Name: Aloysius Lobo

Batch: Morning – DSML BeginnerFeb23

Case-Study: Yulu – Hypothesis Testing

 # Importing all the necessary libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import ttest_ind, chi2_contingency, f_oneway
```

#Importing 'Yulu' csv file

```
df = pd.read_csv('yulu.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

 # Performing basic EDA

1. **Getting shape of dataset**

```
df.shape
```

```
(10886, 12)
```

Observations –

There are total **10886** rows and **12** columns in the entire dataset.

2. **Getting column info on datatype and non-null value counts**

```
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
```

 # Finding missing/null values

```
df.isna().sum()
```

```
datetime      0
season        0
holiday       0
workingday    0
weather       0
temp          0
atemp         0
humidity      0
windspeed     0
casual        0
registered    0
count         0
dtype: int64
```

Observations –

There are no missing values found in the dataset.

 # Columns containing object/categorical datatype

```
cat_cols = df.dtypes == 'object'
cat_cols = list(cat_cols[cat_cols].index)
cat_cols
```

```
['datetime']
```

Observations –

Only ‘datetime’ column contains object datatype

 # Converting all categorical into ‘object’ datatype

```
cat = ['season', 'holiday', 'workingday', 'weather']

for i in cat:
    df[i] = df[i].astype('object')
```

```
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  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  
dtypes: float64(3), int64(4), object(5)
memory usage: 1020.7+ KB
```

 # Getting statistical info on columns with numerical datatypes

```
df.describe()
```

	temp	atemp	humidity	windspeed	casual	registered	count
count	10886.00000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000
mean	20.23086	23.655084	61.886460	12.799395	36.021955	155.552177	191.574132
std	7.79159	8.474601	19.245033	8.164537	49.960477	151.039033	181.144454
min	0.82000	0.760000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	13.94000	16.665000	47.000000	7.001500	4.000000	36.000000	42.000000
50%	20.50000	24.240000	62.000000	12.998000	17.000000	118.000000	145.000000
75%	26.24000	31.060000	77.000000	16.997900	49.000000	222.000000	284.000000
max	41.00000	45.455000	100.000000	56.996900	367.000000	886.000000	977.000000

Observations –

- Count of users renting the bicycle was found to be the maximum when the temperature in the city was 41 degrees Celcius while the count decreased when the temperature was 0.82 degrees Celcius.
- The number of registered customers renting the bicycle is 886 and is more than the non-registered customers which is 367.
- The average humidity is 62 when 50% customers rent the bike.
- The average temperature is 20.5 degrees when 50% customers rent the bike.

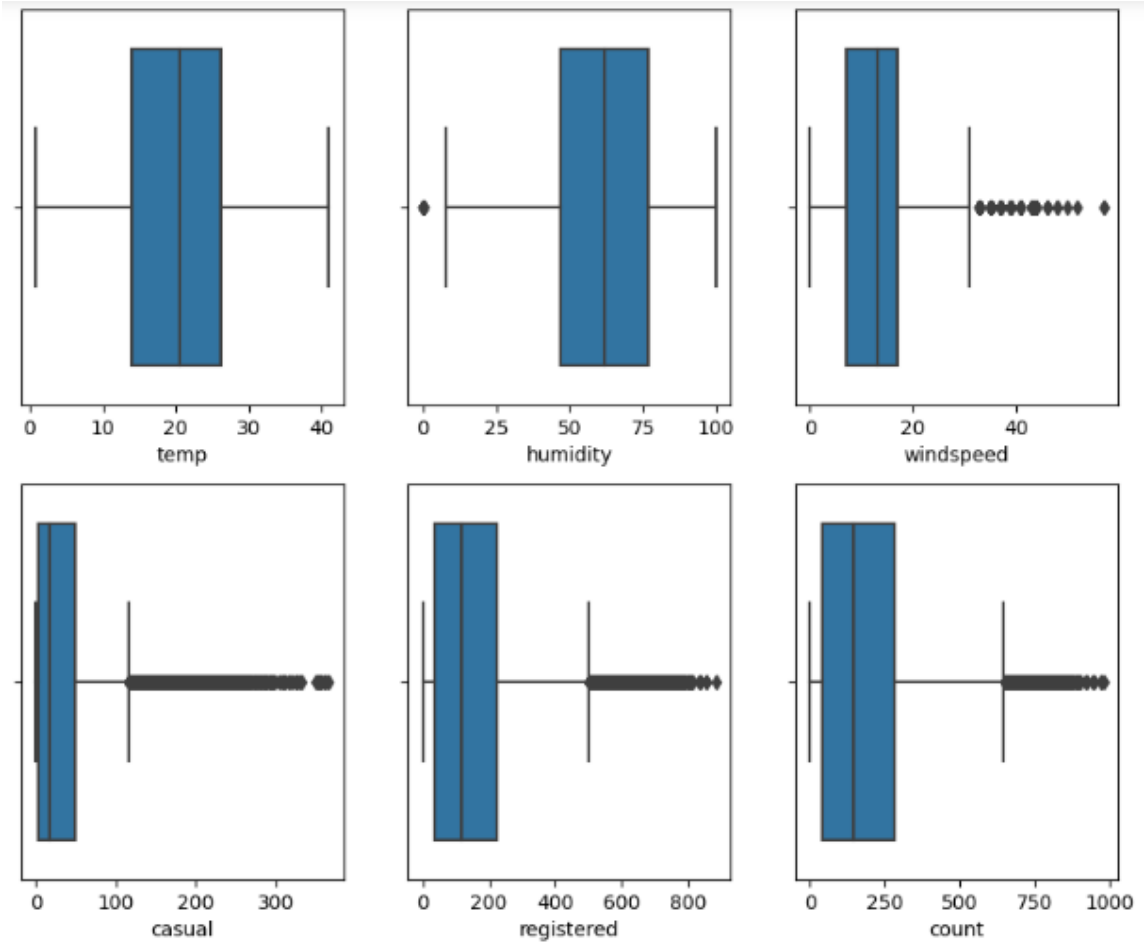
 # Replacing row-names with specific names in categorical columns

```
df = df.replace({'season' : {1:'spring',2:'summer',3:'fall',4:'winter'}})
df = df.replace({'holiday':{ 0:'Not a holiday', 1: 'Holiday'}})
df = df.replace({'workingday':{1:'Working day', 0:'weekend/holiday'}})
df = df.replace({'weather': {1:'Clear/Partly cloudy', 2: 'Mist/Cloudy', 3: 'LightRains/ LightSnow', 4: 'HeavyRains/Thunderstorm/S
```

Seasons	Weather	Holiday	Working day
1 – Spring	1 – Clear/Partly cloudy	0 – Not a holiday	0 – Weekend/Holiday
2 – Summer	2 – Mist/Cloudy	1 – Holiday	1 – Working day
3 – Fall	3 – LightRains/ LightSnow	-	-
4 - Winter	4 – HeavyRains/ Thunderstorms/Snow	-	-

 # Outlier detection

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

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

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

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

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

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

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


plt.show()
```



Observations –

- Maximum outliers are observed in casual, registered and count, while a few are observed in the windspeed data.
- Few customers do rent the bicycle even when the windspeed is over 30.

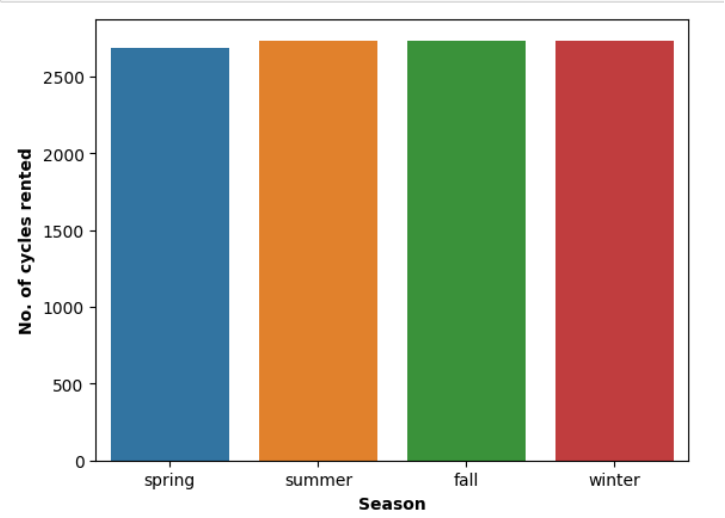
 # Univariate Analysis

Categorical variables

1. **Seasons**

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

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



Observations –

Above analysis on seasons show that except during spring, the cycles rented were similar in all other seasons.

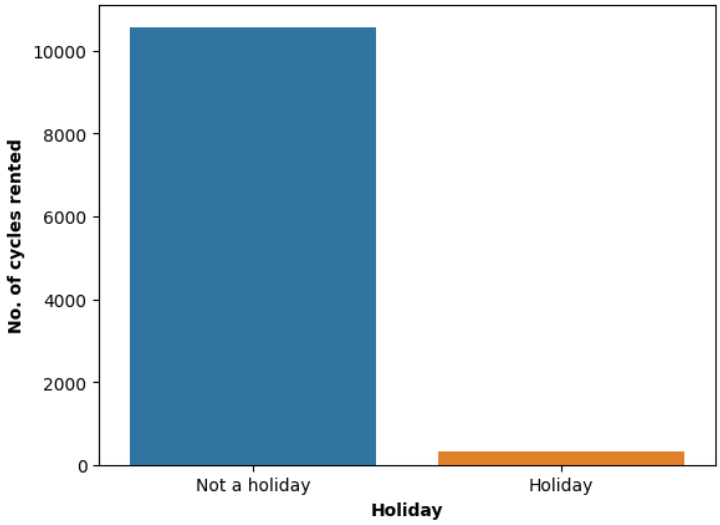
Recommendations -

Seasons do affect the renting of Yulu bicycles.

2. Holiday

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

```
holiday
Not a holiday    10575
Holiday            311
Name: count, dtype: int64
```



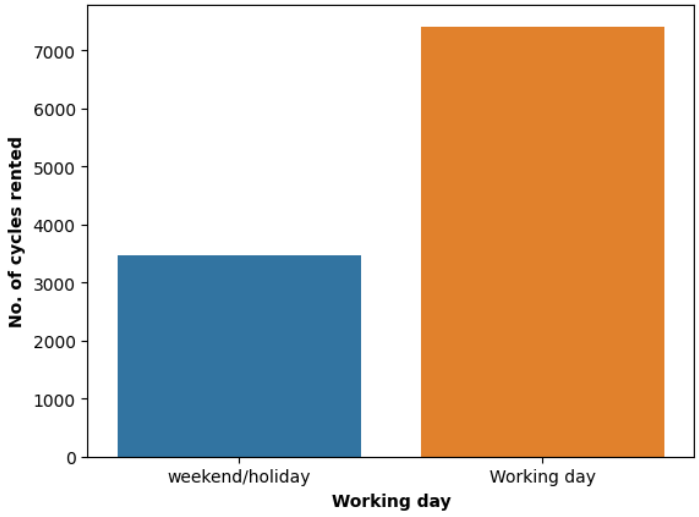
Observations –

Above analysis on holidays show that maximum cycles were rented when it's not a holiday.

3. Working days

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

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



Observations –

Above analysis on working days show that maximum cycles were rented during working days and not on weekends/holidays.

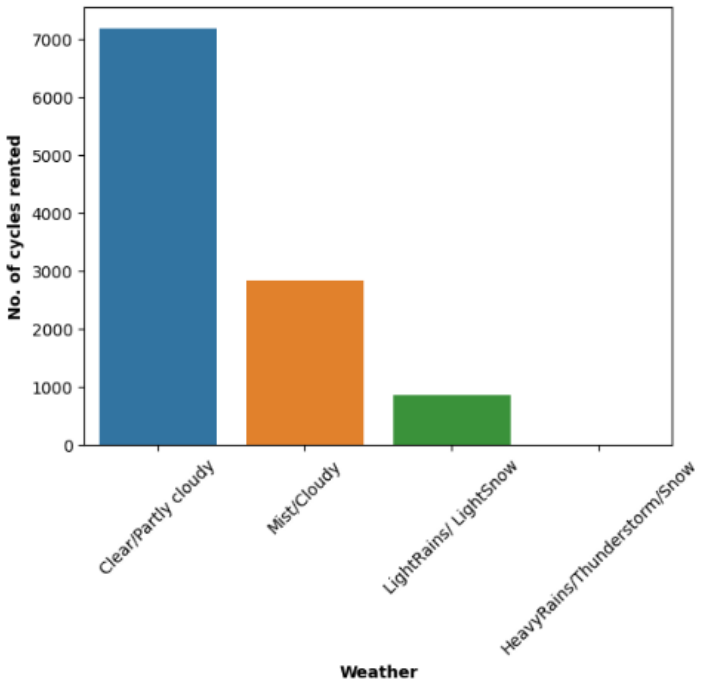
Recommendations -

From above analysis on Holidays and Working days, it is clear that maximum count of bicylces were rented during the Working days as compared to on Weekends/ Holidays. Hence, Yulu must make provisions to increase the Yulu bikes over working days.

4. Weather

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

```
weather
Clear/Partly cloudy             7192
Mist/Cloudy                     2834
LightRains/ LightSnow            859
HeavyRains/Thunderstorm/Snow       1
Name: count, dtype: int64
```



Observations –

- Above analysis show that maximum number of bicyles are rented only when the weather is clear or its partly cloudy.
- During very bad weather like thunderstorms or very heavy snowing, no one has rented Yulu bicyle.

Recommendations -

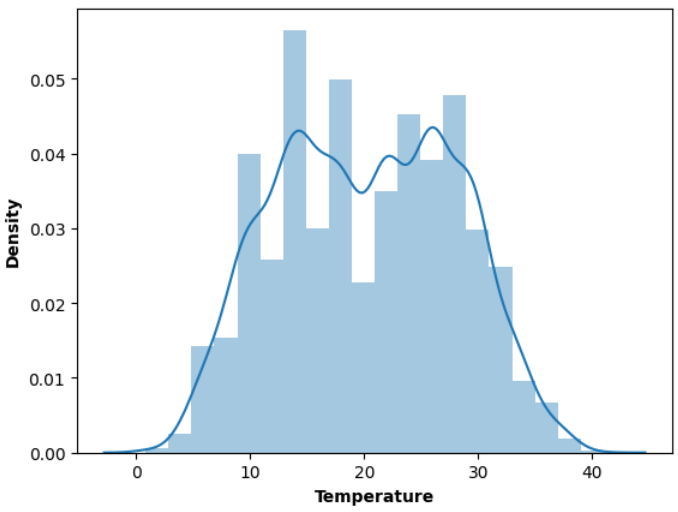
As the analysis show a drop in the bike rents during bad weathers like rains/ thunderstorms. It is thus suggested that Yulu can either drop the rent fares during such bad weather conditions or can introduce a bicycle with some protective covering for snow and rains so that users are comfortable to rent the bikes even in bad weather conditions.

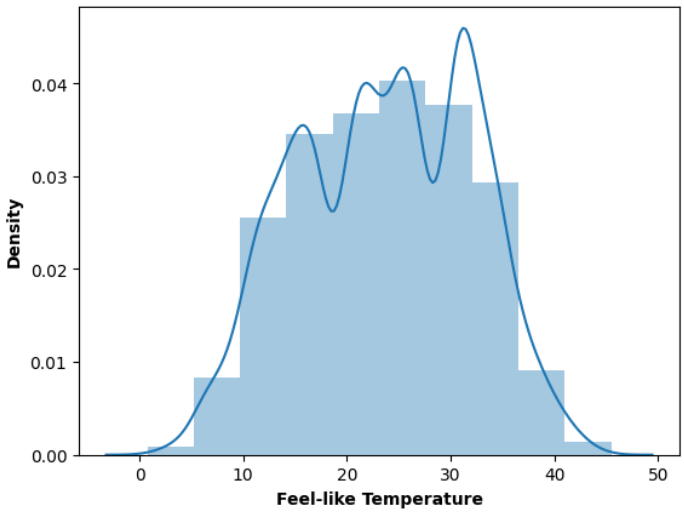
Numerical variables

1. **Temp and Feel-like temperature**

```
sns.distplot(df['temp'], bins = 20)
plt.xlabel('Temperature', fontweight = 'bold')
plt.ylabel('Density', fontweight = 'bold')
plt.show()
```

```
sns.distplot(df['atemp'], bins = 10)
plt.xlabel('Feel-like Temperature', fontweight = 'bold')
plt.ylabel('Density', fontweight = 'bold')
plt.show()
```





Observations –

- Mostly customers have rented bicycle in the temperature range of 15°C to 30°C.
- This is similar to the Feels-like temperature in the range of 15°C to 32°C.
- Some outliers are seen where customers have rented the bike in extreme temperature ranges.

Recommendations –

Yulu should increase the number of bikes for rent when the temperature is above 20 degrees to 30 degrees celcius, as the demand for bikes is more.

 # Hypothesis Testing

1. Working_day

```
# H0: Working day and bicyles rented are independent
# Ha: Number of bicyles rented are dependent on the working day
# significance (alpha) == 0.05

working = df[df['workingday'] == 'Working day']['count']
not_working = df[df['workingday'] == 'weekend/holiday']['count']
t_stat, p_value = ttest_ind(working, not_working, alternative = 'greater')
```

```
p_value
```

```
0.11322402113180674
```

Observations –

- As p_value is greater than the significance value (alpha) i.e. (0.113 > 0.05), we conclude that there are no enough evidences to reject the null hypothesis. Hence, Count of bicycles being rented is independent if it is a working day or a holiday.
- Univariate analysis showed that bicycles are rented more on working days than on holidays/weekends but from the hypothesis testing its clear to confirm that with a 95% confidence they are independent of each other.

Recommendations –

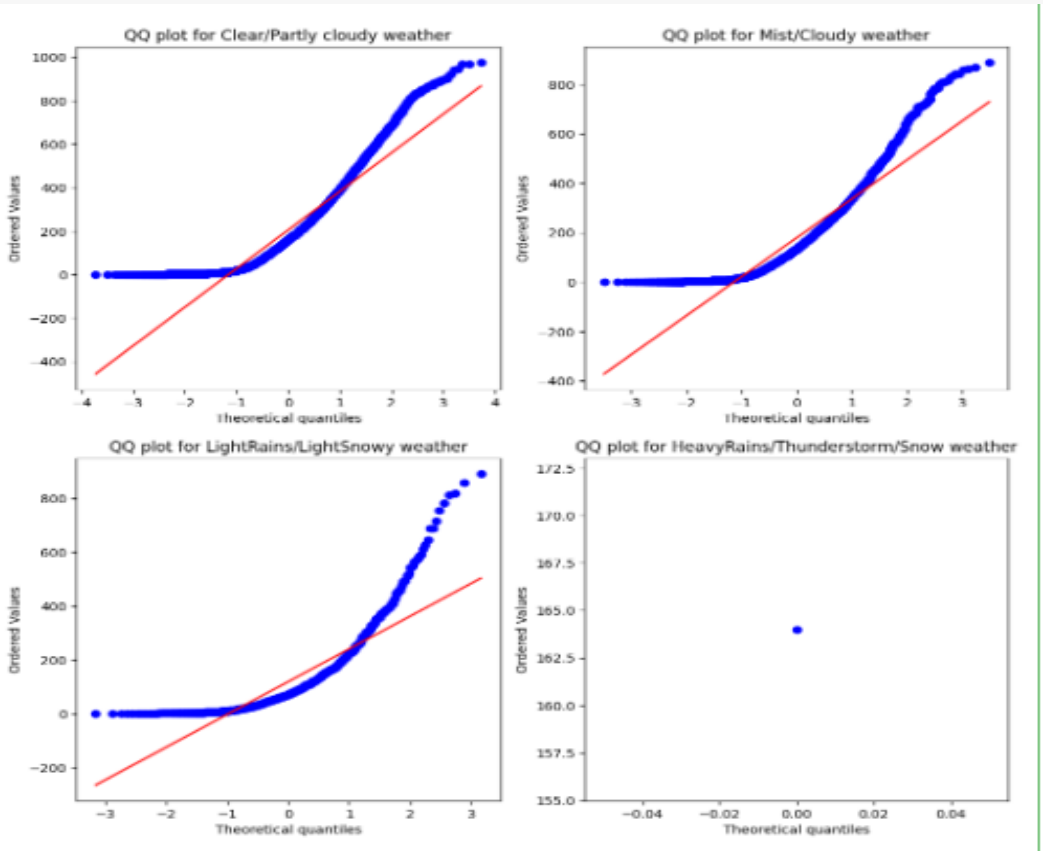
As the count of bikes rented is independent if it's a working day or a holiday, Yulu can make provisions to increase the bikes for rent any day of the week.

2. **Weather**

 # Testing for assumptions for using ANOVA

1. **Testing for Normality using QQ Plot**

```
import scipy.stats as spy
import matplotlib.pyplot as plt

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

plt.subplot(2, 2, 1)
spy.probplot(df.loc[df['weather'] == 'Clear/Partly cloudy', 'count'], plot = plt, dist = 'norm')
plt.title('QQ plot for Clear/Partly cloudy weather')

plt.subplot(2, 2, 2)
spy.probplot(df.loc[df['weather'] == 'Mist/Cloudy', 'count'], plot = plt, dist = 'norm')
plt.title('QQ plot for Mist/Cloudy weather')

plt.subplot(2, 2, 3)
spy.probplot(df.loc[df['weather'] == 'LightRains/ LightSnow', 'count'], plot = plt, dist = 'norm')
plt.title('QQ plot for LightRains/LightSnowy weather')

plt.subplot(2, 2, 4)
spy.probplot(df.loc[df['weather'] == 'HeavyRains/Thunderstorm/Snow', 'count'], plot = plt, dist = 'norm')
plt.title('QQ plot for HeavyRains/Thunderstorm/Snow weather')
plt.plot()
```



Observations –

It's clear from the above analysis that **normality does not exist** for the count of bikes rented during various weather conditions.

2. Testing for Normality using Shapiro-Wilk Test

H0: It's a Gaussian/Normal distribution

Ha: It is not a Gaussian/Normal Distribution

significance(alpha): 0.05

```
from scipy.stats import shapiro

test_stat, p_value = shapiro(df.loc[df['weather'] == 'Clear/Partly cloudy', 'count'].sample(200))
print('p-value for Clear/Partly cloudy: ', p_value,'\n')

test_stat, p_value = shapiro(df.loc[df['weather'] == 'Mist/Cloudy', 'count'].sample(200))
print('p-value for Mist/Cloudy: ',p_value,'\n')

test_stat, p_value = shapiro(df.loc[df['weather'] == 'LightRains/ LightSnow', 'count'].sample(200))
print('p-value for LightRains/LightSnow: ', p_value)
```

```
p-value for Clear/Partly cloudy:  1.5683258511312026e-11

p-value for Mist/Cloudy:  1.3458951730882518e-09

p-value for LightRains/LightSnow:  6.9574667953416734e-15
```

Observations –

Since, p_value < alpha(0.05), we can conclude that with enough evidence we reject the Null Hypothesis (H0) i.e. The above distribution is **not a normal distribution**.

Transforming the data using boxcox transformation and checking if the transformed data follows normal distribution

```
# Transforming the data using boxcox transformation and checking if the transformed data follows normal distribution

import scipy.stats as spy

transformed_weather1 = spy.boxcox(df.loc[df['weather'] == 'Clear/Partly cloudy', 'count'].sample(200))[0]
test_stat, p_value = shapiro(transformed_weather1)
print('p-value after transformation for Clear/Partly cloudy: ', p_value,'\n')

transformed_weather2 = spy.boxcox(df.loc[df['weather'] == 'Mist/Cloudy', 'count'].sample(200))[0]
test_stat, p_value = shapiro(transformed_weather2)
print('p-value after transformation for Mist/Cloudy: ', p_value,'\n')

transformed_weather3 = spy.boxcox(df.loc[df['weather'] == 'LightRains/ LightSnow', 'count'].sample(200))[0]
test_stat, p_value = shapiro(transformed_weather3)
print('p-value after transformation for LightRains/ LightSnow: ', p_value)
```

```
p-value after transformation for Clear/Partly cloudy:  0.0508435033261776

p-value after transformation for Mist/Cloudy:  0.013127652928233147

p-value after transformation for LightRains/ LightSnow:  0.01277914922684431
```

Observations –

- As p_value for weather1 (0.0508) > alpha(0.05), we have enough evidence to fail to reject the Null Hypothesis. Hence, weather-1 i.e. Clear/Partly cloudy weather follows Normal Distribution.
- As p_values for the remaining weather conditions are less than alpha (0.05), we can conclude that with 95% confidence level we reject the null hypothesis. Hence, **it does not follow Normal Distribution.**

3. **Testing for Equal Variance using Levene Test**

```
# H0 : all weather are having equal variances
# Ha: they are not having equal variances

from scipy.stats import levene
statistic, p_value = levene(
 df[df['weather'] == 'Clear/Partly cloudy']['count'],
 df[df['weather'] == 'Mist/Cloudy']['count'],
 df[df['weather'] == 'LightRains/ LightSnow']['count'],
 df[df['weather'] == 'HeavyRains/Thunderstorm/Snow']['count']
 )
```

```
p_value
```

```
3.504937946833238e-35
```

Observations –

As p_value(3.5049e-35) < alpha(0.05) for a 95% confidence level, **all weather conditions have equal variance.**

As the assumptions for ANOVA testing are **not satisfied**, we check the dependency of the count of bikes rented with the weather conditions using **Kruskal Wallis hypothesis test**.

 # Kruskal Wallis test

H0: Number of bikes rented is independent of weather

Ha: Number of bikes rented depends on weather

significance (alpha) : 0.05

```
from scipy.stats import kruskal

df.weather1 = df.loc[df['weather'] == 'Clear/Partly cloudy', 'count']
df.weather2 = df.loc[df['weather'] == 'Mist/Cloudy', 'count']
df.weather3 = df.loc[df['weather'] == 'LightRains/ LightSnow', 'count']

test_stat, p_value = kruskal(df.weather1, df.weather2, df.weather3)
print('p-value: ', p_value)
```

```
p-value:  3.122066178659941e-45
```

Observations –

As p_value is very less than the significance value (alpha) i.e. (3.122e-45 < 0.05), we can conclude that with 95% confidence level, there are enough evidences to reject the null hypothesis. Hence, Count of bicycles being rented is dependent on the weather i.e. it is different for different weather.

Recommendations –

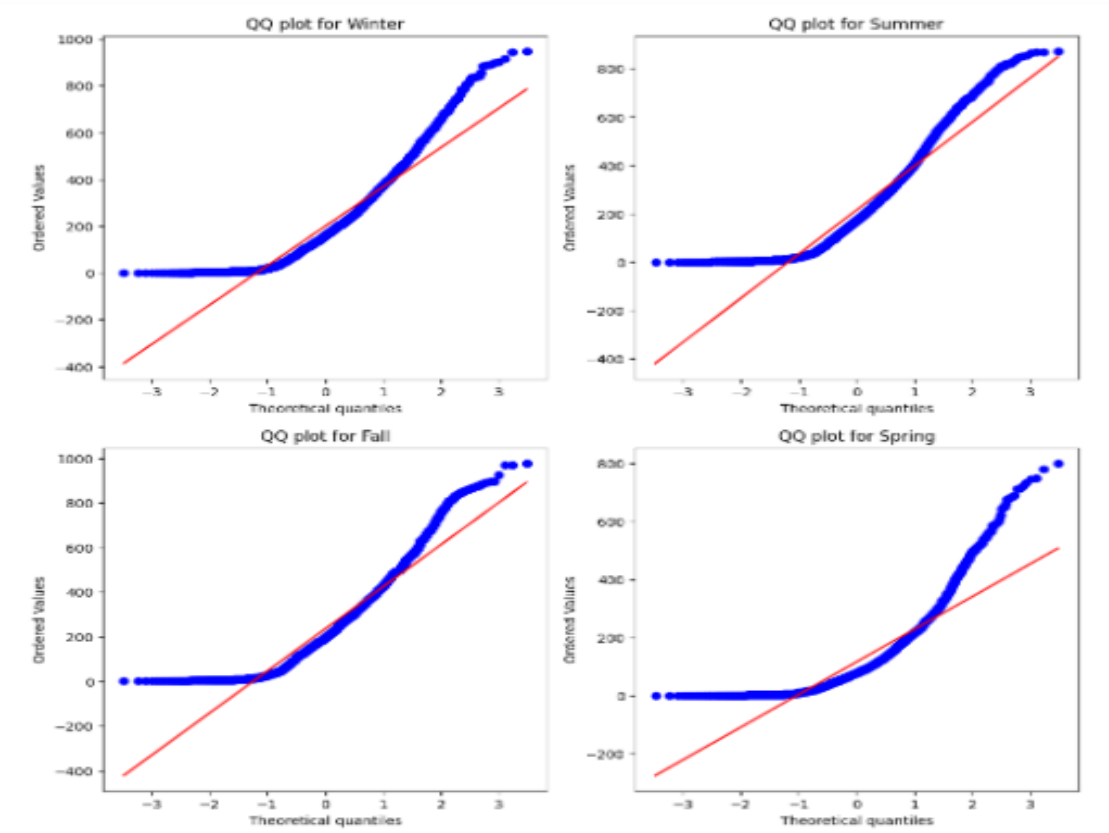
The company should make provisions for bikes to be used in bad weather conditions. Bikes with provisions like protective overhead covering for rains/snow must be incorporated to increase the demands for bikes even during bad weathers.

3. **Seasons**

 # Testing for assumptions of ANOVA

1. **Testing for Normality using QQ Plot**

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

plt.subplot(2, 2, 1)
spy.probplot(df.loc[df['season'] == 'winter', 'count'], plot = plt, dist = 'norm')
plt.title('QQ plot for Winter')

plt.subplot(2, 2, 2)
spy.probplot(df.loc[df['season'] == 'summer', 'count'], plot = plt, dist = 'norm')
plt.title('QQ plot for Summer')

plt.subplot(2, 2, 3)
spy.probplot(df.loc[df['season'] == 'fall', 'count'], plot = plt, dist = 'norm')
plt.title('QQ plot for Fall')

plt.subplot(2, 2, 4)
spy.probplot(df.loc[df['season'] == 'spring', 'count'], plot = plt, dist = 'norm')
plt.title('QQ plot for Spring')
plt.plot()
```



Observations -

It’s clear from the above analysis that **normality do not exist** for the count of bikes rented during various seasons.

2. **Testing for Normality using Shapiro-Wilk Test**

H0: It's a Gaussian/Normal distribution

Ha: It is not a Gaussian/Normal Distribution

significance(alpha): 0.05

```
from scipy.stats import shapiro

test_stat, p_value = shapiro(df.loc[df['season'] == 'winter', 'count'].sample(200))
print('p-value for Winter: ', p_value,'\n')

test_stat, p_value = shapiro(df.loc[df['season'] == 'summer', 'count'].sample(200))
print('p-value for Summer: ',p_value,'\n')

test_stat, p_value = shapiro(df.loc[df['season'] == 'fall', 'count'].sample(200))
print('p-value for Fall: ', p_value, '\n')

test_stat, p_value = shapiro(df.loc[df['season'] == 'spring', 'count'].sample(200))
print('p-value for Spring: ', p_value)
```

```
p-value for Winter:  2.3797198878128256e-08

p-value for Summer:  3.957505989582444e-10

p-value for Fall:  2.1343740108648035e-09

p-value for Spring:  5.803301395478858e-15
```

Observations –

Since, p_value < alpha(0.05), we can conclude that with enough evidence we reject the Null Hypothesis (H0) i.e. The above distribution is **not a normal distribution**.

Transforming the data using boxcox transformation and checking if the transformed data follows normal distribution

```
import scipy.stats as spy

transformed_winter = spy.boxcox(df.loc[df['season'] == 'winter', 'count'].sample(200))[0]
test_stat, p_value = shapiro(transformed_winter)
print('p-value after transformation for Winter: ', p_value,'\n')

transformed_summer = spy.boxcox(df.loc[df['season'] == 'summer', 'count'].sample(200))[0]
test_stat, p_value = shapiro(transformed_summer)
print('p-value after transformation for Summer: ', p_value,'\n')

transformed_fall = spy.boxcox(df.loc[df['season'] == 'fall', 'count'].sample(200))[0]
test_stat, p_value = shapiro(transformed_fall)
print('p-value after transformation for Fall: ', p_value, '\n')

transformed_spring = spy.boxcox(df.loc[df['season'] == 'spring', 'count'].sample(200))[0]
test_stat, p_value = shapiro(transformed_spring)
print('p-value after transformation for Spring: ', p_value)
```

```
p-value after transformation for Winter:  0.0016579122748225927

p-value after transformation for Summer:  0.00030584208434447646

p-value after transformation for Fall:  0.0014314993750303984

p-value after transformation for Spring:  0.01451499667018652
```

Observations –

- As p_values for the every season are less than alpha (0.05), we can conclude that with 95% confidence level we reject the null hypothesis. Hence, it **does not follow Normal Distribution.**

3. **Testing for Equal variance using Levene Test**

```
# H0 : all seasons are having equal variances
# Ha: they are not having equal variances

from scipy.stats import levene
statistic, p_value = levene(
 df[df['season'] == 'winter']['count'],
 df[df['season'] == 'summer']['count'],
 df[df['season'] == 'fall']['count'],
 df[df['season'] == 'spring']['count']
 )
```

```
p_value
```

```
1.0147116860043298e-118
```

Observations -

As p_value(1.015e-118) < alpha(0.05) for a 95% confidence level, **all seasons have equal variance.**

As the assumptions for **ANOVA testing are not satisfied**, we check the dependency of the count of bikes rented with the seasons using **Kruskal Wallis hypothesis test**.

 # Kruskal Wallis test

H0: Number of bikes rented is independent of weather

Ha: Number of bikes rented depends on weather

significance (alpha) : 0.05

```
from scipy.stats import kruskal

df.winter = df.loc[df['season'] == 'winter', 'count']
df.summer = df.loc[df['season'] == 'summer', 'count']
df.fall = df.loc[df['season'] == 'fall', 'count']
df.spring = df.loc[df['season'] == 'spring', 'count']

test_stat, p_value = kruskal(df.winter, df.summer, df.fall, df.spring)
print('p-value: ', p_value)
```

```
p-value:  2.4790083726176776e-151
```

Observations –

As p_value is very less than the significance value (alpha) i.e. (2.479e-151 < 0.05), we can concl ude that with 95% confidence level, there are enough evidences to reject the null hypothesis. Hen ce, Count of bicycles being rented is dependent on the seasons i.e. it is different for different seas ons.

Recommendations –

The company must increase the number of bikes for rent during specific seasons like summer abd fall when the demand for the bikes is high. Discount offers or some special offers can also be giv en during other seasons when the demand is less, in order to increase the sales.

4. **If weather and seasons are dependent using chi2_contingency test**

```
# H0: Weather and seasons are independent
# Ha: Weather and seasons are dependent
# sigificance (alpha) = 0.05

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

chi_stat, p_value, df, expected_values = chi2_contingency(vals)
```

```
p_value
```

```
1.5499250736864862e-07
```

Observations –

As p_value is very less than the significance value (alpha) i.e. (1.549e-07 < 0.05), we conclude that there are enough evidences to reject the null hypothesis. Hence, weather and seasons are dep endent on each other.

SUMMARY

Insights

- In **summer** and **fall** seasons more bikes are rented as compared to other seasons.
- Number of bikes rented is independent if it's a working day or a holiday.
- Whenever there is **rain, thunderstorm, snow or fog**, there were less bikes were rented.
- Whenever the humidity is less than 20, number of bikes rented is very very low.
- Whenever the temperature is less than 10, number of bikes rented is less.
- Whenever the windspeed is greater than 35, number of bikes rented is less.

Recommendations

- In **summer** and **fall** seasons the company should have more bikes in stock to be rented. Because the demand in these seasons is higher as compared to other seasons.
- In very low humid days, company should have less bikes in the stock to be rented.
- Whenever temperature is less than 10 or in very cold days, company should have less bikes.
- For bad weather conditions like rains/thunderstorms, the company should introduce bikes with protective overhead covering, to increase its demand in such weather conditions.