

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

```
In [2]: # Load your dataset into a DataFrame
df = pd.read_csv('Housing.csv')
# Preview the first 5 rows
df.head()
```

```
Out[2]:
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement
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	

```
In [3]: df.dtypes
```

```
Out[3]: price          int64
area          int64
bedrooms      int64
bathrooms     int64
stories       int64
mainroad      object
guestroom     object
basement      object
hotwaterheating object
airconditioning object
parking       int64
prefarea      object
furnishingstatus object
dtype: object
```

```
In [4]: df.dtypes
```

```
Out[4]: price          int64
area          int64
bedrooms      int64
bathrooms     int64
stories       int64
mainroad      object
guestroom     object
basement      object
hotwaterheating object
airconditioning object
parking       int64
prefarea      object
furnishingstatus object
dtype: object
```

```
In [5]: df.describe()
```

```
Out[5]:
```

	price	area	bedrooms	bathrooms	stories	pa
count	5.450000e+02	545.000000	545.000000	545.000000	545.000000	545.00
mean	4.766729e+06	5150.541284	2.965138	1.286239	1.805505	0.69
std	1.870440e+06	2170.141023	0.738064	0.502470	0.867492	0.86
min	1.750000e+06	1650.000000	1.000000	1.000000	1.000000	0.00
25%	3.430000e+06	3600.000000	2.000000	1.000000	1.000000	0.00
50%	4.340000e+06	4600.000000	3.000000	1.000000	2.000000	0.00
75%	5.740000e+06	6360.000000	3.000000	2.000000	2.000000	1.00
max	1.330000e+07	16200.000000	6.000000	4.000000	4.000000	3.00

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

```
Out[6]: 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 [7]: missing_percentage = df.isnull().mean() * 100
threshold = 30
# Filter columns with more than the threshold percentage of missing values
columns_to_drop = missing_percentage[missing_percentage > threshold].index
# Drop the columns
df = df.drop(columns=columns_to_drop)
# Display the resulting dataframe after removing columns with too many missing values
print(f"Columns dropped: {list(columns_to_drop)}")
print(f"Remaining columns: {df.columns}")
```

Columns dropped: []

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

```
In [11]: import pandas as pd
from sklearn.preprocessing import LabelEncoder

# Sample DataFrame (replace with your actual data)
```

```
# df = pd.read_csv('Housing.csv')

label_encoder = LabelEncoder()

for column in df.columns:
    df[column] = label_encoder.fit_transform(df[column])
df.head()
```

```
Out[11]:
```

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	baseme
0	218	232	3	1	2	1	0	
1	217	260	3	3	3	1	0	
2	217	268	2	1	1	1	0	
3	216	237	3	1	1	1	0	
4	215	232	3	0	1	1	1	

```
In [ ]: # Fill missing values in columns with mode
for column in df.select_dtypes(include=['float64']).columns:
    mean_value = df[column].mean() # Get the mode (most frequent value)
    df[column] = df[column].fillna(mean_value)
```

```
In [ ]: # Fill missing values in categorical columns with mode
for column in df.select_dtypes(include=['object']).columns:
    mode_value = df[column].mode()[0] # Get the mode (most frequent value)
    df[column] = df[column].fillna(mode_value)
```

```
In [13]: df.isnull().sum()
```

```
Out[13]: 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 [14]: df
```

Out[14]:

	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement
0	218	232	3	1	2	1	0	
1	217	260	3	3	3	1	0	
2	217	268	2	1	1	1	0	
3	216	237	3	1	1	1	0	
4	215	232	3	0	1	1	1	
...	...	...	...	...	...	...	...	...
540	2	39	1	0	0	1	0	
541	1	15	2	0	0	0	0	
542	0	72	1	0	0	1	0	
543	0	35	2	0	0	0	0	
544	0	90	2	0	1	1	0	

545 rows × 9 columns

```
In [ ]: def replace_outliers_with_nan(column):
Q1 = column.quantile(0.25)
Q3 = column.quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
# Replace outliers with NaN
return column.where((column >= lower_bound) & (column <= upper_bound), np.nan)
# Replace outliers with NaN for each feature
for col in df.columns:
df[col] = replace_outliers_with_nan(df[col])
```

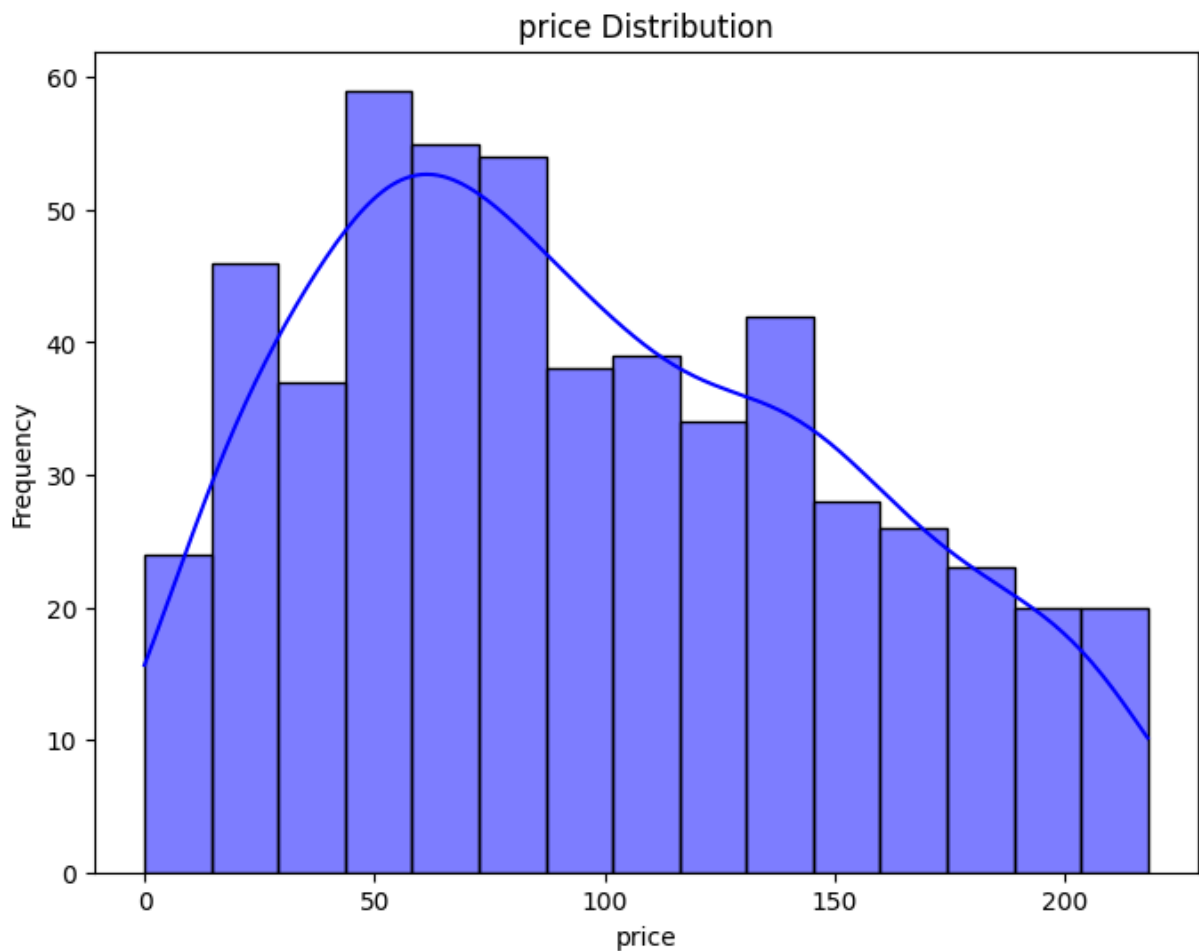
```
In [15]: import pandas as pd
# Example: Load a CSV file into 'data'
data = pd.read_csv("Housing.csv") # Make sure this line was run before
# Now this will work:
null_counts = data.isnull().sum()
print("Sum of null values in each column:")
print(null_counts)
```

Sum of null values in each column:

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 [18]: import matplotlib.pyplot as plt
import seaborn as sns
# 1. Distribution of Age
plt.figure(figsize=(8, 6))
sns.histplot(df['price'], bins=15, kde=True, color='blue')
plt.title('price Distribution')
plt.xlabel('price')
plt.ylabel('Frequency')
plt.show()
```

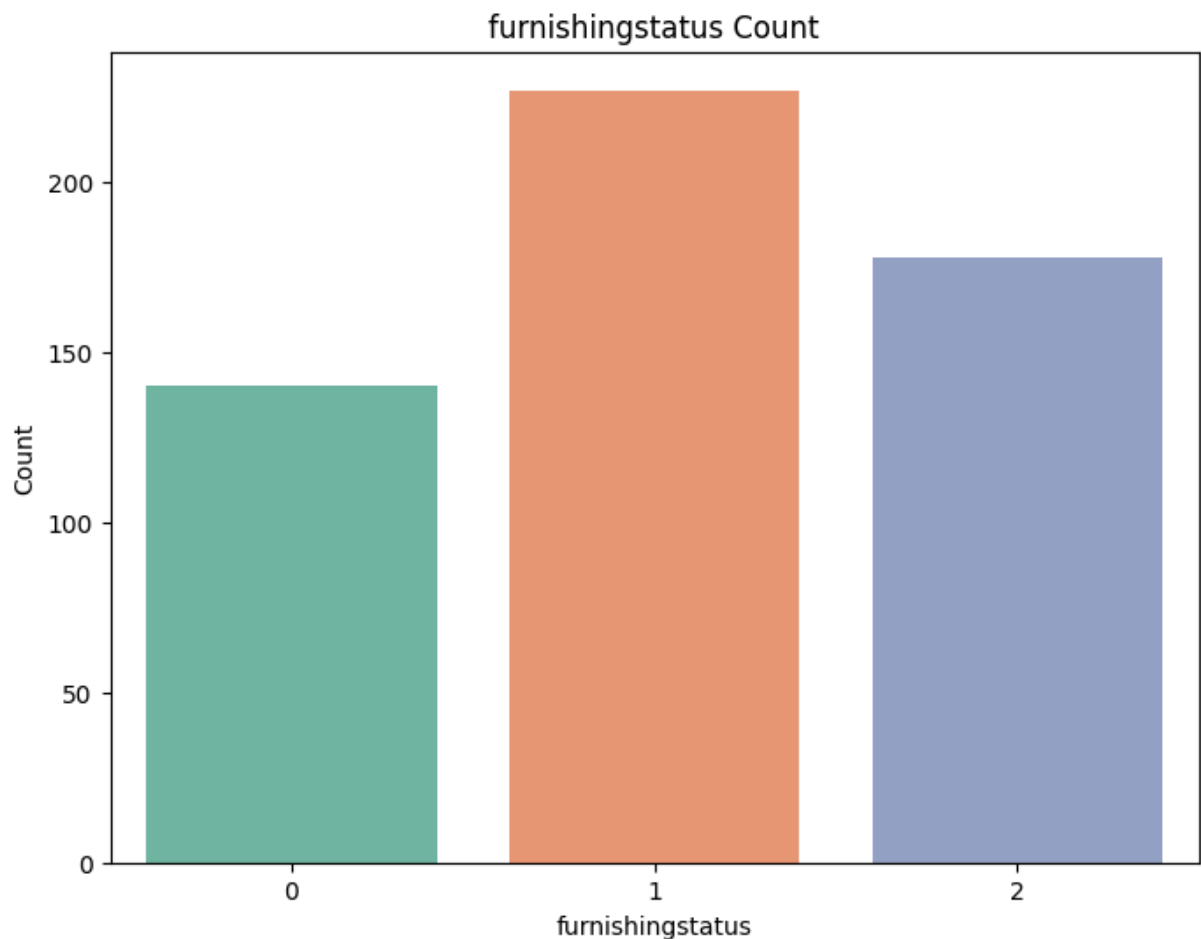


```
In [20]: # 3. Countplot for Classification
plt.figure(figsize=(8, 6))
sns.countplot(x='furnishingstatus', data=df, palette='Set2')
plt.title('furnishingstatus Count')
plt.xlabel('furnishingstatus')
plt.ylabel('Count')
plt.show()
```

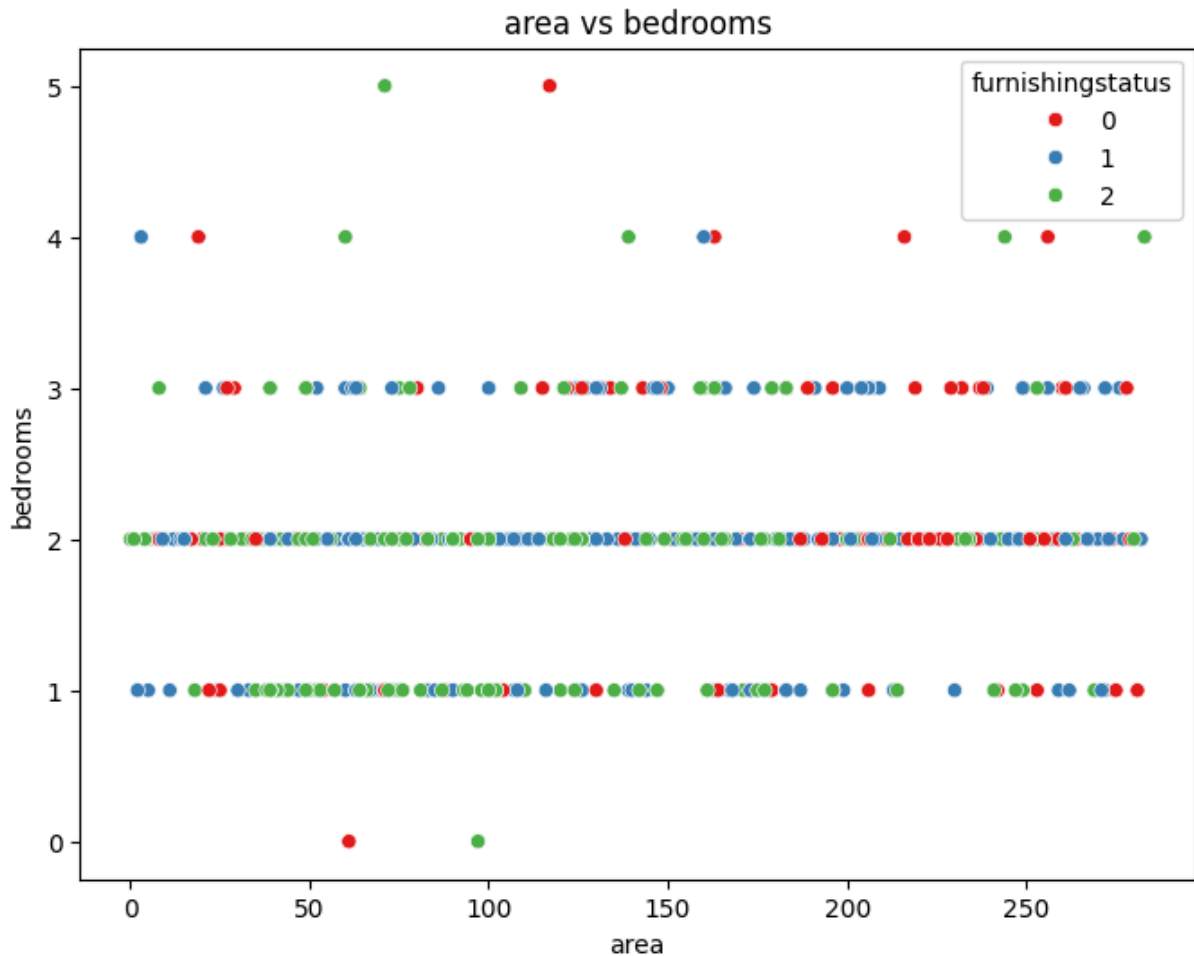
C:\Users\alkya\AppData\Local\Temp\ipykernel_3860\3287708615.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
sns.countplot(x='furnishingstatus', data=df, palette='Set2')
```



```
In [27]: plt.figure(figsize=(8, 6))
sns.scatterplot(x='area', y='bedrooms', data=df, hue='furnishingstatus', pal
plt.title('area vs bedrooms')
plt.xlabel('area')
plt.ylabel('bedrooms')
plt.show()
```

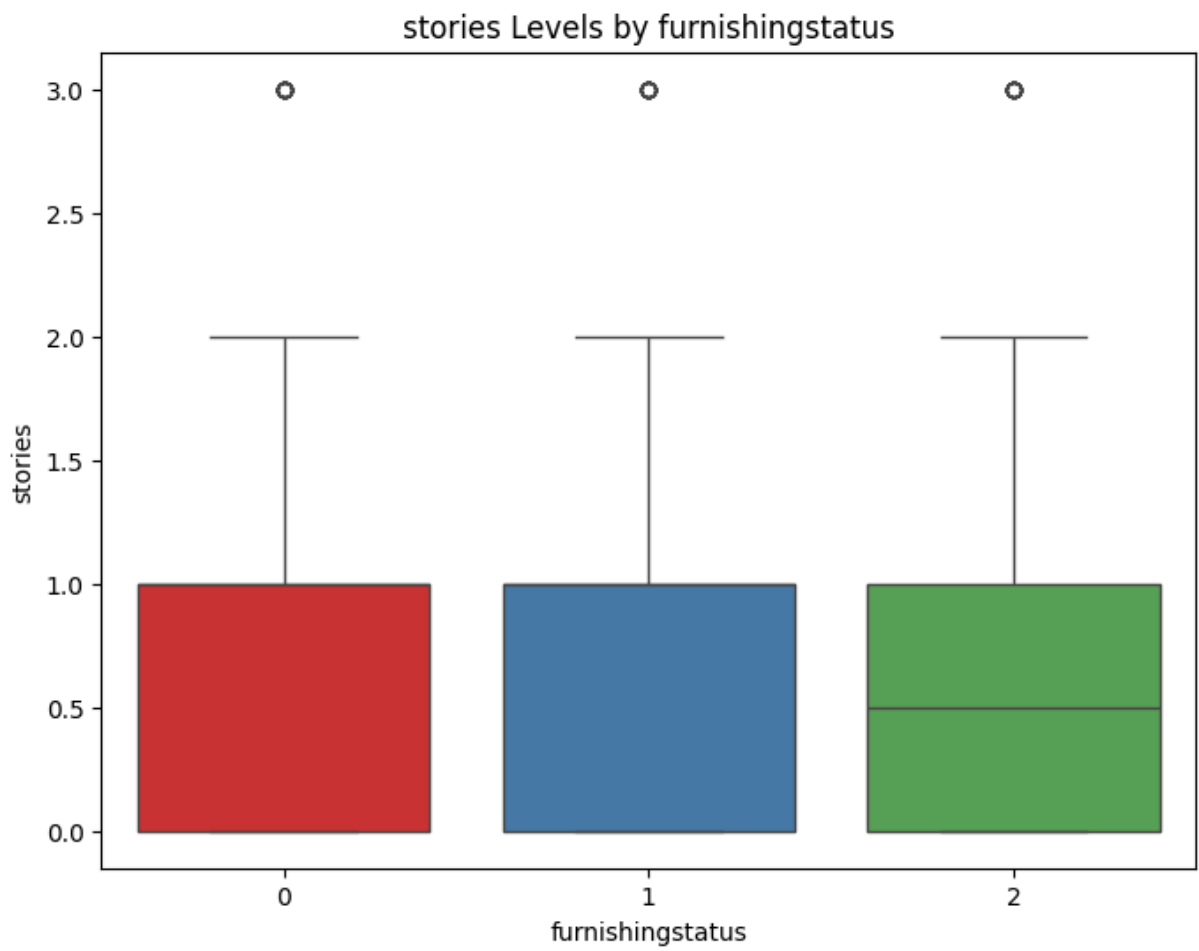


```
In [28]: # Boxplot for Hemoglobin by Classification
plt.figure(figsize=(8, 6))
sns.boxplot(x='furnishingstatus', y='stories', data=df, palette='Set1')
plt.title('stories Levels by furnishingstatus')
plt.xlabel('furnishingstatus')
plt.ylabel('stories')
plt.show()
```

C:\Users\alkya\AppData\Local\Temp\ipykernel_3860\514798808.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

```
sns.boxplot(x='furnishingstatus', y='stories', data=df, palette='Set1')
```



In []: