In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('/home/joseph/Desktop/ml lab/lab1/dataset/Melbourne_housin
df.head(20)
```

Out[2]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Pos
0	Abbotsford	68 Studley St	2	h	NaN	SS	Jellis	3/09/2016	2.5	
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	
2	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	
3	Abbotsford	18/659 Victoria St	3	u	NaN	VB	Rounds	4/02/2016	2.5	
4	Abbotsford	5 Charles St	3	h	1465000.0	SP	Biggin	4/03/2017	2.5	
5	Abbotsford	40 Federation La	3	h	850000.0	PI	Biggin	4/03/2017	2.5	
6	Abbotsford	55a Park St	4	h	1600000.0	VB	Nelson	4/06/2016	2.5	
7	Abbotsford	16 Maugie St	4	h	NaN	SN	Nelson	6/08/2016	2.5	
8	Abbotsford	53 Turner St	2	h	NaN	S	Biggin	6/08/2016	2.5	
9	Abbotsford	99 Turner St	2	h	NaN	S	Collins	6/08/2016	2.5	
10	Abbotsford	129 Charles St	2	h	941000.0	S	Jellis	7/05/2016	2.5	
11	Abbotsford	124 Yarra St	3	h	1876000.0	S	Nelson	7/05/2016	2.5	
12	Abbotsford	121/56 Nicholson St	2	u	NaN	PI	Biggin	7/11/2016	2.5	
13	Abbotsford	17 Raphael St	4	h	NaN	W	Biggin	7/11/2016	2.5	
14	Abbotsford	98 Charles St	2	h	1636000.0	S	Nelson	8/10/2016	2.5	
15	Abbotsford	217 Langridge St	3	h	1000000.0	S	Jellis	8/10/2016	2.5	
16	Abbotsford	18a Mollison St	2	t	745000.0	S	Jellis	8/10/2016	2.5	
17	Abbotsford	6/241 Nicholson St	1	u	300000.0	S	Biggin	8/10/2016	2.5	
18	Abbotsford	10 Valiant St	2	h	1097000.0	S	Biggin	8/10/2016	2.5	
19	Abbotsford	403/609 Victoria St	2	u	542000.0	S	Dingle	8/10/2016	2.5	

20 rows × 21 columns

In [3]:

```
uniqueCounts = df.nunique();
print("Unique count across columns:")
print(uniqueCounts);
```

```
Unique count across columns:
Suburb             351
Address          34009
Rooms               12
Type                 3
Price             2871
Method               9
SellerG            388
Date                78
Distance           215
Postcode           211
Bedroom2            15
Bathroom            11
Car                 15
Landsize          1684
BuildingArea       740
YearBuilt          160
CouncilArea         33
Lattitude        13402
Longtitude       14524
Regionname           8
Propertycount      342
dtype: int64
```

In [4]:
```
print("The dimension of the DataFrame is:")
print(df.shape)
```

```
The dimension of the DataFrame is:
(34857, 21)
```

In [5]:
```
df2=df.fillna(0)
df2.head(30)
```

Out[5]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance
0	Abbotsford	68 Studley St	2	h	0.0	SS	Jellis	3/09/2016	2.5
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5
2	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5
3	Abbotsford	18/659 Victoria St	3	u	0.0	VB	Rounds	4/02/2016	2.5
4	Abbotsford	5 Charles St	3	h	1465000.0	SP	Biggin	4/03/2017	2.5
5	Abbotsford	40 Federation La	3	h	850000.0	PI	Biggin	4/03/2017	2.5
6	Abbotsford	55a Park St	4	h	1600000.0	VB	Nelson	4/06/2016	2.5
7	Abbotsford	16 Maugie St	4	h	0.0	SN	Nelson	6/08/2016	2.5
8	Abbotsford	53 Turner St	2	h	0.0	S	Biggin	6/08/2016	2.5
9	Abbotsford	99 Turner St	2	h	0.0	S	Collins	6/08/2016	2.5
10	Abbotsford	129 Charles St	2	h	941000.0	S	Jellis	7/05/2016	2.5
11	Abbotsford	124 Yarra St	3	h	1876000.0	S	Nelson	7/05/2016	2.5
12	Abbotsford	121/56 Nicholson St	2	u	0.0	PI	Biggin	7/11/2016	2.5
13	Abbotsford	17 Raphael St	4	h	0.0	W	Biggin	7/11/2016	2.5
14	Abbotsford	98 Charles St	2	h	1636000.0	S	Nelson	8/10/2016	2.5
15	Abbotsford	217 Langridge St	3	h	1000000.0	S	Jellis	8/10/2016	2.5
16	Abbotsford	18a Mollison St	2	t	745000.0	S	Jellis	8/10/2016	2.5
17	Abbotsford	6/241 Nicholson St	1	u	300000.0	S	Biggin	8/10/2016	2.5
18	Abbotsford	10 Valiant St	2	h	1097000.0	S	Biggin	8/10/2016	2.5
19	Abbotsford	403/609 Victoria St	2	u	542000.0	S	Dingle	8/10/2016	2.5
20	Abbotsford	2 Rich St	2	h	0.0	SP	Biggin	10/12/2016	2.5
21	Abbotsford	25/84 Trenerry Cr	2	u	760000.0	SP	Biggin	10/12/2016	2.5
22	Abbotsford	106/119 Turner St	1	u	481000.0	SP	Purplebricks	10/12/2016	2.5

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance
23	Abbotsford	411/8 Grosvenor St	2	u	700000.0	VB	Jellis	12/11/2016	2.5
24	Abbotsford	40 Nicholson St	3	h	1350000.0	VB	Nelson	12/11/2016	2.5
25	Abbotsford	123/56 Nicholson St	2	u	750000.0	S	Biggin	12/11/2016	2.5
26	Abbotsford	22 Park St	4	h	1985000.0	S	Biggin	12/11/2016	2.5
27	Abbotsford	13/84 Trenerry Cr	1	u	500000.0	S	Biggin	12/11/2016	2.5
28	Abbotsford	45 William St	2	h	1172500.0	S	Biggin	13/08/2016	2.5
29	Abbotsford	7/20 Abbotsford St	1	u	441000.0	SP	Greg	14/05/2016	2.5

30 rows × 21 columns

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
Suburb              0
Address             0
Rooms               0
Type                0
Price            7610
Method              0
SellerG             0
Date                0
Distance            1
Postcode            1
Bedroom2         8217
Bathroom         8226
Car              8728
Landsize        11810
BuildingArea    21115
YearBuilt       19306
CouncilArea         3
Lattitude        7976
Longtitude       7976
Regionname          3
Propertycount       3
dtype: int64
```

In [7]:
```
df2.shape
```

Out[7]:
```
(34857, 21)
```

In [8]:
```
uniqueCounts = df.nunique();
print("Unique count across columns:")
print(uniqueCounts);
```

```
Unique count across columns:
Suburb             351
Address          34009
Rooms               12
Type                 3
Price             2871
Method               9
SellerG            388
Date                78
Distance           215
Postcode           211
Bedroom2            15
Bathroom            11
Car                 15
Landsize          1684
BuildingArea       740
YearBuilt          160
CouncilArea         33
Lattitude        13402
Longtitude       14524
Regionname           8
Propertycount      342
dtype: int64
```

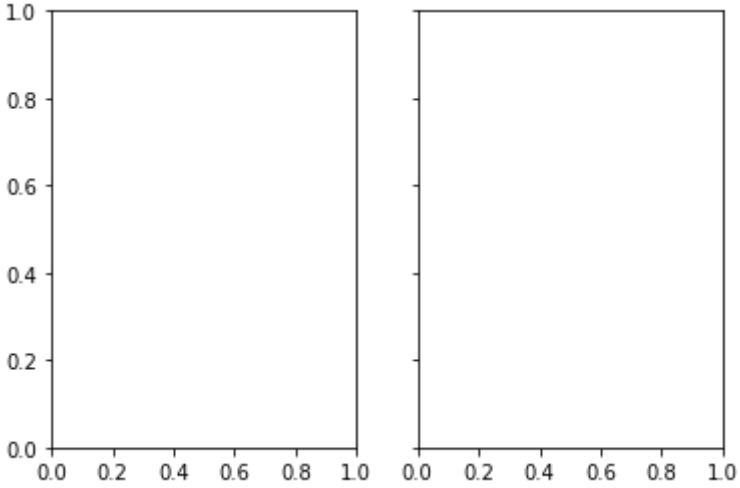
In [9]:
```
df=df[df.Price.notnull()]
df.isnull().sum()
```

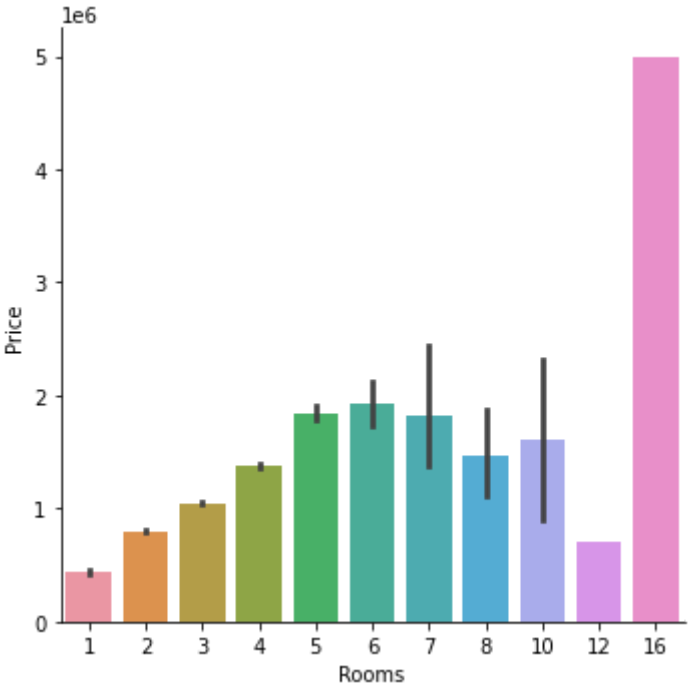
Out[9]:
```
Suburb              0
Address             0
Rooms               0
Type                0
Price               0
Method              0
SellerG             0
Date                0
Distance            1
Postcode            1
Bedroom2         6441
Bathroom         6447
Car              6824
Landsize         9265
BuildingArea    16591
YearBuilt       15163
CouncilArea         3
Lattitude        6254
Longtitude       6254
Regionname          3
Propertycount       3
dtype: int64
```

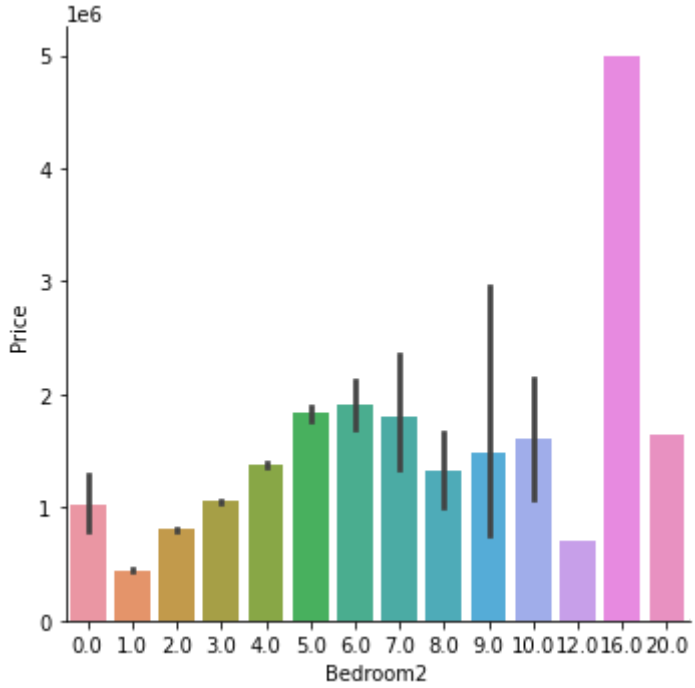
In [10]:
```
df_nobed= df[df.Bedroom2.notnull()]

fig, (ax1, ax2) = plt.subplots(ncols=2, sharey=True)
sns.factorplot(x="Rooms", y="Price",  data=df_nobed, kind="bar", ax = ax1)
sns.factorplot(x="Bedroom2", y="Price",  data=df_nobed, kind="bar",ax = ax2
plt.show()
```

```
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/seabor
n/categorical.py:3717: UserWarning: The `factorplot` function has been rena
med to `catplot`. The original name will be removed in a future release. Pl
ease update your code. Note that the default `kind` in `factorplot` (`'poin
t'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/seabor
n/categorical.py:3775: UserWarning: catplot is a figure-level function and
does not accept target axes. You may wish to try barplot
  warnings.warn(msg, UserWarning)
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/seabor
n/categorical.py:3717: UserWarning: The `factorplot` function has been rena
med to `catplot`. The original name will be removed in a future release. Pl
ease update your code. Note that the default `kind` in `factorplot` (`'poin
t'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/seabor
n/categorical.py:3775: UserWarning: catplot is a figure-level function and
does not accept target axes. You may wish to try barplot
  warnings.warn(msg, UserWarning)
```

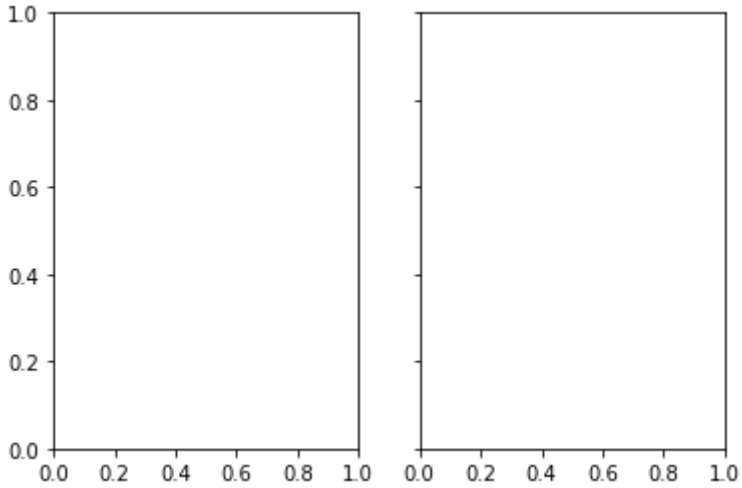


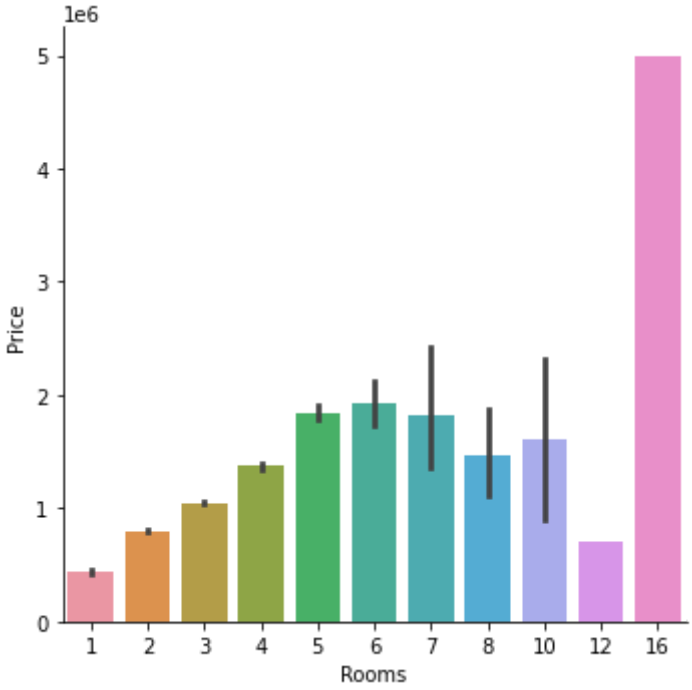


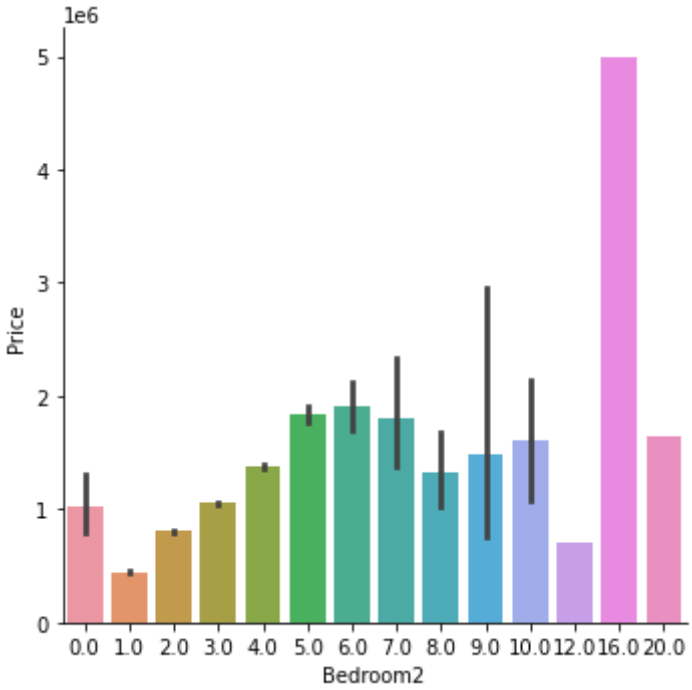


In [11]:
```
fig, (ax1, ax2) = plt.subplots(ncols=2, sharey=True)
sns.factorplot(x="Rooms", y="Price",  data=df_nobed, kind="bar", ax = ax1)
sns.factorplot(x="Bedroom2", y="Price",  data=df_nobed, kind="bar",ax = ax2
fig.show()
```

```
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/seabor
n/categorical.py:3717: UserWarning: The `factorplot` function has been rena
med to `catplot`. The original name will be removed in a future release. Pl
ease update your code. Note that the default `kind` in `factorplot` (`'poin
t'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/seabor
n/categorical.py:3775: UserWarning: catplot is a figure-level function and
does not accept target axes. You may wish to try barplot
  warnings.warn(msg, UserWarning)
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/seabor
n/categorical.py:3717: UserWarning: The `factorplot` function has been rena
med to `catplot`. The original name will be removed in a future release. Pl
ease update your code. Note that the default `kind` in `factorplot` (`'poin
t'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/seabor
n/categorical.py:3775: UserWarning: catplot is a figure-level function and
does not accept target axes. You may wish to try barplot
  warnings.warn(msg, UserWarning)
/tmp/ipykernel_636065/278694422.py:4: UserWarning: Matplotlib is currently
using module://matplotlib_inline.backend_inline, which is a non-GUI backen
d, so cannot show the figure.
  fig.show()
```







In [12]:
```
df.loc[df.Bathroom.isnull(), 'Bathroom']  = df.groupby('Rooms')['Bathroom']
df.head()
```

Out[12]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Post
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	3(
2	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	3(
4	Abbotsford	5 Charles St	3	h	1465000.0	SP	Biggin	4/03/2017	2.5	3(
5	Abbotsford	40 Federation La	3	h	850000.0	PI	Biggin	4/03/2017	2.5	3(
6	Abbotsford	55a Park St	4	h	1600000.0	VB	Nelson	4/06/2016	2.5	3(

5 rows × 21 columns

In [13]:
```
df.loc[df.Car.isnull(), 'Car']  = df.groupby('Type')['Car'].transform(lambd
df.loc[df.Landsize.isnull(), 'Landsize']  = df.groupby('Type')['Landsize'].
```

In [14]:
```
df.isnull().sum()
```

Out[14]:
```
Suburb              0
Address             0
Rooms               0
Type                0
Price               0
Method              0
SellerG             0
Date                0
Distance            1
Postcode            1
Bedroom2         6441
Bathroom            1
Car                 0
Landsize            0
BuildingArea    16591
YearBuilt       15163
CouncilArea         3
Lattitude        6254
Longtitude       6254
Regionname          3
Propertycount       3
dtype: int64
```

In [15]:
```
from numpy.core.fromnumeric import mean
print(mean(df.Price))
print(mean(df.Bedroom2))
print(mean(df.Bathroom))
print(mean(df.Car))
print(mean(df.Landsize))
print(mean(df.BuildingArea))
print(mean(df.YearBuilt))
print(mean(df.Lattitude))
print(mean(df.Longtitude))
print(mean(df.Propertycount))
```

```
1050173.344955408
3.0462366625012014
1.5629734555043613
1.6747196448874575
573.6155711905275
156.83458555743243
1966.6091525984773
-37.806963027199544
144.99671101938742
7566.781089414183
```

In [16]:
```
df = df.fillna(0)
```

In [17]:
```
df['Price'] = df['Price'].replace([0],1050173)
df['Bedroom2'] = df['Bedroom2'].replace([0],3.0)
df['Bathroom'] = df['Bathroom'].replace([0],1.0)
df['Car'] = df['Car'].replace([0],1.0)
df['Landsize'] = df['Landsize'].replace([0],593.59)
df['BuildingArea'] = df['BuildingArea'].replace([0],160.25)
df['YearBuilt'] = df['YearBuilt'].replace([0],1965.28)
df['Lattitude'] = df['Lattitude'].replace([0],-37.810)
df['Longtitude'] = df['Longtitude'].replace([0],145.00)
df['Propertycount'] = df['Propertycount'].replace([0],7572)
```

In [18]:
```
df
```

Out[18]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Dis
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	
2	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	
4	Abbotsford	5 Charles St	3	h	1465000.0	SP	Biggin	4/03/2017	
5	Abbotsford	40 Federation La	3	h	850000.0	PI	Biggin	4/03/2017	
6	Abbotsford	55a Park St	4	h	1600000.0	VB	Nelson	4/06/2016	
...	...	...	...	...	...	...	...	...	...
34852	Yarraville	13 Burns St	4	h	1480000.0	PI	Jas	24/02/2018	
34853	Yarraville	29A Murray St	2	h	888000.0	SP	Sweeney	24/02/2018	
34854	Yarraville	147A Severn St	2	t	705000.0	S	Jas	24/02/2018	
34855	Yarraville	12/37 Stephen St	3	h	1140000.0	SP	hockingstuart	24/02/2018	
34856	Yarraville	3 Tarrengower St	2	h	1020000.0	PI	RW	24/02/2018	

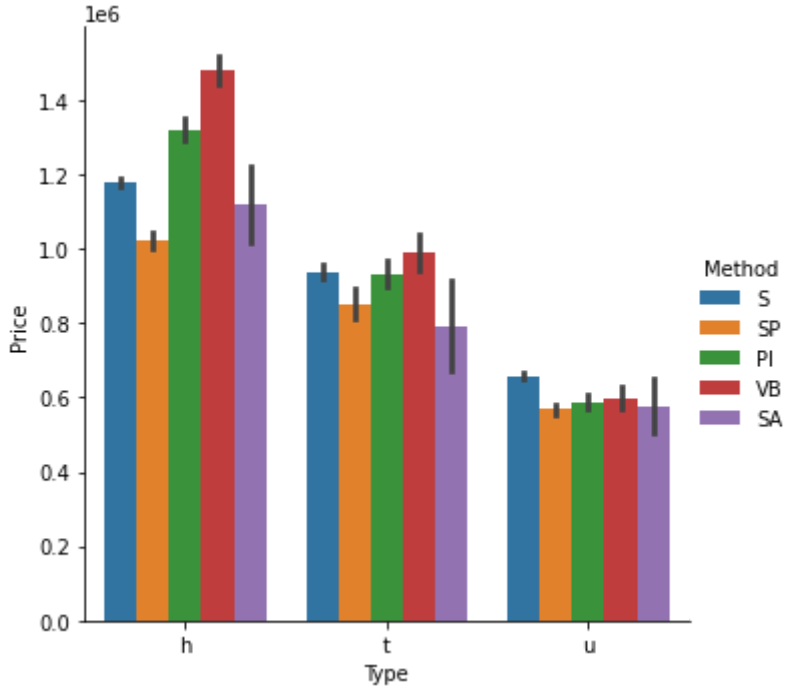
27247 rows × 21 columns

In [19]:
```
df.isnull().sum()
```

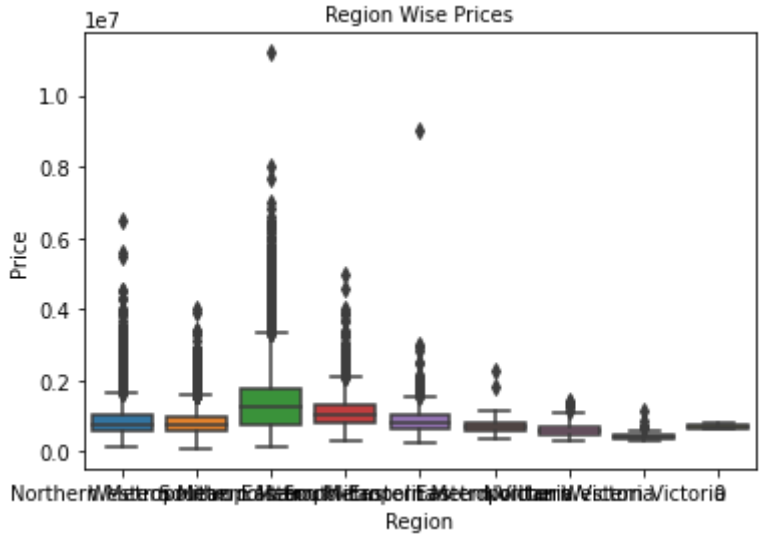
Out[19]:
```
Suburb           0
Address          0
Rooms            0
Type             0
Price            0
Method           0
SellerG          0
Date             0
Distance         0
Postcode         0
Bedroom2         0
Bathroom         0
Car              0
Landsize         0
BuildingArea     0
YearBuilt        0
CouncilArea      0
Lattitude        0
Longtitude       0
Regionname       0
Propertycount    0
dtype: int64
```

In [20]:
```
sns.factorplot(x="Type", y="Price", hue="Method", data=df, kind="bar");
```

```
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/seabor
n/categorical.py:3717: UserWarning: The `factorplot` function has been rena
med to `catplot`. The original name will be removed in a future release. Pl
ease update your code. Note that the default `kind` in `factorplot` (`'poin
t'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
```

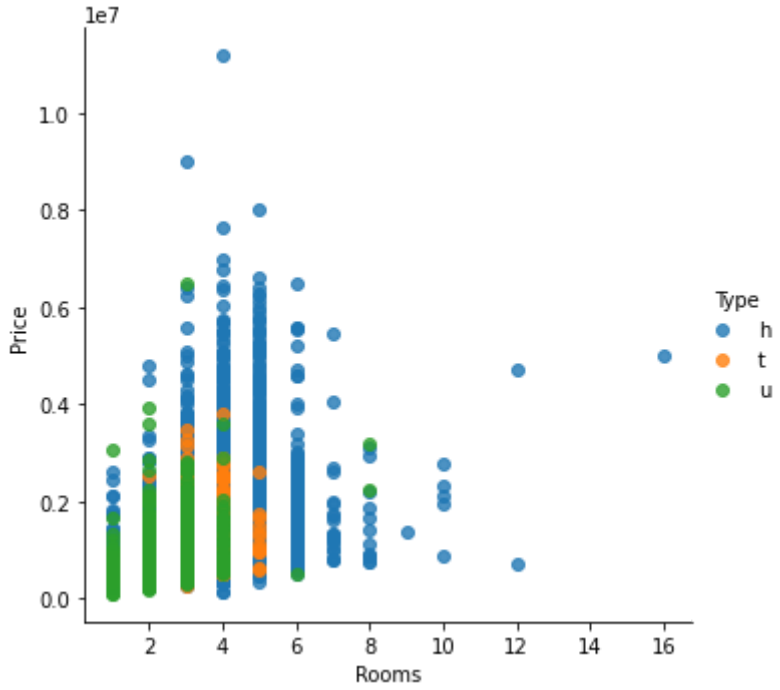


In [21]:
```
b=sns.boxplot(x="Regionname", y="Price", data=df, dodge=False);
b.axes.set_title("Region Wise Prices",fontsize=10)
b.set_xlabel("Region",fontsize=10)
b.set_ylabel("Price",fontsize=10)
b.tick_params(labelsize=10)
```



In [22]:
```
facet = sns.lmplot(data=df, x='Rooms', y='Price', hue='Type',
                   fit_reg=False, legend=True, legend_out=True)
```

```
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/seabor
n/regression.py:592: UserWarning: legend_out is deprecated from the `lmplot
` function signature. Please update your code to pass it using `facet_kws`.
  warnings.warn(msg, UserWarning)
```



In [23]:
```
df.groupby('Regionname')[['SellerG']].count()
```

Out[23]:

Regionname	SellerG
0	3
Eastern Metropolitan	3272
Eastern Victoria	166
Northern Metropolitan	7864
Northern Victoria	166
South-Eastern Metropolitan	1341
Southern Metropolitan	8524
Western Metropolitan	5815
Western Victoria	96

In [24]:
```
df.groupby('Regionname', as_index=False).agg({"Car": "sum"})
```

Out[24]:

	Regionname	Car
0	0	4.855795
1	Eastern Metropolitan	6018.248011
2	Eastern Victoria	341.590006
3	Northern Metropolitan	12774.714606
4	Northern Victoria	353.327965
5	South-Eastern Metropolitan	2572.975491
6	Southern Metropolitan	14139.755843
7	Western Metropolitan	10480.900089
8	Western Victoria	195.718358

In [25]:
```
df.groupby('Regionname', as_index=False).agg({"Car": "sum",'SellerG': "coun
```

Out[25]:

	Regionname	Car	SellerG	Rooms
0	0	4.855795	3	7
1	Eastern Metropolitan	6018.248011	3272	11010
2	Eastern Victoria	341.590006	166	584
3	Northern Metropolitan	12774.714606	7864	22055
4	Northern Victoria	353.327965	166	574
5	South-Eastern Metropolitan	2572.975491	1341	4358
6	Southern Metropolitan	14139.755843	8524	24735
7	Western Metropolitan	10480.900089	5815	17890
8	Western Victoria	195.718358	96	318

In [45]:
```
# Group the data frame by month and item and extract a number of stats from
df.groupby(['Regionname']).agg({'Rooms':sum,      # find the sum of the dur
                                'SellerG': "count" # find the number o
                                })    # get the first date per group
```

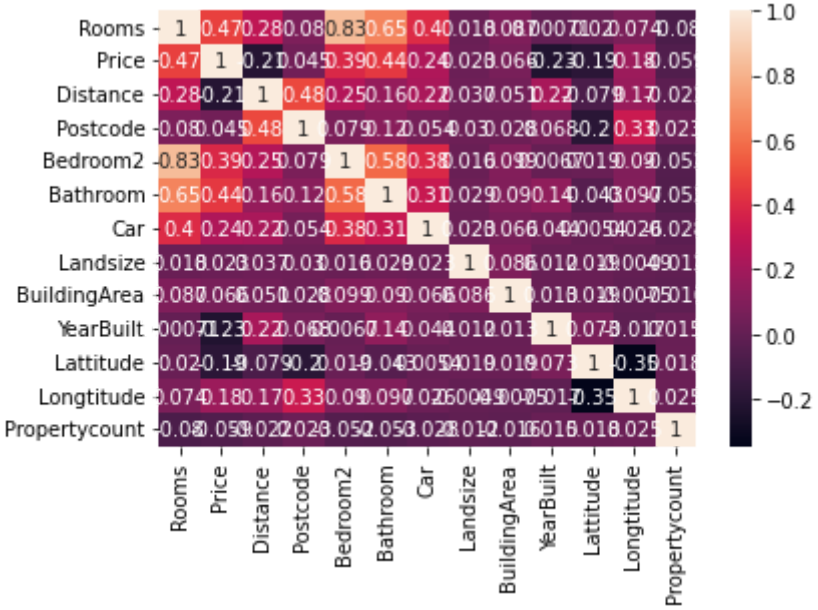
Out[45]:

Regionname	Rooms	SellerG
0	7	3
Eastern Metropolitan	11010	3272
Eastern Victoria	584	166
Northern Metropolitan	22055	7864
Northern Victoria	574	166
South-Eastern Metropolitan	4358	1341
Southern Metropolitan	24735	8524
Western Metropolitan	17890	5815
Western Victoria	318	96

In [27]:
```
df.loc[df.Lattitude.isnull(), 'Lattitude']  = df.groupby('Suburb')['Lattitu
df.loc[df.Longtitude.isnull(), 'Longtitude']  = df.groupby('Suburb')['Longt
```

In [28]:
```
df=df.dropna(axis=0, how='any')
```

In [29]:
```
sns.heatmap(df.corr(), annot = True)
plt.show()
```



In [30]:
```
print(df.dtypes) # dtypes returns a Series with the data type of each colum
print("Categorical Variables:")
print(df.columns[df.dtypes == 'object'])
print("Numeric Variables:")
print(df.columns[(df.dtypes == 'int64') | (df.dtypes == 'float64')])
```

```
Suburb           object
Address          object
Rooms             int64
Type             object
Price           float64
Method           object
SellerG          object
Date             object
Distance        float64
Postcode        float64
Bedroom2        float64
Bathroom        float64
Car             float64
Landsize        float64
BuildingArea    float64
YearBuilt       float64
CouncilArea      object
Lattitude       float64
Longtitude      float64
Regionname       object
Propertycount   float64
dtype: object
Categorical Variables:
Index(['Suburb', 'Address', 'Type', 'Method', 'SellerG', 'Date', 'CouncilAr
ea',
       'Regionname'],
      dtype='object')
Numeric Variables:
Index(['Rooms', 'Price', 'Distance', 'Postcode', 'Bedroom2', 'Bathroom', 'C
ar',
       'Landsize', 'BuildingArea', 'YearBuilt', 'Lattitude', 'Longtitude',
       'Propertycount'],
      dtype='object')
```

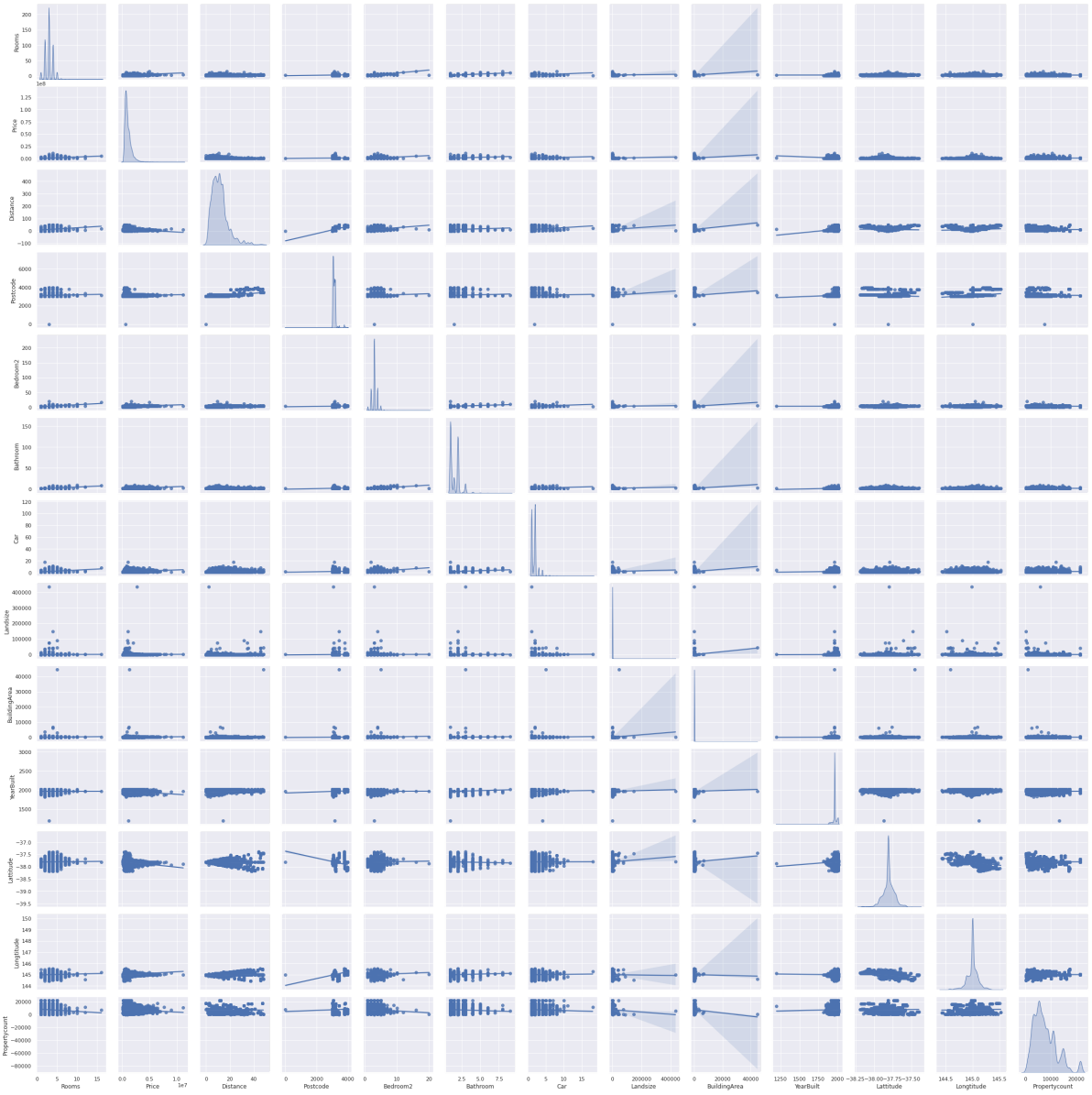
In [212…

```
plt.figure(figsize=(15,8)) #Creating a new figure with given width and heig
sns.pairplot(df, kind='reg', diag_kind='kde') # Plotting multiple pairwise
```

Out[212]:

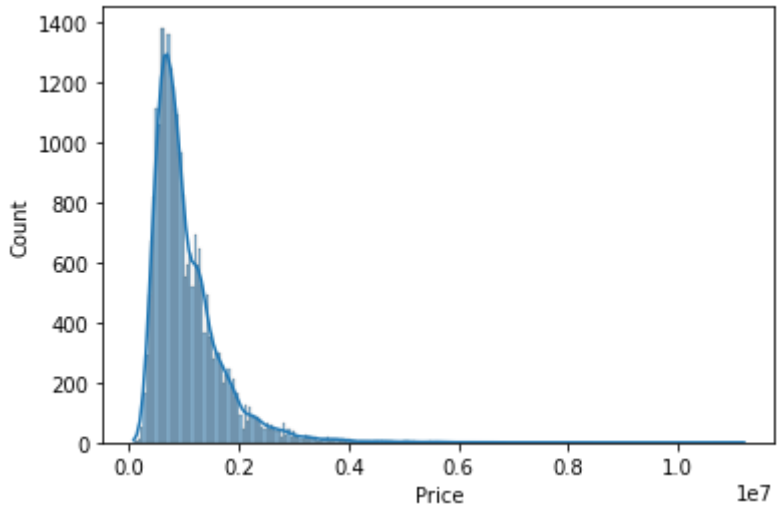
```
<seaborn.axisgrid.PairGrid at 0x7fb2979984d0>

<Figure size 1080x576 with 0 Axes>
```



In [31]:
```
sns.histplot(df, x = "Price",kde = True)
```

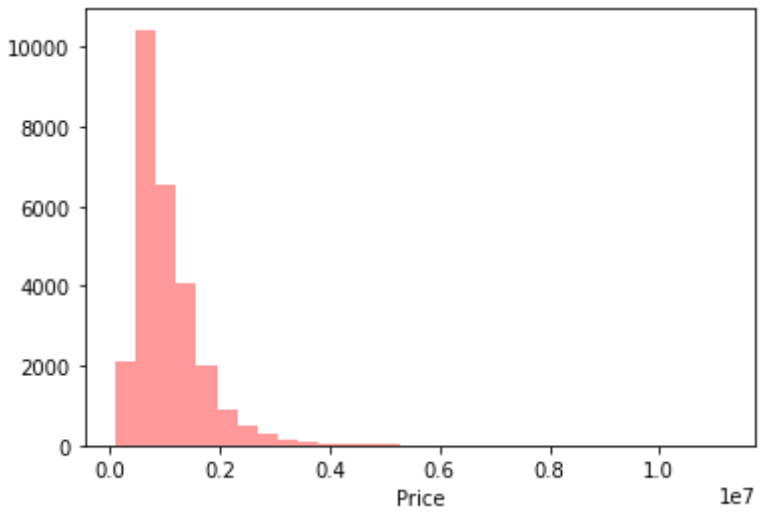
Out[31]:
```
<AxesSubplot:xlabel='Price', ylabel='Count'>
```



In [32]:
```
sns.distplot(df['Price'], kde = False, color ='red', bins = 30)
```

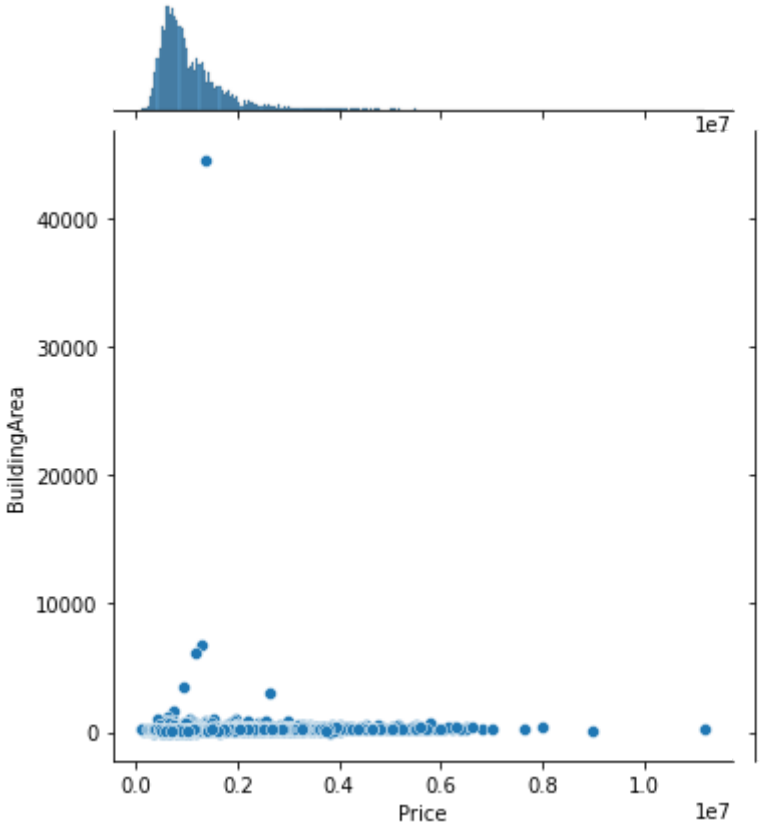
```
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/seabor
n/distributions.py:2619: FutureWarning: `distplot` is a deprecated function
and will be removed in a future version. Please adapt your code to use eith
er `displot` (a figure-level function with similar flexibility) or `histplo
t` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[32]:
```
<AxesSubplot:xlabel='Price'>
```



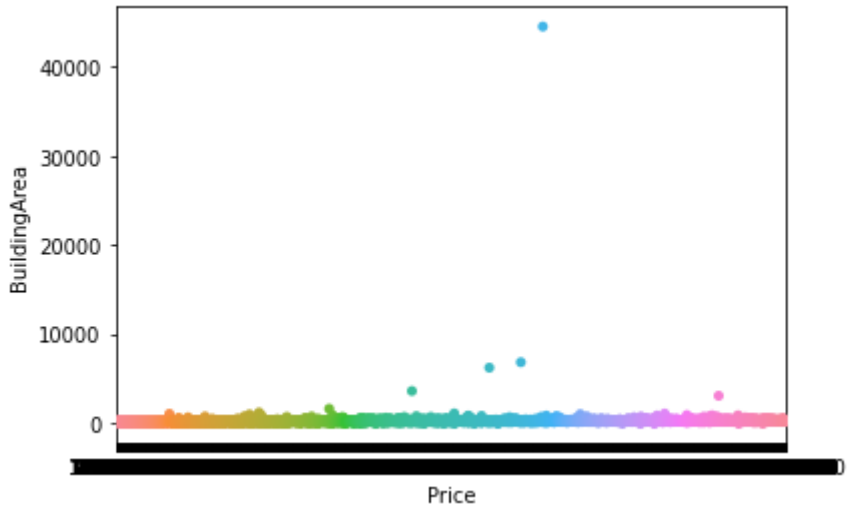
In [33]:
```
sns.jointplot(x = "Price", y = "BuildingArea",
              kind = "scatter", data = df)
```

Out[33]:
```
<seaborn.axisgrid.JointGrid at 0x7efd015ba4d0>
```



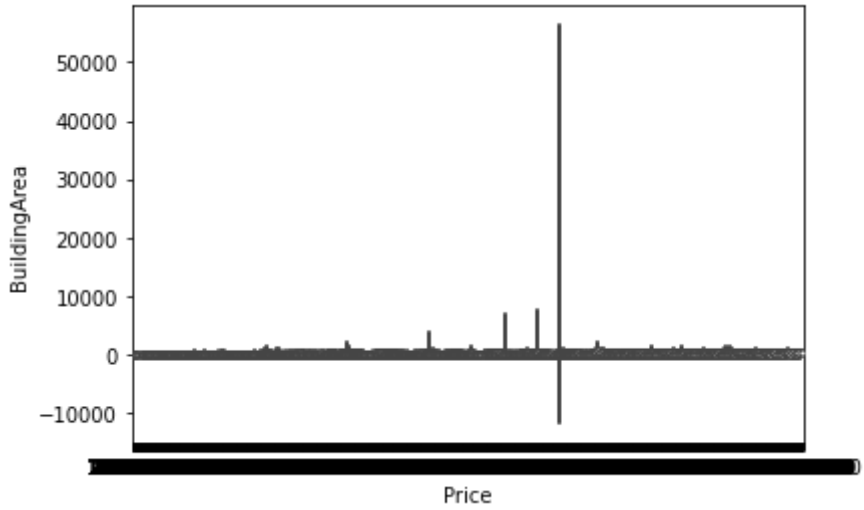
In [34]:
```
sns.stripplot(x="Price", y="BuildingArea", data=df)
```

Out[34]:
```
<AxesSubplot:xlabel='Price', ylabel='BuildingArea'>
```



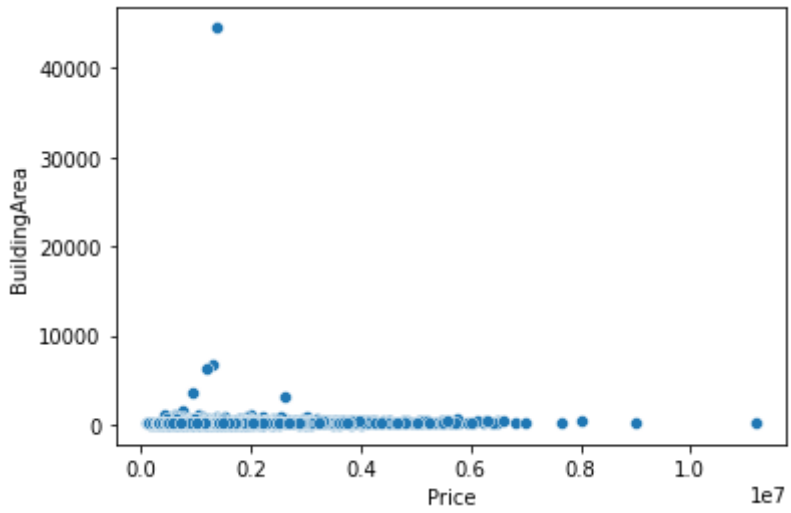
In [35]:
```
sns.violinplot(x ="Price",
               y ="BuildingArea",
               data = df)
```

Out[35]:
```
<AxesSubplot:xlabel='Price', ylabel='BuildingArea'>
```



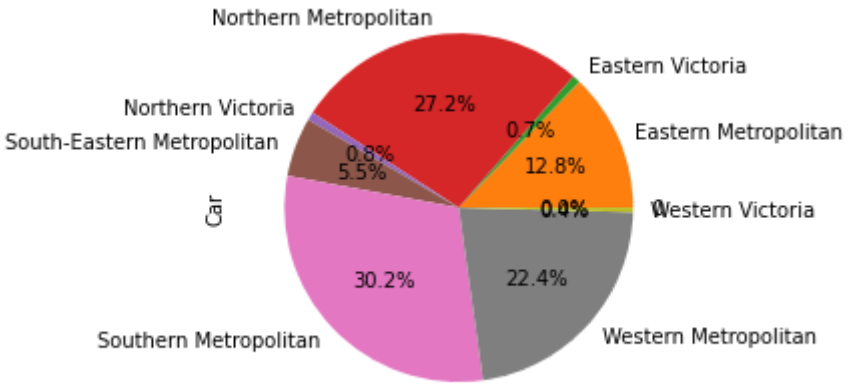
In [36]:
```
sns.scatterplot(x="Price",
                    y="BuildingArea",
                    data=df)
```

Out[36]:
```
<AxesSubplot:xlabel='Price', ylabel='BuildingArea'>
```



In [49]:
```
data = df.groupby("Regionname")["Car"].sum()
```

```
data
data.plot.pie(autopct="%.1f%%");
```



In [50]:
```
df.corr()
```

Out[50]:

	Rooms	Price	Distance	Postcode	Bedroom2	Bathroom	Car	La
Rooms	1.000000	0.465238	0.284283	0.080173	0.834705	0.654197	0.399976	0.0
Price	0.465238	1.000000	-0.211331	0.045002	0.386647	0.440617	0.235202	0.0
Distance	0.284283	-0.211331	1.000000	0.484488	0.251813	0.156197	0.217068	0.0
Postcode	0.080173	0.045002	0.484488	1.000000	0.078579	0.115921	0.054096	0.0
Bedroom2	0.834705	0.386647	0.251813	0.078579	1.000000	0.575976	0.384802	0.0
Bathroom	0.654197	0.440617	0.156197	0.115921	0.575976	1.000000	0.310758	0.0
Car	0.399976	0.235202	0.217068	0.054096	0.384802	0.310758	1.000000	0.0
Landsize	0.018278	0.022520	0.036738	0.030053	0.015832	0.029299	0.022585	1.0
BuildingArea	0.087059	0.066302	0.050686	0.028455	0.099306	0.089627	0.065660	0.0
YearBuilt	0.000712	-0.231196	0.215129	0.067811	0.006708	0.138303	0.044023	0.0
Lattitude	0.020317	-0.190843	-0.079475	-0.195316	0.019405	-0.043060	0.005385	0.0
Longtitude	0.074404	0.175325	0.168095	0.325903	0.090329	0.097237	0.026195	-0.0
Propertycount	-0.079569	-0.059016	-0.021704	0.022888	-0.051718	-0.052991	-0.028421	-0.0

In [52]:
```
pip install -U scikit-learn
```

```
Collecting scikit-learn
  Using cached scikit_learn-1.1.1-cp310-cp310-manylinux_2_17_x86_64.manylin
ux2014_x86_64.whl (30.4 MB)
Collecting threadpoolctl>=2.0.0
  Using cached threadpoolctl-3.1.0-py3-none-any.whl (14 kB)
Requirement already satisfied: scipy>=1.3.2 in ./mlenv/lib/python3.10/site-
packages (from scikit-learn) (1.8.1)
Collecting joblib>=1.0.0
  Using cached joblib-1.1.0-py2.py3-none-any.whl (306 kB)
Requirement already satisfied: numpy>=1.17.3 in ./mlenv/lib/python3.10/site
-packages (from scikit-learn) (1.23.1)
Installing collected packages: threadpoolctl, joblib, scikit-learn
Successfully installed joblib-1.1.0 scikit-learn-1.1.1 threadpoolctl-3.1.0
Note: you may need to restart the kernel to use updated packages.
```

In [53]:
```
import numpy as np
import pandas as pd
```

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier

%matplotlib inline
```

In [54]:

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values = np.nan,
                        strategy ='mean')


print("Original Data : \n", df)
# Fitting the data to the imputer object
imputer = imputer.fit(df)

# Imputing the data
data = imputer.transform(df)

print("Imputed Data : \n", df)
```

```
Original Data :
           Suburb           Address  Rooms Type      Price Method  \
1      Abbotsford      85 Turner St      2    h  1480000.0      S
2      Abbotsford   25 Bloomburg St      2    h  1035000.0      S
4      Abbotsford      5 Charles St      3    h  1465000.0     SP
5      Abbotsford  40 Federation La      3    h   850000.0     PI
6      Abbotsford       55a Park St      4    h  1600000.0     VB
...           ...               ...    ...  ...        ...    ...
34852  Yarraville       13 Burns St      4    h  1480000.0     PI
34853  Yarraville     29A Murray St      2    h   888000.0     SP
34854  Yarraville    147A Severn St      2    t   705000.0      S
34855  Yarraville  12/37 Stephen St      3    h  1140000.0     SP
34856  Yarraville  3 Tarrengower St      2    h  1020000.0     PI

             SellerG        Date  Distance  Postcode  ...  Bathroom      Ca
r  \
1             Biggin   3/12/2016       2.5    3067.0  ...  1.000000  1.0000
0
2             Biggin   4/02/2016       2.5    3067.0  ...  1.000000  1.0000
0
4             Biggin   4/03/2017       2.5    3067.0  ...  2.000000  1.0000
0
5             Biggin   4/03/2017       2.5    3067.0  ...  2.000000  1.0000
0
6             Nelson   4/06/2016       2.5    3067.0  ...  1.000000  2.0000
0
...              ...         ...       ...       ...  ...       ...
...
34852            Jas  24/02/2018       6.3    3013.0  ...  1.000000  3.0000
0
34853        Sweeney  24/02/2018       6.3    3013.0  ...  2.000000  1.0000
0
34854            Jas  24/02/2018       6.3    3013.0  ...  1.000000  2.0000
0
34855  hockingstuart  24/02/2018       6.3    3013.0  ...  1.483603  1.8581
1
34856             RW  24/02/2018       6.3    3013.0  ...  1.000000  1.0000
0

         Landsize  BuildingArea  YearBuilt               CouncilArea  \
1      202.000000        160.25    1965.28        Yarra City Council
2      156.000000         79.00    1900.00        Yarra City Council
4      134.000000        150.00    1900.00        Yarra City Council
5       94.000000        160.25    1965.28        Yarra City Council
6      120.000000        142.00    2014.00        Yarra City Council
...           ...           ...        ...                       ...
34852  593.000000        160.25    1965.28  Maribyrnong City Council
34853   98.000000        104.00    2018.00  Maribyrnong City Council
34854  220.000000        120.00    2000.00  Maribyrnong City Council
34855  648.716257        160.25    1965.28  Maribyrnong City Council
34856  250.000000        103.00    1930.00  Maribyrnong City Council

      Lattitude  Longtitude             Regionname Propertycount
1     -37.79960   144.99840  Northern Metropolitan        4019.0
2     -37.80790   144.99340  Northern Metropolitan        4019.0
4     -37.80930   144.99440  Northern Metropolitan        4019.0
5     -37.79690   144.99690  Northern Metropolitan        4019.0
6     -37.80720   144.99410  Northern Metropolitan        4019.0
...         ...         ...                    ...          ...
34852 -37.81053   144.88467   Western Metropolitan        6543.0
34853 -37.81551   144.88826   Western Metropolitan        6543.0
34854 -37.82286   144.87856   Western Metropolitan        6543.0
34855 -37.81000   145.00000   Western Metropolitan        6543.0
34856 -37.81810   144.89351   Western Metropolitan        6543.0
```

```
[27247 rows x 21 columns]
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [54], in <cell line: 8>()
      6 print("Original Data : \n", df)
      7 # Fitting the data to the imputer object
----> 8 imputer = imputer.fit(df)
     10 # Imputing the data
     11 data = imputer.transform(df)

File ~/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/sklearn/impute/_base.py:345, in SimpleImputer.fit(self, X, y)
    336 if self.verbose != "deprecated":
    337     warnings.warn(
    338         "The 'verbose' parameter was deprecated in version "
    339         "1.1 and will be removed in 1.3. A warning will "
   (...)
    342         FutureWarning,
    343     )
--> 345 X = self._validate_input(X, in_fit=True)
    347 # default fill_value is 0 for numerical input and "missing_value"
    348 # otherwise
    349 if self.fill_value is None:

File ~/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/sklearn/impute/_base.py:302, in SimpleImputer._validate_input(self, X, in_fit)
    296 if "could not convert" in str(ve):
    297     new_ve = ValueError(
    298         "Cannot use {} strategy with non-numeric data:\n{}".format(
    299             self.strategy, ve
    300         )
    301     )
--> 302     raise new_ve from None
    303 else:
    304     raise ve

ValueError: Cannot use mean strategy with non-numeric data:
could not convert string to float: 'Abbotsford'
```

In [55]:
```
df.head()
```

Out[55]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Post
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	3(
2	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	3(
4	Abbotsford	5 Charles St	3	h	1465000.0	SP	Biggin	4/03/2017	2.5	3(
5	Abbotsford	40 Federation La	3	h	850000.0	PI	Biggin	4/03/2017	2.5	3(
6	Abbotsford	55a Park St	4	h	1600000.0	VB	Nelson	4/06/2016	2.5	3(

5 rows × 21 columns

In [56]:
```
import numpy as np
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
import seaborn as sns
```

In [75]:
```
y = df ['Type']
X = df.drop('Address', axis = 1)
X = X.drop('CouncilArea', axis = 1)
X = X.drop('Regionname', axis = 1)
X = X.drop('Suburb', axis = 1)
X = X.drop('SellerG', axis = 1)
X = X.drop('Type', axis = 1)
X = X.drop('Method', axis = 1)

X = X.drop('Date', axis = 1)
# Separating the dependent and independent variable

X_train, X_test, y_train, y_test = train_test_split(
            X, y, test_size = 0.3, random_state = 0)
```

In [76]:
```
K = []
training = []
test = []
scores = {}

for k in range(2, 21):
    clf = KNeighborsClassifier(n_neighbors = k)
    clf.fit(X_train, y_train)

    training_score = clf.score(X_train, y_train)
    test_score = clf.score(X_test, y_test)
    K.append(k)

    training.append(training_score)
    test.append(test_score)
    scores[k] = [training_score, test_score]
```

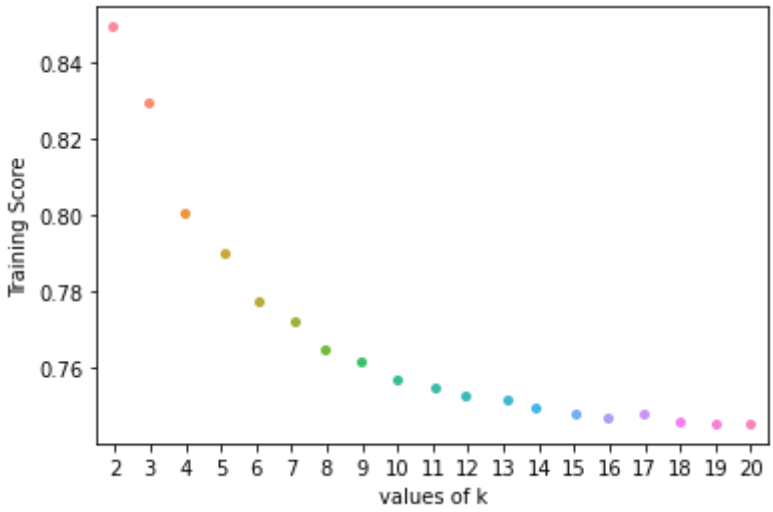
In [77]:
```
for keys, values in scores.items():
    print(keys, ':', values)
```

```
2 : [0.8492554530201343, 0.7155963302752294]
3 : [0.8292260906040269, 0.7138837920489297]
4 : [0.8005977348993288, 0.7217125382262997]
5 : [0.790268456375839, 0.7191437308868501]
6 : [0.7775796979865772, 0.7212232415902141]
7 : [0.7720218120805369, 0.7188990825688073]
8 : [0.7651006711409396, 0.7191437308868501]
9 : [0.7615352348993288, 0.7241590214067278]
10 : [0.7568162751677853, 0.7237920489296636]
11 : [0.7547713926174496, 0.7242813455657492]
12 : [0.7530411073825504, 0.7286850152905199]
13 : [0.7515729865771812, 0.7308868501529052]
14 : [0.7494232382550335, 0.7269724770642202]
15 : [0.7483221476510067, 0.730519877675841]
16 : [0.7470113255033557, 0.7285626911314985]
17 : [0.7482697147651006, 0.7295412844036697]
18 : [0.7462248322147651, 0.7303975535168196]
19 : [0.7454907718120806, 0.7294189602446484]
20 : [0.7455432046979866, 0.7300305810397554]
```

In [78]:
```
ax = sns.stripplot(K, training);
ax.set(xlabel ='values of k', ylabel ='Training Score')
```

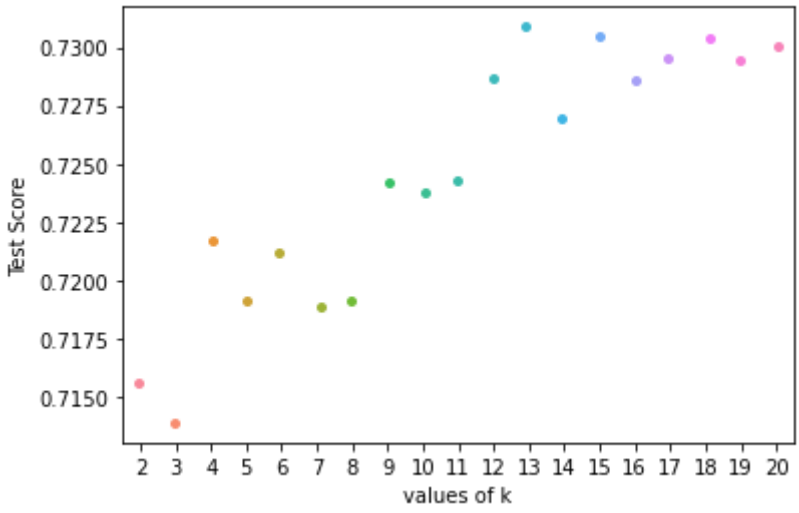
```
plt.show()
```

```
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/seabor
n/_decorators.py:36: FutureWarning: Pass the following variables as keyword
args: x, y. 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 misinterpretation.
  warnings.warn(
```

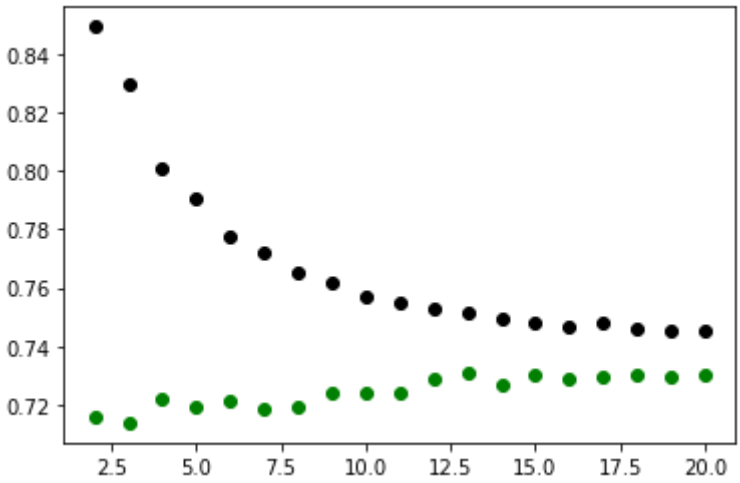


In [79]:
```
ax = sns.stripplot(K, test);
ax.set(xlabel ='values of k', ylabel ='Test Score')
plt.show()
```

```
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/seabor
n/_decorators.py:36: FutureWarning: Pass the following variables as keyword
args: x, y. 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 misinterpretation.
  warnings.warn(
```



In [80]:
```
plt.scatter(K, training, color ='k')
plt.scatter(K, test, color ='g')
plt.show()
# For overlapping scatter plots
```



In [ ]: