In [1]:

```python
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from catboost import CatBoostRegressor
from sklearn.model_selection import train_test_split,KFold, StratifiedKFol
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
import matplotlib.pyplot as plt
from lightgbm import LGBMRegressor
from sklearn.impute import KNNImputer

import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```python
train = pd.read_csv("Housing_dataset_train.csv")
test = pd.read_csv("Housing_dataset_test.csv")
sub = pd.read_csv("Sample_submission.csv")
#var = pd.read_csv("VariableDefinitions.csv")
```

In [3]:

```python
train
```

Out[3]:

	ID	loc	title	bedroom	bathroom	parking_space	price
0	3583	Katsina	Semi-detached duplex	2.0	2.0	1.0	1149999.565
1	2748	Ondo	Apartment	NaN	2.0	4.0	1672416.689
2	9261	Ekiti	NaN	7.0	5.0	NaN	3364799.814
3	2224	Anambra	Detached duplex	5.0	2.0	4.0	2410306.756
4	10300	Kogi	Terrace duplex	NaN	5.0	6.0	2600700.898
...	...	...	...	...	...	...	...
13995	6175	Edo	Bungalow	NaN	7.0	NaN	2367927.861
13996	9704	Kaduna	Apartment	NaN	7.0	5.0	2228516.471
13997	11190	Plateau	Bungalow	8.0	6.0	5.0	2406812.693
13998	9256	Delta	Flat	NaN	6.0	1.0	3348918.718
13999	8787	Nasarawa	NaN	9.0	7.0	5.0	2858516.890

14000 rows × 7 columns

In [4]:
```python
train.describe()
```

Out[4]:

	ID	bedroom	bathroom	parking_space	price
count	14000.000000	12201.000000	12195.000000	12189.000000	1.400000e+04
mean	4862.700357	4.308171	3.134235	3.169825	2.138082e+06
std	3818.348214	2.441165	2.035950	1.599415	1.083057e+06
min	0.000000	1.000000	1.000000	1.000000	4.319673e+05
25%	1672.750000	2.000000	1.000000	2.000000	1.393990e+06
50%	3527.000000	4.000000	2.000000	3.000000	1.895223e+06
75%	8011.250000	6.000000	5.000000	4.000000	2.586699e+06
max	12999.000000	9.000000	7.000000	6.000000	1.656849e+07

How many values are missing in each column?

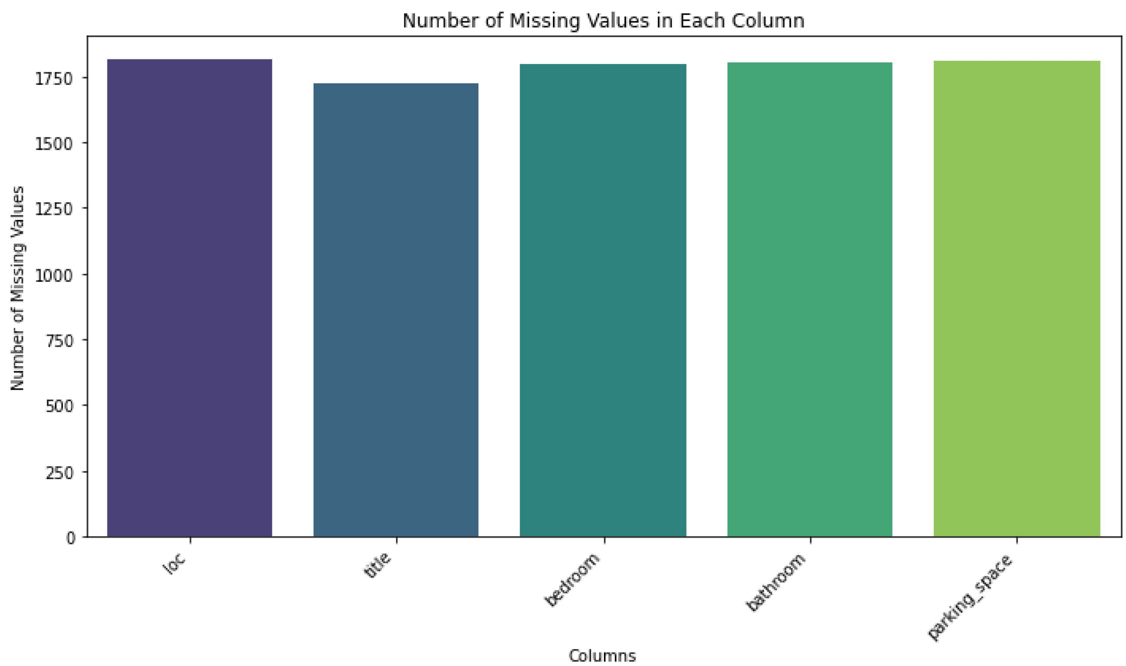
From the plot below, we can see that the location represented as "loc" has the most missing data

In [5]:
```python
data = train.drop(["price"], axis=1).append(test)
data = data.drop("ID", axis=1)
data.isnull().sum()
```

Out[5]:
```
loc              1813
title            1722
bedroom          1799
bathroom         1805
parking_space    1811
dtype: int64
```

In [6]:

```
# Calculate the number of missing values in each column
missing_values_count = data.isnull().sum()

# Create a colorful bar chart
plt.figure(figsize=(10, 6))
sns.barplot(x=missing_values_count.index, y=missing_values_count.values, p

# Set plot properties
plt.xlabel('Columns')
plt.ylabel('Number of Missing Values')
plt.title('Number of Missing Values in Each Column')
plt.xticks(rotation=45, ha='right')

plt.tight_layout()
plt.show()
```



Exploring the data

In [7]:

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


# Step 1: Check General Information
print(data.info())

# Step 2: Summary Statistics
print(data.describe())
```

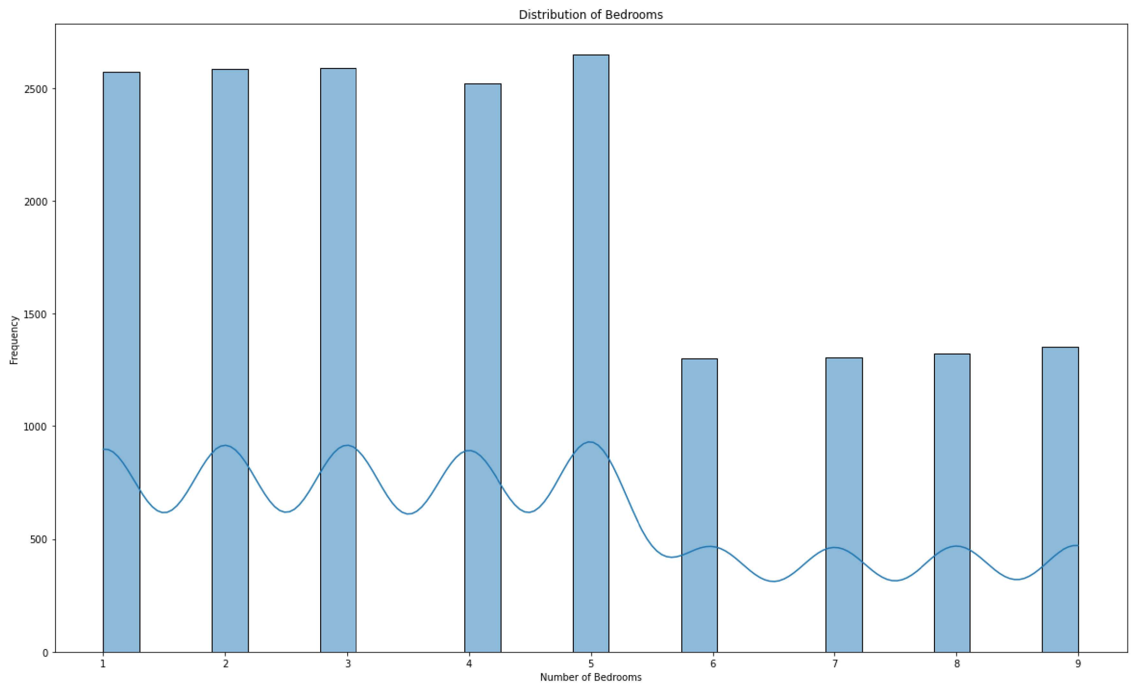
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20000 entries, 0 to 5999
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   loc            18187 non-null  object 
 1   title          18278 non-null  object 
 2   bedroom        18201 non-null  float64
 3   bathroom       18195 non-null  float64
 4   parking_space  18189 non-null  float64
dtypes: float64(3), object(2)
memory usage: 937.5+ KB
None
            bedroom      bathroom  parking_space
count  18201.000000  18195.000000   18189.000000
mean       4.315312      3.124815       3.157458
std        2.445600      2.035028       1.601164
min        1.000000      1.000000       1.000000
25%        2.000000      1.000000       2.000000
50%        4.000000      2.000000       3.000000
75%        6.000000      5.000000       4.000000
max        9.000000      7.000000       6.000000
```

Distribution of bedrooms in the whole dataset

We observe that houses with 5 bedrooms have the most occurence

In [8]:

```
plt.figure(figsize=(20, 12));

# Step 3: Distribution Plots
sns.histplot(data['bedroom'], kde=True)
plt.xlabel('Number of Bedrooms')
plt.ylabel('Frequency')
plt.title('Distribution of Bedrooms')
plt.show()
```

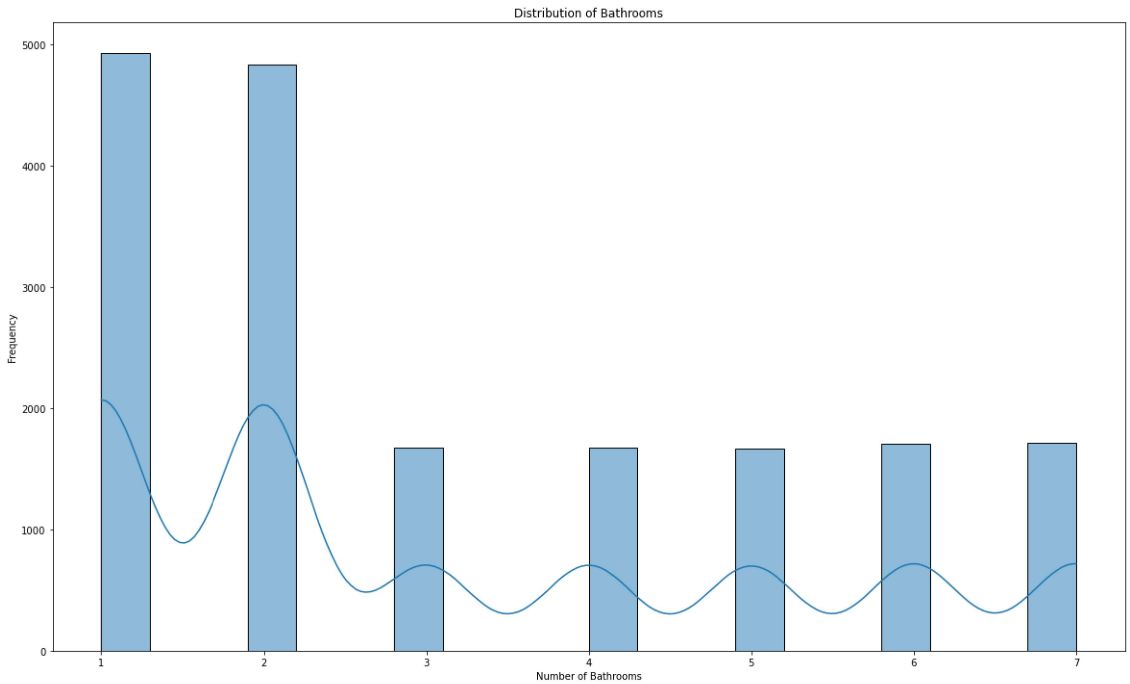


Distribution of Bathrooms in the whole dataset

We Observe that most houses in the dataset have only one bathroom.

In [9]:

```
plt.figure(figsize=(20, 12));
sns.histplot(data['bathroom'], kde=True)
plt.xlabel('Number of Bathrooms')
plt.ylabel('Frequency')
plt.title('Distribution of Bathrooms')
plt.show()
```

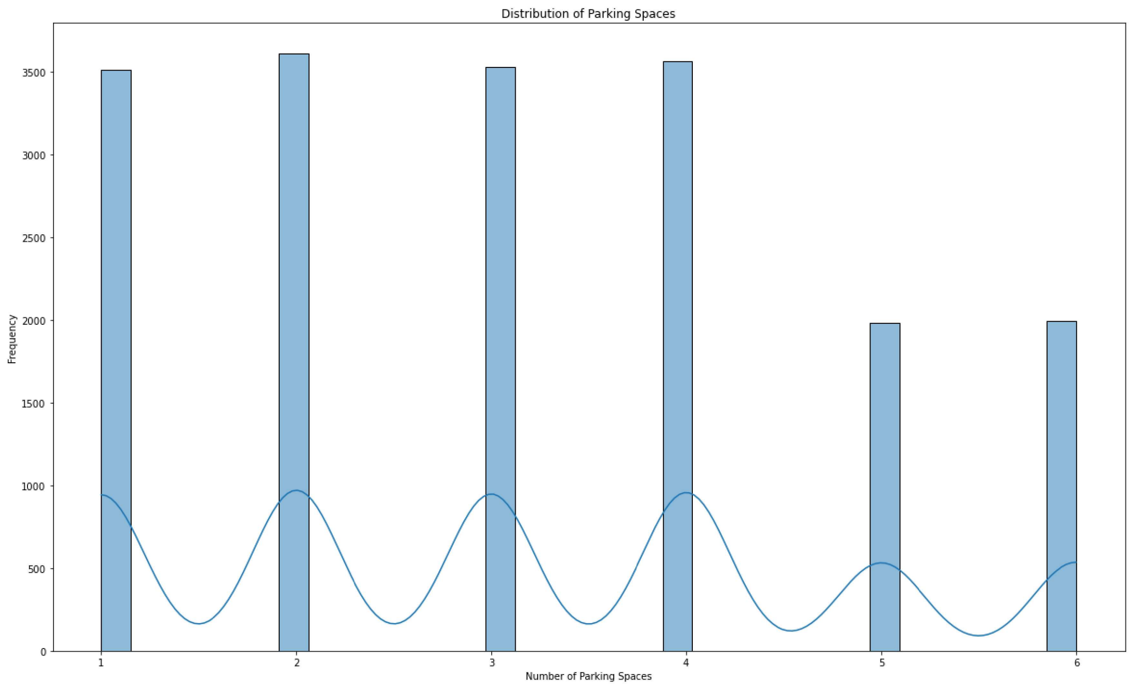


Distribution of Parking Spaces.

Most houses in the dataset have just 2 parking spaces

In [10]:

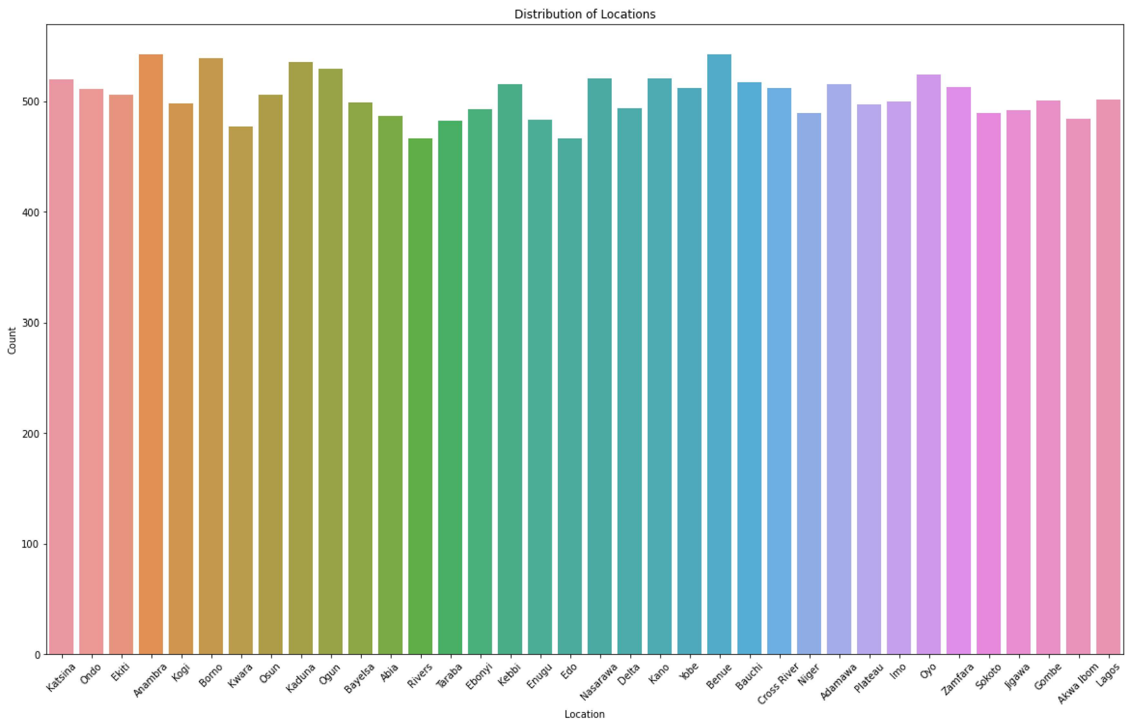
```
plt.figure(figsize=(20, 12));

sns.histplot(data['parking_space'], kde=True)
plt.xlabel('Number of Parking Spaces')
plt.ylabel('Frequency')
plt.title('Distribution of Parking Spaces')
plt.show()
```



Distribution of Location

The Distribution of location in the dataset is also fair.

```
In [11]: # Step 4: Categorical Plots
plt.figure(figsize=(20, 12));
sns.countplot(data['loc'])
plt.xticks(rotation=45)
plt.xlabel('Location')
plt.ylabel('Count')
plt.title('Distribution of Locations')
plt.show()
```

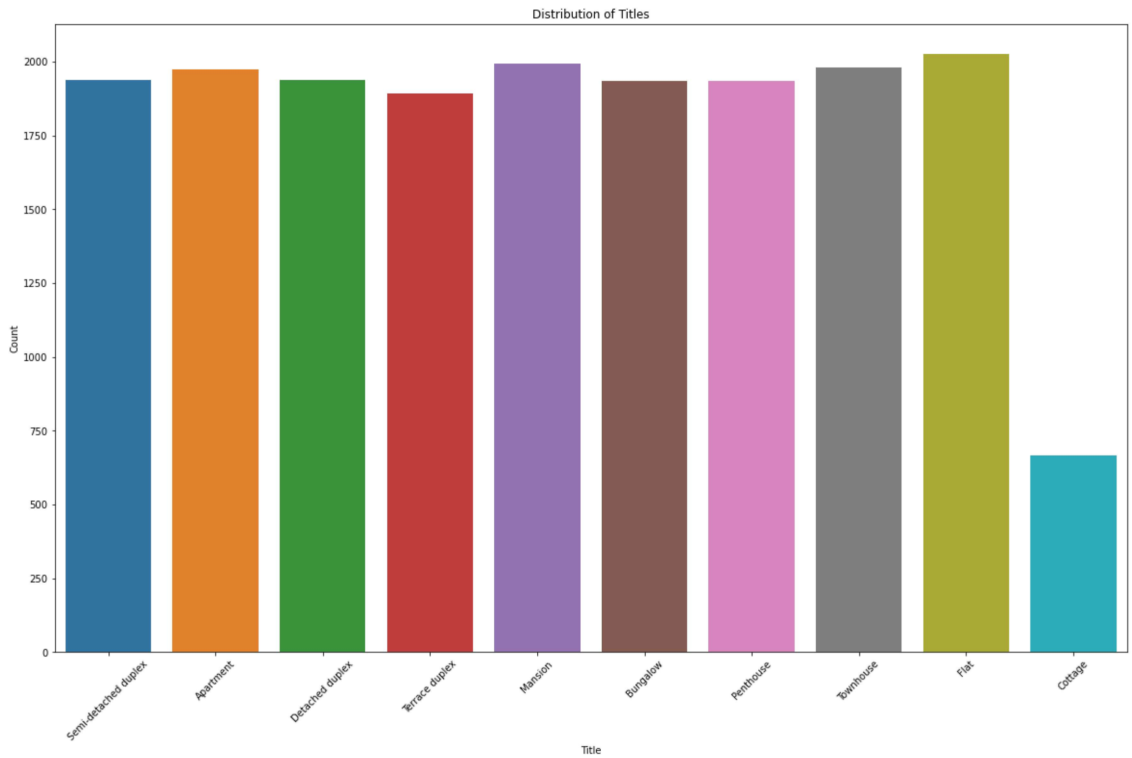


Distribution of House Types

The "cottage" house type in the dataset is quite low compared to the others

In [12]:

```python
plt.figure(figsize=(20, 12));
sns.countplot(data['title'])
plt.xticks(rotation=45)
plt.xlabel('Title')
plt.ylabel('Count')
plt.title('Distribution of Titles')
plt.show()
```

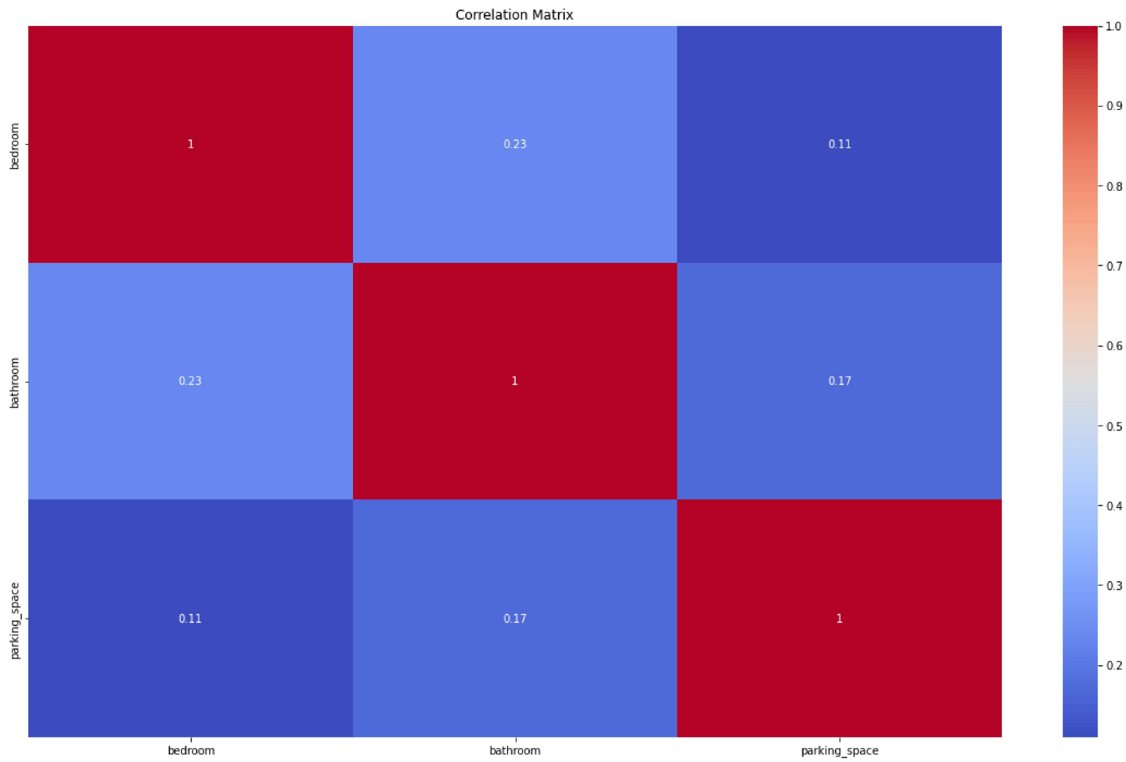


Correlation.

The number of bedrooms and bathrooms seem to be the most correlated of the numerical features

In [13]:

```
# Step 5: Correlation Analysis (if applicable)
plt.figure(figsize=(20, 12));
correlation_matrix = data.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



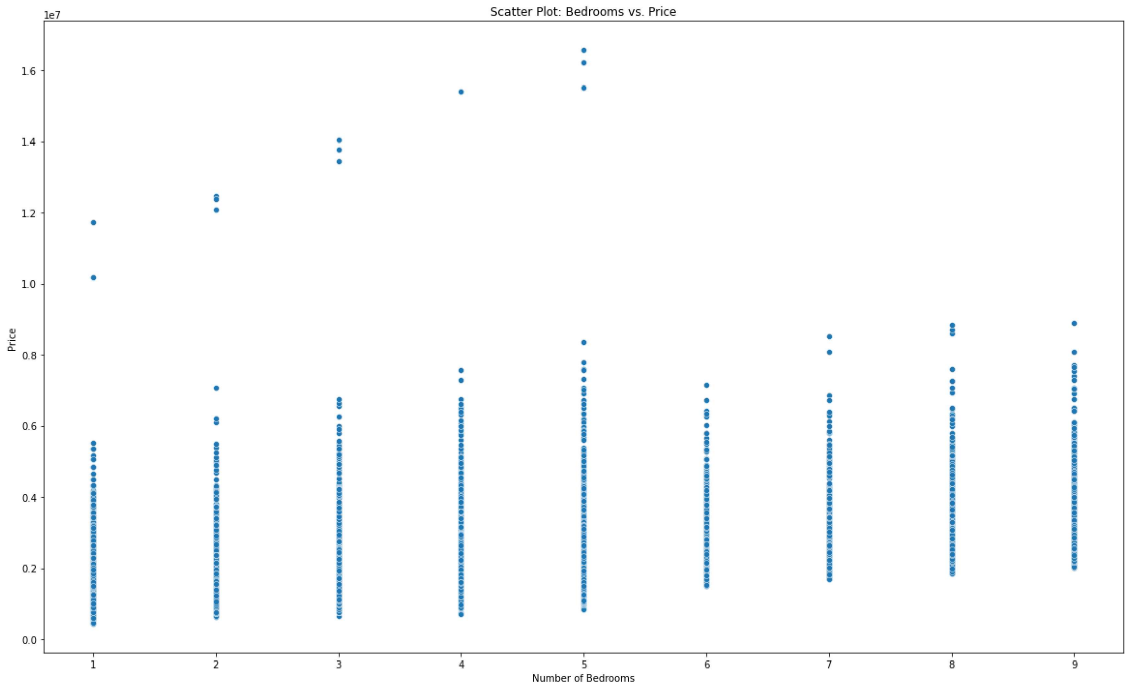
Price Data Exploration

What number of bedrooms have the highest price?

The Scatterplot shows that the houses that have the highest price have 5 bedrooms

In [14]:

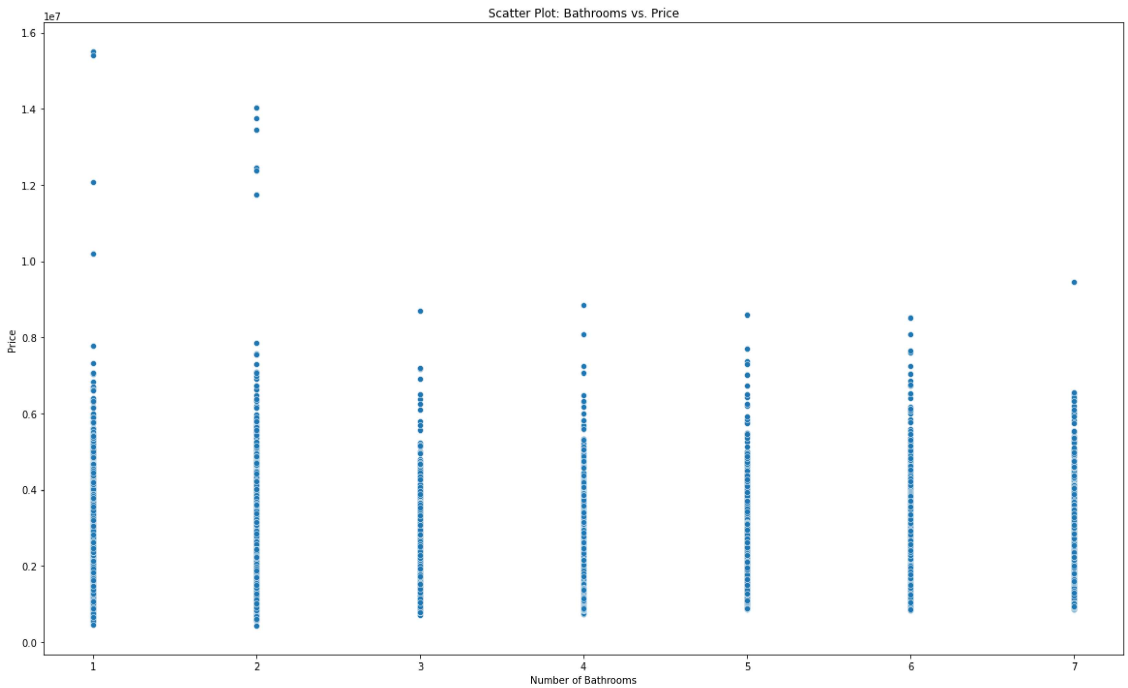
```
# Step 6: Explore Relationships with Price (Scatter Plots)
plt.figure(figsize=(20, 12));
sns.scatterplot(data=train, x='bedroom', y='price')
plt.xlabel('Number of Bedrooms')
plt.ylabel('Price')
plt.title('Scatter Plot: Bedrooms vs. Price')
plt.show()
```



Bedrooms by Prices.

In [15]:

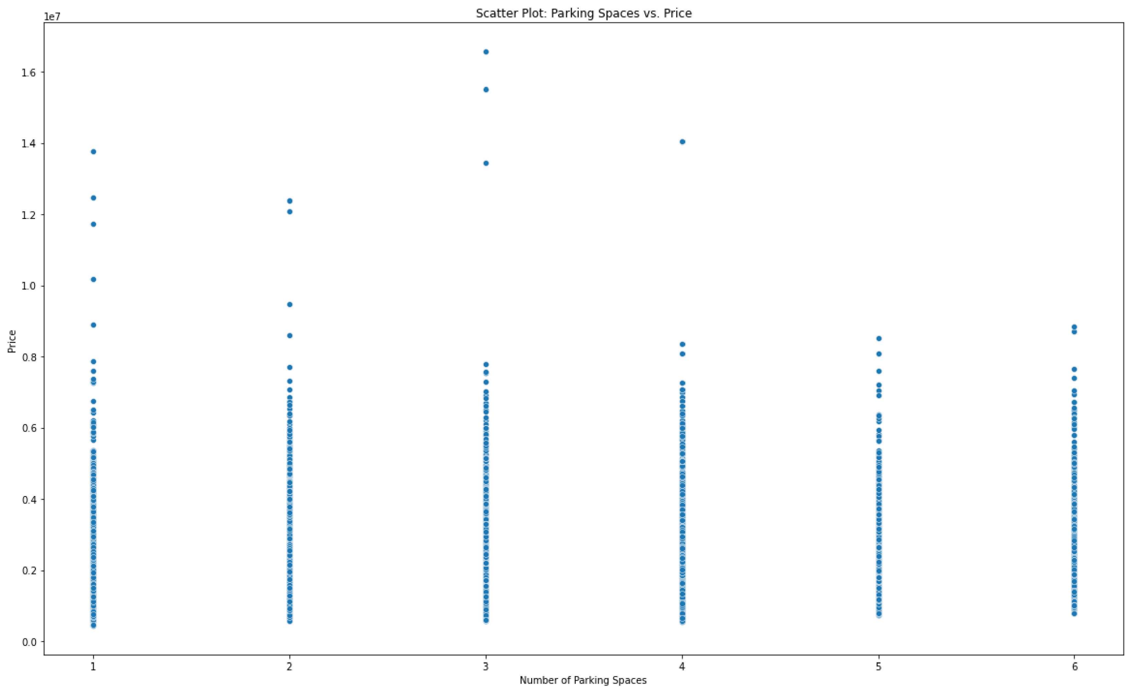
```
plt.figure(figsize=(20, 12));
sns.scatterplot(data=train, x='bathroom', y='price')
plt.xlabel('Number of Bathrooms')
plt.ylabel('Price')
plt.title('Scatter Plot: Bathrooms vs. Price')
plt.show()
```



Parking Spaces by Prices.

In [16]:

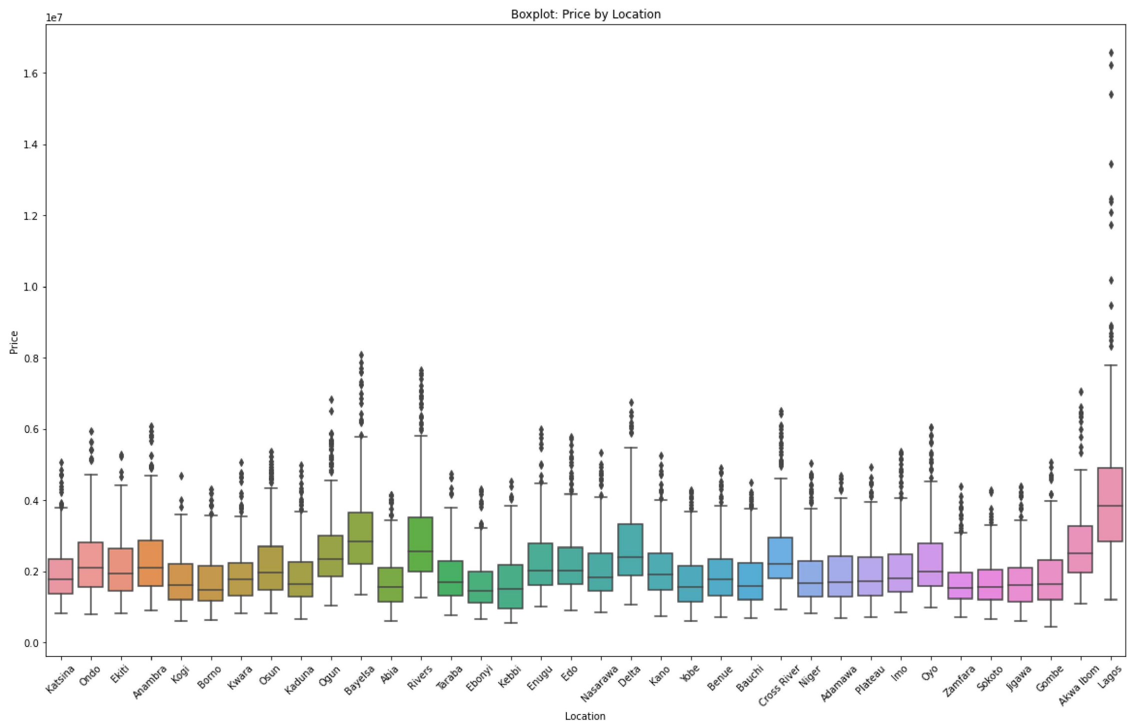
```
plt.figure(figsize=(20, 12));
sns.scatterplot(data=train, x='parking_space', y='price')
plt.xlabel('Number of Parking Spaces')
plt.ylabel('Price')
plt.title('Scatter Plot: Parking Spaces vs. Price')
plt.show()
```



Prices of Houses by Location

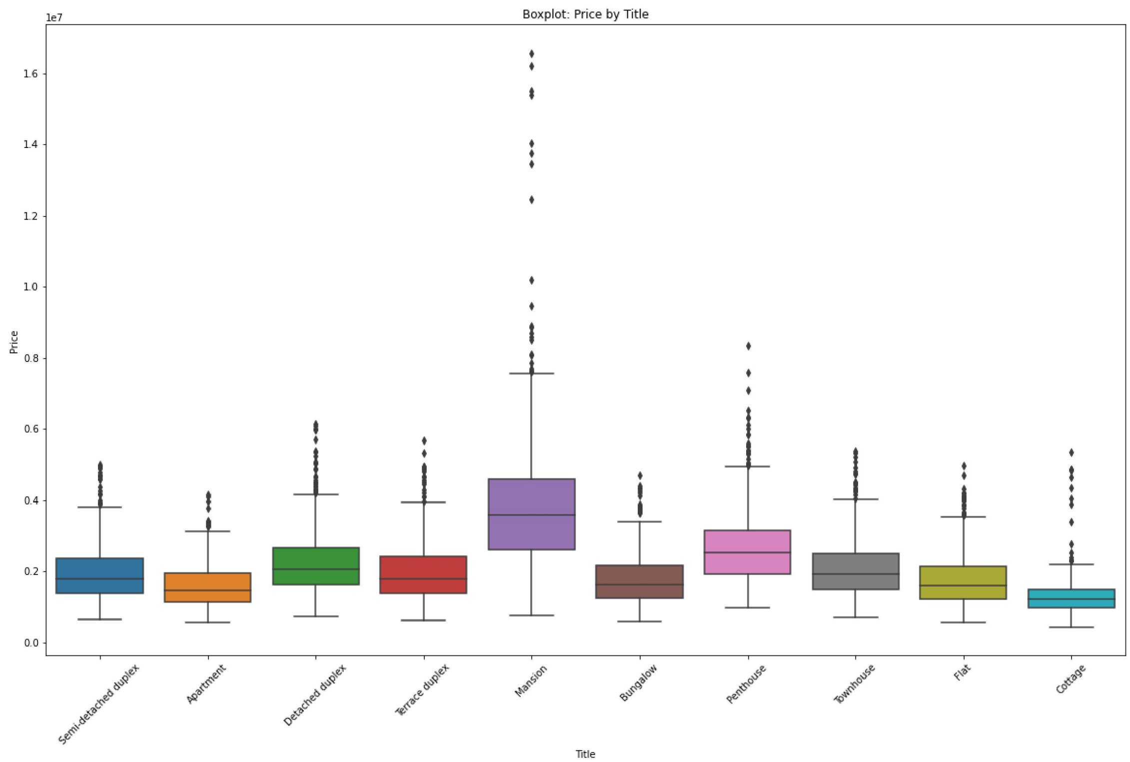
In [17]:

```
plt.figure(figsize=(20, 12));
sns.boxplot(data=train, x='loc', y='price')
plt.xticks(rotation=45)
plt.xlabel('Location')
plt.ylabel('Price')
plt.title('Boxplot: Price by Location')
plt.show()
```



Prices of Houses by House Type

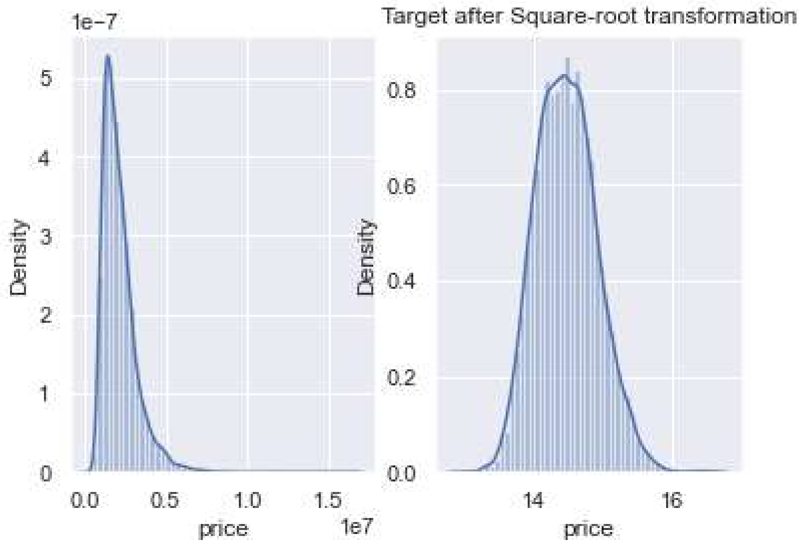
In [18]:
```
plt.figure(figsize=(20, 12));
sns.boxplot(data=train, x='title', y='price')
plt.xticks(rotation=45)
plt.xlabel('Title')
plt.ylabel('Price')
plt.title('Boxplot: Price by Title')
plt.show()
```



Price Disribution

In [19]:

```
sns.set()
y = train.price
y_transformed = pd.Series(np.log(y))

fig, ax = plt.subplots(1, 2)
sns.distplot(y, ax=ax[0])
plt.title("Target after Square-root transformation")
# ax[0].axvline(y_transformed)
sns.distplot(y_transformed, ax=ax[1])
plt.show()
```



In [ ]:

```
```