Yulu - Hypothesis Testing

In [26]:
```
#Importing Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [27]:
```
#Reading the dataset
df=pd.read_csv('https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/original/bike_sharing.csv?1642089089')
```

In [28]:
```
df
```

Out[28]:

	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	0.0000	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0000	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0000	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0000	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0000	0	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...
10881	2012-12-19 19:00:00	4	0	1	1	15.58	19.695	50	26.0027	7	329	336
10882	2012-12-19 20:00:00	4	0	1	1	14.76	17.425	57	15.0013	10	231	241
10883	2012-12-19 21:00:00	4	0	1	1	13.94	15.910	61	15.0013	4	164	168
10884	2012-12-19 22:00:00	4	0	1	1	13.94	17.425	61	6.0032	12	117	129
10885	2012-12-19 23:00:00	4	0	1	1	13.12	16.665	66	8.9981	4	84	88

10886 rows × 12 columns

Non graphical exploratory analysis

In [29]:
```
print(f" Rows: {df.shape[0]} \n Columns: {df.shape[1]}")
```

```
 Rows: 10886
 Columns: 12
```

In [30]:
```
print(f"Column name are {df.columns}")
```

```
Column name are Index(['datetime', 'season', 'holiday', 'workingday', 'weather', 'temp',
       'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count'],
      dtype='object')
```

In [31]:
```
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 [32]:
```
df.describe()
```

Out[32]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	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	191.574132
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.245033	8.164537	49.960477	151.039033	181.144454
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

Checking of nullvalues

In [33]:
```
df.isnull().sum()
```

Out[33]:

	0
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

Checking for duplicated values

In [34]:
```
df.isnull().sum()
```

Out[34]:

	0
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

Counting uniques of each column

In [35]:
```
for i in df.columns:
  print(f"{i} : {df[i].nunique()}")
```

```
datetime : 10886
season : 4
holiday : 2
workingday : 2
weather : 4
temp : 49
atemp : 60
humidity : 89
windspeed : 28
casual : 309
registered : 731
count : 822
```

In [36]:
```
df.groupby("holiday")["count"].sum()
```

Out[36]:

holiday	count
0	2027668
1	57808

dtype: int64

In [37]:
```
df.groupby("workingday")["count"].sum()
```

Out[37]:

workingday	count
0	654872
1	1430604

dtype: int64

In [38]:
```
df.groupby("weather")["count"].sum()
```

Out[38]:

weather	count
1	1476063
2	507160
3	102089
4	164

dtype: int64

In [39]:
```
df.groupby("season")["count"].sum()
```

Out[39]:

season	count
1	312498
2	588282
3	640662
4	544034

dtype: int64

In [40]:
```
df["casual"].sum()
```

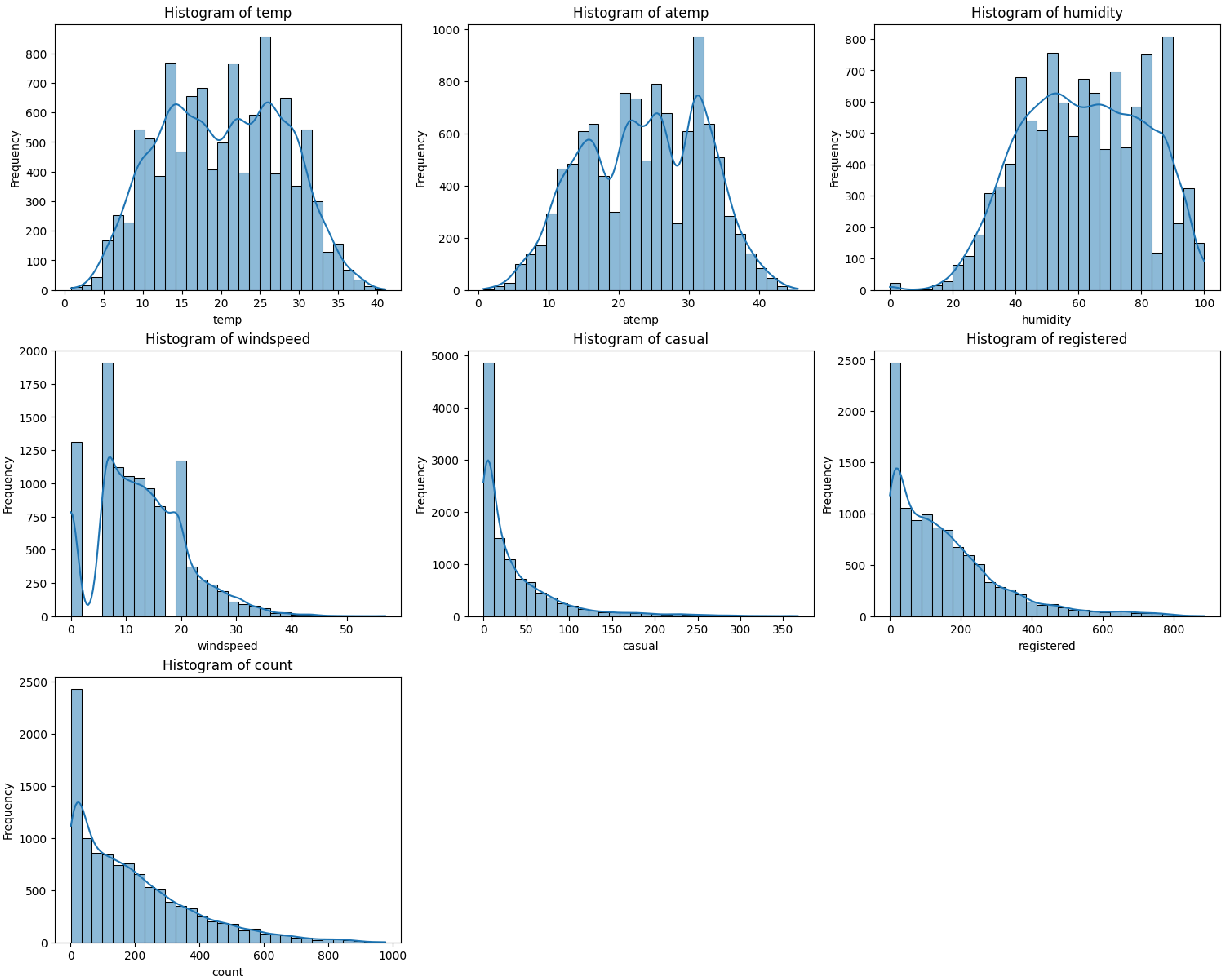
Out[40]: np.int64(392135)

In [41]:
```
df["registered"].sum()
```

Out[41]: np.int64(1693341)

Univariate analysis

In [42]:
```
columns = ["temp", "atemp", "humidity", "windspeed", "casual", "registered", "count"]
n_rows = 3
n_cols = 3
fig, axes = plt.subplots(n_rows, n_cols, figsize=(15, 12), constrained_layout=True)
axes = axes.flatten()
for i, col in enumerate(columns):
  sns.histplot(df[col], kde=True, ax=axes[i], bins=30)
  axes[i].set_title(f"Histogram of {col}")
  axes[i].set_xlabel(col)
  axes[i].set_ylabel("Frequency")
for j in range(len(columns), len(axes)):
  fig.delaxes(axes[j])
plt.show()
```

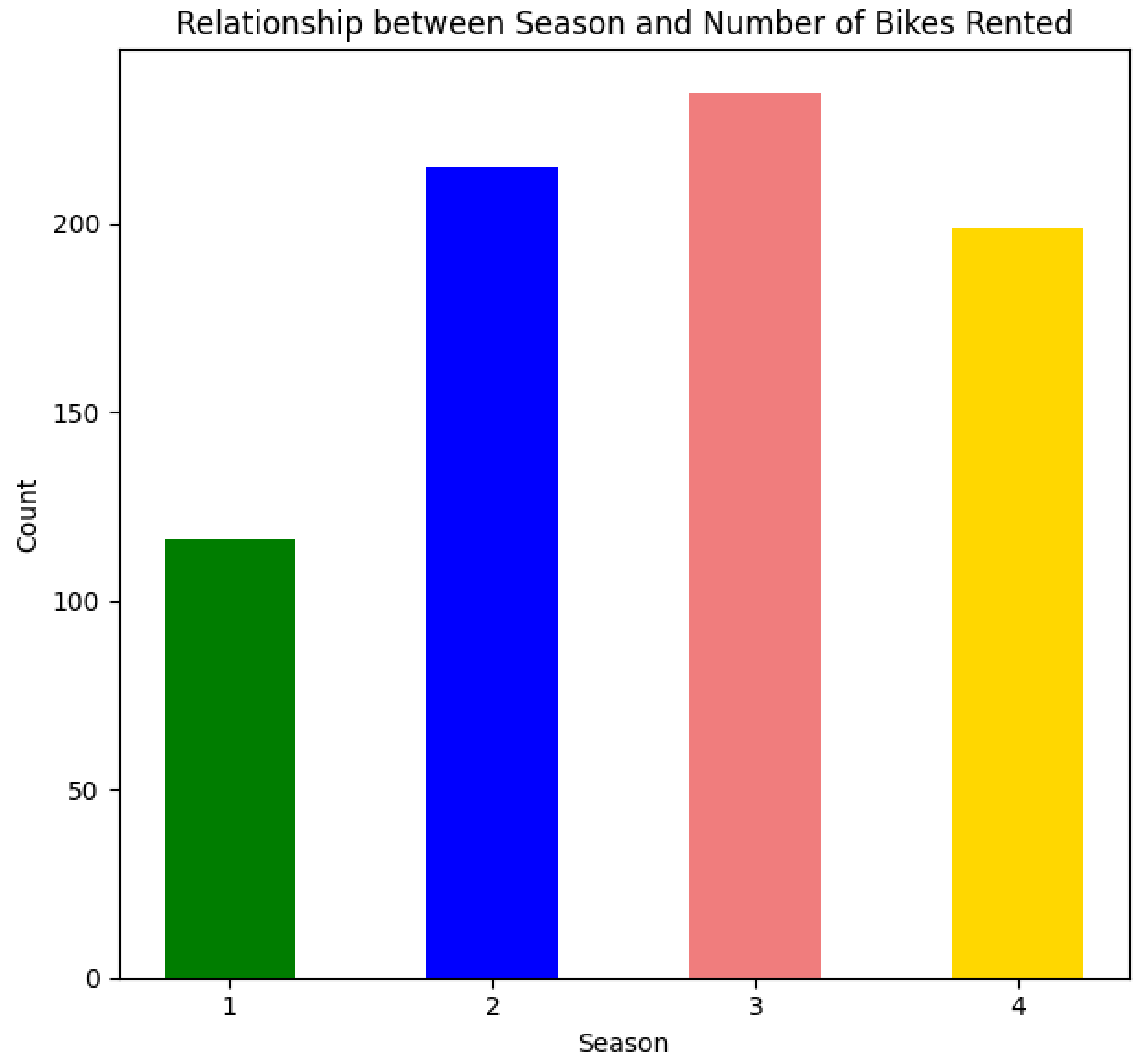


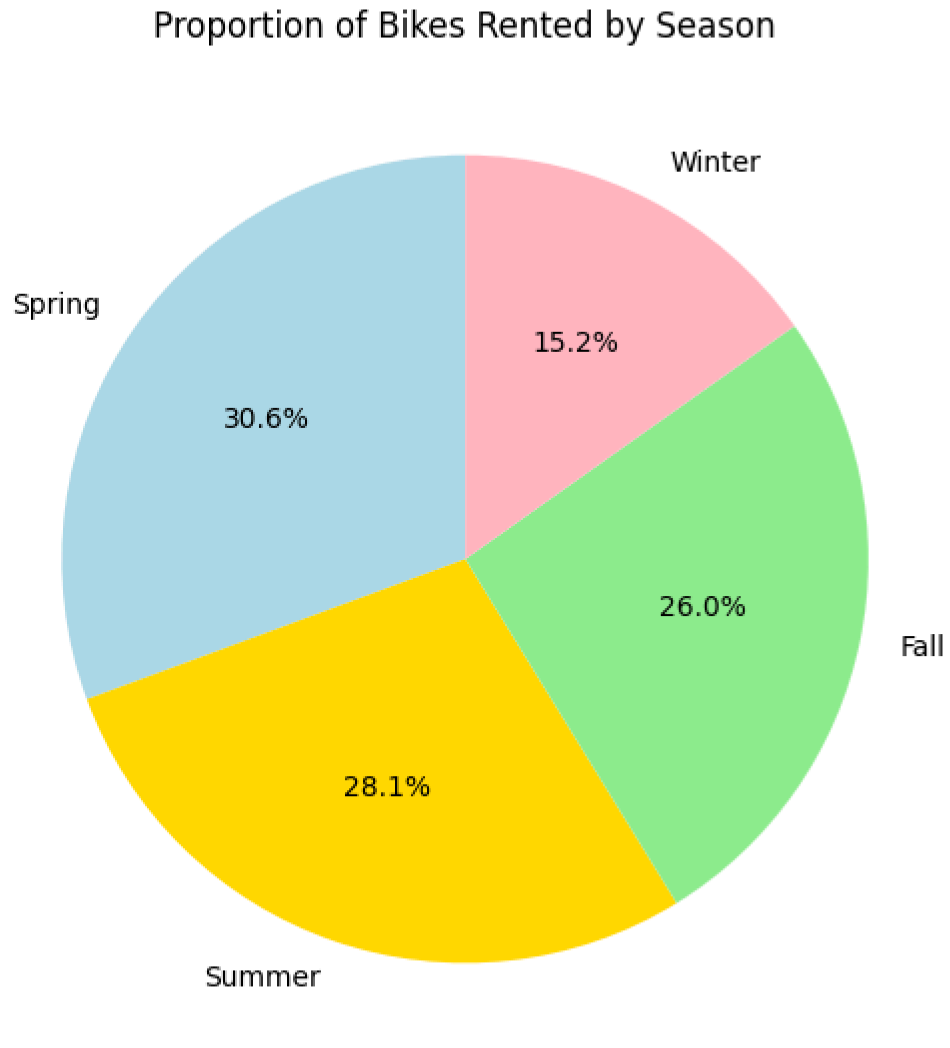
Bivariate analysis

Relationship between season and number of bikes rented

In [43]:
```
# Assuming `seasons` is already calculated
seasons = df.groupby("season")["count"].mean().sort_values(ascending=False)
# Create a figure with two subplots
fig, ax = plt.subplots(1, 2, figsize=(12, 6))
# Bar chart
ax[0].bar(x=seasons.index, height=seasons.values, color=["lightcoral","blue","gold","green"], width=0.5)
ax[0].set_xticks(seasons.index)
ax[0].set_xlabel("Season")
ax[0].set_ylabel("Count")
ax[0].set_title("Relationship between Season and Number of Bikes Rented")
# Pie chart
ax[1].pie(
```

```
    seasons.values,
    labels=["Spring", "Summer", "Fall", "Winter"],
    autopct="%1.1f%%",
    startangle=90,
    colors=["lightblue", "gold", "lightgreen", "lightpink"]
)
ax[1].set_title("Proportion of Bikes Rented by Season")
# Show the plots
plt.tight_layout()
plt.show()
# Print season data
print(seasons)
```






```
season
3    234.417124
2    215.251372
4    198.988296
1    116.343261
Name: count, dtype: float64
```

Observations

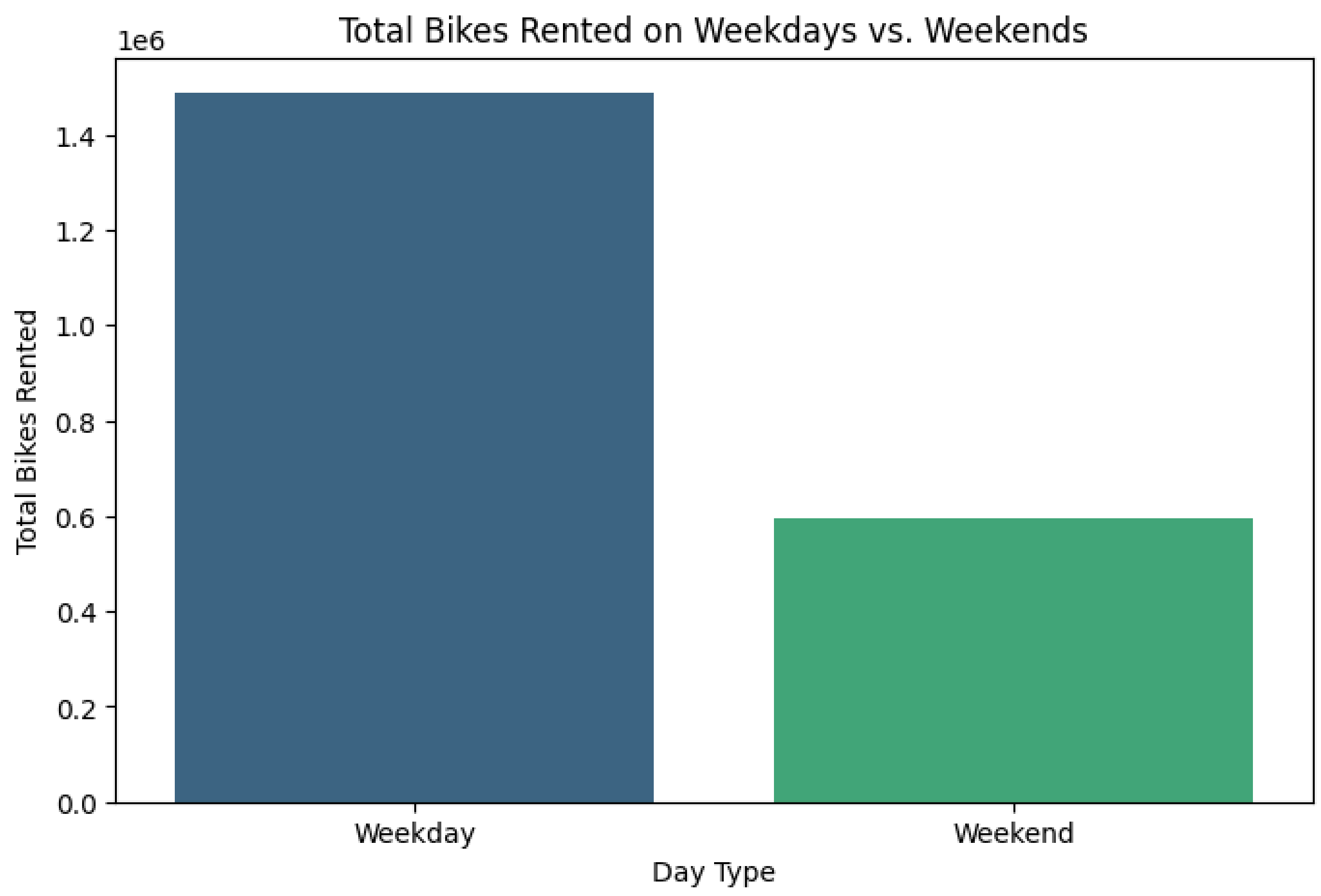
- We can see that Fall(season 3) has the highsest average of bikes rented.
- Summer and Winter has 2nd and 3rd highest average with spring being the least.

No of bikes rented on weekdays and weekend

In [44]:
```
# Convert datetime column to datetime format
df['datetime'] = pd.to_datetime(df['datetime'])
```

In [45]:
```
# Extracting day of the week (0=Monday, 6=Sunday)
df['day_of_week'] = df['datetime'].dt.dayofweek

# Categorizing as Weekday (0-4) and Weekend (5-6)
df['is_weekend'] = df['day_of_week'].apply(lambda x: 'Weekend' if x >= 5 else 'Weekday')

# Grouping by Weekday/Weekend and summing up the count of bikes rented
weekday_vs_weekend = df.groupby('is_weekend')['count'].sum()

# Plotting
plt.figure(figsize=(8, 5))
sns.barplot(x=weekday_vs_weekend.index, y=weekday_vs_weekend.values, palette="viridis")
plt.xlabel("Day Type")
plt.ylabel("Total Bikes Rented")
plt.title("Total Bikes Rented on Weekdays vs. Weekends")
plt.show()
```

```
<ipython-input-45-98cb8b470086>:12: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=weekday_vs_weekend.index, y=weekday_vs_weekend.values, palette="viridis")
```



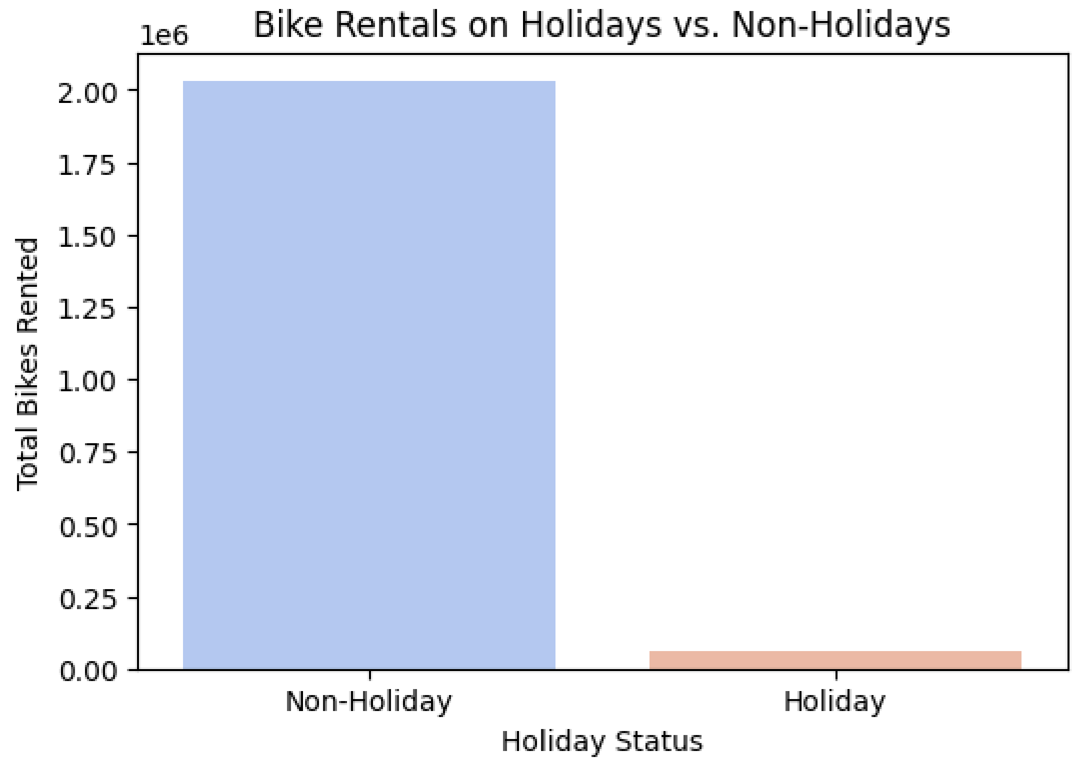
In [46]:
```
weekday_vs_weekend = df.groupby('is_weekend')['count'].sum()
print(weekday_vs_weekend)
```

```
is_weekend
Weekday    1488412
Weekend     597064
Name: count, dtype: int64
```

Realationship between holiday and no of bikes rented

In [47]:
```python
# Grouping data by holiday status
holiday_rentals = df.groupby('holiday')['count'].sum()

# Plotting
plt.figure(figsize=(6, 4))
sns.barplot(x=holiday_rentals.index, y=holiday_rentals.values, palette="coolwarm")

# Customizing plot
plt.xticks([0, 1], ["Non-Holiday", "Holiday"])
plt.xlabel("Holiday Status")
plt.ylabel("Total Bikes Rented")
plt.title("Bike Rentals on Holidays vs. Non-Holidays")
plt.show()

# Display counts
print(holiday_rentals)
```

```
<ipython-input-47-a9ee20b59f3a>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=holiday_rentals.index, y=holiday_rentals.values, palette="coolwarm")
```

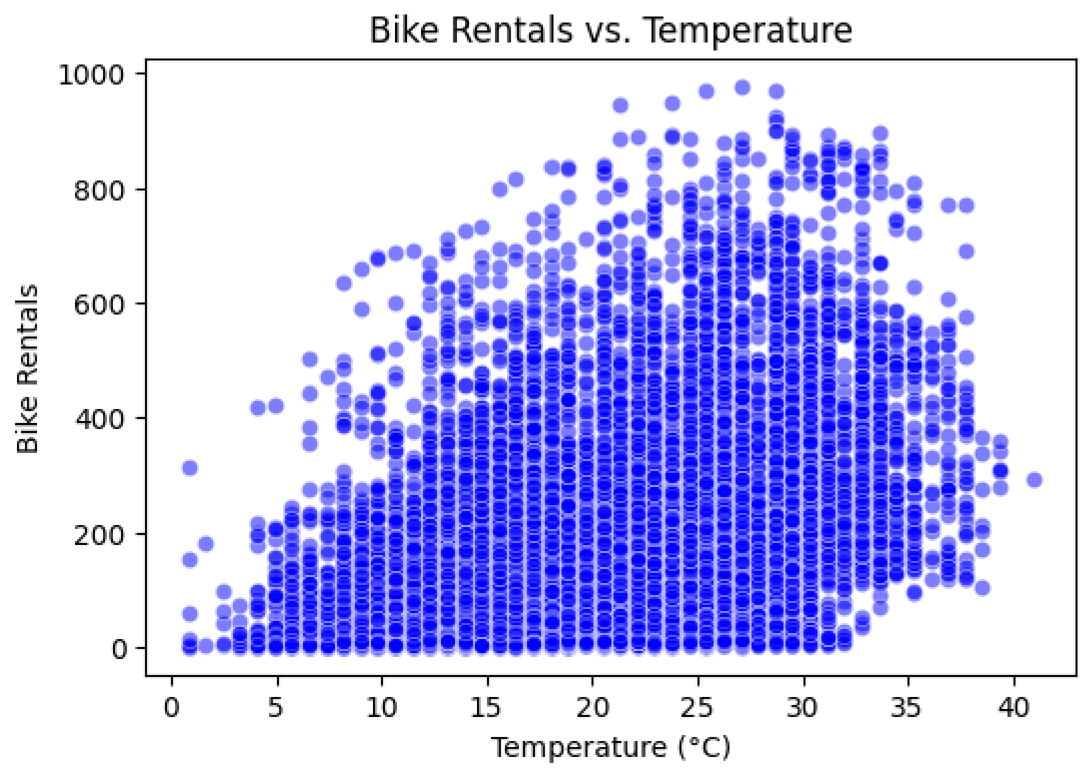


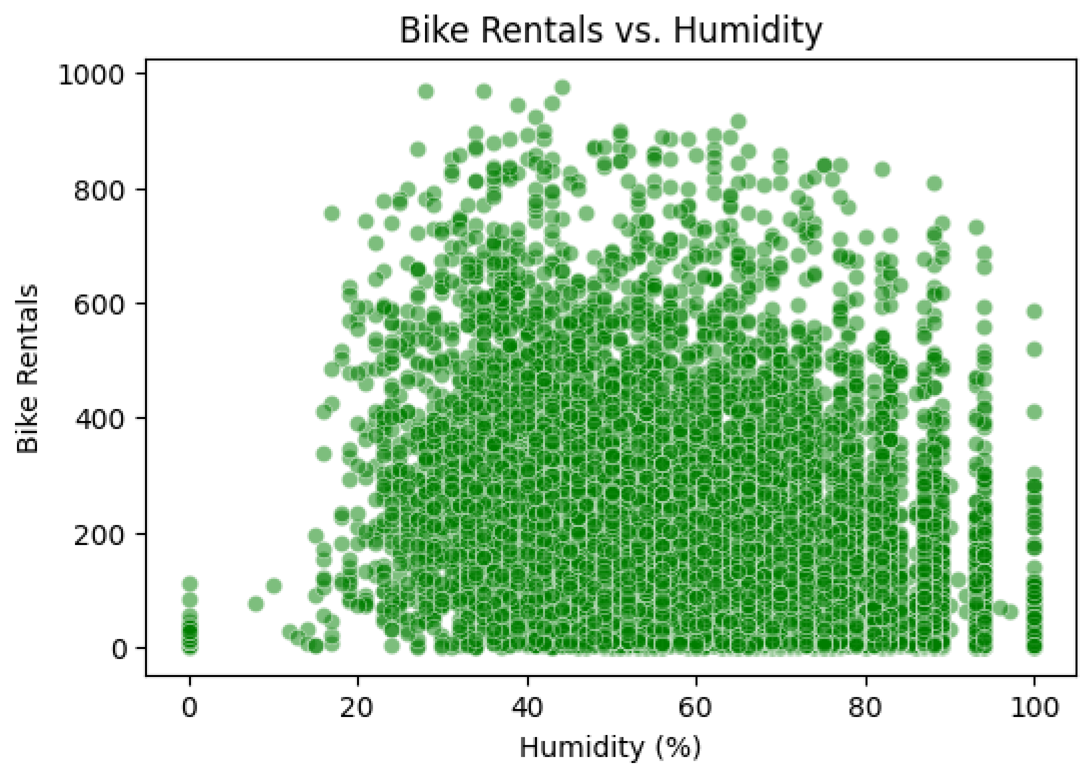

```
holiday
0    2027668
1      57808
Name: count, dtype: int64
```

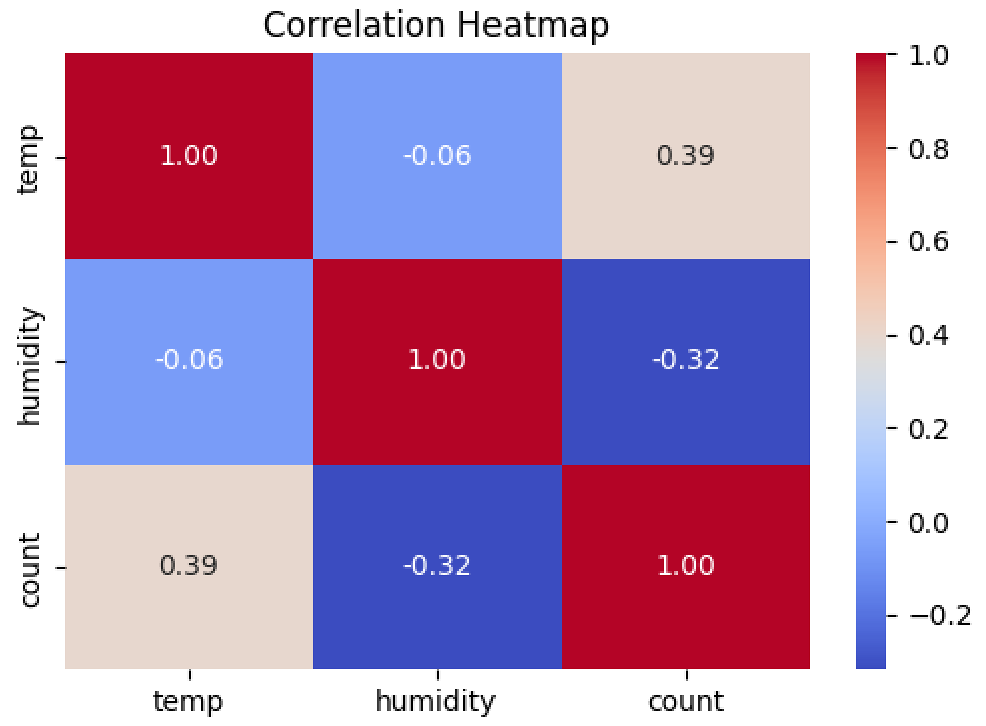
Multivariate analysis

Relationship of bikes rented based on temperature and humidity

In [48]:
```python
# Scatter plot of temperature vs. bike rentals
plt.figure(figsize=(6, 4))
sns.scatterplot(data=df, x='temp', y='count', alpha=0.5, color="blue")
plt.xlabel("Temperature (°C)")
plt.ylabel("Bike Rentals")
plt.title("Bike Rentals vs. Temperature")
plt.show()

# Scatter plot of humidity vs. bike rentals
plt.figure(figsize=(6, 4))
sns.scatterplot(data=df, x='humidity', y='count', alpha=0.5, color="green")
plt.xlabel("Humidity (%)")
plt.ylabel("Bike Rentals")
plt.title("Bike Rentals vs. Humidity")
plt.show()

# Heatmap for correlation between temp, humidity, and bike count
plt.figure(figsize=(6, 4))
sns.heatmap(df[['temp', 'humidity', 'count']].corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```

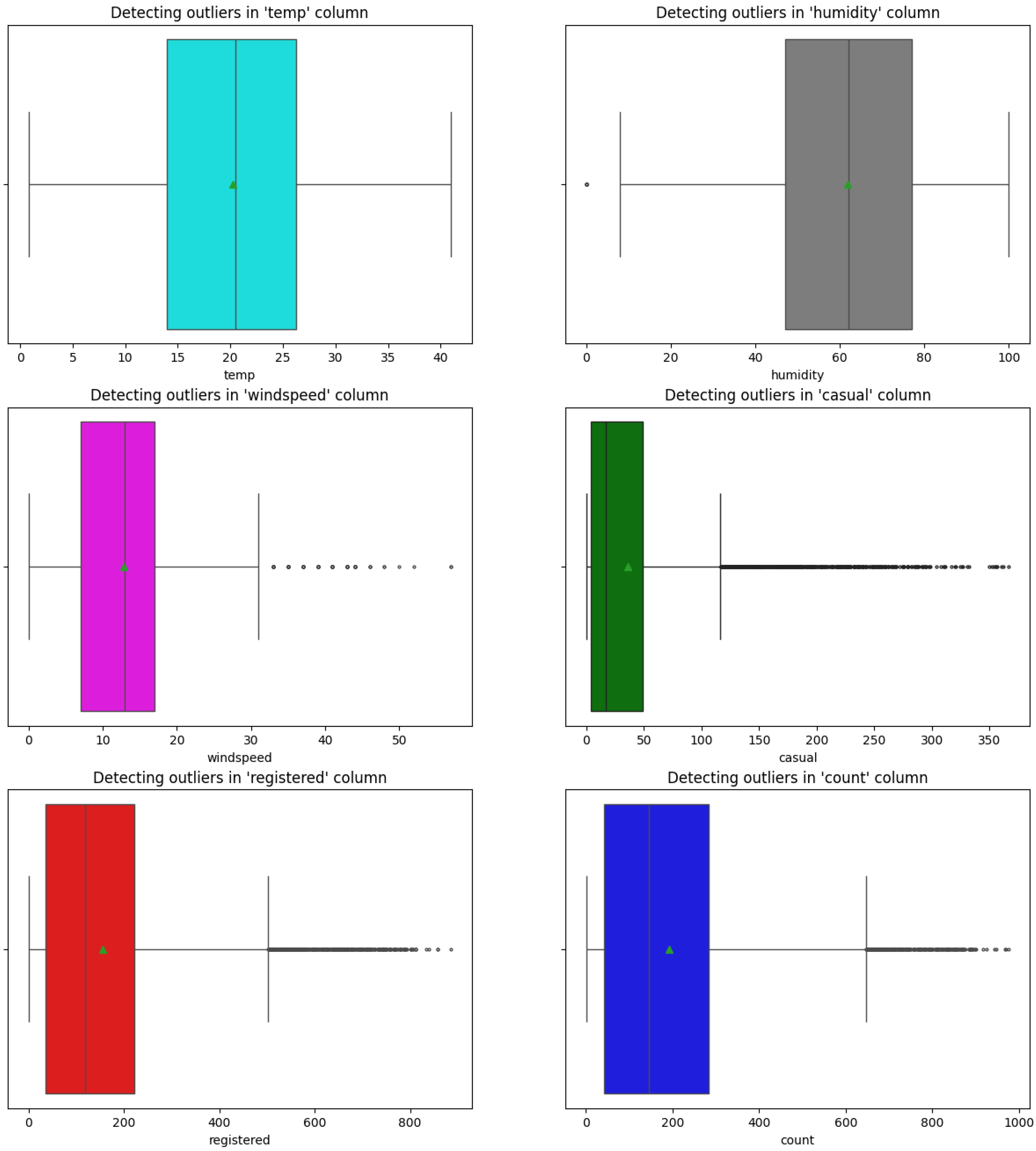






Detecting outliers

In [49]:
```
columns = ['temp', 'humidity', 'windspeed', 'casual', 'registered', 'count']
colors = np.random.permutation(['red', 'blue', 'green', 'magenta', 'cyan', 'gray'])
count = 1
plt.figure(figsize = (15, 16))
for i in columns:
    plt.subplot(3, 2, count)
    plt.title(f"Detecting outliers in '{i}' column")
    sns.boxplot(data = df, x = df[i], color = colors[count - 1], showmeans = True, fliersize = 2)
    plt.plot()
    count += 1
```



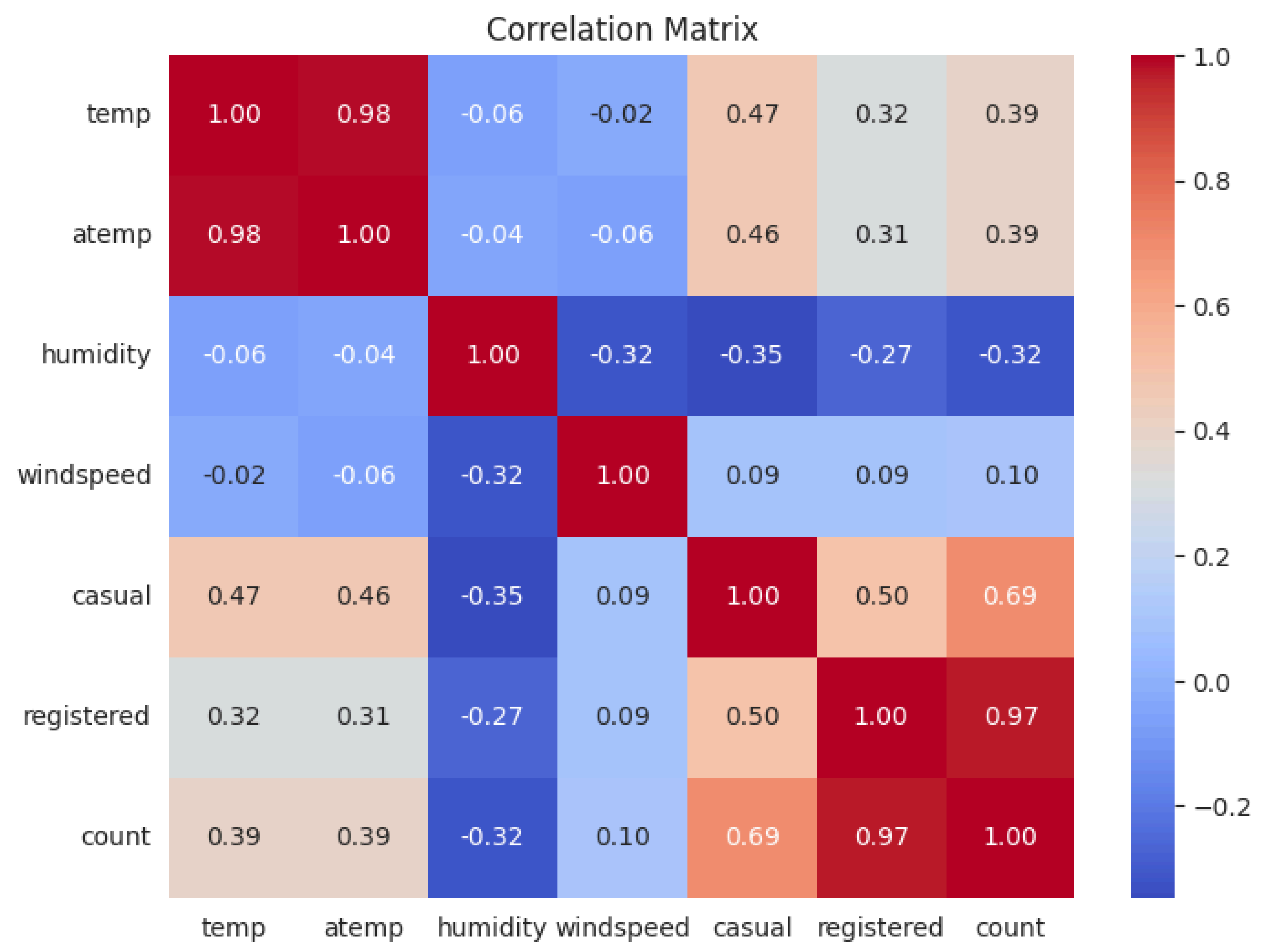
Observations

- There is no outlier in the temp column.
- There are few outliers present in humidity column.
- There are many outliers present in each of the columns : windspeed, casual, registered, count.

Relationship between the Dependent and Independent Variables

In [51]:

```python
data = {
'temp': df["temp"],
'atemp': df["atemp"],
'humidity': df["humidity"],
'windspeed': df["windspeed"],
'casual': df["casual"],
'registered': df["registered"],
'count':df["count"]
}
cor = pd.DataFrame(data)
correlation_matrix = cor.corr()
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap="coolwarm", cbar=True)
plt.title("Correlation Matrix")
plt.show()
```



Observations

- Temperature, atemp, and registered users have the strongest positive correlations with bike rentals.
- Humidity and weather conditions negatively impact rentals.
- Wind speed, holidays, and working days have little effect.

Check if there is significant difference in bike rides between weekdays and weekends

In [57]:

```python
from scipy.stats import ttest_ind

# Separate data into weekdays and weekends
weekdays = df[df["workingday"] == 1]["count"]
weekends = df[df["workingday"] == 0]["count"]

# Perform independent t-test
t_stat, p_value = ttest_ind(weekdays, weekends, equal_var=False)  # Welch's t-test

# Display results
print(f"T-statistic: {t_stat:.4f}")
print(f"P-value: {p_value:.4f}")

# Interpretation
alpha = 0.05
if p_value < alpha:
    print("There is a significant difference in bike rides between weekdays and weekends.")
else:
    print("There is no significant difference in bike rides between weekdays and weekends.")
```

```
T-statistic: 1.2363
P-value: 0.2164
There is no significant difference in bike rides between weekdays and weekends.
```

Check weather the no. of bike rides greater during week days

Hypotheses

Null Hypothesis (H_0): The average number of bike rides during weekdays is less than or equal to the average number of bike rides during weekends.

Alternative Hypothesis (H_1): The average number of bike rides during weekdays is greater than the average number of bike rides during weekends.

In [58]:

```python
# Separate data into weekdays and weekends
weekdays = df[df["workingday"] == 1]["count"]
weekends = df[df["workingday"] == 0]["count"]

# Perform one-tailed t-test (checking if weekdays > weekends)
t_stat, p_value = ttest_ind(weekdays, weekends, equal_var=False, alternative='greater')

# Display results
print(f"T-statistic: {t_stat:.4f}")
print(f"P-value: {p_value:.4f}")

# Interpretation
alpha = 0.05
if p_value < alpha:
    print("The number of bike rides is significantly greater during weekdays.")
else:
    print("There is no significant evidence that bike rides are greater during weekdays.")
```

```
T-statistic: 1.2363
P-value: 0.1082
There is no significant evidence that bike rides are greater during weekdays.
```

Observations

- This test will confirm whether bike rentals are significantly higher on weekdays compared to weekends.

Check if the demand of bikes on rent is the same for different Weather conditions

Hypotheses

Null Hypothesis (H_0): The demand for bikes is same for different weather conditions.

Alternative Hypothesis (H_1): The demand for bikes for different weather changes with different weather conditions

In [62]:
```
df["weather"].unique()
```

Out[62]:
```
array([1, 2, 3, 4])
```

1: Clear, Few clouds, partly cloudy

2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist

3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain+Scattered clouds

4: Heavy Rain + Ice Pellets + Thunderstorm + Mist, Snow + Fog

In [63]:
```
df["weather"].value_counts()
```

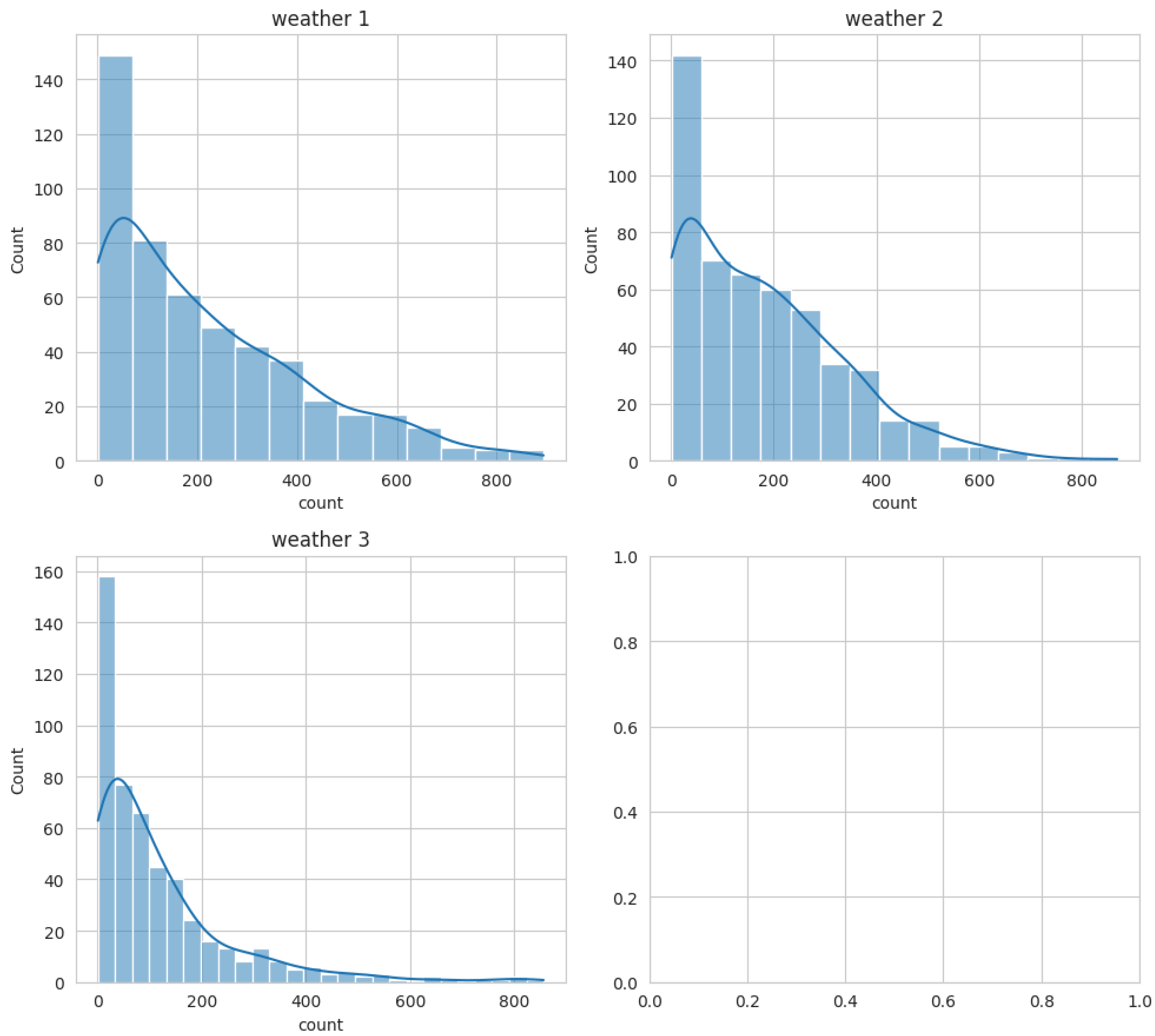
Out[63]:

weather	count
1	7192
2	2834
3	859
4	1

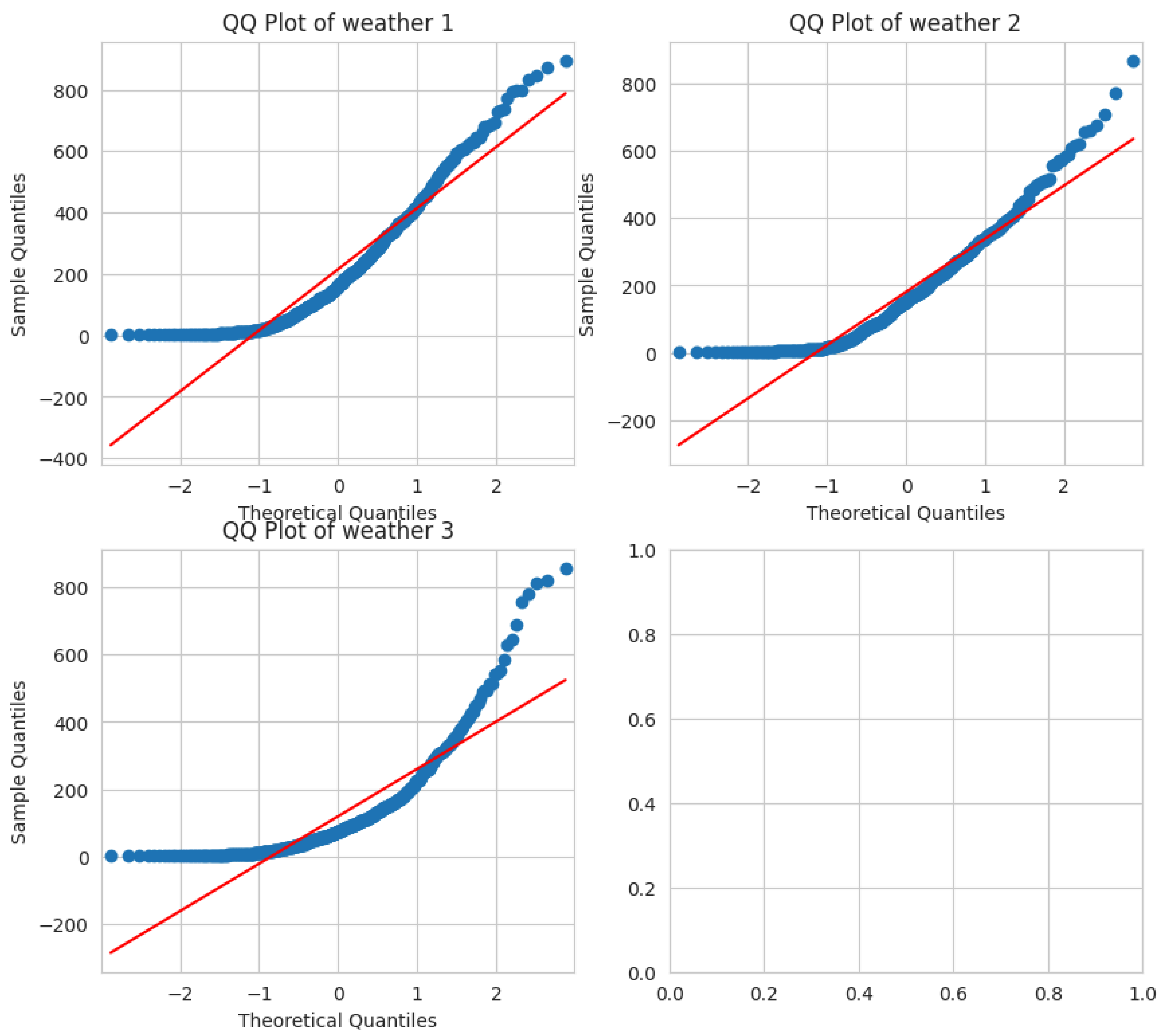
dtype: int64

In [64]:
```
weather1=df[df["weather"]==1]['count'].sample(500)
weather2=df[df["weather"]==2]['count'].sample(500)
weather3=df[df["weather"]==3]['count'].sample(500)
weather4=df[df["weather"]==4]['count'].sample(1)
```

In [66]:
```
#checking the assumptions of test.
import statsmodels.api as sm
all_weathers=[weather1,weather2,weather3]
n_rows, n_cols = 2,2
fig, axes = plt.subplots(n_rows, n_cols, figsize=(10, 9))
axes = axes.flatten()
for idx, data in enumerate(all_weathers):
  sns.histplot(data, kde=True, ax=axes[idx])
  axes[idx].set_title(f"weather {idx+1}")
plt.tight_layout()
```



In [68]:
```
import statsmodels.api as sm # import the statsmodels library
all_weather = [weather1,weather2,weather3]
n_rows, n_cols = 2, 2
fig, axes = plt.subplots(n_rows, n_cols, figsize=(10, 9))
axes = axes.flatten()
for i in range(len(all_weather)):
  sm.qqplot(all_weather[i], line="s", ax=axes[i])
  axes[i].set_title(f"QQ Plot of weather {i + 1}")
```



In [70]:
```
#shapiro_wilk normalacy test
#null:dist is normal
#alternate:dist is not normal
from scipy.stats import shapiro
for i in all_weathers:
  stat,pvalue=shapiro(i)
  print(f"stat:{stat},pvalue:{pvalue}")
if pvalue<alpha:
  print("the dist is not normal")
else:
  print("the dist is normal")
```

```
stat:0.8915731707183179,pvalue:2.6749120279518974e-18
stat:0.9080197050135157,pvalue:7.905258368769154e-17
stat:0.7568267269096141,pvalue:1.70767555941135e-26
the dist is not normal
```

In [72]:
```
#checking of equality of varience
from scipy.stats import levene
lstat,pvalue=levene(weather1,weather2,weather3)
lstat,pvalue
if pvalue<alpha:
  print("the varience is not equal")
else:
  print("the varience is equal")
```

```
the varience is not equal
```

- Hence the assumptions of normality and varience fail we use kw test

In [73]:
```
from scipy.stats import kruskal
stat,pvalue=kruskal(weather1,weather2,weather3)
print(f"k_stat:{stat},p_value:{pvalue}")
if pvalue<alpha:
  print("we reject the null hypothesis")
  print("the demand for bikes is not the same for different weather conditions")
else:
  print("we fail to reject null hypothesis")
  print("the demand for bikes is the same for different weather conditions")
```

```
k_stat:76.64904369125512,p_value:2.2691940746748365e-17
we reject the null hypothesis
the demand for bikes is not the same for different weather conditions
```

Observations

- The p value is very low hence we reject null hypo with confidence.
- This shows weather plays significant role in the usage of bycicles.
- The variability in demand across weather conditions highlights the importance of weather as a critical factor influencing bike usage.

Check if the demand of bikes on rent is the same for different Seasons

In [74]:
```
df["season"].unique() #season: season (1: spring, 2: summer, 3: fall, 4: winter)
```

Out[74]:
```
array([1, 2, 3, 4])
```

In [75]:
```
spring=df[df["season"]==1]["count"].sample(1000)
summer=df[df["season"]==2]["count"].sample(1000)
fall=df[df["season"]==3]["count"].sample(1000)
winter=df[df["season"]==4]["count"].sample(1000)
```

Hypotheses
Null hypothesis= demand of bikes on rent is the same for different Seasons
Alternate hypothesis= demand of bikes on rent is not the same for different Seasons

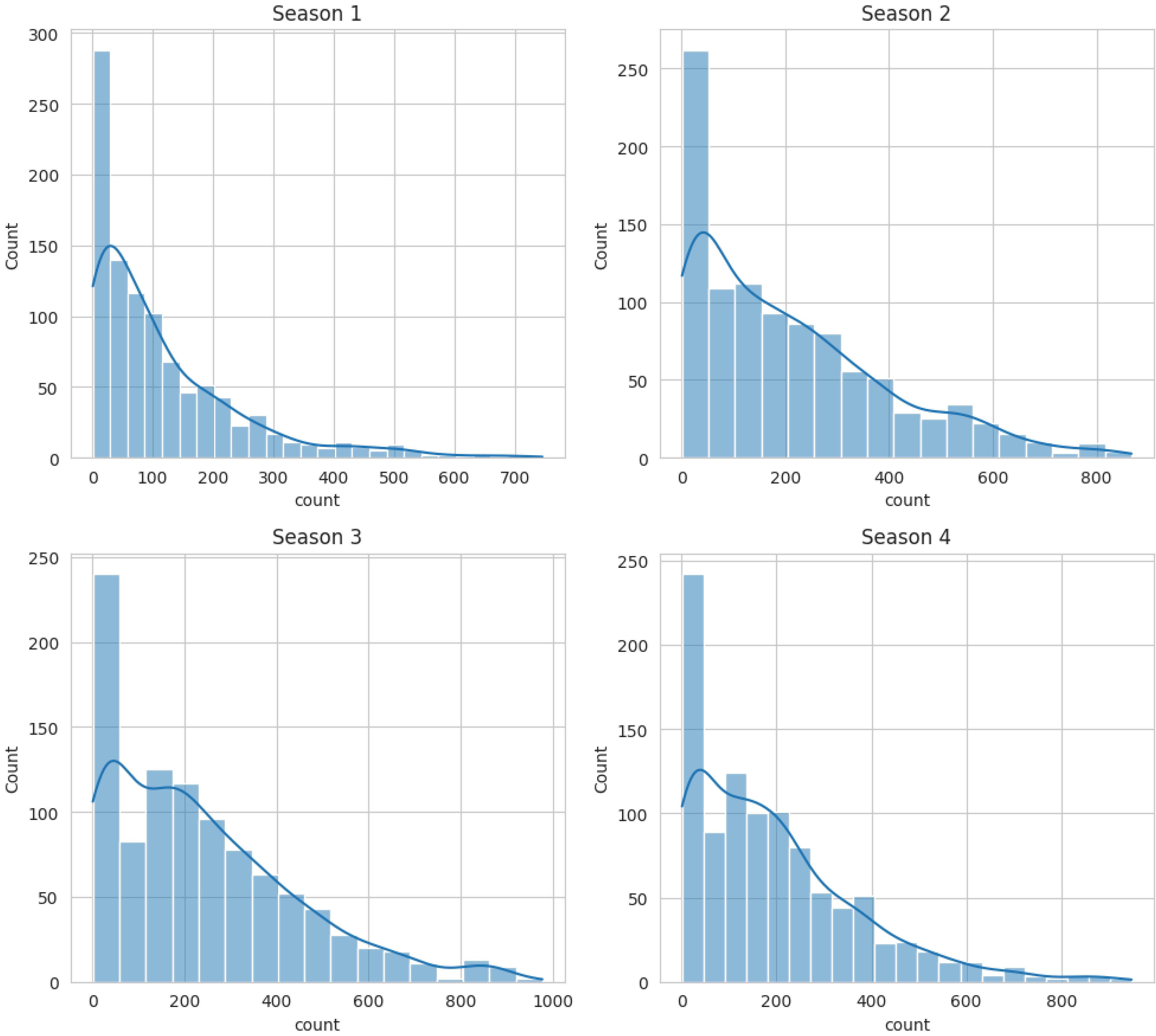
Test assumptions

1. data must be normal
2. its should be independent
3. equality of varience

Normality test

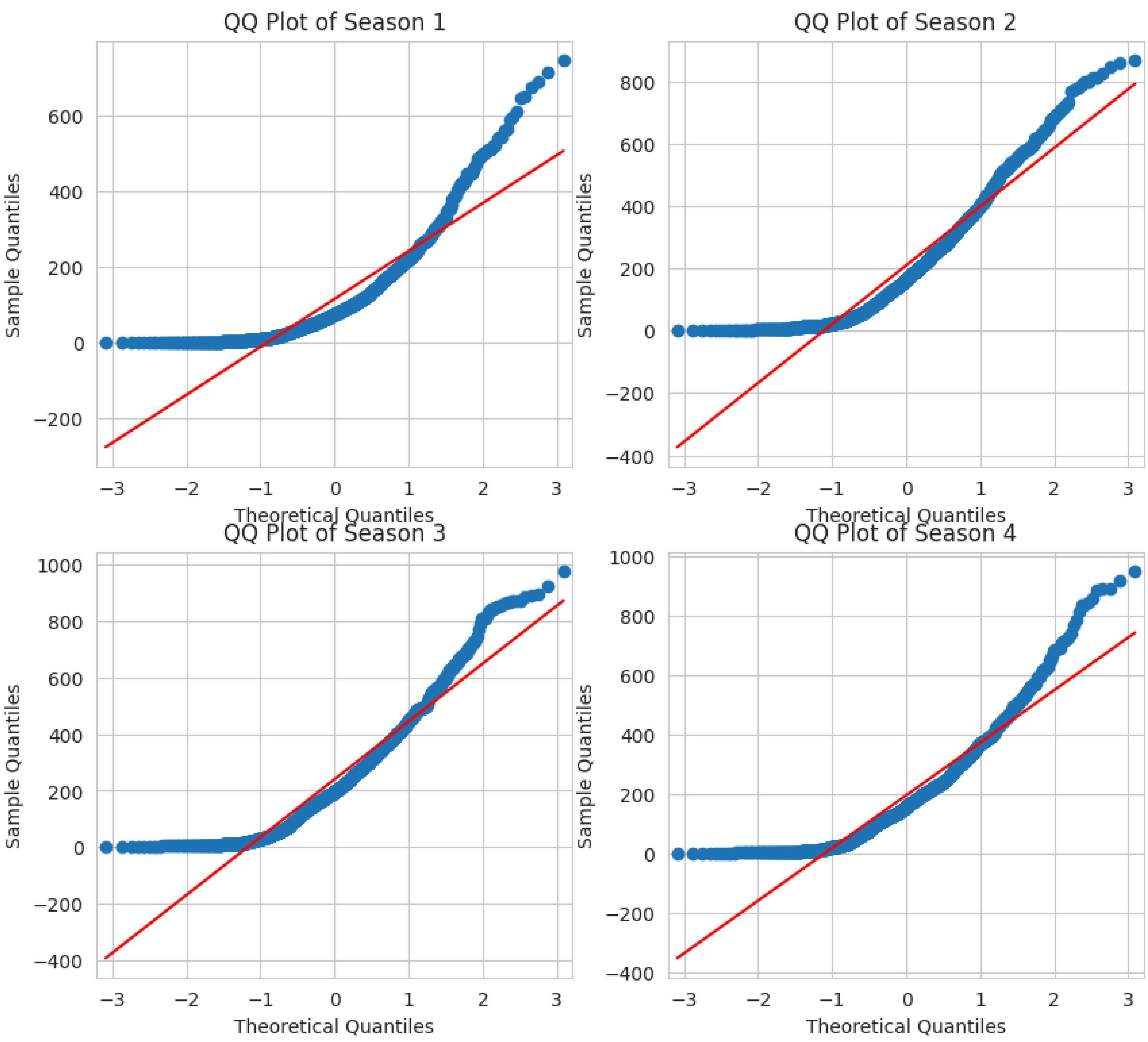
In [77]:

```
import statsmodels.api as sm
all_seasons=[spring,summer,fall,winter]
n_rows, n_cols = 2,2
fig, axes = plt.subplots(n_rows, n_cols, figsize=(10, 9))
axes = axes.flatten()
for idx, data in enumerate(all_seasons):
    sns.histplot(data, kde=True, ax=axes[idx])
    axes[idx].set_title(f"Season {idx+1}")
plt.tight_layout()
```



In [78]:

```
import statsmodels.api as sm # import the statsmodels library
all_seasons = [spring,summer,fall,winter]
n_rows, n_cols = 2, 2
fig, axes = plt.subplots(n_rows, n_cols, figsize=(10, 9))
axes = axes.flatten()

for i in range(len(all_seasons)):
  sm.qqplot(all_seasons[i], line="s", ax=axes[i])
  axes[i].set_title(f"QQ Plot of Season {i + 1}")
```



Shapiro Wilk test of normalacy

In [80]:
```
from scipy.stats import shapiro
for i in all_seasons:
  stat,pvalue=shapiro(i)
  print(f"stat:{stat},pvalue:{pvalue}")
if pvalue<alpha:
  print("the dist is not normal")
else:
  print("the dist is normal")
```

```
stat:0.8077583159243965,pvalue:7.394479963454241e-33
stat:0.8954350778259466,pvalue:1.6385188481885157e-25
stat:0.9089968386491917,pvalue:5.66144634878508e-24
stat:0.8870494506548015,pvalue:2.178231075970518e-26
the dist is not normal
```

levenes test for equality of varience

In [81]:
```
# equality of varience test
from scipy.stats import levene
stat,pvalue=levene(spring,summer,fall,winter)
stat,pvalue
if pvalue<=alpha:
  print("the varience is not equal")
else:
  print("the varience is equal")
```

```
the varience is not equal
```

In [82]:
```
#kw test
from scipy.stats import kruskal
stat,pvalue=kruskal(spring,summer,fall,winter)
stat,pvalue
if pvalue<alpha:
  print("we reject the null hypothesis")
  print("demand of bikes on rent is not the same for different Seasons")
else:
  print("we accept the null hypothesis")
  print("demand of bikes on rent is the same for different Seasons")
```

```
we reject the null hypothesis
demand of bikes on rent is not the same for different Seasons
```

Check if the Weather conditions are significantly different during different Seasons

Hypotheses

Null hypothesis= The weather conditions are independent of season

Alternate hypothesis= The weather conditions are significantly dependent on season

In [83]:
```
#chi_square test
# contigency table
contingency_table=pd.crosstab(df["season"],df["weather"])
contingency_table
```

Out[83]:

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

In [84]:
```
from scipy.stats import chi2_contingency
stat,pvalue,dof,expected=chi2_contingency(contingency_table)
stat,pvalue
```

Out[84]:
```
(np.float64(49.158655596893624), np.float64(1.549925073686492e-07))
```

In [89]:
```
if pvalue<alpha:
  print("we reject the null hypothes is")
  print("The weather conditions are significantly dependent on season")
else:
  print("we fail to reject null hypothes is")
  print ("The weather conditions are independent of season")
```

```
we reject the null hypothes is
The weather conditions are significantly dependent on season
```

observations

- The rejection of the null hypothesis suggests that weather conditions are significantly dependent on the season.
- The demand for bike rentals may fluctuate based on these seasonal weather conditions, such as lower demand during cold or rainy weather and higher demand during sunny or mild weather.

Summary of Hypothesis:

1. Is there any effect of Working Day on the number of electric cycles rented ?

   --The mean hourly count of the total rental bikes is statistically similar for both working and non-working days.

2. Is there any effect of holidays on the number of electric cycles rented ?

   --There is statistically significant dependency of weather and season based on the hourly total number of bikes rented.

3. Is weather dependent on the season ?

   --The hourly total number of rental bikes is statistically different for different weathers.

4. Is the number of cycles rented is similar or different in different weather ?

   --There is no statistically significant dependency of weather 1, 2, 3 on season based on the average hourly total number of bikes rented.

5. Is the number of cycles rented is similar or different in different season ?

   --The hourly total number of rental bikes is statistically different for different seasons.

Recommendations

- **Strategic Seasonal Marketing**: Given the evident seasonal pattern in bike rental counts, Yulu can adapt its marketing strategies strategically. Concentrate on promoting bike rentals during the spring and summer seasons when demand peaks. Consider offering seasonal incentives or exclusive packages to entice more customers during these periods.
- **Dynamic Time-based Pricing**: Leverage the hourly fluctuations in bike rental counts throughout the day. Explore the implementation of dynamic time-based pricing, where rental rates are adjusted to be more affordable during off-peak hours and slightly higher during peak hours. This approach can motivate customers to rent bikes during less congested times, optimizing resource utilization.
- **User-Centric Segmentation**: Considering that approximately 81% of users are registered, and the remaining 19% are casual users, Yulu can tailor its marketing and communication strategies with precision. Offer loyalty programs, exclusive incentives, or personalized recommendations to registered users, fostering repeat business. For casual users, emphasize seamless rental experiences and highlight the advantages of bike rentals for occasional use.
- **Enhanced Weather Data Collection**: Given the limited records for extreme weather conditions, consider enhancing data collection procedures for such scenarios. Accumulating more data on extreme weather conditions can facilitate a deeper understanding of customer behavior, enabling adjustments such as offering specialized bike models for different weather conditions or implementing safety measures during extreme weather.
- **Customer Comfort and Convenience**: With generally high humidity levels and temperatures frequently below 28 degrees Celsius, consider enhancing customer comfort. Provide amenities such as umbrellas, rain jackets, or water bottles to elevate the overall biking experience. These thoughtful touches contribute to a positive customer experience and can encourage repeat business.
- **Social Media Marketing**: Harness the power of social media platforms to promote Yulu's electric bike rental services strategically. Share captivating visuals of biking experiences in diverse weather conditions, showcase customer testimonials, and engage potential customers through interactive posts and contests. Implement targeted advertising campaigns to reach specific customer segments and boost bookings.

In [ ]: