About Yulu

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

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

Yulu has recently suffered considerable dips in its revenues. They have contracted a consulting company to understand the factors on which the demand for these shared electric cycles depends. Specifically, they want to understand the factors affecting the demand for these shared electric cycles in the Indian market.

How you can help here?

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

In [1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import os
from scipy import stats
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# This Python 3 environment comes with many helpful analytics libraries installe

# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly import tools
from plotly.subplots import make_subplots
from plotly.offline import iplot, init_notebook_mode, download_plotlyjs, plot
init_notebook_mode(connected=True)
```

In [3]:
```
import statistics as st
import scipy.stats as stats
from scipy.stats import ttest_ind
```

```
from datetime import datetime
from scipy.stats import mannwhitneyu
from scipy.stats import kruskal
```

In [4]:
```
#csv_path = "C:\Users\deepa\Downloads\yulu_data.csv"
df = pd.read_csv("yulu_data.csv", delimiter=",")
#/kaggle/input/yulu-data/yulu_data.csv.xlsx
```

In [5]:
```
df.head()
```

Out[5]:

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

In [6]:
```
# no of rows amd columns in dataset
print(f"# rows: {df.shape[0]} \n# columns: {df.shape[1]}")
```

```
# rows: 10886
# columns: 12
```

In [7]:
```
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 [8]:
```
df.describe()
```

Out[8]:

	season	holiday	workingday	weather	temp	ater
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000(
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655(
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474€
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760(
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665(
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240(
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060(
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455(

Datatype of following attributes needs to changed to proper data type

- **datetime** - to datetime
- **season** - to categorical
- **holiday** - to categorical
- **workingday** - to categorical
- **weather** - to categorical

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)

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

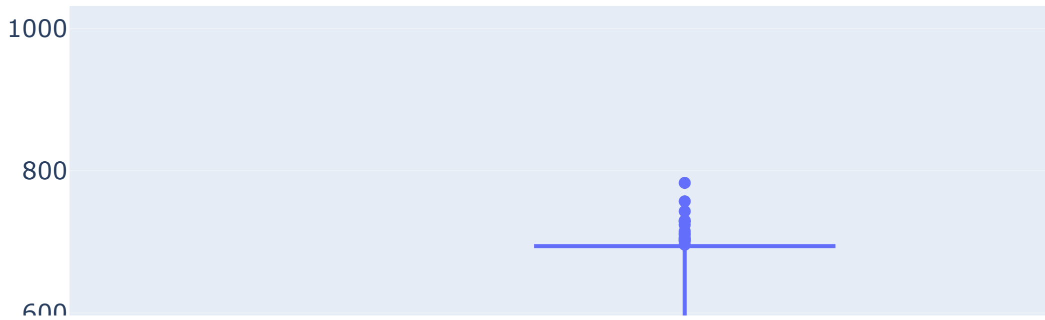
In [9]:
```
df['datetime'] = pd.to_datetime(df['datetime'])

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

In [10]:
```
df.iloc[:, 1:].describe(include='all')
```

Out[10]:

	season	holiday	workingday	weather	temp	atemp	humidit
count	10886.0	10886.0	10886.0	10886.0	10886.00000	10886.000000	10886.00000
unique	4.0	2.0	2.0	4.0	NaN	NaN	Na
top	4.0	0.0	1.0	1.0	NaN	NaN	Na
freq	2734.0	10575.0	7412.0	7192.0	NaN	NaN	Na
mean	NaN	NaN	NaN	NaN	20.23086	23.655084	61.88646
std	NaN	NaN	NaN	NaN	7.79159	8.474601	19.24503
min	NaN	NaN	NaN	NaN	0.82000	0.760000	0.00000
25%	NaN	NaN	NaN	NaN	13.94000	16.665000	47.00000
50%	NaN	NaN	NaN	NaN	20.50000	24.240000	62.00000
75%	NaN	NaN	NaN	NaN	26.24000	31.060000	77.00000
max	NaN	NaN	NaN	NaN	41.00000	45.455000	100.00000

In [11]:
```
px.box(df,x='workingday',y='count')
```



- There are no missing values in the dataset.
- **casual** and **registered** attributes might have outliers because their mean and median are very far away to one another and the value of standard deviation is also high which tells us that there is high variance in the data of these attributes.

In [12]:
```
# detecting missing values in the dataset
df.isnull().sum()
```

Out[12]:
```
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
```

There are no missing values present in the dataset.

In [13]:
```
# minimum datetime and maximum datetime to see the range of datetime values
df['datetime'].min(), df['datetime'].max()
```

Out[13]: (Timestamp('2011-01-01 00:00:00'), Timestamp('2012-12-19 23:00:00'))

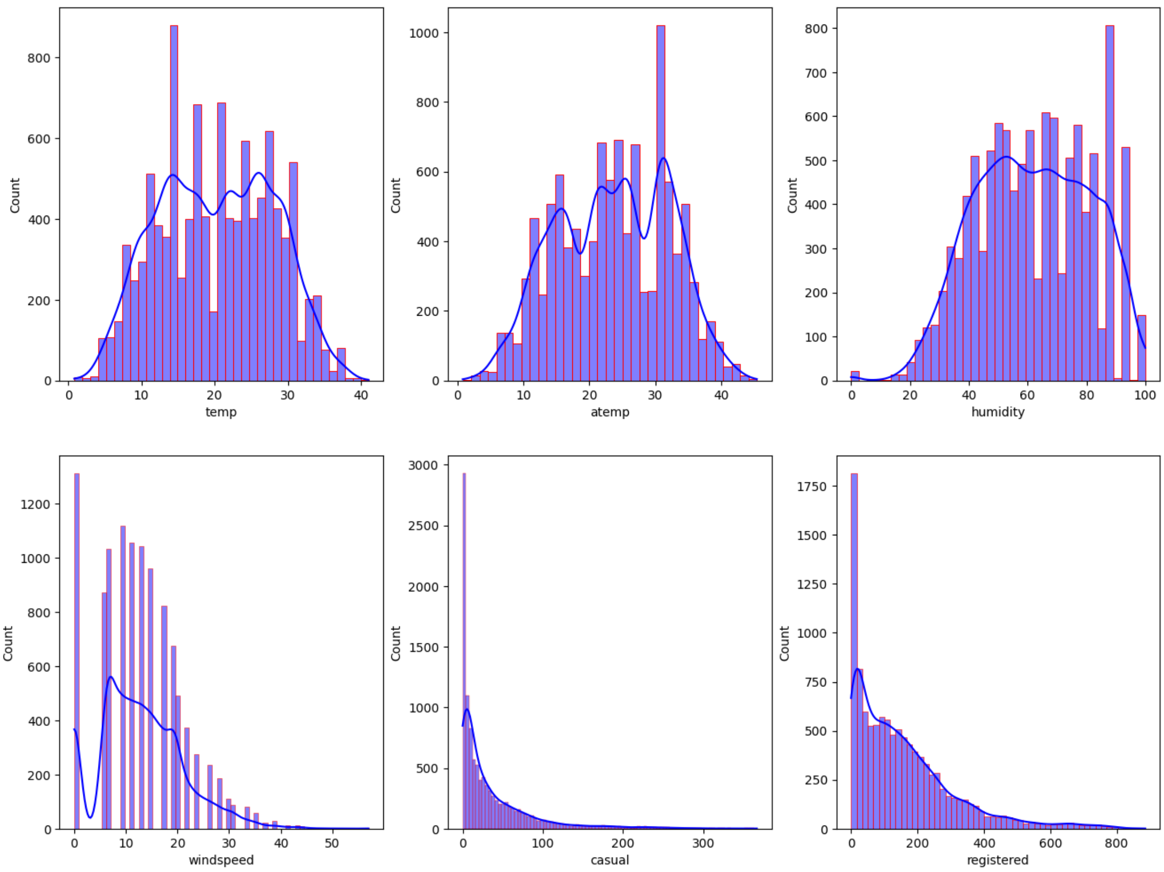
In [14]:
```
# number of unique values in each categorical columns
df[cat_cols].melt().groupby(['variable', 'value'])[['value']].count()
```

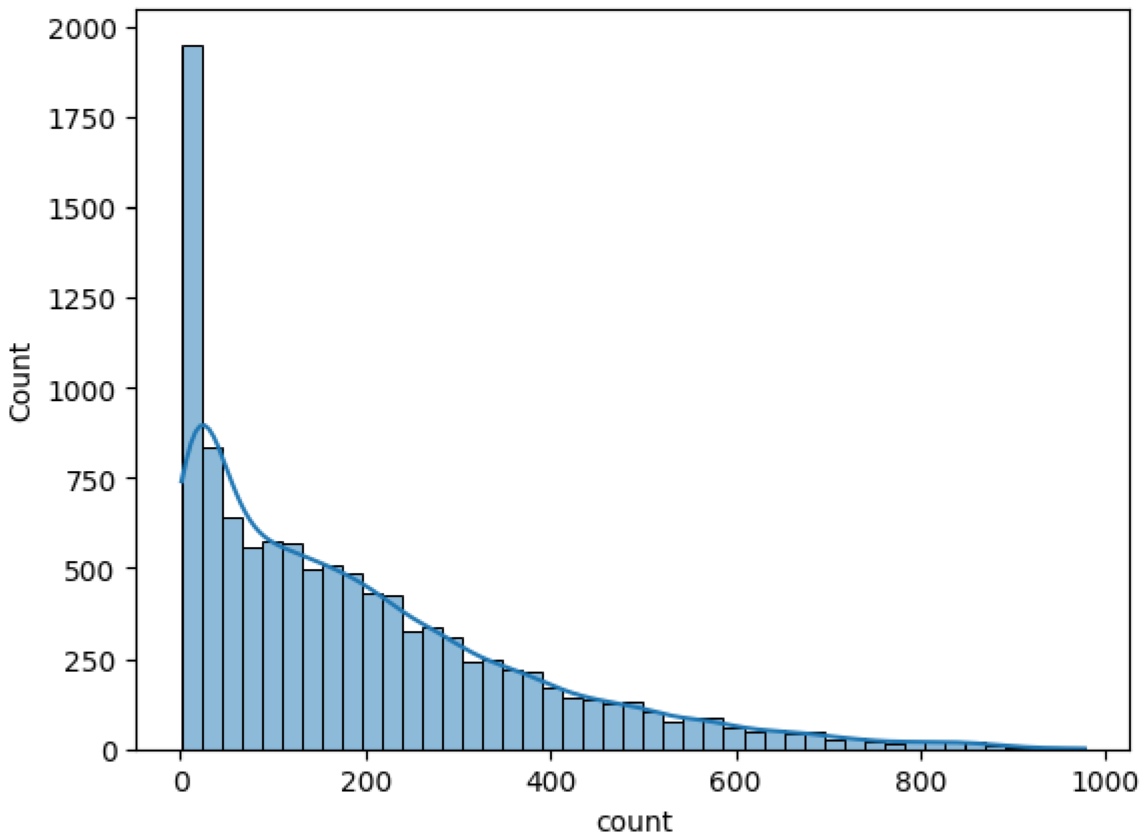
Out[14]:

variable	value	value
holiday	0	10575
	1	311
season	1	2686
	2	2733
	3	2733
	4	2734
weather	1	7192
	2	2834
	3	859
	4	1
workingday	0	3474
	1	7412

Univariate Analysis

In [15]:
```
# understanding the distribution for numerical variables
num_cols = ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered','co

fig, axis = plt.subplots(nrows=2, ncols=3, figsize=(16, 12))

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

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

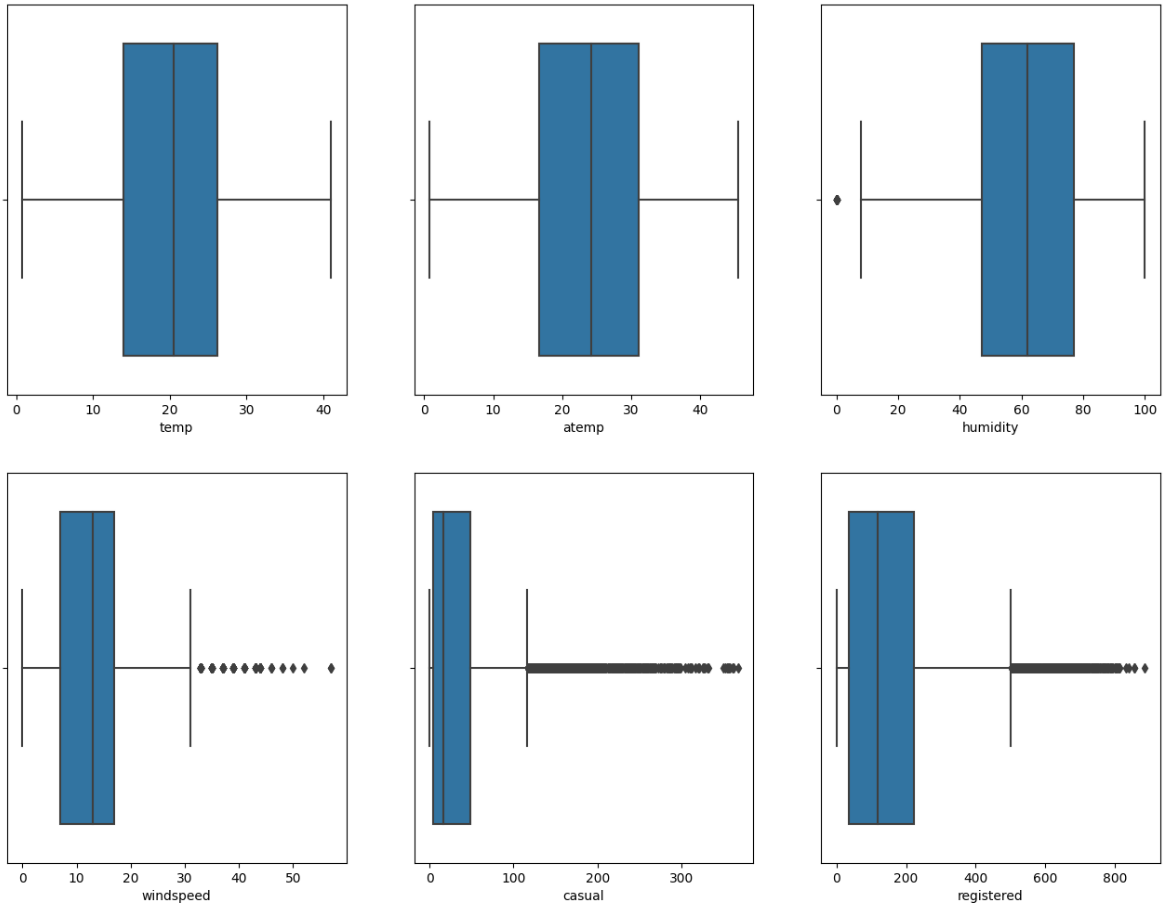


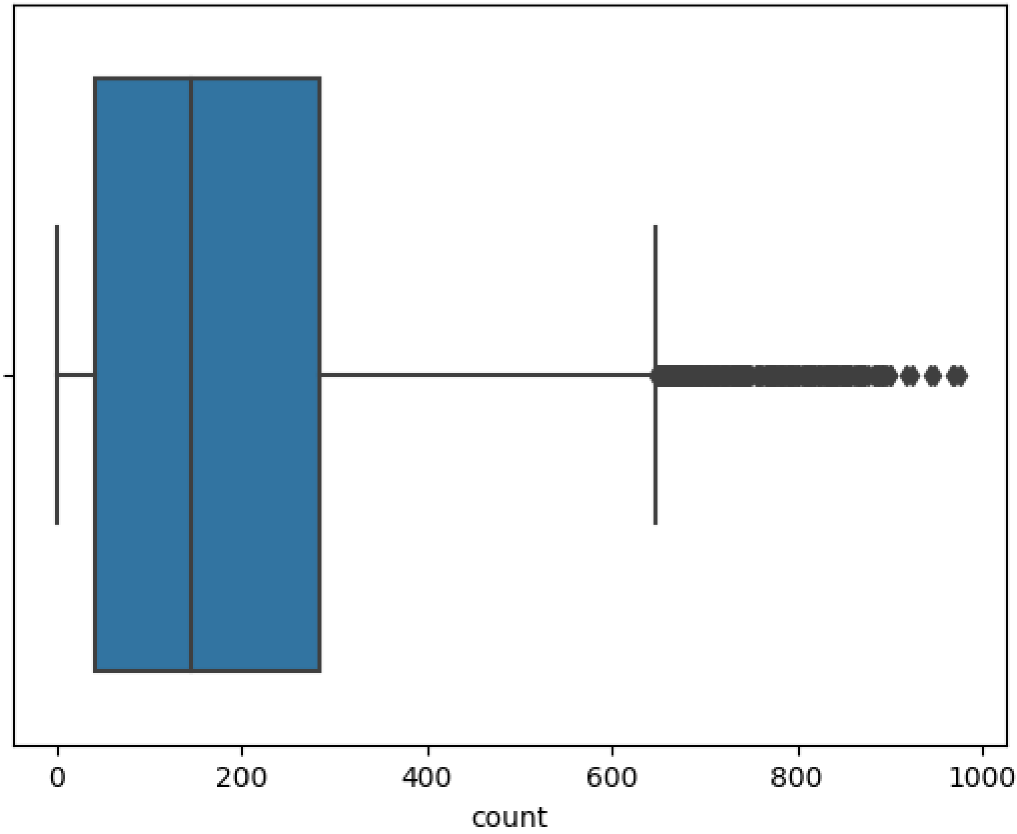


- **casual**, **registered** and **count** somewhat looks like **Log Normal Distrinution**
- **temp**, **atemp** and **humidity** looks like they follows the **Normal Distribution**
- **windspeed** follows the **binomial distribution**

In [16]:
```
# plotting box plots to detect outliers in the data
fig, axis = plt.subplots(nrows=2, ncols=3, figsize=(16, 12))
```

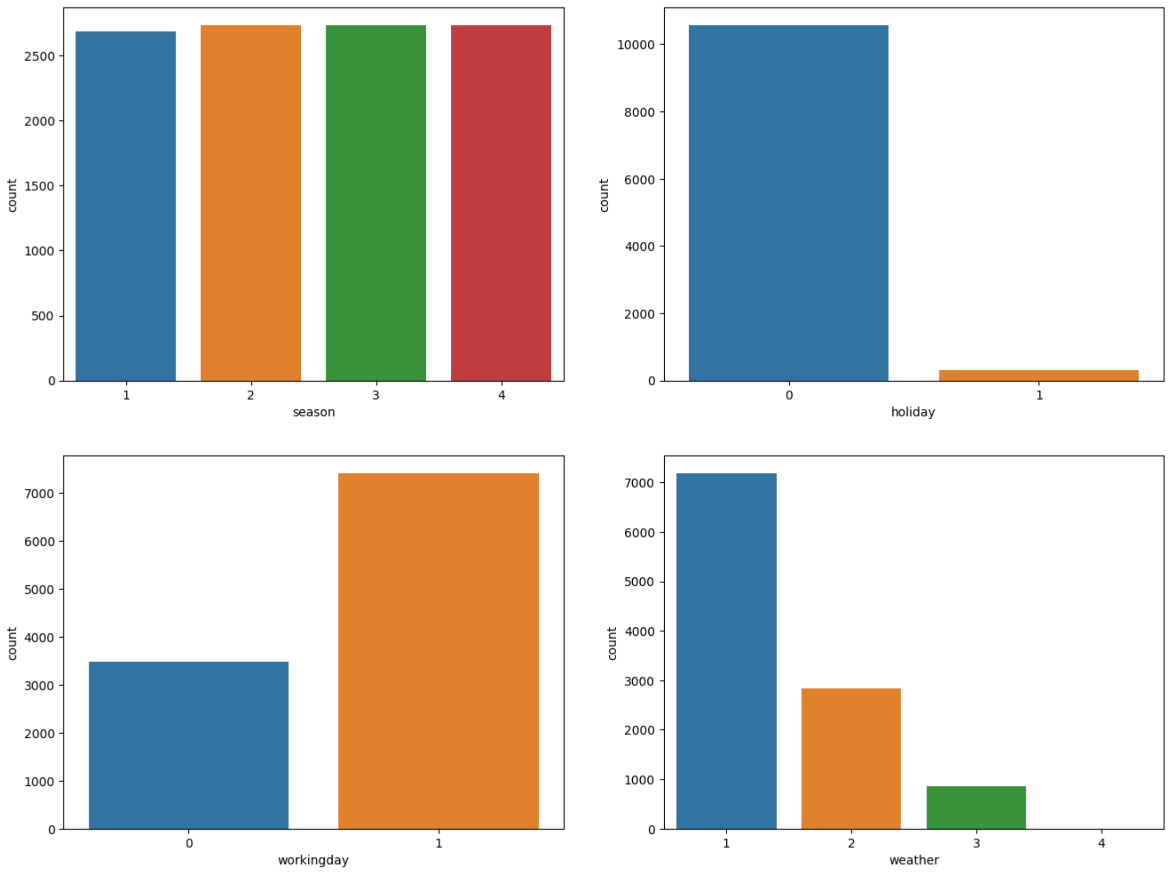
```
index = 0
for row in range(2):
    for col in range(3):
        sns.boxplot(x=df[num_cols[index]], ax=axis[row, col])
        index += 1

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





Looks like **humidity**, **casual**, **registered** and **count** have outliers in the data.

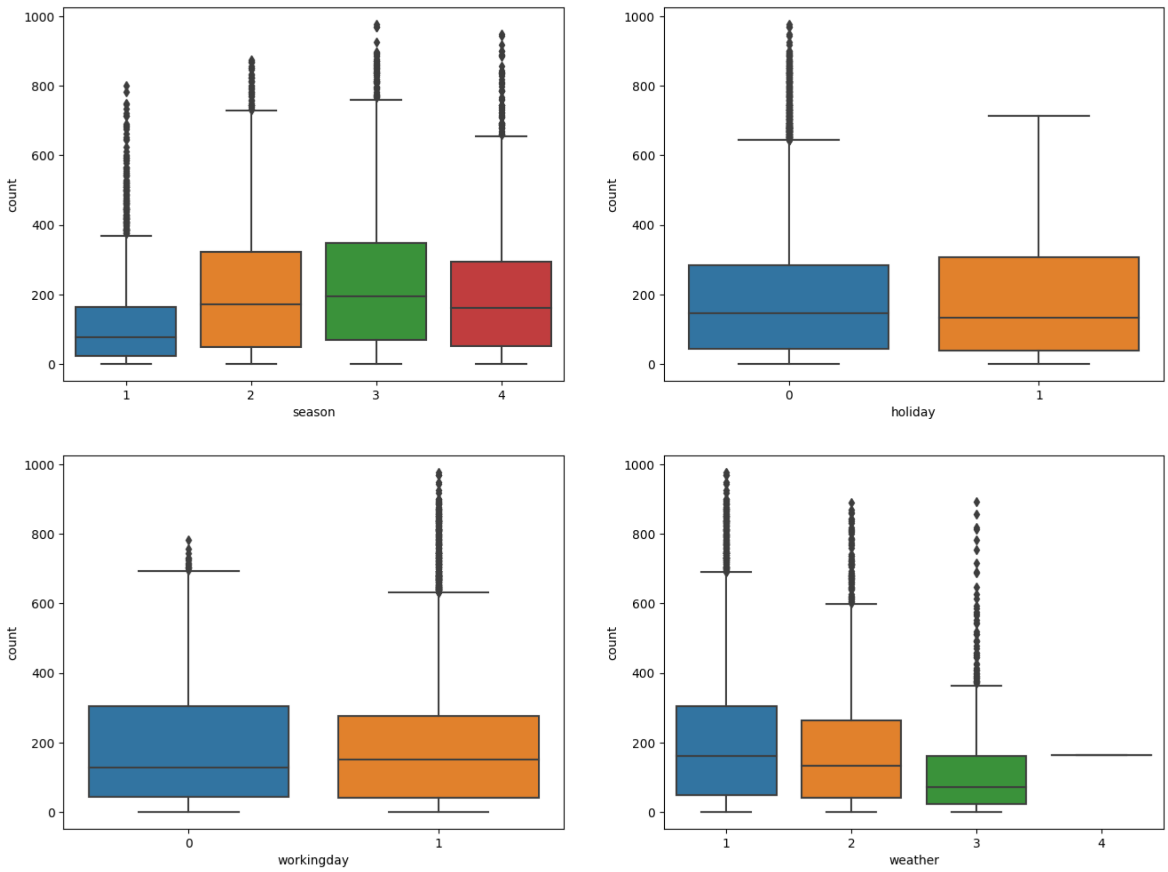
In [17]:
```
# countplot of each categorical column
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(16, 12))

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

plt.show()
```



Data looks common as it should be like equal number of days in each season, more working days and weather is mostly `Clear, Few clouds, partly cloudy, partly cloudy`.

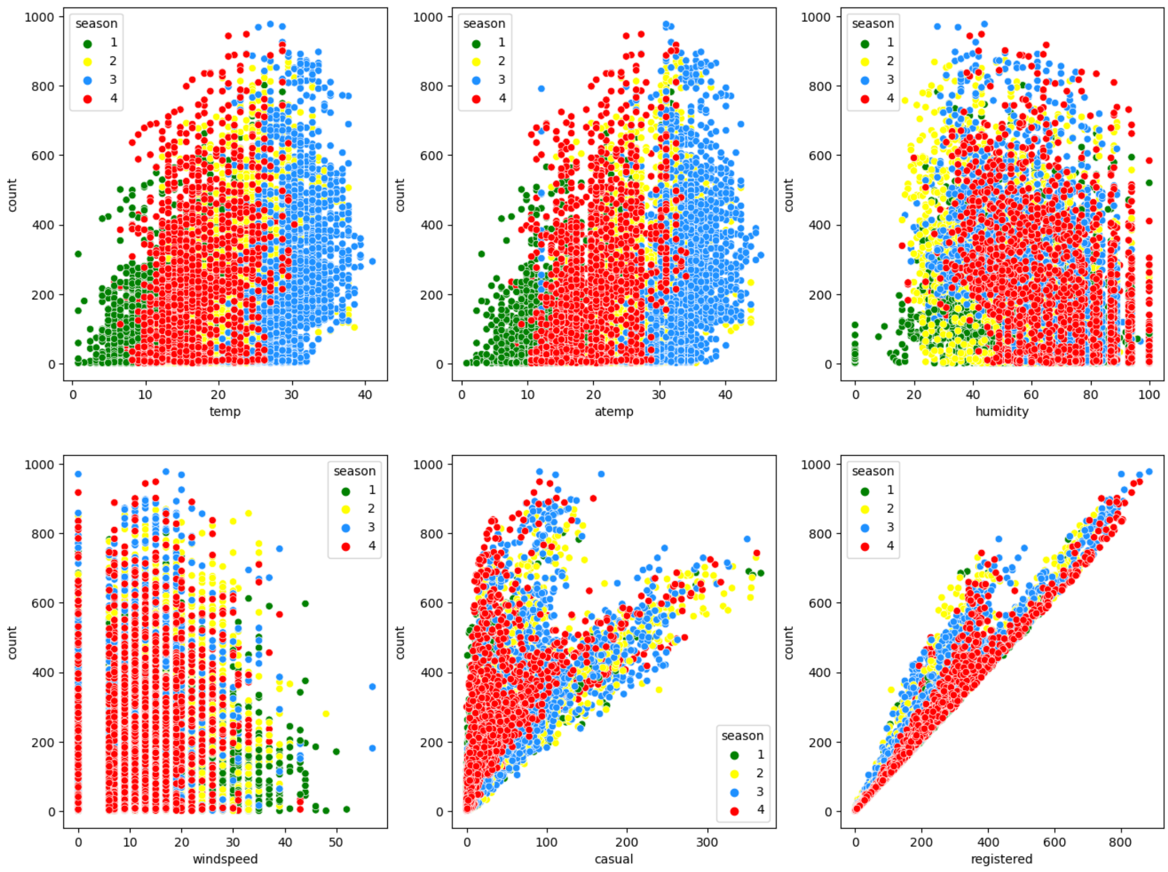
Bi-variate Analysis

In [18]:
```
# plotting categorical variables againt count using boxplots
fig, axis = plt.subplots(nrows=2, ncols=2, figsize=(16, 12))

index = 0
for row in range(2):
    for col in range(2):
        sns.boxplot(data=df, x=cat_cols[index], y='count', ax=axis[row, col])
        index += 1

plt.show()
```



- In **summer** and **fall** seasons more bikes are rented as compared to other seasons.
- Whenever its a **holiday** more bikes are rented.
- It is also clear from the workingday also that whenever day is holiday or weekend, slightly more bikes were rented.
- Whenever there is **rain, thunderstorm, snow or fog**, there were less bikes were rented.

In [19]:
```
# plotting numerical variables against count using scatterplot
fig, axis = plt.subplots(nrows=2, ncols=3, figsize=(16, 12))

index = 0
for row in range(2):
    for col in range(3):
        sns.scatterplot(data=df, x=num_cols[index], y='count', ax=axis[row, col]
                        palette=['green','yellow','dodgerblue','red'])
        index += 1

plt.show()
```



- Whenever the humidity is less than 20, number of bikes rented is very very low.
- Whenever the temperature is less than 10, number of bikes rented is less.
- Whenever the windspeed is greater than 35, number of bikes rented is less.

In [20]:
```
# understanding the correlation between count and numerical variables
df.corr()['count']
```

```
C:\Users\deepa\AppData\Local\Temp\ipykernel_21588\3391090118.py:2: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a future ve
rsion, it will default to False. Select only valid columns or specify the value o
f numeric_only to silence this warning.
```

Out[20]:
```
temp          0.394454
atemp         0.389784
humidity     -0.317371
windspeed     0.101369
casual        0.690414
registered    0.970948
count         1.000000
Name: count, dtype: float64
```

In [21]:
```
#sns.heatmap(df.corr(), annot=True)
#plt.show()
```

Correlation between variables

In [22]:
```
round(df.corr(),2) #rounding-off correlation between our variables
```

```
C:\Users\deepa\AppData\Local\Temp\ipykernel_21588\322788953.py:1: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a future ve
rsion, it will default to False. Select only valid columns or specify the value o
f numeric_only to silence this warning.
```

Out[22]:

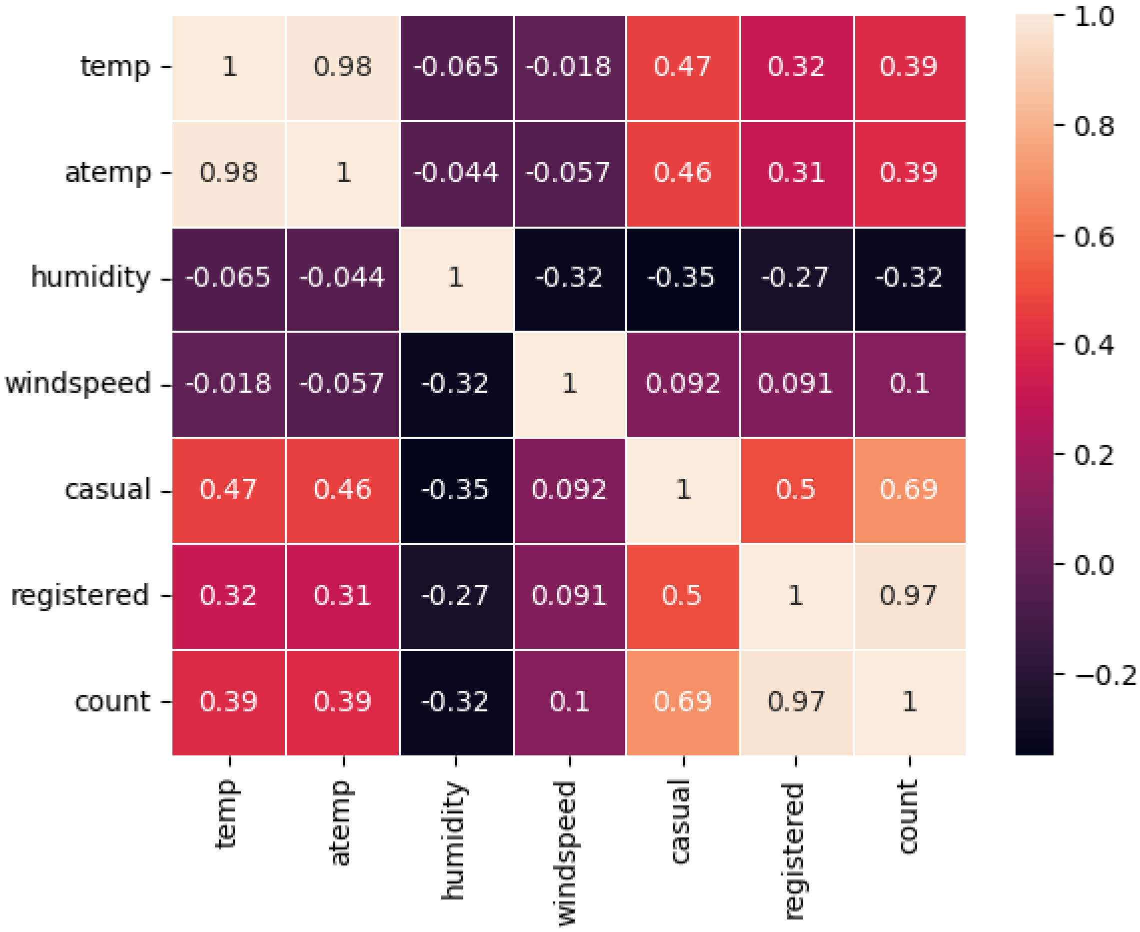
	temp	atemp	humidity	windspeed	casual	registered	count
temp	1.00	0.98	-0.06	-0.02	0.47	0.32	0.39
atemp	0.98	1.00	-0.04	-0.06	0.46	0.31	0.39
humidity	-0.06	-0.04	1.00	-0.32	-0.35	-0.27	-0.32
windspeed	-0.02	-0.06	-0.32	1.00	0.09	0.09	0.10
casual	0.47	0.46	-0.35	0.09	1.00	0.50	0.69
registered	0.32	0.31	-0.27	0.09	0.50	1.00	0.97
count	0.39	0.39	-0.32	0.10	0.69	0.97	1.00

In [23]:

```
sns.heatmap(df.corr(),linewidths=0.5,annot=True)
```

```
C:\Users\deepa\AppData\Local\Temp\ipykernel_21588\2383223086.py:1: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a future ve
rsion, it will default to False. Select only valid columns or specify the value o
f numeric_only to silence this warning.
```

Out[23]: <Axes: >

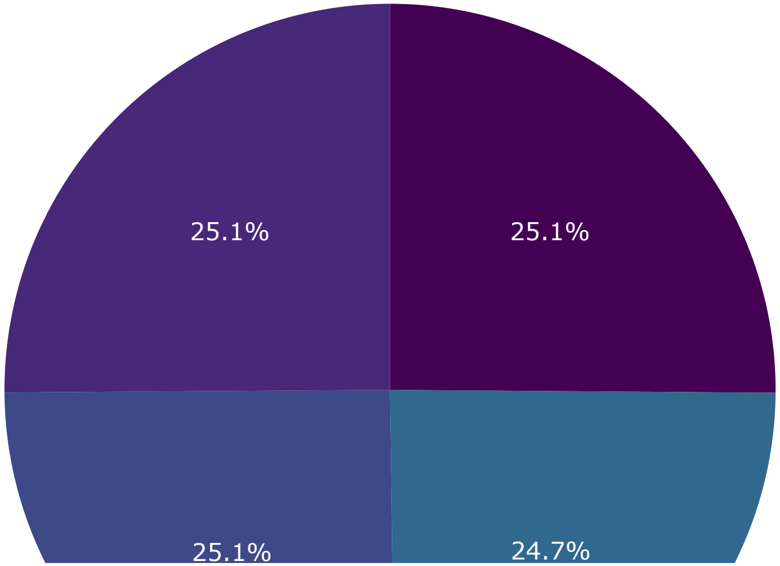


We can see for the "count"

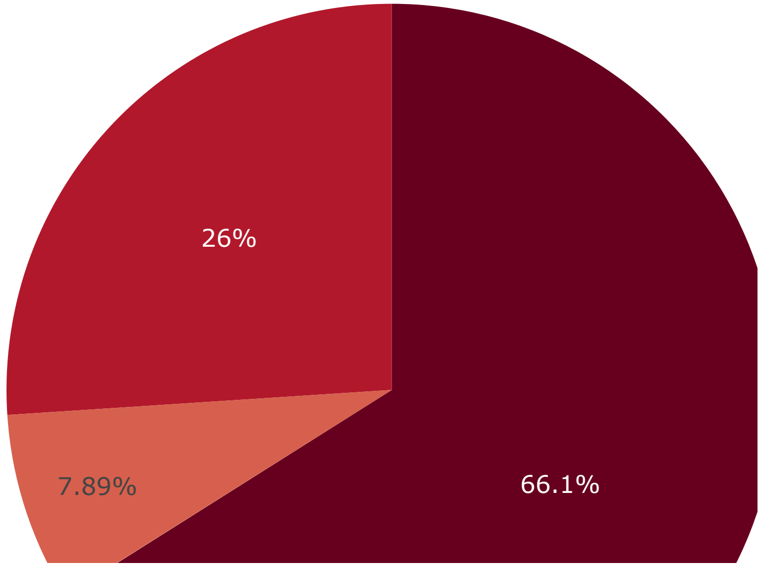
- There is strong positive correlation between count with "casual(0.72)" and "rental(0.91)" which is obvious as it is sum of those two variables
- There is also somewhat a strong positive relation between "count" and "temp(0.39)/atemp(0.38)" which suggests that as the temperature increases, the count also increases
- There is somewhat a strong positive relation (0.43) between "count" and "hour"
- "count" and "humidity" have a negative correlation of -0.32. This means that as humidity increases, the bike rental count tends to decrease.
- This is ofcourse just a first glance, we have to perform various statistical tests in order to actually confirm our findings

Data Visualiztion

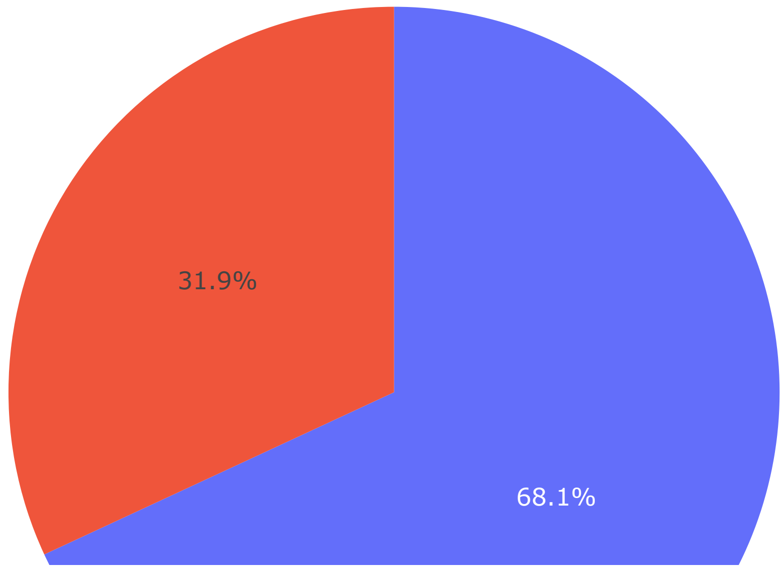
In [24]:
```
#colors = ['gold', 'mediumturquoise', 'darkorange', 'lightgreen']
px.pie(df,names='season',color_discrete_sequence=px.colors.sequential.Viridis)
```



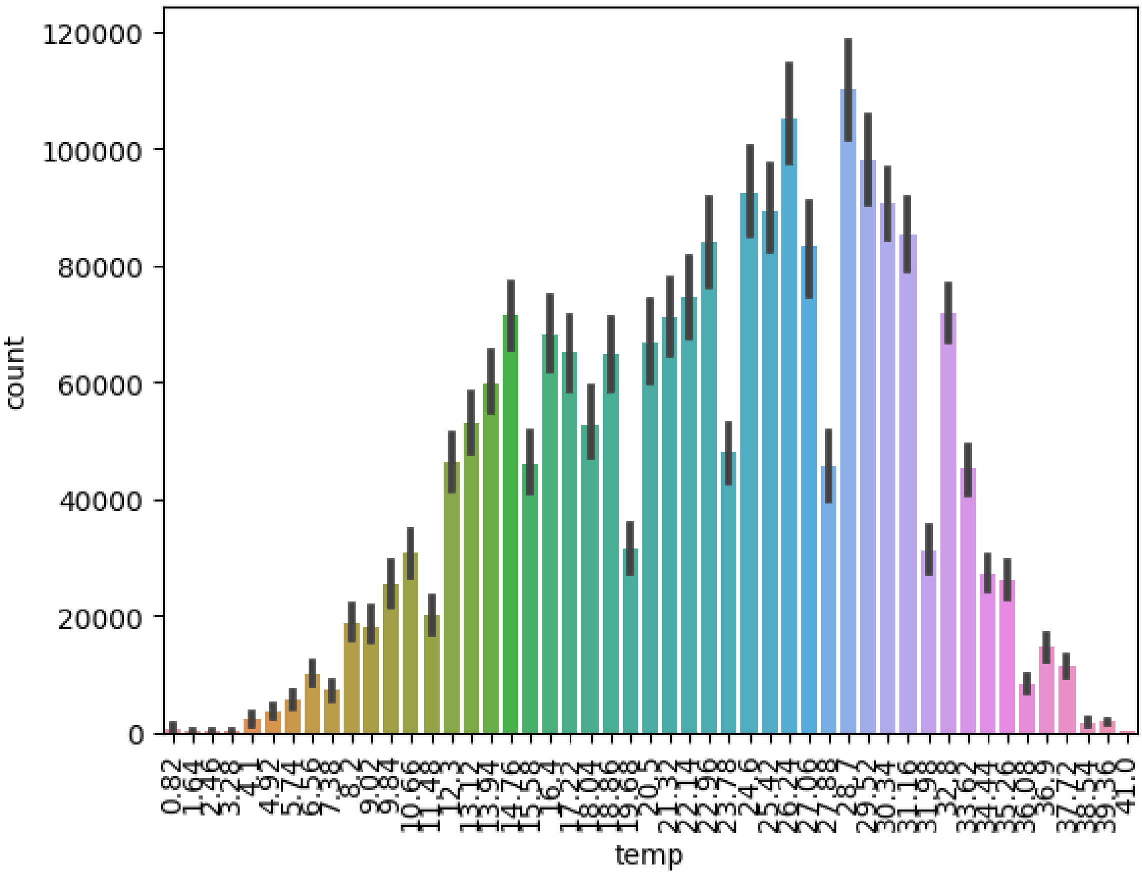

```
In [25]: px.pie(df,names='weather',color_discrete_sequence=px.colors.sequential.RdBu)
```



In [26]:
```
px.pie(df,names='workingday')
```



In [27]:
```
fig = sns.barplot(df,y='count',x='temp',estimator='sum')
fig.set_xticklabels(fig.get_xticklabels(), rotation=90)
plt.show()
```



Hypothesis Testing - ANOVA

Null Hypothesis: Number of cycles rented is similar in different weather and season.

Alternate Hypothesis: Number of cycles rented is not similar in different weather and season.

Significance level (alpha): 0.05

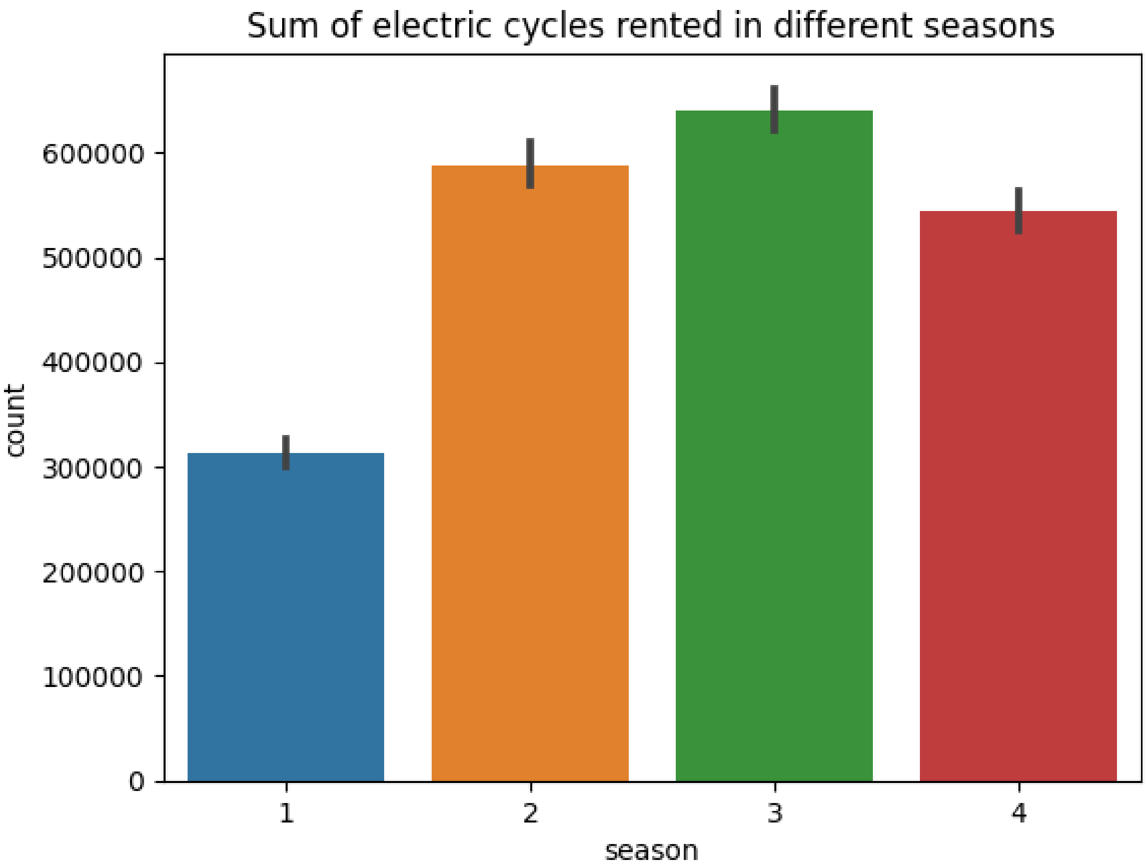
Here, we will use the **ANOVA** to test the hypothess defined above

In [28]:
```
# defining the data groups for the ANOVA

gp1 = df[df['weather']==1]['count'].values
gp2 = df[df['weather']==2]['count'].values
gp3 = df[df['weather']==3]['count'].values
gp4 = df[df['weather']==4]['count'].values

gp5 = df[df['season']==1]['count'].values
gp6 = df[df['season']==2]['count'].values
gp7 = df[df['season']==3]['count'].values
gp8 = df[df['season']==4]['count'].values

# conduct the one-way anova
stats.f_oneway(gp1, gp2, gp3, gp4, gp5, gp6, gp7, gp8)
```

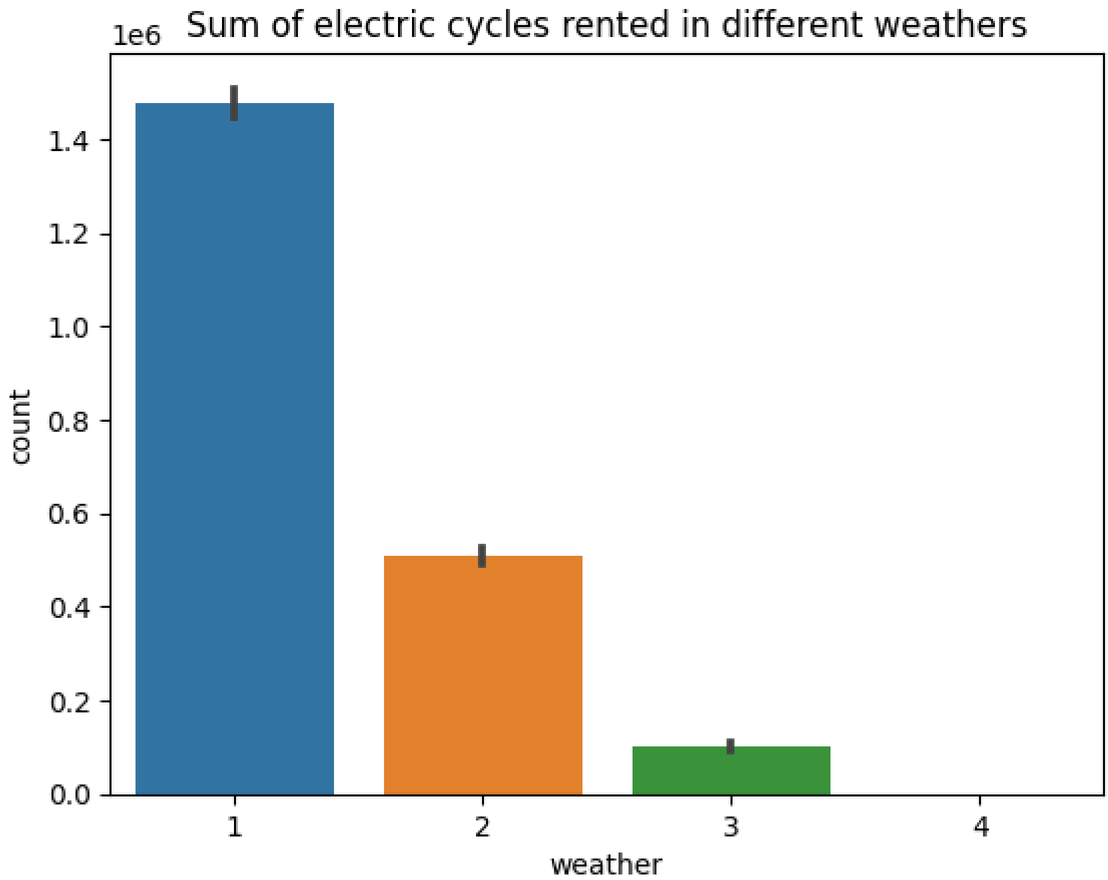
Out[28]:
```
F_onewayResult(statistic=127.96661249562491, pvalue=2.8074771742434642e-185)
```

Since p-value is less than 0.05, we reject the null hypothesis. This implies that Number of cycles rented is not similar in different weather and season conditions

In [29]:
```
# To visualize the dependence of the cycle rentals on the seasons
sns.barplot(df,y='count',x='season',estimator='sum')
plt.title("Sum of electric cycles rented in different seasons")
plt.show()
```



In [30]:
```
# To visualize the dependence of the cycle rentals on the weather
sns.barplot(df,y='count',x='weather',estimator='sum')
plt.title("Sum of electric cycles rented in different weathers")
plt.show()
```



In [31]:
```
# defining the data groups for the ANOVA

gp1 = df[df['weather']==1]['count'].values
gp2 = df[df['weather']==2]['count'].values
gp3 = df[df['weather']==3]['count'].values
gp4 = df[df['weather']==4]['count'].values

gp5 = df[df['season']==1]['count'].values
gp6 = df[df['season']==2]['count'].values
gp7 = df[df['season']==3]['count'].values
gp8 = df[df['season']==4]['count'].values

# conduct the one-way anova
weather_seasons_combo=stats.f_oneway(gp1, gp2, gp3, gp4, gp5, gp6, gp7, gp8)
weather_only=stats.f_oneway(gp1, gp2, gp3, gp4)
seasons_only=stats.f_oneway( gp5, gp6, gp7, gp8)
print(weather_seasons_combo)
print(seasons_only)
print(weather_only)
```

```
F_onewayResult(statistic=127.96661249562491, pvalue=2.8074771742434642e-185)
F_onewayResult(statistic=236.94671081032106, pvalue=6.164843386499654e-149)
F_onewayResult(statistic=65.53024112793271, pvalue=5.482069475935669e-42)
```

In [32]:
```
################################################################################
```

Hypothesis Testing - **chi-square test**

Null Hypothesis (H0): Weather is independent of the season

Alternate Hypothesis (H1): Weather is not independent of the season

Significance level (alpha): 0.05

We will use **chi-square test** to test hypyothesis defined above.

In [33]:
```
data_table = pd.crosstab(df['season'], df['weather'])
print("Observed values:")
data_table
```

```
Observed values:
```

Out[33]:

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 [34]:
```
val = stats.chi2_contingency(data_table)
expected_values = val[3]
expected_values
```

Out[34]:
```
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 [35]:
```
nrows, ncols = 4, 4
dof = (nrows-1)*(ncols-1)
print("degrees of freedom: ", dof)
alpha = 0.05


chi_sqr = sum([(o-e)**2/e for o, e in zip(data_table.values, expected_values)])
chi_sqr_statistic = chi_sqr[0] + chi_sqr[1]
print("chi-square test statistic: ", chi_sqr_statistic)

critical_val = stats.chi2.ppf(q=1-alpha, df=dof)
print(f"critical value: {critical_val}")

p_val = 1-stats.chi2.cdf(x=chi_sqr_statistic, df=dof)
print(f"p-value: {p_val}")

if p_val <= alpha:
    print("\nSince p-value is less than the alpha 0.05, We reject the Null Hypot
    Weather is dependent on the season.")
else:
    print("Since p-value is greater than the alpha 0.05, We do not reject the Nu
```

```
degrees of freedom:  9
chi-square test statistic:  44.09441248632364
critical value: 16.918977604620448
p-value: 1.3560001579371317e-06

Since p-value is less than the alpha 0.05, We reject the Null Hypothesis. Meaning
that    Weather is dependent on the season.
```

- Since p-value is less than the alpha 0.05, We reject the Null Hypothesis. Meaning that Weather is dependent on the season.

In [36]:
```
#################################################################################
```

Hypothesis Testing - 2-Sample T-Test

Null Hypothesis: Working day has no effect on the number of cycles being rented.

Alternate Hypothesis: Working day has effect on the number of cycles being rented.

Significance level (alpha): 0.05

We will use the **2-Sample T-Test** to test the hypothess defined above

In [37]:
```
data_group1 = df[df['workingday']==0]['count'].values
data_group2 = df[df['workingday']==1]['count'].values

np.var(data_group1), np.var(data_group2)
```

Out[37]:
```
(30171.346098942427, 34040.69710674686)
```

Before conducting the two-sample T-Test we need to find if the given data groups have the same variance. If the ratio of the larger data groups to the small data group is less than 4:1 then we can consider that the given data groups have equal variance.

Here, the ratio is 34040.70 / 30171.35 which is less than 4:1

In [38]:
```
stats.ttest_ind(a=data_group1, b=data_group2, equal_var=True)
```

Out[38]:
```
Ttest_indResult(statistic=-1.2096277376026694, pvalue=0.22644804226361348)
```

Since pvalue is greater than 0.05 so we can not reject the Null hypothesis. We don't have the sufficient evidence to say that working day has effect on the number of cycles being rented.

In [39]:
```
##################################################################################
```

Insights

- In **summer** and **fall** seasons more bikes are rented as compared to other seasons.
- Whenever its a **holiday** more bikes are rented.

- It is also clear from the workingday also that whenever day is holiday or weekend, slightly more bikes were rented.
- Whenever the temperature is less than 10, number of bikes rented is less.
- Whenever there is **rain, thunderstorm, snow or fog**, there were less bikes were rented.
- Whenever the humidity is less than 20, number of bikes rented is very very low.
- Whenever the windspeed is greater than 35, number of bikes rented is less.

Recommendations

- With a significance level of 0.05, **workingday** has no effect on the number of bikes being rented.
- In very low humid days, company should have less bikes in the stock to be rented.
- Whenever temprature is less than 10 or in very cold days, company should have less bikes.
- Whenever the windspeed is greater than 35 or in thunderstorms, company should have less bikes in stock to be rented.
- In **summer** and **fall** seasons the company should have more bikes in stock to be rented. Because the demand in these seasons is higher as compared to other seasons.

In [ ]: