```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import sklearn as sk
from sklearn import datasets
import seaborn as sns
```

```
x=pd.read_csv('/content/Housing.csv')
```

```
x.shape
# number of rows are 543 and number of columns are 13
```

```
(545, 13)
```

```
x.head()
# we will get the first four rows output by default
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parking	prefarea
0	13300000	7420	4	2	3	yes	no	no	no	yes	2	yes
1	12250000	8960	4	4	4	yes	no	no	no	yes	3	no
2	12250000	9960	3	2	2	yes	no	yes	no	no	2	yes
3	12215000	7500	4	2	2	yes	no	yes	no	yes	3	yes
4	11410000	7420	4	1	2	yes	yes	yes	no	yes	2	no

```
x.tail()
# we will get the last five rows output by default
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwa
540	1820000	3000	2	1	1	yes	no	yes	
541	1767150	2400	3	1	1	no	no	no	
542	1750000	3620	2	1	1	yes	no	no	
543	1750000	2910	3	1	1	no	no	no	
544	1750000	3850	3	1	2	yes	no	no	

```
x.info()
# the number of column and its data type and null count
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   price             545 non-null    int64
 1   area              545 non-null    int64
 2   bedrooms          545 non-null    int64
 3   bathrooms         545 non-null    int64
 4   stories           545 non-null    int64
 5   mainroad          545 non-null    object
 6   guestroom         545 non-null    object
 7   basement          545 non-null    object
 8   hotwaterheating   545 non-null    object
 9   airconditioning   545 non-null    object
 10  parking           545 non-null    int64
 11  prefarea          545 non-null    object
 12  furnishingstatus  545 non-null    object
dtypes: int64(6), object(7)
memory usage: 55.5+ KB
```

```
x.nunique()
# the number of unique values each column have.
```

```
price               219
area                284
bedrooms              6
bathrooms             4
stories               4
mainroad              2
guestroom             2
basement              2
hotwaterheating       2
airconditioning       2
parking               4
```

```
prefarea            2
furnishingstatus    3
dtype: int64
```

```
x.isnull()
# the null values present in each cell will be repreasented by true and if cell value is present it will show false
```

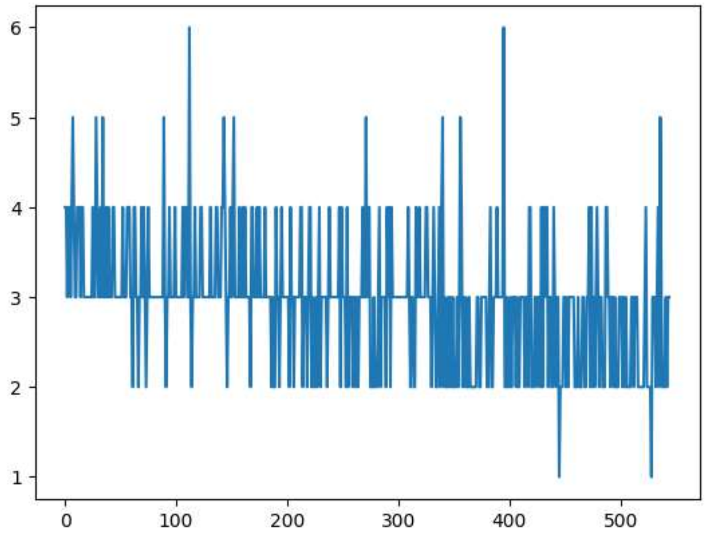
	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwat
0	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	
540	False	False	False	False	False	False	False	False	
541	False	False	False	False	False	False	False	False	
542	False	False	False	False	False	False	False	False	
543	False	False	False	False	False	False	False	False	
544	False	False	False	False	False	False	False	False	

545 rows × 13 columns

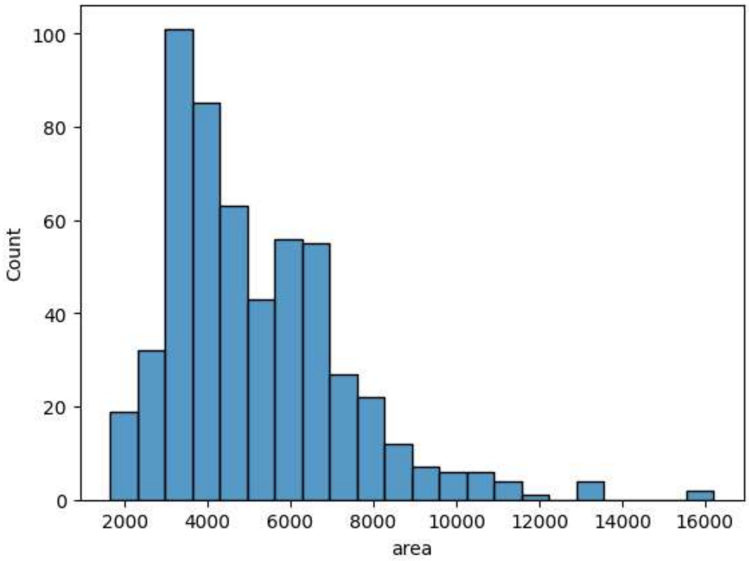
```
x.isnull().sum()
# thier is no null value in the following table
```

```
price               0
area                0
bedrooms            0
bathrooms           0
stories             0
mainroad            0
guestroom           0
basement            0
hotwaterheating     0
airconditioning     0
parking             0
prefarea            0
furnishingstatus    0
dtype: int64
```

```
plt.plot(x.bedrooms)
plt.show()
# Thus we can interpret it that 2 house are having 6 rooms and 2 house are having 1 bedroom, which can be the outlier for the following
```

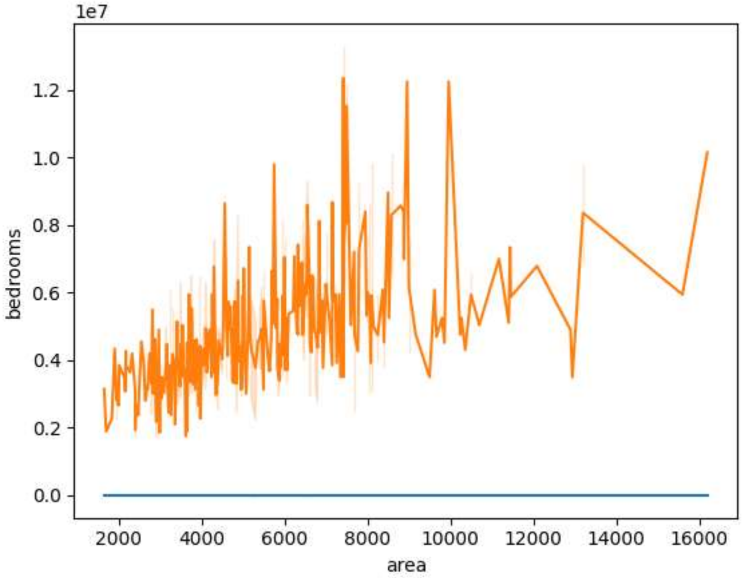


```
sns.histplot(data=x,x='area')
plt.show()
# we can say that count is maximum with area is around 4000 area and between 14000 and 15000 no house. the outlier in terms of area of t
```



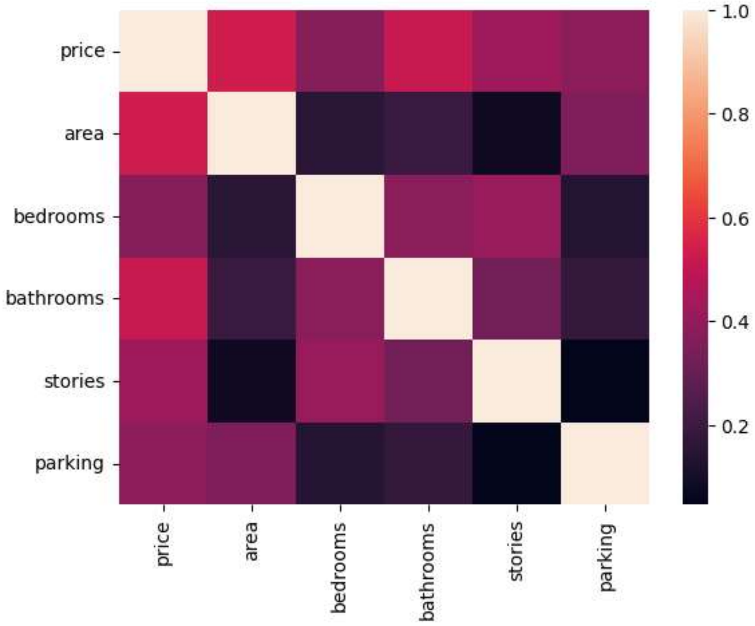
```
sns.lineplot(x='area',y='bedrooms',data=x)
sns.lineplot(x='area',y='price',data=x)
# plt.show()
```

```
<Axes: xlabel='area', ylabel='bedrooms'>
```



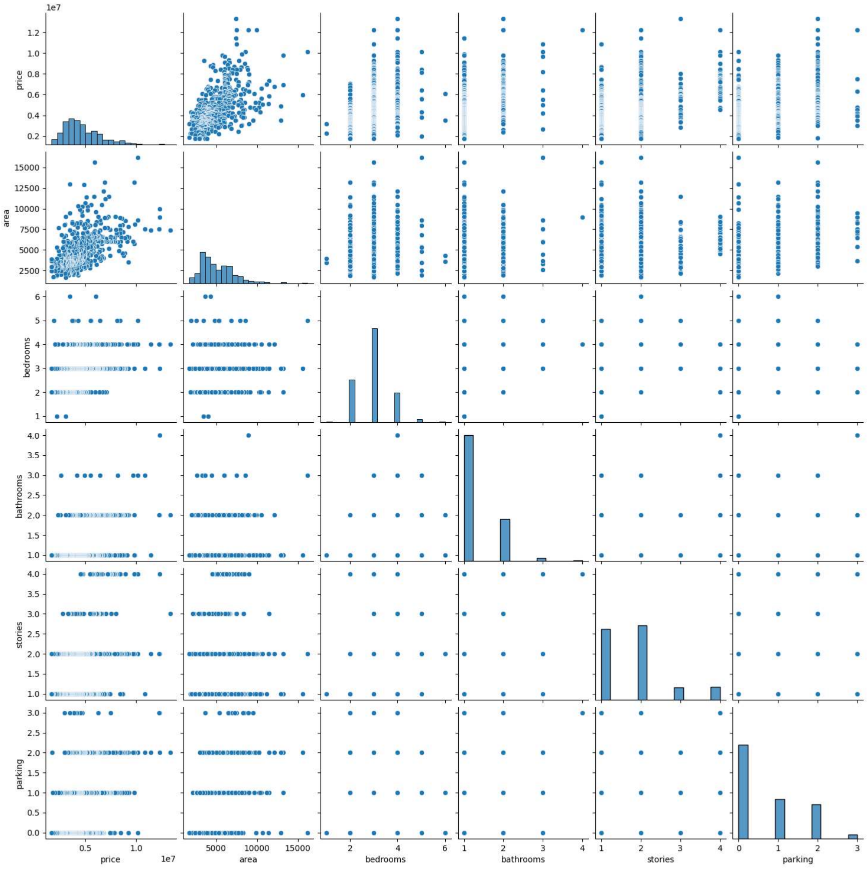
```
corr=x.corr()
sns.heatmap(corr)
plt.show()
# Area has no min connection with stories, parking and stories are also least connected.thus area, stories and parking are not connectec
# but in refernce with the customer it needs to be considered,
# Price, area and bathrroms are having positive correlation among each other.
```

```
<ipython-input-14-a2ade4ad04d6>:1: FutureWarning: The default value of numeric_only i
  corr=x.corr()
```



```
sns.pairplot(data=x)
```

```
<seaborn.axisgrid.PairGrid at 0x783a1b12fca0>
```



```
num=x.select_dtypes(include=['number']).columns
cat=x.select_dtypes(include=['object','category']).columns
print('Cat:',cat)
print('num:',num)

# cat shows the categorical data and num shows the numerical dataset
```

```
    Cat: Index(['mainroad', 'guestroom', 'basement', 'hotwaterheating',
           'airconditioning', 'prefarea', 'furnishingstatus'],
          dtype='object')
    num: Index(['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'parking'], dtype='object')
```

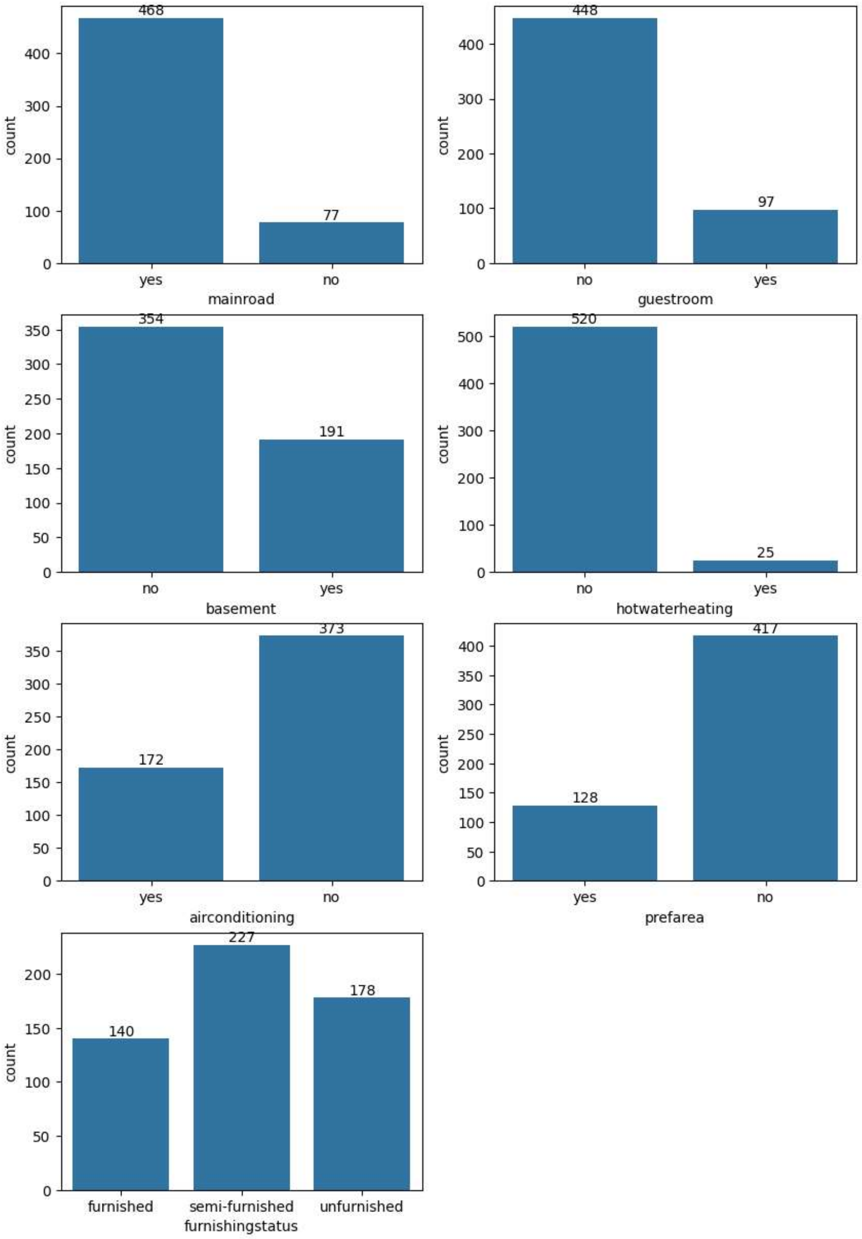
```
cat
```

```
    Index(['mainroad', 'guestroom', 'basement', 'hotwaterheating',
           'airconditioning', 'prefarea', 'furnishingstatus'],
          dtype='object')
```

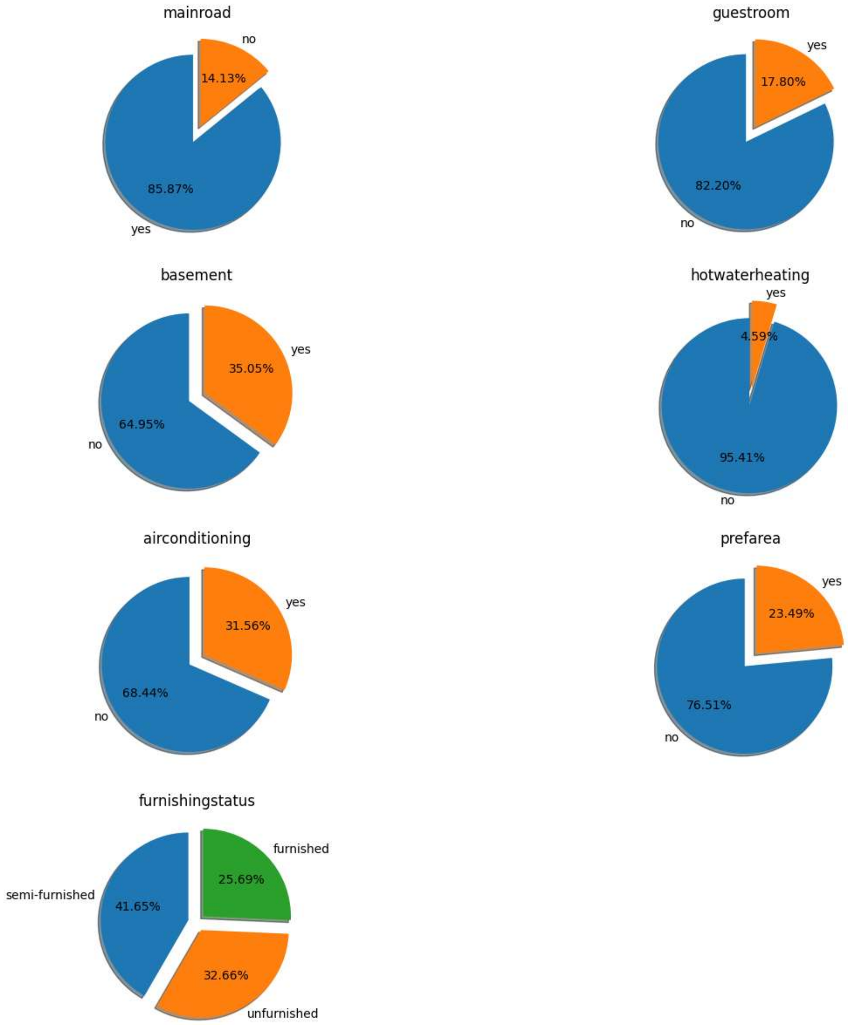
```
for i in cat:
  a=x[i].unique()
  print(i,a)

# the output for categorical value is binary like yes or no.
```

```
    mainroad ['yes' 'no']
    guestroom ['no' 'yes']
    basement ['no' 'yes']
    hotwaterheating ['no' 'yes']
    airconditioning ['yes' 'no']
    prefarea ['yes' 'no']
    furnishingstatus ['furnished' 'semi-furnished' 'unfurnished']
```

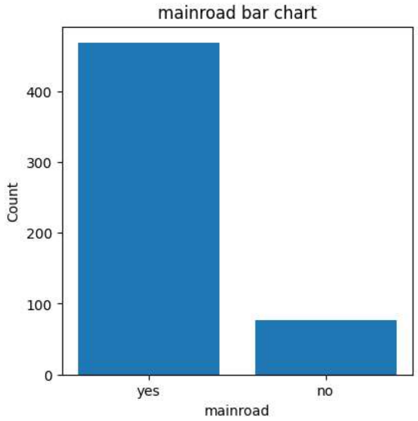
```
plt.figure(figsize=(10,15))
for i,column in enumerate(cat):
  plt.subplot(4,2,i+1)
  ax=sns.countplot(data=x,x=column)
  ax.bar_label(ax.containers[0])
plt.show()

# having the count of categorical dataset individually
```

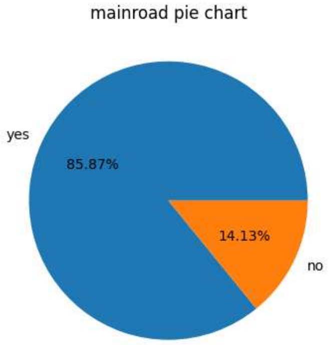


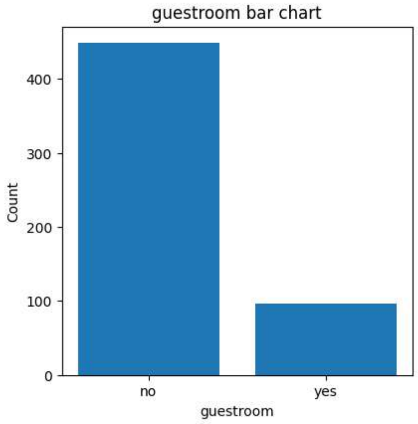
```
plt.figure(figsize=(15,15))
for i, column in enumerate(cat):
  plt.subplot(4,2,i+1)
  x[column].value_counts()
  House=x[column].value_counts(normalize=True).keys()
  count=x[column].value_counts(normalize=True).values
  Data=pd.DataFrame(zip(House,count),columns=[column,'count'])
  n=x[column].nunique()
  l=[0.1 for i in range(n)]
  plt.title(column)
  plt.pie(x=count,labels=House,autopct='%0.2f%%',shadow=True,radius=1,startangle=90,explode=l)
plt.show()

# the outcome in the form of pie chart for categorical value and it's output
```

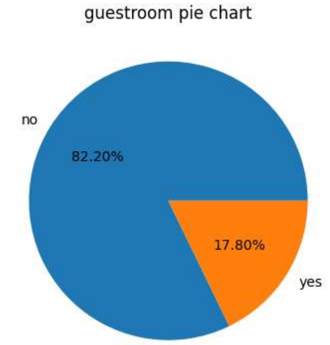


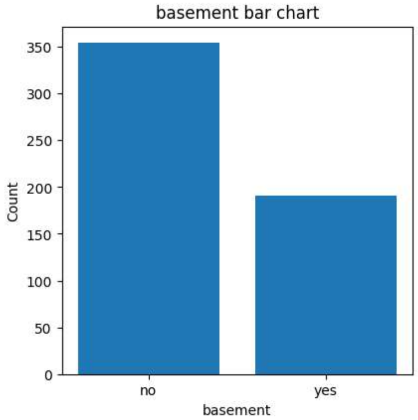
```
for column in cat:

    x[column].value_counts()
    House=x[column].value_counts().keys()
    count=x[column].value_counts().values
    Data=pd.DataFrame(zip(House,count),columns=[column,'Count'])
    Data

    plt.figure(figsize=(10,10))

    plt.subplot(2,2,1)
    plt.title(f'{column} bar chart')
    plt.bar(column,'Count',data=Data)

    plt.xlabel(column)
    plt.ylabel('Count')


    plt.subplot(2,2,2)
    plt.title(f'{column} pie chart')
    plt.pie(x=count,labels=House,autopct='%0.2f%%')

    plt.show()
```

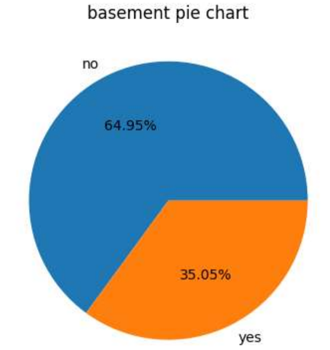

mainroad bar chart


mainroad pie chart


guestroom bar chart


guestroom pie chart


basement bar chart


basement pie chart

hotwaterheating bar chart hotwaterheating pie chart

```
for i in num:
    d_count=round(x[i].count(),2)
    d_max=round(x[i].max(),2)
    d_min=round(x[i].min(),2)
    d_mean=round(x[i].mean(),2)
    d_median=round(x[i].median(),2)
    d_std=round(x[i].std(),2)

    print(i,'count:',d_count)
    print(i,'max:',d_max)
    print(i,'min:',d_min)
    print(i,'mean:',d_mean)
    print(i,'median:',d_median)
    print(i,'std:',d_std)
    print('------------------------------------')

# getting the summary of the data of numerical values
```

```
price count: 545
price max: 13300000
price min: 1750000
price mean: 4766729.25
price median: 4340000.0
price std: 1870439.62
------------------------------------
area count: 545
area max: 16200
area min: 1650
area mean: 5150.54
area median: 4600.0
```

```
     area std: 2170.14
     ------------------------------------
     bedrooms count: 545
     bedrooms max: 6
     bedrooms min: 1
     bedrooms mean: 2.97
     bedrooms median: 3.0
     bedrooms std: 0.74
     ------------------------------------
     bathrooms count: 545
     bathrooms max: 4
     bathrooms min: 1
     bathrooms mean: 1.29
     bathrooms median: 1.0
     bathrooms std: 0.5
     ------------------------------------
     stories count: 545
     stories max: 4
     stories min: 1
     stories mean: 1.81
     stories median: 2.0
     stories std: 0.87
     ------------------------------------
     parking count: 545
     parking max: 3
     parking min: 0
     parking mean: 0.69
     parking median: 0.0
     parking std: 0.86
     ------------------------------------
```

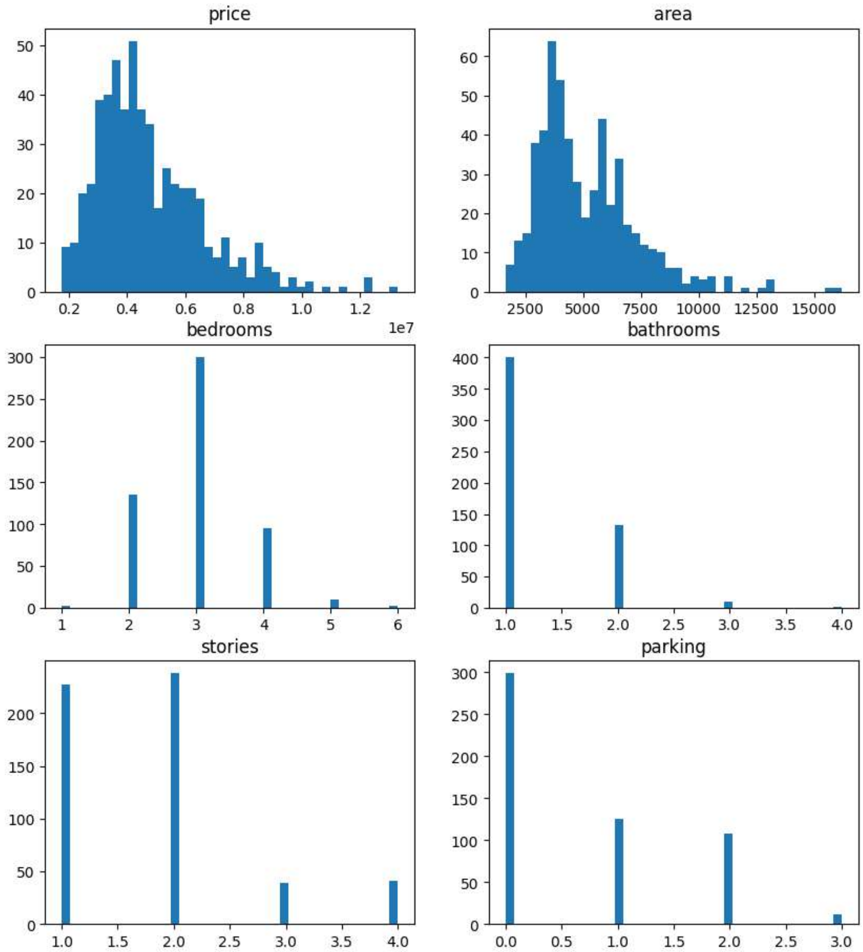
```
for i in num:
    q1=np.quantile(x[i],0.25)
    q2=np.quantile(x[i],0.50)
    q3=np.quantile(x[i],0.75)
    print(i,'q1:',q1)
    print(i,'q2:',q2)
    print(i,'q3:',3)
    print('------------')

# the number in terms of the quartile
```

```
     price q1: 3430000.0
     price q2: 4340000.0
     price q3: 3
     ------------
     area q1: 3600.0
     area q2: 4600.0
     area q3: 3
     ------------
     bedrooms q1: 2.0
     bedrooms q2: 3.0
     bedrooms q3: 3
     ------------
     bathrooms q1: 1.0
     bathrooms q2: 1.0
     bathrooms q3: 3
     ------------
     stories q1: 1.0
     stories q2: 2.0
     stories q3: 3
     ------------
     parking q1: 0.0
     parking q2: 0.0
     parking q3: 3
     ------------
```

```
x.describe()

# the statistical summary of all numerical columns
```

	price	area	bedrooms	bathrooms	stories	parking
count	5.450000e+02	545.000000	545.000000	545.000000	545.000000	545.000000
mean	4.766729e+06	5150.541284	2.965138	1.286239	1.805505	0.693578
std	1.870440e+06	2170.141023	0.738064	0.502470	0.867492	0.861586
min	1.750000e+06	1650.000000	1.000000	1.000000	1.000000	0.000000
25%	3.430000e+06	3600.000000	2.000000	1.000000	1.000000	0.000000
50%	4.340000e+06	4600.000000	3.000000	1.000000	2.000000	0.000000
75%	5.740000e+06	6360.000000	3.000000	2.000000	2.000000	1.000000
max	1.330000e+07	16200.000000	6.000000	4.000000	4.000000	3.000000

```
plt.figure(figsize=(10,15))
for i,column in enumerate(num):
    plt.subplot(4,2,i+1)
    plt.title(column)
    plt.hist(x[column],bins=40)
plt.show()
```

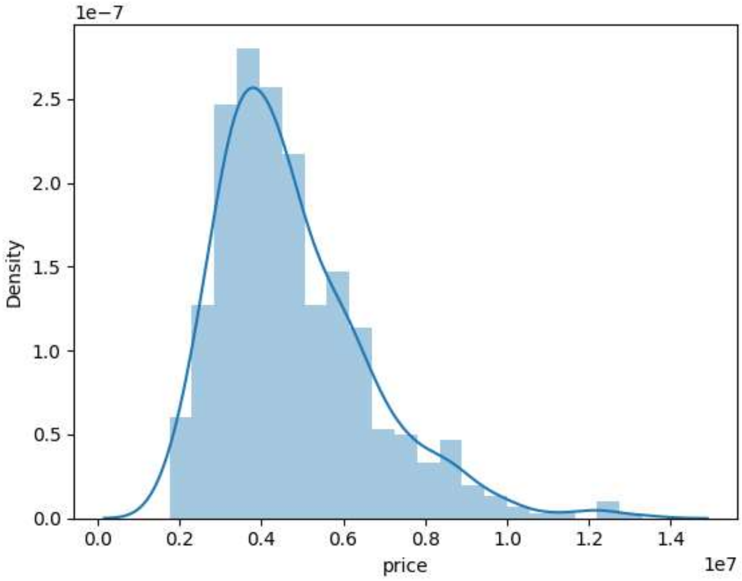


```
num
```

```
Index(['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'parking'], dtype='object')
```

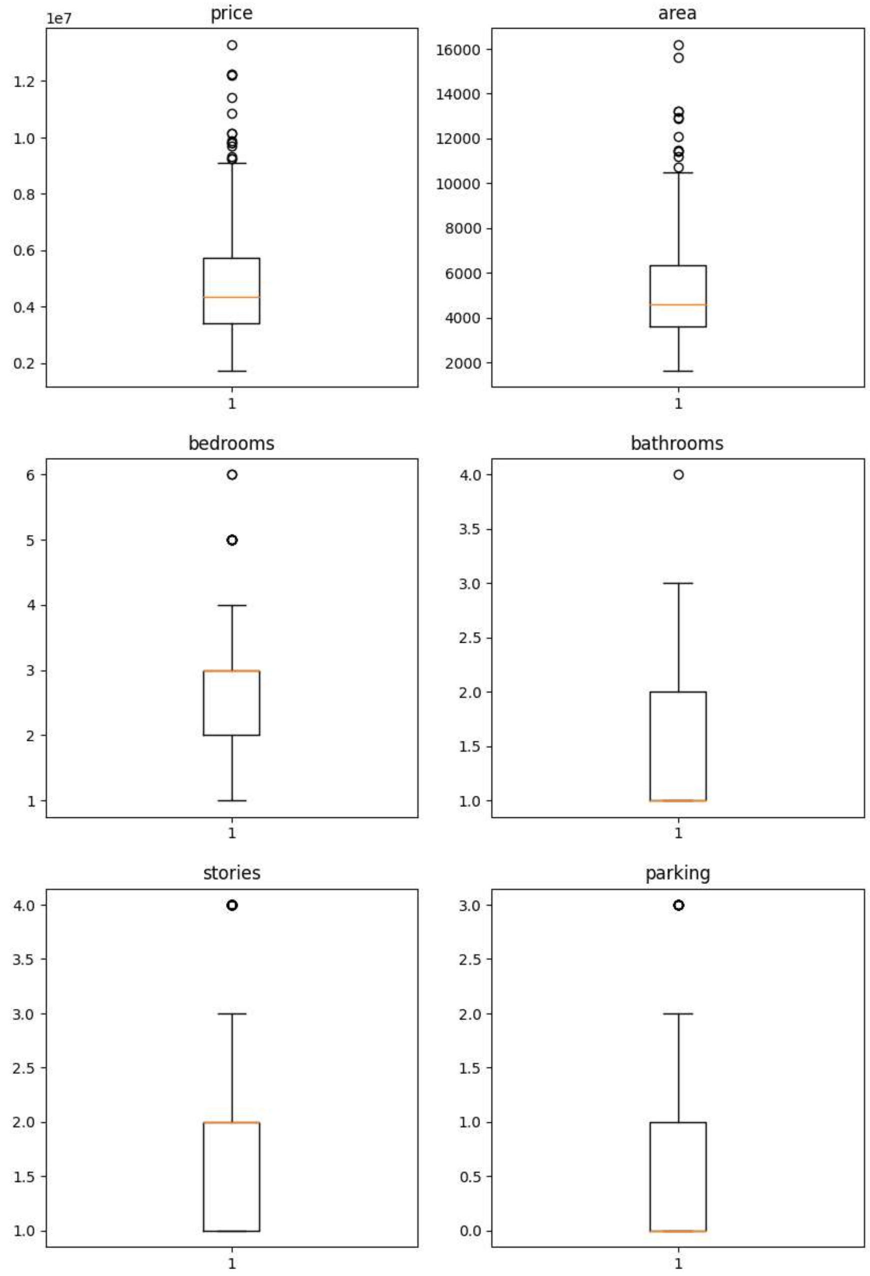
```
for i in num:
  sns.distplot(x[i])
  plt.show()
```

```
<ipython-input-26-b8e3ec97e582>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(x[i])
```



```
<ipython-input-26-b8e3ec97e582>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(x[i])
```

```
plt.figure(figsize=(10,15))
for i, column in enumerate(num):
  plt.subplot(3,2,i+1)
  plt.title(column)
  plt.boxplot(x[column])
plt.show()

# by ploting the box plot we can come to the outliers and the data which is not fitting the statistical model
```



```
num
```

```
    Index(['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'parking'], dtype='object')
```

```
q1=np.quantile(x['price'],0.25)
q2=np.quantile(x['price'],0.50)
q3=np.quantile(x['price'],0.75)

iqr=q3-q1

ub1=q3+(1.5*iqr)
lb1=q1-(1.5*iqr)

con1=x['price']>ub1
con2=x['price']<lb1
outlier=x[con1|con2]
len(outlier)

# length of the outlier in price data  is 15
```

```
    15
```

```
q1=np.quantile(x['price'],0.25)
q2=np.quantile(x['price'],0.50)
q3=np.quantile(x['price'],0.75)

iqr=q3-q1

ub1=q3+(1.5*iqr)
lb1=q1-(1.5*iqr)

con1=x['price']<ub1
con2=x['price']>lb1
non_outlier=x[con1&con2]
len(non_outlier)

# length of the non-outlier data is 530 of prcie data
```

```
530
```

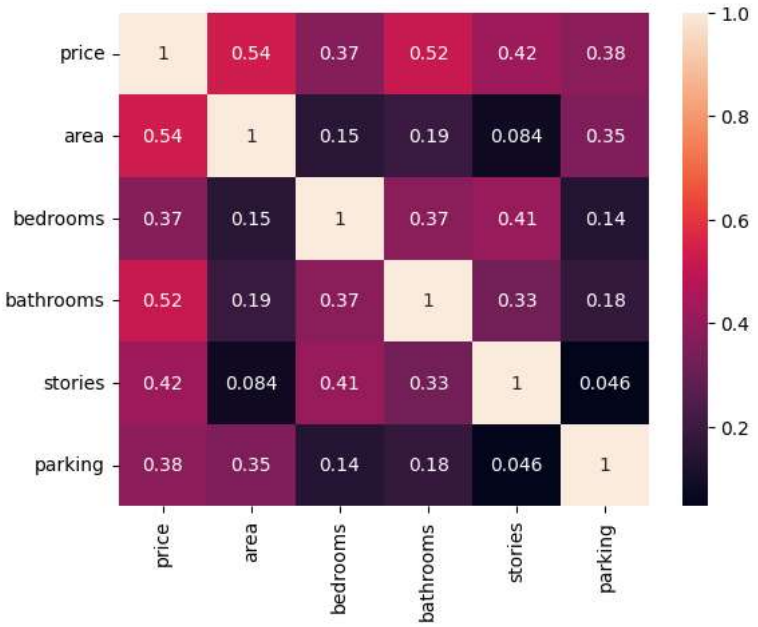
```
## percentage of outlier
v=(len(outlier)/len(x))*100
v
```

```
2.7522935779816518
```

```
corr=x.corr(numeric_only=True)
corr

# correlation of only numeric data, thus it can be concluded that price is co-related to area and number of bathrooms the house is havir
```

	price	area	bedrooms	bathrooms	stories	parking
price	1.000000	0.535997	0.366494	0.517545	0.420712	0.384394
area	0.535997	1.000000	0.151858	0.193820	0.083996	0.352980
bedrooms	0.366494	0.151858	1.000000	0.373930	0.408564	0.139270
bathrooms	0.517545	0.193820	0.373930	1.000000	0.326165	0.177496
stories	0.420712	0.083996	0.408564	0.326165	1.000000	0.045547
parking	0.384394	0.352980	0.139270	0.177496	0.045547	1.000000

```
sns.heatmap(corr,annot=True)
plt.show()
```



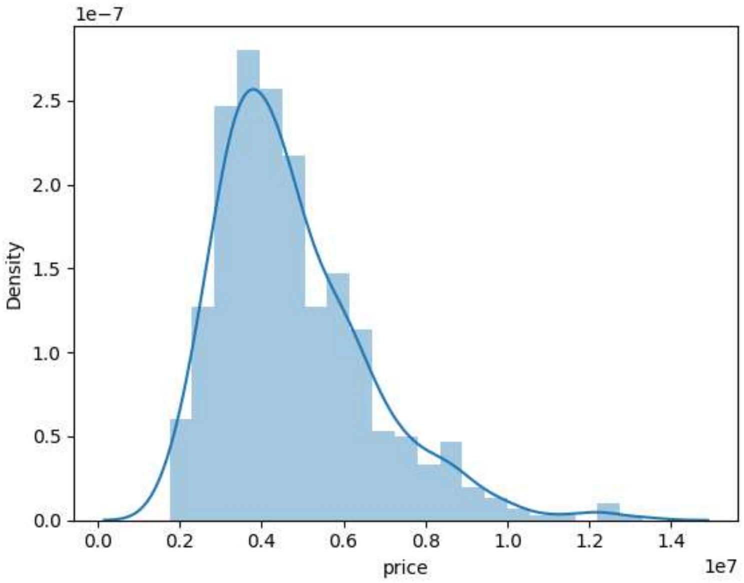
```
for i in (num):
  print(i,x[i].skew())

# the skewness of the data i.e bedrooms and parking data is less skewed
```

```
price 1.2122388370279802
area 1.321188343153483
bedrooms 0.495683940745534473
bathrooms 1.5892635781317528
```

```
        stories 1.0820882904085742
        parking 0.8420623343734072
```

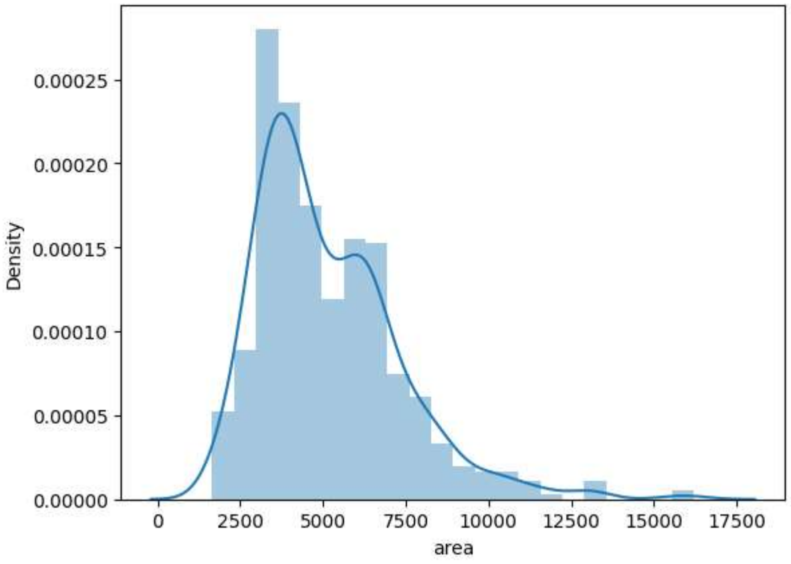
```
sns.distplot(x['price'])
plt.show()
```

```
        <ipython-input-33-ab3e3da2deee>:1: UserWarning:

        `distplot` is a deprecated function and will be removed in seaborn v0.14.0.

        Please adapt your code to use either `displot` (a figure-level function with
        similar flexibility) or `histplot` (an axes-level function for histograms).

        For a guide to updating your code to use the new functions, please see
        https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

          sns.distplot(x['price'])
```



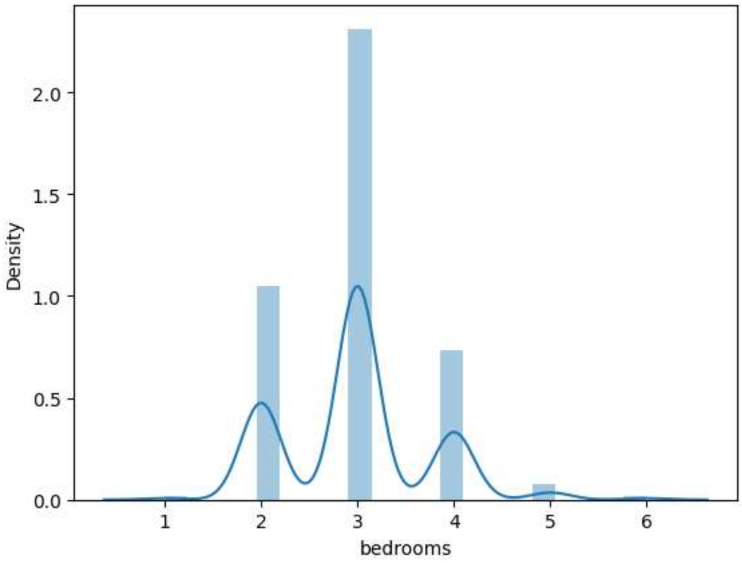
```
sns.distplot(x['area'])
plt.show()
```

```
        <ipython-input-34-70b7c7e6feee>:1: UserWarning:

        `distplot` is a deprecated function and will be removed in seaborn v0.14.0.

        Please adapt your code to use either `displot` (a figure-level function with
        similar flexibility) or `histplot` (an axes-level function for histograms).

        For a guide to updating your code to use the new functions, please see
        https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

          sns.distplot(x['area'])
```



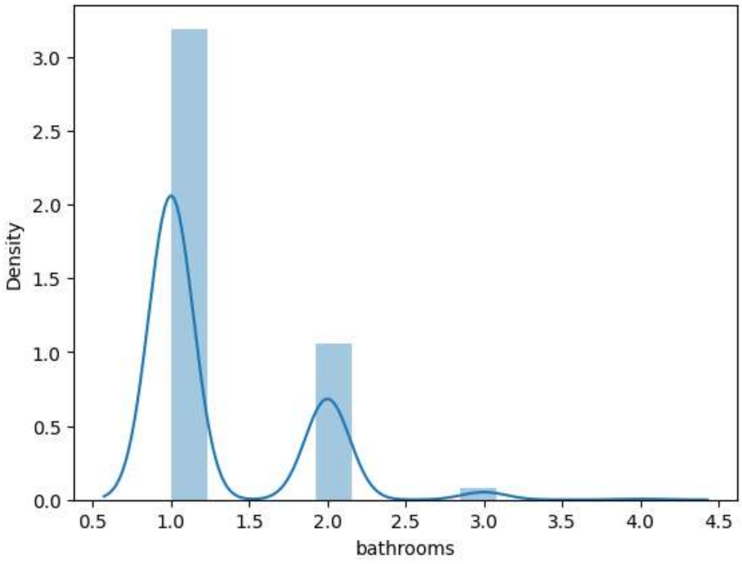
```
sns.distplot(x['bedrooms'])
plt.show()
```

```
<ipython-input-35-ea476cb18230>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(x['bedrooms'])
```



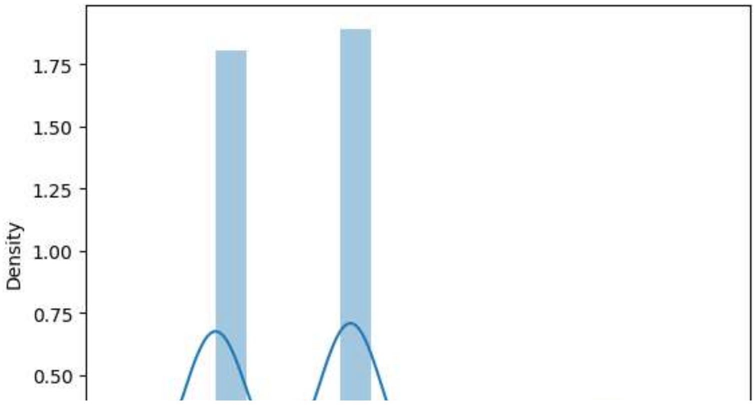
```
sns.distplot(x['bathrooms'])
plt.show()
```

```
<ipython-input-36-5ad89b65b643>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(x['bathrooms'])
```



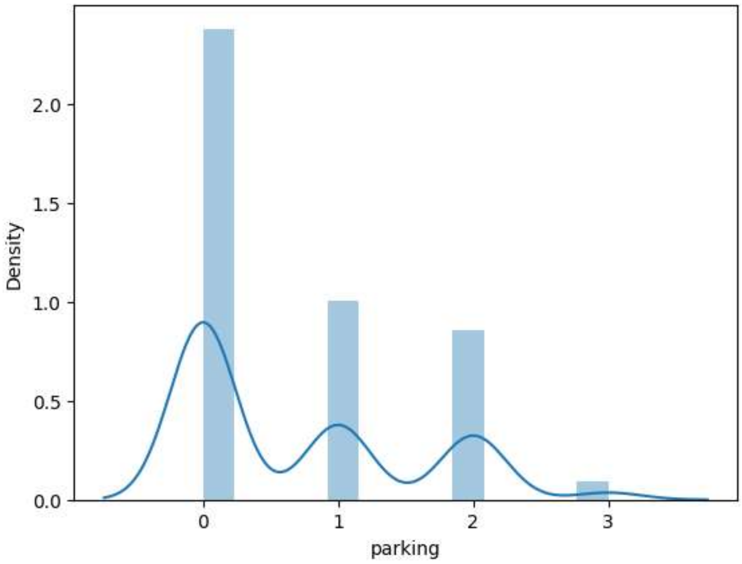
```
sns.distplot(x['stories'])
plt.show()
```

```
<ipython-input-39-a58b139c597c>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(x['stories'])
```



```
sns.distplot(x['parking'])
plt.show()
```

```
<ipython-input-37-8b3abc9240e7>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(x['parking'])
```


Density
parking