In this case study we indent to conduct data analysis on Yulu business data to delve into the factors influencing the demand for their electric cycles in indian market for its Strategic Expansion and Revenue recovery.

Import the dataset and do usual data analysis steps like checking the structure & characteristics of the dataset

a. Examine dataset structure, characteristics, and statistical summary.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sn
import warnings
warnings.filterwarnings('ignore')
from scipy.stats import norm,ttest_1samp,ttest_ind,ttest_rel
from scipy.stats import f_oneway,kruskal,shapiro
from statsmodels.graphics.gofplots import qqplot
```

In [2]:
```
yulu_df=pd.read_csv(r"C:\Users\hp\OneDrive\Desktop\scaler assignment\bike_sharing.csv")
yulu_df
```

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	0.0000	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

Column Profiling:

- datetime: datetime
- season: season (1: spring, 2: summer, 3: fall, 4: winter)
- holiday : whether day is a holiday or not
- workingday : if day is neither weekend nor holiday is 1, otherwise is 0.
- weather: o 1: Clear, Few clouds, partly cloudy o 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist o 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds o 4: Heavy Rain + Ice Pellets + 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

In [3]:
```
yulu_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 [4]:
```
for i in yulu_df.columns[:-7]:
 yulu_df[i] = yulu_df[i].astype('object')
yulu_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  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: float64(3), int64(4), object(5)
memory usage: 1020.7+ KB
```

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

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

In [6]:
```
yulu_df.season.unique()
```

Out[6]:
```
array([1, 2, 3, 4], dtype=object)
```

In [7]:
```
yulu_df.weather.unique()
```

Out[7]:
```
array([1, 2, 3, 4], dtype=object)
```

In [8]:
```
yulu_df.datetime.min(),yulu_df.datetime.max()
```

Out[8]:
```
('2011-01-01 00:00:00', '2012-12-19 23:00:00')
```

In [9]:
```
yulu_df.describe(include='object')
```

Out[9]:

	datetime	season	holiday	workingday	weather
count	10886	10886	10886	10886	10886
unique	10886	4	2	2	4
top	2011-01-01 00:00:00	4	0	1	1
freq	1	2734	10575	7412	7192

In [10]:
```
yulu_df.describe()
```

Out[10]:

	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

b. Identify missing values and perform Imputation using an appropriate method.

In [11]:
```
yulu_df.isna().sum()
```

Out[11]:
```
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
```

Comments: There is no null values in the dataframe

c. Identify and remove duplicate records.

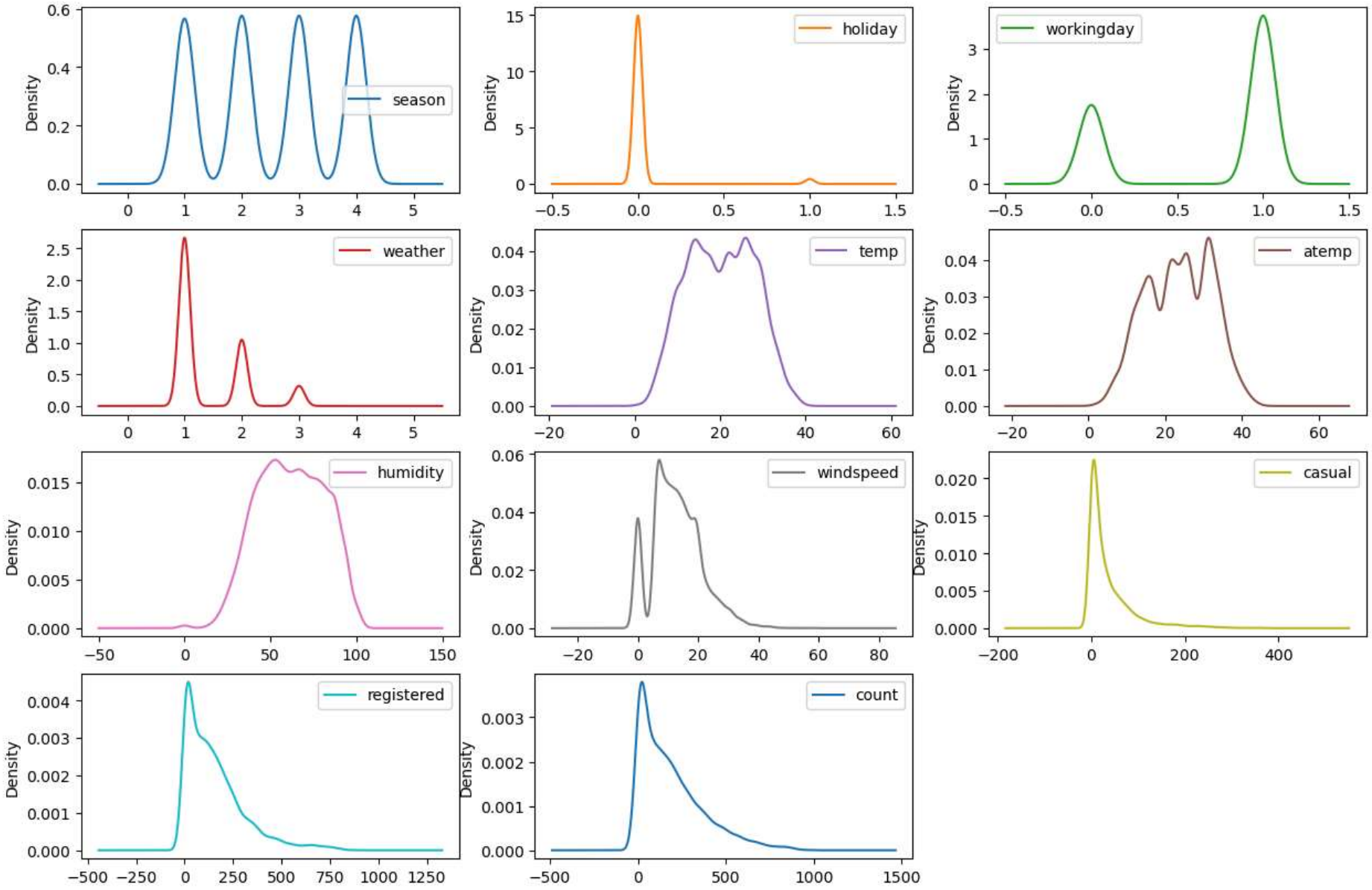
In [12]:
```
df=yulu_df.duplicated().sum()
df
```

Out[12]:
```
0
```

comments: There is no duplicates in the dataset

d. Analyze the distribution of Numerical & Categorical variables, separately

In [13]:
```
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = [15,10]
yulu_df.plot(kind = 'density', subplots = True, layout = (4,3), sharex = False)
plt.show()
```



In [14]:
```
yulu_df.skew()
```

Out[14]:
```
season       -0.007076
holiday       5.660517
workingday   -0.776163
weather       1.243484
temp          0.003691
atemp        -0.102560
humidity     -0.086335
windspeed     0.588767
casual        2.495748
registered    1.524805
count         1.242066
dtype: float64
```

In [15]:
```
yulu_df.kurt()
```

Out[15]:
```
season        -1.355661
holiday       30.046975
workingday    -1.397828
weather        0.395533
temp          -0.914530
atemp         -0.850076
humidity      -0.759818
windspeed      0.630133
casual         7.551629
registered     2.626081
count          1.300093
dtype: float64
```

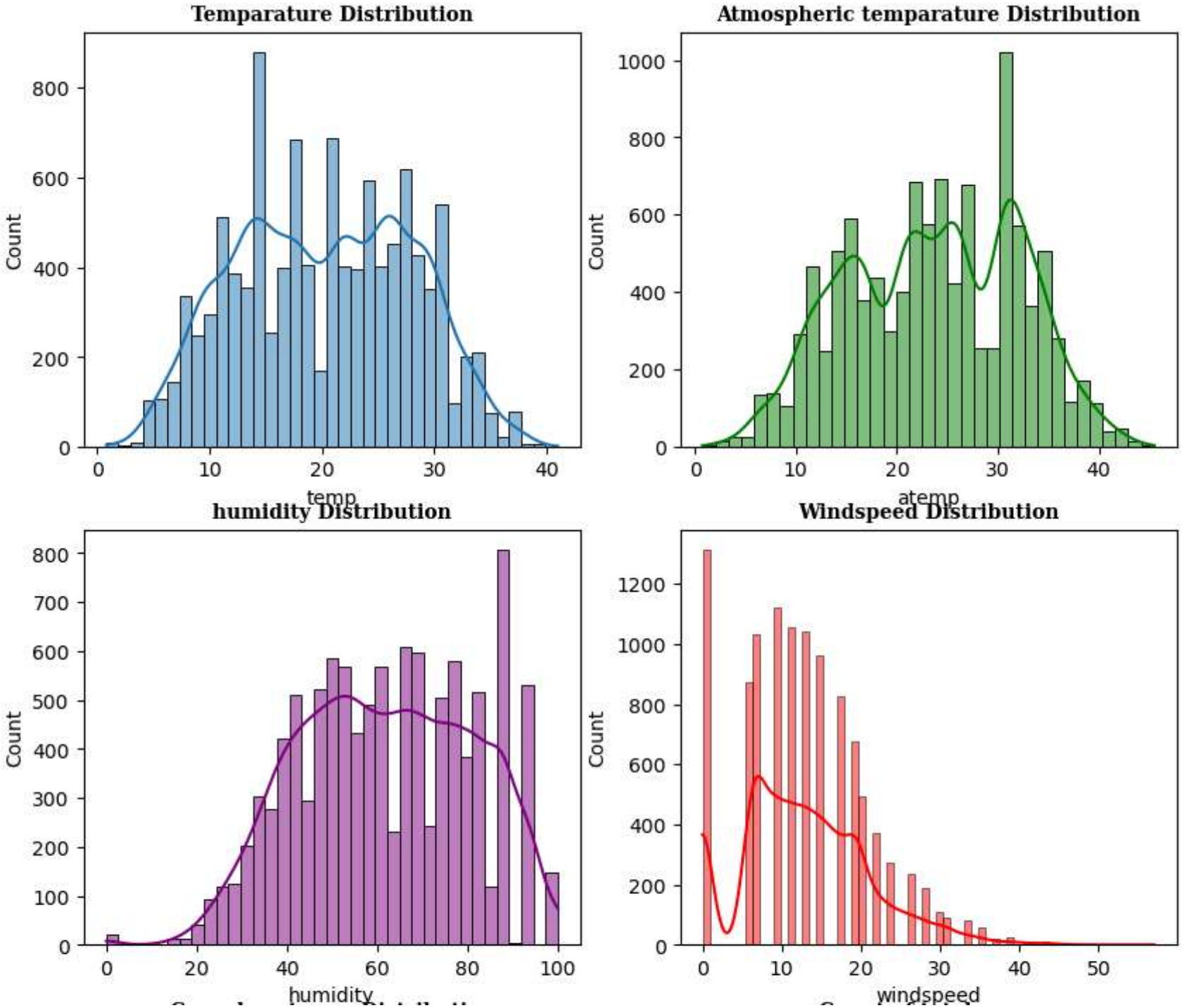
comments: weather ,casual ,registered and count are right skewed distribution Temparture and humidity have normal distribution in comparison to other attributes.

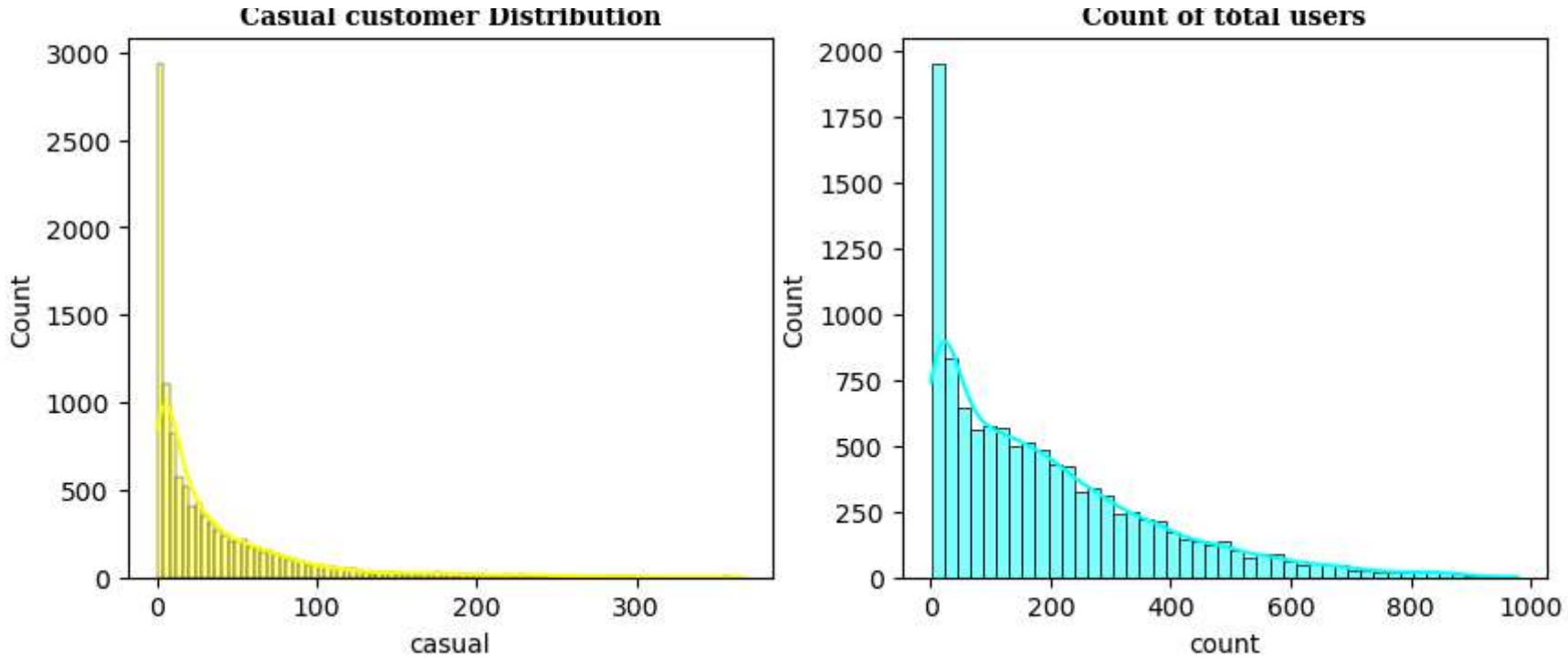
In [16]:
```
plt.figure(figsize=(10,13))
plt.subplot(3,2,1)
sn.histplot(yulu_df['temp'],kde=True) # continous variable
plt.title('Temparature Distribution',{'font':'serif', 'size':10,'weight':'bold'})
plt.subplot(3,2,2)
sn.histplot(yulu_df['atemp'],color='green',kde=True) # continous variable
plt.title('Atmospheric temparature Distribution',{'font':'serif', 'size':10,'weight':'bold'})
plt.subplot(3,2,3)
sn.histplot(yulu_df['humidity'],color='purple',kde=True)  # continous variable
plt.title('humidity Distribution',{'font':'serif', 'size':10,'weight':'bold'})
plt.subplot(3,2,4)
sn.histplot(yulu_df['windspeed'],color='red',kde=True)  # continous variable
```

```
plt.title('Windspeed Distribution',{'font':'serif', 'size':10,'weight':'bold'})
plt.subplot(3,2,5)
sn.histplot(yulu_df['casual'],color='yellow',kde=True) # continous variable
plt.title('Casual customer Distribution',{'font':'serif', 'size':10,'weight':'bold'})
plt.subplot(3,2,6)
sn.histplot(yulu_df['count'],color='cyan',kde=True)  # continous variable
plt.title('Count of total users',{'font':'serif', 'size':10,'weight':'bold'})
```

Out[16]:

```
Text(0.5, 1.0, 'Count of total users')
```

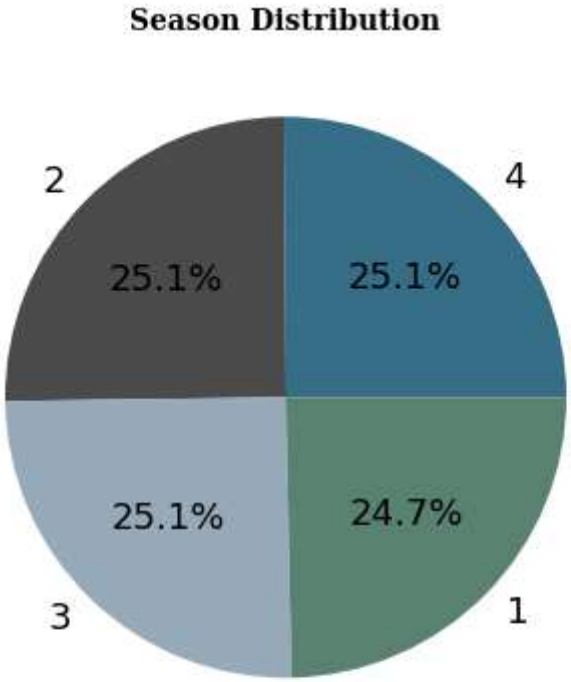

Temparature Distribution
Count
temp
Atmospheric temparature Distribution
Count
atemp
humidity Distribution
Count
humidity
Windspeed Distribution
Count
windspeed

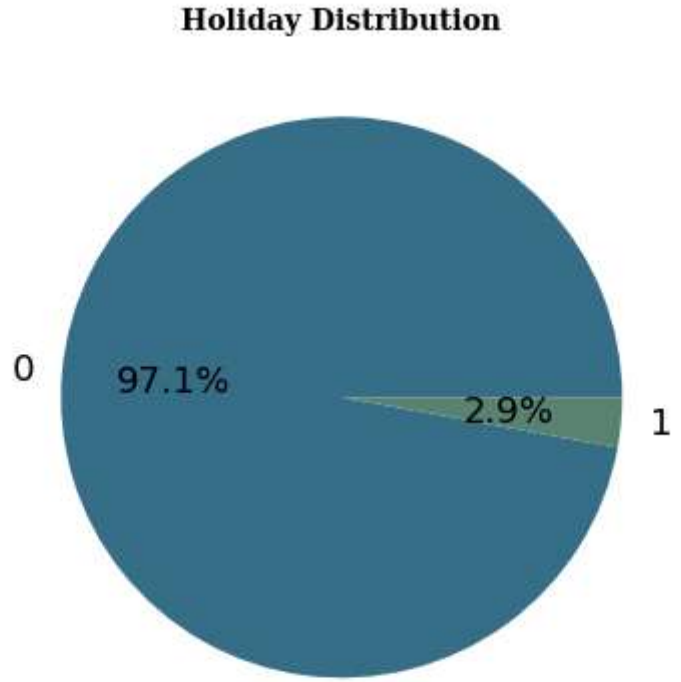


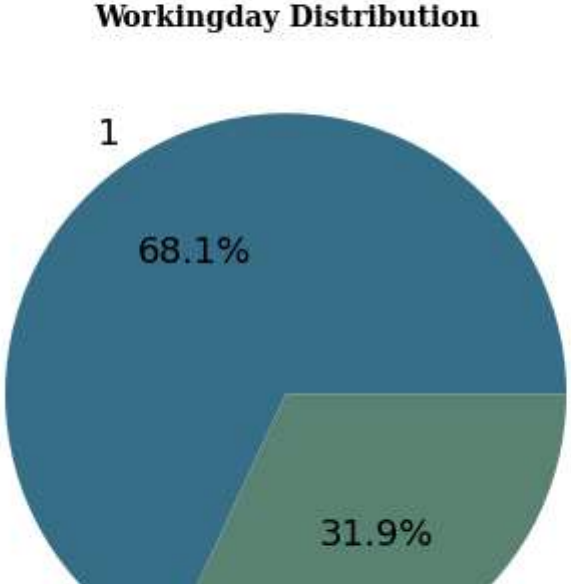
In [17]:

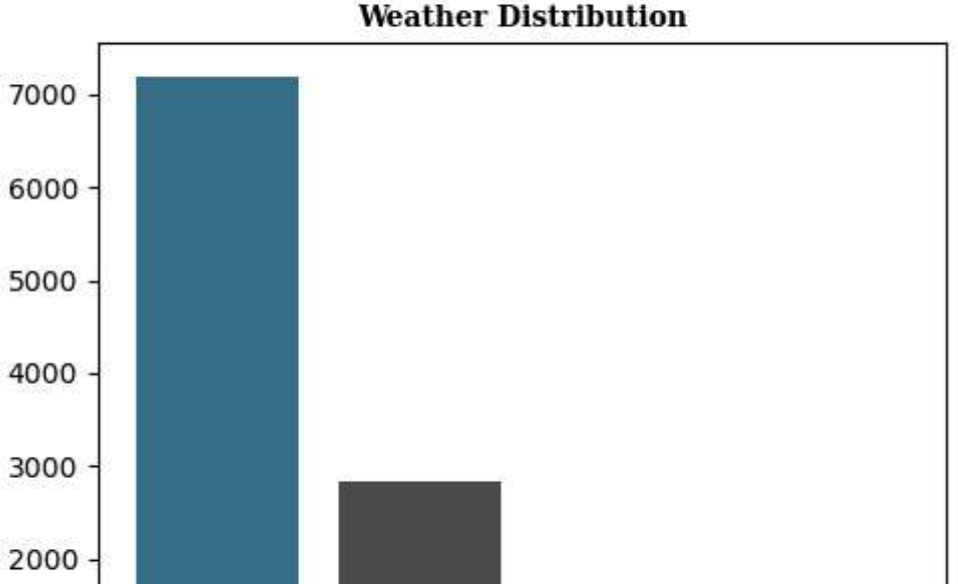
```
fig = plt.figure(figsize = (12,10))
gs = fig.add_gridspec(2,2)
 # creating pie chart for gender disribution
ax0 = fig.add_subplot(gs[0,0])

color_map = ["#3A7089", "#4b4b4c",'#99AEBB','#5C8374']
ax0.pie(yulu_df['season'].value_counts().values,labels = yulu_df['season'].value_counts().index,autopct = '%.1f%%',colors = color
plt.title('Season Distribution',{'font':'serif', 'size':10,'weight':'bold'})


ax1 = fig.add_subplot(gs[0,1])
color_map=["#3A7089",'#5C8374']
ax1.pie(yulu_df['holiday'].value_counts().values,labels = yulu_df['holiday'].value_counts().index,autopct = '%.1f%%',colors = col
plt.title('Holiday Distribution',{'font':'serif', 'size':10,'weight':'bold'})


ax2 = fig.add_subplot(gs[1,0])
color_map=["#3A7089",'#5C8374']
ax2.pie(yulu_df['workingday'].value_counts().values,labels = yulu_df['workingday'].value_counts().index,autopct = '%.1f%%',colors
plt.title('Workingday Distribution',{'font':'serif', 'size':10,'weight':'bold'})
```

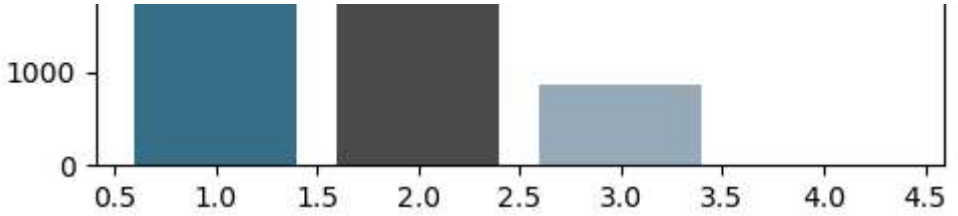
```
ax3 = fig.add_subplot(gs[1,1])
color_map = ["#3A7089", "#4b4b4c",'#99AEBB','#5C8374']
#ax3.pie(yulu_df['weather'].value_counts().values,labels = yulu_df['weather'].value_counts().index,autopct = '%.1f%%',colors = co
ax=plt.bar(yulu_df['weather'].value_counts().index,yulu_df['weather'].value_counts(),color=["#3A7089", "#4b4b4c",'#99AEBB','#5C83
plt.title('Weather Distribution',{'font':'serif', 'size':10,'weight':'bold'})
plt.show()
```


Season Distribution
2
4
25.1%
25.1%
25.1%
24.7%
3
1


Holiday Distribution
0
97.1%
2.9%
1


Workingday Distribution
1
68.1%
31.9%


Weather Distribution
7000
6000
5000
4000
3000
2000

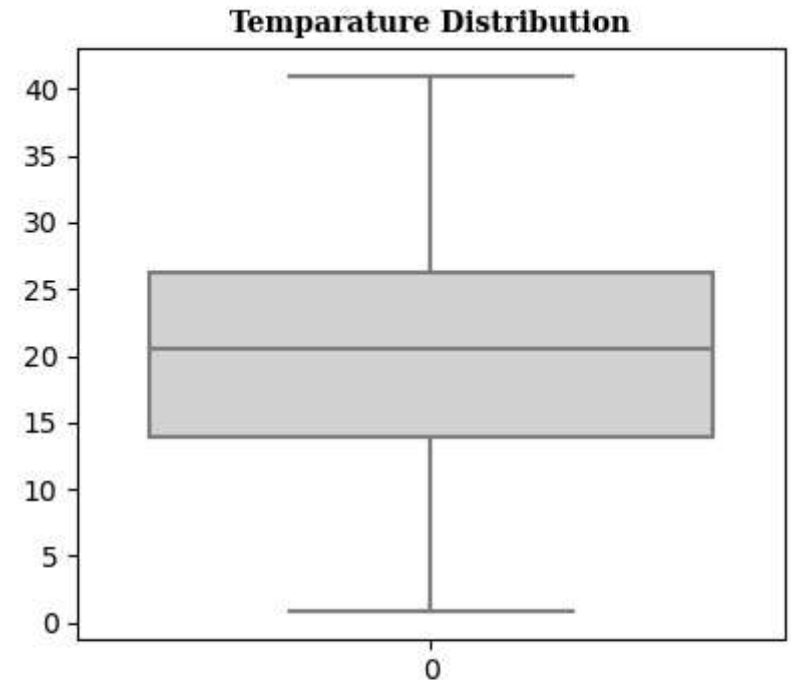


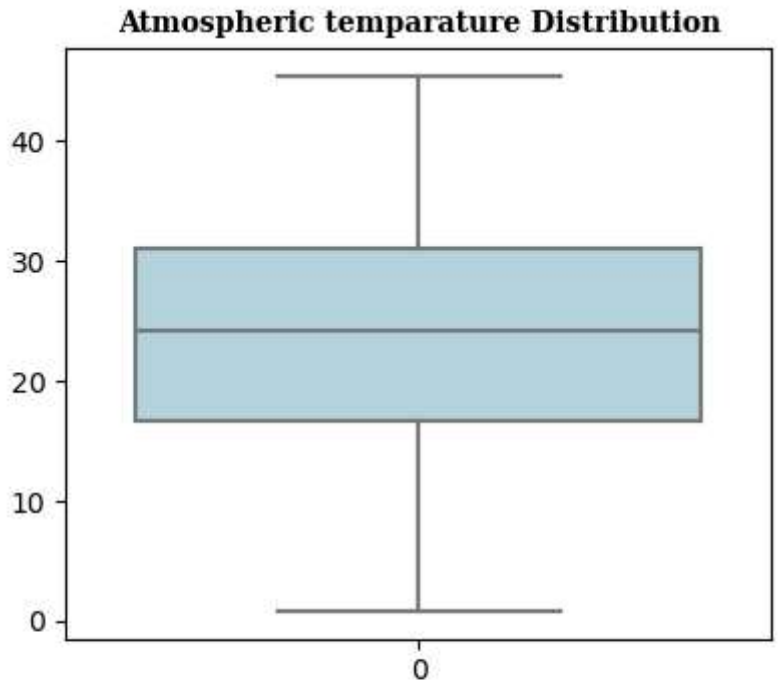
comments: weather 4 is least occured while weather 1 occur predominantly. 68% of the days are working and 32% forms non working days.holidays forms 2.9% of total days.

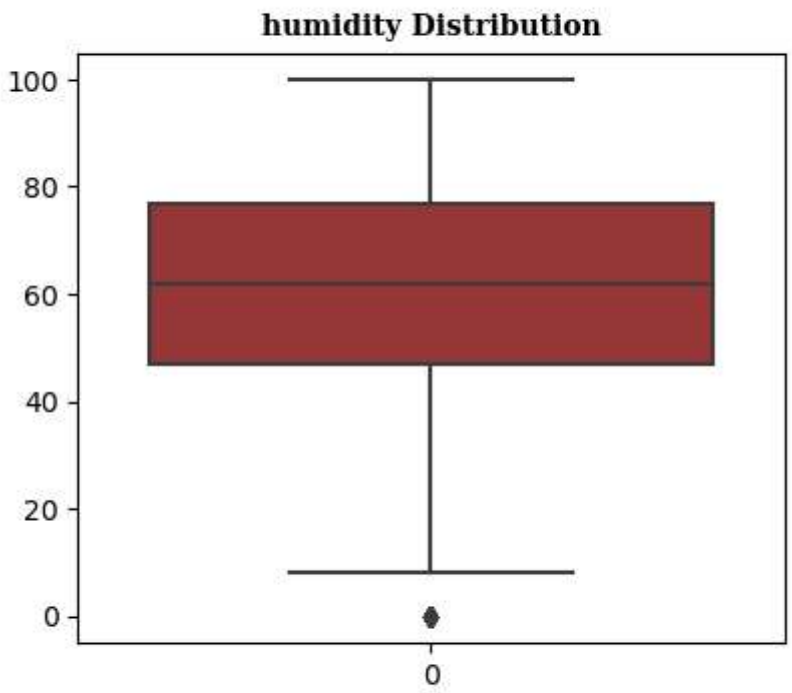
e. Check for Outliers and deal with them accordingly.

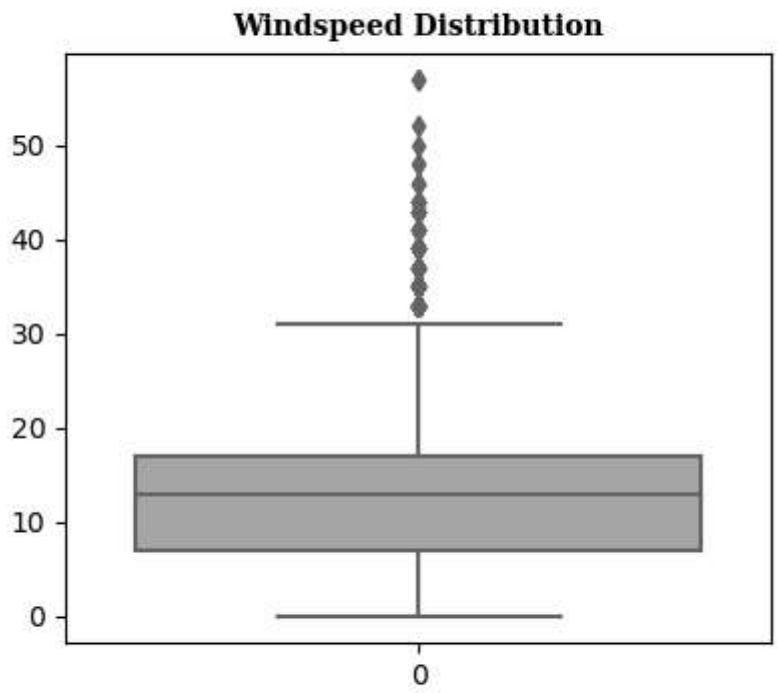
In [18]:
```
plt.figure(figsize=(10,13))
plt.subplot(3,2,1)
sn.boxplot(yulu_df['temp'],color='lightgrey') # continous variable
plt.title('Temparature Distribution',{'font':'serif', 'size':10,'weight':'bold'})
plt.subplot(3,2,2)
sn.boxplot(yulu_df['atemp'],color='lightblue') # continous variable
plt.title('Atmospheric temparature Distribution',{'font':'serif', 'size':10,'weight':'bold'})
plt.subplot(3,2,3)
sn.boxplot(yulu_df['humidity'],color='brown')  # continous variable
plt.title('humidity Distribution',{'font':'serif', 'size':10,'weight':'bold'})
plt.subplot(3,2,4)
sn.boxplot(yulu_df['windspeed'],color='darkgrey')  # continous variable
plt.title('Windspeed Distribution',{'font':'serif', 'size':10,'weight':'bold'})
plt.subplot(3,2,5)
sn.boxplot(yulu_df['casual'],color='skyblue') # continous variable
plt.title('Casual customer Distribution',{'font':'serif', 'size':10,'weight':'bold'})
plt.subplot(3,2,6)
sn.boxplot(yulu_df['count'],color='grey')  # continous variable
plt.title('Count of total users',{'font':'serif', 'size':10,'weight':'bold'})
```

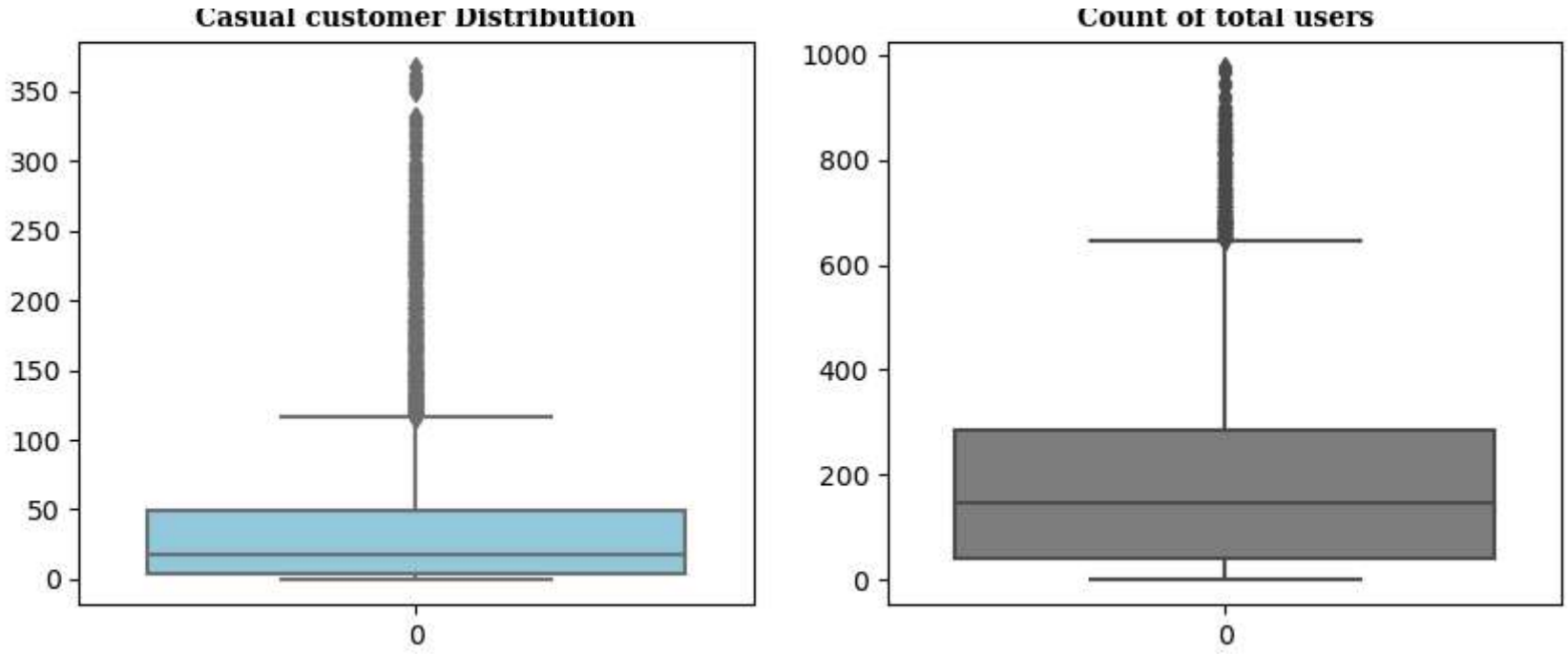
Out[18]:
```
Text(0.5, 1.0, 'Count of total users')
```


Temparature Distribution
40
35
30
25
20
15
10
5
0
0


Atmospheric temparature Distribution
40
30
20
10
0
0


humidity Distribution
100
80
60
40
20
0
0


Windspeed Distribution
50
40
30
20
10
0
0



Comments: temparature and humidity have least outliers Windspeed,count of casual customers and total customers have wider outliers

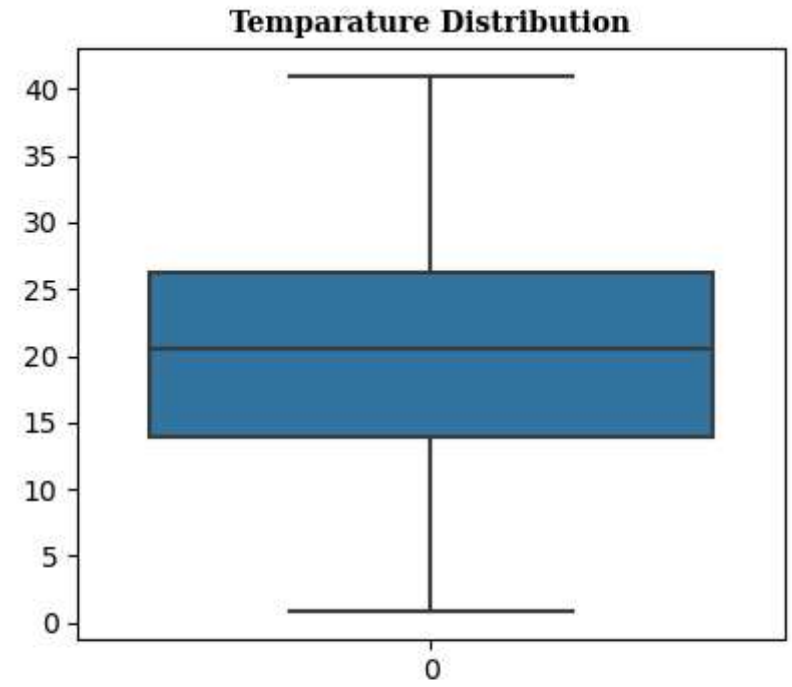
Outlier removal

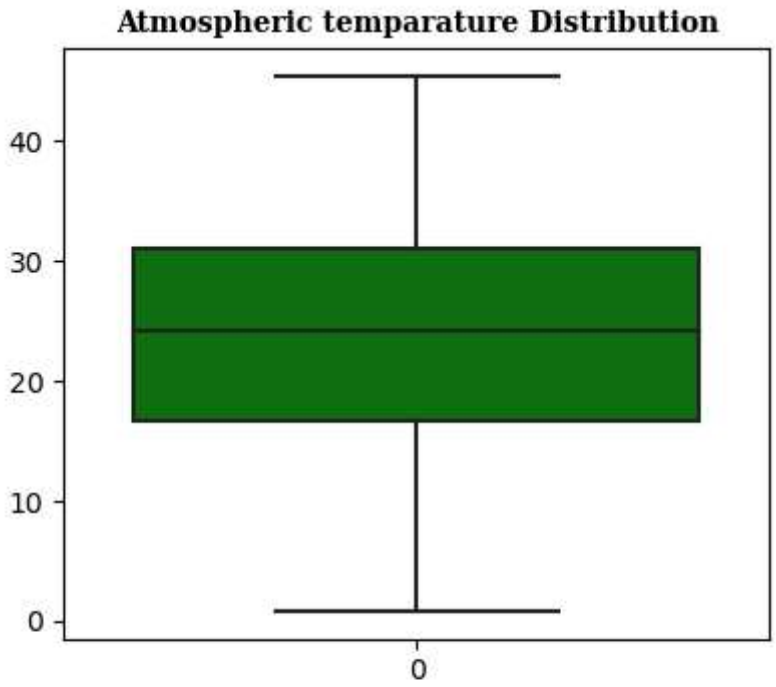
In [19]:
```
# defining function to remove outliers:here np.clip will replace the outliers with limiting values

def clip_data_between_percentiles(data):
    # Calculate the lower and upper percentile values
    lower_limit = np.percentile(data, 25)
    upper_limit = np.percentile(data, 75)
    IQR= upper_limit-lower_limit
    upper_bound = upper_limit+1.5*IQR
    lower_bound = lower_limit-1.5*IQR

    # Clip the data between the lower and upper percentile values
    clipped_df = np.clip(data, lower_bound, upper_bound)

    return pd.DataFrame(clipped_df)
```

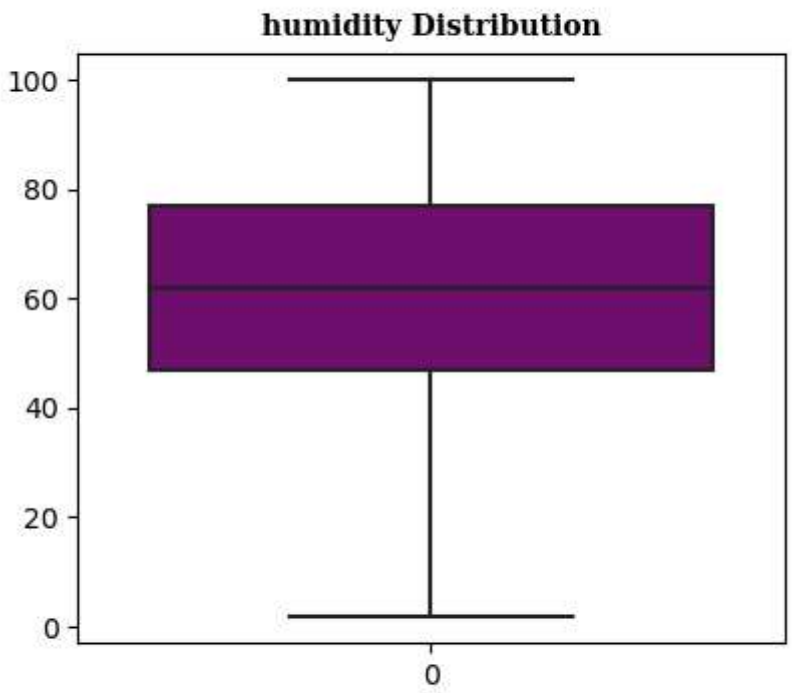
In [20]:
```
# Applying the defined fuction to those continous variables
yulu_df['humidity']=clip_data_between_percentiles(yulu_df['humidity'])
yulu_df['windspeed']=clip_data_between_percentiles(yulu_df['windspeed'])
yulu_df['casual']=clip_data_between_percentiles(yulu_df['casual'])
yulu_df['count']=clip_data_between_percentiles(yulu_df['count'])
```

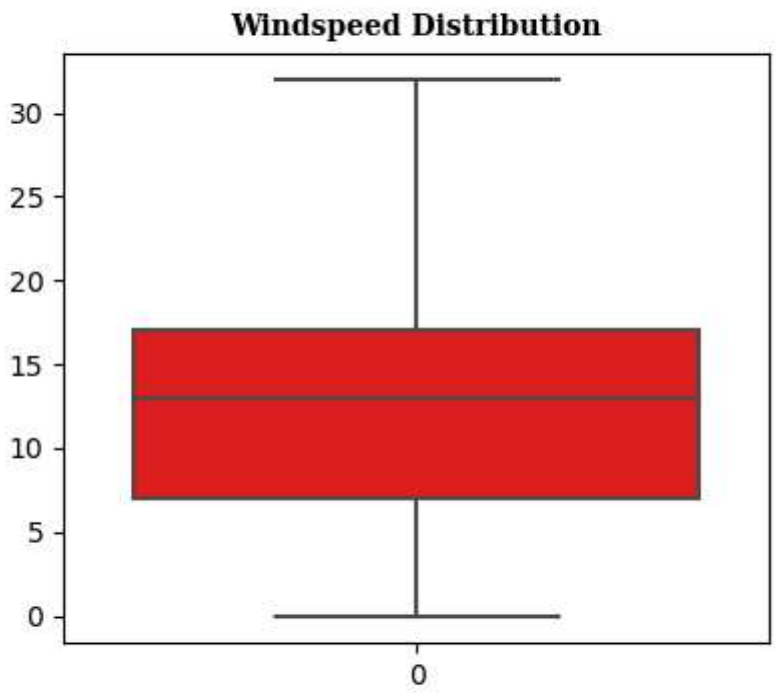
In [21]:
```
plt.figure(figsize=(10,13))
plt.subplot(3,2,1)
sn.boxplot(yulu_df['temp']) # continous variable
plt.title('Temparature Distribution',{'font':'serif', 'size':10,'weight':'bold'})
plt.subplot(3,2,2)
sn.boxplot(yulu_df['atemp'],color='green') # continous variable
plt.title('Atmospheric temparature Distribution',{'font':'serif', 'size':10,'weight':'bold'})
plt.subplot(3,2,3)
sn.boxplot(yulu_df['humidity'],color='purple')  # continous variable
plt.title('humidity Distribution',{'font':'serif', 'size':10,'weight':'bold'})
plt.subplot(3,2,4)
sn.boxplot(yulu_df['windspeed'],color='red')  # continous variable
plt.title('Windspeed Distribution',{'font':'serif', 'size':10,'weight':'bold'})
plt.subplot(3,2,5)
sn.boxplot(yulu_df['casual'],color='yellow') # continous variable
plt.title('Casual customer Distribution',{'font':'serif', 'size':10,'weight':'bold'})
plt.subplot(3,2,6)
sn.boxplot(yulu_df['count'],color='cyan')  # continous variable
plt.title('Count of total users',{'font':'serif', 'size':10,'weight':'bold'})
```

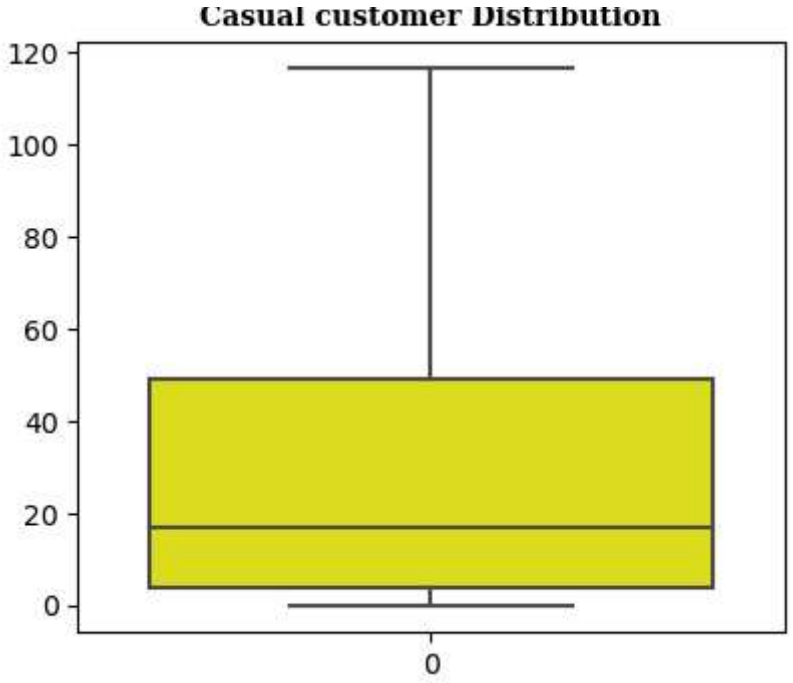
Out[21]:
```
Text(0.5, 1.0, 'Count of total users')
```

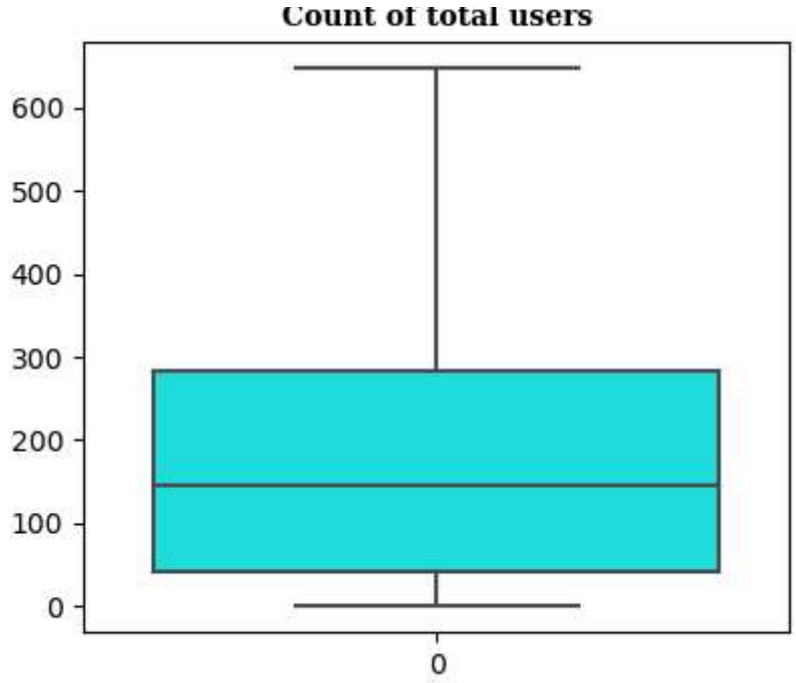

Temparature Distribution
40
35
30
25
20
15
10
5
0
0


Atmospheric temparature Distribution
40
30
20
10
0
0


humidity Distribution
100
80
60
40
20
0
0

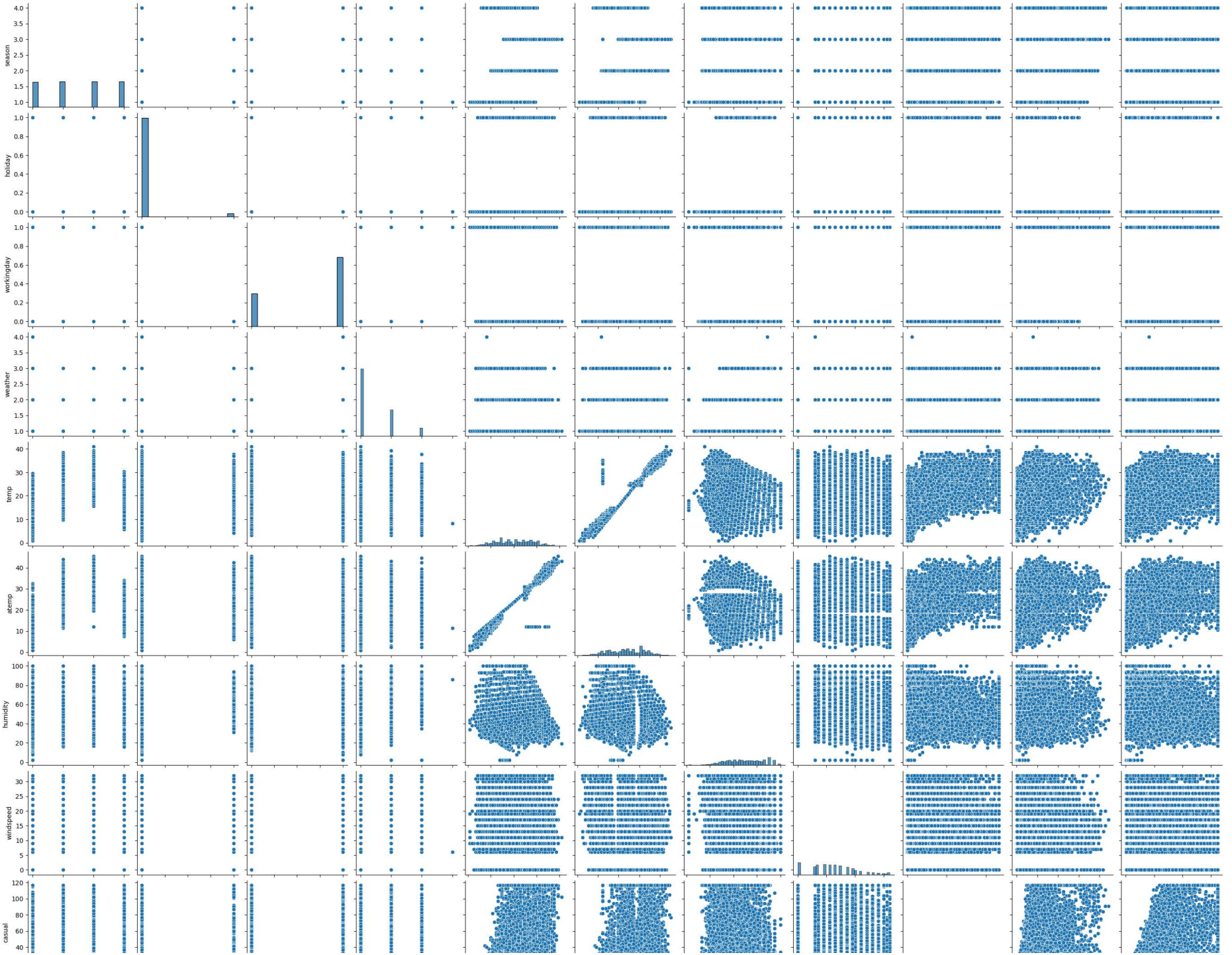

Windspeed Distribution
30
25
20
15
10
5
0
0

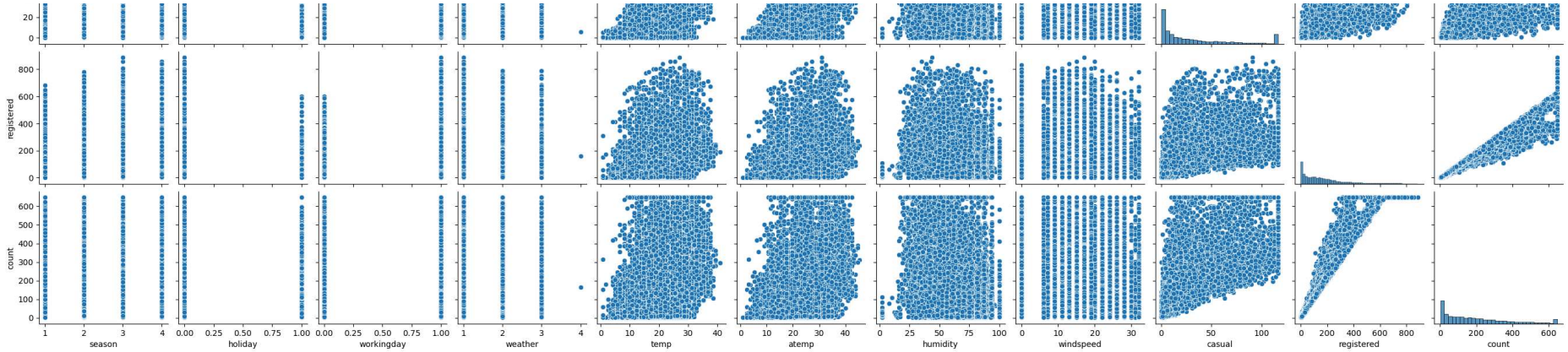




In [22]:
```
# Check correlation among different factors using pair plots.
sn.pairplot(data=yulu_df)
```

Out[22]:
```
<seaborn.axisgrid.PairGrid at 0x29cdd48dd50>
```


season
holiday
workingday
weather
temp
atemp
humidity
windspeed
casual



Comments: pairplot shows higher linear relation between temp and atemp and also between registered customers and total customers

In [23]:
```
# Checking the correlation among different factors
yulu_df.corr()
```

Out[23]:

	temp	atemp	humidity	windspeed	casual	registered	count
temp	1.000000	0.984948	-0.065107	-0.015521	0.542221	0.318571	0.399567
atemp	0.984948	1.000000	-0.043673	-0.055305	0.535456	0.314635	0.395062
humidity	-0.065107	-0.043673	1.000000	-0.320164	-0.378298	-0.265814	-0.323867
windspeed	-0.015521	-0.055305	-0.320164	1.000000	0.110620	0.095128	0.109054
casual	0.542221	0.535456	-0.378298	0.110620	1.000000	0.573120	0.744425
registered	0.318571	0.314635	-0.265814	0.095128	0.573120	1.000000	0.959190
count	0.399567	0.395062	-0.323867	0.109054	0.744425	0.959190	1.000000

In [24]:
```
# Check correlation among different factors using heatmap.
sn.heatmap(yulu_df.corr(),annot=True)
```

Out[24]:
```
<Axes: >
```

Comments: temp and atemp are correlated to 98% and registered and count to 96 %

In [25]:
```
yulu_df.drop(columns=['registered','atemp'],axis=1,inplace=True)
yulu_df
```

Out[25]:

	datetime	season	holiday	workingday	weather	temp	humidity	windspeed	casual	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	81	0.0000	3.0	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	80	0.0000	8.0	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	80	0.0000	5.0	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	75	0.0000	3.0	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	75	0.0000	0.0	1
...	...	...	...	...	...	...	...	...	...	...
10881	2012-12-19 19:00:00	4	0	1	1	15.58	50	26.0027	7.0	336
10882	2012-12-19 20:00:00	4	0	1	1	14.76	57	15.0013	10.0	241
10883	2012-12-19 21:00:00	4	0	1	1	13.94	61	15.0013	4.0	168
10884	2012-12-19 22:00:00	4	0	1	1	13.94	61	6.0032	12.0	129
10885	2012-12-19 23:00:00	4	0	1	1	13.12	66	8.9981	4.0	88

10886 rows × 10 columns

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

In [26]:
```
yulu_df.groupby('workingday')['count'].mean()
```

Out[26]:
```
workingday
0    187.941278
1    189.062736
Name: count, dtype: float64
```

a. Formulate Null Hypothesis (H0) and Alternate Hypothesis (H1)

H0=There is no significant difference mu1=mu2

H1 =There is significant diffrence mu1=!mu2

In [27]:
```
weekday=yulu_df[yulu_df['workingday']==1][['count']]
weekends=yulu_df[yulu_df['workingday']==0][['count']]
weekday
```

Out[27]:

	count
47	5
48	2
49	1
50	3
51	30
...	...
10881	336
10882	241
10883	168
10884	129
10885	88

7412 rows × 1 columns

In [28]:
```
weekends
```

Out[28]:

	count
0	16
1	40
2	32
3	13
4	1
...	...
10809	109
10810	122
10811	106
10812	89
10813	33

3474 rows × 1 columns

In [29]:
```
# since we intent to find relationship between categorical and numerical variables we use ttest independent
test_statistics,p_value=ttest_ind(weekday,weekends)
test_statistics,p_value
```

Out[29]:
```
(array([0.31632604]), array([0.75176111]))
```

In [30]:
```
#our significance interval is set as 5%
if p_value < 0.05:
    print("Reject H0")
    print("The no. of bike rides depend on whether its Weekdays and Weekends")
else:
    print("Fail to reject H0")
    print("The no. of bike rides  doesn't depend on whether its Weekdays and Weekends")
```

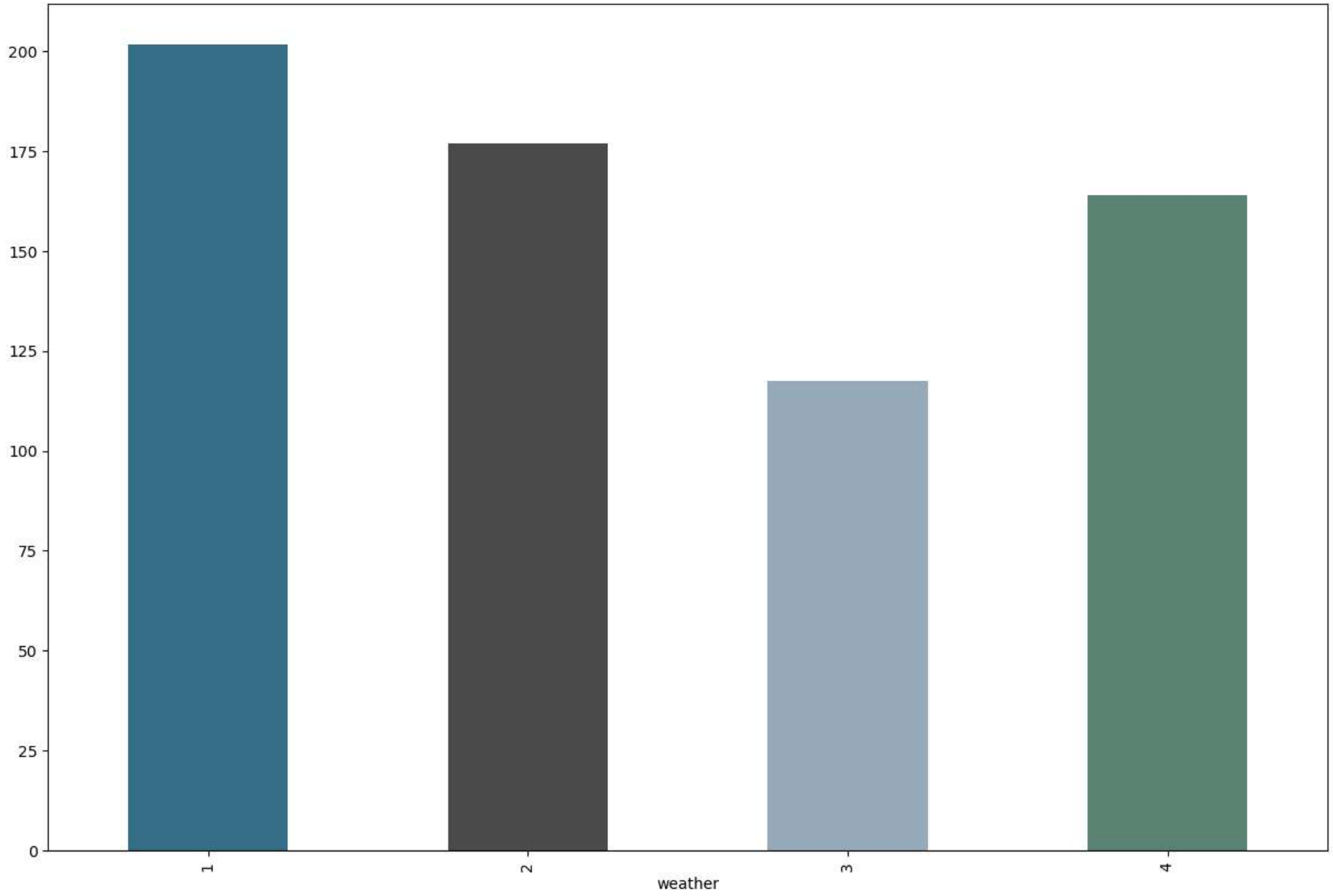
```
Fail to reject H0
The no. of bike rides  doesn't depend on whether its Weekdays and Weekends
```

Comments: From the 2 sample ttest it is found that we have weekday or weekend does not determine much on the no of shared electric cycles.

4. Check if the demand of bicycles on rent is the same for different Weather conditions?

In [31]:
```
yulu_weather=yulu_df.groupby('weather')['count'].mean()
yulu_weather.plot(kind='bar',color= ["#3A7089", "#4b4b4c",'#99AEBB','#5C8374','#6F7597'])
```

Out[31]:
```
<Axes: xlabel='weather'>
```



a. Formulate Null Hypothesis (H0) and Alternate Hypothesis (H1)

H0=the demand of bicycles on rent is the same for different Weather conditions

Ha=the demand of bicycles on rent is not the same for different Weather conditions

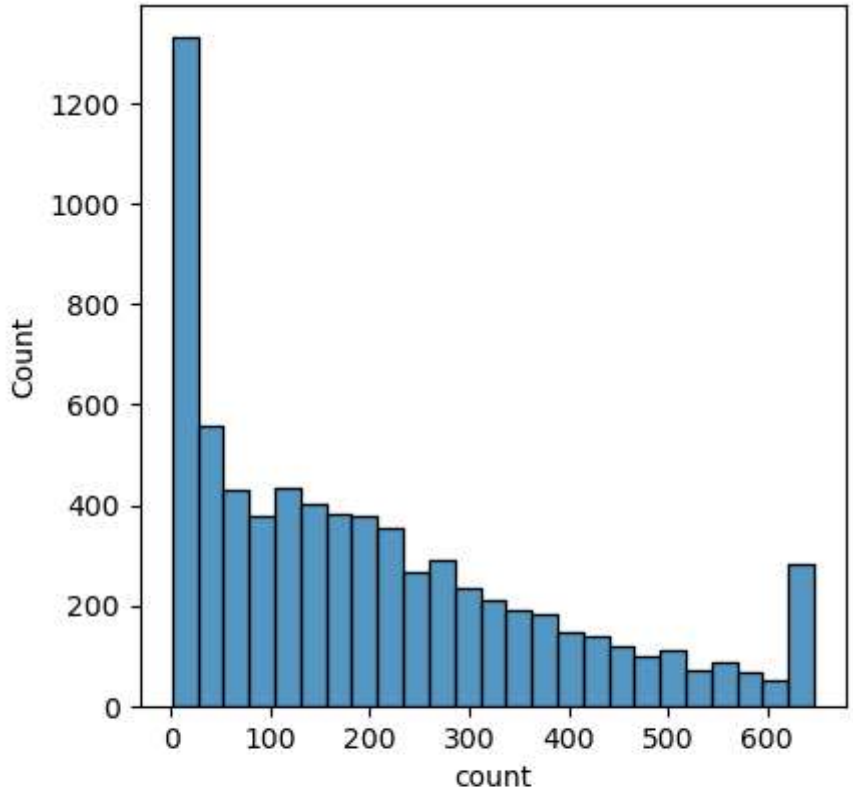
In [32]:
```
#b. Select an appropriate test -
weather1=yulu_df[yulu_df['weather']==1]['count']
weather2=yulu_df[yulu_df['weather']==2]['count']
weather3=yulu_df[yulu_df['weather']==3]['count']
weather4=yulu_df[yulu_df['weather']==4]['count']
```

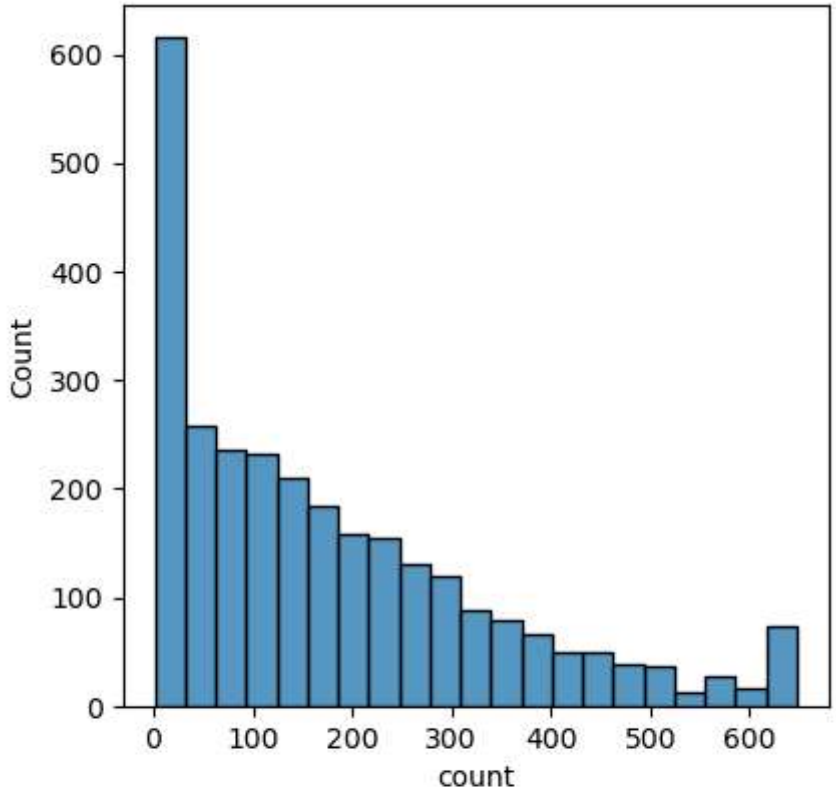
In [33]:
```
#since there are four categories we go for one way anova test
f_stats, p_value = f_oneway(weather1,weather2,weather3,weather4)
f_stats, p_value
```

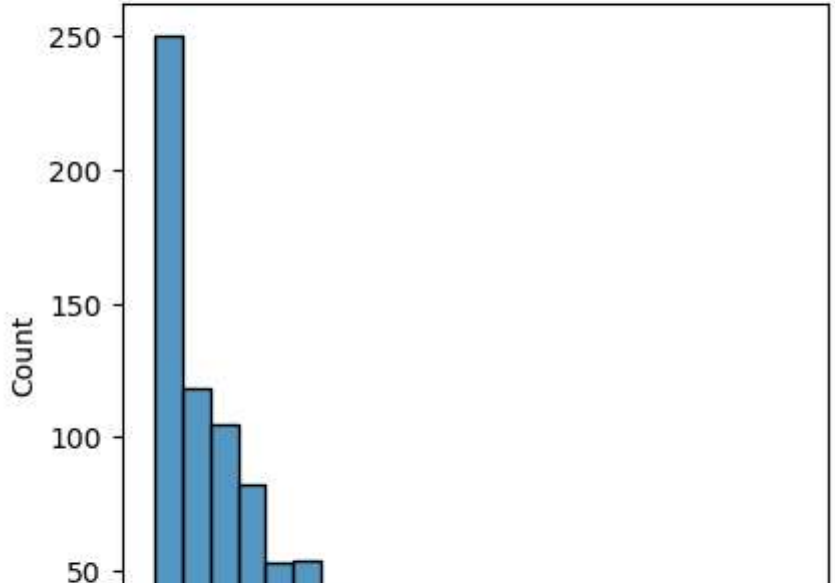
Out[33]:
```
(68.4116520342703, 8.034967610817961e-44)
```

But in order to reley on one way anova test the dististribution should obey assumption

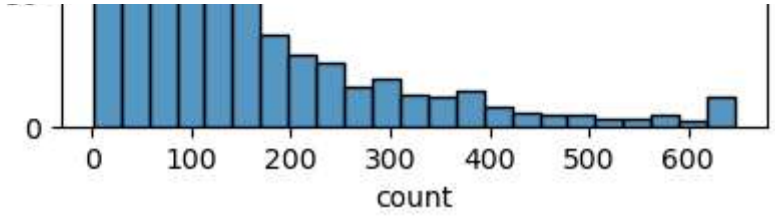
In [34]:
```
# c. Check assumptions of the test
# 1. Use Histogram, Q-Q Plot, Skewness & Kurtosis

plt.figure(figsize=(10,10))
plt.subplot(2,2,1)
sn.histplot(weather1) # continous variable

plt.subplot(2,2,2)
sn.histplot(weather2) # continous variable

plt.subplot(2,2,3)
sn.histplot(weather3)  # continous variable

plt.subplot(2,2,4)
sn.histplot(weather4)  # continous variable

plt.show()
```

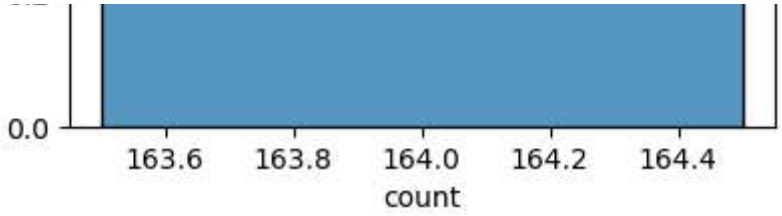

Count
1200
1000
800
600
400
200
0
0
100
200
300
400
500
600
count


Count
600
500
400
300
200
100
0
0
100
200
300
400
500
600
count


Count
250
200
150
100
50


Count
1.0
0.8
0.6
0.4
0.2





In [35]:
```
from scipy.stats import shapiro
# H0: Data is Gaussian
# Ha: Data is not Gaussian

test_stat, p_value=shapiro(weather1)
test_stat, p_value
```

Out[35]:
```
(0.8987792730331421, 0.0)
```

In [36]:
```
if p_value < 0.05:
    print("Reject H0")
    print("Data is not Gaussian")
else:
    print("Fail to reject H0")
    print("Data is Gaussian")
```

```
Reject H0
Data is not Gaussian
```

In [37]:
```
test_stat, p_value=shapiro(weather2)
test_stat, p_value
```

Out[37]:
```
(0.8865376710891724, 1.7712412589065688e-41)
```

In [38]:
```
if p_value < 0.05:
    print("Reject H0")
    print("Data is not Gaussian")
else:
    print("Fail to reject H0")
    print("Data is Gaussian")
```

```
Reject H0
Data is not Gaussian
```

In [39]:
```
test_stat, p_value=shapiro(weather3)
test_stat, p_value
```

Out[39]:
```
(0.788669764995575, 6.402264154069943e-32)
```

In [40]:
```
if p_value < 0.05:
    print("Reject H0")
    print("Data is not Gaussian")
else:
    print("Fail to reject H0")
    print("Data is Gaussian")
```

```
Reject H0
Data is not Gaussian
```

In [41]:
```
#since not following normal distribution we go for Kruskal-Wallis test

# H0: All groups have the same mean
# Ha: One or more groups have different mean

from scipy.stats import kruskal
stat, p_value =kruskal(weather1,weather2,weather3,weather4)
stat, p_value
```

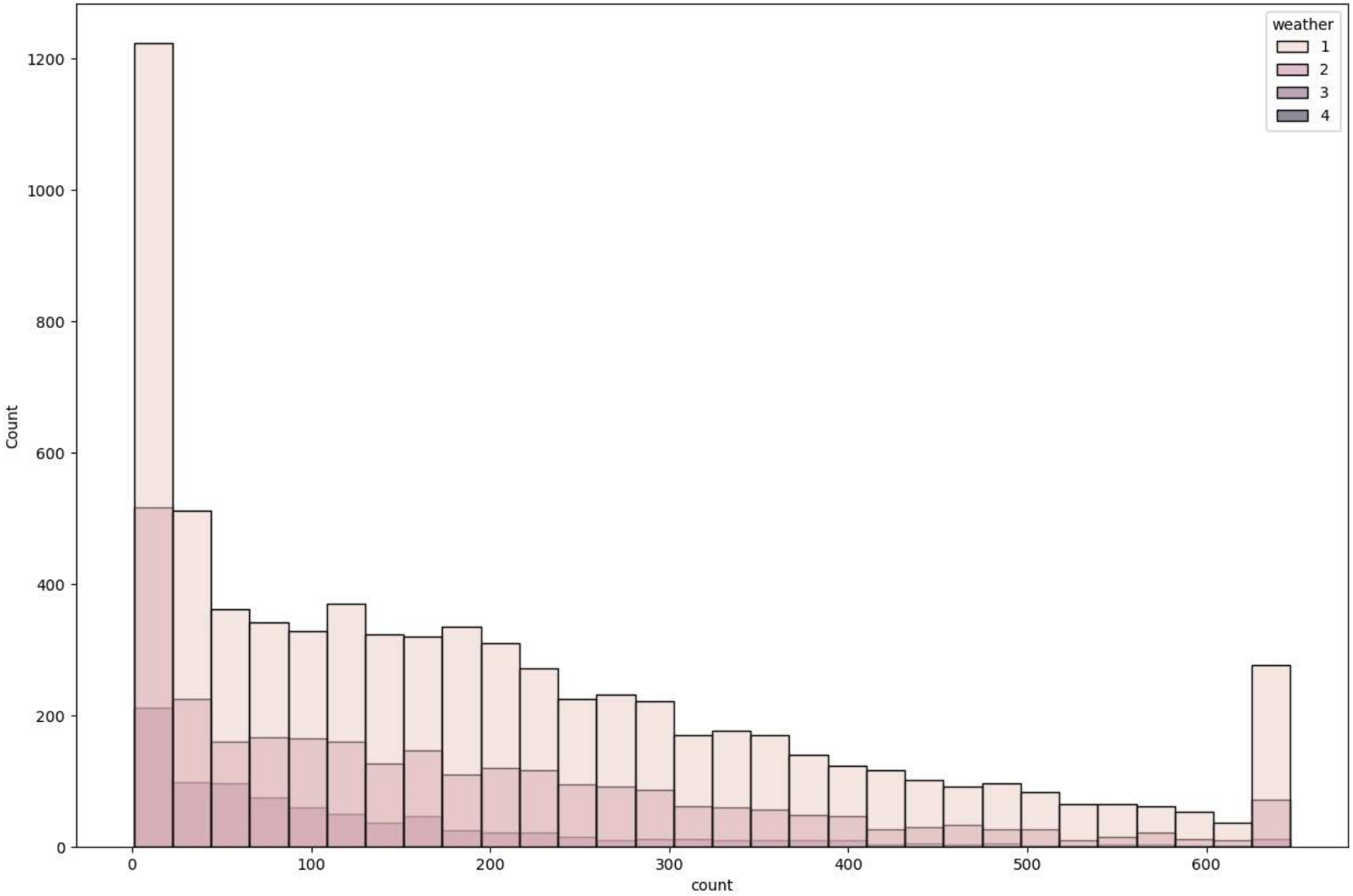
Out[41]:
```
(205.04853208154285, 3.421748763291878e-44)
```

In [42]:
```
if p_value < 0.05:
    print("Reject H0")
    print("Atleast one group have different mean or some are dependent")
else:
    print("Fail to reject H0")
    print("All groups have same mean or they are independent")
```

```
Reject H0
Atleast one group have different mean or some are dependent
```

In [43]:
```
from scipy.stats import levene # Test variance

sn.histplot(data=yulu_df, x="count", hue='weather')
```

Out[43]:
```
<Axes: xlabel='count', ylabel='Count'>
```



In [44]:
```
# H0: Variances are equal
# Ha: Variances are not equal
levene_stat, p_value = levene(weather1,weather2,weather3,weather4)
if p_value < 0.05:
```

```
    print("Variances are not equal")
else:
    print("Variances are equal")
```

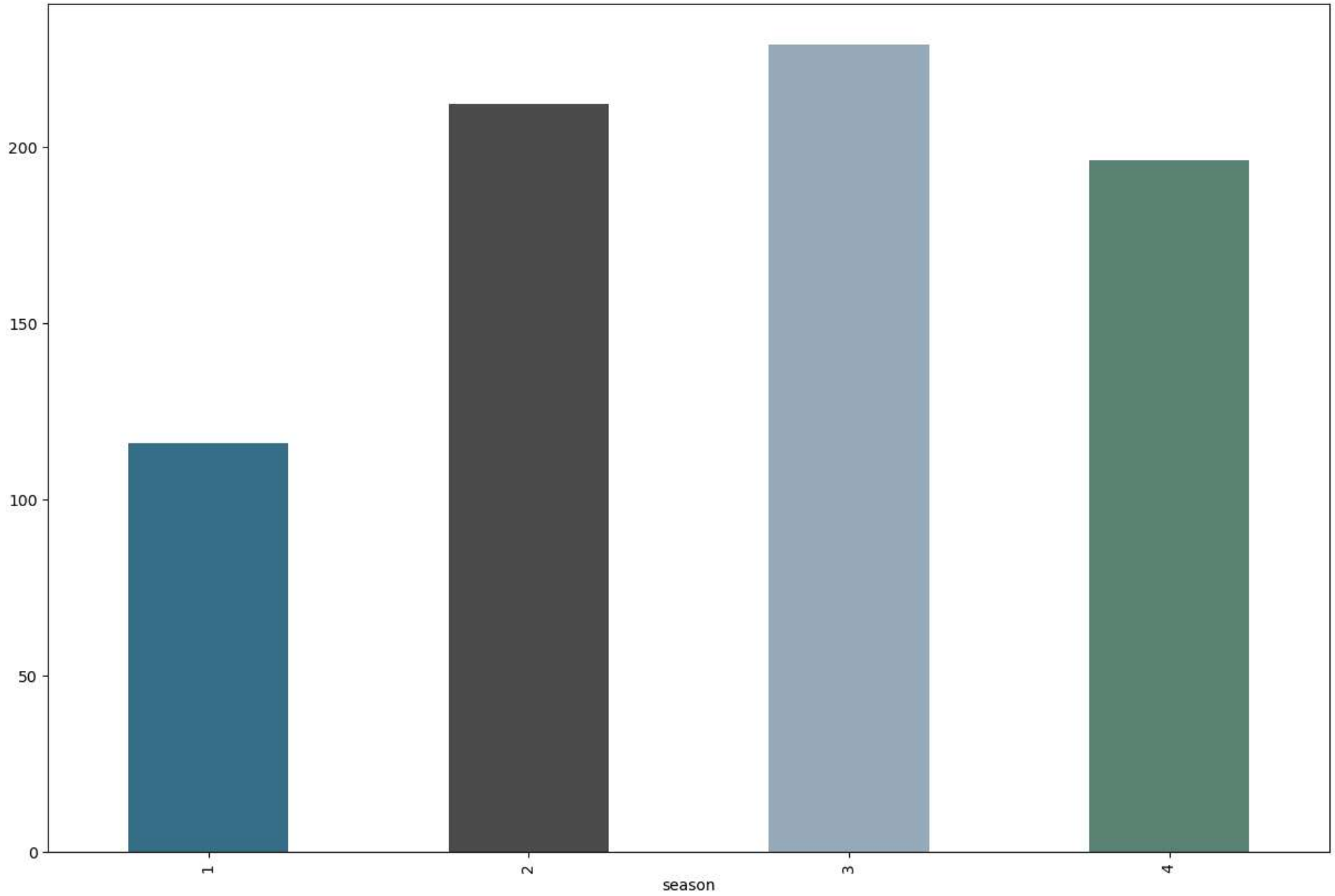
```
Variances are not equal
```

Comments: From the kruskal test it is found that we have atleast some of the weather condition have effect on the count of cycles rented

5. Check if the demand of bicycles on rent is the same for different Seasons?

In [45]:
```
yulu_df.groupby('season')['count'].mean().plot(kind='bar',color= ["#3A7089", "#4b4b4c",'#99AEBB','#5C8374','#6F7597'])
```

Out[45]:
```
<Axes: xlabel='season'>
```



a. Formulate Null Hypothesis (H0) and Alternate Hypothesis (H1)

H0=the demand of bicycles on rent is the same for different Seasons

Ha=the demand of bicycles on rent is not the same for different Seasons

In [46]:
```
season1=yulu_df[yulu_df['season']==1]['count']
season2=yulu_df[yulu_df['season']==2]['count']
season3=yulu_df[yulu_df['season']==3]['count']
season4=yulu_df[yulu_df['season']==4]['count']
```

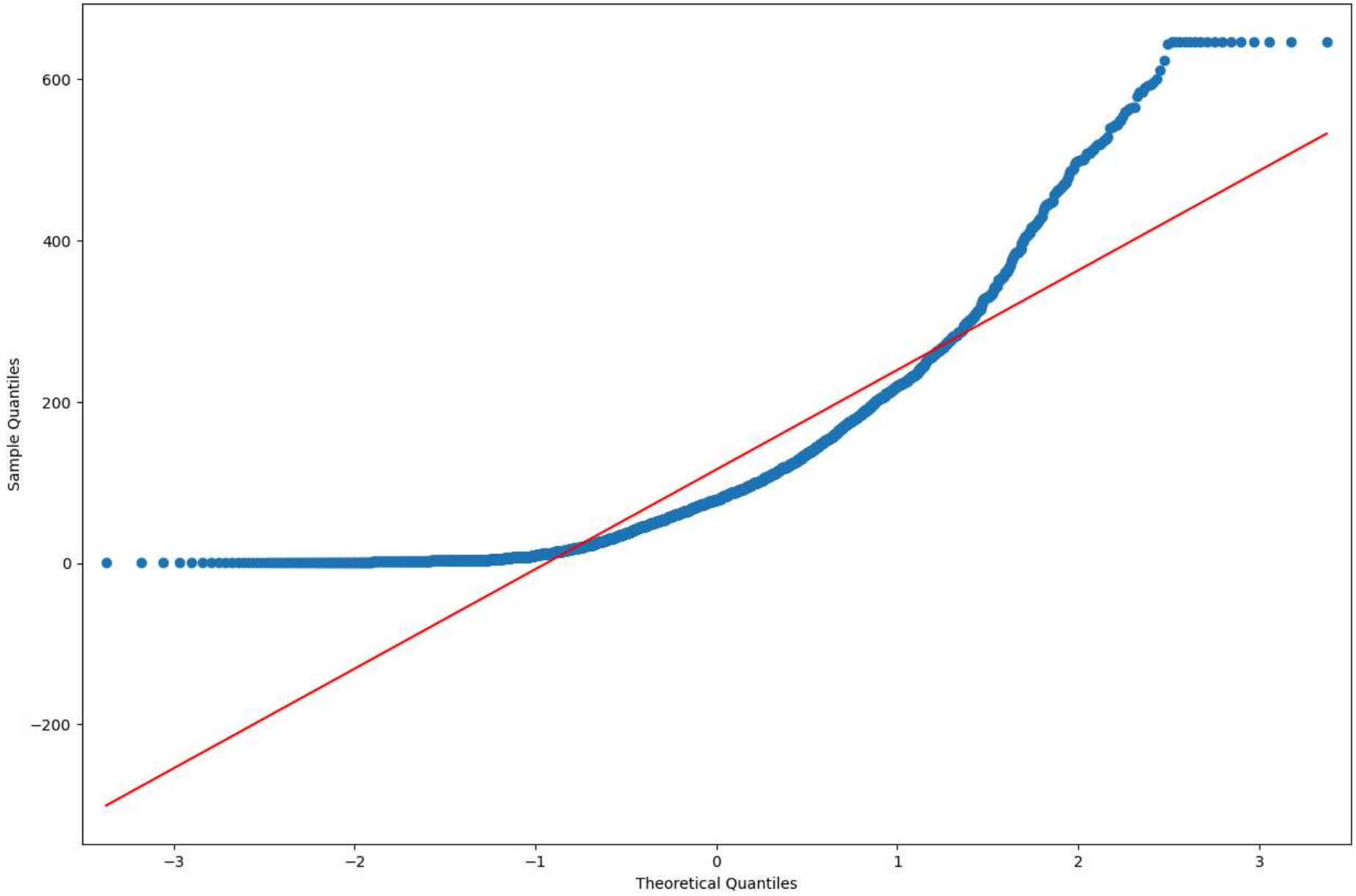
In [47]:
```
#b. Select an appropriate test -Since there are more than two categories of groups we go for anova test
f_oneway(season1,season2,season3,season4)
```

Out[47]:
```
F_onewayResult(statistic=243.33766355201303, pvalue=7.771506553957677e-153)
```

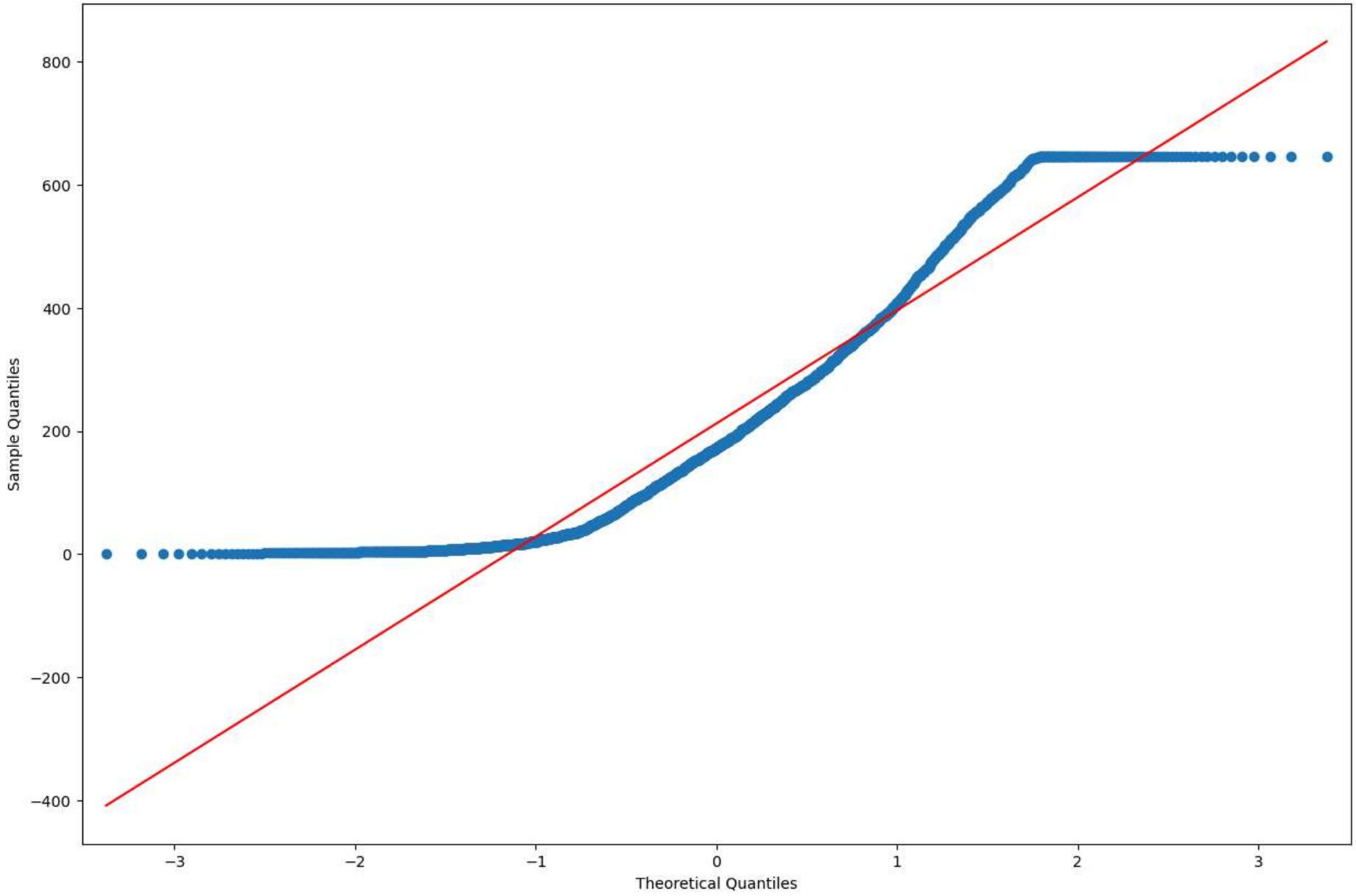
But in order to reley on one way anova test the dististribution should obey assumption

In [48]:
```
# for checking normality we use qqplot
plt.figure(figsize=(4,4))
qqplot(season1, line="s")
plt.show()
```

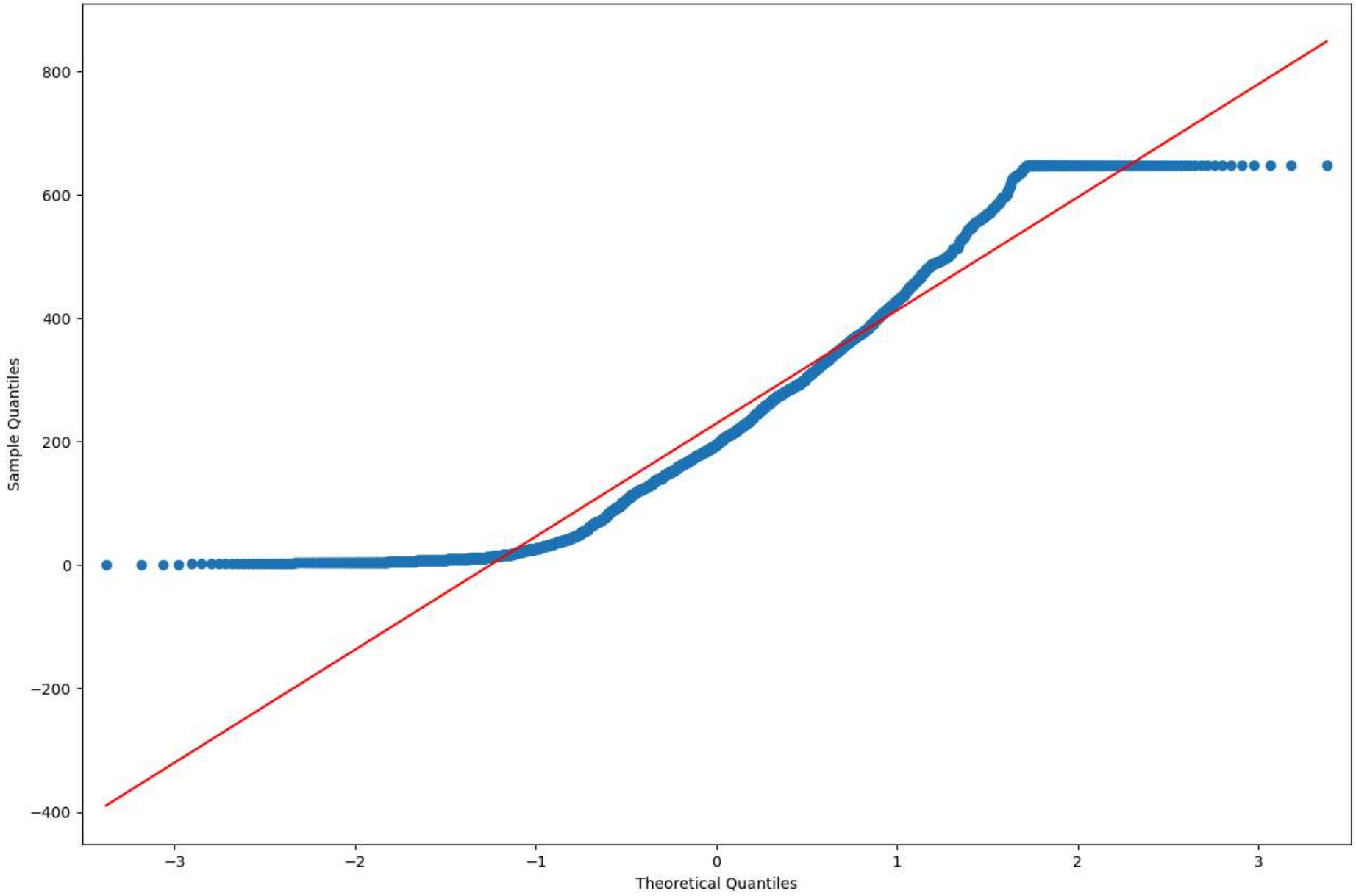
```
<Figure size 400x400 with 0 Axes>
```



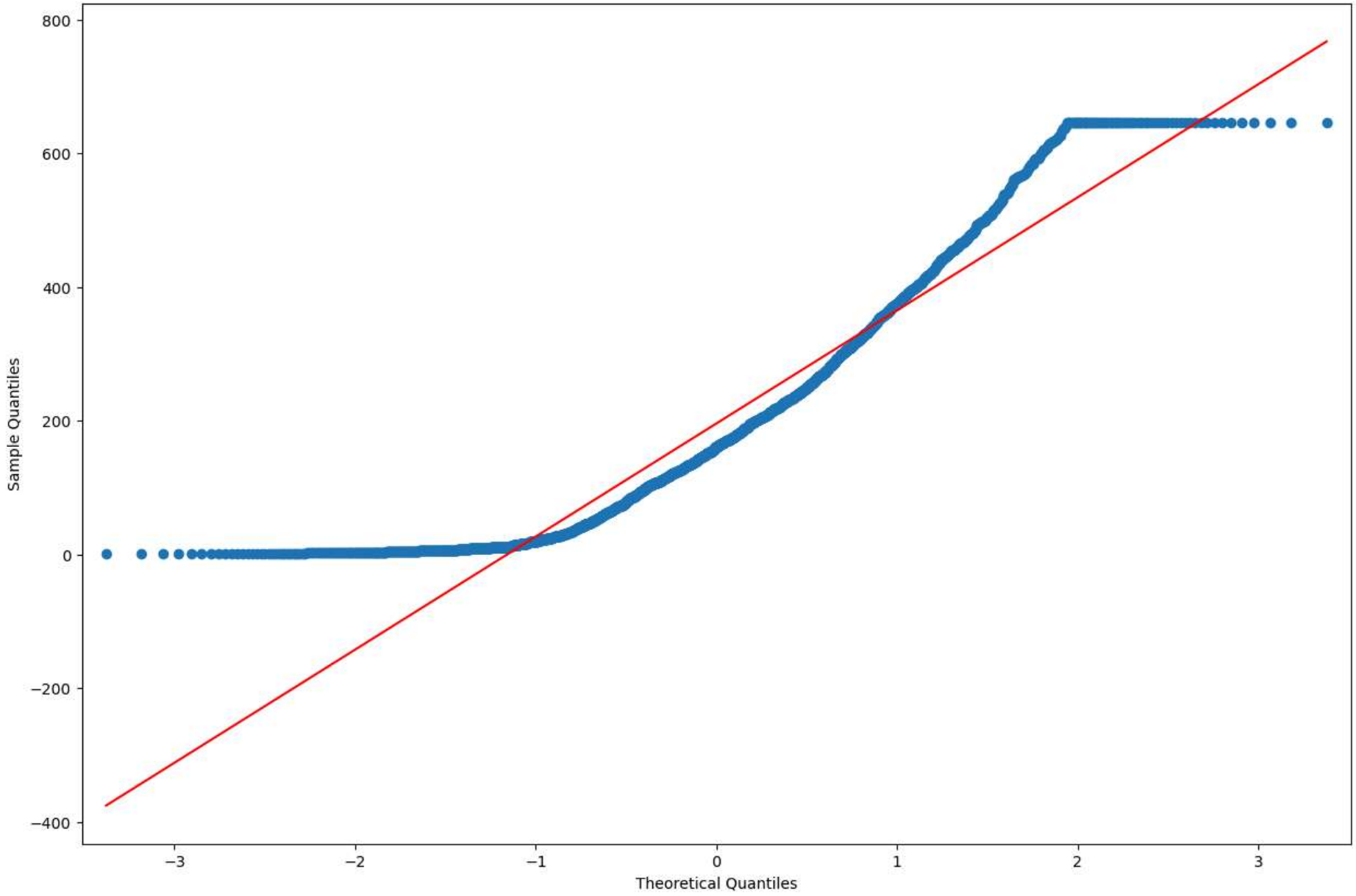
In [49]:
```
qqplot(season2, line="s")
plt.show()
```



In [50]:
```
qqplot(season3, line="s")
plt.show()
```



In [51]:
```
qqplot(season4, line="s")
plt.show()
```



Comments: Since the line is crooked or the dots are scattered, then your data doesn't quite fit and does not follow a Normal Distribution.

In [52]:
```
# For further analysis of NOrmality we use shapiro test
```

```
# H0: Data is Gaussian
# Ha: Data is not Gaussian

from scipy.stats import shapiro
test_stat, p_value=shapiro(season1)
test_stat, p_value
```

Out[52]:
```
(0.8147448301315308, 0.0)
```

In [53]:
```
if p_value < 0.05:
    print("Reject H0")
    print("Data is not Gaussian")
else:
    print("Fail to reject H0")
    print("Data is Gaussian")
```

```
Reject H0
Data is not Gaussian
```

In [54]:
```
test_stat, p_value=shapiro(season2)
test_stat, p_value
```

Out[54]:
```
(0.9027974009513855, 1.3308493050696584e-38)
```

In [55]:
```
if p_value < 0.05:
    print("Reject H0")
    print("Data is not Gaussian")
else:
    print("Fail to reject H0")
    print("Data is Gaussian")
```

```
Reject H0
Data is not Gaussian
```

In [56]:
```
test_stat, p_value=shapiro(season3)
test_stat, p_value
```

Out[56]:
```
(0.9245859384536743, 5.281442164890545e-35)
```

In [57]:
```
if p_value < 0.05:
    print("Reject H0")
    print("Data is not Gaussian")
else:
```

```
    print("Fail to reject H0")
    print("Data is Gaussian")
```

```
Reject H0
Data is not Gaussian
```

In [58]:
```
test_stat, p_value=shapiro(season4)
test_stat, p_value
```

Out[58]:
```
(0.9057514071464539, 3.6797419444712963e-38)
```

In [59]:
```
if p_value < 0.05:
    print("Reject H0")
    print("Data is not Gaussian")
else:
    print("Fail to reject H0")
    print("Data is Gaussian")
```

```
Reject H0
Data is not Gaussian
```

In [60]:
```
#since not following normal distribution
from scipy.stats import kruskal
stat, p_value =kruskal(season1,season2,season3,season4)
stat, p_value
```

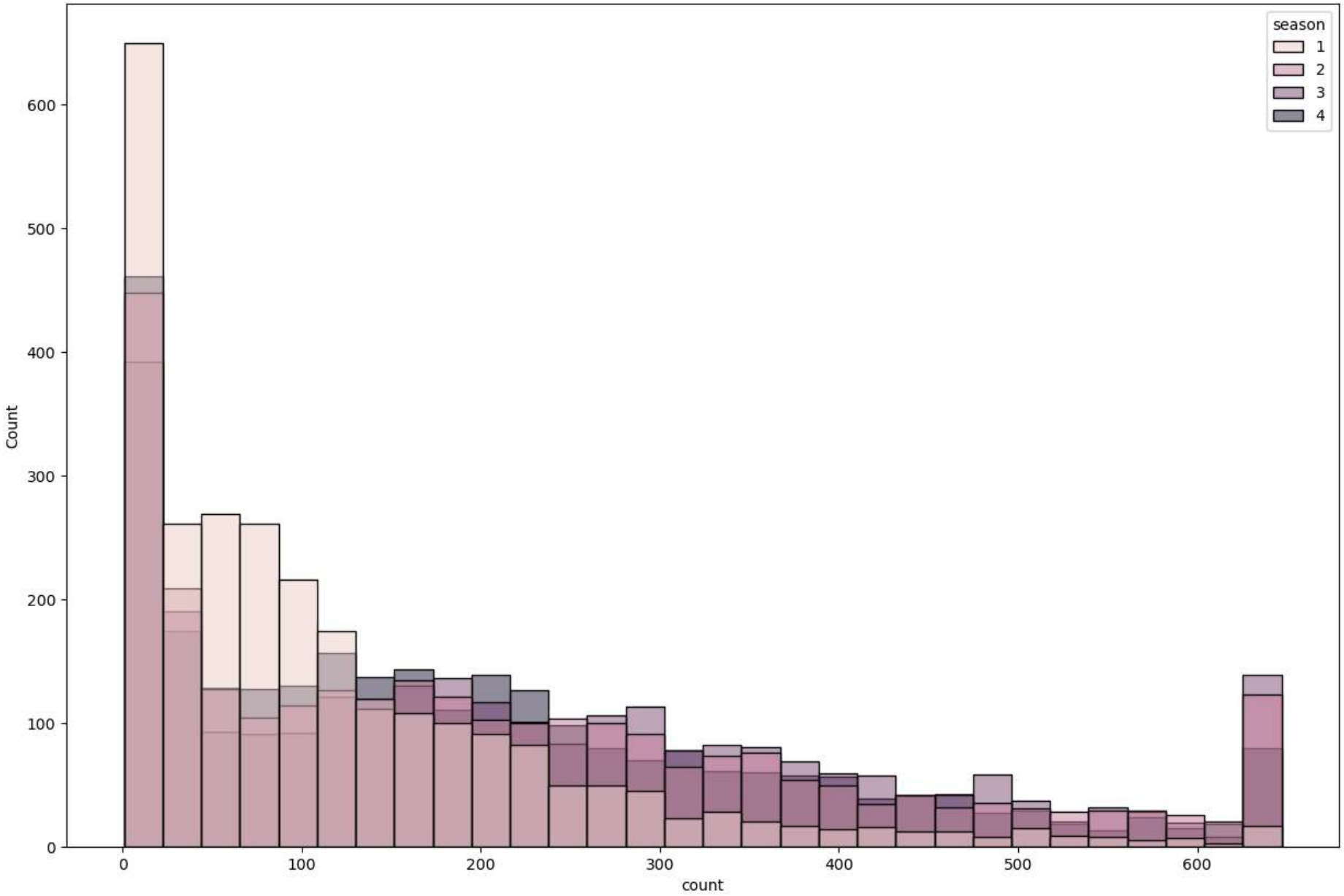
Out[60]:
```
(699.2817665514561, 3.0045514163996123e-151)
```

In [61]:
```
if p_value < 0.05:
    print("Reject H0")
    print("Atleast one group have different mean or some are dependent")
else:
    print("Fail to reject H0")
    print("All groups have same mean or they are independent")
```

```
Reject H0
Atleast one group have different mean or some are dependent
```

In [62]:
```
sn.histplot(data=yulu_df, x="count", hue='season')
```

Out[62]:
```
<Axes: xlabel='count', ylabel='Count'>
```



In [63]:
```
levene_stat, p_value = levene(season1,season2,season3,season4)
if p_value < 0.05:
    print("Variances are not equal")
```

```
Variances are not equal
```

Comments: From the kruskal test it is found that we have atleast some of the seasons have effect on the count of cycles rented

6. Check if the Weather conditions are significantly different during different Seasons?

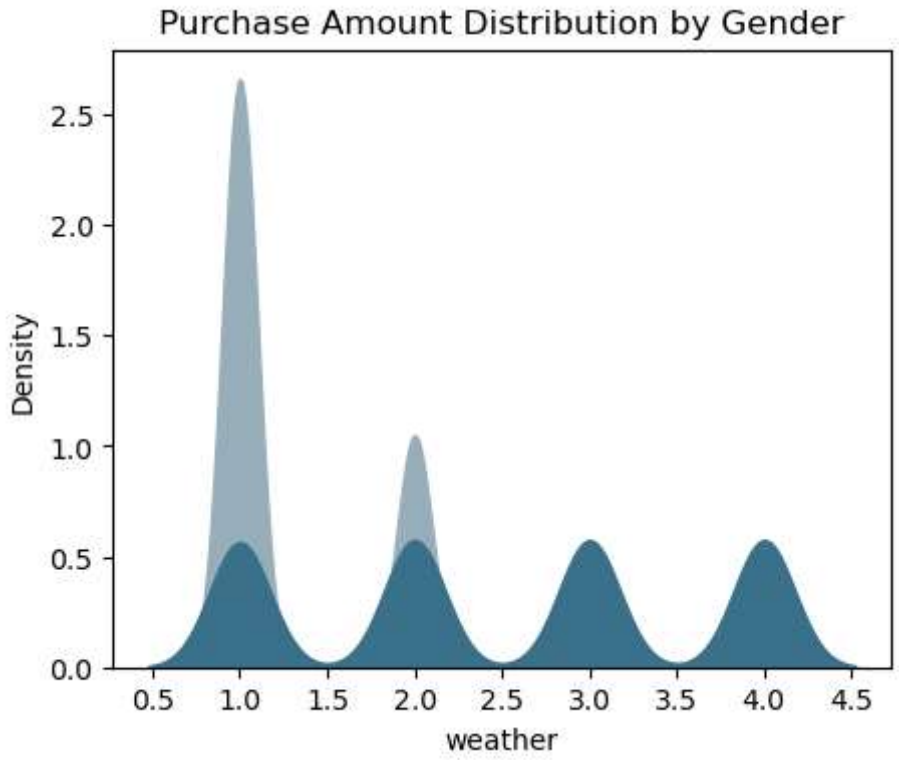
a. Formulate Null Hypothesis (H0) and Alternate Hypothesis (H1)

H0 : season and weather are independent

H1: season and weather are not independent

In [64]:
```
yulu_df['season'] = yulu_df['season'].astype('object')
yulu_df['weather'] = yulu_df['weather'].astype('object')
```

In [65]:
```
plt.figure(figsize=(5,4))
sn.kdeplot(data = yulu_df, x = 'weather', color = '#99AEBB',fill = True, alpha = 1)
sn.kdeplot(data = yulu_df, x = 'season', color = "#3A7089",fill = True, alpha = 1)
plt.title('Purchase Amount Distribution by Gender')
plt.show()
```



In [66]:
```
# Since we are looking for the relationship between two categorical variables we use chi-sqaure test
from scipy.stats import chi2_contingency
```

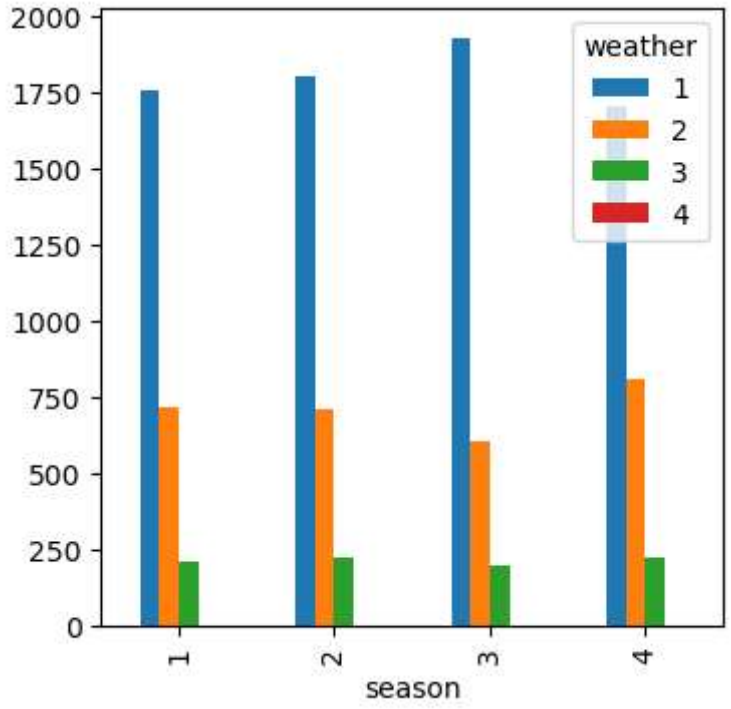
In [67]:
```
cross_tab = pd.crosstab(index=yulu_df['season'], columns=yulu_df['weather'])
cross_tab
```

Out[67]:

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 [68]:
```
cross_tab.plot(kind='bar',figsize=(4,4))
```

Out[68]:
```
<Axes: xlabel='season'>
```



In [69]:
```
obs=[[1759,715,211,1],[1801,708,224,0],[1930,604,199,0],[1702,807,225,0]]
chi_stat, p_value, df, exp_freq = chi2_contingency(obs)
chi_stat, p_value, df, exp_freq
```

Out[69]:
```
(49.158655596893624,
 1.549925073686492e-07,
 9,
 array([[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]]))
```

In [70]:
```
if p_value < 0.05:
    print("Reject H0")
    print("season and weather are not independent")
else:
```

```
    print("Fail to reject H0")
    print("season and weather are independent")
```

```
Reject H0
season and weather are not independent
```

Comments: According to the chi-square test conducted ,Its found that season and weather attributes are not independent.that is season and weather are dependent variables

In [ ]: