Business Case: "Yulu" - Hypothesis Testing

Dataset Description:

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!

Yulu has recently suffered considerable dips in its revenues. They have contracted a consulting company to understand the factors on which the demand for these shared electric cycles depends. Specifically, they want to understand the factors affecting the demand for these shared electric cycles in the Indian market.

Dataset (Column Profiling):

- **datetime:** datetime
- **season:** season (1: spring, 2: summer, 3: fall, 4: winter)
- **holiday:** whether day is a holiday or not (extracted from http://dchr.dc.gov/page/holiday-schedule (http://dchr.dc.gov/page/holiday-schedule))
- **workingday:** if day is neither weekend nor holiday is 1, otherwise is 0.
- **weather:**
  1: Clear, Few clouds, partly cloudy, 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 Pallets + Thunderstorm + Mist, Snow + Fog
- **temp:** temperature in Celsius
- **atemp:** feeling temperature in Celsius
- **humidity:** humidity
- **windspeed:** wind speed
- **casual:** count of casual users
- **registered:** count of registered users
- **count:** count of total rental bikes including both casual and registered

Dataset Link:

Concept Used:

- Bi-Variate Analysis
- 2-sample t-test: testing for difference across populations
- ANNOVA
- Chi-square

How to begin:

- Import the dataset and do usual exploratory data analysis steps like checking the structure & characteristics of the dataset
- Try establishing a relation between the dependent and independent variable (Dependent “Count” & Independent: Workingday, Weather, Season etc)
- Select an appropriate test to check whether:
  - Working Day has effect on number of electric cycles rented
  - No. of cycles rented similar or different in different seasons
  - No. of cycles rented similar or different in different weather
  - Weather is dependent on season (check between 2 predictor variable)
- Set up Null Hypothesis (H0)
- State the alternate hypothesis (H1)
- Check assumptions of the test (Normality, Equal Variance). You can check it using Histogram, Q-Q plot or statistical methods like levene’s test, Shapiro-wilk test (optional)
  - Please continue doing the analysis even If some assumptions fail (levene’s test or Shapiro-wilk test) but double check using visual analysis and report wherever necessary
- Set a significance level (alpha)
- Calculate test Statistics.
- Decision to accept or reject null hypothesis.
- Inference from the analysis

(1) Defining Problem Statement and performing Exploratory Data Analysis (EDA)

How you can help here?

Yulu is India’s leading micro-mobility service provider, which offers unique vehicles for the daily commute. 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.

In [1]:
```
# Import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

# To ignore the warnings & make code more representable
import warnings
warnings.filterwarnings("ignore")

# Load bike_sharing.csv dataset into a pandas dataFrame
url = "https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/original/bike_sharing.csv?1642089089"
df = pd.read_csv(url)
```

Observations on shape of data, data types of all the attributes, conversion of datatypes (If required), missing value detection, statistical summary

In [2]:
```
# Display the first few rows of the dataset
df.head()
```

Out[2]:

	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

In [3]:
```
# Check the shape of the dataset
df.shape
```

Out[3]:
```
(10886, 12)
```

In [4]:
```
print(f"Rows: {df.shape[0]} \nColumns: {df.shape[1]}")
```

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

In [5]:
```
# Check all the columns of the dataset
df.columns
```

Out[5]:
```
Index(['datetime', 'season', 'holiday', 'workingday', 'weather', 'temp',
       'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count'],
      dtype='object')
```

In [6]:
```
# Check data types of attributes
df.dtypes
```

Out[6]:
```
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
```

In [7]:
```
# Check Data Types and Missing Values
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
```

Observations:

- **No null values** found in the data
- datetime column has **object datatype**, the columns temp, atemp, windspeed has **float datatype** while the rest of the column name has **integer datatype**

In [8]:
```
# Converting the datatype of datetime column from object to datetime
df['datetime'] = pd.to_datetime(df['datetime'])

# Converting the datatype of season, holiday, workingday and weather columns from int to categorical
cat_cols= ['season', 'holiday', 'workingday', 'weather']
for col in cat_cols:
    df[col] = df[col].astype('object')
```

In [9]:
```
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  datetime64[ns]
 1   season      10886 non-null  object        
 2   holiday     10886 non-null  object        
 3   workingday  10886 non-null  object        
 4   weather     10886 non-null  object        
 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: datetime64[ns](1), float64(3), int64(4), object(4)
memory usage: 1020.7+ KB
```

In [10]:
```
# Check for missing values & Outlier Detection
df.isnull().sum()
```

Out[10]:
```
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
```

Observations:

1. There are **no missing value**s in the dataset.
2. **casual** and **registered** attributes might have **outliers** because their **mean** and **median** are very far away to one another and the value of **standard deviation** is also high which tells us that there is high variance in the data of these attributes.

In [11]:
```
# Summary statistics
df.describe()
```

Out[11]:

	temp	atemp	humidity	windspeed	casual	registered	count
count	10886.00000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000
mean	20.23086	23.655084	61.886460	12.799395	36.021955	155.552177	191.574132
std	7.79159	8.474601	19.245033	8.164537	49.960477	151.039033	181.144454
min	0.82000	0.760000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	13.94000	16.665000	47.000000	7.001500	4.000000	36.000000	42.000000
50%	20.50000	24.240000	62.000000	12.998000	17.000000	118.000000	145.000000
75%	26.24000	31.060000	77.000000	16.997900	49.000000	222.000000	284.000000
max	41.00000	45.455000	100.000000	56.996900	367.000000	886.000000	977.000000

In [12]:
```
df.iloc[:, 1:].describe(include='all')
```

Out[12]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
count	10886.0	10886.0	10886.0	10886.0	10886.00000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000
unique	4.0	2.0	2.0	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
top	4.0	0.0	1.0	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
freq	2734.0	10575.0	7412.0	7192.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
mean	NaN	NaN	NaN	NaN	20.23086	23.655084	61.886460	12.799395	36.021955	155.552177	191.574132
std	NaN	NaN	NaN	NaN	7.79159	8.474601	19.245033	8.164537	49.960477	151.039033	181.144454
min	NaN	NaN	NaN	NaN	0.82000	0.760000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	NaN	NaN	NaN	NaN	13.94000	16.665000	47.000000	7.001500	4.000000	36.000000	42.000000
50%	NaN	NaN	NaN	NaN	20.50000	24.240000	62.000000	12.998000	17.000000	118.000000	145.000000
75%	NaN	NaN	NaN	NaN	26.24000	31.060000	77.000000	16.997900	49.000000	222.000000	284.000000
max	NaN	NaN	NaN	NaN	41.00000	45.455000	100.000000	56.996900	367.000000	886.000000	977.000000

Observations:

- Temperature (**temp** and **atemp**) has a **mean** around 20-23°C, with a **standard deviation** of around 7-8°C.
- Column **humidity** has a **mean** of approximately 61.9%, with a **standard deviation** of about 19.2%.
- **Windspeed** column has an **average** of 12.8, with a **standard deviation** of 8.16.
- The **casual** and **registered** columns represent **counts of rentals** by different user types, with **'registered' rentals** having a **higher mean and standard deviation** compared to **'casual' rentals**.
- The **count** column represents the **total rental count**, with a **mean** of 191.57 and a **standard deviation** of 181.14. It ranges from 1 to 977.

In [13]:
```
# Minimum Datetime and Maximum Datetime
print("Minimum Datetime: ", df['datetime'].min(), "\nMaximum Datetime: ", df['datetime'].max())
```

```
Minimum Datetime:  2011-01-01 00:00:00
Maximum Datetime:  2012-12-19 23:00:00
```

In [14]:
```
# Number of Unique Values in each Categorical Columns
df[cat_cols].melt().groupby(['variable', 'value'])[['value']].count()
```

Out[14]:

variable	value	value
holiday	0	10575
	1	311
season	1	2686
	2	2733
	3	2733
	4	2734
weather	1	7192
	2	2834
	3	859
	4	1
workingday	0	3474
	1	7412

(2) Univariate Analysis

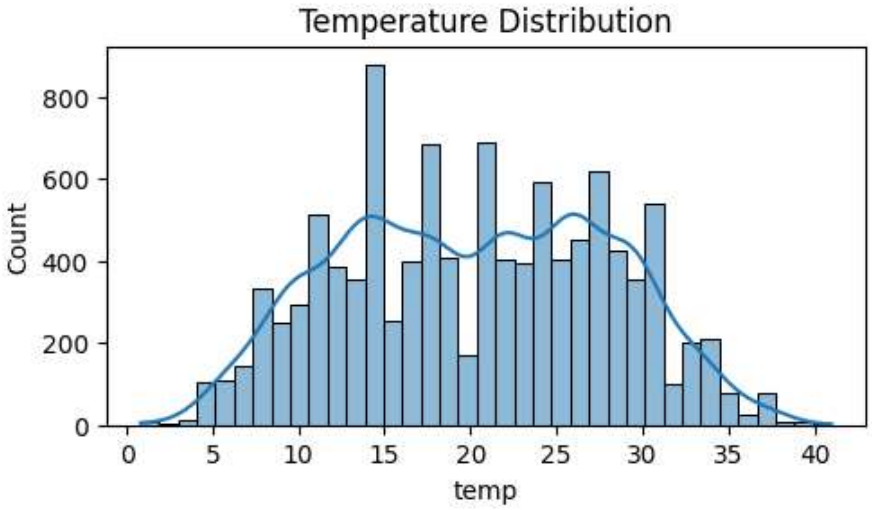
Univariate Analysis (distribution plots of all the continuous variable(s) barplots/countplots of all the categorical variables)

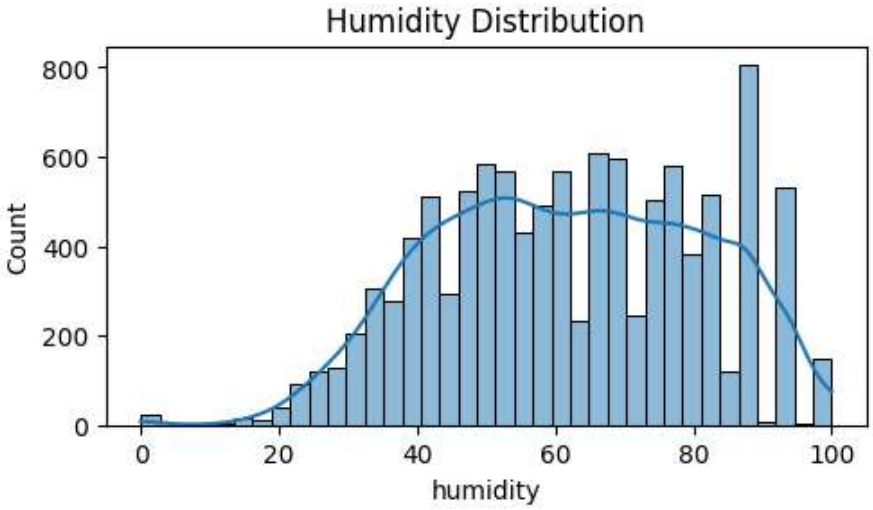
> **NOTE:** Try establishing a relation between the dependent and independent variable(Dependent “Count” & Independent: Workingday, Weather, Season etc)

Understanding the Distribution for Numerical Variables

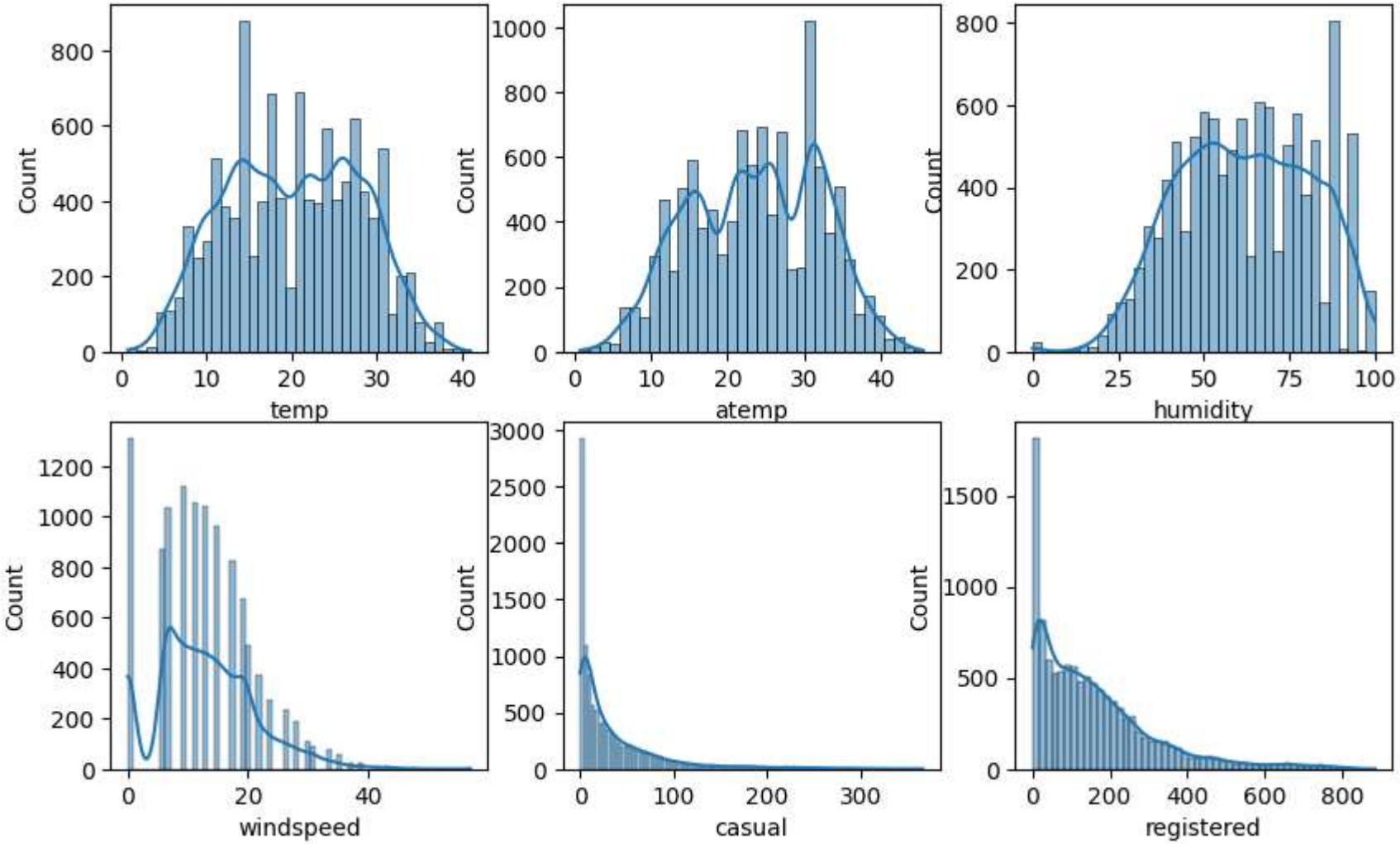
In [15]:
```
# Continuous variables
plt.figure(figsize=(12, 6))
plt.subplot(2, 2, 1)
sns.histplot(df['temp'], kde=True)
plt.title('Temperature Distribution')

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

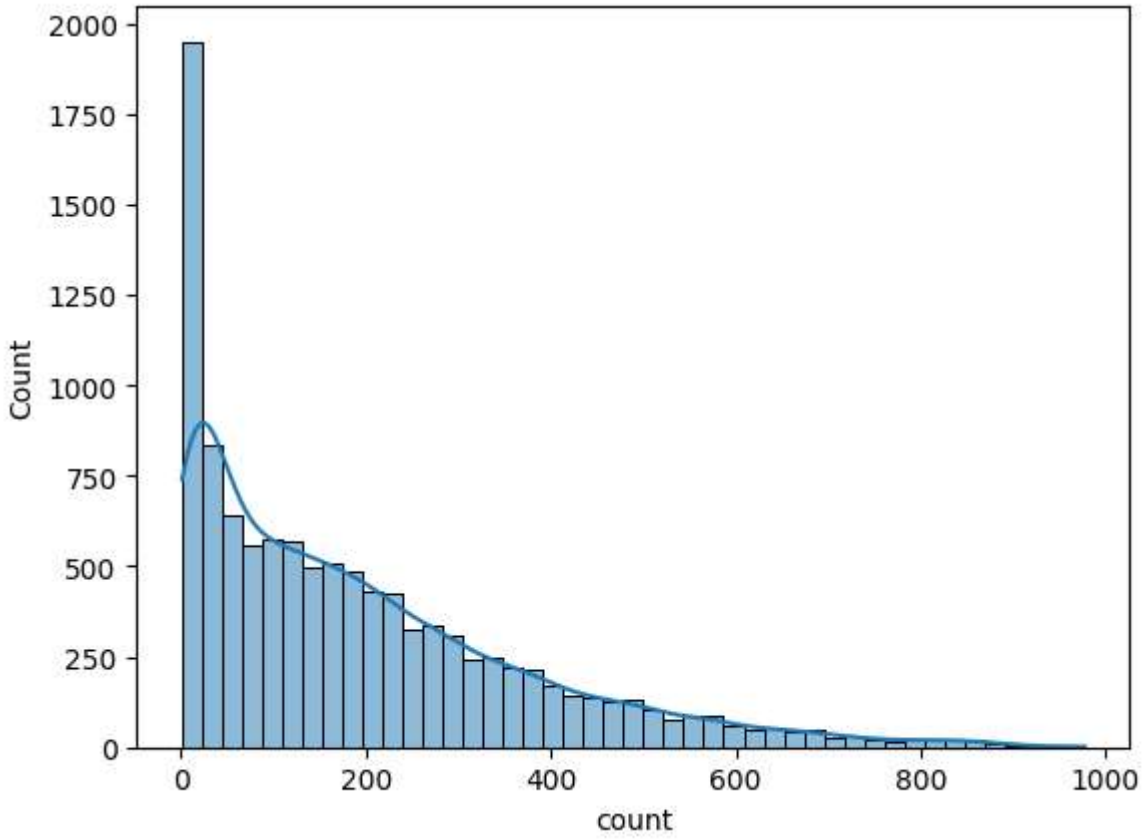
Out[15]:
```
Text(0.5, 1.0, 'Humidity Distribution')
```





In [16]:
```
num_cols = ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered','count']

fig, axis = plt.subplots(nrows=2, ncols=3, figsize=(10, 6))

index = 0
for row in range(2):
    for col in range(3):
        sns.histplot(df[num_cols[index]], ax=axis[row, col], kde=True)
        index += 1

plt.show()
sns.histplot(df[num_cols[-1]], kde=True)
plt.show()
```


Count
temp
atemp
humidity
windspeed
casual
registered

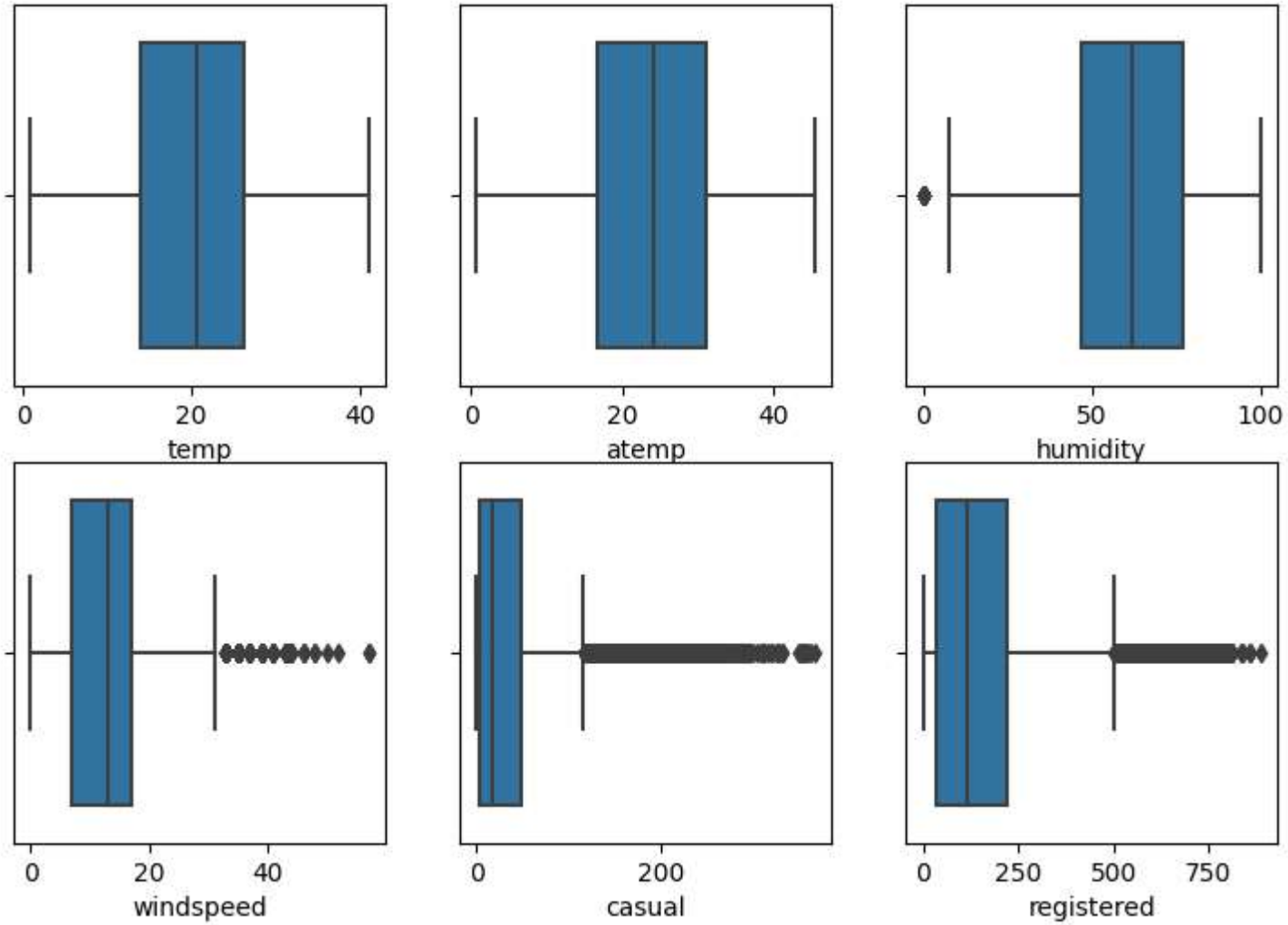


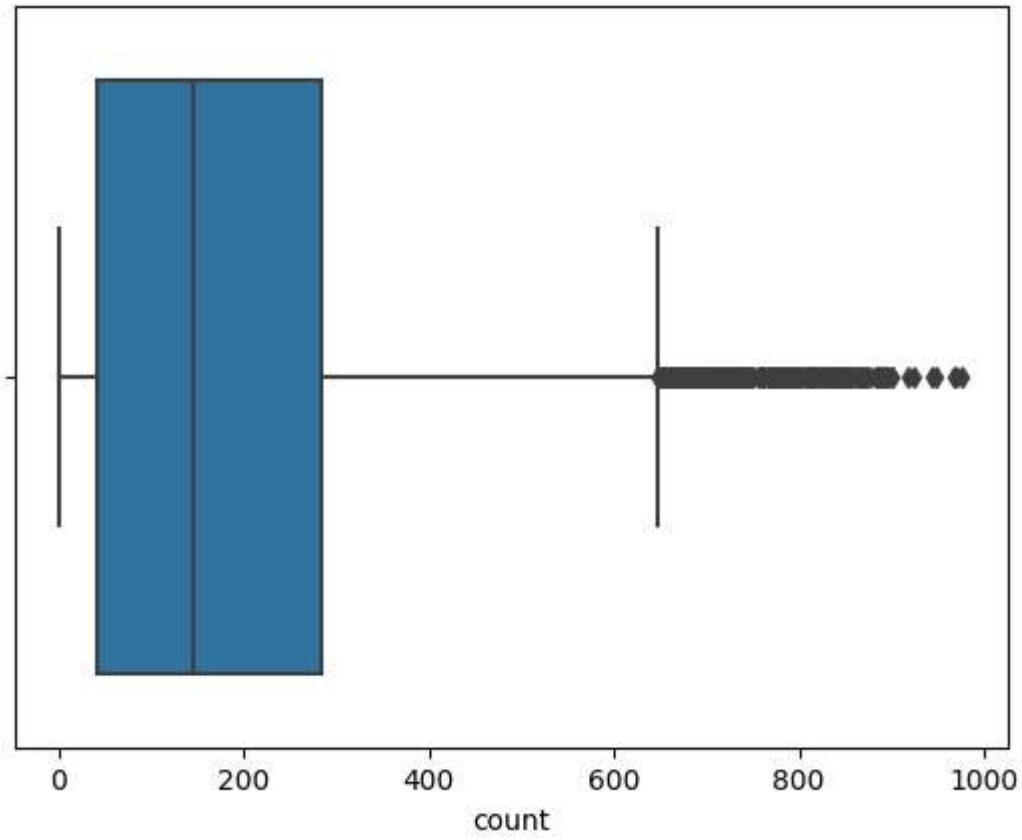
Observations:

- **Casual**, **registered** and **count** somewhat looks like **Log Normal Distributio**n
- **temp**, **atemp** and **humidity** looks like they follows the **Normal Distribution**
- **Windspeed** follows the **binomial distribution**

Plotting Box Plots to Detect Outliers in the Data

In [17]:
```
fig, axis = plt.subplots(nrows=2, ncols=3, figsize=(9, 6))

index = 0
for row in range(2):
    for col in range(3):
        sns.boxplot(x=df[num_cols[index]], ax=axis[row, col])
        index += 1

plt.show()
sns.boxplot(x=df[num_cols[-1]])
plt.show()
```


0
20
40
temp
0
20
40
atemp
0
50
100
humidity
0
20
40
windspeed
0
200
casual
0
250
500
750
registered



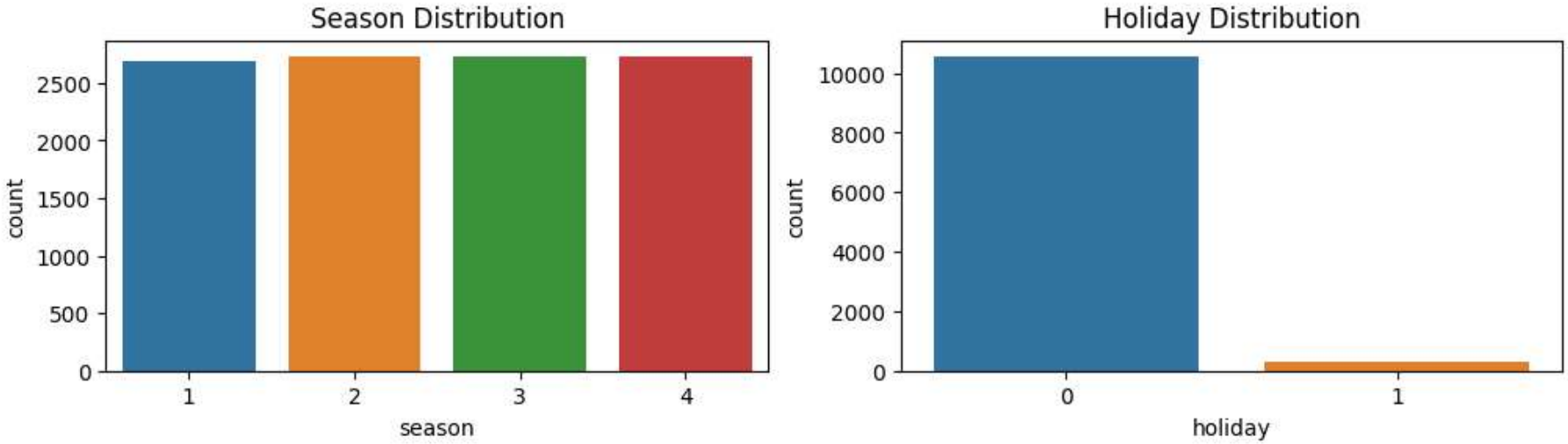
Observations:

- Looks like **humidity**, **casual**, **registered** and **count** have **outliers** in the data.

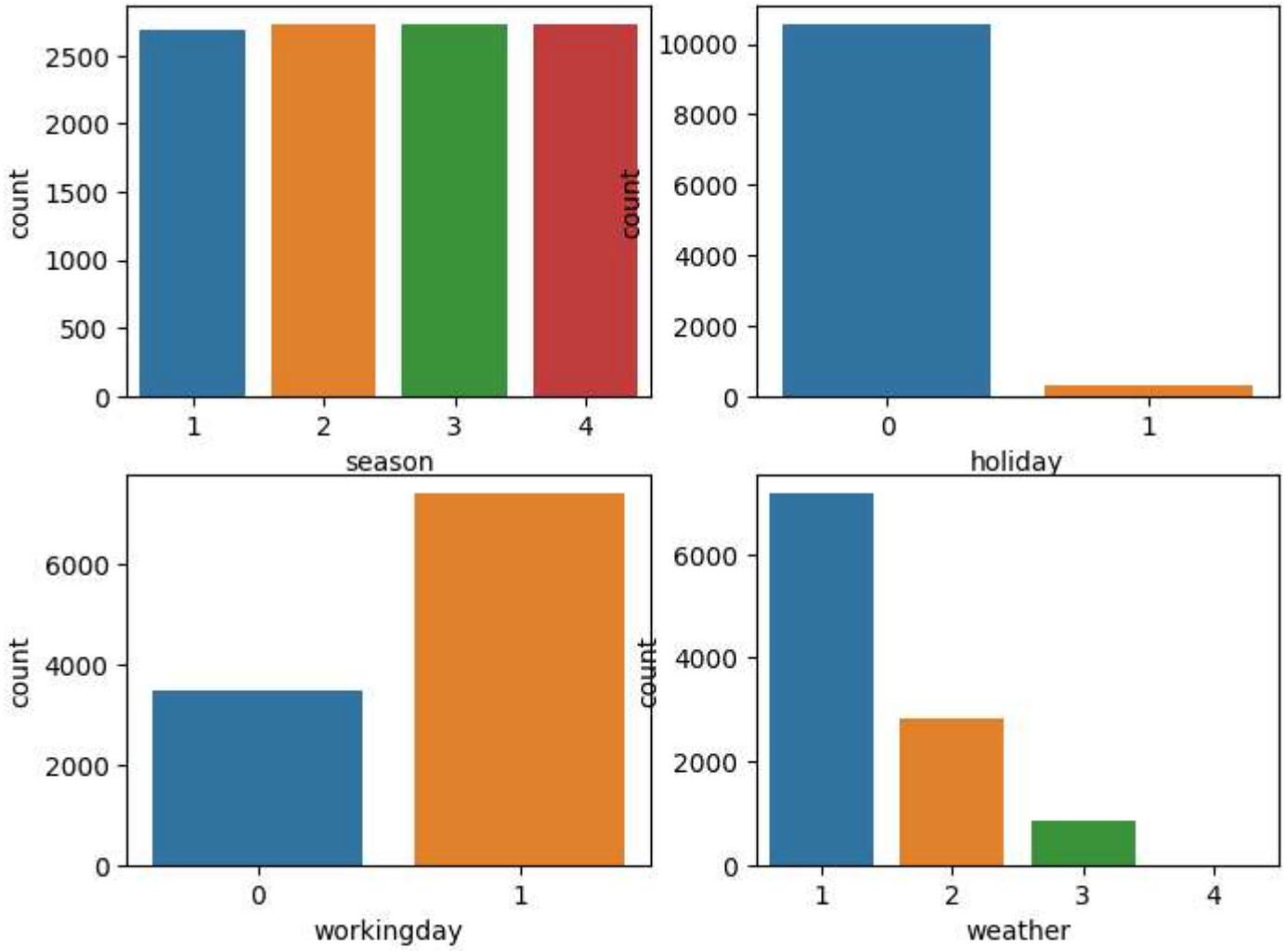
Understanding the Countplot of each Categorical Columns

In [18]:
```
# Categorical variables
plt.figure(figsize=(12, 6))
plt.subplot(2, 2, 1)
sns.countplot(data=df, x='season')
plt.title('Season Distribution')

plt.subplot(2, 2, 2)
sns.countplot(data=df, x='holiday')
plt.title('Holiday Distribution')
```

Out[18]:
```
Text(0.5, 1.0, 'Holiday Distribution')
```



In [19]:
```
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(8, 6))

index = 0
for row in range(2):
    for col in range(2):
        sns.countplot(data=df, x=cat_cols[index], ax=axis[row, col])
        index += 1

plt.show()
```



Observation:

- Data looks common as it should be like equal number of days in each season, more **workingday** and **weather** is mostly Clear, Few clouds, partly cloudy, partly cloudy.

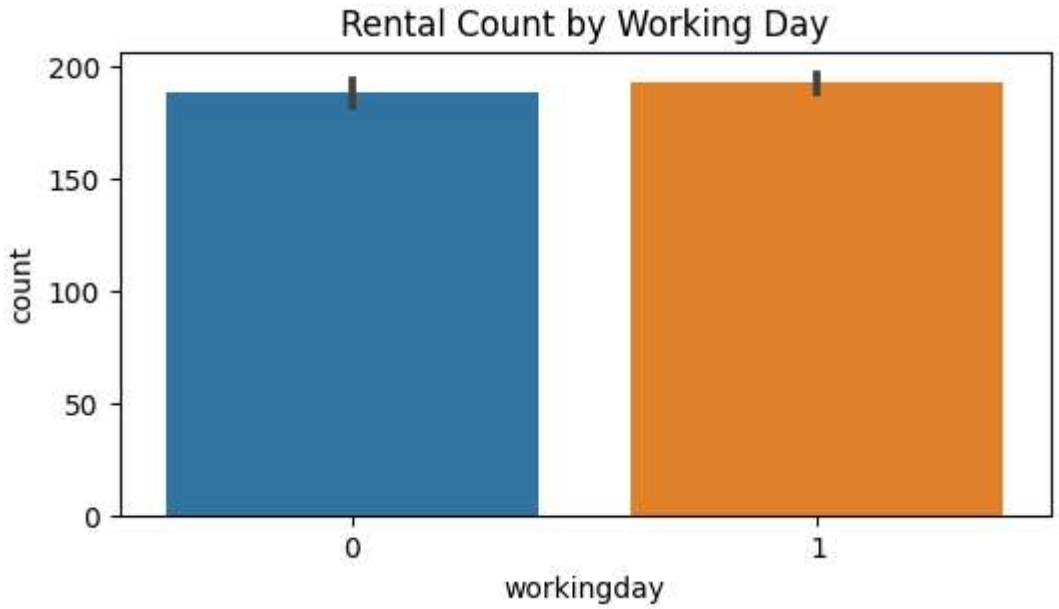
(3) Bivariate Analysis

Bivariate Analysis (Relationships between important variables such as workday and count, season and count, weather and count.

Plotting Categorical Variables againt count using Boxplots

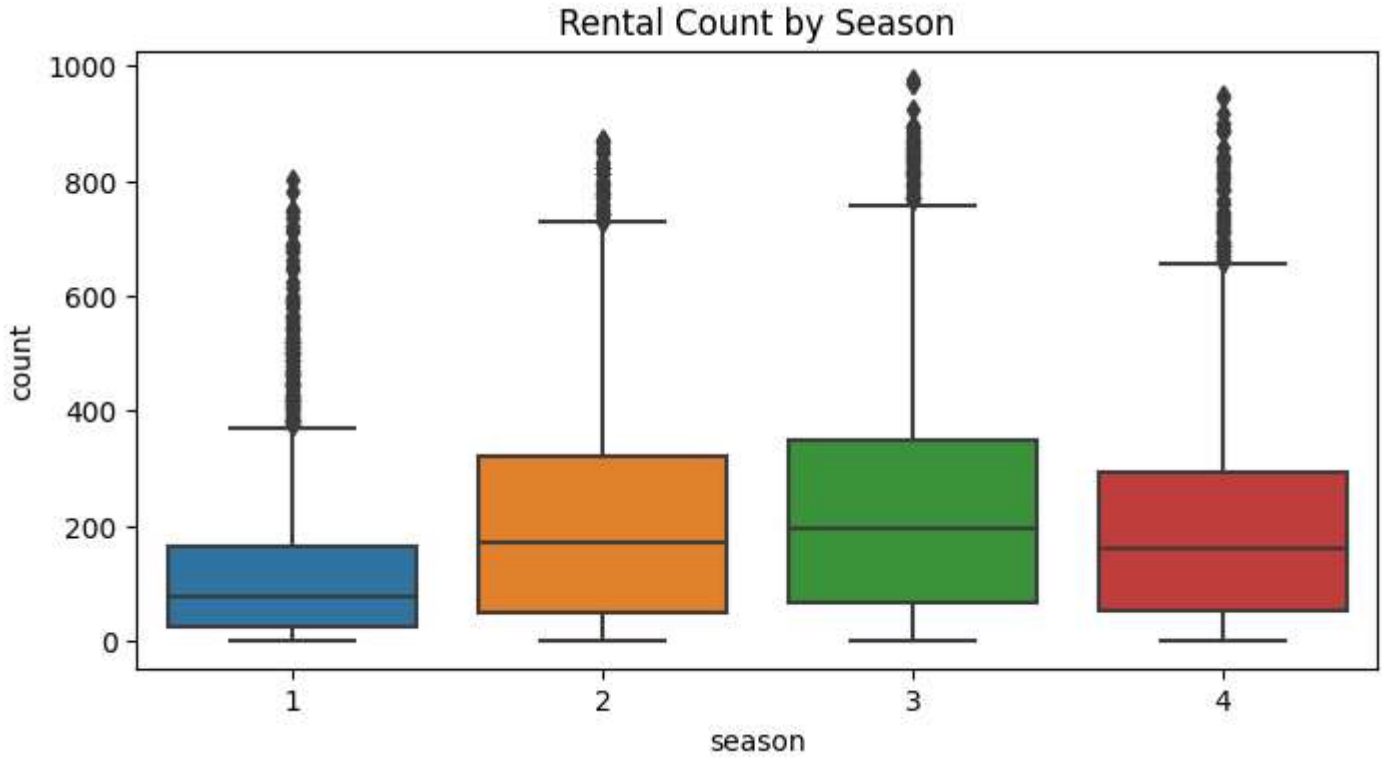
In [20]:
```
# Relationship between 'workingday' and 'count'
plt.figure(figsize=(6, 3))
sns.barplot(data=df, x='workingday', y='count')
plt.title('Rental Count by Working Day')
```

Out[20]:
```
Text(0.5, 1.0, 'Rental Count by Working Day')
```



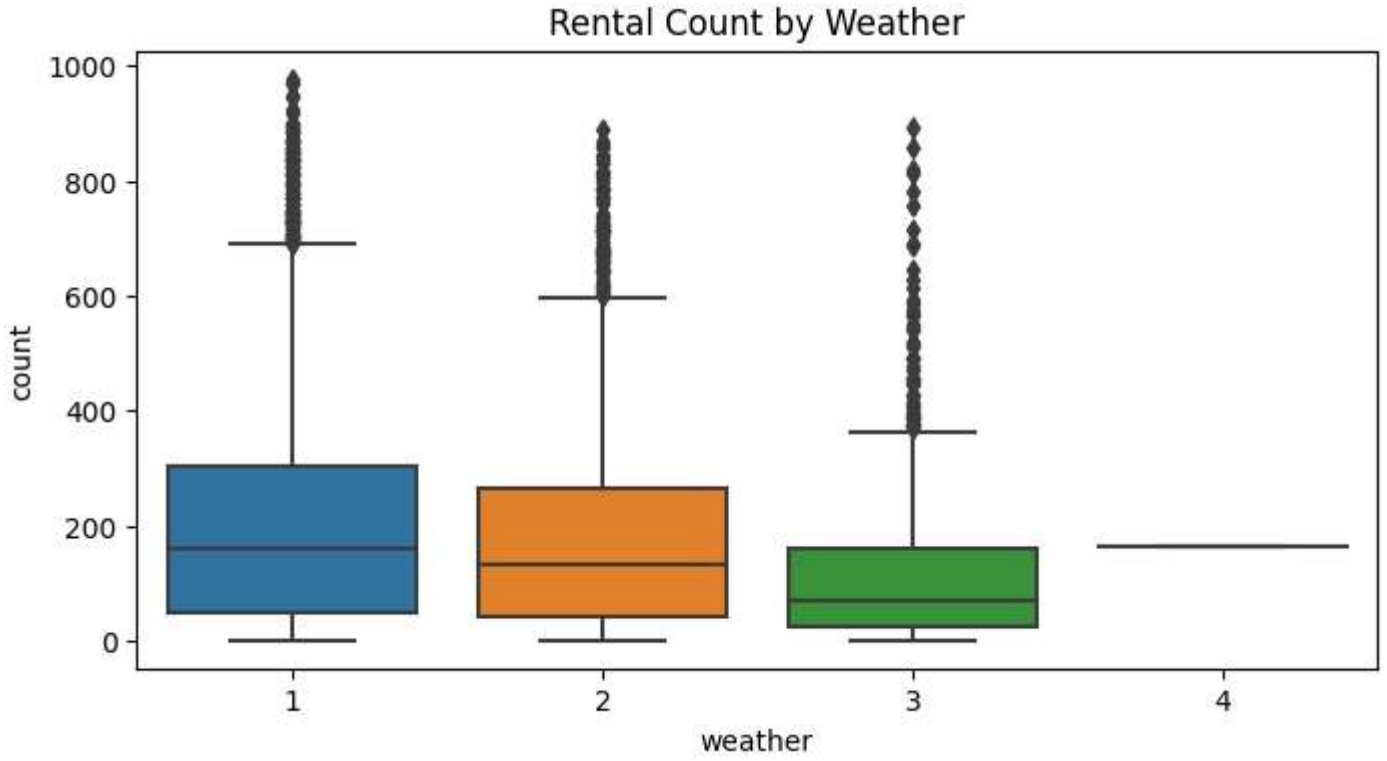
In [21]:
```
# Relationship between 'season' and 'count'
plt.figure(figsize=(8,4))
sns.boxplot(data=df, x='season', y='count')
plt.title('Rental Count by Season')
```

Out[21]:
```
Text(0.5, 1.0, 'Rental Count by Season')
```

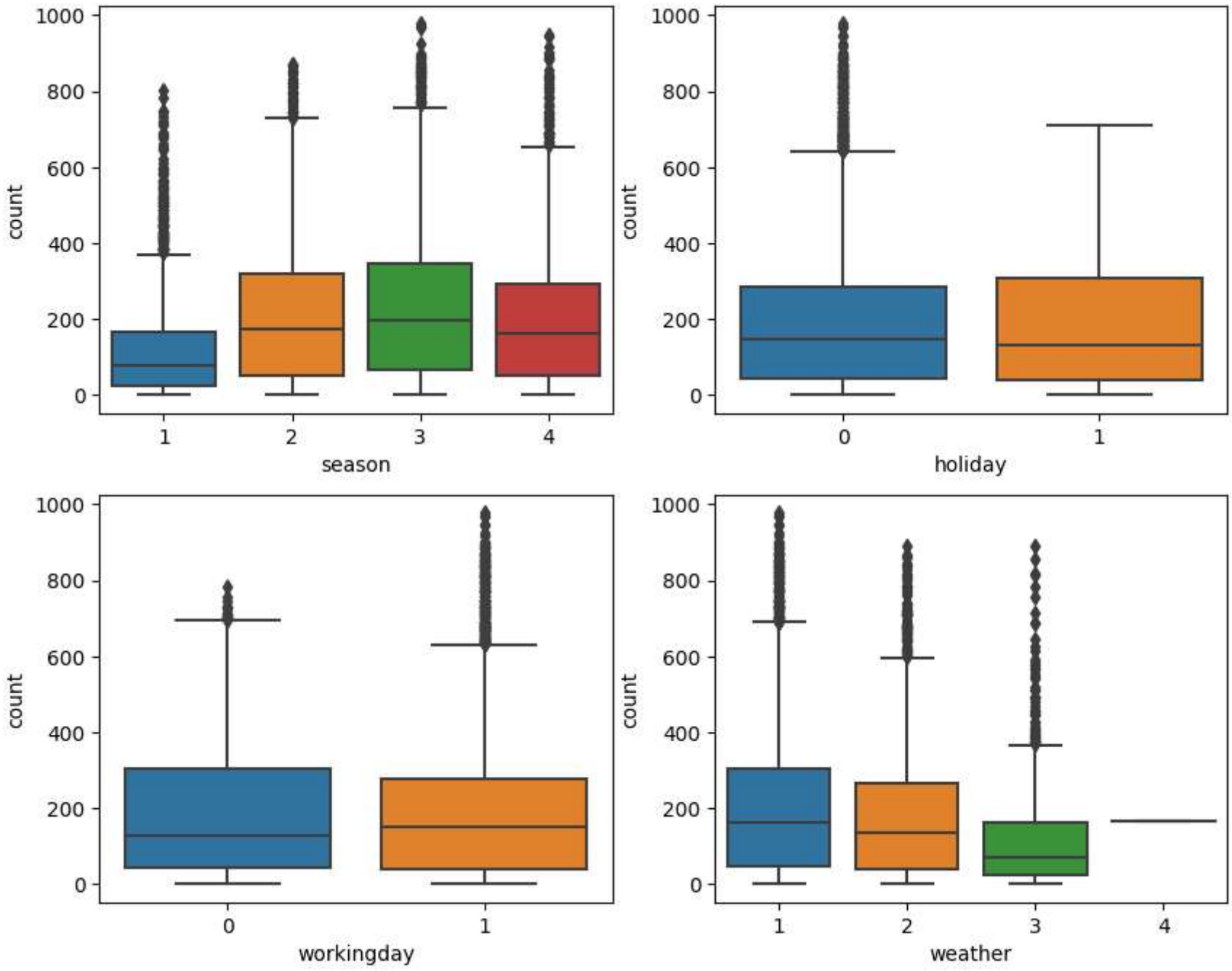


In [22]:
```
# Relationship between 'weather' and 'count'
plt.figure(figsize=(8, 4))
sns.boxplot(data=df, x='weather', y='count')
plt.title('Rental Count by Weather')
```

Out[22]:
```
Text(0.5, 1.0, 'Rental Count by Weather')
```



In [23]:
```
# Plotting Categorical Variables againt count using Boxplots
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(10, 8))

index = 0
for row in range(2):
    for col in range(2):
        sns.boxplot(data=df, x=cat_cols[index], y='count', ax=axis[row, col])
        index += 1

plt.show()
```

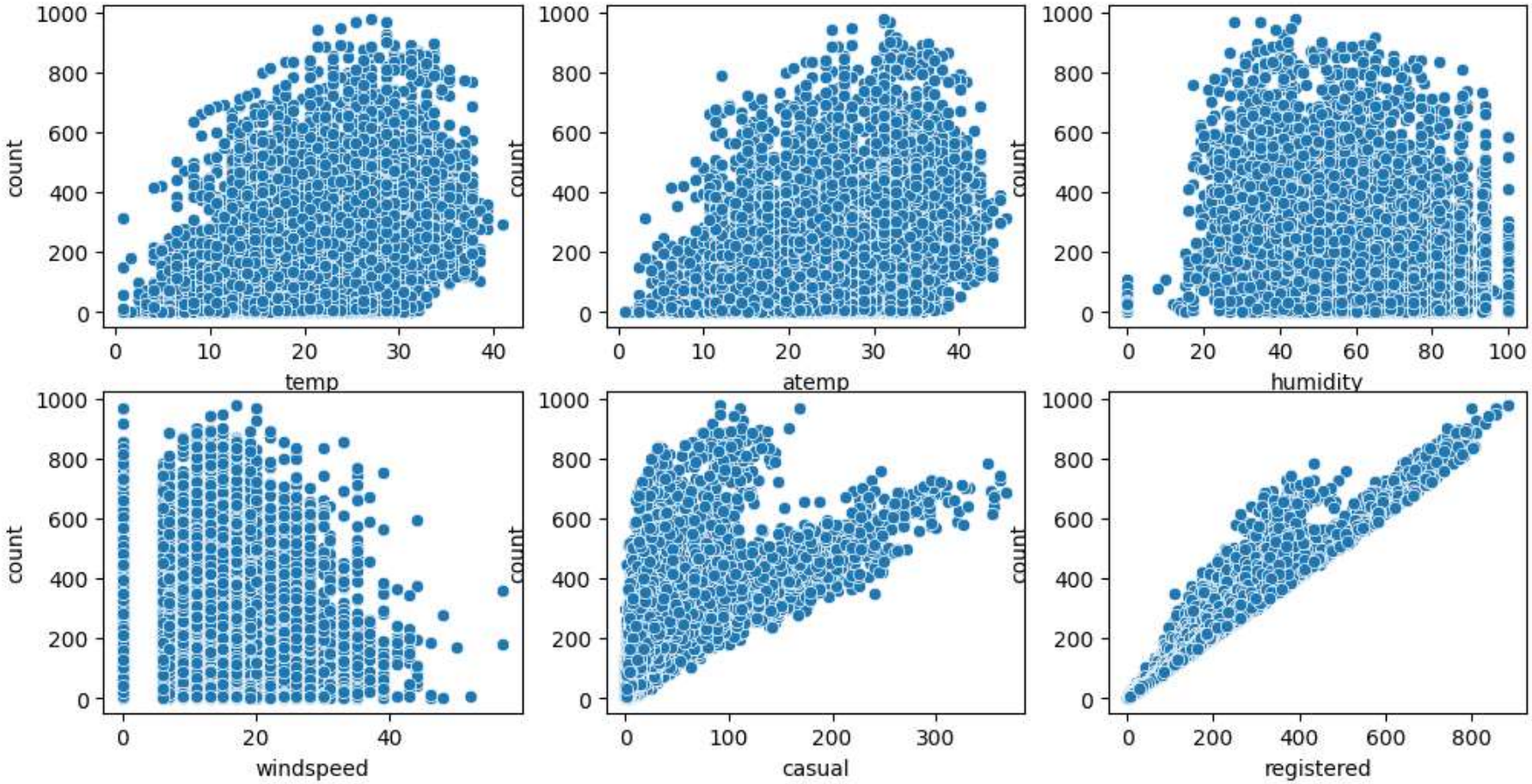

count
season
holiday
workingday
weather
0
200
400
600
800
1000
1
2
3
4

Observations:

- In **summer and fall seasons**, more bikes are rented as compared to other seasons.
- Whenever its a **holiday** more bikes are rented.
- It is also clear from the **workingday** also that whenever day is holiday or weekend, slightly more bikes were rented.
- Whenever there is **rain**, **thunderstorm**, **snow** or **fog**, there were less bikes were rented.

Plotting numerical variables againt count using Scatterplot

In [24]:
```
# Plotting numerical variables againt count using Scatterplot
fig, axis = plt.subplots(nrows=2, ncols=3, figsize=(12, 6))

index = 0
for row in range(2):
    for col in range(3):
        sns.scatterplot(data=df, x=num_cols[index], y='count', ax=axis[row, col])
        index += 1

plt.show()
```

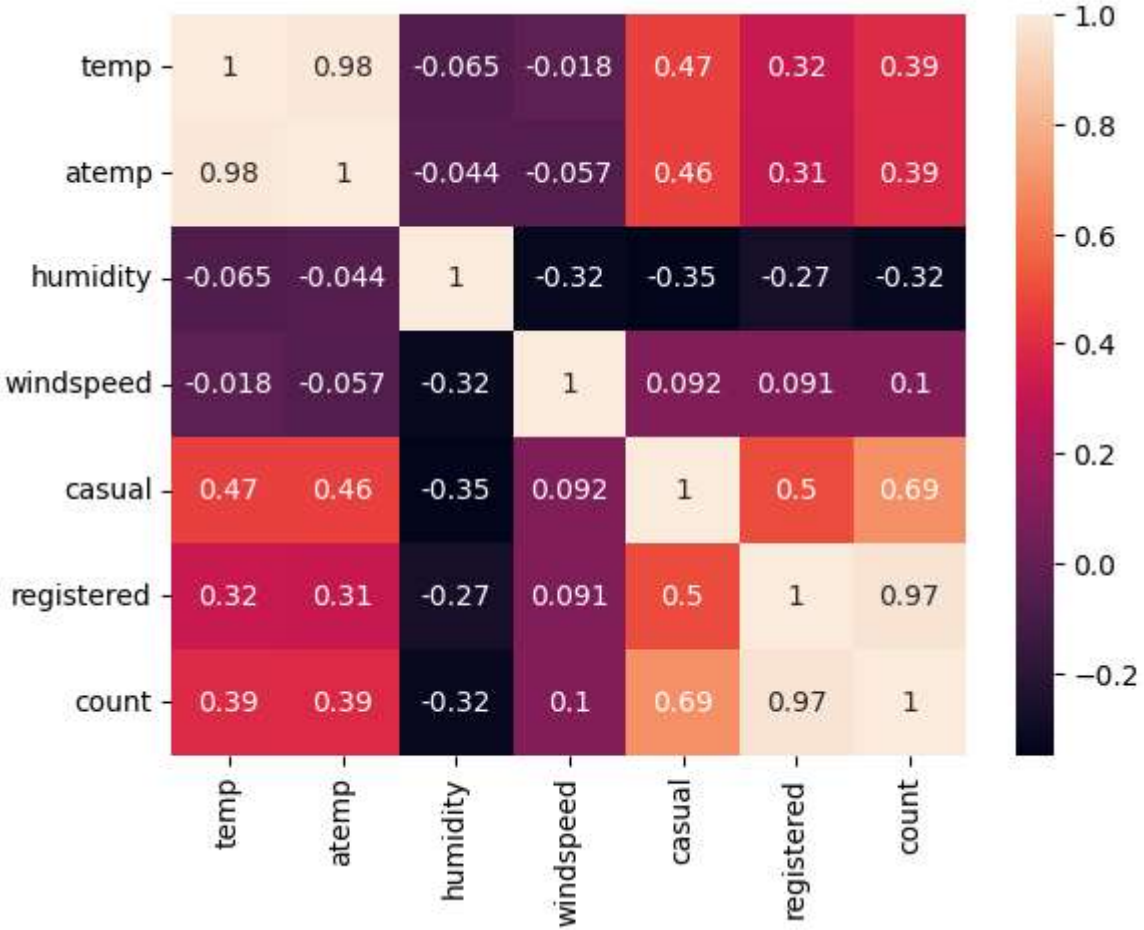


Observations:

- Whenever the **humidity** is **less than 20**, number of bikes rented is very very low.
- Whenever the **temperature** is **less than 10**, number of bikes rented is less.
- Whenever the **windspeed** is **greater than 35**, number of bikes rented is less.

Understanding the Correlation between Count and Numerical Variables

In [25]:
```
df.corr()['count']
sns.heatmap(df.corr(), annot=True)
plt.show()
```



(4) Hypothesis Testing

(4.1) Chi-square test to check if Weather is dependent on the season

Null Hypothesis (H0): Weather is independent of the season
Alternate Hypothesis (H1): Weather is not independent of the season
Significance level (alpha): 0.05

In [26]:
```
data_table = pd.crosstab(df['season'], df['weather'])
print("Observed values:")
data_table
```

```
Observed values:
```

Out[26]:

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 [27]:
```
val = stats.chi2_contingency(data_table)
print(val)
expected_values = val[3]
print(expected_values)
nrows, ncols = 4, 4
dof = (nrows-1)*(ncols-1)
print("degrees of freedom: ", dof)
alpha = 0.05


chi_sqr = sum([(o-e)**2/e for o, e in zip(data_table.values, expected_values)])
chi_sqr_statistic = chi_sqr[0] + chi_sqr[1]
print("chi-square test statistic: ", chi_sqr_statistic)

critical_val = stats.chi2.ppf(q=1-alpha, df=dof)
print(f"critical value: {critical_val}")

p_val = 1-stats.chi2.cdf(x=chi_sqr_statistic, df=dof)
print(f"p-value: {p_val}")

if p_val <= alpha:
    print("\nSince p-value is less than the alpha 0.05, We reject the Null Hypothesis. Meaning that\
    Weather is dependent on the season.")
else:
    print("Since p-value is greater than the alpha 0.05, We do not reject the Null Hypothesis")
```

```
Chi2ContingencyResult(statistic=49.158655596893624, pvalue=1.549925073686492e-07, dof=9, expected_freq=array([[1.7745
4639e+03, 6.99258130e+02, 2.11948742e+02, 2.46738931e-01],
       [1.80559765e+03, 7.11493845e+02, 2.15657450e+02, 2.51056403e-01],
       [1.80559765e+03, 7.11493845e+02, 2.15657450e+02, 2.51056403e-01],
       [1.80625831e+03, 7.11754180e+02, 2.15736359e+02, 2.51148264e-01]]))
[[1.77454639e+03 6.99258130e+02 2.11948742e+02 2.46738931e-01]
 [1.80559765e+03 7.11493845e+02 2.15657450e+02 2.51056403e-01]
 [1.80559765e+03 7.11493845e+02 2.15657450e+02 2.51056403e-01]
 [1.80625831e+03 7.11754180e+02 2.15736359e+02 2.51148264e-01]]
degrees of freedom:  9
chi-square test statistic:  44.09441248632364
critical value: 16.918977604620448
p-value: 1.3560001579371317e-06

Since p-value is less than the alpha 0.05, We reject the Null Hypothesis. Meaning that    Weather is dependent on the
season.
```

(4.2) Sample T-Test to check if Working Day has an effect on the number of electric cycles rented

Null Hypothesis: Working day has no effect on the number of cycles being rented.
Alternate Hypothesis: Working day has effect on the number of cycles being rented.
Significance level (alpha): 0.05

We will use the **2-Sample T-Test** to test the hypothess defined above

In [28]:
```
data_group1 = df[df['workingday']==0]['count'].values
data_group2 = df[df['workingday']==1]['count'].values

print(np.var(data_group1), np.var(data_group2))
np.var(data_group2)// np.var(data_group1)
```

```
30171.346098942427 34040.69710674686
```

Out[28]:
```
1.0
```

Observations:

- Before conducting the **two-sample T-Test** we need to find if the given data groups have the same variance. If the ratio of the larger data groups to the small data group is less than 4:1 then we can consider that the given data groups have equal variance.

- Here, the ratio is 34040.70 / 30171.35 which is less than 4:1

In [29]:
```
stats.ttest_ind(a=data_group1, b=data_group2, equal_var=True)
```

Out[29]:
```
TtestResult(statistic=-1.2096277376026694, pvalue=0.22644804226361348, df=10884.0)
```

Observations:

- Since **pvalue** is greater than 0.05 so **we cannot reject the Null hypothesis**. We don't have the sufficient evidence to say that working day has effect on the number of cycles being rented.

(4.3) ANNOVA to check if No. of cycles rented is similar or different in different (a) weather (b) Season

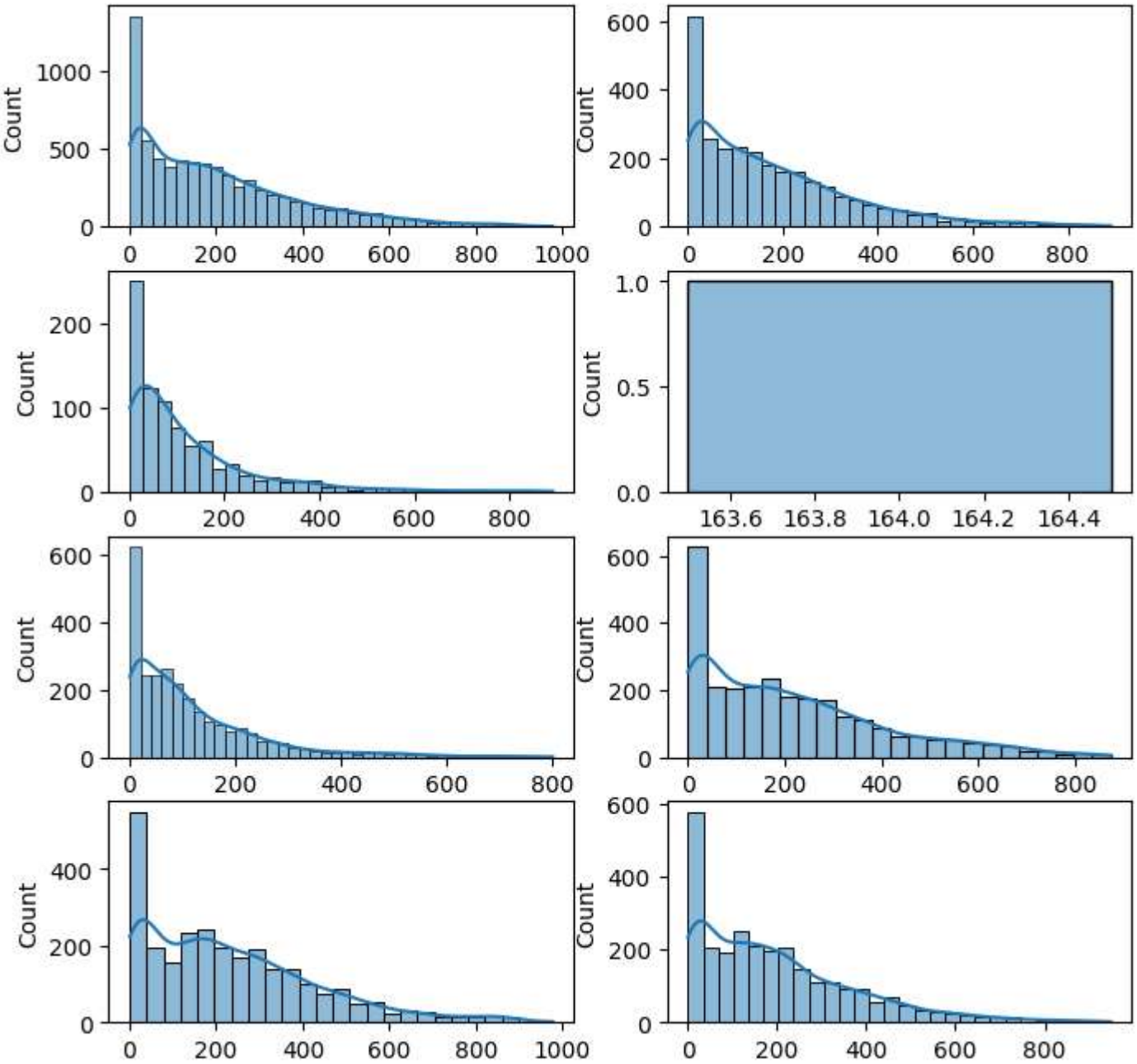
Null Hypothesis: Number of cycles rented is similar in different weather and season. **Alternate Hypothesis:** Number of cycles rented is not similar in different weather and season. **Significance level (alpha):** 0.05

In [30]:
```
# Defining the data groups for the ANOVA
from statsmodels.graphics.gofplots import qqplot
gp1 = df[df['weather']==1]['count'].values
gp2 = df[df['weather']==2]['count'].values
gp3 = df[df['weather']==3]['count'].values
gp4 = df[df['weather']==4]['count'].values

gp5 = df[df['season']==1]['count'].values
gp6 = df[df['season']==2]['count'].values
gp7 = df[df['season']==3]['count'].values
gp8 = df[df['season']==4]['count'].values
groups=[gp1,gp2,gp3,gp4,gp5,gp6,gp7,gp8]
```

Assumptions of ANOVA

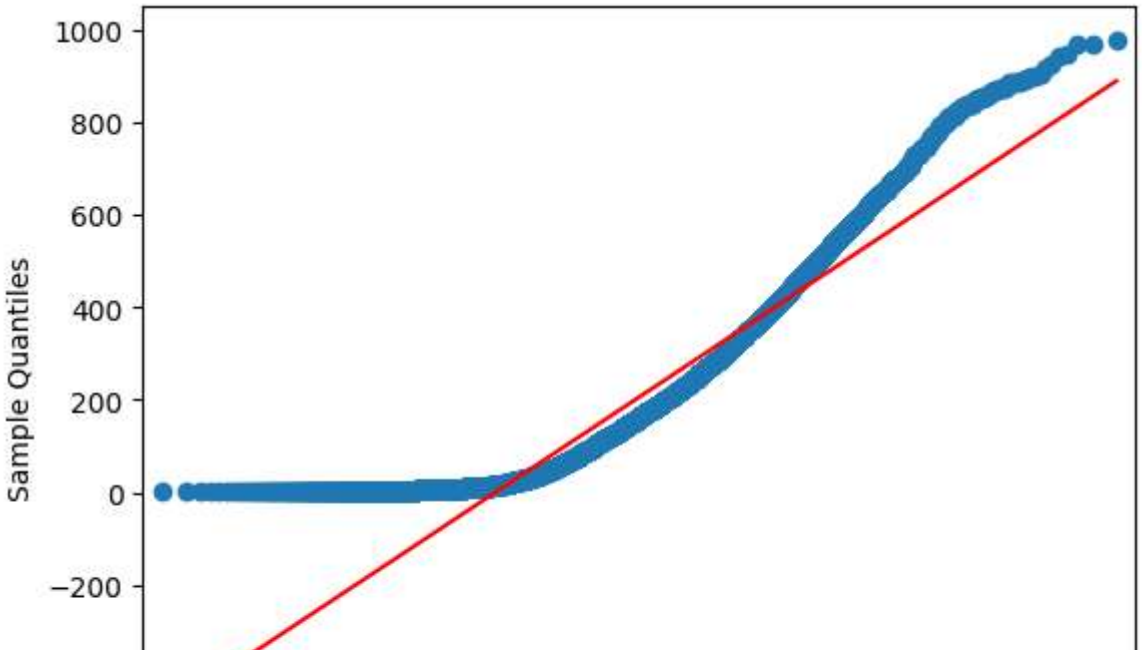
1) Gaussian (Histogram)

In [31]:
```
#  Histogram
fig, axis = plt.subplots(nrows=4, ncols=2, figsize=(8, 8))

index = 0
for row in range(4):
    for col in range(2):
        sns.histplot(groups[index], ax=axis[row, col], kde=True)
        index += 1

plt.show()
```


Count
0
500
1000
0
200
400
600
800
1000
Count
0
200
400
600
800
Count
0
100
200
0
200
400
600
800
Count
0.0
0.5
1.0
163.6
163.8
164.0
164.2
164.4
Count
0
200
400
600
800
0
200
400
600
800
Count
0
200
400
0
200
400
600
800
1000
Count
0
200
400
600
0
200
400
600
800

2) QQ plot

In [32]:
```
index = 0
for row in range(4):
    for col in range(2):
        qqplot(groups[index], line="s")
        index += 1

plt.show()
```



Observations:

- As per above graphs, all groups are **not following Gaussian distribution**
- Data is **Independent**

(4.4) Equal variance: Levene's Test

Null Hypothesis: Variances is similar in different weather and season.
Alternate Hypothesis: Variances is not similar in different weather and season.
Significance level (alpha): 0.05

In [33]:
```
levene_stat, p_value = stats.levene(gp1,gp2,gp3,gp4,gp5,gp6,gp7,gp8)
print(p_value)
if p_value < 0.05:
    print("Reject the Null hypothesis.Variances are not equal")
else:
  print("Fail to Reject the Null hypothesis.Variances are  equal")
```

```
3.463531888897594e-148
Reject the Null hypothesis.Variances are not equal
```

Observations:

- **p_value:** 3.463531888897594e-148
- **Reject the Null hypothesis**. Variances are not equal
- As per **QQ plot** and **Levene's Test**, we **cannot use ANOVA Test**.

(4.5) Assumptions of ANOVA fail, use "Kruskal Wallis"

In [34]:
```
kruskal_stat, p_value = stats.kruskal(gp1,gp2,gp3,gp4,gp5,gp6,gp7,gp8)
print("p_value===",p_value)
if p_value<0.05:
  print("Since p-value is less than 0.05, we reject the null hypothesis")
```

```
p_value=== 4.614440933900297e-191
Since p-value is less than 0.05, we reject the null hypothesis
```

Observations:

- **p_value** = 4.614440933900297e-191

- Since p-value is less than 0.05, **we reject the null hypothesis**. This implies that Number of cycles rented is not similar in different **weather** and **season** conditions.

(5) Final Insights

1. In summer and fall seasons more bikes are rented as compared to other seasons.
2. Whenever its a holiday more bikes are rented.
3. It is also clear from the workingday also that whenever day is holiday or weekend, slightly more bikes were rented.
4. Whenever there is rain, thunderstorm, snow or fog, there were less bikes were rented.
5. Whenever the humidity is less than 20, number of bikes rented is very very low.
6. Whenever the temperature is less than 10, number of bikes rented is less.
7. Whenever the windspeed is greater than 35, number of bikes rented is less.

(6) Final Recommendations (Actionable Items)

"Yulu" is **India’s leading micro-mobility service provider**, which offers unique vehicles for the daily commute.

After having a complete analysis of the **Yulu** dataset, some of the key **actionable recommendations** are shown below:

1. In **summer and fall seasons** the company **should have more bikes in stock to be rented**. Because the demand in these seasons is higher as compared to other seasons.
2. With a **significance level** of 0.05, **workingday** has **no effect** on the number of bikes being rented.
3. In **very low humid days**, company should have **less bikes** in the stock to be rented.
4. Whenever **temperature** is less than 10 or in **very cold days**, company should have **less bikes**.
5. Whenever the **windspeed** is greater than 35 or in thunderstorms, company should have **less bikes in stock to be rented**.

These recommendations provide actionable steps for **Yulu** to **enhance customer satisfaction** and **increase sales** while accommodating the diverse preferences and behaviors of its customer base.