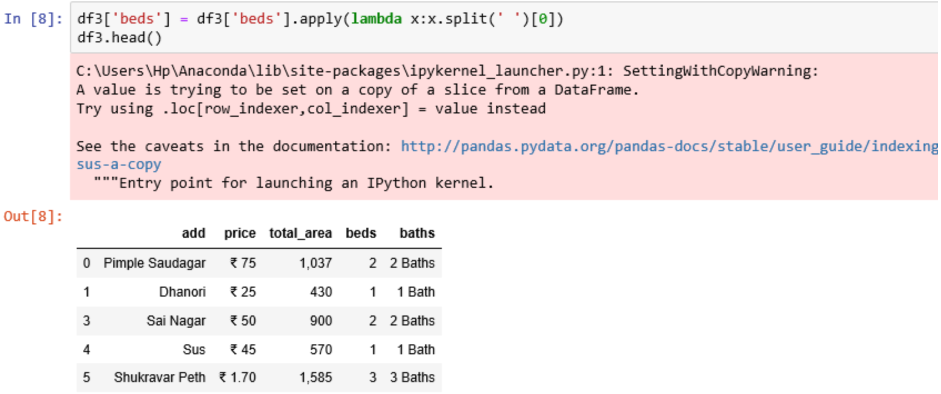
```
In [6]: df3 = df2.dropna(axis=0, subset=['add','price','total_area','beds'])
        df3.head()
```

Out[6]:

	add	price	total_area	beds	baths
0	Pimple Saudagar	₹ 75	1,037	2 BHK	2 Baths
1	Dhanori	₹ 25	430	1 BHK	1 Bath
3	Sai Nagar	₹ 50	900	2 BHK	2 Baths
4	Sus	₹ 45	570	1 BHK	1 Bath
5	Shukravar Peth	₹ 1.70	1,585	3 BHK	3 Baths

```
In [7]: df3.shape

Out[7]: (15102, 5)
```

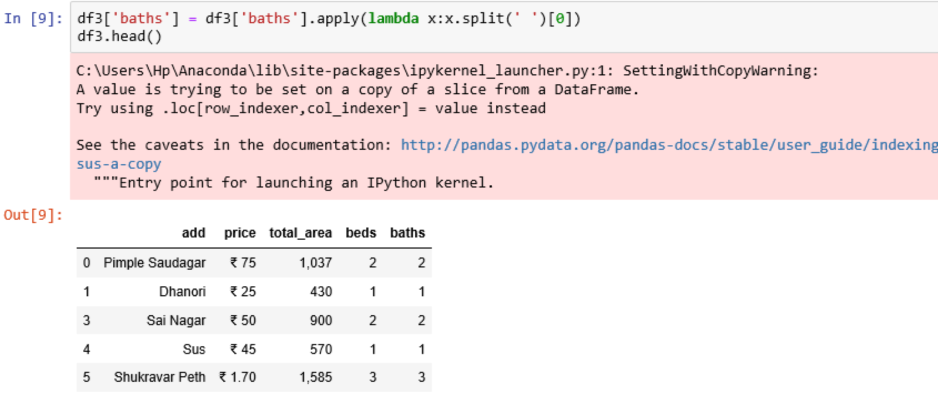
We make our data numerical except “add” column.



```
In [8]: df3['beds'] = df3['beds'].apply(lambda x:x.split(' ')[0])
        df3.head()

C:\Users\Hp\Anaconda\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
sus-a-copy
  """Entry point for launching an IPython kernel.
```

Out[8]:

	add	price	total_area	beds	baths
0	Pimple Saudagar	₹ 75	1,037	2	2 Baths
1	Dhanori	₹ 25	430	1	1 Bath
3	Sai Nagar	₹ 50	900	2	2 Baths
4	Sus	₹ 45	570	1	1 Bath
5	Shukravar Peth	₹ 1.70	1,585	3	3 Baths

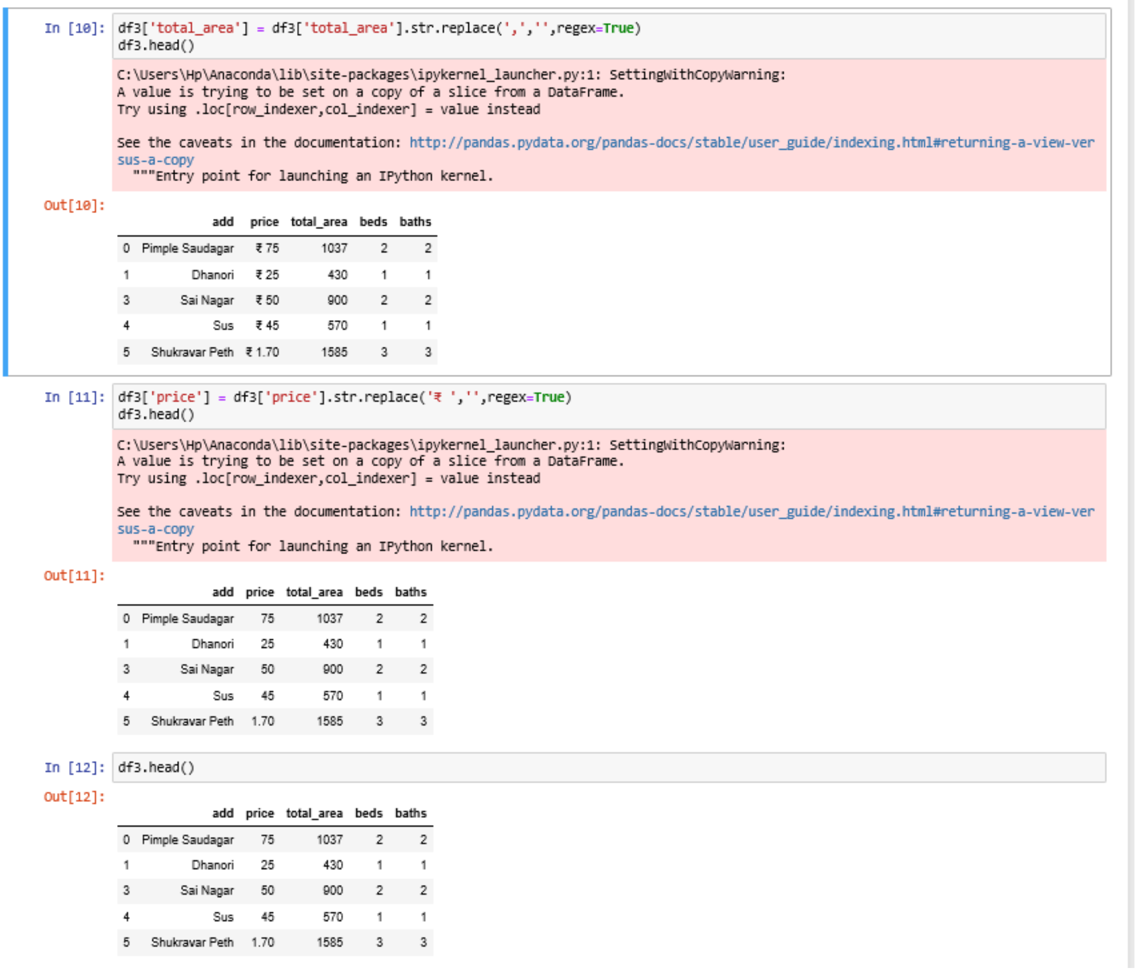


```
In [9]: df3['baths'] = df3['baths'].apply(lambda x:x.split(' ')[0])
        df3.head()

C:\Users\Hp\Anaconda\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
sus-a-copy
  """Entry point for launching an IPython kernel.
```

Out[9]:

	add	price	total_area	beds	baths
0	Pimple Saudagar	₹ 75	1,037	2	2
1	Dhanori	₹ 25	430	1	1
3	Sai Nagar	₹ 50	900	2	2
4	Sus	₹ 45	570	1	1
5	Shukravar Peth	₹ 1.70	1,585	3	3

Now we replace ',' from "total_area" and '₹' from "price" column by replace function.



```
In [10]: df3['total_area'] = df3['total_area'].str.replace(',','',regex=True)
         df3.head()
```

```
C:\Users\Hp\Anaconda\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy
  """Entry point for launching an IPython kernel.
```

Out[10]:

	add	price	total_area	beds	baths
0	Pimple Saudagar	₹ 75	1037	2	2
1	Dhanori	₹ 25	430	1	1
3	Sai Nagar	₹ 50	900	2	2
4	Sus	₹ 45	570	1	1
5	Shukravar Peth	₹ 1.70	1585	3	3

```
In [11]: df3['price'] = df3['price'].str.replace('₹ ','',regex=True)
         df3.head()
```

```
C:\Users\Hp\Anaconda\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy
  """Entry point for launching an IPython kernel.
```

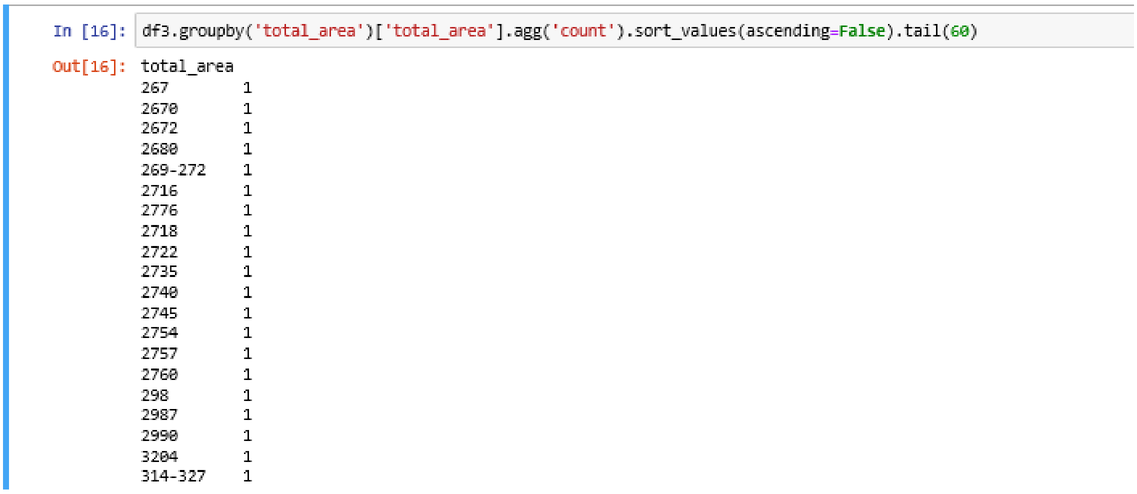
Out[11]:

	add	price	total_area	beds	baths
0	Pimple Saudagar	75	1037	2	2
1	Dhanori	25	430	1	1
3	Sai Nagar	50	900	2	2
4	Sus	45	570	1	1
5	Shukravar Peth	1.70	1585	3	3

```
In [12]: df3.head()
```

Out[12]:

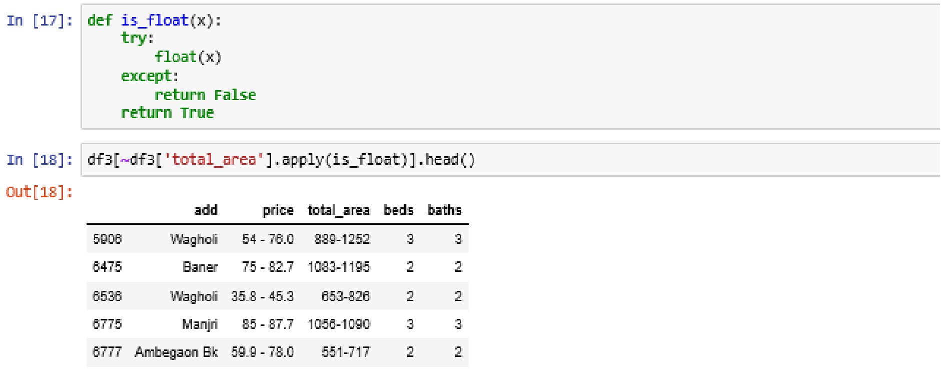
	add	price	total_area	beds	baths
0	Pimple Saudagar	75	1037	2	2
1	Dhanori	25	430	1	1
3	Sai Nagar	50	900	2	2
4	Sus	45	570	1	1
5	Shukravar Peth	1.70	1585	3	3



```
In [16]: df3.groupby('total_area')['total_area'].agg('count').sort_values(ascending=False).tail(60)

Out[16]: total_area
         267        1
         2670       1
         2672       1
         2680       1
         269-272    1
         2716       1
         2776       1
         2718       1
         2722       1
         2735       1
         2740       1
         2745       1
         2754       1
         2757       1
         2760       1
         298        1
         2987       1
         2990       1
         3204       1
         314-327    1
```

If we group our “total_area” column then we can see that above there are some values are in range. So, we will take mean of those values.

- **Feature Engineering**



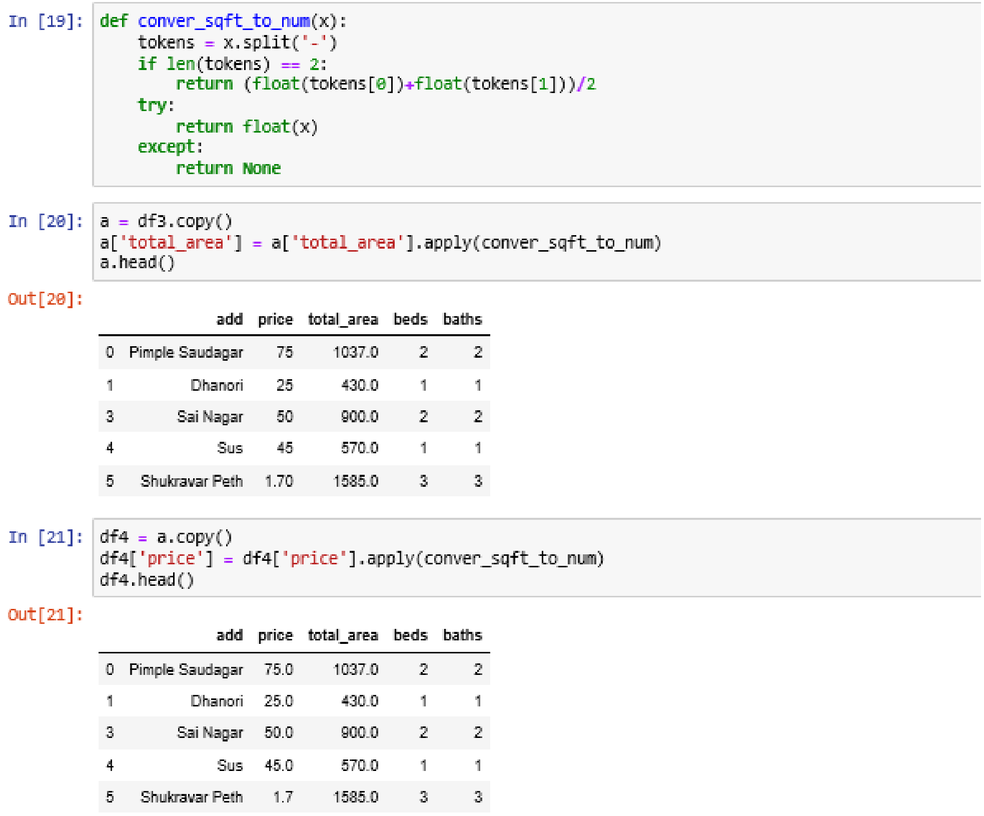
```
In [17]: def is_float(x):
             try:
                 float(x)
             except:
                 return False
             return True
```

```
In [18]: df3[~df3['total_area'].apply(is_float)].head()
```

Out[18]:

	add	price	total_area	beds	baths
5906	Wagholi	54 - 76.0	889-1252	3	3
6475	Baner	75 - 82.7	1083-1195	2	2
6536	Wagholi	35.8 - 45.3	653-826	2	2
6775	Manjri	85 - 87.7	1056-1090	3	3
6777	Ambegaon Bk	59.9 - 78.0	551-717	2	2

We will take mean of those values.



```
In [19]: def conver_sqft_to_num(x):
             tokens = x.split('-')
             if len(tokens) == 2:
                 return (float(tokens[0])+float(tokens[1]))/2
             try:
                 return float(x)
             except:
                 return None
```

```
In [20]: a = df3.copy()
         a['total_area'] = a['total_area'].apply(conver_sqft_to_num)
         a.head()
```

Out[20]:

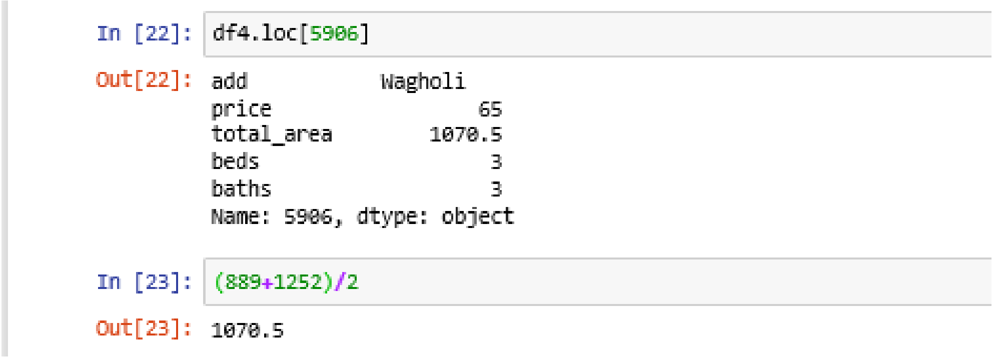
	add	price	total_area	beds	baths
0	Pimple Saudagar	75	1037.0	2	2
1	Dhanori	25	430.0	1	1
3	Sai Nagar	50	900.0	2	2
4	Sus	45	570.0	1	1
5	Shukravar Peth	1.70	1585.0	3	3

```
In [21]: df4 = a.copy()
         df4['price'] = df4['price'].apply(conver_sqft_to_num)
         df4.head()
```

Out[21]:

	add	price	total_area	beds	baths
0	Pimple Saudagar	75.0	1037.0	2	2
1	Dhanori	25.0	430.0	1	1
3	Sai Nagar	50.0	900.0	2	2
4	Sus	45.0	570.0	1	1
5	Shukravar Peth	1.7	1585.0	3	3

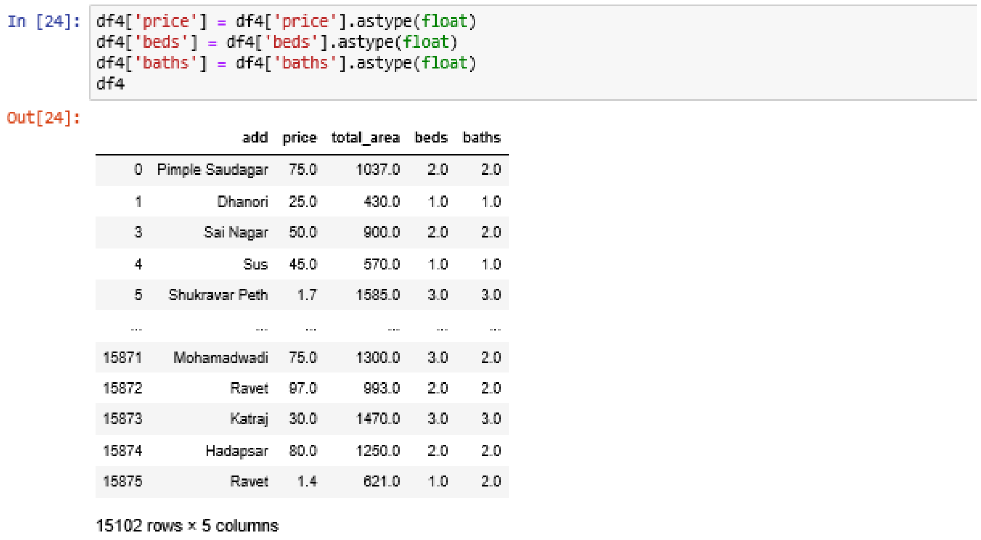
We can see that following values were in range are replaced by mean of index 5906.



```
In [22]: df4.loc[5906]

Out[22]: add           Wagholi
         price              65
         total_area     1070.5
         beds                3
         baths               3
         Name: 5906, dtype: object
```

```
In [23]: (889+1252)/2

Out[23]: 1070.5
```

Here we convert our data in float values for better visualization.



In [24]:

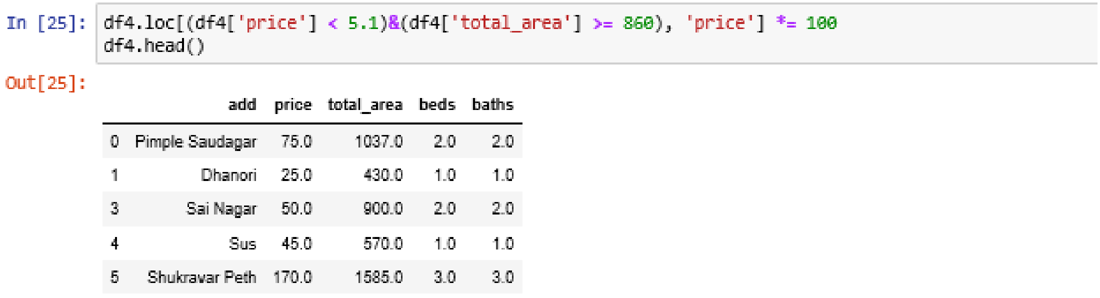
```
df4['price'] = df4['price'].astype(float)
df4['beds'] = df4['beds'].astype(float)
df4['baths'] = df4['baths'].astype(float)
df4
```

Out[24]:

	add	price	total_area	beds	baths
0	Pimple Saudagar	75.0	1037.0	2.0	2.0
1	Dhanori	25.0	430.0	1.0	1.0
3	Sai Nagar	50.0	900.0	2.0	2.0
4	Sus	45.0	570.0	1.0	1.0
5	Shukravar Peth	1.7	1585.0	3.0	3.0
...	...	...	...	...	...
15871	Mohamadwadi	75.0	1300.0	3.0	2.0
15872	Ravet	97.0	993.0	2.0	2.0
15873	Katraj	30.0	1470.0	3.0	3.0
15874	Hadapsar	80.0	1250.0	2.0	2.0
15875	Ravet	1.4	621.0	1.0	2.0

15102 rows × 5 columns

Price column: As we can see in price column values are like 1.7 and total_area is 1585.0 which is not deserving. So, we can say that some prices are in crores, so we convert them into lacs values.

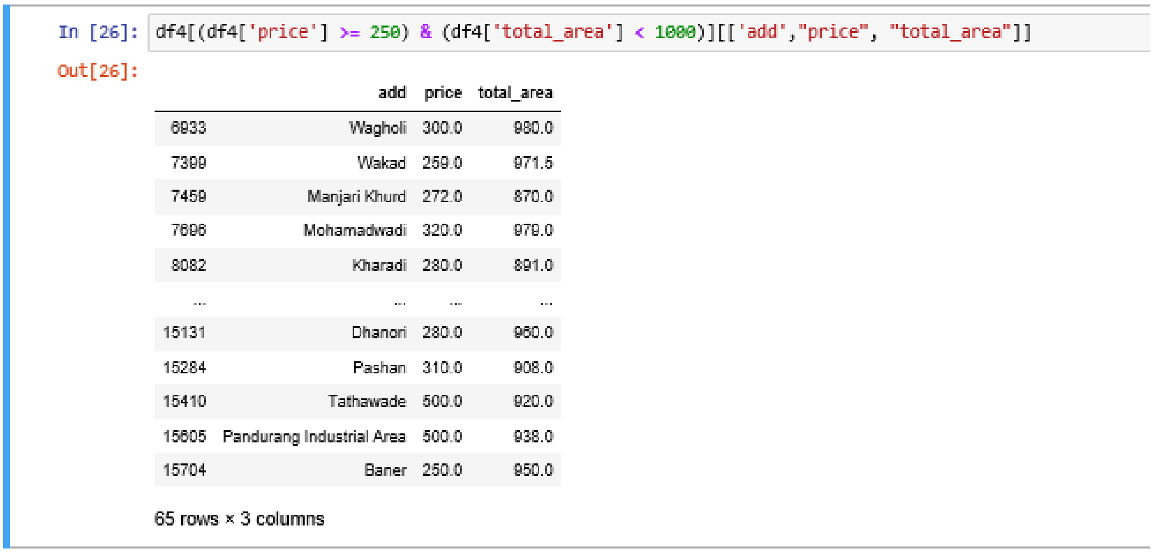


In [25]:

```
df4.loc[(df4['price'] < 5.1)&(df4['total_area'] >= 860), 'price'] *= 100
df4.head()
```

Out[25]:

	add	price	total_area	beds	baths
0	Pimple Saudagar	75.0	1037.0	2.0	2.0
1	Dhanori	25.0	430.0	1.0	1.0
3	Sai Nagar	50.0	900.0	2.0	2.0
4	Sus	45.0	570.0	1.0	1.0
5	Shukravar Peth	170.0	1585.0	3.0	3.0



In [26]:

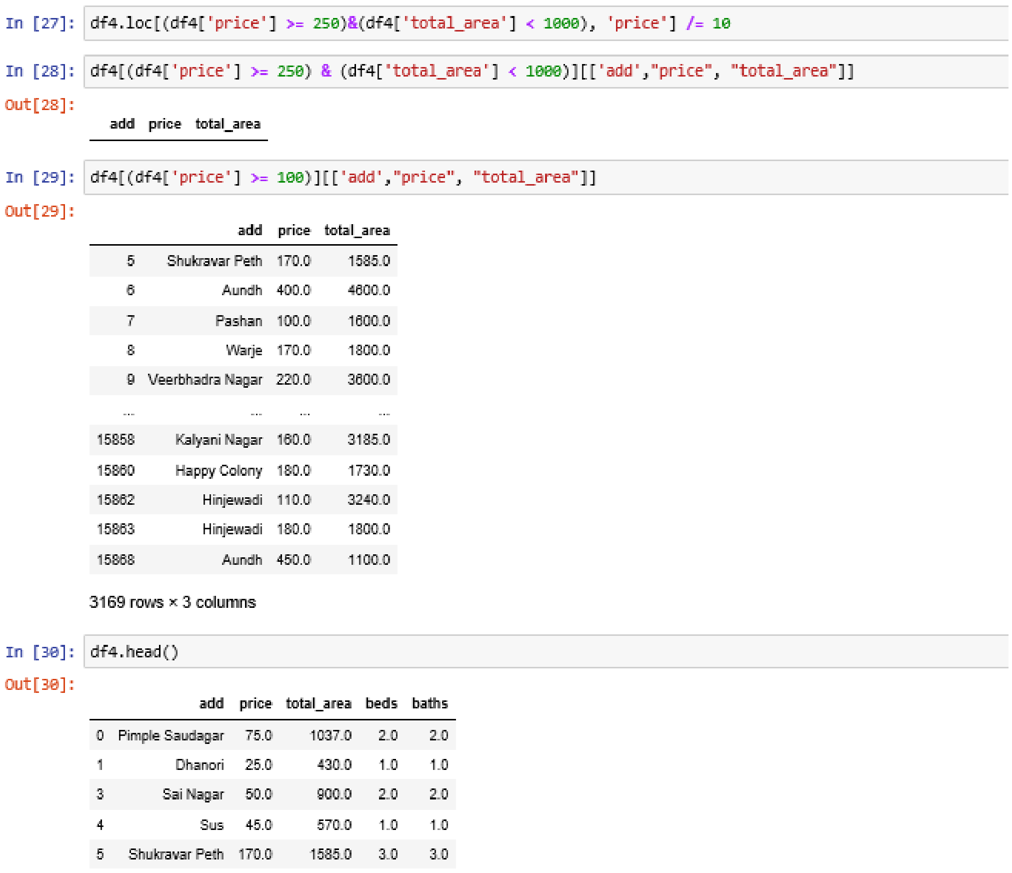
```
df4[(df4['price'] >= 250) & (df4['total_area'] < 1000)][['add',"price", "total_area"]]
```

Out[26]:

	add	price	total_area
6933	Wagholi	300.0	980.0
7399	Wakad	259.0	971.5
7459	Manjari Khurd	272.0	870.0
7696	Mohamadwadi	320.0	979.0
8082	Kharadi	280.0	891.0
...	...	...	...
15131	Dhanori	280.0	960.0
15284	Pashan	310.0	908.0
15410	Tathawade	500.0	920.0
15605	Pandurang Industrial Area	500.0	938.0
15704	Baner	250.0	950.0

65 rows × 3 columns

But in above table, we replaced prices where no need to replace. So, there we have to make as previous.



```
In [27]: df4.loc[(df4['price'] >= 250)&(df4['total_area'] < 1000), 'price'] /= 10
```

```
In [28]: df4[(df4['price'] >= 250) & (df4['total_area'] < 1000)][['add',"price", "total_area"]]
```

Out[28]:

add	price	total_area

```
In [29]: df4[(df4['price'] >= 100)][['add',"price", "total_area"]]
```

Out[29]:

	add	price	total_area
5	Shukravar Peth	170.0	1585.0
6	Aundh	400.0	4600.0
7	Pashan	100.0	1600.0
8	Warje	170.0	1800.0
9	Veerbhadra Nagar	220.0	3600.0
...	...	...	...
15858	Kalyani Nagar	160.0	3185.0
15860	Happy Colony	180.0	1730.0
15862	Hinjewadi	110.0	3240.0
15863	Hinjewadi	180.0	1800.0
15868	Aundh	450.0	1100.0

3169 rows × 3 columns

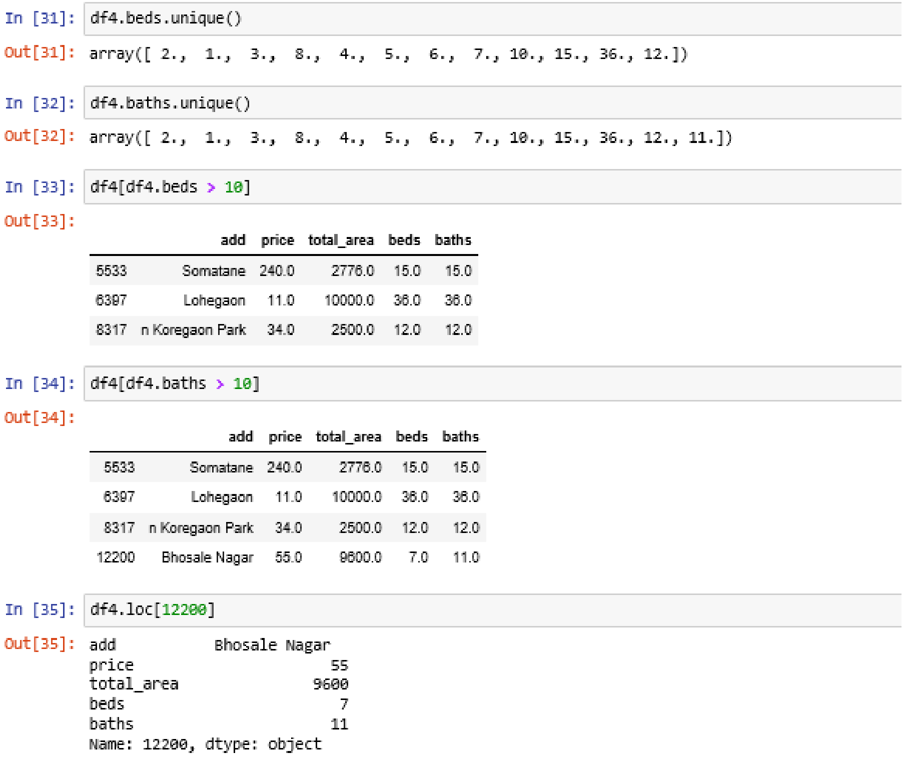
```
In [30]: df4.head()
```

Out[30]:

	add	price	total_area	beds	baths
0	Pimple Saudagar	75.0	1037.0	2.0	2.0
1	Dhanori	25.0	430.0	1.0	1.0
3	Sai Nagar	50.0	900.0	2.0	2.0
4	Sus	45.0	570.0	1.0	1.0
5	Shukravar Peth	170.0	1585.0	3.0	3.0

- **Feature Engineering**

Beds and baths column



```
In [31]: df4.beds.unique()
Out[31]: array([ 2.,  1.,  3.,  8.,  4.,  5.,  6.,  7., 10., 15., 36., 12.])
```

```
In [32]: df4.baths.unique()
Out[32]: array([ 2.,  1.,  3.,  8.,  4.,  5.,  6.,  7., 10., 15., 36., 12., 11.])
```

```
In [33]: df4[df4.beds > 10]
```

Out[33]:

	add	price	total_area	beds	baths
5533	Somatane	240.0	2776.0	15.0	15.0
6397	Lohegaon	11.0	10000.0	36.0	36.0
8317	n Koregaon Park	34.0	2500.0	12.0	12.0

```
In [34]: df4[df4.baths > 10]
```

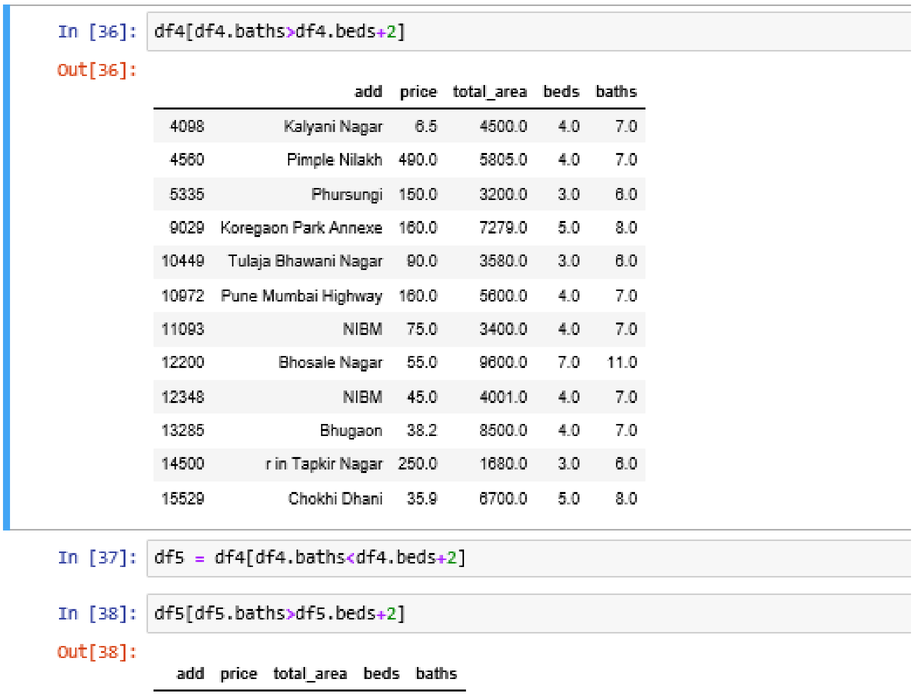
Out[34]:

	add	price	total_area	beds	baths
5533	Somatane	240.0	2776.0	15.0	15.0
6397	Lohegaon	11.0	10000.0	36.0	36.0
8317	n Koregaon Park	34.0	2500.0	12.0	12.0
12200	Bhosale Nagar	55.0	9600.0	7.0	11.0

```
In [35]: df4.loc[12200]
Out[35]: add           Bhosale Nagar
         price                    55
         total_area             9600
         beds                      7
         baths                    11
         Name: 12200, dtype: object
```

As we can see that, bathrooms have much more than bedrooms.

If you have 4-bedroom home and even if you have bathroom in all 4 rooms plus one guest bathroom, you will have total bath = total bed + 1 max. Anything above that is an outlier or a data error and can be removed.



```
In [36]: df4[df4.baths>df4.beds+2]
```

Out[36]:

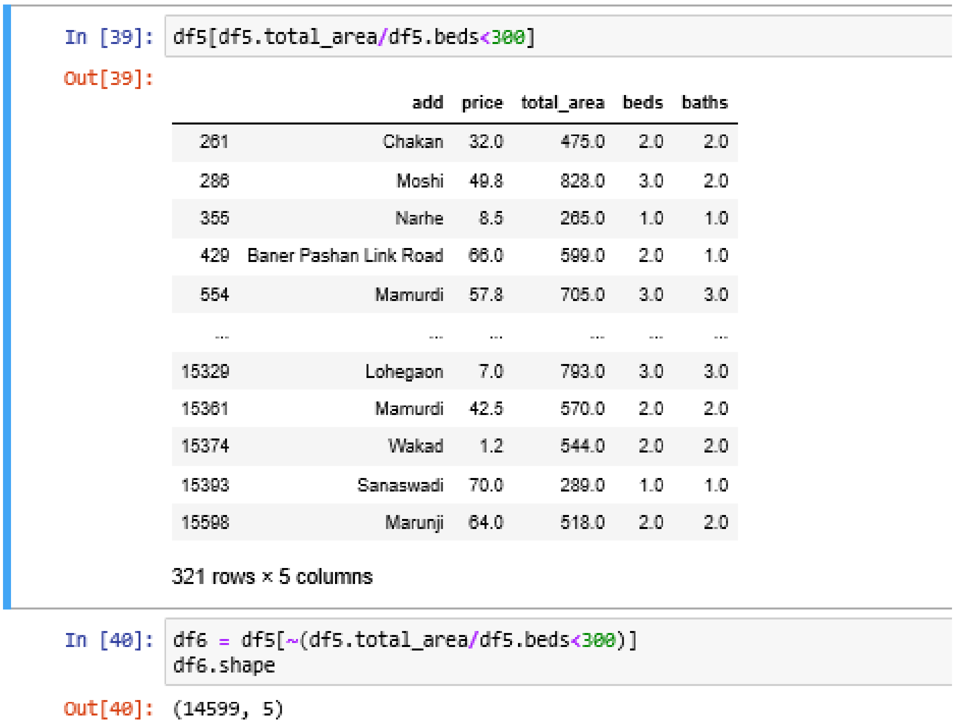
	add	price	total_area	beds	baths
4098	Kalyani Nagar	6.5	4500.0	4.0	7.0
4560	Pimple Nilakh	490.0	5805.0	4.0	7.0
5335	Phursungi	150.0	3200.0	3.0	6.0
9029	Koregaon Park Annexe	160.0	7279.0	5.0	8.0
10449	Tulaja Bhawani Nagar	90.0	3580.0	3.0	6.0
10972	Pune Mumbai Highway	160.0	5600.0	4.0	7.0
11093	NIBM	75.0	3400.0	4.0	7.0
12200	Bhosale Nagar	55.0	9600.0	7.0	11.0
12348	NIBM	45.0	4001.0	4.0	7.0
13285	Bhugaon	38.2	8500.0	4.0	7.0
14500	r in Tapkir Nagar	250.0	1680.0	3.0	6.0
15529	Chokhi Dhani	35.9	6700.0	5.0	8.0

```
In [37]: df5 = df4[df4.baths<df4.beds+2]

In [38]: df5[df5.baths>df5.beds+2]
```

Out[38]:

add	price	total_area	beds	baths

Now if we assume that 1 bedroom approximately cover max to max 300 sqft. We will remove such type of outliers where has bedroom is lesser than 300 sqft.



```
In [39]: df5[df5.total_area/df5.beds<300]
```

Out[39]:

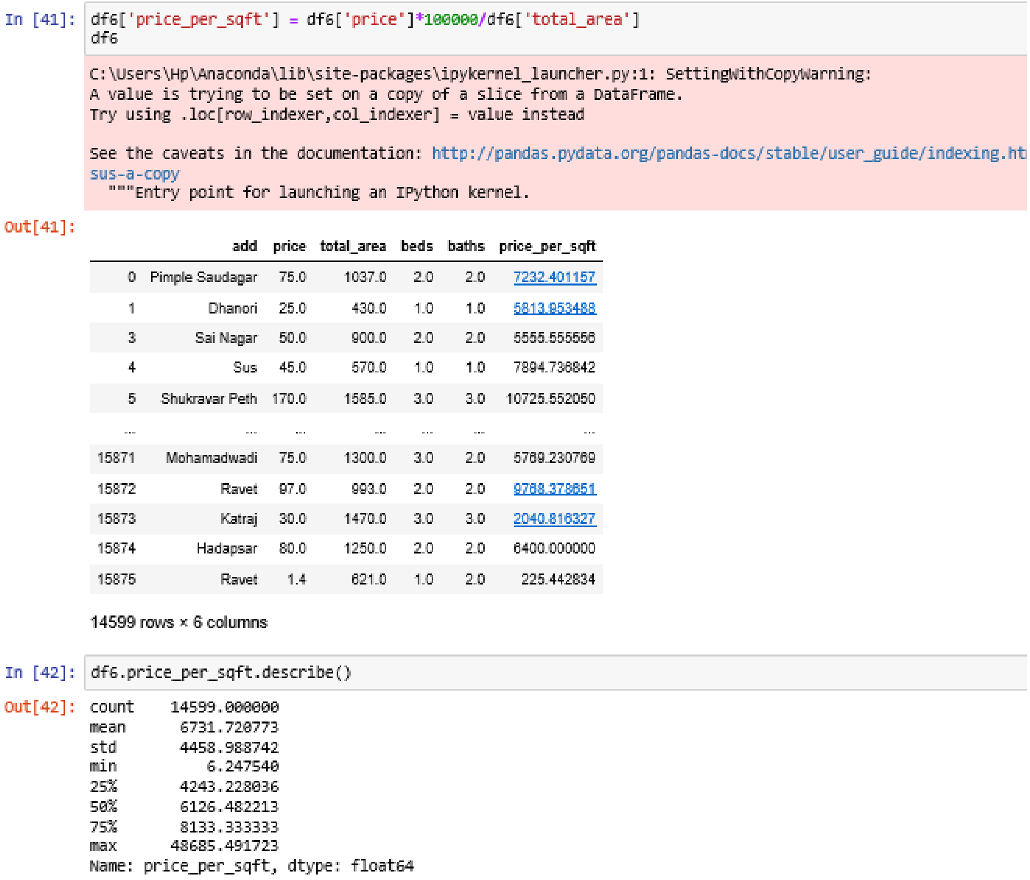
	add	price	total_area	beds	baths
261	Chakan	32.0	475.0	2.0	2.0
286	Moshi	49.8	828.0	3.0	2.0
355	Narhe	8.5	265.0	1.0	1.0
429	Baner Pashan Link Road	66.0	599.0	2.0	1.0
554	Mamurdi	57.8	705.0	3.0	3.0
...	...	...	...	...	...
15329	Lohegaon	7.0	793.0	3.0	3.0
15361	Mamurdi	42.5	570.0	2.0	2.0
15374	Wakad	1.2	544.0	2.0	2.0
15393	Sanaswadi	70.0	289.0	1.0	1.0
15598	Marunji	64.0	518.0	2.0	2.0

321 rows × 5 columns

```
In [40]: df6 = df5[~(df5.total_area/df5.beds<300)]
         df6.shape

Out[40]: (14599, 5)
```

- **Feature Engineering**

Price Per sqft column



```
In [41]: df6['price_per_sqft'] = df6['price']*100000/df6['total_area']
         df6
```

```
C:\Users\Hp\Anaconda\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.ht
sus-a-copy
  """Entry point for launching an IPython kernel.
```

Out[41]:

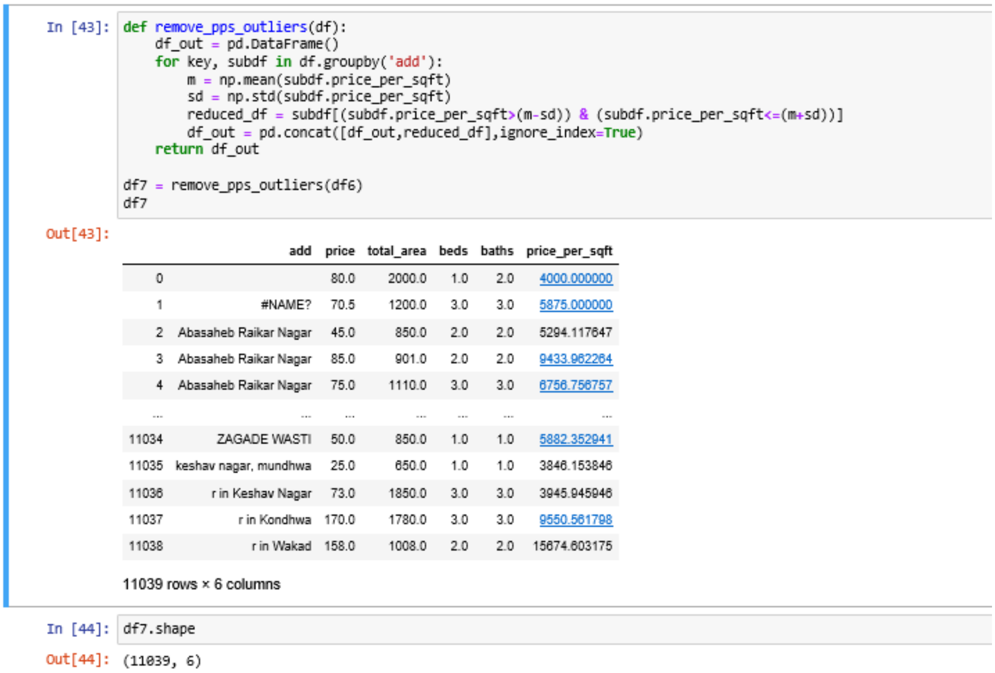
	add	price	total_area	beds	baths	price_per_sqft
0	Pimple Saudagar	75.0	1037.0	2.0	2.0	7232.401157
1	Dhanori	25.0	430.0	1.0	1.0	5813.953488
3	Sai Nagar	50.0	900.0	2.0	2.0	5555.555556
4	Sus	45.0	570.0	1.0	1.0	7894.736842
5	Shukravar Peth	170.0	1585.0	3.0	3.0	10725.552050
...	...	...	...	...	...	...
15871	Mohamadwadi	75.0	1300.0	3.0	2.0	5769.230769
15872	Ravet	97.0	993.0	2.0	2.0	9768.378651
15873	Katraj	30.0	1470.0	3.0	3.0	2040.816327
15874	Hadapsar	80.0	1250.0	2.0	2.0	6400.000000
15875	Ravet	1.4	621.0	1.0	2.0	225.442834

14599 rows × 6 columns

```
In [42]: df6.price_per_sqft.describe()
```

```
Out[42]: count    14599.000000
         mean      6731.720773
         std       4458.988742
         min          6.247540
         25%       4243.228036
         50%       6126.482213
         75%       8133.333333
         max      48685.491723
         Name: price_per_sqft, dtype: float64
```

As we can see, price per sqft column has 6 Rs/sqft min and 48685 Rs/sqft max, which is an outlier. We will remove such type of outliers.



```
In [43]: def remove_pps_outliers(df):
             df_out = pd.DataFrame()
             for key, subdf in df.groupby('add'):
                 m = np.mean(subdf.price_per_sqft)
                 sd = np.std(subdf.price_per_sqft)
                 reduced_df = subdf[(subdf.price_per_sqft>(m-sd)) & (subdf.price_per_sqft<=(m+sd))]
                 df_out = pd.concat([df_out,reduced_df],ignore_index=True)
             return df_out

         df7 = remove_pps_outliers(df6)
         df7
```

Out[43]:

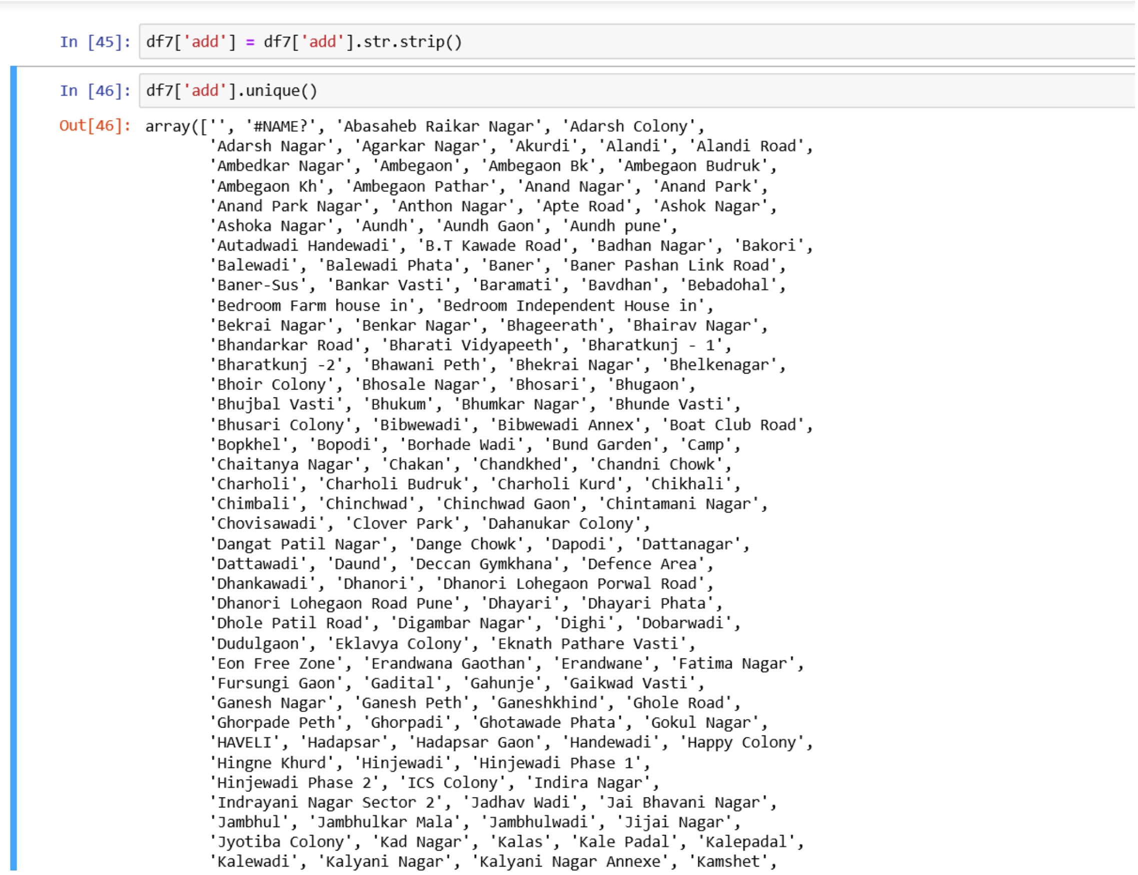
	add	price	total_area	beds	baths	price_per_sqft
0		80.0	2000.0	1.0	2.0	4000.000000
1	#NAME?	70.5	1200.0	3.0	3.0	5875.000000
2	Abasaheb Raikar Nagar	45.0	850.0	2.0	2.0	5294.117647
3	Abasaheb Raikar Nagar	85.0	901.0	2.0	2.0	9433.962264
4	Abasaheb Raikar Nagar	75.0	1110.0	3.0	3.0	6756.756757
...	...	...	...	...	...	...
11034	ZAGADE WASTI	50.0	850.0	1.0	1.0	5882.352941
11035	keshav nagar, mundhwa	25.0	650.0	1.0	1.0	3846.153846
11036	r in Keshav Nagar	73.0	1850.0	3.0	3.0	3945.945946
11037	r in Kondhwa	170.0	1780.0	3.0	3.0	9550.561798
11038	r in Wakad	158.0	1008.0	2.0	2.0	15674.603175

11039 rows × 6 columns

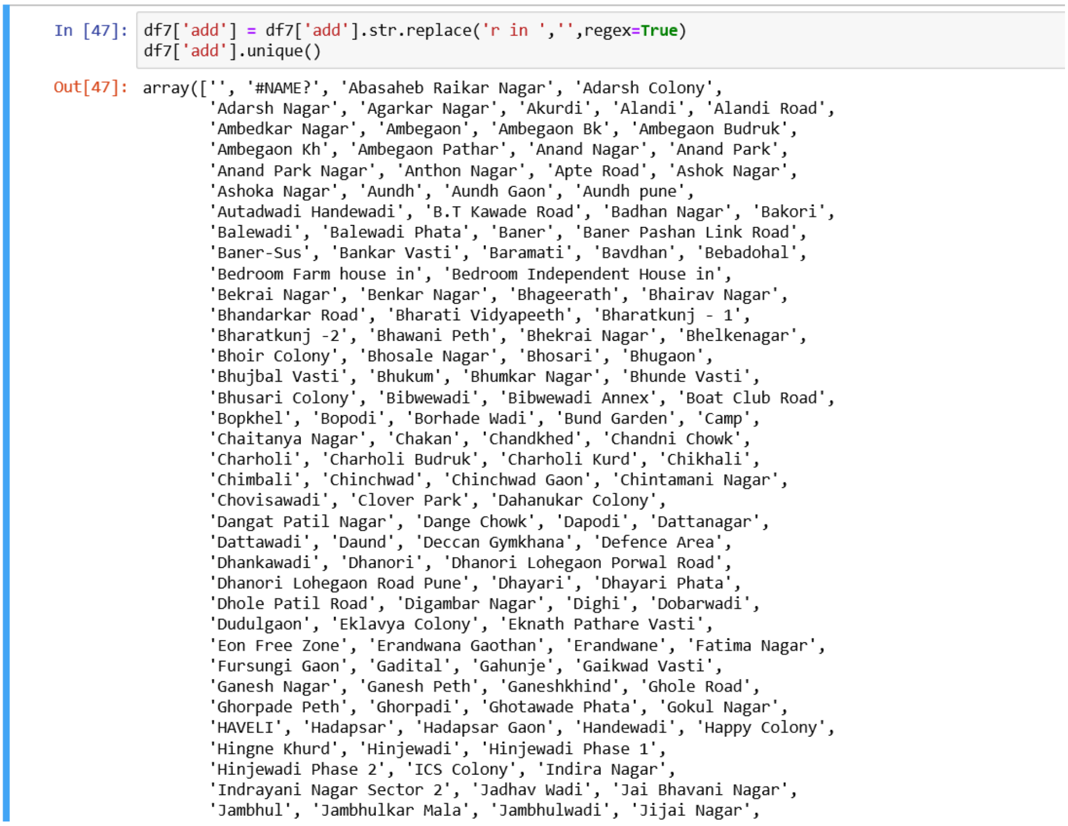
```
In [44]: df7.shape

Out[44]: (11039, 6)
```

- **Feature Engineering**

Add column



```
In [45]: df7['add'] = df7['add'].str.strip()

In [46]: df7['add'].unique()

Out[46]: array(['', '#NAME?', 'Abasaheb Raikar Nagar', 'Adarsh Colony',
                'Adarsh Nagar', 'Agarkar Nagar', 'Akurdi', 'Alandi', 'Alandi Road',
                'Ambedkar Nagar', 'Ambegaon', 'Ambegaon Bk', 'Ambegaon Budruk',
                'Ambegaon Kh', 'Ambegaon Pathar', 'Anand Nagar', 'Anand Park',
                'Anand Park Nagar', 'Anthon Nagar', 'Apte Road', 'Ashok Nagar',
                'Ashoka Nagar', 'Aundh', 'Aundh Gaon', 'Aundh pune',
                'Autadwadi Handewadi', 'B.T Kawade Road', 'Badhan Nagar', 'Bakori',
                'Balewadi', 'Balewadi Phata', 'Baner', 'Baner Pashan Link Road',
                'Baner-Sus', 'Bankar Vasti', 'Baramati', 'Bavdhan', 'Bebadohal',
                'Bedroom Farm house in', 'Bedroom Independent House in',
                'Bekrai Nagar', 'Benkar Nagar', 'Bhageerath', 'Bhairav Nagar',
                'Bhandarkar Road', 'Bharati Vidyapeeth', 'Bharatkunj - 1',
                'Bharatkunj -2', 'Bhawani Peth', 'Bhekrai Nagar', 'Bhelkenagar',
                'Bhoir Colony', 'Bhosale Nagar', 'Bhosari', 'Bhugaon',
                'Bhujbal Vasti', 'Bhukum', 'Bhumkar Nagar', 'Bhunde Vasti',
                'Bhusari Colony', 'Bibwewadi', 'Bibwewadi Annex', 'Boat Club Road',
                'Bopkhel', 'Bopodi', 'Borhade Wadi', 'Bund Garden', 'Camp',
                'Chaitanya Nagar', 'Chakan', 'Chandkhed', 'Chandni Chowk',
                'Charholi', 'Charholi Budruk', 'Charholi Kurd', 'Chikhali',
                'Chimbali', 'Chinchwad', 'Chinchwad Gaon', 'Chintamani Nagar',
                'Chovisawadi', 'Clover Park', 'Dahanukar Colony',
                'Dangat Patil Nagar', 'Dange Chowk', 'Dapodi', 'Dattanagar',
                'Dattawadi', 'Daund', 'Deccan Gymkhana', 'Defence Area',
                'Dhankawadi', 'Dhanori', 'Dhanori Lohegaon Porwal Road',
                'Dhanori Lohegaon Road Pune', 'Dhayari', 'Dhayari Phata',
                'Dhole Patil Road', 'Digambar Nagar', 'Dighi', 'Dobarwadi',
                'Dudulgaon', 'Eklavya Colony', 'Eknath Pathare Vasti',
                'Eon Free Zone', 'Erandwana Gaothan', 'Erandwane', 'Fatima Nagar',
                'Fursungi Gaon', 'Gadital', 'Gahunje', 'Gaikwad Vasti',
                'Ganesh Nagar', 'Ganesh Peth', 'Ganeshkhind', 'Ghole Road',
                'Ghorpade Peth', 'Ghorpadi', 'Ghotawade Phata', 'Gokul Nagar',
                'HAVELI', 'Hadapsar', 'Hadapsar Gaon', 'Handewadi', 'Happy Colony',
                'Hingne Khurd', 'Hinjewadi', 'Hinjewadi Phase 1',
                'Hinjewadi Phase 2', 'ICS Colony', 'Indira Nagar',
                'Indrayani Nagar Sector 2', 'Jadhav Wadi', 'Jai Bhavani Nagar',
                'Jambhul', 'Jambhulkar Mala', 'Jambhulwadi', 'Jijai Nagar',
                'Jyotiba Colony', 'Kad Nagar', 'Kalas', 'Kale Padal', 'Kalepadal',
                'Kalewadi', 'Kalyani Nagar', 'Kalyani Nagar Annexe', 'Kamshet',
```



```
In [47]: df7['add'] = df7['add'].str.replace('r in ','',regex=True)
         df7['add'].unique()

Out[47]: array(['', '#NAME?', 'Abasaheb Raikar Nagar', 'Adarsh Colony',
                'Adarsh Nagar', 'Agarkar Nagar', 'Akurdi', 'Alandi', 'Alandi Road',
                'Ambedkar Nagar', 'Ambegaon', 'Ambegaon Bk', 'Ambegaon Budruk',
                'Ambegaon Kh', 'Ambegaon Pathar', 'Anand Nagar', 'Anand Park',
                'Anand Park Nagar', 'Anthon Nagar', 'Apte Road', 'Ashok Nagar',
                'Ashoka Nagar', 'Aundh', 'Aundh Gaon', 'Aundh pune',
                'Autadwadi Handewadi', 'B.T Kawade Road', 'Badhan Nagar', 'Bakori',
                'Balewadi', 'Balewadi Phata', 'Baner', 'Baner Pashan Link Road',
                'Baner-Sus', 'Bankar Vasti', 'Baramati', 'Bavdhan', 'Bebadohal',
                'Bedroom Farm house in', 'Bedroom Independent House in',
                'Bekrai Nagar', 'Benkar Nagar', 'Bhageerath', 'Bhairav Nagar',
                'Bhandarkar Road', 'Bharati Vidyapeeth', 'Bharatkunj - 1',
                'Bharatkunj -2', 'Bhawani Peth', 'Bhekrai Nagar', 'Bhelkenagar',
                'Bhoir Colony', 'Bhosale Nagar', 'Bhosari', 'Bhugaon',
                'Bhujbal Vasti', 'Bhukum', 'Bhumkar Nagar', 'Bhunde Vasti',
                'Bhusari Colony', 'Bibwewadi', 'Bibwewadi Annex', 'Boat Club Road',
                'Bopkhel', 'Bopodi', 'Borhade Wadi', 'Bund Garden', 'Camp',
                'Chaitanya Nagar', 'Chakan', 'Chandkhed', 'Chandni Chowk',
                'Charholi', 'Charholi Budruk', 'Charholi Kurd', 'Chikhali',
                'Chimbali', 'Chinchwad', 'Chinchwad Gaon', 'Chintamani Nagar',
                'Chovisawadi', 'Clover Park', 'Dahanukar Colony',
                'Dangat Patil Nagar', 'Dange Chowk', 'Dapodi', 'Dattanagar',
                'Dattawadi', 'Daund', 'Deccan Gymkhana', 'Defence Area',
                'Dhankawadi', 'Dhanori', 'Dhanori Lohegaon Porwal Road',
                'Dhanori Lohegaon Road Pune', 'Dhayari', 'Dhayari Phata',
                'Dhole Patil Road', 'Digambar Nagar', 'Dighi', 'Dobarwadi',
                'Dudulgaon', 'Eklavya Colony', 'Eknath Pathare Vasti',
                'Eon Free Zone', 'Erandwana Gaothan', 'Erandwane', 'Fatima Nagar',
                'Fursungi Gaon', 'Gadital', 'Gahunje', 'Gaikwad Vasti',
                'Ganesh Nagar', 'Ganesh Peth', 'Ganeshkhind', 'Ghole Road',
                'Ghorpade Peth', 'Ghorpadi', 'Ghotawade Phata', 'Gokul Nagar',
                'HAVELI', 'Hadapsar', 'Hadapsar Gaon', 'Handewadi', 'Happy Colony',
                'Hingne Khurd', 'Hinjewadi', 'Hinjewadi Phase 1',
                'Hinjewadi Phase 2', 'ICS Colony', 'Indira Nagar',
                'Indrayani Nagar Sector 2', 'Jadhav Wadi', 'Jai Bhavani Nagar',
                'Jambhul', 'Jambhulkar Mala', 'Jambhulwadi', 'Jijai Nagar',
```

Any location having less than 10 data points should be tagged as "other" location. This way number of categories can be reduced by huge amount. Later on, when we do one hot encoding, it will help us with having fewer dummy columns.

In [48]:
```
location_stats = df7.groupby('add')['add'].agg('count').sort_values(ascending=False)
location_stats
```

Out[48]:
```
add
Baner            645
Wakad            567
Kharadi          456
Wagholi          406
NIBM             402
                ...
Mukund Nagar       1
Nakhate Vasti      1
Nana Peth          1
Navi Peth          1
                   1
Name: add, Length: 423, dtype: int64
```

In [49]:
```
location_stats[location_stats<=10]
```

Out[49]:
```
add
Kunj Colony            10
Bhandarkar Road        10
Rajaram Patil Nagar    10
Udyog Nagar            10
Dhayari Phata          10
                       ..
Mukund Nagar            1
Nakhate Vasti           1
Nana Peth               1
Navi Peth               1
                        1
Name: add, Length: 302, dtype: int64
```

In [50]:
```
len(location_stats[location_stats<=10])
```

Out[50]:
```
302
```

In [51]:
```
location_stats_less_than_10 = location_stats[location_stats<=10]
location_stats_less_than_10
```

Out[51]:
```
add
Kunj Colony            10
Bhandarkar Road        10
Rajaram Patil Nagar    10
Udyog Nagar            10
Dhayari Phata          10
                       ..
Mukund Nagar            1
Nakhate Vasti           1
Nana Peth               1
Navi Peth               1
                        1
Name: add, Length: 302, dtype: int64
```

In [52]:
```
df7['add'] = df7['add'].apply(lambda x: 'other' if x in location_stats_less_than_10 else x)
len(df7['add'].unique())
```

Out[52]:
```
122
```

In [53]:
```
df7.tail()
```

Out[53]:

	add	price	total area	beds	baths	price per sqft
11034	other	50.0	850.0	1.0	1.0	5882.352941
11035	other	25.0	650.0	1.0	1.0	3846.153846
11036	Keshav Nagar	73.0	1850.0	3.0	3.0	3945.945946
11037	Kondhwa	170.0	1780.0	3.0	3.0	9550.561798
11038	Wakad	158.0	1008.0	2.0	2.0	15674.603175

In [54]:
```
df8 = df7.copy()
df8
```

Out[54]:

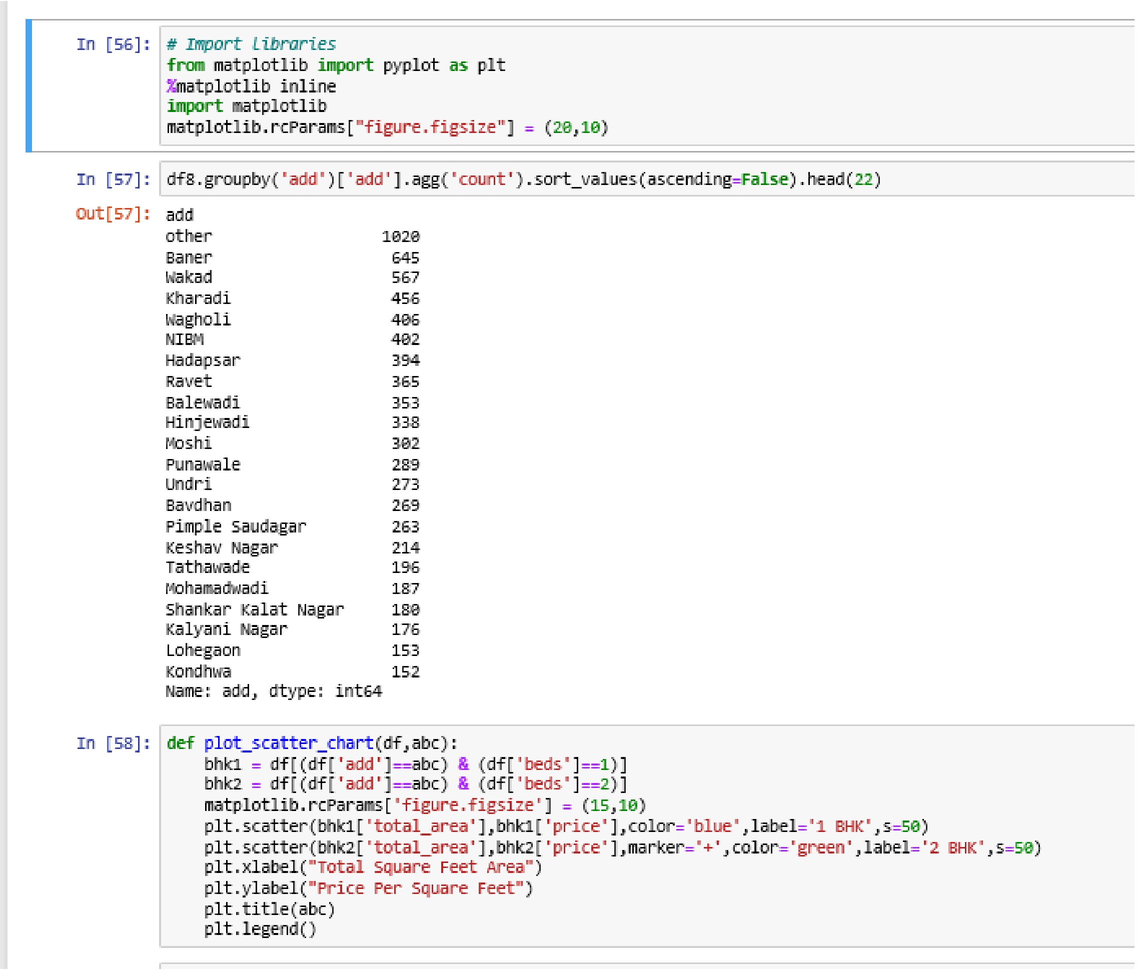
	add	price	total area	beds	baths	price per sqft
0	other	80.0	2000.0	1.0	2.0	4000.000000
1	other	70.5	1200.0	3.0	3.0	5875.000000
2	other	45.0	850.0	2.0	2.0	5294.117647
3	other	85.0	901.0	2.0	2.0	9433.962264
4	other	75.0	1110.0	3.0	3.0	6756.756757
...	...	...	...	...	...	...
11034	other	50.0	850.0	1.0	1.0	5882.352941
11035	other	25.0	650.0	1.0	1.0	3846.153846
11036	Keshav Nagar	73.0	1850.0	3.0	3.0	3945.945946
11037	Kondhwa	170.0	1780.0	3.0	3.0	9550.561798
11038	Wakad	158.0	1008.0	2.0	2.0	15674.603175

11039 rows × 6 columns

In [55]:
```
df8.shape
```

Out[55]:
```
(11039, 6)
```

Let's check if for a given location how does the 1 BHK and 2 BHK property prices look like.

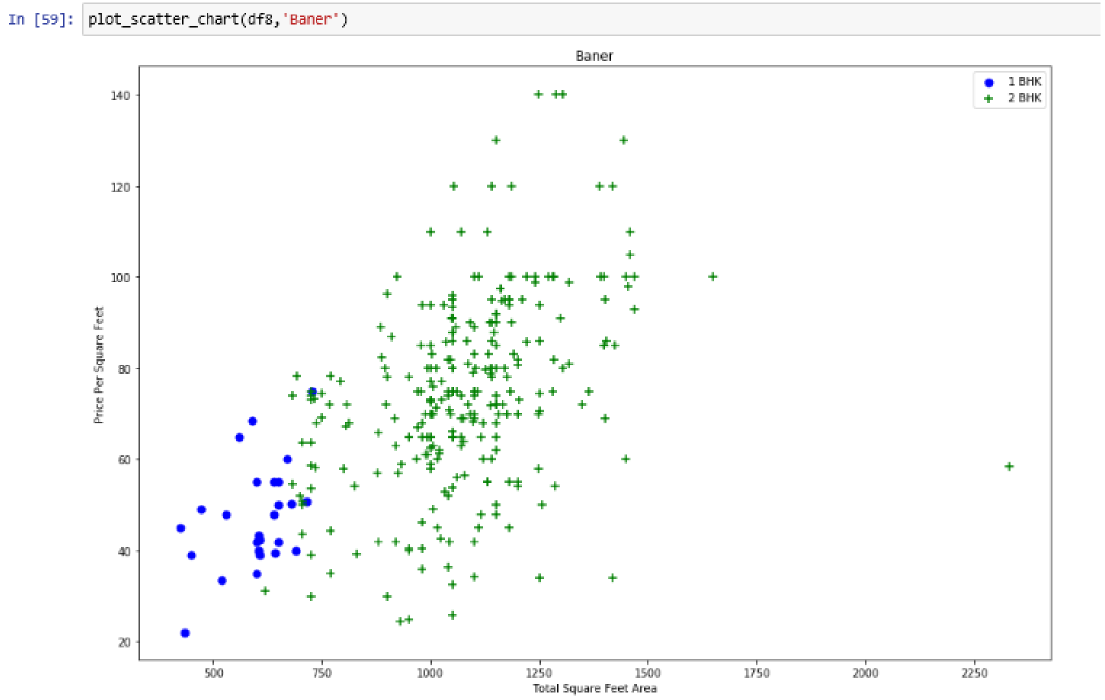


In [56]:
```
# Import Libraries
from matplotlib import pyplot as plt
%matplotlib inline
import matplotlib
matplotlib.rcParams["figure.figsize"] = (20,10)
```

In [57]:
```
df8.groupby('add')['add'].agg('count').sort_values(ascending=False).head(22)
```

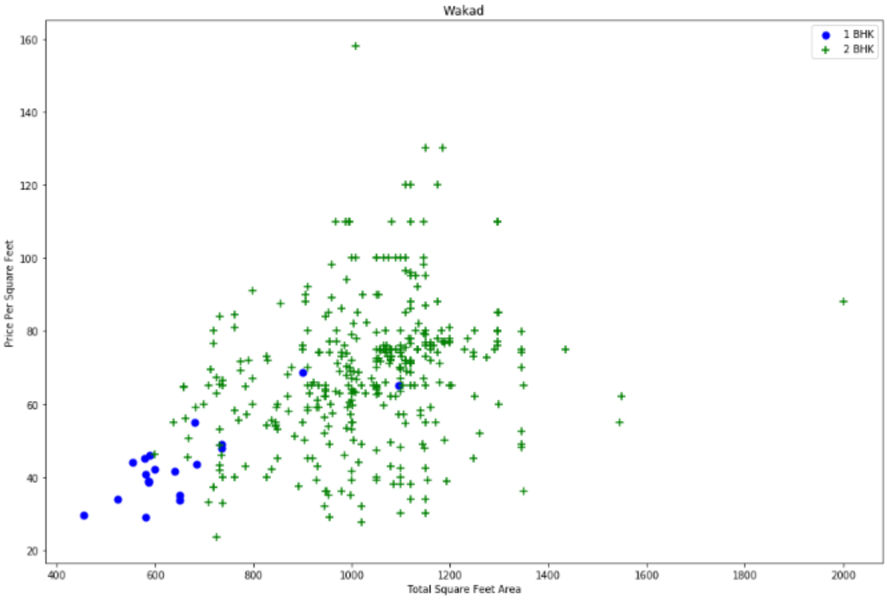
Out[57]:
```
add
other                  1020
Baner                   645
Wakad                   567
Kharadi                 456
Wagholi                 406
NIBM                    402
Hadapsar                394
Ravet                   365
Balewadi                353
Hinjewadi               338
Moshi                   302
Punawale                289
Undri                   273
Bavdhan                 269
Pimple Saudagar         263
Keshav Nagar            214
Tathawade               196
Mohamadwadi             187
Shankar Kalat Nagar     180
Kalyani Nagar           176
Lohegaon                153
Kondhwa                 152
Name: add, dtype: int64
```

In [58]:
```
def plot_scatter_chart(df,abc):
    bhk1 = df[(df['add']==abc) & (df['beds']==1)]
    bhk2 = df[(df['add']==abc) & (df['beds']==2)]
    matplotlib.rcParams['figure.figsize'] = (15,10)
    plt.scatter(bhk1['total_area'],bhk1['price'],color='blue',label='1 BHK',s=50)
    plt.scatter(bhk2['total_area'],bhk2['price'],marker='+',color='green',label='2 BHK',s=50)
    plt.xlabel("Total Square Feet Area")
    plt.ylabel("Price Per Square Feet")
    plt.title(abc)
    plt.legend()
```

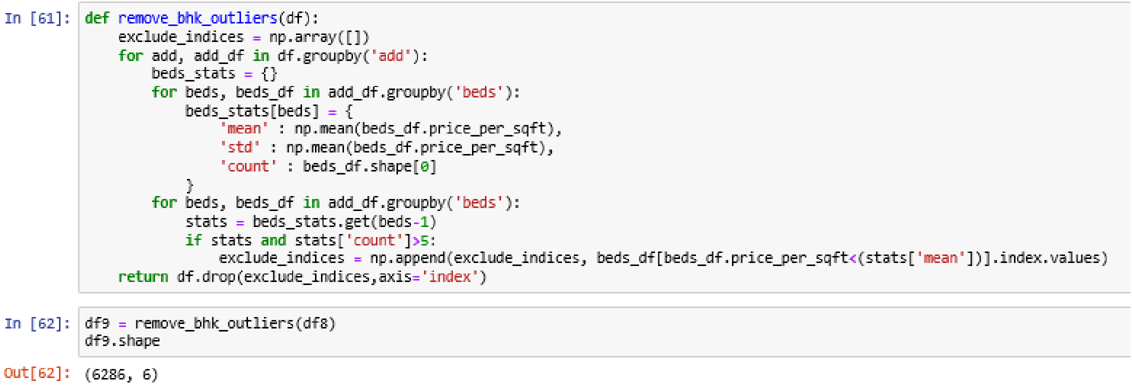


In [59]:
```
plot_scatter_chart(df8,'Baner')
```

```
In [60]: plot_scatter_chart(df8,'Wakad')
```

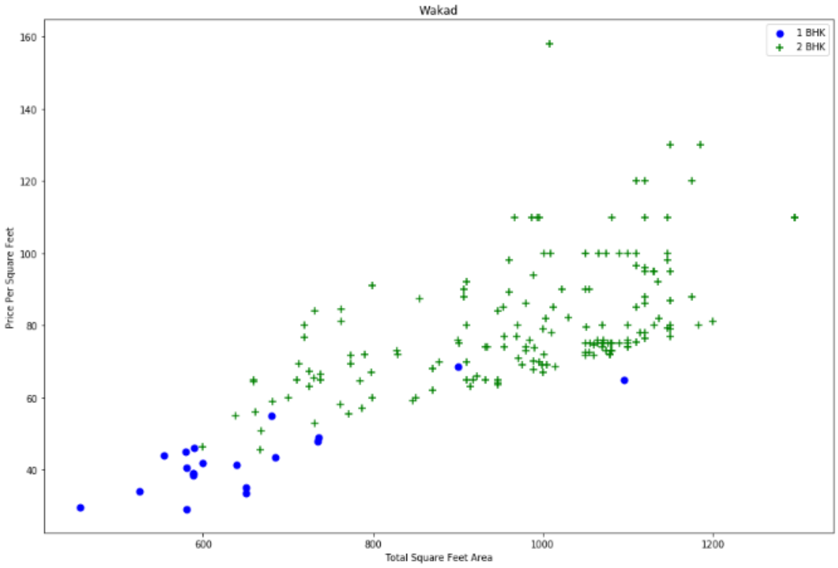


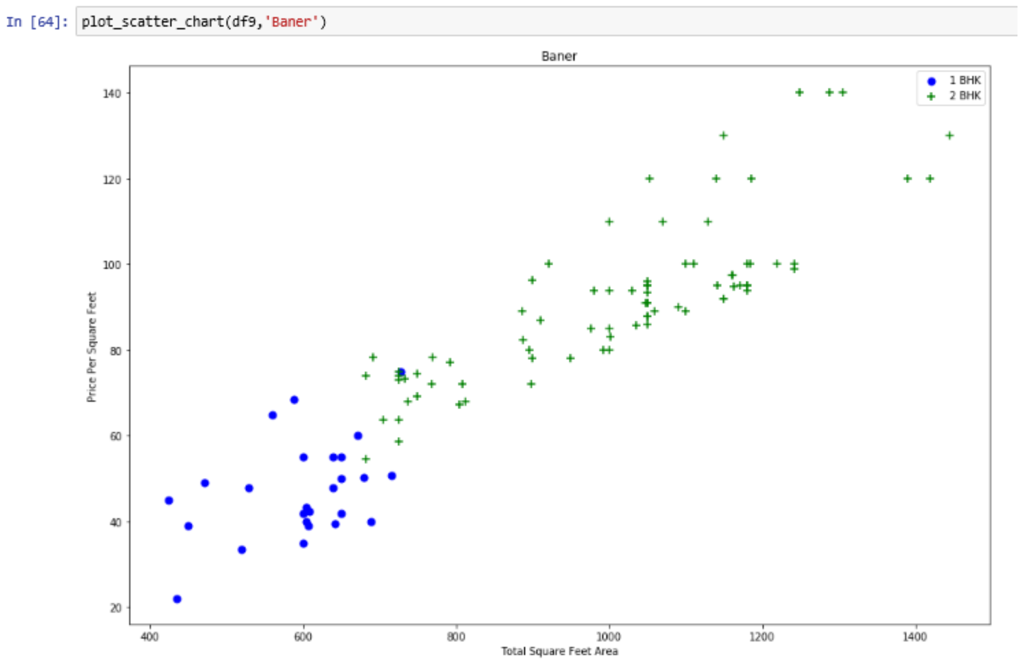
Here we can see that, “+” sign is for 2 BHK and “o” is for 1 BHK. Some 1 BHK apartment prices are higher than 2 BHK apartments, and this are outliers. We have to remove such type of outliers.



```
In [61]: def remove_bhk_outliers(df):
             exclude_indices = np.array([])
             for add, add_df in df.groupby('add'):
                 beds_stats = {}
                 for beds, beds_df in add_df.groupby('beds'):
                     beds_stats[beds] = {
                         'mean' : np.mean(beds_df.price_per_sqft),
                         'std' : np.mean(beds_df.price_per_sqft),
                         'count' : beds_df.shape[0]
                     }
                 for beds, beds_df in add_df.groupby('beds'):
                     stats = beds_stats.get(beds-1)
                     if stats and stats['count']>5:
                         exclude_indices = np.append(exclude_indices, beds_df[beds_df.price_per_sqft<(stats['mean'])].index.values)
             return df.drop(exclude_indices,axis='index')

In [62]: df9 = remove_bhk_outliers(df8)
         df9.shape

Out[62]: (6286, 6)
```

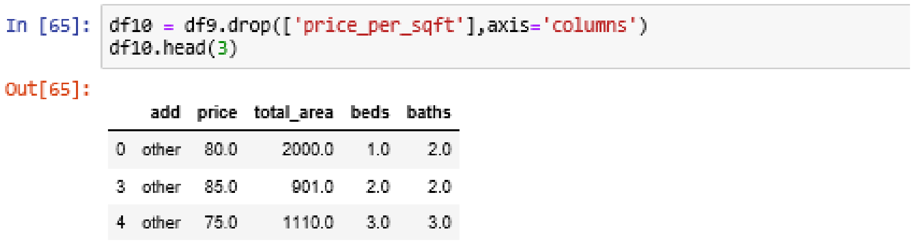
```
In [63]: plot_scatter_chart(df9,'Wakad')
```





```
In [64]: plot_scatter_chart(df9,'Baner')
```

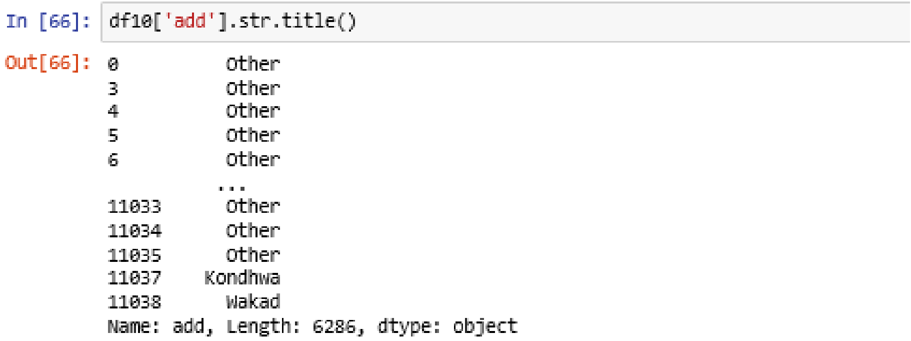
As we can see that in above 2 graphs, outliers are removed.



```
In [65]: df10 = df9.drop(['price_per_sqft'],axis='columns')
         df10.head(3)
```

Out[65]:

	add	price	total_area	beds	baths
0	other	80.0	2000.0	1.0	2.0
3	other	85.0	901.0	2.0	2.0
4	other	75.0	1110.0	3.0	3.0



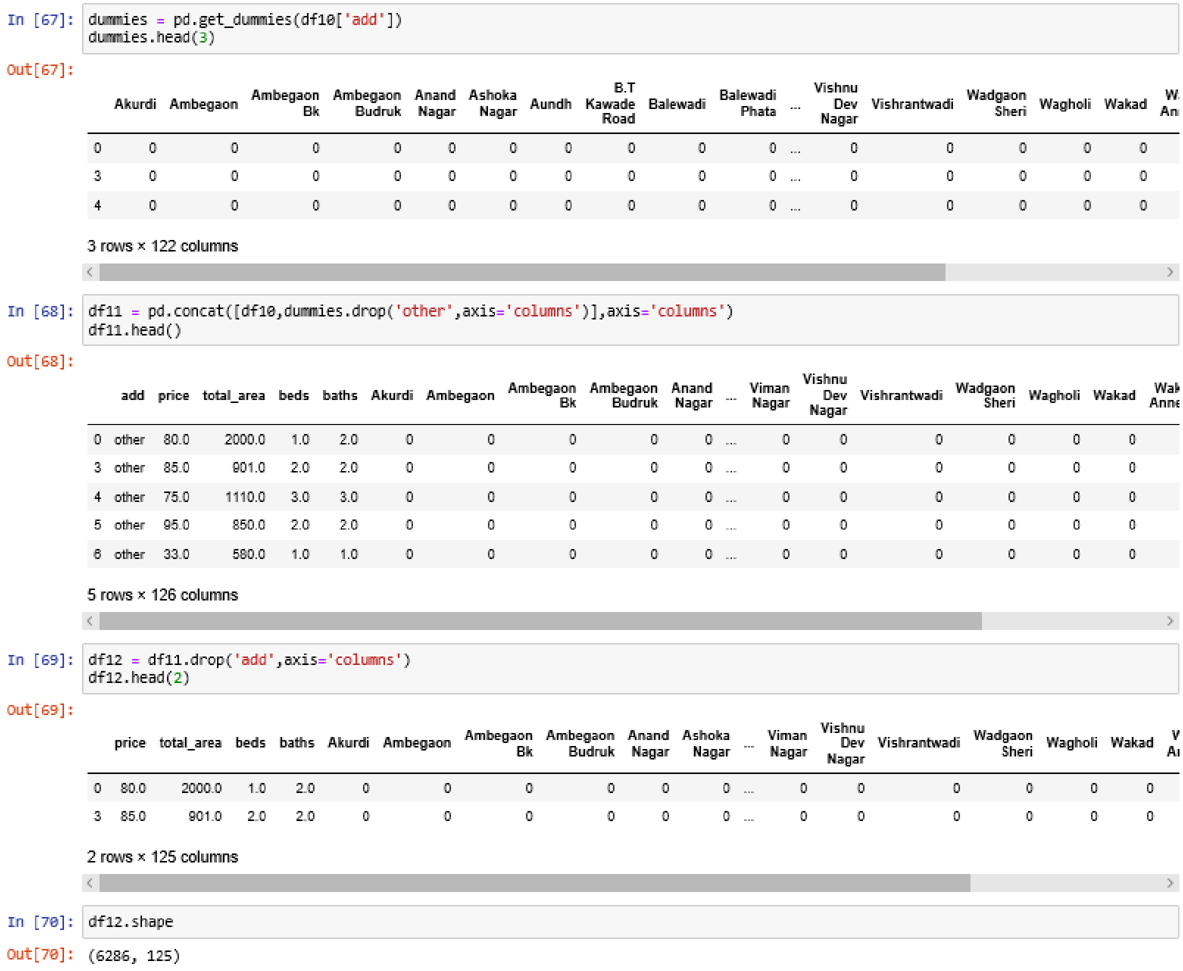
```
In [66]: df10['add'].str.title()
```

```
Out[66]: 0           Other
         3           Other
         4           Other
         5           Other
         6           Other
                    ...
         11033       Other
         11034       Other
         11035       Other
         11037    Kondhwa
         11038       Wakad
         Name: add, Length: 6286, dtype: object
```

In above table, we removed “price_per_sqft” column which is not required.

Use One Hot Encoding for add column

As our whole data is numerical except ”add” column, we have to convert “add” column in to numeric values by adding dummies variable in that.



In [67]:
```
dummies = pd.get_dummies(df10['add'])
dummies.head(3)
```

Out[67]:

	Akurdi	Ambegaon	Ambegaon Bk	Ambegaon Budruk	Anand Nagar	Ashoka Nagar	Aundh	B.T Kawade Road	Balewadi	Balewadi Phata	...	Vishnu Dev Nagar	Vishrantwadi	Wadgaon Sheri	Wagholi	Wakad	W An
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	

3 rows × 122 columns

In [68]:
```
df11 = pd.concat([df10,dummies.drop('other',axis='columns')],axis='columns')
df11.head()
```

Out[68]:

	add	price	total_area	beds	baths	Akurdi	Ambegaon	Ambegaon Bk	Ambegaon Budruk	Anand Nagar	...	Viman Nagar	Vishnu Dev Nagar	Vishrantwadi	Wadgaon Sheri	Wagholi	Wakad	Wak Anne
0	other	80.0	2000.0	1.0	2.0	0	0	0	0	0	...	0	0	0	0	0	0	
3	other	85.0	901.0	2.0	2.0	0	0	0	0	0	...	0	0	0	0	0	0	
4	other	75.0	1110.0	3.0	3.0	0	0	0	0	0	...	0	0	0	0	0	0	
5	other	95.0	850.0	2.0	2.0	0	0	0	0	0	...	0	0	0	0	0	0	
6	other	33.0	580.0	1.0	1.0	0	0	0	0	0	...	0	0	0	0	0	0	

5 rows × 126 columns

In [69]:
```
df12 = df11.drop('add',axis='columns')
df12.head(2)
```

Out[69]:

	price	total_area	beds	baths	Akurdi	Ambegaon	Ambegaon Bk	Ambegaon Budruk	Anand Nagar	Ashoka Nagar	...	Viman Nagar	Vishnu Dev Nagar	Vishrantwadi	Wadgaon Sheri	Wagholi	Wakad	V Ai
0	80.0	2000.0	1.0	2.0	0	0	0	0	0	0	...	0	0	0	0	0	0	
3	85.0	901.0	2.0	2.0	0	0	0	0	0	0	...	0	0	0	0	0	0	

2 rows × 125 columns

In [70]:
```
df12.shape
```

Out[70]:
```
(6286, 125)
```

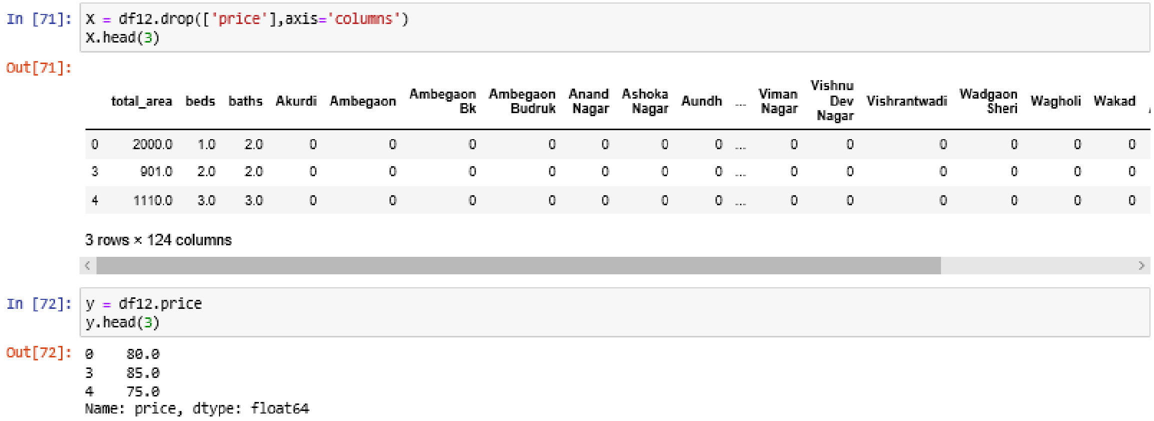
Now we have converted our whole dataset into numerical values. And as we see that we have 6286 rows and 125 columns.

Our project main aim is to predict property values using dataset. Then we have to use **Linear Regression** technique.

Y = a + bX, where Y is dependent variable and X is independent variable.

In our case, our “price” is our dependent variable is i.e. Y and whole dataset is independent variable i.e. X.

So, we will separate our dataset into X and y variable as we require for our predictions.



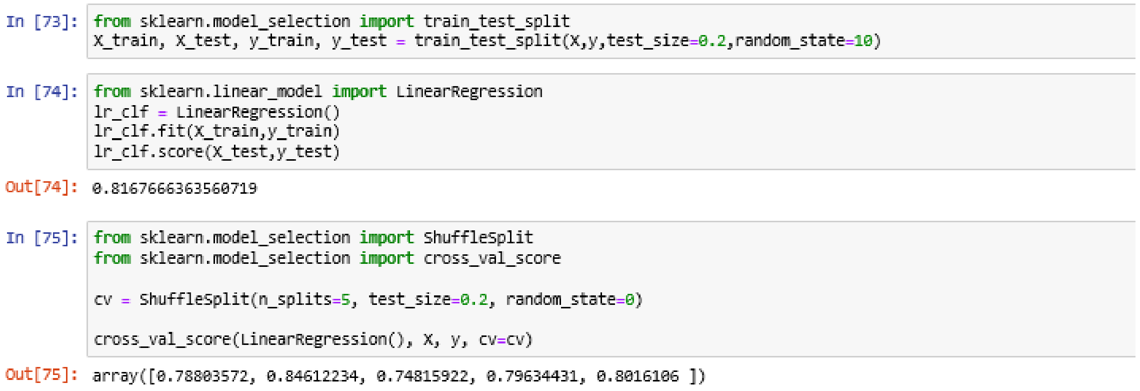
```
In [71]: X = df12.drop(['price'],axis='columns')
         X.head(3)

Out[71]:
```

	total_area	beds	baths	Akurdi	Ambegaon	Ambegaon Bk	Ambegaon Budruk	Anand Nagar	Ashoka Nagar	Aundh	...	Viman Nagar	Vishnu Dev Nagar	Vishrantwadi	Wadgaon Sheri	Wagholi	Wakad
0	2000.0	1.0	2.0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
3	901.0	2.0	2.0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
4	1110.0	3.0	3.0	0	0	0	0	0	0	0	...	0	0	0	0	0	0

3 rows × 124 columns

```
In [72]: y = df12.price
         y.head(3)

Out[72]: 0    80.0
         3    85.0
         4    75.0
         Name: price, dtype: float64
```

We will use K Fold Cross Validation to measure accuracy of our Linear Regression model.

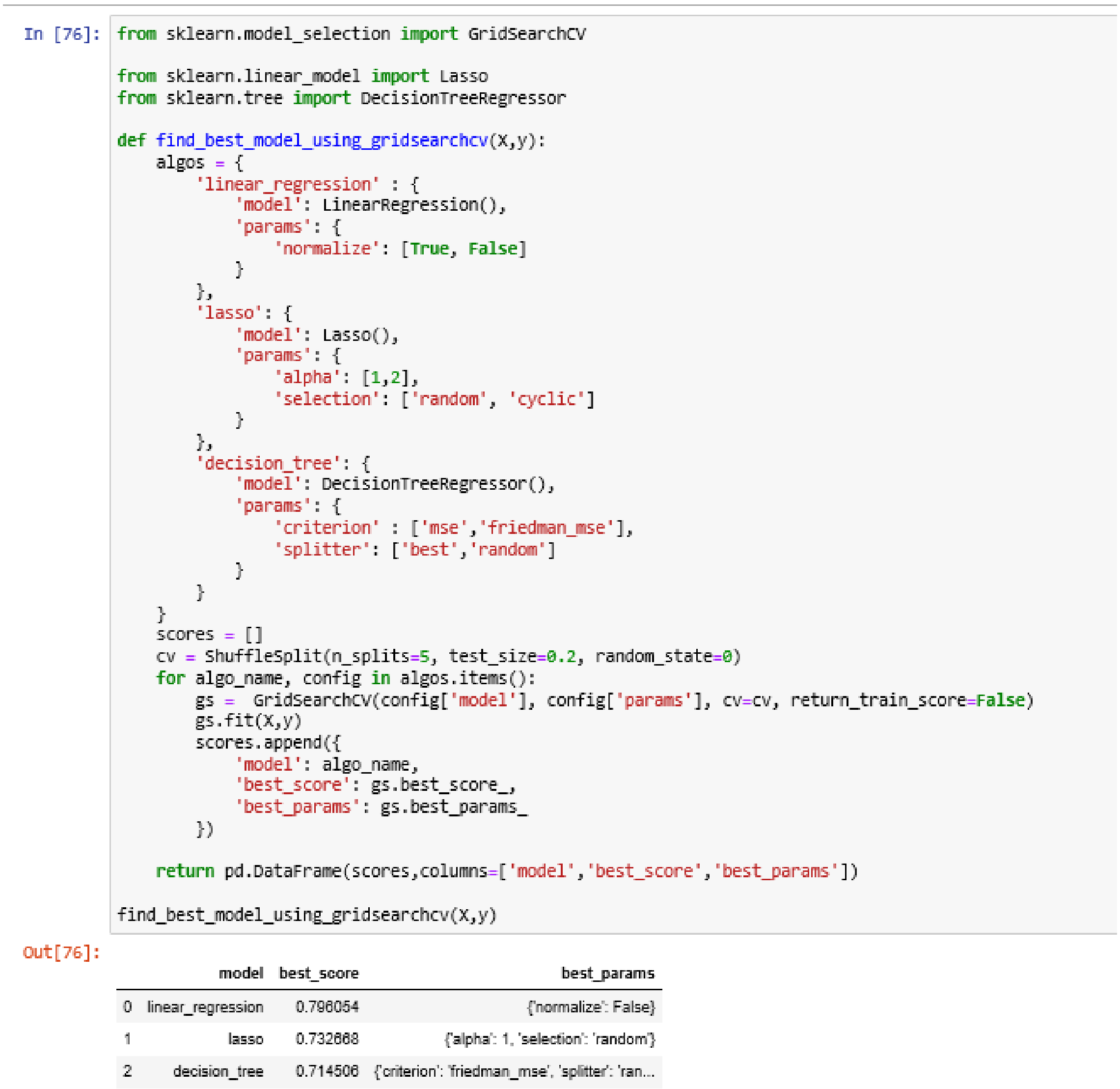


```
In [73]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=10)

In [74]: from sklearn.linear_model import LinearRegression
         lr_clf = LinearRegression()
         lr_clf.fit(X_train,y_train)
         lr_clf.score(X_test,y_test)

Out[74]: 0.8167666363560719

In [75]: from sklearn.model_selection import ShuffleSplit
         from sklearn.model_selection import cross_val_score

         cv = ShuffleSplit(n_splits=5, test_size=0.2, random_state=0)

         cross_val_score(LinearRegression(), X, y, cv=cv)

Out[75]: array([0.78803572, 0.84612234, 0.74815922, 0.79634431, 0.8016106 ])
```

Here for checking accuracy we will take 20% as our test sample and remaining 80% as for model training from whole dataset.

We can see that in 5 iterations we get 3 scores approx. above 80% all the time. This is pretty good but we want to test few other algorithms for regression to see if we can get even better score. We will use GridSearchCV for this purpose.

- **Find best model using GridSearchCV**



In [76]:
```
from sklearn.model_selection import GridSearchCV

from sklearn.linear_model import Lasso
from sklearn.tree import DecisionTreeRegressor

def find_best_model_using_gridsearchcv(X,y):
    algos = {
        'linear_regression' : {
            'model': LinearRegression(),
            'params': {
                'normalize': [True, False]
            }
        },
        'lasso': {
            'model': Lasso(),
            'params': {
                'alpha': [1,2],
                'selection': ['random', 'cyclic']
            }
        },
        'decision_tree': {
            'model': DecisionTreeRegressor(),
            'params': {
                'criterion' : ['mse','friedman_mse'],
                'splitter': ['best','random']
            }
        }
    }
    scores = []
    cv = ShuffleSplit(n_splits=5, test_size=0.2, random_state=0)
    for algo_name, config in algos.items():
        gs =  GridSearchCV(config['model'], config['params'], cv=cv, return_train_score=False)
        gs.fit(X,y)
        scores.append({
            'model': algo_name,
            'best_score': gs.best_score_,
            'best_params': gs.best_params_
        })

    return pd.DataFrame(scores,columns=['model','best_score','best_params'])

find_best_model_using_gridsearchcv(X,y)
```

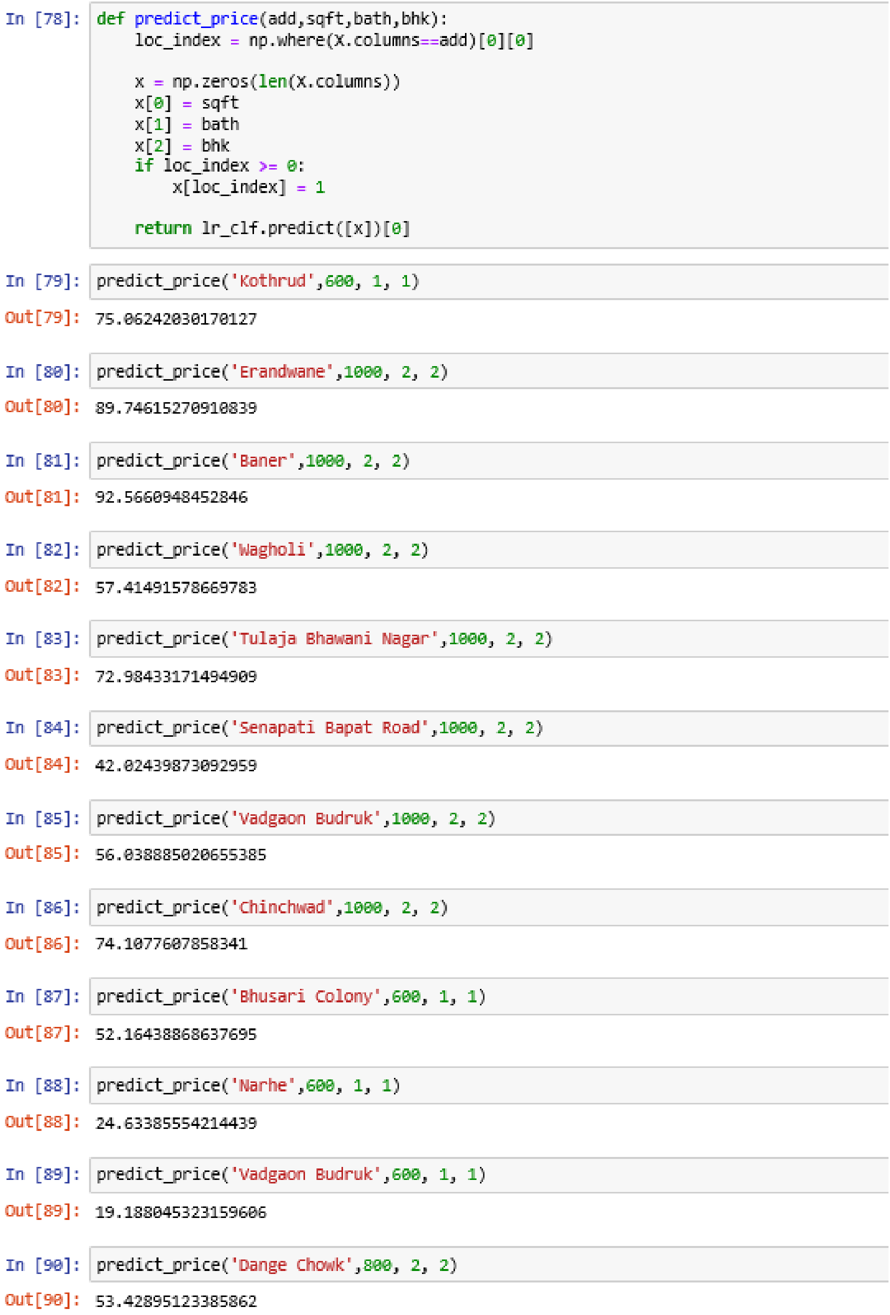
Out[76]:

	model	best_score	best_params
0	linear_regression	0.796054	{'normalize': False}
1	lasso	0.732668	{'alpha': 1, 'selection': 'random'}
2	decision_tree	0.714506	{'criterion': 'friedman_mse', 'splitter': 'ran...

Here we used, Lasso regression technique, Linear regression technique and Decision Tree regression technique for comparing which regression technique gives best score for model building.

Based on above results we can say that the Linear Regression gives the best score. Hence, we will use that.

So, we can use Linear Regression technique for model building for predictions.



```
In [78]: def predict_price(add,sqft,bath,bhk):
             loc_index = np.where(X.columns==add)[0][0]

             x = np.zeros(len(X.columns))
             x[0] = sqft
             x[1] = bath
             x[2] = bhk
             if loc_index >= 0:
                 x[loc_index] = 1

             return lr_clf.predict([x])[0]

In [79]: predict_price('Kothrud',600, 1, 1)

Out[79]: 75.06242030170127

In [80]: predict_price('Erandwane',1000, 2, 2)

Out[80]: 89.74615270910839

In [81]: predict_price('Baner',1000, 2, 2)

Out[81]: 92.5660948452846

In [82]: predict_price('Wagholi',1000, 2, 2)

Out[82]: 57.41491578669783

In [83]: predict_price('Tulaja Bhawani Nagar',1000, 2, 2)

Out[83]: 72.98433171494909

In [84]: predict_price('Senapati Bapat Road',1000, 2, 2)

Out[84]: 42.02439873092959

In [85]: predict_price('Vadgaon Budruk',1000, 2, 2)

Out[85]: 56.038885020655385

In [86]: predict_price('Chinchwad',1000, 2, 2)

Out[86]: 74.1077607858341

In [87]: predict_price('Bhusari Colony',600, 1, 1)

Out[87]: 52.16438868637695

In [88]: predict_price('Narhe',600, 1, 1)

Out[88]: 24.63385554214439

In [89]: predict_price('Vadgaon Budruk',600, 1, 1)

Out[89]: 19.188045323159606

In [90]: predict_price('Dange Chowk',800, 2, 2)

Out[90]: 53.42895123385862
```

Here we can see that, after model building, we have predicted property prices of certain locations from Pune city. Above predicted property prices are in lacs.