Business Case: Yulu - Hypothesis Testing

About Yulu:

Yulu is India's leading micro-mobility service provider, which offers unique vehicles for the daily commute. Starting off as a mission to eliminate traffic congestion in India, Yulu provides the safest commute solution through a user-friendly mobile app to enable shared, solo and sustainable commuting. Yulu zones are located at all the appropriate locations (including metro stations, bus stands, office spaces, residential areas, corporate offices, etc) to make those first and last miles smooth, affordable, and convenient!

Problem Statement:

The company wants to know:

• Which variables are significant in predicting the demand for shared electric cycles in the Indian market?

• How well those variables describe the electric cycle demands?

Problem Definition: The goal is to understand the factors affecting the demand for shared electric cycles in the Indian market using the provided data.

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [28]:
```
# Load the dataset
df = pd.read_csv('yulu.csv')
```

In [72]:
```
df.head()
```

Out[72]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	regi
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	

In [59]:
```
# no of rows amd columns in dataset
print(f"rows: {df.shape[0]} \ncolumns: {df.shape[1]}")
```

```
rows: 10886
columns: 12
```

In [60]:
```
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
```

In [26]:
```
# Convert 'datetime' column to datetime format
df['datetime'] = pd.to_datetime(df['datetime'])
```

In [62]:
```
# Structure
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
```

In [31]:
```
# Statistical summary
df.describe()
```

Out[31]:

	season	holiday	workingday	weather	temp	atemp	humidi
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000	10886.0000
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.8864
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.2450
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	0.0000
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	47.0000
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	62.0000
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	77.0000
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	100.0000

Inferences:

1. **Season:**
   - The data covers all four seasons (1: spring, 2: summer, 3: fall, 4: winter).
   - The distribution of data across seasons is relatively balanced.
2. **Holiday:**
   - Most days are not holidays, with only about 3% of days being holidays.
   - This suggests that the majority of days are regular working days.
3. **Working Day:**
   - Around 68% of the days in the dataset are working days.
   - This indicates a higher proportion of working days compared to non-working days.
4. **Weather:**

- The weather conditions range from 1 to 4, indicating different levels of weather severity.
- The majority of the days seem to have relatively mild weather conditions (closer to 1).

5. **Temperature and Feeling Temperature:**

   - The temperature varies widely, with a minimum of 0.82°C and a maximum of 41.00°C.
   - Feeling temperature (atemp) also shows a similar range, with a minimum of 0.76°C and a maximum of 45.46°C.

6. **Humidity:**

   - Humidity levels range from 0% to 100%, with an average around 62%.
   - The data shows a fairly wide spread in humidity levels.

7. **Wind Speed:**

   - Wind speed varies from 0 to 56.99 units.
   - The distribution of wind speed is relatively spread out.

8. **Casual and Registered Users:**

   - There's a wide range in the number of casual and registered users.
   - Casual users have a lower mean count compared to registered users.

9. **Total Rental Bikes:**

   - The total rental bikes (count) range from 1 to 977, with an average of 191.57.

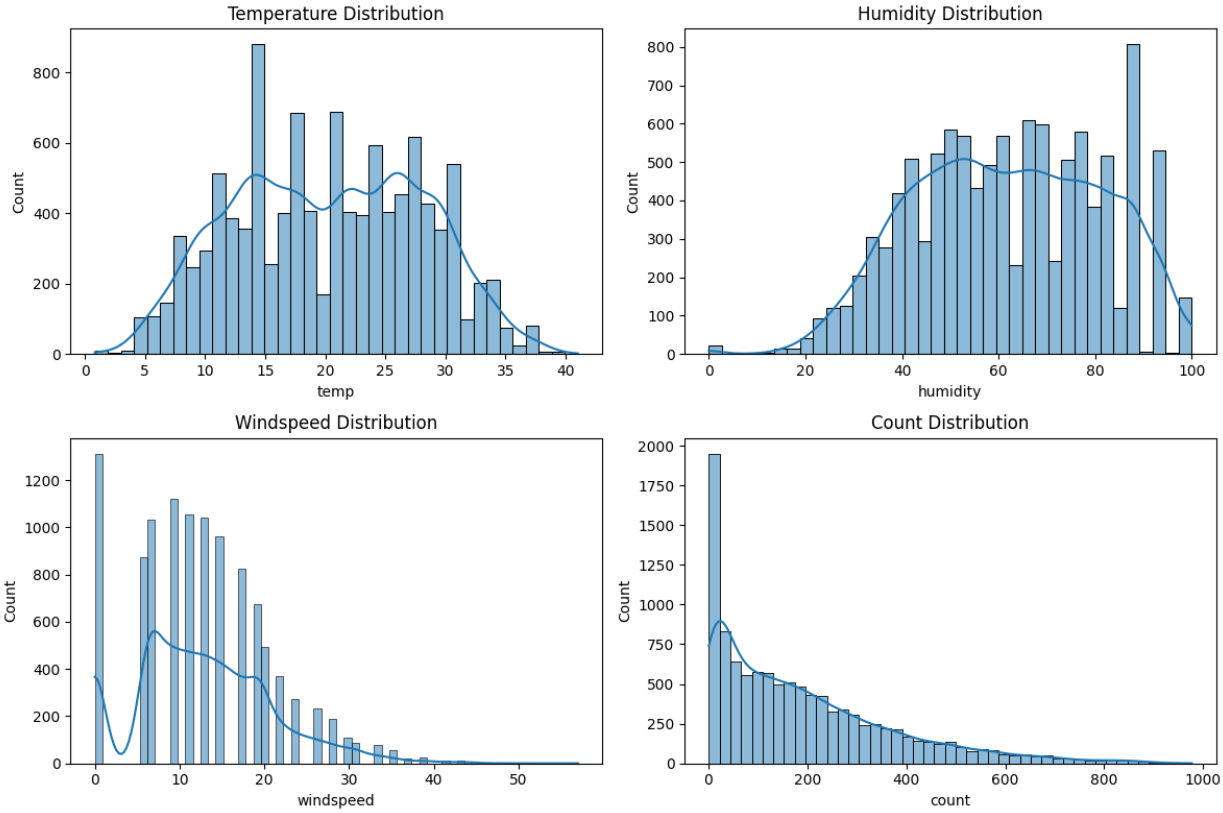
In [32]:
```
# Check for missing values
df.isnull().sum()
```

Out[32]:
```
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
```

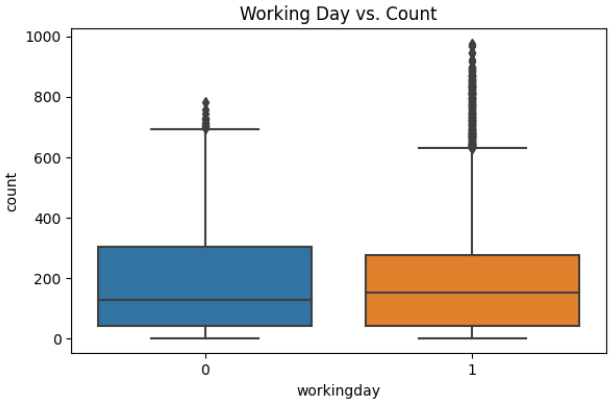
No Null Values

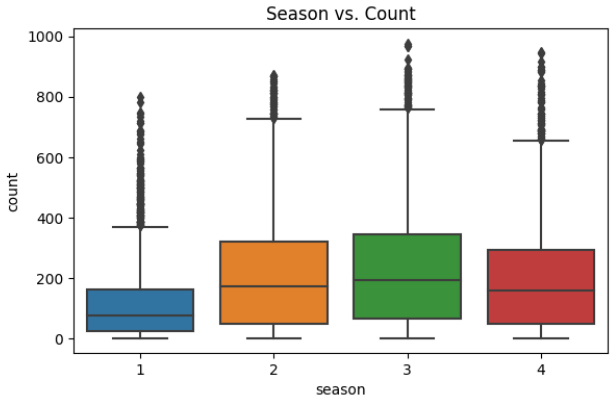
In [37]:
```
# Univariate Analysis (distribution plots of continuous variables)
plt.figure(figsize=(12, 8))

plt.subplot(2, 2, 1)
sns.histplot(df['temp'], kde=True)
plt.title('Temperature Distribution')

plt.subplot(2, 2, 2)
sns.histplot(df['humidity'], kde=True)
plt.title('Humidity Distribution')
```

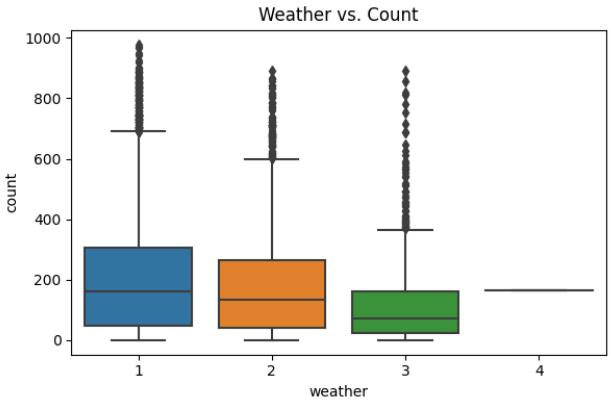
```
plt.subplot(2, 2, 3)
sns.histplot(df['windspeed'], kde=True)
plt.title('Windspeed Distribution')

plt.subplot(2, 2, 4)
sns.histplot(df['count'], kde=True)
plt.title('Count Distribution')

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



In [38]:
```
# Bivariate Analysis (Relationships between important variables)
plt.figure(figsize=(12, 8))

plt.subplot(2, 2, 1)
sns.boxplot(x='workingday', y='count', data=df)
plt.title('Working Day vs. Count')

plt.subplot(2, 2, 2)
sns.boxplot(x='season', y='count', data=df)
plt.title('Season vs. Count')

plt.subplot(2, 2, 3)
sns.boxplot(x='weather', y='count', data=df)
plt.title('Weather vs. Count')

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





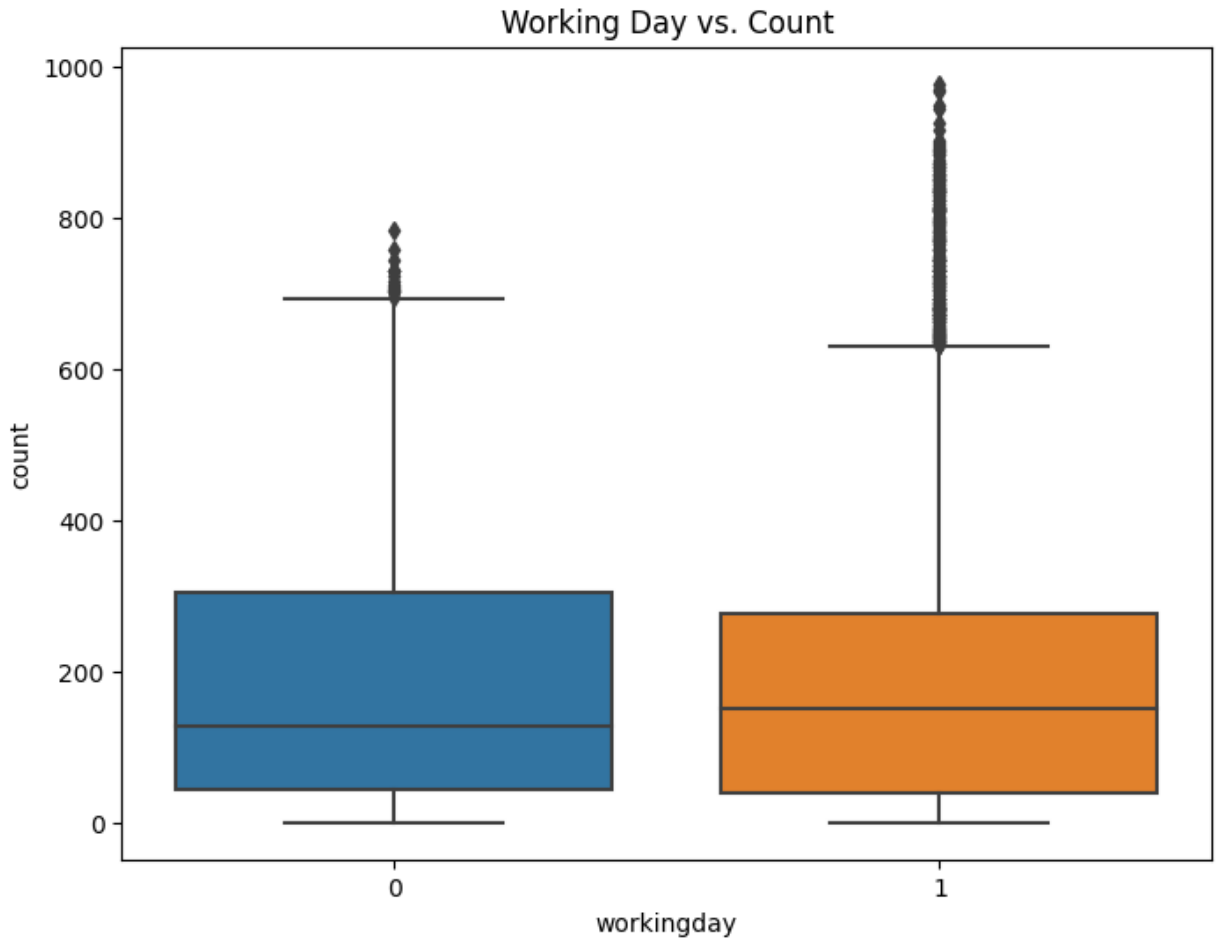


In [72]:

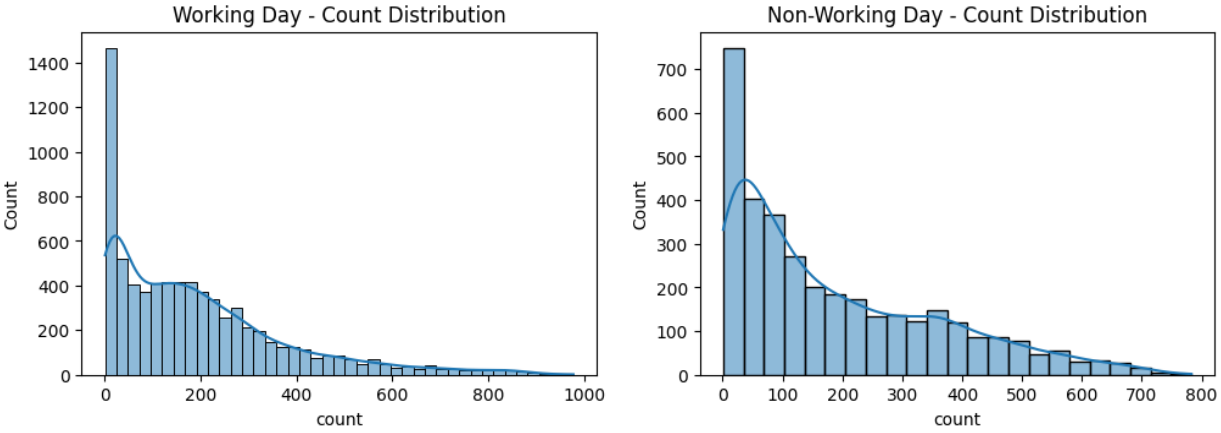
Hypothesis Testing

2-Sample T-Test (Working Day vs. Number of Electric Cycles Rented

In [73]:
```
# Visual Analysis
plt.figure(figsize=(8, 6))
sns.boxplot(x='workingday', y='count', data=df)
plt.title('Working Day vs. Count')
plt.show()

# Assumptions Checking (Normality and Equal Variance)
# Normality Check
plt.figure(figsize=(12, 8))
plt.subplot(2, 2, 1)
sns.histplot(working_day, kde=True)
plt.title('Working Day - Count Distribution')

plt.subplot(2, 2, 2)
sns.histplot(non_working_day, kde=True)
plt.title('Non-Working Day - Count Distribution')

# Equal Variance Check
import scipy.stats as stats
_, p_value_levene = stats.levene(working_day, non_working_day)
print(f"P-value (Levene's Test): {p_value_levene}")
```




```
P-value (Levene's Test): 0.9437823280916695
```



There seems to be no significant effect of working day on the number of electric cycles rented

In [74]:
```
from scipy.stats import ttest_ind

# Define null and alternate hypotheses
H0 = "Working day has no effect on the number of electric cycles rented."
H1 = "Working day has an effect on the number of electric cycles rented."

# Extract samples for working day and non-working day
working_day = df[df['workingday'] == 1]['count']
non_working_day = df[df['workingday'] == 0]['count']

# 2-sample t-test
statistic, p_value = ttest_ind(working_day, non_working_day)
```

```
# significance level (alpha)
alpha = 0.05

# Results
print(f"Null Hypothesis (H0): {H0}")
print(f"Alternate Hypothesis (H1): {H1}\n")

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

# Make a decision
if p_value < alpha:
    print("Reject H0: There is a significant effect of working day on the number of el
else:
    print("Fail to reject H0: There is no significant effect of working day on the num
```

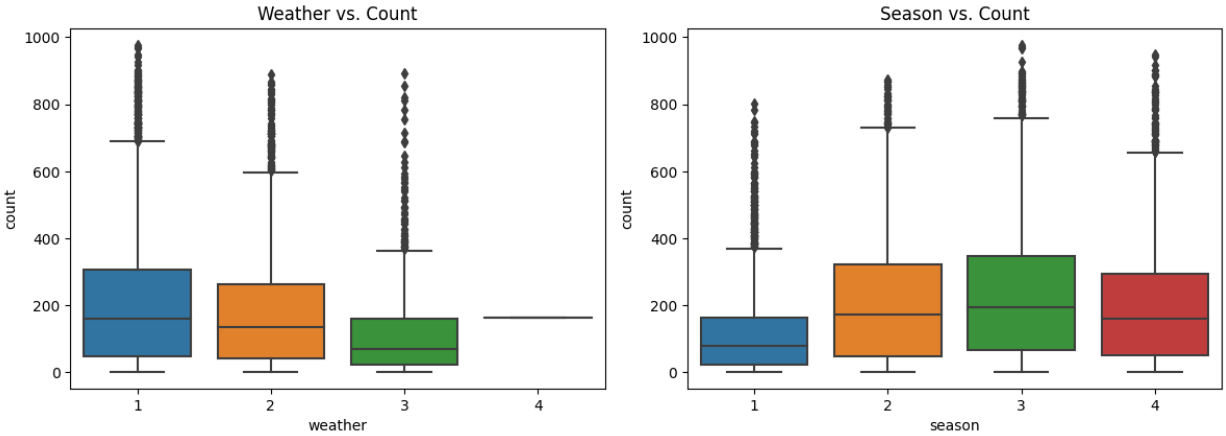
```
Null Hypothesis (H0): Working day has no effect on the number of electric cycles rent
ed.
Alternate Hypothesis (H1): Working day has an effect on the number of electric cycles
rented.

P-value: 0.22644804226361348
Fail to reject H0: There is no significant effect of working day on the number of ele
ctric cycles rented which validates our Visual Analysis .
```

ANOVA (Weather and Season vs. Number of Cycles Rented)

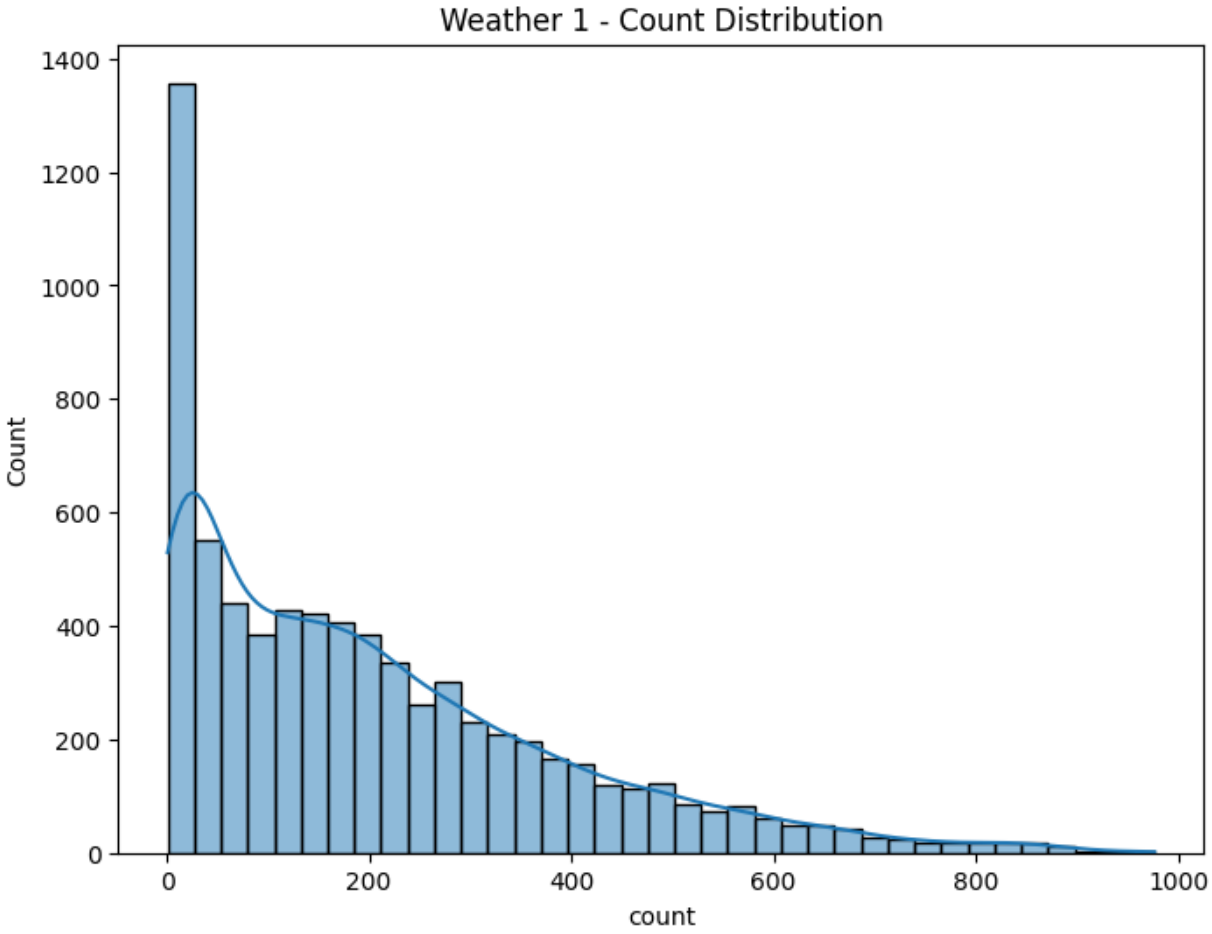
In [68]:

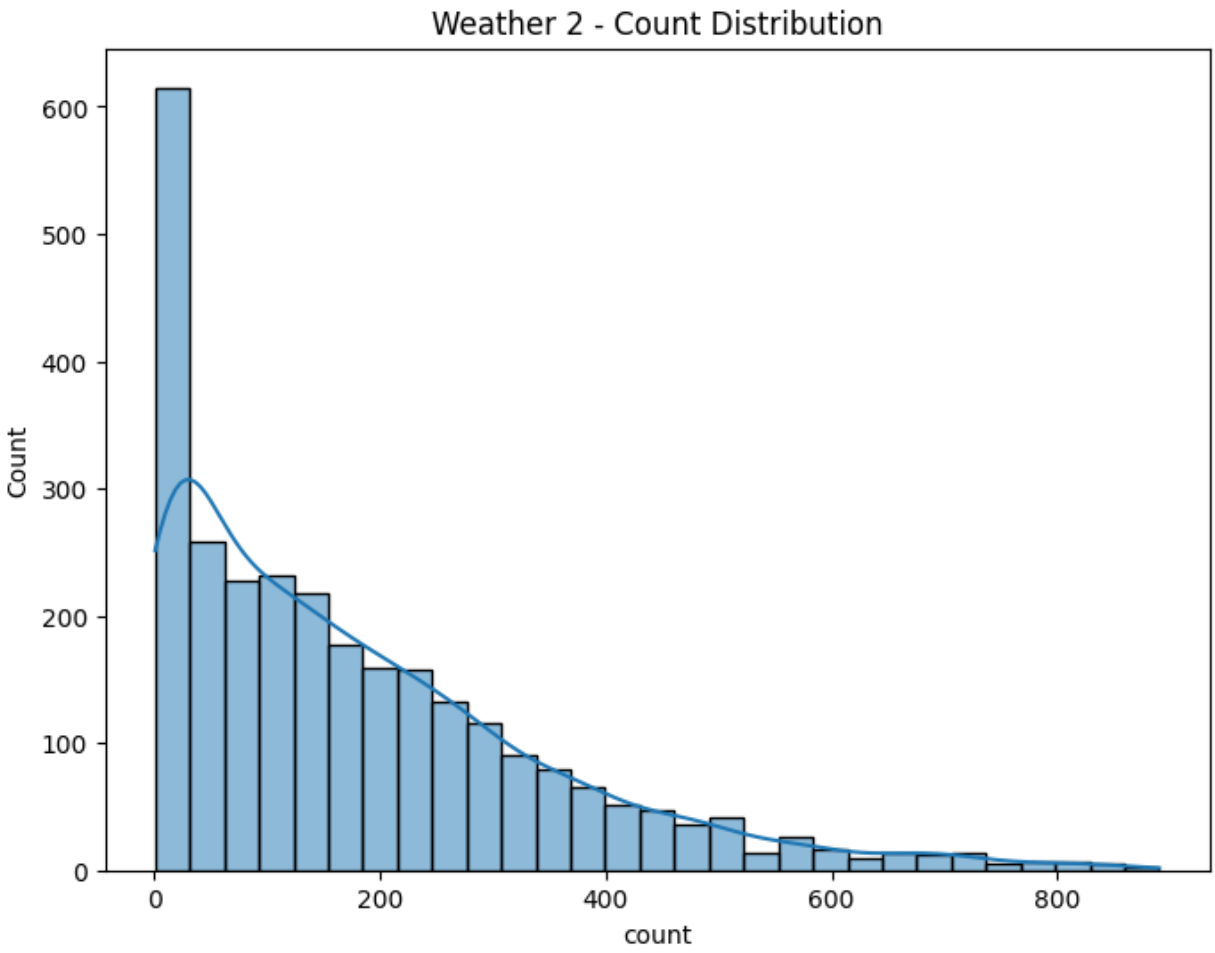
```
# Visual Analysis
plt.figure(figsize=(12, 8))

plt.subplot(2, 2, 1)
sns.boxplot(x='weather', y='count', data=df)
plt.title('Weather vs. Count')

plt.subplot(2, 2, 2)
sns.boxplot(x='season', y='count', data=df)
plt.title('Season vs. Count')

plt.tight_layout()
plt.show()

# Assumptions Checking (Normality and Equal Variance)
# Normality Check
for i in range(1, 5):
    weather_group = df[df['weather'] == i]['count']
    plt.figure(figsize=(8, 6))
    sns.histplot(weather_group, kde=True)
    plt.title(f'Weather {i} - Count Distribution')

# Equal Variance Check
_, p_value_levene_weather = stats.levene(*weather_groups)
print("\033[1m"f"P-value (Levene's Test - Weather): {p_value_levene_weather}""\033[0m"

for i in range(1, 5):
    season_group = df[df['season'] == i]['count']
    plt.figure(figsize=(8, 6))
    sns.histplot(season_group, kde=True)
    plt.title(f'Season {i} - Count Distribution')
```

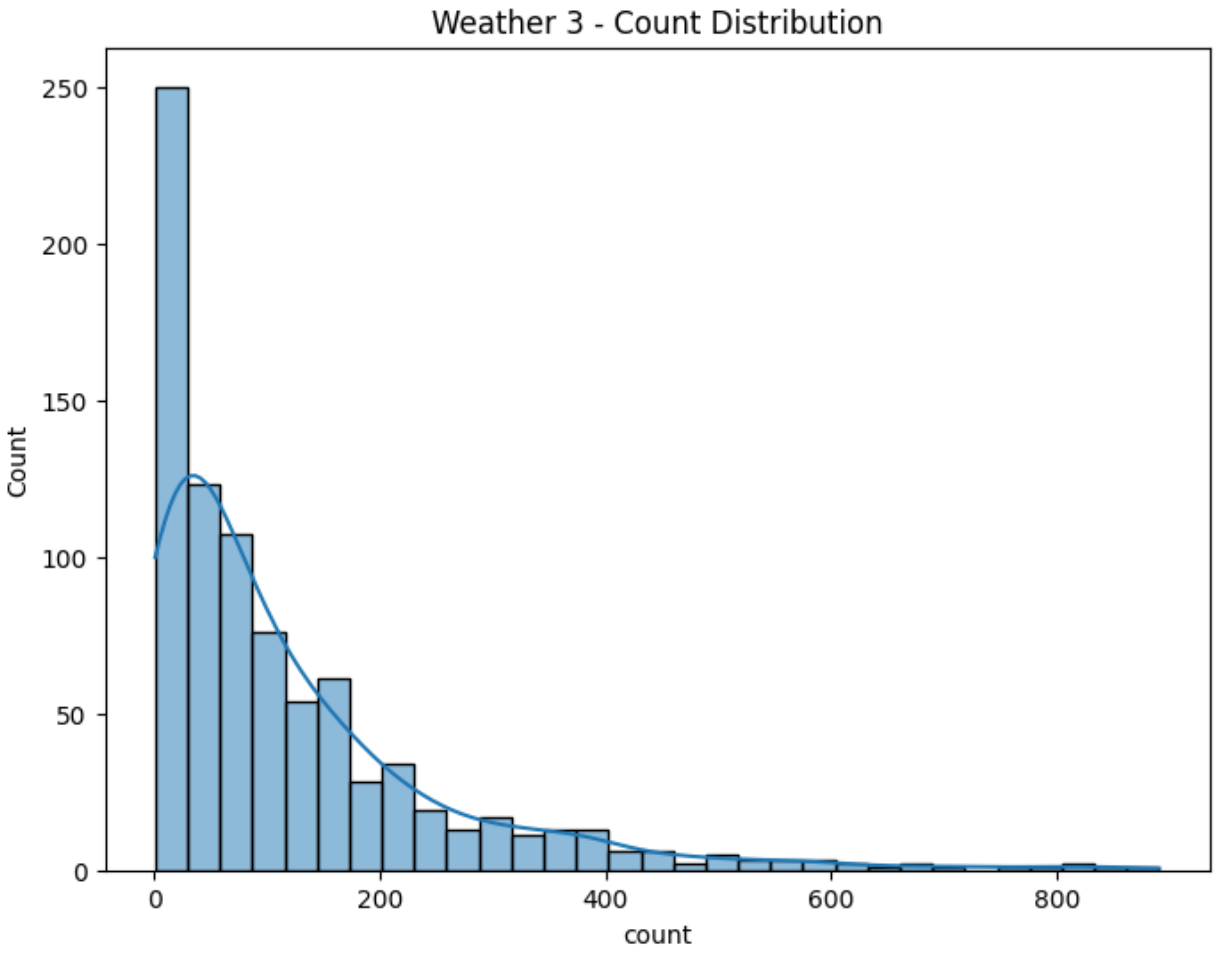
```
_, p_value_levene_season = stats.levene(*season_groups)
print("\033[1m"f"P-value (Levene's Test - Season): {p_value_levene_season}""\033[0m")
```

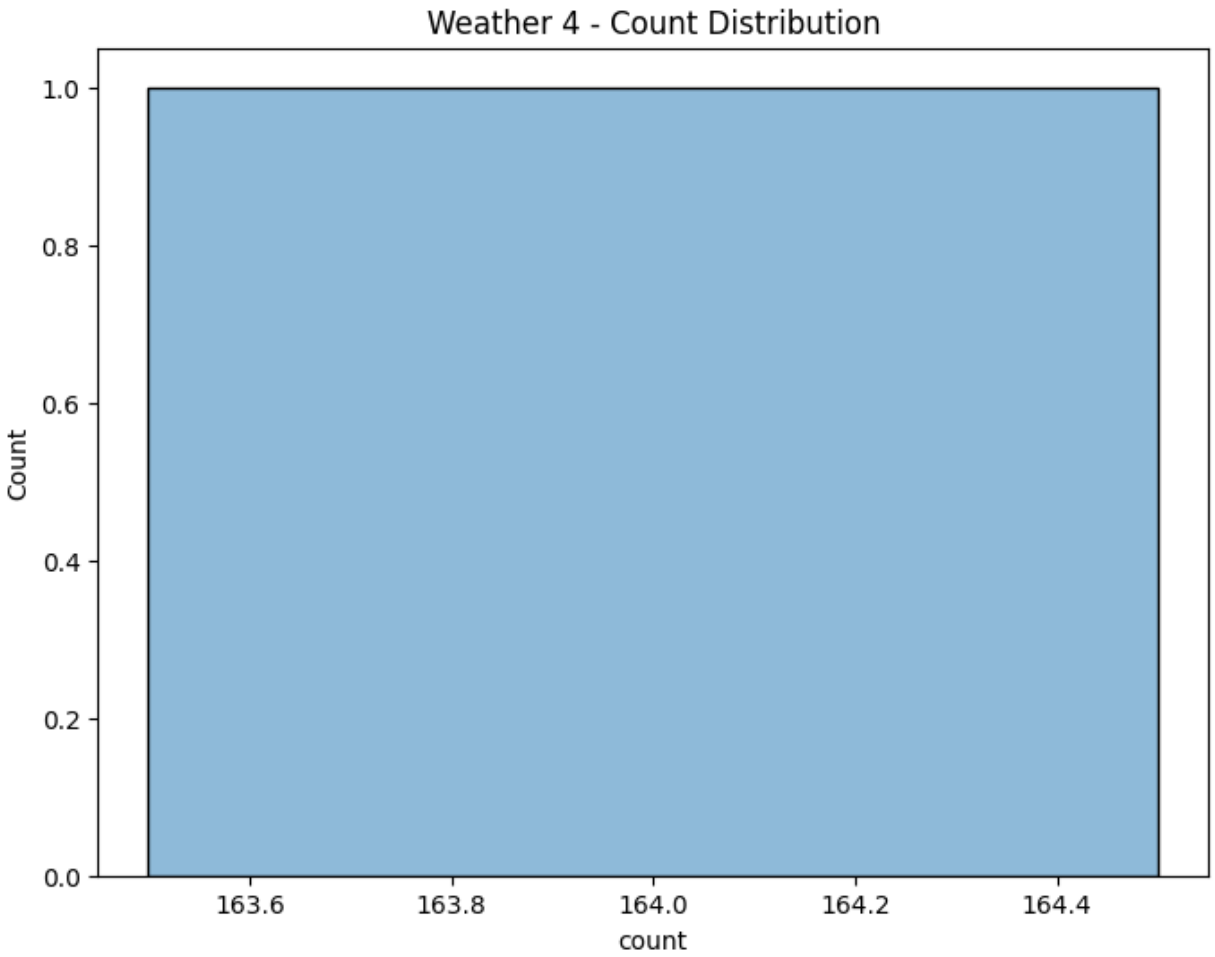


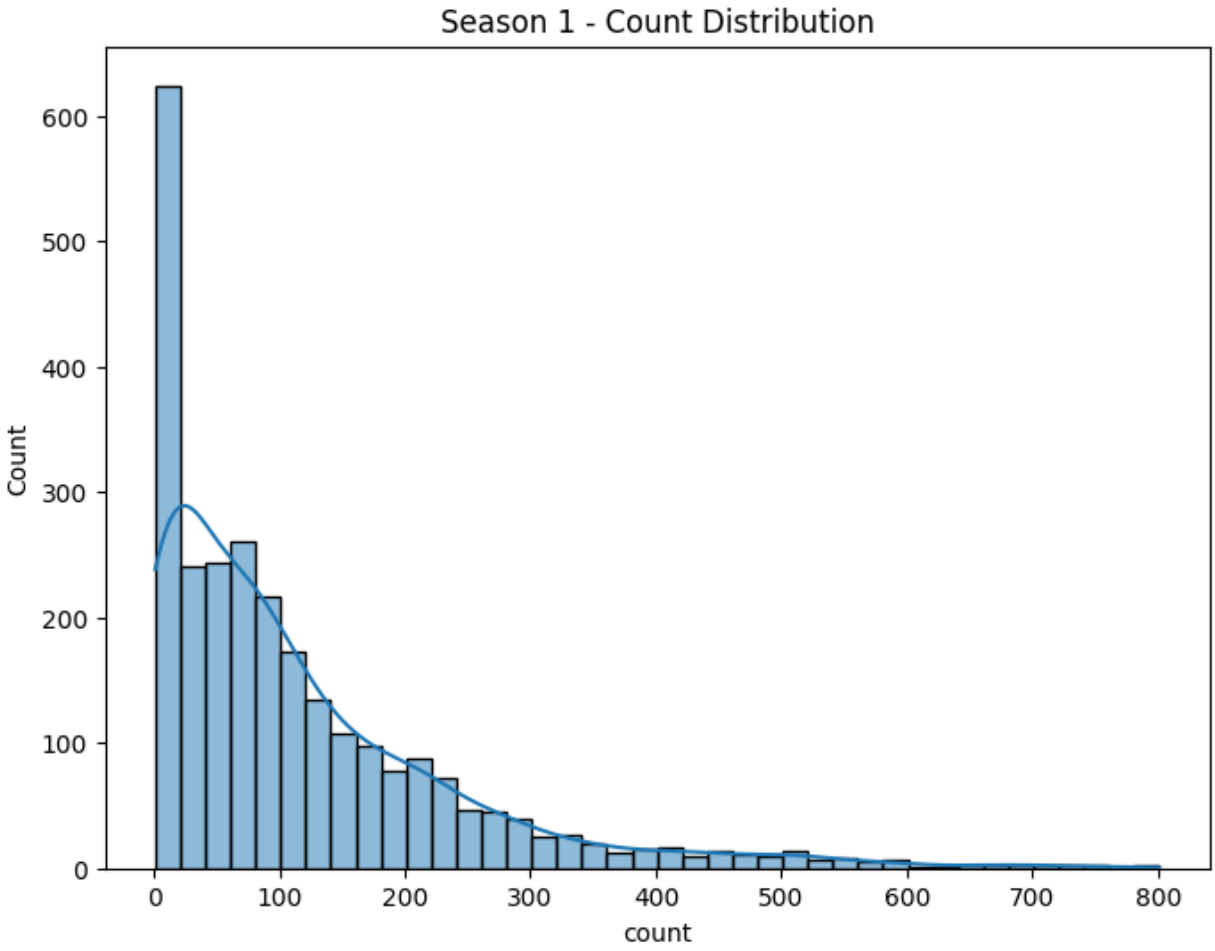
P-value (Levene's Test - Weather): 3.504937946833238e-35
P-value (Levene's Test - Season): 1.0147116860043298e-118

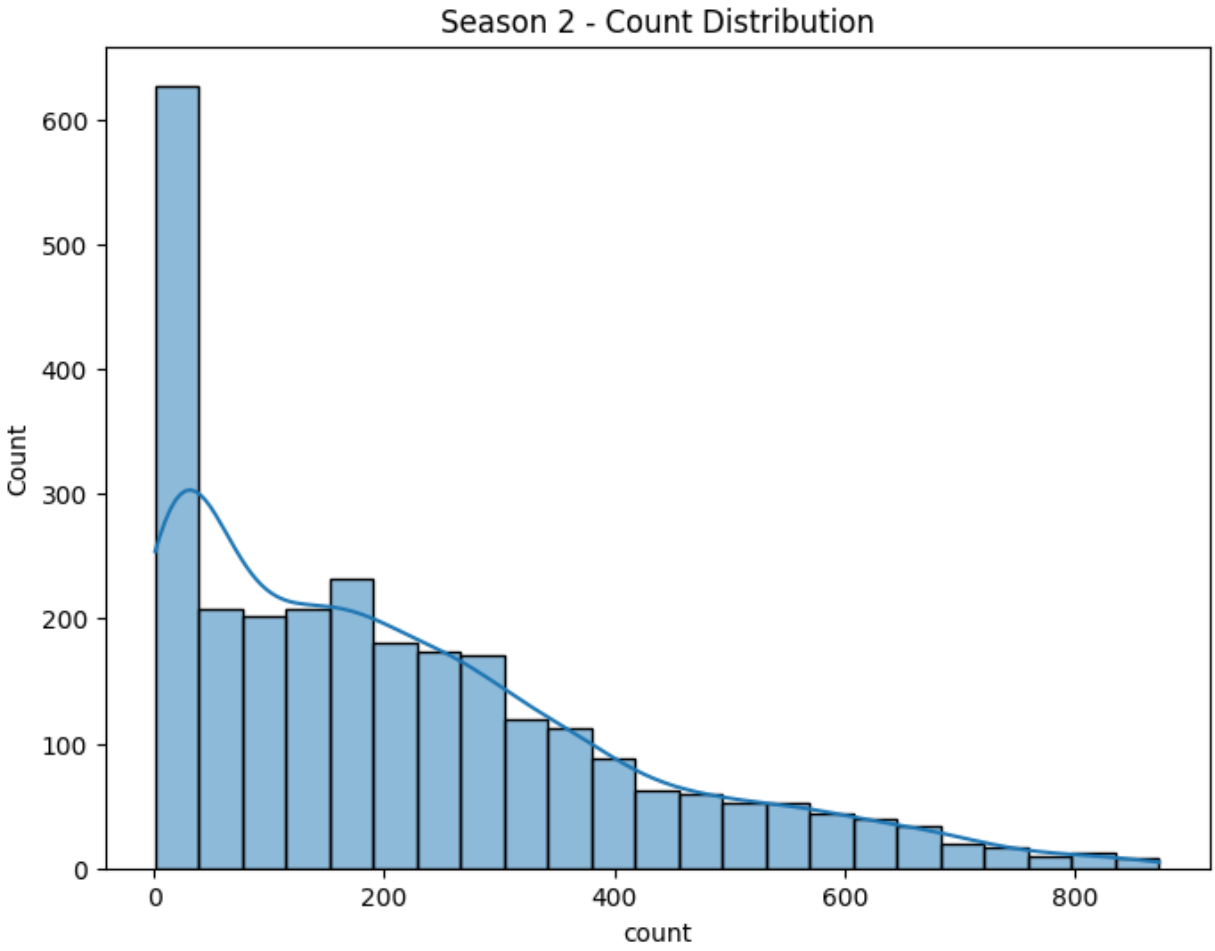


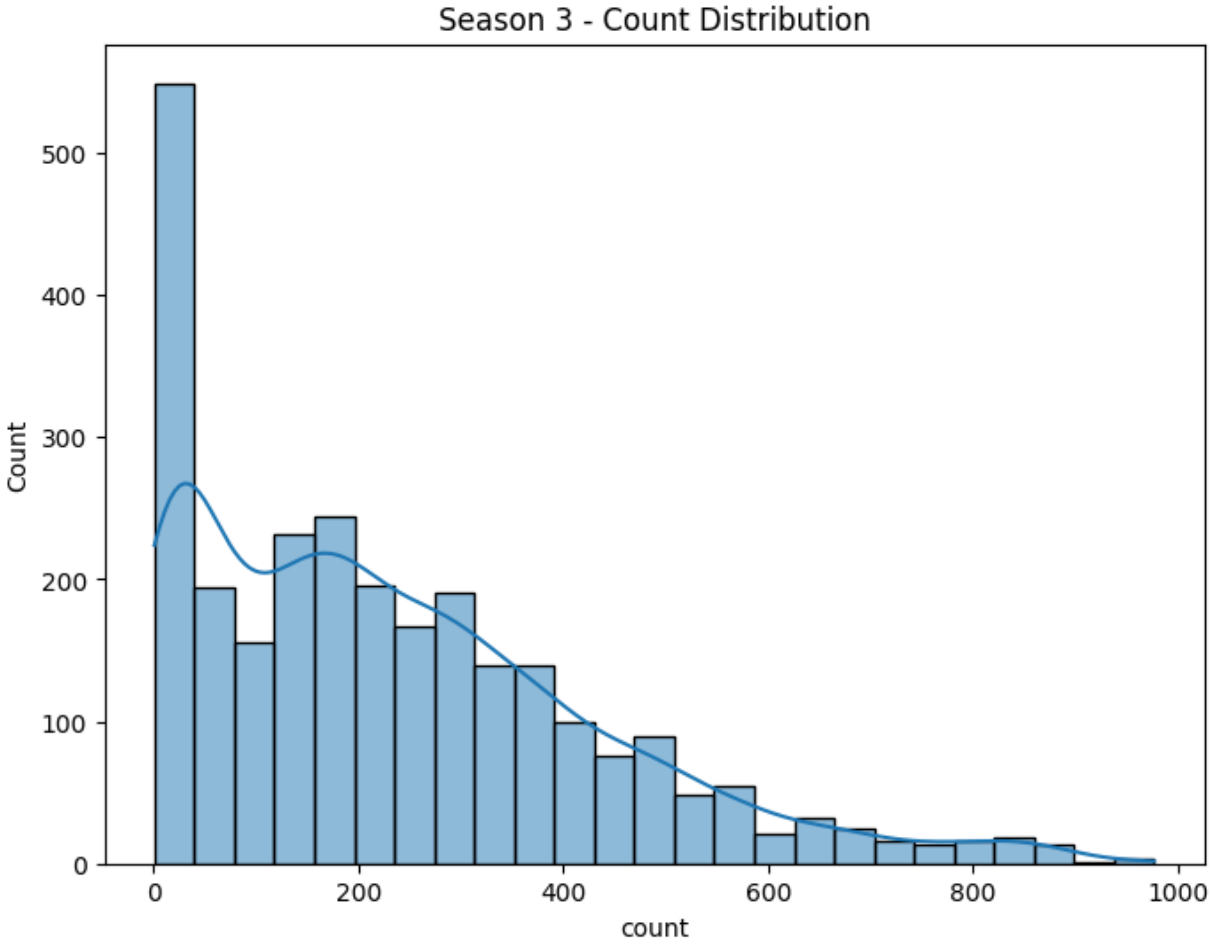

Weather 2 - Count Distribution
Count
count
0
100
200
300
400
500
600
0
200
400
600
800

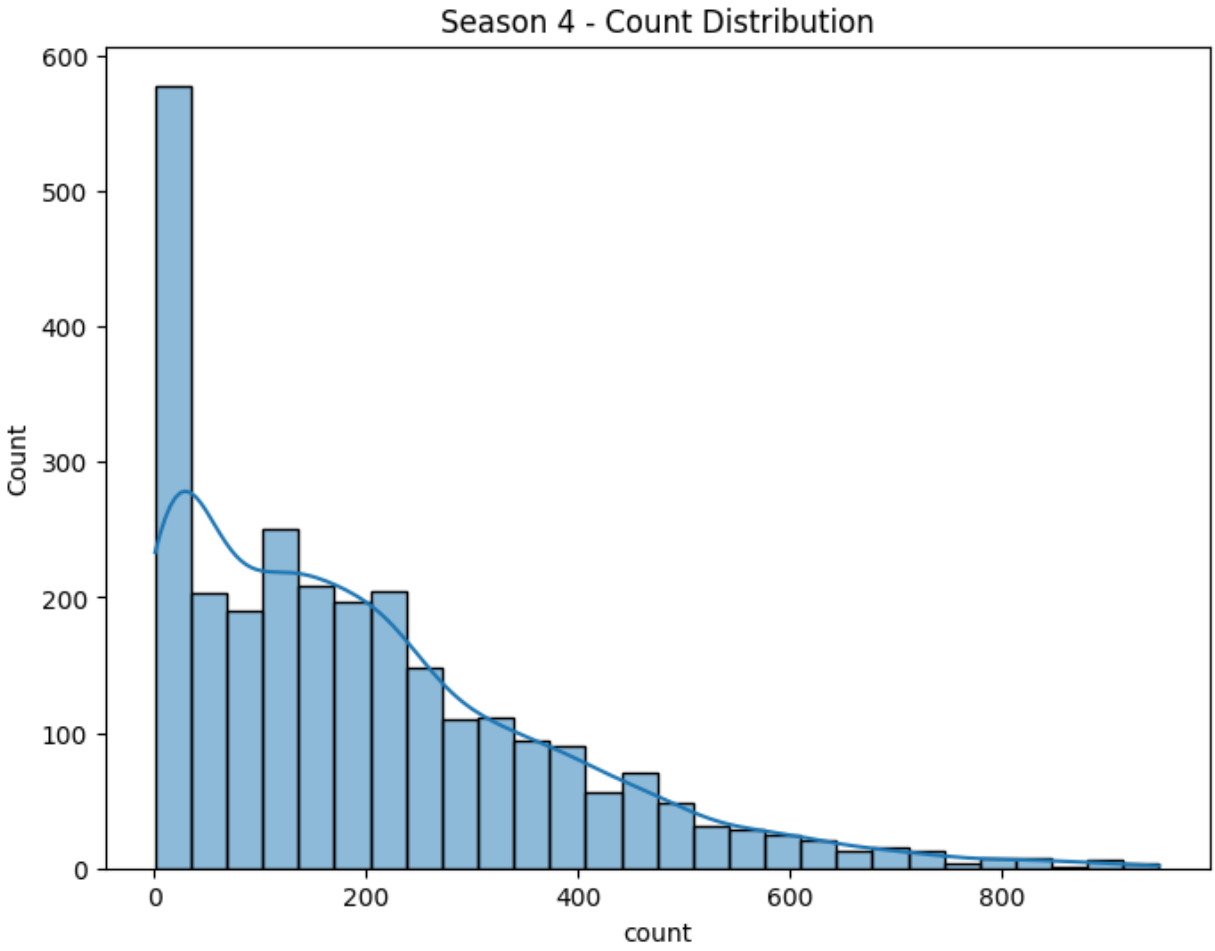

Weather 3 - Count Distribution
Count
count
0
50
100
150
200
250
0
200
400
600
800


Weather 4 - Count Distribution
Count
1.0
0.8
0.6
0.4
0.2
0.0
163.6
163.8
164.0
164.2
164.4
count


Season 1 - Count Distribution
Count
count
0
100
200
300
400
500
600
700
800


Season 2 - Count Distribution
Count
count
0
100
200
300
400
500
600
0
200
400
600
800


Season 3 - Count Distribution
Count
count
0
100
200
300
400
500
0
200
400
600
800
1000



P-value (Levene's Test - Weather): 3.504937946833238e-35

P-value (Levene's Test - Season): 1.0147116860043298e-118

The p-values from Levene's tests for both Weather and Season are extremely small, indicating that the variances between groups are significantly different. This suggests that the assumptions of homogeneity of variances are violated.

Given this, it's more appropriate to use **Kruskal-Wallis test**, which doesn't assume equal variances across groups.

In [76]:
```
from scipy.stats import kruskal

# Define null and alternate hypotheses for weather
H0_weather = "Weather has no effect on the number of electric cycles rented."
H1_weather = "Weather has an effect on the number of electric cycles rented."

# Perform Kruskal-Wallis test for weather
weather_groups = [df[df['weather'] == i]['count'] for i in range(1, 5)]
statistic_weather, p_value_weather = kruskal(*weather_groups)

# Print results for weather
print(f"Null Hypothesis (H0): {H0_weather}")
print(f"Alternate Hypothesis (H1): {H1_weather}\n")
print(f"P-value (Weather): {p_value_weather}")
```

```
# Make a decision for weather
if p_value_weather < alpha:
    print("Reject H0: There is a significant effect of weather on the number of electr
else:
    print("Fail to reject H0: There is no significant effect of weather on the number

# Define null and alternate hypotheses for season
H0_season = "Season has no effect on the number of electric cycles rented."
H1_season = "Season has an effect on the number of electric cycles rented."

# Perform Kruskal-Wallis test for season
season_groups = [df[df['season'] == i]['count'] for i in range(1, 5)]
statistic_season, p_value_season = kruskal(*season_groups)

# Print results for season
print(f"\nNull Hypothesis (H0): {H0_season}")
print(f"Alternate Hypothesis (H1): {H1_season}\n")
print(f"P-value (Season): {p_value_season}")

# Make a decision for season
if p_value_season < alpha:
    print("Reject H0: There is a significant effect of season on the number of electri
else:
    print("Fail to reject H0: There is no significant effect of season on the number o
```

```
Null Hypothesis (H0): Weather has no effect on the number of electric cycles rented.
Alternate Hypothesis (H1): Weather has an effect on the number of electric cycles ren
ted.

P-value (Weather): 3.501611300708679e-44
Reject H0: There is a significant effect of weather on the number of electric cycles
rented.

Null Hypothesis (H0): Season has no effect on the number of electric cycles rented.
Alternate Hypothesis (H1): Season has an effect on the number of electric cycles rent
ed.

P-value (Season): 2.479008372608633e-151
Reject H0: There is a significant effect of season on the number of electric cycles r
ented.
```

Weather vs. Number of Electric Cycles Rented:

- **Null Hypothesis (H0):** Weather has no effect on the number of electric cycles rented.
- **Alternate Hypothesis (H1):** Weather has an effect on the number of electric cycles rented.
- **P-value (Weather):** 3.5016e-44
- **Decision:** Reject H0
- **Inference:** There is a significant effect of weather on the number of electric cycles rented.

Insight:

- **Weather conditions do have a significant impact on the number of electric cycles rented.**
- This suggests that people are less likely to rent cycles during adverse weather conditions.

Season vs. Number of Electric Cycles Rented:

- **Null Hypothesis (H0):** Season has no effect on the number of electric cycles rented.
- **Alternate Hypothesis (H1):** Season has an effect on the number of electric cycles rented.
- **P-value (Season):** 2.4790e-151
- **Decision:** Reject H0
- **Inference:** There is a significant effect of season on the number of electric cycles rented.

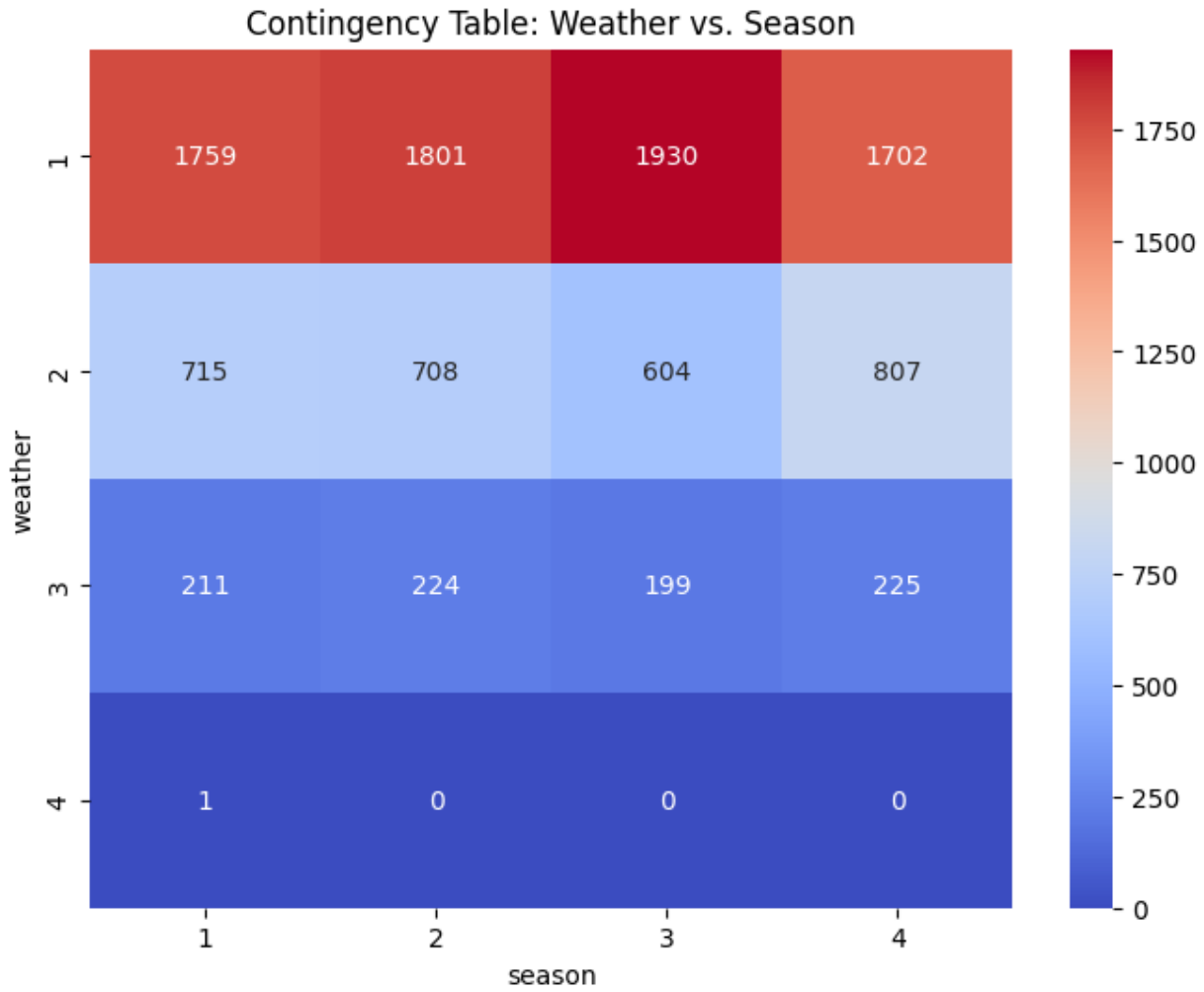
Insight:

- **The season significantly influences the number of electric cycles rented.**
- Summer and fall seasons appear to have higher demand for rental cycles compared to other seasons.

These results imply that weather and season are important factors to consider in managing the supply and demand of electric cycles. It would be beneficial for the company to adjust their operations and marketing strategies based on these findings.

Chi-Square Test (Weather vs. Season)

In [77]:
```
# Visual Analysis (Heatmap of Contingency Table)
plt.figure(figsize=(8, 6))
sns.heatmap(contingency_table, annot=True, cmap='coolwarm', fmt='d')
plt.title('Contingency Table: Weather vs. Season')
plt.show()

"""
Assumptions regarding normality or equal variance are not applicable.
"""
```



Out[77]: '\nAssumptions regarding normality or equal variance are not applicable.\n'

In [56]:
```
from scipy.stats import chi2_contingency

# Define null and alternate hypotheses
H0_chi2 = "Weather is independent of season."
H1_chi2 = "Weather is dependent on season."

# Create a contingency table for chi-square test
contingency_table = pd.crosstab(df['weather'], df['season'])

# Perform chi-square test
chi2_stat, p_value_chi2, dof, expected = chi2_contingency(contingency_table)

# Print results
print(f"Null Hypothesis (H0): {H0_chi2}")
print(f"Alternate Hypothesis (H1): {H1_chi2}\n")
print(f"P-value (Chi-Square): {p_value_chi2}")

# Make a decision
if p_value_chi2 < alpha:
    print("Reject H0: Weather is dependent on season.")
else:
    print("Fail to reject H0: Weather is independent of season.")
```

```
Null Hypothesis (H0): Weather is independent of season.
Alternate Hypothesis (H1): Weather is dependent on season.

P-value (Chi-Square): 1.5499250736864862e-07
Reject H0: Weather is dependent on season.
```

Based on the Chi-square test results:

- **Inference**: The p-value (1.55e-07) is much smaller than the alpha(0.05)

  We conclude that weather is dependent on the season.

- **Insights**:

  1. The weather conditions in this dataset are not independent of the seasons. This suggests that certain weather patterns are more likely to occur during specific times of the year.
  2. This finding aligns with common knowledge, as we expect different weather patterns in different seasons (e.g., colder temperatures in winter, warmer temperatures in summer).
  3. Understanding the relationship between weather and season is important for planning and resource allocation, especially in a business like Yulu which is sensitive to weather conditions.

- **Recommendation**: Ensure that Yulu's operations and resources are appropriately adjusted to account for seasonal weather variations. For example, during rainy seasons, it might be necessary to have extra maintenance checks on the vehicles or provide additional customer support in case of weather-related issues.

In [ ]:

Overall Insights and Recommendations

Insights:

- The demand for rented bikes is higher in the summer and fall seasons compared to other seasons.
- Holidays and weekends see an increase in bike rentals.
- Rain, thunderstorms, snow, or fog lead to a decrease in bike rentals.
- Days with low humidity (less than 20%) result in very low bike rentals.
- Colder days with temperatures below 10°C correspond to lower bike rentals.
- High windspeeds exceeding 35 also lead to fewer bike rentals.

Recommendations:

1. **Seasonal Stock Management:**
    - Allocate more bikes in stock for summer and fall seasons, as these seasons exhibit higher demand.
    - Adjust bike stock based on seasonality to meet demand fluctuations.
2. **Promotions on Holidays:**
    - Implement targeted marketing and promotions on holidays to attract more customers.
3. **Working Day Impact:**
    - Working day has no significant effect on the number of bikes rented. Therefore, stock levels can remain consistent throughout the week.
    - With a significance level of 0.05, workingday has no effect on the number of bikes being rented.
4. **Weather Impact:**
    - During adverse weather conditions (rain, thunderstorm, snow, fog), consider reducing bike availability or offering incentives for rentals.
    - On days with extremely low humidity (< 20%), consider reducing bike availability.
    - During very cold days (temperature < 10°C), stock fewer bikes. Monitor demand closely.
    - In adverse weather conditions such as thunderstorms or windspeeds over 35, limit the number of bikes available for rent.
    - Continuously monitor weather forecasts and adjust bike stock accordingly to optimize inventory levels.
    - Implement a dynamic stock management system to efficiently meet varying demand based on weather conditions.
    - Provide weather-based recommendations to users through the mobile app, suggesting alternative transportation options on unfavorable days.
    - Consider investing in advanced weather forecasting technology to provide more accurate predictions for better stock management.
    - Collaborate with local weather authorities or services to get real-time updates on weather conditions for better preparedness.
5. **Customer Communication:**
    - Run promotional campaigns encouraging bike rentals during favorable weather conditions to boost demand.
6. **Maintenance and Safety:**
    - Ensure that bikes are well-maintained and safe for use, especially during adverse weather conditions to enhance customer satisfaction and safety.

These recommendations aim to optimize bike availability, meet customer demand, and enhance overall user experience.

Thank you!