```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
from scipy.stats import norm, t, ttest_1samp,ttest_ind
from scipy.stats import f, f_oneway, shapiro, levene
from scipy.stats import chi2_contingency
```

```
df = pd.read_csv("bike_sharing.csv")
```

1. Define the Problem Statement, Import the required Libraries and perform Exploratory Data Analysis.

1.a) Examine dataset structure, characteristics, and statistical summary.

```
df.head()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0

```
df.info()
```

```
<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
```

```
df.describe()
```

	season	holiday	workingday	weather	temp	atem
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.00000
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.65508
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.47460
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.76000
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.66500
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.24000
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.06000
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.45500

1.b) Identify missing values and perform Imputation using an appropriate method.

```
df.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
```

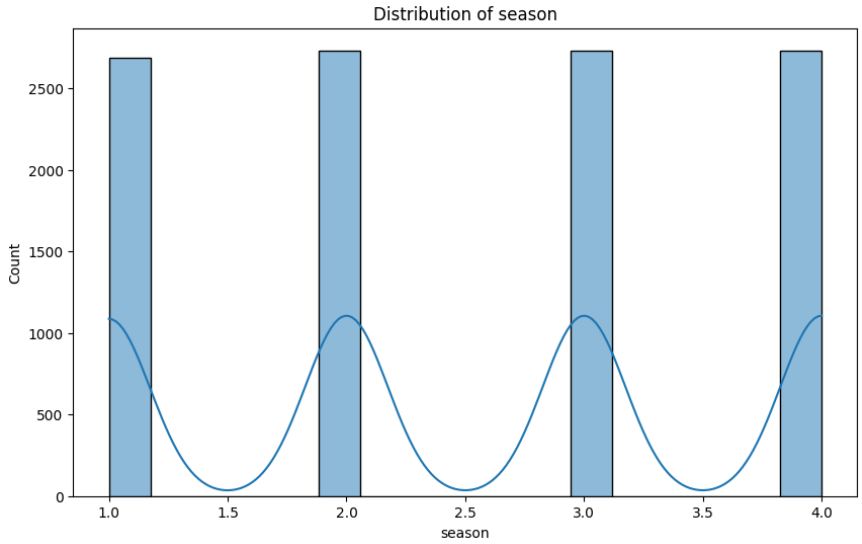
1.c)Identify and remove duplicate records.

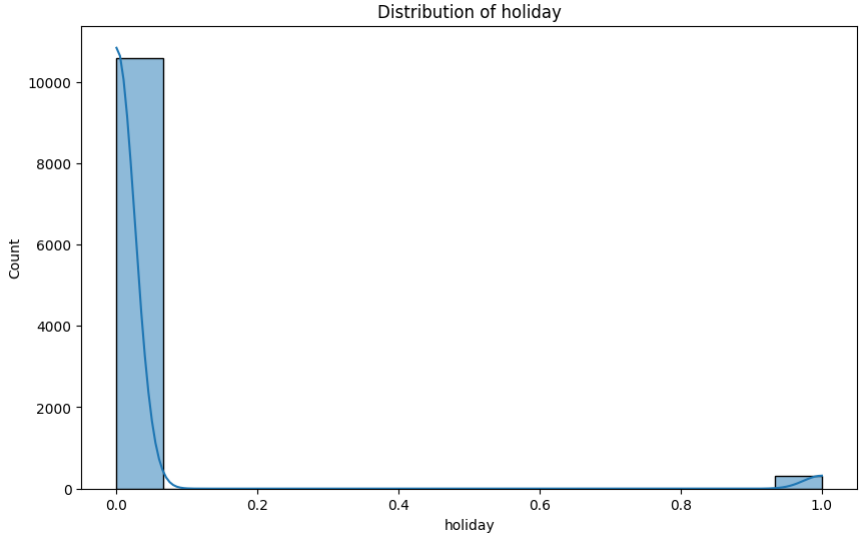
```
# Check for duplicate records
duplicates = df.duplicated()

# Display duplicate records (if any)
print("Duplicate Records:")
print(df[duplicates])
## no dupilicates found
```

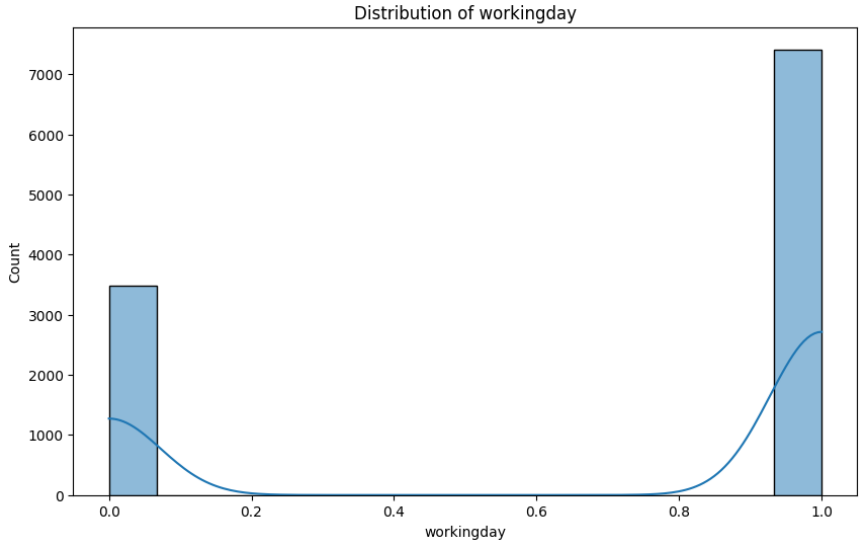
```
Duplicate Records:
Empty DataFrame
Columns: [datetime, season, holiday, workingday, weather, temp, atemp, humidity, windspeed, casual, registered, count]
Index: []
```

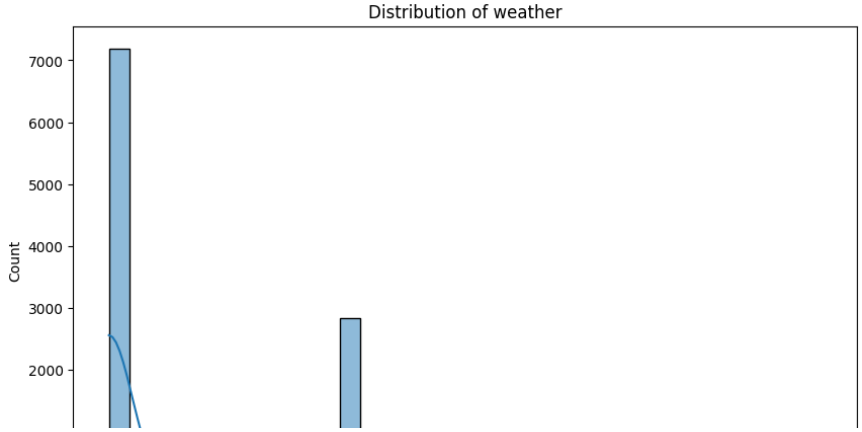
1.d) Analyze the distribution of Numerical & Categorical variables, separately

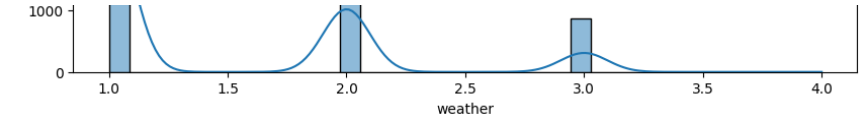
```
# Extracting numerical and categorical columns
numerical_columns = df.select_dtypes(include=['int64', 'float64']).columns
categorical_columns = df.select_dtypes(include=['object']).columns

# Analyze the distribution of Numerical variables
for col in numerical_columns:
    plt.figure(figsize=(10, 6))
    sns.histplot(df[col], kde=True)
    plt.title(f'Distribution of {col}')
    plt.show()

# Analyze the distribution of Categorical variables
for col in categorical_columns:
    plt.figure(figsize=(8, 6))
    sns.countplot(x=col, data=df, palette='viridis')
    plt.title(f'Distribution of {col}')
    plt.show()
```

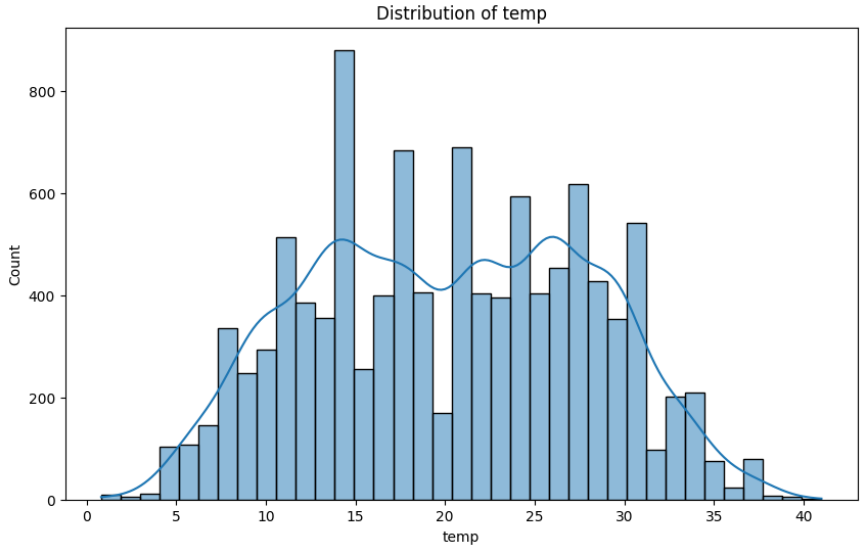

Distribution of season
Count
season
0
500
1000
1500
2000
2500
1.0
1.5
2.0
2.5
3.0
3.5
4.0

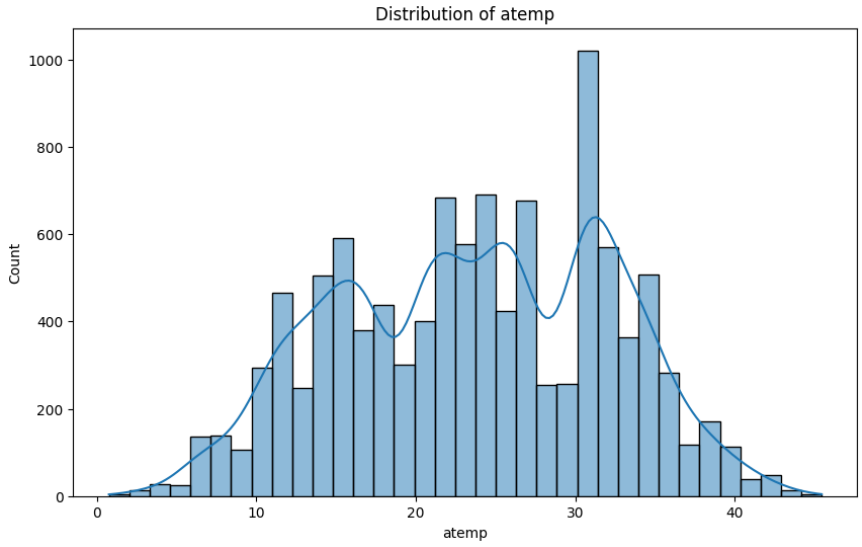

Distribution of holiday
Count
holiday
0
2000
4000
6000
8000
10000
0.0
0.2
0.4
0.6
0.8
1.0

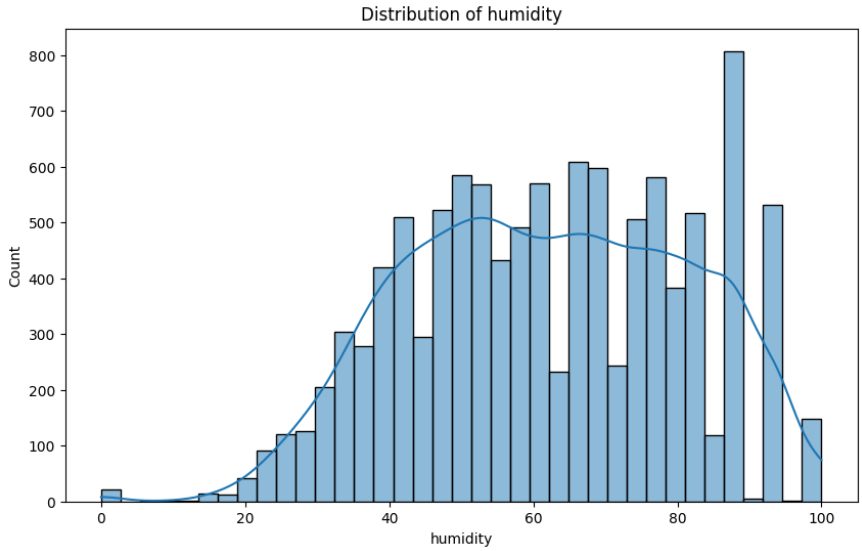

Distribution of workingday
Count
workingday
0
1000
2000
3000
4000
5000
6000
7000
0.0
0.2
0.4
0.6
0.8
1.0

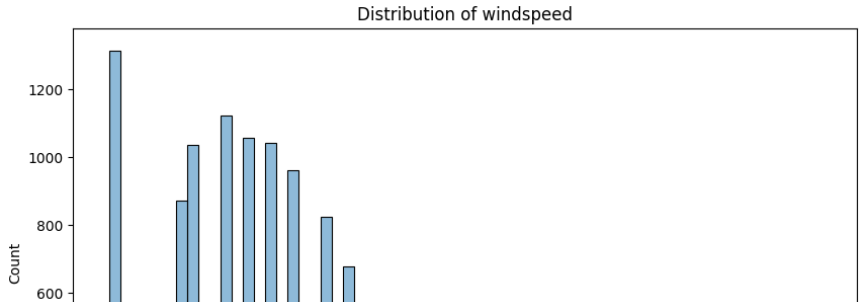

Distribution of weather
Count
0
2000
3000
4000
5000
6000
7000

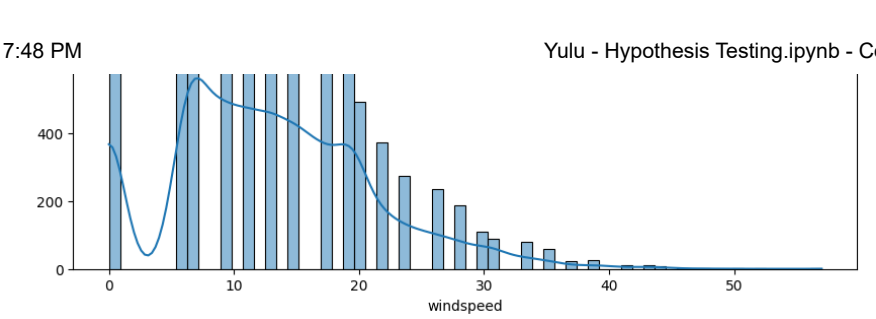

1000
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
weather

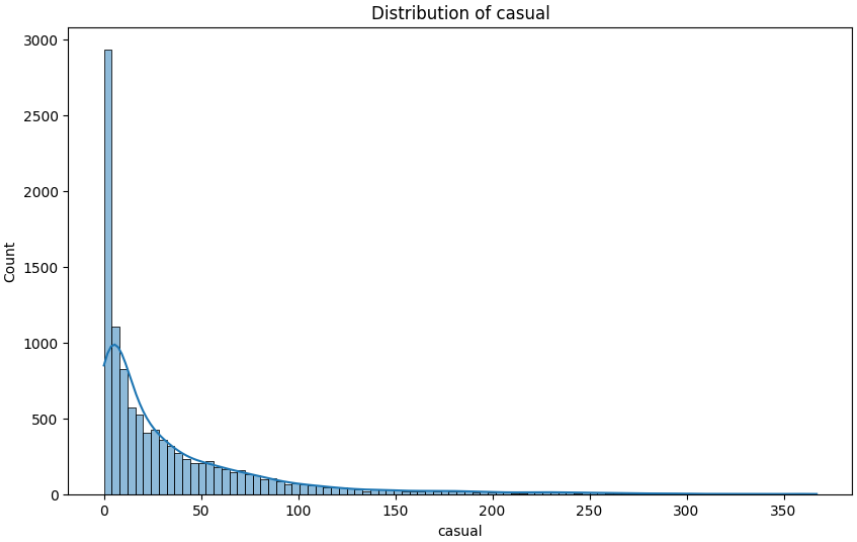

Distribution of temp
800
600
400
200
0
Count
0
5
10
15
20
25
30
35
40
temp

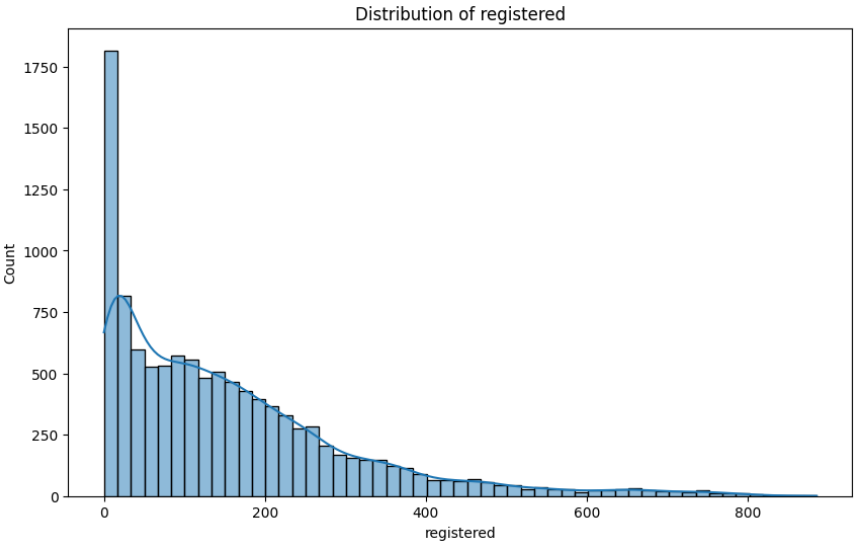

Distribution of atemp
1000
800
600
400
200
0
Count
0
10
20
30
40
atemp

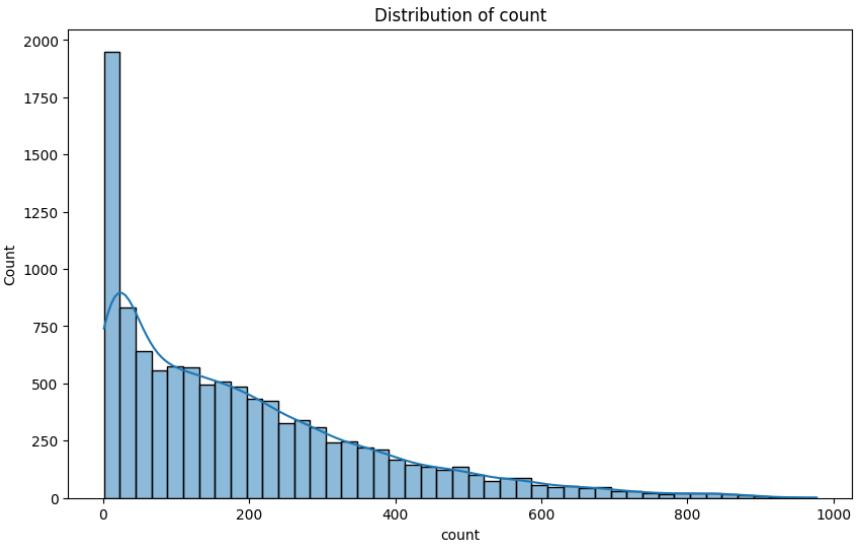

Distribution of humidity
800
700
600
500
400
300
200
100
0
Count
0
20
40
60
80
100
humidity

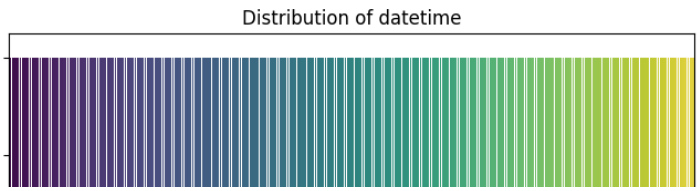

Distribution of windspeed
1200
1000
800
600
Count

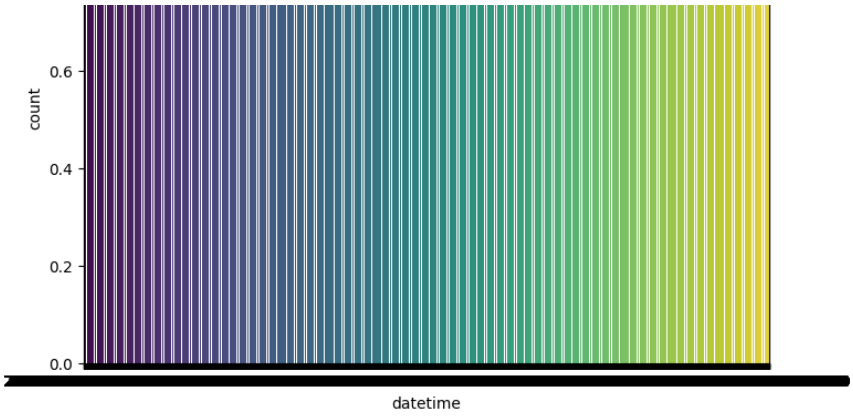

windspeed


Distribution of casual
Count
casual


Distribution of registered
Count
registered

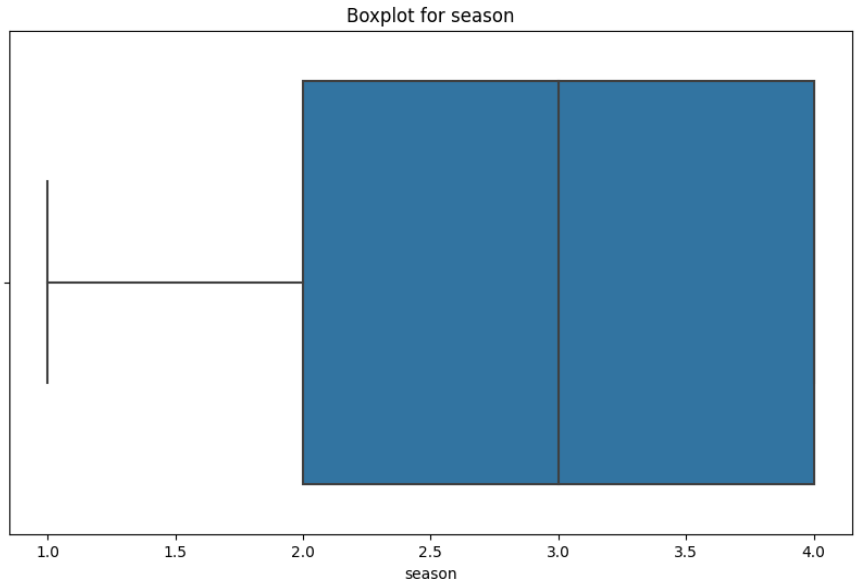

Distribution of count
Count
count

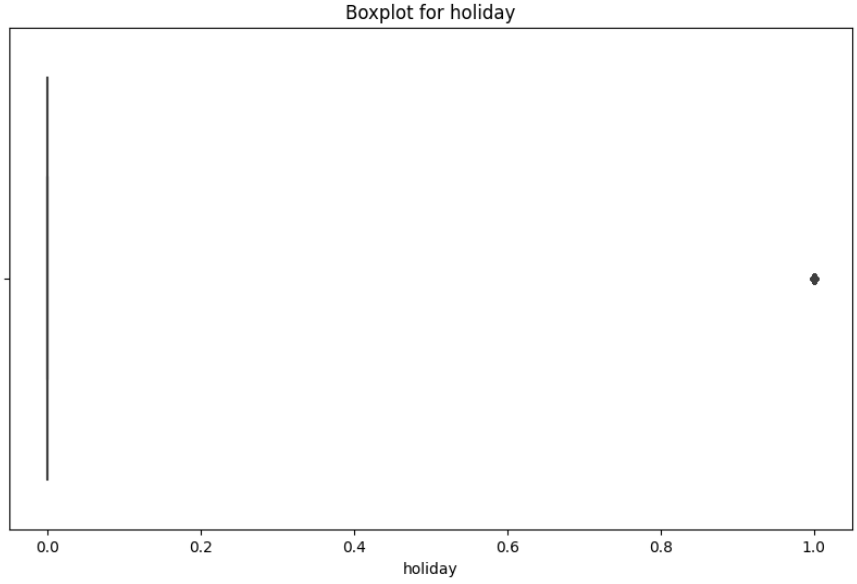

Distribution of datetime

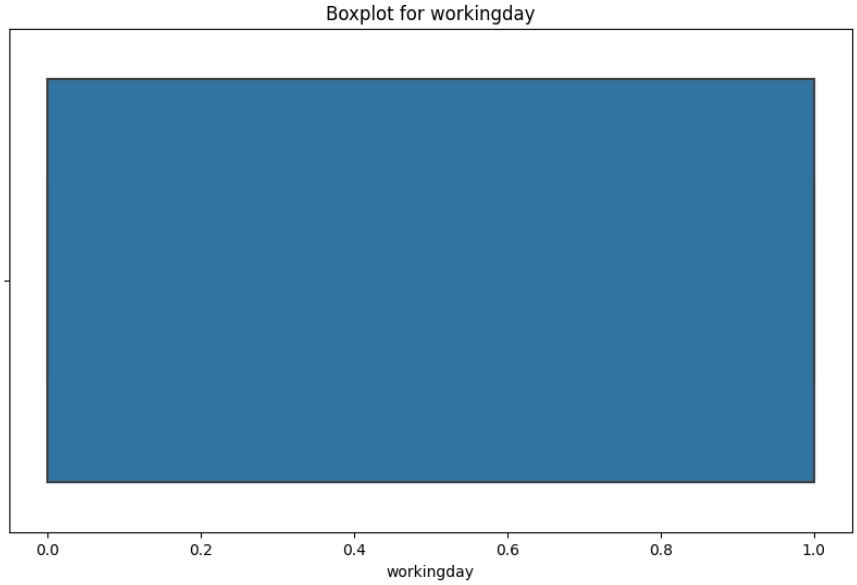


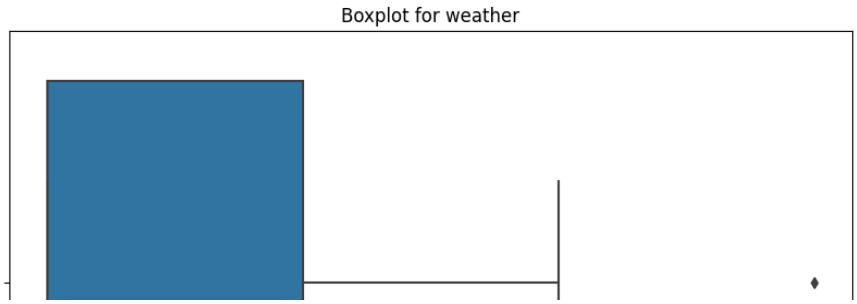
1.e) Check for Outliers and deal with them accordingly.

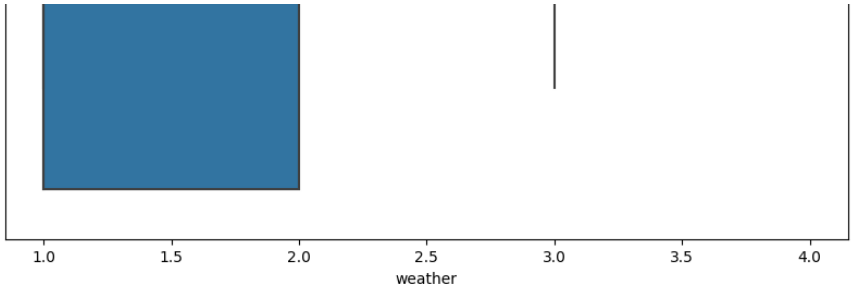
```
# Identify and visualize outliers using boxplots for numerical variables
numerical_columns = df.select_dtypes(include=['int64', 'float64']).columns

for col in numerical_columns:
    plt.figure(figsize=(10, 6))
    sns.boxplot(x=df[col])
    plt.title(f'Boxplot for {col}')
    plt.show()

# Function to handle outliers using IQR
def handle_outliers(data, col):
    Q1 = data[col].quantile(0.25)
    Q3 = data[col].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Identify and remove outliers
    outliers = (data[col] < lower_bound) | (data[col] > upper_bound)
    data_cleaned = data[~outliers]

    return data_cleaned

# Remove outliers for each numerical column
for col in numerical_columns:
    df = handle_outliers(df, col)

# Verify the removal of outliers
for col in numerical_columns:
    plt.figure(figsize=(10, 6))
    sns.boxplot(x=df[col])
    plt.title(f'Boxplot for {col} (After Handling Outliers)')
    plt.show()
```


Boxplot for season
1.0
1.5
2.0
2.5
3.0
3.5
4.0
season

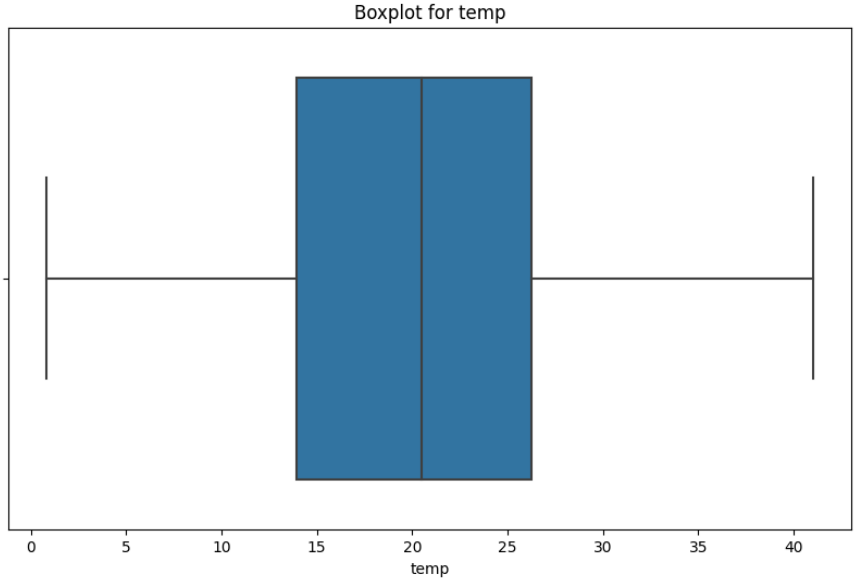

Boxplot for holiday
0.0
0.2
0.4
0.6
0.8
1.0
holiday


Boxplot for workingday
0.0
0.2
0.4
0.6
0.8
1.0
workingday

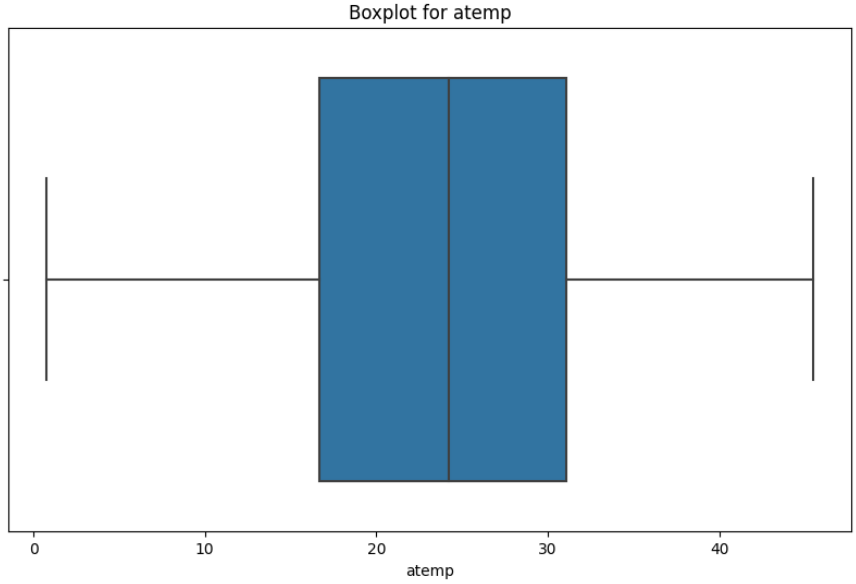

Boxplot for weather



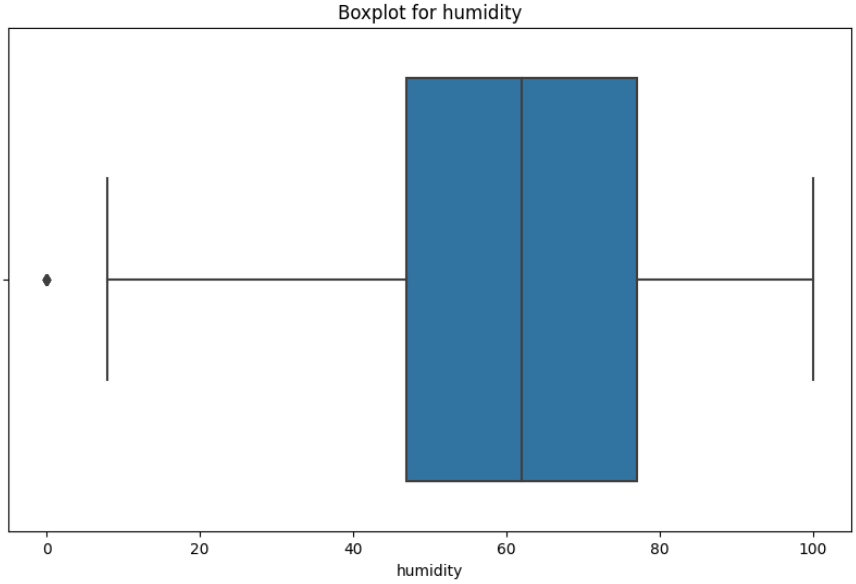
Boxplot for temp



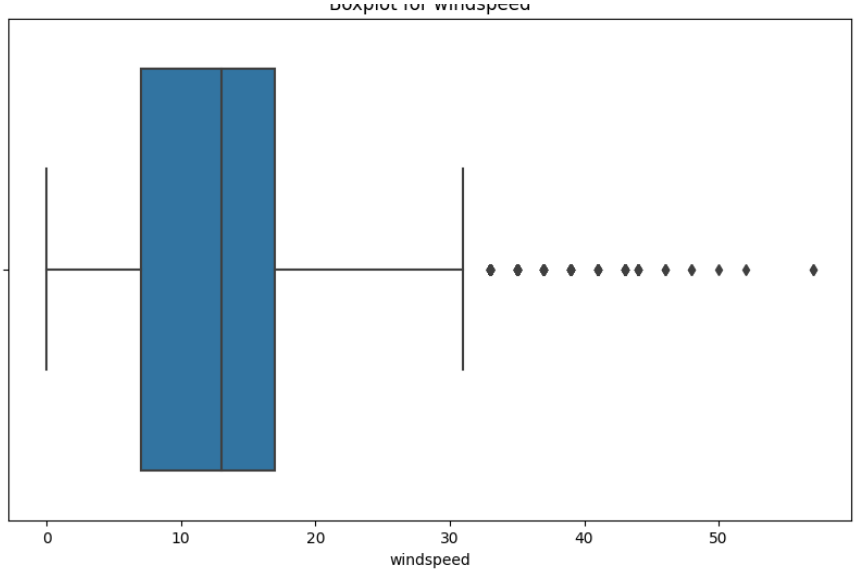
Boxplot for atemp

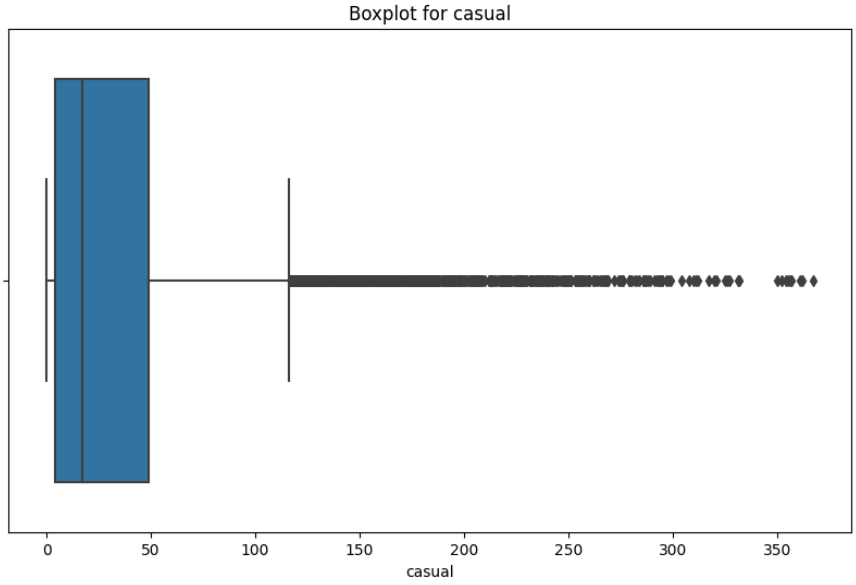


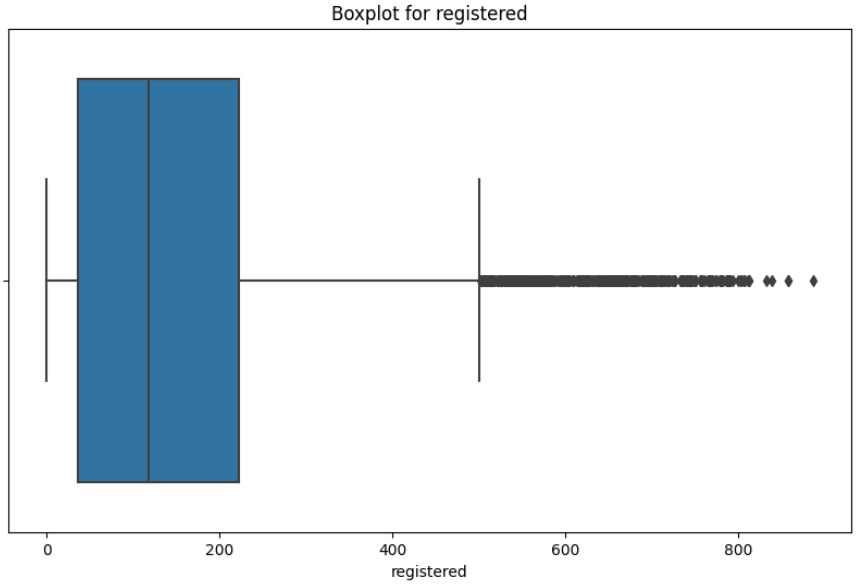
Boxplot for humidity

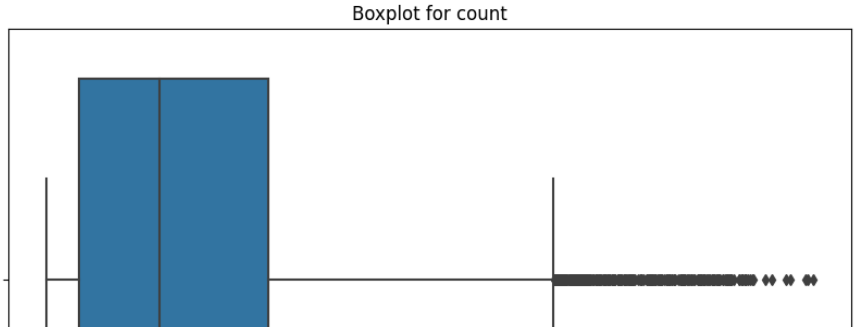


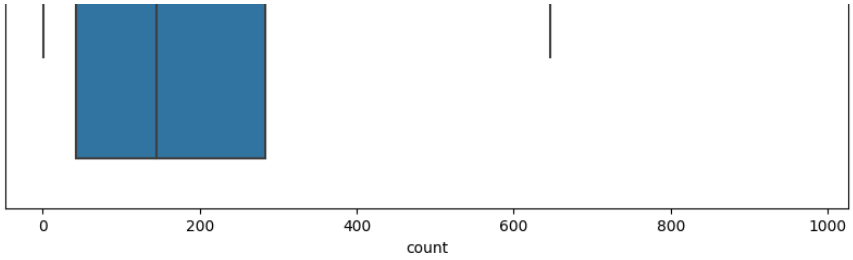
Boxplot for windspeed


0
10
20
30
40
50
windspeed

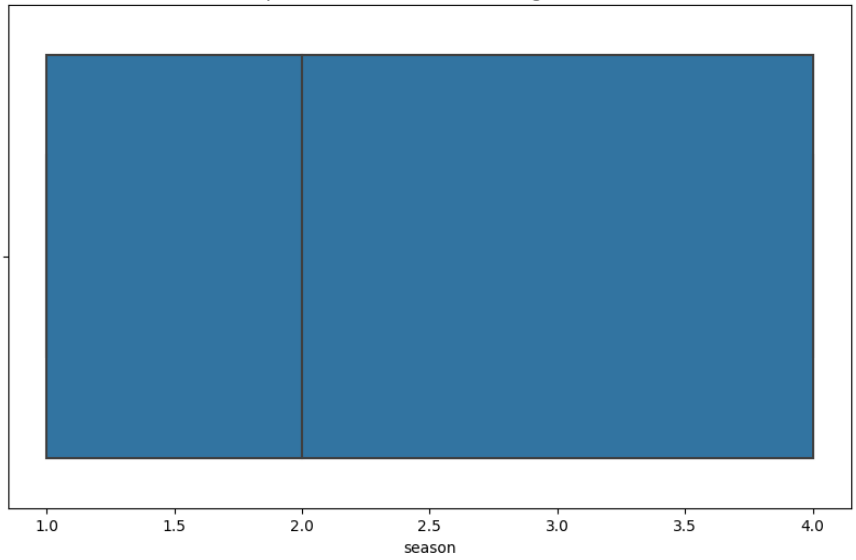

Boxplot for casual
0
50
100
150
200
250
300
350
casual


Boxplot for registered
0
200
400
600
800
registered

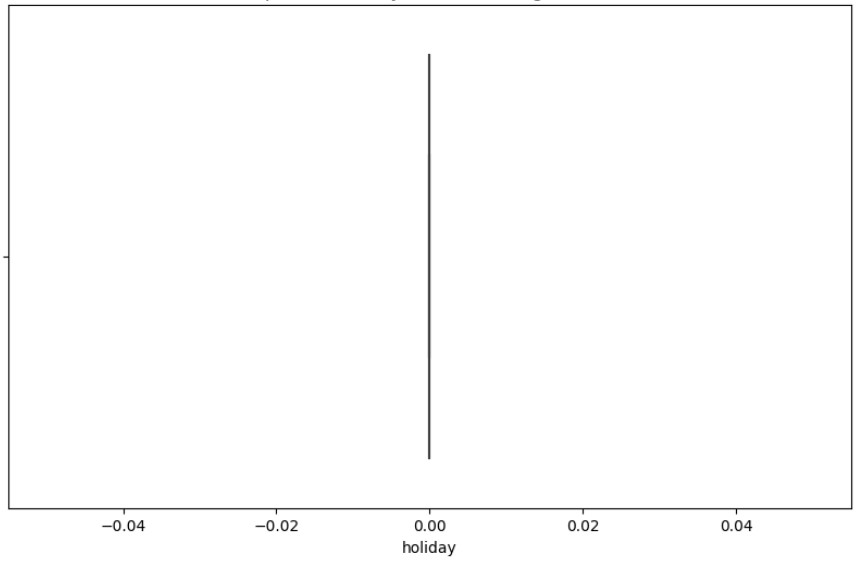

Boxplot for count



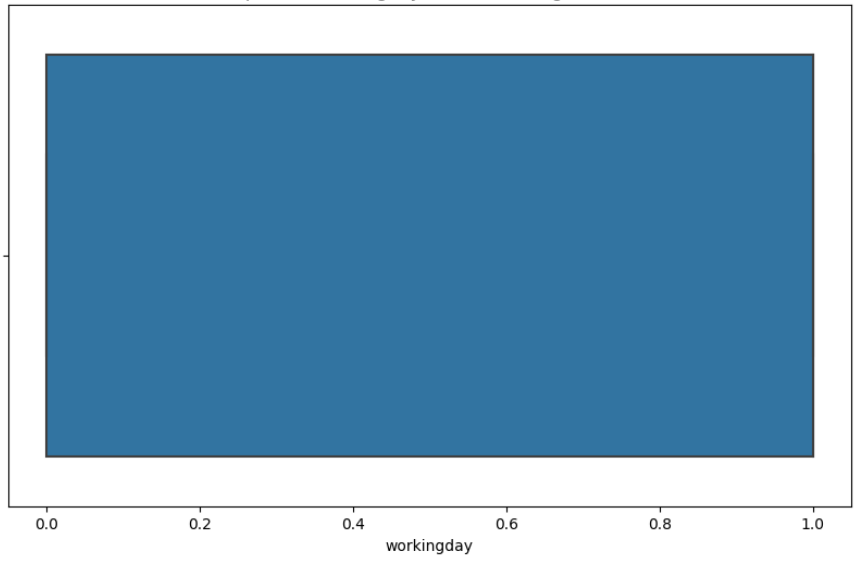
Boxplot for season (After Handling Outliers)



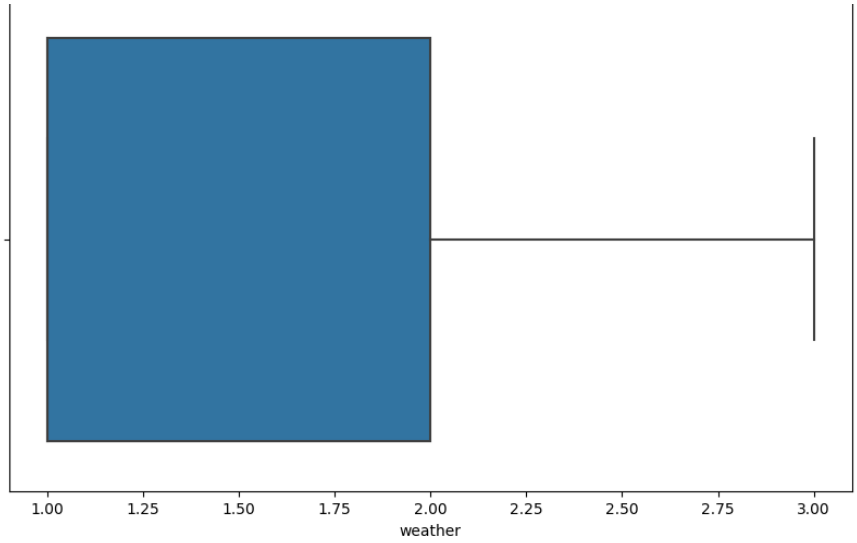
Boxplot for holiday (After Handling Outliers)



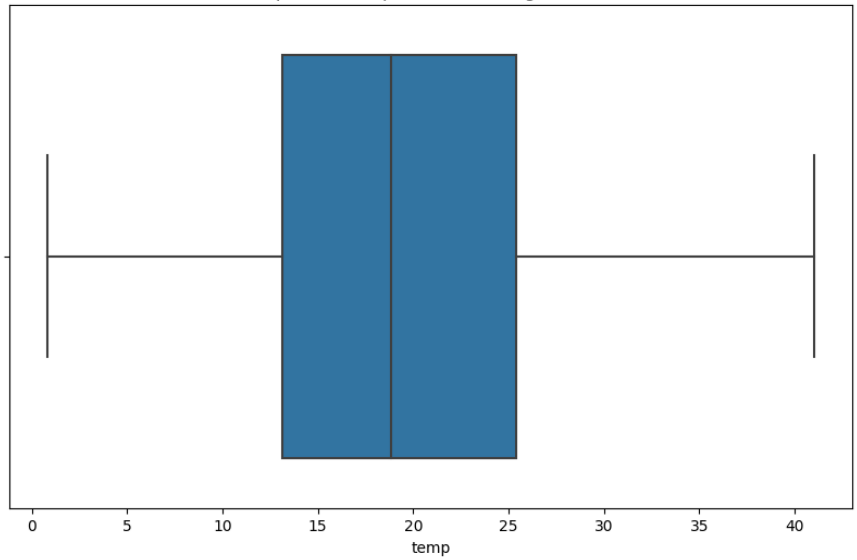
Boxplot for workingday (After Handling Outliers)



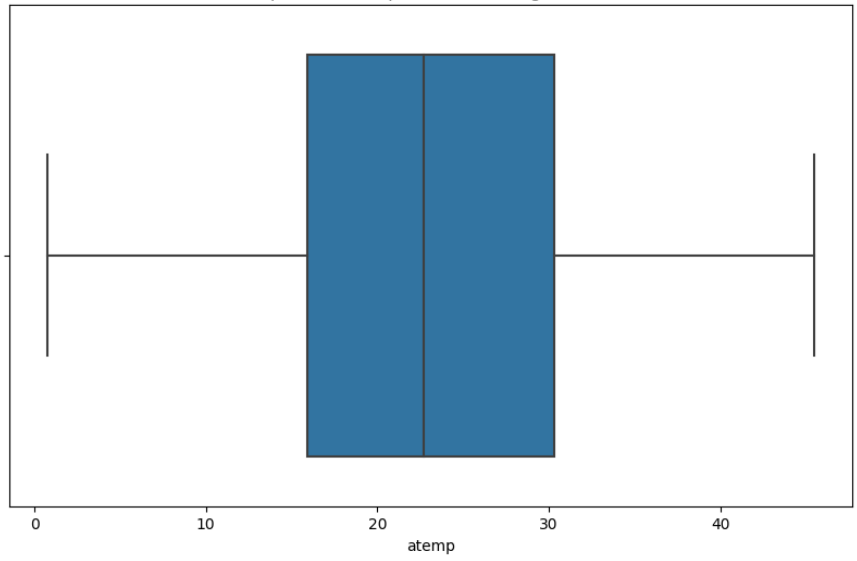
Boxplot for weather (After Handling Outliers)



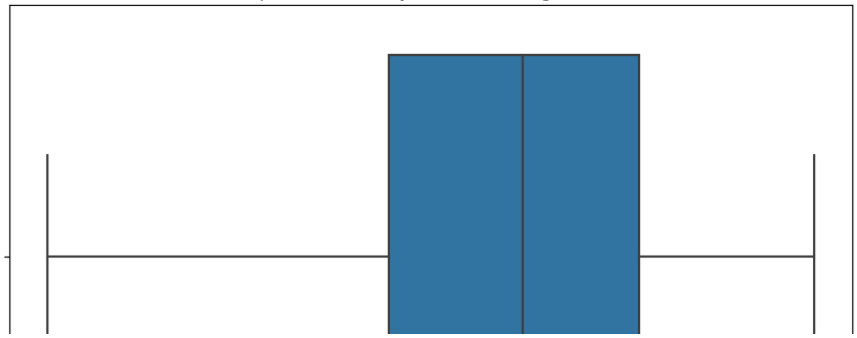
Boxplot for temp (After Handling Outliers)

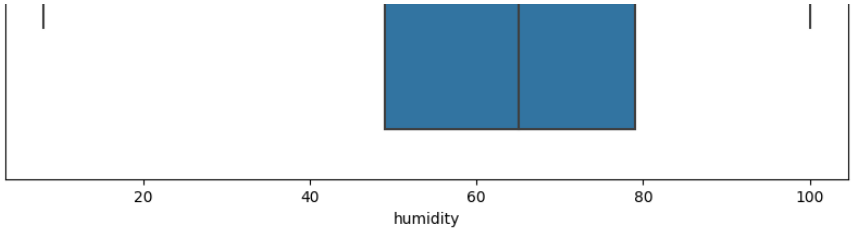


Boxplot for atemp (After Handling Outliers)

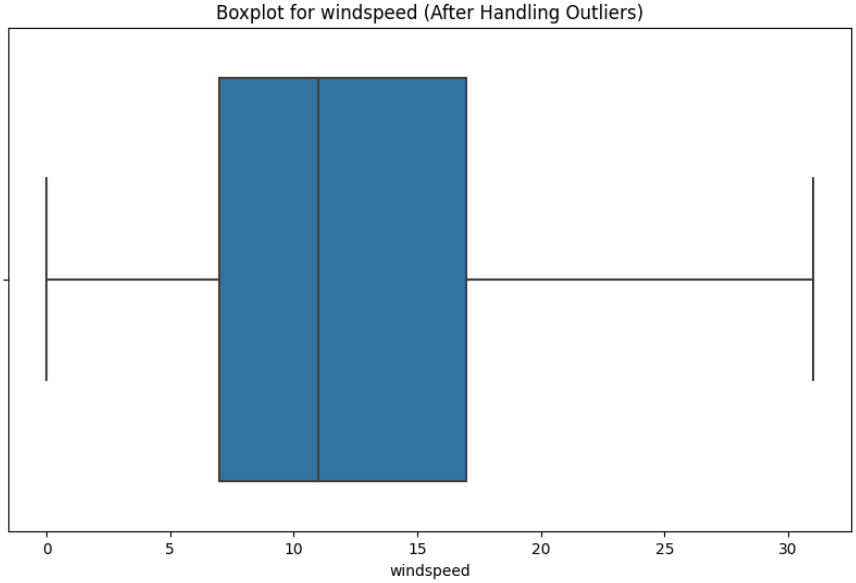


Boxplot for humidity (After Handling Outliers)

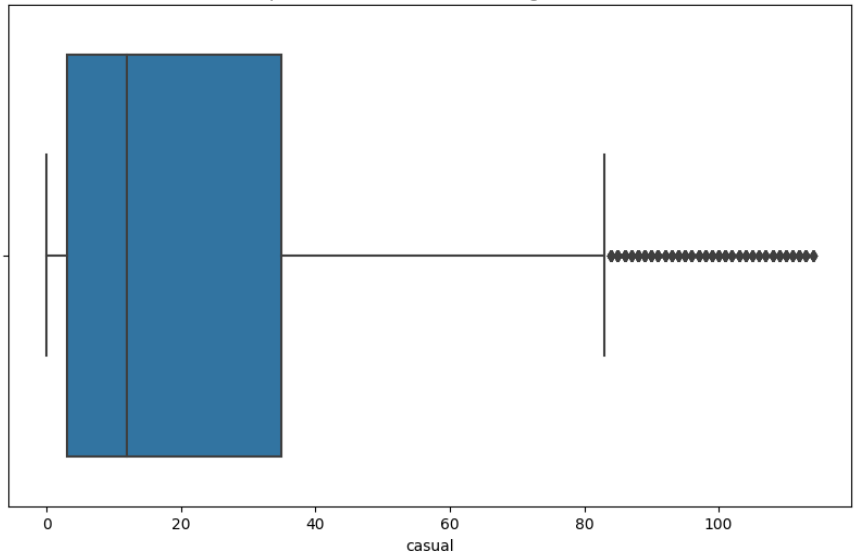




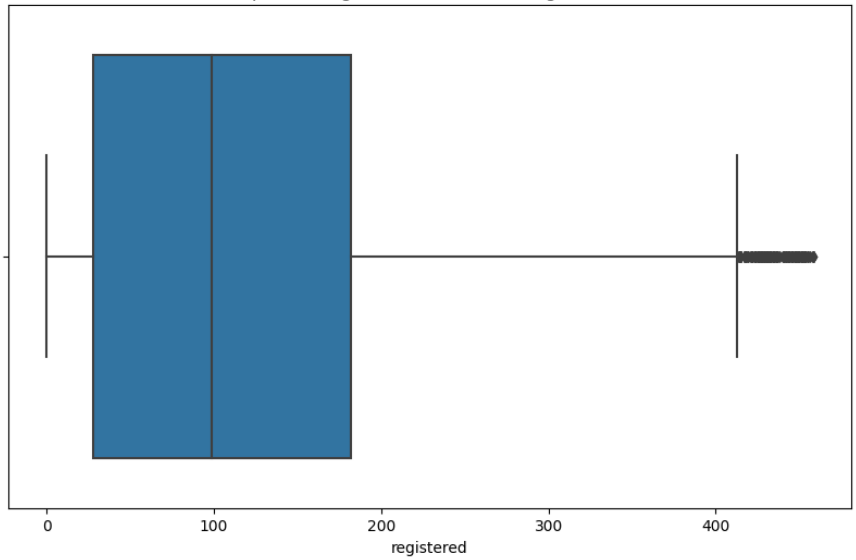
Boxplot for windspeed (After Handling Outliers)



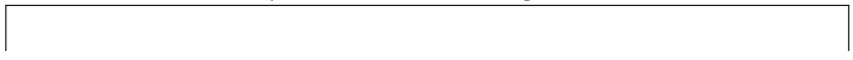
Boxplot for casual (After Handling Outliers)

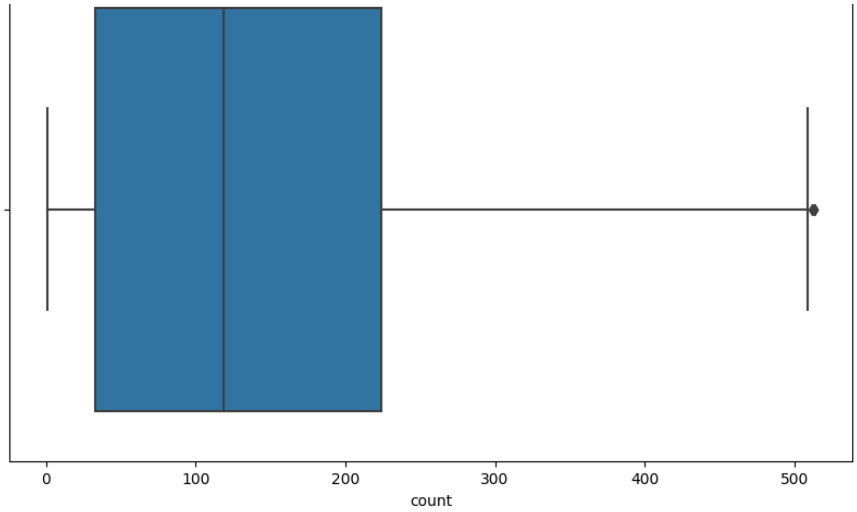


Boxplot for registered (After Handling Outliers)



Boxplot for count (After Handling Outliers)

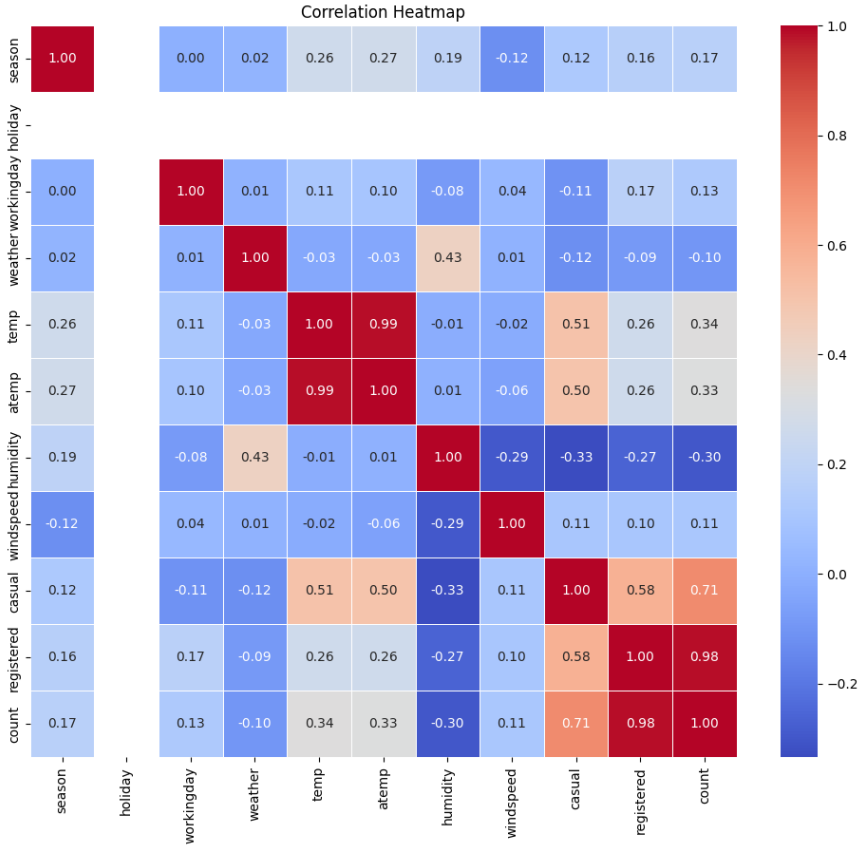



0
100
200
300
400
500
count

2. Try establishing a Relationship between the Dependent and Independent Variables.

```
# Calculate the correlation matrix
correlation_matrix = df.corr()

# Plot a correlation heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```

```
<ipython-input-29-fd973f8ae17f>:2: FutureWarning: The default value of numeric_only i
  correlation_matrix = df.corr()
```



3.Check if there any significant difference between the no. of bike rides on Weekdays and Weekends?

```
## Formulating Hypothesis:-
## Null Hypothesis (H0): There is no significant difference in the number of bike rides between weekdays and weekends.
## Alternate Hypothesis (Ha): There is a significant difference in the number of bike rides between weekdays and weekends.

## Selecting appropriate test
## Test: 2-Sample Independent T-test
from scipy.stats import ttest_ind

## Setting significance level : alpha = 0.05
alpha = 0.05

weekday_counts = df[df['workingday'] == 1]['count']
weekend_counts = df[df['workingday'] == 0]['count']

# Perform 2-sample independent t-test
t_statistic, p_value = ttest_ind(weekday_counts, weekend_counts)

print(f"T-statistic: {t_statistic}")
print(f"P-value: {p_value}")

## Decide whether to accept or reject the Null Hypothesis:
if p_value<alpha:
  print("Interpretation : Reject Ho")
else:
  print("Interpretation : Fail to Reject Ho")
```

```
T-statistic: 12.205234749740082
P-value: 5.375190338688469e-34
Interpretation : Reject Ho
```

Draw inferences & conclusions from the analysis and provide recommendations.:-

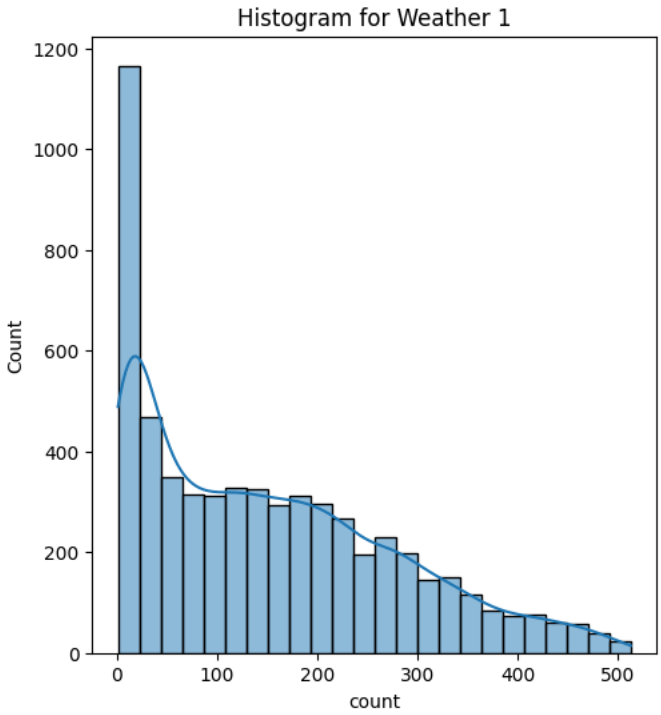
With a T-statistic of 12.205 and an extremely low p-value (5.375e-34), we reject the null hypothesis (H0). The p-value is far less than the significance level (alpha) of 0.05. Therefore, we have strong evidence to suggest that there is a significant difference in the number of bike rides between weekdays and weekends.

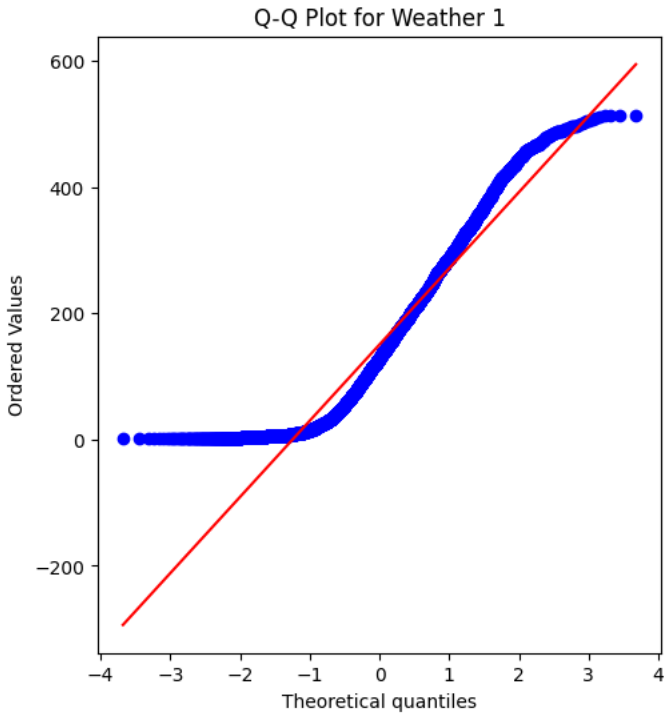
Inferences & Conclusions:

1. **Statistical Significance:** The results of the 2-Sample Independent T-test indicate a highly statistically significant difference in the number of bike rides between weekdays and weekends.
2. **Practical Significance:** The large T-statistic suggests that the observed difference is not only statistically significant but also practically significant.
3. **Recommendations:**
    - **Resource Allocation:** Consider adjusting resource allocation for bike-related services based on the observed difference between weekdays and weekends.
    - **Marketing Strategies:** Tailor marketing strategies or promotions to account for the variation in demand between weekdays and weekends.
4. **Further Analysis:** Explore the specific factors contributing to the significant difference. Investigate whether certain weekdays or weekends exhibit more significant variations. This could provide additional insights for targeted interventions.
5. **Seasonal Considerations:** If relevant, consider how seasonal factors may influence bike rides on weekdays and weekends.

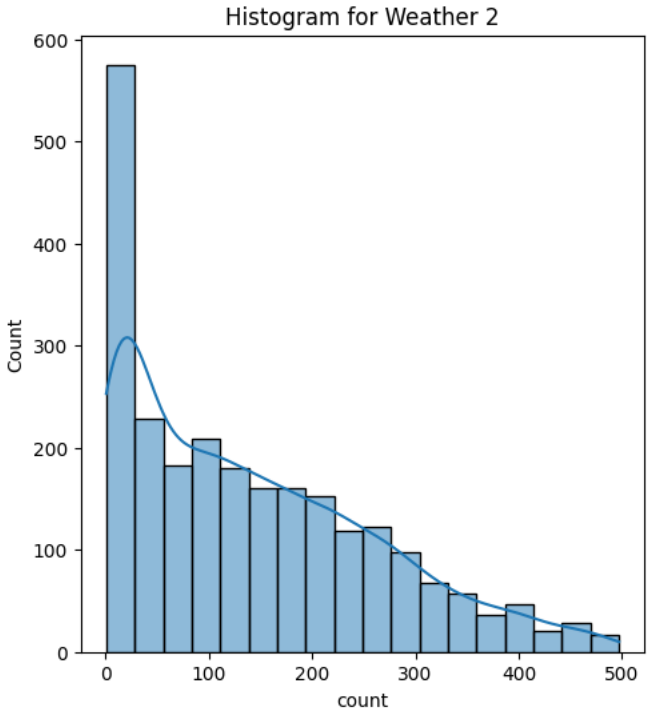
4. Check if the demand of bicycles on rent is the same for different Weather conditions?

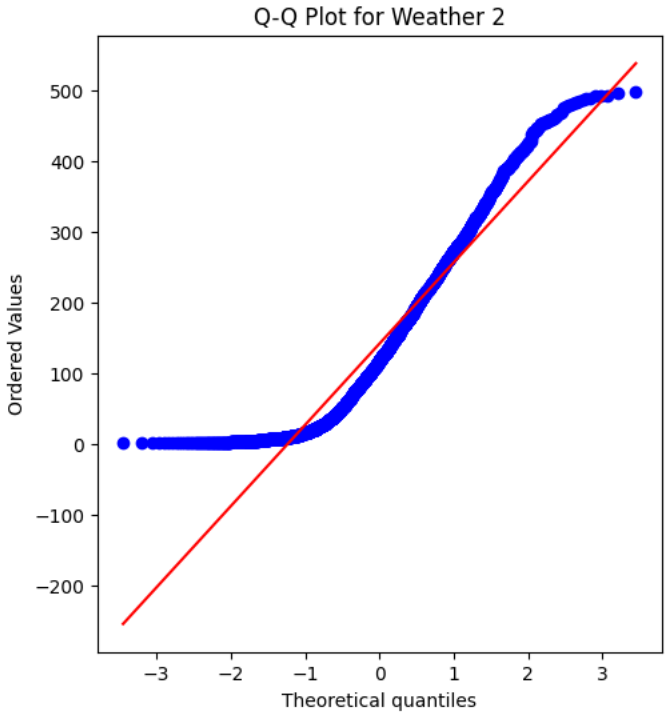
```
##Formulate Null Hypothesis (H0) and Alternate Hypothesis (Ha):
##Null Hypothesis (H0): The demand for bicycles on rent is the same for different weather conditions.
##Alternate Hypothesis (Ha): The demand for bicycles on rent is different for at least one weather condition.

# Select an appropriate test:
# Test: One-way ANOVA test (Analysis of Variance).
from scipy.stats import f, f_oneway

weather_conditions = df['weather'].unique()

# Check assumptions
for condition in weather_conditions:
    subset = df[df['weather'] == condition]['count']

    # Visual analysis
    plt.figure(figsize=(12, 6))
    plt.subplot(1, 2, 1)
    sns.histplot(subset, kde=True)
    plt.title(f'Histogram for Weather {condition}')

    plt.subplot(1, 2, 2)
    stats.probplot(subset, plot=plt)
    plt.title(f'Q-Q Plot for Weather {condition}')

    plt.show()

    # Shapiro-Wilk’s test
    shapiro_test_statistic, shapiro_p_value = shapiro(subset)
    print(f"Shapiro-Wilk’s test for Weather {condition}: Statistic={shapiro_test_statistic}, P-value={shapiro_p_value}")

# Levene’s test for equality of variances
levene_test_statistic, levene_p_value = levene(
    df[df['weather'] == 1]['count'],
    df[df['weather'] == 2]['count'],
    df[df['weather'] == 3]['count']
)

print(f"\nLevene’s test for equality of variances: Statistic={levene_test_statistic}, P-value={levene_p_value}")

# One-way ANOVA test
anova_test_statistic, anova_p_value = f_oneway(
    df[df['weather'] == 1]['count'],
    df[df['weather'] == 2]['count'],
    df[df['weather'] == 3]['count']
)

print(f"\nOne-way ANOVA test: Statistic={anova_test_statistic}, P-value={anova_p_value}")

## Setting significance level : alpha = 0.05
alpha = 0.05
## Decide whether to accept or reject the Null Hypothesis:
if anova_p_value<alpha:
  print("Interpretation : Reject Ho")
else:
  print("Interpretation : Fail to Reject Ho")
```



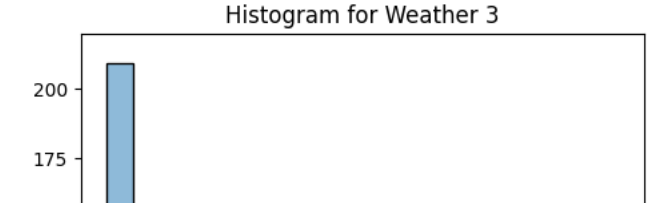


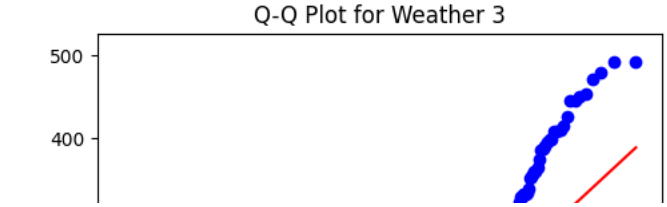

```
/usr/local/lib/python3.10/dist-packages/scipy/stats/_morestats.py:1882: UserWarning: p-value may not be accurate for N > 5000.
  warnings.warn("p-value may not be accurate for N > 5000.")
Shapiro-Wilk's test for Weather 1: Statistic=0.9221206903457642, P-value=0.0
```






```
Shapiro-Wilk's test for Weather 2: Statistic=0.9176433086395264, P-value=1.1781027085019149e-34
```



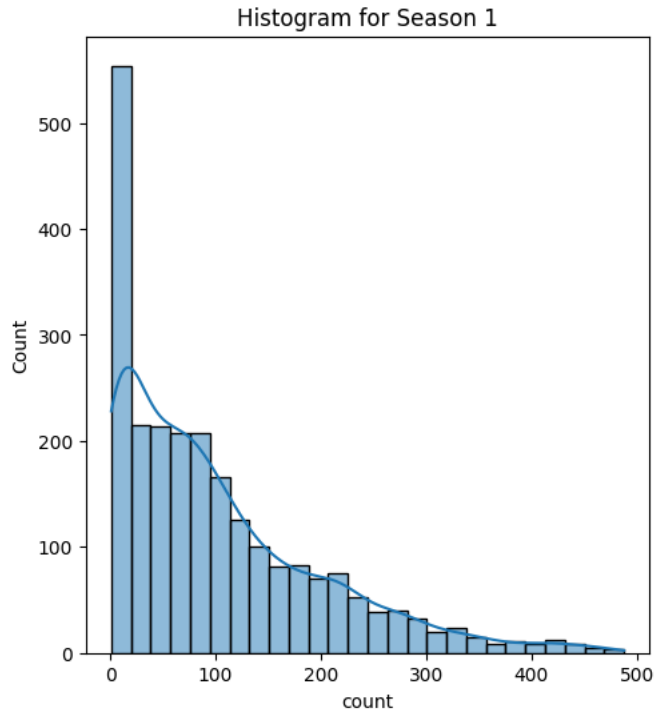


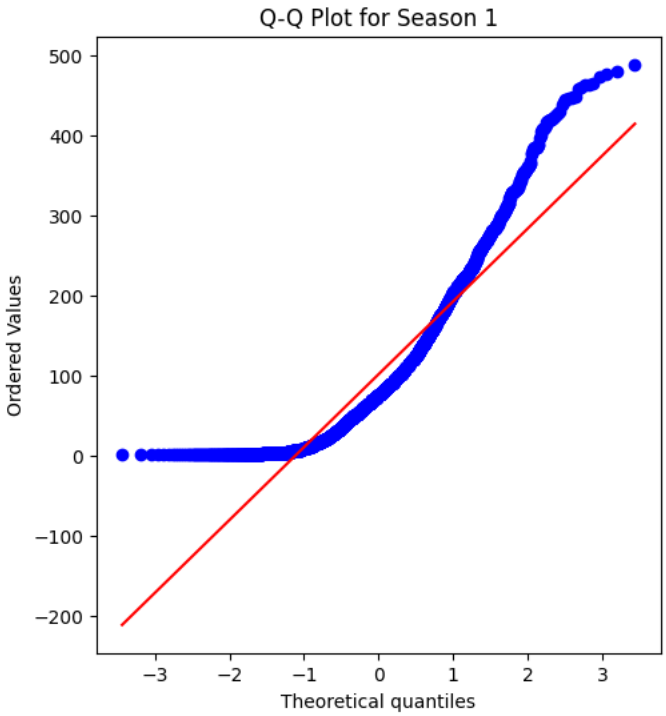
Inferences & Conclusions:

The one-way ANOVA test shows a highly significant p-value (2.675e-25), indicating that there is a significant difference in the demand for bicycles on rent among different weather conditions.

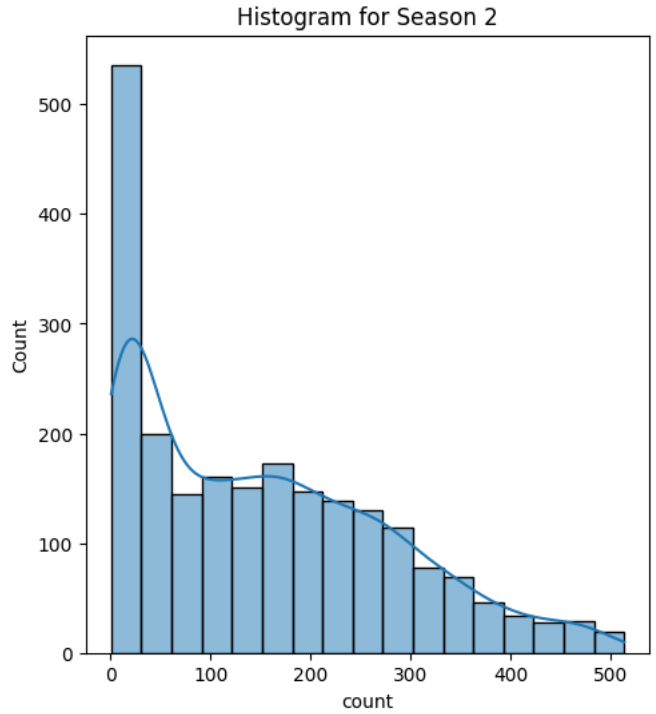
5. Check if the demand of bicycles on rent is the same for different Seasons?

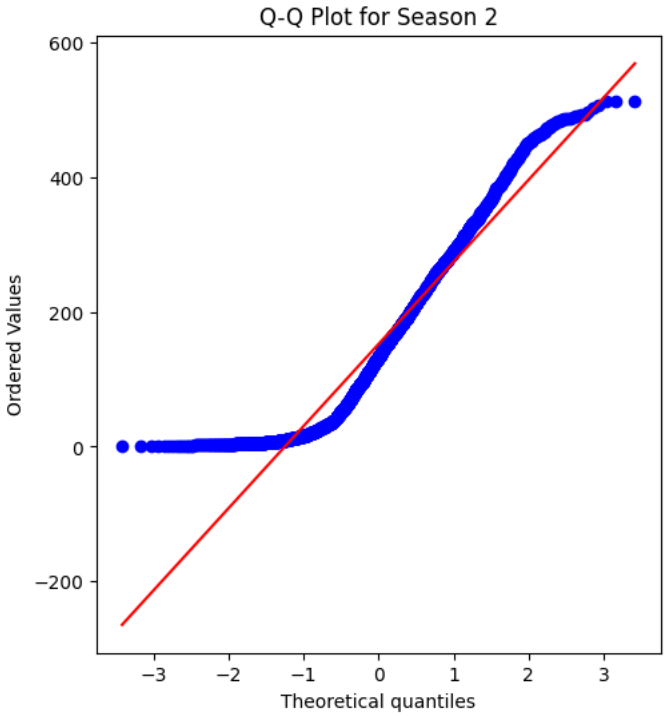
```
##Formulate Null Hypothesis (H0) and Alternate Hypothesis (Ha):
# Null Hypothesis (H0): The demand for bicycles on rent is the same for different seasons.
# Alternate Hypothesis (Ha): The demand for bicycles on rent is different for at least one season.

## Select appropriate test
# Test: One-way ANOVA test (Analysis of Variance).
import scipy.stats as stats
import seaborn as sns

seasons = df['season'].unique()

# Check assumptions
for season in seasons:
    subset = df[df['season'] == season]['count']

    # Visual analysis
    plt.figure(figsize=(12, 6))
    plt.subplot(1, 2, 1)
    sns.histplot(subset, kde=True)
    plt.title(f'Histogram for Season {season}')

    plt.subplot(1, 2, 2)
    stats.probplot(subset, plot=plt)
    plt.title(f'Q-Q Plot for Season {season}')

    plt.show()

    # Shapiro-Wilk’s test
    shapiro_test_statistic, shapiro_p_value = stats.shapiro(subset)
    print(f"Shapiro-Wilk’s test for Season {season}: Statistic={shapiro_test_statistic}, P-value={shapiro_p_value}")

# Levene’s test for equality of variances
levene_test_statistic, levene_p_value = stats.levene(
    df[df['season'] == 1]['count'],
    df[df['season'] == 2]['count'],
    df[df['season'] == 3]['count'],
    df[df['season'] == 4]['count']
)

print(f"\nLevene’s test for equality of variances: Statistic={levene_test_statistic}, P-value={levene_p_value}")

# One-way ANOVA test
anova_test_statistic, anova_p_value = stats.f_oneway(
    df[df['season'] == 1]['count'],
    df[df['season'] == 2]['count'],
    df[df['season'] == 3]['count'],
    df[df['season'] == 4]['count']
)

print(f"\nOne-way ANOVA test: Statistic={anova_test_statistic}, P-value={anova_p_value}")

## Setting significance level : alpha = 0.05
alpha = 0.05
## Decide whether to accept or reject the Null Hypothesis:
if anova_p_value<alpha:
  print("Interpretation : Reject Ho")
else:
  print("Interpretation : Fail to Reject Ho")
```



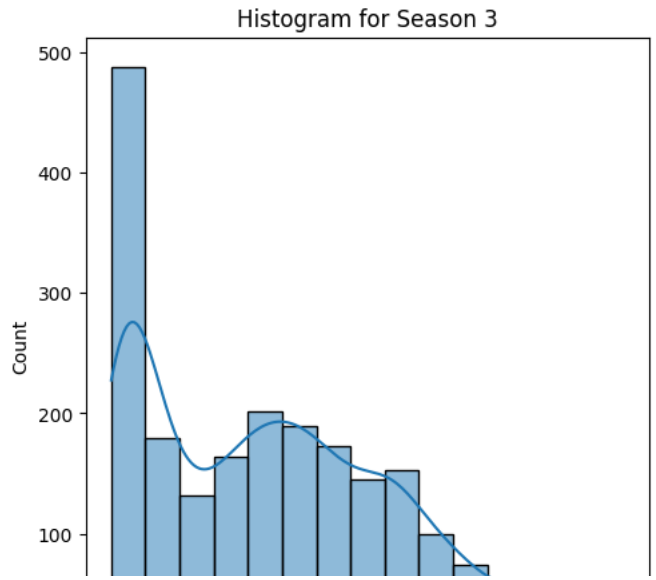


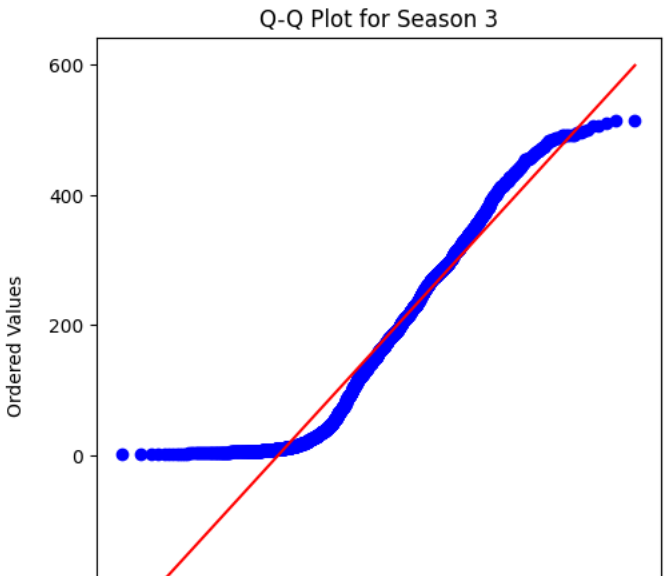

```
Shapiro-Wilk’s test for Season 1: Statistic=0.8720636367797852, P-value=2.099873774760025e-40
```

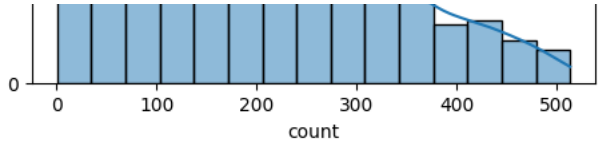


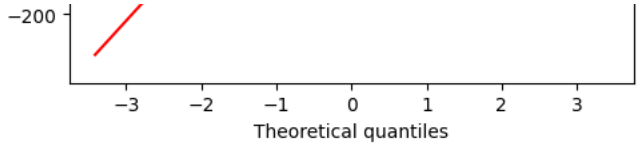



```
Shapiro-Wilk’s test for Season 2: Statistic=0.9223816394805908, P-value=3.047296012963622e-32
```

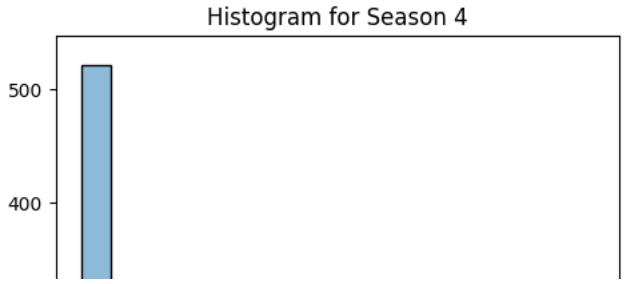


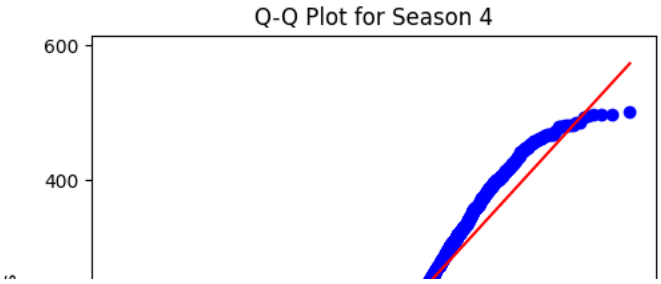







```
Shapiro-Wilk's test for Season 3: Statistic=0.9384109377861023, P-value=4.697499014315342e-29
```





Inferences & Recommendations:

1. **Normality Assumption:**
   - Shapiro-Wilk's tests for each season show extremely low p-values, indicating a significant departure from normality. The visual analysis of histograms and Q-Q plots supports this finding.
2. **Equality of Variance:**
   - Levene's test for equality of variances indicates a very low p-value, rejecting the assumption of equal variances among different seasons.
3. **ANOVA Results:**
   - The one-way ANOVA test shows an extremely significant p-value (1.016e-88), indicating that there is a significant difference in the demand for bicycles on rent among different seasons.

Recommendations:

1. **Seasonal Business Adjustments:**
   - Acknowledge the observed differences in demand among seasons and consider making seasonal adjustments to business strategies, marketing efforts, and resource allocation.
2. **Long-Term Planning:**
   - Use the insights gained from the analysis for long-term business planning, including inventory management, staffing, and promotional activities tailored to specific seasons.
3. **Customer Engagement:**
   - Explore customer preferences and behavior during different seasons to enhance engagement and cater to specific needs.
4. **Monitoring and Adaptation:**
   - Regularly monitor demand patterns and be prepared to adapt strategies based on changing seasonal dynamics.

In summary, while there is a significant difference in the demand for bicycles on rent among different seasons, it's crucial to consider practical implications and explore ways to adapt business strategies for effective seasonal management. The business may benefit from tailoring its approach to better suit the unique demands of each season.

6. Check if the Weather conditions are significantly different during different Seasons?

```
# Formulate Null Hypothesis (H0) and Alternate Hypothesis (Ha):
# Null Hypothesis (H0): Weather conditions are not significantly different during different seasons.
# Alternate Hypothesis (HA): Weather conditions are significantly different during different seasons.

# Select an appropriate test:
# Test: Chi-square test of independence.
from scipy.stats import chi2_contingency
# Create a Contingency Table
contingency_table = pd.crosstab(df['weather'], df['season'])

# Display the contingency table
print(contingency_table)

#  Set a significance level and Calculate the test Statistics / p-value:
## Setting significance level : alpha = 0.05
alpha = 0.05

# Chi-square test
```

Inferences & Recommendations:

1. **Chi-square Test Results:**
    - The Chi-square test of independence yields a highly significant p-value (1.283e-10), indicating a rejection of the null hypothesis.

Recommendations:

1. **Significant Association:**
    - The rejection of the null hypothesis suggests a significant association between weather conditions and seasons.
2. **Business Considerations:**
    - Consider the implications of the significant association for business operations. Weather conditions may impact seasonal activities, consumer behavior, and resource allocation.
3. **Adaptation Strategies:**
    - Based on the findings, develop strategies to adapt to varying weather conditions during different seasons. This may include adjusting marketing campaigns, inventory management, or service offerings.
4. **Monitoring and Adjustment:**
    - Regularly monitor the association between weather conditions and seasons, especially if external factors or climate patterns