About Yulu

Yulu is India’s leading micro-mobility service provider, which offers unique vehicles for the daily commute. Starting off as a mission to eliminate traffic congestion in India, Yulu provides the safest commute solution through a user-friendly mobile app to enable shared, solo and sustainable commuting.

Yulu zones are located at all the appropriate locations (including metro stations, bus stands, office spaces, residential areas, corporate offices, etc) to make those first and last miles smooth, affordable, and convenient!

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.

Problem Statement

Understanding Variables which are significant in predicting the demand for shared electric cycles in the Indian market

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import ttest_ind, kruskal, f_oneway, ttest_ind, levene, shap
from statsmodels.graphics.gofplots import qqplot
```

```
In [4]: df = pd.read_csv(r'C:\Users\mayank.khanduja\Downloads\Yulu.csv')
        df
```

Out[4]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed
0	1/1/2011 0:00	1	0	0	1	9.84	14.395	81	0.0000
1	1/1/2011 1:00	1	0	0	1	9.02	13.635	80	0.0000
2	1/1/2011 2:00	1	0	0	1	9.02	13.635	80	0.0000
3	1/1/2011 3:00	1	0	0	1	9.84	14.395	75	0.0000
4	1/1/2011 4:00	1	0	0	1	9.84	14.395	75	0.0000
...	...	...	...	...	...	...	...	...	...
10881	12/19/2012 19:00	4	0	1	1	15.58	19.695	50	26.0027
10882	12/19/2012 20:00	4	0	1	1	14.76	17.425	57	15.0013
10883	12/19/2012 21:00	4	0	1	1	13.94	15.910	61	15.0013
10884	12/19/2012 22:00	4	0	1	1	13.94	17.425	61	6.0032
10885	12/19/2012 23:00	4	0	1	1	13.12	16.665	66	8.9981

10886 rows × 12 columns

```
In [5]: df.shape
```

```
Out[5]: (10886, 12)
```

```
In [7]: df.nunique()
```

```
Out[7]: datetime      10886
        season            4
        holiday           2
        workingday        2
        weather           4
        temp             49
        atemp            60
        humidity         89
        windspeed        28
        casual          309
        registered      731
        count           822
        dtype: int64
```

In [8]:
```python
df.isna().sum()
# No Null values in dataset
```

Out[8]:
```
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
```

In [9]:
```python
df.describe()
```

Out[9]:

	season	holiday	workingday	weather	temp	atemp	
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000	108
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	1

In [10]:
```
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 [11]:
```
# converting Dtypes
df['datetime'] = pd.to_datetime(df['datetime'])

for i in ['season', 'holiday', 'workingday', 'weather']:
    df[i] = df[i].astype('object')
```

In [12]:
```
# extracting month and year
df['month']=df['datetime'].dt.month
df['year']=df['datetime'].dt.year
```

```
In [13]: df.head()
```

Out[13]:

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

Univariate Analysis

```
In [14]: # segregating categorical and numeric columns
cat_cols= ['season', 'holiday', 'workingday', 'weather']
num_cols = ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered',
```

In [16]:
```
df[cat_cols].melt().groupby(['variable', 'value'])[['value']].count()
```

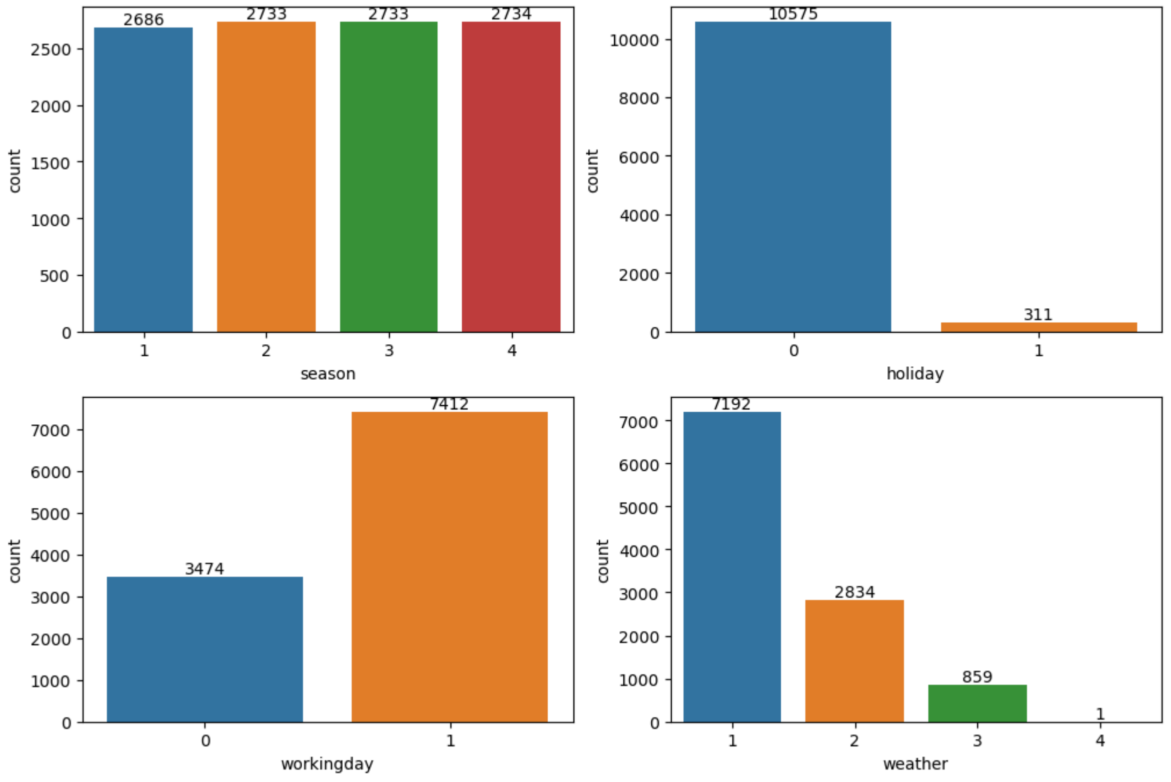
Out[16]:

```
                   value
  variable  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
```

In [20]:
```
#graph for the same
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(12, 8))

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

plt.show()
```



Holiday

1. With 10,575 rows of data, it suggests that the majority of the sales data occurs on non-holiday days.

Season

1. Spring (2,686 entries), Summer (2,733 entries), Fall (2,733 entries), Winter (2,734 entries): The sales data is fairly evenly distributed across all seasons. This implies a steady demand for products throughout the year, with no particular season showing a significant spike or drop in sales.

Weather

1. Clear/Few Clouds/Partly Cloudy (7,192 entries): The majority of sales occur during clear or partly cloudy weather, suggesting that good weather may positively influence customer activity and sales.
2. Mist/Cloudy/Broken Clouds/Few Clouds/Complete Mist (2,834 entries): A significant amount of sales still happens during misty or cloudy conditions, indicating that moderate weather changes don't drastically affect sales.
3. Light Snow/Light Rain + Thunderstorm + Scattered Clouds/Light Rain + Scattered Clouds (860 entries): Sales decrease in less favorable weather conditions like light snow or rain, but are still notable.
4. Heavy Rain + Ice Pallets + Thunderstorm + Mist/Snow + Fog (1 entry): Extreme weather conditions have a substantial negative impact on sales, as seen from the minimal data entries.

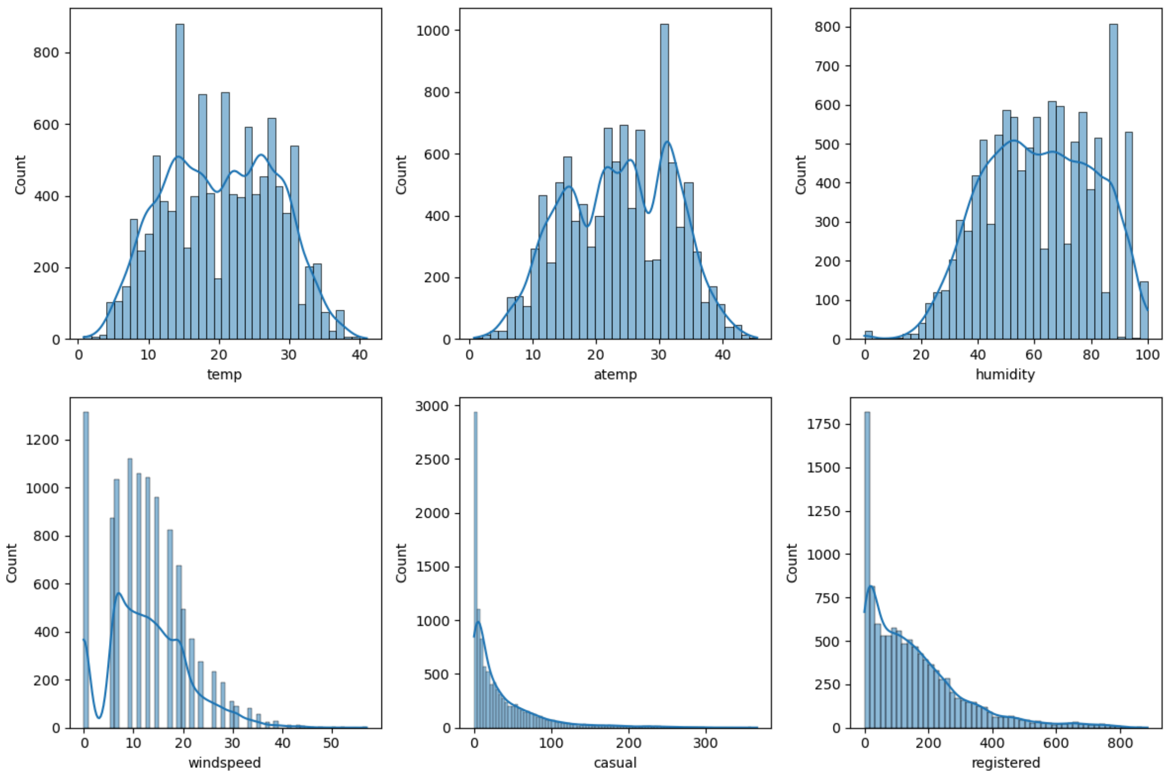
Working Day

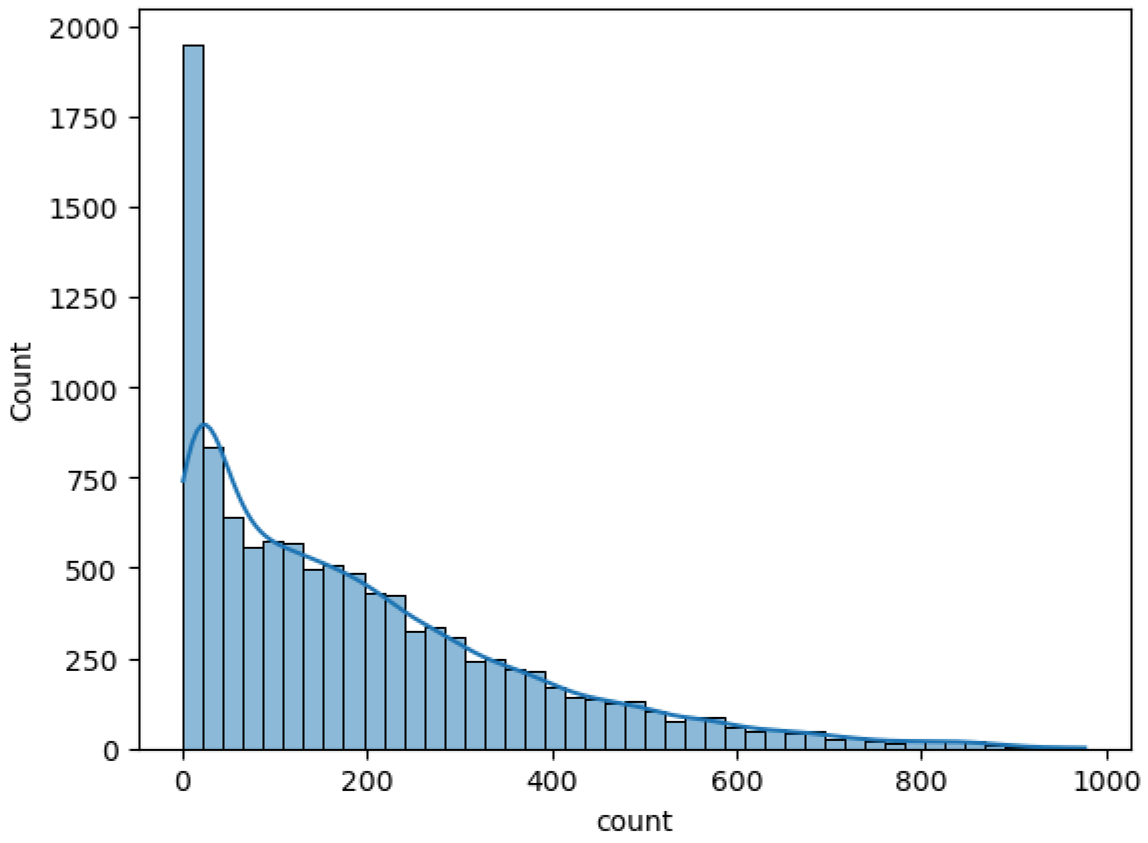
1. Working Day (7,412 entries): Sales are significantly higher on working days, suggesting

In [22]:
```python
# Distributions
fig, axis = plt.subplots(nrows=2, ncols=3, figsize=(12, 8))

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





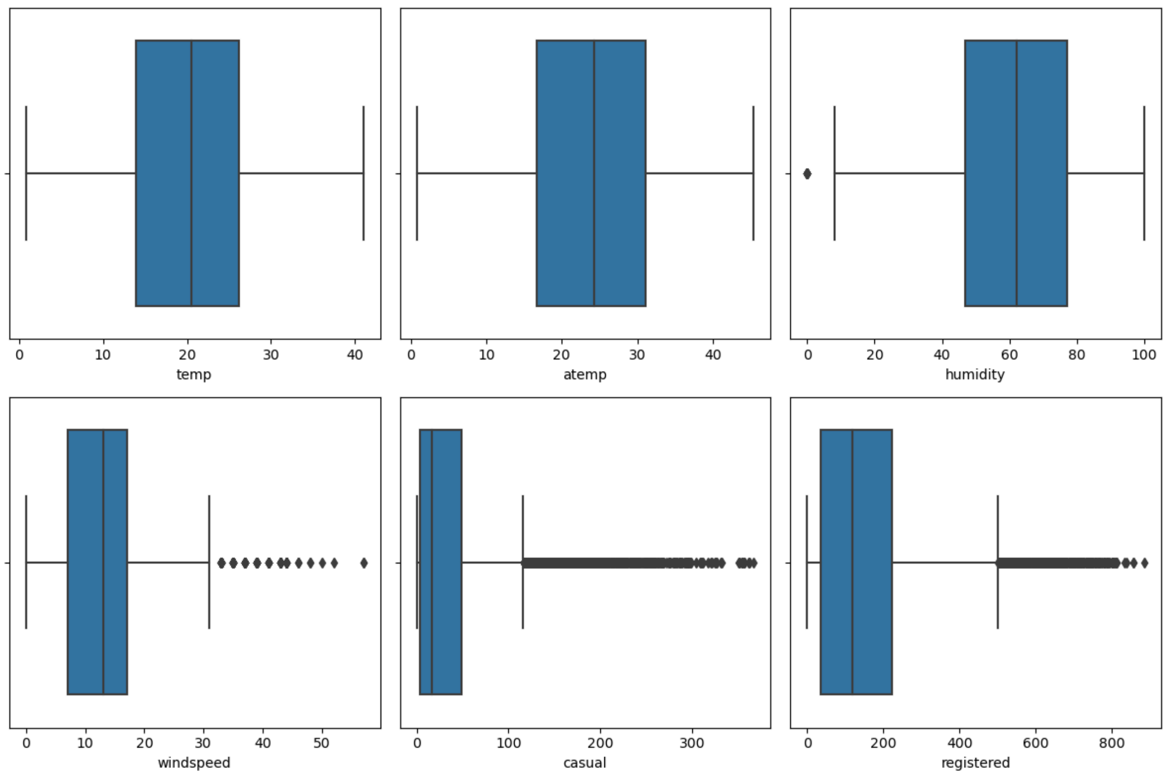
Insights

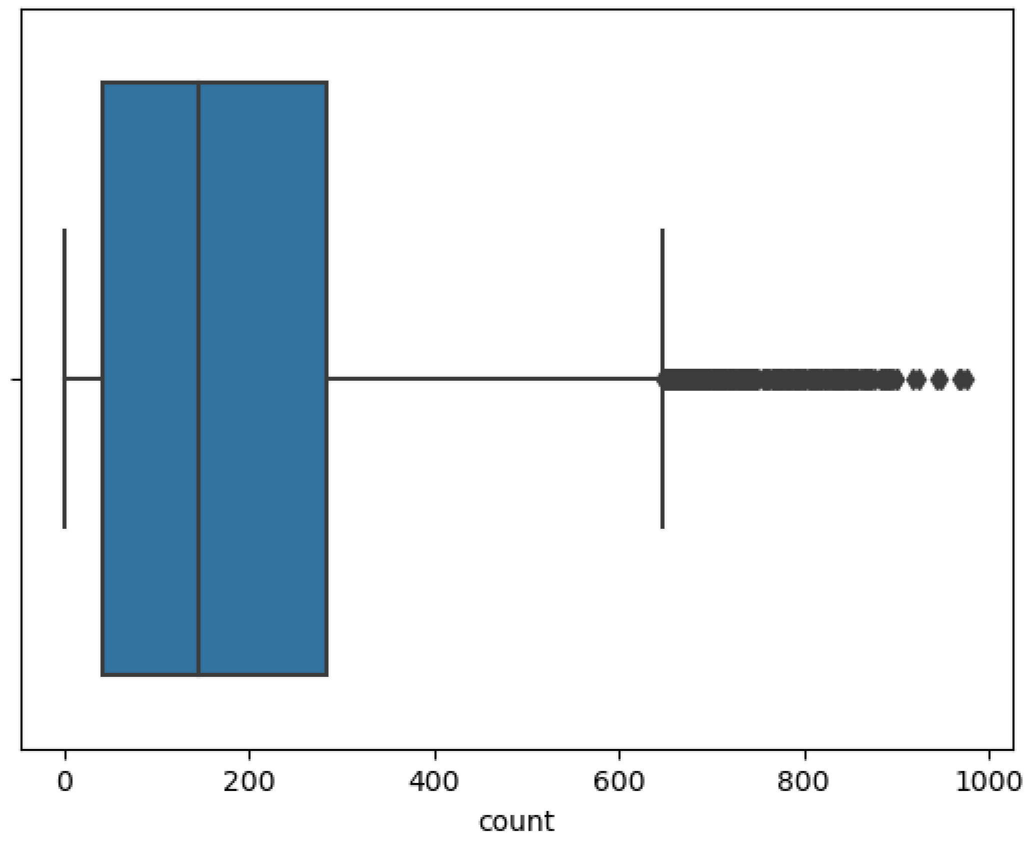
1. Graphs for Casual, registered and Count of bikes are resembles like a Log Normal Distribution.
2. Graphs for Temp, atemp and humidity looks like they follow a Multimodal Normal Distribution.
3. Windspeed seems to have Binomial distribution.

In [23]:
```python
# Outliers
fig, axis = plt.subplots(nrows=2, ncols=3, figsize=(12, 8))

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





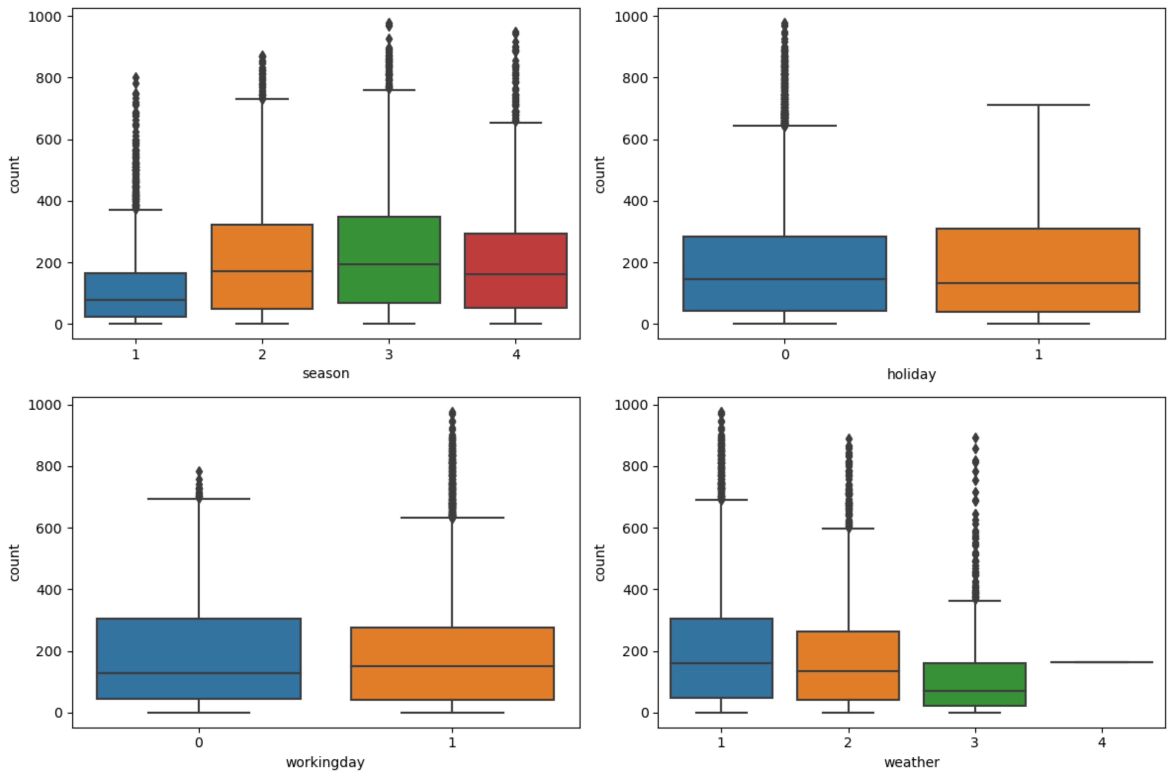
Variables like Windspeed, Casual, Registered and count have outliers.

Bivariate Analysis

In [24]:
```
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(12, 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.tight_layout()
plt.show()
```



Season and Count

1. Median - Highest in summer and fall , Lowest in winter, followed by spring
2. Spread - Largest in summer and fall, indicating more variability while Smallest in winter and spring
3. Outliers - All seasons have some very high outliers
4. Skew - All distributions appear right-skewed, with longer upper whiskers and outliers
5. Trend - Bike rentals tend to be higher in summer and fall

Holiday and Count

1. Median - The median for non-holidays is more than for holidays
2. Spread - The interquartile range (box size) is larger for non-holidays, indicating more variability in rental counts.

3. Outliers - Non-holidays have more high-value outliers
4. Skew - Both distributions are right-skewed
5. Trend - Non-holidays seem to have slightly higher rental counts overall

Weather and Count

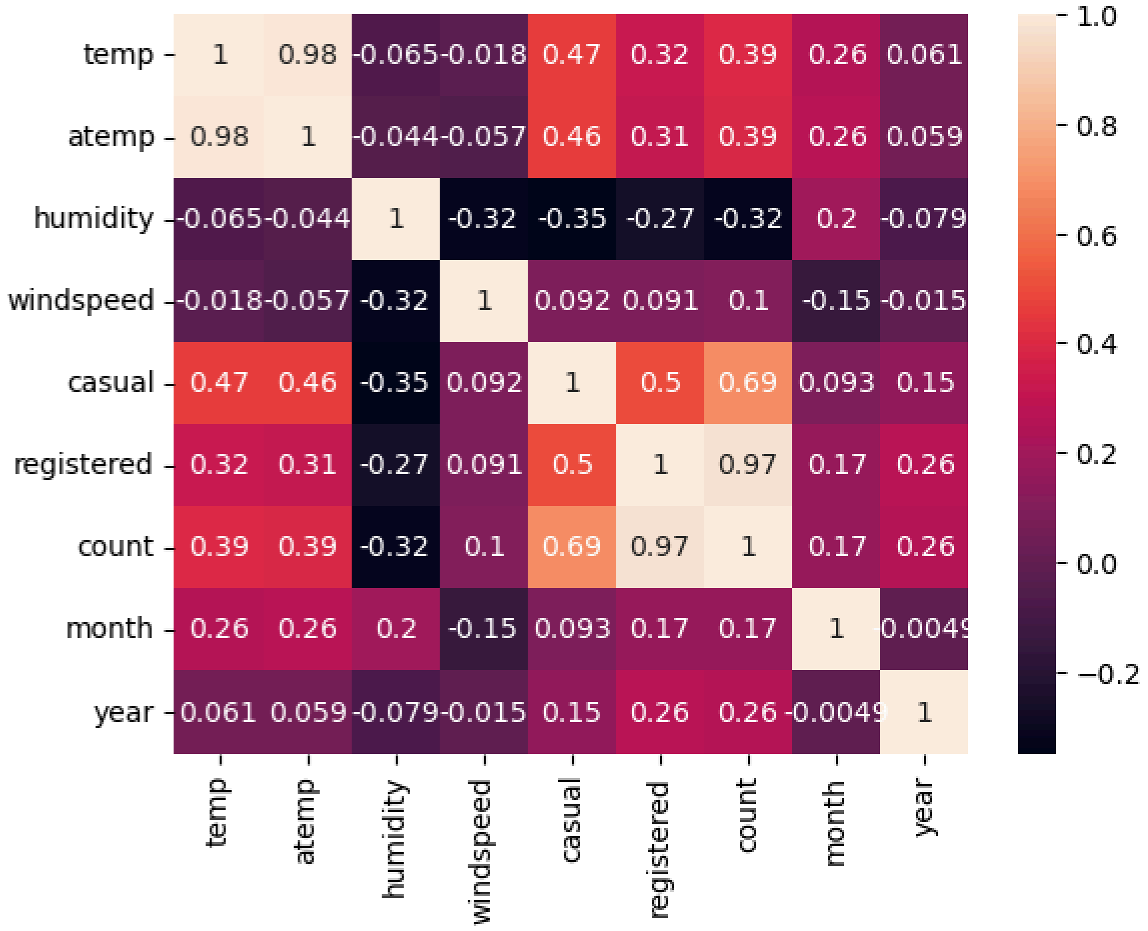
1. Clear weather significantly boosts sales, with high variability and frequent peak sales.
2. Mildly adverse weather slightly reduces sales but maintains a similar spread to clear weather, with fewer peaks.
3. Worse weather further lowers sales and reduces variability, though occasional high sales still occur.
4. Severe weather results in the lowest sales, with minimal variability and no high outliers, indicating that extreme conditions heavily impact consumer activity.

Working Day and Count

1. Working Days shows high rental counts.
2. More outliers in case of Working days.

```
In [25]: # CO-RELATION
         sns.heatmap(df.corr(), annot=True)
         plt.show()
```

```
C:\Users\mayank.khanduja\AppData\Local\Temp\ipykernel_10848\3434107565.py:2:
FutureWarning: The default value of numeric_only in DataFrame.corr is deprec
ated. In a future version, it will default to False. Select only valid colum
ns or specify the value of numeric_only to silence this warning.
  sns.heatmap(df.corr(), annot=True)
```



	temp	atemp	humidity	windspeed	casual	registered	count	month	year
temp	1	0.98	-0.065	-0.018	0.47	0.32	0.39	0.26	0.061
atemp	0.98	1	-0.044	-0.057	0.46	0.31	0.39	0.26	0.059
humidity	-0.065	-0.044	1	-0.32	-0.35	-0.27	-0.32	0.2	-0.079
windspeed	-0.018	-0.057	-0.32	1	0.092	0.091	0.1	-0.15	-0.015
casual	0.47	0.46	-0.35	0.092	1	0.5	0.69	0.093	0.15
registered	0.32	0.31	-0.27	0.091	0.5	1	0.97	0.17	0.26
count	0.39	0.39	-0.32	0.1	0.69	0.97	1	0.17	0.26
month	0.26	0.26	0.2	-0.15	0.093	0.17	0.17	1	0.0049
year	0.061	0.059	-0.079	-0.015	0.15	0.26	0.26	-0.0049	1

1. Temperature and atemp are very highly positvely correlated.
2. Temperature and humidity are highly negatively correlated.
3. Temperature is positively correlated with year, count and casual.
4. Atemp is highly correlated with temp
5. Atemp is negatively correlated with humidity and windspeed.
6. Atemp is positively correlated with year and casual.
7. Humidity is negatively correlated with almost everything.
8. Humidity is very slightly positively correlated with month.
9. Humidity is highly negatively correlated with year followed by humidity and atemp.
10. Windspeed is highly positively correlated with causal and registered.
11. Count is highly postively correlated with registered.

Hypothesis Testing

1. Working Day has effect on number of electric cycles rented
2. No. of cycles rented similar or different in different seasons
3. No. of cycles rented similar or different in different weather
4. Weather is dependent on season

Working Day has effect on number of electric cycles rented

In [26]:
```
#Ho = Working Day has no effect on number of electric cycles rented
#H1 = Working day affects number of electric cycles rented
# at alpha = 95%

df_wd0 = df[df['workingday']==0]
df_wd1 = df[df['workingday']==1]

alpha = 0.05

t_stat, p_value = ttest_ind(df_wd0['count'],df_wd1['count'], alternative = "t

print("t_stat : ",t_stat)
print("p_value : ",p_value)

alpha= 0.05
if p_value < alpha :
    print("We Reject the Null Hypothesis")
else :
    print("We Fail to Reject the Null Hypothesis")
    print('Interpretation: Working day has no effect on Electric Cycles Rente
```

```
t_stat :  -1.2096277376026694
p_value :  0.22644804226361348
We Fail to Reject the Null Hypothesis
Interpretation: Working day has no effect on Electric Cycles Rented
```

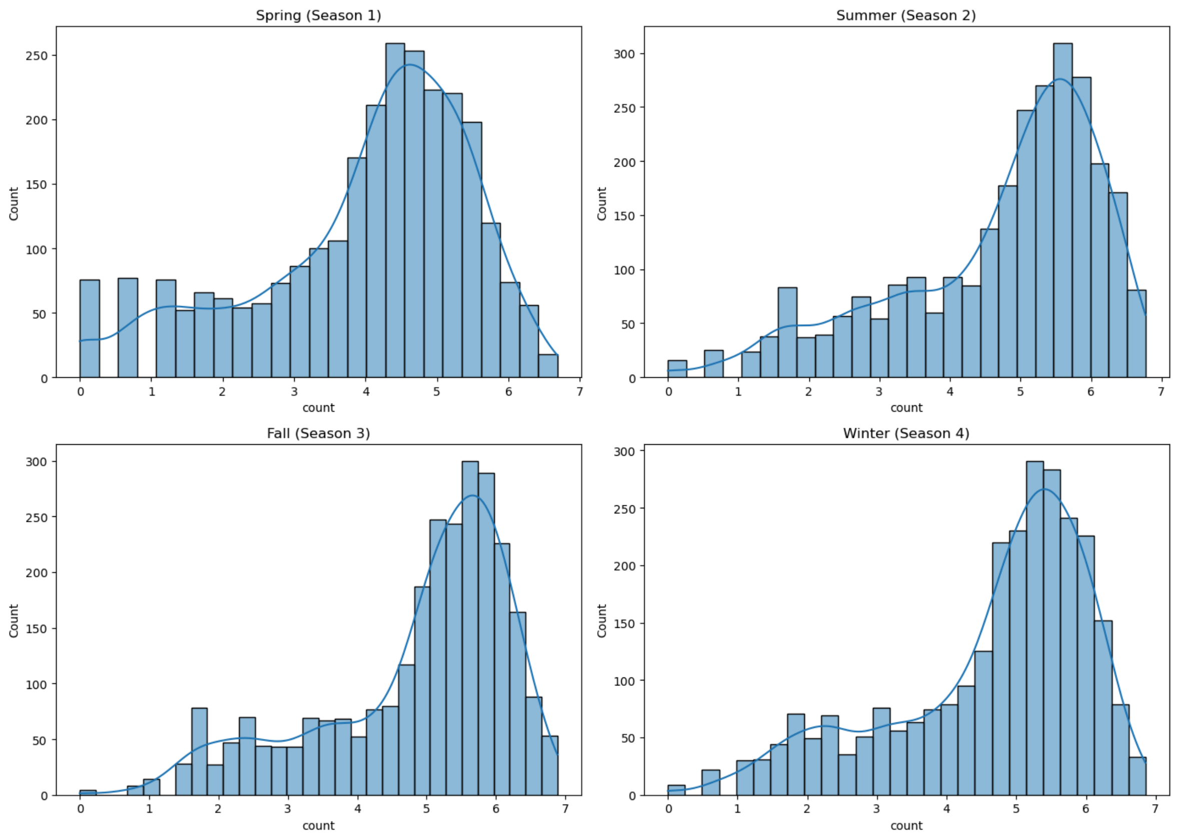
Number of cycles rented similar or different in different seasons

In [27]:
```
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(14, 10))

seasons = [1, 2, 3, 4]
season_labels = ['Spring', 'Summer', 'Fall', 'Winter']

for i, season in enumerate(seasons):
    row, col = divmod(i, 2)
    sns.histplot(data=np.log(df[df['season'] == season]['count']), kde=True,
    axes[row, col].set_title(f'{season_labels[i]} (Season {season})')

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



This test is again a numerical vs categorical test but this time there are 4 categories so we will perform annova

There are 3 conditions which should be satisfied to perform an annova test

1. Data should be gaussian which will be verified by a qqplot and a shapiro test.
2. Data should have equal variances among the categories
3. The rows in categories should not be overlapping in terms of data(which is satisfied)

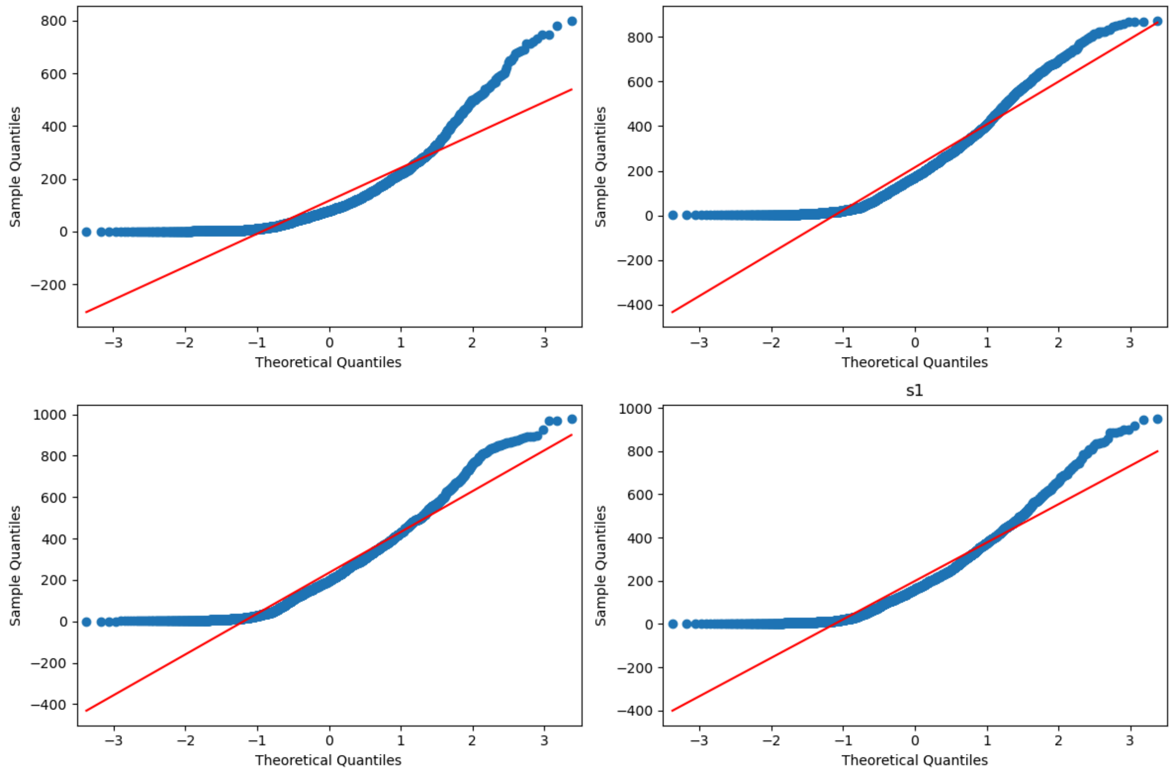
If these three conditions do not satisfy we go with the Kruskal-Wallis test.

```
In [28]: # CHECK NORMALITY OF DATA
df_s1 = df[df['season']==1]
df_s2 = df[df['season']==2]
df_s3 = df[df['season']==3]
df_s4 = df[df['season']==4]


fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(12, 8))

plt.title('s1')
qqplot(df_s1['count'], line='s', ax=axis[0, 0])
qqplot(df_s2['count'], line='s', ax=axis[0, 1])
qqplot(df_s3['count'], line='s', ax=axis[1, 0])
qqplot(df_s4['count'], line='s', ax=axis[1, 1])

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



DATA IS NOT NORMALLY DISTRIBUTED AS PER QQ PLOT

DATA IS NOT NORMALLY DISTRIBUTED AS PER SHAPIRO TEST

```
In [29]: # Shapiro Test
'''
H0 = Data is normally distributed.
H1 = Data is not normally distributed.
alpha = 0.05
'''

def seasons(x):
    s_stat, p_value = shapiro(x.sample(100))

    print('P value:', p_value, end="")

    if p_value < 0.05:
        print(', which is significantly lower than our alpha and hence we REJ
        print('Interpretation: This means that the data is not normally distr
    else:
        print(', which is higher than our alpha and hence we FAIL TO REJECT tl
        print('Interpretation: This means that the data is normally distribut

    return
```

```
In [31]: print('Season 1 count of cycles rented graph')
seasons(df_s1['count'])
```

```
Season 1 count of cycles rented graph
P value: 3.2253269065946055e-12, which is significantly lower than our alpha
and hence we REJECT the null hypothesis.
Interpretation: This means that the data is not normally distributed.
```

```
In [32]: print('Season 2 count of cycles rented graph')
seasons(df_s2['count'])
```

```
Season 2 count of cycles rented graph
P value: 7.671339972148417e-07, which is significantly lower than our alpha
and hence we REJECT the null hypothesis.
Interpretation: This means that the data is not normally distributed.
```

```
In [33]: print('Season 3 count of cycles rented graph')
seasons(df_s3['count'])
```

```
Season 3 count of cycles rented graph
P value: 7.629064930370077e-05, which is significantly lower than our alpha
and hence we REJECT the null hypothesis.
Interpretation: This means that the data is not normally distributed.
```

```
In [34]: print('Season 4 count of cycles rented graph')
seasons(df_s4['count'])
```

```
Season 4 count of cycles rented graph
P value: 2.585158824786049e-07, which is significantly lower than our alpha
and hence we REJECT the null hypothesis.
Interpretation: This means that the data is not normally distributed.
```

In [35]:
```python
# CHECKING VARIANCE
print("Variance of Season 1: ", df_s1['count'].var())
print("Variance of Season 2: ", df_s2['count'].var())
print("Variance of Season 3: ", df_s3['count'].var())
print("Variance of Season 4: ", df_s4['count'].var())
```

```
Variance of Season 1:  15693.56853371715
Variance of Season 2:  36867.01182553239
Variance of Season 3:  38868.5170126629
Variance of Season 4:  31549.720316669263
```

In [36]:
```python
alpha = 0.05

l_stat,p_value = levene(df_s1['count'],df_s2['count'],df_s3['count'],df_s4['c

print("p_value : ",p_value)

if p_value< alpha:
    print("As the p_value is lower than alpha we reject the null hypothesis")
    print("Conclusion : Variances are NOT Equal")
else:
    print("Interpretation : Fail to Reject Ho")
    print("Conclusion : Variances are Equal")
```

```
p_value :  1.0147116860043298e-118
As the p_value is lower than alpha we reject the null hypothesis
Conclusion : Variances are NOT Equal
```

We can clearly see that no condition is satisfied for annova and hence we cannot perform it.

In this case we go with the Kruskal Wallis test.

In [37]:
```python
alpha = 0.05

k_stat, p_value = kruskal(df_s1['count'], df_s2['count'], df_s3['count'], df_

print("p_value : ",p_value)

if p_value < alpha :
    print("As the p_value is lower than alpha we reject the null hypothesis, 
else :
    print("We Fail to Reject the Null Hypothesis")
```

```
p_value :  2.479008372608633e-151
As the p_value is lower than alpha we reject the null hypothesis, which mean
s that season has an effect on electric cycles rented.
```

Number of cycles rented similar or different in different weather

This test is again a numerical vs categorical test but this time there are 4 categories so we will perform annova There are 3 conditions which should be satisfied to perform an annova test

1. Data should be gaussian which will be verified by a qqplot and a shapiro test.
2. Data should have equal variances among the categories
3. The rows in categories should not be overlapping in terms of data(which is satisfied)

If these three conditions do not satisfy we go with the Kruskal-Wallis test.

In [41]:
```
# shapiro test
'''
H0 = Data is normally distributed.
H1 = Data is not normally distributed.
alpha = 0.05
'''

def weather(x):
    s_stat, p_value = shapiro(x.sample(100))

    print('P value:', p_value, end="")

    if p_value < 0.05:
        print(', which is significantly lower than our alpha and hence we REJ
        print('Interpretation: This means that the data is not normally distr
    else:
        print(', which is higher than our alpha and hence we we FAILED TO REJ
        print('Interpretation: This means that the data is normally distribut

    return
```

In [42]:
```
print('Weather 1 count of cycles rented graph')
seasons(df_w1['count'])
```

```
Weather 1 count of cycles rented graph
P value: 2.9676472834694323e-08, which is significantly lower than our alpha
and hence we REJECT the null hypothesis.
Interpretation: This means that the data is not normally distributed.
```

In [43]:
```
print('Weather 2 count of cycles rented graph')
seasons(df_w2['count'])
```

```
Weather 2 count of cycles rented graph
P value: 5.585108375782966e-09, which is significantly lower than our alpha
and hence we REJECT the null hypothesis.
Interpretation: This means that the data is not normally distributed.
```

In [44]:
```
print('Weather 3 count of cycles rented graph')
seasons(df_w3['count'])
```

```
Weather 3 count of cycles rented graph
P value: 2.5925035451734857e-09, which is significantly lower than our alpha
and hence we REJECT the null hypothesis.
Interpretation: This means that the data is not normally distributed.
```

DATA IS NOT NORMALLY DISTRIBUTED

In [45]:
```
# levene Test
print("Variance of Weather 1: ", df_w1['count'].var())
print("Variance of Weather 2: ", df_w2['count'].var())
print("Variance of Weather 3: ", df_w3['count'].var())
print("Variance of Weather 4: ", df_w4['count'].var())
```

```
Variance of Weather 1:  35328.79846268019
Variance of Weather 2:  28347.248993301808
Variance of Weather 3:  19204.775892714213
Variance of Weather 4:  nan
```

In [46]:
```
alpha = 0.05

l_stat,p_value = levene(df_w1['count'],df_w2['count'],df_w3['count'])

print("p_value : ",p_value)

if p_value< alpha:
    print("As the p_value is lower than alpha we reject the null hypothesis")
    print("Conclusion : Variances are NOT Equal")
else:
    print("Interpretation : Fail to Reject Ho")
    print("Conclusion : Variances are Equal")
```

```
p_value :  6.198278710731511e-36
As the p_value is lower than alpha we reject the null hypothesis
Conclusion : Variances are NOT Equal
```

In [47]:
```
'''
We can clearly see that no condition is satisfied for annova and hence we cann
In this case we go with the Kruskal Wallis test.

Let us setup the null and alternate hypothesis for Kruskal-Wallis test.
H0 = Weather has no effect on number of electric cycles rented
H1 = Weather affects the number of electric cycles rented
alpha = 0.05
'''

alpha = 0.05

k_stat, p_value = kruskal(df_w1['count'], df_w2['count'], df_w3['count'])

print("p_value : ",p_value)

if p_value < alpha :
    print("As the p_value is lower than alpha we reject the null hypothesis, w
else :
    print("We Fail to Reject the Null Hypothesis")
```

```
p_value :  3.122066178659941e-45
As the p_value is lower than alpha we reject the null hypothesis, which mean
s that weather has an effect on electric cycles rented.
```

Weather is dependent on season (check between 2 predictor variable)

In [48]:
```
df1 = pd.crosstab(index = df['weather'],columns = df['season'])
df1
```

Out[48]:

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

In [49]:
```
df1.drop([4],inplace = True)
```

In [50]:
```
alpha= 0.05

chi_stat, p_value, dof, expected = chi2_contingency(df1)

print("chi_stat : ",chi_stat)
print("p_value : ",p_value)

if p_value < alpha :
    print("As p_value is lower than alpha we Reject the Null Hypothesis which
else :
    print("We Fail to Reject the Null Hypothesis")
```

```
chi_stat :  46.101457310732485
p_value :  2.8260014509929403e-08
As p_value is lower than alpha we Reject the Null Hypothesis which means tha
t Weather has an impact on seasons.
```

SUMMARY

1. Working Day has effect on number of electric cycles rented?

- We performed ttest as it was Numerical vs Categorical.
- We Fail to Reject the Null Hypothesis and working day has no effect on Electric Cycles Rented.

2. No. of cycles rented similar or different in different seasons?

- It was numerical vs 4 categories hence we decided to do annova.
- We checked all the conditions for Annova and saw that the conditions dont satisfy. Hence, we proceeded with the Kruskal Wallis test.
- As the p_value is lower than alpha we reject the null hypothesis.
- Season affects the number of electric cycles rented.

3. No. of cycles rented similar or different in different weather?

- It was numerical vs 4 categories hence we decided to do annova.
- We observed that the 4th weather was not of any significant use as it only had 1 row hence we did not consider it for further tests.
- We checked all the conditions for Annova and saw that the conditions dont satisfy. Hence, we proceeded with the Kruskal Wallis test.
- As the p_value is lower than alpha we reject the null hypothesis.
- Weather affects the number of electric cycles rented.

4. Weather is dependent on season?

- We went ahead with the chi squared test as it was categorical vs categorical.
- As p_value is lower than alpha we Reject the Null Hypothesis.
- Weather has an impact on seasons.