

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.

Column Profiling:

- datetime: datetime
- season: season (1: spring, 2: summer, 3: fall, 4: winter)
- holiday: whether day is a holiday or not (extracted from <http://dchr.dc.gov/page/holiday-schedule>)
- 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

1. Define the Problem Statement, Import the required Libraries and perform Exploratory Data Analysis.

Problem Statement:

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.

We have been given a dataset of the Count of rental bikes of Yulu along with datetime, atmospheric condition, and day type. We have to analyze attributes affecting the company's revenue and give feedback on conditions affecting it.

Import the Libraries:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels
from scipy.special import comb
from scipy.stats import binom
from scipy.stats import norm,t
from scipy.stats import poisson, expon,geom, ttest_1samp, ttest_ind,ttest_ind_from_stats
from scipy.stats import shapiro, levene, kruskal, chi2, chi2_contingency
from statsmodels.graphics.gofplots import qqplot
```

Importing the dataset:

```
[2] url="https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/original/bike_sharing.csv?1642089089"
df=pd.read_csv(url)
```

```
[3] 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](#)

a. Examine dataset structure, characteristics, and statistical summary.

```
[4] df.shape

(10886, 12)
```


The data has 10886 rows and 12 columns.

```
[5] 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
```

Among all the 12 columns we can see 8 integers, 3 float and 1 object data type column present. Need to change the datatype of some of the column to perform analysis.

Datetime column needs to change into datetime:

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

```
[12] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
#   Column      Non-Null Count  Dtype
---  -
0   datetime    10886 non-null  datetime64[ns]
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: datetime64[ns](1), float64(3), int64(8)
memory usage: 1020.7 KB
```

```
[20] for i in df.columns:
      print(f'{i} column --> has {df[i].nunique()} number of unique values.')
      print('-'*50)

"datetime" column --> has 10886 number of unique values.
-----
"season" column --> has 4 number of unique values.
-----
"holiday" column --> has 2 number of unique values.
-----
"workingday" column --> has 2 number of unique values.
-----
"weather" column --> has 4 number of unique values.
-----
"temp" column --> has 49 number of unique values.
-----
"atemp" column --> has 60 number of unique values.
-----
"humidity" column --> has 89 number of unique values.
-----
"windspeed" column --> has 28 number of unique values.
-----
"casual" column --> has 309 number of unique values.
-----
"registered" column --> has 731 number of unique values.
-----
"count" column --> has 822 number of unique values.
-----
```

From the above result we can see that season, holiday, workingday and weather have very less amount of unique values so we can convert them into categorical columns as well.

```
▶ for i in ['season', 'holiday', 'workingday', 'weather']:
    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  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
dtypes: datetime64[ns](1), float64(3), int64(4), object(4)
memory usage: 1020.7+ KB
```

b. Identify missing values and perform Imputation using an appropriate method.

```
[22] for i in df.columns:
      print(f'Column {i} has {sum(df[i].isna())} number of null values.')
      print('-'*50)
```

```
Column datetime has 0 number of null values.
-----
Column season has 0 number of null values.
-----
Column holiday has 0 number of null values.
-----
Column workingday has 0 number of null values.
-----
Column weather has 0 number of null values.
-----
Column temp has 0 number of null values.
-----
Column atemp has 0 number of null values.
-----
Column humidity has 0 number of null values.
-----
Column windspeed has 0 number of null values.
-----
Column casual has 0 number of null values.
-----
Column registered has 0 number of null values.
-----
Column count has 0 number of null values.
-----
```

There are no nan or null values present in any of the columns.

c. Identify and remove duplicate records.

```
df.nunique()
```

```
datetime    10886
season       4
holiday      2
workingday   2
weather      4
temp        49
atemp       60
humidity     89
windspeed    28
casual      309
registered   731
count       822
dtype: int64
```

```
[27] df.duplicated().value_counts()
```

```
False    10886
Name: count, dtype: int64
```

From above we can say that there are no row wise duplicate data present in the dataset.

d. Analyze the distribution of Numerical & Categorical variables, separately

i. Numerical Variables:

```
[34] #Statistical summary of numeric variables in the dataset
df.describe(exclude = ['datetime', 'object'])
```

	temp	atemp	humidity	windspeed	casual	registered	count
count	10886.00000	10886.00000	10886.00000	10886.00000	10886.00000	10886.00000	10886.00000
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

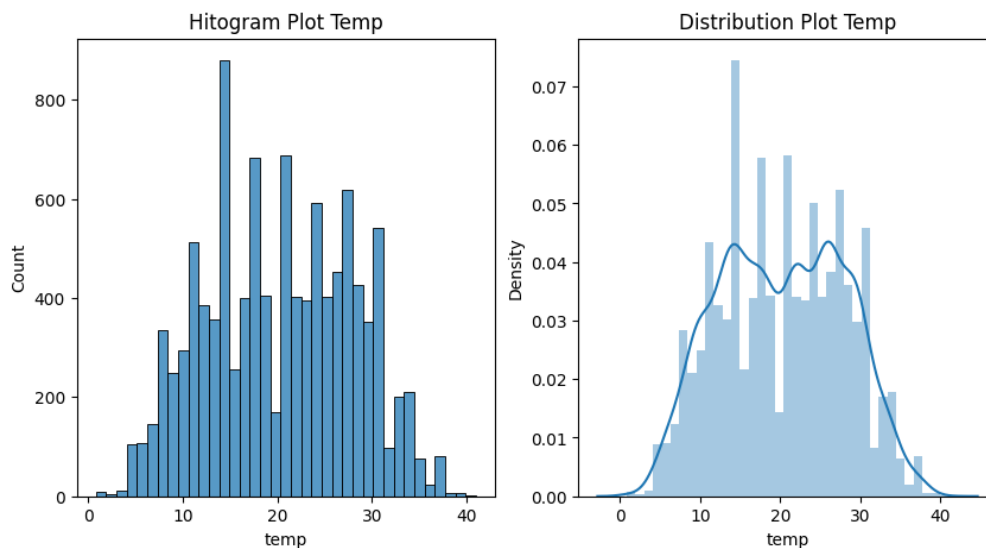
Temp:

```
[41] plt.figure(figsize=(10,5))

plt.subplot(1,2,1)
sns.histplot(df['temp'])
plt.title('Histogram Plot Temp')

plt.subplot(1,2,2)
sns.distplot(df['temp'])
plt.title('Distribution Plot Temp')

plt.show()
```



```
[44] temp_mean = df['temp'].mean().round(2)
temp_std = df['temp'].std().round(2)

print(f'The mean value of temp is {temp_mean} with standard deviation of {temp_std}.')
```

The mean value of temp is 20.23 with standard deviation of 7.79.

Histogram is representing the count of records in various temperature while distplot is representing the density distribution over various temperature.

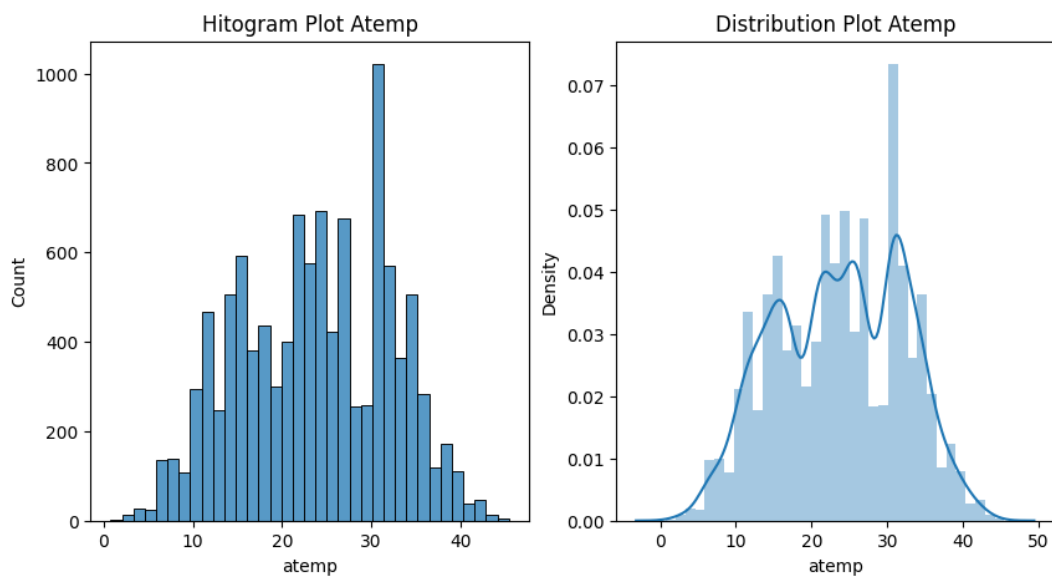
Atemp:

```
plt.figure(figsize=(10,5))

plt.subplot(1,2,1)
sns.histplot(df['atemp'])
plt.title('Histogram Plot Atemp')

plt.subplot(1,2,2)
sns.distplot(df['atemp'])
plt.title('Distribution Plot Atemp')

plt.show()
```



```
[45] atemp_mean = df['atemp'].mean().round(2)
      atemp_std = df['atemp'].std().round(2)

      print(f'The mean value of atemp is {atemp_mean} with standard deviation of {atemp_std}.')
```

The mean value of atemp is 23.66 with standard deviation of 8.47.

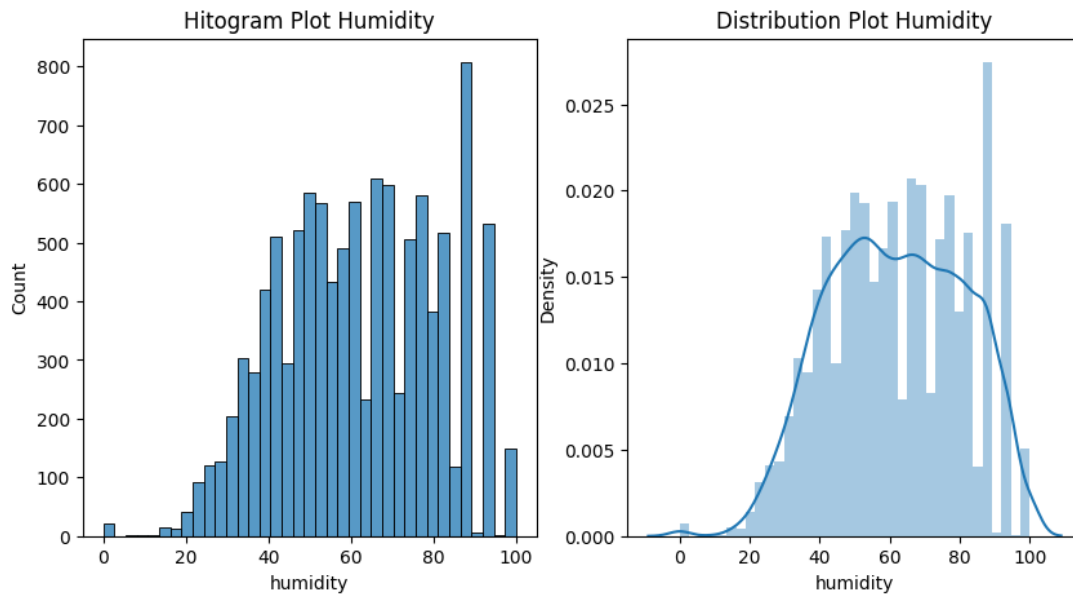
Humidity:

```
[47] plt.figure(figsize=(10,5))

      plt.subplot(1,2,1)
      sns.histplot(df['humidity'])
      plt.title('Histogram Plot Humidity')

      plt.subplot(1,2,2)
      sns.distplot(df['humidity'])
      plt.title('Distribution Plot Humidity')

      plt.show()
```



```
[50] humidity_mean = df['humidity'].mean().round(2)
      humidity_std = df['humidity'].std().round(2)

      print(f'The mean value of humidity is {humidity_mean} with standard deviation of {humidity_std}.')
```

The mean value of humidity is 61.89 with standard deviation of 19.25.

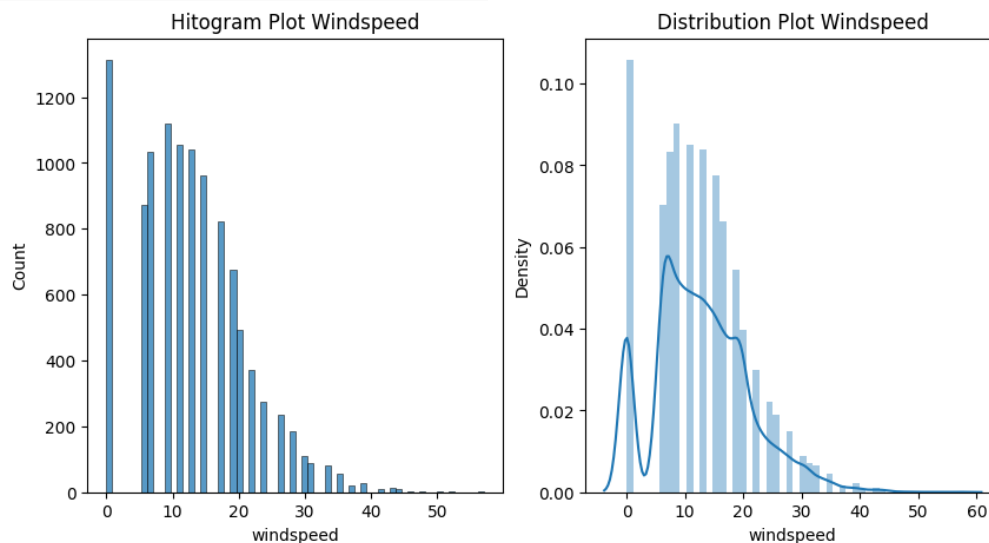
Windspeed:

```
[49] plt.figure(figsize=(10,5))

      plt.subplot(1,2,1)
      sns.histplot(df['windspeed'])
      plt.title('Hitogram Plot Windspeed')

      plt.subplot(1,2,2)
      sns.distplot(df['windspeed'])
      plt.title('Distribution Plot Windspeed')

      plt.show()
```




```
[51] windspeed_mean = df['windspeed'].mean().round(2)
      windspeed_std = df['windspeed'].std().round(2)

      print(f'The mean value of windspeed is {windspeed_mean} with standard deviation of {windspeed_std}.')
```

The mean value of windspeed is 12.8 with standard deviation of 8.16.

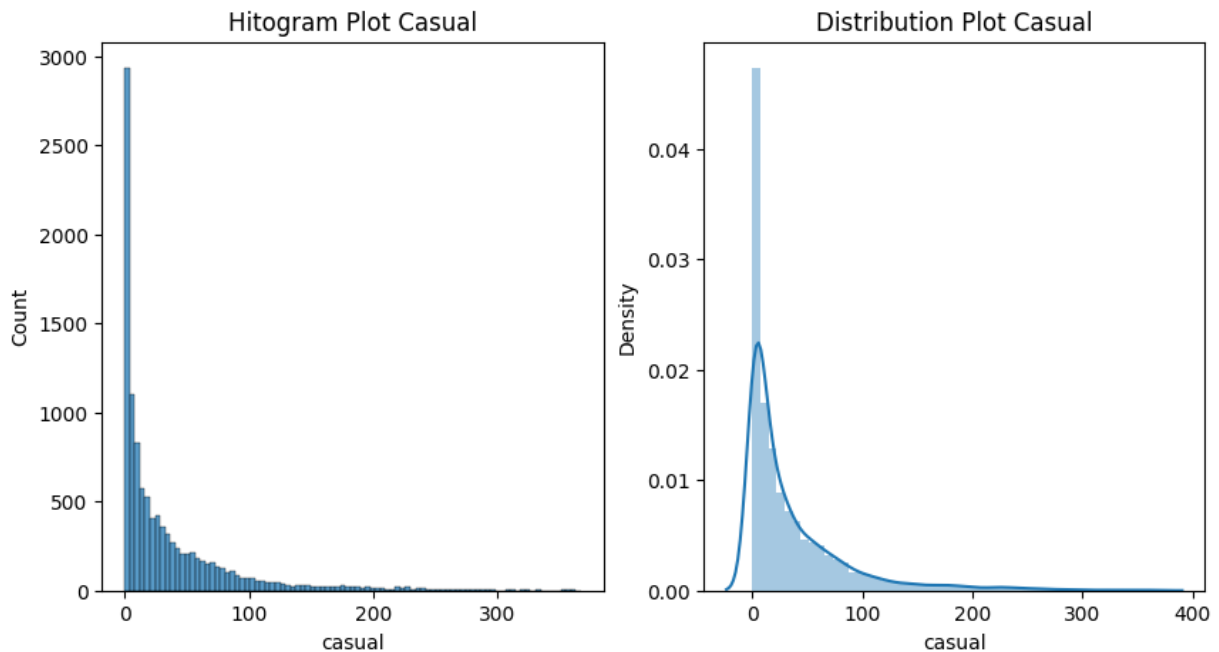
Casual:

```
plt.figure(figsize=(10,5))

plt.subplot(1,2,1)
sns.histplot(df['casual'])
plt.title('Hitogram Plot Casual')

plt.subplot(1,2,2)
sns.distplot(df['casual'])
plt.title('Distribution Plot Casual')

plt.show()
```



```
[53] casual_mean = df['casual'].mean().round(2)
      casual_std = df['casual'].std().round(2)

      print(f'The mean value of casual is {casual_mean} with standard deviation of {casual_std}.')
```

The mean value of casual is 36.02 with standard deviation of 49.96.

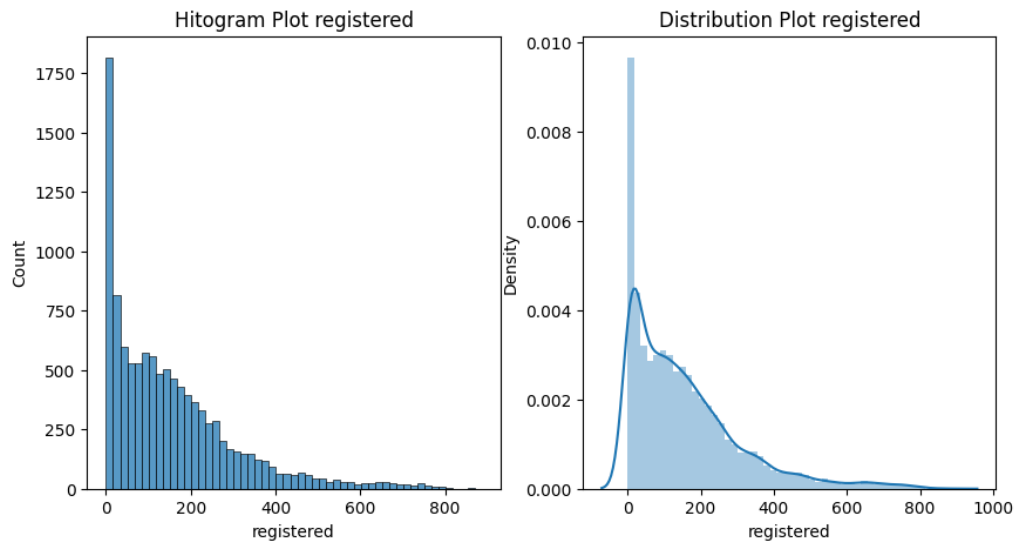
Registered:

```
plt.figure(figsize=(10,5))

plt.subplot(1,2,1)
sns.histplot(df['registered'])
plt.title('Histogram Plot registered')

plt.subplot(1,2,2)
sns.distplot(df['registered'])
plt.title('Distribution Plot registered')

plt.show()
```



```
registered_mean = df['registered'].mean().round(2)
registered_std = df['registered'].std().round(2)

print(f'The mean value of registered is {registered_mean} with standard deviation of {registered_std}.')

The mean value of registered is 155.55 with standard deviation of 151.04.
```

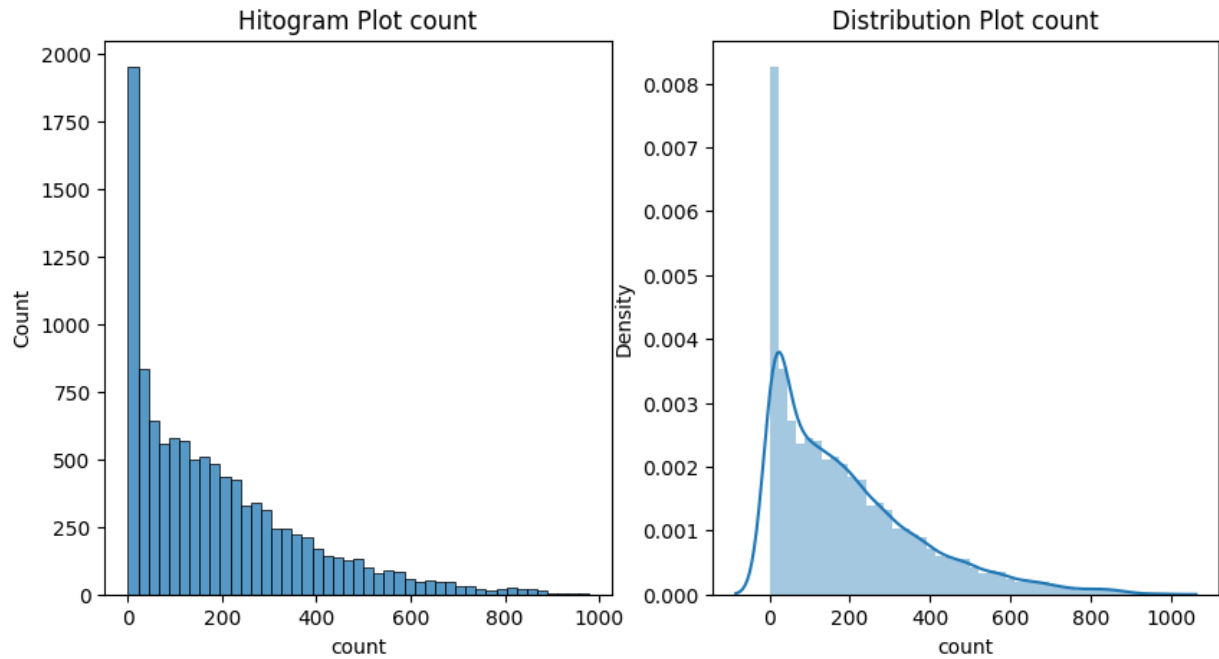
Count:

```
[57] plt.figure(figsize=(10,5))

plt.subplot(1,2,1)
sns.histplot(df['count'])
plt.title('Histogram Plot count')

plt.subplot(1,2,2)
sns.distplot(df['count'])
plt.title('Distribution Plot count')

plt.show()
```



```
count_mean = df['count'].mean().round(2)
count_std = df['count'].std().round(2)

print(f'The mean value of count is {count_mean} with standard deviation of {count_std}.')
```

➞ The mean value of count is 191.57 with standard deviation of 181.14.

ii. Categorical Variable:

```
df.describe(include = 'object')
```

	season	holiday	workingday	weather
count	10886	10886	10886	10886
unique	4	2	2	4
top	4	0	1	1
freq	2734	10575	7412	7192

Season:

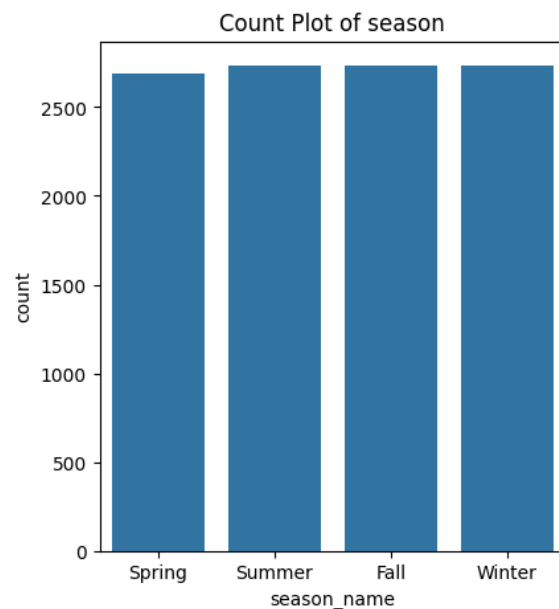
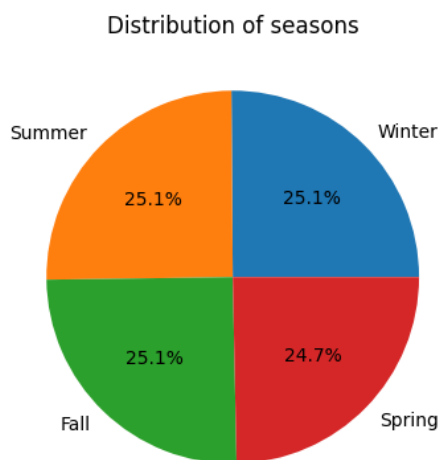
Let's change the numerical values into actual names given.

```
def season(x):  
    if x == 1:  
        return 'Spring'  
    elif x == 2:  
        return 'Summer'  
    elif x == 3:  
        return 'Fall'  
    else:  
        return 'Winter'  
  
df['season_name'] = df['season'].apply(season)
```

```
[76] df.head()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	season_name
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16	Spring
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40	Spring
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32	Spring
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13	Spring
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1	Spring

```
[77] plt.figure(figsize=(10,5))  
  
plt.subplot(1,2,1)  
plt.pie(df['season_name'].value_counts().values, labels = df['season_name'].value_counts().index, autopct = '%1.1f%%')  
plt.title('Distribution of seasons')  
  
plt.subplot(1,2,2)  
sns.countplot(data = df, x = 'season_name')  
plt.title('Count Plot of season')  
  
plt.show()
```



Holiday:

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

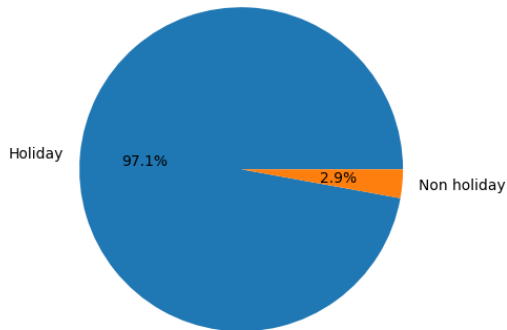
plt.subplot(1,2,1)
plt.pie(df['holiday'].value_counts().values, labels = ['Holiday', 'Non holiday'], autopct = '%1.1f%%')
plt.title('Distribution of holiday')

plt.subplot(1,2,2)
sns.countplot(data = df, x = 'holiday')
plt.title('Count Plot of holiday')

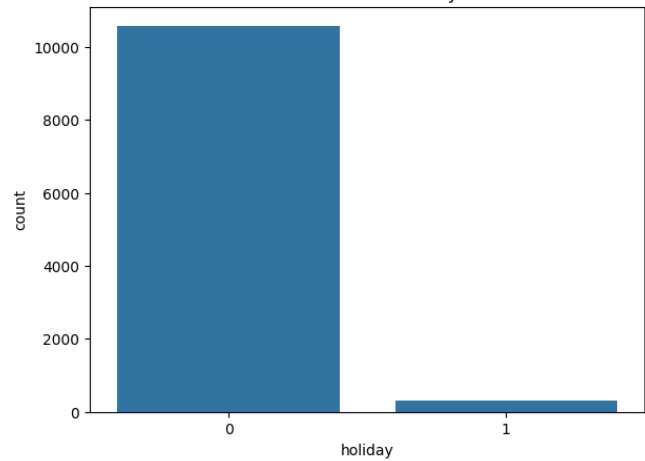
plt.show()
```



Distribution of holiday



Count Plot of holiday



Workingday:

```
[83] plt.figure(figsize=(15,5))

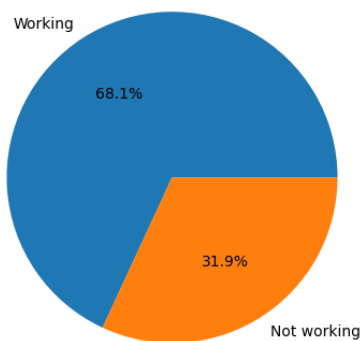
plt.subplot(1,2,1)
plt.pie(df['workingday'].value_counts().values, labels = ['Working', 'Not working'], autopct = '%1.1f%%')
plt.title('Distribution of workingday')

plt.subplot(1,2,2)
sns.countplot(data = df, x = 'workingday')
plt.title('Count Plot of workingday')

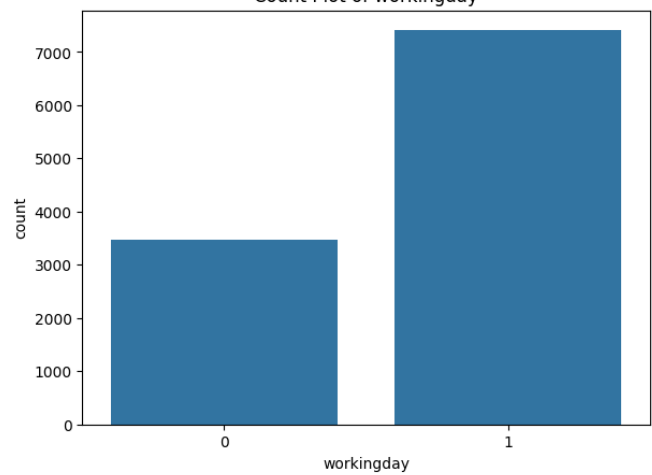
plt.show()
```



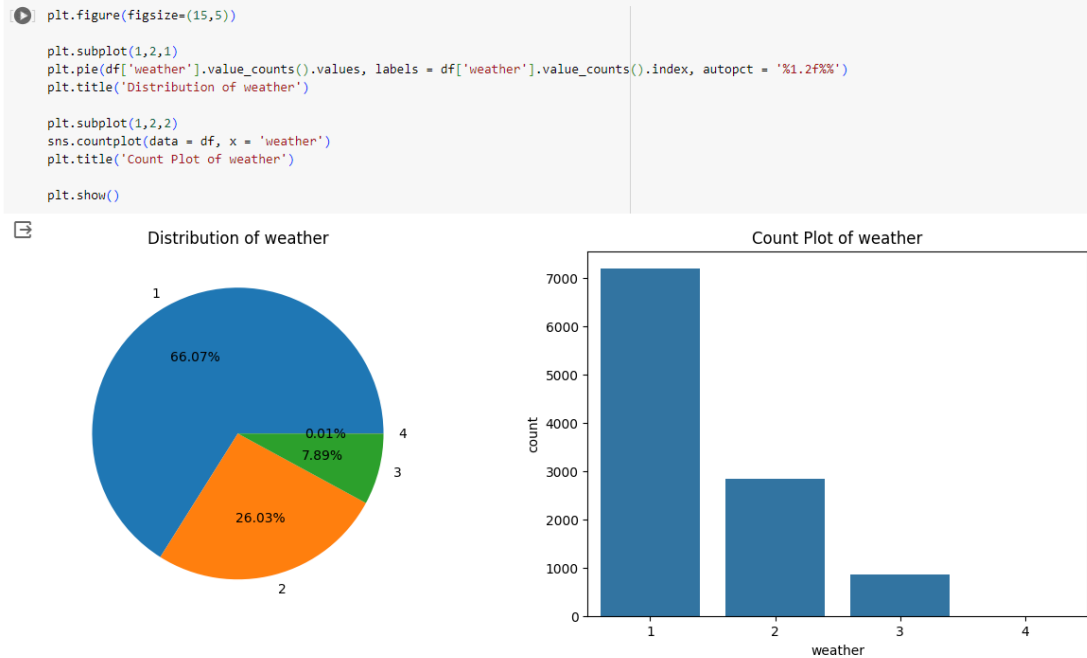
Distribution of workingday



Count Plot of workingday



Weather:



e. Check for Outliers and deal with them accordingly.

We have total 7 numerical variables are there let's calculate the outliers for all.

```
[99] for i in df.describe(exclude = ['datetime', 'object']).columns:
    q1 = np.quantile(df[i], 0.25).round(2)
    q3 = np.quantile(df[i], 0.75).round(2)
    iqr = (q3-q1).round(2)
    iqr_range = ((q1-1.5*iqr).round(2), (q3+1.5*iqr).round(2))
    print(f'For numerical variable "{i}"')
    print(f'25% value is = {q1}')
    print(f'75% value is = {q3}')
    print(f'IQR value is = {iqr}')
    print(f'The outliers are values less than {iqr_range[0]} and greater than {iqr_range[1]}')
    print('-'*50)
```

For numerical variable "temp"

25% value is = 13.94

75% value is = 26.24

IQR value is = 12.3

The outliers are values less than -4.51 and greater than 44.69

For numerical variable "atemp"

25% value is = 16.66

75% value is = 31.06

IQR value is = 14.4

The outliers are values less than -4.94 and greater than 52.66

For numerical variable "humidity"

25% value is = 47.0

75% value is = 77.0

IQR value is = 30.0

The outliers are values less than 2.0 and greater than 122.0

```

For numerical variable "windspeed"
25% value is = 7.0
75% value is = 17.0
IQR value is = 10.0
The outliers are values less than -8.0 and greater than 32.0
-----
For numerical variable "casual"
25% value is = 4.0
75% value is = 49.0
IQR value is = 45.0
The outliers are values less than -63.5 and greater than 116.5
-----
For numerical variable "registered"
25% value is = 36.0
75% value is = 222.0
IQR value is = 186.0
The outliers are values less than -243.0 and greater than 501.0
-----
For numerical variable "count"
25% value is = 42.0
75% value is = 284.0
IQR value is = 242.0
The outliers are values less than -321.0 and greater than 647.0
-----

```

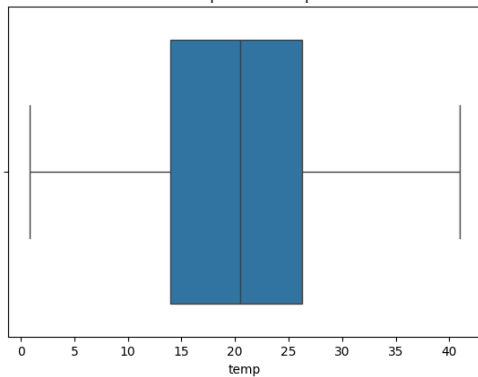
```

[ ] plt.figure(figsize=(15,22))
    cnt = 1

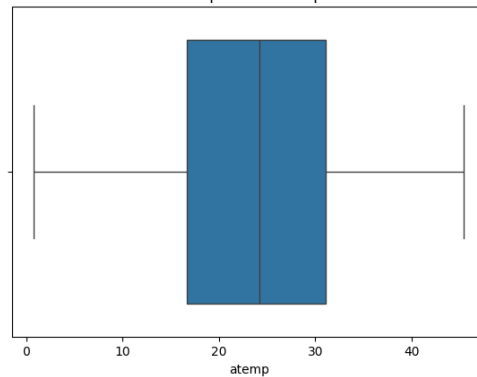
    for i in df.describe(exclude = ['datetime', 'object']).columns:
        plt.subplot(4,2,cnt)
        sns.boxplot(data = df, x = df[i])
        plt.title(f'Outlier in box plot for "{i}" column.')
        cnt += 1

```

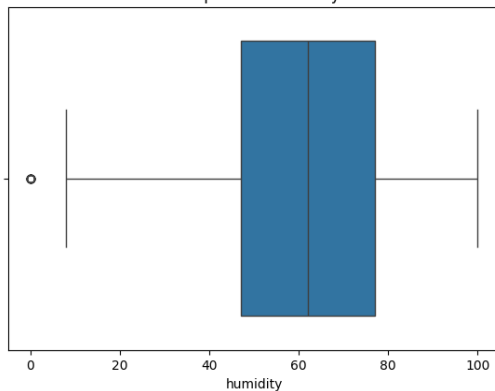
Outlier in box plot for "temp" column.



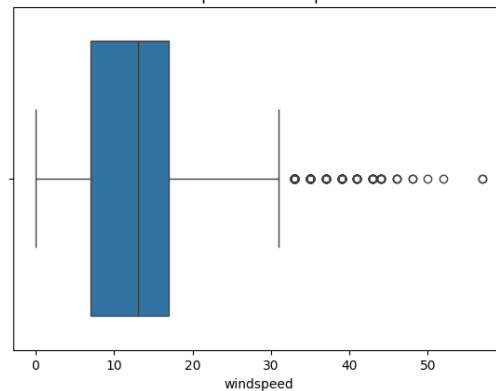
Outlier in box plot for "atemp" column.

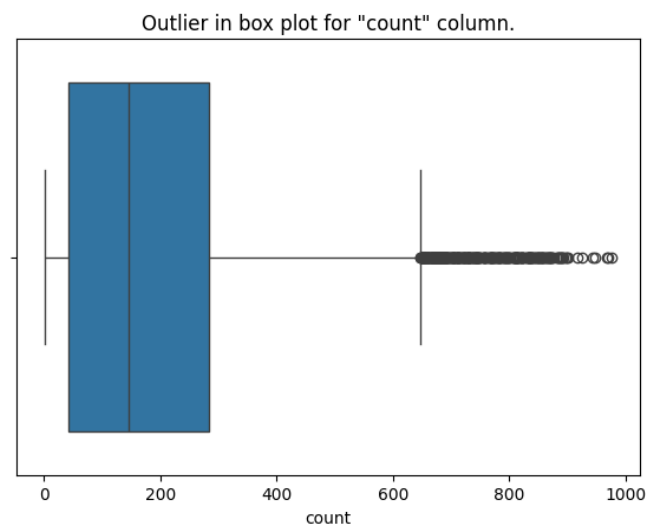
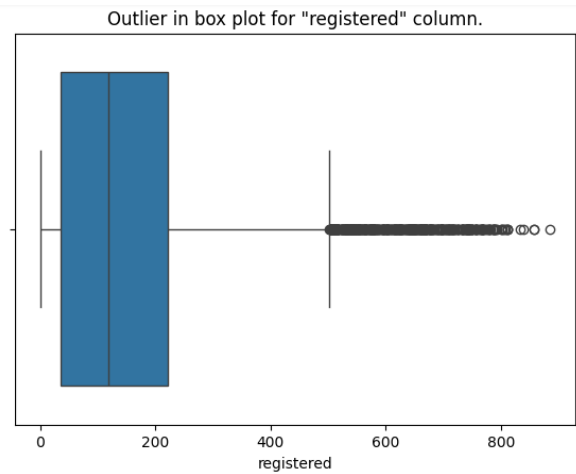
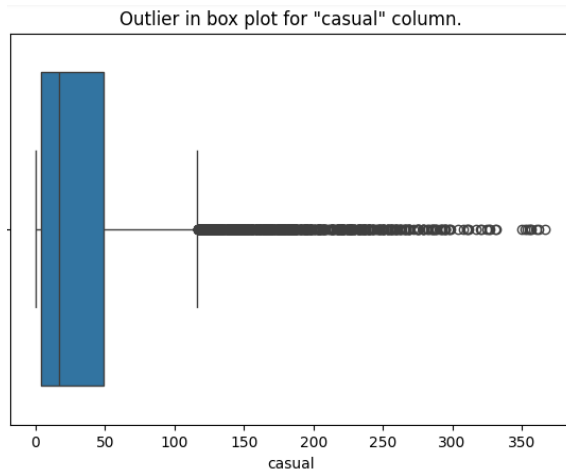


Outlier in box plot for "humidity" column.



Outlier in box plot for "windspeed" column.





Bivariate Analysis:

Season wise count based on working day:

```
plt.figure(figsize = (20, 5))
```

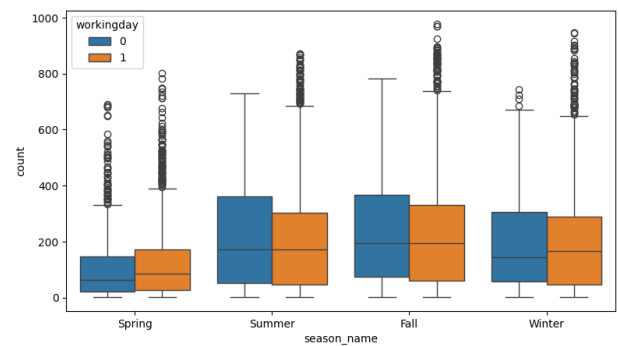
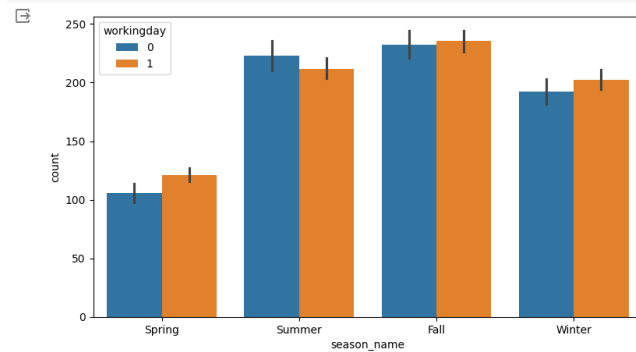
```
plt.subplot(1,2,1)
```

```
sns.barplot(data = df, x = 'season_name', y = 'count', hue = 'workingday')
```

```
plt.subplot(1,2,2)
```

```
sns.boxplot(data = df, x = 'season_name', y = 'count', hue = 'workingday')
```

```
plt.show()
```



There no such difference in count for working and non-working days but we can see spring has less number of count among all the seasons.

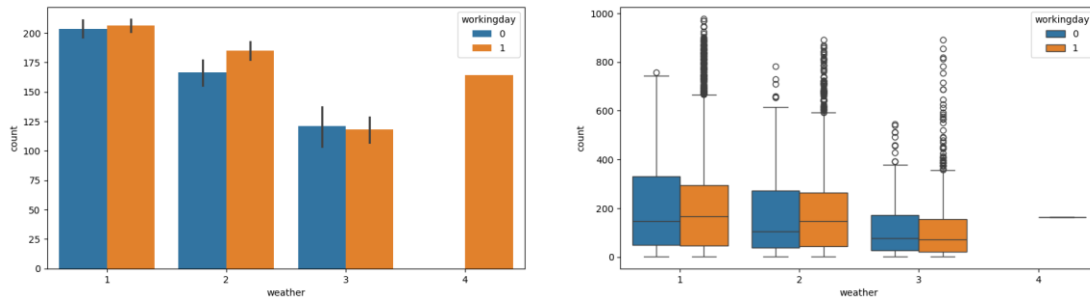
Weather wise count based on working day:

```
plt.figure(figsize = (20, 5))

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

plt.subplot(1,2,2)
sns.boxplot(data = df, x = 'weather', y = 'count', hue = 'workingday')

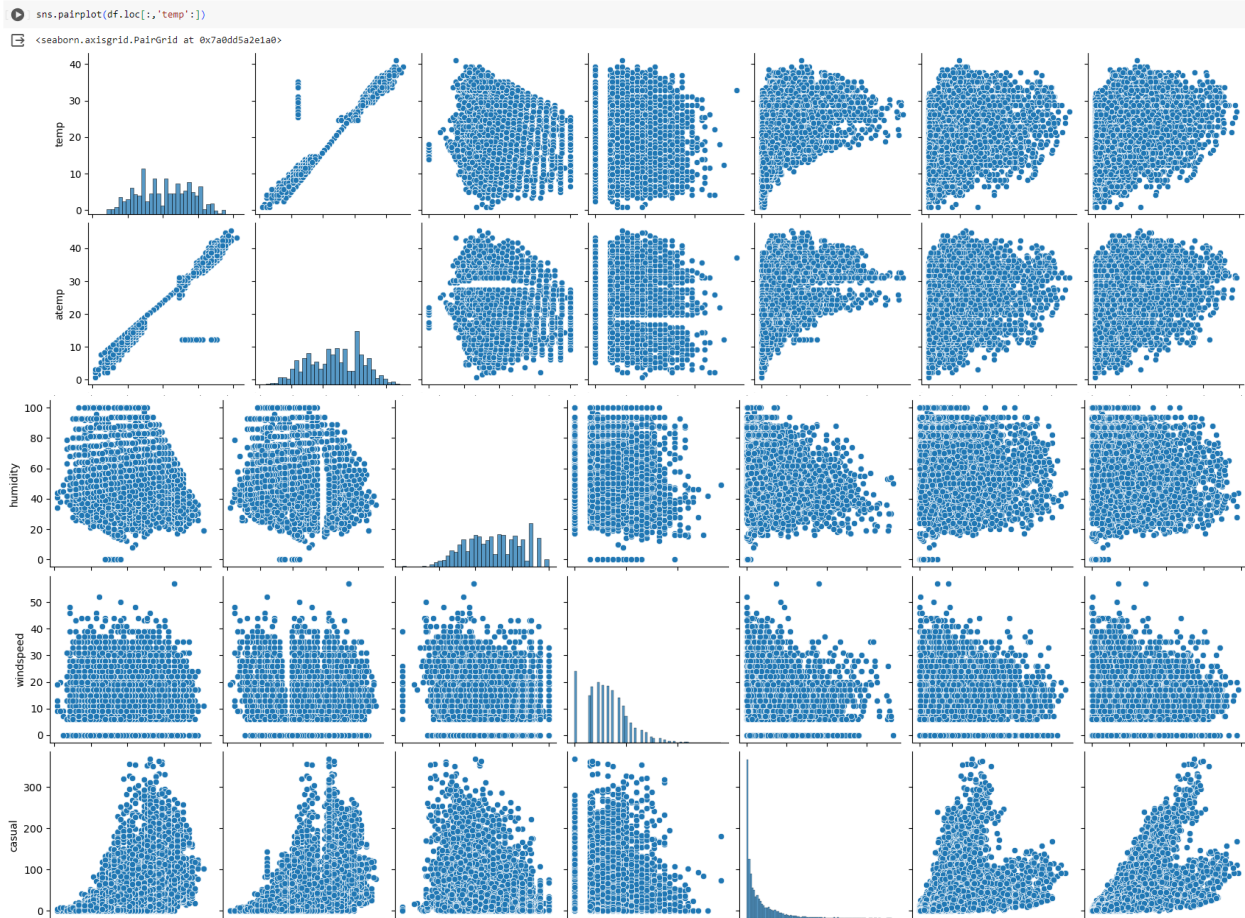
plt.show()
```

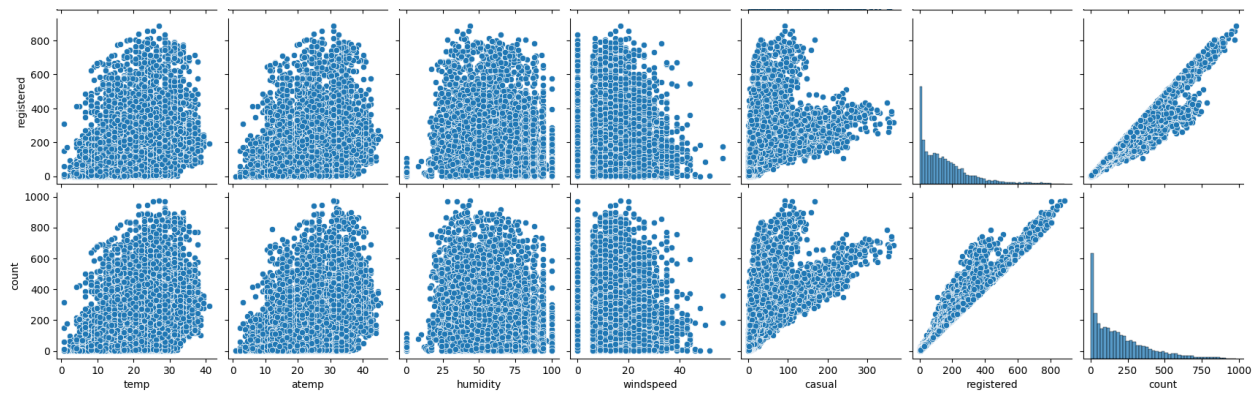


The hourly count of total rental bikes is higher in the clear and cloudy weather, followed by the misty weather and rainy weather. There are very few records for extreme weather conditions.

2. Try establishing a Relationship between the Dependent and Independent Variables.

a. Pair plot among the numerical variables:





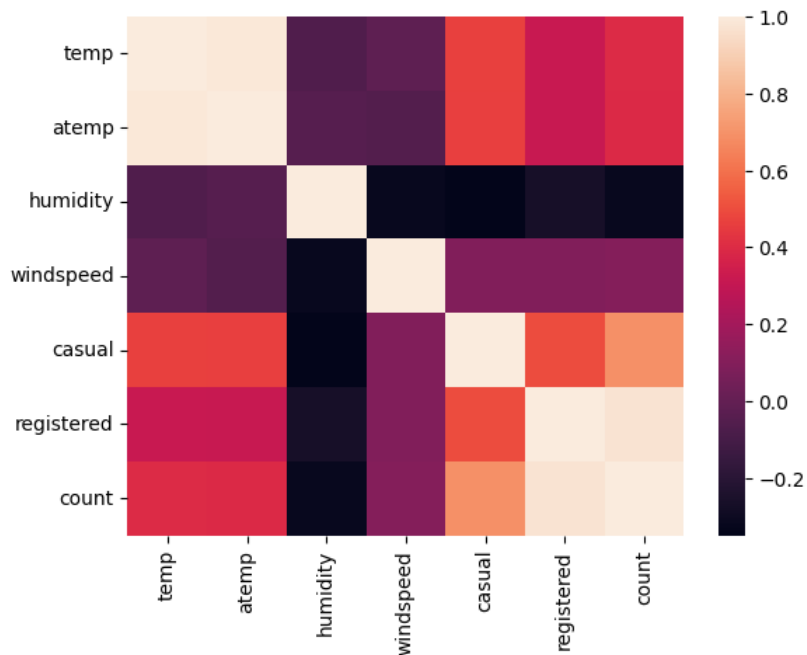
b. Correlation between variables:

```
[115] df.loc[:, 'temp':'count'].corr()
```

	temp	atemp	humidity	windspeed	casual	registered	count
temp	1.000000	0.984948	-0.064949	-0.017852	0.467097	0.318571	0.394454
atemp	0.984948	1.000000	-0.043536	-0.057473	0.462067	0.314635	0.389784
humidity	-0.064949	-0.043536	1.000000	-0.318607	-0.348187	-0.265458	-0.317371
windspeed	-0.017852	-0.057473	-0.318607	1.000000	0.092276	0.091052	0.101369
casual	0.467097	0.462067	-0.348187	0.092276	1.000000	0.497250	0.690414
registered	0.318571	0.314635	-0.265458	0.091052	0.497250	1.000000	0.970948
count	0.394454	0.389784	-0.317371	0.101369	0.690414	0.970948	1.000000

```
[116] sns.heatmap(data = df.loc[:, 'temp':'count'].corr())
```

<Axes: >



3. Check if there any significant difference between the no. of bike rides on Weekdays and Weekends?

```
[118] df['workingday'].value_counts()
```

```
workingday
1    7412
0    3474
Name: count, dtype: int64
```

```
[119] working_day_count = df[df['workingday'] == 1]['count']
      non_working_day_count = df[df['workingday'] == 0]['count']
```

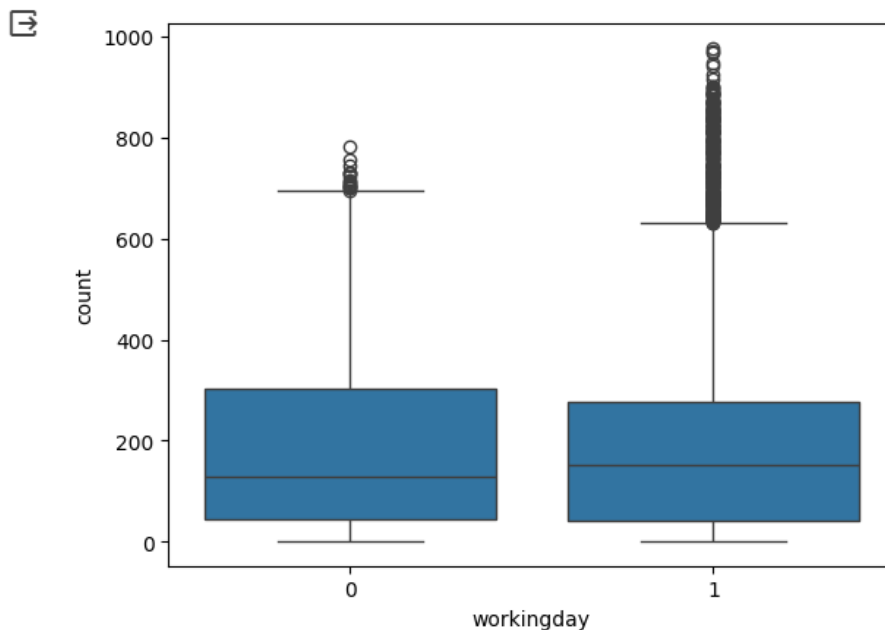
```
[121] working_day_count.mean(), working_day_count.std()
```

```
(193.01187263896384, 184.5136590421481)
```

```
[122] non_working_day_count.mean(), non_working_day_count.std()
```

```
(188.50662061024755, 173.7240153250003)
```

```
▶ sns.boxplot(data = df, x = 'workingday', y = 'count')
  plt.show()
```



Step 01: Set up the null hypothesis.

NULL Hypothesis H0: The mean value of count in working day is same as non-working day. Means there is no effect of working day on electric cycle rented count.

Alternative Hypothesis HA: The means are not same. Means there is effect of working days on electric cycle rent count.

Step 02: Checking for basic assumptions for the hypothesis

We have two categories to compare so we will perform 2 sample T Test.

Step 03: Define the test statistics

We will compare the mean of the samples.

Step 04: Calculate the p value and define alpha

We set our alpha as 0.05

Step 05: We will compare the p value and alpha

If $p > \alpha$ means p is high null will fly, we will fail to reject null

If $p < \alpha$ means p is low null will go, we reject null.

```
[126] t_stat, p_value = ttest_ind(working_day_count, non_working_day_count, alternative = 'two-sided')
      t_stat, p_value
      (1.2096277376026694, 0.22644804226361348)

[127] alpha = 0.05

if p_value > alpha:
    print('We fail to reject null, The mean value of count in working day is same as non-working day. Means there is no effect of working day on electric cycle rented count.')
else:
    print('We reject the null, The means are not same. Means there is effect of working days on electric cycle rent count.')

We fail to reject null, The mean value of count in working day is same as non-working day. Means there is no effect of working day on electric cycle rented count.
```

Insight:

Here $p_value = 0.22$ which is greater than 0.05, hence the mean of electric cycle rent have no effect due to working days.

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.

4. Check if the demand of bicycles on rent is the same for different Weather conditions?

As we have 4 types of weather condition we need to perform Anova or Kruskal Walis test to determine the significance of weather on electric cycle rent count.

```
[189] weather_1 = df[df['weather'] == 1]['count']
      weather_2 = df[df['weather'] == 2]['count']
      weather_3 = df[df['weather'] == 3]['count']
      weather_4 = df[df['weather'] == 4]['count']
```

```
[190] weather_4
```

```
5631    164
      Name: count, dtype: int64
```

As weather_4 has only one row we will ignore it in our calculation.

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

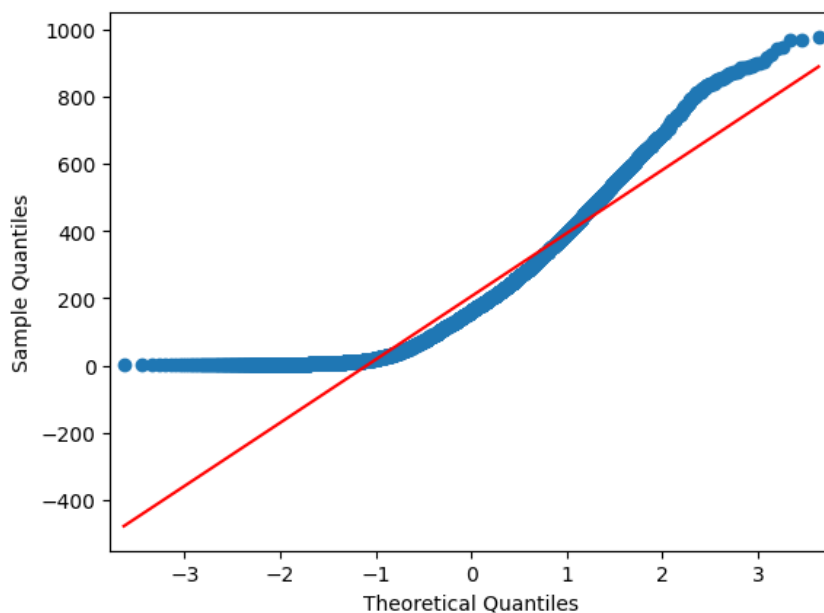
Weather 01:

```
[191] # Let's perform Shapiro and QQ Plot to check the normal distribution
      alpha = 0.05
      test_stat, p_value = shapiro(weather_1)

      print('P Value is : ', round(p_value,4))
      if p_value > alpha:
          print('Fail to reject null, means the weather_1 follows Gaussian distribution.')
      else:
          print('Reject null, means the weather_1 does not follows Gaussian distribution.')
```

```
P Value is : 0.0
Reject null, means the weather_1 does not follows Gaussian distribution.
```

```
[195] qqplot(weather_1, line = 's')
      plt.show()
```



Insight:

From both the analysis we can conclude that the weather_1 doesn't follow normal distribution.

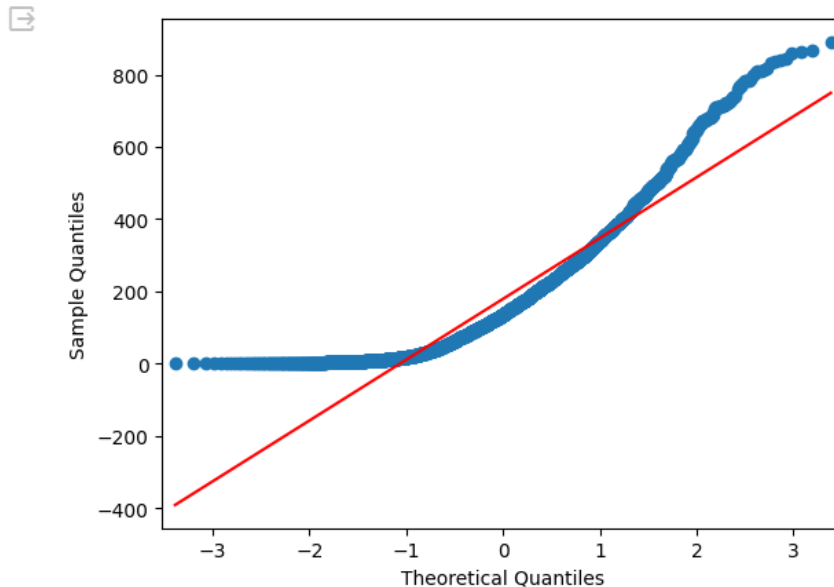
Weather 02:

```
[196] # Let's perform Shapiro and QQ Plot to check the normal distribution
      alpha = 0.05
      test_stat, p_value = shapiro(weather_2)

      print('P Value is : ', round(p_value,4))
      if p_value > alpha:
          print('Fail to reject null, means the weather_2 follows Gaussian distribution.')
      else:
          print('Reject null, means the weather_2 does not follows Gaussian distribution.')
```

➡ P Value is : 0.0
Reject null, means the weather_2 does not follows Gaussian distribution.

```
[197] qqplot(weather_2, line = 's')
      plt.show()
```



Insight:

Weather_02 doesn't follow the normal distribution.

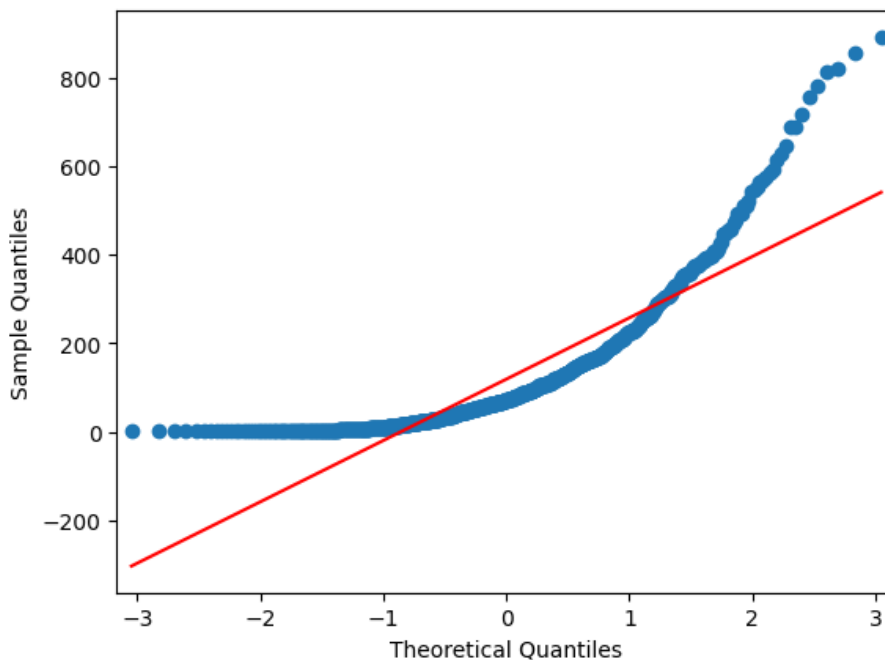
Weather 03:

```
[▶] # Let's perform Shapiro and QQ Plot to check the normal distribution
     alpha = 0.05
     test_stat, p_value = shapiro(weather_3)

     print('P Value is : ', round(p_value,4))
     if p_value > alpha:
         print('Fail to reject null, means the weather_3 follows Gaussian distribution.')
     else:
         print('Reject null, means the weather_3 does not follows Gaussian distribution.')
```

➡ P Value is : 0.0
Reject null, means the weather_3 does not follows Gaussian distribution.

```
[199] qqplot(weather_3, line = 's')  
      plt.show()
```



Insight:

Weather_03 doesn't follow the normal distribution.

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

```
[200] #Levene Test  
  
alpha = 0.05  
  
test_stat, p_value = levene(weather_1, weather_2, weather_3)  
print('P Value is : ', round(p_value,4))  
if p_value > alpha:  
    print('Fail to reject null, variance of all the samples are same.')  
else:  
    print('Reject null, variance of all the samples are different.')  
  
P Value is : 0.0  
Reject null, variance of all the samples are different.
```

Insight:

As all the samples are not following normal distribution and have different variance we cannot perform Anova we need to 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

```
[201] # Kruskal Wallis
      alpha=0.05

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 Hypotheis, 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
```

5. Check if the demand of bicycles on rent is the same for different Seasons?

```
[202] df['season_name'].value_counts()

season_name
Winter    2734
Summer    2733
Fall      2733
Spring    2686
Name: count, dtype: int64
```

As we have 4 types of seasons so we need to perform Anova or Kruskal Walis test to determine the significance of seasons on electric cycle rent count.

```
▶ winter_count = df[df['season_name'] == 'Winter']['count']
summer_count = df[df['season_name'] == 'Summer']['count']
fall_count = df[df['season_name'] == 'Fall']['count']
spring_count = df[df['season_name'] == 'Spring']['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

Winter Season:

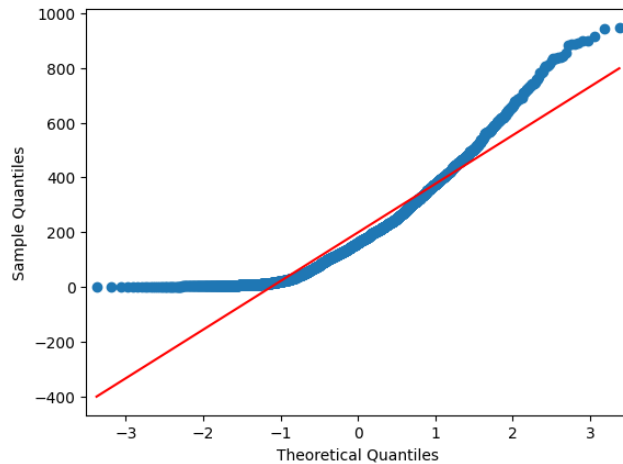
```
▶ # Let's perform Shapiro and QQ Plot to check the normal distribution
alpha = 0.05
test_stat, p_value = shapiro(winter_count)

print('P Value is : ', round(p_value,4))
if p_value > alpha:
    print('Fail to reject null, means the winter_count follows Gaussian distribution.')
else:
    print('Reject null, means the winter_count does not follows Gaussian distribution.')

P Value is : 0.0
Reject null, means the winter_count does not follows Gaussian distribution.
```



```
[205] qqplot(winter_count, line = 's')
plt.show()
```



Here Plot not matching with straight line so based on that we can say that sample does not follow normal distribution

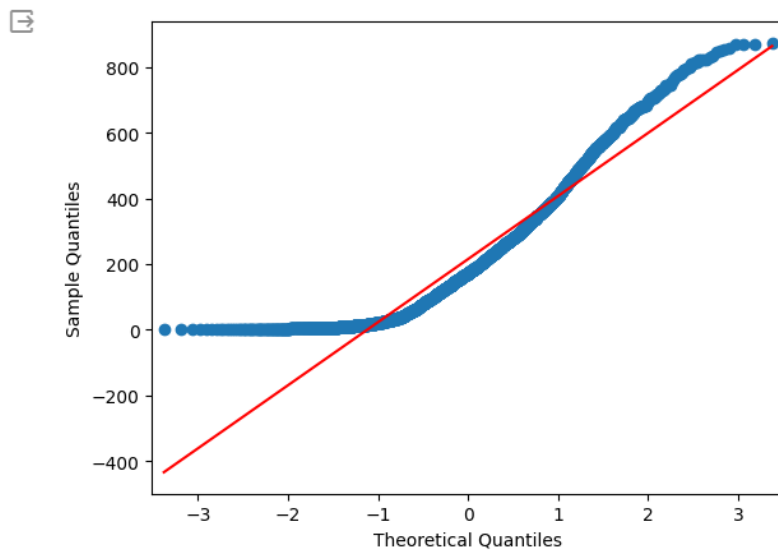
Summer session:

```
[206] # Let's perform Shapiro and QQ Plot to check the normal distribution
alpha = 0.05
test_stat, p_value = shapiro(summer_count)

print('P Value is : ', round(p_value,4))
if p_value > alpha:
    print('Fail to reject null, means the summer_count follows Gaussian distribution.')
else:
    print('Reject null, means the summer_count does not follows Gaussian distribution.')
```

P Value is : 0.0
Reject null, means the summer_count does not follows Gaussian distribution.

```
[207] qqplot(summer_count, line = 's')
plt.show()
```



Here Plot not matching with straight line so based on that we can say that sample does not follow normal distribution

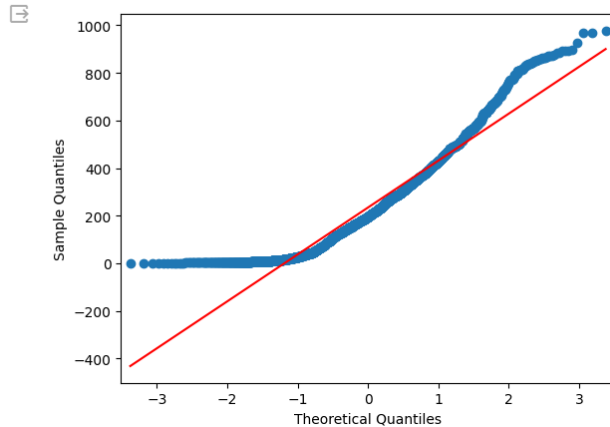
Fall Session:

```
[208] # Let's perform Shapiro and QQ Plot to check the normal distribution
alpha = 0.05
test_stat, p_value = shapiro(fall_count)

print('P Value is : ', round(p_value,4))
if p_value > alpha:
    print('Fail to reject null, means the fall_count follows Gaussian distribution.')
else:
    print('Reject null, means the fall_count does not follows Gaussian distribution.')

P Value is : 0.0
Reject null, means the fall_count does not follows Gaussian distribution.
```

```
[209] qqplot(fall_count, line = 's')
plt.show()
```



Here Plot not matching with straight line so based on that we can say that sample does not follow normal distribution.

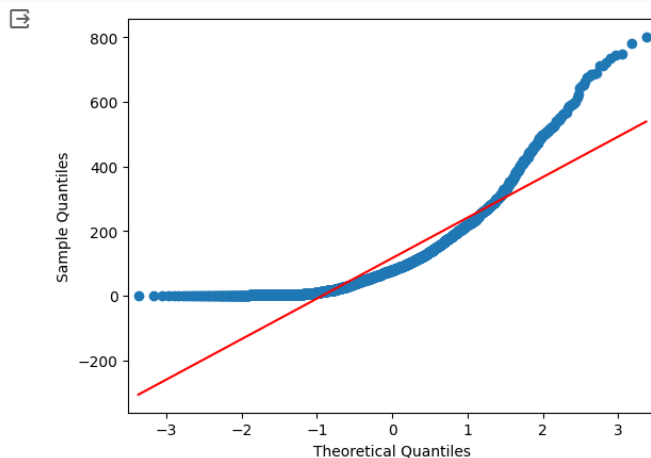
Spring Session:

```
[210] # Let's perform Shapiro and QQ Plot to check the normal distribution
alpha = 0.05
test_stat, p_value = shapiro(spring_count)

print('P Value is : ', round(p_value,4))
if p_value > alpha:
    print('Fail to reject null, means the spring_count follows Gaussian distribution.')
else:
    print('Reject null, means the spring_count does not follows Gaussian distribution.')

P Value is : 0.0
Reject null, means the spring_count does not follows Gaussian distribution.
```

```
[211] qqplot(spring_count, line = 's')
plt.show()
```



Here Plot not matching with straight line so based on that we can say that sample does not follow normal distribution.

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

```
[212] #Levene Test

alpha = 0.05

test_stat, p_value = levene(summer_count, winter_count, fall_count, spring_count)
print('P Value is : ', round(p_value,4))
if p_value > alpha:
    print('Fail to reject null, variance of all the samples are same.')
else:
    print('Reject null, variance of all the samples are different.')

P Value is : 0.0
Reject null, variance of all the samples are different.
```

Insight:

As all the samples are not following normal distribution and have different variance we cannot perform Anova we need to go for Kruskal Wallis test.

Null and Alternate Hypothesis for Kruskal Wallis Test

H0: mean of total rental bikes of different seasons are same

Ha: mean of total rental bikes of different seasons are not same

```
[213] # Kruskal Wallis
alpha=0.05

test_statistics,p_value=kruskal(summer_count, winter_count, fall_count, spring_count)
print("p-value:", round(p_value,4))
if p_value < alpha:
    print("Reject Null Hypotheis, mean of total rental bikes of different seasons are not same")
else:
    print("Fail to Reject Null Hypothesis, mean of total rental bikes of different seasons are same")

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

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

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

6. Check if the Weather conditions are significantly different during different Seasons?

Here we are checking between two categorical variables hence we need to perform Chi2 test.

```
[215] w_s = pd.crosstab(df['season_name'], df['weather'])  
w_s
```

weather	1	2	3	4
season_name				
Fall	1930	604	199	0
Spring	1759	715	211	1
Summer	1801	708	224	0
Winter	1702	807	225	0

Next steps:

[Generate code with w_s](#)

[View recommendations](#)

```
[216] w_s.loc[:, :3]
```

weather	1	2	3
season_name			
Fall	1930	604	199
Spring	1759	715	211
Summer	1801	708	224
Winter	1702	807	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

```
[219] alpha = 0.05  
  
test_statistics, p_value, dof, exp = chi2_contingency(w_s.loc[:, :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
```

From the Chi-Square Test, we can say that weather and season are dependent 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

1. **Seasonal Marketing:** Since there is a clear seasonal pattern in the count of rental bikes, Yulu can adjust its marketing strategies accordingly. Focus on promoting bike rentals during the spring and summer months when there is higher demand. Offer seasonal discounts or special packages to attract more customers during these periods.
2. **Time-based Pricing:** Take advantage of the hourly fluctuation in bike rental counts throughout the day. Consider implementing time-based pricing where rental rates are lower during off-peak hours and higher during peak hours. This can encourage customers to rent bikes during less busy times, balancing out the demand and optimizing the resources.
3. **Weather-based Promotions:** Recognize the impact of weather on bike rentals. Create weather-based promotions that target customers during clear and cloudy weather, as these conditions show the highest rental counts. Yulu can offer weather-specific discounts to attract more customers during these favorable weather conditions.
4. **User Segmentation:** Given that around 81% of users are registered, and the remaining 19% are casual, Yulu can tailor its marketing and communication strategies accordingly. Provide loyalty programs, exclusive offers, or personalized recommendations for registered users to encourage repeat business. For casual users, focus on providing a seamless rental experience and promoting the benefits of bike rentals for occasional use.
5. **Optimize Inventory:** Analyze the demand patterns during different months and adjust the inventory accordingly. During months with lower rental counts such as January, February, and March, Yulu can optimize its inventory levels to avoid excess bikes. On the other hand, during peak months, ensure having sufficient bikes available to meet the higher demand.
6. **Improve Weather Data Collection:** Given the lack of records for extreme weather conditions, consider improving the data collection process for such scenarios. Having more data on extreme weather conditions can help to understand customer behavior and adjust the operations accordingly, such as offering specialized bike models for different weather conditions or implementing safety measures during extreme weather.
7. **Customer Comfort:** Since humidity levels are generally high and temperature is often below 28 degrees Celsius, consider providing amenities like umbrellas, rain jackets, or water bottles to

enhance the comfort and convenience of the customers. These small touches can contribute to a positive customer experience and encourage repeat business.

8. **Collaborations with Weather Services:** Consider collaborating with weather services to provide real-time weather updates and forecasts to potential customers. Incorporate weather information into your marketing campaigns or rental app to showcase the ideal biking conditions and attract users who prefer certain weather conditions.
9. **Seasonal Bike Maintenance:** Allocate resources for seasonal bike maintenance. Before the peak seasons, conduct thorough maintenance checks on the bike fleet to ensure they are in top condition. Regularly inspect and service bikes throughout the year to prevent breakdowns and maximize customer satisfaction.
10. **Customer Feedback and Reviews:** Encourage customers to provide feedback and reviews on their biking experience. Collecting feedback can help identify areas for improvement, understand customer preferences, and tailor the services to better meet customer expectations.
11. **Social Media Marketing:** Leverage social media platforms to promote the electric bike rental services. Share captivating visuals of biking experiences in different weather conditions, highlight customer testimonials, and engage with potential customers through interactive posts and contests. Utilize targeted advertising campaigns to reach specific customer segments and drive more bookings.
12. **Special Occasion Discounts:** Since Yulu focusses on providing a sustainable solution for vehicular pollution, it should give special discounts on the occasions like Zero Emissions Day (21st September), Earth day (22nd April), World Environment Day (5th June) etc. in order to attract new users.