In [ ]:
```
#DIGITAL ASSIGNMENT - 3
#Girish Kumar A
#20MID0170
```

In [30]:
```
import pandas as pd
import numpy as np
from scipy import stats
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
import seaborn as sns
```

In [31]:
```
# Step 1: Load the dataset
data = pd.read_csv("Housing.csv")
```

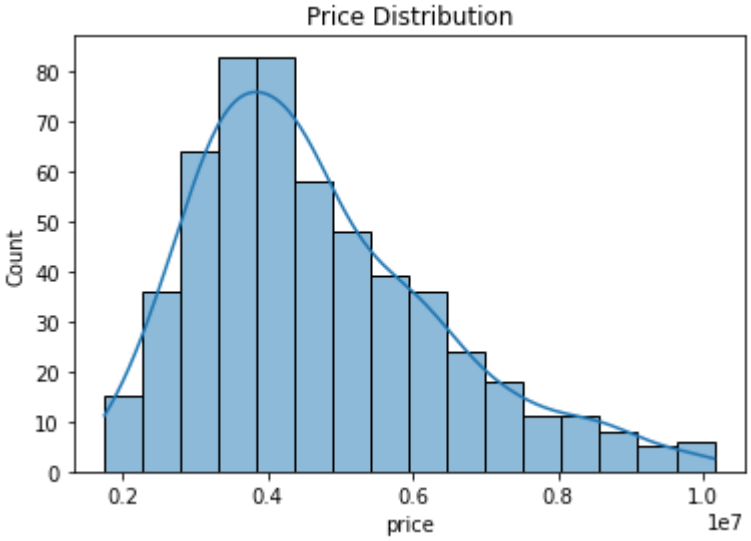
In [32]:
```
# Step 2: Perform data exploration and preprocessing
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 12 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  furnishingstatus  545 non-null    object
dtypes: int64(6), object(6)
memory usage: 51.2+ KB
None
```

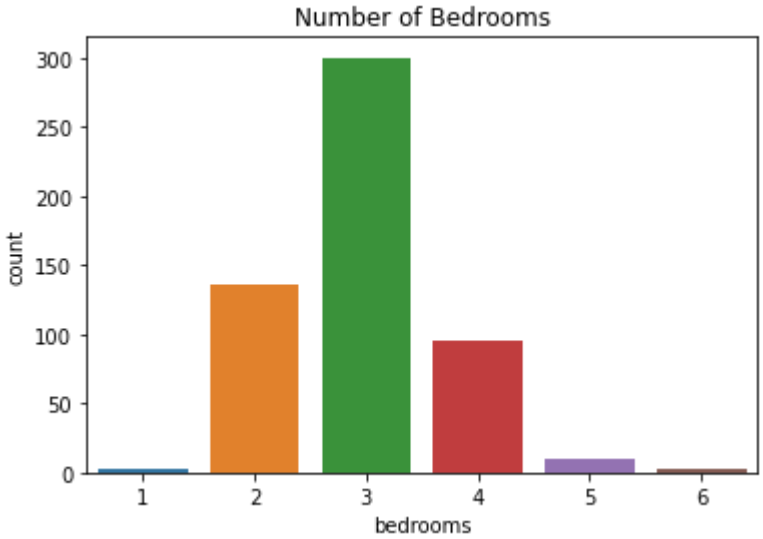
In [55]:
```
# Step 3: Visualizations (univariate, bivariate, multivariate)
# Perform exploratory data analysis (EDA) using visualizations

# Univariate analysis
sns.histplot(data['price'], kde=True)
plt.title('Price Distribution')
plt.show()

sns.countplot(data['bedrooms'])
plt.title('Number of Bedrooms')
plt.show()
```



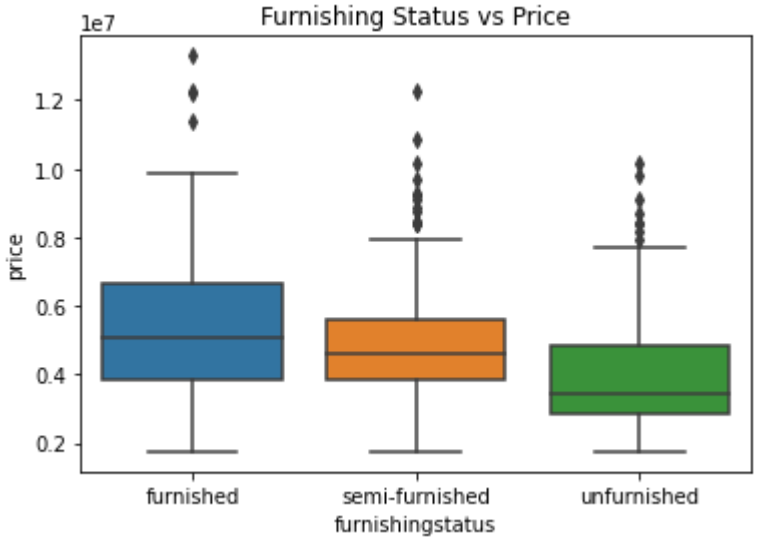

```
C:\Users\Neethu\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pa
ss the following variable as a keyword arg: x. From version 0.12, the only valid positio
nal argument will be `data`, and passing other arguments without an explicit keyword wil
l result in an error or misinterpretation.
  warnings.warn(
```



In [34]:

```
# Bivariate analysis
sns.scatterplot(x='area', y='price', data=data)
plt.title('Area vs Price')
plt.show()

sns.boxplot(x='furnishingstatus', y='price', data=data)
plt.title('Furnishing Status vs Price')
plt.show()
```



In [35]:

```
# Multivariate analysis
sns.pairplot(data)
plt.show()
```

In [36]:

```
# Step 4: Descriptive statistics
print(data.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
```

In [37]:

```
# Step 5: Handle missing values
data.dropna(inplace=True)  # Remove rows with missing values or impute missing values
print("After handling missing values:")
print(data.info())
```

```
After handling missing values:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 545 entries, 0 to 544
Data columns (total 12 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  furnishingstatus  545 non-null    object
dtypes: int64(6), object(6)
memory usage: 55.4+ KB
None
```

In [38]:

```
# Step 6: Handle outliers
# Identify outliers using techniques like z-score, IQR, etc. and replace or remove them
# Identify outliers using Z-score
z_scores = np.abs(stats.zscore(data['price']))
threshold = 3  # Set the threshold for outliers

outlier_indices = np.where(z_scores > threshold)[0]
outlier_values = data.loc[outlier_indices, 'price']

# Handle outliers by replacing them with the median value
median_value = data['price'].median()
data.loc[outlier_indices, 'price'] = median_value
print("After handling outliers:")
print(data.describe())
```

```
After handling outliers:
              price          area    bedrooms   bathrooms     stories  \
count  5.450000e+02    545.000000  545.000000  545.000000  545.000000   
mean   4.681894e+06   5150.541284    2.965138    1.286239    1.805505   
std    1.703666e+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%    5.600000e+06   6360.000000    3.000000    2.000000    2.000000   
max    1.015000e+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
```

In [48]:

```
# Step 7: Perform one-hot encoding on categorical variables
print("Original data:")
print(data.head())

data_encoded = pd.get_dummies(data, drop_first=True)
print("Encoded data:")
print(data_encoded.head())
```

```
Original data:
     price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
0  4340000  7420         4          2        3      yes        no       no
1  4340000  8960         4          4        4      yes        no       no
2  4340000  9960         3          2        2      yes        no      yes
3  4340000  7500         4          2        2      yes        no      yes
4  4340000  7420         4          1        2      yes       yes      yes

  hotwaterheating airconditioning  parking furnishingstatus
0              no             yes        2        furnished
1              no             yes        3        furnished
2              no              no        2   semi-furnished
3              no             yes        3        furnished
4              no             yes        2        furnished
Encoded data:
     price  area  bedrooms  bathrooms  stories  parking  mainroad_yes  \
0  4340000  7420         4          2        3        2             1
1  4340000  8960         4          4        4        3             1
2  4340000  9960         3          2        2        2             1
3  4340000  7500         4          2        2        3             1
4  4340000  7420         4          1        2        2             1

   guestroom_yes  basement_yes  hotwaterheating_yes  airconditioning_yes  \
0              0             0                    0                    1
1              0             0                    0                    1
2              0             1                    0                    0
3              0             1                    0                    1
4              1             1                    0                    1

   furnishingstatus_semi-furnished  furnishingstatus_unfurnished
0                                0                             0
1                                0                             0
2                                1                             0
3                                0                             0
4                                0                             0
```

In [47]:

```
# Step 8: Split the data into dependent and independent variables
X = data_encoded.drop('price', axis=1)  # Independent variables
y = data_encoded['price']  # Dependent variable

print("Independent variables:")
print(X.head())
```

```
print("Dependent variable:")
print(y.head())
```

```
Independent variables:
   area  bedrooms  bathrooms  stories  parking  mainroad_yes  guestroom_yes  \
0  7420         4          2        3        2             1              0
1  8960         4          4        4        3             1              0
2  9960         3          2        2        2             1              0
3  7500         4          2        2        3             1              0
4  7420         4          1        2        2             1              1

   basement_yes  hotwaterheating_yes  airconditioning_yes  \
0             0                    0                    1
1             0                    0                    1
2             1                    0                    0
3             1                    0                    1
4             1                    0                    1

   furnishingstatus_semi-furnished  furnishingstatus_unfurnished
0                                0                             0
1                                0                             0
2                                1                             0
3                                0                             0
4                                0                             0
Dependent variable:
0    4340000
1    4340000
2    4340000
3    4340000
4    4340000
Name: price, dtype: int64
```

In [49]:

```
# Step 9: Scale the independent variables
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

print("Scaled independent variables:")
print(X_scaled[:5])
```

```
Scaled independent variables:
[[ 1.04672629  1.40341936  1.42181174  1.37821692  1.51769249  0.40562287
  -0.46531479 -0.73453933 -0.2192645   1.4726183  -0.84488844 -0.6964292 ]
 [ 1.75700953  1.40341936  5.40580863  2.53202371  2.67940935  0.40562287
  -0.46531479 -0.73453933 -0.2192645   1.4726183  -0.84488844 -0.6964292 ]
 [ 2.21823241  0.04727831  1.42181174  0.22441013  1.51769249  0.40562287
  -0.46531479  1.3613975  -0.2192645  -0.67906259  1.18358821 -0.6964292 ]
 [ 1.08362412  1.40341936  1.42181174  0.22441013  2.67940935  0.40562287
  -0.46531479  1.3613975  -0.2192645   1.4726183  -0.84488844 -0.6964292 ]
 [ 1.04672629  1.40341936 -0.57018671  0.22441013  1.51769249  0.40562287
   2.14908276  1.3613975  -0.2192645   1.4726183  -0.84488844 -0.6964292 ]]
```

In [50]:

```
# Step 10: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_
print("Train-test split:")
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

```
Train-test split:
X_train shape: (436, 12)
X_test shape: (109, 12)
y_train shape: (436,)
y_test shape: (109,)
```

In [51]:

```
# Step 11: Build the model
model = LinearRegression()

# Step 12: Train the model
model.fit(X_train, y_train)
print("Model training complete.")
```

```
Model training complete.
```

In [52]:

```
# Step 13: Test the model
y_pred = model.predict(X_test)
print("Model testing complete.")
```

```
Model testing complete.
```

In [53]:

```
# Step 14: Measure the performance using metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R-squared Score:", r2)
```

```
Mean Squared Error: 1463448549682.701
R-squared Score: 0.6273081687625075
```