

# Business Case : Yulu – Hypothesis Testing

## 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.

## Objective

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

## Importing Libraries

```
[1] import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as spy
import datetime as dt
```

```
[2] df = pd.read_csv("/content/bike_sharing.csv")
df.head()
```



	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



Next steps:

[Generate code with df](#)[View recommended plots](#)[New interactive sheet](#)

Dimensions of Data:

```
[3] df.shape
```

```
(10886, 12)
```



```
df.columns
```



```
Index(['datetime', 'season', 'holiday', 'workingday', 'weather', 'temp',
      'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count'],
      dtype='object')
```

Column Profiling:

- **datetime:** datetime
- **season:** season (1: spring, 2: summer, 3: fall, 4: winter)
- **holiday:** whether day is a holiday or not
- **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

Checking null values

	<code>df.isna().sum()</code>
	
	<b>0</b>
<b>datetime</b>	0
<b>season</b>	0
<b>holiday</b>	0
<b>workingday</b>	0
<b>weather</b>	0
<b>temp</b>	0
<b>atemp</b>	0
<b>humidity</b>	0
<b>windspeed</b>	0
<b>casual</b>	0
<b>registered</b>	0
<b>count</b>	0
<b>dtype:</b>	int64

---


## Datatype of the columns


	<code>df.dtypes</code>																										
	<table><tr><th></th><th></th></tr><tr><td><b>datetime</b></td><td>object</td></tr><tr><td><b>season</b></td><td>int64</td></tr><tr><td><b>holiday</b></td><td>int64</td></tr><tr><td><b>workingday</b></td><td>int64</td></tr><tr><td><b>weather</b></td><td>int64</td></tr><tr><td><b>temp</b></td><td>float64</td></tr><tr><td><b>atemp</b></td><td>float64</td></tr><tr><td><b>humidity</b></td><td>int64</td></tr><tr><td><b>windspeed</b></td><td>float64</td></tr><tr><td><b>casual</b></td><td>int64</td></tr><tr><td><b>registered</b></td><td>int64</td></tr><tr><td><b>count</b></td><td>int64</td></tr></table>			<b>datetime</b>	object	<b>season</b>	int64	<b>holiday</b>	int64	<b>workingday</b>	int64	<b>weather</b>	int64	<b>temp</b>	float64	<b>atemp</b>	float64	<b>humidity</b>	int64	<b>windspeed</b>	float64	<b>casual</b>	int64	<b>registered</b>	int64	<b>count</b>	int64
<b>datetime</b>	object																										
<b>season</b>	int64																										
<b>holiday</b>	int64																										
<b>workingday</b>	int64																										
<b>weather</b>	int64																										
<b>temp</b>	float64																										
<b>atemp</b>	float64																										
<b>humidity</b>	int64																										
<b>windspeed</b>	float64																										
<b>casual</b>	int64																										
<b>registered</b>	int64																										
<b>count</b>	int64																										
	<b>dtype:</b> object																										


## Converting the datatype of datetime column from object to datetime

```
[7] df['datetime'] = pd.to_datetime(df['datetime'])
```

 `df["datetime"].min()`

 Timestamp('2011-01-01 00:00:00')

 `df["datetime"].max()`

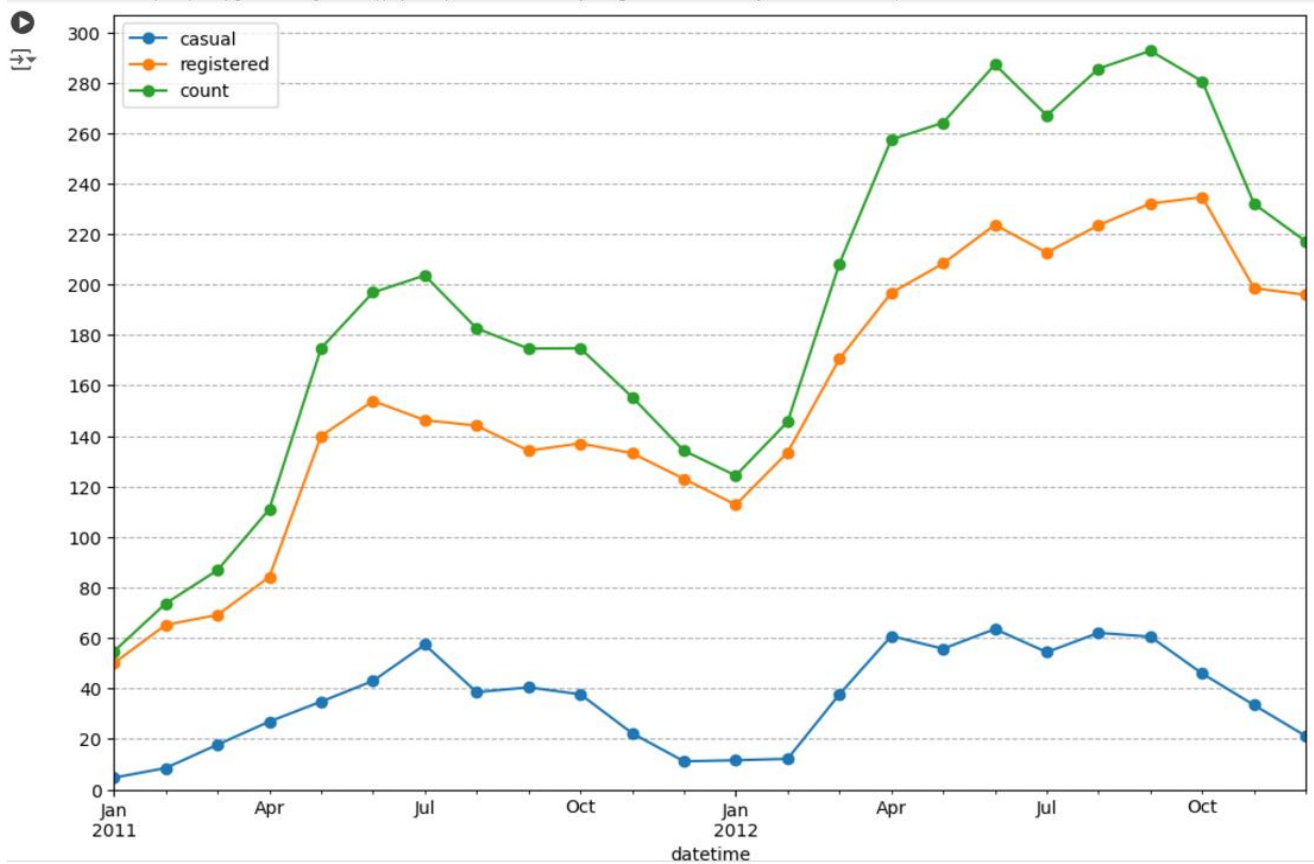
 Timestamp('2012-12-19 23:00:00')

```
df['day'] = df['datetime'].dt.day_name()
df.set_index('datetime', inplace = True)
```

```
plt.figure(figsize = (12, 8))
```

```
df.resample('M')['casual'].mean().plot(kind = 'line', legend = 'casual', marker = 'o')
df.resample('M')['registered'].mean().plot(kind = 'line', legend = 'registered', marker = 'o')
df.resample('M')['count'].mean().plot(kind = 'line', legend = 'count', marker = 'o')

plt.grid(axis = 'y', linestyle = '--')
plt.yticks(np.arange(0, 301, 20))
plt.ylim(0,)
plt.show()
```



I want to know if there is an increase in the average hourly count of rental bikes from the year 2011 to 2012

```
df1 = df.resample('Y')['count'].mean().to_frame().reset_index()

df1['prev_count'] = df1['count'].shift(1)

df1['growth_percent'] = (df1['count'] - df1['prev_count']) * 100 / df1['prev_count']
df1
```

<ipython-input-17-d9b4c71cfba8>:1: FutureWarning: 'Y' is deprecated and will be removed in a future version  
df1 = df.resample('Y')['count'].mean().to\_frame().reset\_index()

	datetime	count	prev_count	growth_percent
0	2011-12-31	144.223349	NaN	NaN
1	2012-12-31	238.560944	144.223349	65.410764

Insights:

- This data suggests that there was substantial growth in the count of the variable over the course of one year.
- The mean total hourly count of rental bikes is 144 for the year 2011 and 239 for the year 2012. An annual growth rate of 65.41 % can be seen in the demand of electric vehicles on an hourly basis.

**How does the average hourly count of rental bikes varies for different month ?**

```
[18] df.reset_index(inplace = True)
```

```
df1 = df.groupby(by = df['datetime'].dt.month)['count'].mean().reset_index()
df1.rename(columns = {'datetime' : 'month'}, inplace = True)

df1['prev_count'] = df1['count'].shift(1)

df1['growth_percent'] = (df1['count'] - df1['prev_count']) * 100 / df1['prev_count']
df1.set_index('month', inplace = True)
df1
```

	count	prev_count	growth_percent
month			
1	90.366516	NaN	NaN
2	110.003330	90.366516	21.730188
3	148.169811	110.003330	34.695751
4	184.160616	148.169811	24.290241
5	219.459430	184.160616	19.167406
6	242.031798	219.459430	10.285440
7	235.325658	242.031798	-2.770768
8	234.118421	235.325658	-0.513007
9	233.805281	234.118421	-0.133753
10	227.699232	233.805281	-2.611596
11	193.677278	227.699232	-14.941620
12	175.614035	193.677278	-9.326465

### Insights:

- The count of rental bikes shows an increasing trend from January to March, with a significant growth rate of 34.70% between February and March.
- The growth rate starts to stabilize from April to June, with a relatively smaller growth rate.
- From July to September, there is a slight decrease in the count of rental bikes, with negative growth rates.
- The count further declines from October to December, with the largest drop observed between October and November (-14.94%).

```

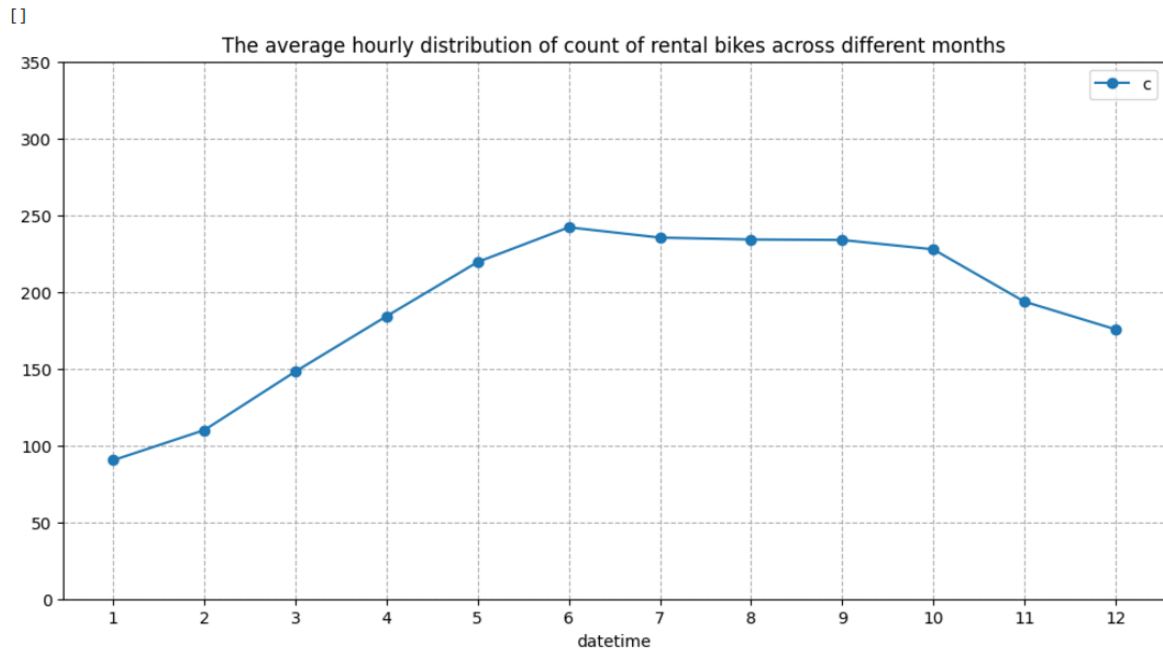
plt.figure(figsize = (12, 6))

plt.title("The average hourly distribution of count of rental bikes across different months")

df.groupby(by = df['datetime'].dt.month)['count'].mean().plot(kind = 'line', marker = 'o')

plt.ylim(0,) # Setting the y-axis limits to start from zero
plt.xticks(np.arange(1, 13)) # Setting the x-ticks to represent the months from 1 to 12
plt.legend('count') # Adding a legend to the plot for the 'count' line.
plt.yticks(np.arange(0, 400, 50))
# Adding gridlines to both the x and y axes with a dashed line style
plt.grid(axis = 'both', linestyle = '--')
plt.plot()

```



### Insights:

- The average hourly count of rental bikes is the highest in the month of June followed by July and August.
- The average hourly count of rental bikes is the lowest in the month of January followed by February and March.

*Overall, these trends suggest a seasonal pattern in the count of rental bikes, with higher demand during the spring and summer months, a slight decline in the fall, and a further decrease in the winter months. It could be useful for the rental bike company to consider these patterns for resource allocation, marketing strategies, and operational planning throughout the year.*

**What is the distribution of average count of rental bikes on an hourly basis in a single day ?**

```
df1 = df.groupby(by = df['datetime'].dt.hour)['count'].mean().reset_index()
df1.rename(columns = {'datetime' : 'hour'}, inplace = True)

df1['prev_count'] = df1['count'].shift(1)

df1['growth_percent'] = (df1['count'] - df1['prev_count']) * 100 / df1['prev_count']
df1.set_index('hour', inplace = True)
df1
```



0	55.138462	NaN	NaN
1	33.859031	55.138462	-38.592718
2	22.899554	33.859031	-32.367959
3	11.757506	22.899554	-48.656179
4	6.407240	11.757506	-45.505110
5	19.767699	6.407240	208.521293
6	76.259341	19.767699	285.777526
7	213.116484	76.259341	179.462793
8	362.769231	213.116484	70.221104
9	221.780220	362.769231	-38.864655
10	175.092308	221.780220	-21.051432
11	210.674725	175.092308	20.322091
12	256.508772	210.674725	21.755835
13	257.787281	256.508772	0.498427
14	243.442982	257.787281	-5.564393
15	254.298246	243.442982	4.459058
16	316.372807	254.298246	24.410141
17	468.765351	316.372807	48.168661
18	430.859649	468.765351	-8.086285
19	315.278509	430.859649	-26.825705
20	228.517544	315.278509	-27.518833
21	173.370614	228.517544	-24.132471
22	133.576754	173.370614	-22.953059
23	89.508772	133.576754	-32.990757



### Insights:

- During the early morning hours (hours 0 to 5), there is a significant decrease in the count, with negative growth percentages ranging from -38.59% to -48.66%.
- However, starting from hour 5, there is a sudden increase in count, with a sharp positive growth percentage of 208.52% observed from hour 4 to hour 5.
- The count continues to rise significantly until reaching its peak at hour 17, with a growth percentage of 48.17% compared to the previous hour.
- After hour 17, there is a gradual decrease in count, with negative growth percentages ranging from -8.08% to -32.99% during the late evening and nighttime hours.

## Univariate Analysis


```
plt.figure(figsize = (12,10))

plt.subplot(2,2,1)
sns.distplot(df["temp"])

plt.subplot(2,2,2)
sns.distplot(df["atemp"])

plt.subplot(2,2,3)
sns.distplot(df["humidity"])

plt.subplot(2,2,4)
sns.distplot(df["windspeed"])
```

 <ipython-input-27-73b6ecc1daba>:13: UserWarning:

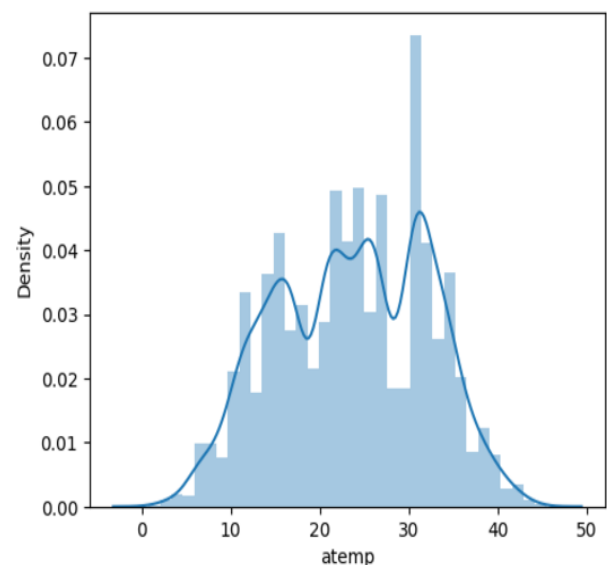
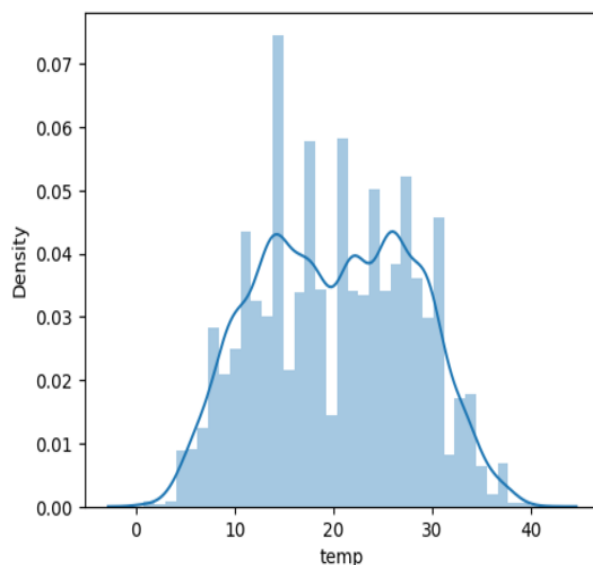
``distplot`` is a deprecated function and will be removed in seaborn v0.14.0.

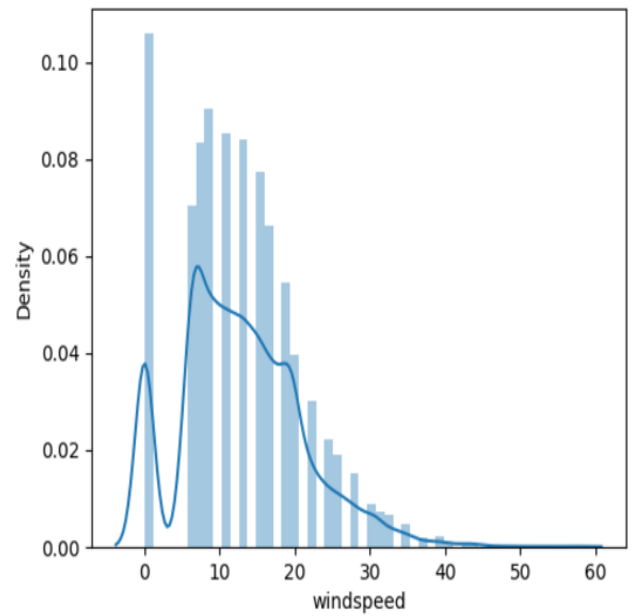
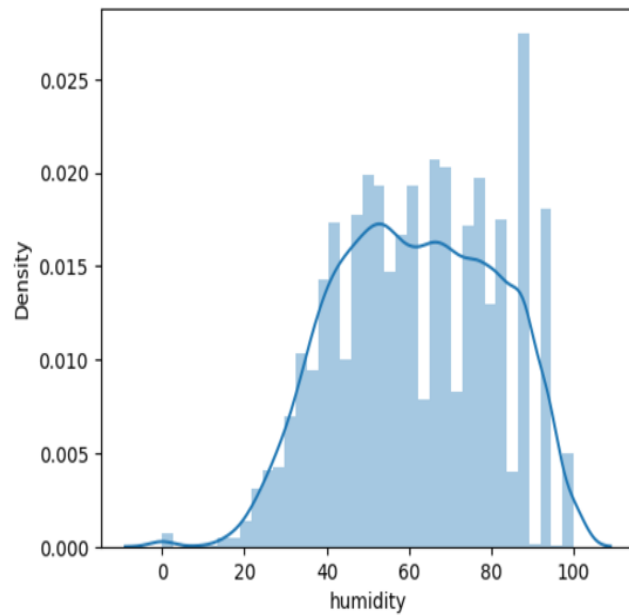
Please adapt your code to use either ``displot`` (a figure-level function with similar flexibility) or ``histplot`` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see <https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

```
sns.distplot(df["windspeed"])
<Axes: xlabel='windspeed', ylabel='Density'>
```

```
sns.distplot(df["windspeed"])
<Axes: xlabel='windspeed', ylabel='Density'>
```





```
plt.figure(figsize = (12,10))

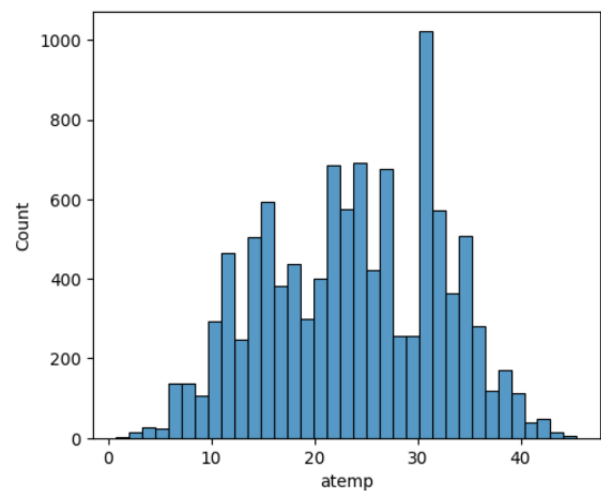
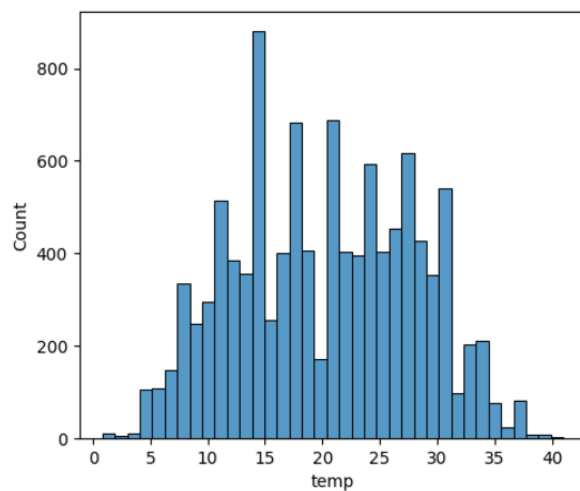
plt.subplot(2,2,1)
sns.histplot(df["temp"])

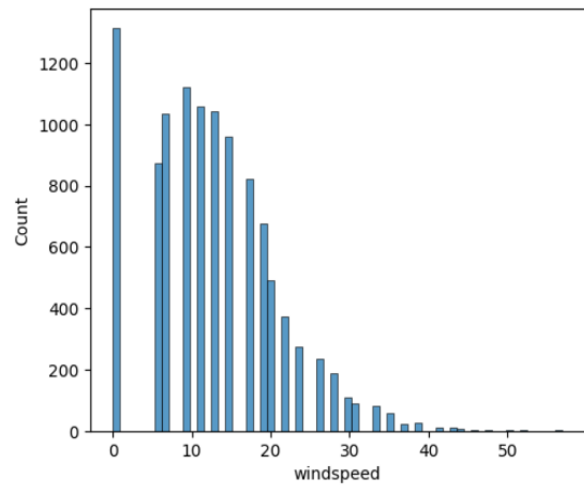
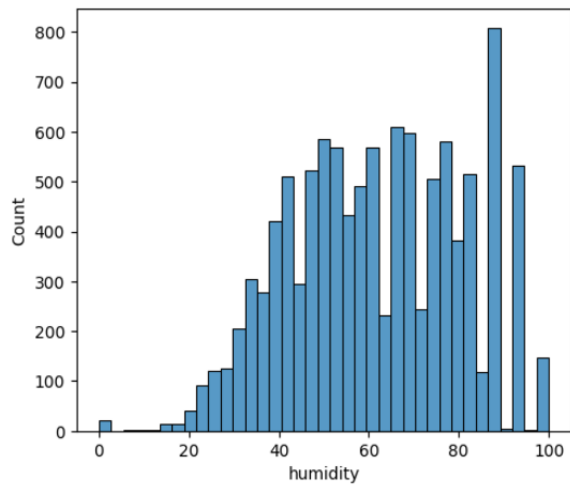
plt.subplot(2,2,2)
sns.histplot(df["atemp"])

plt.subplot(2,2,3)
sns.histplot(df["humidity"])

plt.subplot(2,2,4)
sns.histplot(df["windspeed"])
```

<Axes: xlabel='windspeed', ylabel='Count'>





```

▶ plt.figure(figsize = (12,10))

plt.subplot(2,2,1)
sns.boxplot(df["temp"])

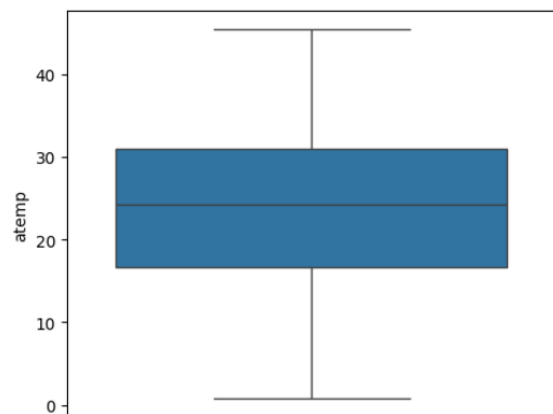
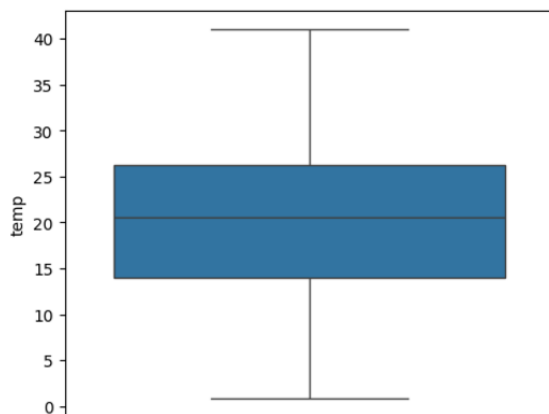
plt.subplot(2,2,2)
sns.boxplot(df["atemp"])

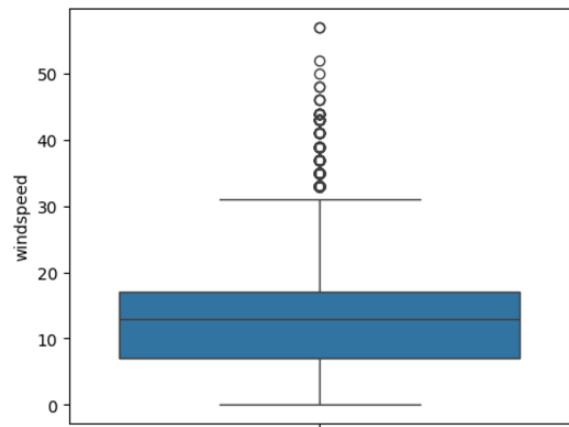
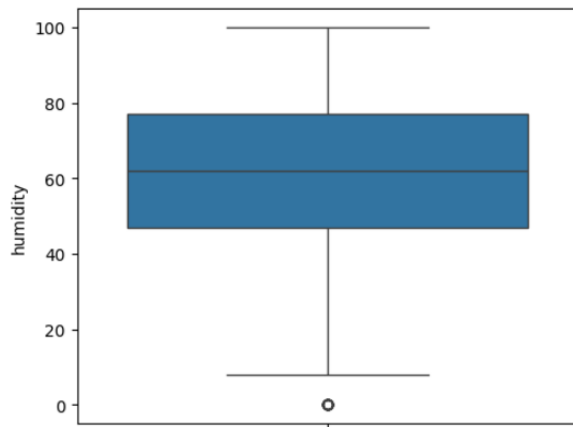
plt.subplot(2,2,3)
sns.boxplot(df["humidity"])

plt.subplot(2,2,4)
sns.boxplot(df["windspeed"])

```

<Axes: ylabel='windspeed'>





```
plt.figure(figsize = (12,10))

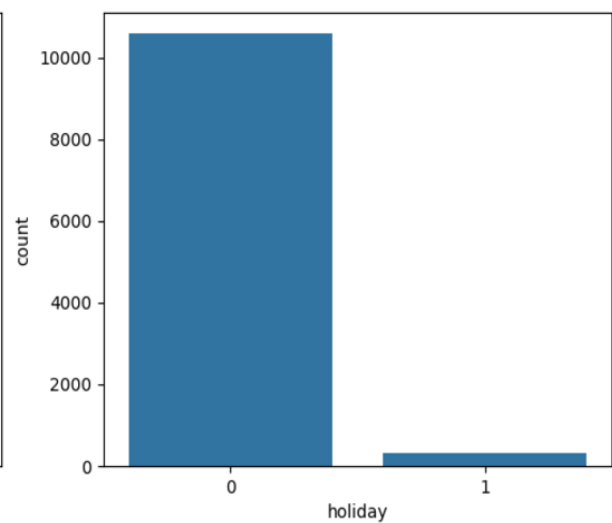
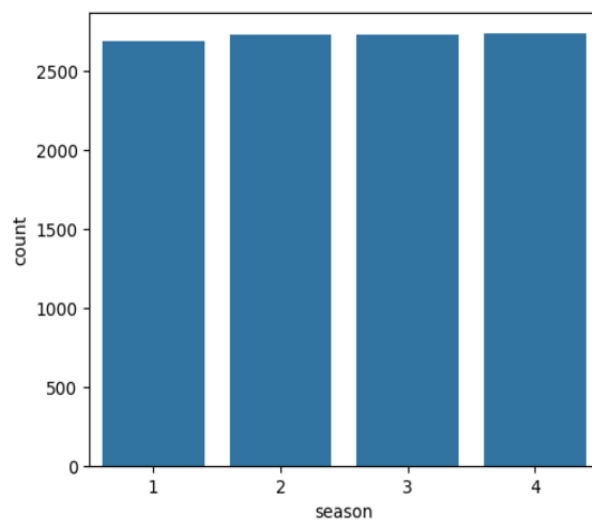
plt.subplot(2,2,1)
sns.countplot(data=df,x="season")

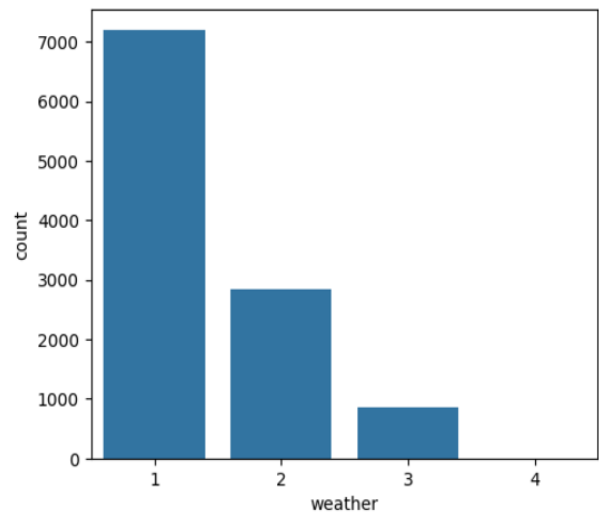
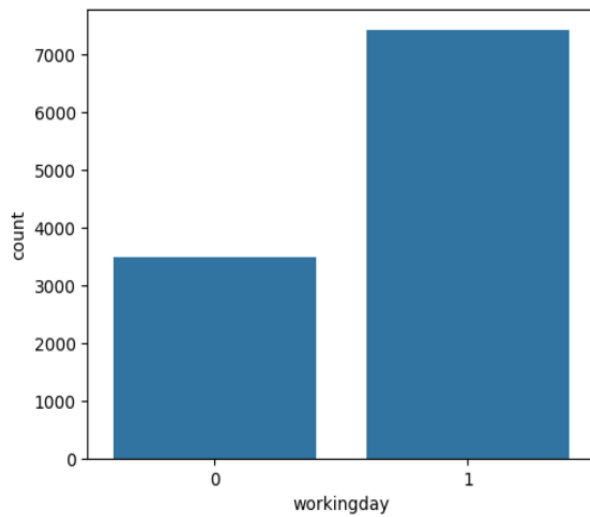
plt.subplot(2,2,2)
sns.countplot(data=df, x="holiday")

plt.subplot(2,2,3)
sns.countplot(data=df,x="workingday")

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

<Axes: xlabel='weather', ylabel='count'>





## Bivariate Analysis

```
plt.figure(figsize=(12,10))

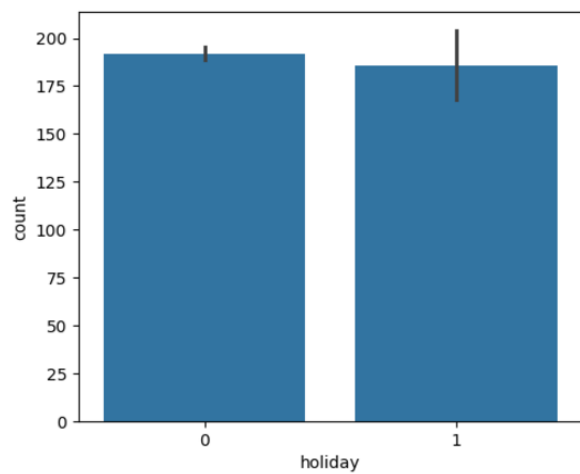
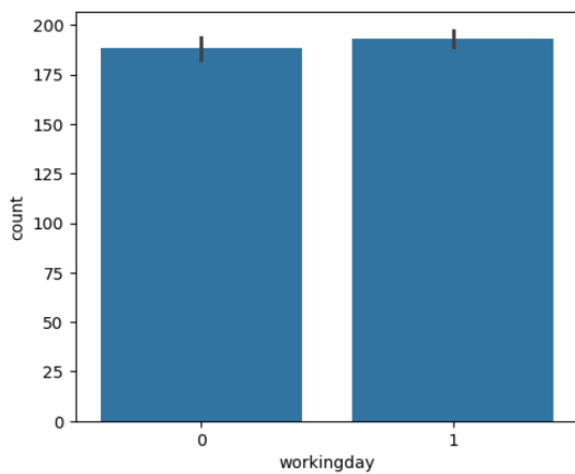
plt.subplot(2,2,1)
sns.barplot(data=df, x="workingday", y="count")

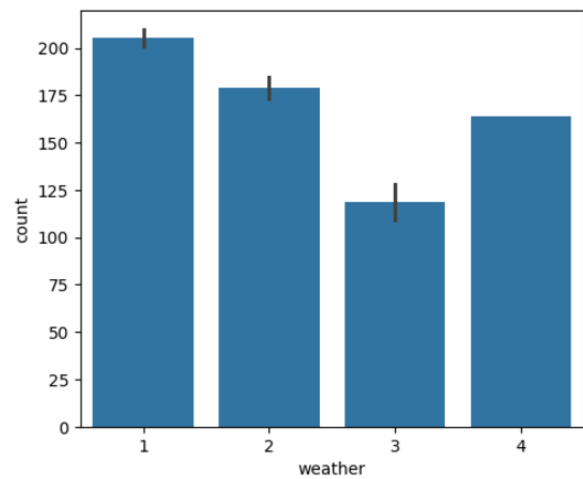
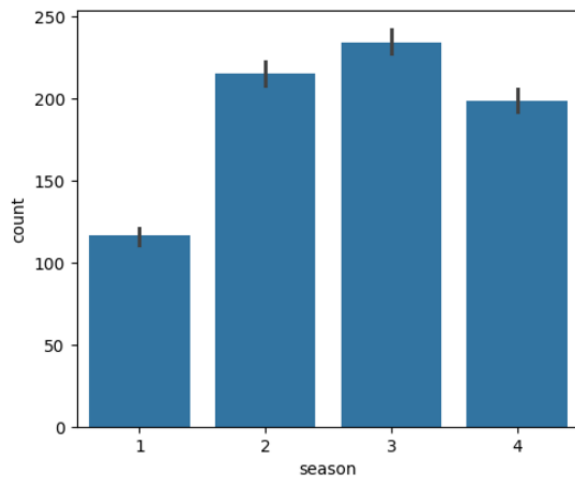
plt.subplot(2,2,2)
sns.barplot(data=df, x="holiday", y="count")

plt.subplot(2,2,3)
sns.barplot(data=df, x="season", y="count")

plt.subplot(2,2,4)
sns.barplot(df, x="weather", y="count")
```

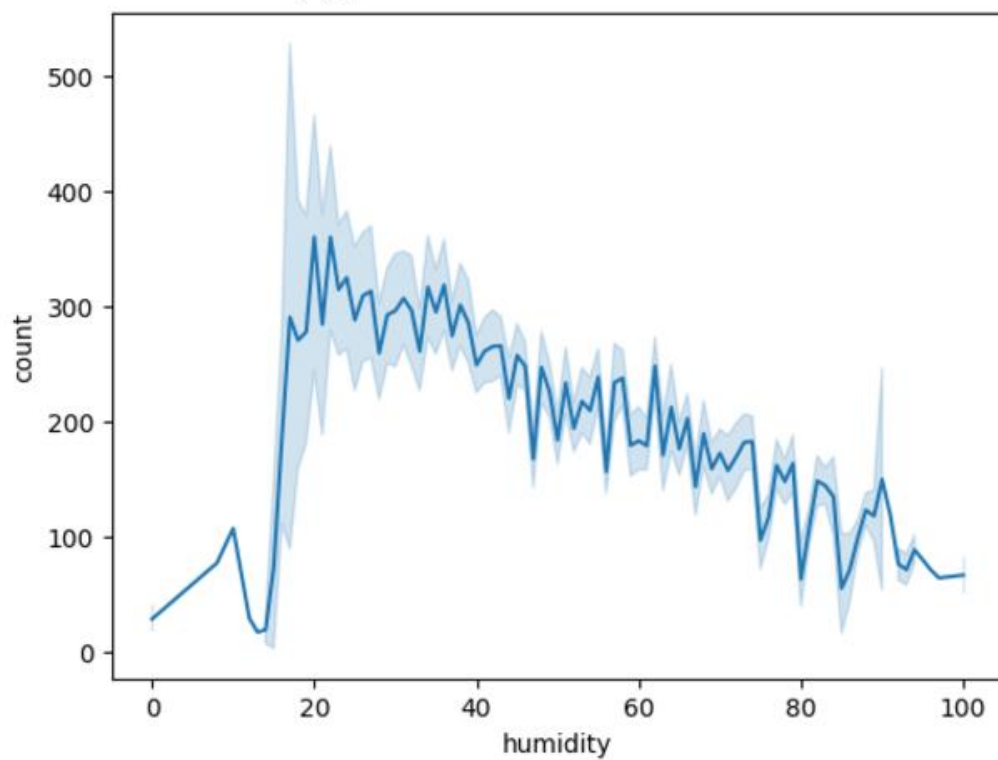
<Axes: xlabel='weather', ylabel='count'>





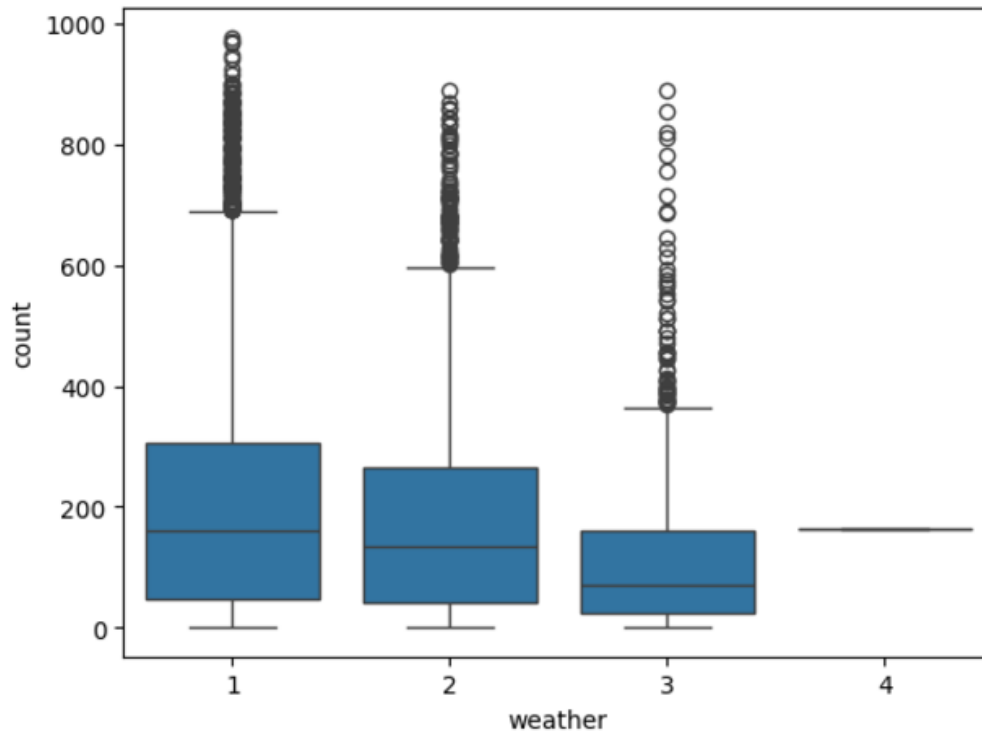
```
sns.lineplot(df, x="humidity",y="count")
```

```
<Axes: xlabel='humidity', ylabel='count'>
```



```
sns.boxplot(df, x="weather", y="count")
```

```
<Axes: xlabel='weather', ylabel='count'>
```



```
#Correlation between different attributes of dataframe
df["season"]=df["season"].astype("object")
df["holiday"]=df["holiday"].astype("object")
df["workingday"]=df["workingday"].astype("object")
df["weather"]=df["weather"].astype("object")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 13 columns):
#   Column          Non-Null Count  Dtype  
---  -
0   datetime         10886 non-null  datetime64[ns]
1   season           10886 non-null  object  
2   holiday          10886 non-null  object  
3   workingday       10886 non-null  object  
4   weather          10886 non-null  object  
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   
12  day              10886 non-null  object  
dtypes: datetime64[ns](1), float64(3), int64(4), object(5)
memory usage: 1.1+ MB
```





humidity: It gives the humidity at a given time, and its value ranges from 0.0 to 100.00

windspeed: It gives the values of windspeed at a given time, and its value ranges from 0.0 to 56.99

casual: It gives a count of casual users at a given time, and its value ranges from 0 to 367

registered: It gives a count of casual users at a given time, and its value ranges from 0 to 886


count: It gives a count of total rental bikes including both casual and registered, and its value ranges from 1 to 977.


Here Outliers are found by IQR method in casual, registered, and count columns, but as dropping or morphing of outliers may affect our statistical significance, so Its better to keep them in our data.


## 2 Sample T-Test

```
[61] #Filtering count based on working day
      working_day_count= df.loc[df["workingday"]==1,"count"]
      non_working_day_count=df.loc[df["workingday"]==0,"count"]
```

```
[62] #Mean and Standard Deviation of count during working day
      working_day_count.mean(), working_day_count.std()
```

 (193.01187263896384, 184.5136590421483)

```
 #Mean and Standard Deviation of count during Non-working day
      non_working_day_count.mean(), non_working_day_count.std()
```

 (188.50662061024755, 173.72401532500032)

**Ho** : mean of working day and non working day is same :  $\mu_1 = \mu_2$

**Ha** : mean of working day is higher than non working day :  $\mu_1 > \mu_2$

*#Let us set significance level 0.05, confidence level 95%*

alpha=0.05

```
[64] #Let us set significance level 0.05, confidence level 95%
      alpha=0.05
```

```
[66] from scipy.stats import ttest_ind

      test_statistic, p_value = ttest_ind(working_day_count, non_working_day_count, alternative="greater")
      test_statistic, p_value
```

```
(1.2096277376026694, 0.11322402113180674)
```

```
if p_value < alpha:
    print("Reject Null Hypothesis Ho")
else:
    print("Fail to Reject Null Hypothesis Ho")
```

```
Fail to Reject Null Hypothesis Ho
```

We have considered a confidence level of 95% in the Test.

The 2 Sample T-Test between the count attributes of the working day and the non-working day has been carried out and We found from the 2 Sample T-test that the means of both samples have no statistically significant difference.

## ANOVA Test

```
[68] #Filtering count based on weather category
      weather_1 = df.loc[df["weather"]==1, "count"]
      weather_2 = df.loc[df["weather"]==2, "count"]
      weather_3 = df.loc[df["weather"]==3, "count"]
      weather_4 = df.loc[df["weather"]==4, "count"]
```

```
weather_4
```

```
count
5631  164
dtype: int64
```

We will do shapiro Test for checking whether our sample follows Gaussian Distribution or not

Null and Alternate Hypothesis for Shapiro Test

**H0:** The sample follows Gaussian Distribution

**Ha:** The sample does not follow Gaussian Distribution

```
from scipy.stats import shapiro

test_statistics, p_value = shapiro(weather_1)
print("p-value:", round(p_value,4))
if p_value < alpha:
    print("Reject Null Hypothesis, Sample does not follow Gaussian Distribution")
else:
    print("Fail to Reject Null Hypothesis, Sample follows Gaussian Distribution")
```

```
↗ p-value: 0.0
Reject Null Hypothesis, Sample does not follow Gaussian Distribution
/usr/local/lib/python3.10/dist-packages/scipy/stats/_axis_nan_policy.py:531: UserWarning: scipy.stats.shapiro:
    res = hypotest_fun_out(*samples, **kws)
```

```
[74] test_statistics, p_value = shapiro(weather_2)
print("p-value:", round(p_value,4))
if p_value < alpha:
    print("Reject Null Hypothesis, Sample does not follow Gaussian Distribution")
else:
    print("Fail to Reject Null Hypothesis, Sample follows Gaussian Distribution")
```

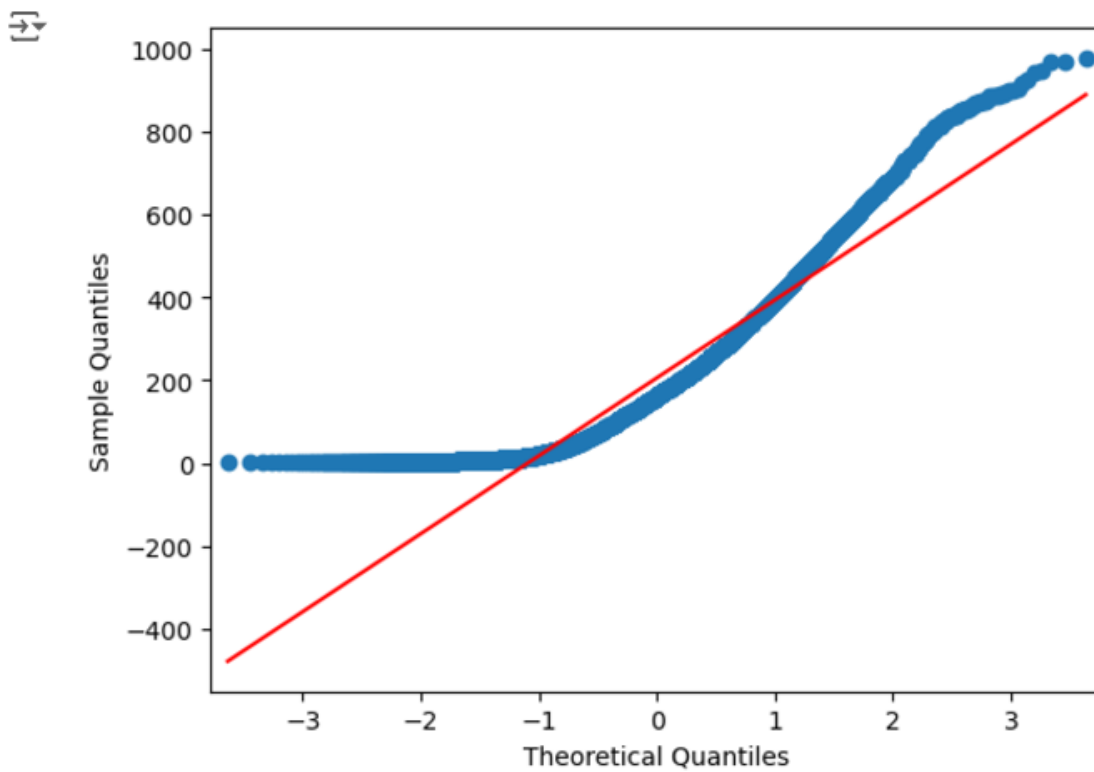
```
↗ p-value: 0.0
Reject Null Hypothesis, Sample does not follow Gaussian Distribution
```

```
test_statistics, p_value = shapiro(weather_3)
print("p-value:", round(p_value,4))
if p_value < alpha:
    print("Reject Null Hypothesis, Sample does not follow Gaussian Distribution")
else:
    print("Fail to Reject Null Hypothesis, Sample follows Gaussian Distribution")
```

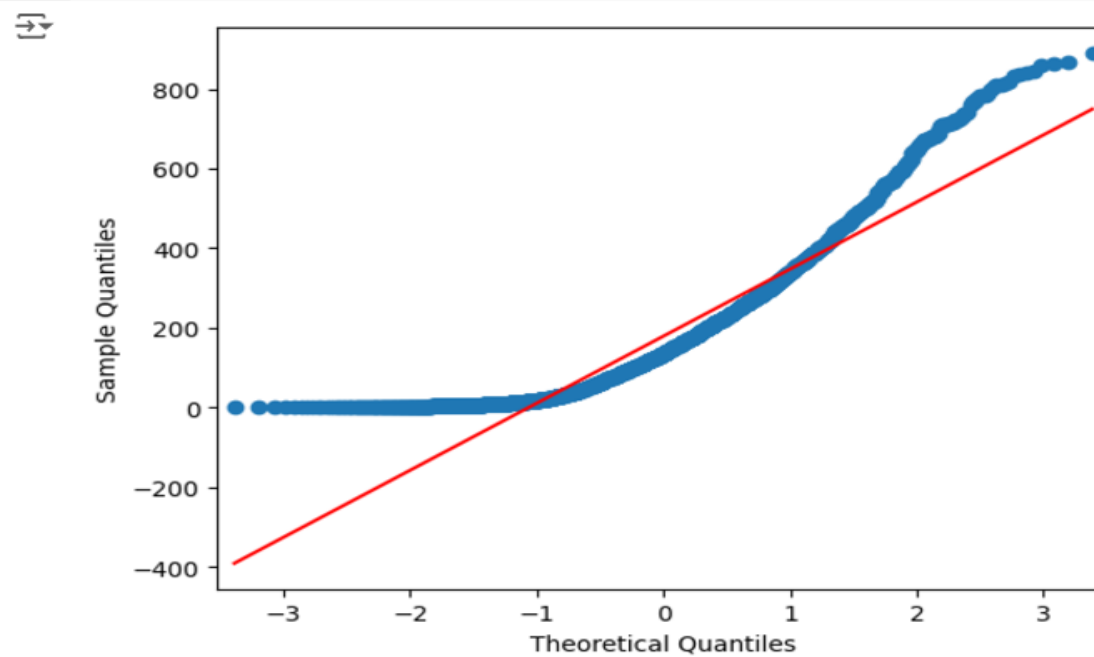
```
↗ p-value: 0.0
Reject Null Hypothesis, Sample does not follow Gaussian Distribution
```

**QQ – plot**

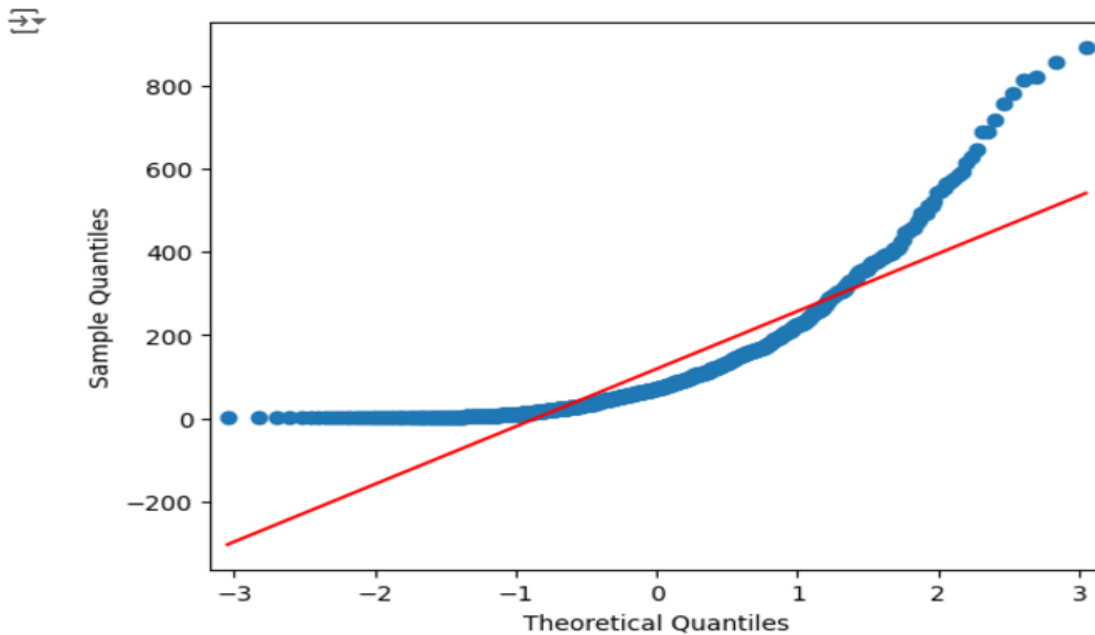
```
#Let's check for normality based on q-q plot  
qqplot(weather_1,line="s")  
plt.show()
```



```
#Let's check for normality based on q-q plot  
qqplot(weather_2,line="s")  
plt.show()
```



```
#Let's check for normality based on q-q plot
qqplot(weather_3,line="s")
plt.show()
```



## Levene test

We will do levene test to check whether variance of the samples are same or not

Null Hypothesis and Alternate Hypothesis for Levene Test

**H0:** Variances of the samples are same

**Ha:** Variances of the samples are not same

```
[79] #Let us set significance level 0.05, confidence level 95%
alpha=0.05
```

```
test_statistics, p_value=levene(weather_1,weather_2, weather_3)
print("p-value:", round(p_value,4))
if p_value < alpha:
    print("Reject Null Hypothesis, Variances of the samples are not same")
else:
    print("Fail to Reject Null Hypothesis, Variances of the samples are same")
```

```
p-value: 0.0
Reject Null Hypothesis, Variances of the samples are not same
```

As we have done shapiro and Q-Q Plot for checking Normality and Levene Test for checking Variance.

We have found that Samples do not follow Gaussian Distribution and do not have similar variance. So we will go for Kruskal-Wallis Test

Null and Alternate Hypothesis for Kruskal Wallis Test

**H0:** mean of total rental bikes of different weathers are same

**Ha:** mean of total rental bikes of different weathers are not same

```
[81] #Let us set significance level 0.05, confidence level 95%  
      alpha=0.05
```

```
#p-value calculation  
test_statistics,p_value=kruskal(weather_1,weather_2,weather_3)  
print("p-value:", round(p_value,4))  
if p_value < alpha:  
    print("Reject Null Hypothesis, mean of total rental bikes of different weathers are not same")  
else:  
    print("Fail to Reject Null Hypothesis, mean of total rental bikes of different weathers are same")
```

```
↗ p-value: 0.0  
Reject Null Hypothesis, mean of total rental bikes of different weathers are not same
```

```
#Filtering count based on weather category  
season_1 = df.loc[df["season"]==1, "count"]  
season_2 = df.loc[df["season"]==2, "count"]  
season_3 = df.loc[df["season"]==3, "count"]  
season_4 = df.loc[df["season"]==4, "count"]
```

We will do shapiro Test for checking whether our sample follows Gaussian Distribution or not

Null and Alternate Hypothesis for Shapiro Test

**H0:** The sample follows Gaussian Distribution

**Ha:** The sample does not follow Gaussian Distribution

```
[84] #Let us set significance level 0.05, confidence level 95%
      alpha=0.05
```

```
▶ #p-value calculation
test_statistics, p_value = shapiro(season_1)
print("p-value:", round(p_value,4))
if p_value < alpha:
    print("Reject Null Hypothesis, Sample does not follow Gaussian Distribution")
else:
    print("Fail to Reject Null Hypothesis, Sample follows Gaussian Distribution")
```

```
⇒ p-value: 0.0
   Reject Null Hypothesis, Sample does not follow Gaussian Distribution
```

```
▶ #p-value calculation
test_statistics, p_value = shapiro(season_2)
print("p-value:", round(p_value,4))
if p_value < alpha:
    print("Reject Null Hypothesis, Sample does not follow Gaussian Distribution")
else:
    print("Fail to Reject Null Hypothesis, Sample follows Gaussian Distribution")
```

```
⇒ p-value: 0.0
   Reject Null Hypothesis, Sample does not follow Gaussian Distribution
```

```
[87] #p-value calculation
test_statistics, p_value = shapiro(season_3)
print("p-value:", round(p_value,4))
if p_value < alpha:
    print("Reject Null Hypothesis, Sample does not follow Gaussian Distribution")
else:
    print("Fail to Reject Null Hypothesis, Sample follows Gaussian Distribution")
```

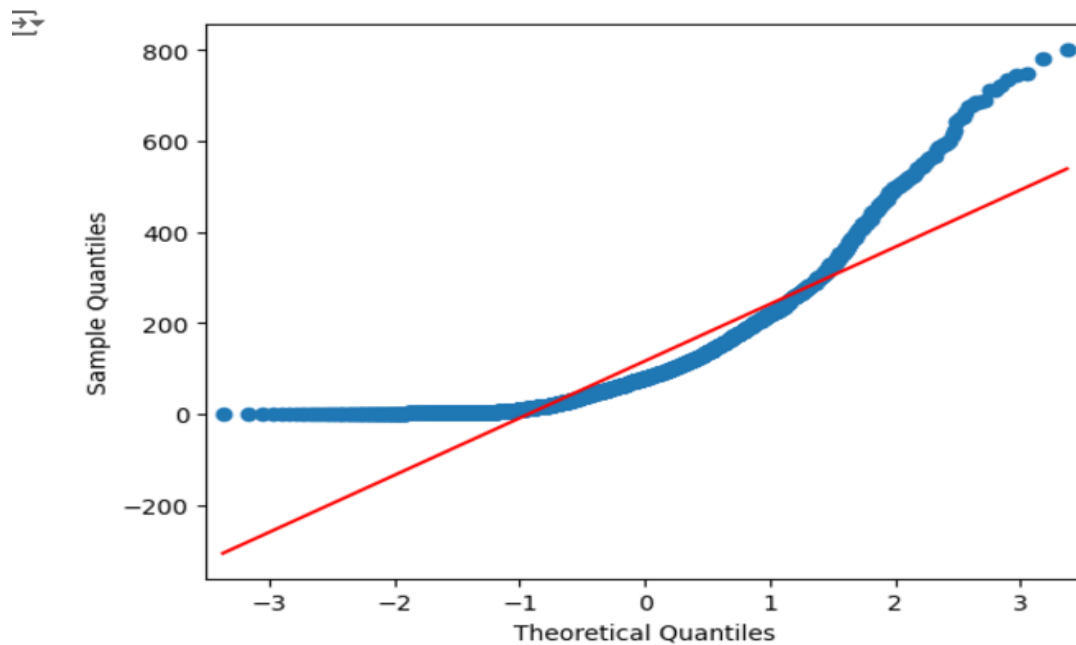
```
⇒ p-value: 0.0
   Reject Null Hypothesis, Sample does not follow Gaussian Distribution
```

```
[88] #p-value calculation
test_statistics, p_value = shapiro(season_4)
print("p-value:", round(p_value,4))
if p_value < alpha:
    print("Reject Null Hypothesis, Sample does not follow Gaussian Distribution")
else:
    print("Fail to Reject Null Hypothesis, Sample follows Gaussian Distribution")
```

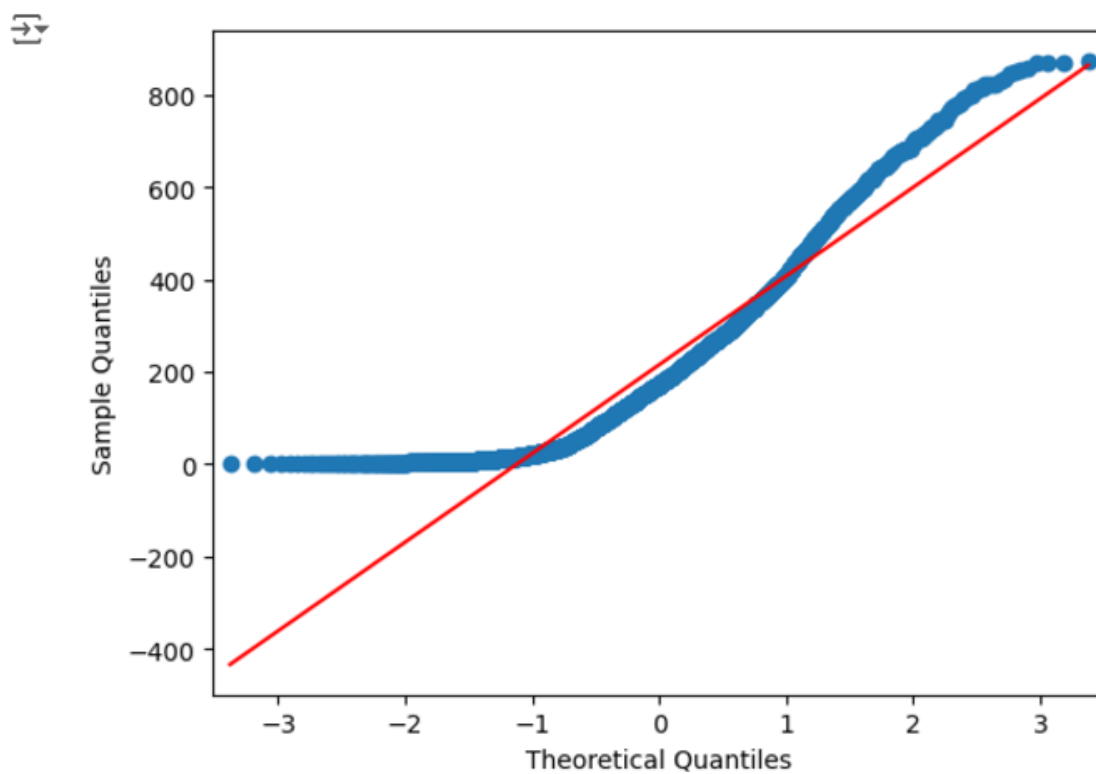
```
⇒ p-value: 0.0
   Reject Null Hypothesis, Sample does not follow Gaussian Distribution
```



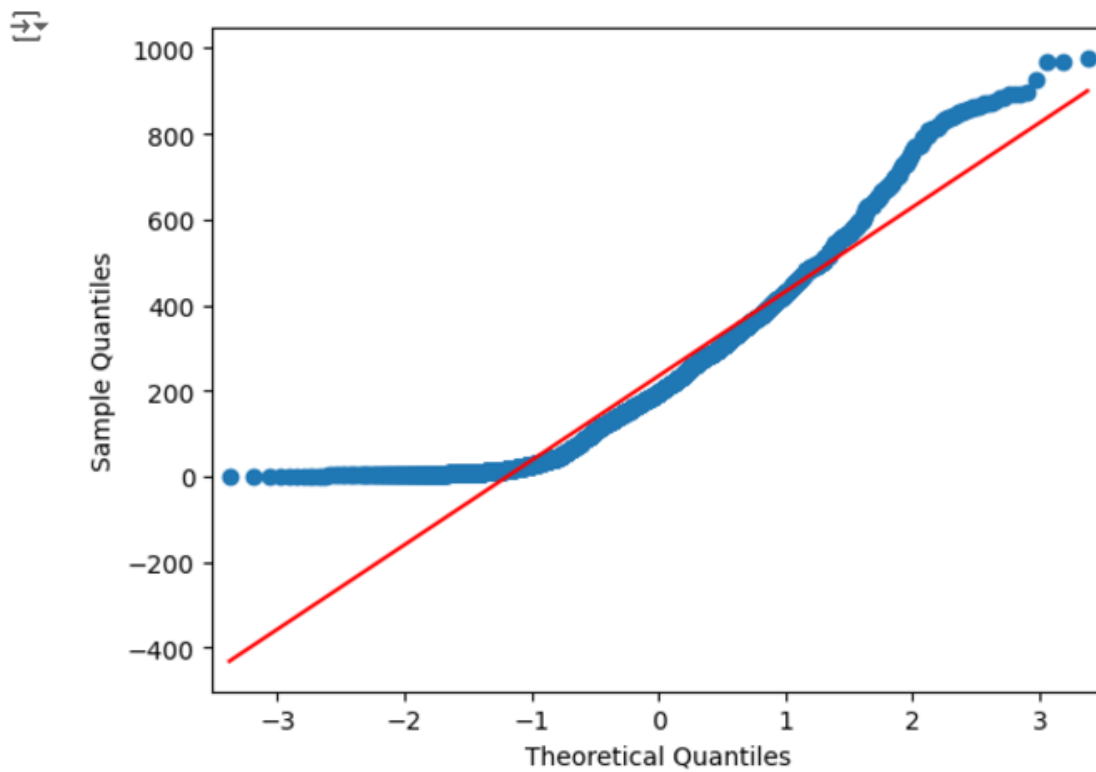
```
#Let's check for normality based on q-q plot  
qqplot(season_1,line="s")  
plt.show()
```



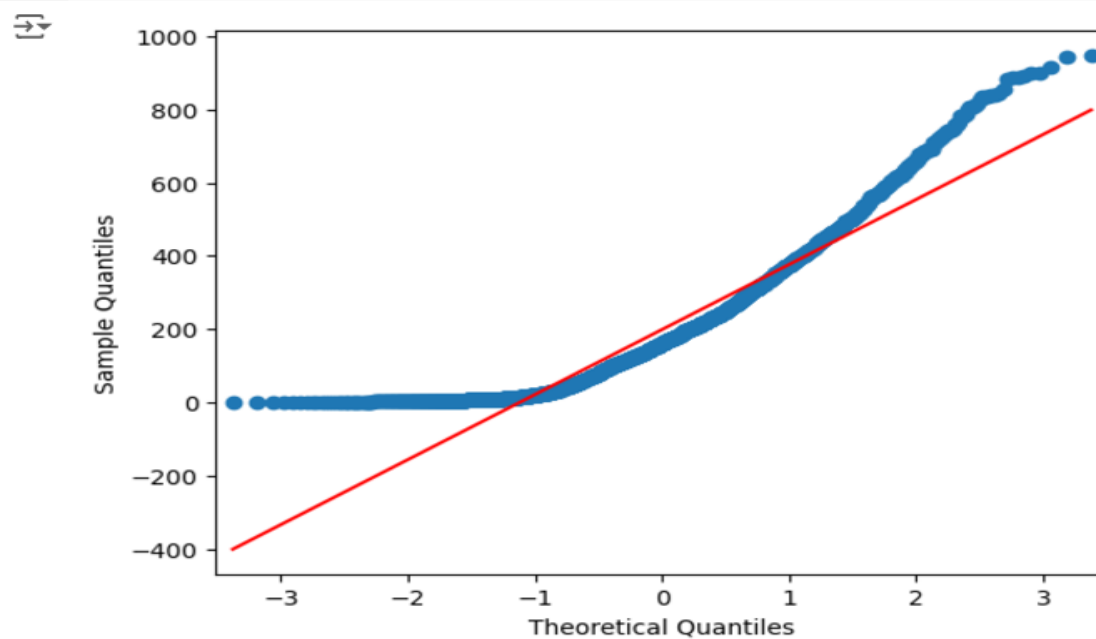
```
#Let's check for normality based on q-q plot  
qqplot(season_2,line="s")  
plt.show()
```



```
#Let's check for normality based on q-q plot  
qqplot(season_3,line="s")  
plt.show()
```



```
#Let's check for normality based on q-q plot  
qqplot(season_4,line="s")  
plt.show()
```



**We have considered a confidence level of 95% in the Test.**

For the assumptions testing like the shapiro-wilk test, q-q plot, and levene test.


As samples fail for normality tests and variance tests, we have carried out Kruskal Wallis Test.

From the Kruskal Wallis Test, It can be said that the Means of total rental bikes for different weathers has a statistically significant difference.

From the Kruskal Wallis Test, It can be said that the Means of total rental bikes for different seasons has a statistically significant difference.

## Chi-square Test

```
#Creating Contingency table between categorical attributes weather and season
ws= pd.crosstab(df["weather"], df["season"])
ws
```



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

Next steps: [Generate code with ws](#)

[View recommended plots](#)

[New interactive sheet](#)

```
#Here in our contingency table there is value count of 1 and 0 for weather type 4
#we can not do chi-square test as minimum frequency to run chi-square test is 5
ws.loc[1:3,:]
```



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

## Here For Chi-Square Test between weather and Season

Null and Alternate Hypothesis

**H0:** Seasons and weather are independent

**Ha:** Seasons and weather are dependent on each other

```
[99] #Let us set significance level 0.05, confidence level 95%
      alpha=0.05
```

```
▶ #p-value calculation
test_statistics,p_value, dof, exp=chi2_contingency(ws)
print("p-value:", round(p_value,4))
if p_value < alpha:
    print("Reject Null Hypothesis, Seasons and weather are dependent on each other")
else:
    print("Fail to Reject Null Hypothesis, Seasons and weather are independent")
```

```
➡ p-value: 0.0
   Reject Null Hypothesis, Seasons and weather are dependent on each other
```

```
▶ #p-value calculation
test_statistics,p_value, dof, exp=chi2_contingency(ws.loc[1:3,:])
print("p-value:", round(p_value,4))
if p_value < alpha:
    print("Reject Null Hypothesis, Seasons and weather are dependent on each other")
else:
    print("Fail to Reject Null Hypothesis, Seasons and weather are independent")
```

```
➡ p-value: 0.0
   Reject Null Hypothesis, Seasons and weather are dependent on each other
```

### **We have considered a confidence level of 95% in the Test.**

From the Chi-Square Test, We can say that weather and season are depended on each other.

## **Insights**

Weather and seasons are dependent on each other.

Total rental bikes are depended on the weather. the mean value for the total rental bikes for the weather 1st category is high compared to others.

Total rental bikes are also depended on the seasons. the mean value for total rental bikes for fall is higher compared to other, during spring there is the lowest number of users.

There is no statistical difference in the mean of the total rental bikes on working days and non-working days

Most days in the city are of the weather of 1st category.

Temperature and total rental bikes are correlated and humidity and total rental bikes are negatively correlated.

casual users and total rental bikes are less correlated compared to registered users and total rental bikes.

## **Recommendations**

During spring, Yulu should provide some discounts and offers to increase the use of rental bikes.

During weather of rain, The mean of total rental bikes is lower than others. As Yulu provides bike services, customers can't use it in rainy times. so Yulu should provide some roofs or cab services during this weather.

As humidity increases the total number of rental bikes decreases, so, Yulu should provide benefits during these humid days.

Yulu can increase the use of rental bikes by providing some city tour offers, events, or campaigns during non-working days.

Yulu can convert its casual users to registered users by providing some discounts or registration offers to convert casual users to registered users.

As mostly there is clear weather, Yulu should focus on the increase in total rental bikes during clear weather days.