

```
import pandas as pd
```

```
import numpy as np
```

```
pwd
```

```
'C:\\Users\\Administrator'
```

```
df = pd.read_csv("Housing.csv")
```

```
df.head()
```

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

	hotwaterheating	airconditioning	parking	prefarea	furnishingstatus
0	no	yes	2	yes	furnished
1	no	yes	3	no	furnished
2	no	no	2	yes	semi-furnished
3	no	yes	3	yes	furnished
4	no	yes	2	no	furnished

```
df.describe()
```

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

```
count    parking
545.000000
mean      0.693578
std       0.861586
min       0.000000
25%       0.000000
50%       0.000000
75%       1.000000
max       3.000000
```

Handle Missing Values

```
missing_counts = df.isnull().sum()
```

```
df.isnull().sum()
```

```
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
```

```
len(df)
```

```
545
```

There is no need to create a cleaned dataset (CSV) if there are no missing values.

Detect Outliers using IQR

```
numeric_cols =  
['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'parking']
```

```
Q1 = df[numeric_cols].quantile(0.25)
```

```
Q3 = df[numeric_cols].quantile(0.75)
```

```
IQR = Q3 - Q1
```

```
lower_bound = Q1 - 1.5 * IQR
```

```

upper_bound = Q3 + 1.5 * IQR

outlier_mask = (df[numeric_cols] < (Q1 - 1.5 * IQR)) |
(df[numeric_cols] > (Q3 + 1.5 * IQR))

outliers = outlier_mask.any(axis=1)

print(f"Total outlier rows detected (IQR method): {outliers.sum()}")
Total outlier rows detected (IQR method): 82

#Detect Outliers using Z-score

from scipy import stats

z_scores = np.abs(stats.zscore(df[numeric_cols]))
z_outliers = (z_scores > 3).any(axis=1)

print(f"Total outlier rows detected (Z-score method):
{z_outliers.sum()}")
Total outlier rows detected (Z-score method): 23

df['price'] = pd.to_numeric(df['price'], errors='coerce')
df = df.dropna(subset=['price'])
df['price'] = df['price'].astype(float)
cap_price = np.percentile(df['price'], 99)
cap_area = np.percentile(df['area'], 99)
clean_df = df.copy()

# Handle Outliers by Capping

clean_df.loc[clean_df['price'] > cap_price, 'price'] = cap_price
clean_df.loc[clean_df['area'] > cap_area, 'area'] = cap_area

print("Old max price vs new max price:", df['price'].max(), "->",
clean_df['price'].max())

Old max price vs new max price: 13300000.0 -> 10541999.999999963

print("Old max area vs new max area:", df['area'].max(), "->",
clean_df['area'].max())

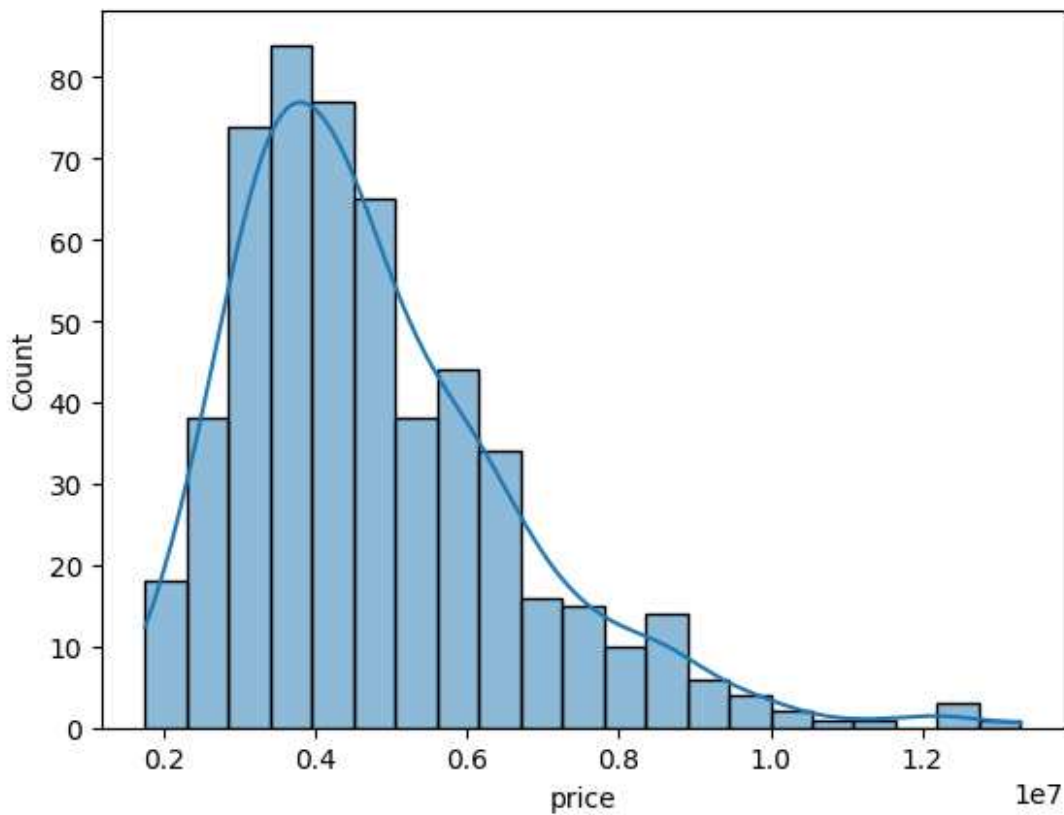
Old max area vs new max area: 16200 -> 12543.599999999955

clean_df.to_csv('Housing_cleaned.csv', index=False)

#####

```

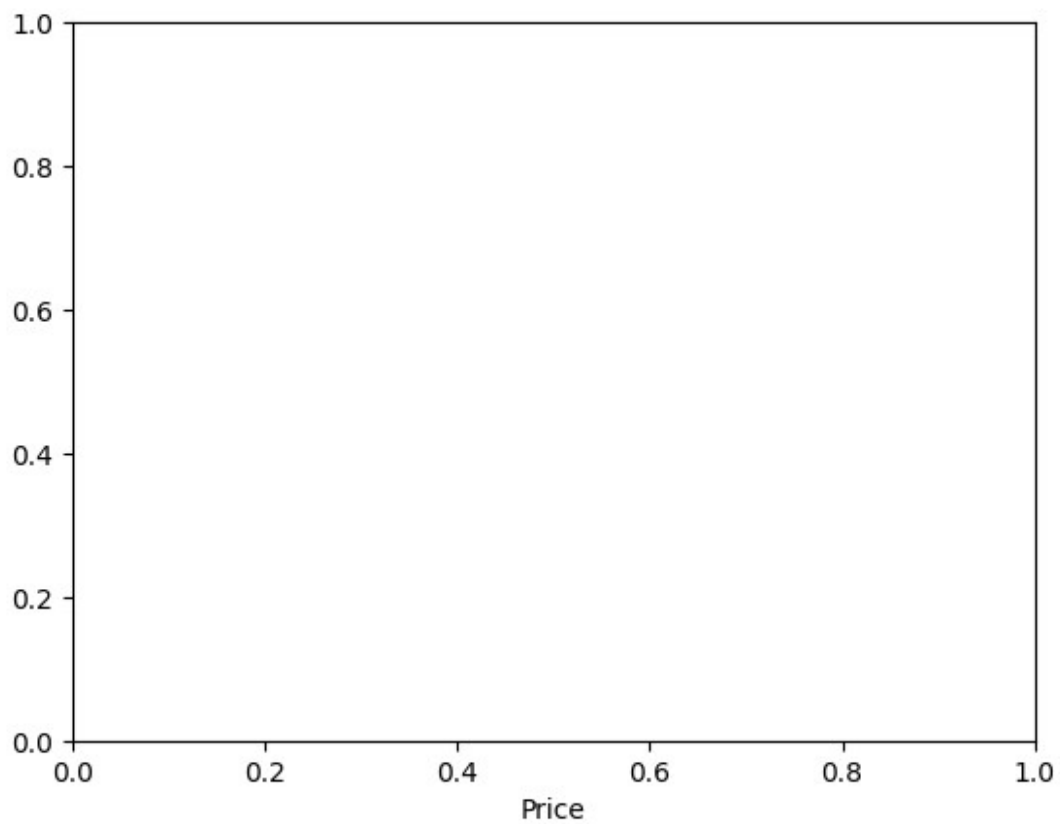
```
import matplotlib.pyplot as plt
import seaborn as sns
# Data Visualization
# Histogram of Price (original data)
plt.figure(figsize=(6,4))
<Figure size 600x400 with 0 Axes>
<Figure size 600x400 with 0 Axes>
sns.histplot(df['price'], kde=True)
<Axes: xlabel='price', ylabel='Count'>
```



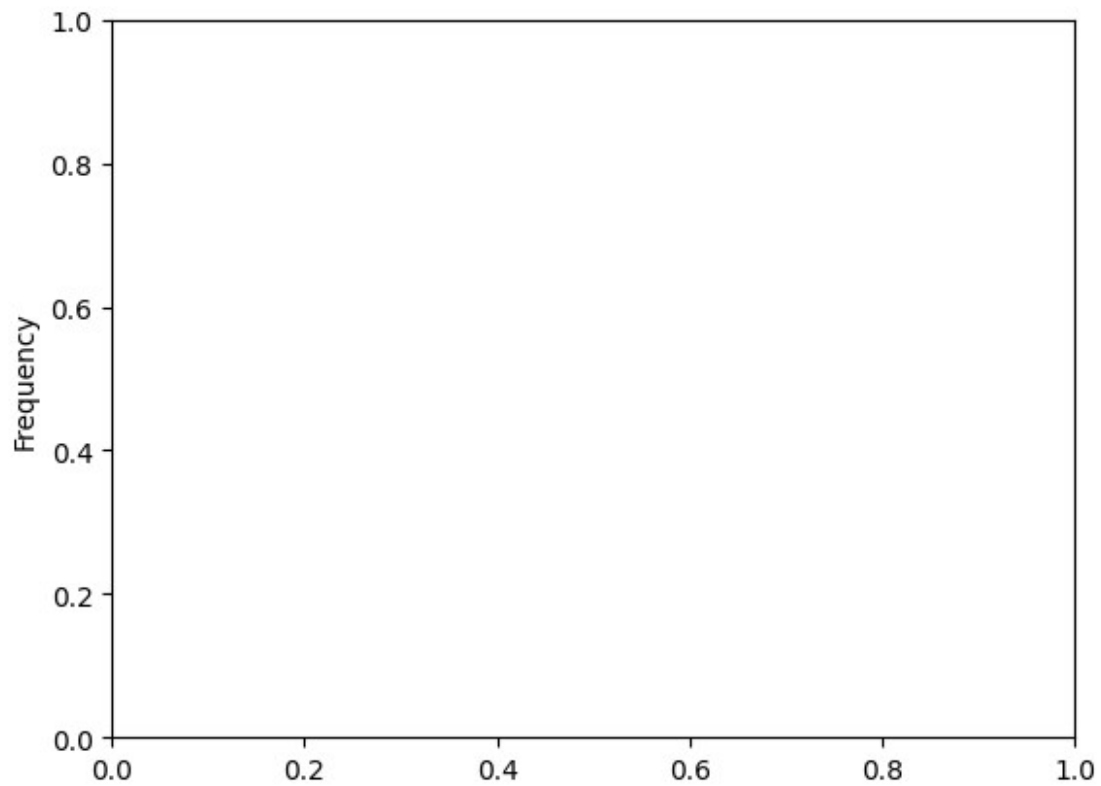
```
plt.title('Distribution of House Prices')
Text(0.5, 1.0, 'Distribution of House Prices')
```



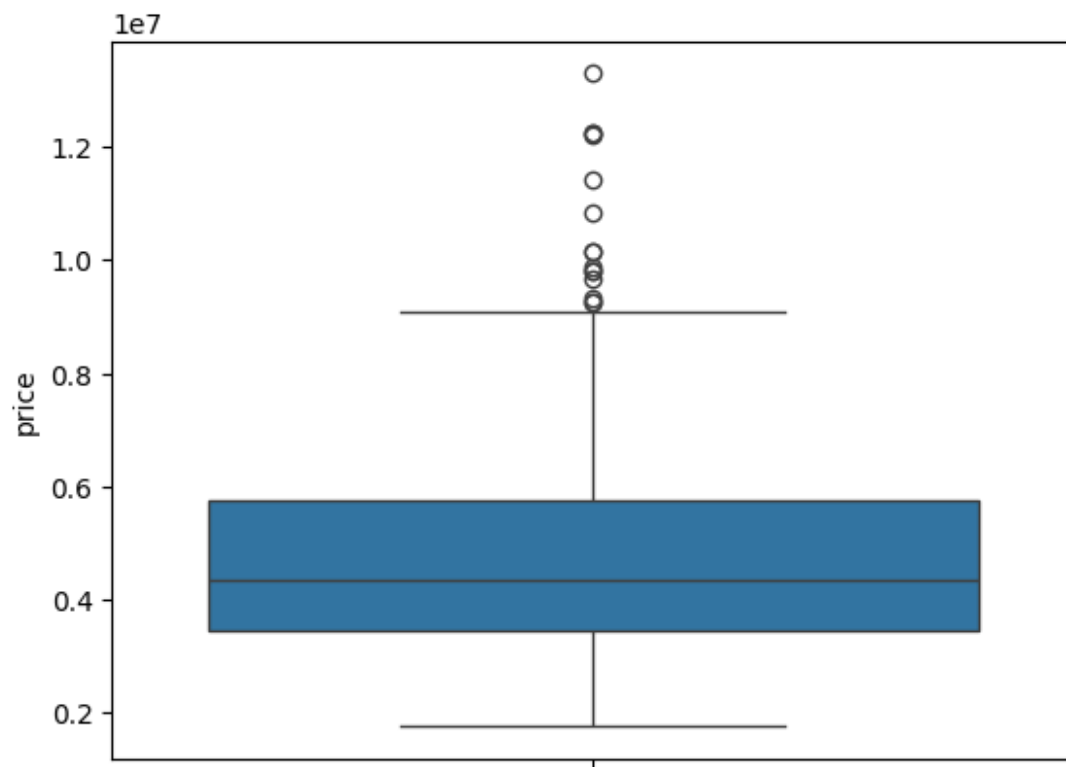
```
plt.xlabel('Price')  
Text(0.5, 0, 'Price')
```



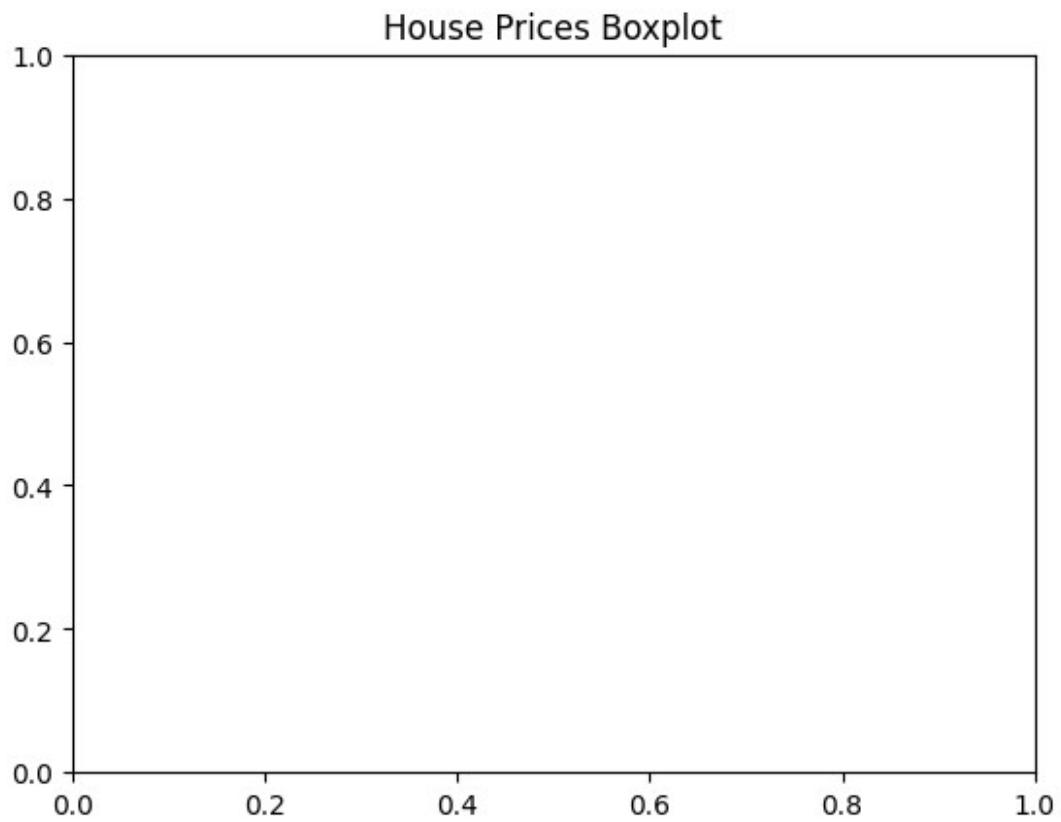
```
plt.ylabel('Frequency')  
Text(0, 0.5, 'Frequency')
```



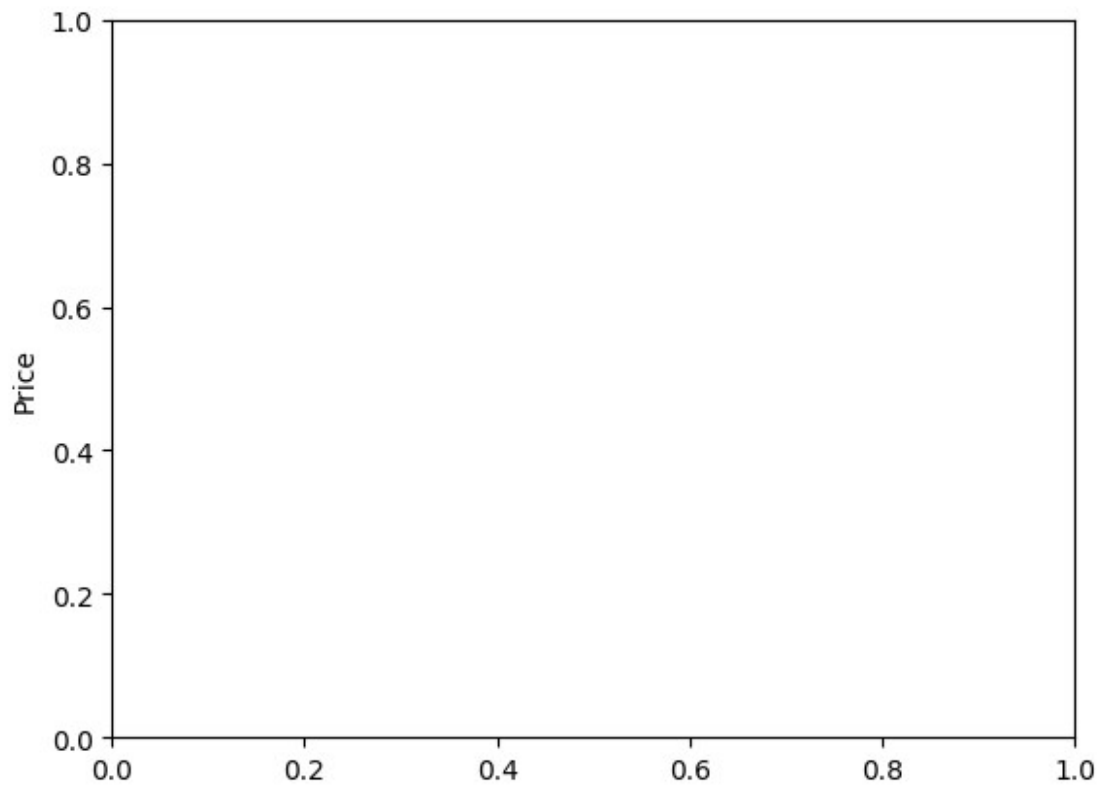
```
plt.show()
# Boxplot of Price
plt.figure(figsize=(4,6))
<Figure size 400x600 with 0 Axes>
<Figure size 400x600 with 0 Axes>
sns.boxplot(y=df['price'])
<Axes: ylabel='price'>
```



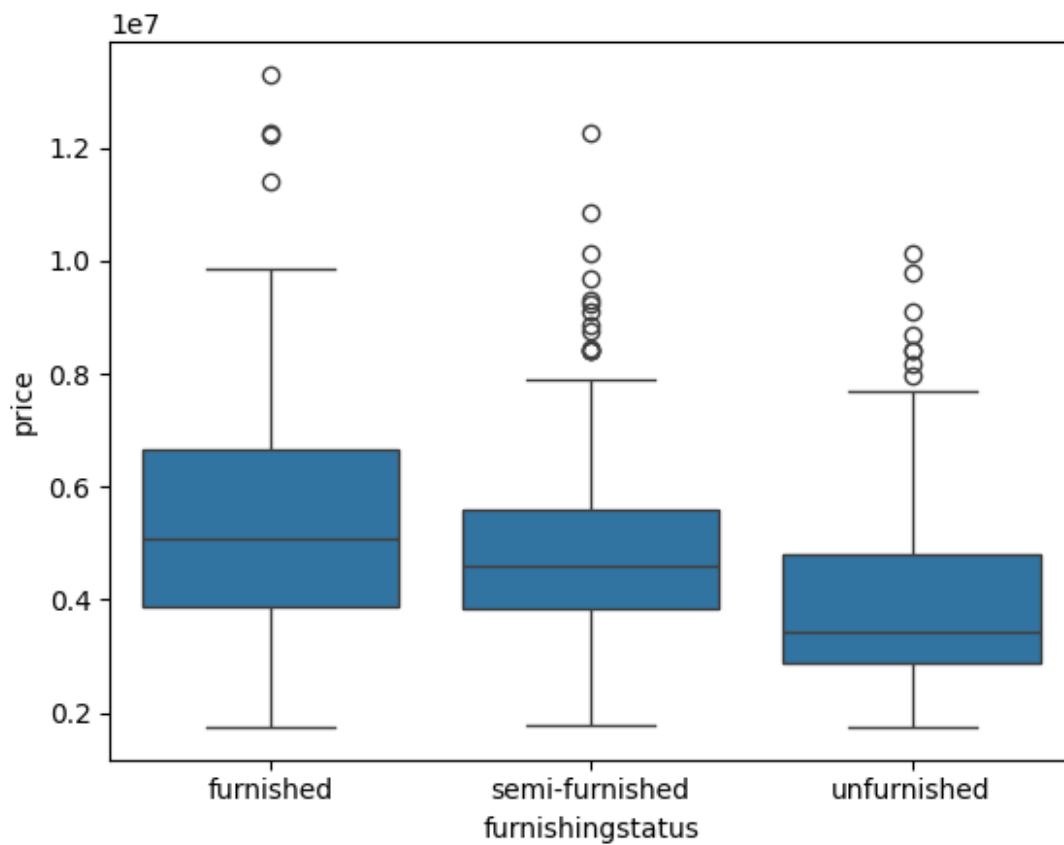
```
plt.title('House Prices Boxplot')  
Text(0.5, 1.0, 'House Prices Boxplot')
```



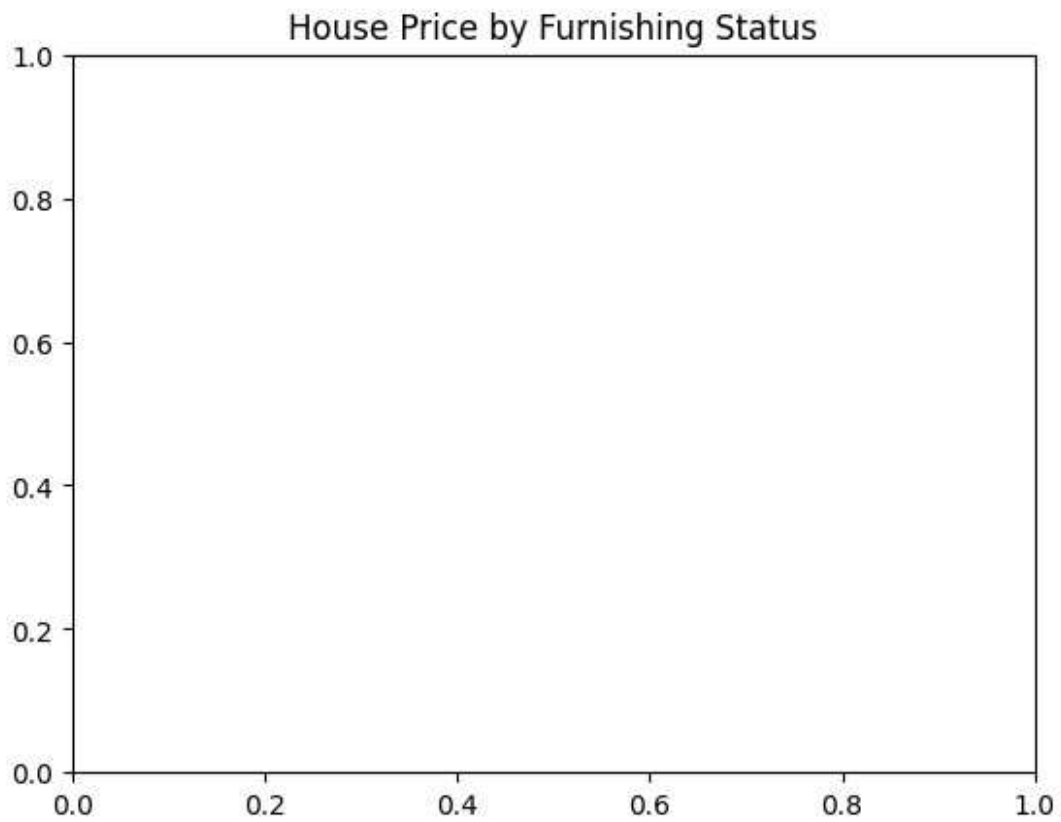
```
plt.ylabel('Price')  
Text(0, 0.5, 'Price')
```



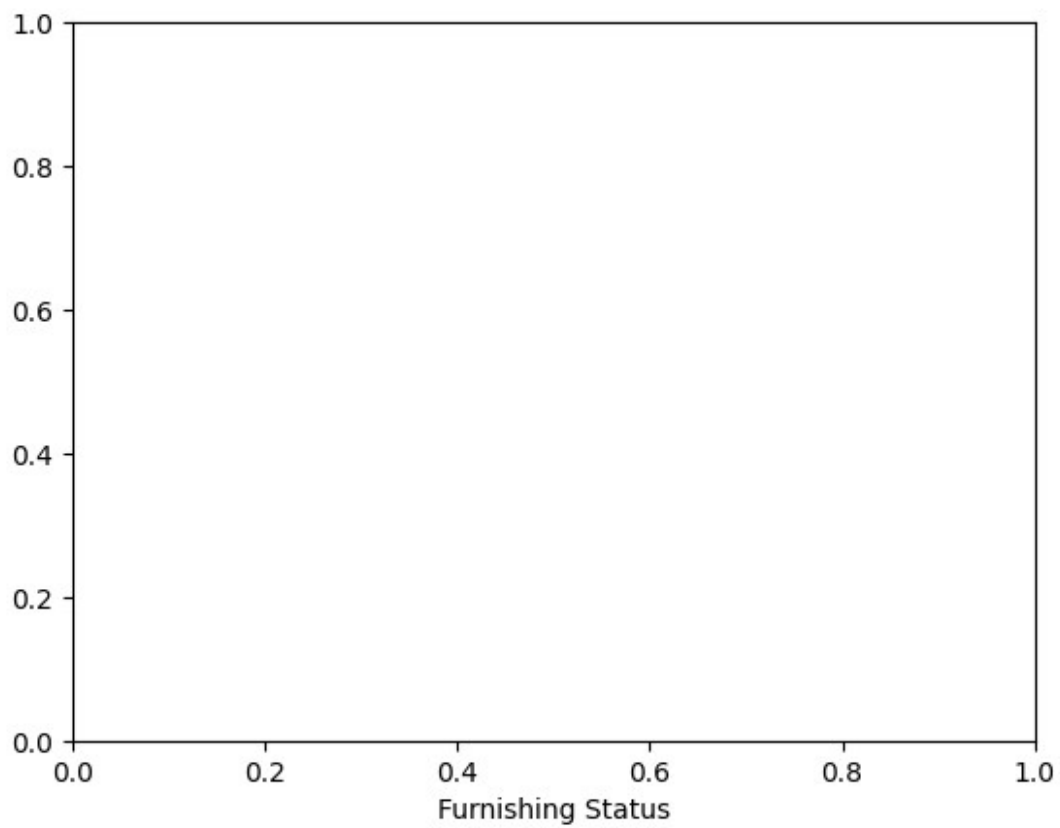
```
plt.show()
# Boxplot of Price by furnishing status
plt.figure(figsize=(6,4))
<Figure size 600x400 with 0 Axes>
<Figure size 600x400 with 0 Axes>
sns.boxplot(x=df['furnishingstatus'], y=df['price'])
<Axes: xlabel='furnishingstatus', ylabel='price'>
```



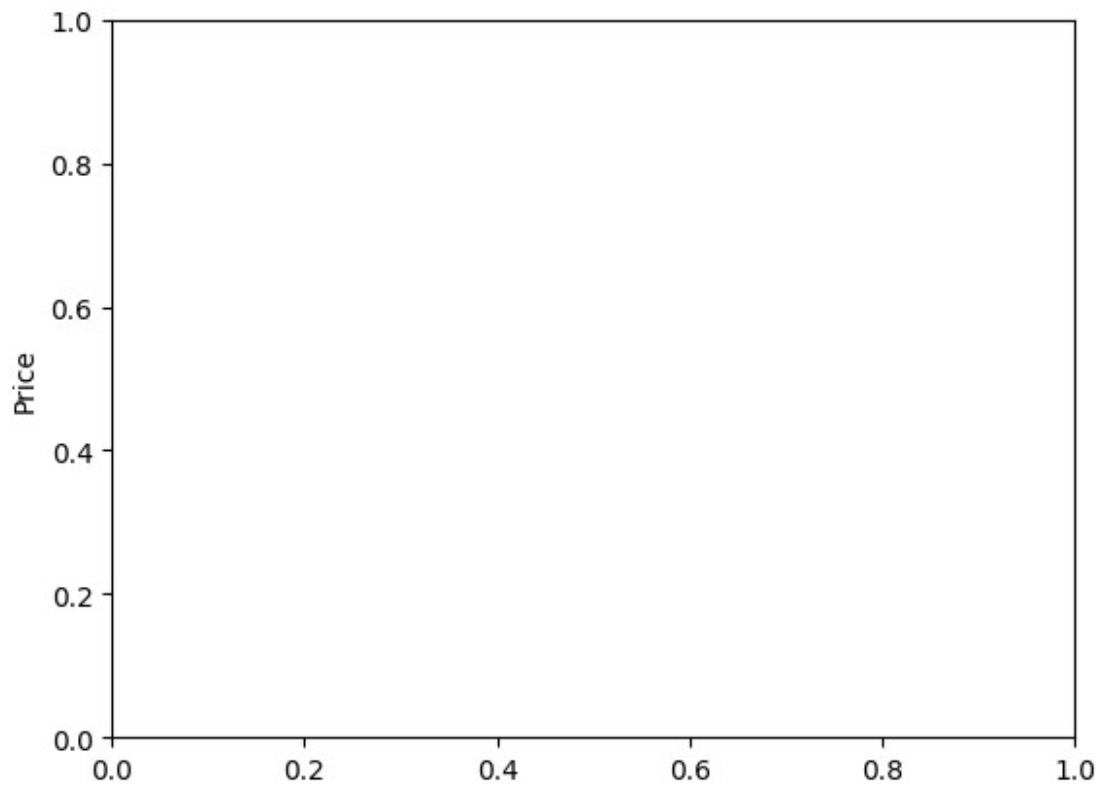
```
plt.title('House Price by Furnishing Status')  
Text(0.5, 1.0, 'House Price by Furnishing Status')
```



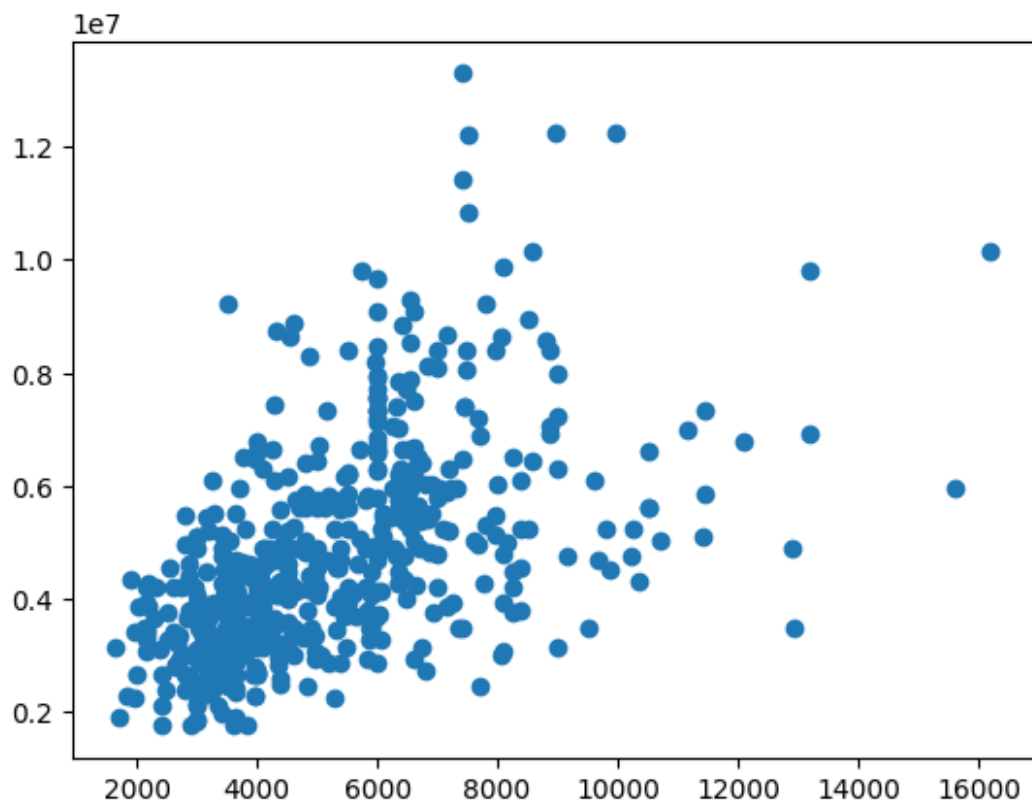
```
plt.xlabel('Furnishing Status')  
Text(0.5, 0, 'Furnishing Status')
```



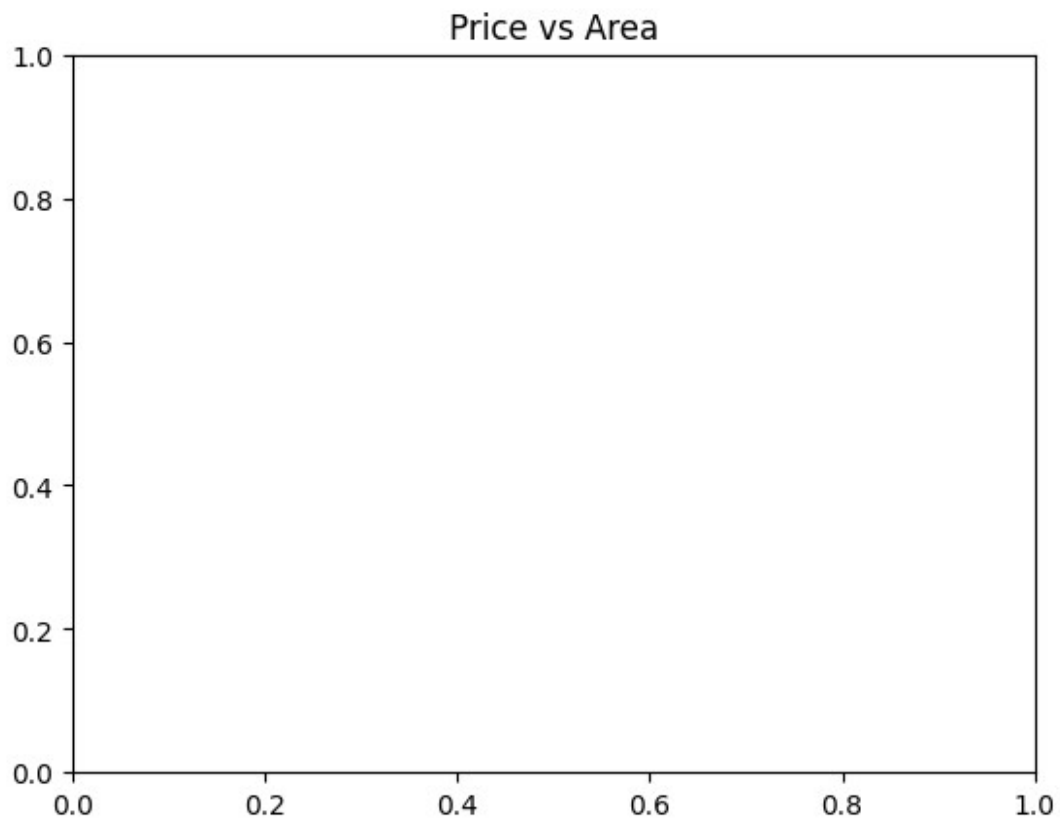
```
plt.ylabel('Price')  
Text(0, 0.5, 'Price')
```



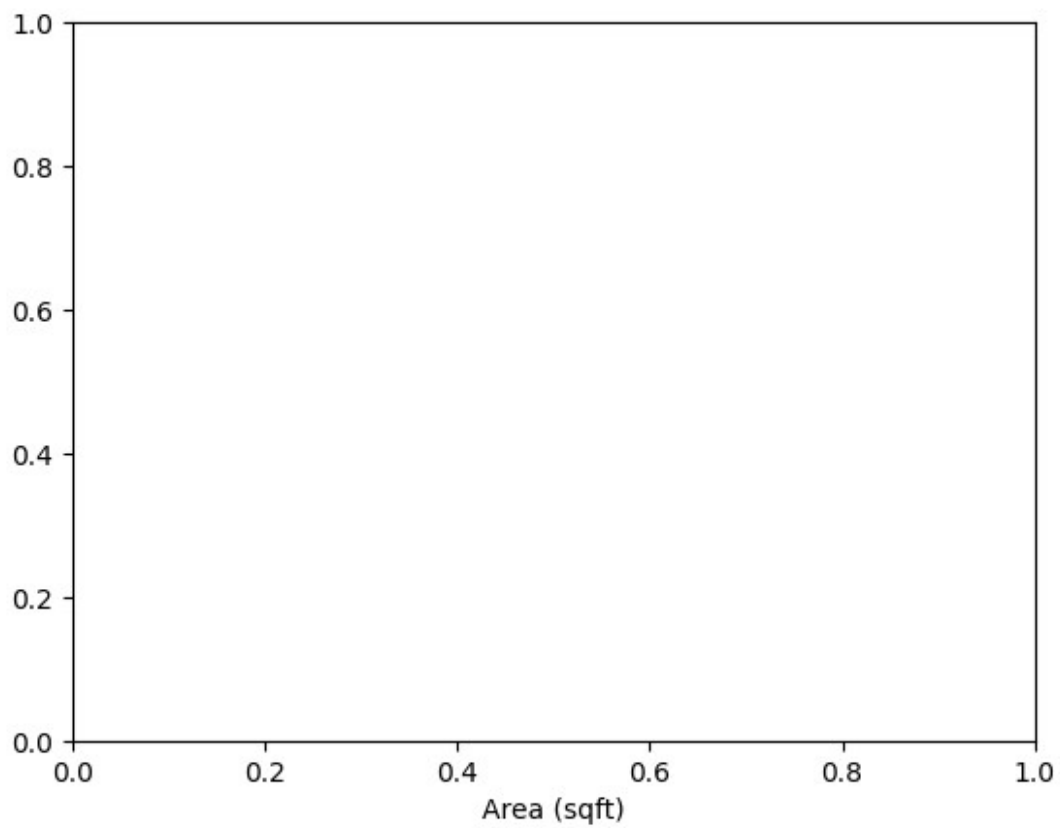
```
plt.show()
# Scatter plot of Price vs Area
plt.figure(figsize=(6,5))
<Figure size 600x500 with 0 Axes>
<Figure size 600x500 with 0 Axes>
plt.scatter(df['area'], df['price'])
<matplotlib.collections.PathCollection at 0x278193854f0>
```



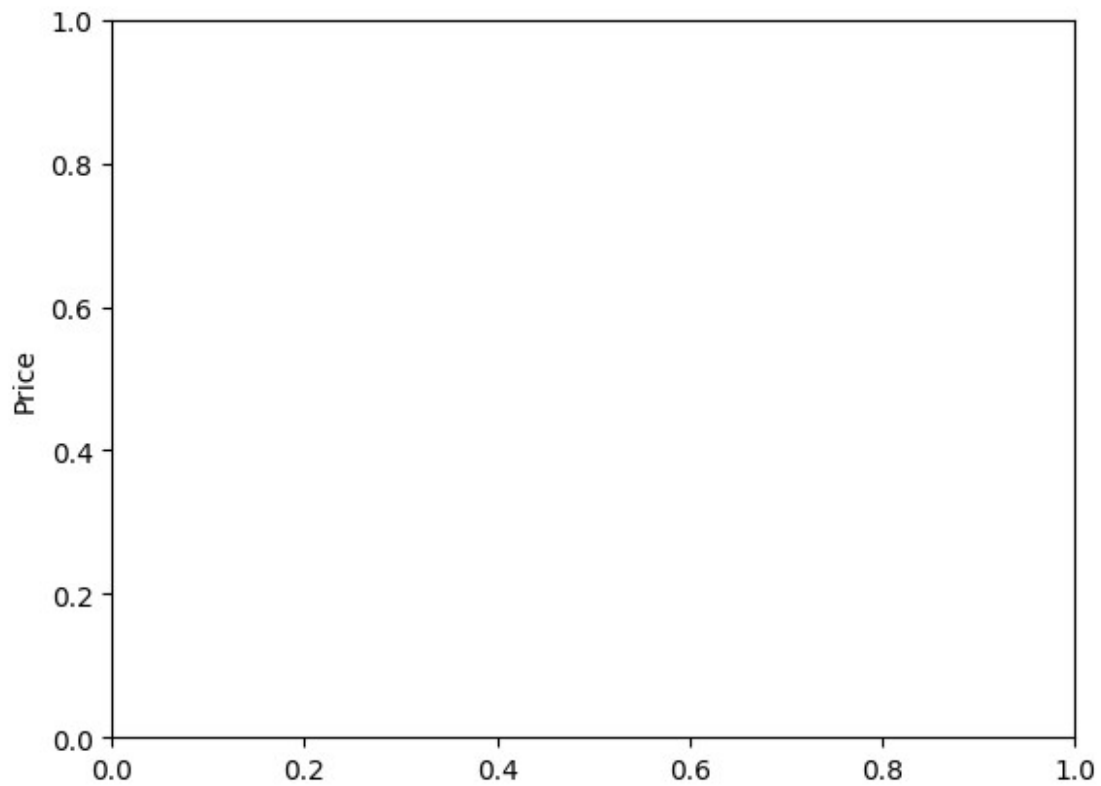
```
plt.title('Price vs Area')  
Text(0.5, 1.0, 'Price vs Area')
```



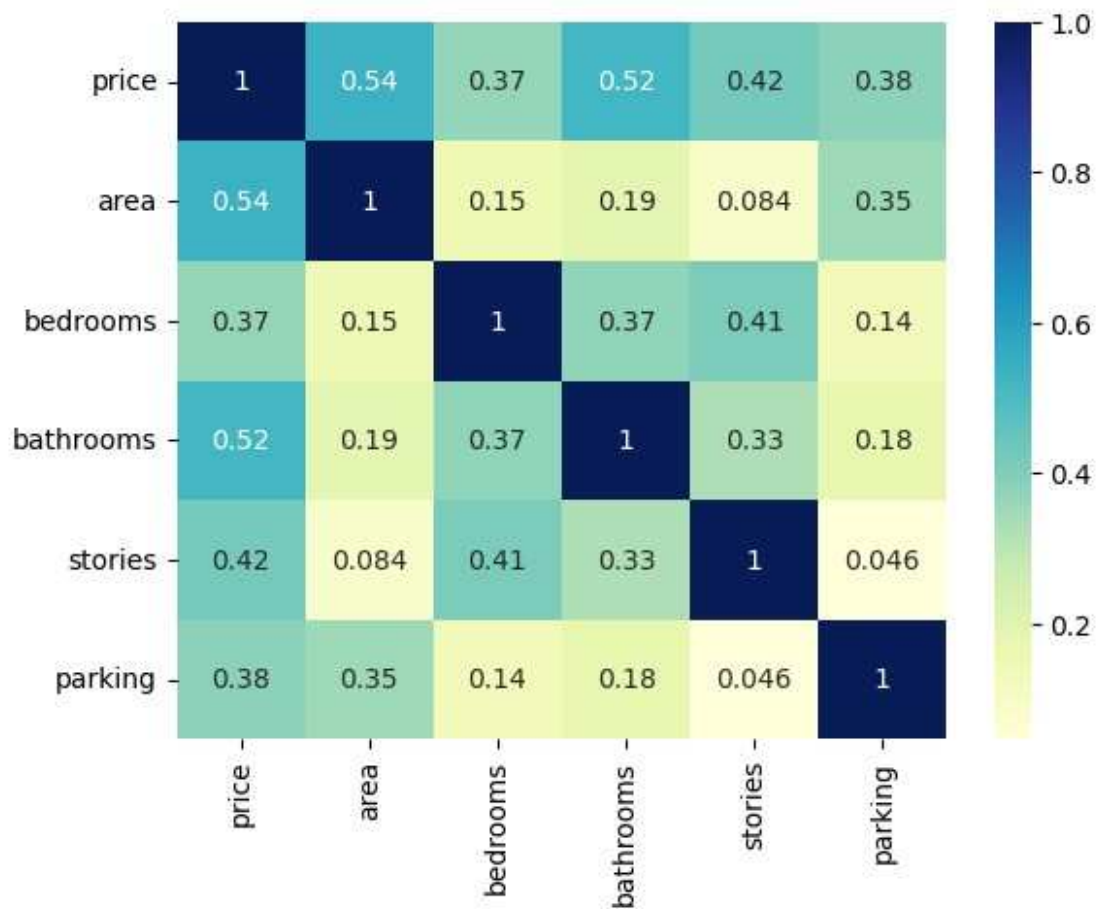
```
plt.xlabel('Area (sqft)')  
Text(0.5, 0, 'Area (sqft)')
```



```
plt.ylabel('Price')  
Text(0, 0.5, 'Price')
```

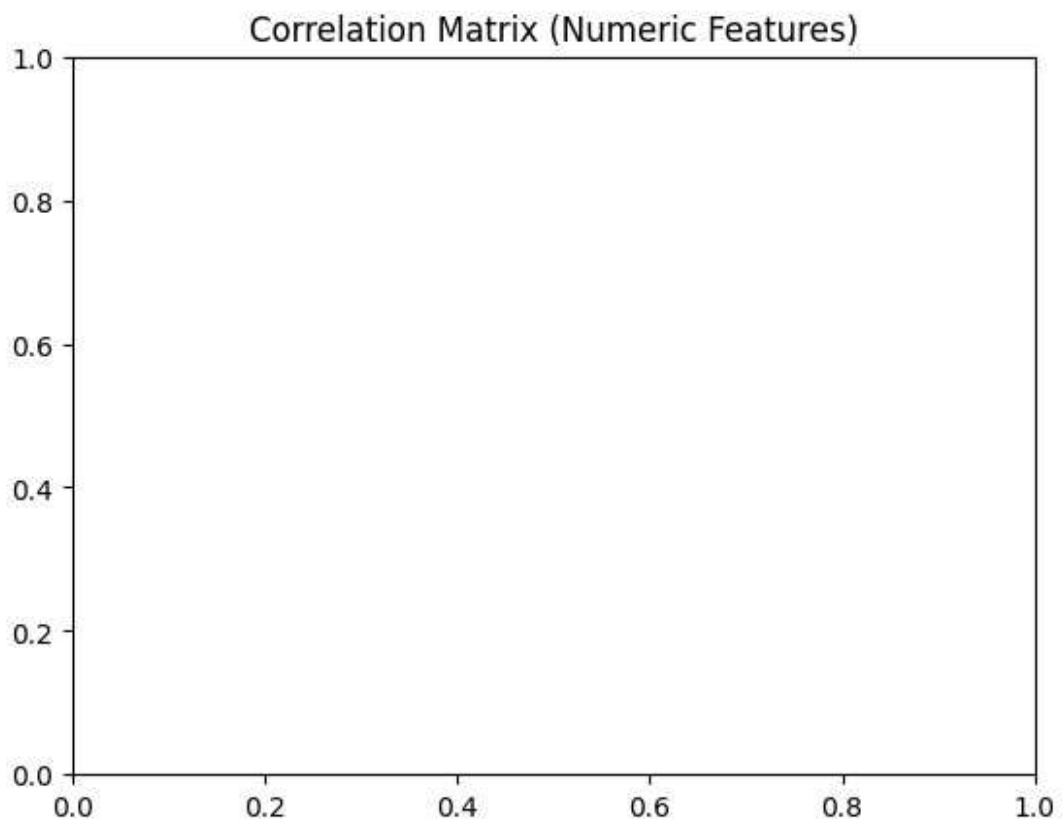


```
plt.show()
# Correlation heatmap for numeric features
plt.figure(figsize=(6,5))
<Figure size 600x500 with 0 Axes>
<Figure size 600x500 with 0 Axes>
corr = df[numeric_cols].corr()
sns.heatmap(corr, annot=True, cmap='YlGnBu')
<Axes: >
```



```
plt.title('Correlation Matrix (Numeric Features)')
```

```
Text(0.5, 1.0, 'Correlation Matrix (Numeric Features)')
```



```
plt.show()
```

```
In [1]: import pandas as pd
In [2]: import numpy as np

In [6]: pwd
Out[6]: 'C:\Users\Administrator'

In [7]: df = pd.read_csv('Housing.csv')

In [14]: df.head()
Out[14]:
   price  area  bedrooms  bathrooms  stories  mainroad  guestroom  basement  hotwaterheating  airconditioning  parking  prefarea  furnishingstatus
0  13300000  7420      4        2        3      yes      no      no      no      yes      2      yes      furnished
1  12250000  8960      4        2        4      yes      no      no      no      yes      3      no      semi-furnished
2  12250000  9980      3        2        2      yes      no      yes      no      no      2      yes      semi-furnished
3  12150000  7900      4        2        2      yes      no      yes      no      yes      3      yes      furnished
4  11410000  7420      4        1        2      yes      yes      yes      no      yes      2      no      unfurnished

In [15]: df.describe()
Out[15]:
           price      area  bedrooms  bathrooms  stories  parking
count  5.400000e+02  545.000000  545.000000  545.000000  545.000000  545.000000
mean    4.766729e+06  5190.541284  2.865136  1.286229  1.805505  0.683578
std     1.870440e+06  2170.141023  0.738864  0.552470  0.867482  0.861588
min    -1.765000e+06  -1650.000000  0.000000  0.000000  0.000000  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  6300.000000  3.000000  2.000000  2.000000  1.000000
max    1.330000e+07  16200.000000  6.000000  4.000000  4.000000  3.000000

In [30]: # Handle Missing Values

In [18]: missing_counts = df.isnull().sum()

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

In [20]: len(df)
Out[20]: 545

In [31]: # There is no need to create a cleaned dataset (CSV) if there are no missing values.

In [ ]: # Detect Outliers using IQR

In [22]: numeric_cols = ['price', 'area', 'bedrooms', 'bathrooms', 'stories', 'parking']

In [23]: Q1 = df[numeric_cols].quantile(0.25)

In [24]: Q3 = df[numeric_cols].quantile(0.75)

In [25]: IQR = Q3 - Q1

In [37]: lower_bound = Q1 - 1.5 * IQR

In [38]: upper_bound = Q3 + 1.5 * IQR

In [39]: outlier_mask = (df[numeric_cols] < (Q1 - 1.5 * IQR)) | (df[numeric_cols] > (Q3 + 1.5 * IQR))

In [40]: outliers = outlier_mask.any(axis=1)

In [44]: print(f'Total outlier rows detected (IQR method): {outliers.sum()}')
Total outlier rows detected (IQR method): 82

In [42]: #Detect Outliers using Z-score

In [45]: from scipy import stats

In [46]: z_scores = np.abs(stats.zscore(df[numeric_cols]))

In [47]: z_outliers = (z_scores > 3).any(axis=1)

In [48]: print(f'Total outlier rows detected (Z-score method): {z_outliers.sum()}')
Total outlier rows detected (Z-score method): 23

In [56]: df['price'] = pd.to_numeric(df['price'], errors='coerce')

In [63]: df = df.dropna(subset=['price'])

In [64]: df['price'] = df['price'].astype(float)

In [65]: cap_price = np.percentile(df['price'], 99)

In [66]: cap_area = np.percentile(df['area'], 99)

In [67]: clean_df = df.copy()

In [74]: # Handle Outliers by Capping

In [75]: clean_df.loc[clean_df['price'] > cap_price, 'price'] = cap_price

In [76]: clean_df.loc[clean_df['area'] > cap_area, 'area'] = cap_area

In [77]: print(f'Old max price vs new max price:', df['price'].max(), '->', clean_df['price'].max())
Old max price vs new max price: 13300800.0 -> 10541999.99999999963

In [78]: print(f'Old max area vs new max area:', df['area'].max(), '->', clean_df['area'].max())
Old max area vs new max area: 16288 -> 12543.5999999999955

In [80]: clean_df.to_csv('Housing_cleaned.csv', index=False)

In [90]: #####

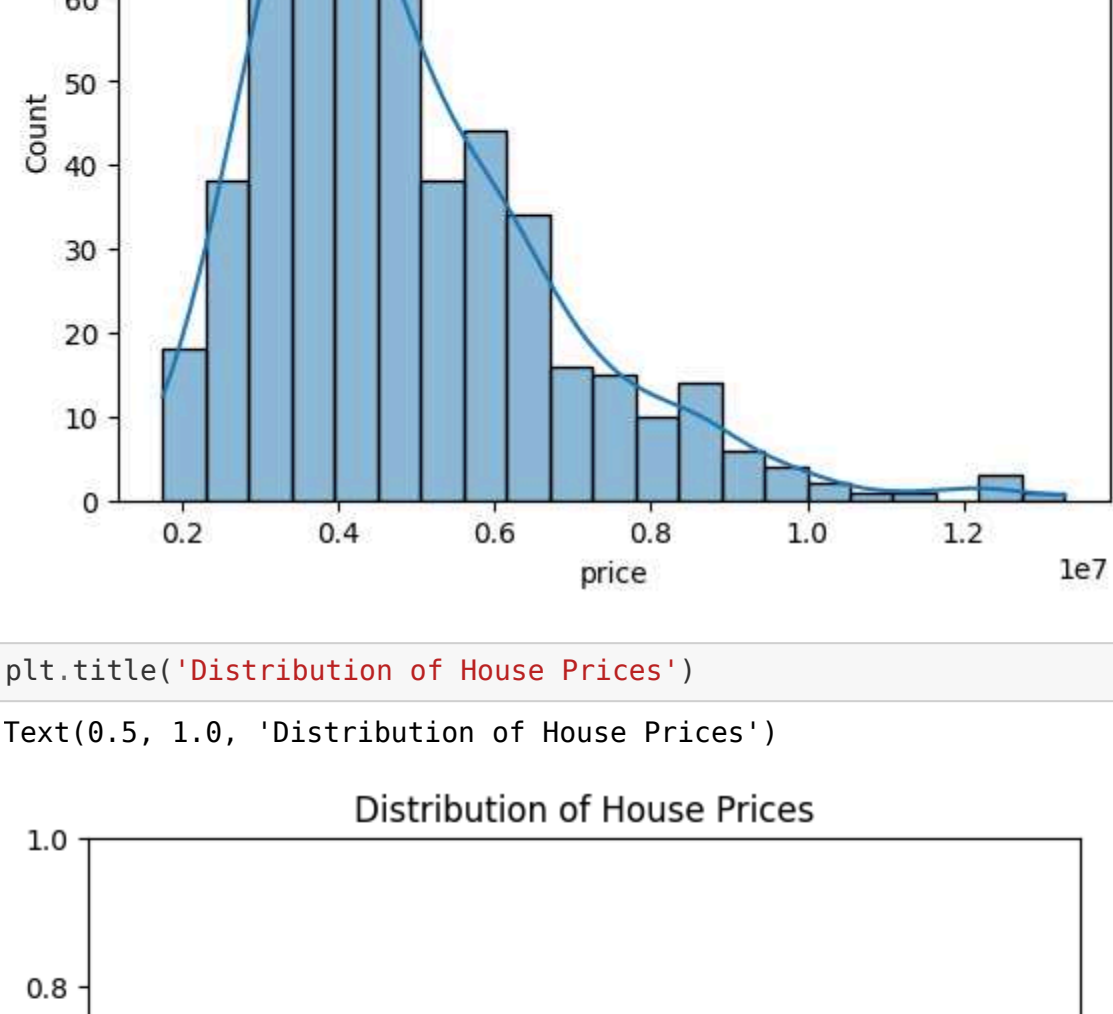
In [99]: import matplotlib.pyplot as plt

In [100]: import seaborn as sns

In [101]: # Data Visualization

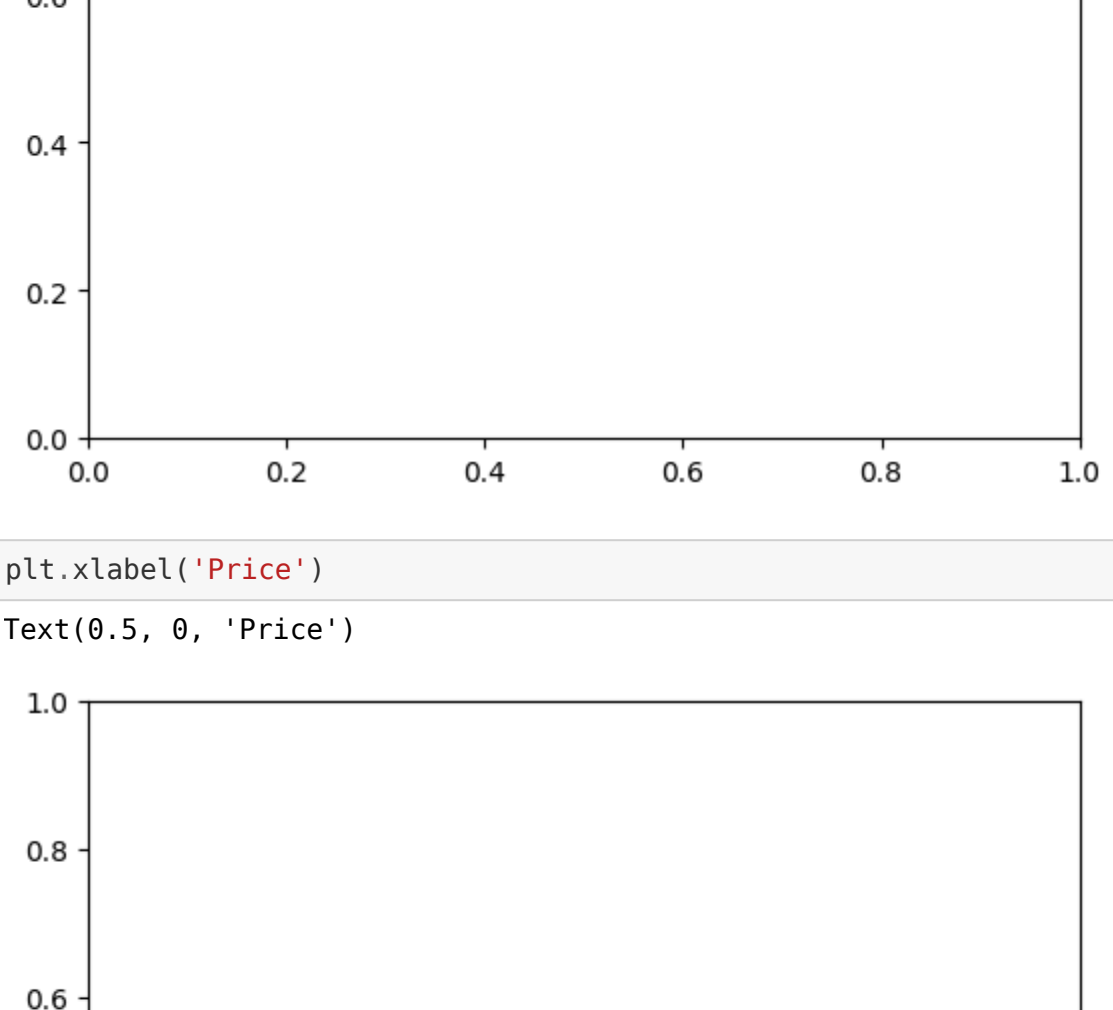
In [102]: # Histogram of Price (original data)

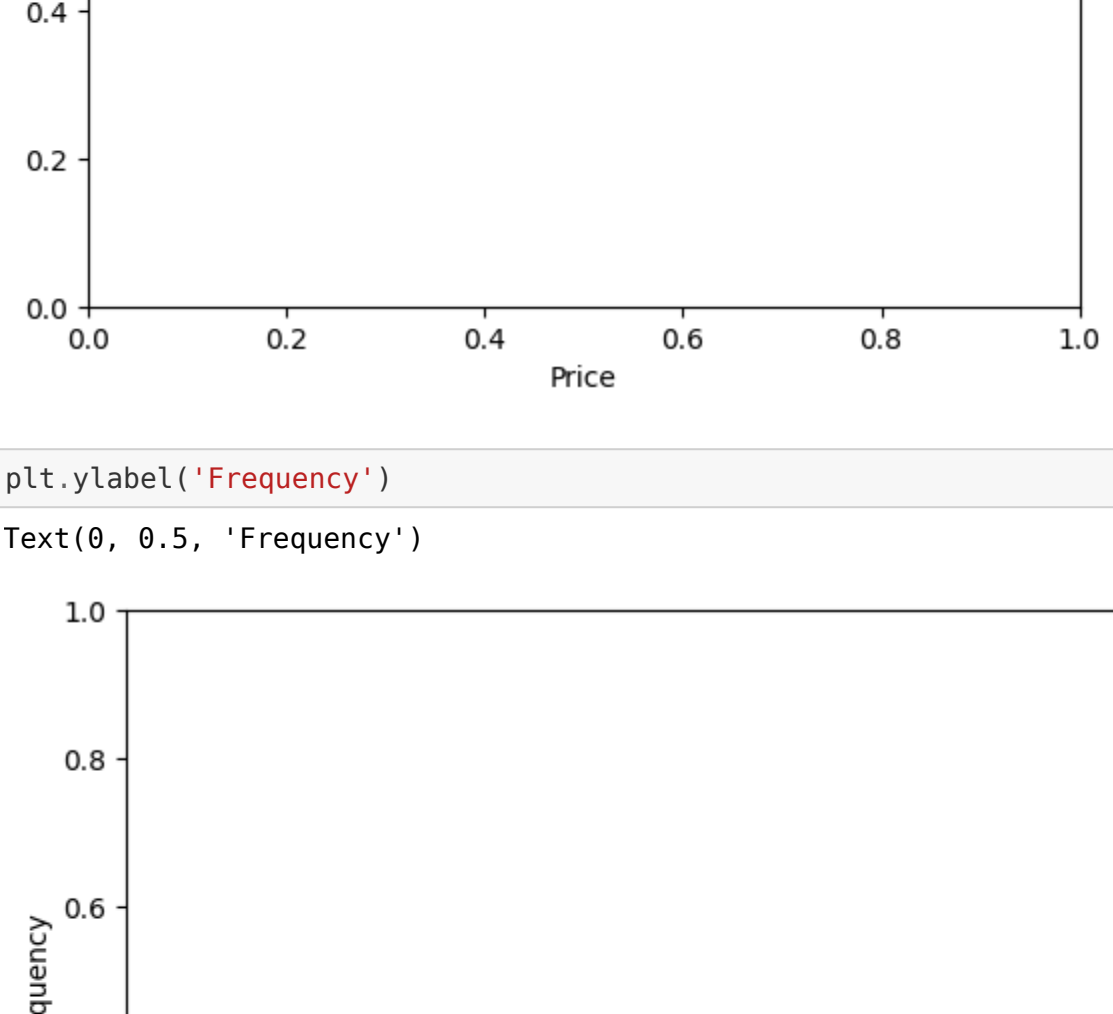
In [103]: plt.figure(figsize=(6,4))
Out[103]: <Figure size 600x400 with 0 Axes>
<Figure size 600x400 with 0 Axes>

In [104]: sns.histplot(df['price'], kde=True)
Out[104]: <Axes: xlabel='price', ylabel='Count'>


In [105]: plt.title('Distribution of House Prices')

In [106]: Text(0.5, 1.0, 'Distribution of House Prices')

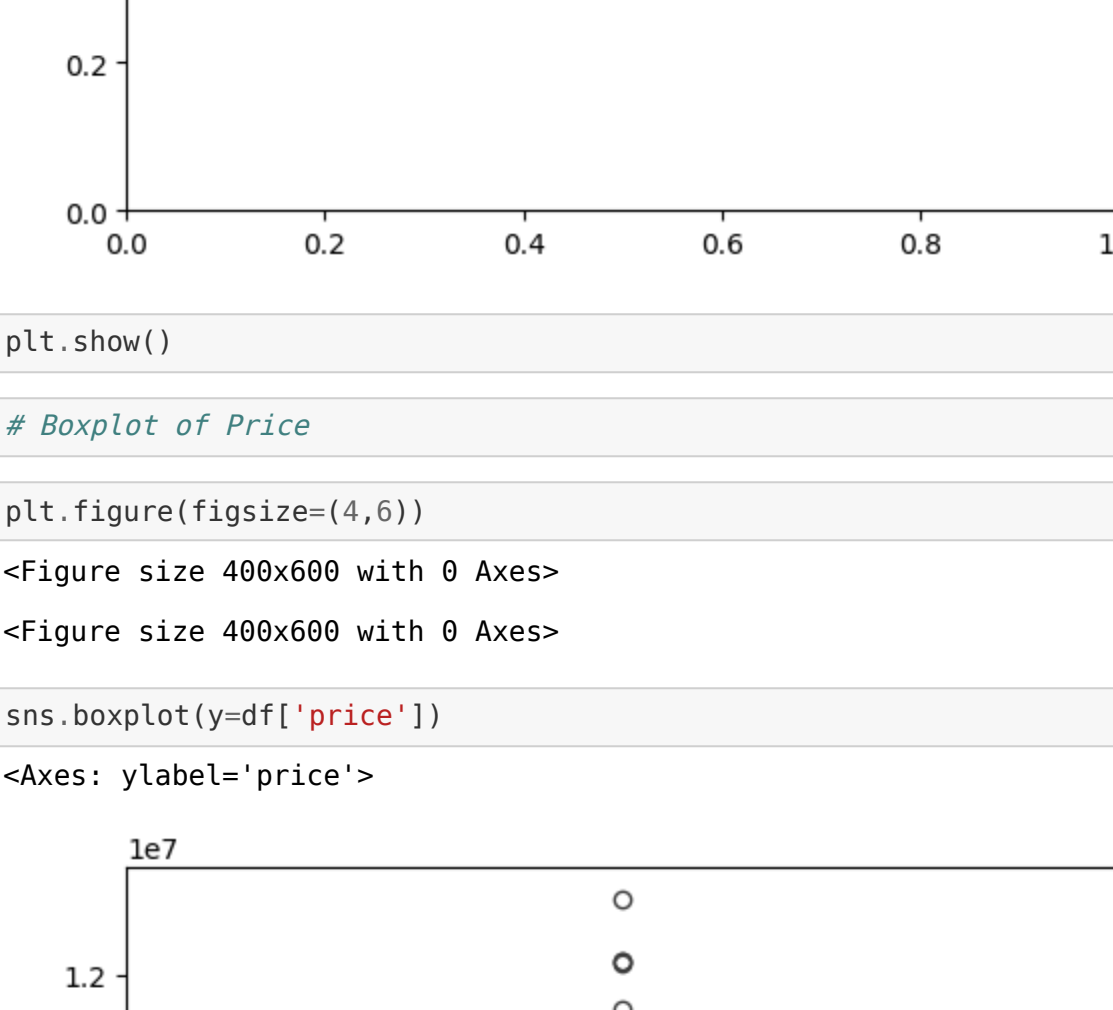
In [106]: plt.xlabel('Price')
Out[106]: Text(0.5, 0, 'Price')


In [106]: plt.ylabel('Frequency')
Out[107]: Text(0, 0.5, 'Frequency')


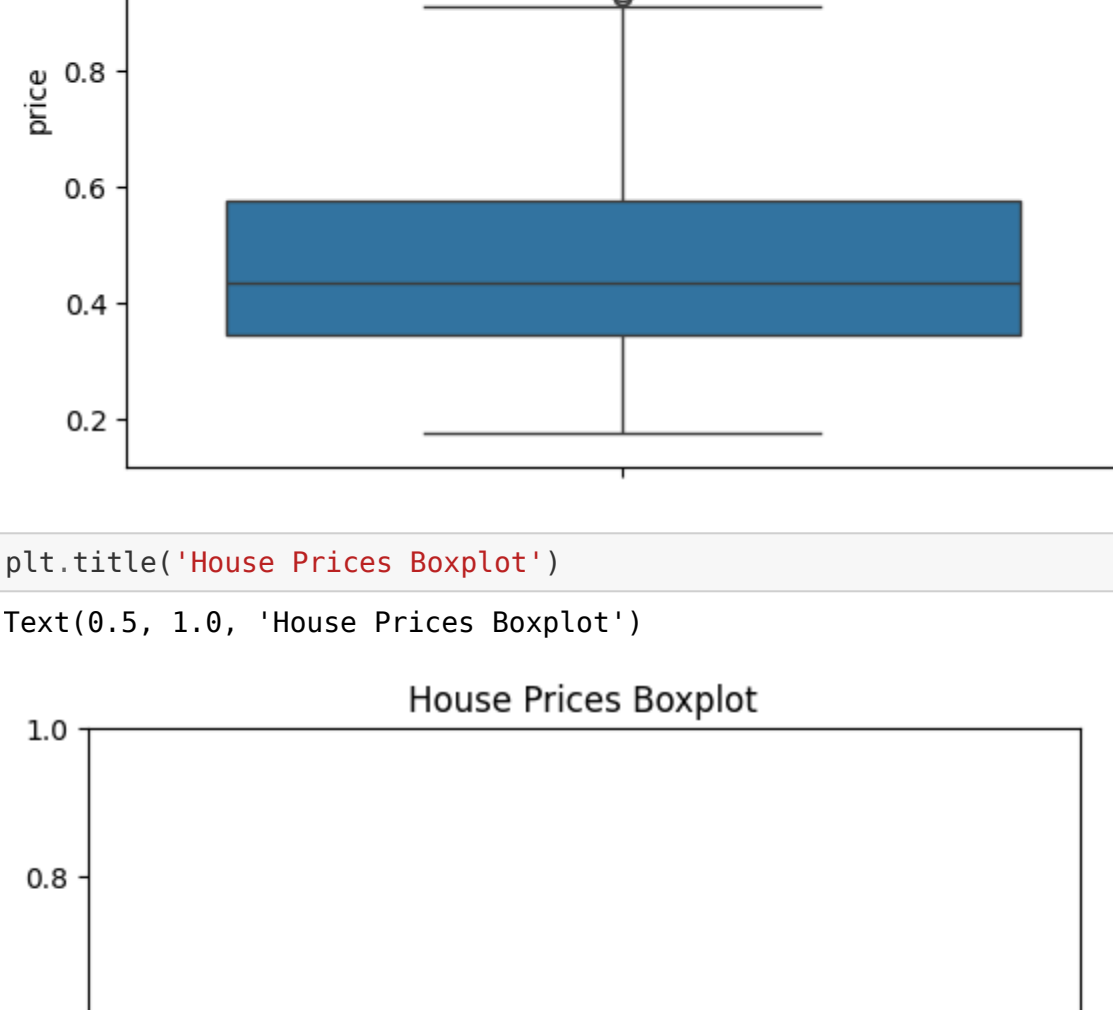
In [108]: plt.show()

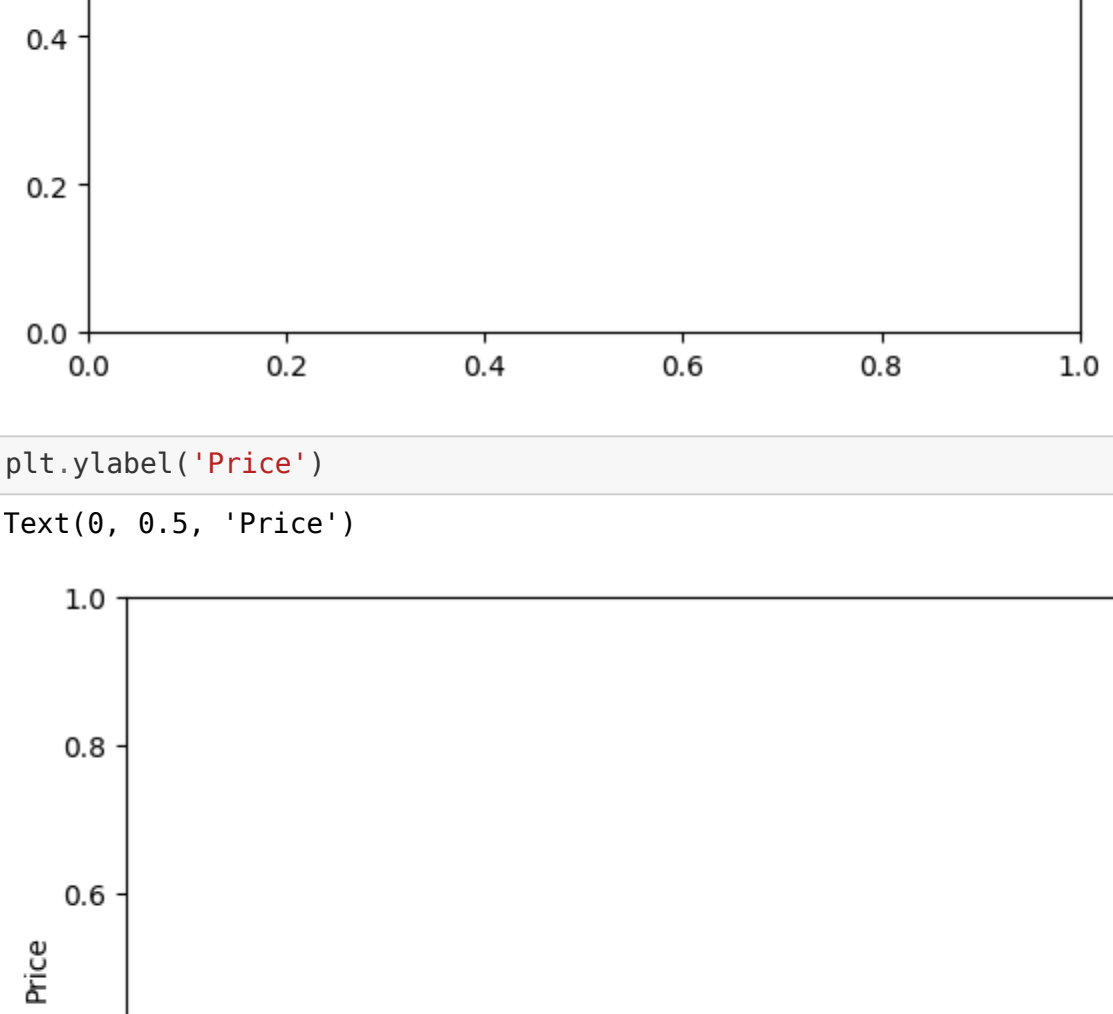
In [109]: # Boxplot of Price

In [110]: plt.figure(figsize=(4,6))
Out[110]: <Figure size 400x600 with 0 Axes>
<Figure size 400x600 with 0 Axes>

In [111]: sns.boxplot(y=df['price'])
Out[111]: <Axes: xlabel='price'>


In [112]: plt.title('House Prices Boxplot')

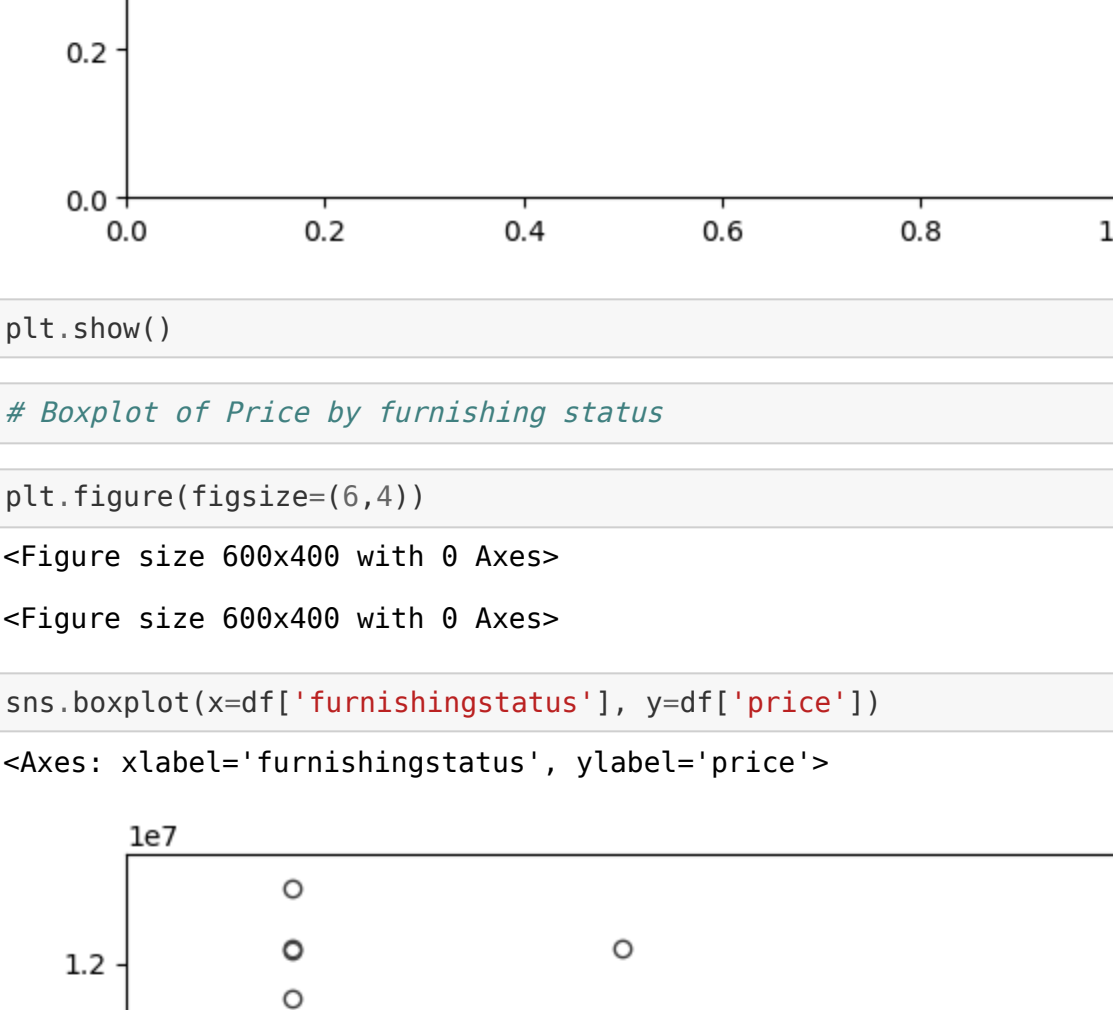
In [113]: Text(0.5, 1.0, 'House Prices Boxplot')


In [113]: plt.ylabel('Price')
Out[113]: Text(0, 0.5, 'Price')


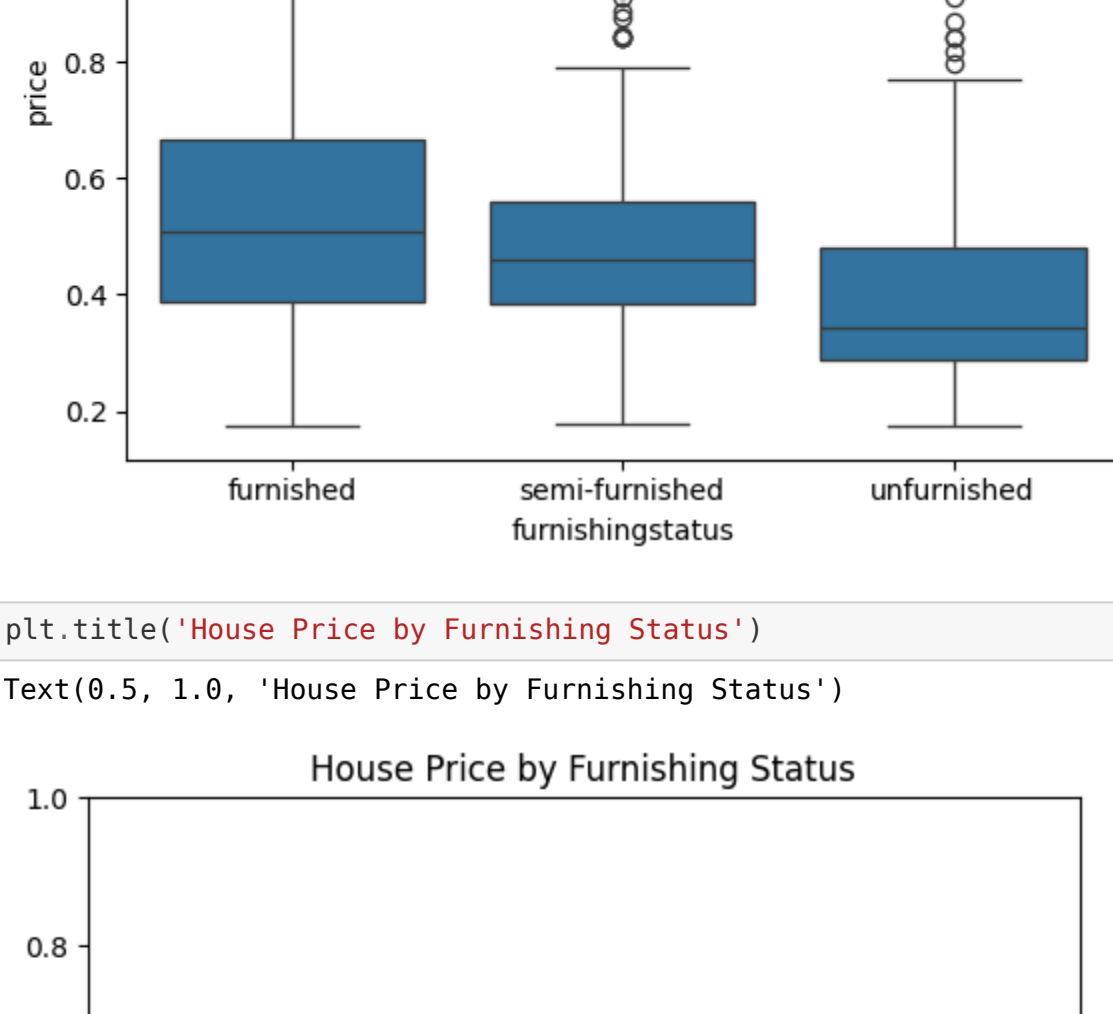
In [114]: plt.show()

In [115]: # Boxplot of Price by furnishing status

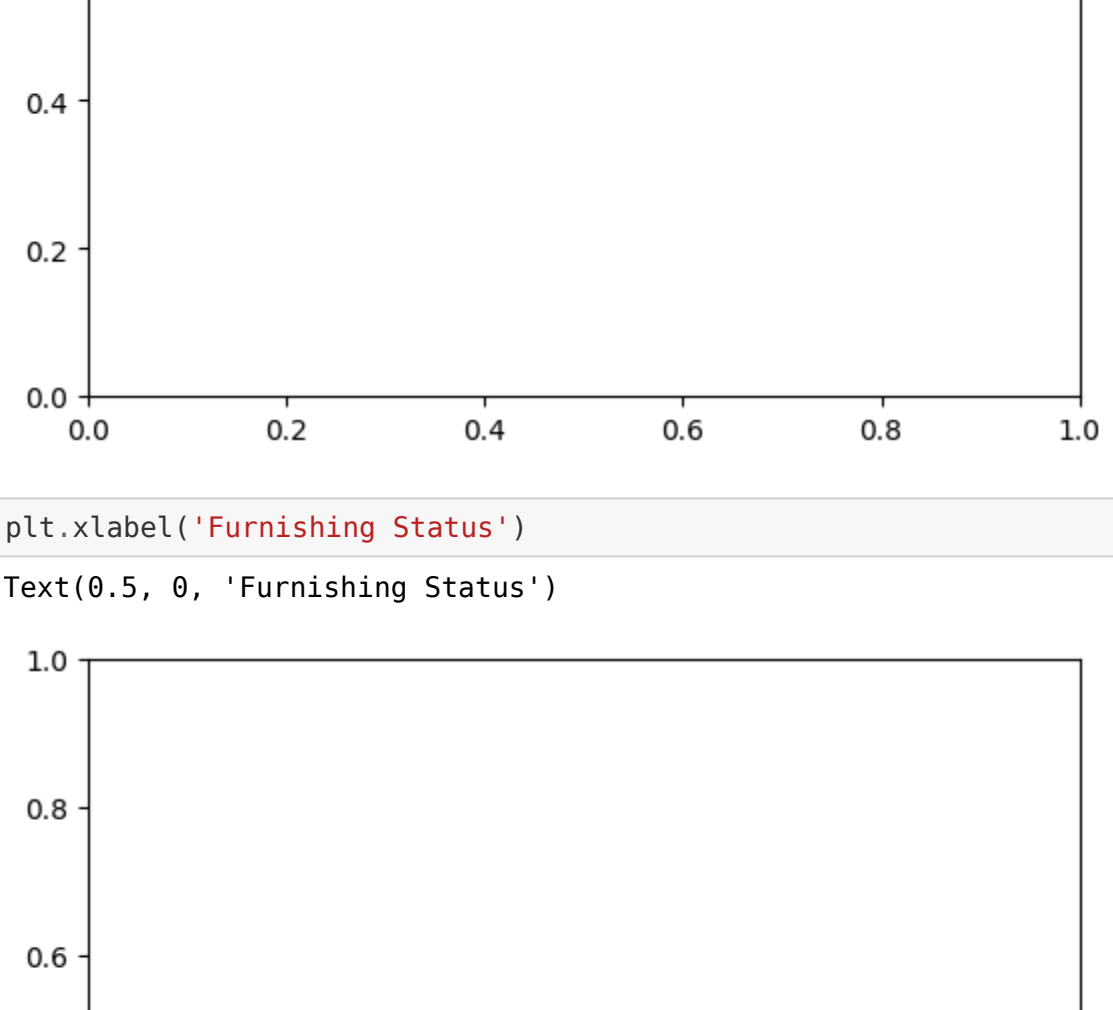
In [116]: plt.figure(figsize=(6,4))
Out[116]: <Figure size 600x400 with 0 Axes>
<Figure size 600x400 with 0 Axes>

In [117]: sns.boxplot(x=df['furnishingstatus'], y=df['price'])
Out[117]: <Axes: xlabel='furnishingstatus', ylabel='price'>


In [118]: plt.title('House Price by Furnishing Status')

In [119]: Text(0.5, 1.0, 'House Price by Furnishing Status')


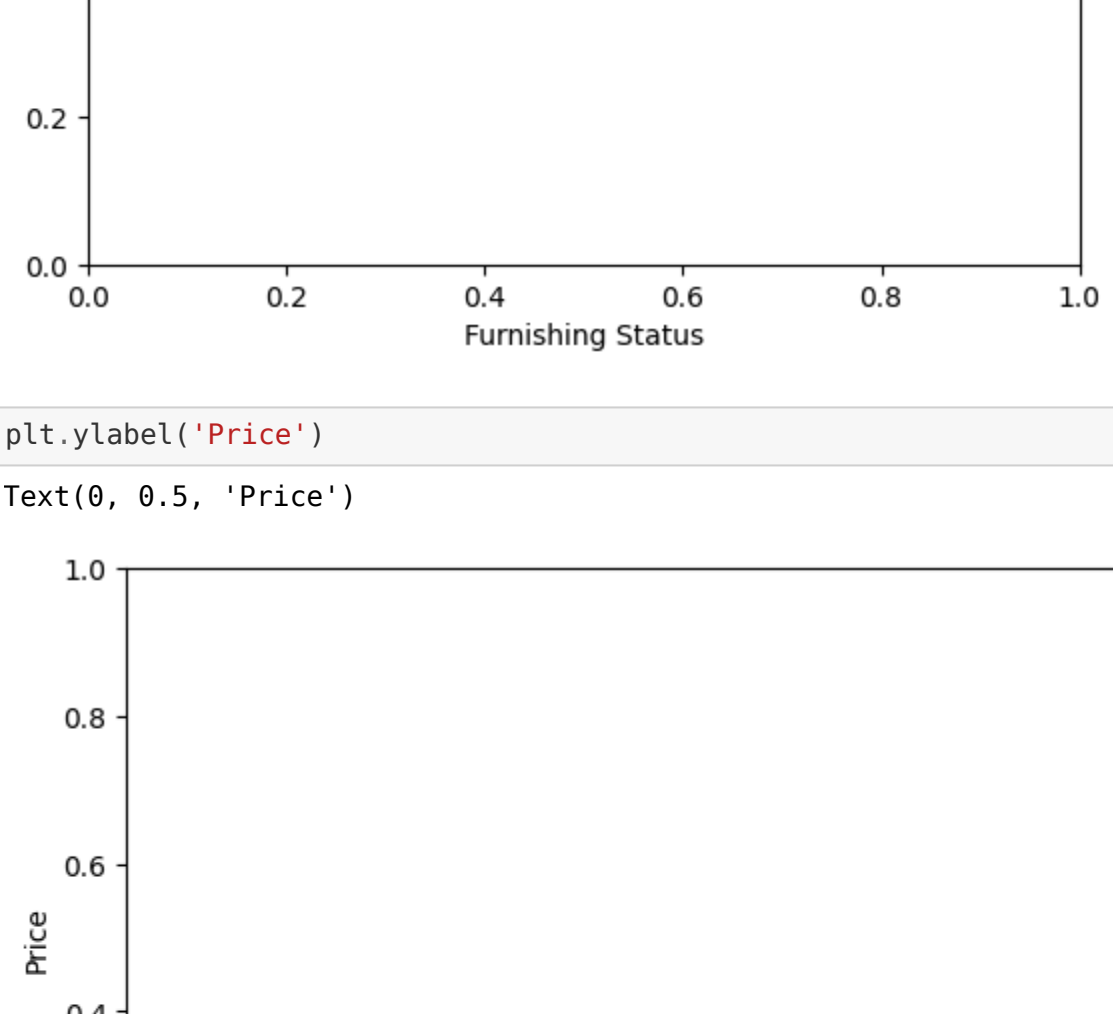
In [119]: plt.xlabel('Furnishing Status')

In [120]: Text(0, 0.5, 'Price')


In [121]: plt.show()

In [122]: # Scatter plot of Price vs Area

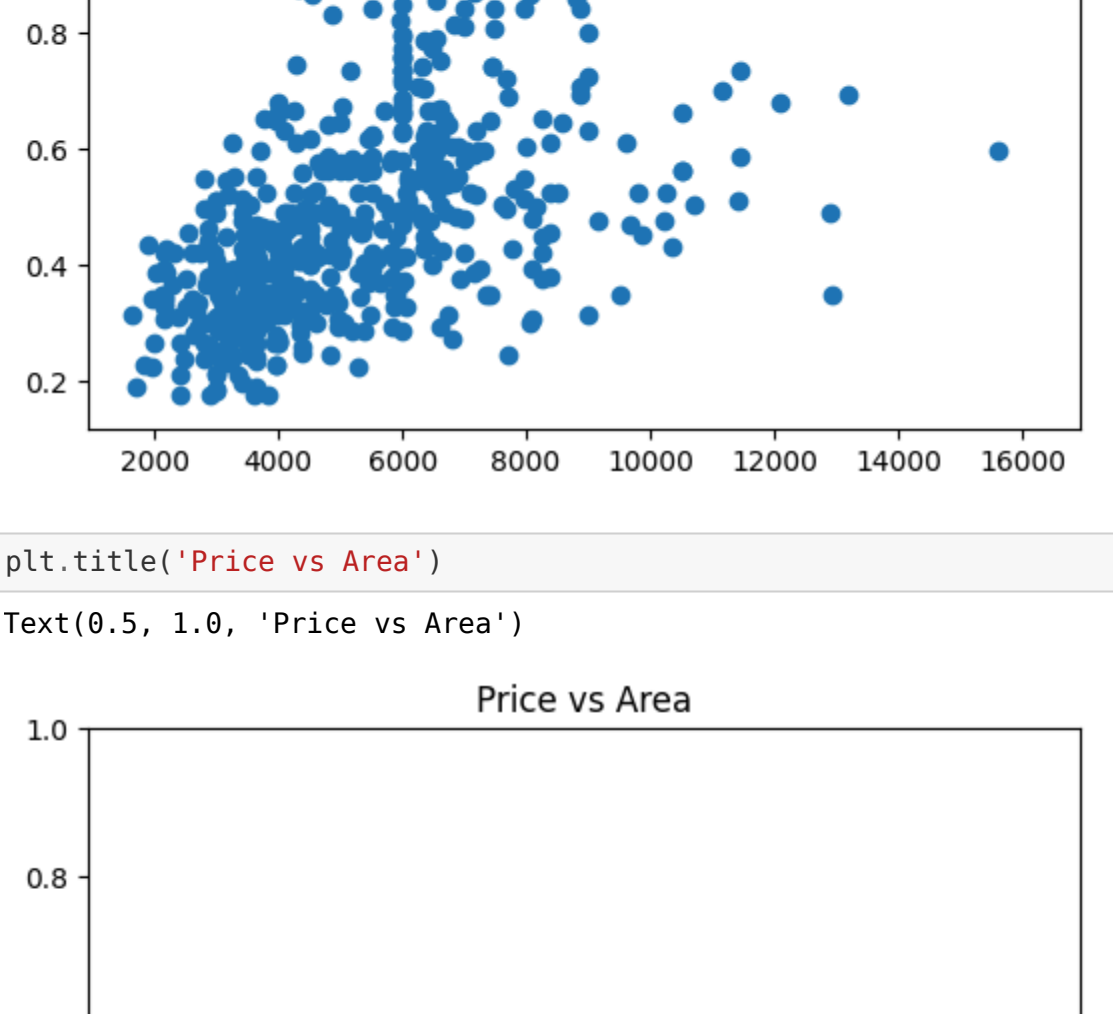
In [123]: plt.figure(figsize=(6,5))
Out[123]: <Figure size 600x500 with 0 Axes>
<Figure size 600x500 with 0 Axes>

In [124]: plt.scatter(df['area'], df['price'])
Out[124]: <matplotlib.collections.PathCollection at 0x278193854f0>


In [125]: plt.title('Price vs Area')

In [126]: Text(0.5, 1.0, 'Price vs Area')


In [126]: plt.xlabel('Area (sqft)')

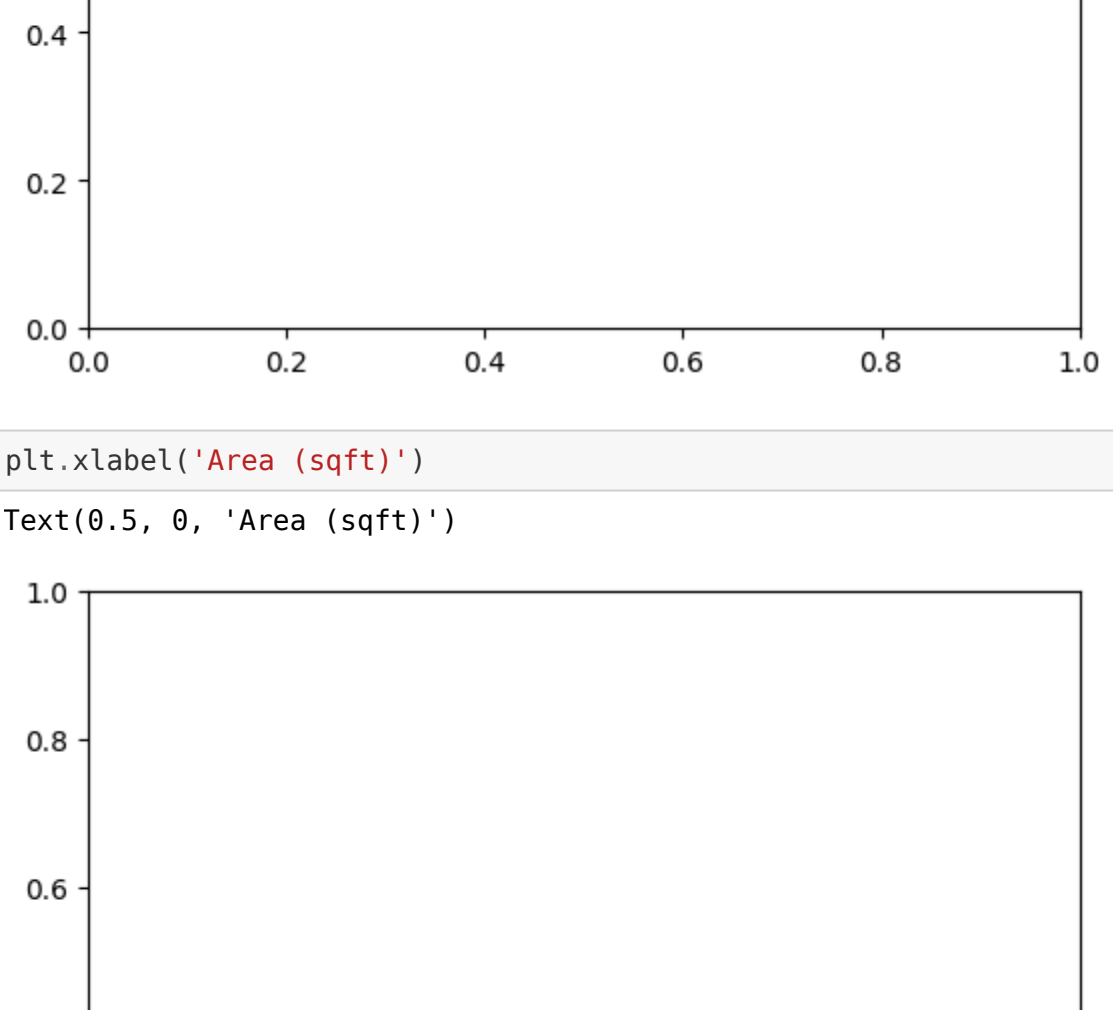
In [127]: plt.ylabel('Price')
Out[127]: Text(0, 0.5, 'Price')


In [128]: plt.show()

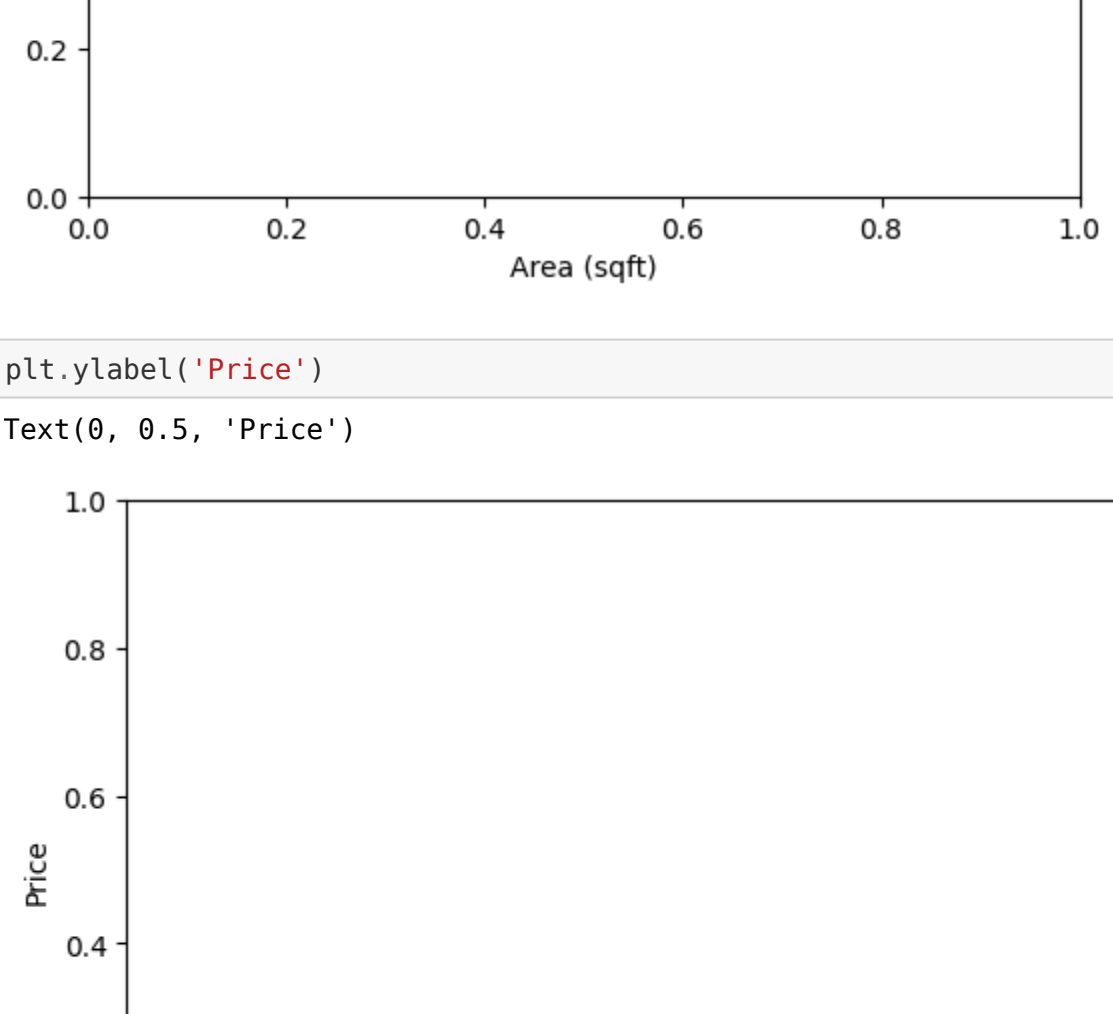
In [129]: # Correlation heatmap for numeric features

In [130]: plt.figure(figsize=(6,5))
Out[130]: <Figure size 600x500 with 0 Axes>
<Figure size 600x500 with 0 Axes>

In [131]: corr = df[numeric_cols].corr()

In [132]: sns.heatmap(corr, annot=True, cmap='YlGnBu')
Out[132]: <Axes: >


In [133]: plt.title('Correlation Matrix (Numeric Features)')

In [133]: Text(0.5, 1.0, 'Correlation Matrix (Numeric Features)')


In [134]: plt.show()

In [ ]:

In [ ]:
```