In [308…
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm,ttest_1samp,ttest_ind,ttest_rel,f_oneway,chisquare,chi2_c
from scipy.stats import kstest,levene,kruskal,shapiro

import warnings
warnings.filterwarnings('ignore')
```

In [309…
```
def bold_text(text):
    bold_start = '\033[1m'
    bold_end = '\033[0m'
    return bold_start + text + bold_end
```

Define Problem Statement and perform Exploratory Data Analysis

Definition of problem

Yulu is India's leading micro-mobility service provider, which offers unique vehicles for the daily commute. Yulu has recently suffered considerable dips in its revenues.They want to understand the factors affecting the demand for these shared electric cycles in the Indian market.

The company wants to know:

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

Observations on shape of data, data types of all the attributes

In [310…
```
df = pd.read_csv('Yulu.csv')
```

In [311…
```
df.shape
```

Out[311]:
```
(10886, 12)
```

In [312…
```
df.head()
```

Out[312]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	reg
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	

In [313…

```
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 [314…

```
df.describe()
```

Out[314]:

	season	holiday	workingday	weather	temp	atemp	humid
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000	10886.0000
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.8864
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.2450
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	0.0000
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	47.0000
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	62.0000
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	77.0000
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	100.0000

In [315…

```
df.datetime = pd.to_datetime(df.datetime)

df['date'] = df['datetime'].dt.date
df['month'] = df['datetime'].dt.month
df['hour'] = df['datetime'].dt.hour
```

In [316…

```
cols_list =['season','holiday','workingday']
```

In [317…

```
days_df = df[['date','season','holiday','workingday']].drop_duplicates(keep = 'fir
df['windspeed_bins'] = pd.cut(df['windspeed'],bins = [-0.0000001,10,20,30,40,50,60],la


for i in cols_list:
    print(bold_text(i.upper()+':'))
    print(f'Value Counts of {i} columns is:\n{days_df[i].value_counts()}\n\n')

print(bold_text('weather'.upper()+':'))
print(f'Value Counts of weather columns is:\n{df["weather"].value_counts()}\n\n')


print(bold_text('humidity'.upper()+':'))
print(f'Value Counts of humidity columns is:\n{df["humidity"].value_counts()}\n\n')

print(bold_text('windspeed'.upper()+':'))
print(f'Value Counts of windspeed columns is:\n{df["windspeed_bins"].value_counts()}\r
```

SEASON:

```
Value Counts of season columns is:
1    114
2    114
3    114
4    114
Name: season, dtype: int64
```

HOLIDAY:

```
Value Counts of holiday columns is:
0    443
1     13
Name: holiday, dtype: int64
```

WORKINGDAY:

```
Value Counts of workingday columns is:
1    311
0    145
Name: workingday, dtype: int64
```

WEATHER:

```
Value Counts of weather columns is:
1    7192
2    2834
3     859
4       1
Name: weather, dtype: int64
```

HUMIDITY:

```
Value Counts of humidity columns is:
88    368
94    324
83    316
87    289
70    259
     ...
8       1
10      1
97      1
96      1
91      1
Name: humidity, Length: 89, dtype: int64
```

WINDSPEED:

```
Value Counts of windspeed columns is:
10-20    5052
0-10     4339
20-30    1068
30-40     387
40-50      36
50-60       4
Name: windspeed_bins, dtype: int64
```

In [318…
```
print("The dataset was recorded over",bold_text(str(df['date'].nunique())),"days.")
```

```
The dataset was recorded over 456 days.
```

In [319…
```
df.isna().sum()
```

Out[319]:
```
datetime          0
season            0
holiday           0
workingday        0
weather           0
temp              0
atemp             0
humidity          0
windspeed         0
casual            0
registered        0
count             0
date              0
month             0
hour              0
windspeed_bins    0
dtype: int64
```

In [320…
```
df.duplicated().sum()
```

Out[320]:
```
0
```

In [321…
```
new_df = df.groupby('date')['date'].count()
new_df.index[new_df<24]
```

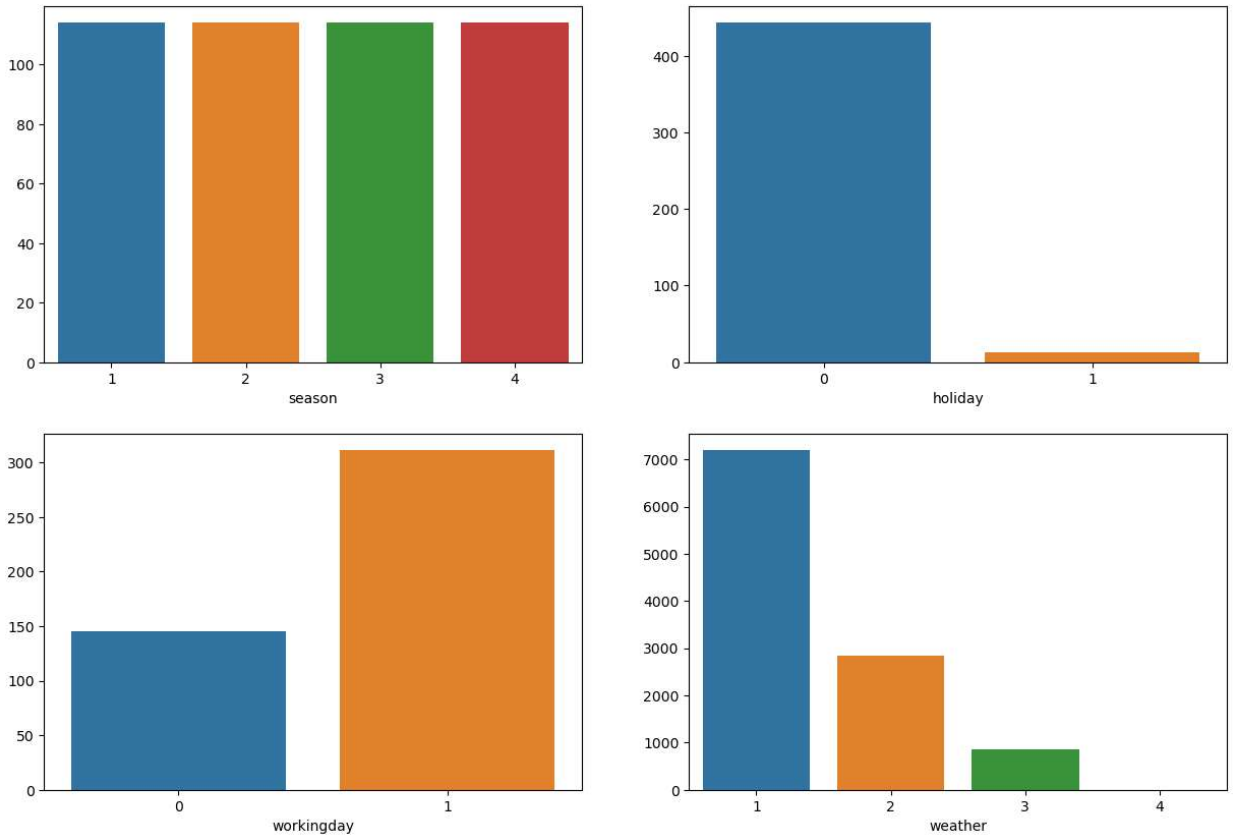
Out[321]:
```
Index([2011-01-02, 2011-01-03, 2011-01-04, 2011-01-05, 2011-01-06, 2011-01-07,
       2011-01-11, 2011-01-12, 2011-01-14, 2011-01-18, 2011-01-19, 2011-02-01,
       2011-02-03, 2011-02-04, 2011-02-09, 2011-02-10, 2011-02-11, 2011-02-13,
       2011-02-15, 2011-02-16, 2011-03-06, 2011-03-07, 2011-03-10, 2011-03-11,
       2011-03-13, 2011-03-14, 2011-03-15, 2011-03-16, 2011-03-18, 2011-04-11,
       2011-09-06, 2011-09-08, 2011-09-12, 2011-10-19, 2012-01-02, 2012-01-10,
       2012-01-17, 2012-02-06, 2012-03-11, 2012-04-02, 2012-04-11, 2012-11-08],
      dtype='object', name='date')
```

Univariate Analysis

In [322…
```
cols_list =['season','holiday','workingday','weather']
```

In [323…
```
plt.figure(figsize=(15,5*2))
plt.subplot(2,2,1)
sns.countplot(data = days_df,x = 'season')

plt.ylabel('')

plt.subplot(2,2,2)
sns.countplot(data = days_df,x = 'holiday')
plt.ylabel('')

plt.subplot(2,2,3)
sns.countplot(data = days_df,x = 'workingday')
plt.ylabel('')
```

```
plt.subplot(2,2,4)
sns.countplot(data = df,x = 'weather')
plt.ylabel('')

plt.show()
```



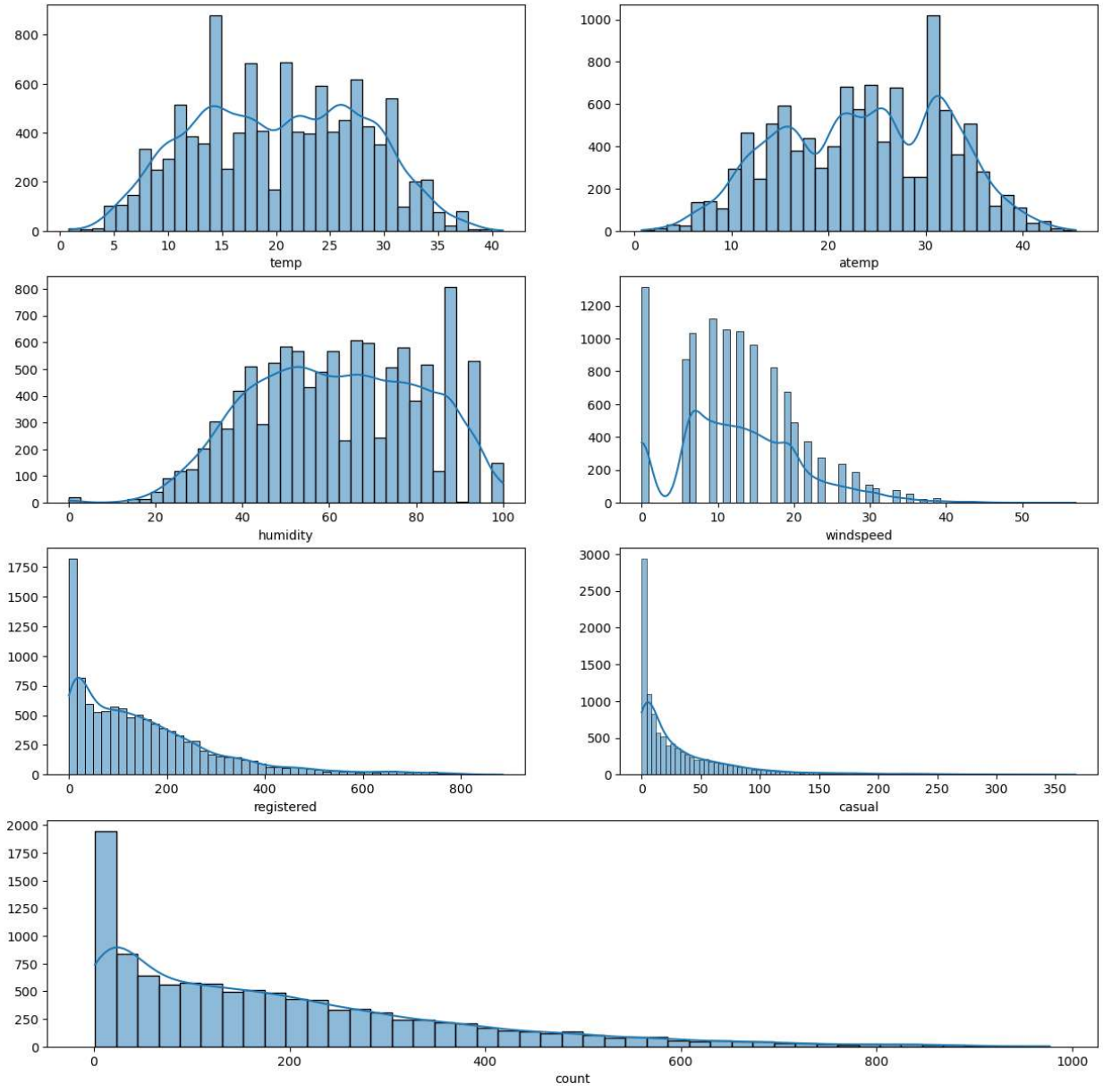
Observations:

1. We have dataset which is equally divided among all seasons
2. This dataset has very few holiday records.
3. Most of the records are of workingday.
4. Most of the records are of weather1.

In [324…

```
plt.figure(figsize=(15,5*3))

plt.subplot(4,2,1)
sns.histplot(data = df,x = 'temp',kde=True)
plt.ylabel('')

plt.subplot(4,2,2)
sns.histplot(data = df,x = 'atemp',kde=True)
plt.ylabel('')

plt.subplot(4,2,3)
sns.histplot(data = df,x = 'humidity',kde=True)
plt.ylabel('')

plt.subplot(4,2,4)
sns.histplot(data = df,x = 'windspeed',kde=True)
plt.ylabel('')
```

```
plt.subplot(4,2,5)
sns.histplot(data = df,x = 'registered',kde=True)
plt.ylabel('')

plt.subplot(4,2,6)
sns.histplot(data = df,x = 'casual',kde=True)
plt.ylabel('')

plt.subplot(4,1,4)
sns.histplot(data = df,x = 'count',kde=True)
plt.ylabel('')

plt.show()
```

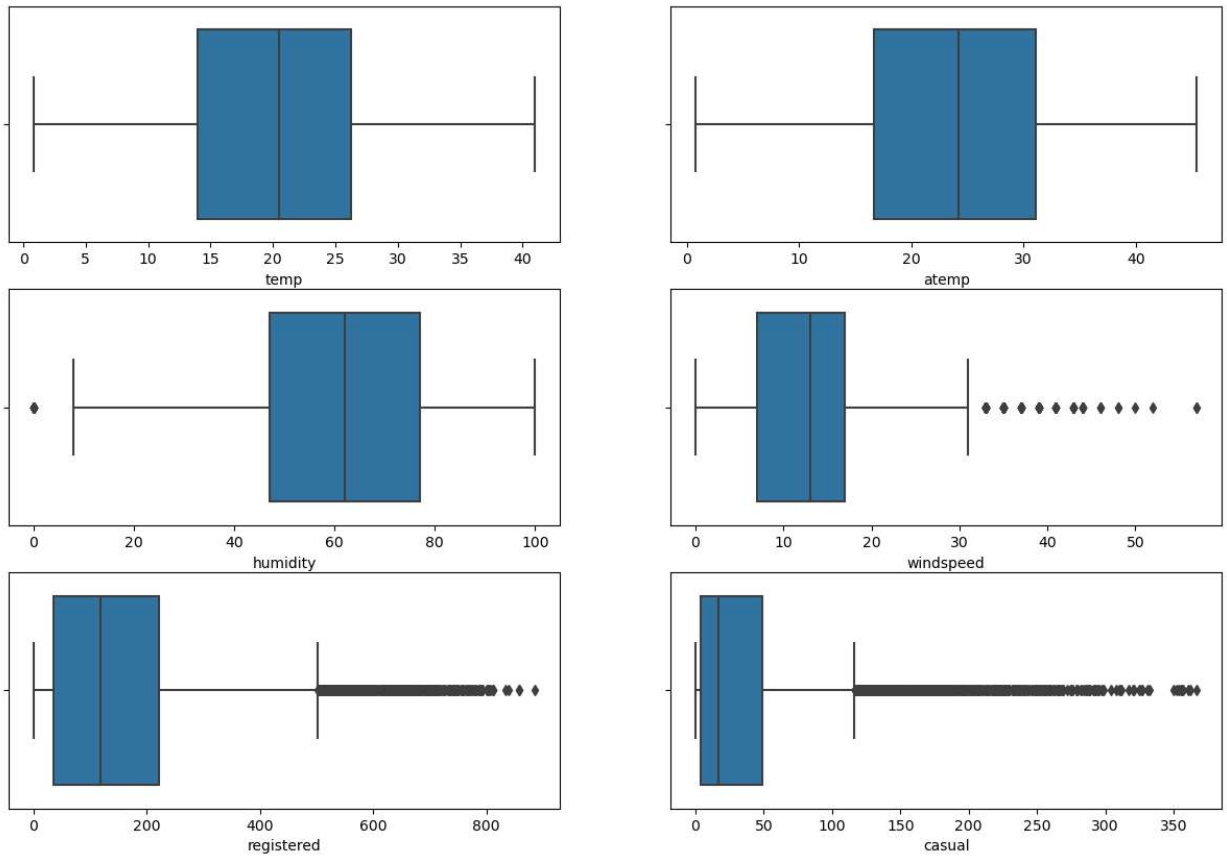


Observations

1. Registered, Casual and Count are Right Skewed, which seems valid as there's less probability of having high number of rides

In [325…

```
plt.figure(figsize=(15,5*2))

plt.subplot(3,2,1)
sns.boxplot(data = df,x = 'temp',)

plt.subplot(3,2,2)
sns.boxplot(data = df,x = 'atemp')

plt.subplot(3,2,3)
sns.boxplot(data = df,x = 'humidity')

plt.subplot(3,2,4)
sns.boxplot(data = df,x = 'windspeed')

plt.subplot(3,2,5)
sns.boxplot(data = df,x = 'registered')

plt.subplot(3,2,6)
sns.boxplot(data = df,x = 'casual')

plt.show()
```

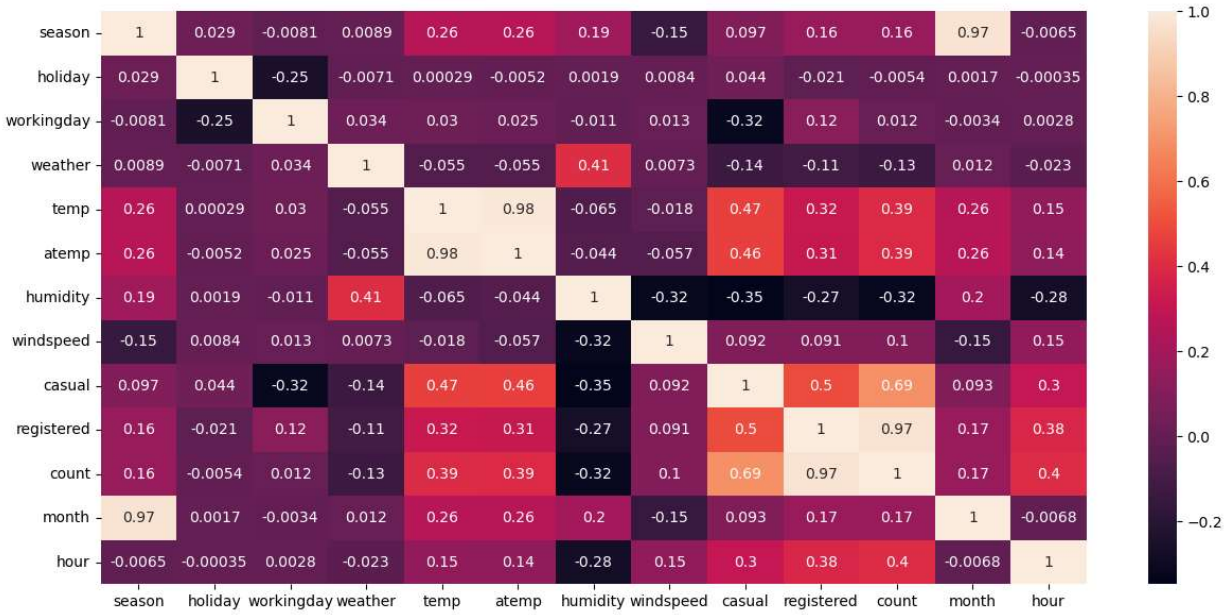


Observations

1. We can see some outliers for casual, windspeed and registered.

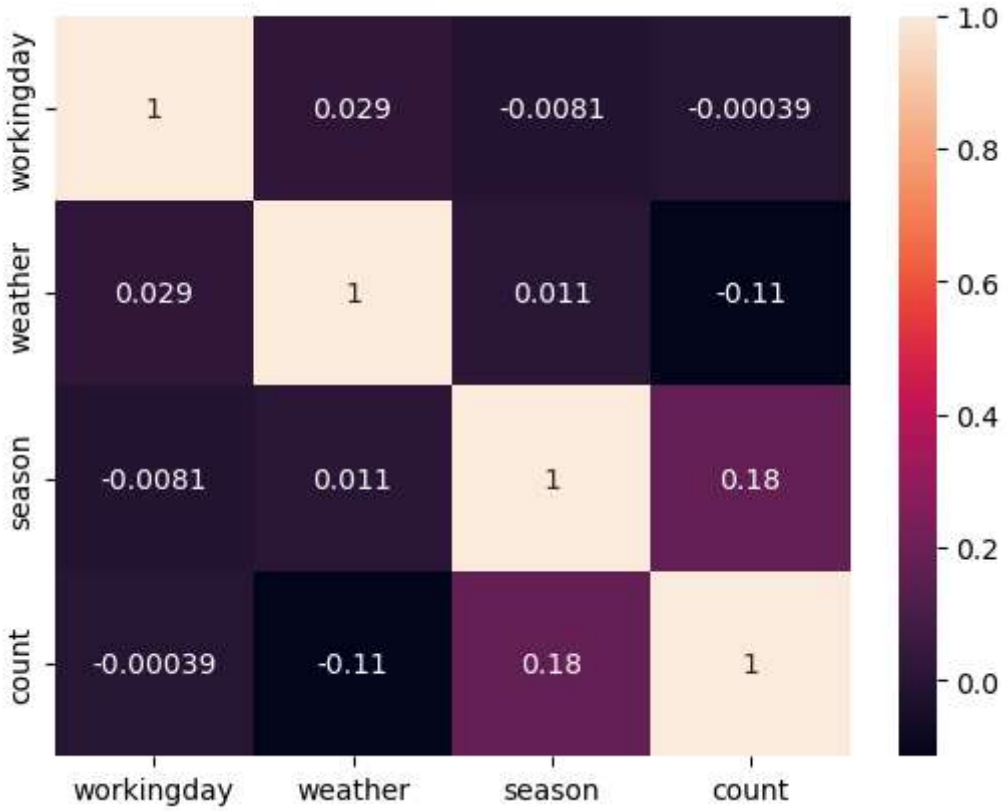
In [326…

```
plt.figure(figsize = (15,7))
sns.heatmap(df.corr(),annot = True)
plt.show()
```



In [327…

```
sns.heatmap(df[['workingday','weather','season','count']].corr(method = 'spearman'
plt.show()
```



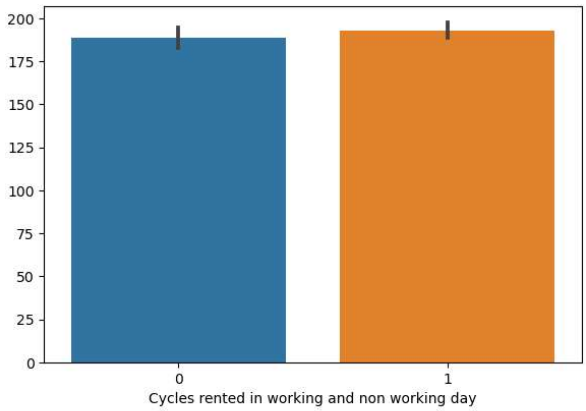
Observations

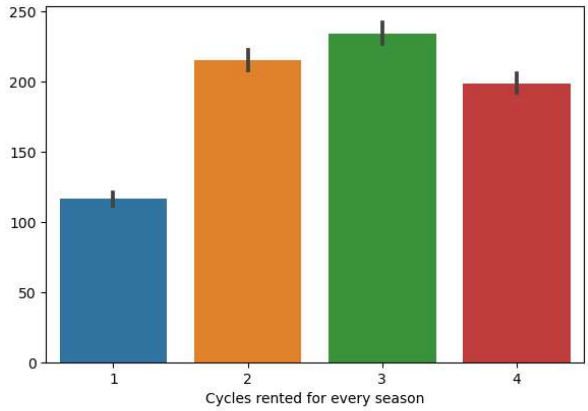
1. Count is highly correlated with registered.
2. Count is correlated with casual.
3. Count is lowly correlated with temp and atemp.
4. Count is negatively low correlated with weather and humidity.
5. Count is not correlated(independent) of holiday, workingday, windspeed.

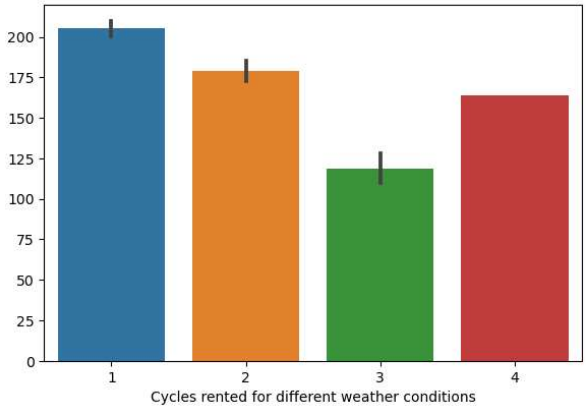
6. Temp and Atemp are highly correlated.

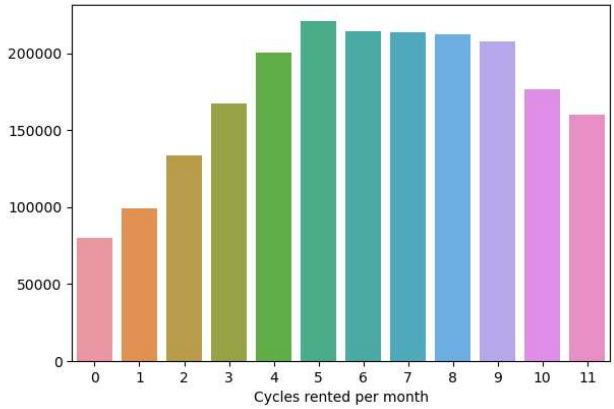
In [328…

```
month_count = df.groupby('month')['count'].sum().reset_index(drop = True)

plt.figure(figsize = (15,5*2))

plt.subplot(2,2,1)
sns.barplot(data =df, x = 'workingday', y = 'count')
plt.ylabel('')
plt.xlabel('Cycles rented in working and non working day')

plt.subplot(2,2,2)
sns.barplot(data =df, x = 'season', y = 'count')
plt.ylabel('')
plt.xlabel("Cycles rented for every season")

plt.subplot(2,2,3)
sns.barplot(data =df, x = 'weather', y = 'count')
plt.ylabel('')
plt.xlabel('Cycles rented for different weather conditions')

plt.subplot(2,2,4)
sns.barplot(x = month_count.index, y = month_count)
plt.ylabel('')
plt.xlabel('Cycles rented per month')

plt.show()
```









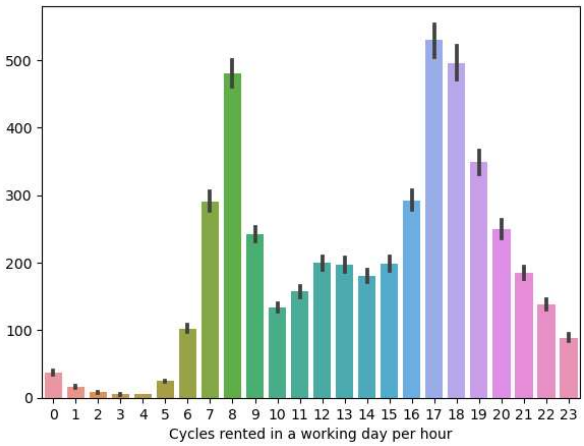
Observations

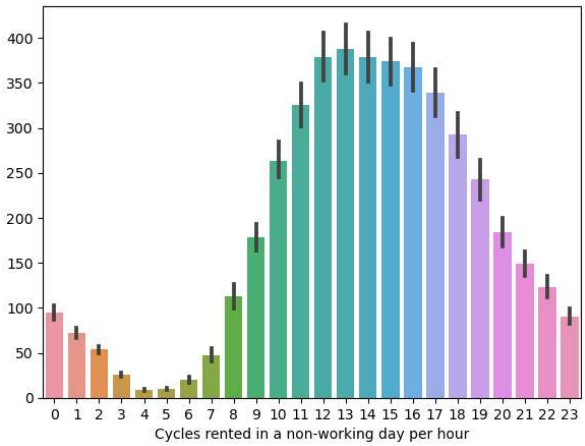
1. Most cycles were rented in weather 1

2. Most cycles were rented season 3.
3. We can observe some seasonality, most of the cycles were rented from 4th month to 9th month

In [329…

```
working = df[df['workingday'] == 1]
nonworking = df[df['workingday'] == 0]

plt.figure(figsize = (15,5))

plt.subplot(1,2,1)
sns.barplot(data = working, x = 'hour',y='count')
plt.xlabel('Cycles rented in a working day per hour')
plt.ylabel('')

plt.subplot(1,2,2)
sns.barplot(data = nonworking, x = 'hour',y='count')
plt.xlabel('Cycles rented in a non-working day per hour')
plt.ylabel('')

plt.show()
```





Observations

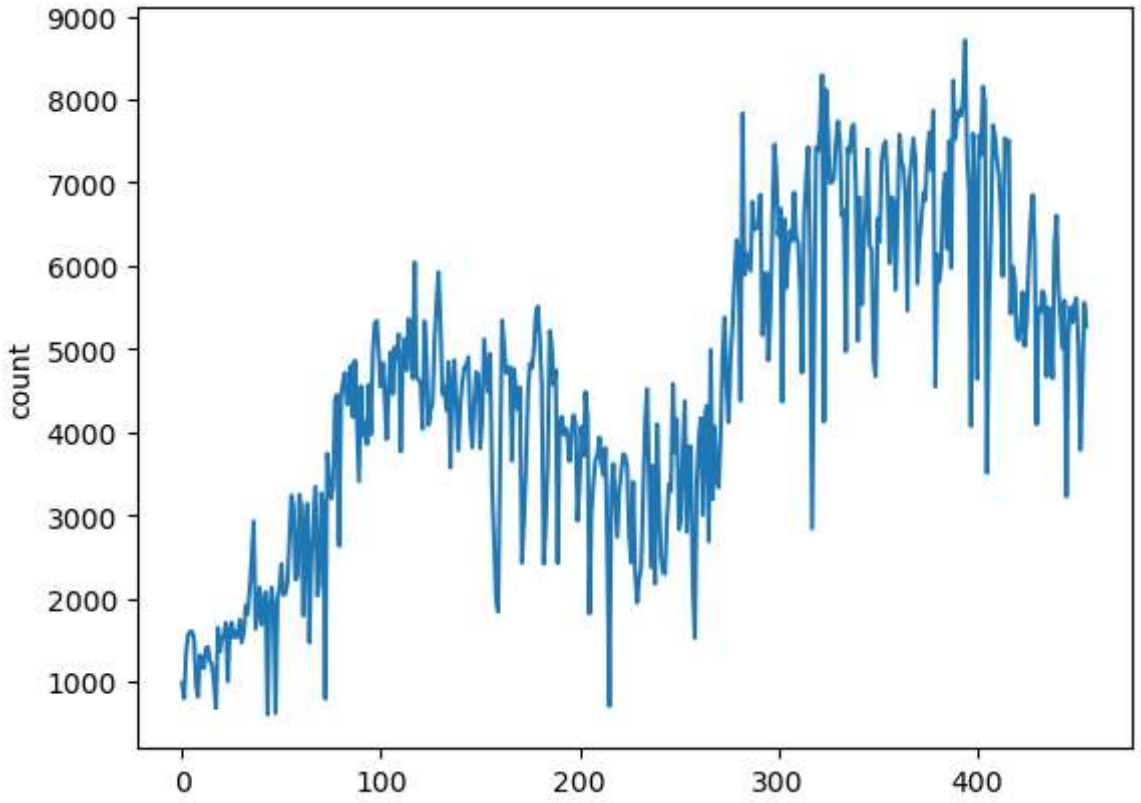
1. In a workingday most of the cycles were rented in hour 8,17 and 18.
2. In a non-working day most cycless were rened between 12th to 16th hour

In [330…

```
cycles_count = df.groupby('date')['count'].sum().reset_index(drop = True)
```

In [331…

```
sns.lineplot(x = cycles_count.index,y = cycles_count);
```



Observation

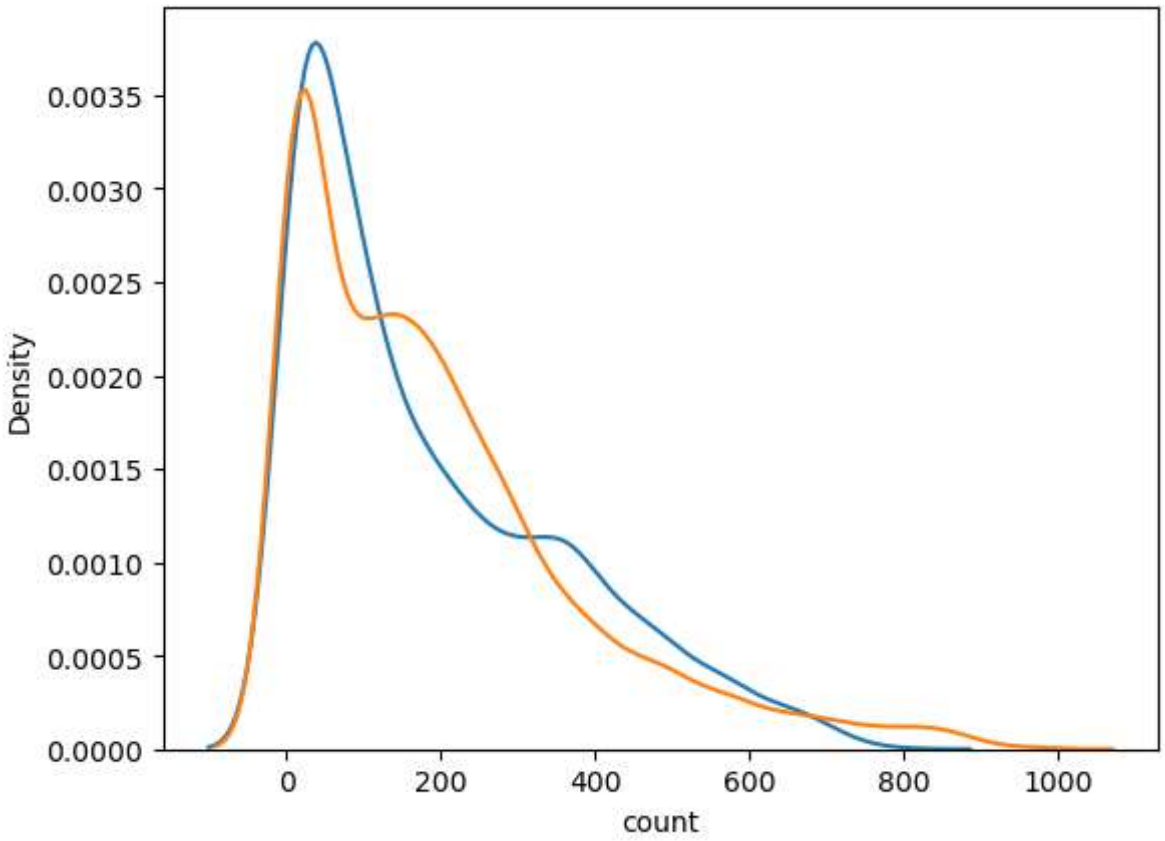
1. It can be observed that no of cycles rented are growing.

Hypothesis Testing

Working Day has effect on number of electric cycles rented

In [332…
```
sns.kdeplot(df.loc[df['workingday'] == 0,'count'])
sns.kdeplot(df.loc[df['workingday'] == 1,'count'])
```

Out[332]:
```
<Axes: xlabel='count', ylabel='Density'>
```



In [333…

```
Ho = 'The average number of cycles rented for working day is same as holiday'
Ha = 'The average number of cycles rented for working day is not same as holiday'
alpha = 0.05
t_stat,p_value = ttest_ind(df.loc[df['workingday'] == 0,'count'],df.loc[df['workingday
print("p_value:",p_value)
if p_value < alpha:
    print("Reject Ho, Interpretation:",bold_text(Ha))
else:
    print("Fail to Reject Ho, Interpretation:",bold_text(Ho))
```

```
p_value: 0.22644804226361348
Fail to Reject Ho, Interpretation: The average number of cycles rented for working da
y is same as holiday
```

In [334…

```
Ho = 'The average number of cycles rented for working day is same as holiday'
Ha = 'The average number of cycles rented for working day is not same as holiday'
alpha = 0.05
t_stat,p_value = kstest(df.loc[df['workingday'] == 0,'count'],df.loc[df['workingday']
print("p_value:",p_value)
if p_value < alpha:
    print("Reject Ho, Interpretation:",bold_text(Ha))
else:
    print("Fail to Reject Ho, Interpretation:",bold_text(Ho))
```

```
p_value: 8.003959300341833e-07
Reject Ho, Interpretation: The average number of cycles rented for working day is not
same as holiday
```

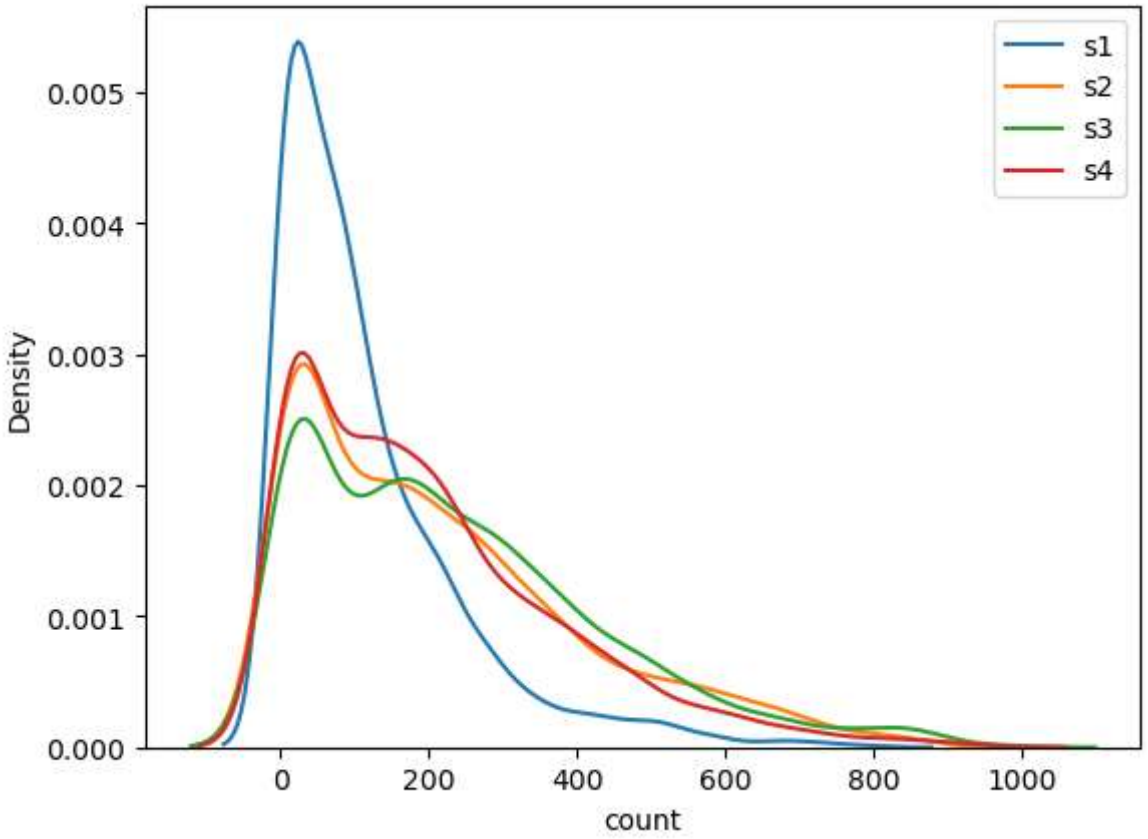
Conclusion:

1. The average number of cycles rented for working day is not same as holiday

No. of cycles rented similar or different in different seasons

In [335…

```
s1 =  df.loc[df['season'] == 1,'count']
s2 =  df.loc[df['season'] == 2,'count']
s3 =  df.loc[df['season'] == 3,'count']
s4 =  df.loc[df['season'] == 4,'count']

sns.kdeplot(df.loc[df['season'] == 1,'count'],label = 's1')
sns.kdeplot(df.loc[df['season'] == 2,'count'],label = 's2')
sns.kdeplot(df.loc[df['season'] == 3,'count'],label = 's3')
sns.kdeplot(df.loc[df['season'] == 4,'count'],label = 's4')
plt.legend()
plt.show()
```



In [336…

```
Ho = 'The average number of cycles rented for every season is same'
Ha = 'Atleast one season has different average number rented cycles'
alpha = 0.05
f_stat,p_value = f_oneway(s1,s2,s3,s4)
print("p_value:",p_value)
if p_value < alpha:
    print("Reject Ho, Interpretation:",bold_text(Ha))
else:
    print("Fail to Reject Ho, Interpretation:",bold_text(Ho))
```

```
p_value: 6.164843386499654e-149
Reject Ho, Interpretation: Atleast one season has different average number rented cyc
les
```

In [337…
```
print("Alpha =",alpha)
print("P_value for Shapiro's test season1 is",shapiro(s1)[1])
print("P_value for Shapiro's test season2 is",shapiro(s2)[1])
print("P_value for Shapiro's test season3 is",shapiro(s3)[1])
print("P_value for Shapiro's test season4 is",shapiro(s4)[1])
print("\nP_value for Levene's test: ", levene(s1,s2,s3,s4)[1])
```

```
Alpha = 0.05
P_value for Shapiro's test season1 is 0.0
P_value for Shapiro's test season2 is 6.039093315091269e-39
P_value for Shapiro's test season3 is 1.043458045587339e-36
P_value for Shapiro's test season4 is 1.1301682309549298e-39

P_value for Levene's test:  1.0147116860043298e-118
```

From the pvalues it is clearly visible that atleast one of the season has different variance, all the seasons are not of gaussian distirbution So we cannot perform `ANOVA`, we need to perform `KRUSKAL'S TEST` to test our hypothesis

In [338…
```
alpha = 0.05
k_stat,p_value = kruskal(s1,s2,s3,s4)
print("p_value:",p_value)
if p_value < alpha:
    print("Reject Ho, Interpretation:",bold_text(Ha))
else:
    print("Fail to Reject Ho, Interpretation:",bold_text(Ho))
```

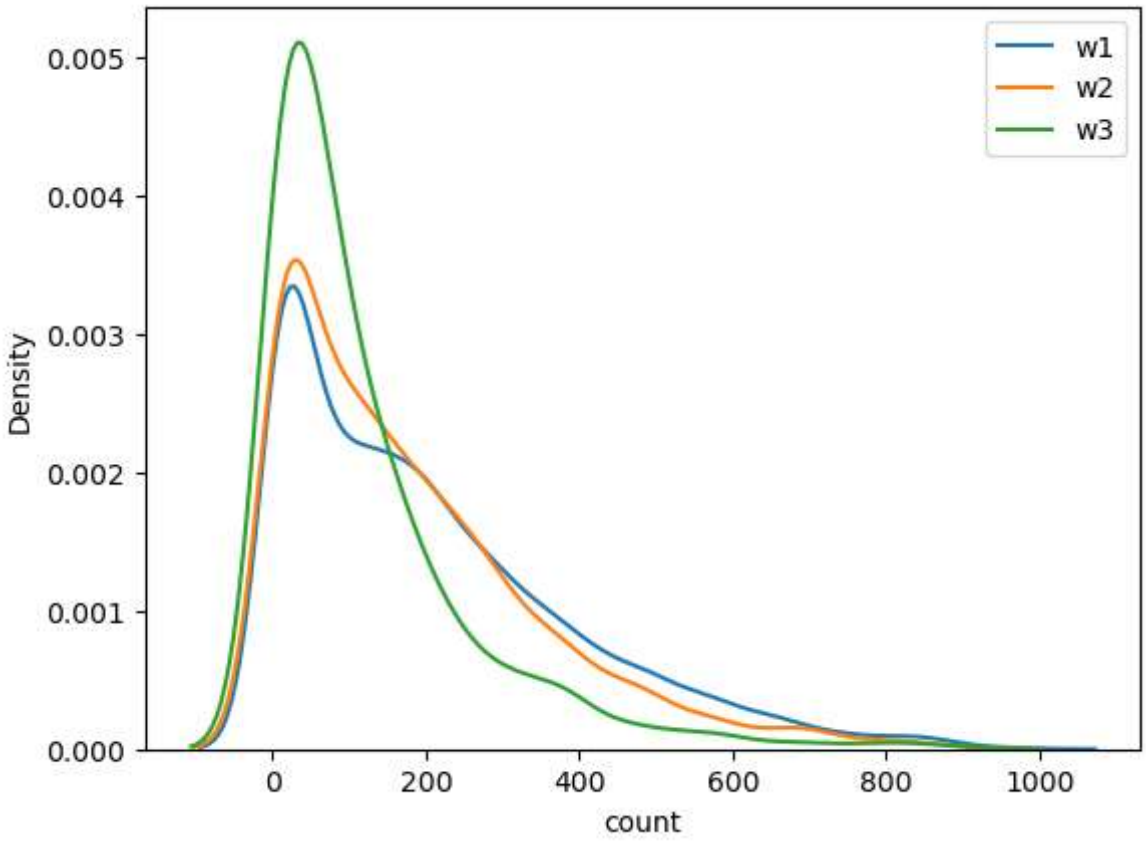
```
p_value: 2.479008372608633e-151
Reject Ho, Interpretation: Atleast one season has different average number rented cyc
les
```

Conclusion:

1. Atleast one season has different average number rented cycles

No. of cycles rented similar or different in different weather

In [339…
```
w1 =  df.loc[df['weather'] == 1,'count']
w2 =  df.loc[df['weather'] == 2,'count']
w3 =  df.loc[df['weather'] == 3,'count']
w4 =  df.loc[df['weather'] == 4,'count']

sns.kdeplot(df.loc[df['weather'] == 1,'count'],label = 'w1')
sns.kdeplot(df.loc[df['weather'] == 2,'count'],label = 'w2')
sns.kdeplot(df.loc[df['weather'] == 3,'count'],label = 'w3')
sns.kdeplot(df.loc[df['weather'] == 4,'count'],label = 'w4')
plt.legend()
plt.show()
```



In [340…

```
Ho = 'The average number of rented cycles for every weather is same'
Ha = 'Atleast one weather has different average number of rented cycles'
alpha = 0.05
f_stat,p_value = f_oneway(w1,w2,w3,w4)
print("p_value:",p_value)

if p_value < alpha:
    print("Reject Ho, Interpretation:",bold_text(Ha))
else:
    print("Fail to Reject Ho, Interpretation:",bold_text(Ho))
```

```
p_value: 5.482069475935669e-42
Reject Ho, Interpretation: Atleast one weather has different average number of rented cycles
```

In [341…

```
print("Alpha =",alpha)
print("P_value for Shapiro's test season1 is",shapiro(w1)[1])
print("P_value for Shapiro's test season2 is",shapiro(w2)[1])
print("P_value for Shapiro's test season3 is",shapiro(w3)[1])
# print("P_value for Shapiro's test season4 is",shapiro(s4)[1])
print("\nP_value for Levene's test: ", levene(w1,w2,w3,w4)[1])
```

```
Alpha = 0.05
P_value for Shapiro's test season1 is 0.0
P_value for Shapiro's test season2 is 9.781063280987223e-43
P_value for Shapiro's test season3 is 3.876090133422781e-33

P_value for Levene's test:  3.504937946833238e-35
```

From the pvalues it is clearly visible that atleast one of the weather has different variance, all the weather are not of gaussian distirbution

So we cannot perform `ANOVA`, we need to perform `KRUSKAL'S TEST` to test our hypothesis

In [342…

```
w1 =  df.loc[df['weather'] == 1,'count']
w2 =  df.loc[df['weather'] == 2,'count']
w3 =  df.loc[df['weather'] == 3,'count']
w4 =  df.loc[df['weather'] == 4,'count']

alpha = 0.05
k_stat,p_value = kruskal(w1,w2,w3,w4)
print("p_value:",p_value)

if p_value < alpha:
    print("Reject Ho, Interpretation:",bold_text(Ha))
else:
    print("Fail to Reject Ho, Interpretation:",bold_text(Ho))
```

```
p_value: 3.501611300708679e-44
Reject Ho, Interpretation: Atleast one weather has different average number of rented cycles
```

Conclusion

1. Atleast one weather has different average number of rented cycles

Weather is dependent on season

In [343…

```
pd.crosstab(df['weather'],df['season'])
```

Out[343]:

season	1	2	3	4
weather				
1	1759	1801	1930	1702
2	715	708	604	807
3	211	224	199	225
4	1	0	0	0

In [344…

```
Ho = 'Weather is independent on Season'
Ha = 'Weather is dependent on Season'
alpha = 0.05

chi_stat,p_value,*a = chi2_contingency(pd.crosstab(df['weather'],df['season']))
print("p_value:",p_value)

if p_value < alpha:
    print("Reject Ho, Interpretation:",bold_text(Ha))
else:
    print("Fail to Reject Ho, Interpretation:",bold_text(Ho))
```

```
p_value: 1.5499250736864862e-07
Reject Ho, Interpretation: Weather is dependent on Season
```

Conclusion

1. Weather is dependent on season