

✓ Yulu

Define the Problem Statement and Perform Exploratory Data Analysis (EDA)



Problem Statement Identify which variables significantly predict the demand for shared electric cycles in the Indian market. Determine how well those variables describe the electric cycle demands.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import ttest_ind, f_oneway, chi2_contingency

# Load the dataset
yulu_data = pd.read_csv('yulu_data.csv')

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



	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16	
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40	
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32	
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13	
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1	

Next steps:

[View recommended plots](#)

Examine Dataset Structure

```
# Shape of the dataset
print(yulu_data.shape)

# Info about the dataset
print(yulu_data.info())

# Statistical summary
print(yulu_data.describe())
```

```
➞ (10886, 12)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
#   Column          Non-Null Count  Dtype
---  -
0   datetime        10886 non-null  object
1   season          10886 non-null  int64
2   holiday         10886 non-null  int64
3   workingday      10886 non-null  int64
4   weather         10886 non-null  int64
5   temp            10886 non-null  float64
6   atemp           10886 non-null  float64
7   humidity        10886 non-null  int64
8   windspeed       10886 non-null  float64
9   casual          10886 non-null  int64
10  registered      10886 non-null  int64
11  count           10886 non-null  int64
dtypes: float64(3), int64(8), object(1)
memory usage: 1020.7+ KB
None
```

	season	holiday	workingday	weather	temp
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000

mean	2.506614	0.028569	0.680875	1.418427	20.23086
std	1.116174	0.166599	0.466159	0.633839	7.79159
min	1.000000	0.000000	0.000000	1.000000	0.82000
25%	2.000000	0.000000	0.000000	1.000000	13.94000
50%	3.000000	0.000000	1.000000	1.000000	20.50000
75%	4.000000	0.000000	1.000000	2.000000	26.24000
max	4.000000	1.000000	1.000000	4.000000	41.00000

	count	atemp	humidity	windspeed	casual	registered \
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000
mean		23.655084	61.886460	12.799395	36.021955	155.552177
std		8.474601	19.245033	8.164537	49.960477	151.039033
min		0.760000	0.000000	0.000000	0.000000	0.000000
25%		16.665000	47.000000	7.001500	4.000000	36.000000
50%		24.240000	62.000000	12.998000	17.000000	118.000000
75%		31.060000	77.000000	16.997900	49.000000	222.000000
max		45.455000	100.000000	56.996900	367.000000	886.000000

	count
count	10886.000000
mean	191.574132
std	181.144454
min	1.000000
25%	42.000000
50%	145.000000
75%	284.000000
max	977.000000

Identify Missing Values

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

```

→ datetime    0
  season       0
  holiday      0
  workingday   0
  weather      0

```

```
temp      0
atemp     0
humidity  0
windspeed 0
casual    0
registered 0
count     0
dtype: int64
```

Remove Duplicates

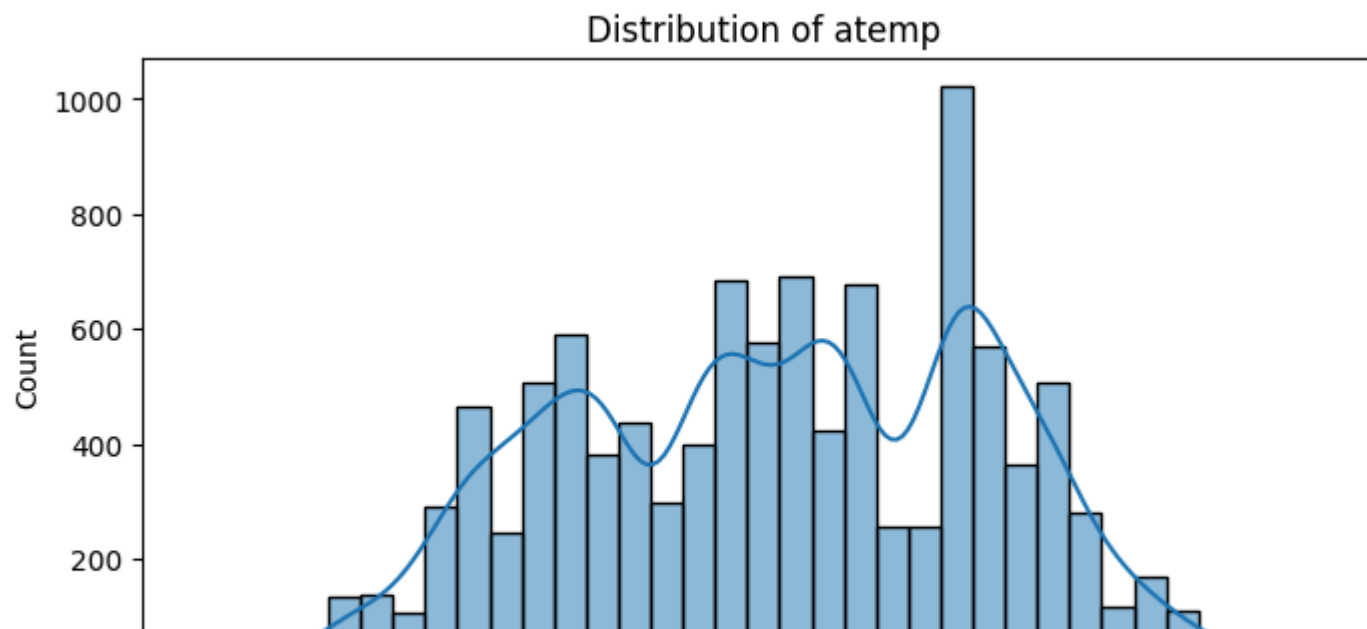
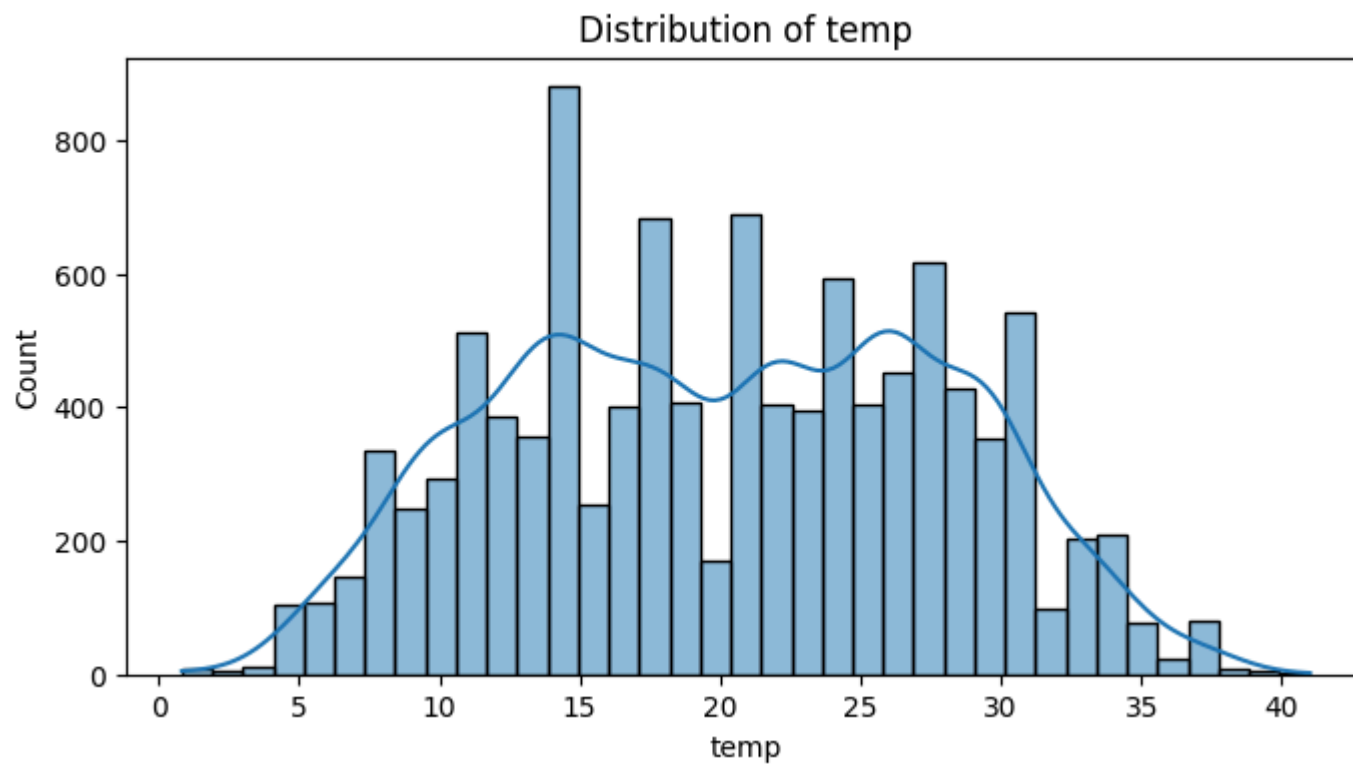
```
# Check for duplicates
duplicates = yulu_data.duplicated().sum()
print(f'Duplicates: {duplicates}')

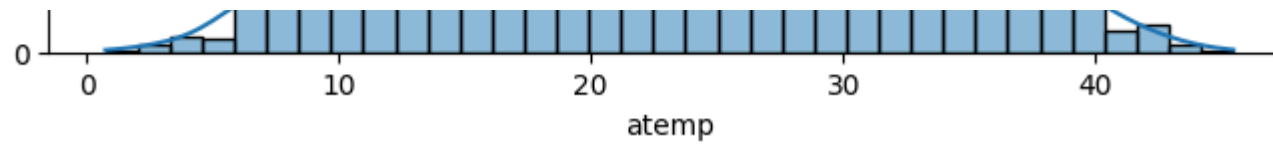
# Drop duplicates if any
yulu_data.drop_duplicates(inplace=True)
```

↔ Duplicates: 0

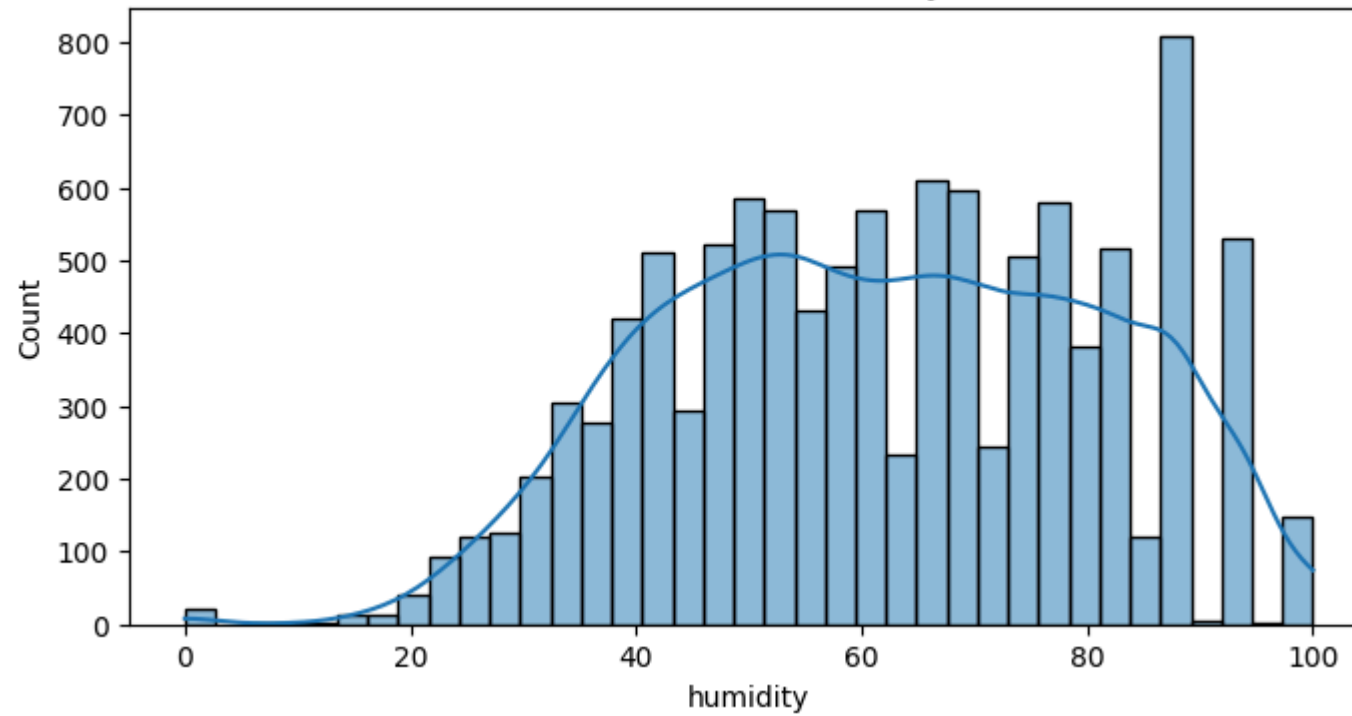
Univariate Analysis

```
# Plot histograms for numerical features
numerical_features = ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count']
for feature in numerical_features:
    plt.figure(figsize=(8, 4))
    sns.histplot(yulu_data[feature], kde=True)
    plt.title(f'Distribution of {feature}')
    plt.show()
```

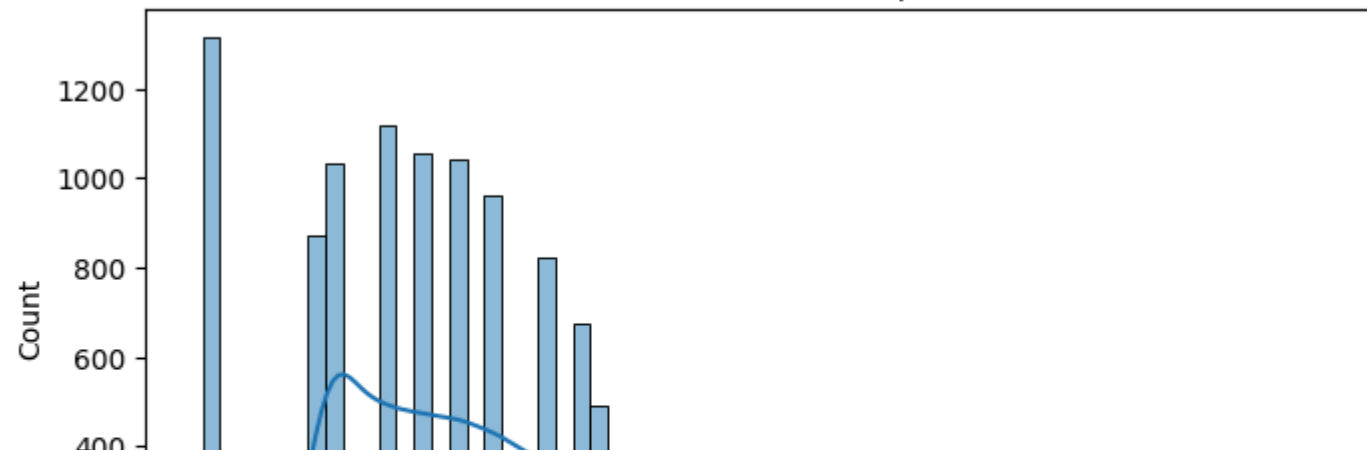


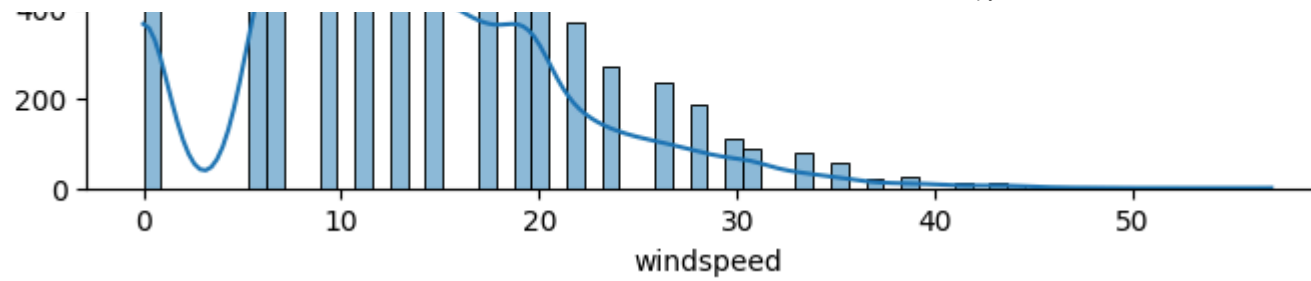


Distribution of humidity

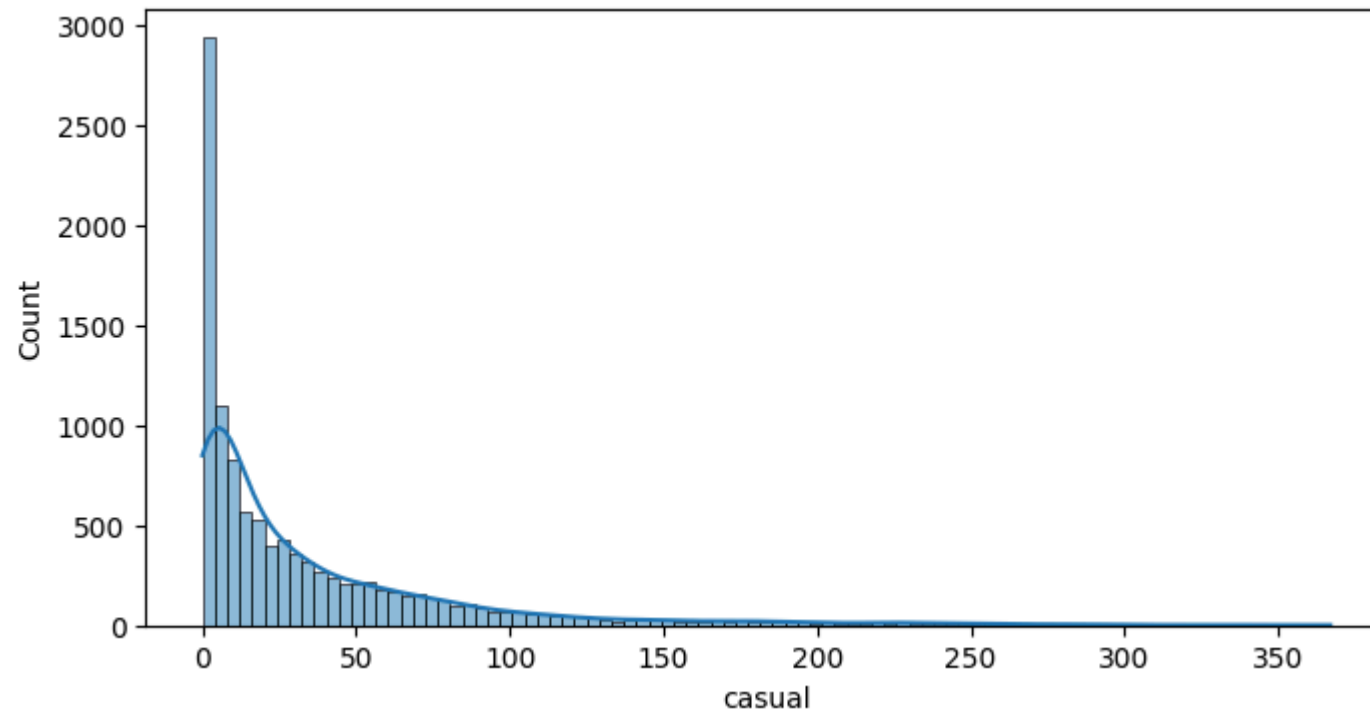


Distribution of windspeed



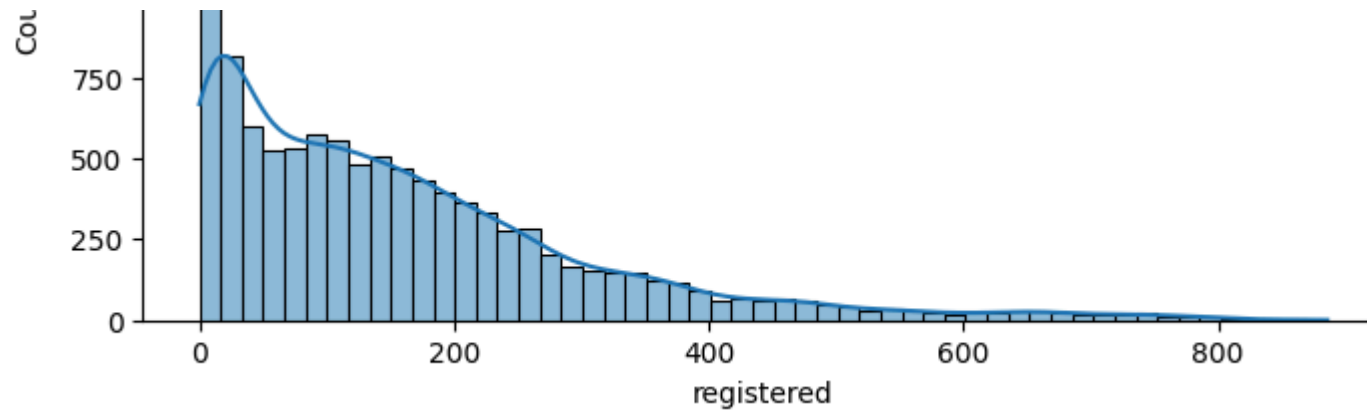


Distribution of casual

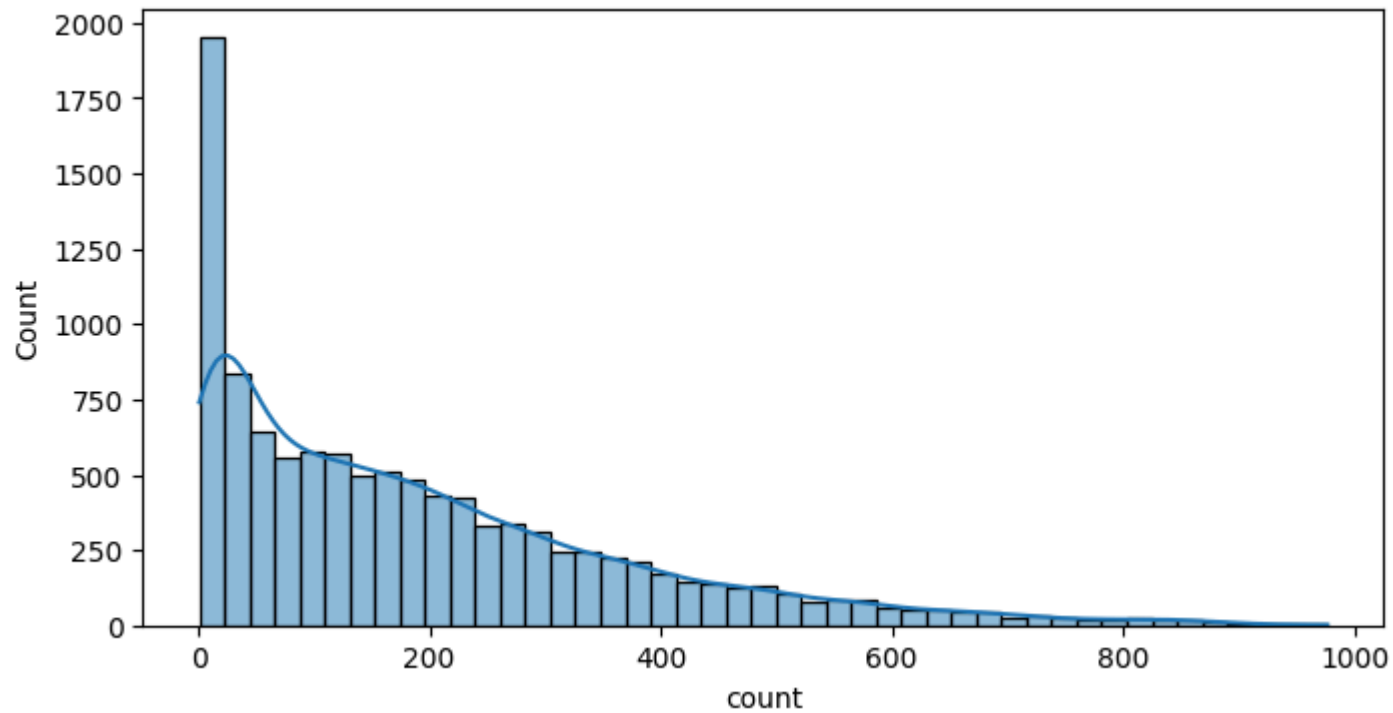


Distribution of registered

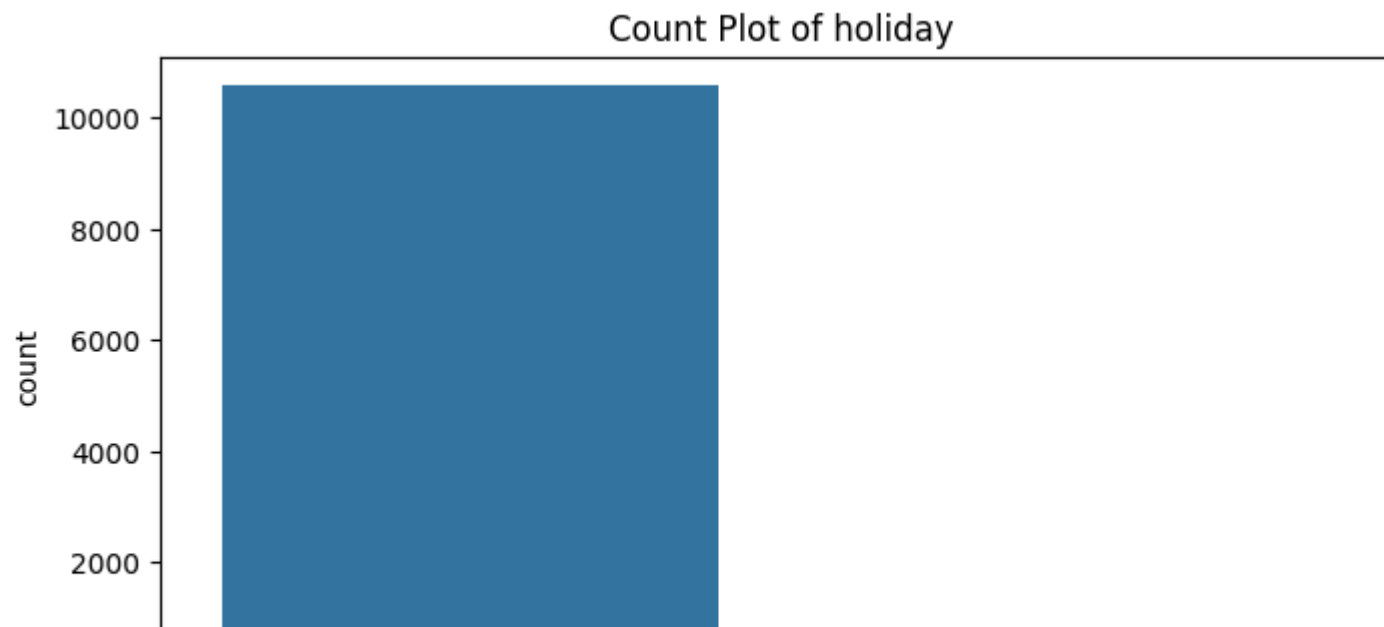
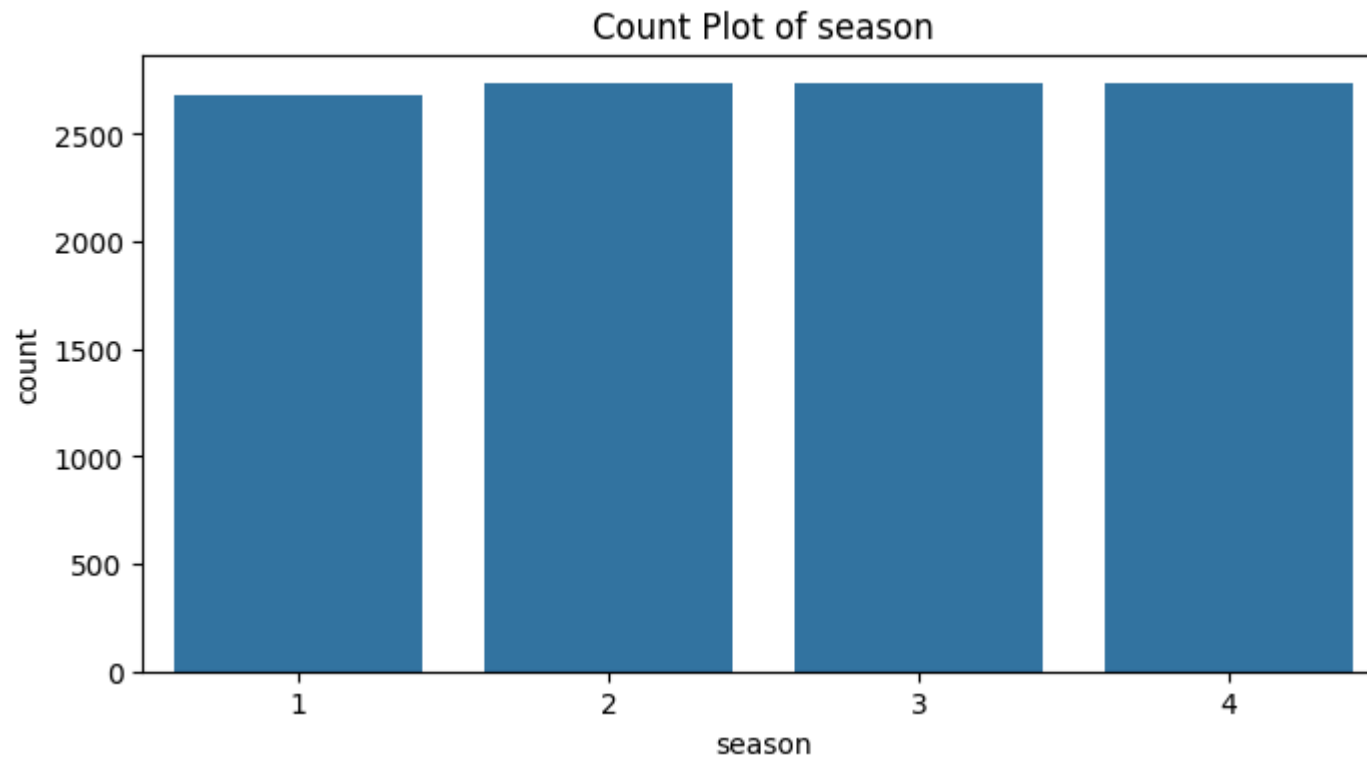




Distribution of count

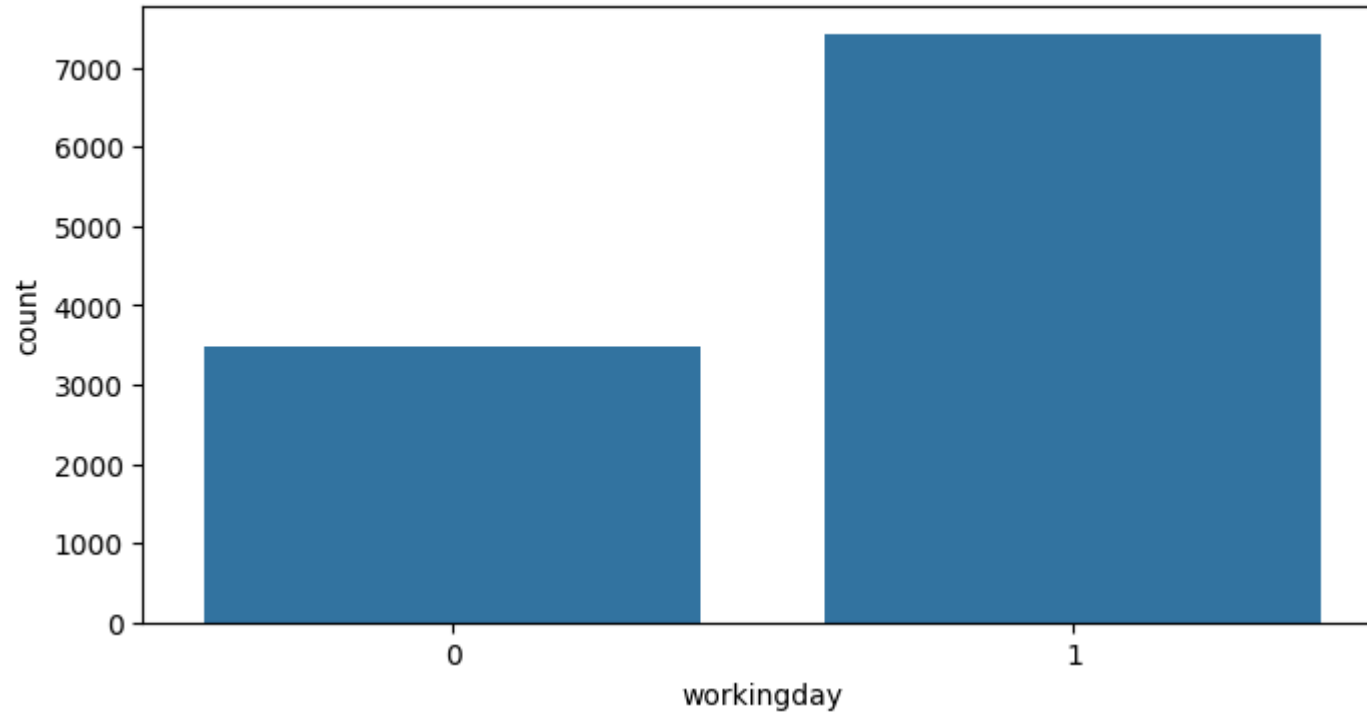



```
# Plot count plots for categorical features
categorical_features = ['season', 'holiday', 'workingday', 'weather']
for feature in categorical_features:
    plt.figure(figsize=(8, 4))
    sns.countplot(x=yulu_data[feature])
    plt.title(f'Count Plot of {feature}')
    plt.show()
```

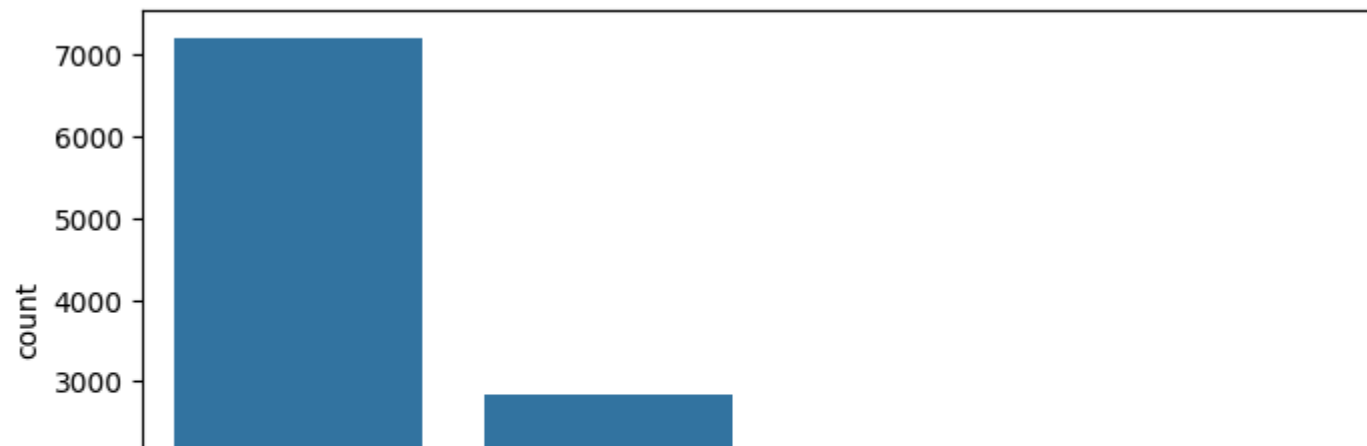


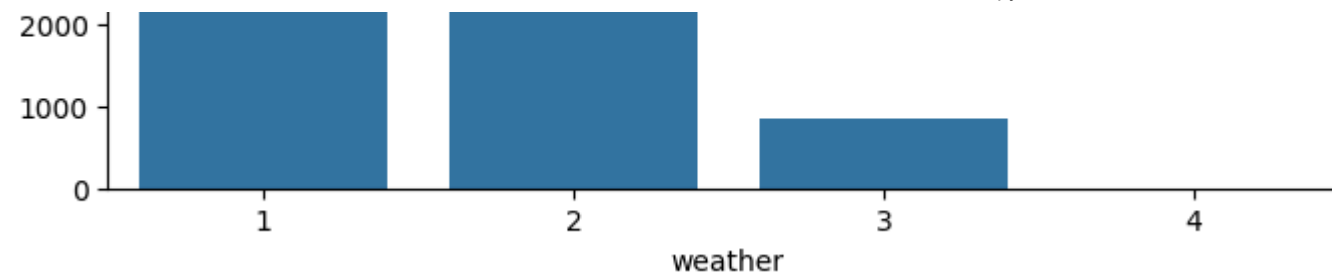


Count Plot of workingday



Count Plot of weather



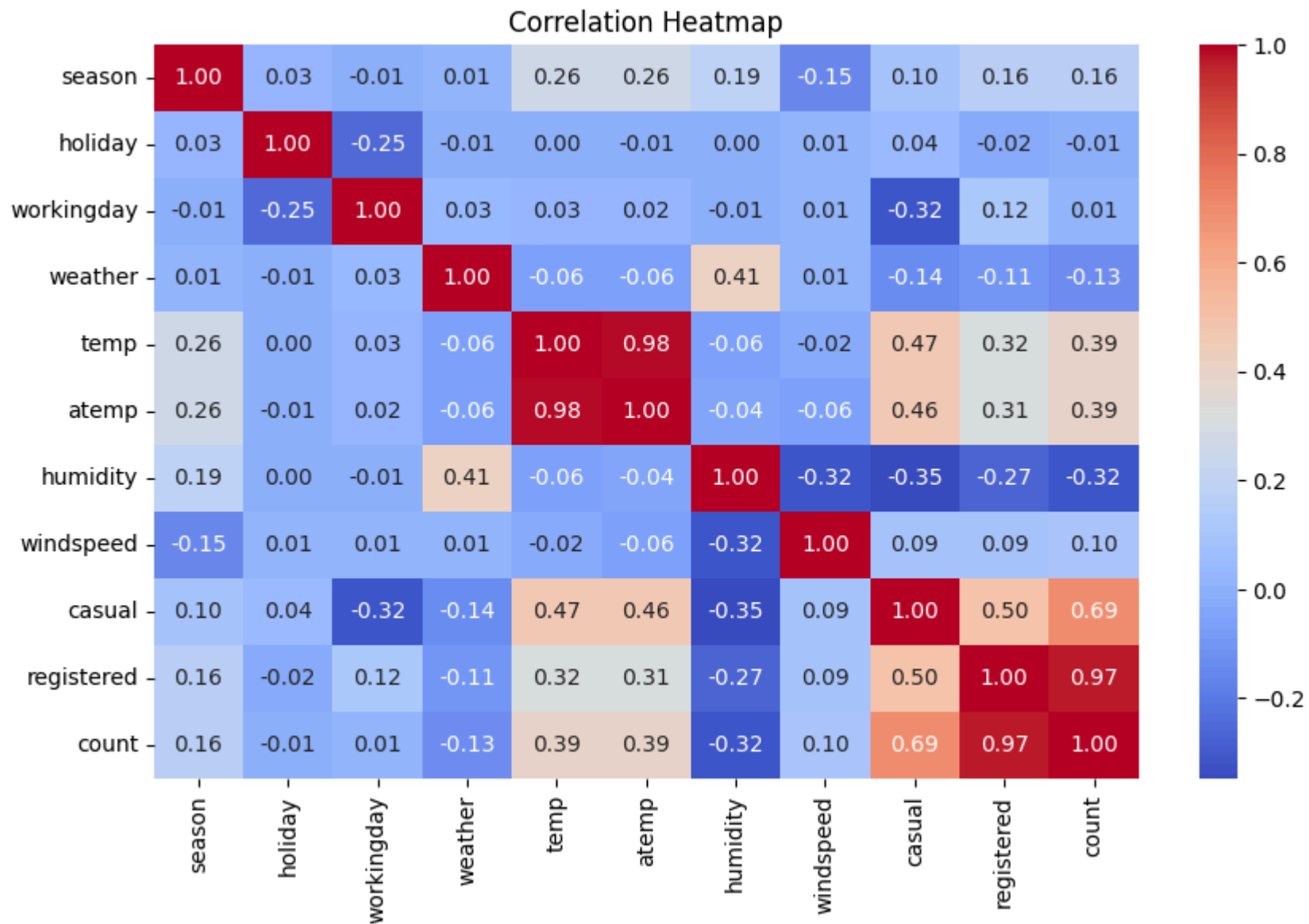


Bivariate Analysis

```
# Correlation heatmap for numerical variables
# Ensure that we only take the numeric columns from the DataFrame
numeric_data = yulu_data.select_dtypes(include=[float, int])

# Calculate the correlation matrix
corr_matrix = numeric_data.corr()

# Plot the heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap')
plt.show()
```



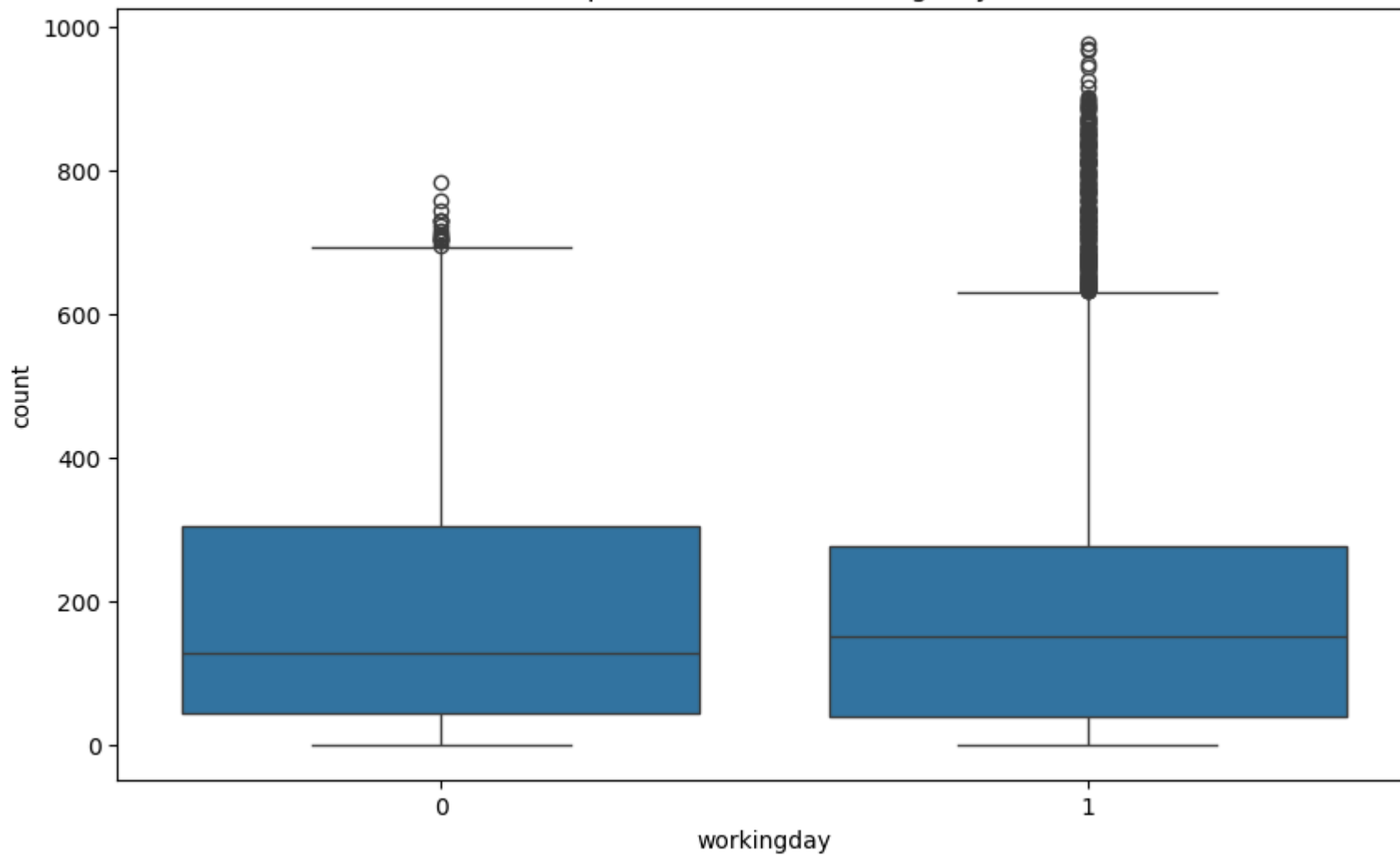
```
# Scatter plots and boxplots for relationships with 'count'
plt.figure(figsize=(10, 6))
sns.boxplot(x='workingday', y='count', data=yulu_data)
plt.title('Boxplot of Count vs Working Day')
plt.show()

plt.figure(figsize=(10, 6))
sns.boxplot(x='season', y='count', data=yulu_data)
plt.title('Boxplot of Count vs Season')
plt.show()

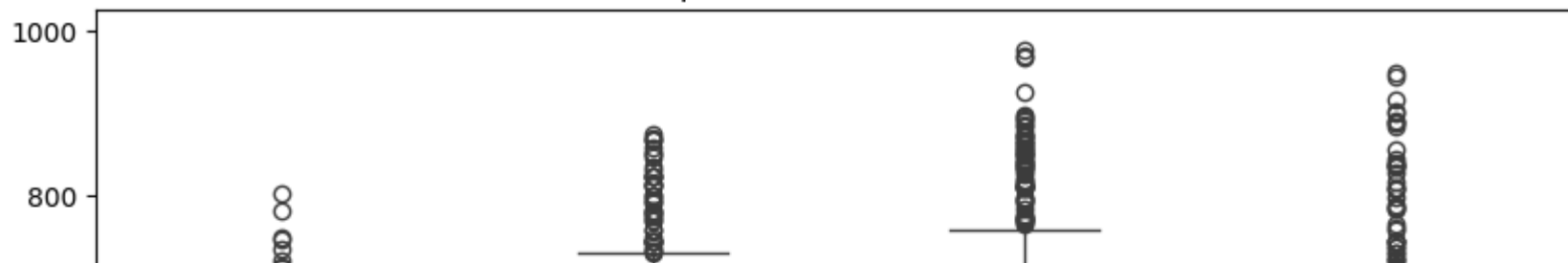
plt.figure(figsize=(10, 6))
sns.boxplot(x='weather', y='count', data=yulu_data)
plt.title('Boxplot of Count vs Weather')
plt.show()
```

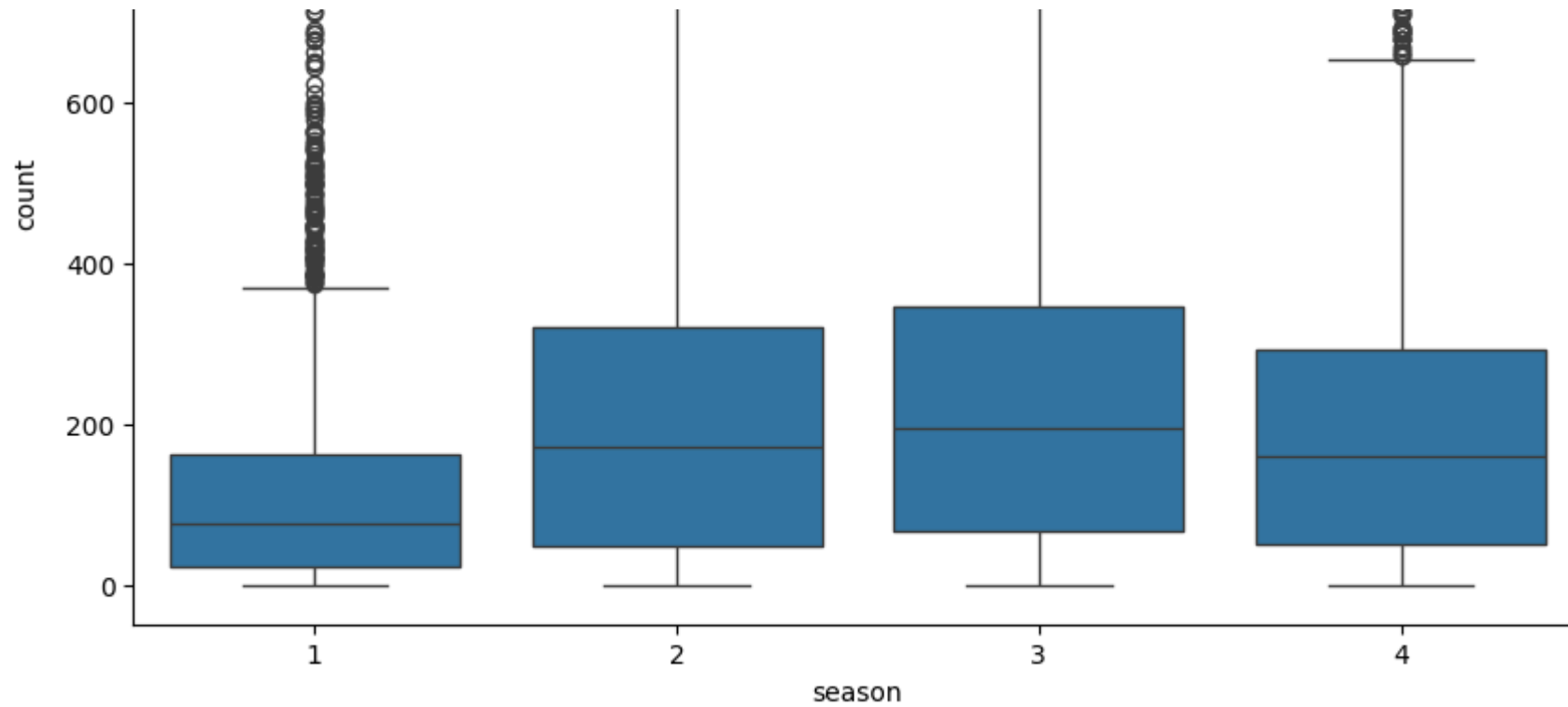


Boxplot of Count vs Working Day

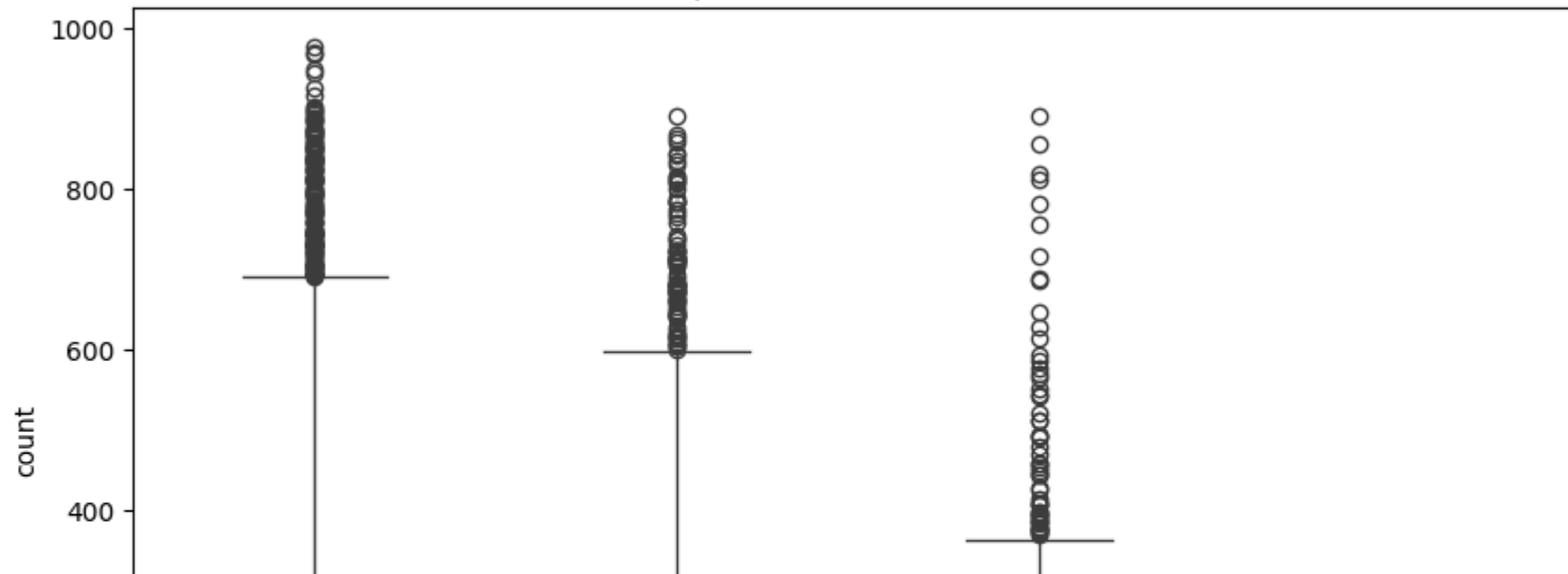


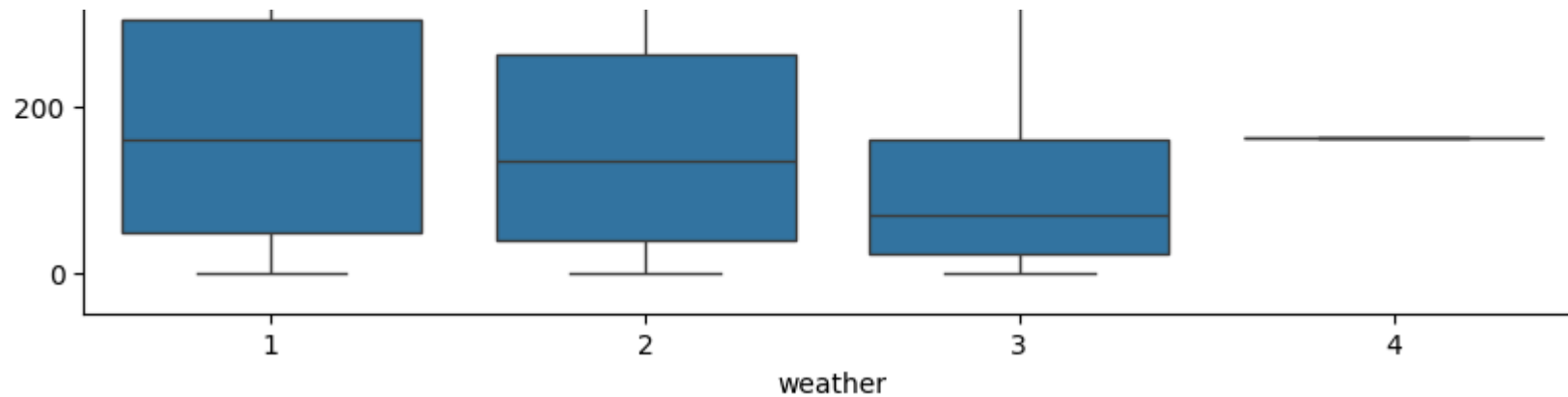
Boxplot of Count vs Season





Boxplot of Count vs Weather





Hypothesis Testing

Effect of Working Day on Number of Electric Cycles Rented Hypotheses: H0: No significant difference in the number of electric cycles rented between working days and non-working days. H1: Significant difference in the number of electric cycles rented between working days and non-working days.

```
# Separate data into working day and non-working day
workingday_count = yulu_data[yulu_data['workingday'] == 1]['count']
non_workingday_count = yulu_data[yulu_data['workingday'] == 0]['count']

# Perform 2-sample t-test
t_stat, p_value = ttest_ind(workingday_count, non_workingday_count)
alpha = 0.05

print(f'T-statistic: {t_stat}, P-value: {p_value}')

if p_value <= alpha:
    print("Reject the null hypothesis: There is a significant difference in the number of electric cycles rented between working day")
else:
    print("Fail to reject the null hypothesis: There is no significant difference in the number of electric cycles rented between wc")
```

→ T-statistic: 1.2096277376026694, P-value: 0.22644804226361348
Fail to reject the null hypothesis: There is no significant difference in the number of electric cycles rented between working c

ANOVA: Number of Cycles Rented in Different Weathers and Seasons Hypotheses for Weather:

H0: No significant difference in the number of cycles rented across different weather conditions. H1: Significant difference in the number of cycles rented across different weather conditions. Hypotheses for Season:

H0: No significant difference in the number of cycles rented across different seasons. H1: Significant difference in the number of cycles rented across different seasons.

```
# Groups for weather
weather_groups = [yulu_data[yulu_data['weather'] == i]['count'] for i in yulu_data['weather'].unique()]

# Perform ANOVA for weather
f_stat_weather, p_value_weather = f_oneway(*weather_groups)
print(f'Weather - F-statistic: {f_stat_weather}, P-value: {p_value_weather}')

if p_value_weather <= alpha:
    print("Reject the null hypothesis for weather: There is a significant difference in the number of cycles rented across different
else:
    print("Fail to reject the null hypothesis for weather: There is no significant difference in the number of cycles rented across
```

→ Weather - F-statistic: 65.53024112793271, P-value: 5.482069475935669e-42
Reject the null hypothesis for weather: There is a significant difference in the number of cycles rented across different weathe

```
# Groups for season
season_groups = [yulu_data[yulu_data['season'] == i]['count'] for i in yulu_data['season'].unique()]

# Perform ANOVA for season
f_stat_season, p_value_season = f_oneway(*season_groups)
print(f'Season - F-statistic: {f_stat_season}, P-value: {p_value_season}')

if p_value_season <= alpha:
    print("Reject the null hypothesis for season: There is a significant difference in the number of cycles rented across different seasons")
else:
    print("Fail to reject the null hypothesis for season: There is no significant difference in the number of cycles rented across different seasons")
```

Season - F-statistic: 236.94671081032106, P-value: 6.164843386499654e-149
Reject the null hypothesis for season: There is a significant difference in the number of cycles rented across different seasons

Chi-Square Test: Dependency Between Weather and Season Hypotheses: H0: Weather conditions are independent of the season. H1: Weather conditions are dependent on the season.

```
# Create a contingency table
contingency_table = pd.crosstab(yulu_data['weather'], yulu_data['season'])

# Perform Chi-square test
chi2_stat, p_value, dof, expected = chi2_contingency(contingency_table)
print(f'Chi2 Statistic: {chi2_stat}, P-value: {p_value}')

if p_value <= alpha:
    print("Reject the null hypothesis: Weather conditions are dependent on the season.")
else:
    print("Fail to reject the null hypothesis: Weather conditions are independent of the season.")
```

Chi2 Statistic: 49.15865559689363, P-value: 1.5499250736864862e-07
Reject the null hypothesis: Weather conditions are dependent on the season.

Draw Inferences and Recommendations Key Findings: Working Days:

If the p-value is significant, it indicates a difference in demand between working and non-working days. Yulu can target specific promotions for working days or weekends to balance the demand. Weather Conditions:

A significant p-value indicates weather conditions affect the number of cycles rented. Yulu can prepare for adverse weather conditions by optimizing bike availability and maintenance schedules. Seasons:

A significant p-value suggests seasonal variations in demand. Yulu can use this insight to plan for peak and off-peak seasons, adjusting their inventory and marketing strategies accordingly. Dependency Between Weather and Season:

If weather conditions are dependent on the season, Yulu can better predict and plan for expected weather patterns based on the season, ensuring a smoother operational process.

Conclusion This solution includes the problem definition, EDA, hypothesis testing, and key findings with appropriate visualizations and