Yulu Case Study

DAV 3 - Hypothesis Testing

About

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!

Problem

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.

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

Exploratory Data Analysis

Data Cleaning & Exploration

In [1]:
```
# Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('bike_sharing.csv')
df.head()
```

Out[2]:

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

In [3]:
```
df.rename(columns={'atemp':'feeling_temp','temp':'actual_temp','count':'total_users
```

Observation 💡

In [5]:
```
rows = df.shape[0]
cols = df.shape[1]
print(f'This data set contains \n {rows} rows \n {cols} columns')
```

```
This data set contains
 10886 rows
 12 columns
```

In [6]:
```
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   actual_temp       10886 non-null  float64
 6   feeling_temp      10886 non-null  float64
 7   humidity          10886 non-null  int64  
 8   windspeed         10886 non-null  float64
 9   casual_users      10886 non-null  int64  
 10  registered_users  10886 non-null  int64  
 11  total_users       10886 non-null  int64  
dtypes: float64(3), int64(8), object(1)
memory usage: 1020.7+ KB
```

In [7]:
```
df.dtypes
```

Out[7]:
```
datetime             object
season                int64
holiday               int64
workingday            int64
weather               int64
actual_temp         float64
feeling_temp        float64
humidity              int64
windspeed           float64
casual_users          int64
registered_users      int64
total_users           int64
dtype: object
```

Observation 💡

- There is Only One Datetime object present in the given df **typecasting needed**
- There are 4 Categorical Variables present in the given df i.e., **season, holiday, working_day, and weather**
- Remaining 6 Columns are Continous Variables

In [8]:
```
df.isnull().sum() / df.shape[0] # Null Value %ge
```

Out[8]:
```
datetime            0.0
season              0.0
holiday             0.0
workingday          0.0
weather             0.0
actual_temp         0.0
feeling_temp        0.0
humidity            0.0
windspeed           0.0
casual_users        0.0
registered_users    0.0
total_users         0.0
dtype: float64
```

In [9]:
```
df[df.duplicated()]
```

Out[9]:

datetime	season	holiday	workingday	weather	actual_temp	feeling_temp	humidity	...

Observation 💡

- No Null Values in any column
- No Duplicate Values in entire df

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

Out[10]:

	season	holiday	workingday	weather	actual_temp	feeling_temp
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000

In [11]:
```
df.describe(include='object')
```

Out[11]:

	datetime
count	10886
unique	10886
top	2011-01-01 00:00:00
freq	1

In [12]:
```
df.isnull().sum() / df.shape[0] # Null Value %ge
```

Out[12]:
```
datetime            0.0
season              0.0
holiday             0.0
workingday          0.0
weather             0.0
actual_temp         0.0
feeling_temp        0.0
humidity            0.0
windspeed           0.0
casual_users        0.0
registered_users    0.0
total_users         0.0
dtype: float64
```

Season Encoding

In [13]:
```
def season_encode(season):
    if season == 1:
        return 'spring'
    elif season == 2:
        return 'summer'
    elif season == 3:
        return 'fall'
    else:
        return 'winter'
```

In [14]:
```
df['season'] = df['season'].map(season_encode)
```

Holiday Encoding

In [15]:
```
def holiday_encode(holiday):
    if holiday == 1:
        return 'Y'
    else:
        return 'N'
```

In [16]:
```
df['holiday'] = df['holiday'].map(holiday_encode)
```

Workingday Encoding

In [17]:
```
def workingday_encode(workingday):
    if workingday == 1:
        return 'Y'
    else:
        return 'N'
```

In [18]:
```
df['workingday'] = df['workingday'].map(workingday_encode)
```

weather Encoding

In [19]:
```
def weather_encode(weather):
    if weather == 1:
        return 'Clear'
    elif weather == 2:
        return 'Cloudy'
    elif weather == 3:
        return 'Light Rain'
    else:
        return 'Heavy Rain'
```

In [20]:
```
df['weather'] = df['weather'].map(weather_encode)
```

In [21]:
```
df.head()
```

Out[21]:

	datetime	season	holiday	workingday	weather	actual_temp	feeling_temp	humidity
0	2011-01-01 00:00:00	spring	N	N	Clear	9.84	14.395	81
1	2011-01-01 01:00:00	spring	N	N	Clear	9.02	13.635	80
2	2011-01-01 02:00:00	spring	N	N	Clear	9.02	13.635	80
3	2011-01-01 03:00:00	spring	N	N	Clear	9.84	14.395	75
4	2011-01-01 04:00:00	spring	N	N	Clear	9.84	14.395	75

Datetime typecasting

In [22]:
```
df['datetime']= pd.to_datetime(df['datetime'])
```

In [23]:
```
df['date'] = df['datetime'].dt.date
df['year'] = df['datetime'].dt.strftime('%Y')
```

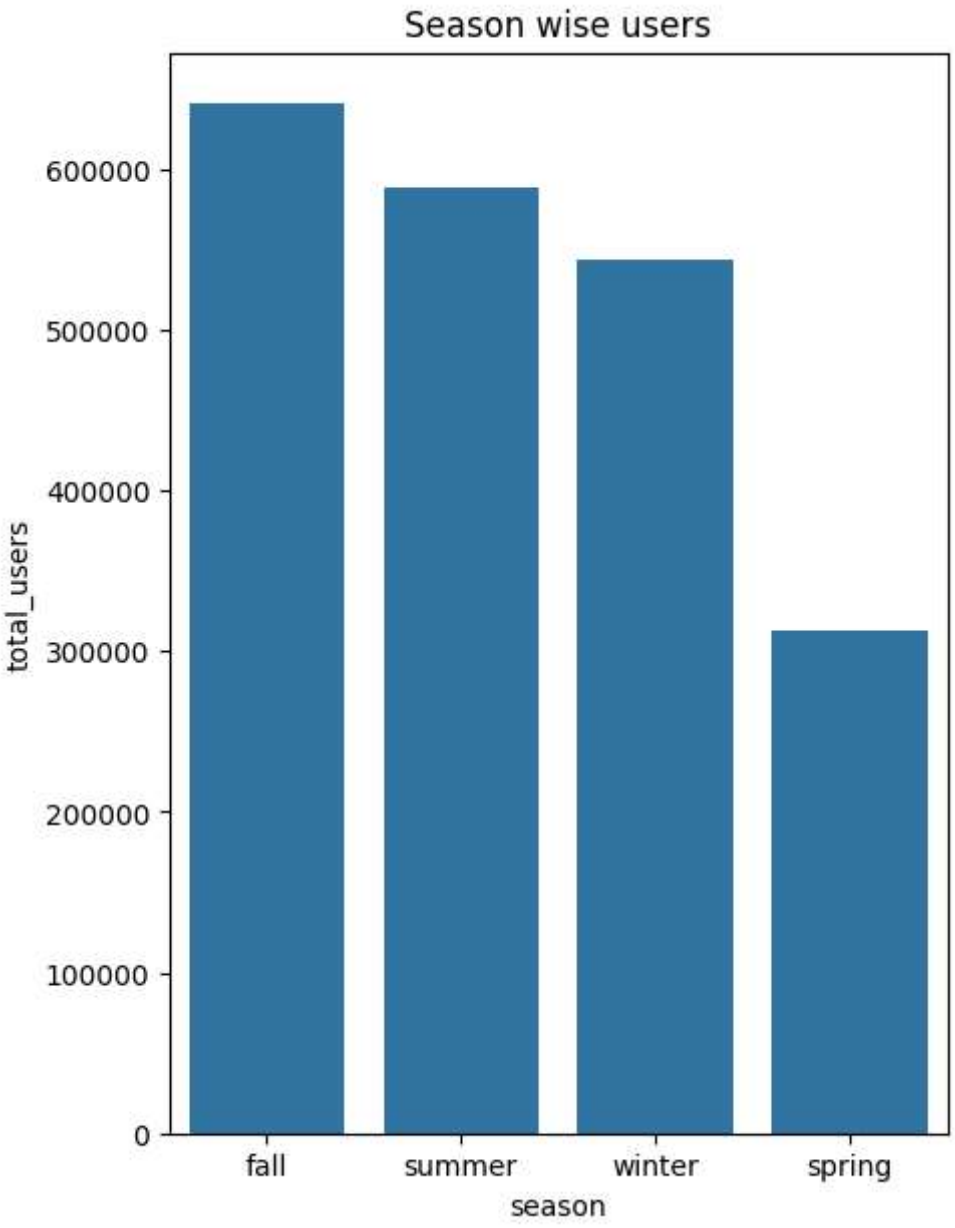
```
df['month'] = df['datetime'].dt.strftime('%B')
```

In [24]:
```
df['Hours'] = df['datetime'].dt.strftime("%H")
df['Hours'] = df['Hours'].astype(np.int64)
```

Univariate Analysis

Seasonwise Users

In [140…
```
plt.figure(figsize=(5,7))
sns.barplot(df.groupby('season')['total_users'].sum().round().sort_values(ascending
plt.title('Season wise users')
plt.show()
```



In [27]:
```
np.round(df.groupby('season')['total_users'].aggregate('sum').sort_values(ascending
```

Out[27]:
```
season
fall      30.72
summer    28.21
winter    26.09
spring    14.98
Name: total_users, dtype: float64
```

Observation 💡

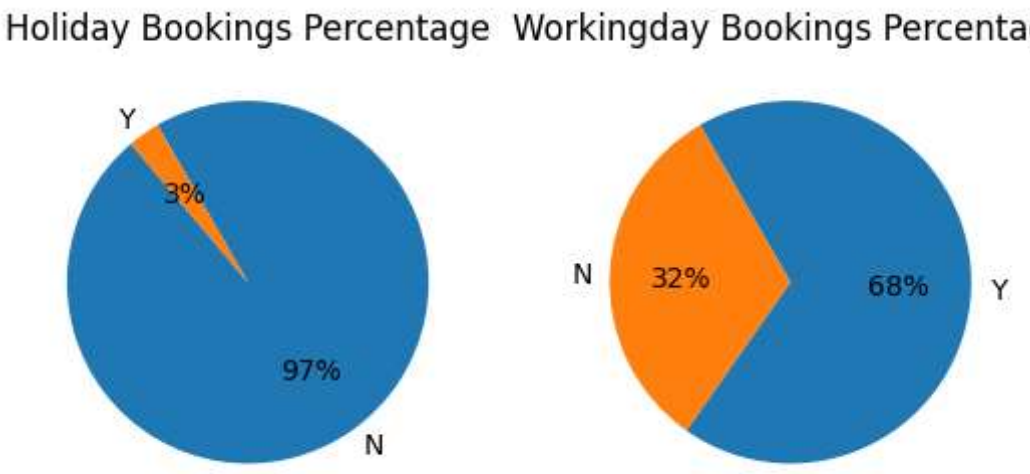
- Spring had low avalibility of users

Holidays and Working Days

In [28]:
```
df['holiday'].value_counts()
```

Out[28]:
```
holiday
N    10575
Y      311
Name: count, dtype: int64
```

In [29]:
```
hcounts = df['holiday'].value_counts().values.tolist()
labels = df['holiday'].value_counts().index
hlabels = list([labels[0],labels[1]])
hcounts,hlabels

wdcounts = df['workingday'].value_counts().values.tolist()
labels = df['workingday'].value_counts().index
wdlabels = list([labels[0],labels[1]])
wdcounts,wdlabels

plt.subplots(1,2)

plt.subplot(1,2,1)
plt.pie(x=hcounts,labels=hlabels,startangle=120,autopct='%1.0f%%',counterclock=Fals
plt.title('Holiday Bookings Percentage')

plt.subplot(1,2,2)
plt.pie(x=wdcounts,labels=wdlabels,startangle=120,autopct='%1.0f%%',counterclock=Fa
plt.title('Workingday Bookings Percentage')

plt.show()
```



In [30]:
```
holiday_per = round((len(df[df['holiday'] =='Y'])/df.shape[0])*100,2)
working_day_per = round((len(df[df['workingday'] =='Y'])/df.shape[0])*100,2)

print(f'Holiday Bookings Pecentage : {holiday_per}')
print(f'workingday Bookings Pecentage : {working_day_per}')
```

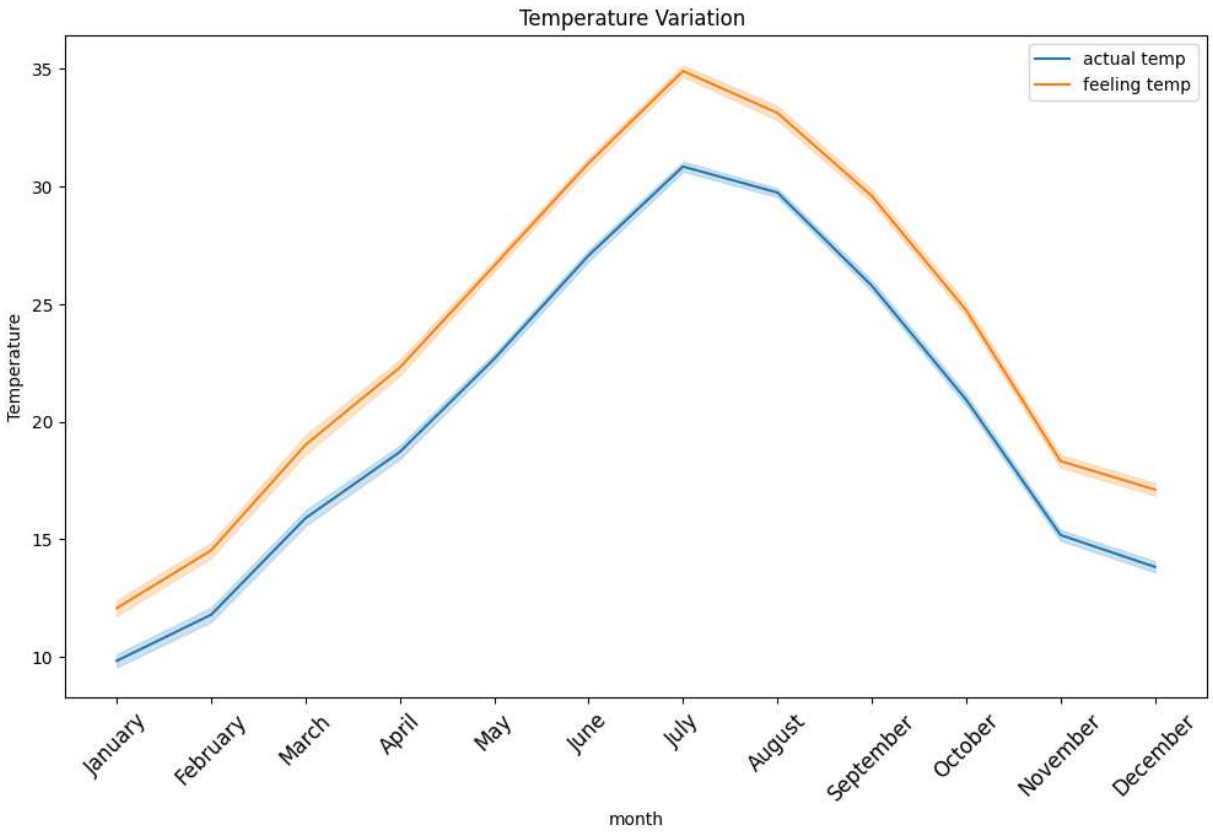
```
Holiday Bookings Pecentage : 2.86
workingday Bookings Pecentage : 68.09
```

Observation 💡

- Only 3% of Bookings made on Holiday !!
- Where as Workingday bookings merely 68% !

Temperature Variation

In [142…
```
plt.figure(figsize=(12,7))
sns.lineplot(data=df,x='month',y='actual_temp',label='actual temp')
sns.lineplot(data=df,x='month',y='feeling_temp',label='feeling temp')
plt.ylabel('Temperature')
plt.xticks(rotation=45,fontsize=12)
plt.title('Temperature Variation ')
plt.show()
```



Numerical Variables Attributes

In [32]:
```
# Columns Segregation
num_cols = [df.columns[i] for i in range(cols) if df[df.columns[i]].dtype=='int64'
cat_cols = [df.columns[i] for i in range(cols) if df[df.columns[i]].dtype=='object'
print(f'Numerical_Columns : {num_cols}')
print(f'Categorical_Columns : {cat_cols}')
```

```
Numerical_Columns : ['actual_temp', 'feeling_temp', 'humidity', 'windspeed', 'casual
_users', 'registered_users', 'total_users']
Categorical_Columns : ['season', 'holiday', 'workingday', 'weather']
```

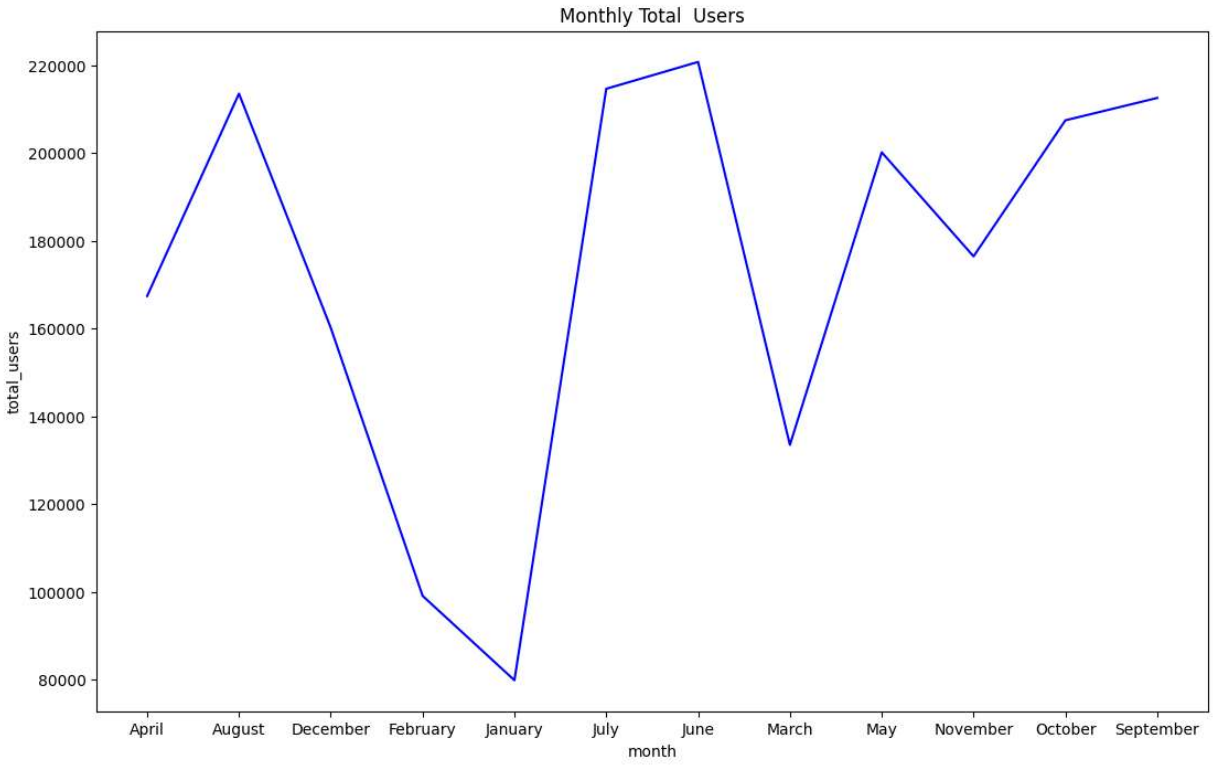
Observation 💡

- These are the Observations of Numerical variables across Months
  - Unsual Windspeeds Happening in Spring
  - Low Availibility of users in First 3 Months (spring)
  - Casual Users are Very High in Middle Months
  - There is Constant availibility of Registered Users from June

Bivariate Analysis

Monthly Users

In [35]:
```
users_per_year = df.groupby('month')['total_users'].sum()
plt.figure(figsize=(13,8))
sns.lineplot(users_per_year,color='b')
plt.title('Monthly Total  Users')
plt.show()
```

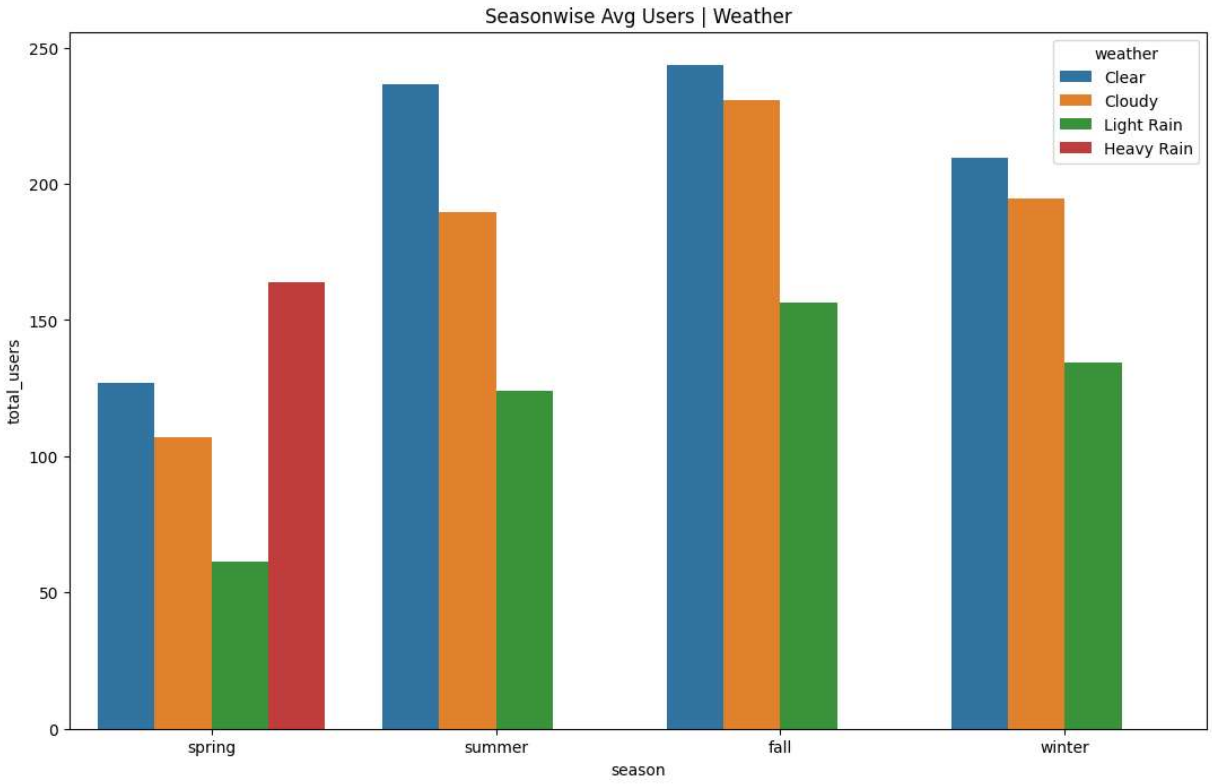


Observation 💡

- In Spring Season Users Engagement is low

Seasonwise Average Users

In [36]:
```
plt.figure(figsize=(13,8))
plt.title('Seasonwise Avg Users | Weather')
sns.barplot(data = df,x='season',y='total_users',hue='weather',errorbar=None,estima
plt.show()
```



In [37]:
```
sampl = df[df['season']=='spring'].groupby(['season','weather'])['total_users'].mea
sampl.groupby('weather')['total_users'].sum() /sampl['total_users'].sum()*100
```

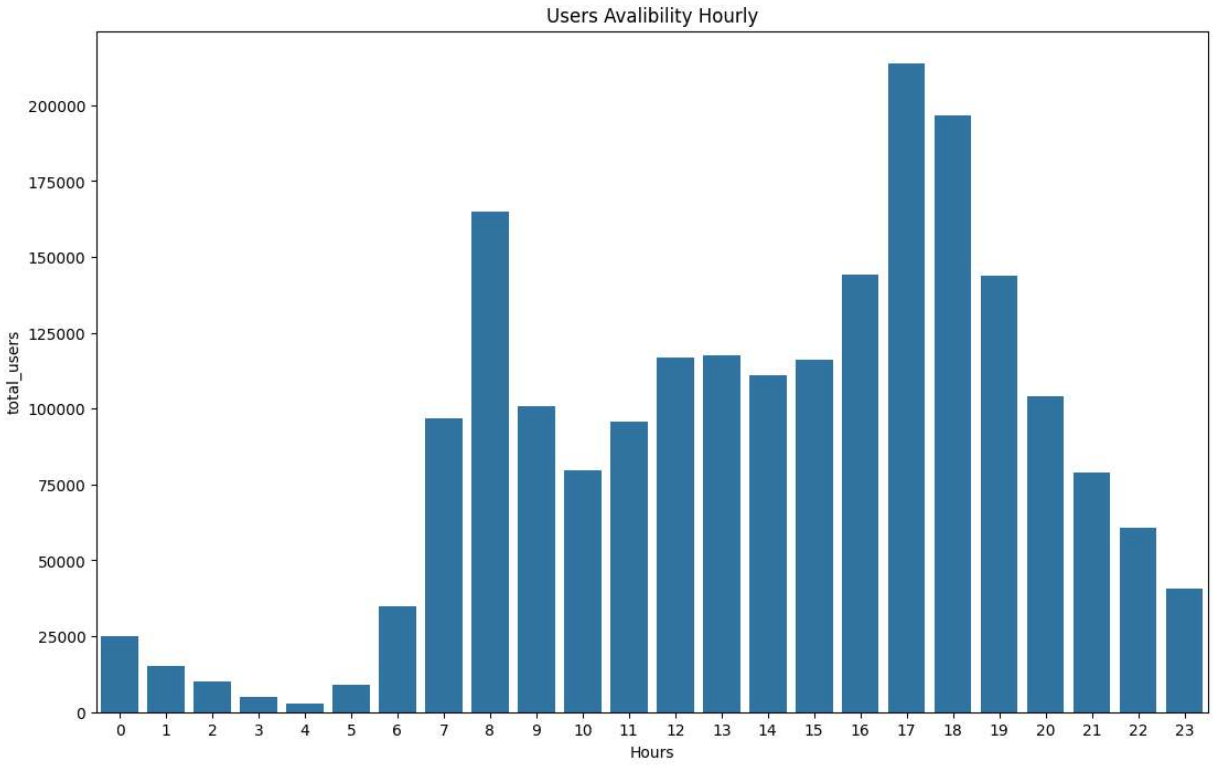
Out[37]:
```
weather
Clear         27.629066
Cloudy        23.287940
Heavy Rain    35.739914
Light Rain    13.343080
Name: total_users, dtype: float64
```

Observation 💡

1. The Reason why users engagement is low in Spring season because
   - Rain
   - Approx 50% users Availability on heavy rains

Hourly Users Avalibility

In [38]:
```
plt.figure(figsize=(13,8))
plt.title('Users Avalibility Hourly')
sns.barplot(df.groupby('Hours')['total_users'].sum())
plt.show()
```



Observation 💡

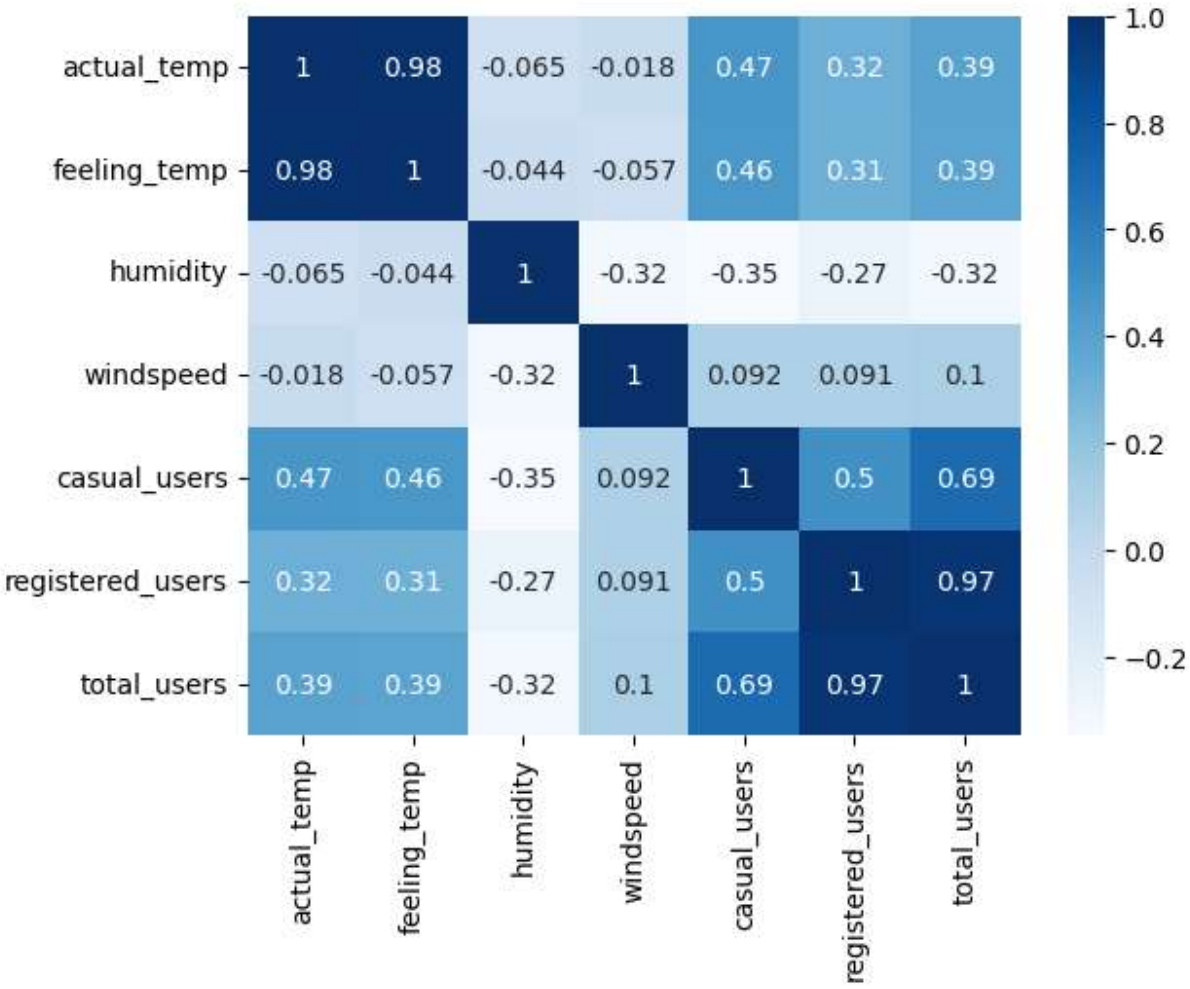
- Peak Hours are between 6AM to Midnight
  - Especially Office Starting timinings from 7AM-9AM had more than 1L+ users avaialable
  - Same goes for Office Return timinings also from 4PM - 7PM
- In Early Morning time No Users Available

Correlation

In [39]:
```
num_df = df[num_cols]
```

In [40]:
```
sns.heatmap(num_df.corr(),cmap='Blues', annot=True)
```

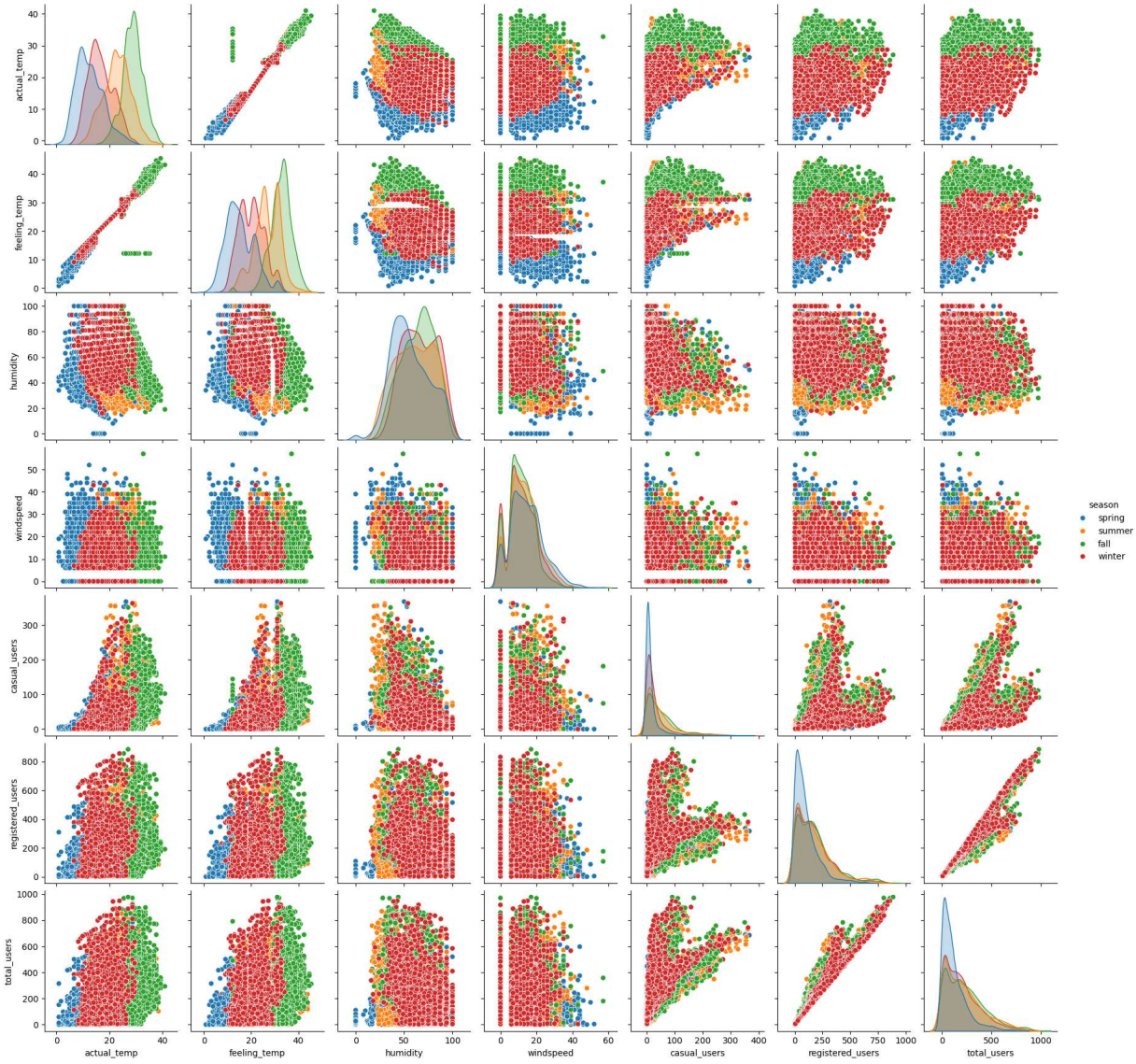
Out[40]: <Axes: >



In [42]:
```
num_cols.append('season')
```

In [45]:
```
sns.pairplot(num_df,hue='season')
```

Out[45]: <seaborn.axisgrid.PairGrid at 0x2731e383310>



Statistical Testing

In [46]:
```
# Libraries
from scipy.stats import norm,binom,geom #Binom,Geometric
from scipy.stats import ttest_1samp,ttest_rel,ttest_ind # T-Test
from scipy.stats import chisquare,chi2_contingency # Chi-Squared Test
from scipy.stats import pearsonr,spearmanr # Correlation Test
from scipy.stats import f_oneway # Anova Test
from scipy.stats import shapiro
from scipy.stats import levene
import statsmodels.api as sm
from scipy.stats import kruskal
import scipy.stats as stats
```

1) is Working Day has effect on number of electric cycles rented

In [47]:
```
# Data
working_day_df = df[df['workingday']=='Y']['total_users']
non_working_day_df = df[df['workingday']!='Y']['total_users']
```

In [48]:
```
# 1 Setup  Hypothesis
h0 = 'Working Day has no effect on number of electric cycles rented'
h1 = 'Working Day has effect on number of electric cycles rented'

alpha = 0.05 #(Default )
# 2 Choose Test
print('''
We are going with t-test independence because we are identifying relationship betwe
Here we are considering 2 Categoricals with no std,population mean so t_test_ind (2
''')

# 3 Assumptions (If Needed )

# 4 Compute Pvalue
t_stat,pvalue = ttest_ind(working_day_df,non_working_day_df)
print(f't stat :{t_stat} \n pvalue {pvalue}')

print('=============================================================================
# 5 Justification
if pvalue<alpha:
    print(f'Rejecting Null Hypothesis \n {h0}')
else:
    print(f'Failed to Reject Null Hypothesis \n {h1}')
print('=============================================================================
print(np.round(df.groupby('workingday')['total_users'].sum() / df['total_users'].su
print(f'69% of users are Available on Working day so Workingday has impact on Renta
```

```
We are going with t-test independence because we are identifying relationship betwee
n 2 Categorical with group of numericals
Here we are considering 2 Categoricals with no std,population mean so t_test_ind (2s
amples)

t stat :1.2096277376026694
 pvalue 0.22644804226361348
================================================================================
Failed to Reject Null Hypothesis
 Working Day has effect on number of electric cycles rented
================================================================================
workingday
N    31.4
Y    68.6
Name: total_users, dtype: float64
69% of users are Available on Working day so Workingday has impact on Rental Bikes
```

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

In [49]:
```
# Data
spring_cycles = df[df['season'] == 'spring']['total_users']
summer_cycles = df[df['season'] == 'summer']['total_users']
```

```
fall_cycles = df[df['season'] == 'summer']['total_users']
winter_cycles = df[df['season'] == 'winter']['total_users']
```

In [50]:

```
print(' #1 Setup Hypothesis')
print('''
h0 = No. of cycles rented same throughout all seasons
h1 = No. of cycles rented different in different seasons
''')
h0 = 'No. of cycles rented same throughout all seasons'
h1 = 'No. of cycles rented different in different seasons'
print('================================================================================


print('# 2 Choose Test')
print('''
We are going with Oneway ANOVA because we are identifying relationship between 4 Ca
Here we are considering more than 2 Categoricals so we are going with Oneway ANOVA
''')
print('================================================================================

print('# 3 Assumptions')
print('''
1. It Should follow Normal Distribution
    QQ Plot ❌
    Shapiro-wilk test (Samples in between 50-200) ❌❌
2. Equal Variance
    Levence Test ❌❌❌
3. Independent

''')
print('================================================================================

print('# QQ plot check - for Normal Distribution check')
plt.subplots(2,2, figsize = (15,10))

plt.subplot(2,2,1)
stats.probplot(spring_cycles, dist="norm", plot=plt)
plt.title('spring_cycles - QQ plot')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Ordered Values')

plt.subplot(2,2,2)
stats.probplot(summer_cycles, dist="norm", plot=plt)
plt.title('summer_cycles - QQ plot')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Ordered Values')

plt.subplot(2,2,3)
stats.probplot(fall_cycles, dist="norm", plot=plt)
plt.title('fall_cycles - QQ plot')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Ordered Values')

plt.subplot(2,2,4)
stats.probplot(winter_cycles, dist="norm", plot=plt)
plt.title('winter_cycles - QQ plot')
```

```
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Ordered Values')

plt.suptitle('QQ plots For All Seasons')
plt.show()
print(f'Conclusion1 ❌ : No Sample Data is following Sample Distribution')
print('==========================================================================
print('# Shapiro check - for Normal Distribution check sample_size is 100')

sample_size = 100

spring_cycles_sample = df[df['season'] == 'spring']['total_users'].sample(sample_si
summer_cycles_sample = df[df['season'] == 'summer']['total_users'].sample(sample_si
fall_cycles_sample = df[df['season'] == 'fall']['total_users'].sample(sample_size)
winter_cycles_sample = df[df['season'] == 'winter']['total_users'].sample(sample_si

shapiro_results_spring = stats.shapiro(spring_cycles_sample)
shapiro_results_summer = stats.shapiro(summer_cycles_sample)
shapiro_results_fall = stats.shapiro(fall_cycles_sample)
shapiro_results_winter = stats.shapiro(winter_cycles_sample)


plt.subplots(2, 2, figsize=(15, 10))

plt.subplot(2, 2, 1)
stats.probplot(spring_cycles_sample, dist="norm", plot=plt)
plt.title('spring_cycles - Shapiro plot')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Ordered Values')


plt.subplot(2, 2, 2)
stats.probplot(summer_cycles_sample, dist="norm", plot=plt)
plt.title('summer_cycles - Shapiro plot')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Ordered Values')


plt.subplot(2, 2, 3)
stats.probplot(fall_cycles_sample, dist="norm", plot=plt)
plt.title('fall_cycles- Shapiro plot')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Ordered Values')


plt.subplot(2, 2, 4)
stats.probplot(winter_cycles_sample, dist="norm", plot=plt)
plt.title('winter_cycles- Shapiro plot')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Ordered Values')


plt.suptitle('Shapiro plots For All Seasons')
plt.show()

print(f'Conclusion2 ❌❌ : No Sample Data is following Sample Distribution')
```

```
print('================================================================================')
print('# Levence Test - For Equal Variance By using Box Plot')

plt.figure(figsize=(15, 6))
plt.boxplot([spring_cycles_sample, summer_cycles_sample, fall_cycles_sample, winter
            labels=['Spring', 'Summer', 'Fall', 'Winter'])
plt.title('Total Users Distribution by Season')
plt.xlabel('Season')
plt.ylabel('Total Users')
plt.show()
print(f'Conclusion3 ❌❌❌  - No Equal Variance Among Seasons')



print('================================================================================')
print('# 4 Compute Pvalue')
print("After Checking All Assumptions We are decided to move with 'Kruskal-Wallis t

kruskal_results = kruskal(spring_cycles_sample, summer_cycles_sample, fall_cycles_s

print("Kruskal-Wallis Test Results:")
print("Kruskal-Wallis Statistic:", kruskal_results.statistic)
print("P-value:", kruskal_results.pvalue)
print('================================================================================')
alpha = 0.05 #(Default )
pvalue = kruskal_results.pvalue
print('# 5 Justification')
if pvalue<alpha:
    print(f'Rejecting Null Hypothesis \n {h0}')
else:
    print(f'Failed to Reject Null Hypothesis \n {h1}')
print('================================================================================')
df.groupby('season')['total_users'].sum().sort_values(ascending=False) / df['total_
print(f'In Each Season we have different Rental Cycles are Available')
```

#1 Setup Hypothesis

h0 = No. of cycles rented same throughout all seasons
h1 = No. of cycles rented different in different seasons

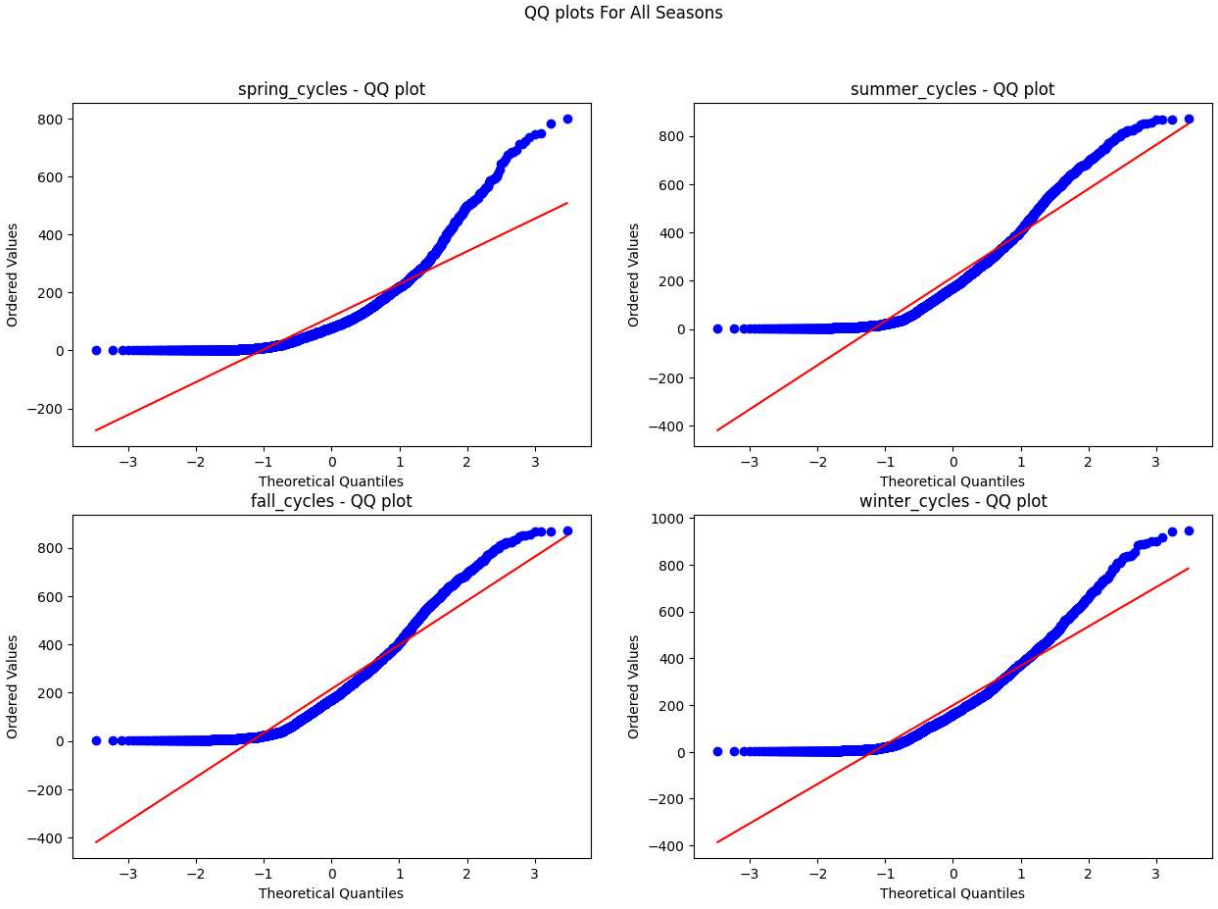
==========
2 Choose Test

We are going with Oneway ANOVA because we are identifying relationship between 4 Categorical with group of numericals
Here we are considering more than 2 Categoricals so we are going with Oneway ANOVA

==========
3 Assumptions

1. It Should follow Normal Distribution
    QQ Plot ❌
    Shapiro-wilk test (Samples in between 50-200) ❌❌
2. Equal Variance
    Levence Test ❌❌❌
3. Independent

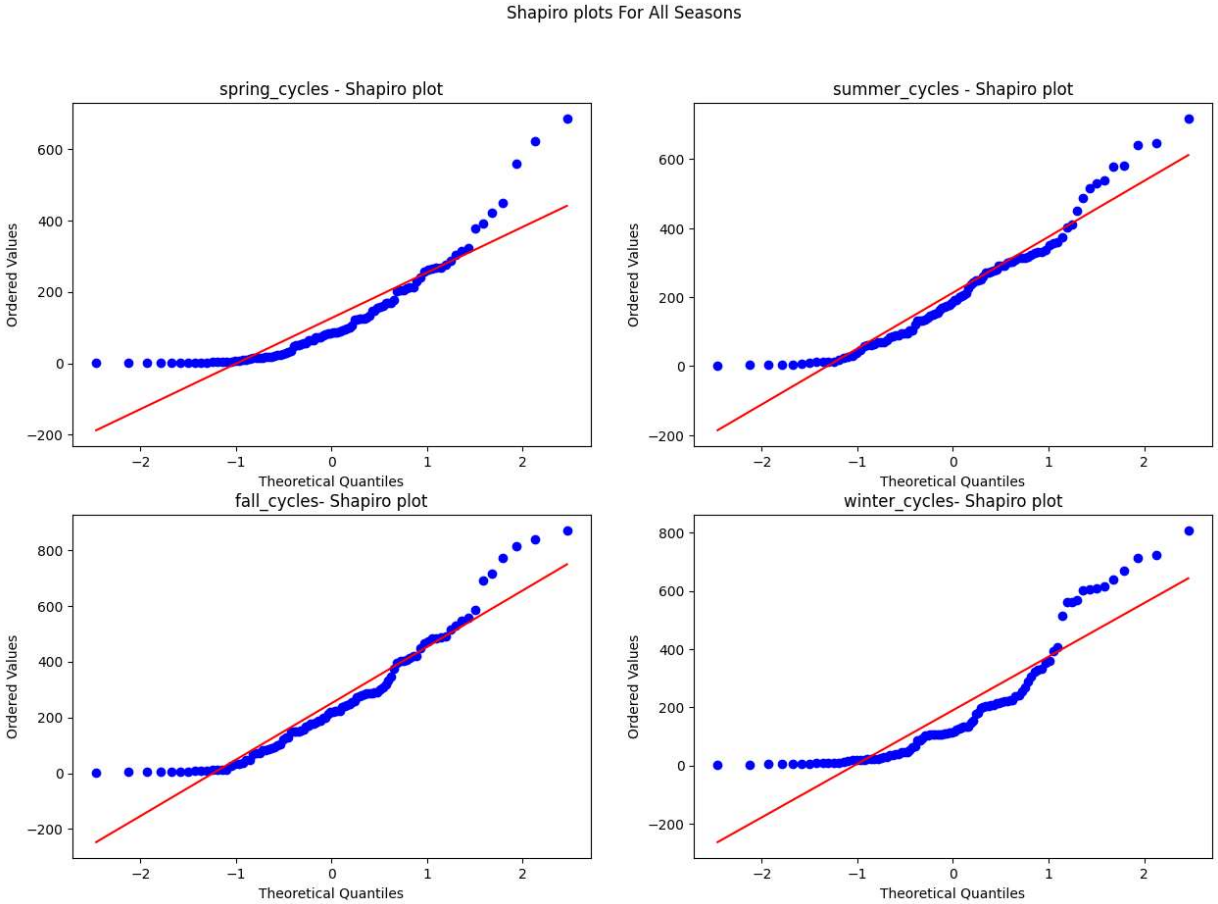
==========
QQ plot check - for Normal Distribution check



Conclusion1 ❌ : No Sample Data is following Sample Distribution

==========

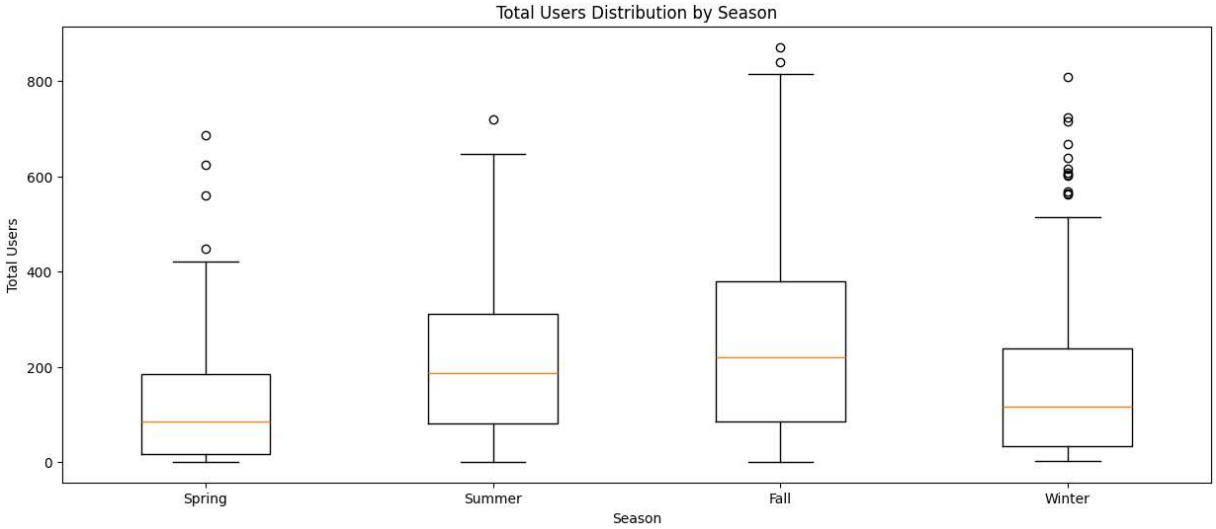
Shapiro check - for Normal Distribution check sample_size is 100



Conclusion2 ❌❌ : No Sample Data is following Sample Distribution

==========

Levence Test - For Equal Variance By using Box Plot



```
Conclusion3 ❌❌❌  - No Equal Variance Among Seasons
================================================================================
============
# 4 Compute Pvalue
After Checking All Assumptions We are decided to move with 'Kruskal-Wallis test'
Kruskal-Wallis Test Results:
Kruskal-Wallis Statistic: 28.447644408263326
P-value: 2.9252848964201597e-06
================================================================================
============
# 5 Justification
Rejecting Null Hypothesis
 No. of cycles rented same throughout all seasons
================================================================================
============
In Each Season we have different Rental Cycles are Available
```

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

In [94]:
```
# Data
Clear_weather_cycles = df[df['weather'] == 'Clear']['total_users']
Cloudy_weather_cycles = df[df['weather'] == 'Cloudy']['total_users']
Light_weather_Rain_cycles = df[df['weather'] == 'Light Rain']['total_users']
Heavy_Rain_weather_cycles = df[df['weather'] == 'Heavy Rain']['total_users']
```

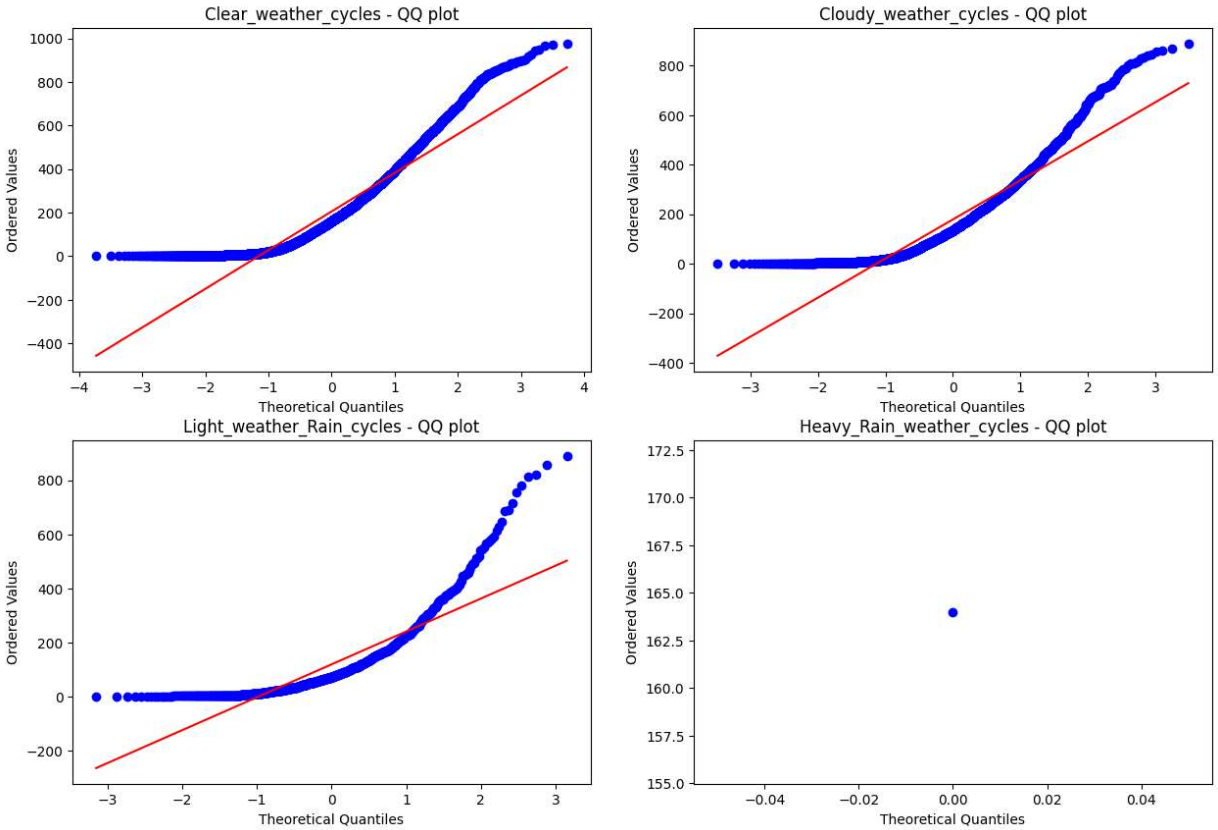
In [139…
```
print(' #1 Setup Hypothesis')
print('''
h1 = No. of cycles rented same at any weather
h0 = No. of cycles rented different in weather conditions
''')
h1 = 'No. of cycles rented same at any weather'
h0 = 'No. of cycles rented different in weather conditions'
print('================================================================================


print('# 2 Choose Test')
print('''
We are going with Oneway ANOVA because we are identifying relationship between 4 Ca
Here we are considering more than 2 Categoricals so we are going with Oneway ANOVA
''')
print('================================================================================

print('# 3 Assumptions')
print('''
1. It Should follow Normal Distribution
    QQ Plot ❌
    Shapiro-wilk test (Samples in between 50-200) ❌❌
2. Equal Variance
    Levence Test ❌❌❌
3. Independent

''')
print('================================================================================
```

```
print('# QQ plot check - for Normal Distribution check')
plt.subplots(2,2, figsize = (15,10))

plt.subplot(2,2,1)
stats.probplot(Clear_weather_cycles, dist="norm", plot=plt)
plt.title('Clear_weather_cycles - QQ plot')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Ordered Values')

plt.subplot(2,2,2)
stats.probplot(Cloudy_weather_cycles, dist="norm", plot=plt)
plt.title('Cloudy_weather_cycles - QQ plot')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Ordered Values')

plt.subplot(2,2,3)
stats.probplot(Light_weather_Rain_cycles, dist="norm", plot=plt)
plt.title('Light_weather_Rain_cycles - QQ plot')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Ordered Values')

plt.subplot(2,2,4)
stats.probplot(Heavy_Rain_weather_cycles, dist="norm", plot=plt)
plt.title('Heavy_Rain_weather_cycles - QQ plot')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Ordered Values')

plt.suptitle('QQ plots For All Weather Conditions')
plt.show()
print(f'Conclusion1 ❌ : No Sample Data is following Sample Distribution')
print('================================================================================
print('# Shapiro check - for Normal Distribution check sample_size is 100')


plt.subplots(2, 2, figsize=(15, 10))

plt.subplot(2, 2, 1)
Clear_weather_cycles = df[df['weather'] == 'Clear']['total_users']
stats.shapiro(Clear_weather_cycles[:55])
stats.probplot(Clear_weather_cycles[:55], dist="norm", plot=plt)
plt.title('Clear_weather_cycles - Shapiro plot')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Ordered Values')


plt.subplot(2, 2, 2)
Cloudy__weather_cycles = df[df['weather'] == 'Cloudy_']['total_users']
stats.shapiro(Cloudy_weather_cycles[:55])
stats.probplot(Cloudy_weather_cycles[:55], dist="norm", plot=plt)
plt.title('Cloudy_weather_cycles - Shapiro plot')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Ordered Values')


plt.subplot(2, 2, 3)
```

```
Light_weather_cycles = df[df['weather'] == 'Light Rain']['total_users']
stats.shapiro(Light_weather_cycles[:55])
stats.probplot(Light_weather_cycles[:55], dist="norm", plot=plt)
plt.title('Light_weather_cycles - Shapiro plot')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Ordered Values')

Heavy_weather_cycles = df[df['weather'] == 'Heavy Rain']['total_users']
plt.subplot(2, 2, 4)
#stats.shapiro(Heavy_weather_cycles[:1])
#stats.probplot(Heavy_weather_cycles[:1], dist="norm", plot=plt)  Only 1 Row is pre
plt.title('Heavy_weather_cycles - Shapiro plot')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Ordered Values')


plt.suptitle('Shapiro plots For All Weather Conditions')
plt.show()

print(f'Conclusion2 ❌❌ : No Sample Data is following Sample Distribution')
print('=============================================================================
print('# Levence Test - For Equal Variance By using Box Plot')

plt.figure(figsize=(15, 6))
plt.boxplot([Clear_weather_cycles,Cloudy_weather_cycles,Light_weather_Rain_cycles,H
            labels=['Clear', 'Cloudy', 'Light_Rain', 'Heavy_Rain'])
plt.title('Total Users Distribution by Weather')
plt.xlabel('Weather')
plt.ylabel('Total Users')
plt.suptitle('Variance For All Weather Conditions')
plt.show()
print(f'Conclusion3 ❌❌❌  - No Equal Variance Weather Conditions')



print('=============================================================================
print('# 4 Compute Pvalue')
print("After Checking All Assumptions We are decided to move with 'Kruskal-Wallis t

kruskal_results = kruskal(Clear_weather_cycles,Cloudy_weather_cycles,Light_weather_

print("Kruskal-Wallis Test Results:")
print("Kruskal-Wallis Statistic:", kruskal_results.statistic)
print("P-value:", kruskal_results.pvalue)
print('=============================================================================
alpha = 0.05 #(Default )
pvalue = kruskal_results.pvalue
print('# 5 Justification')
if pvalue<alpha:
    print(f'Rejecting Null Hypothesis \n {h0}')
else:
    print(f'Failed to Reject Null Hypothesis \n {h1}')
```

```
 #1 Setup Hypothesis

h1 = No. of cycles rented same at any weather
h0 = No. of cycles rented different in weather conditions

================================================================================
============
# 2 Choose Test

We are going with Oneway ANOVA because we are identifying relationship between 4 Cat
egorical with group of numericals
Here we are considering more than 2 Categoricals so we are going with Oneway ANOVA

================================================================================
============
# 3 Assumptions

1. It Should follow Normal Distribution
    QQ Plot ❌
    Shapiro-wilk test (Samples in between 50-200) ❌❌
2. Equal Variance
    Levence Test ❌❌❌
3. Independent


================================================================================
============
# QQ plot check - for Normal Distribution check
c:\users\lokesh\appdata\local\programs\python\python39\lib\site-packages\scipy\stats
\_stats_mstats_common.py:182: RuntimeWarning: invalid value encountered in scalar di
vide
  slope = ssxym / ssxm
c:\users\lokesh\appdata\local\programs\python\python39\lib\site-packages\scipy\stats
\_stats_mstats_common.py:196: RuntimeWarning: invalid value encountered in sqrt
  t = r * np.sqrt(df / ((1.0 - r + TINY)*(1.0 + r + TINY)))
c:\users\lokesh\appdata\local\programs\python\python39\lib\site-packages\scipy\stats
\_stats_mstats_common.py:199: RuntimeWarning: invalid value encountered in scalar di
vide
  slope_stderr = np.sqrt((1 - r**2) * ssym / ssxm / df)
```

QQ plots For All Weather Conditions

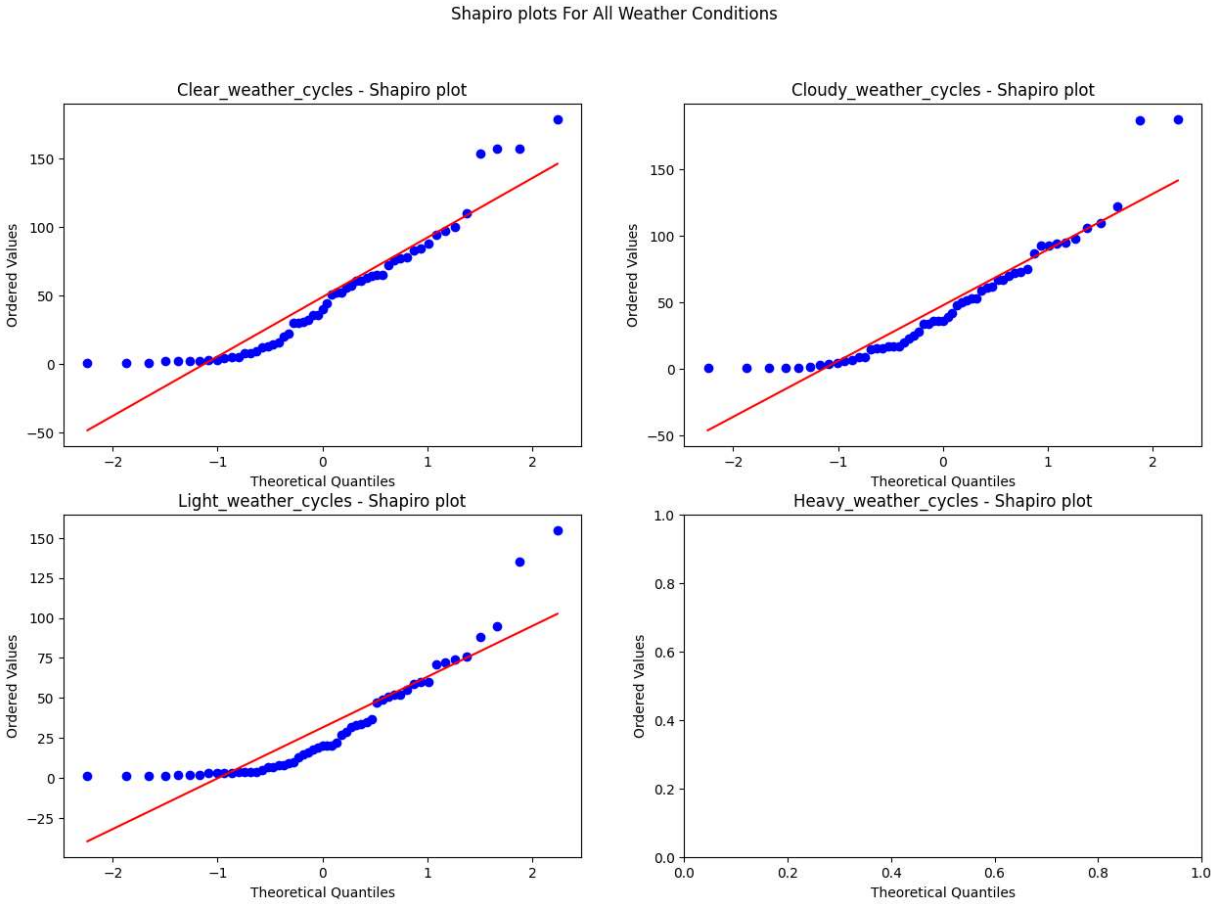


Conclusion1 ❌ : No Sample Data is following Sample Distribution

==========

==========

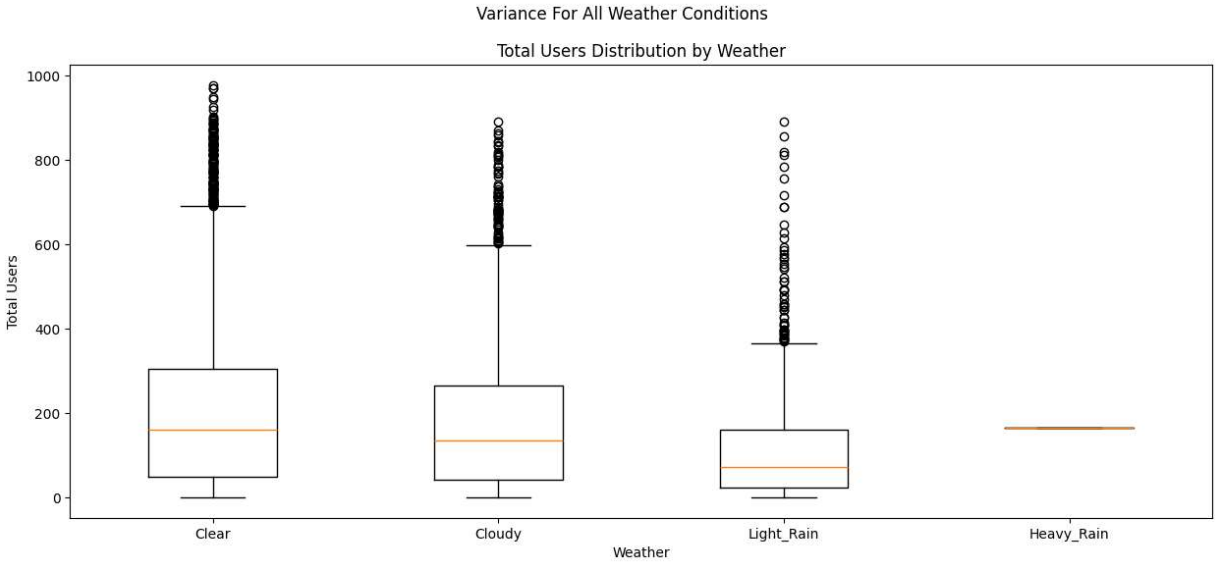
Shapiro check - for Normal Distribution check sample_size is 100



Conclusion2 ❌❌ : No Sample Data is following Sample Distribution

==========

Levence Test - For Equal Variance By using Box Plot



```
Conclusion3 ❌❌❌  - No Equal Variance Weather Conditions
================================================================================
============
# 4 Compute Pvalue
After Checking All Assumptions We are decided to move with 'Kruskal-Wallis test'
Kruskal-Wallis Test Results:
Kruskal-Wallis Statistic: 205.00216514479087
P-value: 3.501611300708679e-44
================================================================================
============
# 5 Justification
Rejecting Null Hypothesis
 No. of cycles rented different in weather conditions
```

4 Check if the Weather conditions are significantly different during different Seasons

In [126…

```
data = pd.crosstab(index=df['weather'],columns=df['season'],margins=True)
data
```

Out[126…

season weather	fall	spring	summer	winter	All
Clear	1930	1759	1801	1702	7192
Cloudy	604	715	708	807	2834
Heavy Rain	0	1	0	0	1
Light Rain	199	211	224	225	859
All	2733	2686	2733	2734	10886

In [138…

```
print('# 1 Setup  Hypothesis')
print('''
h0 = 'Weather conditions are significantly different during different Seasons'
h1 = 'Weather conditions are significantly same in all seasons'
''')
h0 = 'Weather conditions are significantly different during different Seasons'
h1 = 'Weather conditions are significantly same in all seasons'

print('# 2 Choose Test')
print('''
We are going with chisqaured independence because we are identifying relationship b
Here we are considering 8 Categoricals categorical Vs Categorical
''')

# 3 Assumptions (If Needed )

# 4 Compute Pvalue
observed_data = [[1930,1759,1801,1702],[604,715,708,807],[0,1,0,0],[199,211,224,225
chi_stat,pvalue,dof,expected = chi2_contingency(observed)
print(f'chi_stat :{chi_stat} \n pvalue {pvalue} \n degree_of_freedom :{dof}')
```

```python
expected = expected.astype(np.int64)
print()
print(f'Expected \n {expected}')

print('================================================================================')
# 5 Justification
alpha = 0.05 #(Default )
if pvalue<alpha:
    print(f'Rejecting Null Hypothesis \n {h0}')
else:
    print(f'Failed to Reject Null Hypothesis \n {h1}')
print('================================================================================')
```

```
# 1 Setup  Hypothesis

h0 = 'Weather conditions are significantly different during different Seasons'
h1 = 'Weather conditions are significantly same in all seasons'

# 2 Choose Test

We are going with chisqaured independence because we are identifying relationship be
tween 8 Categorical with group of numericals
Here we are considering 8 Categoricals categorical Vs Categorical

chi_stat :49.15865559689363
 pvalue 1.5499250736864862e-07
 degree_of_freedom :9

Expected
 [[1805 1774 1805 1806]
 [ 711  699  711  711]
 [   0    0    0    0]
 [ 215  211  215  215]]
================================================================================
Rejecting Null Hypothesis
 Weather conditions are significantly different during different Seasons
================================================================================
```