Assignment week - 3 of Applied Data Science

Submitted by - Rahul Kumar , Registration number - 20BEC1186

Mail ID - rahulkumar2020a@vitstudent.ac.in (mailto:rahulkumar2020a@vitstudent.ac.in)

Question 1:

1. Download the dataset: Dataset

Question 2:

Load the Dataset into the tool

In [1]:
```
import pandas as pd
data = pd.read_csv("Housing.csv")
data
```

Out[1]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	pa
0	13300000	7420	4	2	3	yes	no	no	no	yes	
1	12250000	8960	4	4	4	yes	no	no	no	yes	
2	12250000	9960	3	2	2	yes	no	yes	no	no	
3	12215000	7500	4	2	2	yes	no	yes	no	yes	
4	11410000	7420	4	1	2	yes	yes	yes	no	yes	
...	...	...	...	...	...	...	...	...	...	...	
540	1820000	3000	2	1	1	yes	no	yes	no	no	
541	1767150	2400	3	1	1	no	no	no	no	no	
542	1750000	3620	2	1	1	yes	no	no	no	no	
543	1750000	2910	3	1	1	no	no	no	no	no	
544	1750000	3850	3	1	2	yes	no	no	no	no	

545 rows × 12 columns

Question 3:

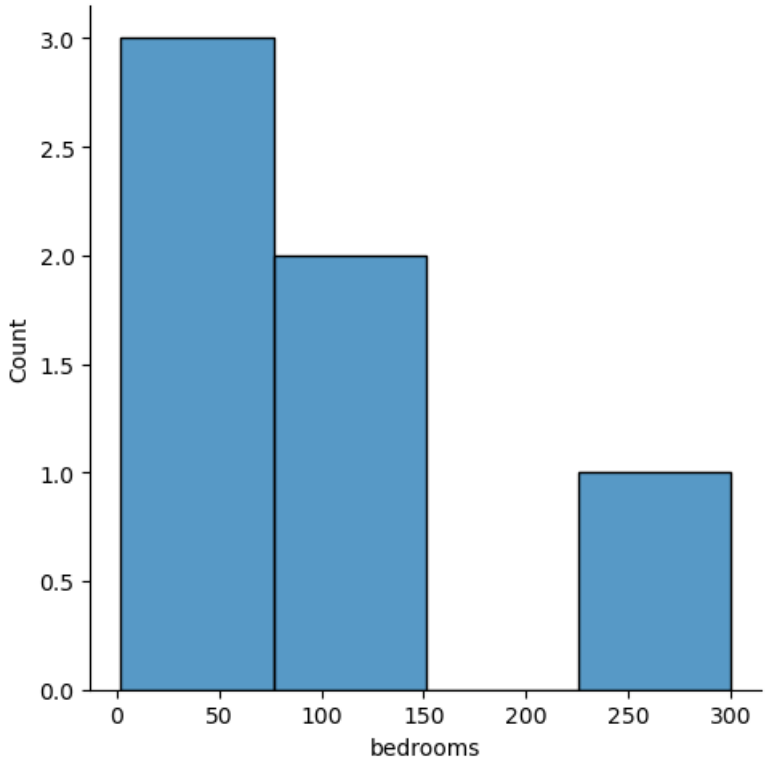
3. Perform below Visualizations.

• Univariate Analysis

In [26]:
```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
sns.displot(data.bedrooms.value_counts())
```

Out[26]:
```
<seaborn.axisgrid.FacetGrid at 0x21b251c7220>
```

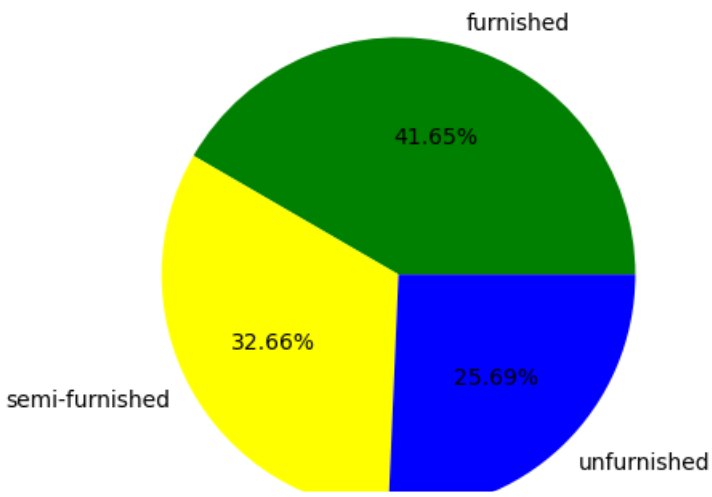
In [27]:
```
plt.show()
```



In [29]:
```
plt.pie(data.furnishingstatus.value_counts(),colors=['green','yellow','blue'],labels=['furnished','
```

Out[29]:
```
([<matplotlib.patches.Wedge at 0x21b2401a340>,
  <matplotlib.patches.Wedge at 0x21b2401adc0>,
  <matplotlib.patches.Wedge at 0x21b2400fa00>],
 [Text(0.28521128309432414, 1.0623815340995388, 'furnished'),
  Text(-0.9645476294288756, -0.5288174264934321, 'semi-furnished'),
  Text(0.7608233961924185, -0.7944480850289933, 'unfurnished')],
 [Text(0.15556979077872224, 0.5794808367815666, '41.65%'),
  Text(-0.5261168887793867, -0.2884458689964175, '32.66%'),
  Text(0.4149945797413191, -0.43333531910672357, '25.69%')])
```

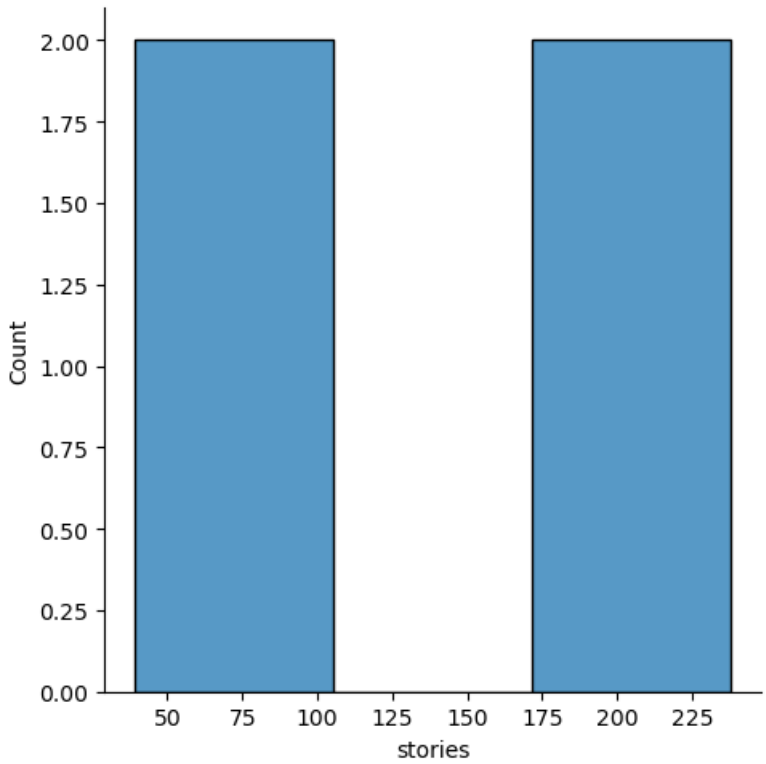
In [30]:
```
plt.show()
```



In [21]:
```
sns.displot(data.stories.value_counts())
```

Out[21]:
```
<seaborn.axisgrid.FacetGrid at 0x21b250e4370>
```

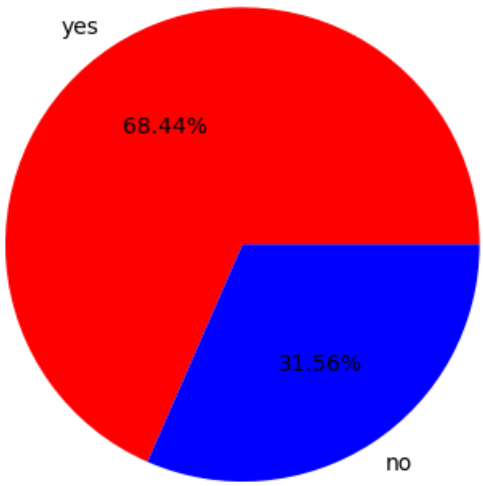
In [22]:
```
plt.show()
```



In [31]:
```
plt.pie(data.airconditioning.value_counts(),colors=['red','blue'],labels=['yes','no'],autopct='%.2f
```

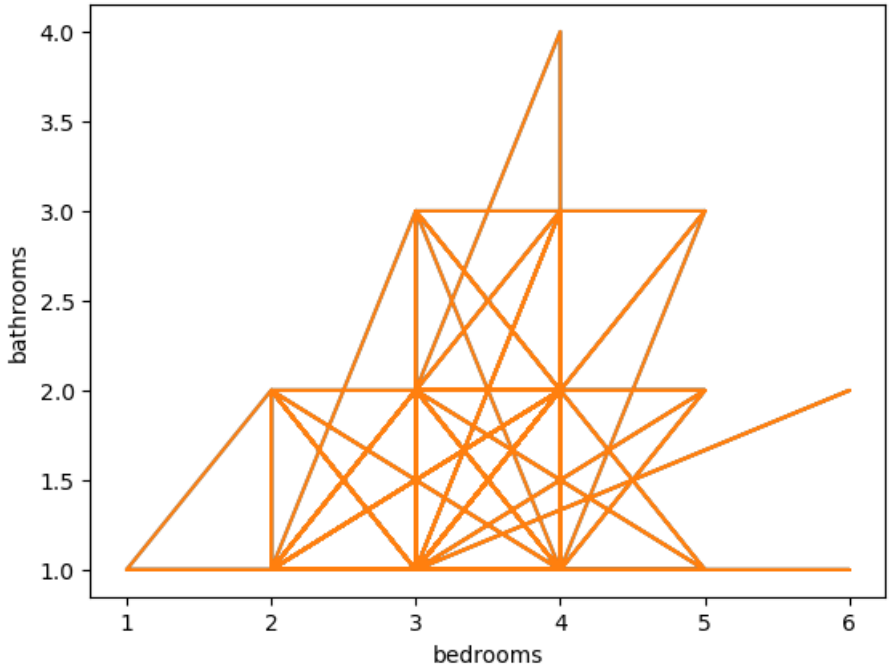
Out[31]:
```
([<matplotlib.patches.Wedge at 0x21b262d6b20>,
  <matplotlib.patches.Wedge at 0x21b262d6e20>],
 [Text(-0.6022016005293036, 0.9205179152628944, 'yes'),
  Text(0.6022016005293035, -0.9205179152628945, 'no')],
 [Text(-0.32847360028871103, 0.5021006810524877, '68.44%'),
  Text(0.3284736002887109, -0.5021006810524878, '31.56%')])
```

In [32]:
```
plt.show()
```

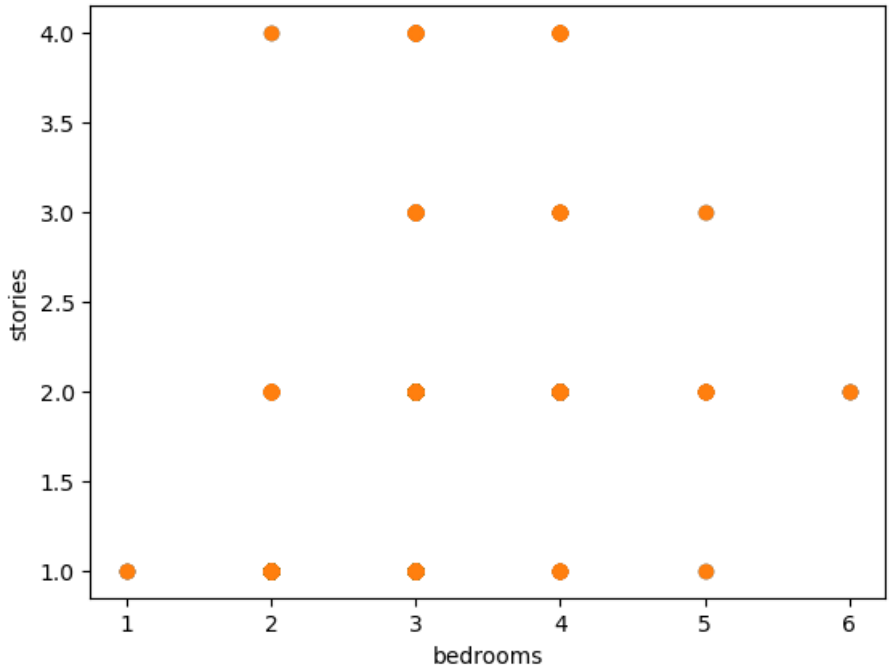


- **Bivariate Analysis**

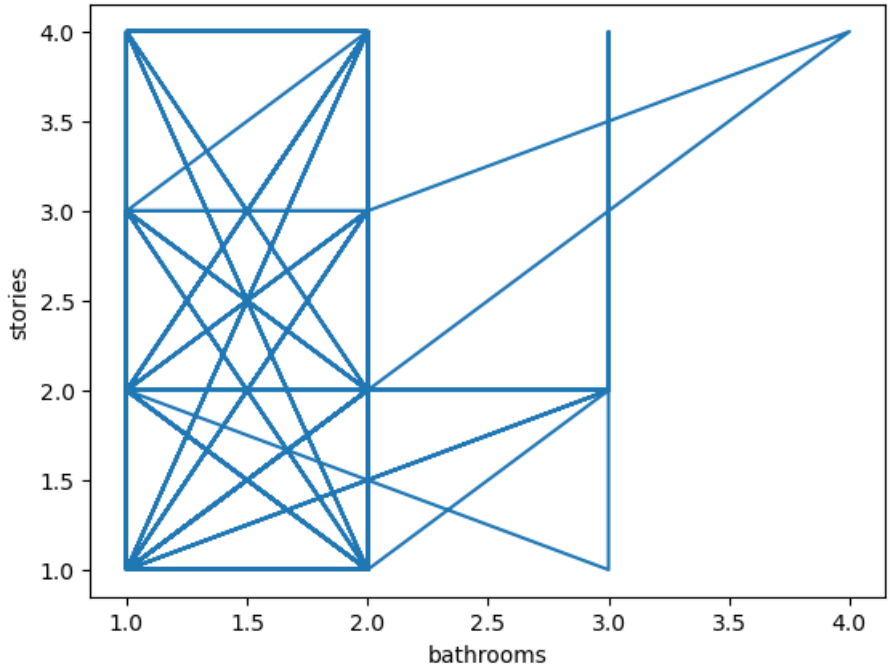
In [34]:
```
plt.plot(data['bedrooms'],data['bathrooms'])
plt.xlabel('bedrooms')
plt.ylabel('bathrooms')
plt.show()
```



In [37]:
```
plt.scatter(data['bedrooms'],data['stories'])
plt.xlabel('bedrooms')
plt.ylabel('stories')
plt.show()
```

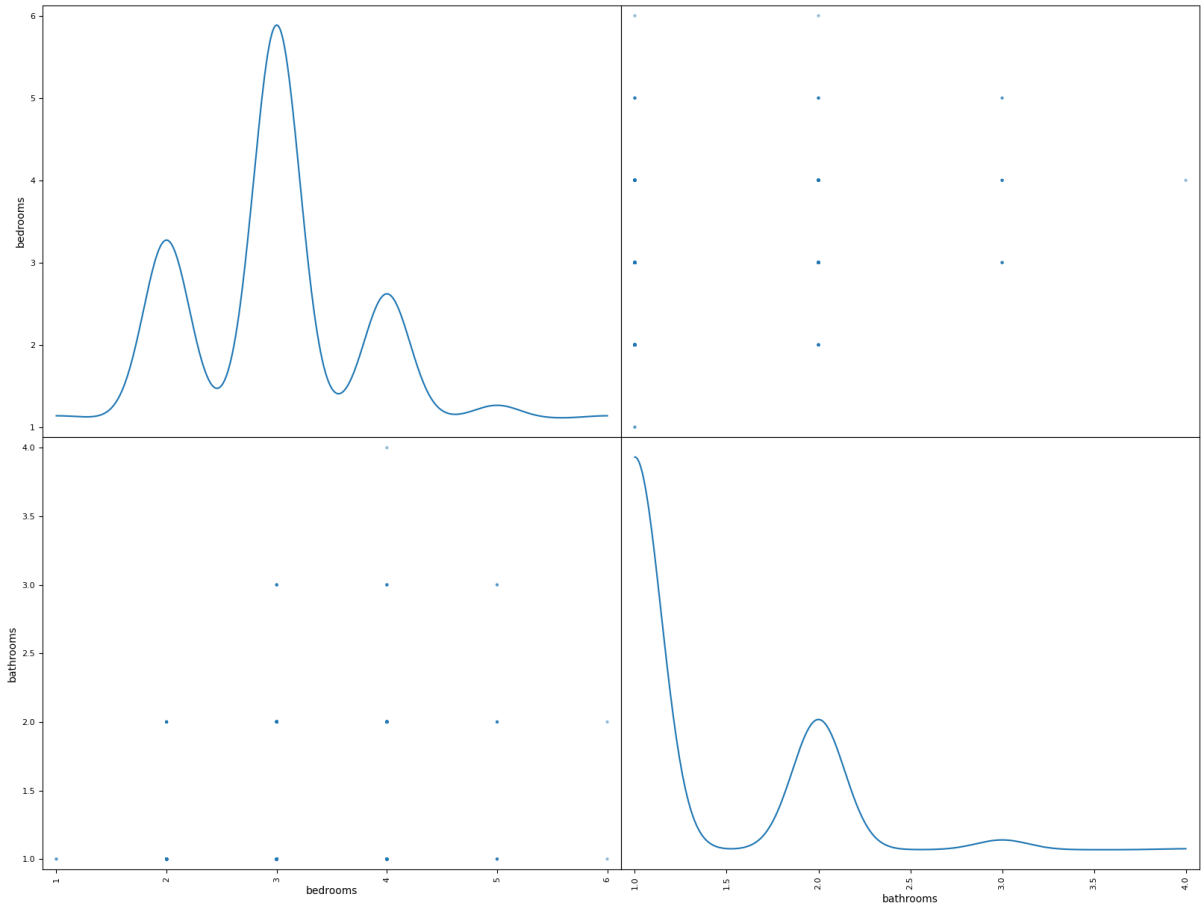


In [38]:
```
plt.plot(data['bathrooms'],data['stories'])
plt.xlabel('bathrooms')
plt.ylabel('stories')
plt.show()
```

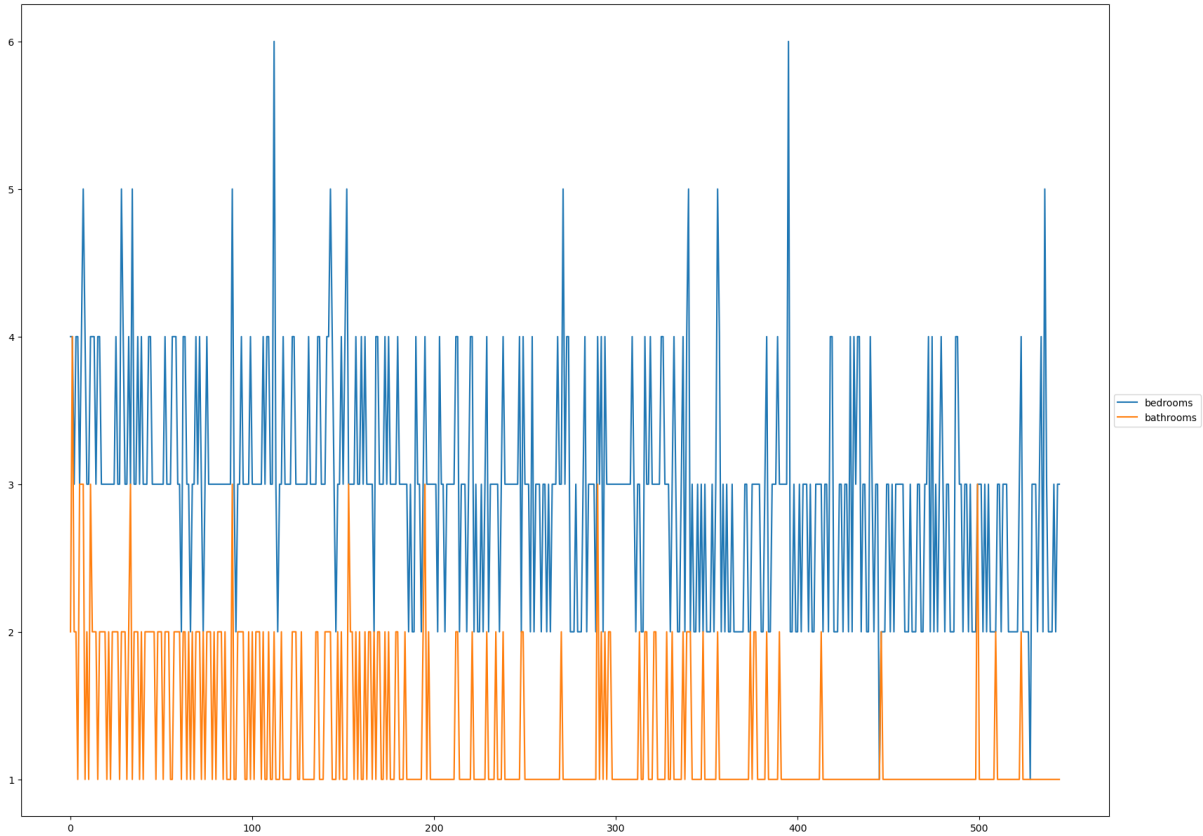


• Multivariate Analysis

In [39]:
```
pd.plotting.scatter_matrix(data.loc[:, 'bedrooms':'bathrooms'], diagonal="kde",figsize=(20,15))
plt.show()
```



In [42]:
```
ax = data[["bedrooms","bathrooms"]].plot(figsize=(20,15))
ax.legend(loc='center left', bbox_to_anchor=(1, 0.5));
plt.show()
```



4. Perform descriptive statistics on the dataset.

In [43]:
```
# to describe the data set
data.describe()
```

Out[43]:

	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

In [46]:
```
#the variance of the data set the
data.var(numeric_only = True )
```

Out[46]:
```
price        3.498544e+12
area         4.709512e+06
bedrooms     5.447383e-01
bathrooms    2.524757e-01
stories      7.525432e-01
parking      7.423300e-01
dtype: float64
```

In [48]:
```
#the mean for the data
data.mean(numeric_only = True)
```

Out[48]:
```
price        4.766729e+06
area         5.150541e+03
bedrooms     2.965138e+00
bathrooms    1.286239e+00
stories      1.805505e+00
parking      6.935780e-01
dtype: float64
```

In [49]:
```
# median and mode for the dataset and for the numeric only
data.median(numeric_only = True)
```

Out[49]:
```
price        4340000.0
area            4600.0
bedrooms           3.0
bathrooms          1.0
stories            2.0
parking            0.0
dtype: float64
```

In [50]:
```
data.mode()
```

Out[50]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	parl
0	3500000	6000.0	3.0	1.0	2.0	yes	no	no	no	no	
1	4200000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

In [51]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 12 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  furnishingstatus  545 non-null    object
dtypes: int64(6), object(6)
memory usage: 51.2+ KB
```

5. Check for Missing values and deal with them.

In [52]:
```
data.isnull().any()
```

Out[52]:
```
price               False
area                False
bedrooms            False
bathrooms           False
stories             False
mainroad            False
guestroom           False
basement            False
hotwaterheating     False
airconditioning     False
parking             False
furnishingstatus    False
dtype: bool
```

In [54]:
```
data.isnull().sum()
```

Out[54]:
```
price               0
area                0
bedrooms            0
bathrooms           0
stories             0
mainroad            0
guestroom           0
basement            0
hotwaterheating     0
airconditioning     0
parking             0
furnishingstatus    0
dtype: int64
```

Hence there is no null space in the data set so we dont need to fill the space

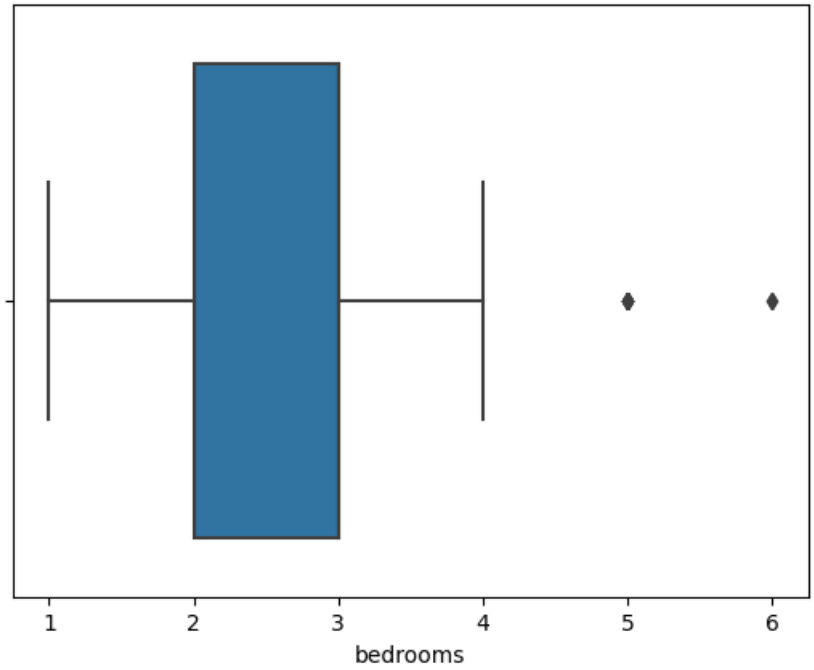
6. Find the outliers and replace them outliers

In [60]:
```
sns.boxplot(data.bedrooms)
```

```
C:\Users\Dell\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the foll
owing variable as a keyword arg: x. From version 0.12, the only valid positional argument will be
`data`, and passing other arguments without an explicit keyword will result in an error or misint
erpretation.
  warnings.warn(
```

Out[60]:
```
<AxesSubplot:xlabel='bedrooms'>
```

In [61]:
```
plt.show()
```



In [62]:
```
perc99=data.bedrooms.quantile(0.99)
perc99
```

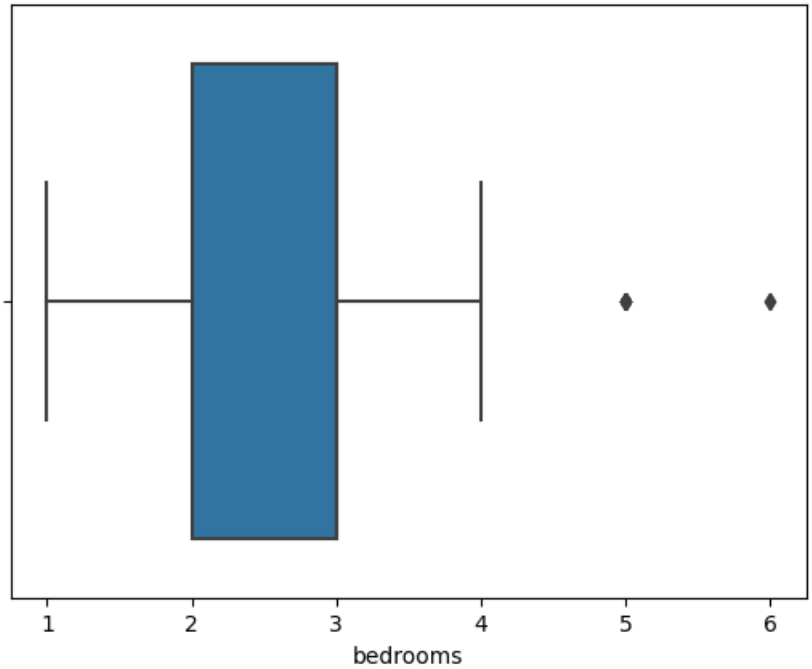
Out[62]:
```
5.0
```

In [63]:
```
data_new = pd.read_csv("Housing.csv")
data_new=data_new[data_new.bedrooms<=perc99]
sns.boxplot(data.bedrooms)
```

```
C:\Users\Dell\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the foll
owing variable as a keyword arg: x. From version 0.12, the only valid positional argument will be
`data`, and passing other arguments without an explicit keyword will result in an error or misint
erpretation.
  warnings.warn(
```

Out[63]:
```
<AxesSubplot:xlabel='bedrooms'>
```

In [64]:
```
plt.show()
```



7. Check for Categorical columns and perform encoding.

In [66]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

In [68]:
```
data.airconditioning=le.fit_transform(data.airconditioning)
```

In [69]:
```
data.guestroom=le.fit_transform(data.guestroom)
```

In [70]:
```
data.mainroad=le.fit_transform(data.mainroad)
```

In [71]:
```
data.basement=le.fit_transform(data.basement)
```

In [72]:
```
data.hotwaterheating=le.fit_transform(data.hotwaterheating)
```

In [73]:
```
data.furnishingstatus=le.fit_transform(data.furnishingstatus)
```

In [74]:
```
data.head()
```

Out[74]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	park
0	13300000	7420	4	2	3	1	0	0	0	1	
1	12250000	8960	4	4	4	1	0	0	0	1	
2	12250000	9960	3	2	2	1	0	1	0	0	
3	12215000	7500	4	2	2	1	0	1	0	1	
4	11410000	7420	4	1	2	1	1	1	0	1	

In [75]:
```
y=data['guestroom']
```

In [76]:
```
y.head()
```

Out[76]:
```
0    0
1    0
2    0
3    0
4    1
Name: guestroom, dtype: int32
```

In [77]:
```
X=data.drop(columns=['guestroom'],axis=1)
```

In [78]:
```
X.head()
```

Out[78]:

	price	area	bedrooms	bathrooms	stories	mainroad	basement	hotwaterheating	airconditioning	parking	furnishi
0	13300000	7420	4	2	3	1	0	0	1	2	
1	12250000	8960	4	4	4	1	0	0	1	3	
2	12250000	9960	3	2	2	1	1	0	0	2	
3	12215000	7500	4	2	2	1	1	0	1	3	
4	11410000	7420	4	1	2	1	1	0	1	2	

8. Split the data into dependent and independent variables.

In [79]:
```
X=data.drop(columns=['guestroom','furnishingstatus','mainroad','basement','hotwaterheating','aircor
```

In [80]:
```
name=X.columns
```

In [81]:
```
name
```

Out[81]:
```
Index(['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'parking'], dtype='object')
```

9. Scale the independent variables

In [83]:
```
from sklearn.preprocessing import MinMaxScaler
scale=MinMaxScaler()
```

In [84]:
```
X_scaled=scale.fit_transform(X)
```

In [85]:
```
X_scaled
```

Out[85]:
```
array([[1.        , 0.39656357, 0.6       , 0.33333333, 0.66666667,
        0.66666667],
       [0.90909091, 0.5024055 , 0.6       , 1.        , 1.        ,
        1.        ],
       [0.90909091, 0.57113402, 0.4       , 0.33333333, 0.33333333,
        0.66666667],
       ...,
       [0.        , 0.13539519, 0.2       , 0.        , 0.        ,
        0.        ],
       [0.        , 0.08659794, 0.4       , 0.        , 0.        ,
        0.        ],
       [0.        , 0.15120275, 0.4       , 0.        , 0.33333333,
        0.        ]])
```

In [86]:
```
X=pd.DataFrame(X_scaled,columns=name)
```

In [87]:
```
X
```

Out[87]:

	price	area	bedrooms	bathrooms	stories	parking
0	1.000000	0.396564	0.6	0.333333	0.666667	0.666667
1	0.909091	0.502405	0.6	1.000000	1.000000	1.000000
2	0.909091	0.571134	0.4	0.333333	0.333333	0.666667
3	0.906061	0.402062	0.6	0.333333	0.333333	1.000000
4	0.836364	0.396564	0.6	0.000000	0.333333	0.666667
...	...	...	...	...	...	...
540	0.006061	0.092784	0.2	0.000000	0.000000	0.666667
541	0.001485	0.051546	0.4	0.000000	0.000000	0.000000
542	0.000000	0.135395	0.2	0.000000	0.000000	0.000000
543	0.000000	0.086598	0.4	0.000000	0.000000	0.000000
544	0.000000	0.151203	0.4	0.000000	0.333333	0.000000

545 rows × 6 columns

10. Split the data into training and testing

In [90]:
```
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=0)
```

In [91]:
```
X=pd.DataFrame(X,columns=name)
```

In [92]:
```
X_train.head()
```

Out[92]:

	price	area	bedrooms	bathrooms	stories	parking
542	0.000000	0.135395	0.2	0.0	0.0	0.0
496	0.081818	0.161512	0.2	0.0	0.0	0.0
484	0.096970	0.095533	0.2	0.0	0.0	0.0
507	0.072727	0.134021	0.2	0.0	0.0	0.0
252	0.239394	0.564261	0.4	0.0	0.0	0.0

In [93]:
```
X_test.head()
```

Out[93]:

	price	area	bedrooms	bathrooms	stories	parking
239	0.245455	0.161512	0.4	0.0	0.333333	0.333333
113	0.375152	0.547766	0.4	0.0	0.000000	0.666667
325	0.195455	0.124399	0.6	0.0	0.333333	0.000000
66	0.448485	0.793814	0.2	0.0	0.000000	0.333333
479	0.103030	0.138144	0.6	0.0	0.333333	0.000000

In [94]:
```
y_train
```

Out[94]:
```
542    0
496    0
484    0
507    0
252    0
      ..
70     0
277    0
9      1
359    0
192    1
Name: guestroom, Length: 436, dtype: int32
```

In [95]:
```
y_test
```

Out[95]:
```
239    0
113    0
325    0
66     0
479    0
      ..
76     0
132    0
311    0
464    0
155    0
Name: guestroom, Length: 109, dtype: int32
```

11. Build the Model

In [96]:
```
y=data["price"]
X=data.drop(columns=["price"],axis=1)
```

In [97]:
```
data.head()
```

Out[97]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioning	park
0	13300000	7420	4	2	3	1	0	0	0	1	
1	12250000	8960	4	4	4	1	0	0	0	1	
2	12250000	9960	3	2	2	1	0	1	0	0	
3	12215000	7500	4	2	2	1	0	1	0	1	
4	11410000	7420	4	1	2	1	1	1	0	1	

In [99]:
```
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.01,random_state=0)
```

In [100]:
```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
```

12. Train the Model

In [101]:
```
model.fit(x_train,y_train)
```

Out[101]:
```
LinearRegression()
```

13. Test the Model

In [102]:
```
pred=model.predict(x_test)
```

In [103]:
```
pred
```

Out[103]:
```
array([4132602.10623296, 5909768.21867566, 4502527.18510612,
       7290885.77184122, 2924314.19759268, 7149365.39640723])
```

In [104]:
```
y_test
```

Out[104]:
```
239    4585000
113    6083000
325    4007500
66     6930000
479    2940000
103    6195000
Name: price, dtype: int64
```

14. Measure the performance using Metrics

In [105]:
```
E=pred-y_test
E
```

Out[105]:
```
239   -452397.893767
113   -173231.781324
325    495027.185106
66     360885.771841
479    -15685.802407
103    954365.396407
Name: price, dtype: float64
```

In [108]:
```
from sklearn.metrics import r2_score , confusion_matrix
R2_Score = r2_score(pred , y_test)*100
```

In [109]:
```
R2_Score
```

Out[109]:
```
90.12083024768475
```