

YULU DATA ANALYSIS – BY SUHASINI MOTTANNAVAR

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

```
! wget "https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/original/bike_sharing.csv?1642089089"
```

```
--2025-04-02 11:34:05-- https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/original/bike_sharing.csv?1642089089
Resolving d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)... 18.164.173.110, 18.164.173.58, 18.164.173.117, ...
Connecting to d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)|18.164.173.110|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 648353 (633K) [text/plain]
Saving to: 'bike_sharing.csv?1642089089'

bike_sharing.csv?16 100%[=====] 633.16K --.-KB/s in 0.04s

2025-04-02 11:34:05 (15.7 MB/s) - 'bike_sharing.csv?1642089089' saved [648353/648353]
```

```
df = pd.read_csv("bike_sharing.csv?1642089089")
```

```
df.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:

[Generate code with df](#)

[View recommended plots](#)

[New interactive sheet](#)

```
df.shape
```

```
(10886, 12)
```

```
df.dtypes
```


	0
datetime	object
season	int64
holiday	int64
workingday	int64
weather	int64
temp	float64
atemp	float64
humidity	int64
windspeed	float64
casual	int64
registered	int64
count	int64
dtype:	object

```
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	atemp	humidity	windspeed	casual	registered	
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.886460	12.799395	36.021955	155.552177	191.552177
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.245033	8.164537	49.960477	151.039033	181.039033
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	47.000000	7.001500	4.000000	36.000000	42.000000
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	62.000000	12.998000	17.000000	118.000000	145.000000
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	77.000000	16.997900	49.000000	222.000000	284.000000
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	100.000000	56.996900	367.000000	886.000000	977.000000

```
# Check for missing values in the dataset
missing_values = df.isnull().sum()
```

```
# Display columns with missing values
missing_values[missing_values > 0]
```




```
0
```

```
dtype: int64
```

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

```
# Remove duplicates if any exist
df = df.drop_duplicates()
```

```
# Display the count of removed duplicates
duplicate_records
```

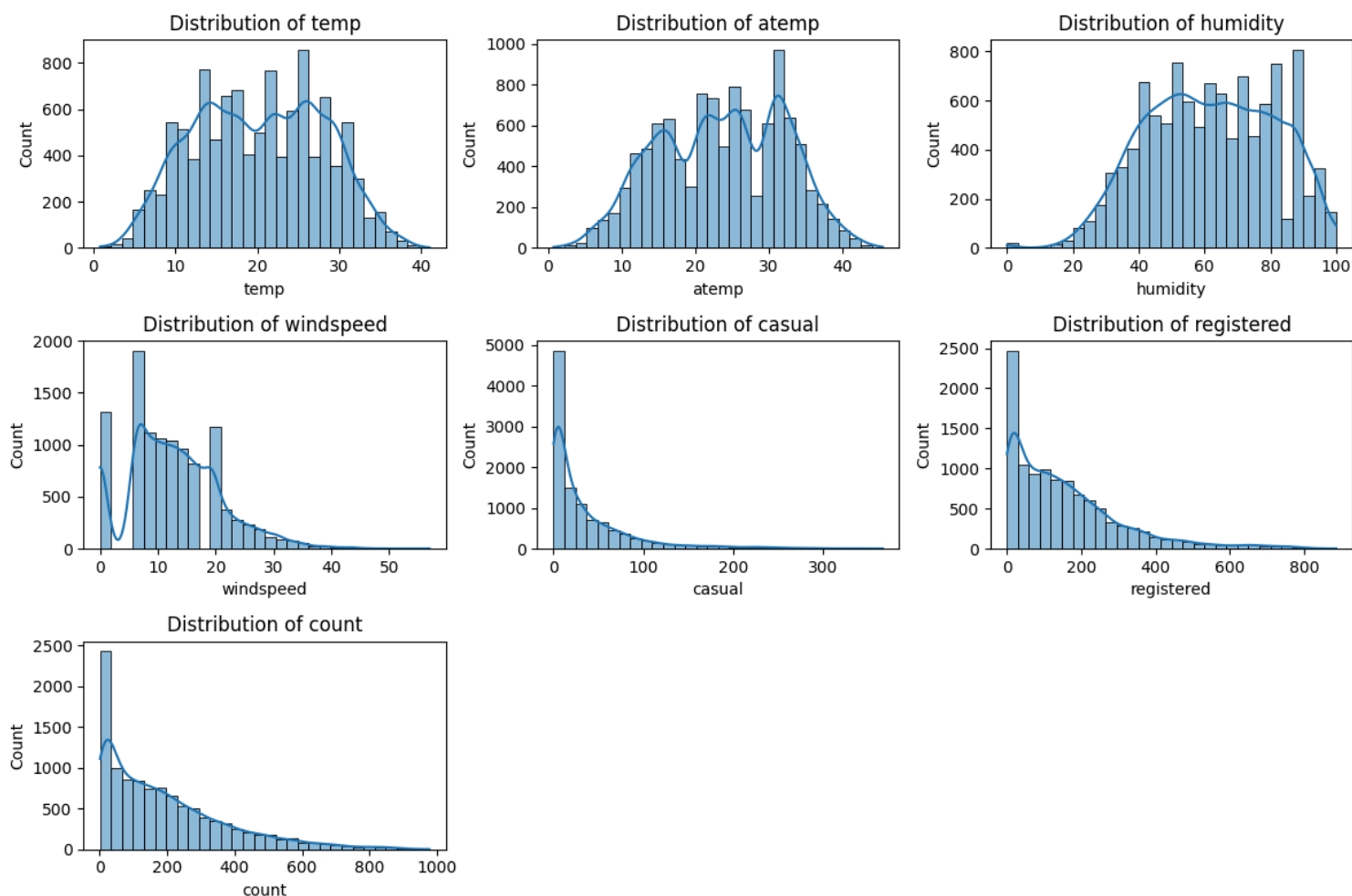


```
np.int64(0)
```

Analyzing the Distribution of Numerical & Categorical Variables

```
# Select numerical columns
numerical_cols = ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count']
```

```
# Plot distribution of numerical variables
plt.figure(figsize=(12, 8))
for i, col in enumerate(numerical_cols):
    plt.subplot(3, 3, i+1)
    sns.histplot(df[col], kde=True, bins=30)
    plt.title(f'Distribution of {col}')
plt.tight_layout()
plt.show()
```



From the multiple histo plots we can see the following.

1. Temperature (temp):

- Appears to be normally distributed, with most values ranging between 10 and 30.

2. atemp:

- Similar to temp, indicating a normal distribution.
- Most values range between 10 and 35.

The bike rental demand is likely influenced by temperature, with higher rentals in comfortable temperature ranges.

3. Humidity (humidity):

- The distribution is fairly normal but slightly skewed.
- Most humidity values are between 40% and 80%, with very few extreme values close to 0 or 100.

Extremely high humidity might reduce bike rentals, while moderate humidity levels may be ideal.

4. Windspeed (windspeed):

- The distribution is highly right-skewed, with many occurrences at lower wind speeds (0-10 km/h).
- There are multiple peaks, suggesting some form of data grouping or recording issues.

High wind speeds may discourage bike rentals, while low wind speeds could be more favorable.

5. Casual Users (casual) & Registered Users (registered):

- Both distributions are highly right-skewed, meaning most users rent bikes in lower numbers.
- There are a few cases of extremely high rentals, possibly during peak hours or special events.

Casual users are fewer compared to registered users. Marketing strategies could focus on converting casual users into registered users.

6. Total Bike Rentals (count):

- The count distribution is right-skewed, indicating that on most days, bike rentals are relatively low.
- However, there are some days with very high rentals, suggesting peak periods.

The rental count might be influenced by external factors such as weather, weekends, or holidays.

Conclusion

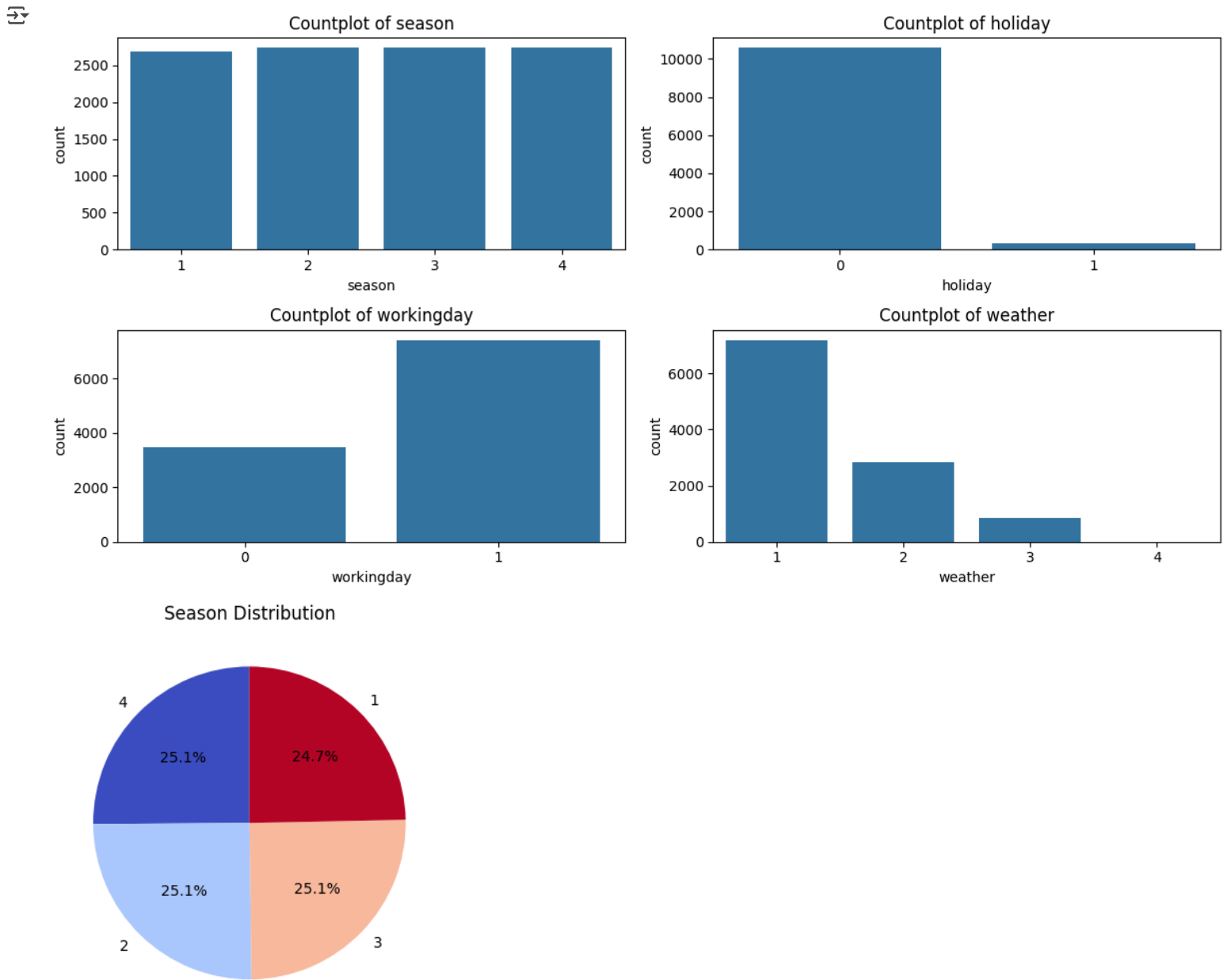
- Temperature and humidity appear to be normally distributed, making them reliable for predictive modeling.

- Casual, registered users, and rental count distributions are right-skewed, indicating that most rental days have fewer users, with occasional peaks.
- Windspeed shows anomalies with multiple peaks, suggesting data inconsistencies or weather fluctuations.

```
# Select categorical columns
categorical_cols = ['season', 'holiday', 'workingday', 'weather']

# Plot count plots
plt.figure(figsize=(12, 6))
for i, col in enumerate(categorical_cols):
    plt.subplot(2, 2, i+1)
    sns.countplot(x=df[col])
    plt.title(f'Countplot of {col}')
plt.tight_layout()
plt.show()

# Pie chart for season
df['season'].value_counts().plot.pie(autopct='%1.1f%%', startangle=90, cmap='coolwarm')
plt.title('Season Distribution')
plt.ylabel('')
plt.show()
```



These bar plots show the frequency distribution of categorical variables.

- Season:
 - The seasons are evenly distributed across all four categories, suggesting a balanced dataset in terms of seasonal representation.
- Holiday:

- Most days are not holidays (label 0), with only a small fraction being holidays (label 1).
- This suggests a potential imbalance in data, which might influence analysis related to holidays.

3. Working Day:

- A larger number of observations fall under working days (1), indicating that most data points are from weekdays rather than weekends.

4. Weather:

- Weather condition 1 (Clear/Partly Cloudy) is the most frequent, followed by condition 2 (Misty/Cloudy).
- Conditions 3 (Light Rain/Snow) and 4 (Heavy Rain/Snow) are rare, indicating that extreme weather conditions are infrequent.

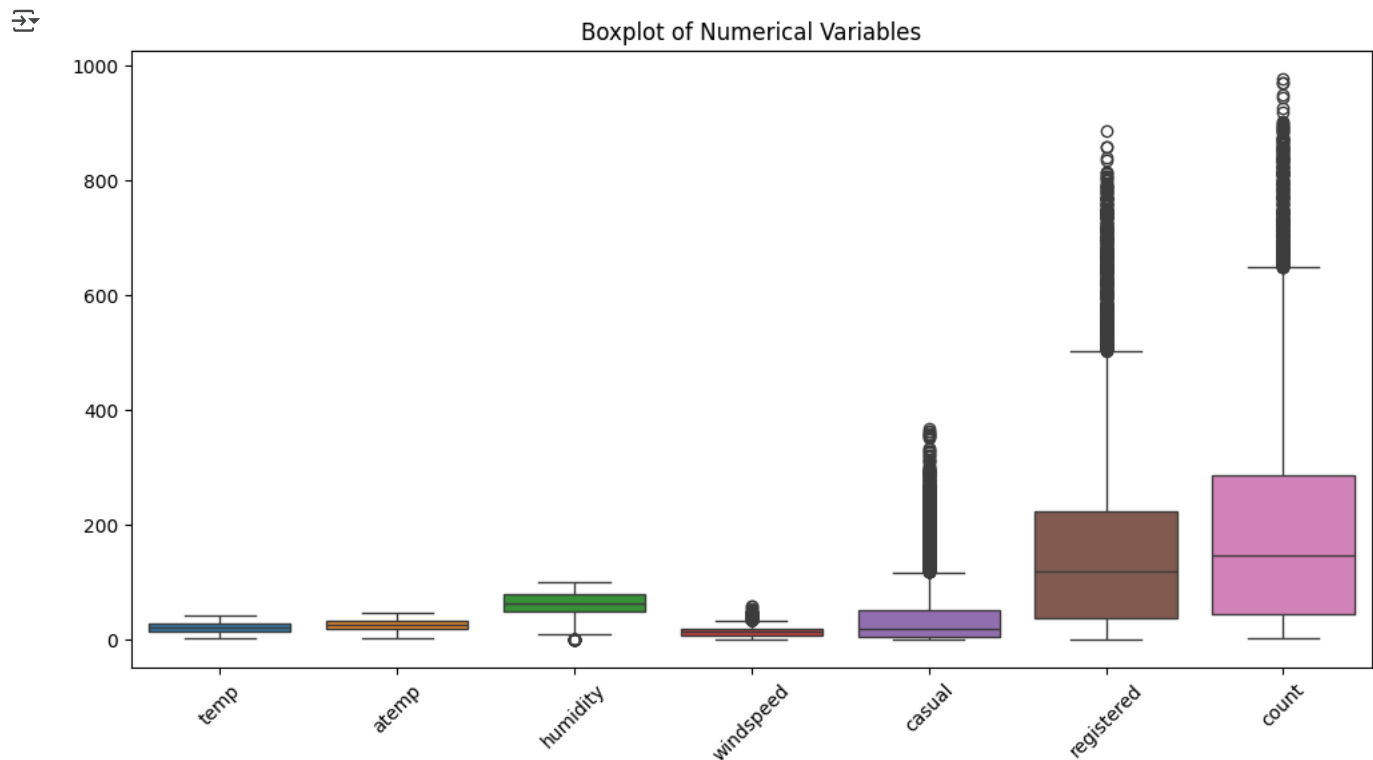
5. Pie Chart of Season Distribution:

- The seasons are nearly evenly distributed, with all four seasons having approximately 25% representation.
- There is no seasonal bias in the dataset.

Handling the Outliers

```
# Select numerical columns
numerical_cols = ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count']
```

```
# Plot boxplots
plt.figure(figsize=(12, 6))
sns.boxplot(data=df[numerical_cols])
plt.xticks(rotation=45)
plt.title("Boxplot of Numerical Variables")
plt.show()
```



1. temp & atemp

- Both have a small range and no significant outliers.
- Indicates that temperature values are fairly consistent with limited variation.

2. Humidity

- Some minor outliers are present.
- Distribution appears slightly spread out, but most values lie within a normal range.

3. Windspeed

- A few extreme outliers at the high end.
- Data is concentrated in a narrow range, suggesting many repeated values (likely due to fixed measurement intervals).

4. Casual

- Highly skewed with many outliers.
- The box is small, indicating that most values are near zero, but a few days have a very high number of casual users.

5. Registered

- Broader distribution compared to casual.

- Some high-value outliers suggest occasional spikes in demand.

6. Total Count


- Similar pattern to registered, indicating total usage is primarily driven by registered users.
- The presence of outliers suggests occasional days with very high rentals.

```
# Compute Q1 (25th percentile) and Q3 (75th percentile)
Q1 = df[numerical_cols].quantile(0.25)
Q3 = df[numerical_cols].quantile(0.75)
IQR = Q3 - Q1

# Define the lower and upper bound
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR


# Remove outliers
df_cleaned = df[~((df[numerical_cols] < lower_bound) | (df[numerical_cols] > upper_bound)).any(axis=1)]

# Print dataset size before and after removing outliers
print(f"Original Dataset Size: {df.shape[0]}")
print(f"Cleaned Dataset Size: {df_cleaned.shape[0]}")
```

 Original Dataset Size: 10886
Cleaned Dataset Size: 9518

```
# Clip outliers to the upper and lower bounds
df_clipped = df.copy()
for col in numerical_cols:
    df_clipped[col] = df[col].clip(lower=lower_bound[col], upper=upper_bound[col])

# Check if extreme values are now within limits
df_clipped.describe()
```

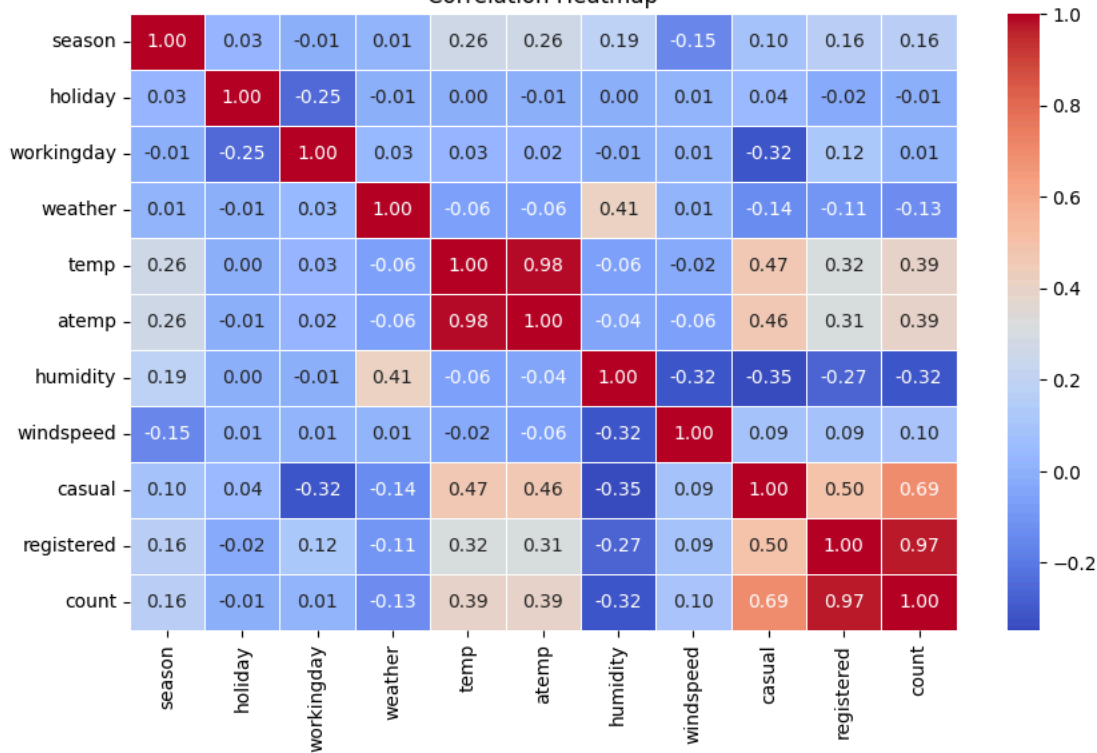
	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.890502	12.703540	31.520898	150.488058	188.000000
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.232240	7.882794	35.502108	135.370102	172.000000
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	2.000000	0.000000	0.000000	0.000000	1.000000
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	47.000000	7.001500	4.000000	36.000000	42.000000
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	62.000000	12.998000	17.000000	118.000000	145.000000
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	77.000000	16.997900	49.000000	222.000000	284.000000
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	100.000000	31.992500	116.500000	501.000000	647.000000

```
# Compute correlation matrix
# Exclude non-numeric columns, like 'datetime', from correlation calculation
corr_matrix = df.select_dtypes(include=np.number).corr()

# Plot heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)
plt.title("Correlation Heatmap")
plt.show()
```



Correlation Heatmap



Explanation of the Correlation Heatmap

A correlation heatmap visually represents the relationship between numerical variables in a dataset using colors and numerical values.

Color Interpretation:

- Red (close to 1) → Strong positive correlation (one variable increases, the other also increases).
- Blue (close to -1) → Strong negative correlation (one variable increases, the other decreases).
- White/Neutral (around 0) → No correlation (no direct relationship between variables).

Diagonal Values (1.00):

- Each variable is perfectly correlated with itself.

1. Correlation Between Features

- temp & atemp (0.98): Very strong positive correlation, meaning actual temperature (temp) and apparent temperature (atemp) move together. This suggests one of them might be redundant in predictive modeling.
- casual & count (0.69) and registered & count (0.97): The total count (count) is highly influenced by the number of registered users (registered) rather than casual users. This implies that registered users form the bulk of bike rentals.
- casual & registered (0.50): Moderate positive correlation suggests that both user types tend to rent bikes under similar conditions.

2. Influence of Weather Conditions

- weather & humidity (0.41): Indicates that higher humidity levels are associated with certain weather conditions (e.g., cloudy, misty).
- weather & windspeed (0.01): Almost no correlation, meaning wind speed does not strongly influence weather categories.
- weather & count (-0.13): Negative correlation implies that adverse weather conditions reduce bike rentals.

3. Working Days & Holidays

- workingday & casual (-0.32): Negative correlation suggests that casual users rent bikes more on non-working days (weekends or holidays).
- workingday & registered (0.12): Slight positive correlation means registered users are slightly more active on working days.

4. Effect of Environmental Factors on Rentals

- temp & count (0.39) and atemp & count (0.39): Moderate positive correlation indicates that higher temperatures are linked to more bike rentals.
- humidity & count (-0.32): Negative correlation suggests that higher humidity levels reduce bike usage.
- windspeed & count (0.10): Weak positive correlation, meaning wind speed has little impact on rentals.

Final Analysis & Insights

- Temperature is a key factor influencing bike rentals. Higher temperatures increase rentals, while high humidity reduces them.
- Registered users dominate the total rental count, making them more reliable for demand forecasting.
- Casual users prefer non-working days, while registered users rent bikes more consistently.

- Bad weather (high humidity, adverse conditions) reduces bike rentals. Future models should account for this trend.
- Wind speed has little effect on bike rentals, so it may not be a significant variable in predictions.

Recommendation:

- Focus on temperature, season, and working days when predicting bike rental demand.
- Consider separating casual and registered users for more precise modeling.
- Humidity and weather conditions should be included as controlling factors.

```
# Set threshold for high correlation (e.g., 0.8)
threshold = 0.8
```

```
# Find pairs of highly correlated features
high_corr_vars = set()
for i in range(len(corr_matrix.columns)):
    for j in range(i):
        if abs(corr_matrix.iloc[i, j]) > threshold:
            colname = corr_matrix.columns[i]
            high_corr_vars.add(colname)
```

```
print("Highly Correlated Variables to Remove:", high_corr_vars)
```

```
# Drop highly correlated variables
df_reduced = df.drop(columns=high_corr_vars)
```

```
🔗 Highly Correlated Variables to Remove: {'count', 'atemp'}
```

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

Null Hypothesis (H_0): There is no significant difference in the average number of bike rides between weekdays and weekends.

Alternative Hypothesis (H_1): There is a significant difference in the average number of bike rides between weekdays and weekends.

```
from scipy import stats
```

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

```
# Extract day of the week (Monday = 0, Sunday = 6)
df['day_of_week'] = df['datetime'].dt.weekday
```

```
# Define weekdays (0-4) and weekends (5-6)
weekdays = df[df['day_of_week'] < 5]['count']
weekends = df[df['day_of_week'] >= 5]['count']
```

```
# Perform independent T-test
t_stat, p_value = stats.ttest_ind(weekdays, weekends, equal_var=False)
```

```
# Print results
print(f"T-Statistic: {t_stat:.4f}")
print(f"P-Value: {p_value:.4f}")
```

```
# Interpretation
if p_value < 0.05:
    print("Reject the Null Hypothesis ( $H_0$ ): There is a significant difference in bike rides between weekdays and weekends.")
else:
    print("Fail to Reject the Null Hypothesis ( $H_0$ ): No significant difference in bike rides between weekdays and weekends.")
```

```
🔗 T-Statistic: 1.0590
P-Value: 0.2897
Fail to Reject the Null Hypothesis ( $H_0$ ): No significant difference in bike rides between weekdays and weekends.
```

No significant difference in bike rides between weekdays and weekends.

Inference: Demand for bikes is consistent across weekdays and weekends.

Recommendations:

- Maintain a steady supply of bikes.
- Focus on overall improvements rather than weekday/weekend-specific changes.
- Consider external factors like weather, time of day, and holidays.

Checking if the demand of bicycles on rent is the same for different Weather conditions

Null Hypothesis (H_0): The average number of bike rentals is the same across different weather conditions.

Alternative Hypothesis (H_1): The average number of bike rentals differs across different weather conditions.


```
# Group the bike rental count based on weather conditions
weather_groups = [df[df['weather'] == i]['count'] for i in df['weather'].unique()]

# Perform One-Way ANOVA test
f_stat, p_value = stats.f_oneway(*weather_groups)

# Print results
print(f"F-Statistic: {f_stat:.4f}")
print(f"P-Value: {p_value:.4f}")

# Interpretation
if p_value < 0.05:
    print("Reject the Null Hypothesis (H₀): There is a significant difference in bike rentals across different weather conditions.")
else:
    print("Fail to Reject the Null Hypothesis (H₀): No significant difference in bike rentals across different weather conditions.")
```

```
🔍 F-Statistic: 65.5302
P-Value: 0.0000
Reject the Null Hypothesis (H₀): There is a significant difference in bike rentals across different weather conditions.
```

```
# Select 'count' column grouped by 'weather'
weather_conditions = df['weather'].unique()

plt.figure(figsize=(12, 6))

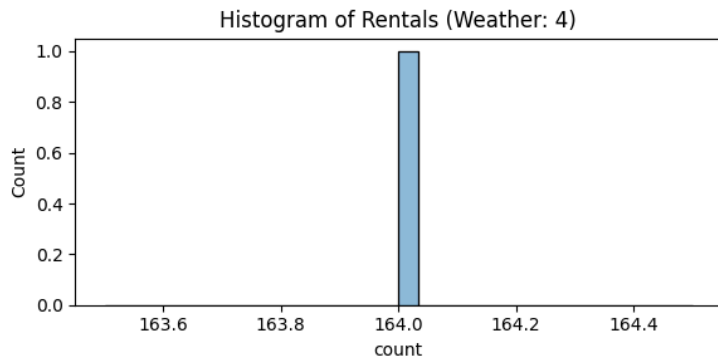
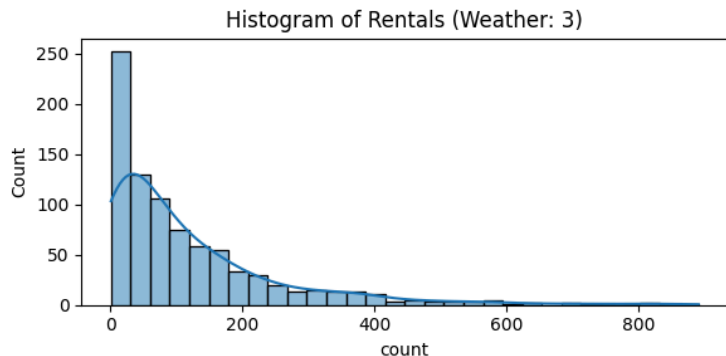
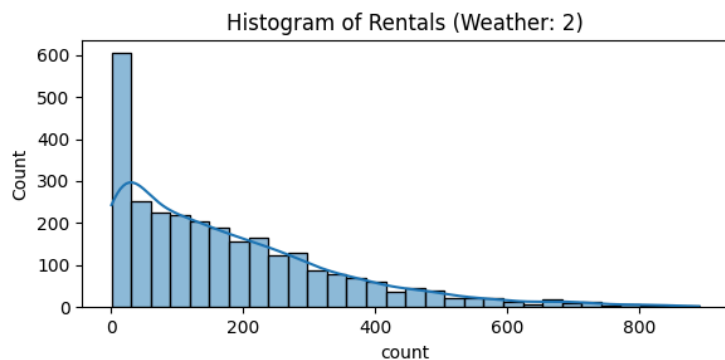
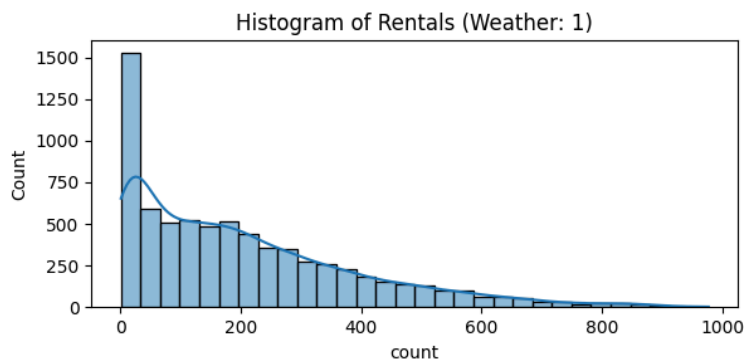
for i, weather in enumerate(weather_conditions, 1):
    plt.subplot(2, 2, i)
    sns.histplot(df[df['weather'] == weather]['count'], kde=True, bins=30)
    plt.title(f'Histogram of Rentals (Weather: {weather})')

plt.tight_layout()
plt.show()

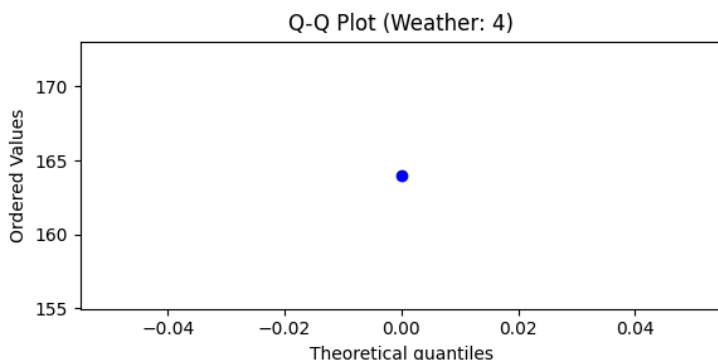
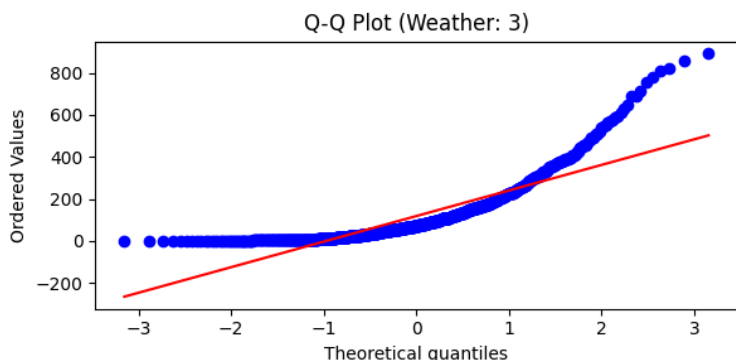
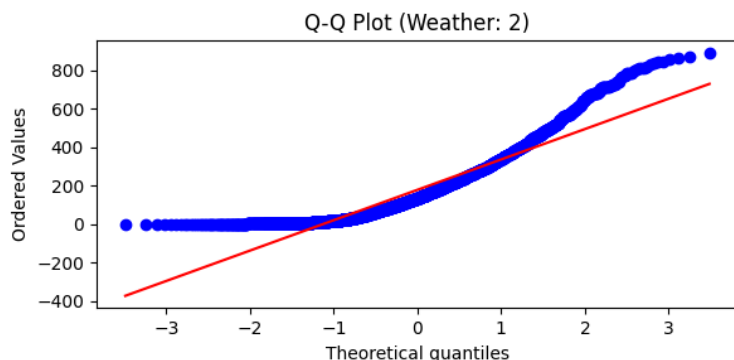
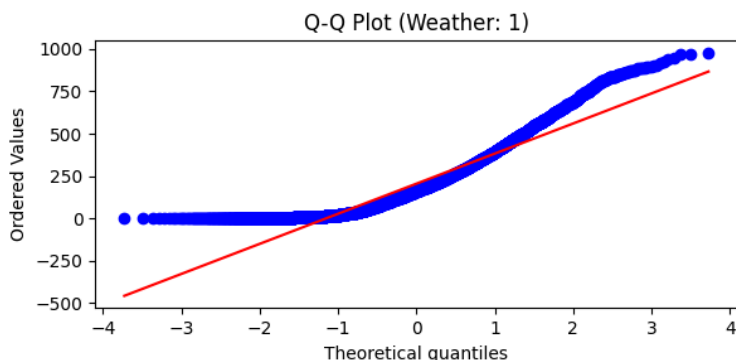
# Q-Q Plot
plt.figure(figsize=(12, 6))

for i, weather in enumerate(weather_conditions, 1):
    plt.subplot(2, 2, i)
    stats.probplot(df[df['weather'] == weather]['count'], dist="norm", plot=plt)
    plt.title(f'Q-Q Plot (Weather: {weather})')

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



```
/usr/local/lib/python3.11/dist-packages/scipy/stats/_stats_py.py:10919: RuntimeWarning: invalid value encountered in scalar divide
  slope = ssym / ssxm
/usr/local/lib/python3.11/dist-packages/scipy/stats/_stats_py.py:10933: RuntimeWarning: invalid value encountered in sqrt
  t = r * np.sqrt(df / ((1.0 - r + TINY)*(1.0 + r + TINY)))
/usr/local/lib/python3.11/dist-packages/scipy/stats/_stats_py.py:10936: RuntimeWarning: invalid value encountered in scalar divide
  slope_stderr = np.sqrt((1 - r**2) * ssym / ssxm / df)
```



Histograms of Rentals Based on Weather

The histograms show the distribution of bike rentals under different weather conditions.

The four subplots represent different weather conditions:

- Weather 1 (Clear, Few Clouds): Most frequent rentals, showing a long-tail distribution.
- Weather 2 (Misty, Cloudy): Fewer rentals compared to Weather 1 but follows a similar trend.
- Weather 3 (Light Snow, Light Rain): Rentals decrease significantly under these conditions.
- Weather 4 (Heavy Rain, Thunderstorm, Snow): Almost no rentals, indicating users avoid renting in extreme weather.

Key Takeaways:

Rentals are highest in clear weather and gradually decrease with worsening conditions. Extreme weather (Weather 4) leads to negligible rentals.

Q-Q Plots of Rentals Based on Weather

These plots compare the distribution of rentals to a normal distribution.

Interpretation:

- The blue dots represent actual data, while the red line represents a normal distribution.
- If the dots align with the red line, the data follows a normal distribution.
- The plots for Weather 1, 2, and 3 show deviations from normality, indicating skewness.
- Weather 4 has only one data point, suggesting it lacks enough data for meaningful statistical interpretation.

Key Takeaways:

- The rental data is not normally distributed, particularly for Weather 1, 2, and 3.
- The distribution is right-skewed, meaning most rentals are on the lower end with occasional high values.
- Extreme weather (Weather 4) has too little data to make meaningful statistical assumptions.

Conclusion

- Weather significantly impacts bike rentals, rentals are highest in clear weather and lowest in extreme weather.
- Humidity negatively affects rentals, while temperature has a positive effect.
- Registered users dominate bike rentals, showing a strong correlation with the total count.
- The data is right-skewed, meaning most rentals are low with some extreme high values.

```
# Check Skewness & Kurtosis for each weather condition
for weather in weather_conditions:
    subset = df[df['weather'] == weather]['count']
    skewness = subset.skew()
    kurtosis = subset.kurt()
    print(f"Weather {weather}: Skewness = {skewness:.2f}, Kurtosis = {kurtosis:.2f}")
```

```
Weather 1: Skewness = 1.14, Kurtosis = 0.96
Weather 2: Skewness = 1.29, Kurtosis = 1.59
Weather 3: Skewness = 2.19, Kurtosis = 6.00
Weather 4: Skewness = nan, Kurtosis = nan
```

Weather 1: Skewness = 1.14, Kurtosis = 0.96

- Right-skewed (moderate positive skewness).
- Flatter than normal (kurtosis < 3), meaning fewer extreme values.

Weather 2: Skewness = 1.29, Kurtosis = 1.59

- More right-skewed than Weather 1.
- Flatter distribution than normal but more extreme values compared to Weather 1.

Weather 3: Skewness = 2.19, Kurtosis = 6.00

- Highly right-skewed, meaning most values are concentrated at lower values with a long tail on the right.
- Highly peaked with heavy tails (kurtosis > 3), meaning there are more extreme values (outliers).

Weather 4: Skewness = NaN, Kurtosis = NaN

- "NaN" (Not a Number) means that the data for this weather condition is either missing or constant (all values are the same).
- If the data contains only a single value (e.g., all zeroes or all ones), skewness and kurtosis cannot be calculated.

```
from scipy.stats import shapiro
```

```
# Perform Shapiro-Wilk Test
for weather in weather_conditions:
    stat, p = shapiro(df[df['weather'] == weather]['count'])
    print(f"Shapiro-Wilk Test for Weather {weather}: p-value = {p:.4f}")

    if p < 0.05:
        print("    -> Data is NOT normally distributed (Reject H₀).")
    else:
        print("    -> Data is normally distributed (Fail to reject H₀).")
```

```
Shapiro-Wilk Test for Weather 1: p-value = 0.0000
    -> Data is NOT normally distributed (Reject H₀).
Shapiro-Wilk Test for Weather 2: p-value = 0.0000
    -> Data is NOT normally distributed (Reject H₀).
Shapiro-Wilk Test for Weather 3: p-value = 0.0000
    -> Data is NOT normally distributed (Reject H₀).
Shapiro-Wilk Test for Weather 4: p-value = nan
    -> Data is normally distributed (Fail to reject H₀).
/usr/local/lib/python3.11/dist-packages/scipy/stats/_axis_nan_policy.py:573: UserWarning: scipy.stats.shapiro: For N > 5000, computed
res = hypotest_fun_out(*samples, **kws)
<ipython-input-32-d2c7995d7fef>:5: SmallSampleWarning: One or more sample arguments is too small; all returned values will be NaN. Se
stat, p = shapiro(df[df['weather'] == weather]['count'])
```

```

from scipy.stats import levene

# Perform Levene's test
levene_stat, levene_p = levene(
    df[df['weather'] == 1]['count'],
    df[df['weather'] == 2]['count'],
    df[df['weather'] == 3]['count'],
    df[df['weather'] == 4]['count']
)

print(f"Levene's Test: p-value = {levene_p:.4f}")

if levene_p < 0.05:
    print("    -> Variances are NOT equal (Reject H₀). Consider Welch ANOVA instead.")
else:
    print("    -> Variances are equal (Fail to reject H₀).")

```

```

🔍 Levene's Test: p-value = 0.0000
    -> Variances are NOT equal (Reject H₀). Consider Welch ANOVA instead.

```

Significantly Affects Bike Rentals

Inference: Bike rental demand varies across different weather conditions.

Recommendations:

- Increase bike availability in favorable weather.
- Offer discounts on rainy/snowy days to encourage rentals.
- Provide real-time weather updates to users for better trip planning.

Checking if the demand of bicycles on rent is the same for different Seasons

Null Hypothesis (H_0): The average number of bike rentals is the same across different seasons.

Alternative Hypothesis (H_1): The average number of bike rentals differs across different seasons.

```

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

plt.figure(figsize=(12, 6))

for i, season in enumerate(seasons, 1):
    plt.subplot(2, 2, i)
    sns.histplot(df[df['season'] == season]['count'], kde=True, bins=30)
    plt.title(f'Histogram of Rentals (Season: {season})')

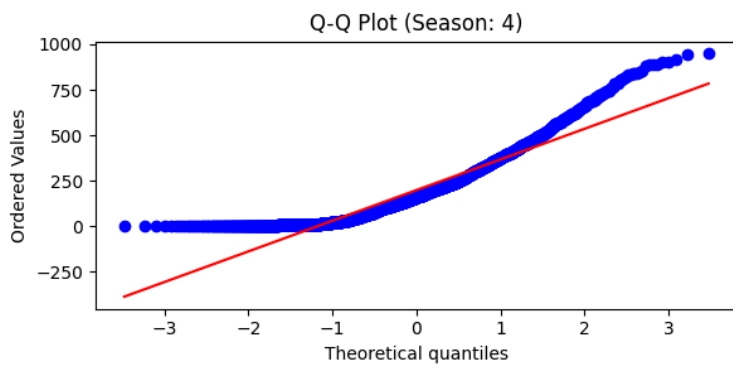
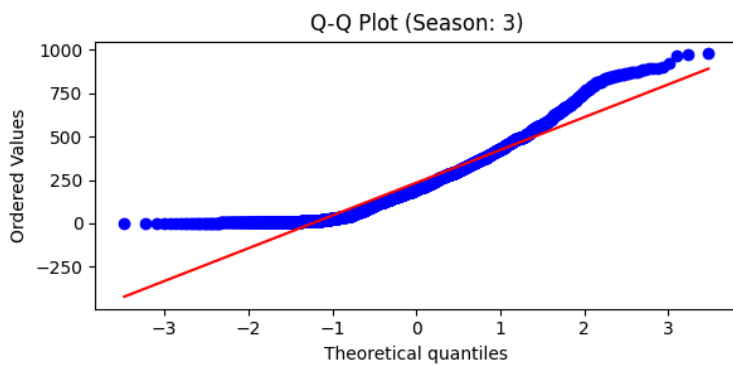
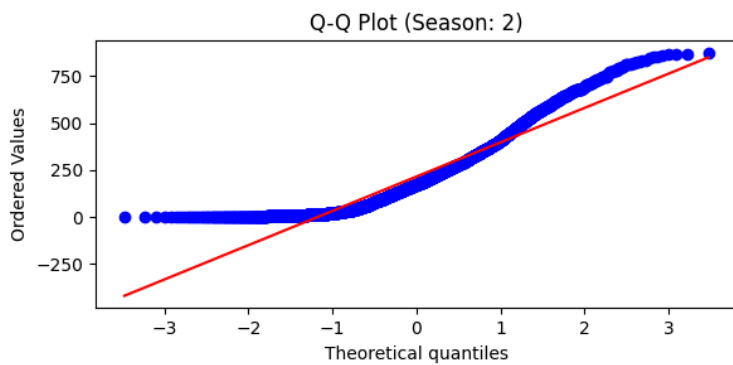
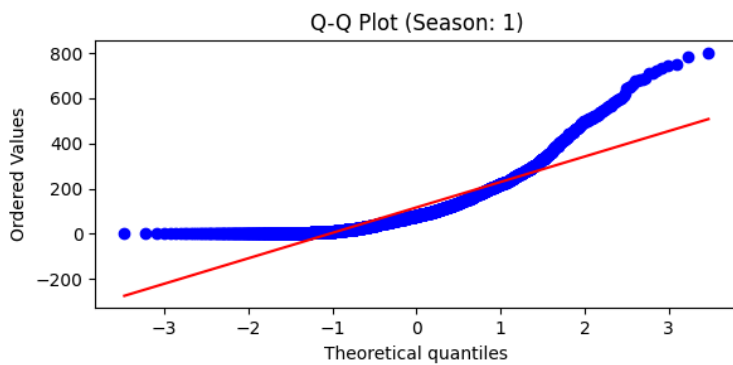
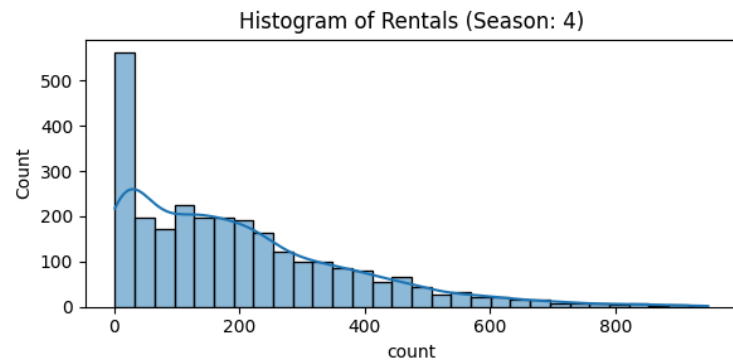
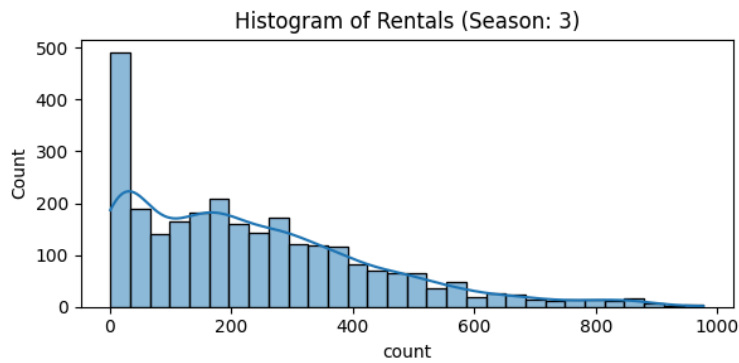
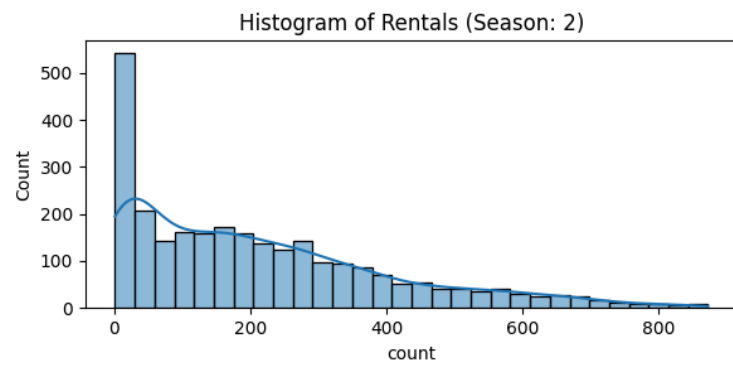
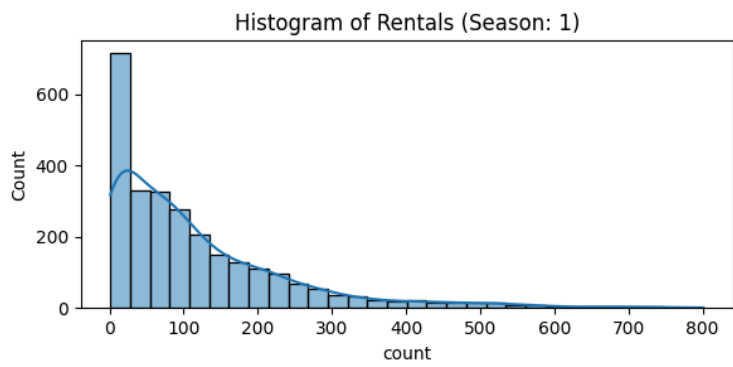
plt.tight_layout()
plt.show()

# Q-Q Plot
plt.figure(figsize=(12, 6))

for i, season in enumerate(seasons, 1):
    plt.subplot(2, 2, i)
    stats.probplot(df[df['season'] == season]['count'], dist="norm", plot=plt)
    plt.title(f'Q-Q Plot (Season: {season})')

plt.tight_layout()
plt.show()

```



Histogram of Rentals by Season

- The histograms show the distribution of rental counts across four seasons.
- The x-axis represents the rental count, while the y-axis represents the frequency of occurrences.
- The shape of the histograms suggests a right-skewed distribution (positive skewness).
- A majority of the rental counts are concentrated at the lower end, with fewer occurrences of higher rental values.
- The density curve (in blue) confirms this skewness, as the tail extends toward the right.

Q-Q Plot of Rentals by Season

- The Q-Q (Quantile-Quantile) plots compare the actual rental count distribution with a theoretical normal distribution.
- The x-axis represents theoretical quantiles, while the y-axis represents the observed values.
- If the data followed a normal distribution, the points would align along the red diagonal line.
- In all four plots, the points deviate from the red line, especially in the tails.
- This confirms that the rental data is not normally distributed, with heavier tails on the right.

Observations:

- Skewed Distribution: The rental data across all seasons is right-skewed, indicating that most rentals are low, but there are some high outliers.
- Non-Normality: The Q-Q plots reinforce that the data is not normally distributed.
- Seasonal Patterns: The distributions appear similar across all four seasons, suggesting a consistent trend in rental behavior.

```
# Check Skewness & Kurtosis for each season
for season in seasons:
    subset = df[df['season'] == season]['count']
    skewness = subset.skew()
    kurtosis = subset.kurt()
    print(f"Season {season}: Skewness = {skewness:.2f}, Kurtosis = {kurtosis:.2f}")
```

```
Season 1: Skewness = 1.89, Kurtosis = 4.31
Season 2: Skewness = 1.00, Kurtosis = 0.43
Season 3: Skewness = 0.99, Kurtosis = 0.70
Season 4: Skewness = 1.17, Kurtosis = 1.27
```

Season 1: Skewness = 1.89, Kurtosis = 4.31

- Highly right-skewed, meaning most values are lower with a few extreme high values.
- Leptokurtic (Kurtosis > 3), indicating a sharp peak and more extreme values (outliers).
- Interpretation: There are some days with exceptionally high values (e.g., demand or rentals).

Season 2: Skewness = 1.00, Kurtosis = 0.43

- Moderately right-skewed but closer to a normal distribution.
- Very flat distribution (platykurtic), meaning fewer extreme values.
- Interpretation: More balanced demand/rentals with fewer spikes.

Season 3: Skewness = 0.99, Kurtosis = 0.70

- Similar to Season 2, but slightly more skewed.
- Still relatively flat, meaning fewer extreme events.
- Interpretation: Most values are near the average, with occasional high values.

Season 4: Skewness = 1.17, Kurtosis = 1.27

- Slightly more right-skewed than Seasons 2 & 3.
- More variation than Seasons 2 & 3 but still not extremely peaked.
- Interpretation: Somewhat more irregular demand but still relatively balanced.

```
# Perform Shapiro-Wilk Test
for season in seasons:
    stat, p = shapiro(df[df['season'] == season]['count'])
    print(f"Shapiro-Wilk Test for Season {season}: p-value = {p:.4f}")

    if p < 0.05:
        print("    -> Data is NOT normally distributed (Reject H₀).")
    else:
        print("    -> Data is normally distributed (Fail to reject H₀).")
```

```
Shapiro-Wilk Test for Season 1: p-value = 0.0000
    -> Data is NOT normally distributed (Reject H₀).
Shapiro-Wilk Test for Season 2: p-value = 0.0000
    -> Data is NOT normally distributed (Reject H₀).
Shapiro-Wilk Test for Season 3: p-value = 0.0000
    -> Data is NOT normally distributed (Reject H₀).
Shapiro-Wilk Test for Season 4: p-value = 0.0000
    -> Data is NOT normally distributed (Reject H₀).
```

```
# Perform Levene's test
levene_stat, levene_p = levene(
    df[df['season'] == 1]['count'],
    df[df['season'] == 2]['count'],
    df[df['season'] == 3]['count'],
    df[df['season'] == 4]['count']
)
```

```
print(f"Levene's Test: p-value = {levene_p:.4f}")
```

```
if levene_p < 0.05:
    print("    -> Variances are NOT equal (Reject H₀). Consider Welch ANOVA instead.")
else:
    print("    -> Variances are equal (Fail to reject H₀).")
```

```
Levene's Test: p-value = 0.0000
    -> Variances are NOT equal (Reject H₀). Consider Welch ANOVA instead.
```

```
# Group the bike rental count based on seasons
season_groups = [df[df['season'] == i]['count'] for i in df['season'].unique()]

# Perform One-Way ANOVA test
f_stat, p_value = stats.f_oneway(*season_groups)

# Print results
print(f"F-Statistic: {f_stat:.4f}")
print(f"P-Value: {p_value:.4f}")

# Decision Rule
if p_value < 0.05:
    print("Reject the Null Hypothesis (H0): There is a significant difference in bike rentals across different seasons.")
else:
    print("Fail to Reject the Null Hypothesis (H0): No significant difference in bike rentals across different seasons.")
```

```
🔗 F-Statistic: 236.9467
P-Value: 0.0000
Reject the Null Hypothesis (H0): There is a significant difference in bike rentals across different seasons.
```

Season Significantly Affects Bike Rentals ($p \leq 0.05$)

Inference: Bike rental demand varies across different seasons.

Recommendations:

- Increase bike availability in peak seasons (e.g., summer).
- Offer discounts or promotions in low-demand seasons (e.g., winter).
- Plan for seasonal maintenance and stocking of bikes.

Checking if Weather Conditions Vary Significantly Across Different Seasons

Null Hypothesis (H_0): Weather conditions are independent of seasons (i.e., there is no relationship between the two).

Alternative Hypothesis (H_1): Weather conditions depend on seasons (i.e., there is a significant relationship between the two).

```
# Create a contingency table (crosstab) for Weather vs. Season
contingency_table = pd.crosstab(df['season'], df['weather'])

# Display the table
print(contingency_table)

🔗 weather season
      1      2      3      4
1  1759   715   211     1
2   1801   708   224     0
3   1930   604   199     0
4   1702   807   225     0

from scipy.stats import chi2_contingency

# Perform Chi-Square Test
chi2_stat, p_value, dof, expected = chi2_contingency(contingency_table)

# Print results
print(f"Chi-Square Statistic: {chi2_stat:.4f}")
print(f"P-Value: {p_value:.4f}")
print(f"Degrees of Freedom: {dof}")

if p_value < 0.05:
    print("Reject the Null Hypothesis (H0): Weather conditions are significantly different across different seasons.")
else:
    print("Fail to Reject the Null Hypothesis (H0): No significant relationship between weather conditions and seasons.")
```

```
🔗 Chi-Square Statistic: 49.1587
P-Value: 0.0000
Degrees of Freedom: 9
Reject the Null Hypothesis (H0): Weather conditions are significantly different across different seasons.
```

Weather Conditions Are Significantly Different Across Seasons ($p \leq 0.05$) Inference: Certain weather conditions are more common in specific seasons.

Recommendations:

- Adjust bike availability based on seasonal weather patterns.
- Plan for increased bike demand during clear weather seasons.
- Offer weather-based pricing incentives.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.