Yulu Business Case Study- Hypothesis testing

By Parth Patel

Business Objective - problem statement

Yulu is India's leading micro-mobility service provider, which offers unique vehicles for the daily commute. Yulu zones are located at all the appropriate locations. Yulu has recently suffered considerable dips in its revenues. Through this case study, we want to understand the factors on which the demand for these shared electric cycles depends. Specifically, we want to understand the factors affecting the demand for these shared electric cycles in the Indian market.

Importing Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from scipy.stats import ttest_rel,ttest_ind,ttest_1samp
from scipy.stats import chi2_contingency, chisquare
from scipy.stats import f_oneway, kruskal, shapiro, levene
from scipy.stats import spearmanr
from scipy.stats import norm
from statsmodels.graphics.gofplots import qqplot
from scipy.stats import probplot

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

```
df = pd.read_csv("bike_sharing.csv")
df.head()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windsp
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	
	2011-01-								

```
df.head()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windsp
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	
	2011-01-								

```
df.shape
```

```
(10886, 12)
```

```
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
```

Statistical Summary

```
df.describe()
```

	season	holiday	workingday	weather	temp	atem
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.00000
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.65508
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.47460
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.76000
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.66500
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.24000
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.06000
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.45500

Non Graphical Analysis

since we have numerical data for season, weather etc, I am changing them into categorical data for simplicity in analysis

```
def season(s):
    if s==1:
        return 'spring'
    if s==2:
        return 'summer'
    if s==3:
        return 'fall'
    if s==4:
        return 'winter'


df['season'] = df.season.apply(season)
df['season'] = df['season'].astype('O')  #changing the dtype as well


df['holiday']=df.holiday.apply(lambda x: 'holiday' if x==1 else 'no holiday')
df['holiday'] = df['holiday'].astype('O') #changing the dtype as well


df['workingday']=df.workingday.apply(lambda x: 'working day' if x==1 else 'weeken
df['workingday'] = df['workingday'].astype('O') #changing the dtype as well


def weather(x):
    if x==1:
        return 'clear'
    if x==2:
        return 'cloudy'
    if x==3:
        return 'Light rain'
    if x==4:
        return 'heavy rain'


df['weather'] = df.weather.apply(weather)
df['weather'] = df['weather'].astype('O')
```

```
df.head()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	win
0	2011-01-01 00:00:00	spring	no holiday	weekend/holiday	clear	9.84	14.395	81	
1	2011-01-01 01:00:00	spring	no holiday	weekend/holiday	clear	9.02	13.635	80	
	2011-01-								

```
df.season.value_counts()
```

```
season
winter    2734
summer    2733
fall      2733
spring    2686
Name: count, dtype: int64
```

```
df.weather.value_counts()
```

```
weather
clear         7192
cloudy        2834
Light rain     859
heavy rain       1
Name: count, dtype: int64
```

```
df.holiday.value_counts()
```

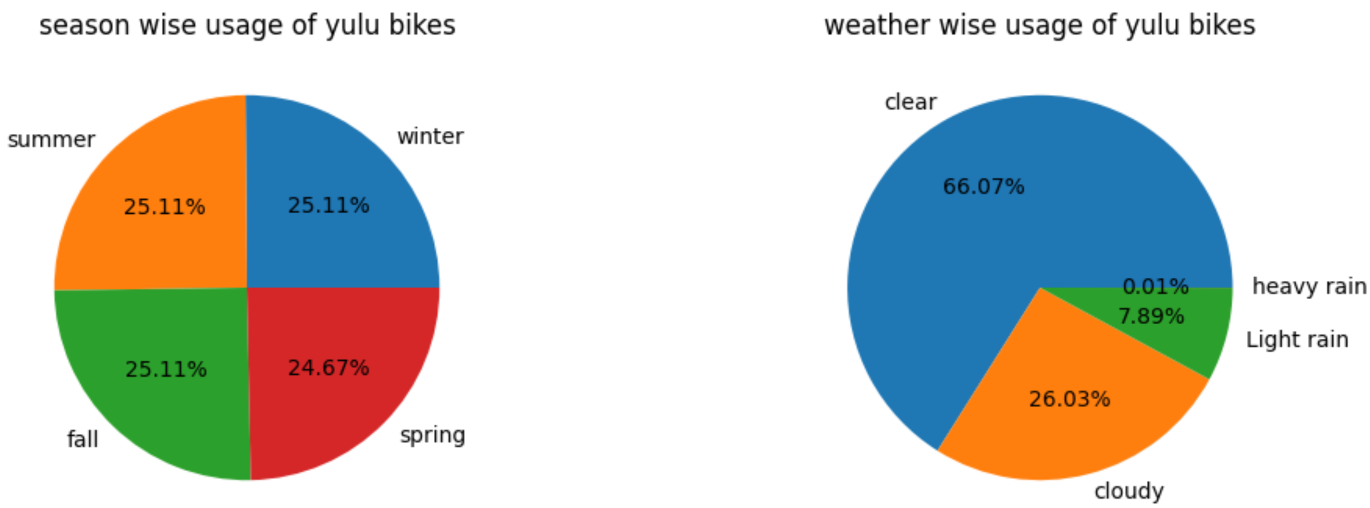
```
holiday
no holiday    10575
holiday         311
Name: count, dtype: int64
```

```
df.workingday.value_counts()
```

```
workingday
working day        7412
weekend/holiday    3474
Name: count, dtype: int64
```

Visual Analysis - Univariate and Bivariate Graphs

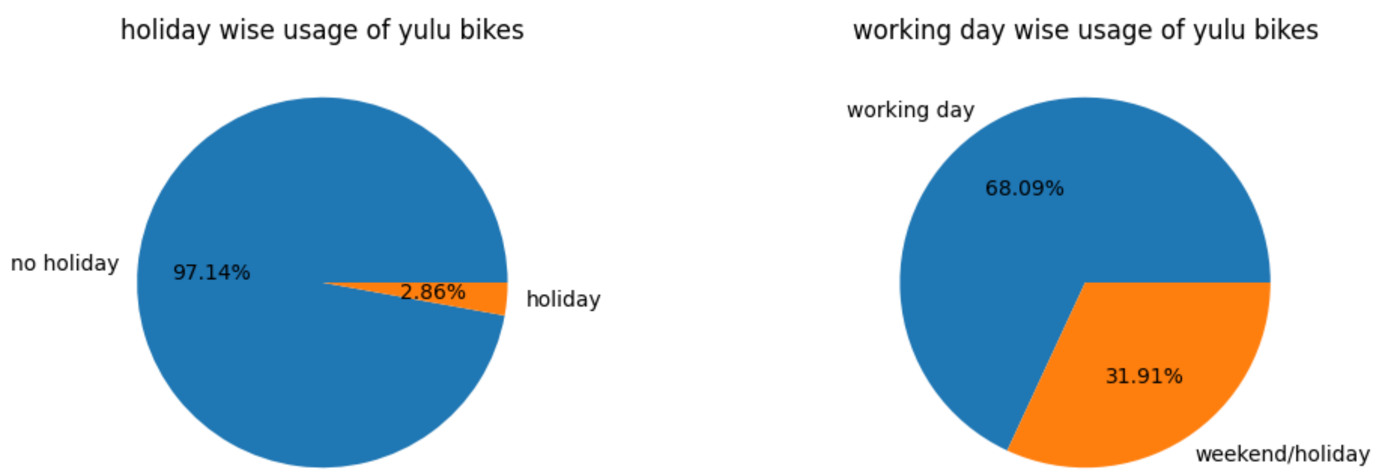
```
plt.figure(figsize=(12,4))
plt.subplot(1,2,1)
plt.pie(df['season'].value_counts().values, labels = df['season'].value_counts().
plt.title('season wise usage of yulu bikes')

plt.subplot(1,2,2)
plt.pie(df['weather'].value_counts().values,labels=df['weather'].value_counts().i
plt.title('weather wise usage of yulu bikes')

plt.show()
```



Yulu Bike usage is almost same for all seasons.

Yulu Bike usage is high when weather is clear.

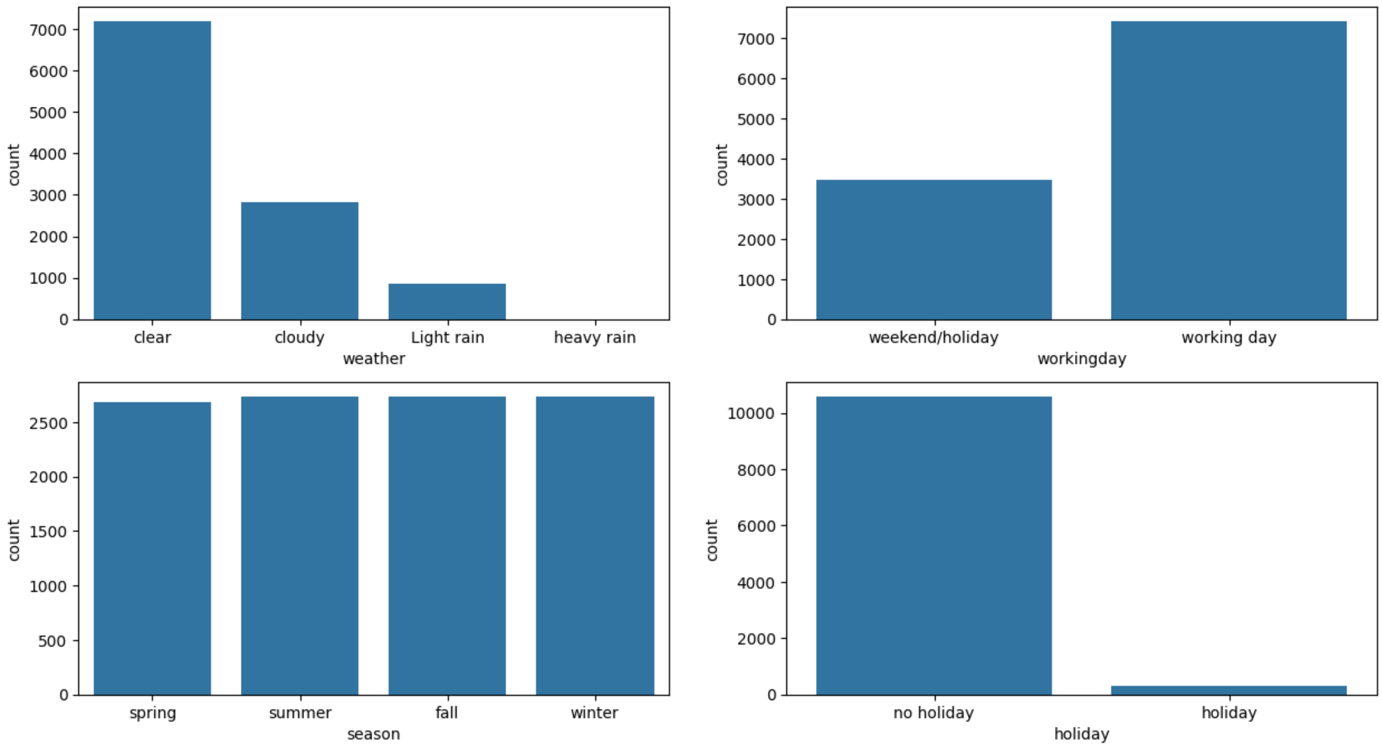
```
plt.figure(figsize=(12,4))
plt.subplot(1,2,1)
plt.pie(df['holiday'].value_counts().values, labels = df['holiday'].value_counts(
plt.title('holiday wise usage of yulu bikes')

plt.subplot(1,2,2)
plt.pie(df['workingday'].value_counts().values,labels=df['workingday'].value_coun
plt.title('working day wise usage of yulu bikes')

plt.show()
```



Yulu Bike usage is high on "no holiday" days.

Yulu Bike usage is high on working days.

```
plt.figure(figsize=(15,8))
plt.subplot(2,2,1)
sns.countplot(data=df, x='weather')
plt.subplot(2,2,2)
sns.countplot(data=df, x='workingday')
plt.subplot(2,2,3)
sns.countplot(data=df, x='season')
plt.subplot(2,2,4)
sns.countplot(data=df, x='holiday')
plt.show()
```


count
weather
clear
cloudy
Light rain
heavy rain
workingday
weekend/holiday
working day
season
spring
summer
fall
winter
holiday
no holiday
holiday

These graphs suggests that yulu bikes usage is high on no-holiday working day, having clear weather.

There is not much impact of the season.

for "temp" and "atemp" column, I am creating categories, for values in these columns

```
bins = [0,10,20,30,40,50]
groups = ['low', 'low-moderate', 'moderate', 'moderate-high', 'high']  #Categorie
df['temp_cat'] = pd.cut(df['temp'], bins, labels = groups)
df['temp_cat'] = df['temp_cat'].astype('O')
```

```
df.groupby('temp_cat')['count'].mean()
```

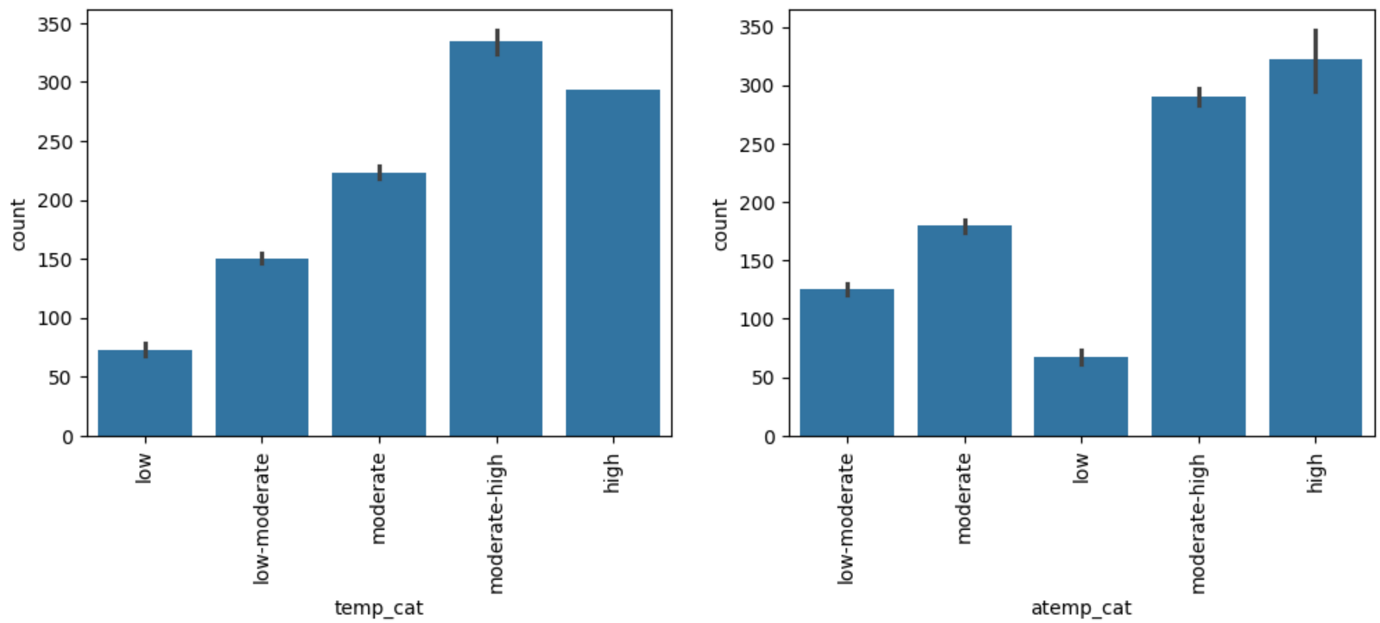
```
temp_cat
high             294.000000
low               73.185862
low-moderate     150.465053
moderate         223.411398
moderate-high    334.306516
Name: count, dtype: float64
```

```
bins = [0,10,20,30,40,50]
groups = ['low', 'low-moderate', 'moderate', 'moderate-high', 'high']  #Categorie
df['atemp_cat'] = pd.cut(df['atemp'], bins, labels = groups)
df['atemp_cat'] = df['atemp_cat'].astype('O')
```

```
df.head()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	win
0	2011-01-01 00:00:00	spring	no holiday	weekend/holiday	clear	9.84	14.395	81	
1	2011-01-01 01:00:00	spring	no holiday	weekend/holiday	clear	9.02	13.635	80	
2	2011-01-01 02:00:00	spring	no holiday	weekend/holiday	clear	9.02	13.635	80	
3	2011-01-01 03:00:00	spring	no holiday	weekend/holiday	clear	9.84	14.395	75	
4	2011-01-01 04:00:00	spring	no holiday	weekend/holiday	clear	9.84	14.395	75	

```
plt.figure(figsize=(12,4))
plt.subplot(1,2,1)
sns.barplot(data=df, x='temp_cat', y='count', estimator='mean')
plt.xticks(rotation=90)
plt.subplot(1,2,2)
sns.barplot(data=df, x='atemp_cat', y='count', estimator='mean')
plt.xticks(rotation=90)
plt.show()
```



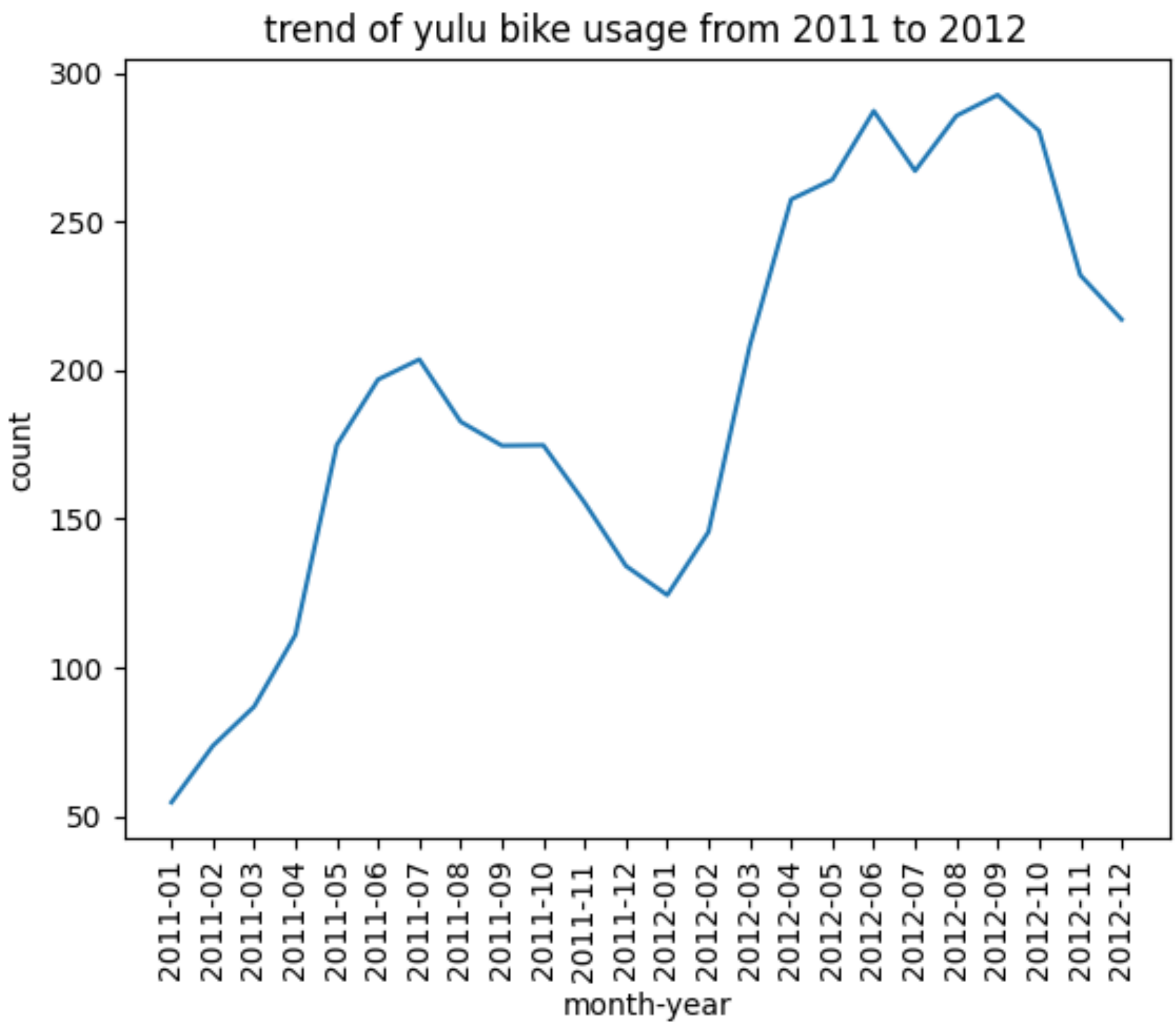
These graph suggests that yulu bike usage is high when temperature is moderate-high to high.

To check month on month growth and usage trend, I am creating a month-year column from datetime column

```
# df['date'] = pd.to_datetime(df.datetime).dt.date
# df['month'] = pd.to_datetime(df.datetime).dt.month
# df['year'] = pd.to_datetime(df.datetime).dt.year
df['month-year'] = pd.to_datetime(df.datetime).dt.strftime('%Y-%m')
df_date = df.groupby(['month-year'])['count'].mean().reset_index()
df_date.head()
```

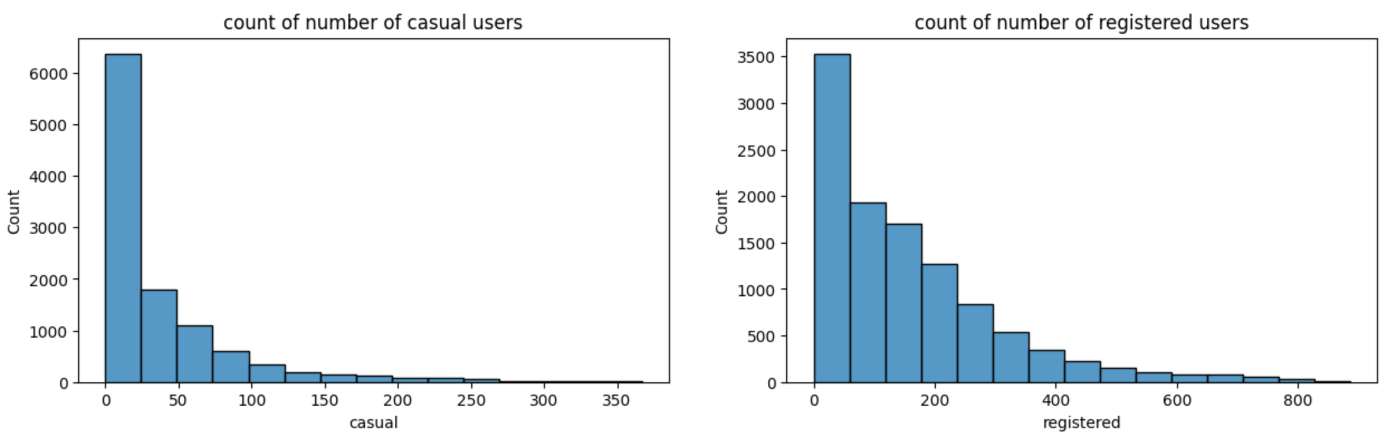
	month-year	count
0	2011-01	54.645012
1	2011-02	73.641256
2	2011-03	86.849776
3	2011-04	111.026374
4	2011-05	174.809211

```
sns.lineplot(data=df_date, x='month-year', y='count')
plt.xticks(rotation=90)
plt.title("trend of yulu bike usage from 2011 to 2012")
plt.show()
```



The graph indicates that the mean usage of Yulu bikes has displayed an upward trend since the beginning of 2011, albeit with periodic decreases towards the end of each year. Similarly, at the start of 2012, there was a noticeable increase in usage, followed by a subsequent decline towards the end of the year.

```
plt.figure(figsize=(15,4))
plt.subplot(1,2,1)
sns.histplot(data=df, x='casual', bins=15 )
plt.title('count of number of casual users')
plt.subplot(1,2,2)
sns.histplot(data=df, x='registered', bins=15)
plt.title('count of number of registered users')

plt.show()
```

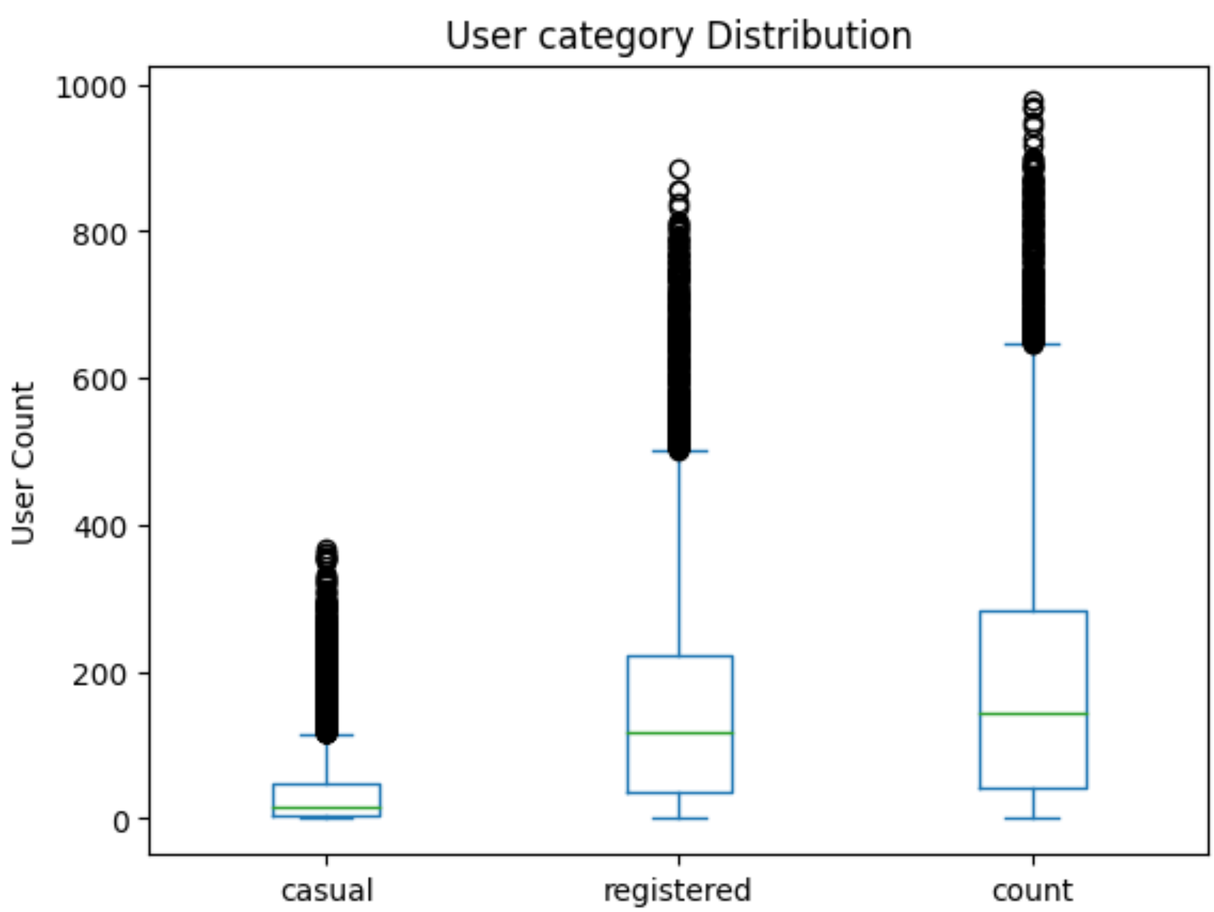


It's evident that the number of registered users significantly surpasses that of casual users, highlighting a positive aspect for Yulu's user base.

Outlier Analysis

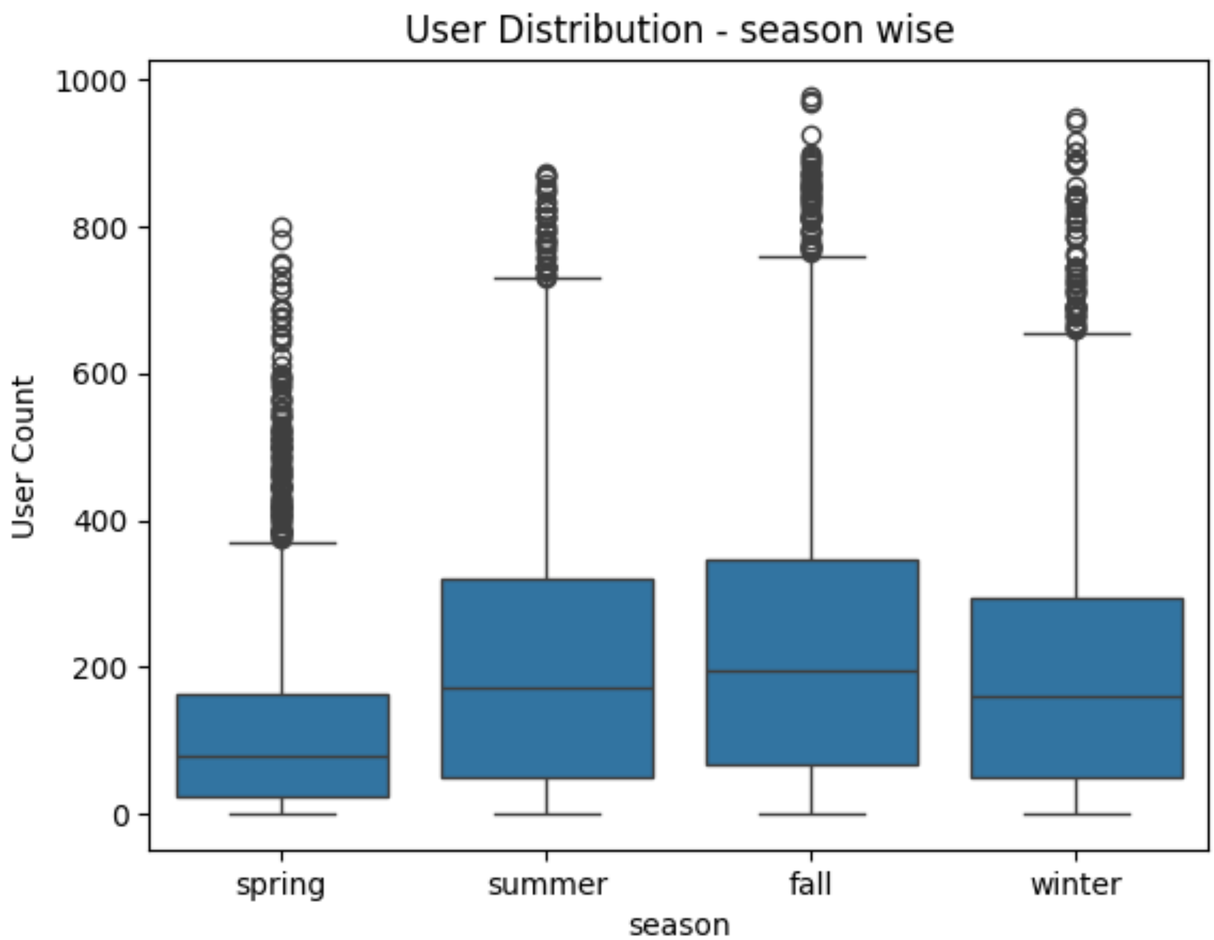
Checking Outliers using Boxplot

```
df1 = df[['casual','registered','count']]

df1.plot(kind='box')

plt.ylabel('User Count')
plt.title('User category Distribution')
plt.show()
```



Outliers exists in all three columns.

```
sns.boxplot(data=df, x='season', y='count')
plt.ylabel('User Count')
plt.title('User Distribution - season wise')
plt.show()
```



Removing Outlier using IQR Method

```
num_cols = [i for i in df.columns if df[i].dtypes != 'O']  #getting all numerical
num_cols
```

```
['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count']
```

```
def Get_Numerical_Outlier_indices(df, cols):
    out_ind = []
    for col in cols:
        q1 = df[col].quantile(0.25)
        q2 = df[col].quantile(0.75)
        iqr = q2-q1
        rare_ind = df[((df[col]<(q1-(1.5*iqr)))|(df[col]>(q2+(1.5*iqr))))].index
        out_ind.extend(rare_ind)

    out_ind = set(out_ind)
    return out_ind

numerical_outlier_indices = Get_Numerical_Outlier_indices(df, num_cols)
```

```
outlier_len = len(numerical_outlier_indices)   #number of outliers in dataset
orig_len =  len(df)
print(f'original length of data: {orig_len}')
print(f'outliers length: {outlier_len}')
```

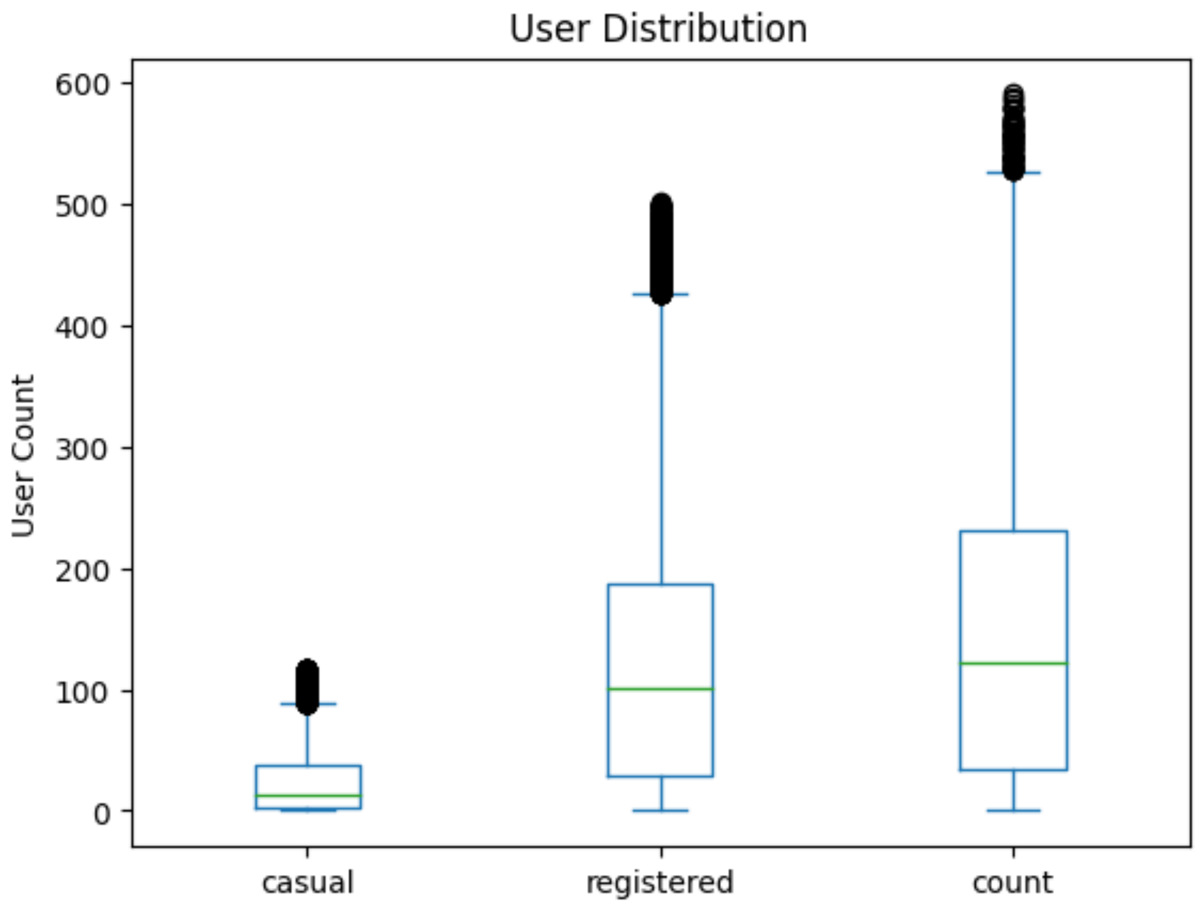
```
original length of data: 10886
outliers length: 1368
```

```
df = df.drop(numerical_outlier_indices) #dropping outlier rows
```

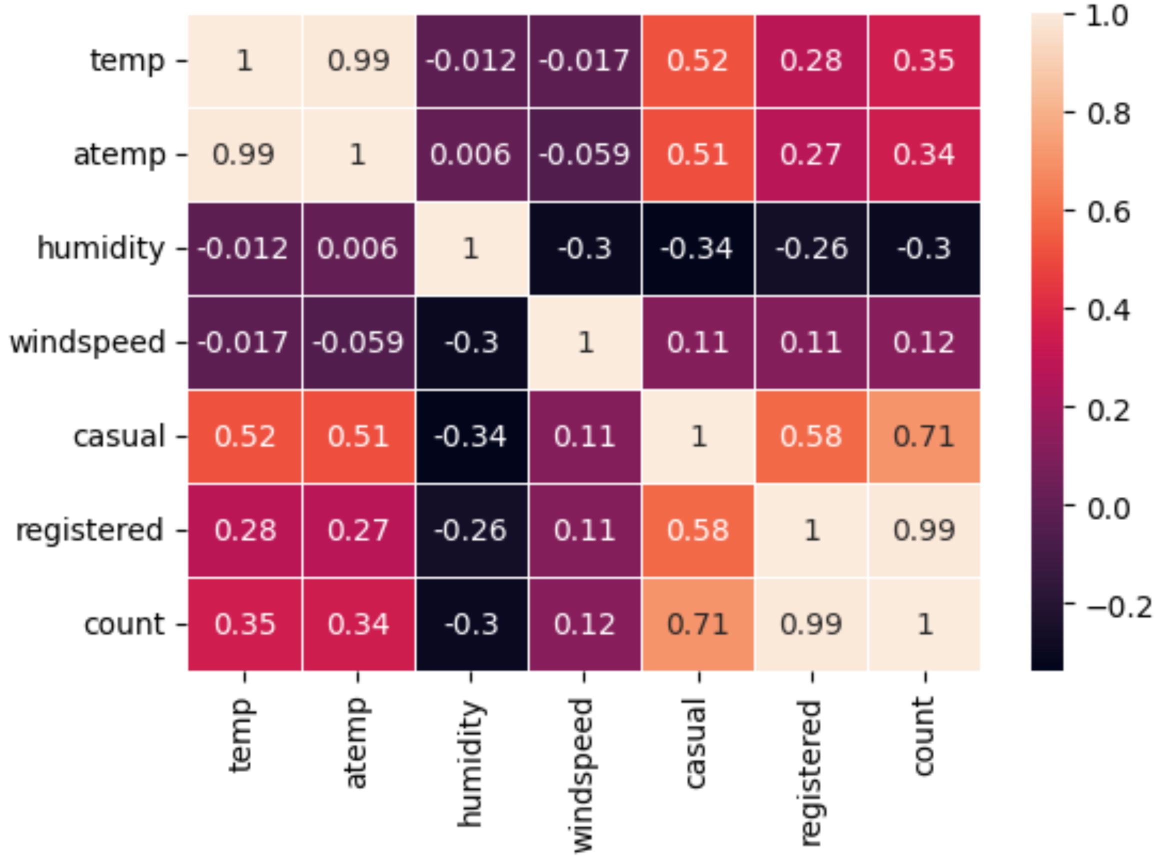
```
data_len = len(df) #dataset left with us
print(f'data left with us after outlier removal: {data_len}')
print(f'% of data left after outlier removal: {round(data_len*100/orig_len,2)}%')
```

```
data left with us after outlier removal: 9518
% of data left after outlier removal: 87.43%
```

```
df2 = df[['casual','registered','count']]

df2.plot(kind='box')

plt.ylabel('User Count')
plt.title('User Distribution')
plt.show()
```

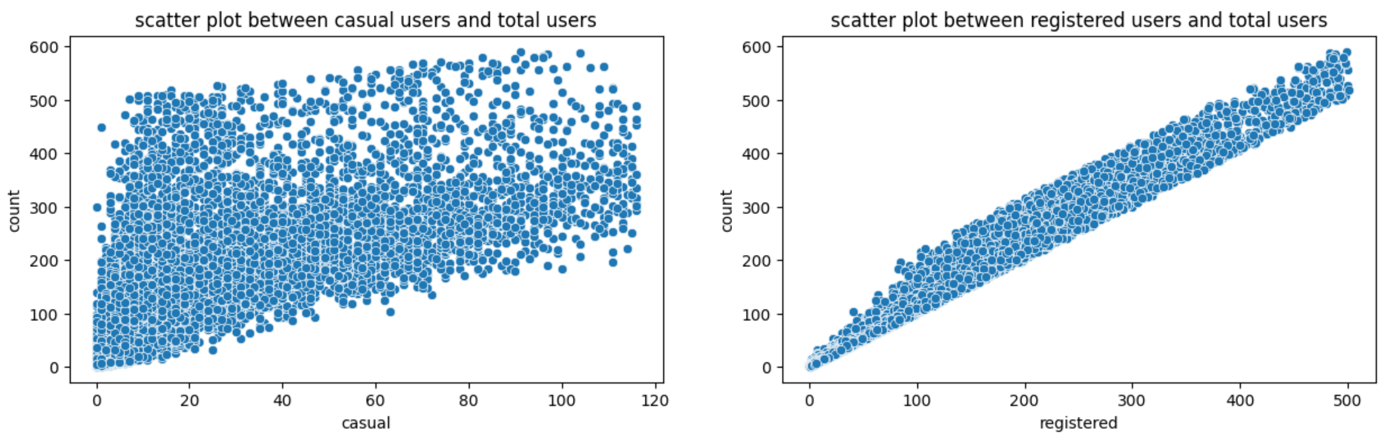


```
plt.figure(figsize=(6,4))
sns.heatmap(df.corr(numeric_only=True), annot=True, linewidth=.5)
plt.show()
```



Based on this observation, it can be concluded that registered users make a substantial contribution to the total user base, given the strong correlation between registered users and overall user numbers.

```
plt.figure(figsize=(15,4))
plt.subplot(1,2,1)
sns.scatterplot(data=df, x='casual', y='count')
plt.title("scatter plot between casual users and total users")
plt.subplot(1,2,2)
sns.scatterplot(data=df, x='registered', y='count')
plt.title("scatter plot between registered users and total users")
plt.show()
```



The scatterplot reaffirms a strong correlation between registered customers and the total number of Yulu bike users.

Hypothesis Testing

Test Stats:

alpha = 0.05 (95% significance level)

1.Working Day has effect on number of electric cycles rented

H0: Working Day has **no** effect on number of electric bikes rented

Ha: Working Day has effect on number of electric bikes rented

Assumptions

Observations in each sample are normally distributed (Guassian curve)

Observations in each sample are independent and identically distributed.

```
working = df[df['workingday']=='working day']
nonworking = df[df['workingday']=='weekend/holiday']
```

H0: data has gaussian distribution

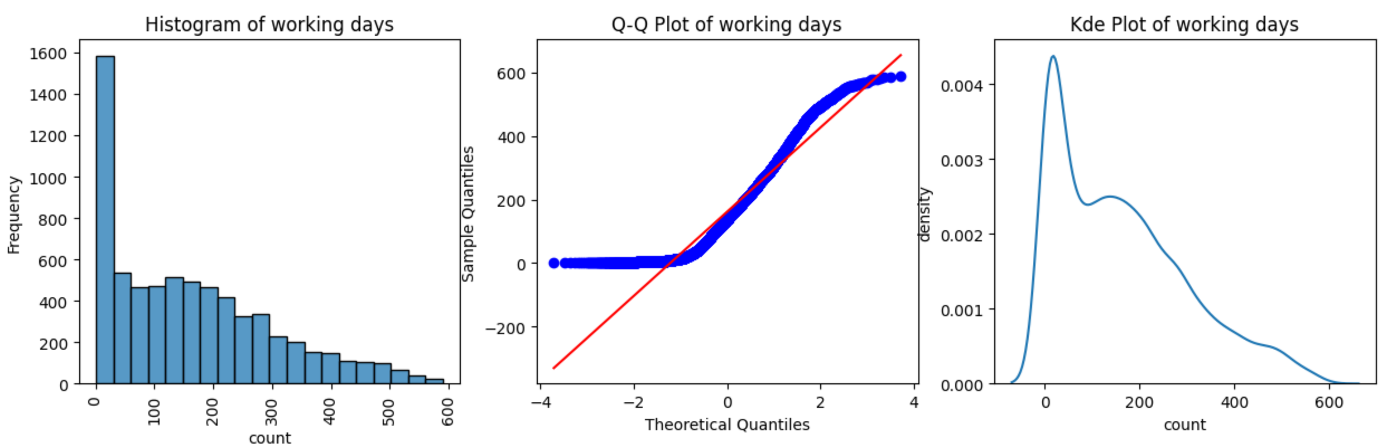
Ha: data does not have gaussian distribution

```
shapiro(working['count']) , shapiro(nonworking['count'])
```

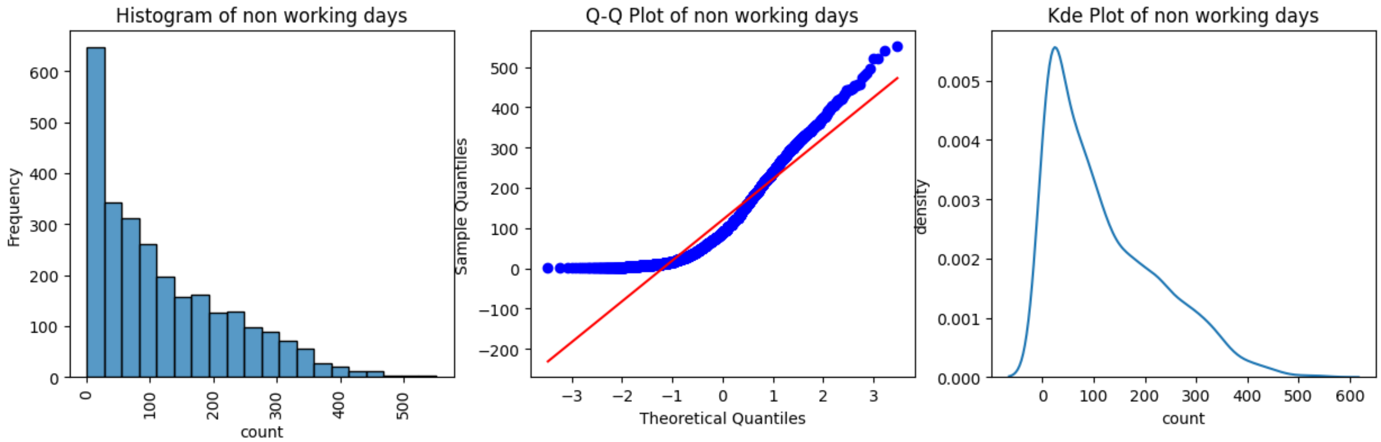
```
(ShapiroResult(statistic=0.9170000553131104, pvalue=0.0),
 ShapiroResult(statistic=0.8963597416877747, pvalue=1.6402717005353784e-39))
```

The p-values for both datasets are less than the alpha value (0.05), indicating that we reject the null hypothesis, suggesting that the data is not Gaussian. Let's further verify this using a QQ plot.

```
plt.figure(figsize=(15, 4))
plt.subplot(1,3,1)
sns.histplot(working['count'], bins=20)
plt.xlabel('count')
plt.ylabel('Frequency')
plt.title(f'Histogram of working days')
plt.xticks(rotation=90)

plt.subplot(1,3,2)
probplot(working['count'], dist='norm', plot=plt)
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Sample Quantiles')
plt.title(f'Q-Q Plot of working days')

plt.subplot(1,3,3)
sns.kdeplot(working['count'])
plt.xlabel('count')
plt.ylabel('density')
plt.title(f'Kde Plot of working days')

plt.show()
```



```
plt.figure(figsize=(15, 4))
plt.subplot(1,3,1)
sns.histplot(nonworking['count'], bins=20)
plt.xlabel('count')
plt.ylabel('Frequency')
plt.title(f'Histogram of non working days')
plt.xticks(rotation=90)

plt.subplot(1,3,2)
probplot(nonworking['count'], dist='norm', plot=plt)
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Sample Quantiles')
plt.title(f'Q-Q Plot of non working days')

plt.subplot(1,3,3)
sns.kdeplot(nonworking['count'])
plt.xlabel('count')
plt.ylabel('density')
plt.title(f'Kde Plot of non working days')

plt.show()
```



The QQ plot indicates that our data is not Gaussian. In practical scenarios with large datasets, assumptions may not always hold true. Therefore, we will proceed with a 2-sample t-test to assess whether the data is independent of each other.

```
#Mean of both groups

working_mean = working['count'].mean()
nonworking_mean = nonworking['count'].mean()
working_mean,nonworking_mean
```

```
(161.97010309278352, 120.68108504398828)
```

```
# Standard deviation of both group

working_std = working['count'].std()
nonworking_std = nonworking['count'].std()
working_std,nonworking_std
```

```
(138.58857204299835, 106.74781110470883)
```

```
stats, p = ttest_ind(working['count'],nonworking['count'])
print(f'p-value: {p}')
if p < 0.05:
    print('reject null hypothesis: bike usage depends on working day')
else:
    print('fail to reject null hypothesis: bike usage does not depends on working
```

```
p-value: 5.384896180235767e-44
reject null hypothesis: bike usage depends on working day
```

Test result: yulu bike usage depends on working day

2. No. of cycles rented similar or different in different seasons

H0: cycle usage is independent of season

Ha: cycle usage depends on season

Assumptions

Observations in each sample are normally distributed.

Observations in each sample should have same variance

```
df.season.value_counts()
```

```
season
winter    2475
spring    2463
summer    2292
fall      2288
Name: count, dtype: int64
```

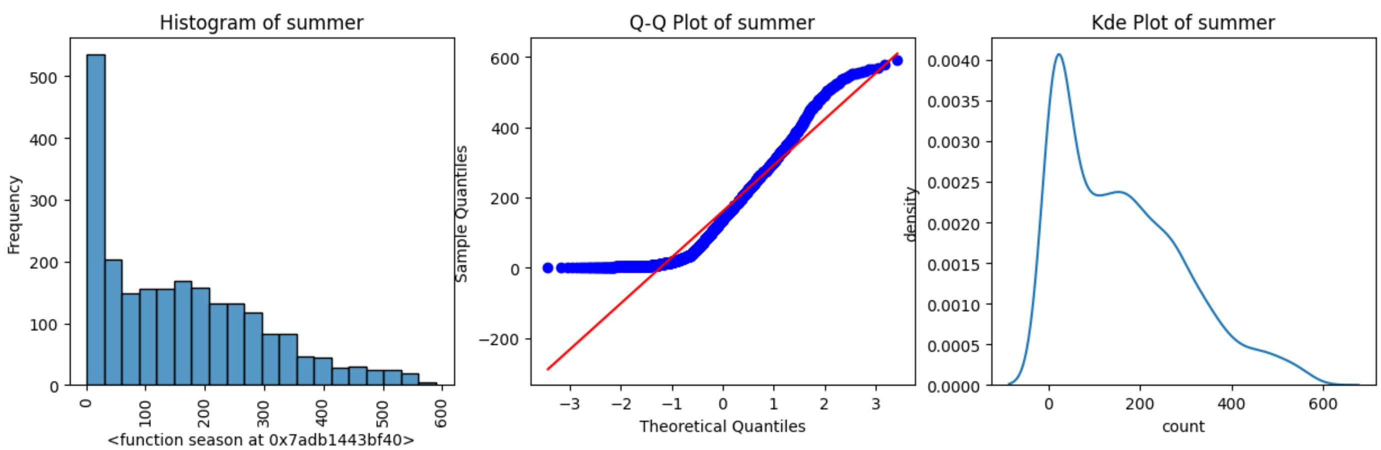
```
summer = df[df['season']=='summer']
winter = df[df['season']=='winter']
fall = df[df['season']=='fall']
spring = df[df['season']=='spring']
```

```
#Check if data is gaussian

# h0: data has gaussian distribution
# ha: data does not have gaussian distribution

shapiro(summer['count']), shapiro(winter['count']), shapiro(fall['count']), shapi
```

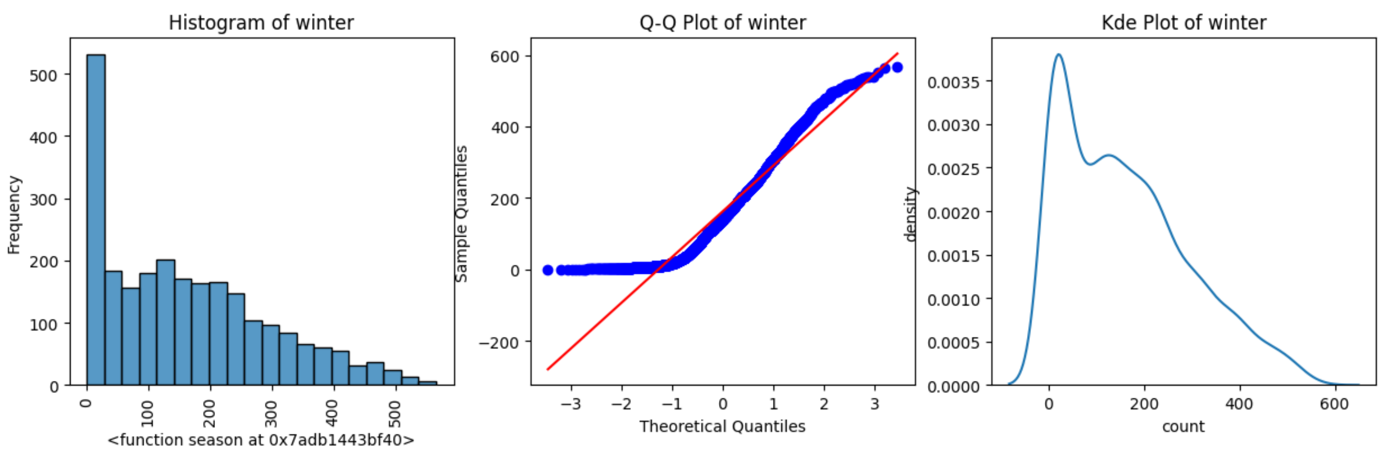
```
(ShapiroResult(statistic=0.9176210165023804, pvalue=1.2426929547549821e-33),
 ShapiroResult(statistic=0.9272552728652954, pvalue=4.5287389233367154e-33),
 ShapiroResult(statistic=0.9323311448097229, pvalue=5.115096899524057e-31),
 ShapiroResult(statistic=0.8594179153442383, pvalue=2.0725204287364044e-42))
```

p-values for data is less than alpha, meaning **reject null hypothesis: Data is not guassian** Let's check with qq-plot:

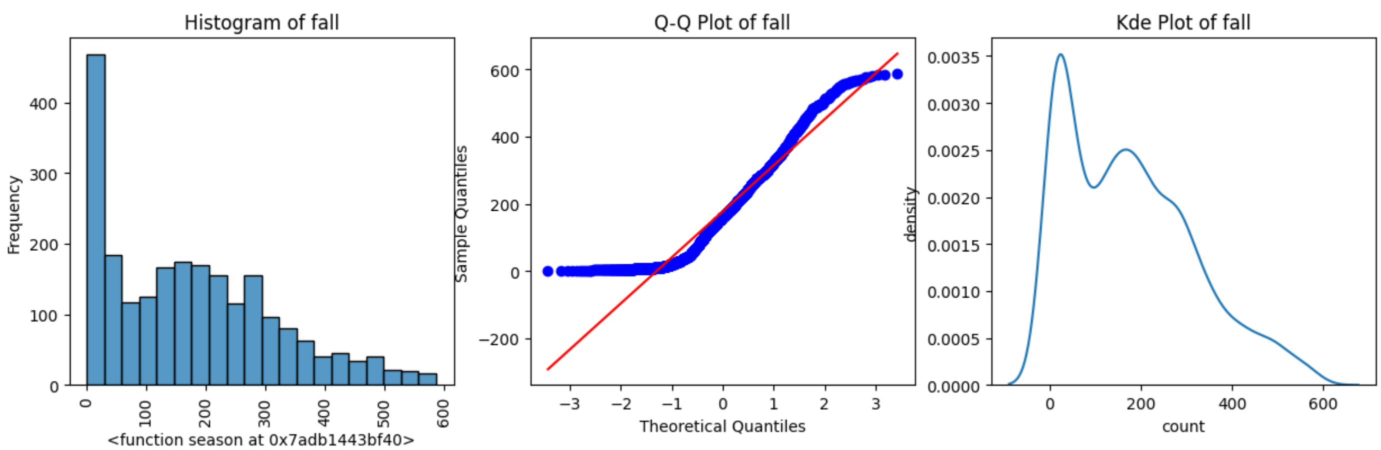
```
plt.figure(figsize=(15, 4))
plt.subplot(1,3,1)
sns.histplot(summer['count'], bins=20)
plt.xlabel(f'{season}')
plt.ylabel('Frequency')
plt.title(f'Histogram of summer')
plt.xticks(rotation=90)
plt.subplot(1,3,2)
probplot(summer['count'], dist='norm', plot=plt)
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Sample Quantiles')
plt.title(f'Q-Q Plot of summer')

plt.subplot(1,3,3)
sns.kdeplot(summer['count'])
#     plt.xlabel(column)
plt.ylabel('density')
plt.title(f'Kde Plot of summer')

plt.show()
```



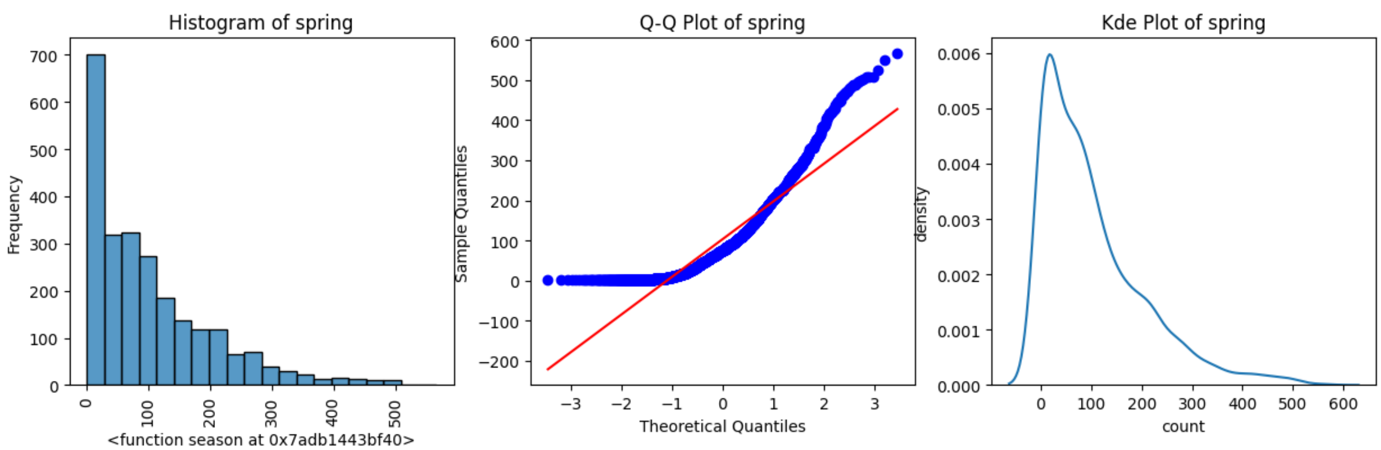
```
plt.figure(figsize=(15, 4))
plt.subplot(1,3,1)
sns.histplot(winter['count'], bins=20)
plt.xlabel(f'{season}')
plt.ylabel('Frequency')
plt.title(f'Histogram of winter')
plt.xticks(rotation=90)
plt.subplot(1,3,2)
probplot(winter['count'], dist='norm', plot=plt)
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Sample Quantiles')
plt.title(f'Q-Q Plot of winter')

plt.subplot(1,3,3)
sns.kdeplot(winter['count'])
#     plt.xlabel(column)
plt.ylabel('density')
plt.title(f'Kde Plot of winter')

plt.show()
```



```
plt.figure(figsize=(15, 4))
plt.subplot(1,3,1)
sns.histplot(fall['count'], bins=20)
plt.xlabel(f'{season}')
plt.ylabel('Frequency')
plt.title(f'Histogram of fall')
plt.xticks(rotation=90)
plt.subplot(1,3,2)
probplot(fall['count'], dist='norm', plot=plt)
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Sample Quantiles')
plt.title(f'Q-Q Plot of fall')

plt.subplot(1,3,3)
sns.kdeplot(fall['count'])
#     plt.xlabel(column)
plt.ylabel('density')
plt.title(f'Kde Plot of fall')

plt.show()
```



```
plt.figure(figsize=(15, 4))
plt.subplot(1,3,1)
sns.histplot(spring['count'], bins=20)
plt.xlabel(f'{season}')
plt.ylabel('Frequency')
plt.title(f'Histogram of spring')
plt.xticks(rotation=90)
plt.subplot(1,3,2)
probplot(spring['count'], dist='norm', plot=plt)
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Sample Quantiles')
plt.title(f'Q-Q Plot of spring')

plt.subplot(1,3,3)
sns.kdeplot(spring['count'])
#     plt.xlabel(column)
plt.ylabel('density')
plt.title(f'Kde Plot of spring')

plt.show()
```



qq-plot suggests that data is not guassian. Let's check for variance using levene test:

H0: variance is same

Ha: variance is different

```
summer = df[df['season']=='summer']['count']
winter = df[df['season']=='winter']['count']
fall = df[df['season']=='fall']['count']
spring = df[df['season']=='spring']['count']


stats, p = levene(summer, winter, fall, spring)
print(f'p-value: {p}')
if p < 0.05:
    print('reject null hypothesis: variance is different')
else:
    print('fail to reject null hypothesis: variance is same ')
```

```
p-value: 6.687186315723853e-87
reject null hypothesis: variance is different
```

Assumptions are not holding true, but still proceeding with ANOVA test:

```
stats, p = f_oneway(summer, winter, fall, spring)
print(f'p-value: {p}')
if p < 0.05:
    print('reject null hypothesis: bike usage depends on season')
else:
    print('fail to reject null hypothesis: bike usage is independent of season ')
```

```
p-value: 1.328514170995064e-98
reject null hypothesis: bike usage depends on season
```

As mentioned, for large practical data, assumptions sometimes do not hold true. So applying Kruskal test:

```
stats, p = kruskal(summer, winter, fall, spring)
print(f'p-value: {p}')
if p < 0.05:
    print('reject null hypothesis: bike usage depends on season')
else:
    print('fail to reject null hypothesis: bike usage is independent of season ')
```

```
p-value: 9.09294670507136e-93
reject null hypothesis: bike usage depends on season
```

Test result: Yulu bike usage depends on season

3. No. of cycles rented similar or different in different weather

H0: cycle usage is independent of weather

Ha: cycle usage dependent on weather

Assumptions:

Observations in each sample are normally distributed.

Observations in each sample should have same variance

```
df.weather.value_counts()
```

```
weather
clear         6176
cloudy        2568
Light rain     773
heavy rain       1
Name: count, dtype: int64
```

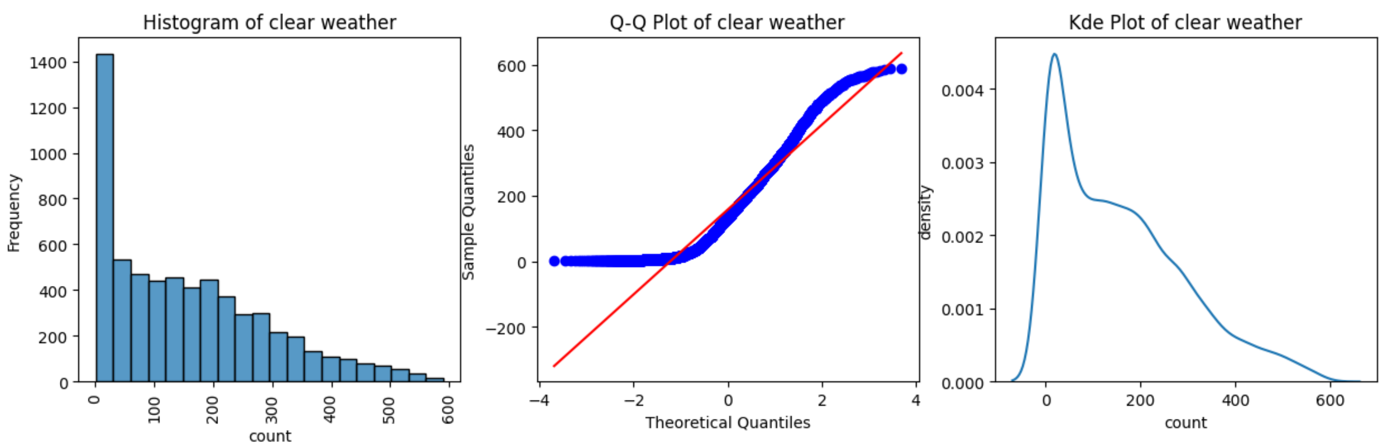
```
clear = df[df['weather']=='clear']
cloudy = df[df['weather']=='cloudy']
lightRain = df[df['weather']=='Light rain']
# heavyRain = df[df['weather']=='heavy rain']
```

```
shapiro(clear['count']), shapiro(cloudy['count']), shapiro(lightRain['count'])
```

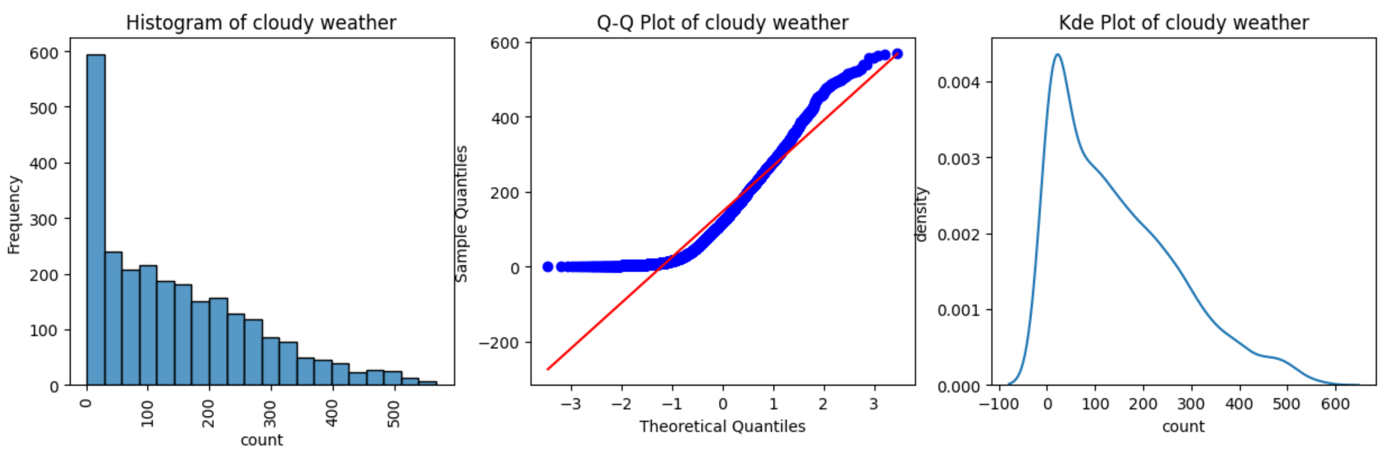
```
(ShapiroResult(statistic=0.9150597453117371, pvalue=0.0),
 ShapiroResult(statistic=0.91085284948349, pvalue=2.1035535621065664e-36),
 ShapiroResult(statistic=0.8443405628204346, pvalue=7.518643302497174e-27))
```

p-values for data is less than alpha, meaning **reject null hypothesis: Data is not guassian** Let's check with qq-plot:

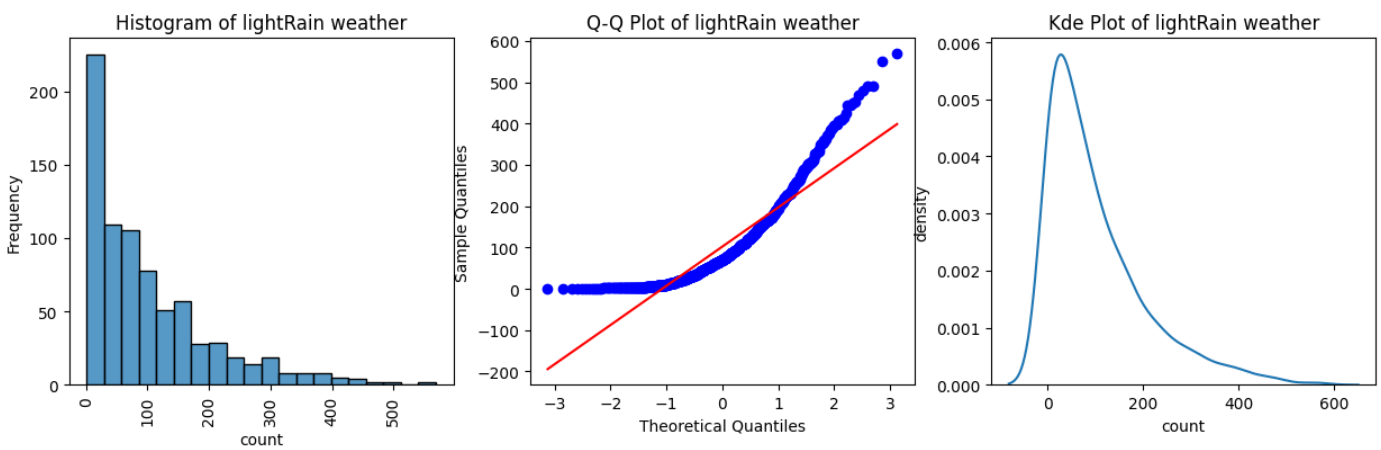
```
plt.figure(figsize=(15, 4))
plt.subplot(1,3,1)
sns.histplot(clear['count'], bins=20)
# plt.xlabel('clear')
plt.ylabel('Frequency')
plt.title(f'Histogram of clear weather')
plt.xticks(rotation=90)
plt.subplot(1,3,2)
probplot(clear['count'], dist='norm', plot=plt)
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Sample Quantiles')
plt.title(f'Q-Q Plot of clear weather')

plt.subplot(1,3,3)
sns.kdeplot(clear['count'])
#     plt.xlabel(column)
plt.ylabel('density')
plt.title(f'Kde Plot of clear weather')

plt.show()
```



```
plt.figure(figsize=(15, 4))
plt.subplot(1,3,1)
sns.histplot(cloudy['count'], bins=20)
# plt.xlabel('clear')
plt.ylabel('Frequency')
plt.title(f'Histogram of cloudy weather')
plt.xticks(rotation=90)
plt.subplot(1,3,2)
probplot(cloudy['count'], dist='norm', plot=plt)
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Sample Quantiles')
plt.title(f'Q-Q Plot of cloudy weather')

plt.subplot(1,3,3)
sns.kdeplot(cloudy['count'])
#     plt.xlabel(column)
plt.ylabel('density')
plt.title(f'Kde Plot of cloudy weather')

plt.show()
```



```
plt.figure(figsize=(15, 4))
plt.subplot(1,3,1)
sns.histplot(lightRain['count'], bins=20)
# plt.xlabel('clear')
plt.ylabel('Frequency')
plt.title(f'Histogram of lightRain weather')
plt.xticks(rotation=90)
plt.subplot(1,3,2)
probplot(lightRain['count'], dist='norm', plot=plt)
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Sample Quantiles')
plt.title(f'Q-Q Plot of lightRain weather')

plt.subplot(1,3,3)
sns.kdeplot(lightRain['count'])
#     plt.xlabel(column)
plt.ylabel('density')
plt.title(f'Kde Plot of lightRain weather')

plt.show()
```



qq-plot suggests that data is **not guassian**. Let's check for variance using levene test:

H0: variance is same

Ha: variance is different

```
clear = df[df['weather']=='clear']['count']
cloudy = df[df['weather']=='cloudy']['count']
lightRain = df[df['weather']=='Light rain']['count']


stat, p = levene(clear, cloudy, lightRain)
print(f'p-value: {p}')
if p < 0.05:
    print('reject null hypothesis: variance is different')
else:
    print('fail to reject null hypothesis: variance is same ')
```

```
p-value: 1.1479762859567072e-28
reject null hypothesis: variance is different
```

Assumptions are not holding true, still applying ANOVA test

```
stats, p = f_oneway(clear, cloudy, lightRain)
if p < 0.05:
    print('reject null hypothesis: bike usage depends on weather')
else:
    print('fail to reject null hypothesis: bike usage is independent of weather '
```

```
reject null hypothesis: bike usage depends on weather
```

As mentioned, for large practical data, assumptions sometimes do not hold true. So applying Kruskal test:

```
stats, p = kruskal(clear, cloudy, lightRain)
print(f'p-value: {p}')
if p < 0.05:
    print('reject null hypothesis: bike usage depends on weather')
else:
    print('fail to reject null hypothesis: bike usage is independent of weather '
```

```
p-value: 7.1193803165392e-26
reject null hypothesis: bike usage depends on weather
```

Test result: yulu bike usage depends on weather

4. Weather is dependent on season

H0: weather is independent of season

Ha: weather depends on season

Applying **chi-sqaure test** to check the dependency between season and weather

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

stats, p, dof, e = chi2_contingency(contingency_table)
print(f'p-value: {p}')
if p < 0.05:
    print('reject null hypothesis: weather depends on season')
else:
    print('fail to reject null hypothesis: weather is independent of season')
```

```
p-value: 1.0976664201931212e-07
reject null hypothesis: weather depends on season
```

Test result: weather is dependent on season

Insights and Recommendation

1. **Working Day Dependency:**
    - **Boost Availability during Workdays**: Given the higher Yulu bike usage on weekdays, consider enhancing bike availability on workdays, particularly in areas with dense office and commercial activity.
    - Tailored Incentives for Commuters: Encourage commuter adoption by introducing tailored incentives, such as discounts or loyalty programs, specifically for those using Yulu bikes on weekdays.
2. **Seasonal Dependency:**
    - **Flexible Fleet Management**: Adapt bike fleet sizes according to seasonal demand fluctuations, increasing capacity during peak seasons like spring and summer.
    - **Strategic Seasonal Marketing**: Craft targeted marketing initiatives aligned with seasonal trends, promoting Yulu bike rides to seasonal destinations or events.
3. **Weather Dependency:**
    - **Real-Time Weather Updates**: Implement a system for delivering real-time weather alerts to riders, notifying them of optimal biking conditions during clear weather.
4. **User Retention and Engagement:**
    - **Personalized Offers for Users**: Enhance user engagement by offering personalized discounts and promotions, tailored to registered users' preferences and behavior.
    - Continuous App Improvement: Maintain a focus on app optimization, ensuring a seamless and user-friendly experience to drive user retention and satisfaction.