Data Exploration and Preprocessing:- Analyze the dataset for trends, missing values, and outliers. Perform data cleaning, feature engineering, and transformations.

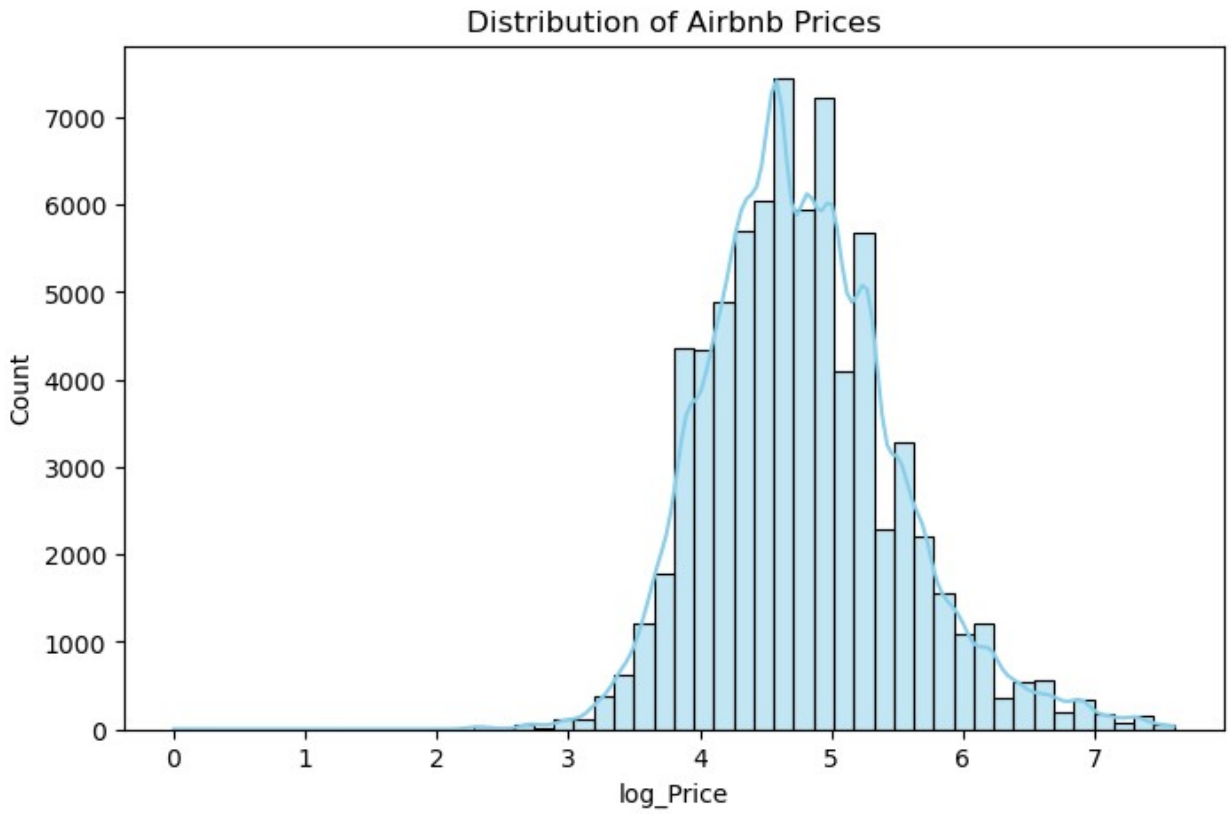
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split

# Load Dataset
df = pd.read_csv(r"C:\Users\Pravallika\Downloads\Airbnbdata.csv")

# Quick Overview
print("Dataset Shape:", df.shape)
print("\nDataset Info:")
print(df.info())
print("\nMissing Values:\n", df.isnull().sum())
```

```
Dataset Shape: (74111, 29)

Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 74111 entries, 0 to 74110
Data columns (total 29 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   id                      74111 non-null  int64
 1   log_price               74111 non-null  float64
 2   property_type           74111 non-null  object
 3   room_type               74111 non-null  object
 4   amenities               74111 non-null  object
 5   accommodates            74111 non-null  int64
 6   bathrooms               73911 non-null  float64
 7   bed_type                74111 non-null  object
 8   cancellation_policy     74111 non-null  object
 9   cleaning_fee            74111 non-null  bool
 10  city                    74111 non-null  object
 11  description             74111 non-null  object
 12  first_review            58247 non-null  object
 13  host_has_profile_pic    73923 non-null  object
 14  host_identity_verified  73923 non-null  object
 15  host_response_rate      55812 non-null  object
 16  host_since              73923 non-null  object
 17  instant_bookable        74111 non-null  object
 18  last_review             58284 non-null  object
 19  latitude                74111 non-null  float64
 20  longitude               74111 non-null  float64
 21  name                    74111 non-null  object
 22  neighbourhood           67239 non-null  object
```

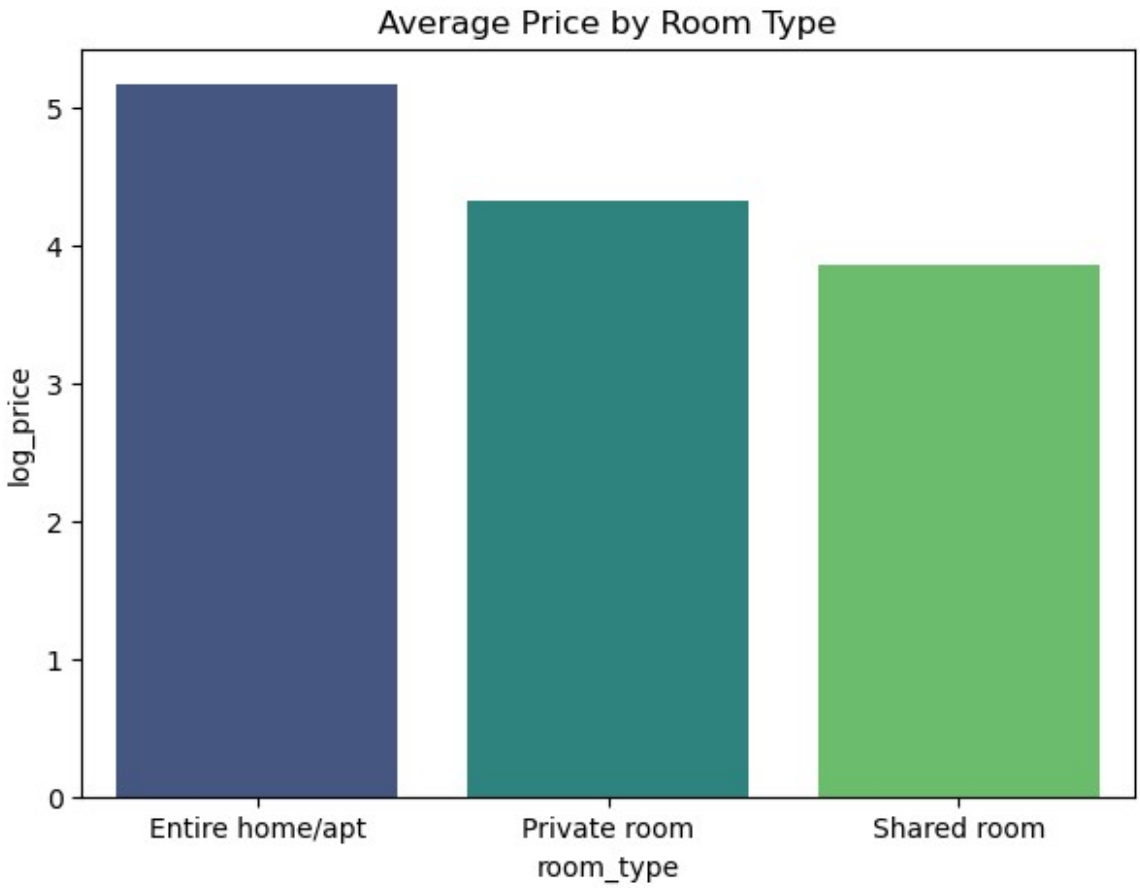
```
 23  number_of_reviews       74111 non-null  int64
 24  review_scores_rating    57389 non-null  float64
 25  thumbnail_url           65895 non-null  object
 26  zipcode                 73143 non-null  object
 27  bedrooms                74020 non-null  float64
 28  beds                    73980 non-null  float64
dtypes: bool(1), float64(7), int64(3), object(18)
memory usage: 15.9+ MB
None

Missing Values:
 id                            0
log_price                      0
property_type                  0
room_type                      0
amenities                      0
accommodates                   0
bathrooms                    200
bed_type                       0
cancellation_policy            0
cleaning_fee                   0
city                           0
description                    0
first_review               15864
host_has_profile_pic         188
host_identity_verified       188
host_response_rate         18299
host_since                   188
instant_bookable               0
last_review                15827
latitude                       0
longitude                      0
name                           0
neighbourhood               6872
number_of_reviews              0
review_scores_rating       16722
thumbnail_url               8216
zipcode                      968
bedrooms                      91
beds                         131
dtype: int64
```

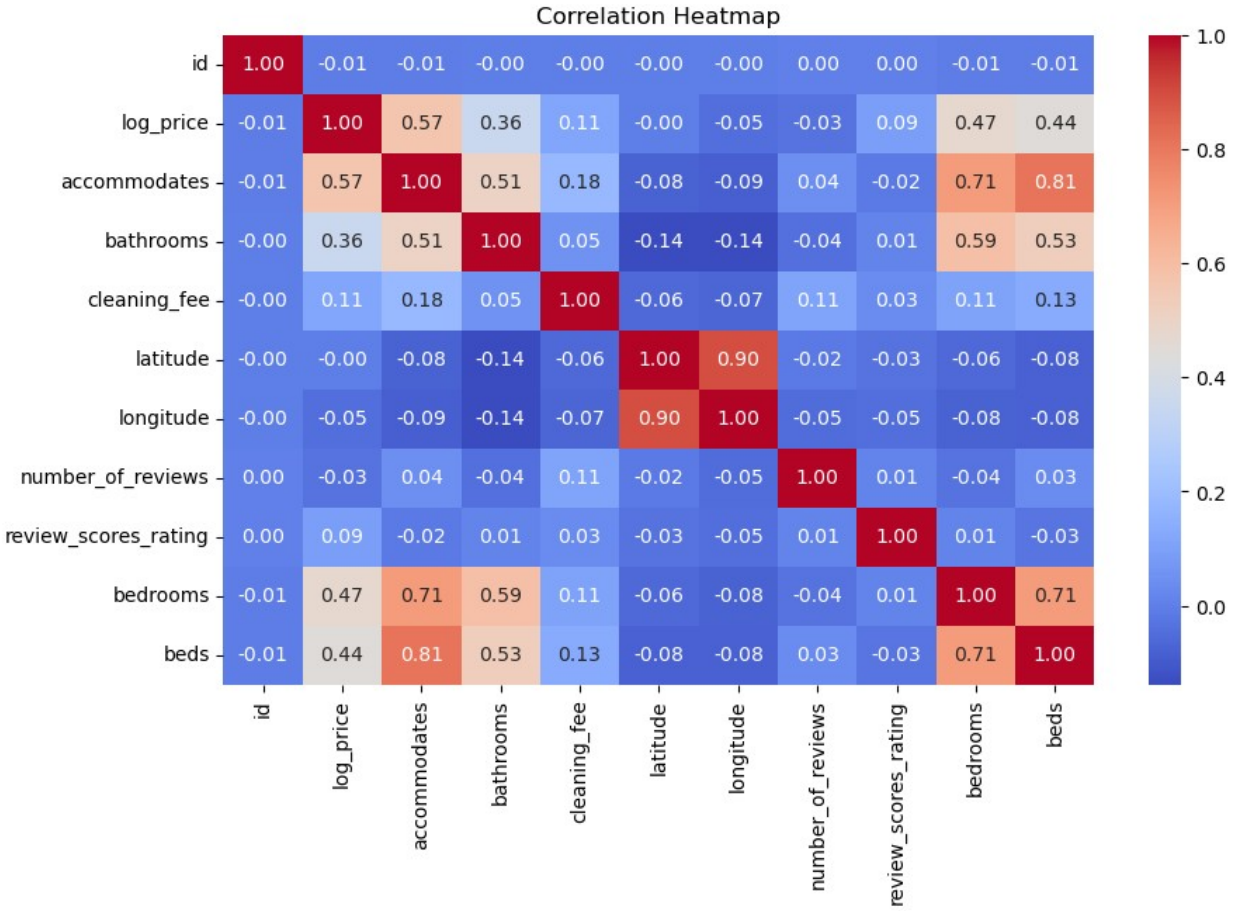
```python
# Analyze Trends
# Price Distribution

plt.figure(figsize=(8,5))
sns.histplot(df['log_price'], bins=50, kde=True, color='skyblue')
plt.title("Distribution of Airbnb Prices")
plt.xlabel("log_Price")
plt.ylabel("Count")
```

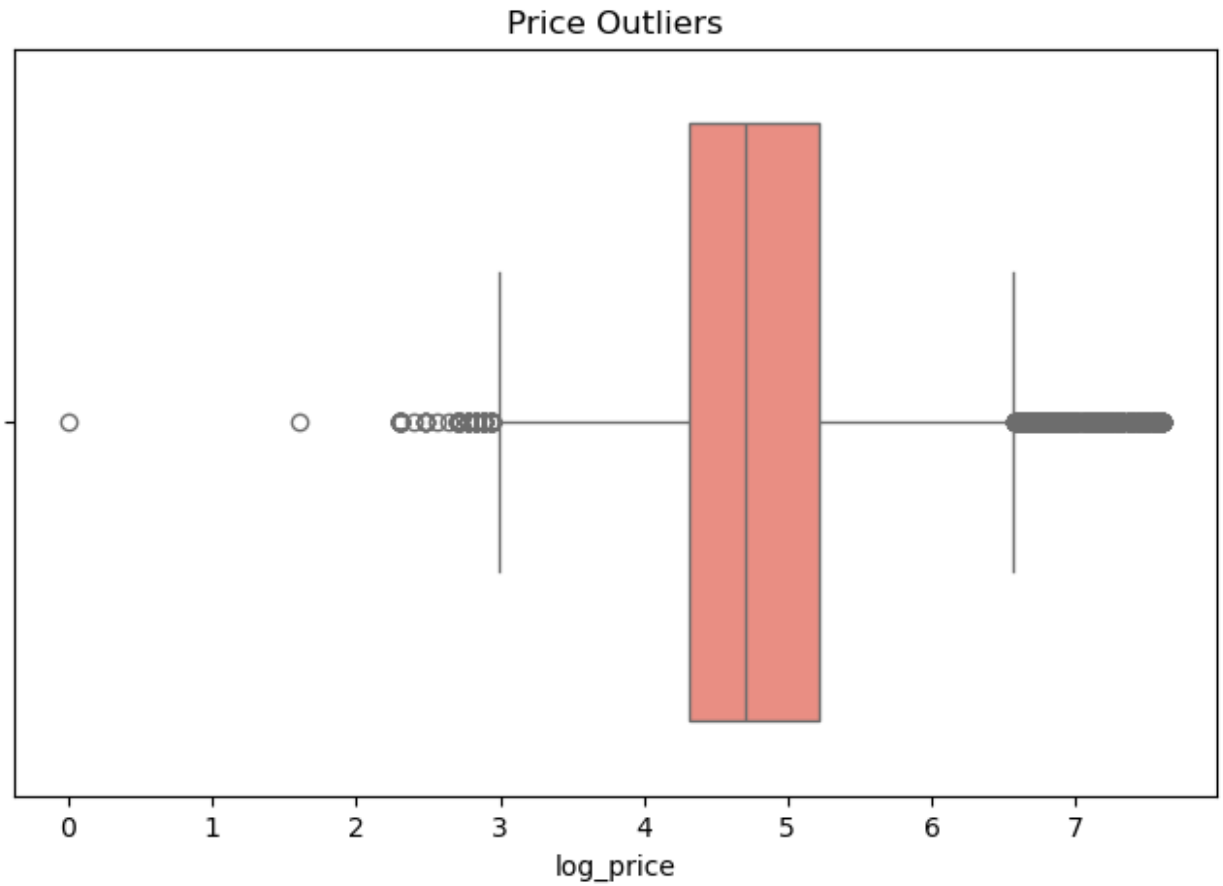
```
plt.show()

# Average Price by Room Type
if 'room_type' in df.columns:
    plt.figure(figsize=(7,5))
    sns.barplot(x='room_type', y='log_price', data=df,
estimator='mean', errorbar=None, palette='viridis')
    plt.title("Average Price by Room Type")
    plt.show()


# Correlation Heatmap
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm',
fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()

# Check for Outliers

plt.figure(figsize=(8,5))
sns.boxplot(x=df['log_price'], color='salmon')
plt.title("Price Outliers")
plt.show()

Q1 = df['log_price'].quantile(0.25)
Q3 = df['log_price'].quantile(0.75)
IQR = Q3 - Q1
upper = Q3 + 1.5 * IQR
lower = Q1 - 1.5 * IQR
print(f"Outlier Threshold: {lower:.2f} to {upper:.2f}")
```




```
C:\Users\Pravallika\AppData\Local\Temp\
ipykernel_23216\1686290662.py:14: 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.barplot(x='room_type', y='log_price', data=df, estimator='mean',
errorbar=None, palette='viridis')
```


Average Price by Room Type
log_price
0
1
2
3
4
5
Entire home/apt
Private room
Shared room
room_type


Correlation Heatmap
id 1.00 -0.01 -0.01 -0.00 -0.00 -0.00 -0.00 0.00 0.00 -0.01 -0.01
log_price -0.01 1.00 0.57 0.36 0.11 -0.00 -0.05 -0.03 0.09 0.47 0.44
accommodates -0.01 0.57 1.00 0.51 0.18 -0.08 -0.09 0.04 -0.02 0.71 0.81
bathrooms -0.00 0.36 0.51 1.00 0.05 -0.14 -0.14 -0.04 0.01 0.59 0.53
cleaning_fee -0.00 0.11 0.18 0.05 1.00 -0.06 -0.07 0.11 0.03 0.11 0.13
latitude -0.00 -0.00 -0.08 -0.14 -0.06 1.00 0.90 -0.02 -0.03 -0.06 -0.08
longitude -0.00 -0.05 -0.09 -0.14 -0.07 0.90 1.00 -0.05 -0.05 -0.08 -0.08
number_of_reviews 0.00 -0.03 0.04 -0.04 0.11 -0.02 -0.05 1.00 0.01 -0.04 0.03
review_scores_rating 0.00 0.09 -0.02 0.01 0.03 -0.03 -0.05 0.01 1.00 0.01 -0.03
bedrooms -0.01 0.47 0.71 0.59 0.11 -0.06 -0.08 -0.04 0.01 1.00 0.71
beds -0.01 0.44 0.81 0.53 0.13 -0.08 -0.08 0.03 -0.03 0.71 1.00
id
log_price
accommodates
bathrooms
cleaning_fee
latitude
longitude
number_of_reviews
review_scores_rating
bedrooms
beds
1.0
0.8
0.6
0.4
0.2
0.0




```
Outlier Threshold: 2.96 to 6.57
```

```
# Handle Missing Values
df['review_scores_rating'] = df['review_scores_rating'].fillna(0)
df = df.dropna(subset=['log_price'])

# Remove Outliers

df = df[(df['log_price'] >= lower) & (df['log_price'] <= upper)]
print("Data shape after removing outliers:", df.shape)
```

```
Data shape after removing outliers: (72579, 29)
```

```
# Feature Engineering

if 'amenities' in df.columns:
    df['num_amenities'] = df['amenities'].apply(lambda x:
len(str(x).split(',')))

# Encode categorical variables

categorical_cols = df.select_dtypes(include='object').columns
le = LabelEncoder()
```

```
for col in categorical_cols:
    df[col] = le.fit_transform(df[col].astype(str))

# Feature Transformation

scaler = StandardScaler()
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns
df[numeric_cols] = scaler.fit_transform(df[numeric_cols])

print("□ Data Preprocessing Complete")
```

```
□ Data Preprocessing Complete
```

Model Development:- Build a regression model to predict listing prices.

```
# Split Data into Train, Validation, and Test Sets

X = df.drop('log_price', axis=1)
y = df['log_price']

X_train, X_temp, y_train, y_temp = train_test_split(X, y,
test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp,
test_size=0.5, random_state=42)

print(f"Train: {X_train.shape}, Validation: {X_val.shape}, Test:
{X_test.shape}")

# Impute missing values using the mean of the training set'

for col in X_train.columns:
    if X_train[col].isnull().any():
        mean_val = X_train[col].mean()
        X_train[col] = X_train[col].fillna(mean_val)
        X_val[col] = X_val[col].fillna(mean_val)
        X_test[col] = X_test[col].fillna(mean_val)


# Train Multiple Regression Models

from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor

lr = LinearRegression()
rf = RandomForestRegressor(random_state=42)
xgb = XGBRegressor(random_state=42)

lr.fit(X_train, y_train)
rf.fit(X_train, y_train)
```

```
xgb.fit(X_train, y_train)

print("□ Model Training Completed")
```

```
Train: (50805, 29), Validation: (10887, 29), Test: (10887, 29)
□ Model Training Completed
```

Model Evaluation Evaluate the model's performance using appropriate metrics like RMSE, MAE, and R^2.

```
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score

def evaluate_model(model, X, y):
    preds = model.predict(X)
    rmse = np.sqrt(mean_squared_error(y, preds))
    mae = mean_absolute_error(y, preds)
    r2 = r2_score(y, preds)
    return rmse, mae, r2

models = {'Linear Regression': lr, 'Random Forest': rf, 'XGBoost':
xgb}

print("\n□ Model Evaluation Results:")
for name, model in models.items():
    rmse, mae, r2 = evaluate_model(model, X_test, y_test)
    print(f"\n{name}:")
    print(f"RMSE: {rmse:.3f}")
    print(f"MAE:  {mae:.3f}")
    print(f"R²:   {r2:.3f}")

# Visual comparison

results = pd.DataFrame({
    'Model': ['Linear Regression', 'Random Forest', 'XGBoost'],
    'RMSE': [evaluate_model(lr, X_test, y_test)[0],
             evaluate_model(rf, X_test, y_test)[0],
             evaluate_model(xgb, X_test, y_test)[0]],
    'R2 Score': [evaluate_model(lr, X_test, y_test)[2],
                 evaluate_model(rf, X_test, y_test)[2],
                 evaluate_model(xgb, X_test, y_test)[2]]
})

plt.figure(figsize=(8,5))
sns.barplot(x='Model', y='R2 Score', data=results, palette='magma')
plt.title("Model Comparison by R² Score")
plt.show()
```

```
□ Model Evaluation Results:
```

```
Linear Regression:
RMSE: 0.692
MAE:  0.537
R²:   0.524

Random Forest:
RMSE: 0.555
MAE:  0.417
R²:   0.694

XGBoost:
RMSE: 0.542
MAE:  0.408
R²:   0.708

C:\Users\Pravallika\AppData\Local\Temp\
ipykernel_23216\834269041.py:32: 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.barplot(x='Model', y='R2 Score', data=results, palette='magma')
```