Problem statement:

1. Which variables are significant in predicting the demand for shared electric cycles in the Indian market?
2. How well those variables describe the electric cycle demands

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import ttest_ind,f_oneway,chi2,chisquare,chi2_contingency,ttest_rel
```

In [2]:
```
df   =   pd.read_csv("https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/original/bike_sharing.cs
```

In [3]:
```
df.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]:
```
# split datetime column to date and time

df[['date','time']] = df['datetime'].str.split(expand=True)
```

In [5]:
```
del df['datetime']
```

In [6]:
```
df['date'] = pd.to_datetime(df['date'])
```

In [7]:
```
df['year'] = df['date'].dt.year.astype('object')
df['month'] = df['date'].dt.month.astype('object')
```

In [163…
```
df['month-wise'] = df['month'].astype('str') + '-' + df['year'].astype('str')
```

In [164…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   season      10886 non-null  category
 1   holiday     10886 non-null  category
 2   workingday  10886 non-null  category
 3   weather     10886 non-null  category
 4   temp        10886 non-null  float64
 5   atemp       10886 non-null  float64
 6   humidity    10886 non-null  int64
 7   windspeed   10886 non-null  float64
 8   casual      10886 non-null  int64
 9   registered  10886 non-null  int64
 10  count       10886 non-null  int64
 11  date        10886 non-null  datetime64[ns]
 12  time        10886 non-null  object
 13  year        10886 non-null  object
 14  month       10886 non-null  object
 15  month-wise  10886 non-null  object
dtypes: category(4), datetime64[ns](1), float64(3), int64(4), object(4)
memory usage: 1.0+ MB
```

Characteristics of the data

In [9]:
```
df.shape
```

Out[9]:
```
(10886, 16)
```

In [10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   season      10886 non-null  int64
 1   holiday     10886 non-null  int64
 2   workingday  10886 non-null  int64
 3   weather     10886 non-null  int64
 4   temp        10886 non-null  float64
 5   atemp       10886 non-null  float64
 6   humidity    10886 non-null  int64
 7   windspeed   10886 non-null  float64
 8   casual      10886 non-null  int64
 9   registered  10886 non-null  int64
 10  count       10886 non-null  int64
 11  date        10886 non-null  datetime64[ns]
 12  time        10886 non-null  object
 13  year        10886 non-null  object
 14  month       10886 non-null  object
 15  month-wise  10886 non-null  object
dtypes: datetime64[ns](1), float64(3), int64(8), object(4)
memory usage: 1.3+ MB
```

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

Out[11]:
```
season        0
holiday       0
workingday    0
weather       0
temp          0
atemp         0
humidity      0
windspeed     0
casual        0
registered    0
count         0
date          0
time          0
year          0
month         0
month-wise    0
dtype: int64
```

In [68]:
```
# there are no null values
# convert datetime column to datetime datatype
# the dataframe has 10886 rows and 12 columns
```

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

Out[13]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casu
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000	10886.000000	10886.000000	10886.0000
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.886460	12.799395	36.0219
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.245033	8.164537	49.9604
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	0.000000	0.000000	0.0000
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	47.000000	7.001500	4.0000
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	62.000000	12.998000	17.0000
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	77.000000	16.997900	49.0000
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	100.000000	56.996900	367.0000

In [14]:
```
for i in df.columns:
    print(f'{i} : {df[i].nunique()}')
```

```
season : 4
holiday : 2
workingday : 2
weather : 4
temp : 49
atemp : 60
humidity : 89
windspeed : 28
casual : 309
registered : 731
count : 822
date : 456
time : 24
year : 2
month : 12
month-wise : 13
```

In [15]:
```
# categories: season, holiday, workingday, weather
```

In [16]:
```
# converting these four columns to categories

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

In [17]:
```
df.dtypes
```

Out[17]:
```
season              category
holiday             category
workingday          category
weather             category
temp                 float64
atemp                float64
humidity               int64
windspeed            float64
casual                 int64
registered             int64
count                  int64
date          datetime64[ns]
time                  object
year                  object
month                 object
month-wise            object
dtype: object
```

In [76]:
```
for i in ['season','holiday','workingday','weather']:
    print()
    print(df[i].value_counts())
    print('----------------------------------------')
```

```

4    2734
2    2733
3    2733
1    2686
Name: season, dtype: int64
----------------------------------------

0    10575
1      311
Name: holiday, dtype: int64
----------------------------------------

1    7412
0    3474
Name: workingday, dtype: int64
----------------------------------------

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

In [22]:
```
print('start date of the data:', df['date'].min())
print('end date of the data:',df['date'].max())
```

```
start date of the data: 2011-01-01 00:00:00
end date of the data: 2012-12-19 00:00:00
```

Univariate Analysis

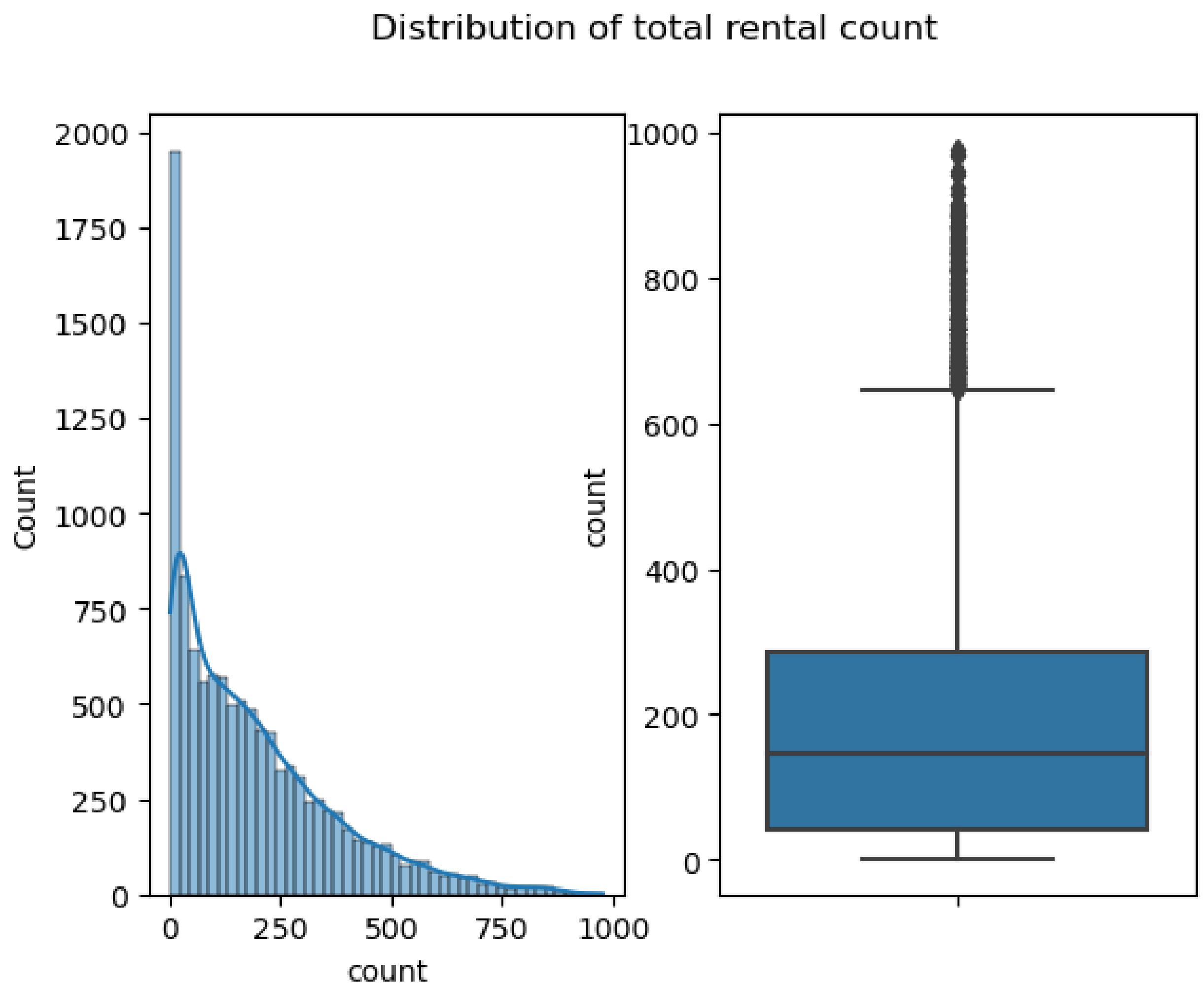
In [23]:
```
df.head()
```

Out[23]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	date	time	year	month
0	1	0	0	1	9.84	14.395	81	0.0	3	13	16	2011-01-01	00:00:00	2011	1
1	1	0	0	1	9.02	13.635	80	0.0	8	32	40	2011-01-01	01:00:00	2011	1
2	1	0	0	1	9.02	13.635	80	0.0	5	27	32	2011-01-01	02:00:00	2011	1
3	1	0	0	1	9.84	14.395	75	0.0	3	10	13	2011-01-01	03:00:00	2011	1
4	1	0	0	1	9.84	14.395	75	0.0	0	1	1	2011-01-01	04:00:00	2011	1

In [212…
```
plt.subplot(121)
sns.histplot(df['count'],kde=True)
plt.suptitle("Distribution of total rental count")

plt.subplot(122)
sns.boxplot(data = df,y = 'count')
plt.show()
```

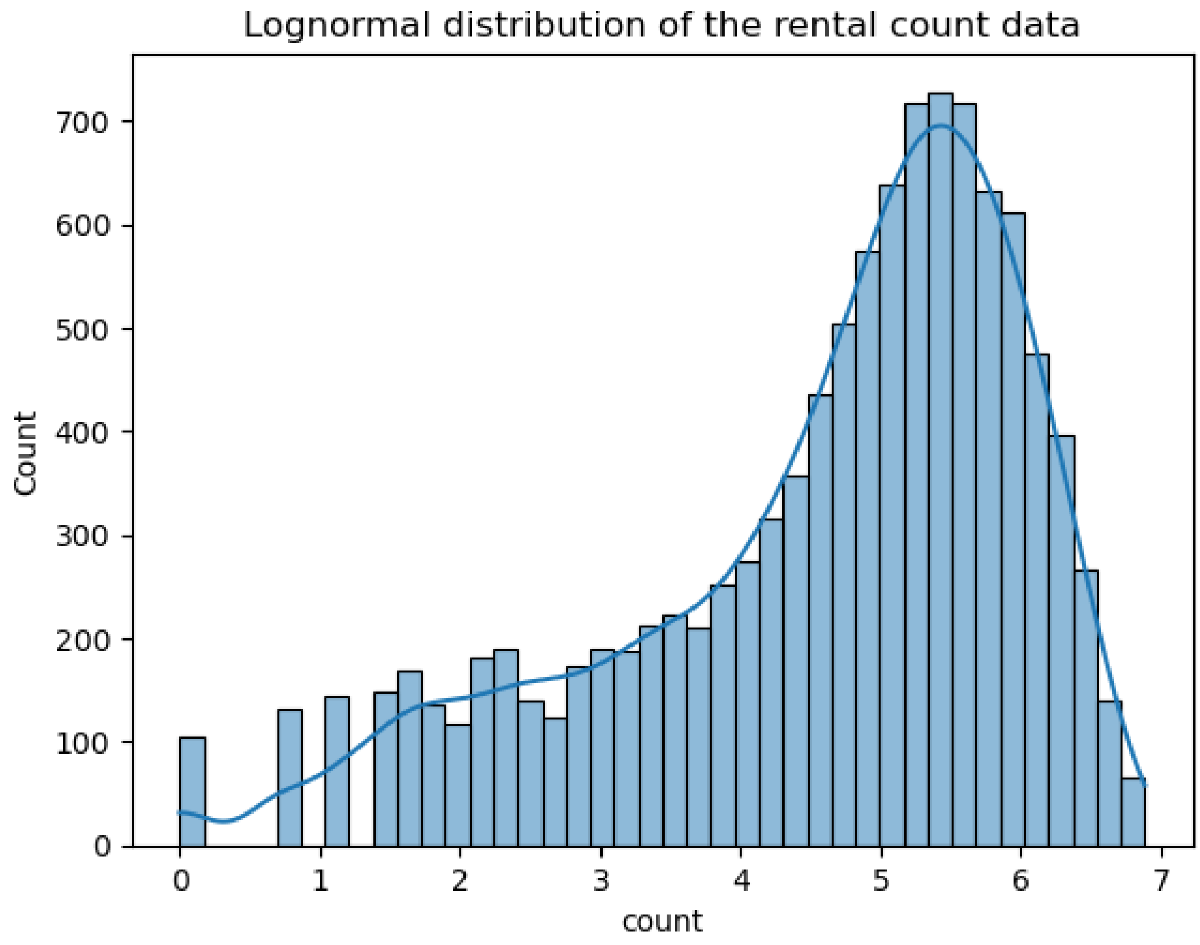


Insights:

1. We can observe that most of the data lies between 0 to 650, which means that 0-650 cycles are rented each hour.
2. It is visible that most data lies close to 0 and this skewness is evident in both histogram and the boxplot.

We can apply logarithmic transformation in an attempt to normalise the data.

In [186…
```
sns.histplot(np.log(df['count']),kde=True)
plt.title("Lognormal distribution of the rental count data")
plt.show()
```



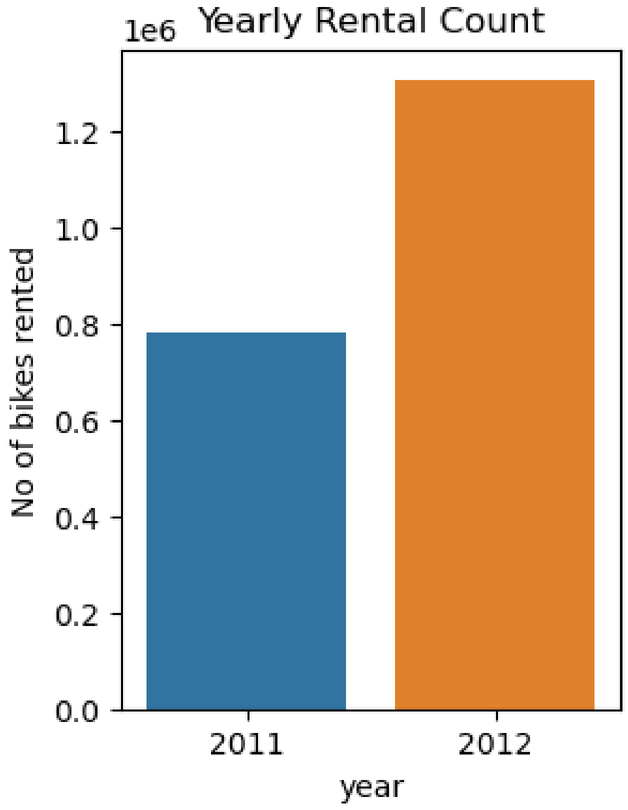
Even after applying logarithmic transormation on the data, we can see from the histplot that the distribution of the population is not normal.

In [210…

```
yearly = df.groupby(['year'])['count'].sum().reset_index()
print("Percentage increase in rentals from 2011 to 2012: ",end='')
print(f'{np.round((yearly["count"][1]   -   yearly["count"][0])/(yearly["count"][0]),2)*100}%')

fig = plt.figure(figsize=(3,4))
sns.barplot(data = df.groupby(['year'])['count'].sum().reset_index(),x = 'year',y='count')
plt.title("Yearly Rental Count")
plt.ylabel('No of bikes rented')
plt.show()
```

```
Percentage increase in rentals from 2011 to 2012: 67.0%
```

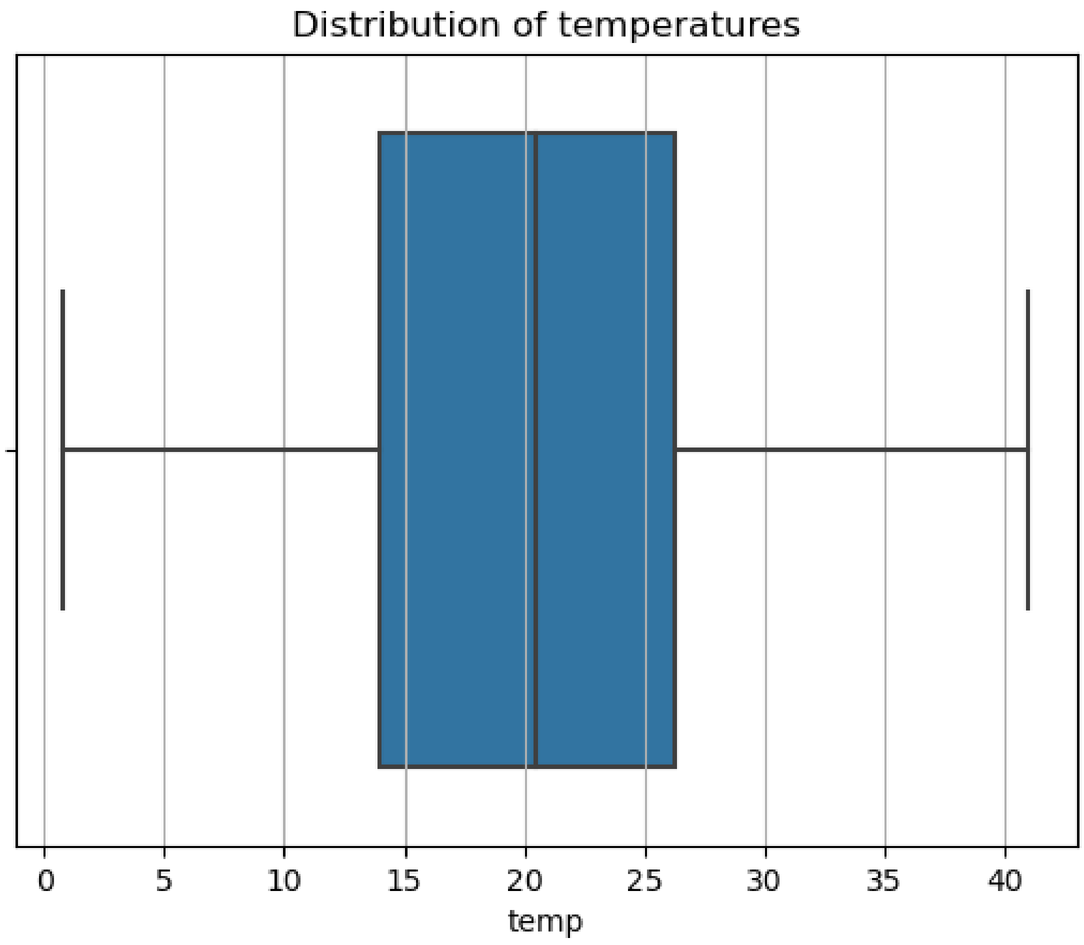


In [190…

```
min_temp = np.min(df['temp'])
max_temp = np.max(df['temp'])
print(f'Min and max values of temperature from the data are {min_temp} Celsius and {max_temp} Celsius')

plt.grid()
sns.boxplot(data=df,x = 'temp')
plt.title('Distribution of temperatures')
plt.show()
```

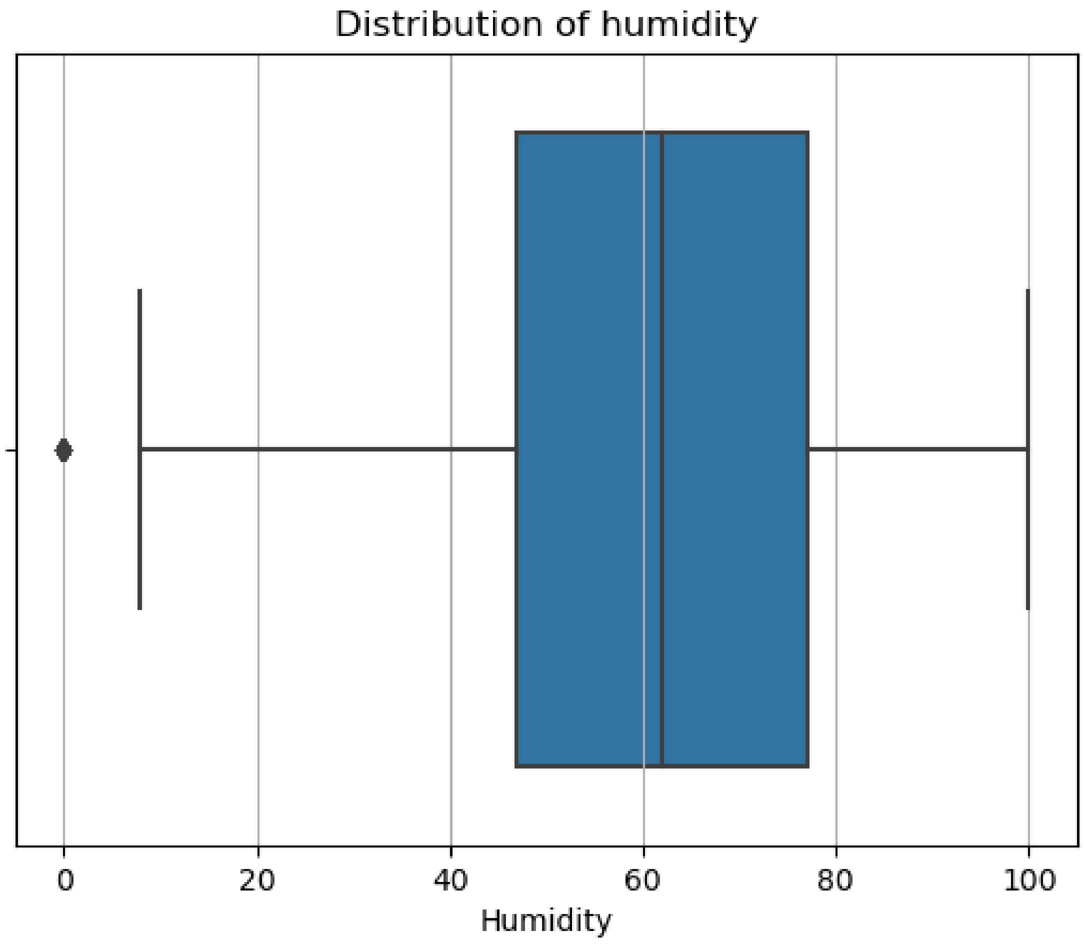
```
Min and max values of temperature from the data are 0.82 Celsius and 41.0 Celsius
```



In [196…
```
min_humidity = np.min(df['humidity'])
max_humidity = np.max(df['humidity'])
print(f'Min and max values of humidity from the data are {min_humidity} and {max_humidity}')

plt.grid()
sns.boxplot(data=df,x = 'humidity')
plt.title('Distribution of humidity')
plt.xlabel('Humidity')
plt.show()
```

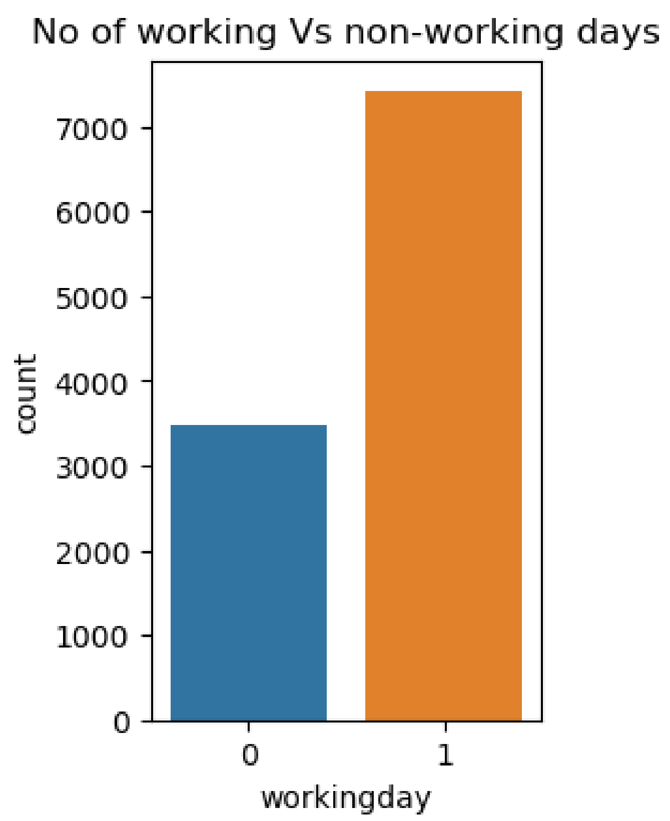
```
Min and max values of humidity from the data are 0 and 100
```

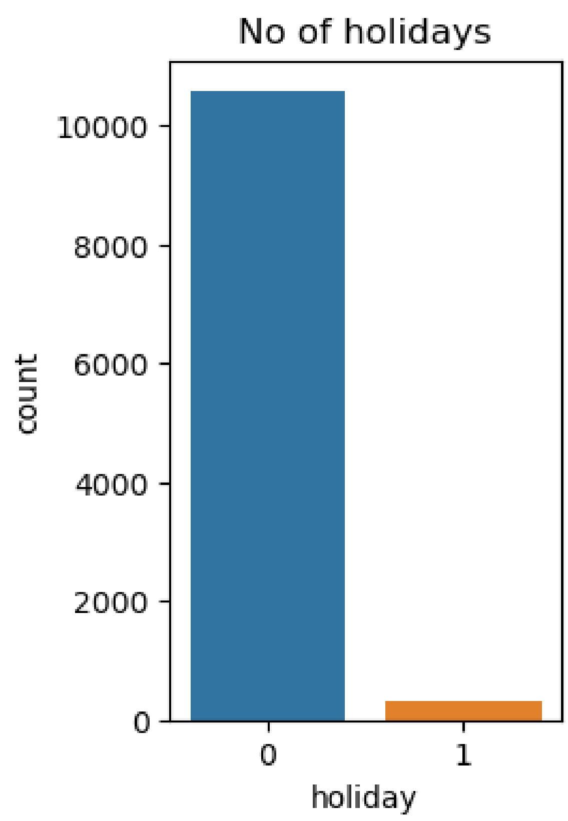


In [276…
```
# count of workingday and holidays

plt.figure(figsize=(8,4))
plt.subplot(1,3,1)
sns.countplot(data = df,x = 'workingday')
plt.title('No of working Vs non-working days')

plt.subplot(1,3,3)
sns.countplot(data = df,x = 'holiday')
plt.title('No of holidays')
plt.show()
```

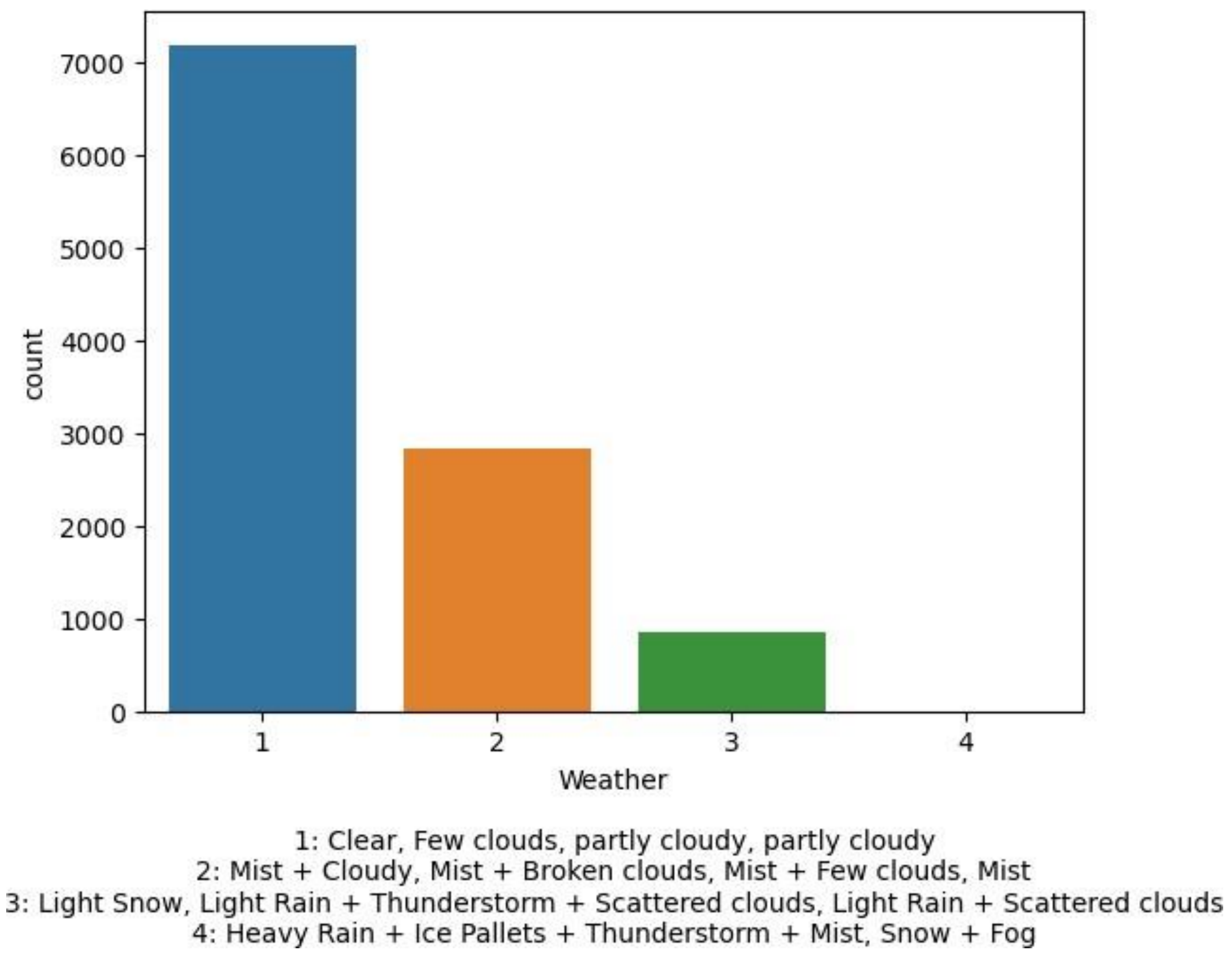




In [328…

```
# Weather distribution

sns.countplot(data=df,x='weather')
plt.xlabel('Weather\n\n1: Clear, Few clouds, partly cloudy, partly cloudy\n2: Mist + Cloudy, Mist + Broken clouds
plt.show()
```



Insights:

1. The Inter Quartile Range(IQR) for humidity lies to the right side of the distribution, which means that the data is left skewed. This means that most of the time the humidity is 45% - 78% (approx) according to the given data.
2. During most hours in a day, the temperature lies between 14 - 26 degree Celsius with min temperature in the given time period being 0.82 degree Celsius and maximum being 41 degree Celsius.
3. The Percentage increase in rentals from 2011 to 2012 is 67.0%.
4. There are more no of days with clear/partly cloudy weather than any other weather type. Misty/cloudy follows and there are almost no rentals during Heavy rain/thunderstorms.

Bivariate analysis

In [165…

```
df.head()
```

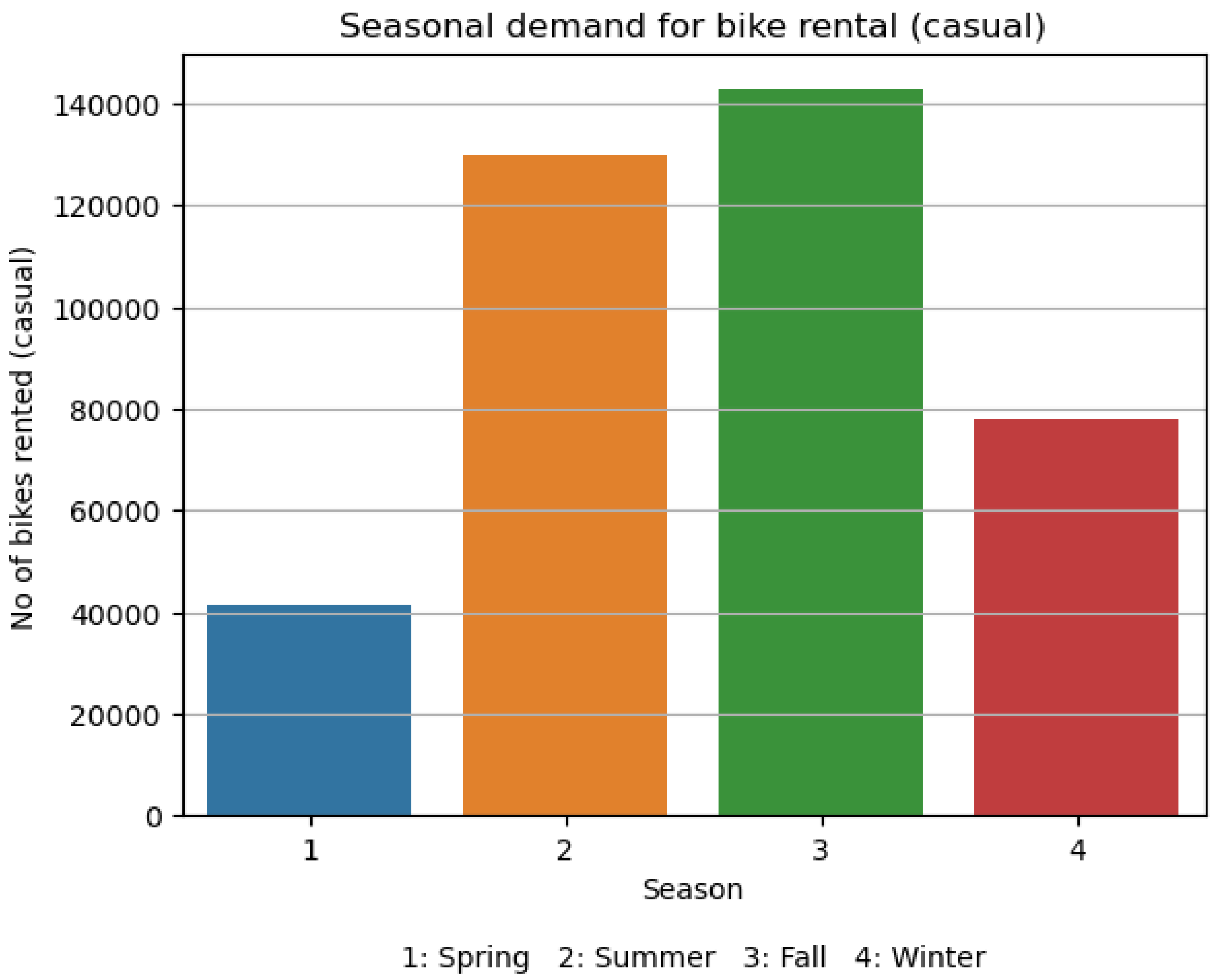
Out[165]:

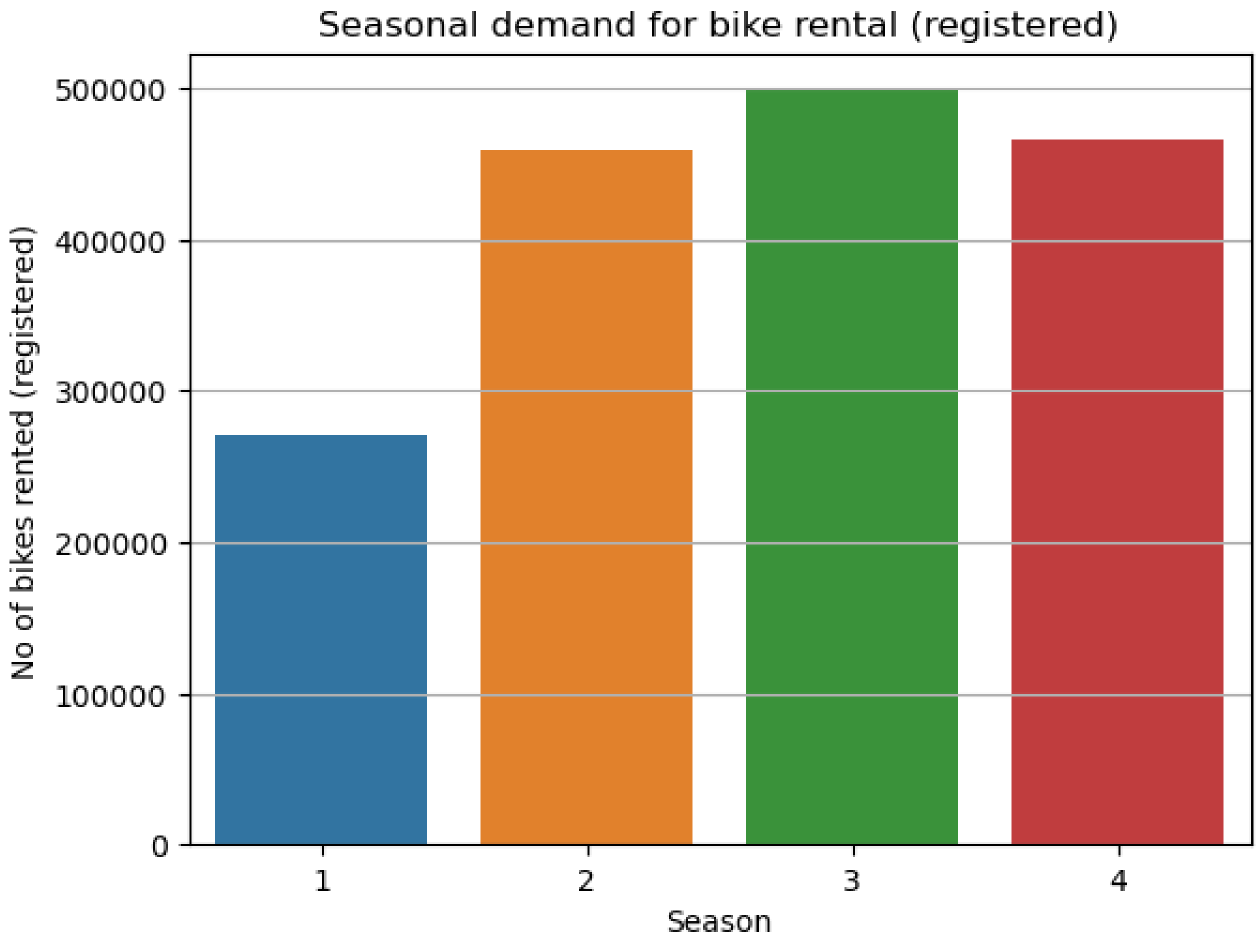
	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	date	time	year	month
0	1	0	0	1	9.84	14.395	81	0.0	3	13	16	2011-01-01	00:00:00	2011	1
1	1	0	0	1	9.02	13.635	80	0.0	8	32	40	2011-01-01	01:00:00	2011	1
2	1	0	0	1	9.02	13.635	80	0.0	5	27	32	2011-01-01	02:00:00	2011	1
3	1	0	0	1	9.84	14.395	75	0.0	3	10	13	2011-01-01	03:00:00	2011	1
4	1	0	0	1	9.84	14.395	75	0.0	0	1	1	2011-01-01	04:00:00	2011	1

In [270…

```
for i in ['casual','registered']:
    plt.grid()
    sns.barplot(data = df.groupby('season')[i].sum().reset_index(),x = 'season',y = i)
    plt.xlabel('Season\n\n1: Spring   2: Summer   3: Fall   4: Winter')
    plt.ylabel(f'No of bikes rented ({i})')
    plt.title(f'Seasonal demand for bike rental ({i})')
    plt.show()

    # do the same graph for casual rentals and registered rentals
```





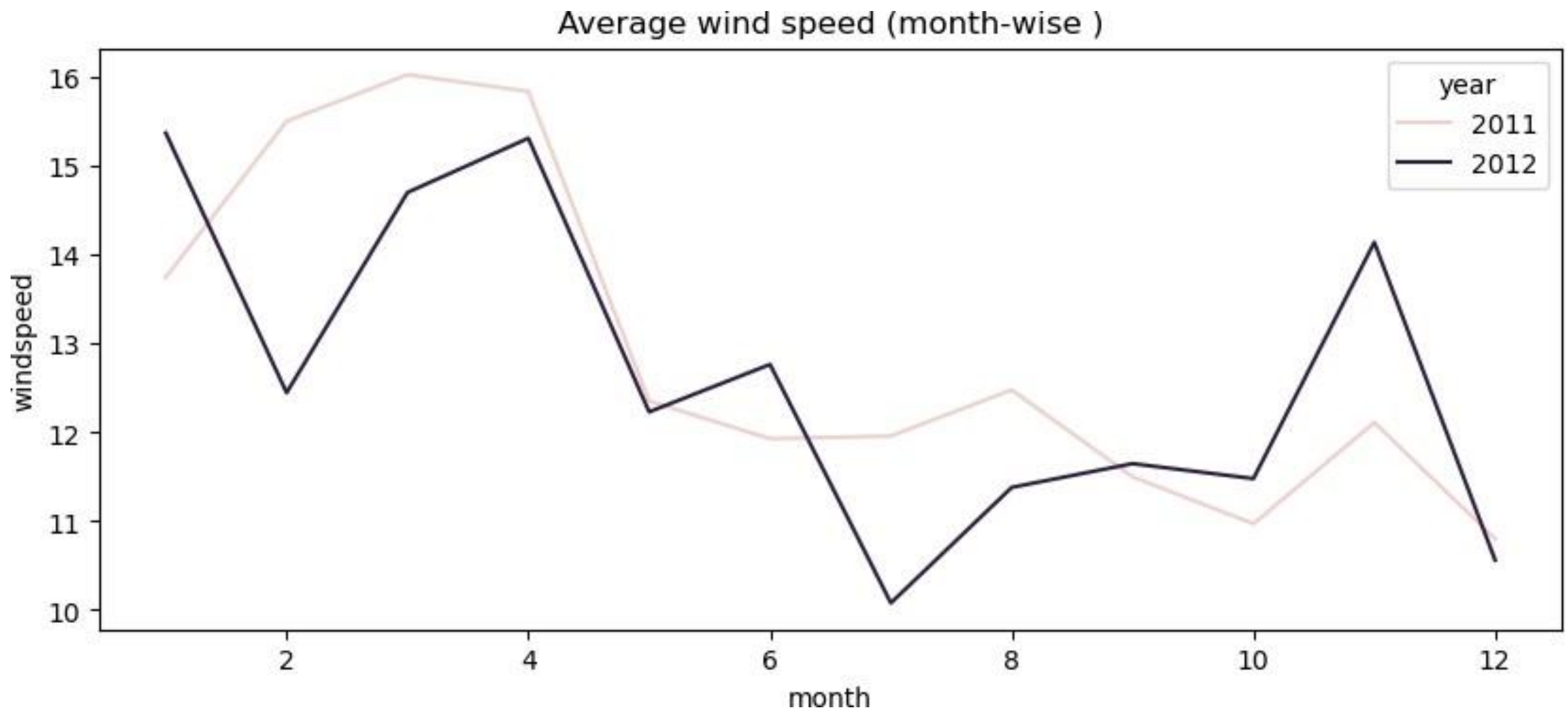
1: Spring   2: Summer   3: Fall   4: Winter

Insights:

1. We can observe that spring and winter rentals among casual cyclists is lower than registered.
2. Cycle rentals in fall is the highest followed by summer.

In [269…

```
plt.figure(figsize=(10,4))
sns.lineplot(data= df.groupby(['month','year'])['windspeed'].mean().reset_index(), x='month',y='windspeed',hue='y
plt.title('Average wind speed (month-wise )')
plt.show()
```

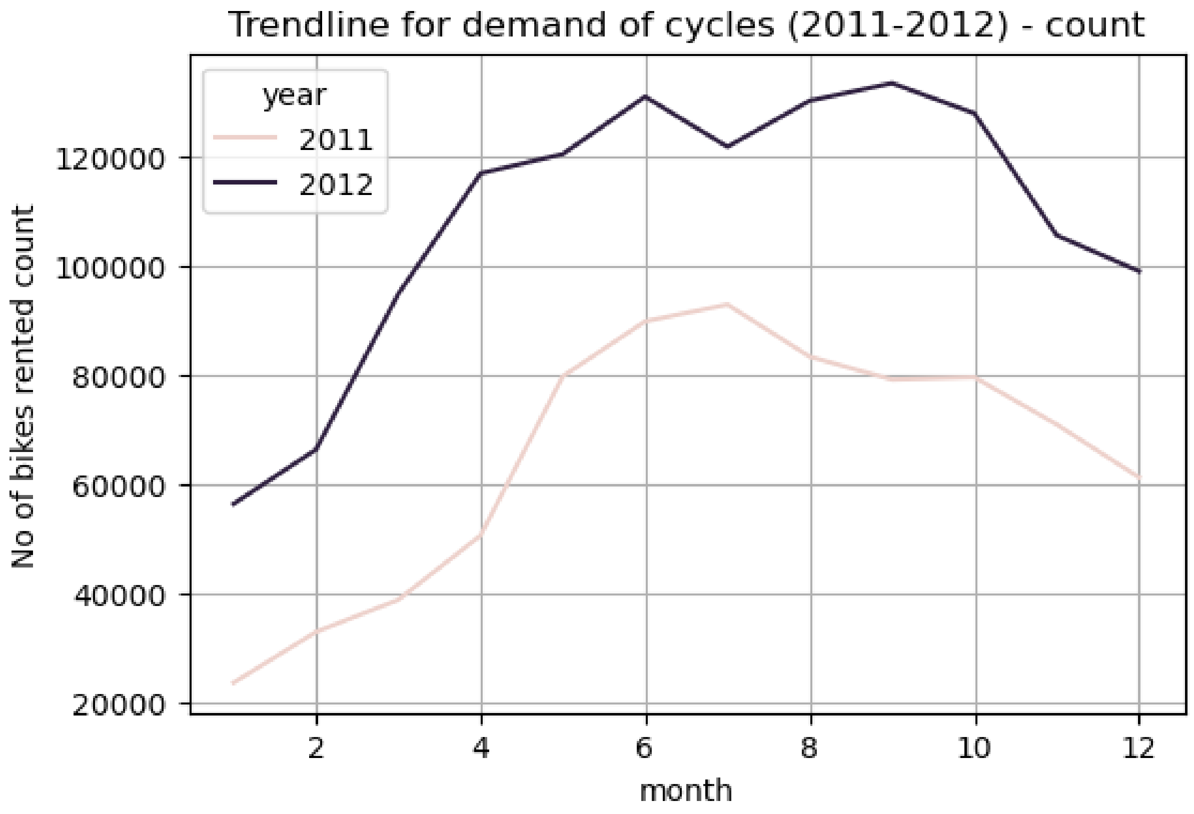


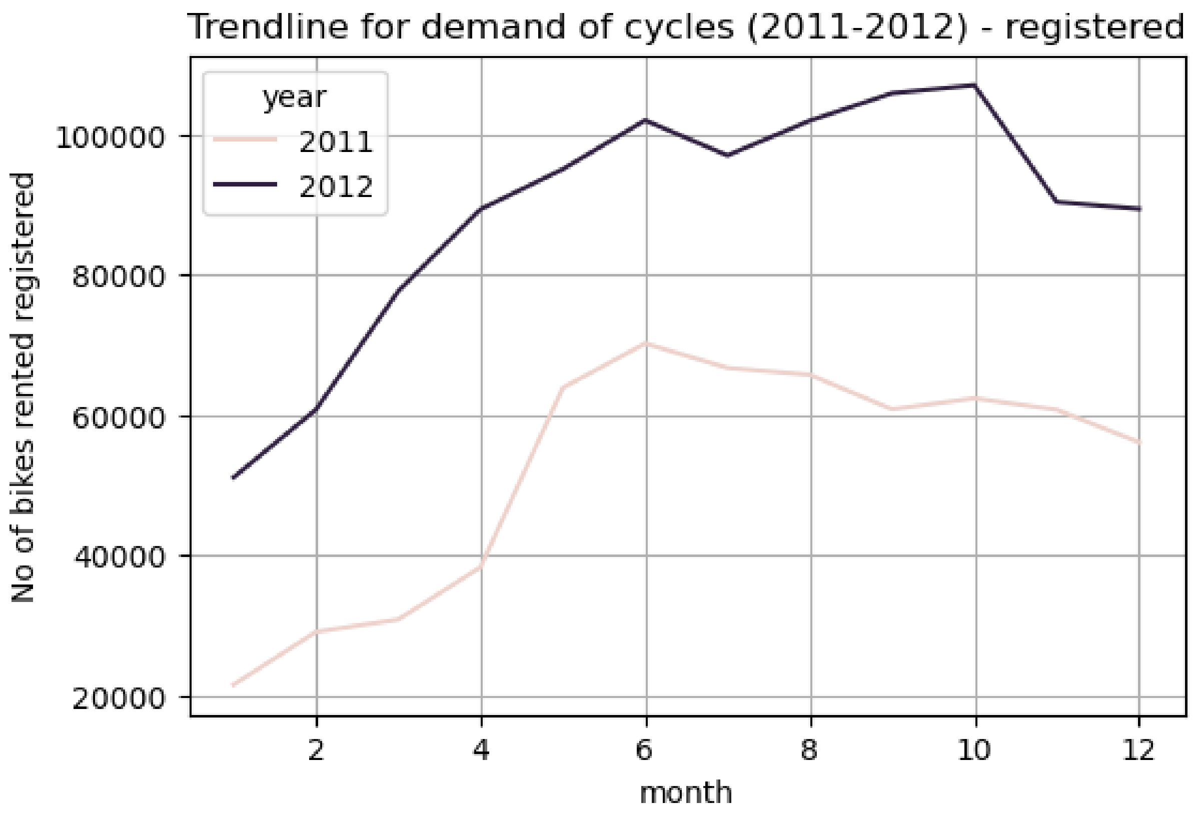
In [333…

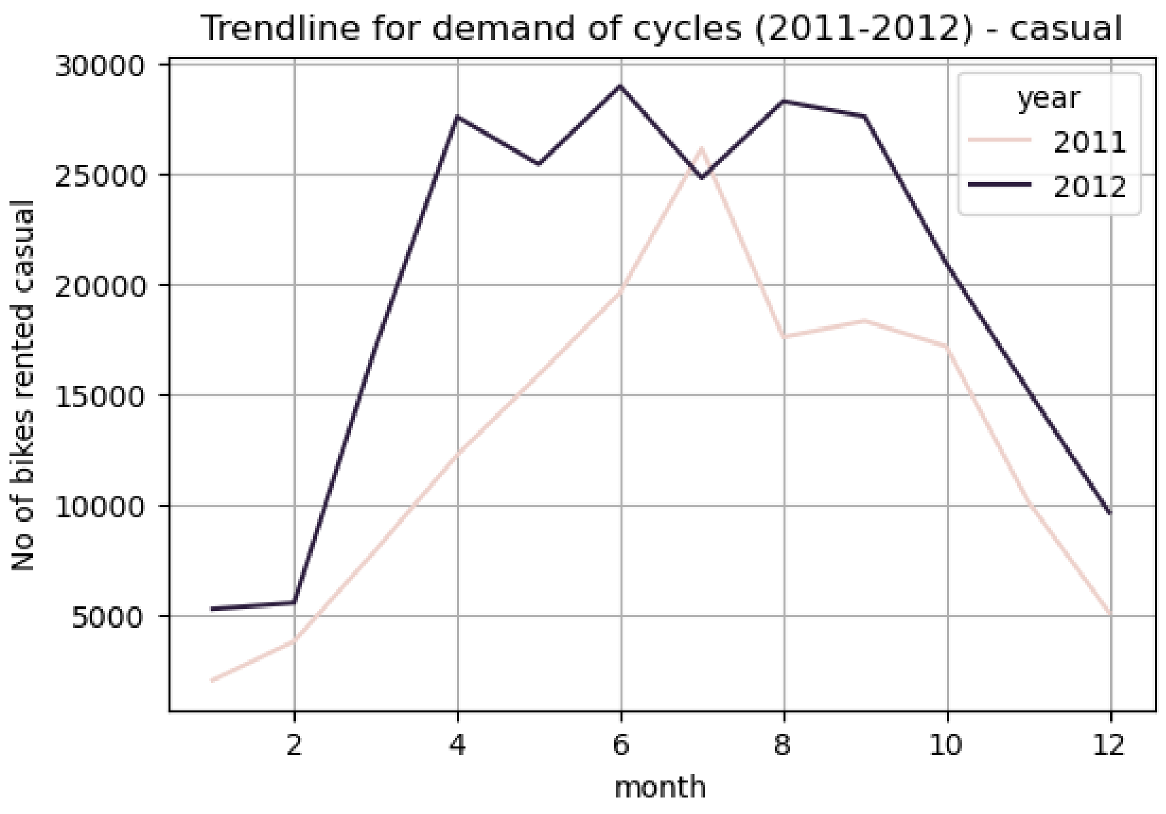
```
# Trends for casual and registered cyclists

for i in ['count','registered','casual']:

    fig = plt.figure(figsize=(6,4))
    plt.grid()
    sns.lineplot(data = df.groupby(['year','month'])[i].sum().reset_index(),x = 'month',y=i,hue='year')
    plt.ylabel(f'No of bikes rented {i}')
    plt.title(f'Trendline for demand of cycles (2011-2012) - {i}')
    plt.show()
```


Trendline for demand of cycles (2011-2012) - count
year
2011
2012
No of bikes rented count
month


Trendline for demand of cycles (2011-2012) - registered
year
2011
2012
No of bikes rented registered
month


Trendline for demand of cycles (2011-2012) - casual
year
2011
2012
No of bikes rented casual
month

Insights:

Windspeed plot:

1. Windspeeds seem comparitively higher in the first and last few months compared to the rest of the year.
2. The demand seems to be somewhat inline with the graph of windspeed;lower in the first 3 and last 3 months of the year and higher in the middle of the year.

Casual Rentals trendline Observations:

1. We can note that the demand increases exponentially in the first four months and decreses from 9th to 12th months.
2. in the months from 4 to 9 for 2011 sees a slow rise in rentals till 7th month and a decline since then, whereas 2012 sees a sharp increase from 2nd to 4th months and the demand is generally constant from 4th to 9th month before a rapid decline.

Total Rentals Trendline Observations:

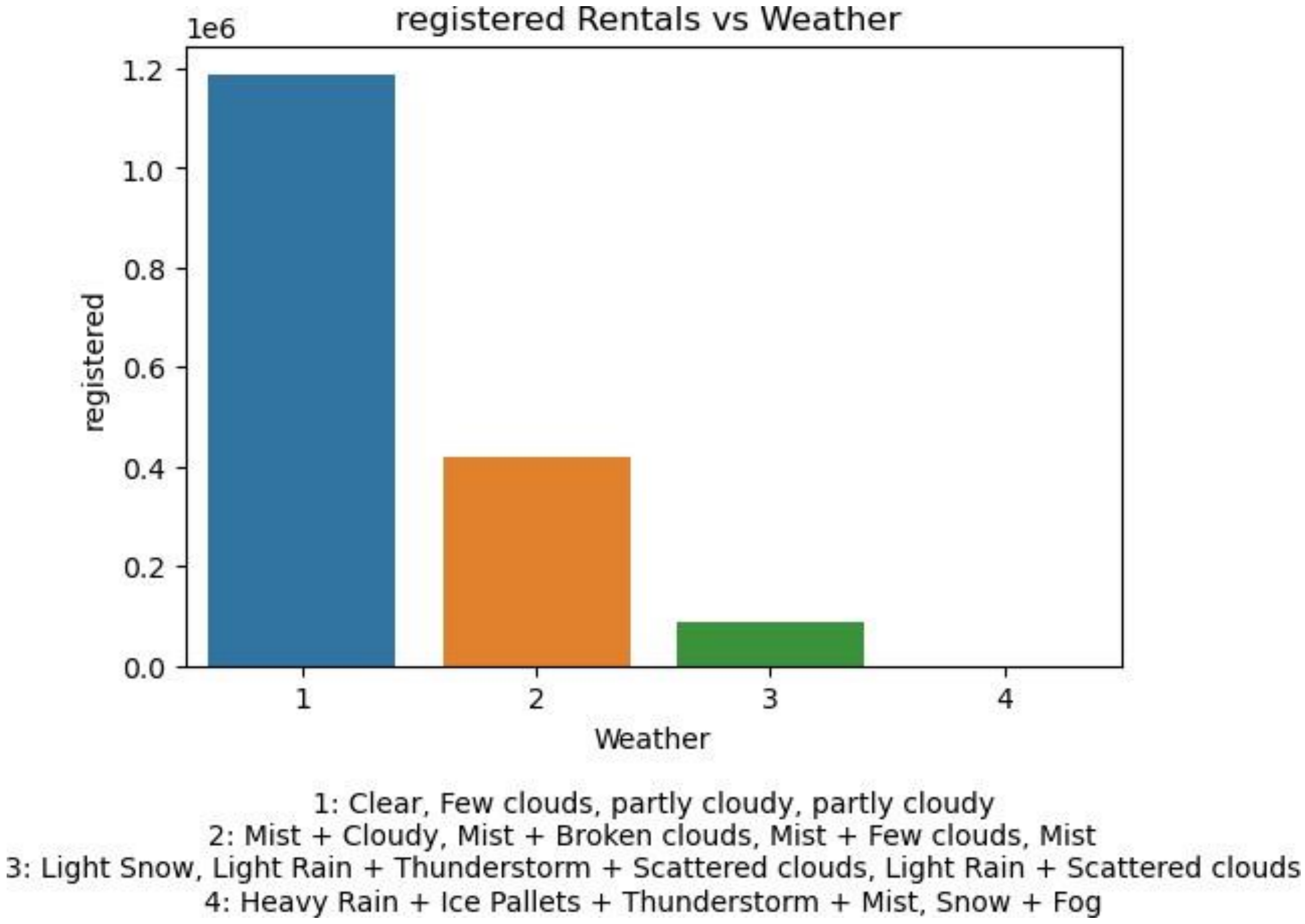
1. We can note that the total count of bikes in use has increased from the start of the year to the end of the year.
2. the overall count for the year 2012 increase significantly since the previous year 2011.

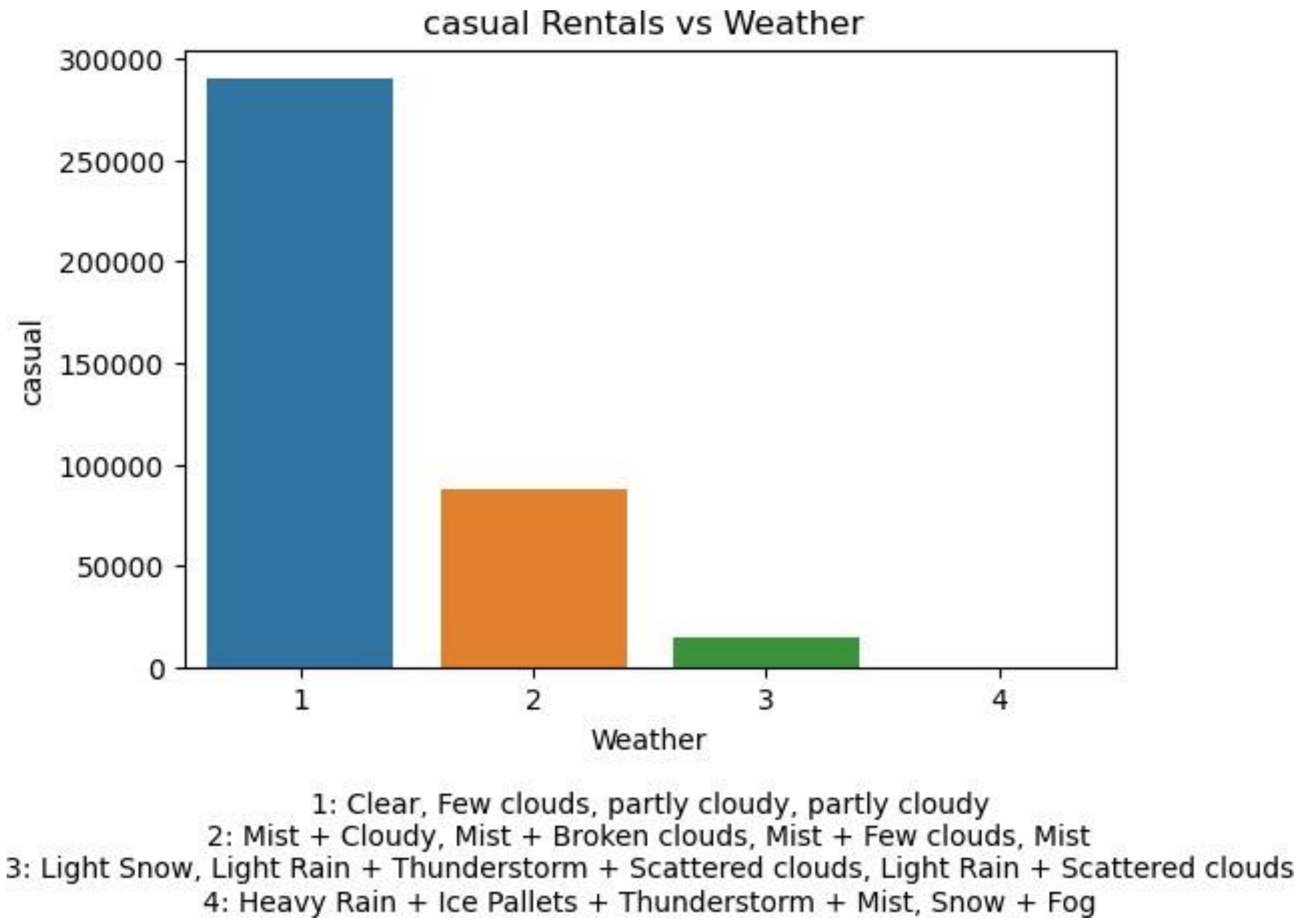
Registered Rentals trendline Observations:

1. The trendline for registered rentals is very similar to that of 'Total rentals'. Casual rentals trend differs from Total and Registered.

In [329…

```
for i in ['registered','casual']:
    fig = plt.figure(figsize=(6,4))
    sns.barplot(data  =  df.groupby('weather')[i].sum().reset_index(),x='weather',y=i)
    plt.xlabel('Weather\n\n1: Clear, Few clouds, partly cloudy, partly cloudy\n2: Mist + Cloudy, Mist + Broken cl
    plt.title(f'{i} Rentals vs Weather ')
    plt.show()
```





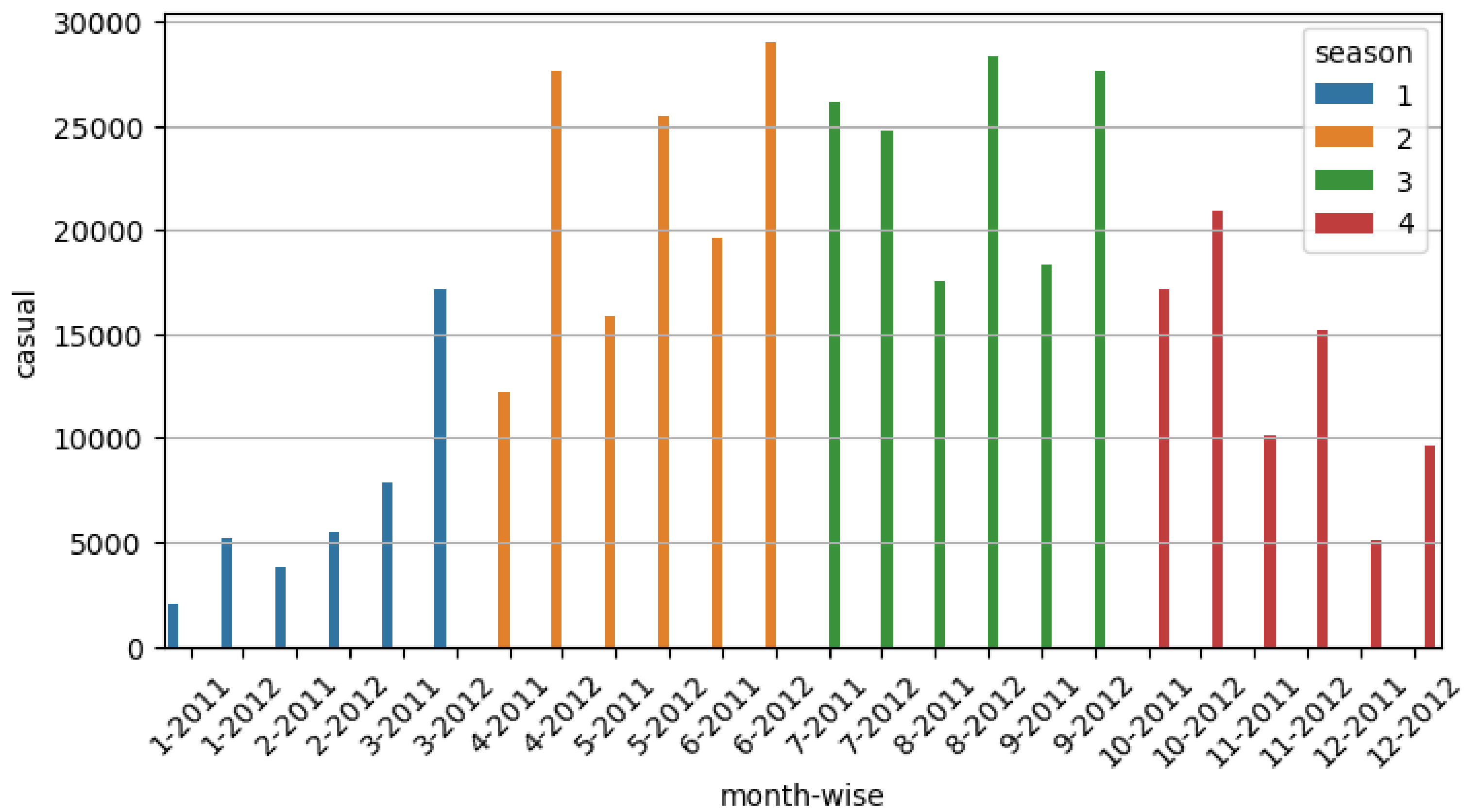
Insight:

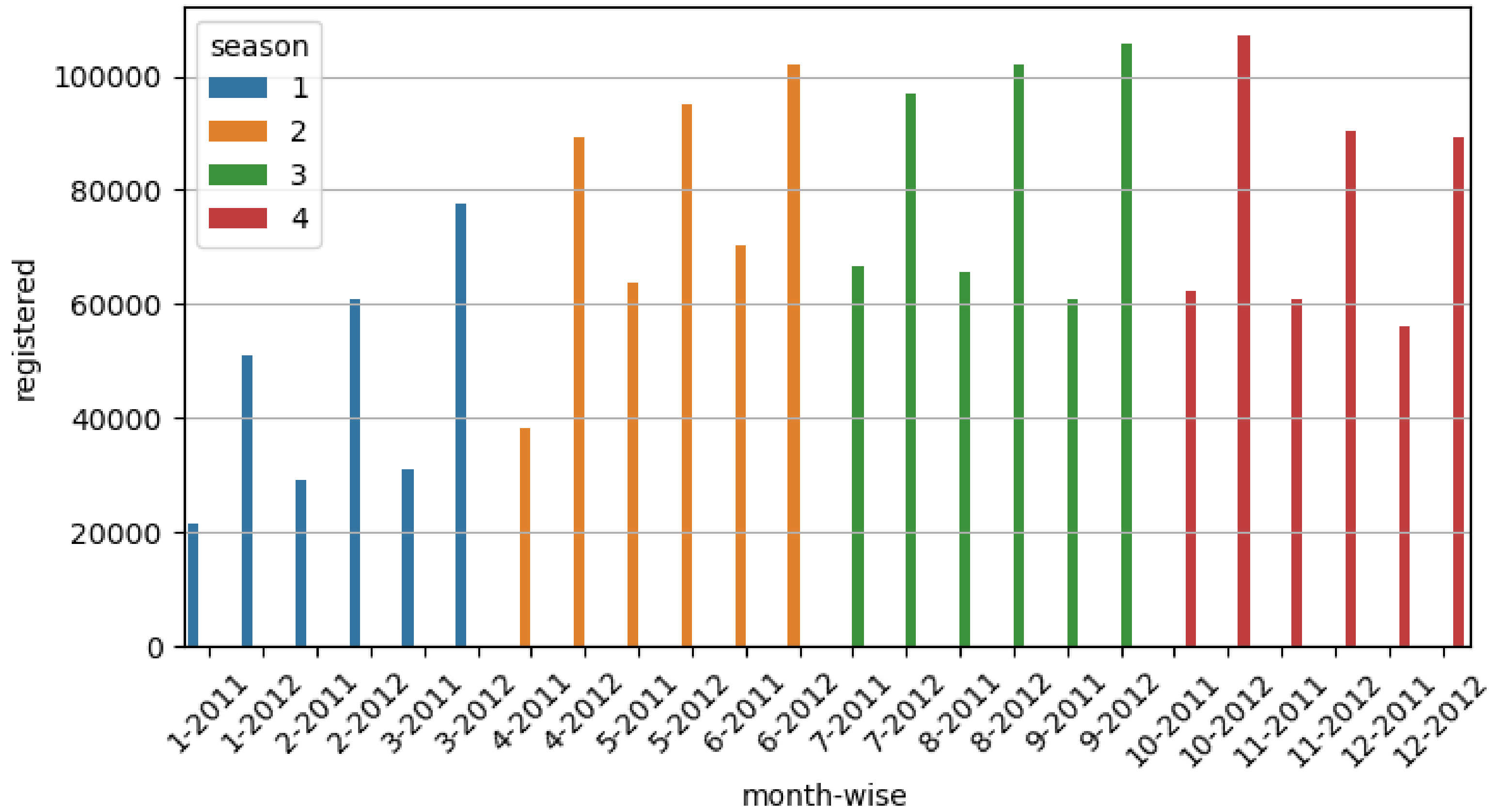
1. The trend for registered and casual is similar.

In [330…

```
# checking impact of different categories on casual

for i in ['casual','registered']:
    df_season = df.groupby(['month-wise','season'])[i].sum().reset_index()
    df_season = df_season.where(df_season[i]>0).dropna()
    df_season = df_season.sort_values(by=['season','month-wise'])
    fig = plt.figure(figsize=(8,4))
    plt.grid()
    sns.barplot(data = df_season,x = 'month-wise',y= i,hue='season')
    plt.xticks(rotation=45)
    plt.show()
```



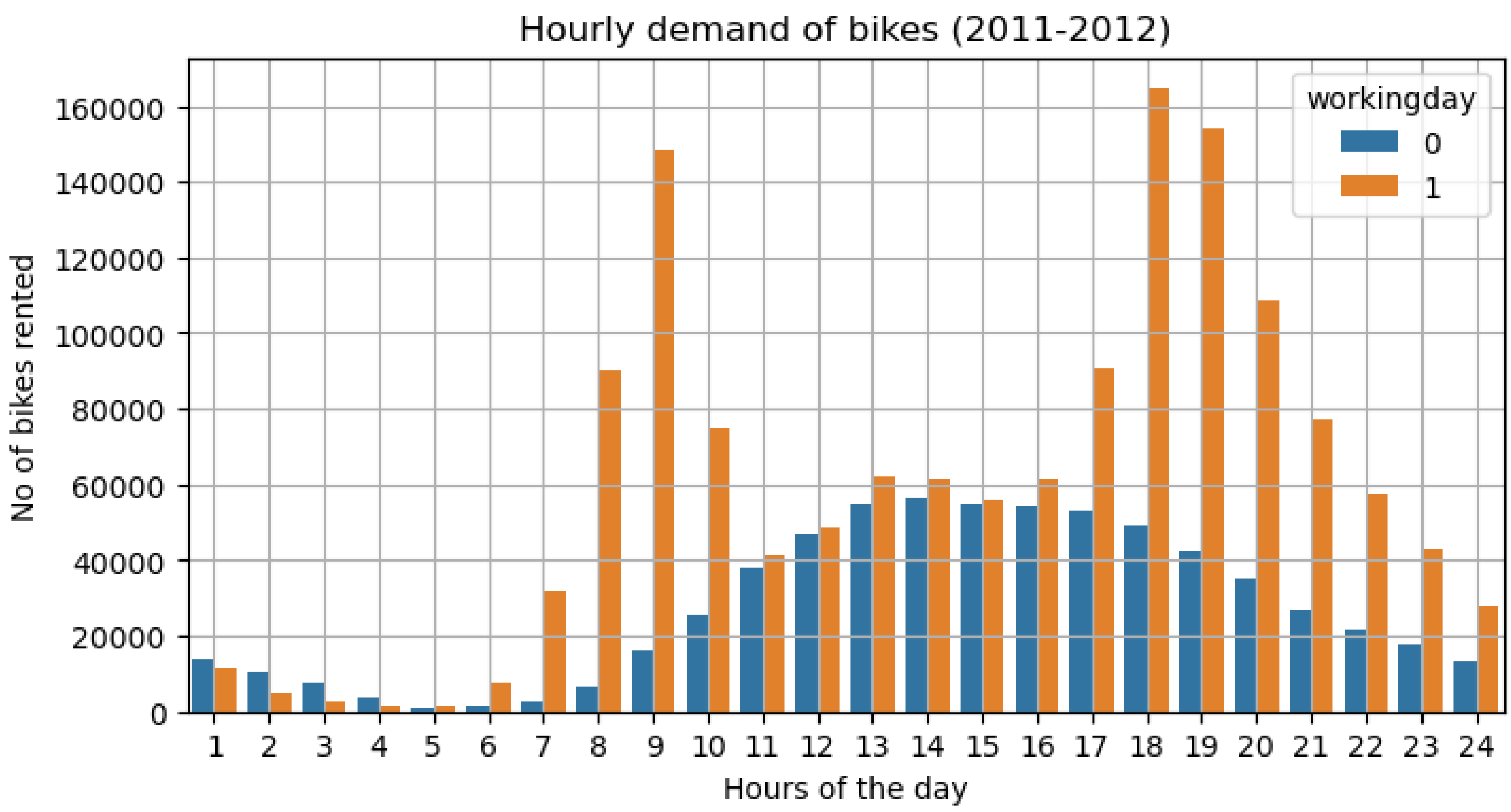


Insights:

1. We can see that casual and registered rentals follow a similar pattern with respect to season (despite casual rental being significantly lower than registered rentals.
2. We can also confirm the observation made earlier that the rentals during seasons 1 and 4 are much lower compared to seasons 2 and 3 for casual cyclists.

In [322…
```
hrs = [i for i in range(1,25)]

fig = plt.figure(figsize=(8,4))
sns.barplot(data = df.groupby(['time','workingday'])['count'].sum().reset_index(),x = 'time',y='count',hue='worki
plt.xlabel('Hours of the day')
plt.ylabel('No of bikes rented')
plt.title('Hourly demand of bikes (2011-2012)')
plt.grid()
plt.xticks([i for i in range(0,24)],hrs)
plt.show()
```

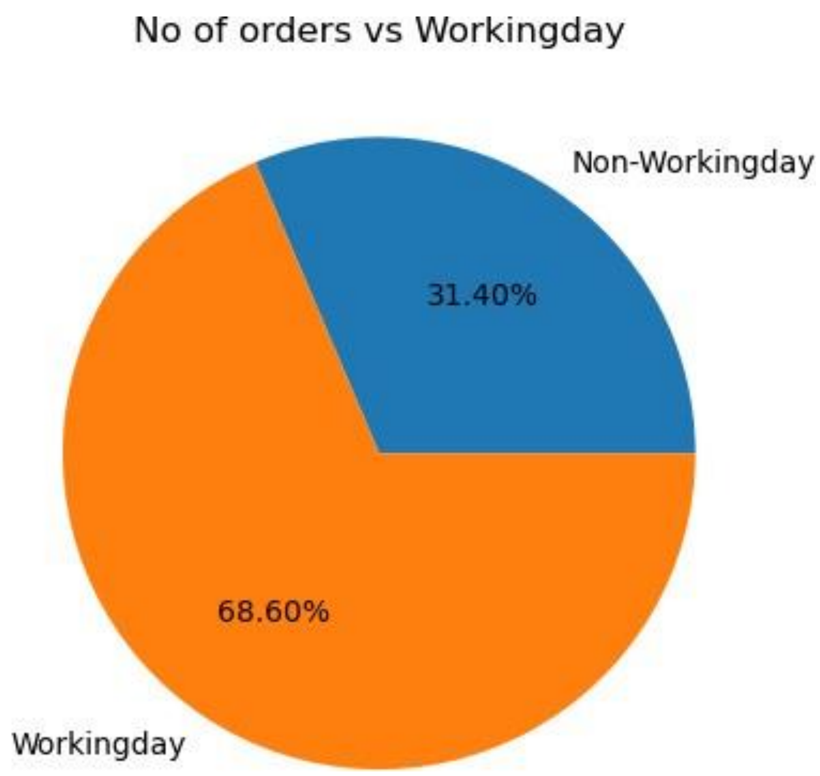


Insights:

1. The hourly demand in bar graph shows that Yulu has from 7AM to 7PM.
2. The demand is particularly high during 8AM and 4PM to 6PM.
3. Non working day demand is lower in general than workingday demand.

In [323…
```
# ratio of working days to non workingdays

plt.pie(df.groupby('workingday')['count'].sum(),labels = ['Non-Workingday','Workingday'],autopct='%1.2f%%')
plt.title('No of orders vs Workingday')
plt.show()
```



Insights:

1. There is more demand(68%) for bike rentals during a working day/non-holiday.

Working day effect on cycles rented

In [39]:
```
from scipy.stats import ttest_ind,chi2_contingency,f_oneway,kruskal
```

1. We will test how different the continous values of rental count is with respect to working and non working days.
2. Independent 2 sample ttest is best for this as the variable is continous and the samples are independent.

Null and alternate hypothesis :

Ho - mean of workingday rentals = mean of non workingday rentals

Ha - mean of workingday rentals > mean of non workingday rentals

In [40]:
```
df_wd = df.loc[df['workingday']==1]
df_notwd = df.loc[df['workingday']==0]
```

In [41]:
```
tstat, pvalue = ttest_ind(df_wd['count'],df_notwd['count'], alternative='greater')
print('test statistic:',tstat)
print("pvalue:",pvalue)
# perform twotailed/left tailed test
```

```
test statistic: 1.2096277376026694
pvalue: 0.11322402113180674
```

Results:

1. The pvalue(0.11) > alpha(0.05), hence we fail to reject the null hypothesis.
2. This implies that mean of workingday rentals is not significantly different from mean of non workingday rentals.

No. of cycles rented vs Seasons

> Setting a signiifcance value of 5% or 0.05

There are 4 categories in seasons field. Since we need to compare more than 2 groups, we could use ANOVA to find if there are significantly similar or different.

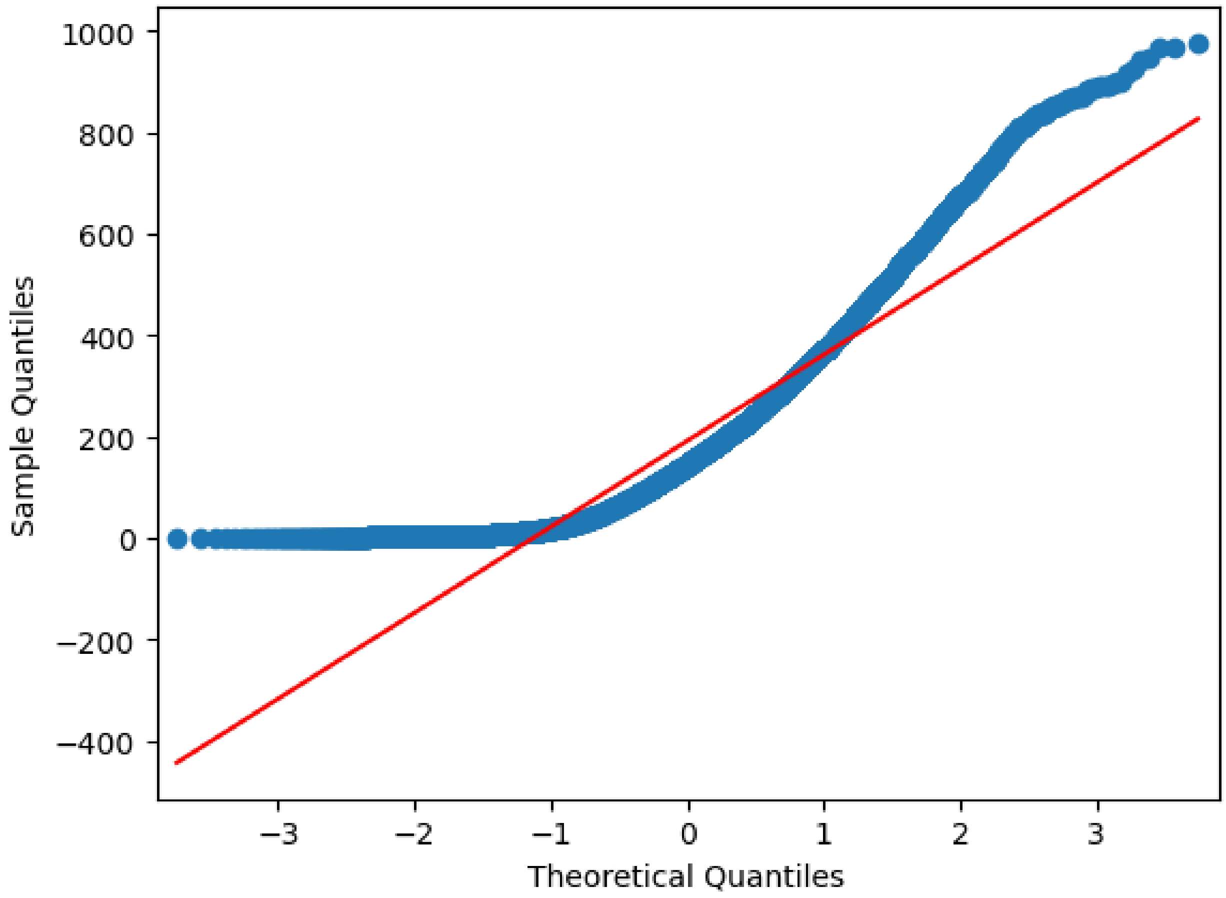
But first we have to check the assumtions of ANOVA

1. The variances of groups are similar/there is no significance difference between them.
2. The samples are taken from normally distributed population.

This can be done using visual representation (QQ plot) and levene's test to compare variances and Shapiro Wilk's test for normality.

QQ PLOT

In [42]:
```
from statsmodels.graphics.gofplots import qqplot
qqplot(df['count'], line ='r')
plt.show()
```



Insights:

1. If the dotted line almost superimposes the red reference line, then we can conclude that the population distribution is normal. But we can observe that that the data is heavily skewed towards 0 i.e. right skewed.

Levene's Test

Ho - The variances of samples are not significantly different.

Ha - The variance of samples (seasons) are significantly different.

In [43]:
```
# CHECK VARIANCES OF ALL SAMPLES

from scipy.stats import levene

df_s1 = df.loc[df['season']==1]['count']
df_s2 = df.loc[df['season']==2]['count']
df_s3 = df.loc[df['season']==3]['count']
df_s4 = df.loc[df['season']==4]['count']
```

In [57]:
```
levene(df_s1,df_s2,df_s3,df_s4)
```

Out[57]:
```
LeveneResult(statistic=187.7706624026276,   pvalue=1.0147116860043298e-118)
```

Insights:

1. pval is very low, we can reject the null hypothesis and conclude that varinces of the samples are significantly different.

Shapiro Wilk test

In [67]:
```
from scipy.stats import shapiro
print("Normality test for Season 1's data:",shapiro(df_s1))
print("Normality test for Season 2's data:",shapiro(df_s2))
print("Normality test for Season 3's data:",shapiro(df_s3))
print("Normality test for Season 4's data:",shapiro(df_s4))
```

```
Normality test for Season 1's data: ShapiroResult(statistic=0.8087388873100281, pvalue=0.0)
Normality test for Season 2's data: ShapiroResult(statistic=0.900481641292572, pvalue=6.039093315091269e-39)
Normality test for Season 3's data: ShapiroResult(statistic=0.9148160815238953, pvalue=1.043458045587339e-36)
Normality test for Season 4's data: ShapiroResult(statistic=0.8954644799232483, pvalue=1.1301682309549298e-39)
```

Decision:

1. The very very low pvalue (1*10^-118) indicates that the null hypothesis can be rejected.

2. this means that the variances of the samples drawn are not similar/ they are significantly different.

Anova

We will still go ahead with testing our hypothesis using ANOVA..

Null and alternate hypothesis :

Ho - there is no significant difference between the groups of all 4 seasons

Ha - there is a significant difference between the groups of all 4 seasons

In [45]:
```
fstat,pvalue_anova = f_oneway(df_s1,df_s2,df_s3,df_s4)
print('f statistic:',fstat)
print("pvalue:",pvalue_anova)
```

```
f statistic: 236.94671081032106
pvalue: 6.164843386499654e-149
```

Decision:

1. The p value of the ANOVA test is very very low (less than 5% significance value-0.05) and hence we can reject the null hypothesis.
2. This implies that the no of cylcles rented has significant impact of season.

Kruskal Wallis test

As the assumptions of ANOVA are failing, we can use Kruskal Wallis test to confirm out observations with ANOVA.

Null and alternate hypothesis :

Ho - there is no significant difference between the groups of all 4 seasons

Ha - there is a significant difference between the groups of all 4 seasons

In [46]:
```
# using the same samples

kruskal(df_s1,df_s2,df_s3,df_s4)
```

Out[46]:
```
KruskalResult(statistic=699.6668548181988,   pvalue=2.479008372608633e-151)
```

Insights:

1. Even with Kruskal's test we can conclude that pvalue <<< 5% significance value(0.05)and reject null hypothesis.

No. of cyles rented vs Weather

We will repeat the same procedure as seasons for weather variable also.

Ho - The variances of samples is not significantly different

Ha - The variances of samples is significantly different

In [47]:
```
df_w1 = df.loc[df['weather']==1]['count']
df_w2 = df.loc[df['weather']==2]['count']
df_w3 = df.loc[df['weather']==3]['count']
df_w4 = df.loc[df['weather']==4]['count']

levene(df_w1,df_w2,df_w3,df_w4)
```

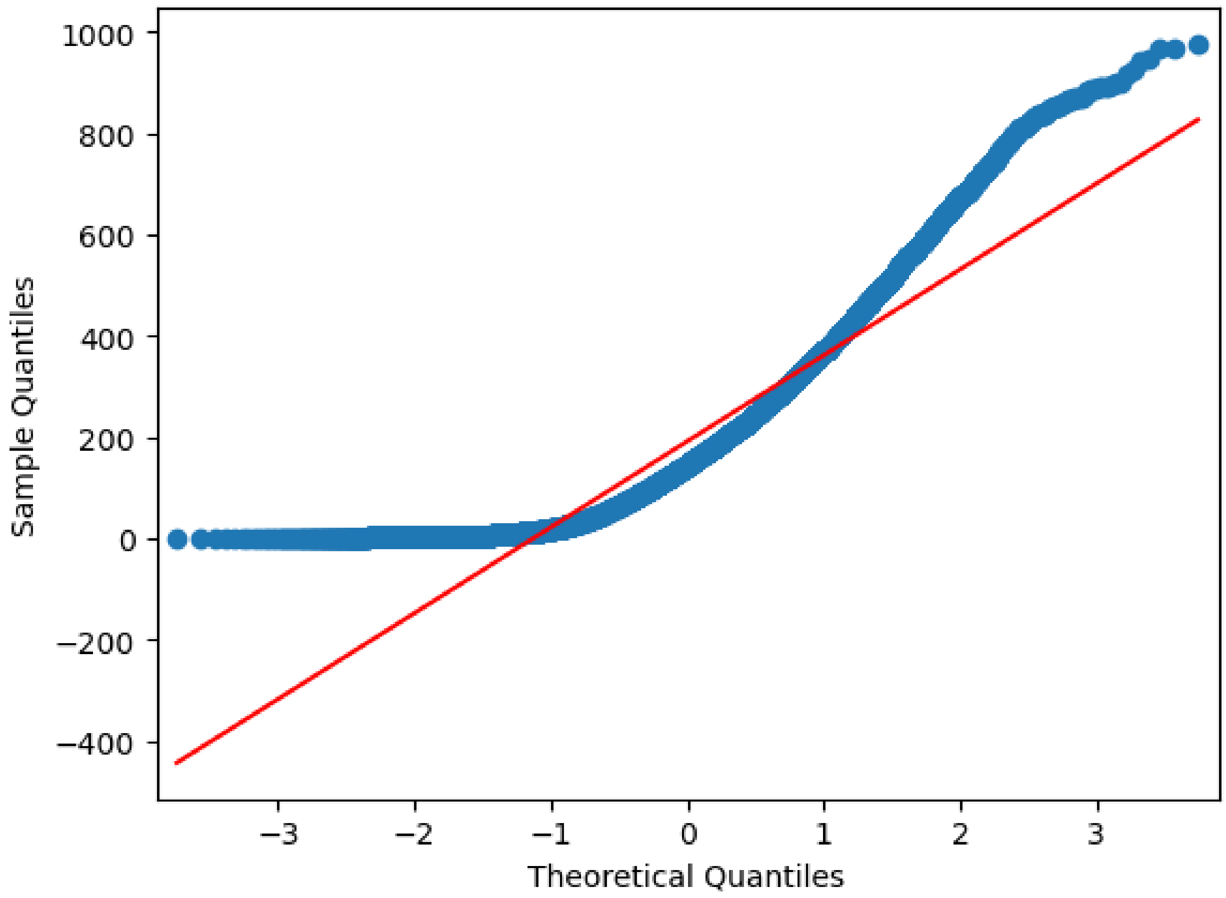
Out[47]:
```
LeveneResult(statistic=54.85106195954556,    pvalue=3.504937946833238e-35)
```

Insight:

1. the pval is very low and hence the variances are significantly different.

In [48]:
```
# QQ plot

qqplot(df['count'], line ='r')
plt.show()
```



Insights:

1. the qq plot yields the same results as before and hence the population distribution is not normal.

Anova

Null and alternate hypothesis :

Ho - there is no significant difference between the groups of all 4 weathers

Ha - there is a significant difference between the groups of all 4 weathers

In [49]:
```
fstat2,pvalue_anova2 = f_oneway(df_w1,df_w2,df_w3,df_w4)
print('f statistic:',fstat2)
print("pvalue:",pvalue_anova2)
```

```
f statistic: 65.53024112793271
pvalue: 5.482069475935669e-42
```

In [334…
```
# perform kruskal's test

kruskal(df_w1,df_w2,df_w3,df_w4)
```

Out[334]:
```
KruskalResult(statistic=205.00216514479087,    pvalue=3.501611300708679e-44)
```

Decision:

Kruskal Wallis test also confirms the results from ANOVA.

We can reject the null hypothesis, as pvalue < 0.05(significance value) and conclude that there is a significant difference in samples drawn from each weather.

Weather vs Season

Null and alternate hypothesis :

Ho - seasons are independent of weather

Ha - seasons are dependent on weather

In [50]:
```
weather_seasons = pd.crosstab(df['season'],df['weather'],margins=True)
weather_seasons
```

Out[50]:

weather season	1	2	3	4	All
1	1759	715	211	1	2686
2	1801	708	224	0	2733
3	1930	604	199	0	2733
4	1702	807	225	0	2734
All	7192	2834	859	1	10886

In [51]:
```
chi_stat, chi_pval, dof, obs_val = chi2_contingency(weather_seasons)
print('pvalue',chi_pval)
```

```
pvalue 3.1185273325126814e-05
```

Decision:

The pvalue for the chi square test is very low and less than 0.05, which leads us to conclude that we can reject the null hypothesis

Recommendations:

1. We could roll out offers for casual cyclists ("Encourage cycling") during spring to boost casual sales/increase registrations.
2. As the non working day rentals are generally lower than workingday rentals, the company could increase non workingday sales by offering deals on weekends/holidays
3. Season are dependent on weather.
4. Weather and season has an impact on the rental count.
5. The rentals on working day and non working day are not significantly different.