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.

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

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

Importing the required libraries and dataset

In [1]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
```

In [2]:
```
Yuludf = pd.read_csv(r"H:\Scaler\Hypothesis testing\Yulu\bike_sharing.csv")
```

In [3]:
```
Yuludf.head()
```

Out[3]:

	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 [4]:
```
Yuludf.tail()
```

Out[4]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
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

In [5]:
```
Yuludf.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 [6]:
```
Yuludf.shape
```

Out[6]: (10886, 12)

In [7]:
```
#The yulu dataset has 12 columns and 10886 rows.
#The columns named datetime is of object data type.
#The column named temp,atemp and windspeed is of float64 data type.
#The rest of the columns are of int64 data type
```

In [8]:
```
Yuludf.describe()
```

Out[8]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	co
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.886460	12.799395	36.021955	155.552177	191.574
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.245033	8.164537	49.960477	151.039033	181.144
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	0.000000	0.000000	0.000000	0.000000	1.000
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	47.000000	7.001500	4.000000	36.000000	42.000
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	62.000000	12.998000	17.000000	118.000000	145.000
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	77.000000	16.997900	49.000000	222.000000	284.000
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	100.000000	56.996900	367.000000	886.000000	977.000

In [9]:
```
Yuludf.isna().sum().sum()
```

Out[9]: 0

In [10]:
```
#There are no null values in the dataset
```

Checking for outliers

In [11]:
```
def find_outliers_IQR(df):
    q1 = df.quantile(0.25)
    q3 = df.quantile(0.75)
    IQR = q3 - q1
    outliers = df[((df<(q1-1.5*IQR)) | (df>(q3+1.5*IQR)))]
    return outliers
```

In [12]:
```
season_outliers = find_outliers_IQR(Yuludf['season'])

print('number of outliers: '+ str(len(season_outliers)))

print('max outlier value: '+ str(season_outliers.max()))

print('min outlier values: '+ str(season_outliers.min()))
```

```
number of outliers: 0
max outlier value: nan
min outlier values: nan
```

In [13]:
```
weather_outliers = find_outliers_IQR(Yuludf['weather'])

print('number of outliers: '+ str(len(weather_outliers)))

print('max outlier value: '+ str(weather_outliers.max()))

print('min outlier values: '+ str(weather_outliers.min()))
```

```
number of outliers: 1
max outlier value: 4
min outlier values: 4
```

In [14]:
```
temp_outliers = find_outliers_IQR(Yuludf['temp'])

print('number of outliers: '+ str(len(temp_outliers)))

print('max outlier value: '+ str(temp_outliers.max()))

print('min outlier values: '+ str(temp_outliers.min()))
```

```
number of outliers: 0
max outlier value: nan
min outlier values: nan
```

In [15]:
```
casual_outliers = find_outliers_IQR(Yuludf['casual'])

print('number of outliers: '+ str(len(casual_outliers)))

print('max outlier value: '+ str(casual_outliers.max()))

print('min outlier values: '+ str(casual_outliers.min()))
```

```
number of outliers: 749
max outlier value: 367
min outlier values: 117
```

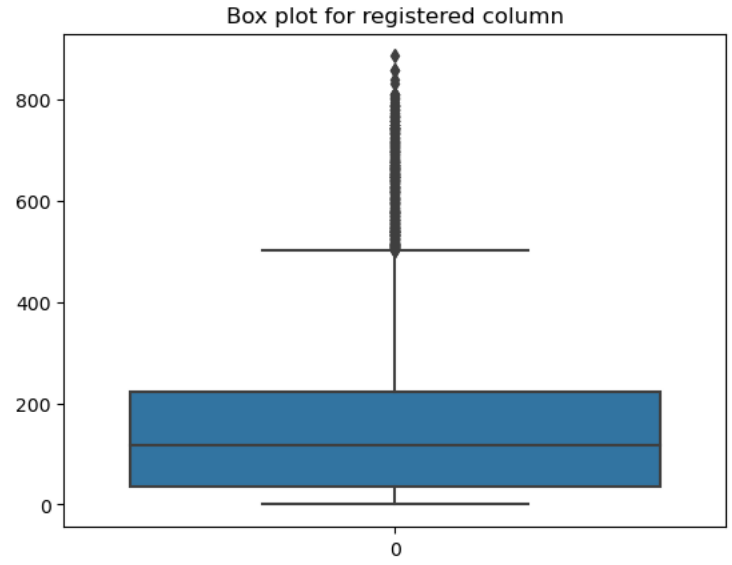
Outliers for the casual column are in between 117 and 367

In [16]:
```
registered_outliers = find_outliers_IQR(Yuludf['registered'])

print('number of outliers: '+ str(len(registered_outliers)))

print('max outlier value: '+ str(registered_outliers.max()))

print('min outlier values: '+ str(registered_outliers.min()))
```

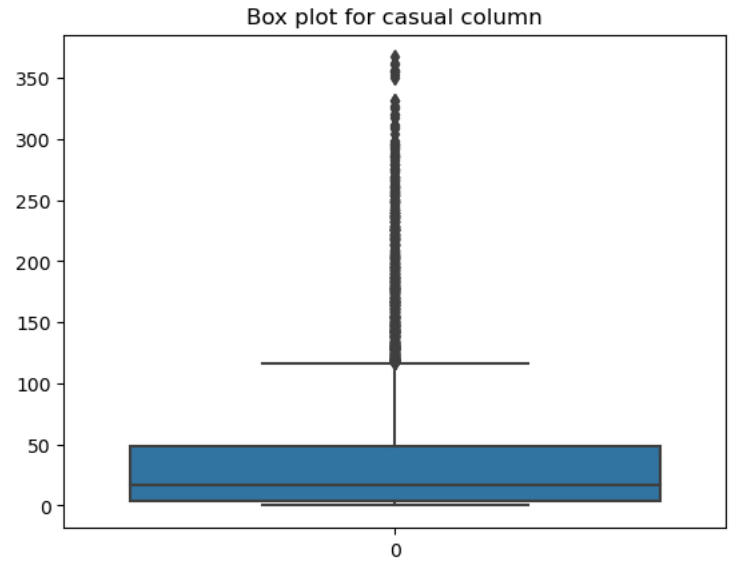
```
number of outliers: 423
max outlier value: 886
min outlier values: 502
```

Outliers for the registered column are in between 502 & 886

In [17]:
```
sns.boxplot(data=Yuludf['registered'])
plt.title("Box plot for registered column")
plt.show()
```



In [18]:
```
sns.boxplot(data=Yuludf['casual'])
plt.title("Box plot for casual column")
plt.show()
```

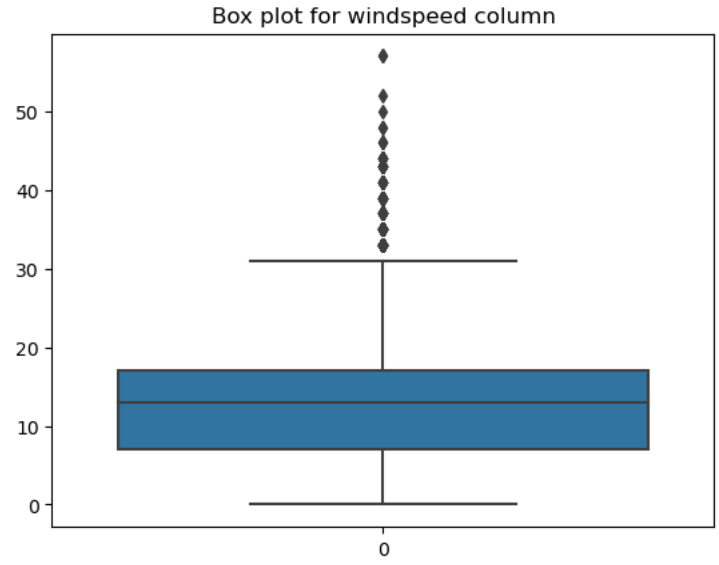


In [19]:
```
windspeed_outliers = find_outliers_IQR(Yuludf['windspeed'])

print('number of outliers: '+ str(len(windspeed_outliers)))

print('max outlier value: '+ str(windspeed_outliers.max()))

print('min outlier values: '+ str(windspeed_outliers.min()))
```

```
number of outliers: 227
max outlier value: 56.9969
min outlier values: 32.9975
```

Outliers for the windspeed column are in between 32.9 and 56.9

In [20]:
```
sns.boxplot(data=Yuludf['windspeed'])
plt.title("Box plot for windspeed column")
plt.show()
```



In [21]:
```
temp_outliers = find_outliers_IQR(Yuludf['temp'])

print('number of outliers: '+ str(len(temp_outliers)))

print('max outlier value: '+ str(temp_outliers.max()))

print('min outlier values: '+ str(temp_outliers.min()))
```

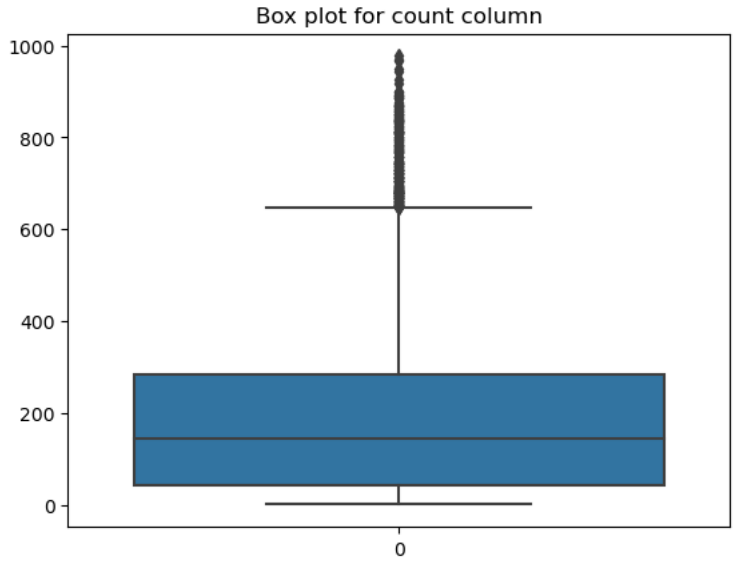
```
number of outliers: 0
max outlier value: nan
min outlier values: nan
```

In [22]:
```
count_outliers = find_outliers_IQR(Yuludf['count'])

print('number of outliers: '+ str(len(count_outliers)))

print('max outlier value: '+ str(count_outliers.max()))

print('min outlier values: '+ str(count_outliers.min()))
```

```
number of outliers: 300
max outlier value: 977
min outlier values: 648
```

Outliers for the count column are in between 648 and 977

In [23]:
```
sns.boxplot(data=Yuludf['count'])
plt.title("Box plot for count column")
plt.show()
```

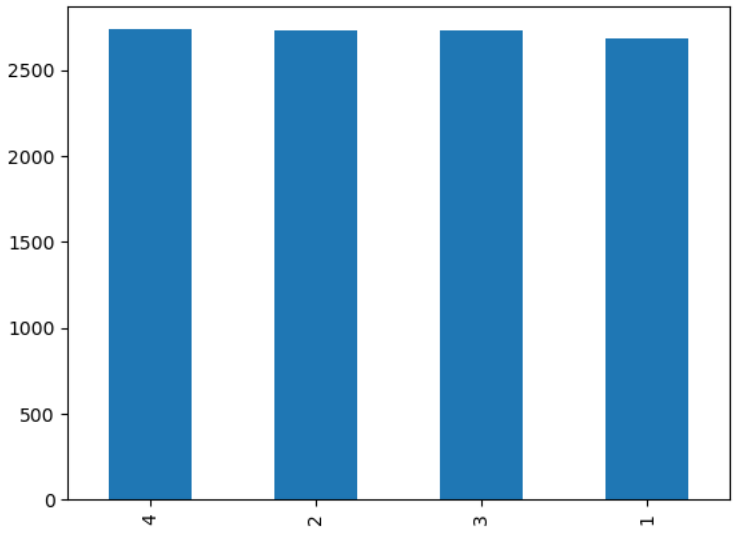


In [24]:
```
Yuludf.head()
```

Out[24]:

	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 [25]:
```
Yuludf['season'].value_counts().plot(kind='bar')
plt.show()
```



In [26]:
```
Yuludf['season'].value_counts()
```

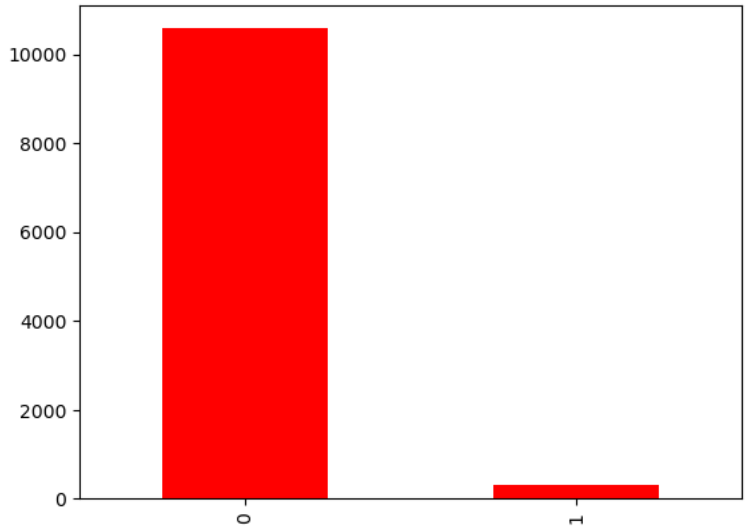
Out[26]:
```
4    2734
2    2733
3    2733
1    2686
Name: season, dtype: int64
```

The 4 seasons seem to be almost equally split

In [27]:
```
Yuludf['holiday'].value_counts()
```

Out[27]:
```
0    10575
1      311
Name: holiday, dtype: int64
```

In [28]:
```
Yuludf['holiday'].value_counts().plot(kind='bar',color='red')
plt.show()
```

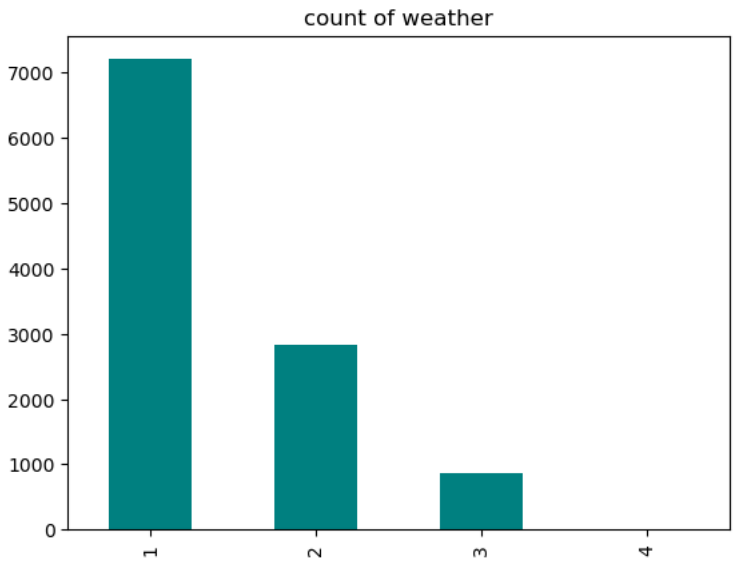


There seems to be higher count of 0 in comparison to the count 1 indicating that the number of holidays was 311 and then number of non holidays was 10575

In [29]:
```
Yuludf['weather'].value_counts()
```

Out[29]:
```
1    7192
2    2834
3     859
4       1
Name: weather, dtype: int64
```

In [30]:
```
Yuludf['weather'].value_counts().plot(kind='bar',color='teal')
plt.title('count of weather')
plt.show()
```



The highest count for the weather column is for the number 1

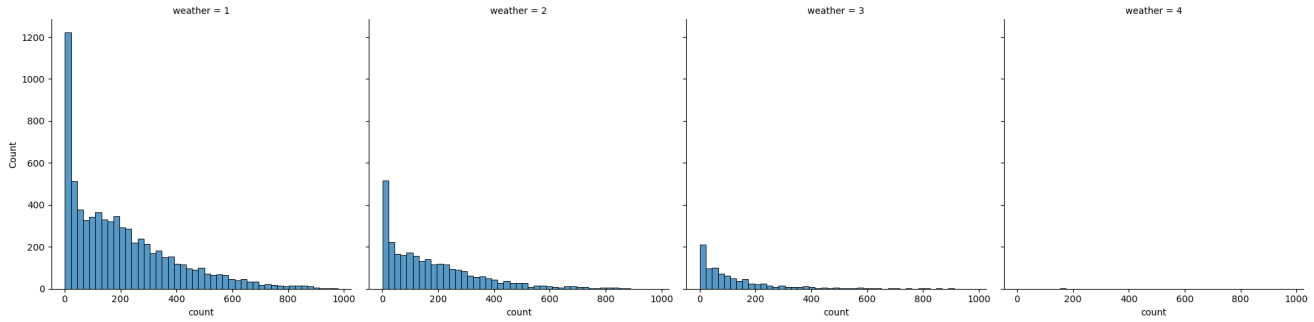
1: Clear, Few clouds, partly cloudy, partly cloudy

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

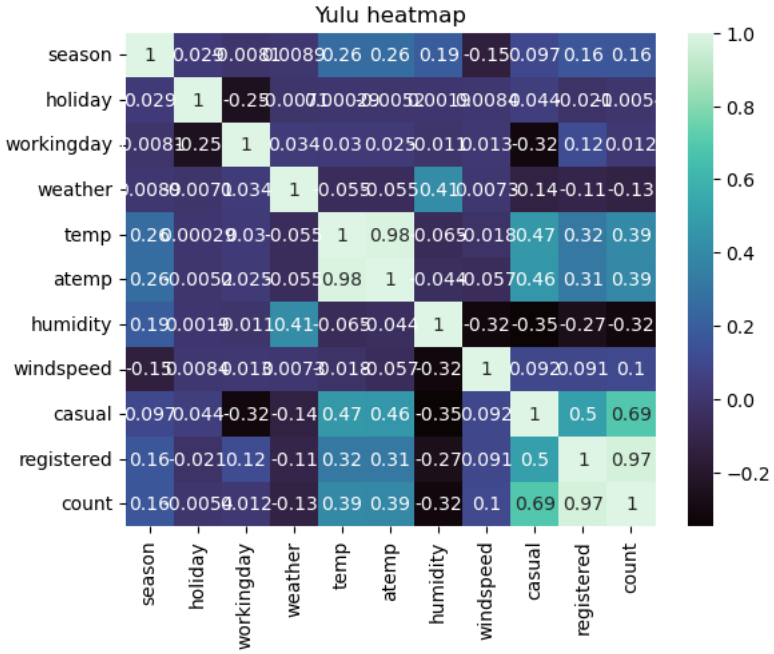
3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds

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

In [31]:
```
sns.displot(Yuludf, x="count", col="weather")
plt.show()
```

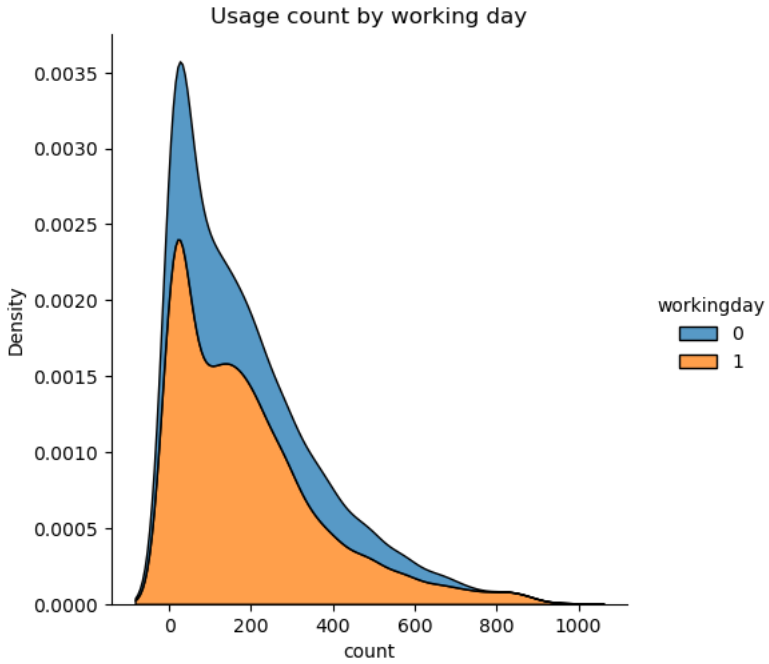


In [32]:
```
ax = sns.heatmap(Yuludf.corr(),annot=True,cmap='mako')
plt.title('Yulu heatmap')
plt.show()
```



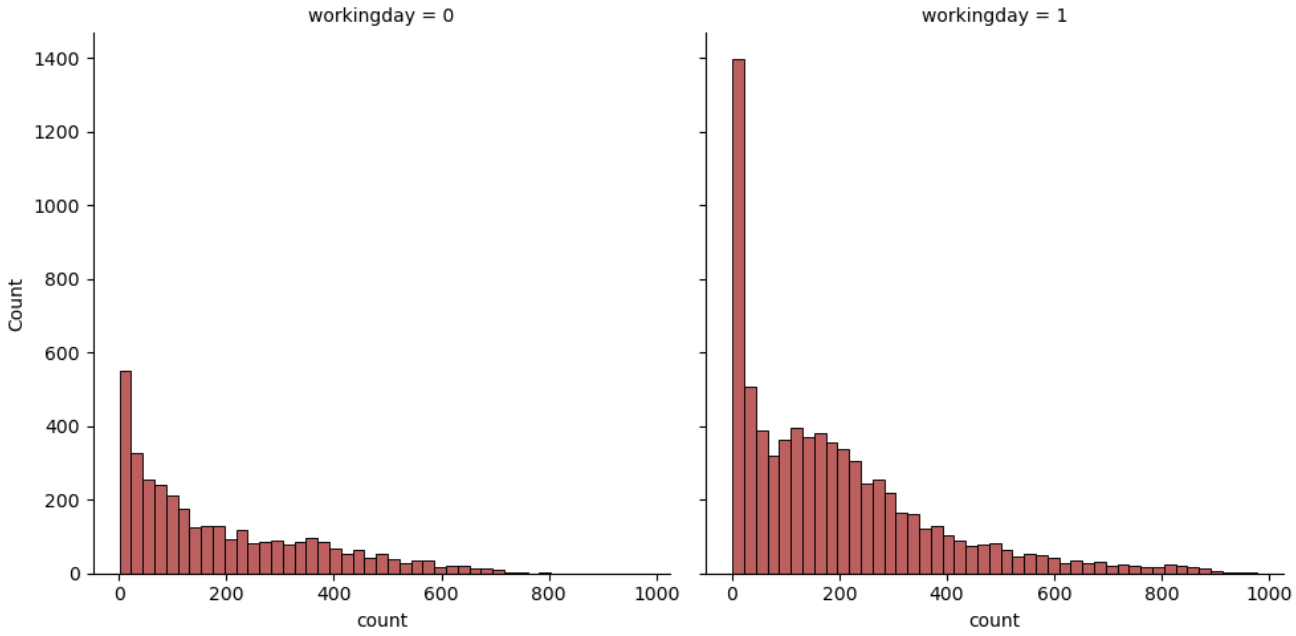
BIvariate analysis

In [33]:
```
sns.displot(Yuludf, x="count", hue="workingday", kind="kde", multiple="stack")
plt.title('Usage count by working day')
plt.show()
```



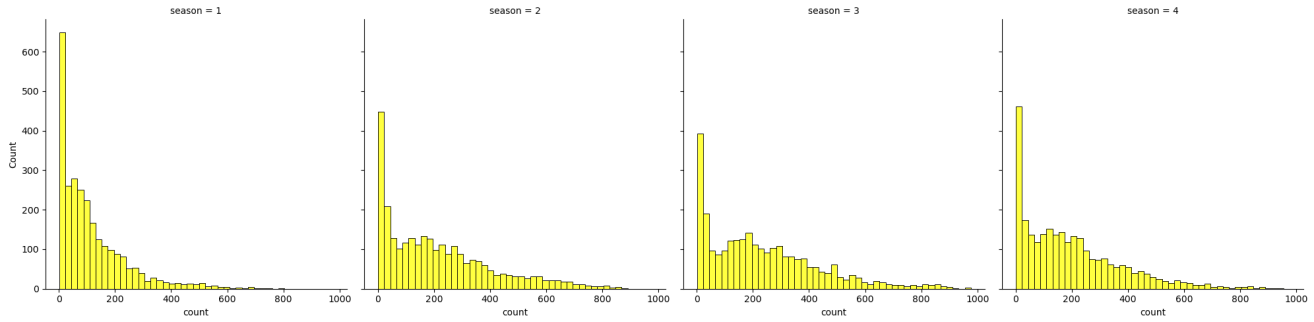
The count of cycle rented on a working day is much higher than the count on a non working day .

In [34]:
```
sns.displot(Yuludf, x="count", col="workingday",color='brown')
#plt.title('Product purchased by Gender')
plt.show()
```



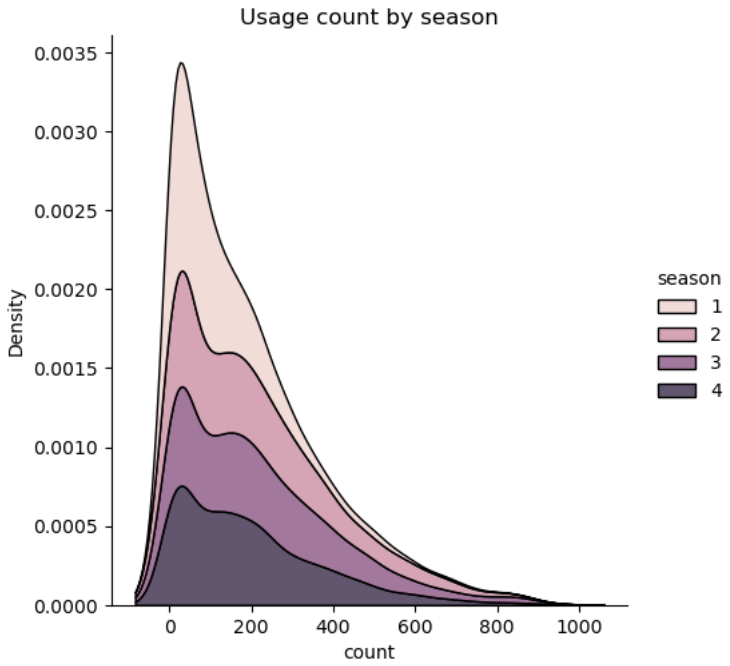
The plot above shows the difference between the count of cycles rented on a working vs non working day and it can be clearly seen that the count is higher on a working day.

In [35]:
```
sns.displot(Yuludf, x="count", col="season",color='yellow')
#plt.title('Product purchased by Gender')
plt.show()
```

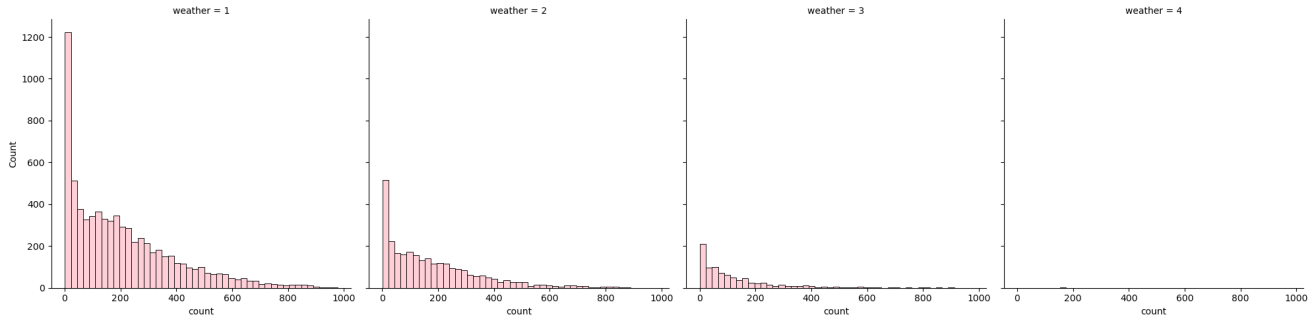


From the above graphs it can be seen that the count of cycles rented is higher on season 1 followed by season 2 , season 3 and then season 4

In [36]:
```
sns.displot(Yuludf, x="count", hue="season", kind="kde", multiple="stack")
plt.title('Usage count by season')
plt.show()
```



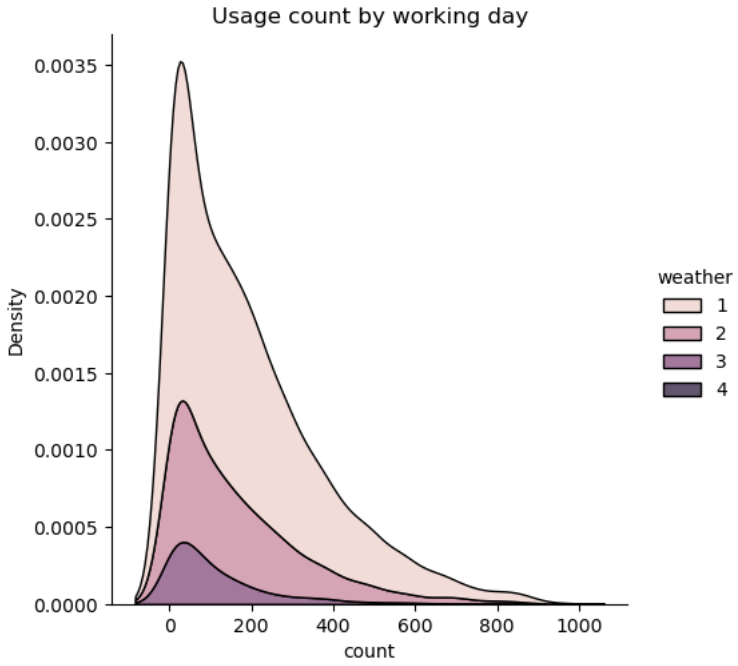
In [37]:
```
sns.displot(Yuludf, x="count", col="weather",color='pink')
#plt.title('Product purchased by Gender')
plt.show()
```



From the above plots it can be seen that the number of cycles rented is higher in weather 1 followed by weather 2 , weather 3 and then weather 4

In [38]:
```
sns.displot(Yuludf, x="count", hue="weather", kind="kde", multiple="stack")
plt.title('Usage count by working day')
plt.show()
```

```
C:\Users\india\AppData\Local\Temp\ipykernel_16872\2236272899.py:1: UserWarning: Dataset has 0 variance; skipping density estima
te. Pass `warn_singular=False` to disable this warning.
  sns.displot(Yuludf, x="count", hue="weather", kind="kde", multiple="stack")
```



In [39]:
```
Yuludf
```

Out[39]:

	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

2- Sample T-Test

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

In [40]:
```
Significance_level = 0.5
```

2 Sample t test to heck if working day has an effect on the count of cycles rented.

Here we assume that working day has no effect on the count of cycles rented.

Ho = There is no effect of working day on the count of cycles rented .

Ha = There is a effect on the count of cycles rented due to the working day .

In [41]:
```
working = Yuludf[Yuludf['workingday']==1]['count']
```

In [42]:
```
working
```

Out[42]:
```
47         5
48         2
49         1
50         3
51        30
         ...
10881    336
10882    241
10883    168
10884    129
10885     88
Name: count, Length: 7412, dtype: int64
```

In [43]:
```
notworking = Yuludf[Yuludf['workingday']==0]['count']
```

In [44]:
```
notworking
```

Out[44]:
```
0         16
1         40
2         32
3         13
4          1
         ...
10809    109
10810    122
10811    106
10812     89
10813     33
Name: count, Length: 3474, dtype: int64
```

In [45]:
```
from scipy.stats import ttest_ind
```

In [46]:
```
x,pvalue = ttest_ind(working,notworking,alternative='greater')
print(pvalue)
```

```
0.11322402113180674
```

In [47]:
```
if pvalue<Significance_level:
    print("Reject null hypothesis")
else:
    print("Failed to reject null hypothesis")
```

```
Reject null hypothesis
```

Working day has a effect on the number of cycles rented.

ANNOVA TEST

1)ANNOVA to check if No. of cycles rented is similar or different in different weather

Here we assume that the number of ccles rented is similar in different weather conditions

Ho= Number of cycles rented is similar in different weather conditions

Ha = Number of cycles rented is different in different weather conditions

In [48]:
```
sample1 = Yuludf[Yuludf['weather']==1]['count']
sample2 = Yuludf[Yuludf['weather']==2]['count']
sample3 = Yuludf[Yuludf['weather']==3]['count']
sample4 = Yuludf[Yuludf['weather']==4]['count']
```

In [49]:
```
from scipy.stats import f_oneway
```

In [50]:
```
x_1,pvalue_1 = f_oneway(sample1,sample2,sample3,sample4)
```

In [51]:
```
print(pvalue_1)
```

```
5.482069475935669e-42
```

In [52]:
```
if pvalue_1<Significance_level:
    print("Reject null hypothesis")
else:
    print("Failed to reject null hypothesis")
```

```
Reject null hypothesis
```

Since null hypothesis is rejected the number of cycles rented is different in different weather conditions

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

Here we assume that the number of cycles rented is similar in different seasons

Ho = Number of cycles is similar in different seasons

Ha = Number of cycles is not similar in different seasons

In [53]:
```
sample5 = Yuludf[Yuludf['season']==1]['count']
sample6 = Yuludf[Yuludf['season']==2]['count']
sample7 = Yuludf[Yuludf['season']==3]['count']
sample8 = Yuludf[Yuludf['season']==4]['count']
```

In [54]:
```
x_2,pvalue_2 =  f_oneway(sample5,sample6,sample7,sample8)
```

In [55]:
```
print(pvalue_2)
```

```
6.164843386499654e-149
```

In [57]:
```
if pvalue_2<Significance_level:
        print("Reject null hypothesis")
    else:
        print("Failed to reject null hypothesis")
```

```
  File <tokenize>:3
    else:
    ^
IndentationError: unindent does not match any outer indentation level
```

Since the null hypthesis is rejected the number of cycles rented is not similar in different seasons

Chi-square test

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

Here we assume that the weather is dependent on the season

Ho = Weather is dependent on season

Ha = Weather is not dependent on season

In [ ]:
```
from scipy import stats
from scipy.stats import chi2_contingency
from scipy.stats import chi2
from scipy.stats import chisquare
```

In [ ]:
```
contingency = pd.crosstab(Yuludf['weather'],Yuludf['season'])
print(contingency)
```

In [ ]:
```
p_value3 = chi2_contingency(contingency)[1]
print(p_value)
```

In [ ]:
```
if p_value3<Significance_level:
    print("Reject null hypothesis")
else:
    print("Failed to reject null hypothesis")
```

Since null hypothesis is rejected the weather is not dependent on the season

CONCLUSION

The count of cycle rented on a working day is much higher than the count on a non working day . This suggests that more people are in need of cycles during their working day which could be needed for them to commute to work . Increasing the availability of cycles on a working day will increase the profit

Usage count by season Checking the usage count by season it can be seen that season 1 has the highest count.

season1: spring

season2: summer

season3: fall

season4: winter

Spring season has the highest count of cycles rented followed by summer and fall and finally winter .

Increasing marketing and advertisement and also the number of cycles available during spring will greatly benefit the company and increases the profit.

Usage count by weather weather:

weather1: Clear, Few clouds, partly cloudy, partly cloudy

weather2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist

weather3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds

weather4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

Checking the usage count by weather it can be seen that weather 1 has the highest count of cycles being rented. Weather indicates a clear weather with few clouds and this is clearly a preferrable time for using a cycle . Weather 2 has the next highest count of cycles being rented followed by weather 3 . Weahter 4 has almost no cycles being rented and since weather 4 indicates heavy rain , ice pallets and thunderstorm it can be clearly understood that people do not want to commute by a cycle during such weather.

Predicting the weather and placing the cycles and managing the count of cycles appropriately will be a good startegy to have a good profit and performance for the company . It is best to not have any cycles for rent during weather 4 .

2- Sample T-Test to check if Working Day has an effect on the number of electric cycles rented. By performing the 2 sample t test it can be clearly seen that the working day has an effect on the number of cycles being rented. By understanding the public and general holidays in a year it will be a good startey to have the count of cycles appropriately stacked. More cycles are required during the working day and less during non working day .

ANNOVA test to check if the number of cycles rented is similar in different weather conditions. By using the ANNOVA test it can be seen that the number of cycles rented is not similar in different weather conditions. Hence it is a good stategy to plan the availability of cycles as per the predicted weather conditions.

Chi-square test to check if Weather is dependent on the season From chi square test it can be seen that the weather is not dependent on season . Therefore using the information of predicted weather to stack the number of cycles for rent would be more appropriate strategy than to use seasons as weather can change within a short interval of time.

END OF ANALYSIS

In [ ]: