Problem Statement

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.

In [191…

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import ttest_ind, shapiro, levene, kruskal, chi2_contingenc
from statsmodels.graphics.gofplots import qqplot
```

In [232…

```
# Loading dataset
df = pd.read_csv("/Users/bose/Downloads/yulu.csv")
```

In [3]:

```
df.head()
```

Out[3]:

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

In [6]:

```
df.tail()
```

Out[6]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed
10881	2012-12-19 19:00:00	4	0	1	1	15.58	19.695	50	26.0027
10882	2012-12-19 20:00:00	4	0	1	1	14.76	17.425	57	15.0013
10883	2012-12-19 21:00:00	4	0	1	1	13.94	15.910	61	15.0013
10884	2012-12-19 22:00:00	4	0	1	1	13.94	17.425	61	6.0032
10885	2012-12-19 23:00:00	4	0	1	1	13.12	16.665	66	8.998

In [4]:
```
df.shape
```

Out[4]:
```
(10886, 12)
```

The dataset has **10886 rows** and **12 columns**.

In [8]:
```
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
```

Is there any Null values in the datasaet ?

In [233…
```
df.isna().sum()
```

Out[233]:
```
datetime      0
season        0
holiday       0
workingday    0
weather       0
temp          0
atemp         0
humidity      0
windspeed     0
casual        0
registered    0
count         0
dtype: int64
```

No Null Values in the dataset

Is there any Duplicate values in the dataset ?

In [235…
```
df.duplicated().sum()
```

Out[235]:
```
0
```

No Duplicate Values in the dataset

In [5]:
```
# Datatypes of Columns -
df.dtypes
```

Out[5]:
```
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 [16]:
```
# Converting datatype of 'datetime' column from object to datetime
df['datetime'] = pd.to_datetime(df['datetime'])
```

In [17]:
```
# Converting categorical attributes to category
df['season'] = df['season'].astype('object')
df['holiday'] = df['holiday'].astype('object')
df['workingday'] = df['workingday'].astype('object')
df['weather'] = df['weather'].astype('object')
```

In [18]:
```
# Minimum Date
df['datetime'].min()
```

Out[18]:
```
Timestamp('2011-01-01 00:00:00')
```

In [19]:
```
# Maximum Date
df['datetime'].max()
```

Out[19]:
```
Timestamp('2012-12-19 23:00:00')
```

In [20]:
```
# Time Period for which data has been given -
df['datetime'].max() - df['datetime'].min()
```

Out[20]:
```
Timedelta('718 days 23:00:00')
```

In [22]:
```
df.describe()
```

Out[22]:

	temp	atemp	humidity	windspeed	casual	registere
count	10886.00000	10886.000000	10886.000000	10886.000000	10886.000000	10886.0000
mean	20.23086	23.655084	61.886460	12.799395	36.021955	155.5521
std	7.79159	8.474601	19.245033	8.164537	49.960477	151.0390
min	0.82000	0.760000	0.000000	0.000000	0.000000	0.0000
25%	13.94000	16.665000	47.000000	7.001500	4.000000	36.0000
50%	20.50000	24.240000	62.000000	12.998000	17.000000	118.0000
75%	26.24000	31.060000	77.000000	16.997900	49.000000	222.0000
max	41.00000	45.455000	100.000000	56.996900	367.000000	886.0000

In [ ]:
```
df.describe(include='object')
```

Out[ ]:

	season	holiday	workingday	weather
count	10886	10886	10886	10886
unique	4	2	2	4
top	4	0	1	1
freq	2734	10575	7412	7192

Non-Visual Analysis -

Season

In [221…
```
# season wise unique value & count -
df["season"].value_counts()
```

Out[221]:
```
4    2734
2    2733
3    2733
1    2686
Name: season, dtype: int64
```

In [222…
```
df["season"].value_counts(normalize = True).round(4) * 100
```

Out[222]:
```
4    25.11
2    25.11
3    25.11
1    24.67
Name: season, dtype: float64
```

Observations -

- We have **4 seasons** in the given dataset
- **All 4 seasons** are almost **equally distributed**

Holiday

In [223…
```
# holiday wise unique value & count -
df["holiday"].value_counts()
```

Out[223]:
```
0    10575
1      311
Name: holiday, dtype: int64
```

In [224…
```
df["holiday"].value_counts(normalize = True).round(2) * 100
```

Out[224]:
```
0    97.0
1     3.0
Name: holiday, dtype: float64
```

Observation -

- In the given data, **97% accounts for non-holidays(10,575 days)**
- **Only 3% are holidays(311 days)**

Workingday

In [225…
```
# working day wise unique value & count -
df["workingday"].value_counts()
```

Out[225]:
```
1    7412
0    3474
Name: workingday, dtype: int64
```

In [226…
```
df["workingday"].value_counts(normalize = True).round(2) * 100
```

Out[226]:
```
1    68.0
0    32.0
Name: workingday, dtype: float64
```

Observation -

- In the given data, **68% accounts for working day(7412 days)**
- And **32% accounts for non-working days(3474 days)**

Weather

In [227…
```
# weather wise unique value & count -
df["weather"].value_counts()
```

Out[227]:
```
1    7192
2    2834
3     859
4       1
Name: weather, dtype: int64
```

In [228…
```
df["weather"].value_counts(normalize = True).round(4) * 100
```

Out[228]:
```
1    66.07
2    26.03
3     7.89
4     0.01
Name: weather, dtype: float64
```

Observation -

- There are **4 different types weathers** in the given data.

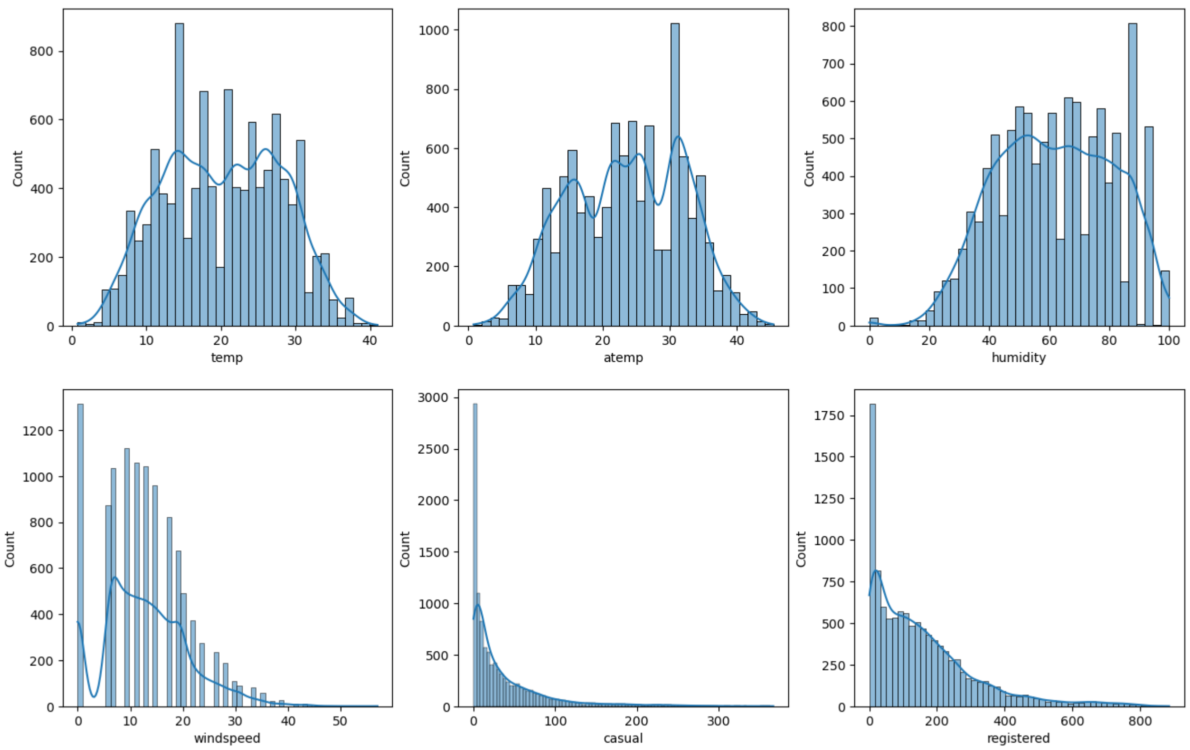
- **Most preferred weather** is **weather 1(Clear)**
- **Least preferred weather** is **weather 4(Heavy Rain)**

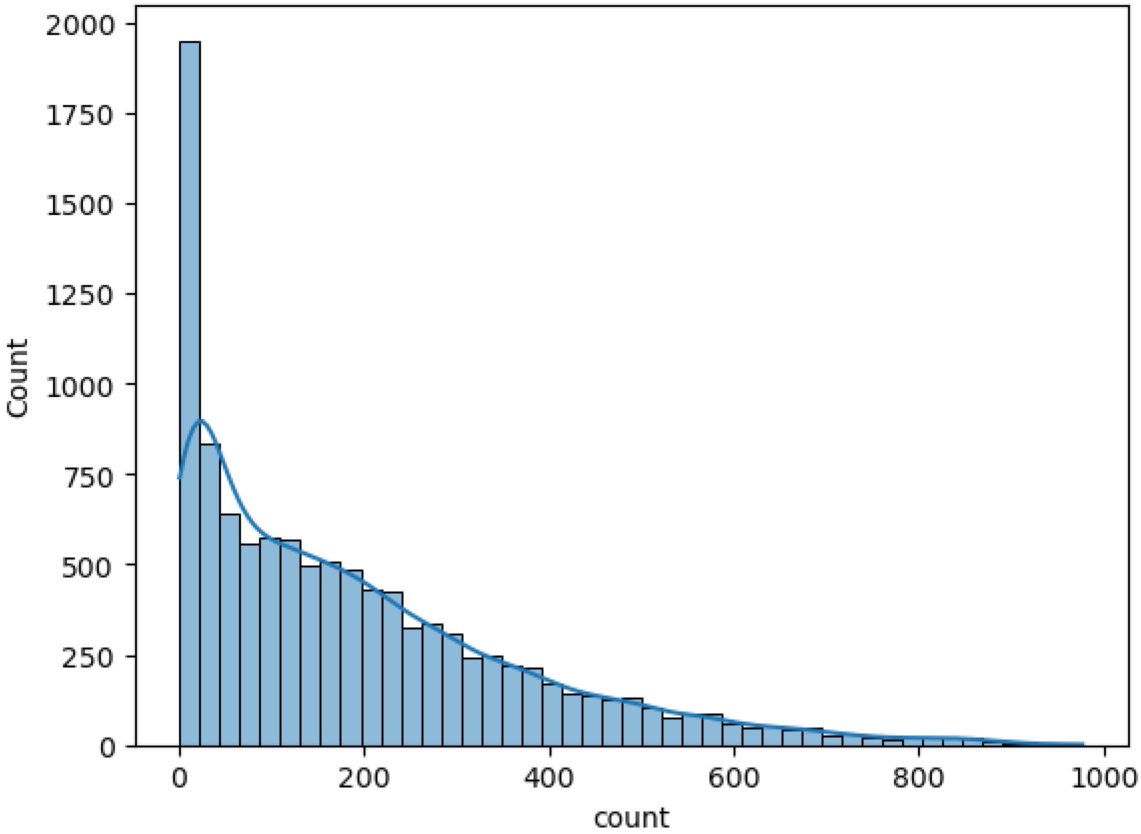
In [25]:
```
#Splitting the columns into numerical and categorical

num_cols = ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered'
cat_cols = ['season', 'holiday', 'workingday', 'weather']
```

Visual Analysis

Univariate Analysis -

In [38]:
```
# Plotting Distplots for all the numerical columns
num_cols = ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered'

fig, axis = plt.subplots(nrows=2, ncols=3, figsize=(16, 10))

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

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

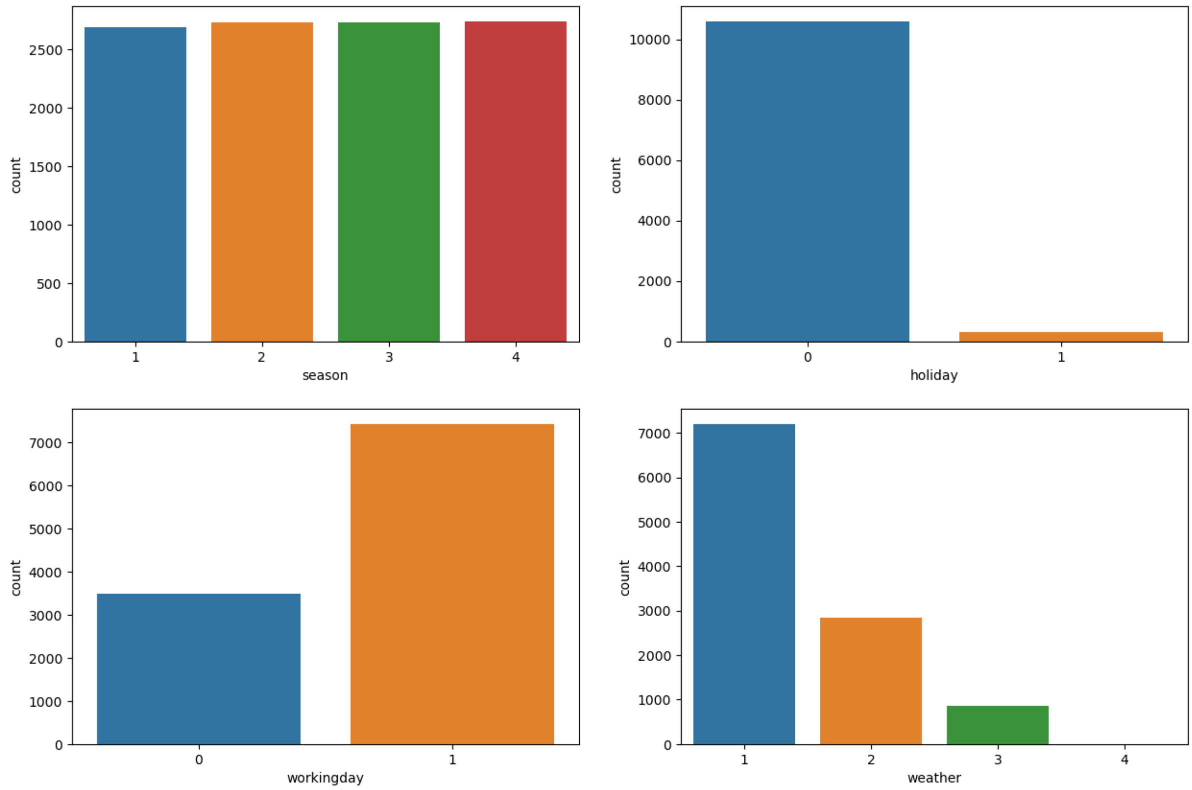




Observations -

- Variables **temp, atemp,humidity** seem to be following a **Normal** distribution
- **Range of temp is (0-40)**. **Median value is 20.5**.
- Range of **atemp** is **(0-45)**. **Median** value is **24.24**.
- **humidity** has a range of **(0-100)**. **Median** value for humidity is **62**.
- **windspeed** has a range of **(0-57)**.**Median** value for windspeed is **12.998**.
- **casual** has a range of **(0-367)**. **Median** value is **17**.
- **registered** has a range of **(0-886)**. **Median** value for registerd column is **118**.
- **count** has a range of **(0-977)**. **Median** value for count column is **147**.

In [39]:
```
# Plotting Countplot for all the categorical columns
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(15, 10))

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

plt.show()
```

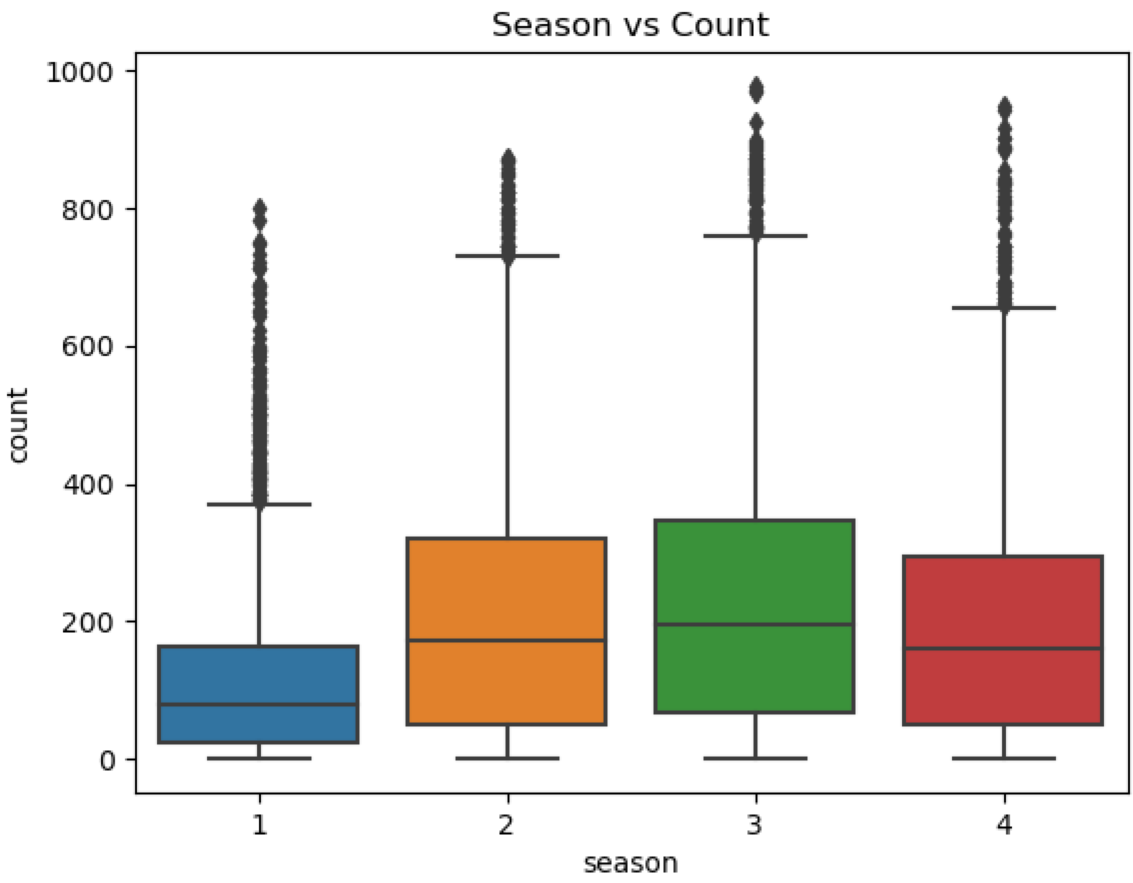


Observations -

- All **4 seasons** seem to be **equally distributed**.
- No of **holidays is 311** compared to **non-holidays 10,575**.
- No of **working days is 7412**. No of **Non-working days is 3474**.
- **Weather 1** (Clear) is the **most common weather appearing on 7192 days**. Whereas **weather 4** (heavy rain) is the **least common weather** appearing on just 1 day.

Bivariate Analysis -

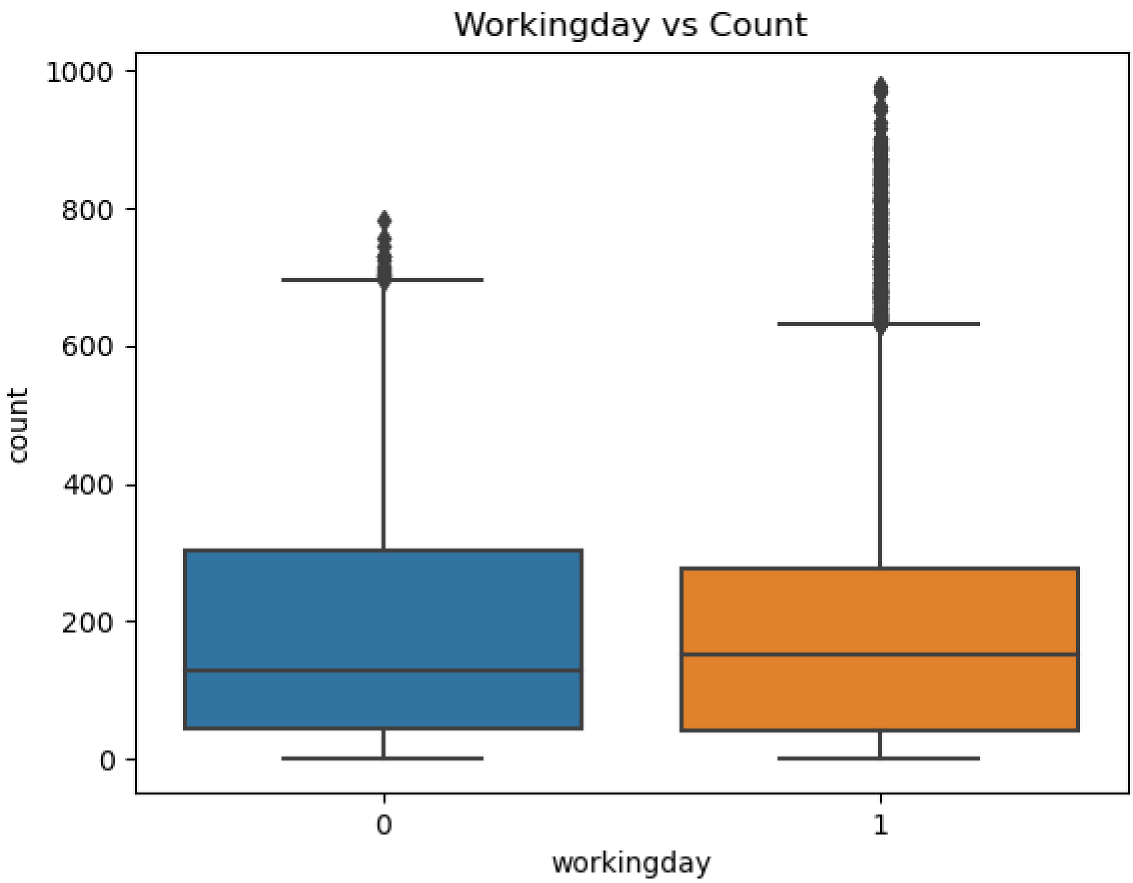
In [46]:
```
# Plotting relationship between season and count
sns.boxplot(data=df, x='season', y='count')
plt.title('Season vs Count')
plt.show()
```



Observations -

- **Most bikes** are rented during **season 3(Fall)**
- **Least bike** are rented during **season 1(Spring)**

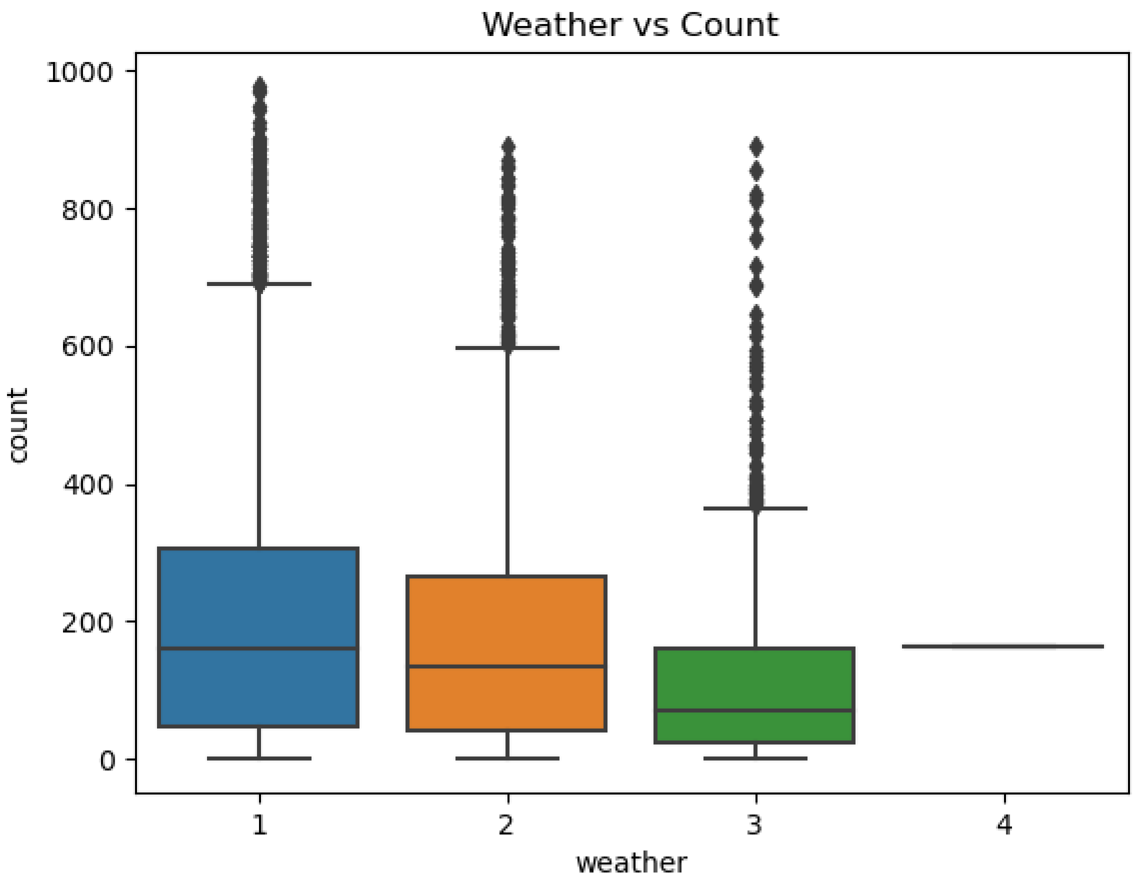
In [47]:
```
# Plotting relationship between workingday and count
sns.boxplot(data=df, x='workingday', y='count')
plt.title('Workingday vs Count')
plt.show()
```



Observation -

- **Median value for count** of bike rentals is slightly **higher for working day**
- maximum value for number of rentals is higher for non-workingday

In [48]:
```
# Plotting relationship between weather and count
sns.boxplot(data=df, x='weather', y='count')
plt.title('Weather vs Count')
plt.show()
```



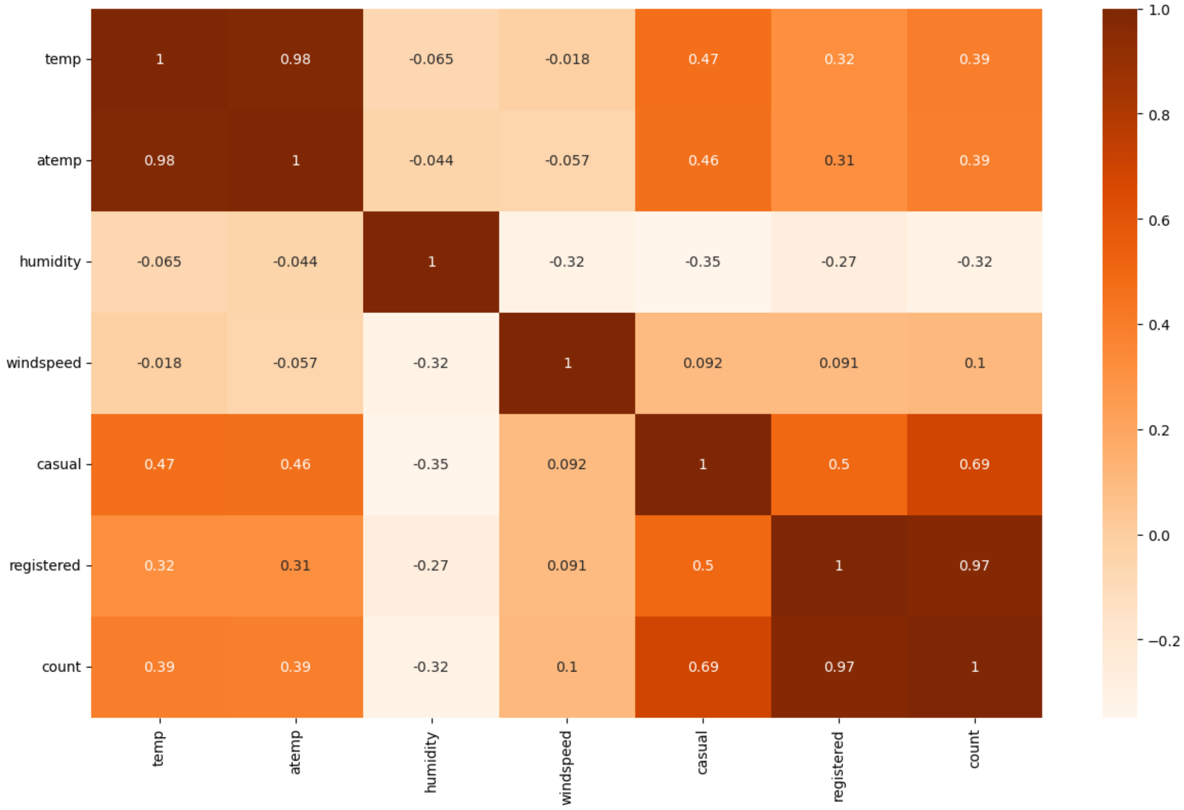
Observations -

- **Weather 1** (Clear) is the **most preferred weather** for renting bikes.
- **Weather 4** (Heavy Rain) is the **least preferred weather**.

Multivariate Analysis -

In [231…

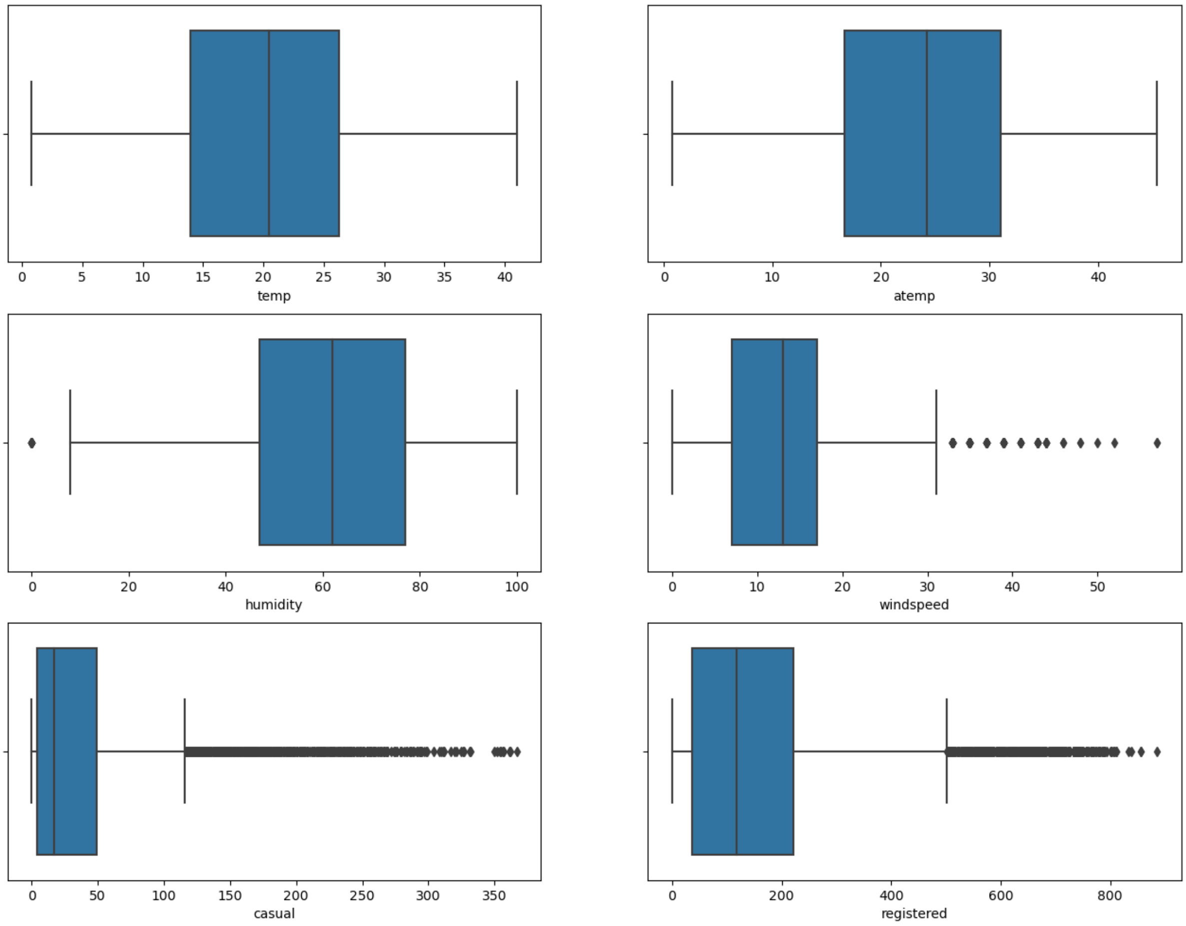
```
# Heatmap -
plt.figure(figsize = (15, 9))
sns.heatmap(df.corr(), annot = True, cmap = "Oranges")
plt.xticks(rotation = 90)
plt.yticks(rotation = 0)
plt.show()
```

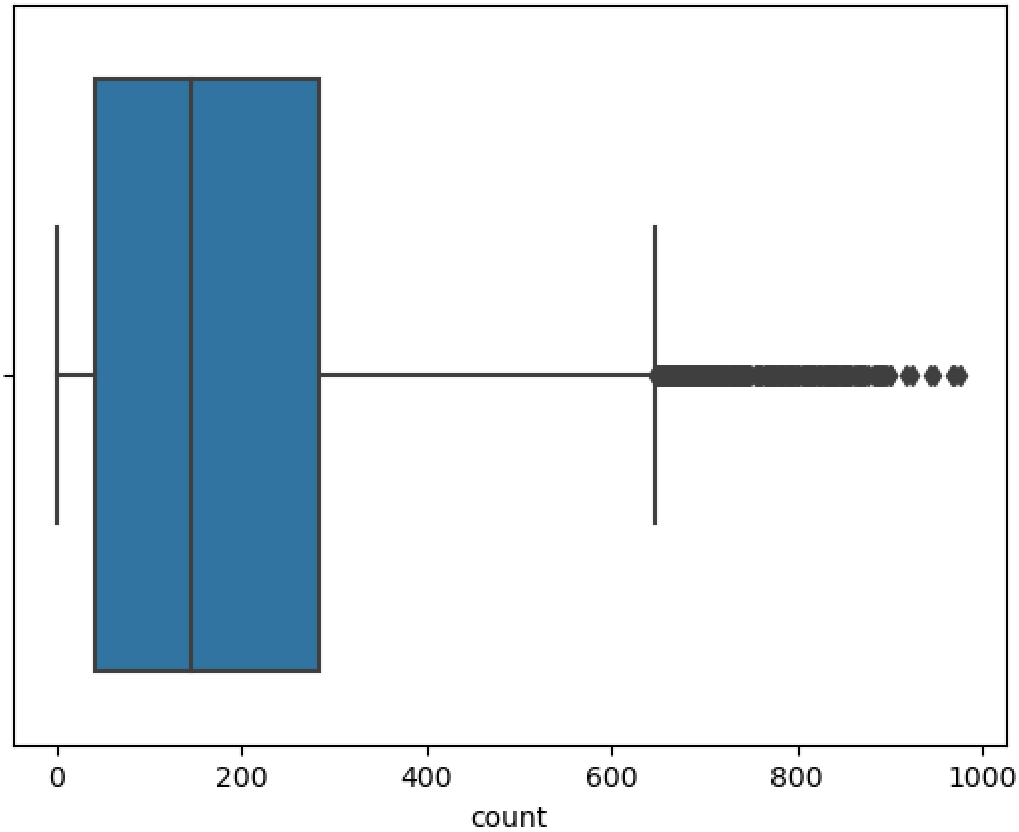


Observations -

- **temp has a negative correlation with humidity and windspeed**
- **humidity has a negative correlation with every other attribute**
- **windspeed doesn't have any strong correleation** with the other attributes
- **casual has a high corrrelation(0.69) with count**
- **registered has very high correlation with count(0.97)**

In [50]:
```
# Plotting boxplot for each numerical column to find outliers
fig, axis = plt.subplots(nrows=3, ncols=2, figsize=(16, 12))

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

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





Observations -

- There are **no outliers for temp and atemp** columns
- **Rest of the columns (humidity, windspeed, casual, registered and count) have outliers**
- humidity has an outlier near 0

- windspeed have a few outliers above the value of 30
- casual have a lot of outliers above the value 120
- registered have many outliers above the value 500
- count have many outliers above the value 640

Hypothesis Testing

1. 2-Sample T-Test to check if Working Day has an effect on the number of electric cycles rented

- H0 : Working day **has no effect** on the number of electric cycles rented
- Ha : Working day **have an effect** on the number of electric cycles rented

Let us take the **Significance Level as 95%**.

- **alpha = 0.05**

In [51]:
```
df.head()
```

Out[51]:

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

In [70]:
```
working = df[df['workingday']==1]
working.head()
```

Out[70]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	c
47	2011-01-03 00:00:00	1	0	1	1	9.02	9.850	44	23.9994	
48	2011-01-03 01:00:00	1	0	1	1	8.20	8.335	44	27.9993	
49	2011-01-03 04:00:00	1	0	1	1	6.56	6.820	47	26.0027	
50	2011-01-03 05:00:00	1	0	1	1	6.56	6.820	47	19.0012	
51	2011-01-03 06:00:00	1	0	1	1	5.74	5.305	50	26.0027	

In [85]:
```
working_day = working['count']
working_day.head()
```

Out[85]:
```
47     5
48     2
49     1
50     3
51    30
Name: count, dtype: int64
```

In [86]:
```
non_working = df[df['workingday']==0]
non_working.head()
```

Out[86]:

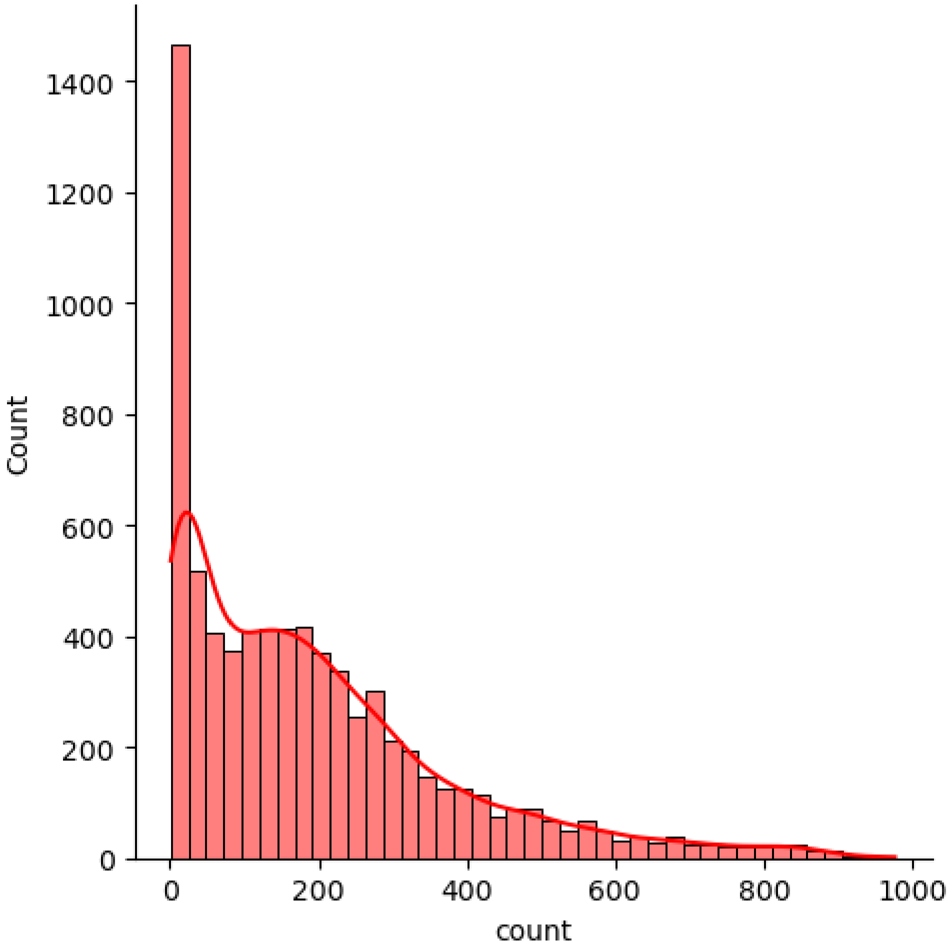
	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	ca
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	
4	2011-01-01 04:00:00	1	0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

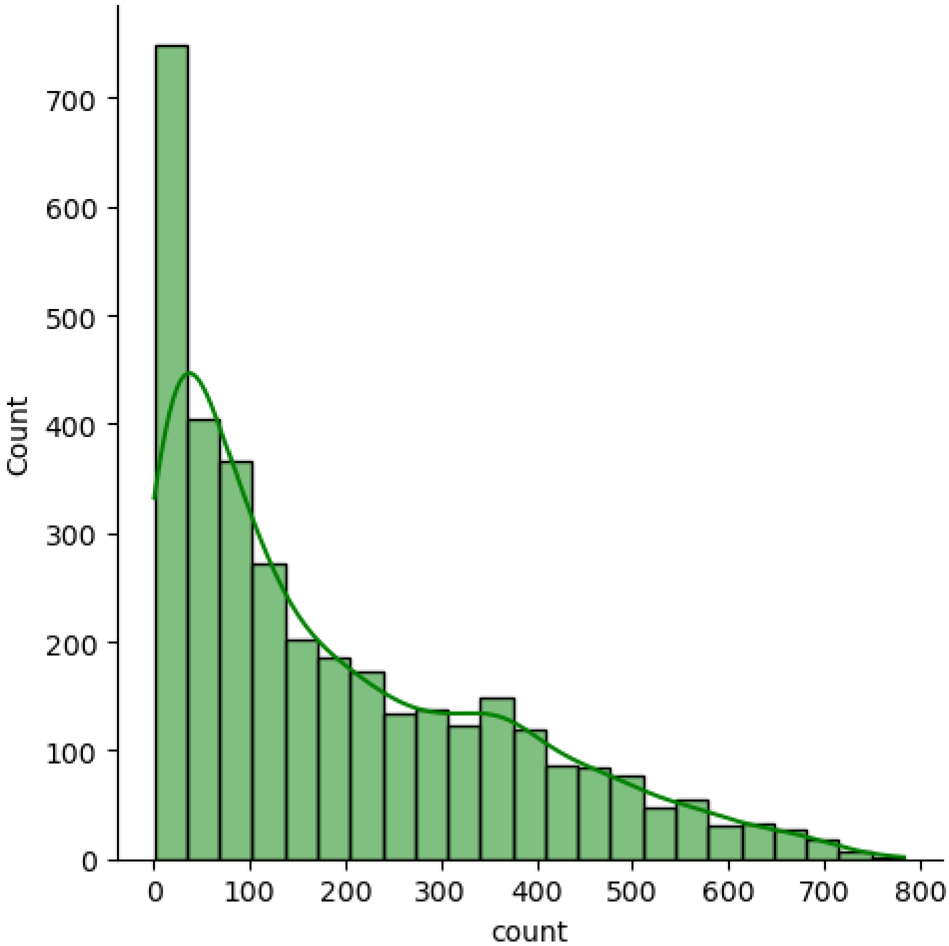
In [87]:
```
non_working_day = non_working['count']
non_working_day.head()
```

Out[87]:
```
0     16
1     40
2     32
3     13
4      1
Name: count, dtype: int64
```

Assumptions -

In [115…
```
sns.displot(working, x = 'count',kde=True, color='r')
sns.displot(non_working, x = 'count', kde=True, color='g')
plt.show()
```





In [106…
```
np.var(working_day.values).round(2)
```

Out[106]:
```
34040.7
```

In [107…
```
np.var(non_working_day.values).round(2)
```

Out[107]:
```
30171.35
```

It can be see that the data values are continuous for both data sets and the variances are almost equal

In [69]:
```
t_stat, p_value = ttest_ind(working_day, non_working_day)
t_stat, p_value
```

Out[69]:
```
(1.2096277376026694, 0.22644804226361348)
```

Inference -

- Here, **p_value > alpha**
- Since p_value is greater than alpha, **we cannot reject the Null Hypothesis**.
- Therefore **working day has no effect on the numer of electric cycles being rented**.

2. ANNOVA to check if No. of cycles rented is similar or different in different

- 1. weather
- 1. season

Assumptions for ANNOVA test -

- Data should be gaussian
- Data should be independant
- Equal variance between the data

A) Weather

In [109…

```
df.head()
```

Out[109]:

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

In [111…

```
df['weather'].value_counts()
```

Out[111]:

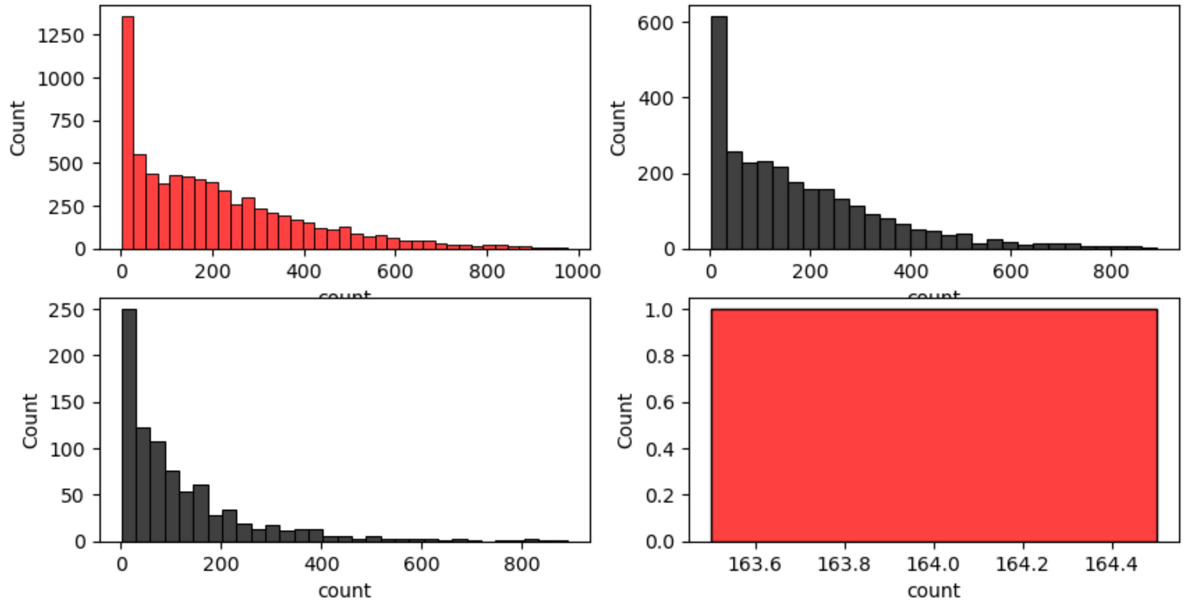
```
1    7192
2    2834
3     859
4       1
Name: weather, dtype: int64
```

In [167…

```
weather1 = df[df['weather']==1]
weather2 = df[df['weather']==2]
weather3 = df[df['weather']==3]
weather4 = df[df['weather']==4]
```

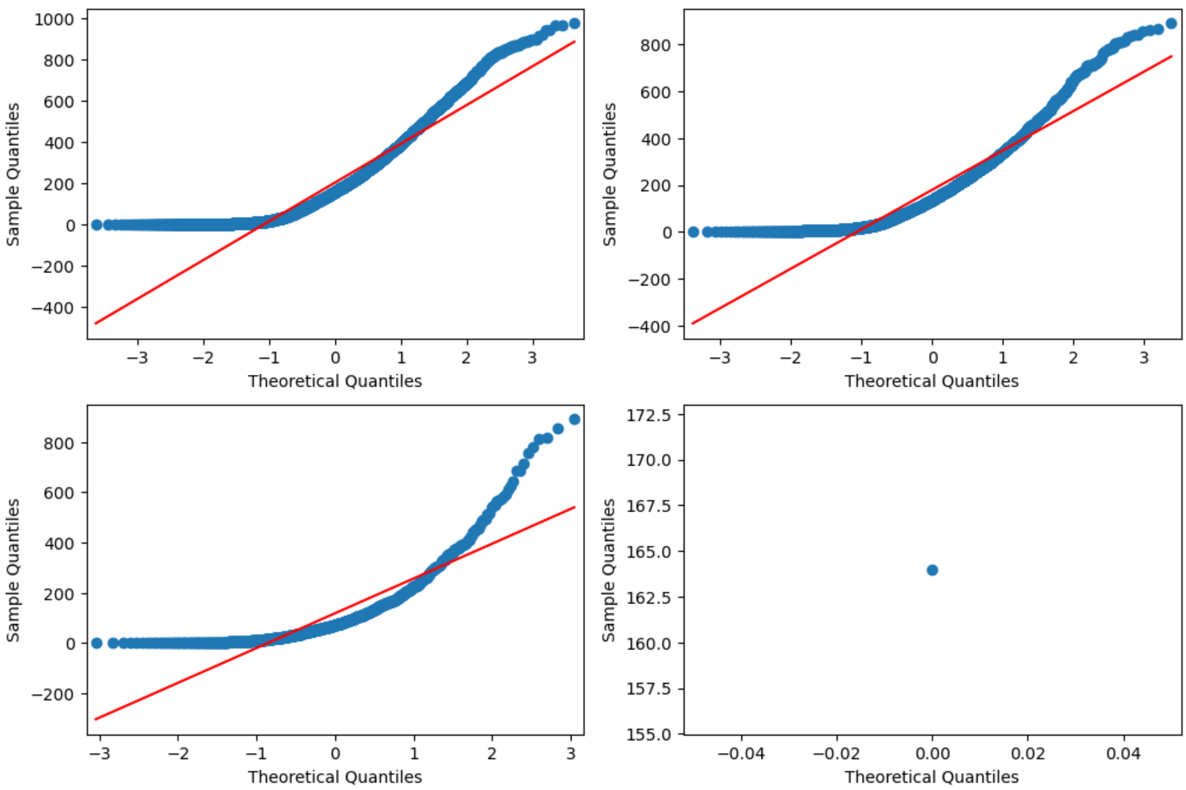
In [172…

```
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(10,5))

sns.histplot(weather1,x='count', ax=axis[0,0],color='r')
sns.histplot(weather2,x='count', ax=axis[0,1],color='k')
sns.histplot(weather3,x='count', ax=axis[1,0],color='k')
sns.histplot(weather4,x='count', ax=axis[1,1],color='r')
plt.show()
```



In [220…

```
# Plotting QQPlot to check whether given data is Gaussian
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(12,8))

qqplot(weather1,line='s', ax=axis[0,0])
qqplot(weather2,line='s', ax=axis[0,1],color='r')
qqplot(weather3,line='s', ax=axis[1,0],color='r')
qqplot(weather4,line='s', ax=axis[1,1])
plt.show()
```



In [206…

```
# Shapiro test - (To check whether data is Gaussian or not)
'''
Assumptions -
    H0 : Data follows gaussian distribution
    Ha : Data do not follow gaussian distribution
'''

total_count = df['count'].sample(100)
```

```python
test_stat, p_value = shapiro(total_count)
print('p_value :',p_value)
```

```
p_value : 2.874224662718916e-07
```

As **p_value < 0.5, we reject the null hypothesis**. And Therefor the **data does not follow Gaussian distribution**.

In [205…

```python
# Levene test - (To check variance of groups)
'''
Assumptions -
    H0 : Variances are equal
    Ha : Variances are not equal
'''

weather1 = df[df['weather']==1]["count"]
weather2 = df[df['weather']==2]["count"]
weather3 = df[df['weather']==3]["count"]
weather4 = df[df['weather']==4]["count"]

levene_stat, p_value = levene(weather1, weather2, weather3, weather4)
print('p_value :',p_value)
```

```
p_value : 3.504937946833238e-35
```

As **p-value < 0.5, we reject the null hypothesis**. And therefore **the Variance is not equal** between the data.

Observation -

- From the QQPlot, Shapiro test and Levene Test we can see that the **data does not satisfy the Assumptions for Annova Test**.
- Hence **we have to use Kruskal-Wallis method** here.

Kruskal Wallis method -

- *H0: No of cycles rented is similar in diiferent weather*
- *Ha: No of cycles rented is different in different weather*

In [186…

```python
weather1 = df[df['weather']==1]["count"]
weather2 = df[df['weather']==2]["count"]
weather3 = df[df['weather']==3]["count"]
weather4 = df[df['weather']==4]["count"]
```

In [157…

```python
weather1.mean()
```

Out[157]:

```
205.23679087875416
```

In [158…

```python
weather2.mean()
```

Out[158]:

```
178.95553987297106
```

In [159…

```python
weather3.mean()
```

Out[159]:

```
118.84633294528521
```

In [160…

```python
weather4.mean()
```

Out[160]: 164.0

In [201…
```
# Kruskal-Wallis Test
k_stat, p_value = kruskal(weather1,weather2,weather3,weather4)
print('p_value :',p_value)
```

```
p_value : 3.501611300708679e-44
```

Inference -

- Here **p_value < 0.5, so we reject the Null Hypothesis**.
- Therefore we can conclude that **No of cycles rented is different in different weather**.

B) Season

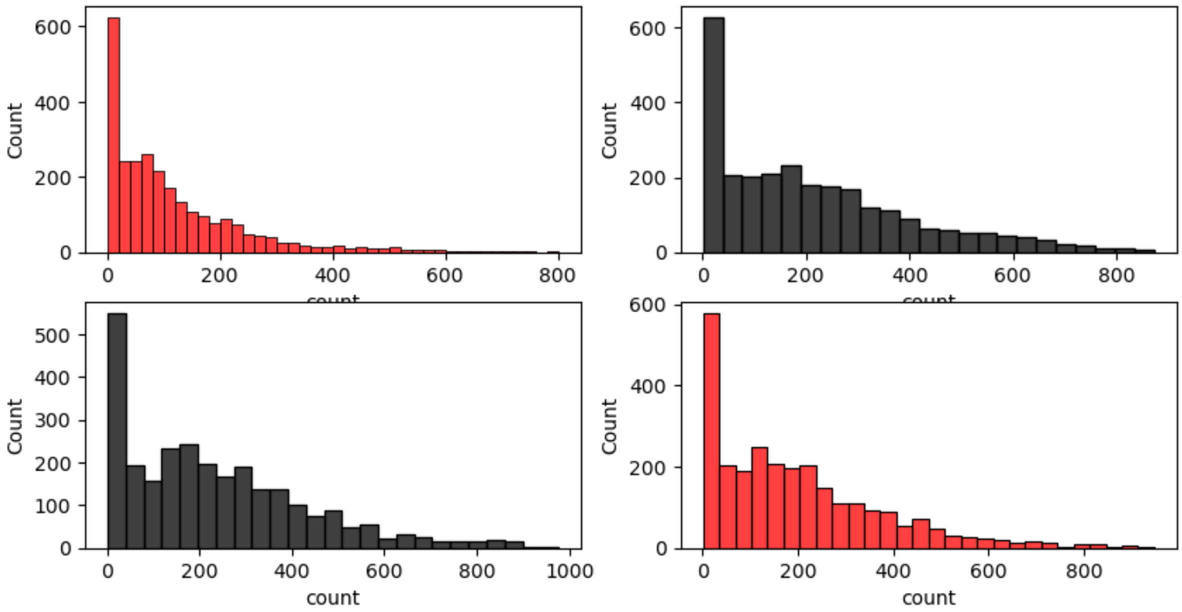
In [162…
```
df.head()
```

Out[162]:

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

In [163…
```
spring = df[df["season"] == 1]
summer = df[df["season"] == 2]
fall = df[df["season"] == 3]
winter = df[df["season"] == 4]
```

In [171…
```
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(10,5))

sns.histplot(spring,x='count', ax=axis[0,0],color='r')
sns.histplot(summer,x='count', ax=axis[0,1],color='k')
sns.histplot(fall,x='count', ax=axis[1,0],color='k')
sns.histplot(winter,x='count', ax=axis[1,1],color='r')
plt.show()
```



In [198…

```
# Levene test - (To check variance of groups)
'''
Assumptions -
    H0 : Variances are equal
    Ha : Variances are not equal
'''
spring_1 = df[df["season"] == 1]["count"]
summer_2 = df[df["season"] == 2]["count"]
fall_3 = df[df["season"] == 3]["count"]
winter_4 = df[df["season"] == 4]["count"]

levene_stat, p_value = levene(spring_1, summer_2, fall_3, winter_4)
print('p_value :',p_value)
```

```
p_value : 1.0147116860043298e-118
```

As **p-value < 0.5, we reject the null hypothesis**. And therefore the **Variance is not equal between the data**.

Observation -

- From the above graphs and Levene test we can see that the **data does not satisfy the Assumptions of Annova**.

Kruskal Wallis method -

- *H0: No of cycles rented is similar in diiferent season*
- *Ha: No of cycles rented is different in different season*

In [179…

```
spring_1.mean()
```

Out[179]:

```
116.34326135517499
```

In [180…

```
summer_2.mean()
```

Out[180]:

```
215.25137211855105
```

In [181…

```
fall_3.mean()
```

Out[181]:
```
234.417124039517
```

In [182…
```
winter_4.mean()
```

Out[182]:
```
198.98829553767374
```

In [200…
```
# Kruskal-Wallis Test -
k_stat, p_value = kruskal(spring_1, summer_2, fall_3, winter_4)
print('p_value :',p_value)
```

```
p_value : 2.479008372608633e-151
```

- Here **p_value < 0.5, so we reject the Null Hypothesis**.
- Therefore we can conclude that **No of cycles rented is different in different season**.

3. Chi-square test to check if Weather is dependent on the season

- **H0: Weather and Season are independent**
- **Ha: Weather is dependent on Season**

In [190…
```
data = pd.crosstab(index = df["weather"], columns = df["season"])
data
```

Out[190]:

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 [197…
```
chi2_stat, p_value, dof, exp_freq = chi2_contingency(data)
print('p_value :',p_value)
```

```
p_value : 1.5499250736864862e-07
```

Inference -

- Here **p_value < 0.5, so we reject the Null Hypothesis**.
- Therefore we can conclude that **Weather is dependent on Season**.

Insights -

- **Majority of bike rentals come during Fall season**. And the **least no of bike rentals** is during **Spring season**.
- **On average bike rentals are higher on workingday** compared to non-working days
- **Weather 1(Clear) is the most preferred weather** for bike rentals
- Customers hardly rent bikes during Heavy rains.

- **Annova test couldn't be performed** on the data as it **did not hold the Assumptions of Annova**. So **using Kruskal-Wallis test** we **proved** that, **the Number of cycles rented depends on the weather**. It is different for different weather.
- **Using Kruskal-Wallis test** we **proved** that, **the Number of cycles rented depends on the season**. It is different for different seasons.
- **Using 2 Sample T-Test** we **proved** that, **Working day has an effect on the number of electric cycles being rented**.
- **Using Chi-square Test** we **proved** that, **Weather is dependent on seasoon**.

Recommendation -

- The **participation is a bit low on non-working days**. Yulu can **conduct certain events like a bike marathon** to **increase participation** during non-working days
- Weather 1 is the most preferred weather by customers. **Yulu can add more safety equipments and additional features** to improve the quality of their bikes to be **more adaptable other weather conditions**. And these features should be brought forward to the customers through **awareness programs**.
- **Participation during Spring(season 1) is low**. Yulu can **conduct social events** during this period to **engage their customers**.
- **Fall season is the season with the most number of bike rentals**. Yulu has to **make sure that there is enough number of bikes** at thier Yulu Hubs during this season so that they don't run out, as it will affect the customer satisfaction.
- Yulu can use reward programs on their mobile apps to motivate their customers and keep them engaged to the service.
- Number of bike rentals seem to be very low during heavy rains. During this time the number of bikes at Yulu stations can be reduced. And company can perform repair and maintanence activies on the bike during this time.

```
In [ ]:
```