

```
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

```

        parking
count    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())

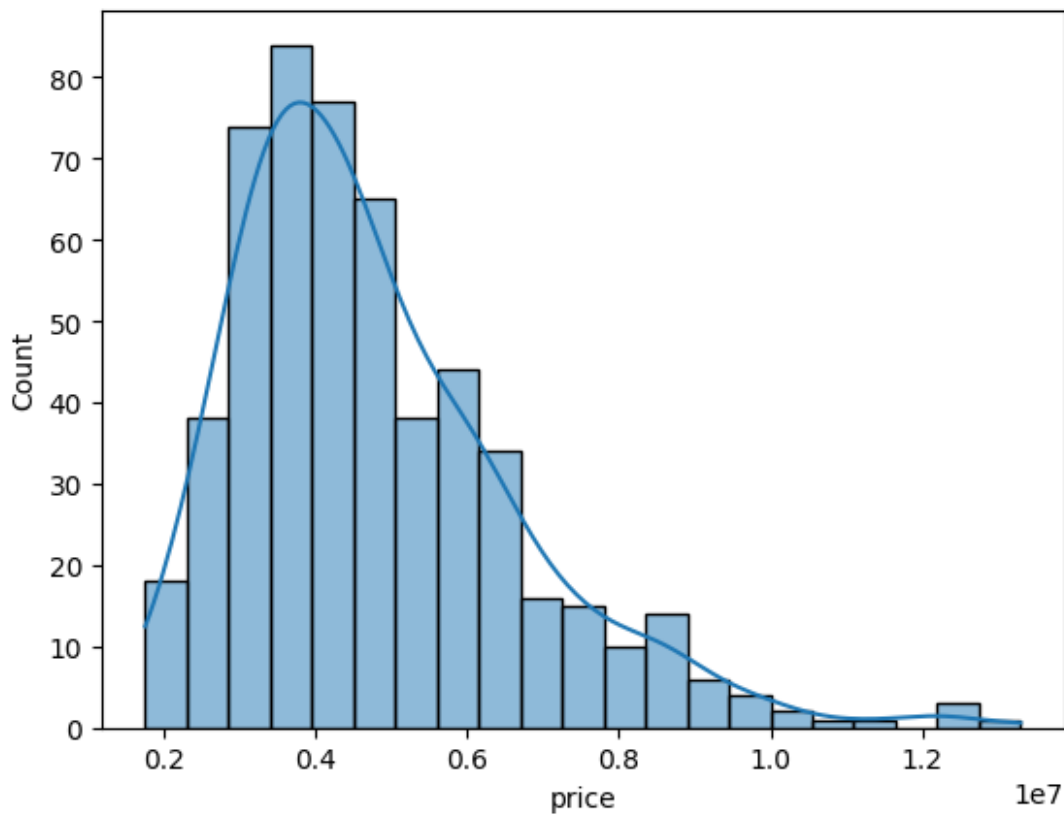
Old max area vs new max area: 16200 -> 12543.599999999955

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

#####

```

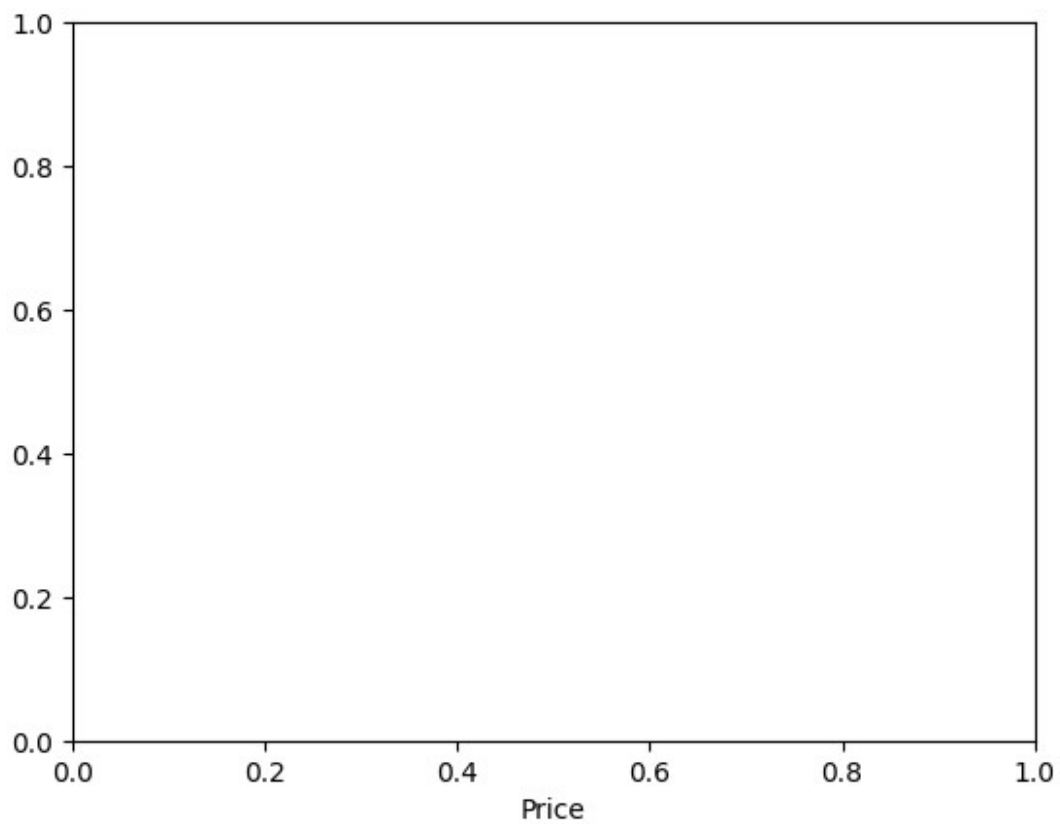
```
import matplotlib.pyplot as plt
import seaborn as sns
# Data Visualization
# Histogram of Price (original data)
plt.figure(figsize=(6,4))
<Figure size 600x400 with 0 Axes>
<Figure size 600x400 with 0 Axes>
sns.histplot(df['price'], kde=True)
<Axes: xlabel='price', ylabel='Count'>
```



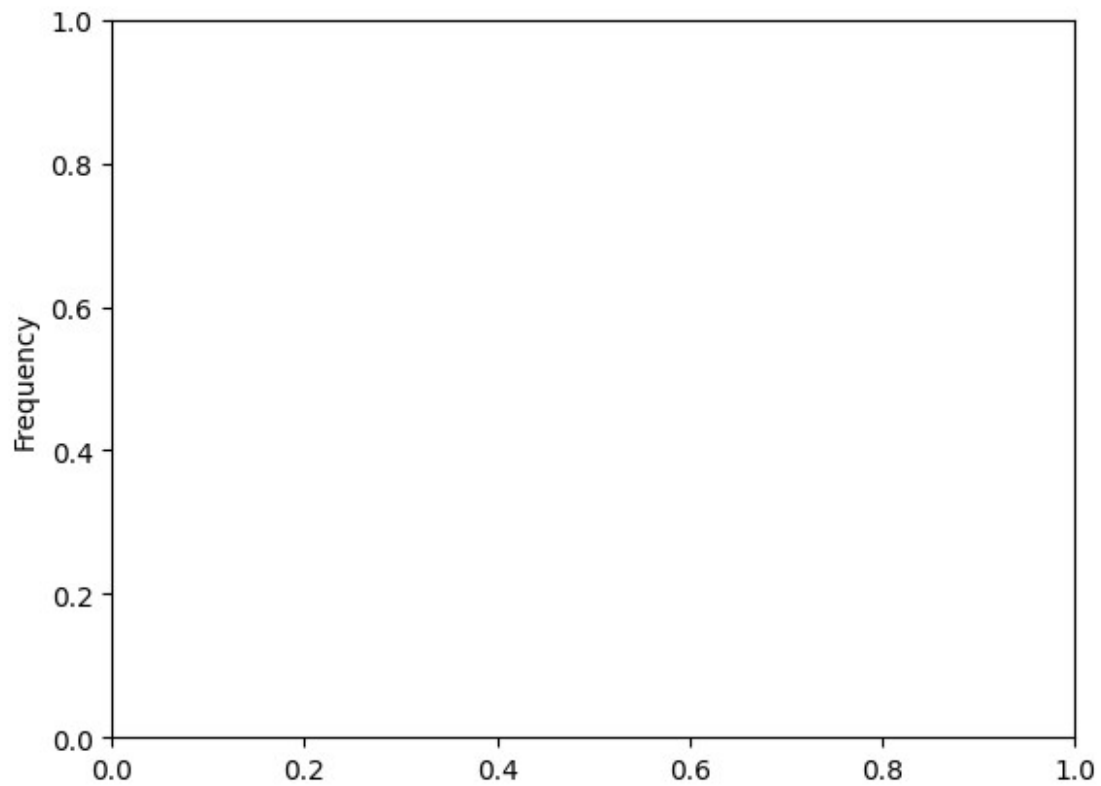
```
plt.title('Distribution of House Prices')
Text(0.5, 1.0, 'Distribution of House Prices')
```



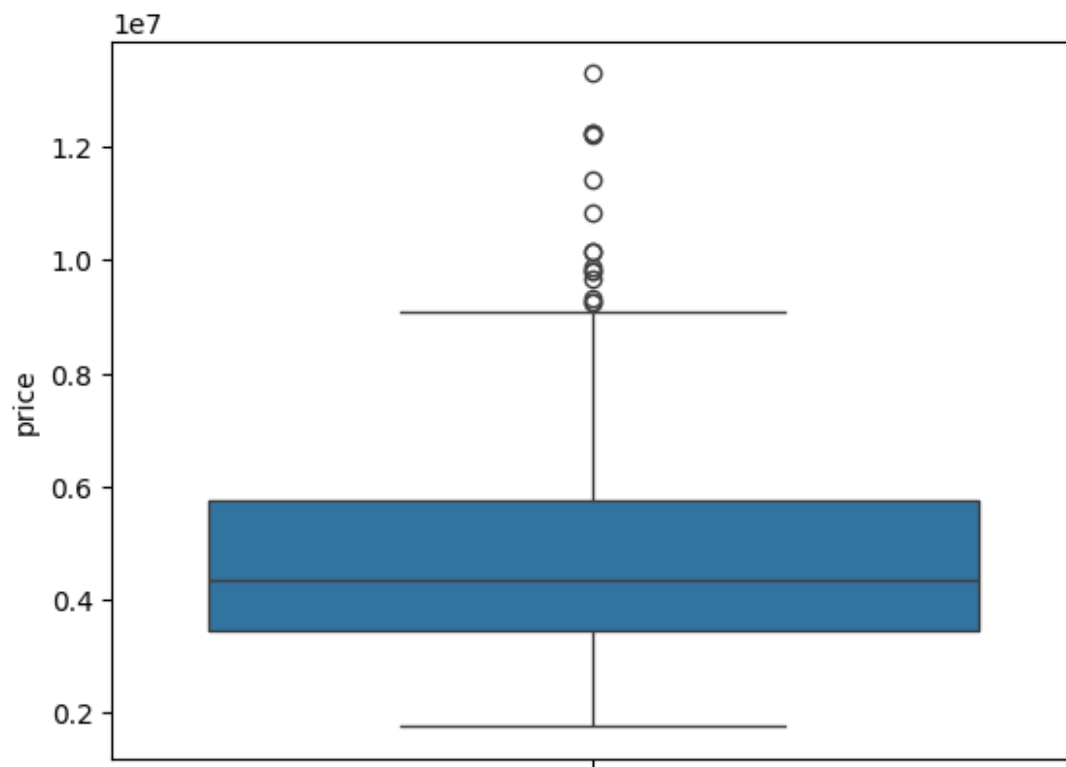
```
plt.xlabel('Price')  
Text(0.5, 0, 'Price')
```



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

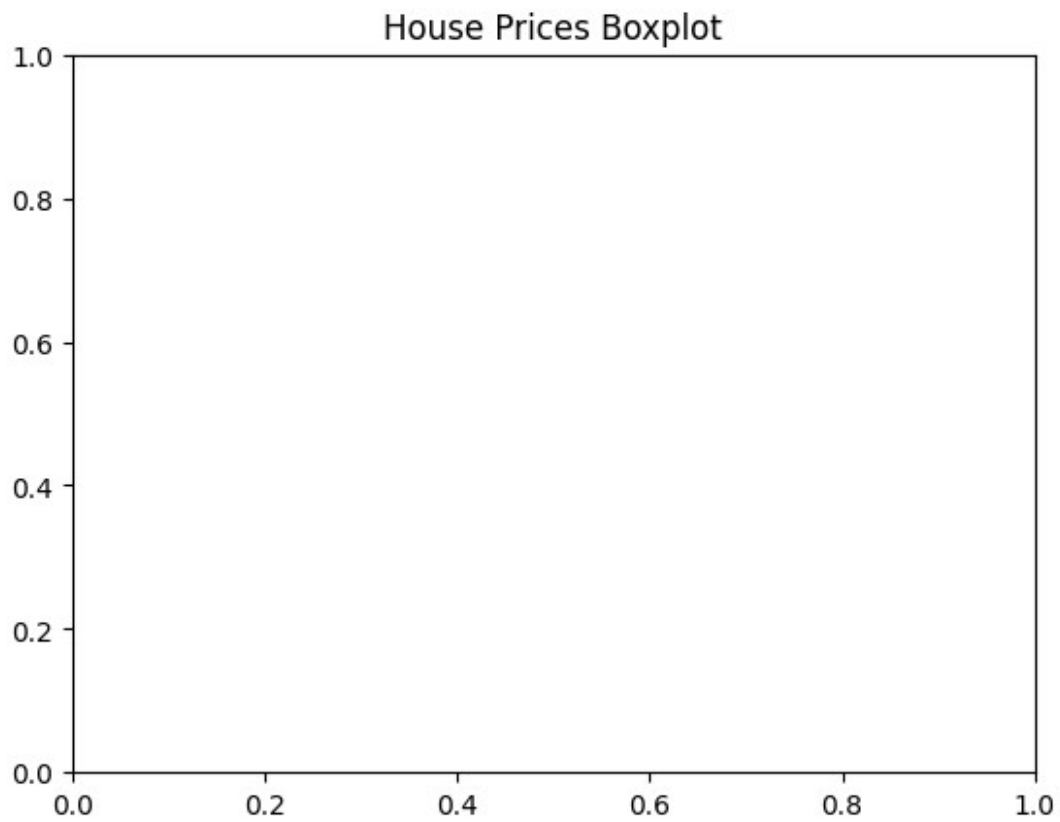


```
plt.show()
# Boxplot of Price
plt.figure(figsize=(4,6))
<Figure size 400x600 with 0 Axes>
<Figure size 400x600 with 0 Axes>
sns.boxplot(y=df['price'])
<Axes: ylabel='price'>
```

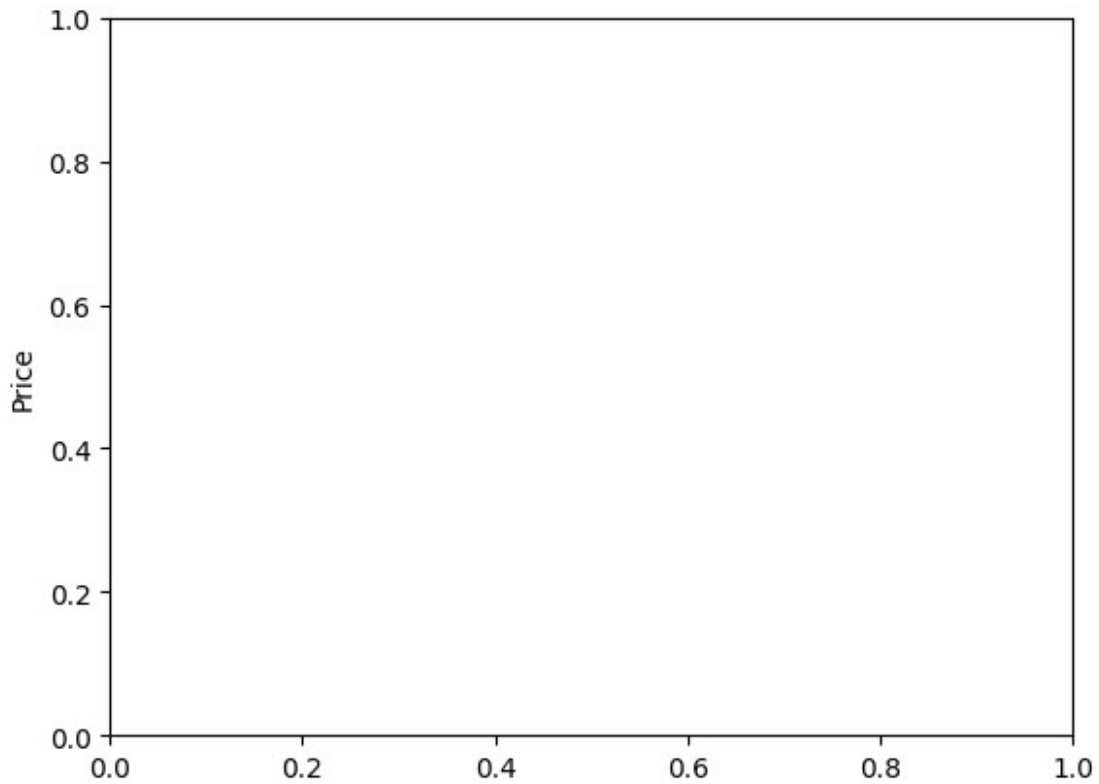


```
plt.title('House Prices Boxplot')
```

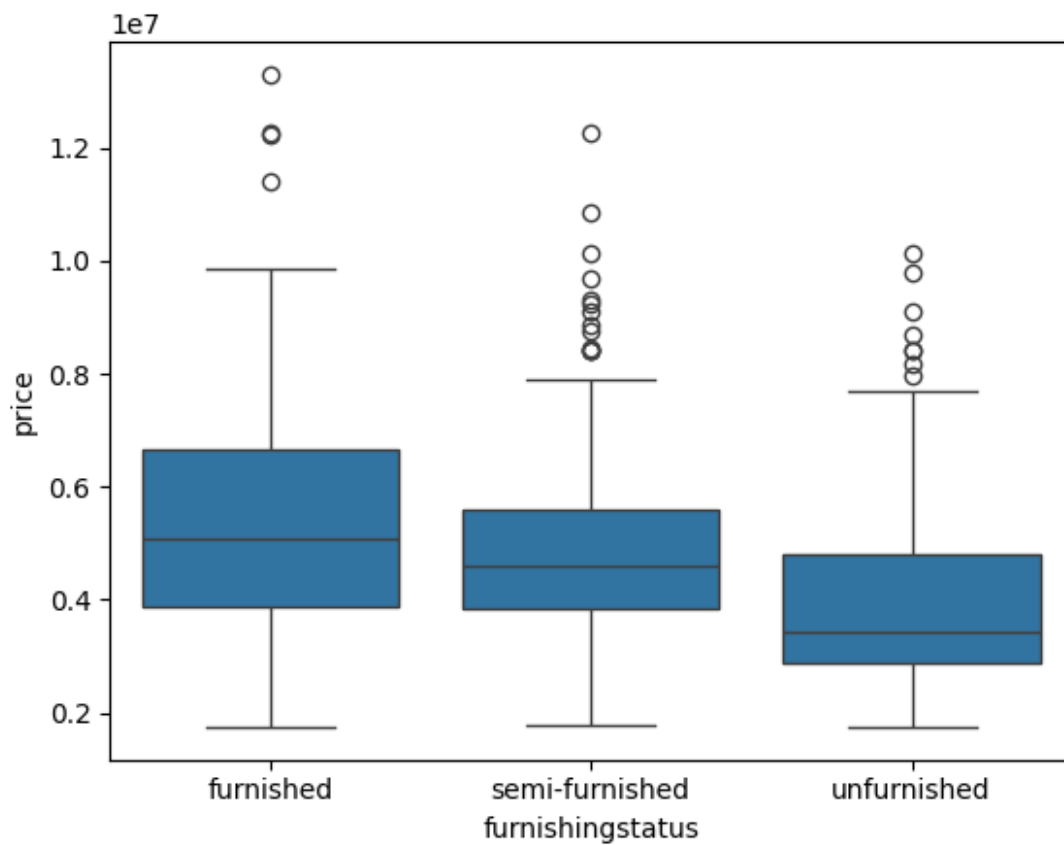
```
Text(0.5, 1.0, 'House Prices Boxplot')
```



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

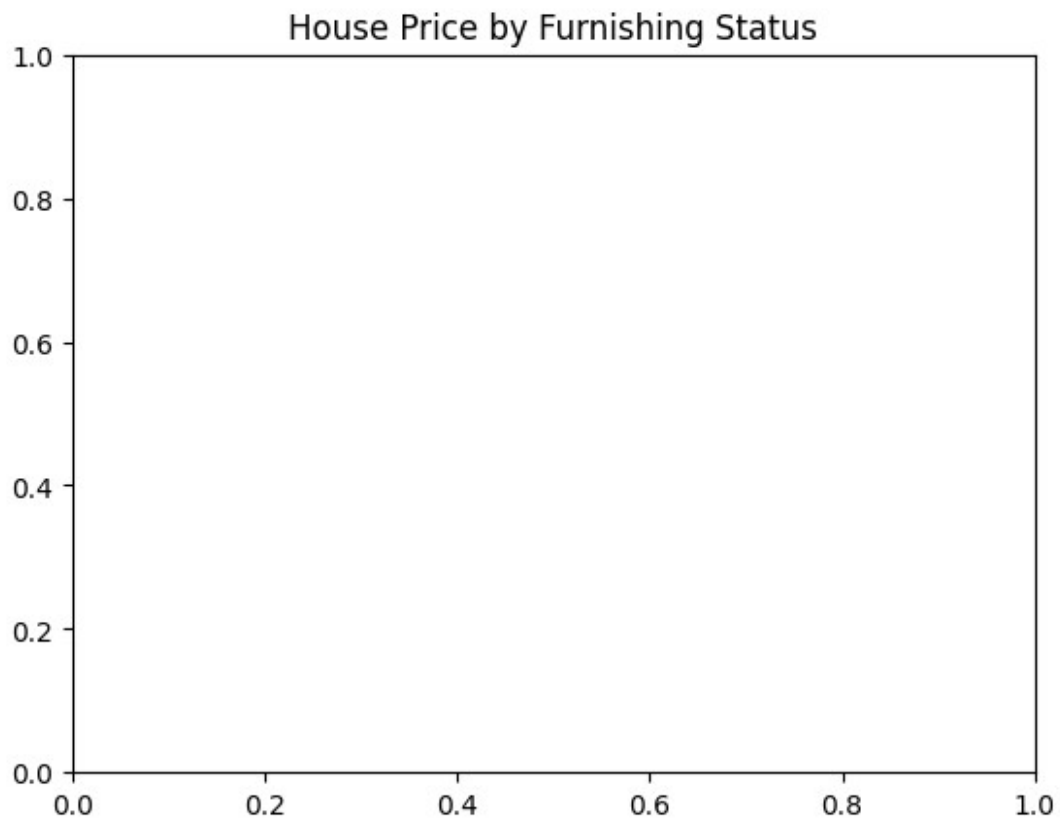


```
plt.show()
# Boxplot of Price by furnishing status
plt.figure(figsize=(6,4))
<Figure size 600x400 with 0 Axes>
<Figure size 600x400 with 0 Axes>
sns.boxplot(x=df['furnishingstatus'], y=df['price'])
<Axes: xlabel='furnishingstatus', ylabel='price'>
```

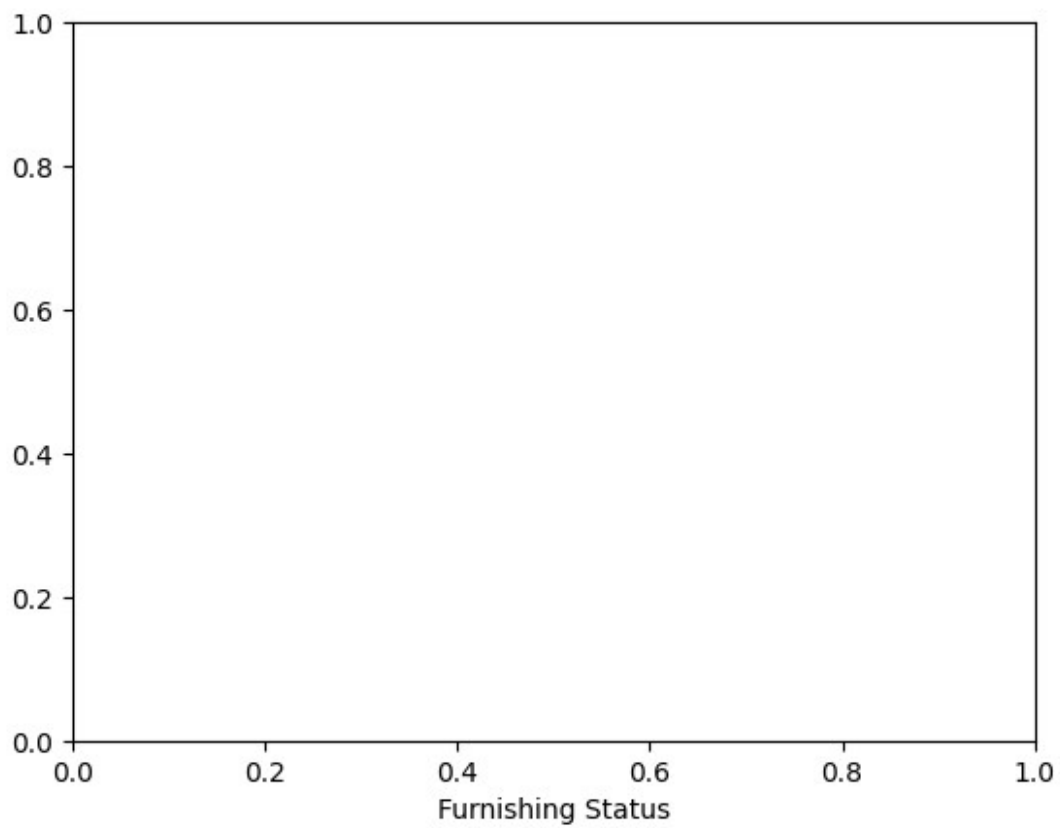


```
plt.title('House Price by Furnishing Status')
```

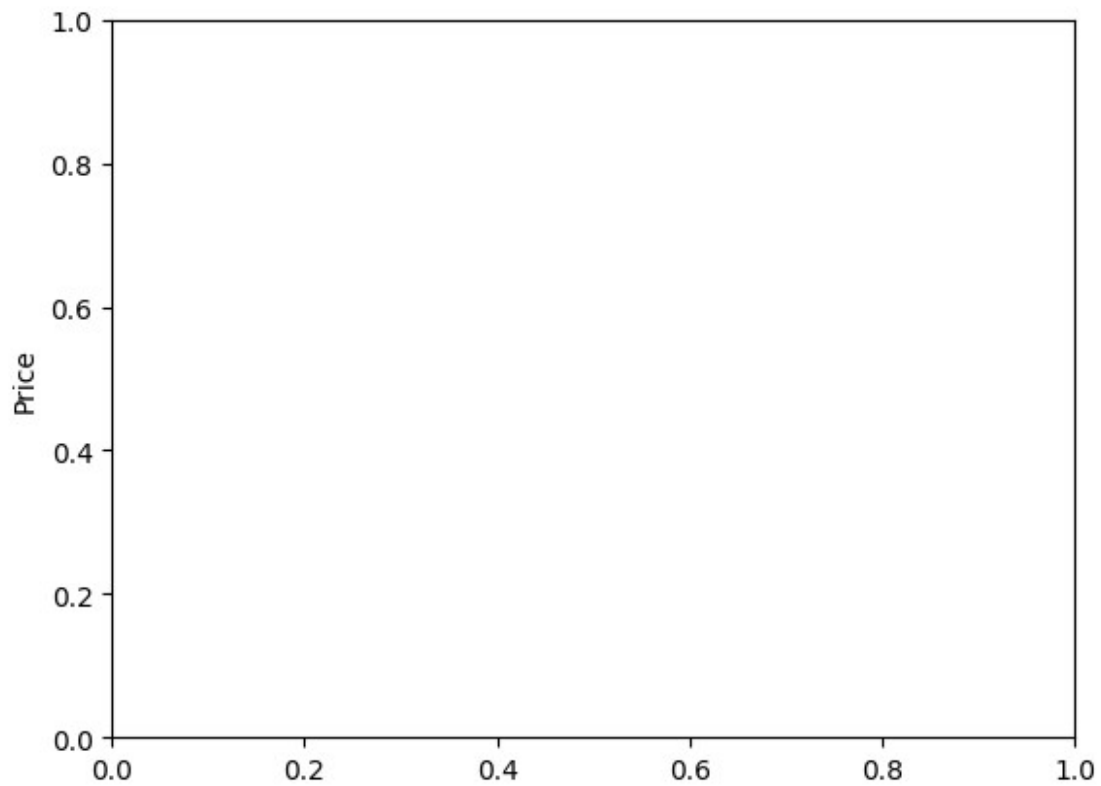
```
Text(0.5, 1.0, 'House Price by Furnishing Status')
```



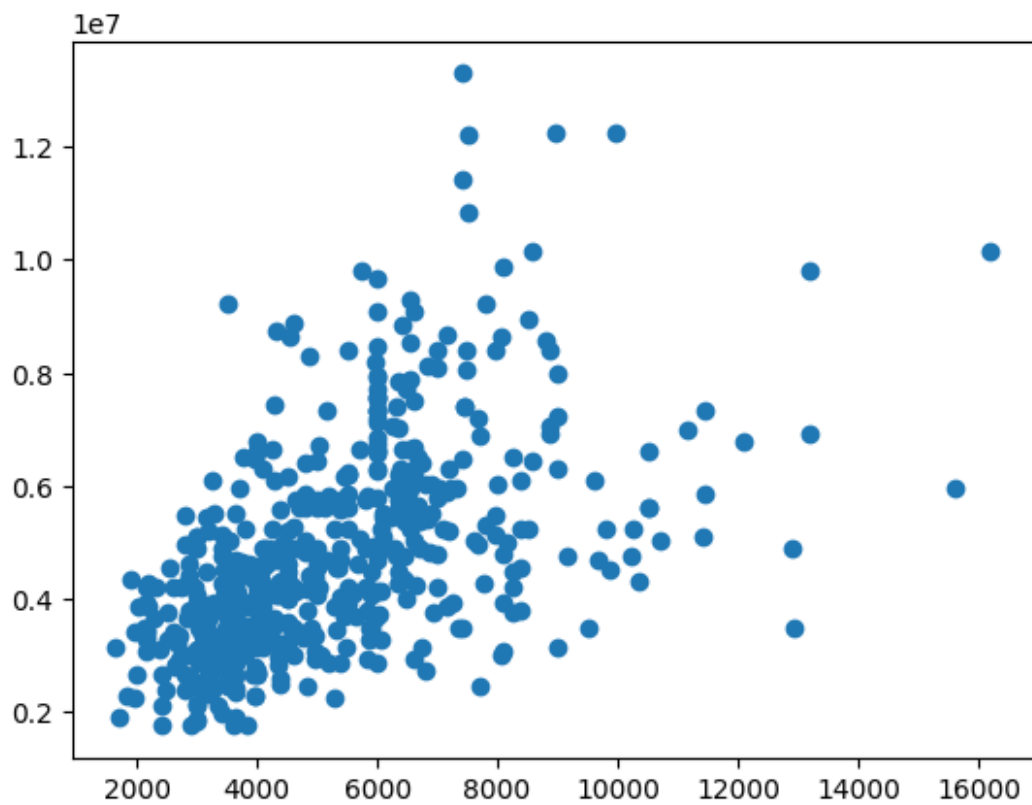
```
plt.xlabel('Furnishing Status')  
Text(0.5, 0, 'Furnishing Status')
```



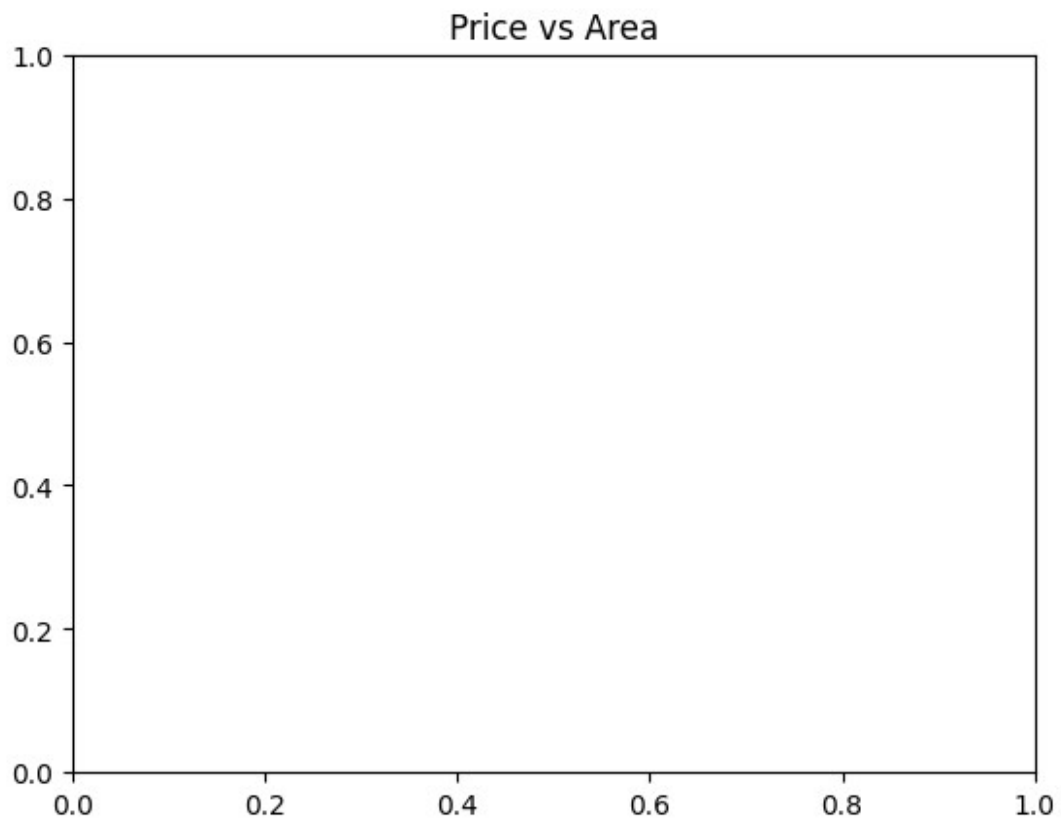
```
plt.ylabel('Price')  
Text(0, 0.5, 'Price')
```



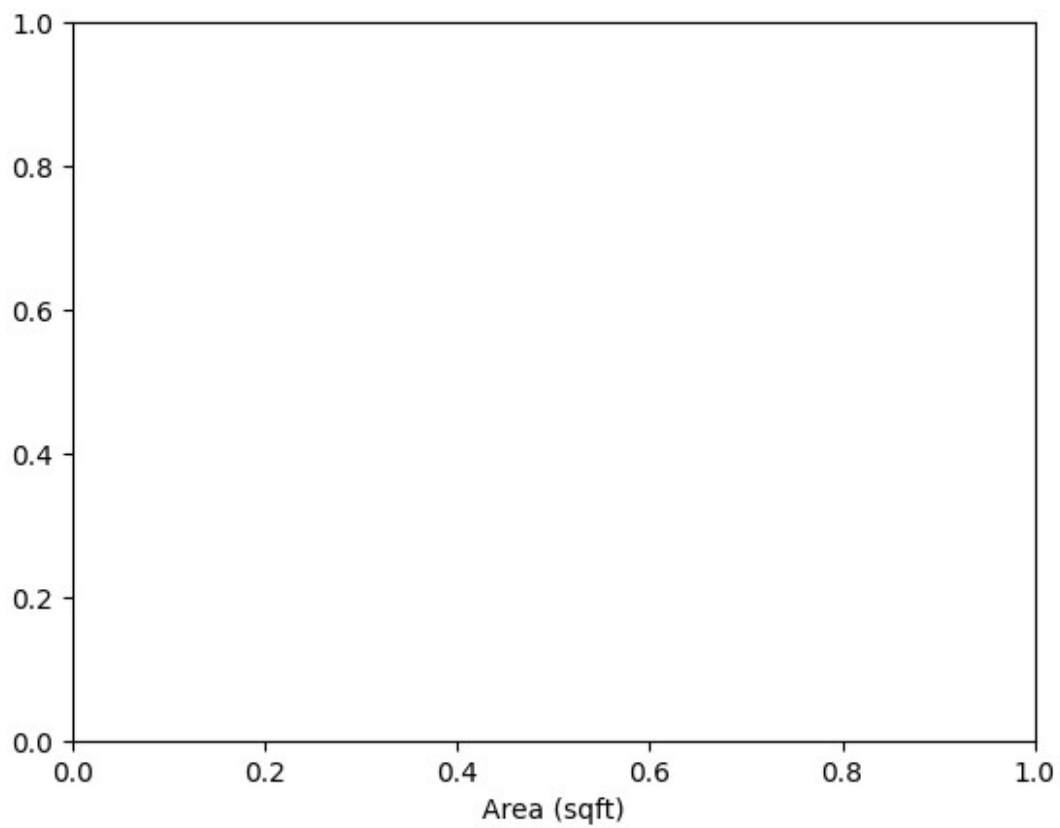
```
plt.show()
# Scatter plot of Price vs Area
plt.figure(figsize=(6,5))
<Figure size 600x500 with 0 Axes>
<Figure size 600x500 with 0 Axes>
plt.scatter(df['area'], df['price'])
<matplotlib.collections.PathCollection at 0x278193854f0>
```



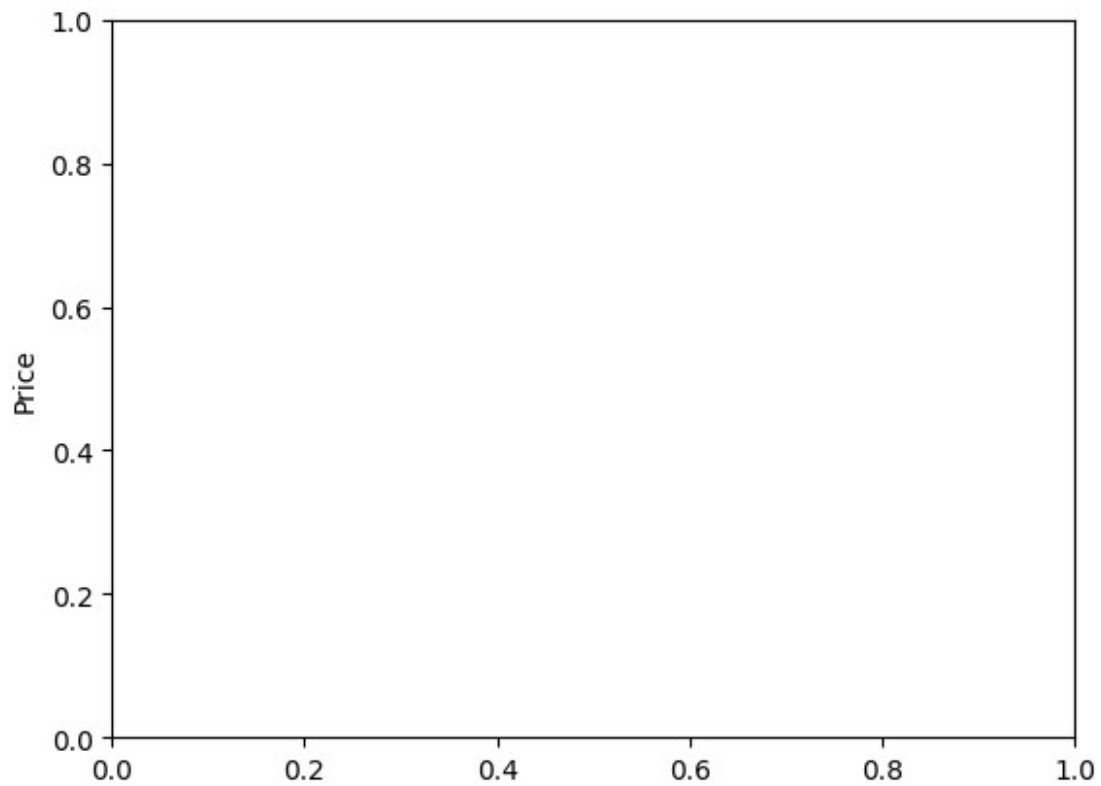
```
plt.title('Price vs Area')  
Text(0.5, 1.0, 'Price vs Area')
```



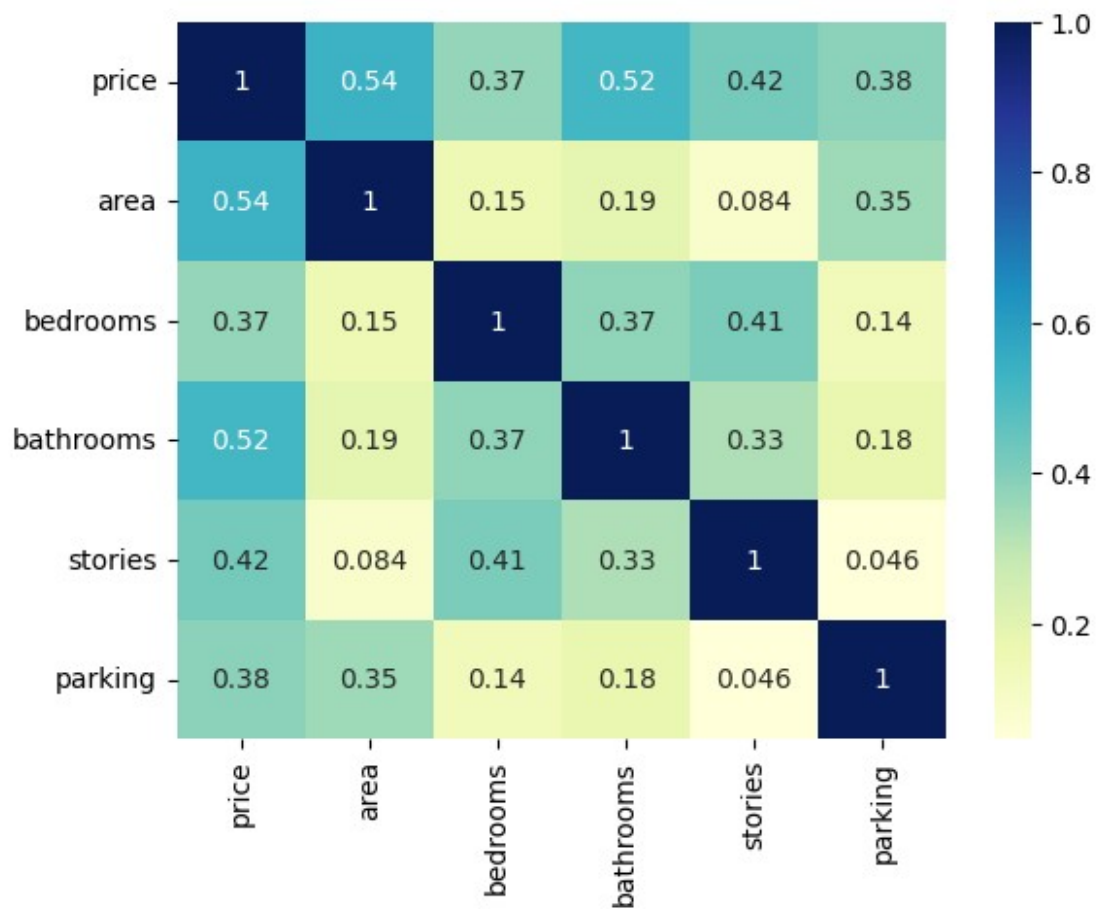
```
plt.xlabel('Area (sqft)')  
Text(0.5, 0, 'Area (sqft)')
```



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

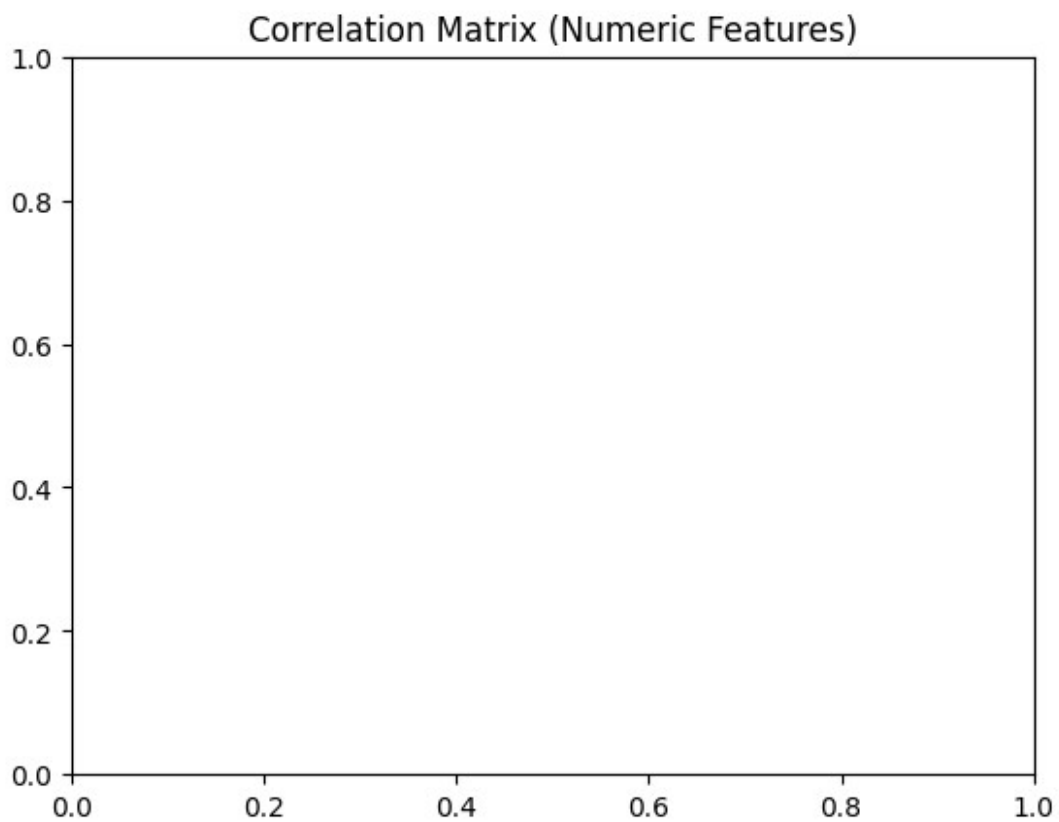


```
plt.show()
# Correlation heatmap for numeric features
plt.figure(figsize=(6,5))
<Figure size 600x500 with 0 Axes>
<Figure size 600x500 with 0 Axes>
corr = df[numeric_cols].corr()
sns.heatmap(corr, annot=True, cmap='YlGnBu')
<Axes: >
```



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

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



```
plt.show()
```