In [6]:
```python
import numpy as np
import pandas as pd
```

In [7]:
```python
df = pd.read_csv('House Price India.csv')
```

Out[7]:

	id	Date	number of bedrooms	number of bathrooms	living area	lot area	number of floors	waterfront present	number of views	condit of hou
0	6762810145	42491	5	2.50	3650	9050	2.0	0	4	
1	6762810635	42491	4	2.50	2920	4000	1.5	0	0	
2	6762810998	42491	5	2.75	2910	9480	1.5	0	0	
3	6762812605	42491	4	2.50	3310	42998	2.0	0	0	
4	6762812919	42491	3	2.00	2710	4500	1.5	0	0	

5 rows × 23 columns

In [8]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14620 entries, 0 to 14619
Data columns (total 23 columns):
 #   Column                                 Non-Null Count  Dtype  
---  ------                                 --------------  -----  
 0   id                                     14620 non-null  int64  
 1   Date                                   14620 non-null  int64  
 2   number of bedrooms                     14620 non-null  int64  
 3   number of bathrooms                    14620 non-null  float64
 4   living area                            14620 non-null  int64  
 5   lot area                               14620 non-null  int64  
 6   number of floors                       14620 non-null  float64
 7   waterfront present                     14620 non-null  int64  
 8   number of views                        14620 non-null  int64  
 9   condition of the house                 14620 non-null  int64  
 10  grade of the house                     14620 non-null  int64  
 11  Area of the house(excluding basement)  14620 non-null  int64  
 12  Area of the basement                   14620 non-null  int64  
 13  Built Year                             14620 non-null  int64  
 14  Renovation Year                        14620 non-null  int64  
 15  Postal Code                            14620 non-null  int64  
 16  Lattitude                              14620 non-null  float64
 17  Longitude                              14620 non-null  float64
 18  living_area_renov                      14620 non-null  int64  
 19  lot_area_renov                         14620 non-null  int64  
 20  Number of schools nearby               14620 non-null  int64  
 21  Distance from the airport              14620 non-null  int64  
 22  Price                                  14620 non-null  int64  
dtypes: float64(4), int64(19)
memory usage: 2.6 MB
```

Descriptive Analysis

In [9]:
```
df.describe()
```

Out[9]:

	id	Date	number of bedrooms	number of bathrooms	living area	lot area
count	1.462000e+04	14620.000000	14620.000000	14620.000000	14620.000000	1.462000e+04
mean	6.762821e+09	42604.538646	3.379343	2.129583	2098.262996	1.509328e+04
std	6.237575e+03	67.347991	0.938719	0.769934	928.275721	3.791962e+04
min	6.762810e+09	42491.000000	1.000000	0.500000	370.000000	5.200000e+02
25%	6.762815e+09	42546.000000	3.000000	1.750000	1440.000000	5.010750e+03
50%	6.762821e+09	42600.000000	3.000000	2.250000	1930.000000	7.620000e+03
75%	6.762826e+09	42662.000000	4.000000	2.500000	2570.000000	1.080000e+04
max	6.762832e+09	42734.000000	33.000000	8.000000	13540.000000	1.074218e+06

8 rows × 23 columns

Handling Missing Values

In [10]:
```
df.isnull().sum()
```

Out[10]:
```
id                                         0
Date                                       0
number of bedrooms                         0
number of bathrooms                        0
living area                                0
lot area                                   0
number of floors                           0
waterfront present                         0
number of views                            0
condition of the house                     0
grade of the house                         0
Area of the house(excluding basement)      0
Area of the basement                       0
Built Year                                 0
Renovation Year                            0
Postal Code                                0
Lattitude                                  0
Longitude                                  0
living_area_renov                          0
lot_area_renov                             0
Number of schools nearby                   0
Distance from the airport                  0
Price                                      0
dtype: int64
```

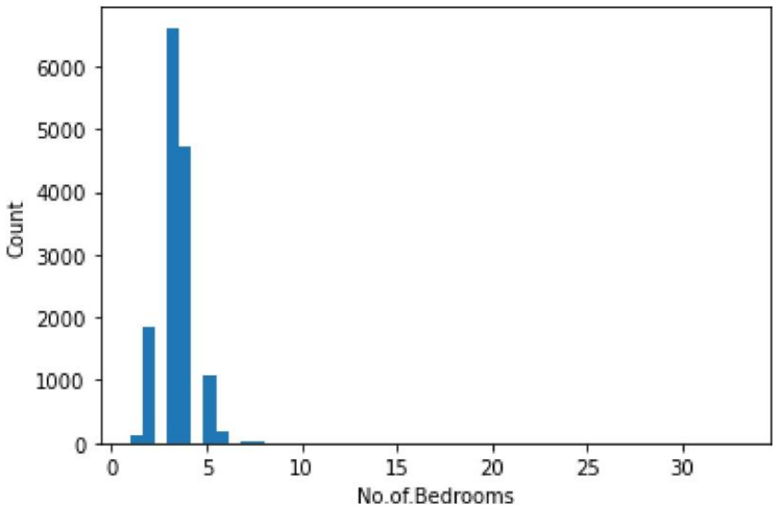
The above information shows that the none of the columns contains any null value in it. We don't need to perform any specific operations to handle the missing values.

Univariate Analysis

Histogram

In [11]:
```
plt.hist(df['number of bedrooms'],bins=50)
```

Out[11]: Text(0, 0.5, 'Count')



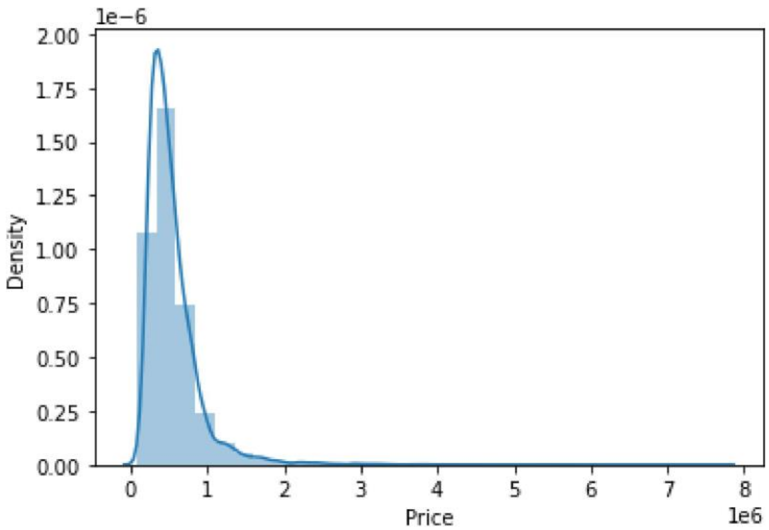
From the above graph we can clearly see that the peek count above 6000 is at range between 0 to 5. As the no.of.bedrooms increases after 5 the count values decreases tremoundously.

Distplot

In [12]:
```
sns.distplot(df['Price'],bins=30)
```

```
C:\Users\priya\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[12]: <AxesSubplot:xlabel='Price', ylabel='Density'>



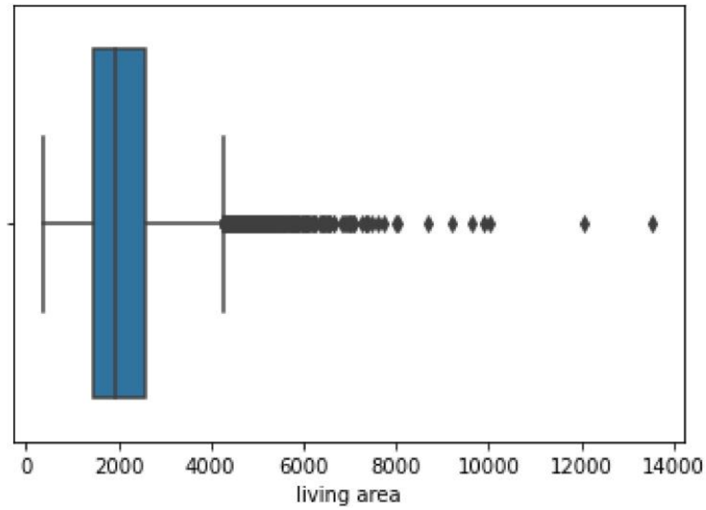
From the above distplot we came to know that the price distributes at peek between 0 and 1 related to density of the distribution.

Boxplot

In [12]:
```
sns.distplot(df['Price'],bins=30)
```

```
C:\Users\priya\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureW
arning: Pass the following variable as a keyword arg: x. From version 0.12, t
he only valid positional argument will be `data`, and passing other arguments
without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[13]: <AxesSubplot:xlabel='living area'>



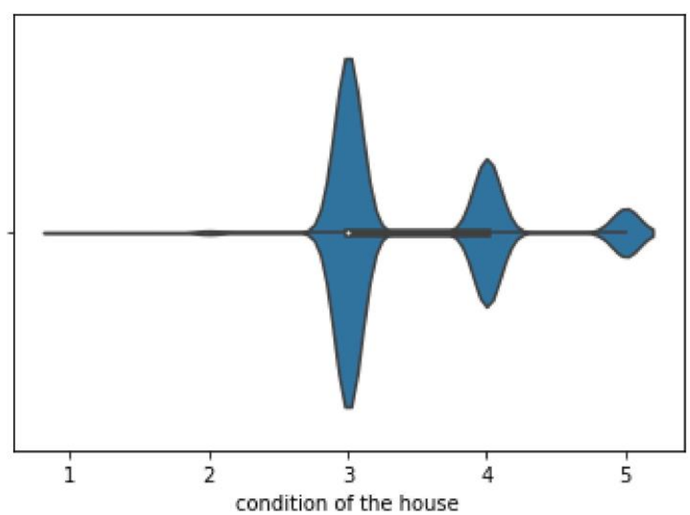
Boxplot is also used for detect the outlier in data set. It captures the summary of the data efficiently with a simple box and whiskers and allows us to compare easily across groups.

Boxplot for living area and it contains many outliers and many outliers present in the features. The above one is a sample for detecting outliers.

Violinplot

In [14]:
```
sns.violinplot(x=df['condition of the house'])
```

Out[14]: <AxesSubplot:xlabel='condition of the house'>



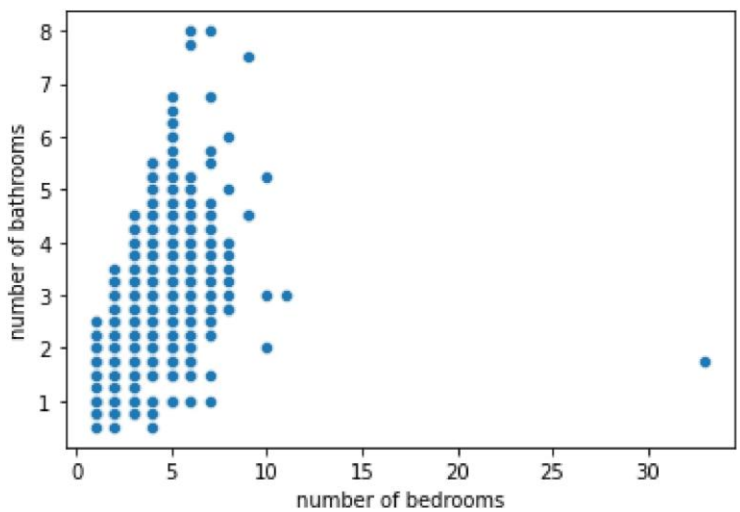
violinplot is used to vizualize the distribution numerical data and it shows the full distribution of data. The mean value of the variable "condition of the house" lies in 3 and the interquartile ranges between 3 to 4. The rest thin lines represents the rest distributions, except for the points that are determined to be the outliers. The higher probability lies in 3 and lowest probability lies above 5.

Bivariate Analysis

Scatterplot

In [15]:
```
sns.scatterplot(x=df['number of bedrooms'],y=df['number of bathrooms'])
```

Out[15]:
```
<AxesSubplot:xlabel='number of bedrooms', ylabel='number of bathrooms'>
```

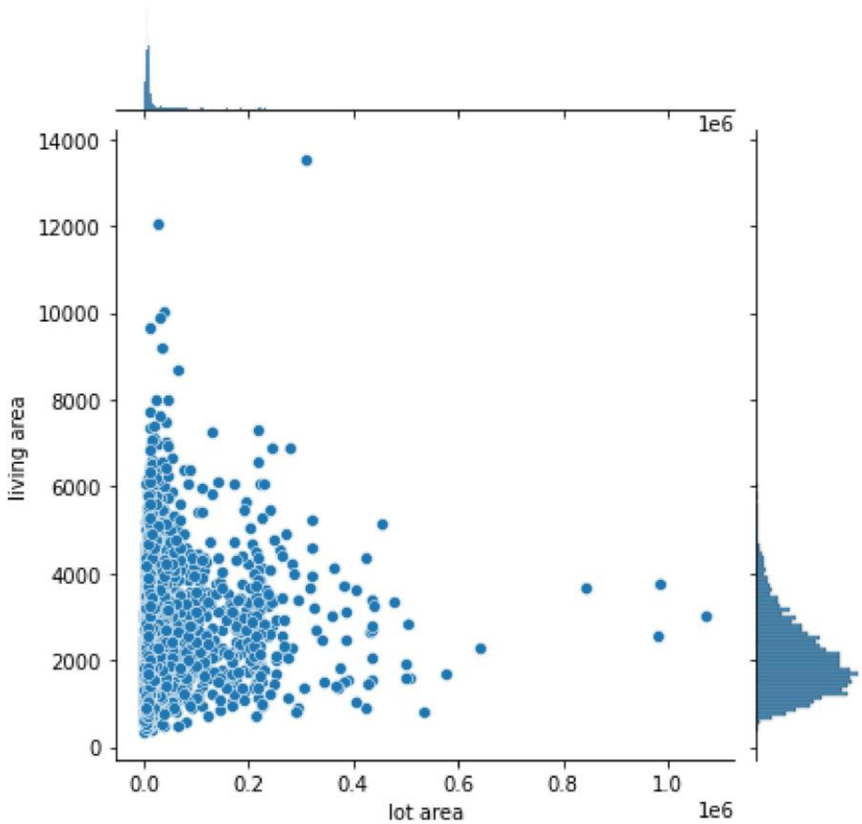


The scatterplot is used to show distributions between two variables. For no.of.bathrooms and no.of.bedrooms as far as the bathroom increases the bedroom number increases. And there are some outliers present in them.

Jointplot

```
In [16]: sns.jointplot(data = df,x = 'lot area',y = 'living area')
```

```
Out[16]: <seaborn.axisgrid.JointGrid at 0x2b56adb7f70>
```

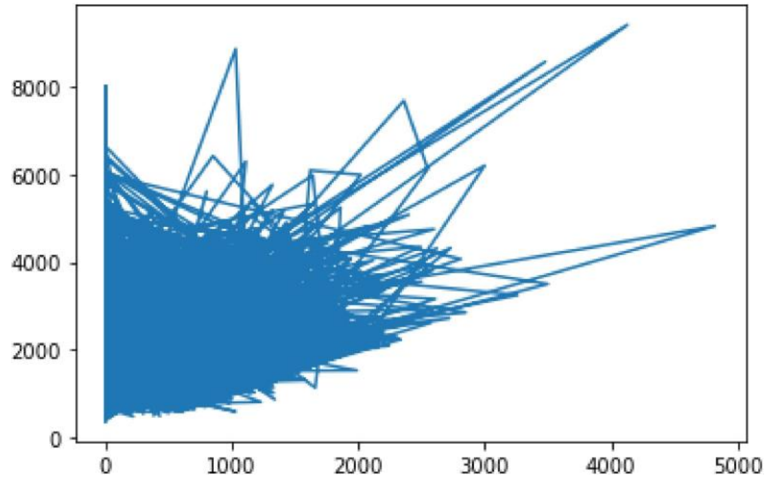


The relation between living area vs lot area and univariate of these has been shown. As far as the living area increases the lot area increases slighter and present many outliers between them. Univariate distribution of lot area remains same with slight increase in area but for living area the peak value is achieved at 2000 by gradual increase in it and then decreases until at a range of 5000.

Line plot

In [16]:
```
sns.jointplot(data = df,x = 'lot area',y = 'living area')
```

Out[17]: [<matplotlib.lines.Line2D at 0x2b56d2860d0>]



Multivariate Analysis

Pairplot

In [18]:
```
X = df[['number of bedrooms','number of bathrooms','lot area','living area','P
```

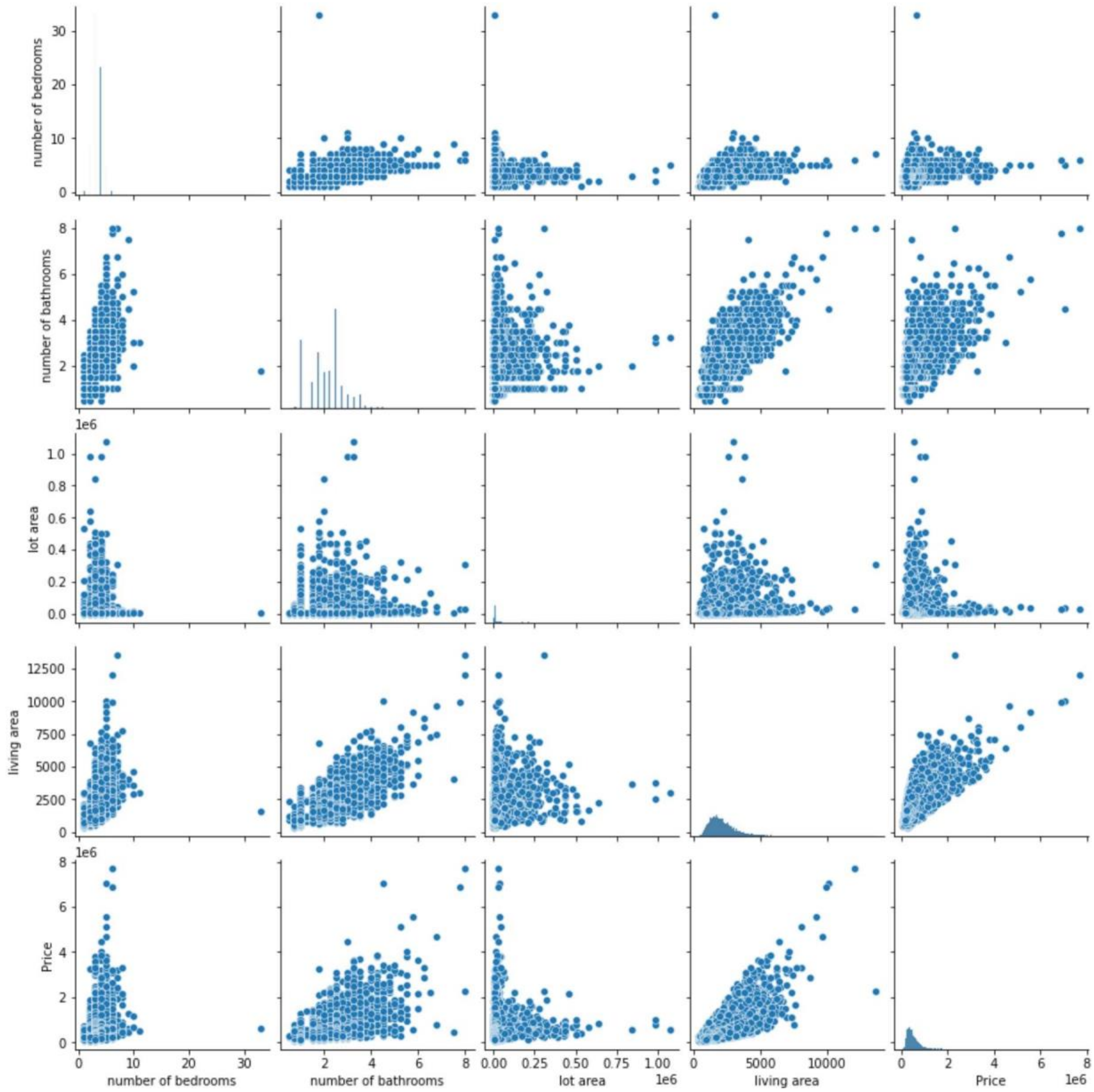
Out[18]:

	number of bedrooms	number of bathrooms	lot area	living area	Price
0	5	2.50	9050	3650	2380000
1	4	2.50	4000	2920	1400000
2	5	2.75	9480	2910	1200000
3	4	2.50	42998	3310	838000
4	3	2.00	4500	2710	805000
...	...	...	...	...	...
14615	2	1.50	20000	1556	221700
14616	3	2.00	7000	1680	219200
14617	2	1.00	6120	1070	209000
14618	4	1.00	6621	1030	205000
14619	3	1.00	4770	900	146000

14620 rows × 5 columns

In [16]:
```
sns.jointplot(data = df,x = 'lot area',y = 'living area')
```

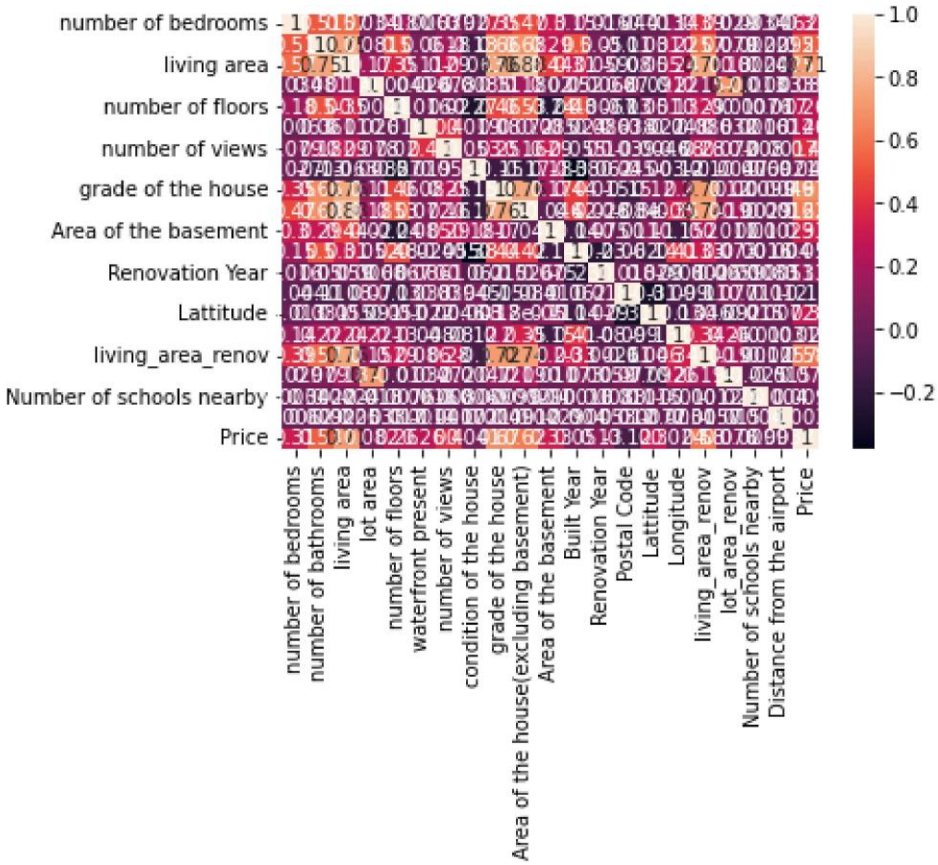
Out[19]: <seaborn.axisgrid.PairGrid at 0x2b56a776d00>



From pairplot we can clearly see that some variable are linear to some variable and logistic to some variables. Most of the variables are linear to other variables. But in all variables outliers present in it.

In [16]:
```
sns.jointplot(data = df,x = 'lot area',y = 'living area')
```

Out[20]: <AxesSubplot:>



In [21]:
```
a=df.groupby('number of bedrooms')['Price'].median()
```

Out[21]: <AxesSubplot:xlabel='number of bedrooms'>