```
import pandas as pd

# Load the dataset
data = pd.read_csv("/content/AirBNB.csv")

# Display the first few rows of the dataset
data.head()

# Check for missing values
print(data.isnull().sum())

# Fill missing values or drop rows/columns based on the data
data = data.dropna()  # For simplicity, we drop rows with missing values here
```

```
id                          0
room_type                   5
accommodates                3
bathrooms                 203
cancellation_policy         8
cleaning_fee                4
instant_bookable            0
review_scores_rating    16722
bedrooms                   92
beds                      131
log_price                   0
dtype: int64
<ipython-input-3-9fecff44891c>:4: DtypeWarning: Columns (5) have mixed types. Specify dtype option on import or set low_memory=False.
  data = pd.read_csv("/content/AirBNB.csv")
```

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

# Load the dataset
data = pd.read_csv("/content/AirBNB.csv")

# Display the first few rows of the dataset to check column names
print(data.columns)  # Print column names to inspect

# Check for missing values
print(data.isnull().sum())
```

```
Index(['id', 'room_type', 'accommodates', 'bathrooms', 'cancellation_policy',
       'cleaning_fee', 'instant_bookable', 'review_scores_rating', 'bedrooms',
       'beds', 'log_price'],
      dtype='object')
id                          0
room_type                   5
accommodates                3
bathrooms                 203
cancellation_policy         8
cleaning_fee                4
instant_bookable            0
review_scores_rating    16722
bedrooms                   92
beds                      131
log_price                   0
dtype: int64
```

```
<ipython-input-12-7bf5733477e1>:6: DtypeWarning: Columns (5) have mixed types. Specify dtype option on import or set low_memory=False.
  data = pd.read_csv("/content/AirBNB.csv")
```

```
# Assuming 'log_price' is the correct column name for the price
data = data.dropna(subset=['log_price'])  # Keep rows with valid price values
```

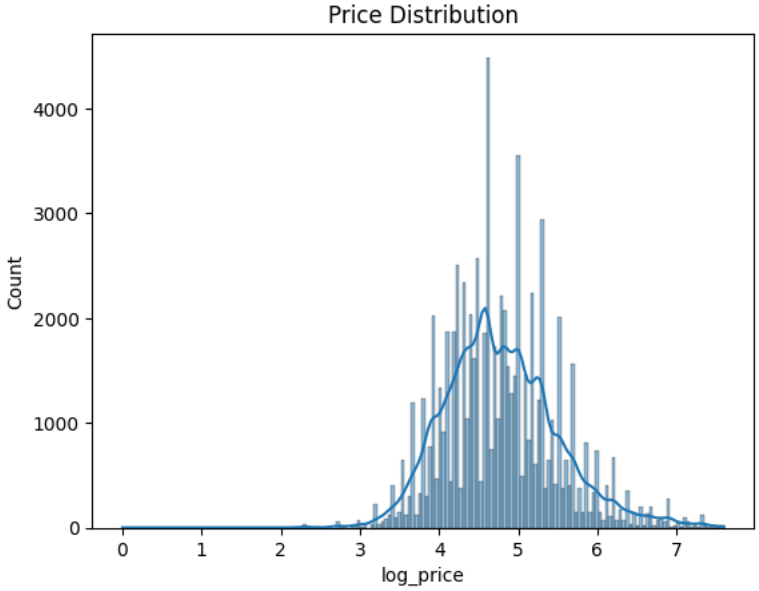
```
print(data.columns)
```

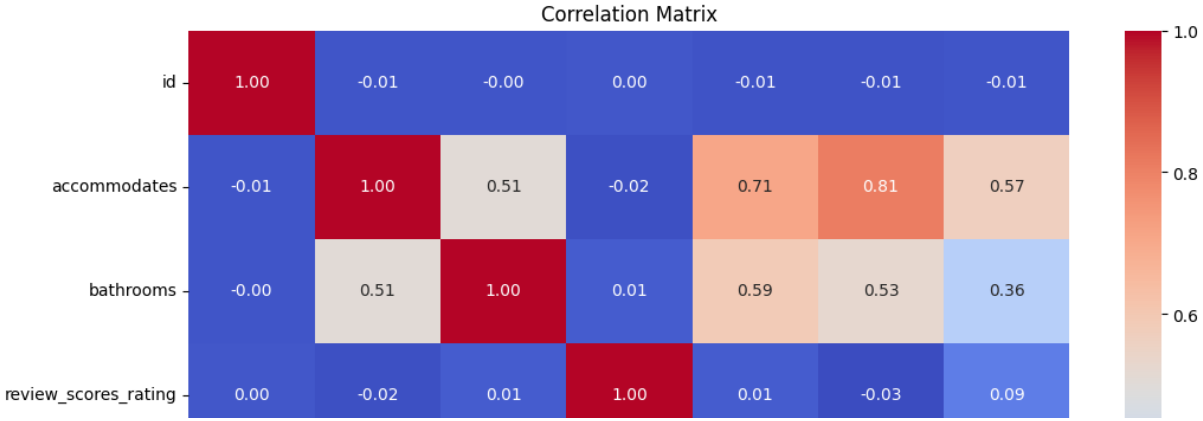
```
Index(['id', 'room_type', 'accommodates', 'bathrooms', 'cancellation_policy',
       'cleaning_fee', 'instant_bookable', 'review_scores_rating', 'bedrooms',
       'beds', 'log_price'],
      dtype='object')
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# Basic statistics of numerical columns
print(data.describe())

# Distribution of the price column
# Replace 'log_price' with the actual name of the price column in your dataset if it's different
sns.histplot(data['log_price'], kde=True)
plt.title('Price Distribution')
plt.show()

# Correlation matrix to see relationships between features and target variable
# Select only numerical features for correlation calculation
numerical_data = data.select_dtypes(include=['number']) # Select numerical columns only

correlation = numerical_data.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(correlation, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix')
plt.show()
```

	id	accommodates	bathrooms	review_scores_rating	\
count	7.411100e+04	74108.000000	73908.000000	57389.000000	
mean	1.126662e+07	3.155125	1.235272	94.067365	
std	6.081735e+06	2.153603	0.582054	7.836556	
min	3.440000e+02	1.000000	0.000000	20.000000	
25%	6.261964e+06	2.000000	1.000000	92.000000	
50%	1.225415e+07	2.000000	1.000000	96.000000	
75%	1.640226e+07	4.000000	1.000000	100.000000	
max	2.123090e+07	16.000000	8.000000	100.000000	

	bedrooms	beds	log_price
count	74019.000000	73980.000000	74111.000000
mean	1.265797	1.710868	4.782069
std	0.852149	1.254142	0.717394
min	0.000000	0.000000	0.000000
25%	1.000000	1.000000	4.317488
50%	1.000000	1.000000	4.709530
75%	1.000000	2.000000	5.220356
max	10.000000	18.000000	7.600402





```
# Convert categorical variables using one-hot encoding
data = pd.get_dummies(data, drop_first=True)

# Check if 'price' column exists, if not, use 'log_price'
if 'price' not in data.columns:
    if 'log_price' in data.columns:  # Assuming 'log_price' is the original price column
        data = data.rename(columns={'log_price': 'price'})  # Rename to 'price'
    else:
        raise KeyError("Neither 'price' nor 'log_price' column found in the DataFrame")

# Separate the target variable (price) and features
X = data.drop('price', axis=1)
y = data['price']

# Scale the features (optional but can improve model performance)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.impute import SimpleImputer  # Import SimpleImputer

# ... (your existing code for data loading and preprocessing) ...

# Simple Linear Regression
X_simple = X[['accommodates']]  # Example: Using 'accommodates' as the feature

# Impute missing values in X_simple using the mean
imputer = SimpleImputer(strategy='mean')  # Create an imputer instance
X_simple = imputer.fit_transform(X_simple)  # Fit and transform to impute NaNs

# Continue with train-test split and model training
X_train, X_test, y_train, y_test = train_test_split(X_simple, y, test_size=0.2, random_state=42)

# Initialize and fit the model
model_simple = LinearRegression()
model_simple.fit(X_train, y_train)

# ... (rest of your code for prediction and evaluation) ...
```

```
▾ LinearRegression
LinearRegression()
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.impute import SimpleImputer  # Import SimpleImputer

# ... (your existing code for data loading and preprocessing) ...
```

```
# Impute missing values in X_scaled using the mean
imputer = SimpleImputer(strategy='mean')  # Create an imputer instance
X_scaled = imputer.fit_transform(X_scaled)  # Fit and transform to impute NaNs

# Multiple Linear Regression
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42

# Initialize and fit the model
model_multiple = LinearRegression()
model_multiple.fit(X_train, y_train)

# Predict the prices
y_pred_multiple = model_multiple.predict(X_test)

# Evaluate the model
mse_multiple = mean_squared_error(y_test, y_pred_multiple)
r2_multiple = r2_score(y_test, y_pred_multiple)
print(f"Multiple Linear Regression MSE: {mse_multiple}")
```

```
Multiple Linear Regression MSE: 0.2470154773286076
Multiple Linear Regression R^2: 0.519168727882675
```