IMPORTING LIBRARIES

In [12]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
from scipy import stats
```

READING FILE USING PANDAS

In [2]:
```
data=pd.read_csv('Housing.csv')
```

head() , tail()

In [3]:
```
top=data.head(10)
top
```

Out[3]:

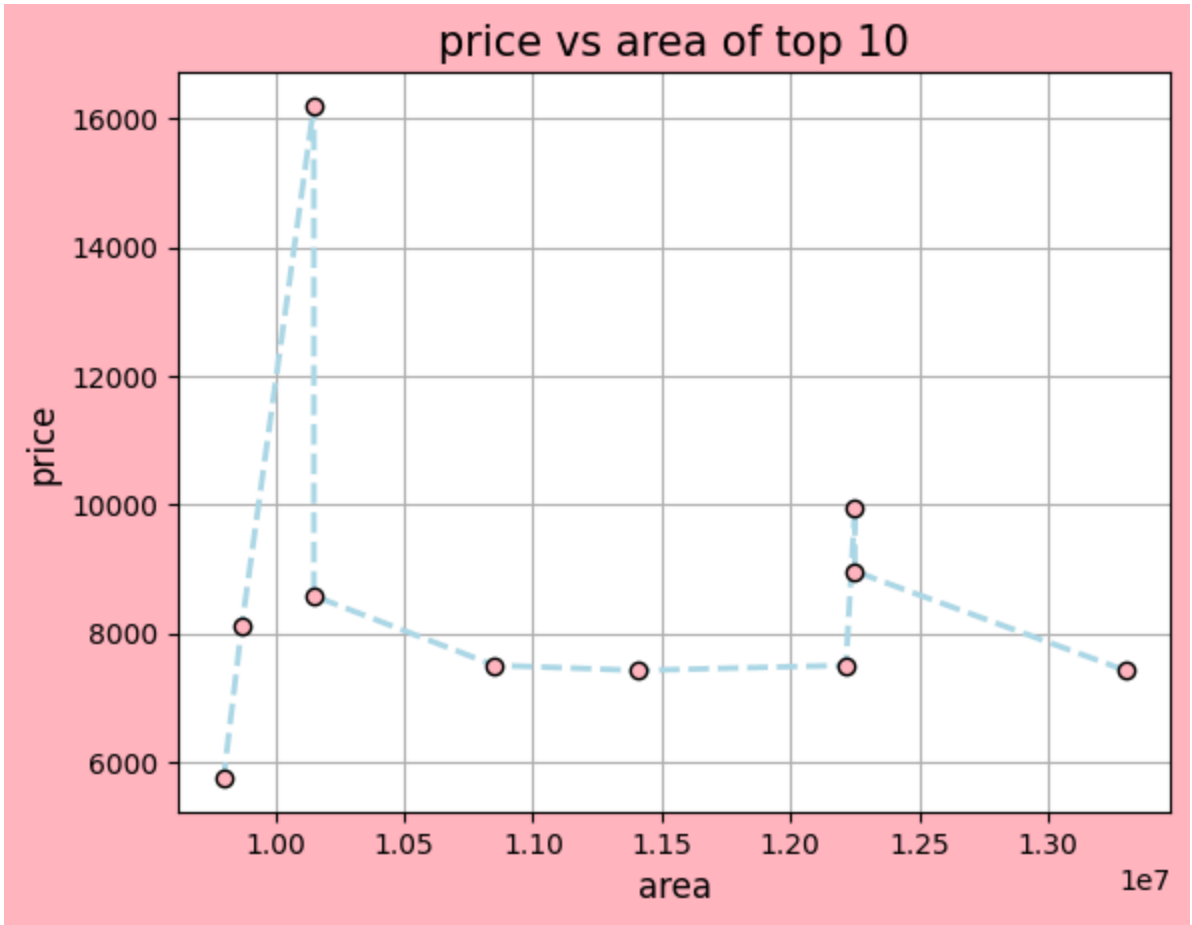
	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airco
0	13300000	7420	4	2	3	yes	no	no	no	
1	12250000	8960	4	4	4	yes	no	no	no	
2	12250000	9960	3	2	2	yes	no	yes	no	
3	12215000	7500	4	2	2	yes	no	yes	no	
4	11410000	7420	4	1	2	yes	yes	yes	no	
5	10850000	7500	3	3	1	yes	no	yes	no	
6	10150000	8580	4	3	4	yes	no	no	no	
7	10150000	16200	5	3	2	yes	no	no	no	
8	9870000	8100	4	1	2	yes	yes	yes	no	
9	9800000	5750	3	2	4	yes	yes	no	no	

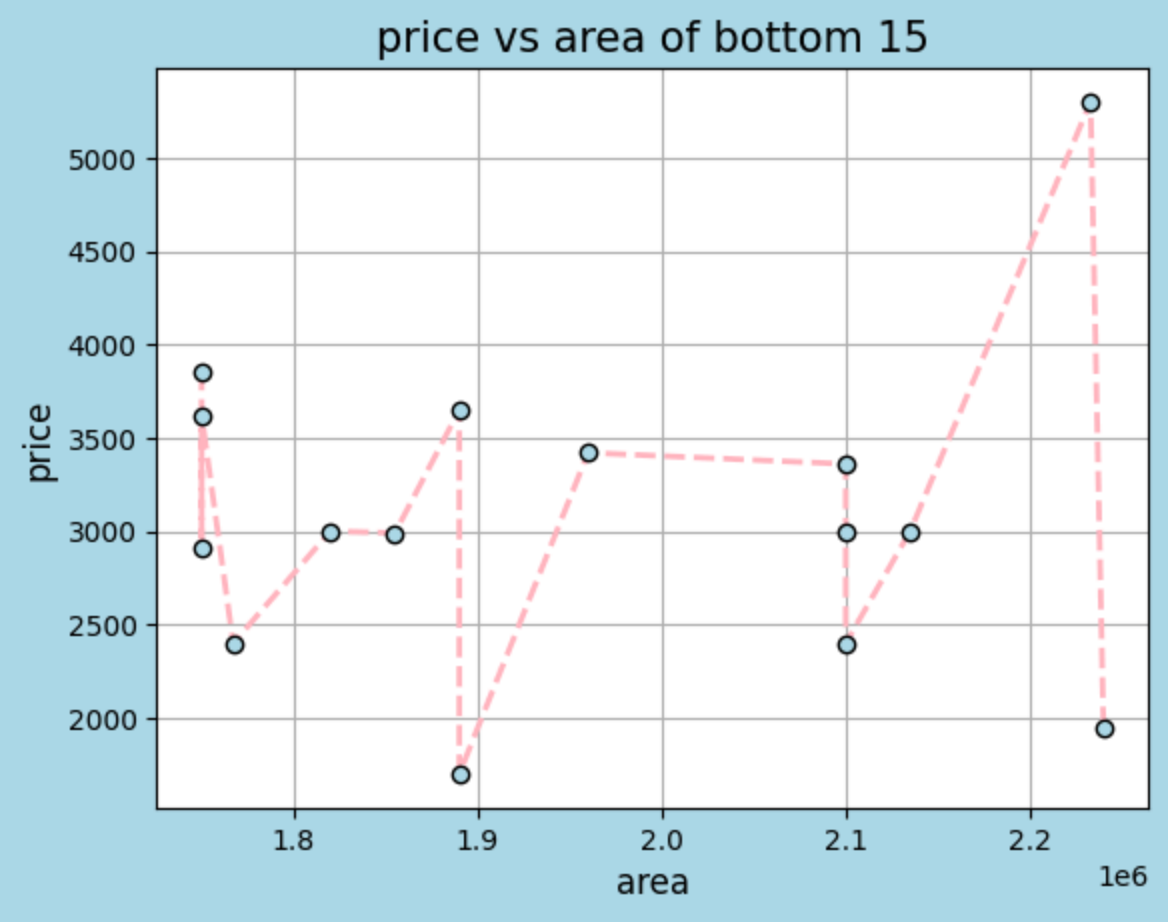
In [436]:
```
bottom=data.tail(15)
bottom
```

Out[436]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airco
530	2240000	1950	3	1	1	no	no	no	yes	
531	2233000	5300	3	1	1	no	no	no	no	
532	2135000	3000	2	1	1	no	no	no	no	
533	2100000	2400	3	1	2	yes	no	no	no	
534	2100000	3000	4	1	2	yes	no	no	no	
535	2100000	3360	2	1	1	yes	no	no	no	
536	1960000	3420	5	1	2	no	no	no	no	
537	1890000	1700	3	1	2	yes	no	no	no	
538	1890000	3649	2	1	1	yes	no	no	no	
539	1855000	2990	2	1	1	no	no	no	no	
540	1820000	3000	2	1	1	yes	no	yes	no	
541	1767150	2400	3	1	1	no	no	no	no	
542	1750000	3620	2	1	1	yes	no	no	no	
543	1750000	2910	3	1	1	no	no	no	no	
544	1750000	3850	3	1	2	yes	no	no	no	

In [437]:
```
x1=top['price']
y1=top['area']
plt.figure(facecolor='lightpink',frameon=True)
plt.plot(x1,y1,
         linewidth=2,color='lightblue',linestyle='--',
         marker='o',mec='k',mfc='lightpink')
plt.xlabel('area',fontsize=12)
plt.ylabel('price',fontsize=12)
plt.title('price vs area of top 10',fontsize=15,color='k')
plt.grid()
plt.show()

x=bottom['price']
y=bottom['area']
plt.figure(facecolor='lightblue',frameon=True)
plt.plot(x,y,
         linewidth=2,color='lightpink',linestyle='--',
         marker='o',mec='k',mfc='lightblue')
plt.xlabel('area',fontsize=12)
plt.ylabel('price',fontsize=12)
plt.title('price vs area of bottom 15',fontsize=15,color='k')
plt.grid()
plt.show()
```





describe(),info() , quantile(), percentile()

In [438]:
```
data.describe()
```

Out[438]:

	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 [18]:
```
data.describe(percentiles=(0.1,0.2,0.4,0.6,0.8,1.0))
```

Out[18]:

	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
10%	2.835000e+06	3000.000000	2.000000	1.000000	1.000000	0.000000
20%	3.290000e+06	3450.000000	2.000000	1.000000	1.000000	0.000000
40%	3.990000e+06	4065.000000	3.000000	1.000000	1.000000	0.000000
50%	4.340000e+06	4600.000000	3.000000	1.000000	2.000000	0.000000
60%	4.830000e+06	5400.000000	3.000000	1.000000	2.000000	1.000000
80%	6.093500e+06	6600.000000	3.000000	2.000000	2.000000	2.000000
100%	1.330000e+07	16200.000000	6.000000	4.000000	4.000000	3.000000
max	1.330000e+07	16200.000000	6.000000	4.000000	4.000000	3.000000

In [29]:
```
data['price'].describe()
```

Out[29]:
```
count    5.450000e+02
mean     4.766729e+06
std      1.870440e+06
min      1.750000e+06
25%      3.430000e+06
50%      4.340000e+06
75%      5.740000e+06
max      1.330000e+07
Name: price, dtype: float64
```

In [28]:
```
data['price'].quantile()
```

Out[28]:
```
4340000.0
```

In [31]:
```
data['price'].quantile(0.25)
```

Out[31]:
```
3430000.0
```

In [439]:
```
data.info()
```

```
<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
```

isnull() and duplicated()

In [440]:
```
data.isnull().sum()#tells us if there is any missing value
'''to performm better analysis if in case there were missing values we wouldve
used fillna() to change null values to mean etc or simply remove them using dropna()'''
```

Out[440]:
```
'to performm better analysis if in case there were missing values we wouldve \nused fi
llna() to change null values to mean etc or simply remove them using dropna()'
```

In [441]:
```
plt.figure(figsize=(20,5),facecolor='lightpink')
plt.plot(data.isnull().sum(),color='k')
```

Out[441]:
```
[<matplotlib.lines.Line2D at 0x1bbf5ed1f10>]
```

In [442]:
```
plt.figure(figsize=(20,5),facecolor='lightblue')
plt.plot(data[data.duplicated()].sum(),color='k')
```

Out[442]:
```
[<matplotlib.lines.Line2D at 0x1bbf5e31650>]
```

columns and shape

In [443]:
```
data.columns#gives the name of the features given in the data
```

Out[443]:
```
Index(['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'mainroad',
       'guestroom', 'basement', 'hotwaterheating', 'airconditioning',
       'parking', 'prefarea', 'furnishingstatus'],
      dtype='object')
```

In [444]:
```
data.shape#give sthe shape of the 'matrix' that is data
```

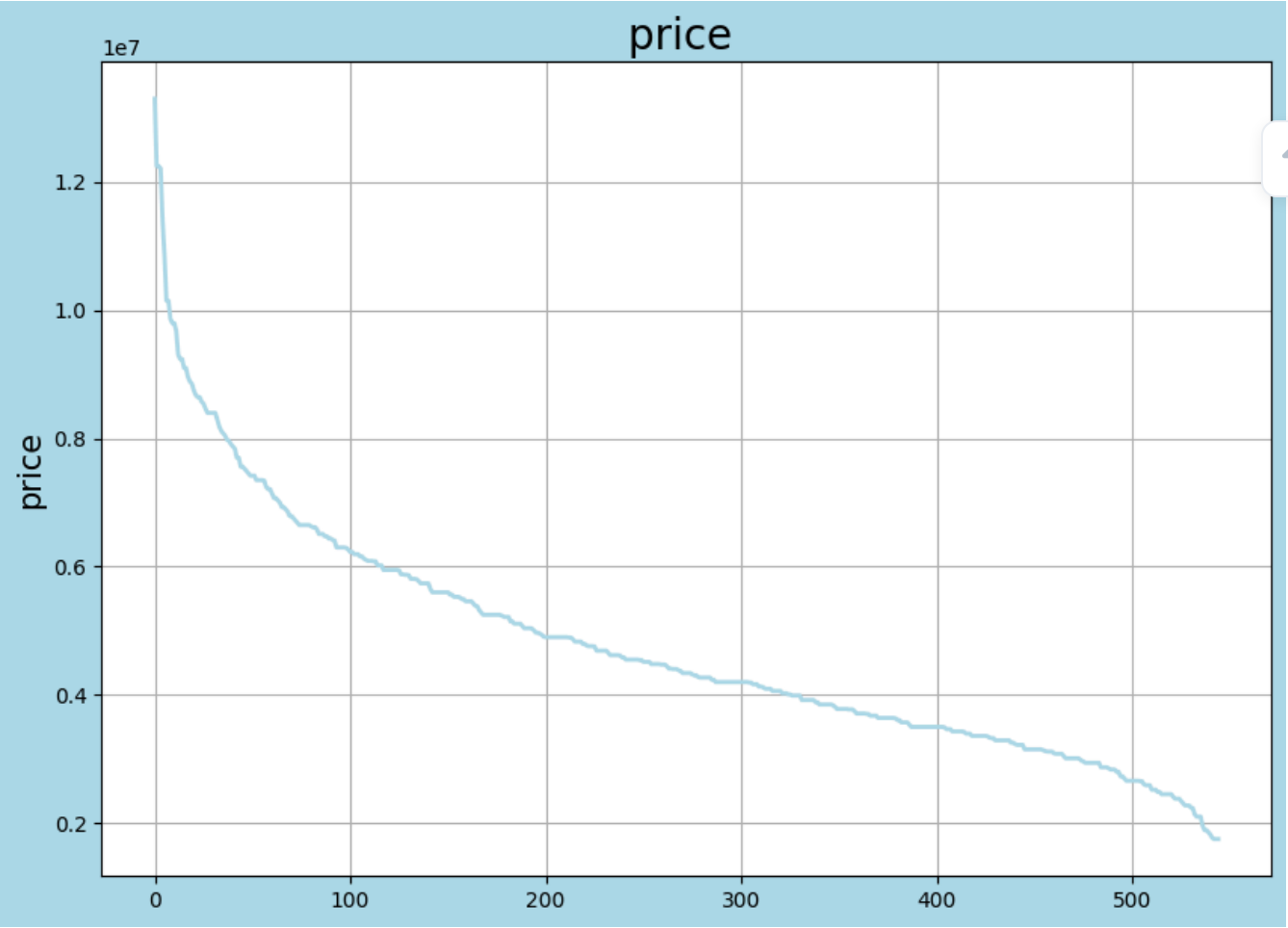
Out[444]:
```
(545, 13)
```

min(),max(),std(),mean().... and showing min and max area on graph

In [445]:
```
data['price'].min()#gives the cheapest house offerred
```

Out[445]:
```
1750000
```

In [446]:
```
plt.figure(facecolor='lightblue',figsize=(10,7))
plt.plot(data['price'],
         linewidth=2,color='lightblue')
plt.title('price ',fontsize=20,color='k')
plt.ylabel("price",fontsize=15)
plt.grid()
plt.show()
```



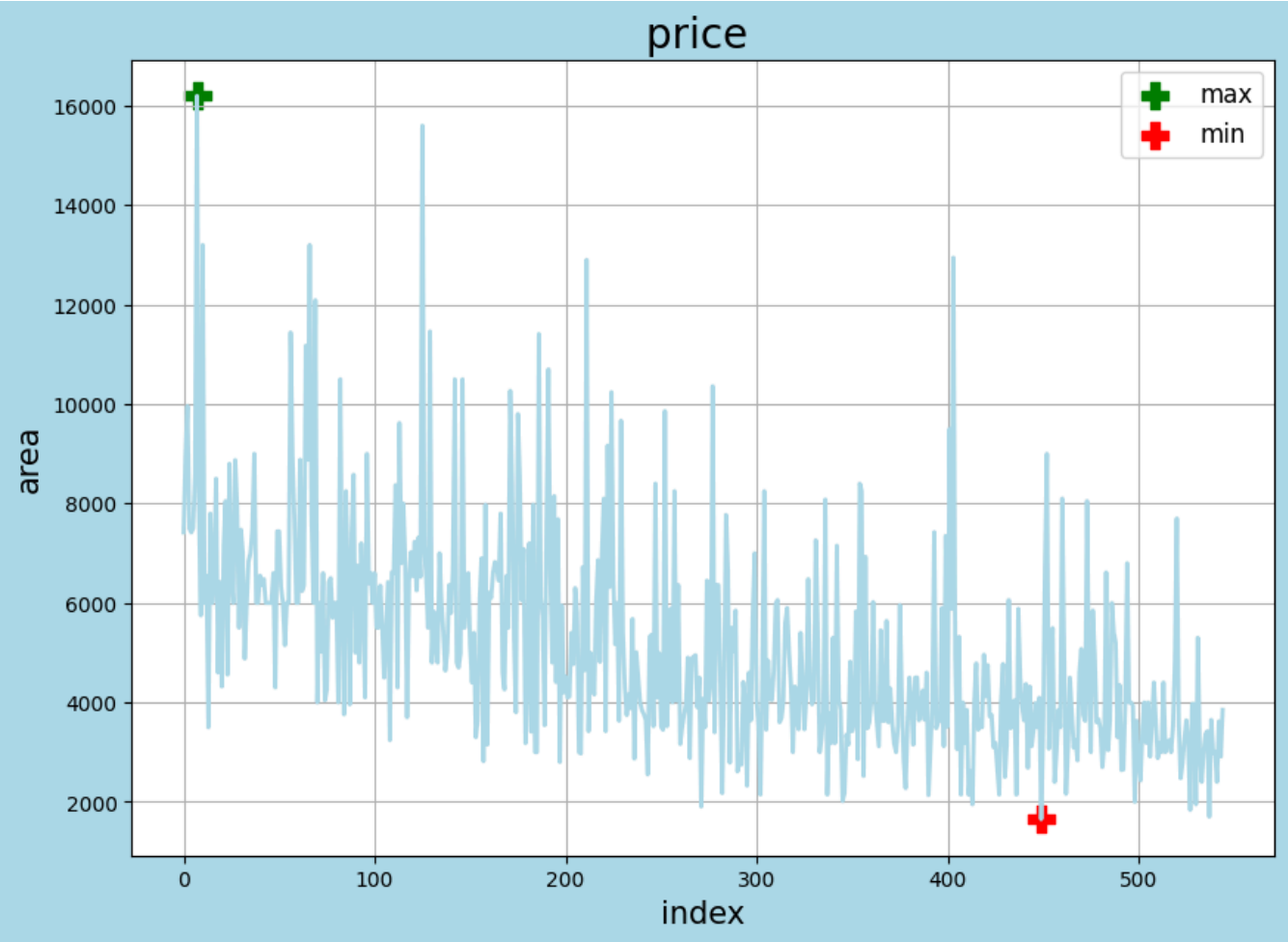
In [447]:
```
data['area'].std()#gives what is the span of area from the mean area
```

Out[447]: 2170.141022508803

In [448]:
```
data['area'].mean()#the avg area of house offered
```

Out[448]: 5150.54128440367

In [449]:
```
max_area= data['area'].max()
min_area = data['area'].min()

max_index = data['area'].idxmax()
min_index = data['area'].idxmin()

plt.figure(facecolor='lightblue',figsize=(10,7))
plt.plot(data['area'],
         linewidth=2,color='lightblue')

plt.scatter(max_index,max_area,marker='P',color='g',s=150,label='max')
plt.scatter(min_index,min_area,marker='P',color='r',s=150,label='min')

plt.title('price ',fontsize=20,color='k')
plt.ylabel("area",fontsize=15)
plt.xlabel("index",fontsize=15)
plt.legend(fontsize=12)
plt.grid()
plt.show()
```



slicing

In [450]:
```
data.iloc[273]#middle row
```

Out[450]:
```
price                 4340000
area                     3500
bedrooms                    4
bathrooms                   1
stories                     2
mainroad                  yes
guestroom                  no
basement                   no
hotwaterheating            no
airconditioning            no
parking                     2
prefarea                   no
furnishingstatus    furnished
Name: 273, dtype: object
```

In [451]:
```
'''since the no. of rows are odd , thus the true middle rows are two '''
data.iloc[272:274]
```

Out[451]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airco
272	4340000	4075	3	1	1	yes	yes	yes	no	
273	4340000	3500	4	1	2	yes	no	no	no	

In [452]:
```
data.iloc[:,0:2]#gives only price and area of the houses
```

Out[452]:

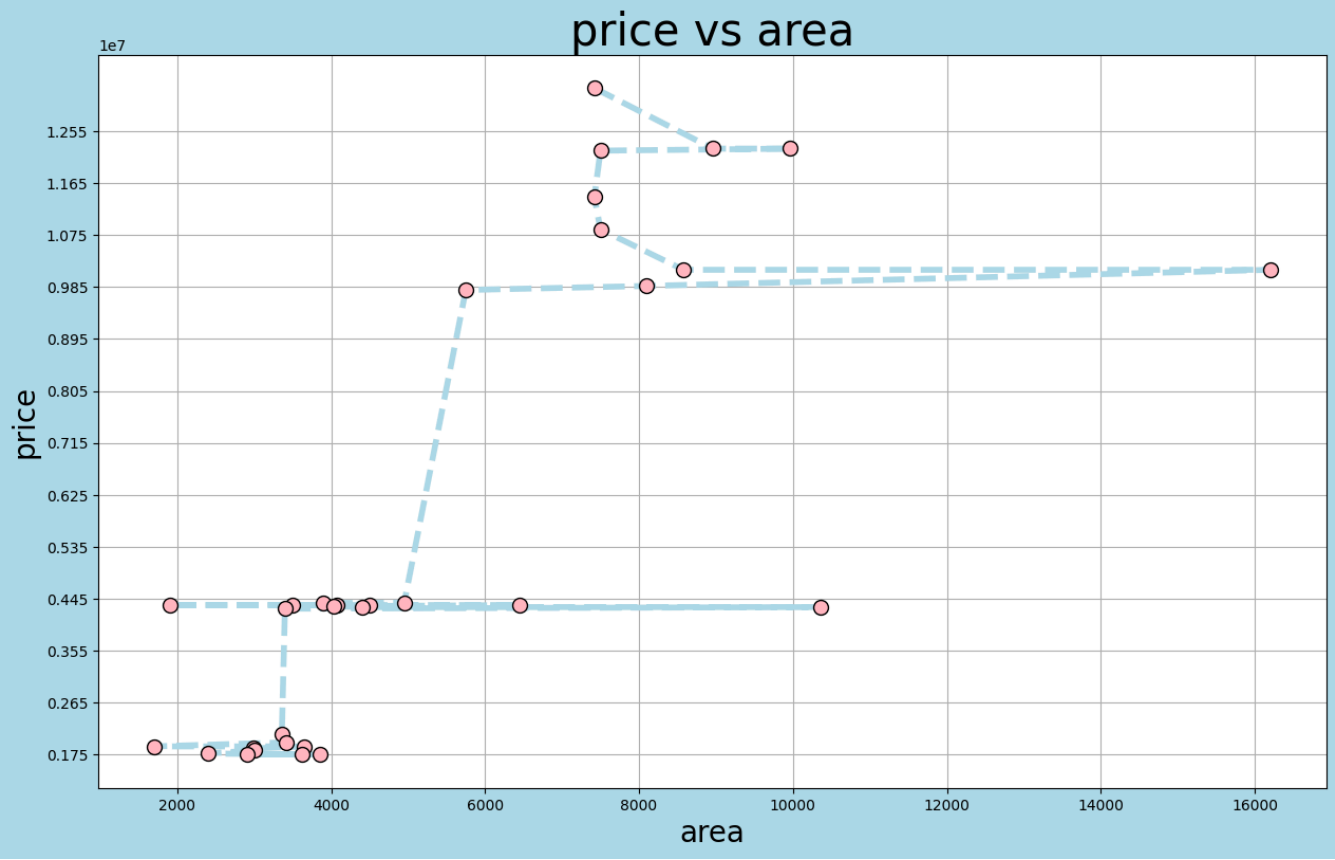
	price	area
0	13300000	7420
1	12250000	8960
2	12250000	9960
3	12215000	7500
4	11410000	7420
...	...	...
540	1820000	3000
541	1767150	2400
542	1750000	3620
543	1750000	2910
544	1750000	3850

545 rows × 2 columns

concatenate multiple data set to create a graph

In [453]:
```
d1=data.head(10)
d2=data.tail(10)
d3=data.iloc[268:279]
d = pd.concat([d1, d3,d2])
#getting starting ending and middle rows
```

In [454]:
```
x=d['area']
y=d['price']
plt.figure(figsize=(15,9),facecolor='lightblue')
plt.plot(x,y,linewidth=4,linestyle='--',color='lightblue',marker='o',ms=10,mec='k',mfc='
plt.xlabel('area',fontsize=20)
plt.ylabel('price',fontsize=20)
plt.yticks(np.arange(1750000,13300000,step=900000))
plt.title('price vs area',fontsize=30,color='k')
plt.grid()
plt.show()
```



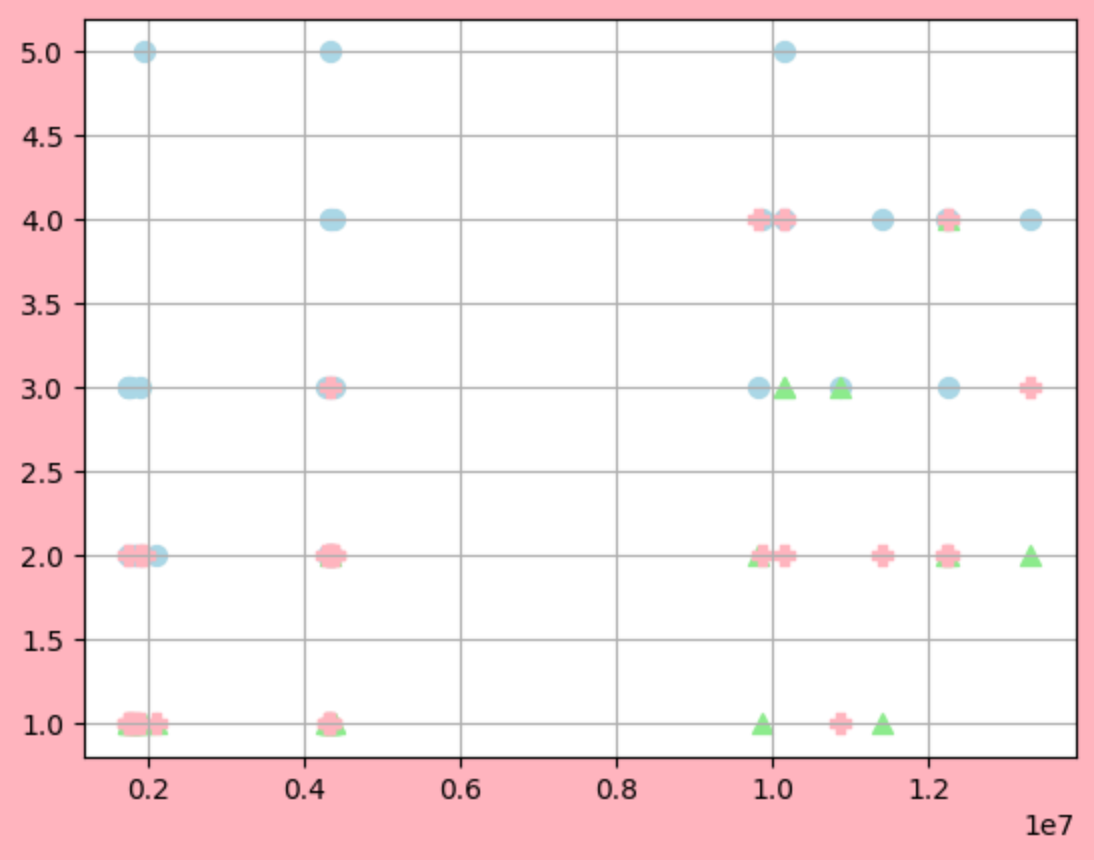
scatter plot

In [455]:
```
print(d['bedrooms'].max())
print(d['bathrooms'].max())
data['stories'].max()
```

```
5
4
```

Out[455]:
```
4
```

In [456]:
```
x=d['price']
y1=d['bedrooms']
y2=d['bathrooms']
y3=d['stories']
plt.figure(facecolor='lightpink')
plt.scatter(x,y1,color='lightblue',s=50,label='bedrooms',marker='o')
plt.scatter(x,y2,color='lightgreen',s=50,label='bathrooms',marker='^')
plt.scatter(x,y3,color='lightpink',s=50,label='stories',marker='P')
plt.grid()
plt.show()
```



TAKING SAMPLE

In [457]:
```
#making a smaller data frame by picking the indexes at regular intervals
arr=(np.linspace(1,544,20)).astype(int)
arr
```

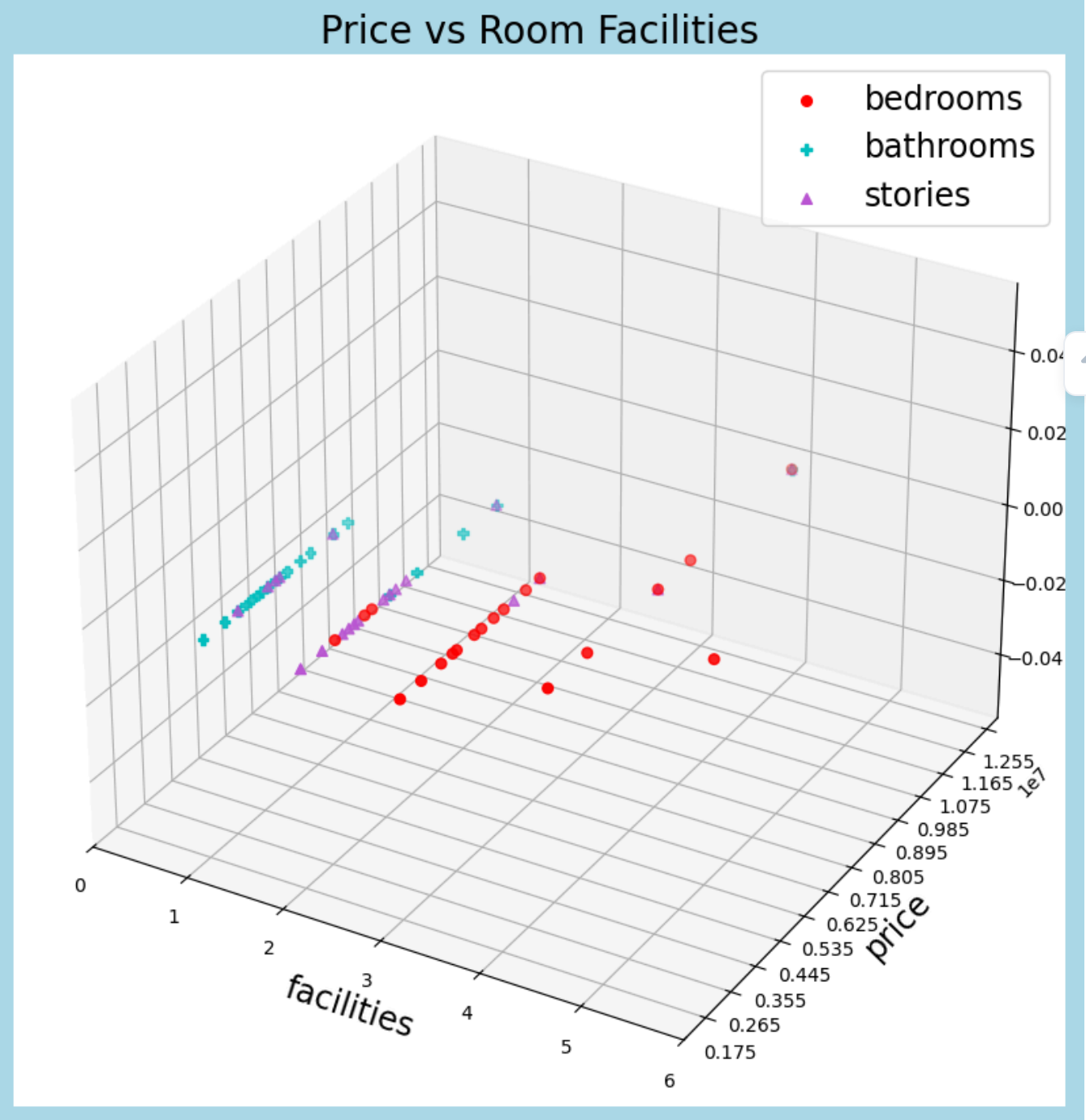
Out[457]:
```
array([  1,  29,  58,  86, 115, 143, 172, 201, 229, 258, 286, 315, 343,
       372, 401, 429, 458, 486, 515, 544])
```

In [458]:
```
house=data.iloc[arr]
house
```

Out[458]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	a
1	12250000	8960	4	4	4	yes	no	no	no	
29	8400000	5500	4	2	2	yes	no	yes	no	
58	7210000	7680	4	2	4	yes	yes	no	no	
86	6510000	6670	3	1	3	yes	no	yes	no	
115	6020000	8000	3	1	1	yes	yes	yes	no	
143	5600000	4800	5	2	3	no	no	yes	yes	
172	5250000	8400	3	1	2	yes	yes	yes	no	
201	4900000	4095	3	1	2	no	yes	yes	no	
229	4690000	9667	4	2	2	yes	yes	yes	no	
258	4480000	4040	3	1	2	yes	no	no	no	
286	4235000	2787	3	1	1	yes	no	yes	no	

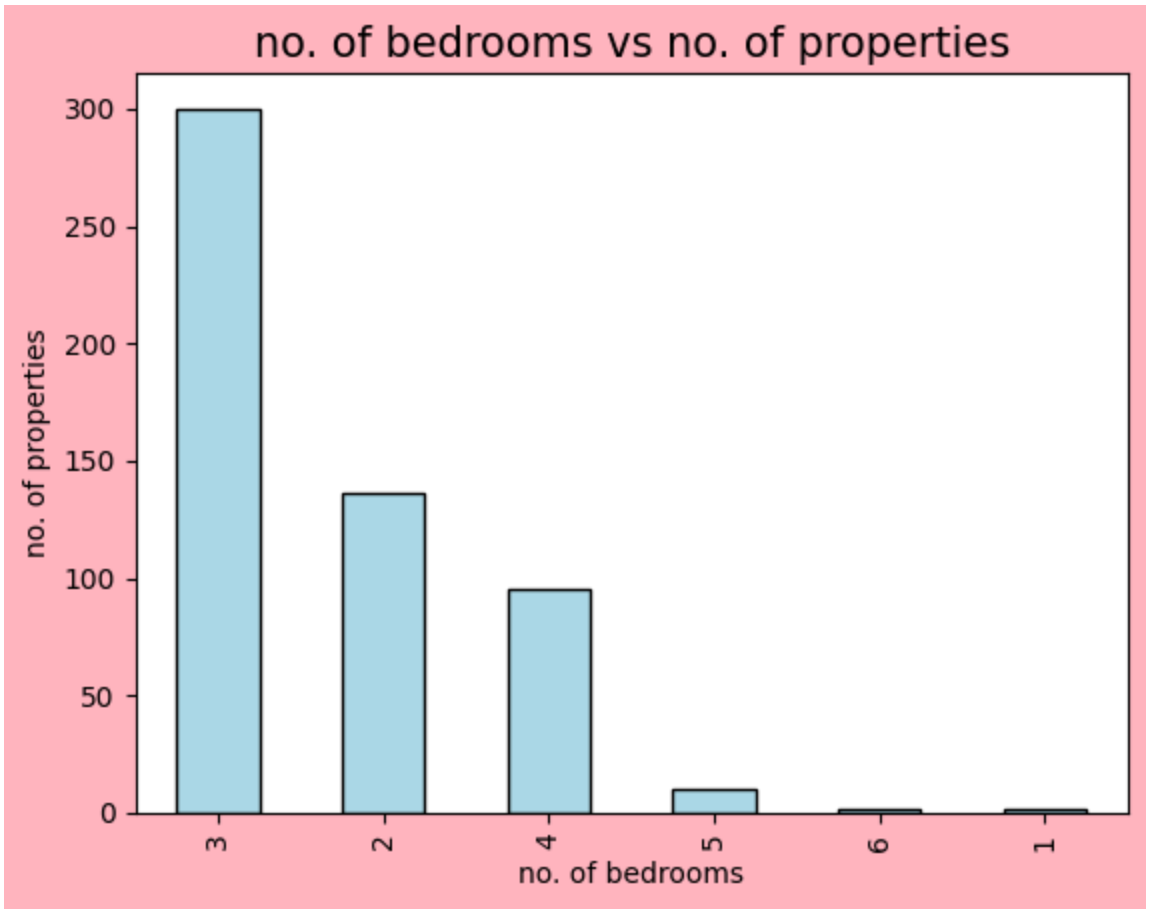
In [459]:
```
fig=plt.figure(figsize=(10,20),facecolor='lightblue')
fig.add_subplot(projection='3d')
x1=house['price']
y1=house['bedrooms']
y2=house['bathrooms']
y3=house['stories']

plt.scatter(y1,x1, marker='o',color='red', label='bedrooms',s=30)
plt.scatter(y2,x1, marker='P',color='c', label='bathrooms',s=30)
plt.scatter(y3,x1, marker='^',color='mediumorchid', label='stories',s=30)
plt.title('Price vs Room Facilities',fontsize=20)
plt.ylabel('price',fontsize=17,color='k')
plt.xlabel('facilities',fontsize=17,color='k')
plt.xticks(np.arange(0,7,step=1))
plt.yticks(np.arange(1750000,13300000,step=900000))
plt.legend(fontsize=17)
plt.grid(axis='x',linestyle='--')
plt.show()
```



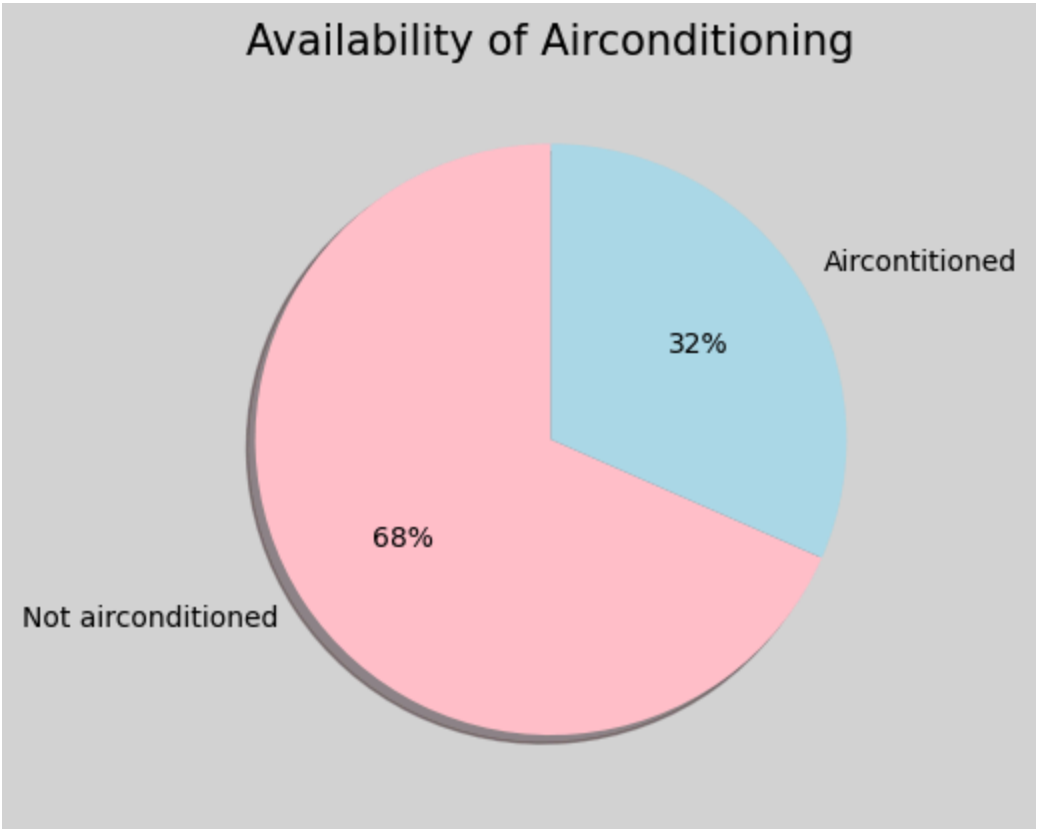
pie chart and bar plot to show internal facilities

In [460]:
```
count1=data['bedrooms'].value_counts()
```

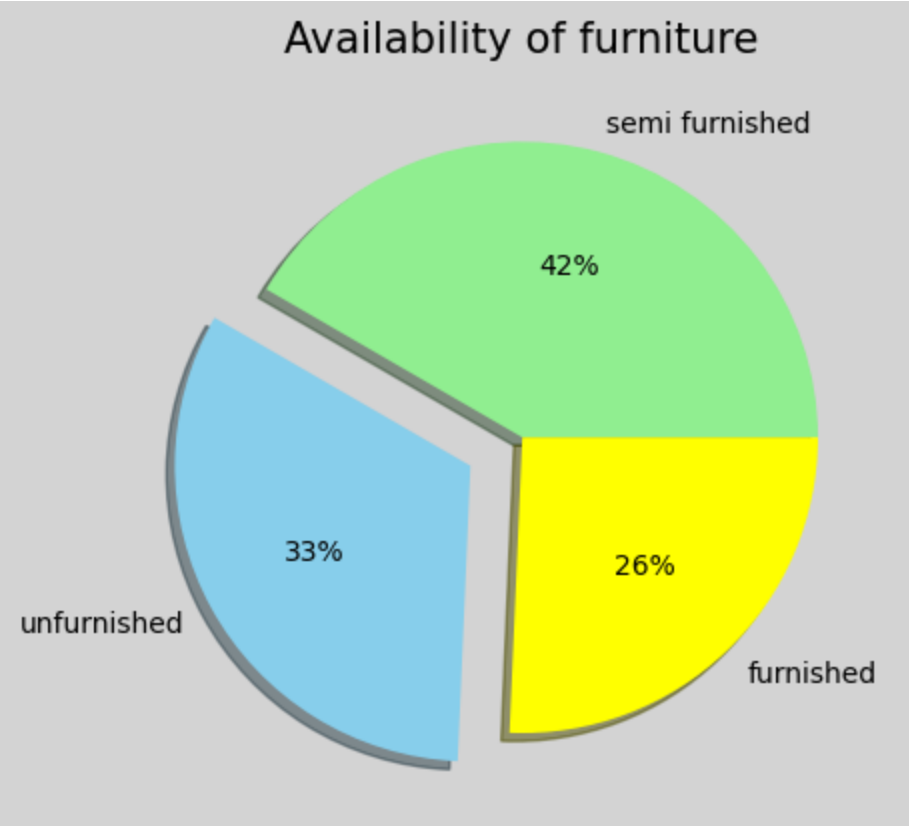
In [461]:
```
plt.figure(facecolor='lightpink')
count1.plot(kind='bar',color='lightblue',edgecolor='k')
plt.title('no. of bedrooms vs no. of properties',fontsize=15)
plt.xlabel('no. of bedrooms')
plt.ylabel('no. of properties')
plt.show()
```



In [462]:
```
plt.figure(facecolor='lightgrey')
count1=data['airconditioning'].value_counts()
cols1=['pink','lightblue']
label1=['Not airconditioned','Aircontitioned']
plt.pie(count1,colors=cols1,autopct='%1.0f%%',shadow=True,startangle=90,labels=label1)
plt.title('Availability of Airconditioning', fontsize=15)
plt.show()
```



In [463]:
```
plt.figure(facecolor='lightgrey')
count2=data['furnishingstatus'].value_counts()
cols2=['lightgreen','skyblue','yellow']
label2=['semi furnished','unfurnished','furnished']
plt.pie(count2,colors=cols2,autopct='%1.0f%%',shadow=True,labels=label2,explode=[0,0.2,0
plt.title('Availability of furniture',fontsize=15)
plt.show()
```

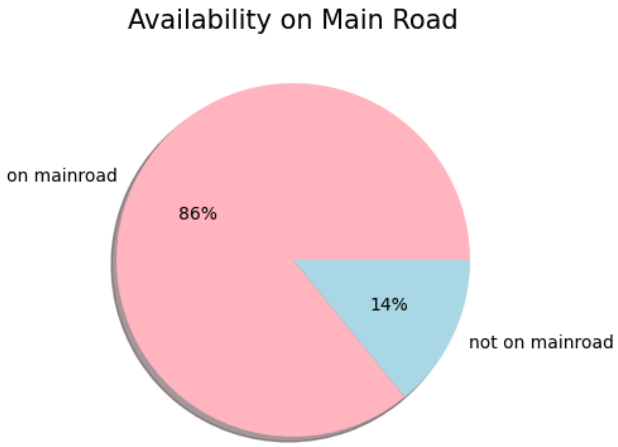


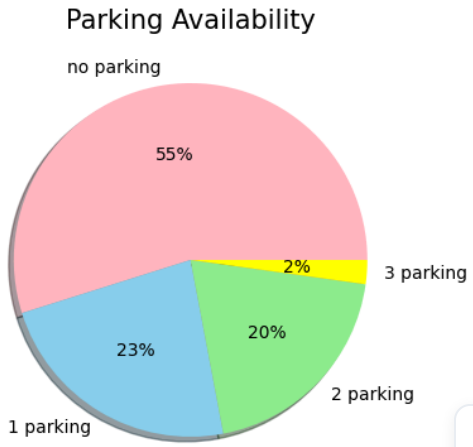
SUBPLOT TO SHOW EXTERNAL FACILITIES

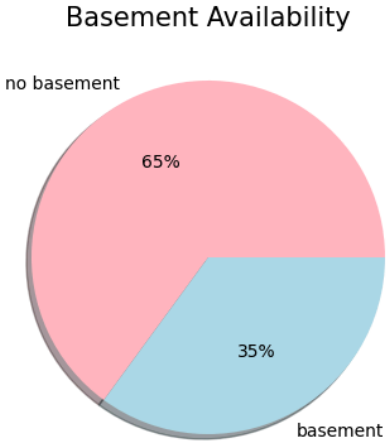
In [464]:
```
count1=data['mainroad'].value_counts()
count2=data['parking'].value_counts()
count3=data['basement'].value_counts()
count4=data['prefarea'].value_counts()
print(count1,'\n \n',count2,'\n \n',count3,'\n \n',count4)
```

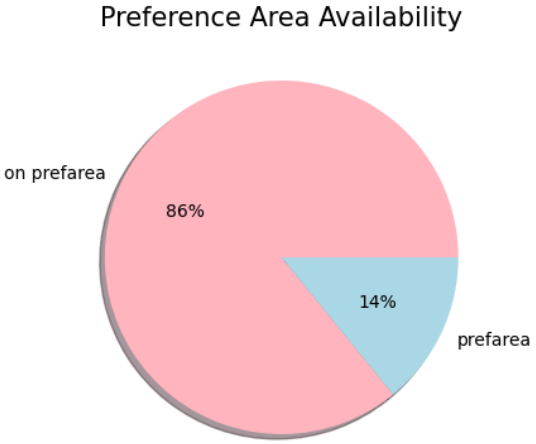
```
mainroad
yes    468
no      77
Name: count, dtype: int64

 parking
0    299
1    126
2    108
3     12
Name: count, dtype: int64

 basement
no     354
yes    191
Name: count, dtype: int64

 prefarea
no     417
yes    128
Name: count, dtype: int64
```

In [465]:
```
cols1=['lightpink','lightblue']
cols2=['lightpink','skyblue','lightgreen','yellow']
label1=['on mainroad','not on mainroad']
label2=['no parking','1 parking','2 parking','3 parking']
label3=['no basement','basement']
label4=['on prefarea','prefarea']
fig,sub=plt.subplots(2,2,figsize=(14,10))
sub[0,0].pie(count1,
                  colors=cols1,
                  autopct='%1.0f%%',
                  shadow=True,
                  labels=label1)
sub[0, 0].set_title('Availability on Main Road', fontsize=15)
sub[0,1].pie(count2,
                  colors=cols2,
                  autopct='%1.0f%%',
                  shadow=True,
                  labels=label2)
sub[1,0].pie(count3,
                  colors=cols1,
                  autopct='%1.0f%%',
                  shadow=True,
                  labels=label3)
sub[0, 1].set_title('Parking Availability', fontsize=15)

sub[1, 0].set_title('Basement Availability', fontsize=15)
sub[1,1].pie(count1,
                  colors=cols1,
                  autopct='%1.0f%%',
                  shadow=True,
                  labels=label4)
sub[1, 1].set_title('Preference Area Availability', fontsize=15)
plt.show()
```





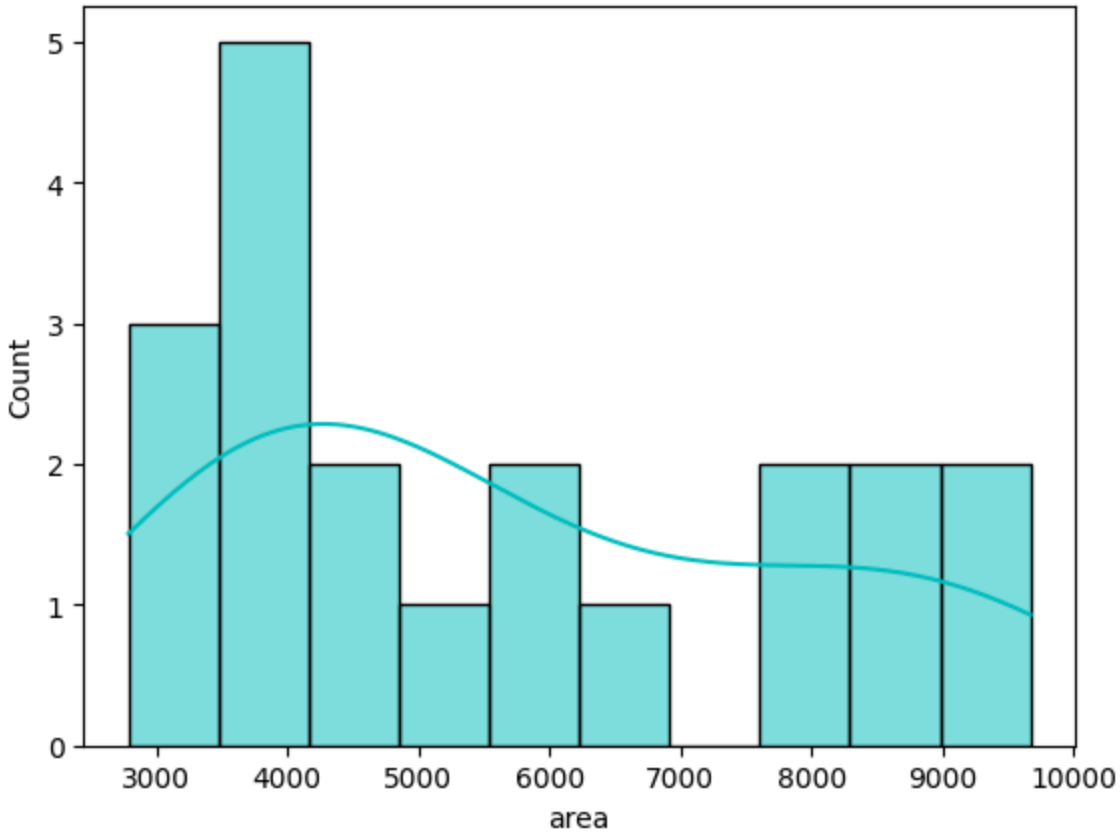




USING SEABORN

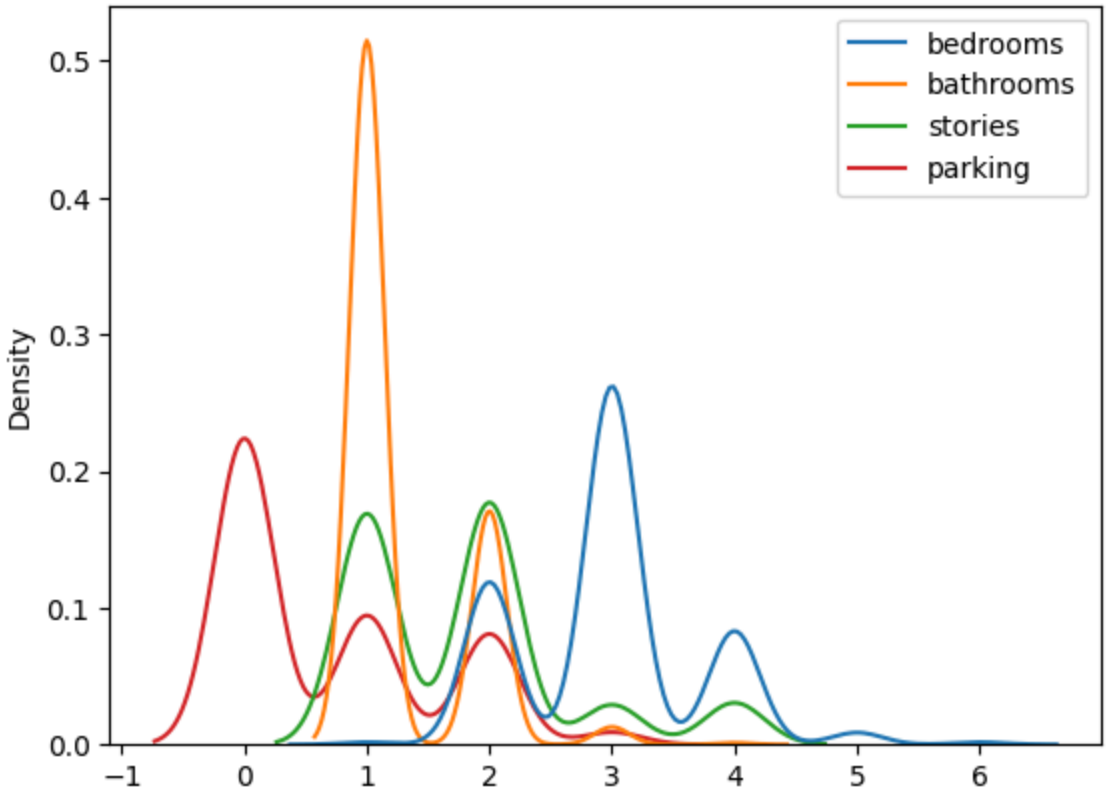
In [466]:
```
sns.histplot(house['area'],color='c',bins=10,fill=True,
             kde=True)
```

Out[466]: <Axes: xlabel='area', ylabel='Count'>



In [467]:
```
sns.kdeplot(data.iloc[:,2:],fill=False,)
```

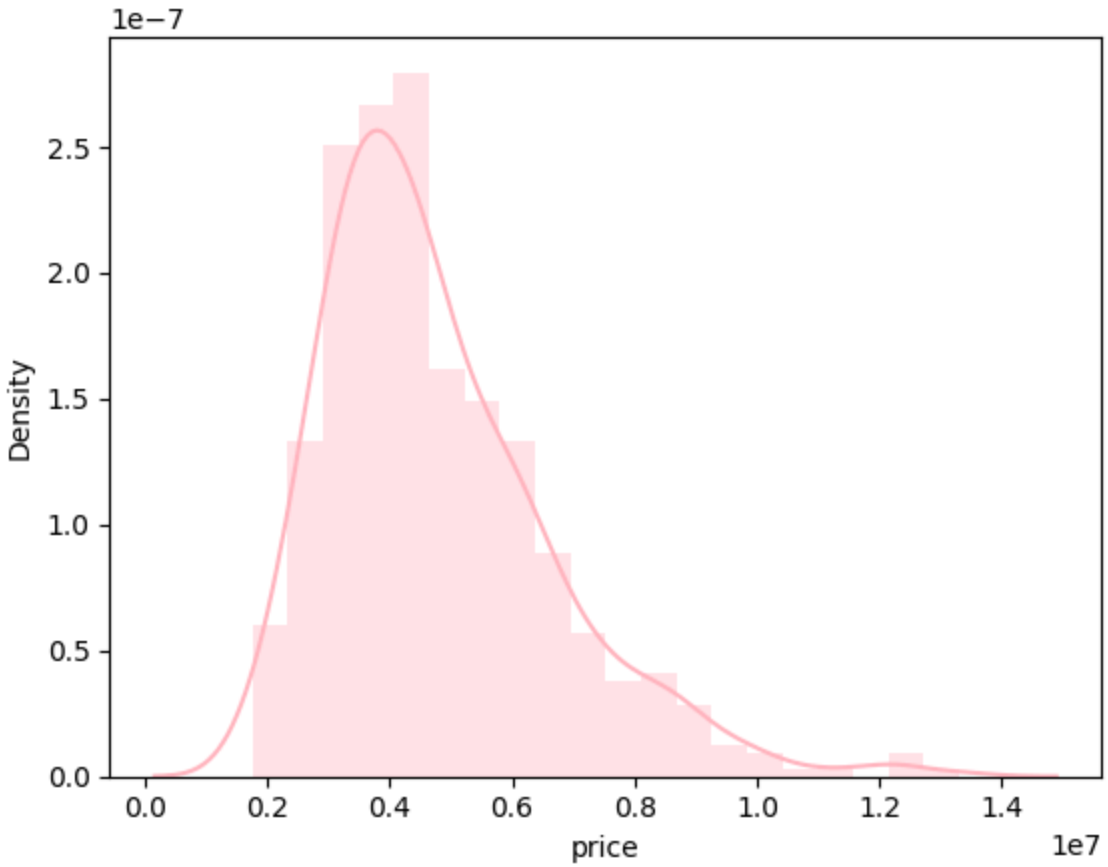
Out[467]: <Axes: ylabel='Density'>



In [468]:
```
sns.distplot(data['price'],color='lightpink',bins=20)
```

```
C:\Users\weite\AppData\Local\Temp\ipykernel_14036\2936774675.py: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 (https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(data['price'],color='lightpink',bins=20)
```

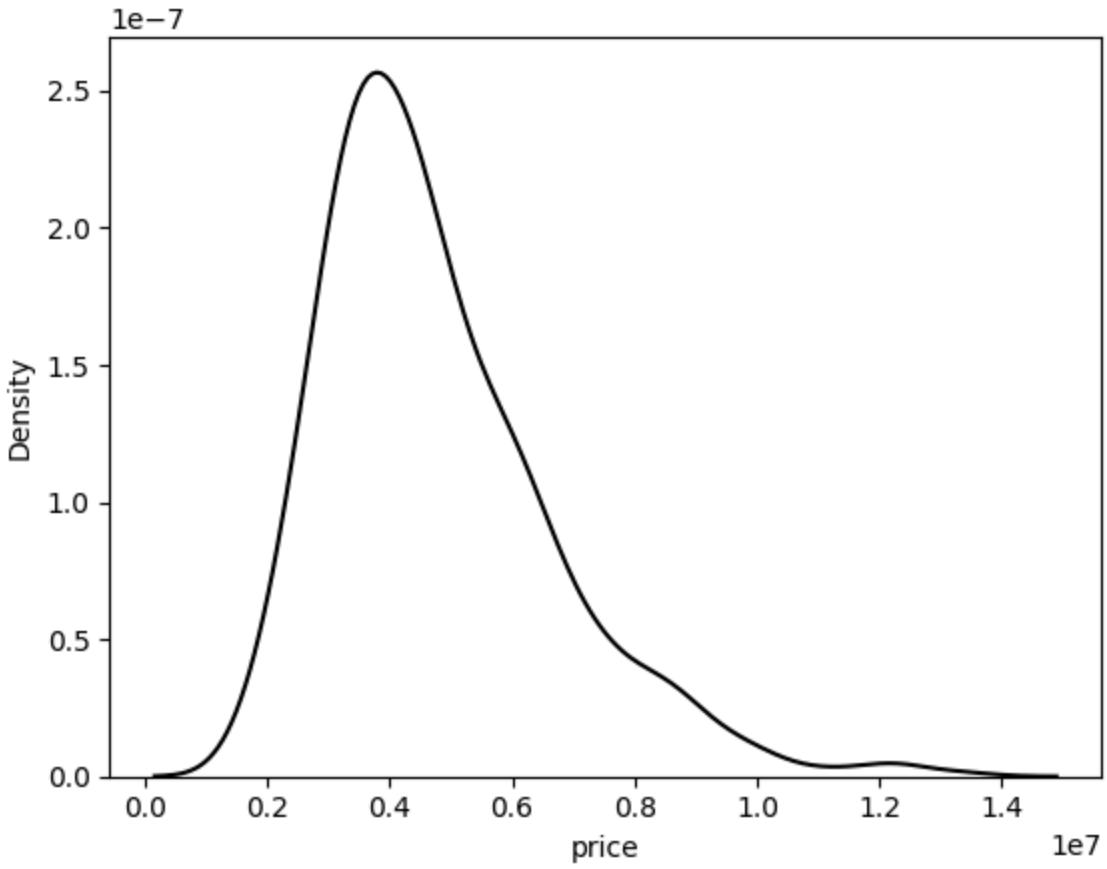
Out[468]:
```
<Axes: xlabel='price', ylabel='Density'>
```



In [469]:
```
sns.distplot(data['price'],color='k',bins=20,hist=False)
```

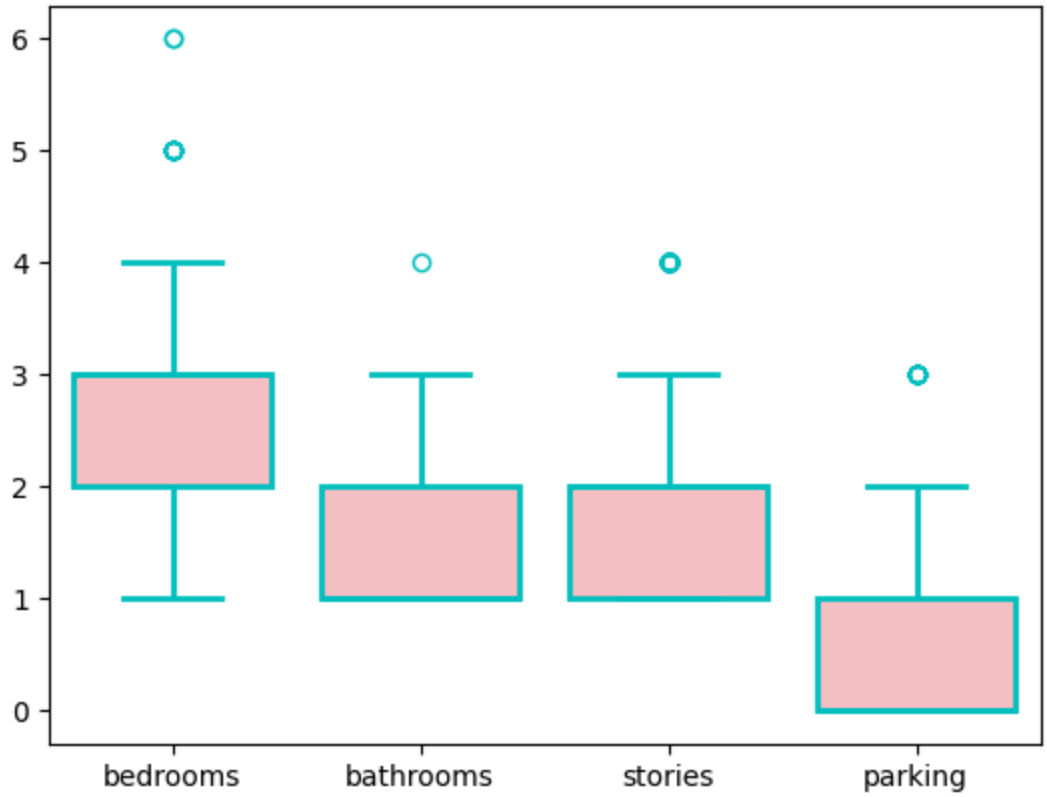
```
C:\Users\weite\AppData\Local\Temp\ipykernel_14036\1126506230.py:1: UserWarning:

`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 `kdeplot` (an axes-level function for kernel density plots).

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

  sns.distplot(data['price'],color='k',bins=20,hist=False)
```

Out[469]:
```
<Axes: xlabel='price', ylabel='Density'>
```



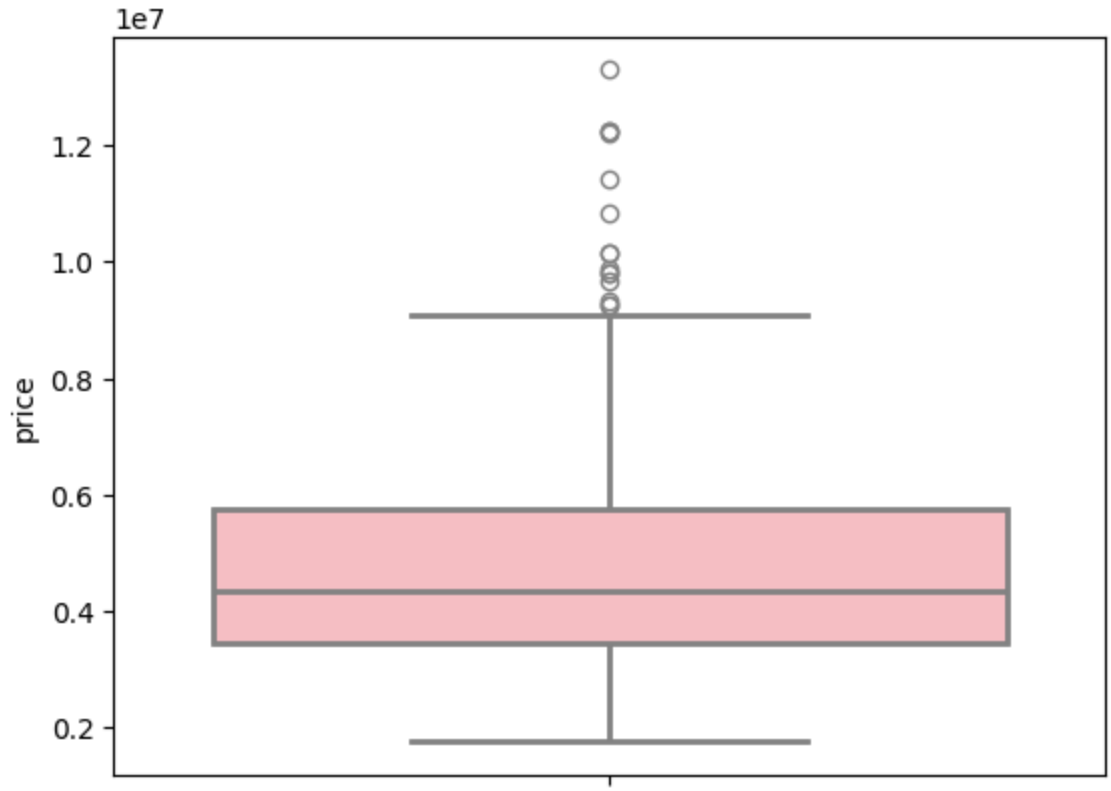
In [470]:
```
sns.boxplot(data.iloc[:,2:],color='lightpink',linecolor='c',linewidth=2)
```

Out[470]: <Axes: >

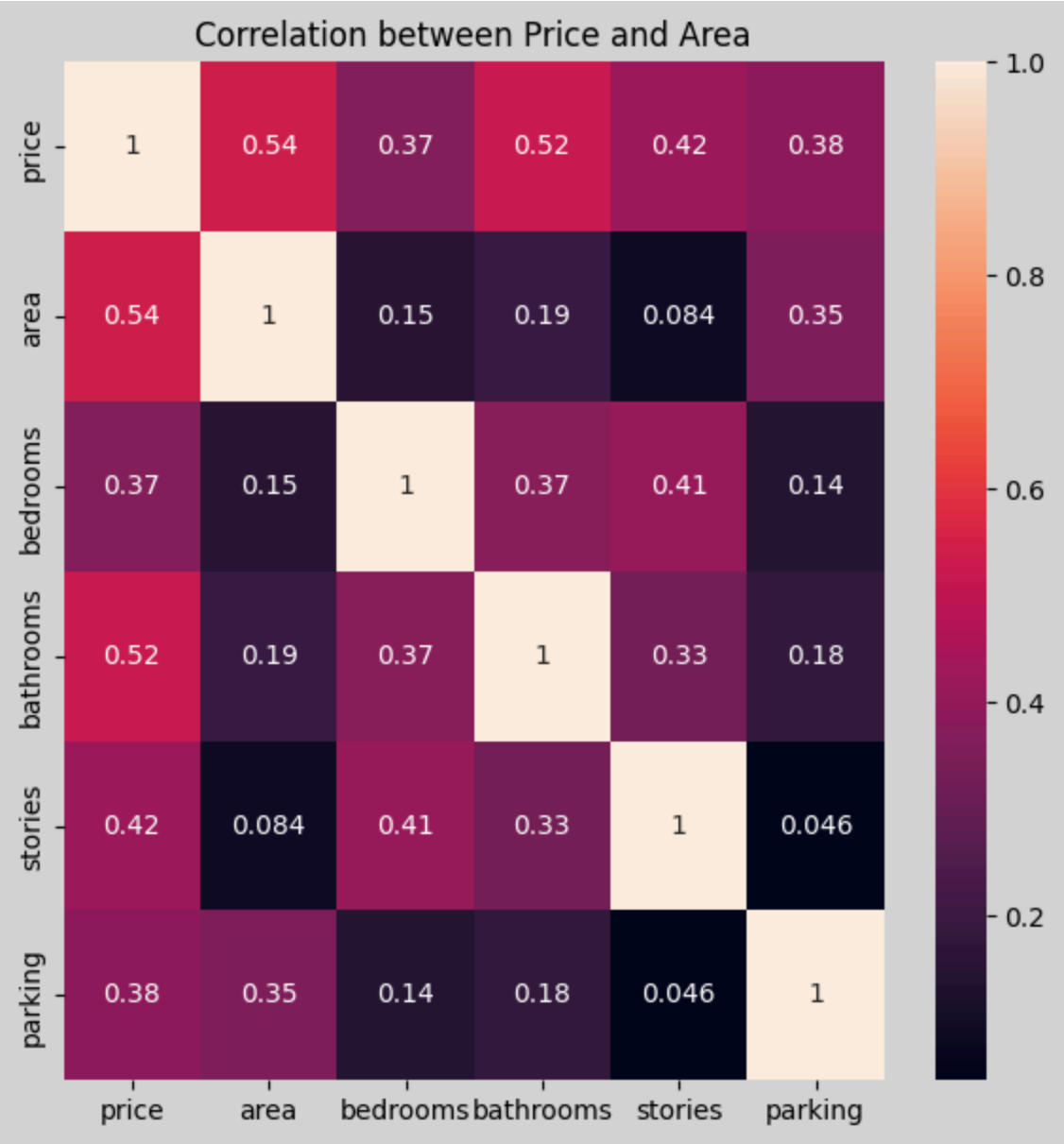


In [39]:
```
sns.boxplot(data['price'],color='lightpink',linewidth=2)
```

Out[39]: <Axes: ylabel='price'>



In [491]:
```
plt.figure(figsize=(7,7),facecolor='lightgrey')
df_numeric = data.select_dtypes(include=['number'])
df = df_numeric.corr(method='pearson')
sns.heatmap(df,annot=True)
plt.title('Correlation between Price and Area')
plt.show()
```



sampling using scipy

In [7]:
```
#systematic
n=int(input('enter equal interval : '))
sample_data=data.iloc[::n]
print("systematic  sampling:")
sample_data
```

```
enter equal interval : 50
systematic  sampling:
```

Out[7]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airc
0	13300000	7420	4	2	3	yes	no	no	no	
50	7420000	7440	3	2	4	yes	no	no	no	
100	6230000	6600	3	2	1	yes	no	yes	no	
150	5600000	5136	3	1	2	yes	yes	yes	no	
200	4900000	4520	3	1	2	yes	no	yes	no	
250	4515000	3510	3	1	3	yes	no	no	no	
300	4200000	4079	3	1	3	yes	no	no	no	
350	3780000	3420	2	1	2	yes	no	no	yes	
400	3500000	3512	2	1	1	yes	no	no	no	
450	3150000	3450	3	1	2	yes	no	yes	no	
500	2660000	2800	3	1	1	yes	no	no	no	

In [8]:
```
#random
n=int(input('enter no. of random samples: '))
sample_data=data.sample(n,random_state=400)
print("simple random sampling:")
sample_data
```

```
enter no. of random samples: 20
simple random sampling:
```

Out[8]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airco
371	3640000	3570	3	1	2	yes	no	yes	no	
268	4382000	4950	4	1	2	yes	no	no	no	
59	7210000	6000	3	2	4	yes	yes	no	no	
437	3290000	5880	3	1	1	yes	no	no	no	
189	5040000	3540	2	1	1	no	yes	yes	no	
250	4515000	3510	3	1	3	yes	no	no	no	
74	6650000	4040	3	1	2	yes	no	yes	yes	
54	7350000	6000	3	2	2	yes	yes	no	no	
27	8400000	8875	3	1	1	yes	no	no	no	
232	4655000	3745	3	1	2	yes	no	yes	no	
297	4200000	3640	3	2	2	yes	no	yes	no	
242	4550000	3640	3	1	2	yes	no	no	no	
376	3640000	4130	3	2	2	yes	no	no	no	
70	6790000	4000	3	2	2	yes	no	yes	no	
22	8645000	8050	3	1	1	yes	yes	yes	no	
68	6860000	6000	3	1	1	yes	no	no	no	
114	6020000	6800	2	1	1	yes	yes	yes	no	
309	4130000	4632	4	1	2	yes	no	no	no	
255	4480000	5885	2	1	1	yes	no	no	no	
429	3325000	4775	4	1	2	yes	no	no	no	

In [11]:
```
#stratisfied
data.groupby('bathrooms').get_group(2)
```

Out[11]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airc
0	13300000	7420	4	2	3	yes	no	no	no	
2	12250000	9960	3	2	2	yes	no	yes	no	
3	12215000	7500	4	2	2	yes	no	yes	no	
9	9800000	5750	3	2	4	yes	yes	no	no	
12	9310000	6550	4	2	2	yes	no	no	no	
...	...	...	...	...	...	...	...	...	...	
390	3500000	2135	3	2	2	no	no	no	no	
413	3430000	1950	3	2	2	yes	no	yes	no	
446	3150000	3986	2	2	1	no	yes	yes	no	
509	2590000	3600	2	2	2	yes	no	yes	no	
523	2380000	2787	4	2	2	yes	no	no	no	

133 rows × 13 columns

In [14]:
```
strata=data.groupby('bathrooms',group_keys=False).apply(lambda x:x.sample(min(len(x),2))
strata
```

Out[14]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airc
97	6300000	6400	3	1	1	yes	yes	yes	no	
282	4270000	2175	3	1	2	no	yes	yes	no	
83	6580000	6000	3	2	4	yes	no	no	no	
377	3640000	2850	3	2	2	no	no	yes	no	
5	10850000	7500	3	3	1	yes	no	yes	no	
195	4970000	4410	4	3	2	yes	no	yes	no	
1	12250000	8960	4	4	4	yes	no	no	no	

In [15]:
```
strata=data.groupby('bathrooms').apply(lambda x:x.sample(min(len(x),2)))
strata
```

Out[15]:

bathrooms		price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterh
1	8	9870000	8100	4	1	2	yes	yes	yes	
	431	3290000	3180	4	1	2	yes	no	yes	
2	124	5950000	6525	3	2	4	yes	no	no	
	29	8400000	5500	4	2	2	yes	no	yes	
3	5	10850000	7500	3	3	1	yes	no	yes	
	11	9681000	6000	4	3	2	yes	yes	yes	
4	1	12250000	8960	4	4	4	yes	no	no	

In [5]:
```
#cluster
clusters=data.bedrooms.unique()
clusters
```

Out[5]:
```
array([4, 3, 5, 2, 6, 1], dtype=int64)
```

In [16]:
```
for i in np.random.choice(clusters,size=2,replace=False):
    filtered_data=data[data['bedrooms']==i]
    cs=filtered_data.sample(3)
```

In [17]:
```
cs
```

Out[17]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airco
89	6440000	8580	5	3	2	yes	no	no	no	
340	3850000	5300	5	2	2	yes	no	no	no	
34	8120000	6840	5	1	2	yes	yes	yes	no	

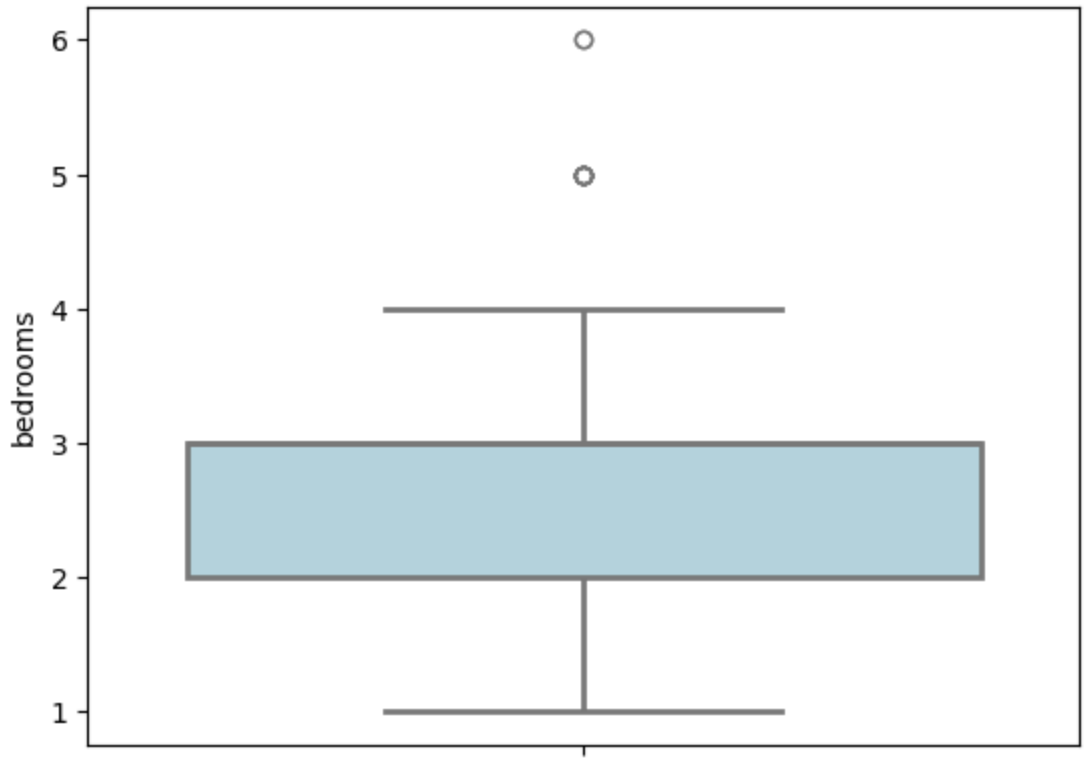
In [ ]:
```

```

outlliers using boxplot

In [63]:
```
print(data['bedrooms'].mean())
sns.boxplot(data['bedrooms'],color='lightblue',linewidth=2)
```

```
2.9651376146788992
```

Out[63]: <Axes: ylabel='bedrooms'>



In [64]:
```
df=data[(data['bedrooms']<=4)]
df
```

Out[64]:

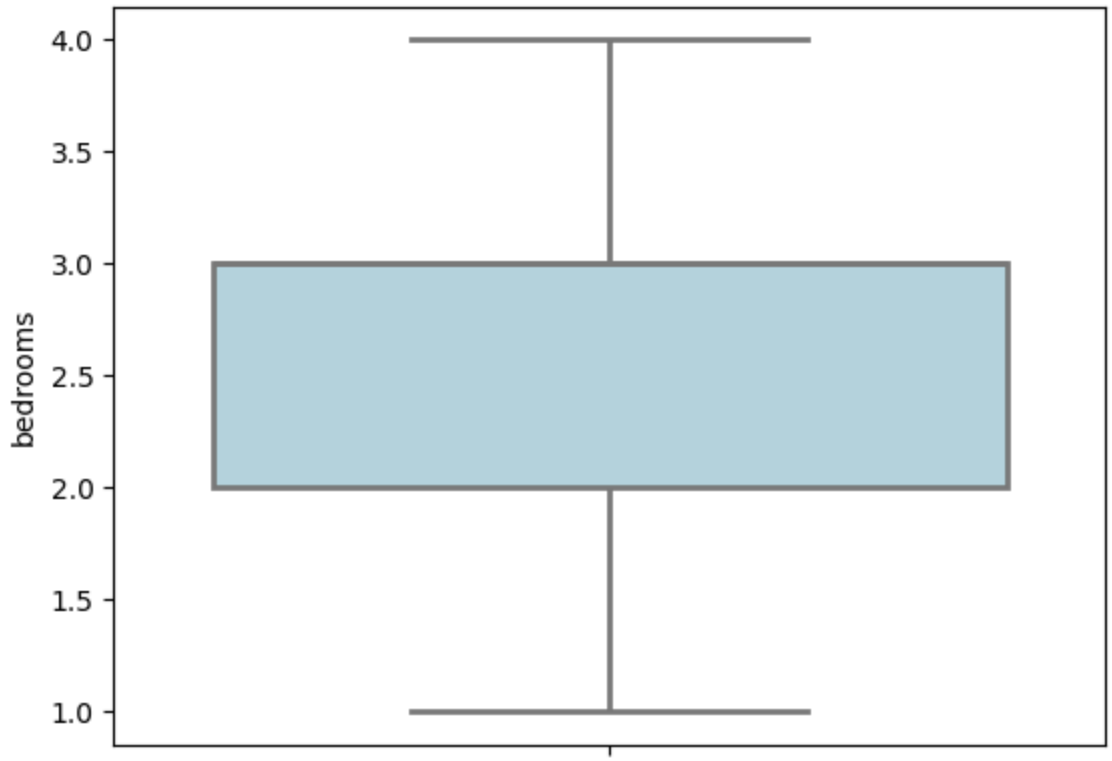
	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airc
0	13300000	7420	4	2	3	yes	no	no	no	
1	12250000	8960	4	4	4	yes	no	no	no	
2	12250000	9960	3	2	2	yes	no	yes	no	
3	12215000	7500	4	2	2	yes	no	yes	no	
4	11410000	7420	4	1	2	yes	yes	yes	no	
...	...	...	...	...	...	...	...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

533 rows × 13 columns

In [65]:
```
print(df['bedrooms'].mean(numeric_only=True))
sns.boxplot(df['bedrooms'],color='lightblue',linewidth=2)
```

```
2.9155722326454034
```

Out[65]: <Axes: ylabel='bedrooms'>



outliers using iqr

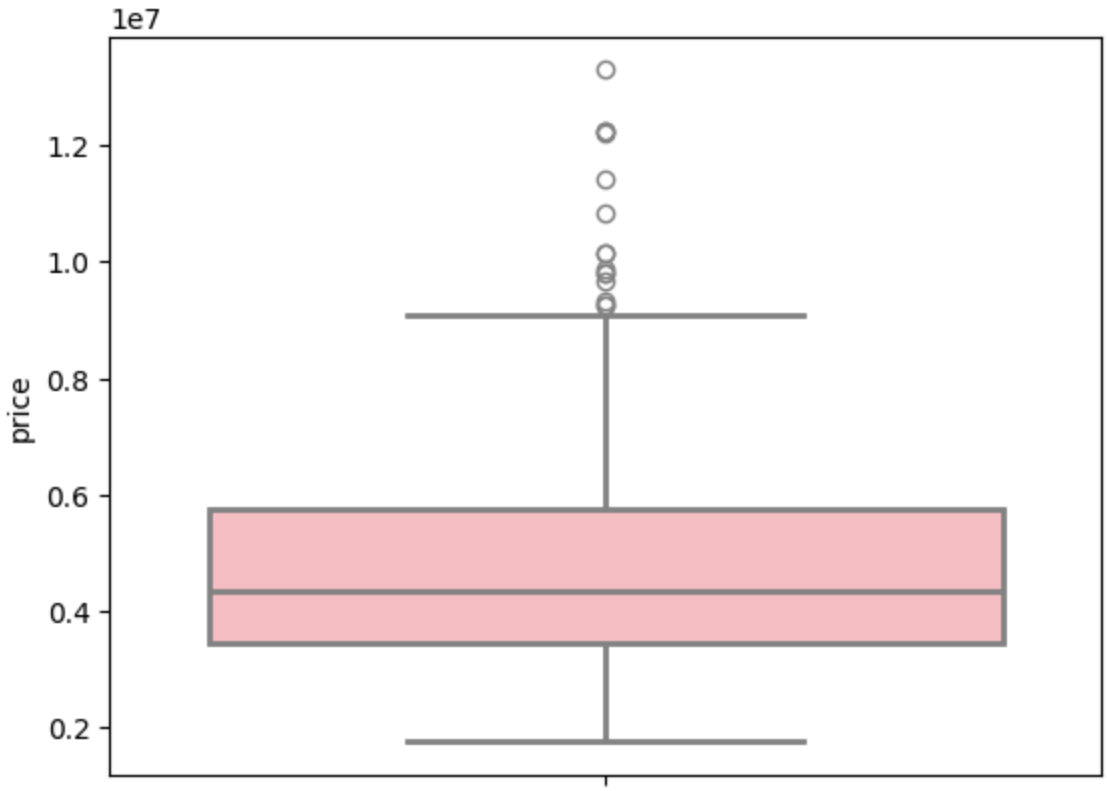
In [20]:
```
q1=data['price'].quantile(0.25)
q3=data['price'].quantile(0.75)
IQR=q3-q1
lower_limit=q1-1.5*IQR
upper_limit=q3+1.5*IQR
outliers=[x for x in data['price'] if x<lower_limit or x>upper_limit]
print(outliers)
print(upper_limit)
```

```
[13300000, 12250000, 12250000, 12215000, 11410000, 10850000, 10150000, 10150000, 98700
00, 9800000, 9800000, 9681000, 9310000, 9240000, 9240000]
9205000.0
```

In [21]:
```
#initial
print(data['price'].mean())
sns.boxplot(data['price'],color='lightpink',linewidth=2)
```

```
4766729.247706422
```

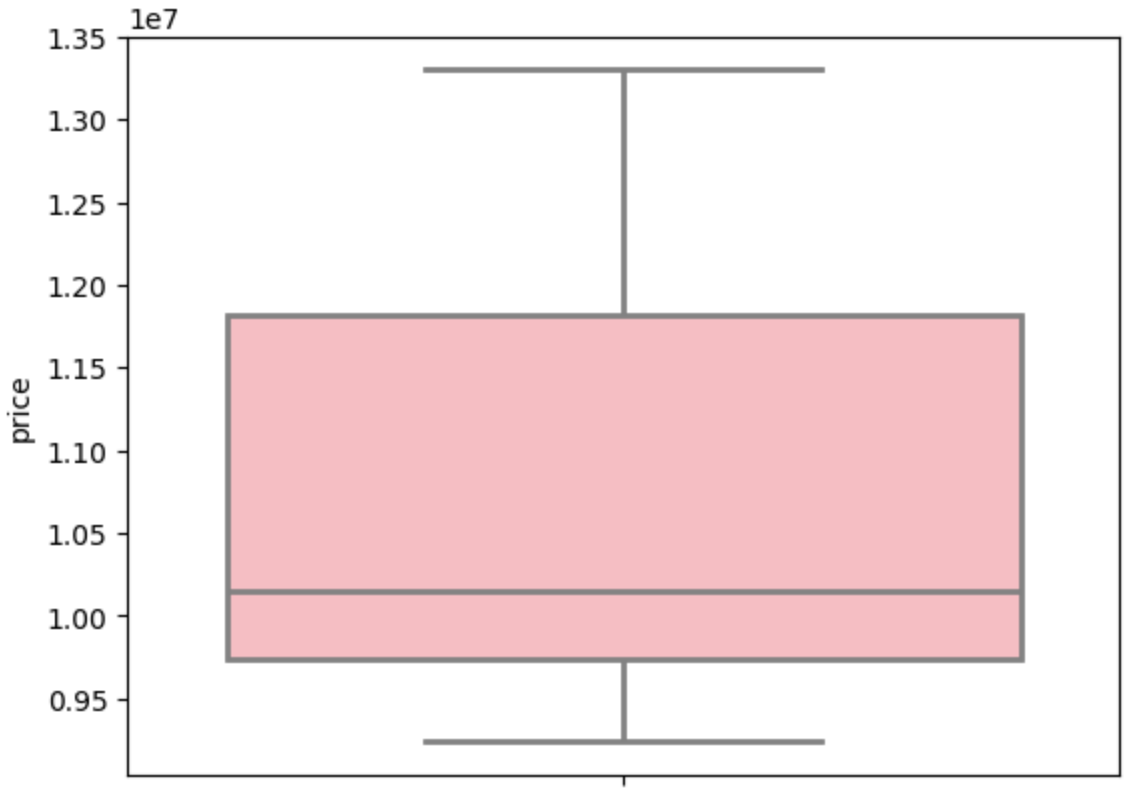
Out[21]: <Axes: ylabel='price'>



In [22]:
```
df=data[(data['price']<lower_limit) |(data['price']>upper_limit) ]
print(df['price'].mean())
sns.boxplot(df['price'],color='lightpink',linewidth=2)
```

```
10634400.0
```

Out[22]: <Axes: ylabel='price'>



In [23]:
```
id=data[(data['price']<lower_limit)|(data['price']>8855000.0)].index
id
```

Out[23]:
```
Index([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18], dtype='int64')
```

In [24]:
```
data.drop(index=id,axis=0,inplace=True)
```

In [25]:
```
data[(data['price']<lower_limit)|(data['price']>8855000.0)]
```

Out[25]:

price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterheating	airconditioni

CONFIDENCE INTERVAL

$$CI = \bar{x} \pm z\frac{s}{\sqrt{n}}$$

CI = confidence interval

$\bar{x}$ = sample mean

z = confidence level value

s = sample standard deviation

n = sample size

population and sample

In [3]:
```
population=data['price']
population
```

Out[3]:
```
0      13300000
1      12250000
2      12250000
3      12215000
4      11410000
         ...
540     1820000
541     1767150
542     1750000
543     1750000
544     1750000
Name: price, Length: 545, dtype: int64
```

In [8]:
```
sample_data=data.sample(50,random_state=42)['price']
sample_data
```

Out[8]:
```
316     4060000
77      6650000
360     3710000
90      6440000
493     2800000
209     4900000
176     5250000
249     4543000
516     2450000
426     3353000
6      10150000
497     2660000
422     3360000
424     3360000
529     2275000
499     2660000
498     2660000
55      7350000
476     2940000
486     2870000
72      6720000
163     5425000
538     1890000
174     5250000
304     4193000
2      12250000
463     3080000
184     5110000
10      9800000
512     2520000
70      6790000
398     3500000
79      6650000
483     2940000
429     3325000
296     4200000
210     4900000
431     3290000
394     3500000
523     2380000
158     5495000
367     3675000
76      6650000
199     4907000
451     3150000
255     4480000
83      6580000
137     5740000
473     3003000
540     1820000
Name: price, dtype: int64
```

fucntion to caclulate CI

In [21]:
```
def cal_CI(sample , z):
    sample_mean=sample.mean()
    sample_std=sample.std()
    sample_size=len(sample)
    margin_of_error=z*(sample_std/np.sqrt(sample_size))
    lower_limit=sample_mean-margin_of_error
    upper_limit=sample_mean+margin_of_error
    if(sample_mean<upper_limit and sample_mean>lower_limit):
        print(" no significant difference in means of populationa and sample ")
    else:
        print("significant difference in means of populationa and sample ")
```

In [22]:
```
cal_CI(sample_data,1.96)
```

```
 no significant difference in means of populationa and sample
```

calculating CI of area to test

In [23]:
```
cal_CI(data.sample(1,random_state=42)['area'],1.96)
```

```
significant difference in means of populationa and sample
```

In [24]:
```
cal_CI(data.sample(50,random_state=42)['area'],1.96)
```

```
 no significant difference in means of populationa and sample
```

T-test

In [26]:
```
def ttest(sample1, sample2,alpha):
    t, p_value=stats.ttest_ind(sample1,sample2)
    if p_value<alpha:
        print(" reject Ho : there is a significant differnce")
    else:
        print("accept Ho: there is no significant difference ")
```

In [28]:
```
sample_1=data.sample(20,random_state=42)['price']
sample_2=data.sample(20,random_state=42)['price']
ttest(sample_1,sample_2,0.05)
```

```
accept Ho: there is no significant difference
```