In [ ]:
```
#Importing the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import ttest_ind,chi2_contingency,kruskal,shapiro,levene
from statsmodels.graphics.gofplots import qqplot
```

In [ ]:
```
!pip install pingouin
import pingouin as pg
```

```
Collecting pingouin
  Downloading pingouin-0.5.4-py2.py3-none-any.whl (198 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 198.9/198.9 kB 2.3 MB/s eta 0:00:00
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from pingouin) (1.25.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from pingouin) (1.11.4)
Requirement already satisfied: pandas>=1.5 in /usr/local/lib/python3.10/dist-packages (from pingouin) (2.0.3)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from pingouin) (3.7.1)
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (from pingouin) (0.13.1)
Requirement already satisfied: statsmodels in /usr/local/lib/python3.10/dist-packages (from pingouin) (0.14.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from pingouin) (1.2.2)
Collecting pandas-flavor (from pingouin)
  Downloading pandas_flavor-0.6.0-py3-none-any.whl (7.2 kB)
Requirement already satisfied: tabulate in /usr/local/lib/python3.10/dist-packages (from pingouin) (0.9.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.5->pingouin) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.5->pingouin) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.5->pingouin) (2024.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->pingouin) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->pingouin) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->pingouin) (4.51.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->pingouin) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->pingouin) (24.0)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->pingouin) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->pingouin) (3.1.2)
Requirement already satisfied: xarray in /usr/local/lib/python3.10/dist-packages (from pandas-flavor->pingouin) (2023.7.0)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->pingouin) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->pingouin) (3.5.0)
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.10/dist-packages (from statsmodels->pingouin) (0.5.6)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.6->statsmodels->pingouin) (1.16.0)
Installing collected packages: pandas-flavor, pingouin
Successfully installed pandas-flavor-0.6.0 pingouin-0.5.4
```

In [ ]:

```python
#Importing the dataset
df=pd.read_csv("https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/
df.head()
```

Out[ ]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0

In [ ]:

```
#checking the shape of the dataset
df.shape
```

Out[ ]:

```
(10886, 12)
```

In [ ]:

```
#Checking the datatypes of the variables
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
```

In [ ]:

```
#Checking for null values
df.isnull().sum()
```

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

In [ ]:
```
#checking for unique groups in the columns of the dataset
df.nunique()
```

Out[ ]:
```
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
```

In [ ]:
```
#statistical analysis using describe method
df.describe().T
```

Out[ ]:

	count	mean	std	min	25%	50%	75%	max
season	10886.0	2.506614	1.116174	1.00	2.0000	3.000	4.0000	4.0000
holiday	10886.0	0.028569	0.166599	0.00	0.0000	0.000	0.0000	1.0000
workingday	10886.0	0.680875	0.466159	0.00	0.0000	1.000	1.0000	1.0000
weather	10886.0	1.418427	0.633839	1.00	1.0000	1.000	2.0000	4.0000
temp	10886.0	20.230860	7.791590	0.82	13.9400	20.500	26.2400	41.0000
atemp	10886.0	23.655084	8.474601	0.76	16.6650	24.240	31.0600	45.4550
humidity	10886.0	61.886460	19.245033	0.00	47.0000	62.000	77.0000	100.0000
windspeed	10886.0	12.799395	8.164537	0.00	7.0015	12.998	16.9979	56.9969
casual	10886.0	36.021955	49.960477	0.00	4.0000	17.000	49.0000	367.0000
registered	10886.0	155.552177	151.039033	0.00	36.0000	118.000	222.0000	886.0000
count	10886.0	191.574132	181.144454	1.00	42.0000	145.000	284.0000	977.0000

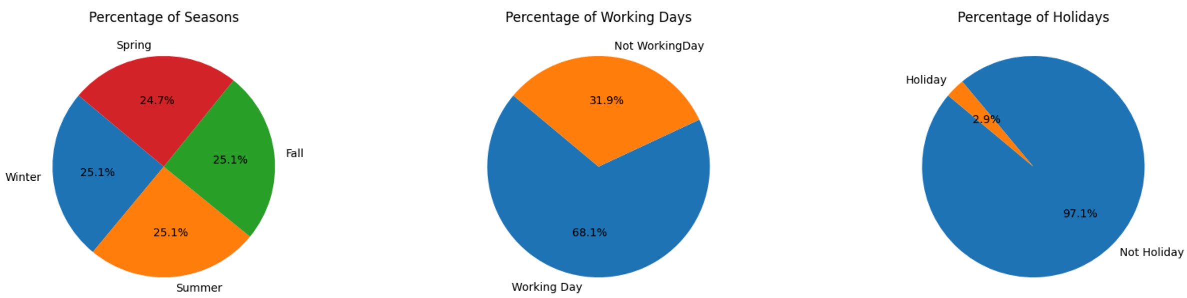
In [ ]:
```
#Checking the duplicate rows
duplicates=df[df.duplicated()]
print(duplicates)
```

```
Empty DataFrame
Columns: [datetime, season, holiday, workingday, weather, temp, atemp, humidity, w
indspeed, casual, registered, count]
Index: []
```

Univariate Analysis

In [ ]:

```
#Pie Plot
# Mapping numerical values to categorical labels
season_mapping = {1: 'Spring', 2: 'Summer', 3: 'Fall', 4: 'Winter'}
df['season_category'] = df['season'].map(season_mapping)
workingday_mapping = {1: 'Working Day', 0:'Not WorkingDay'}
df['workingday_category'] = df['workingday'].map(workingday_mapping)
holiday_mapping = {1: 'Holiday', 0:'Not Holiday'}
df['holiday_category'] = df['holiday'].map(holiday_mapping)
weather_mapping = {1: 'Clear, Few clouds, partly cloudy', 2: 'Mist + Cloudy, Mist +
df['weather_category'] = df['weather'].map(weather_mapping)
# Plotting the pie chart
plt.figure(figsize=(20, 10))
plt.subplot(2,3,1)
df['season_category'].value_counts().plot.pie(autopct='%1.1f%%', startangle=140)
plt.title('Percentage of Seasons')
plt.ylabel('')  # Hides the y-label
plt.subplot(2,3,2)
df['workingday_category'].value_counts().plot.pie(autopct='%1.1f%%', startangle=140
plt.title('Percentage of Working Days')
plt.ylabel('')  # Hides the y-label
plt.subplot(2,3,3)
df['holiday_category'].value_counts().plot.pie(autopct='%1.1f%%', startangle=140)
plt.title('Percentage of Holidays')
plt.ylabel('')  # Hides the y-label
plt.show()
```



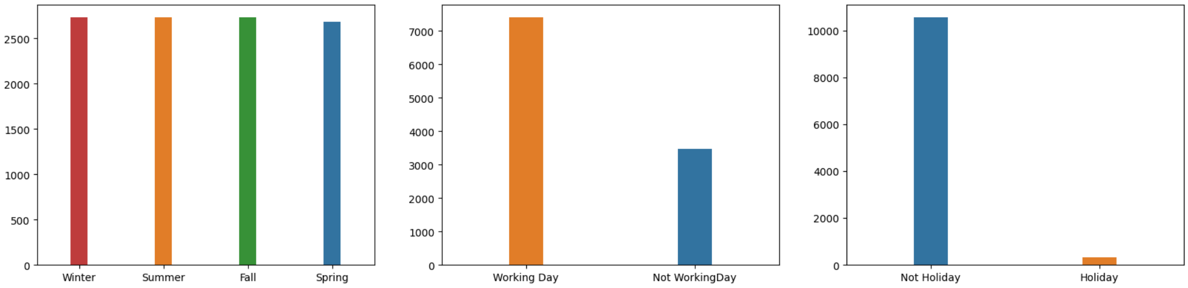
Insight -:

- Bicycle usage is evenly distributed across all four seasons, with each season representing approximately 25% of total usage
- Bicycle usage is significantly higher on working days (68.1%) compared to non-working days (31.9%)
- Bicycle usage is overwhelmingly higher on non-holidays (97.1%) compared to holidays (2.9%).

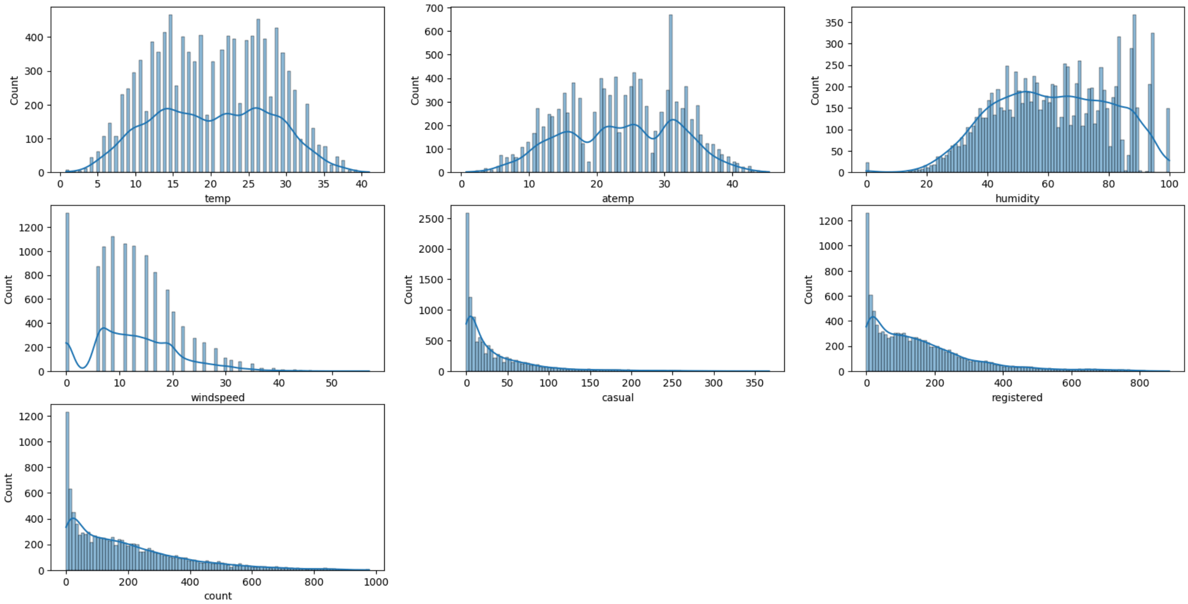
In [ ]:

```
#CountPlot
plt.figure(figsize=(20,10))
plt.subplot(2,3,1)
sns.countplot(x=df["season_category"],order=df["season_category"].value_counts().in
plt.xlabel('')
plt.ylabel('')
plt.subplot(2,3,2)
```

```
sns.countplot(x=df["workingday_category"],order=df["workingday_category"].value_cou
plt.xlabel('')
plt.ylabel('')
plt.subplot(2,3,3)
sns.countplot(x=df["holiday_category"],order=df["holiday_category"].value_counts().
plt.xlabel('')
plt.ylabel('')
plt.show()
```



In [ ]:
```
#HistPlot
plt.figure(figsize=(20,10))
plt.subplot(3,3,1)
sns.histplot(df["temp"],bins=100,kde=True)
plt.subplot(3,3,2)
sns.histplot(df["atemp"],bins=100,kde=True)
plt.subplot(3,3,3)
sns.histplot(df["humidity"],bins=100,kde=True)
plt.subplot(3,3,4)
sns.histplot(df["windspeed"],bins=100,kde=True)
plt.subplot(3,3,5)
sns.histplot(df["casual"],bins=100,kde=True)
plt.subplot(3,3,6)
sns.histplot(df["registered"],bins=100,kde=True)
plt.subplot(3,3,7)
sns.histplot(df["count"],bins=100,kde=True)
plt.show()
```

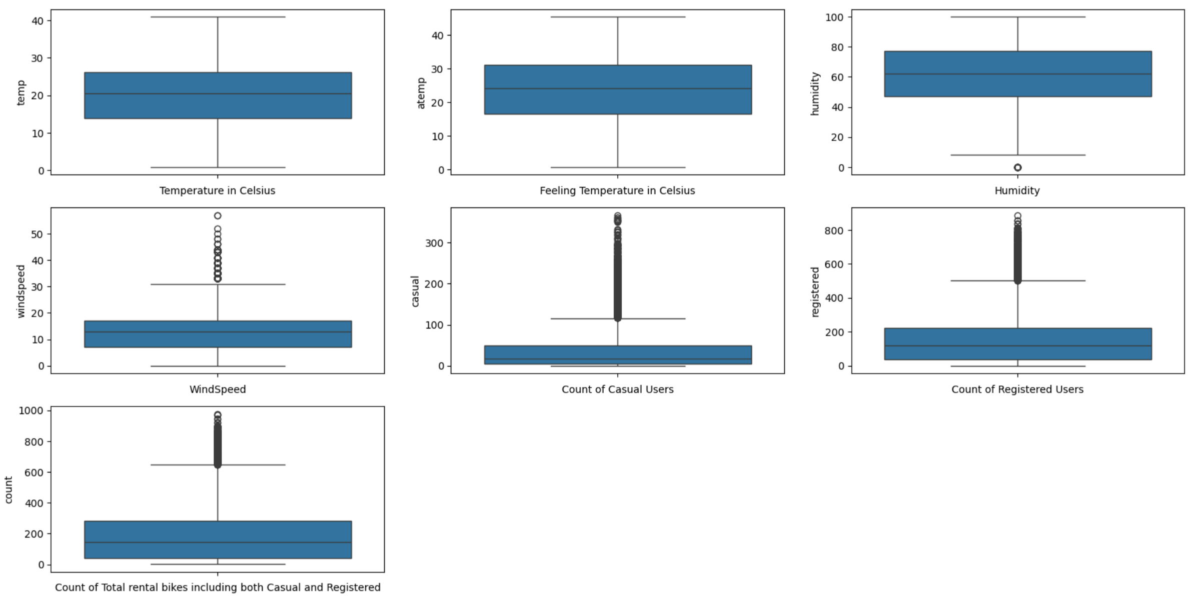


Insight -:

- Most of the data points lie between 10 and 30 degrees Celsius, indicating this is the common temperature range.
- The feels-like temperature closely follows the actual temperature, which is expected.

- High humidity levels are common, with a significant number of data points close to 100%.
- Low wind speeds are very common, with a majority of observations below 20 units.
- Most casual user counts are low, indicating occasional use by casual users.
- There is a broader range of registered user counts, but low counts are still common.
- Low total user counts are common, but there is a long tail indicating occasional high usage days.

In [ ]:
```
#BoxPlot
plt.figure(figsize=(20,10))
plt.subplot(3,3,1)
sns.boxplot(df["temp"])
plt.xlabel("Temperature in Celsius")
plt.subplot(3,3,2)
sns.boxplot(df["atemp"])
plt.xlabel("Feeling Temperature in Celsius")
plt.subplot(3,3,3)
sns.boxplot(df["humidity"])
plt.xlabel("Humidity")
plt.subplot(3,3,4)
sns.boxplot(df["windspeed"])
plt.xlabel("WindSpeed")
plt.subplot(3,3,5)
sns.boxplot(df["casual"])
plt.xlabel("Count of Casual Users")
plt.subplot(3,3,6)
sns.boxplot(df["registered"])
plt.xlabel("Count of Registered Users")
plt.subplot(3,3,7)
sns.boxplot(df["count"])
plt.xlabel("Count of Total rental bikes including both Casual and Registered")
plt.show()
```



In [ ]:
```
# List of numerical columns to check for outliers
numerical_columns = ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registere
for column in numerical_columns:
  def calculate_iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
```

```
    upper_bound = Q3 + 1.5 * IQR
    return lower_bound, upper_bound
```

In [ ]:
```
# Remove outliers from the DataFrame
for column in numerical_columns:
    lower_bound, upper_bound = calculate_iqr(df, column)
    df = df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]

# Check the shape of the dataset after removal
print('Dataset shape after removing outliers:', df.shape)
```

```
Dataset shape after removing outliers: (9364, 16)
```

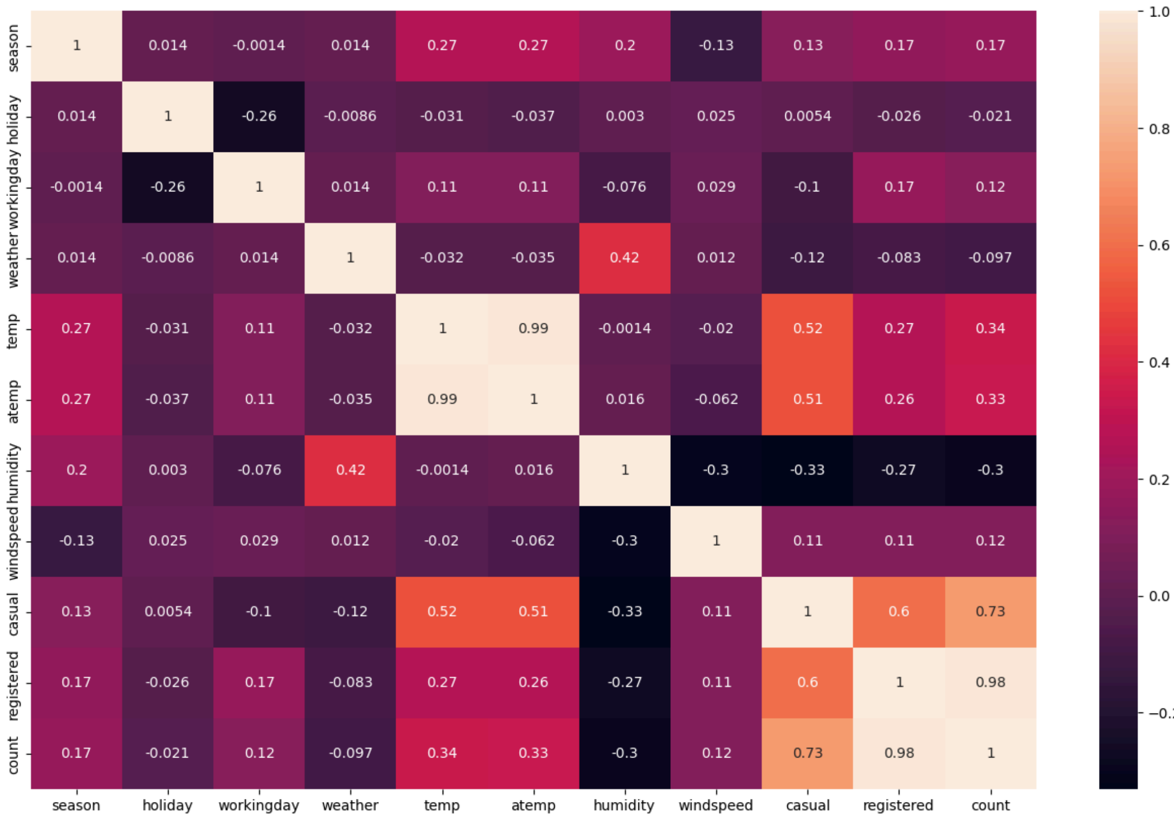
In [ ]:
```
# Clip outliers for each numerical column
for column in numerical_columns:
    lower_bound, upper_bound = calculate_iqr(df, column)
    df[column] = df[column].clip(lower=lower_bound, upper=upper_bound)

# Check the shape of the dataset after clipping
print('Dataset shape after clipping outliers:', df.shape)
```

```
Dataset shape after clipping outliers: (9364, 16)
```

In [ ]:
```
#Shape after removing Outliers
df.shape
```

Out[ ]:
```
(9364, 16)
```

In [ ]:
```
#Heatmap
plt.figure(figsize=(16,10))
sns.heatmap(data=df.corr(numeric_only=True),annot=True)
plt.show()
```



	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
season	1	0.014	-0.0014	0.014	0.27	0.27	0.2	-0.13	0.13	0.17	0.17
holiday	0.014	1	-0.26	-0.0086	-0.031	-0.037	0.003	0.025	0.0054	-0.026	-0.021
workingday	-0.0014	-0.26	1	0.014	0.11	0.11	-0.076	0.029	-0.1	0.17	0.12
weather	0.014	-0.0086	0.014	1	-0.032	-0.035	0.42	0.012	-0.12	-0.083	-0.097
temp	0.27	-0.031	0.11	-0.032	1	0.99	-0.0014	-0.02	0.52	0.27	0.34
atemp	0.27	-0.037	0.11	-0.035	0.99	1	0.016	-0.062	0.51	0.26	0.33
humidity	0.2	0.003	-0.076	0.42	-0.0014	0.016	1	-0.3	-0.33	-0.27	-0.3
windspeed	-0.13	0.025	0.029	0.012	-0.02	-0.062	-0.3	1	0.11	0.11	0.12
casual	0.13	0.0054	-0.1	-0.12	0.52	0.51	-0.33	0.11	1	0.6	0.73
registered	0.17	-0.026	0.17	-0.083	0.27	0.26	-0.27	0.11	0.6	1	0.98
count	0.17	-0.021	0.12	-0.097	0.34	0.33	-0.3	0.12	0.73	0.98	1

In [ ]:
```
corr_matrix=df.corr(numeric_only=True)
```

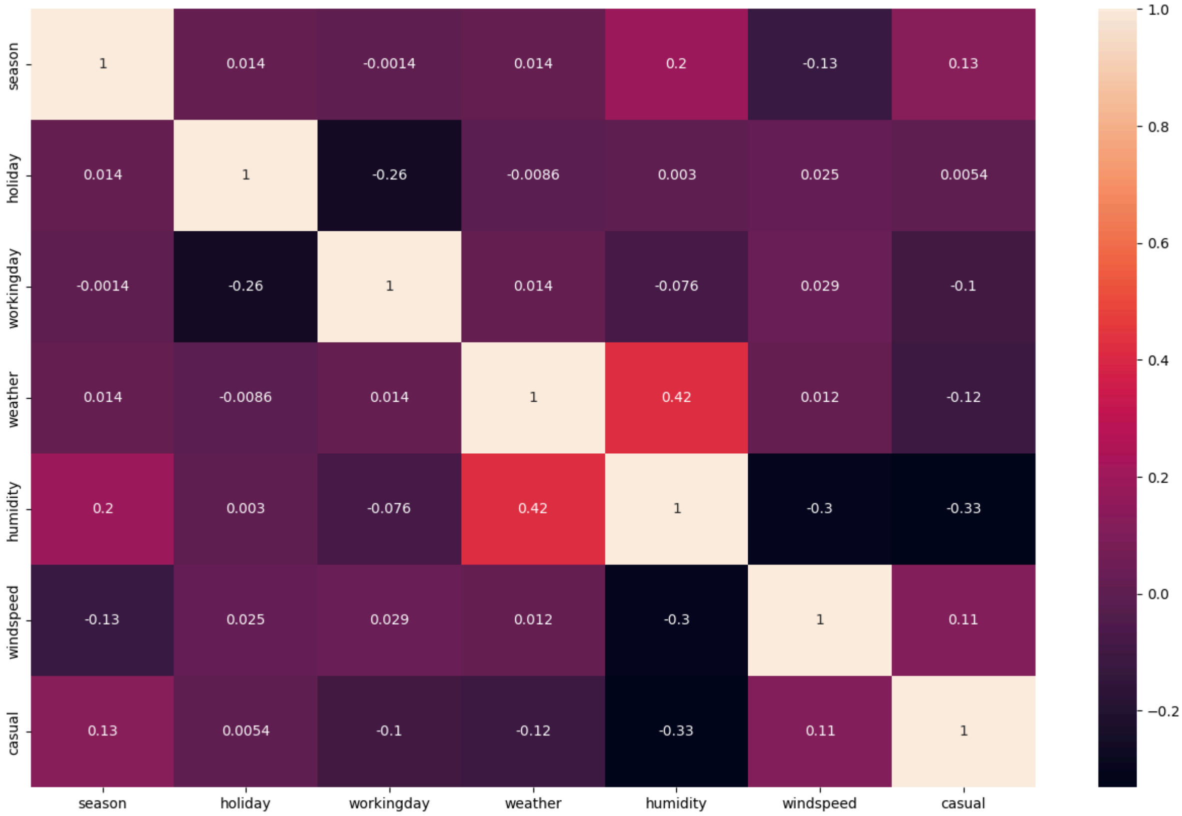
In [ ]:
```
# Identify highly correlated pairs (correlation coefficient > 0.8 or < -0.8)
high_corr = corr_matrix.abs().unstack().sort_values(kind="quicksort", ascending=Fal
```

```
high_corr = high_corr[high_corr != 1]  # Remove self-correlation

# Display highly correlated pairs
print(high_corr[high_corr > 0.8])
```

```
atemp       temp          0.986274
temp        atemp         0.986274
registered  count         0.984985
count       registered    0.984985
dtype: float64
```

In [ ]:
```
# Let's assume 'temp' and 'atemp' are highly correlated and you decide to remove th
df_reduced = df.drop(columns=['atemp','temp','registered','count'])
```

In [ ]:
```
#Heatmap after removing highly Correlated Variables
plt.figure(figsize=(16,10))
sns.heatmap(data=df_reduced.corr(numeric_only=True),annot=True)
plt.show()
```

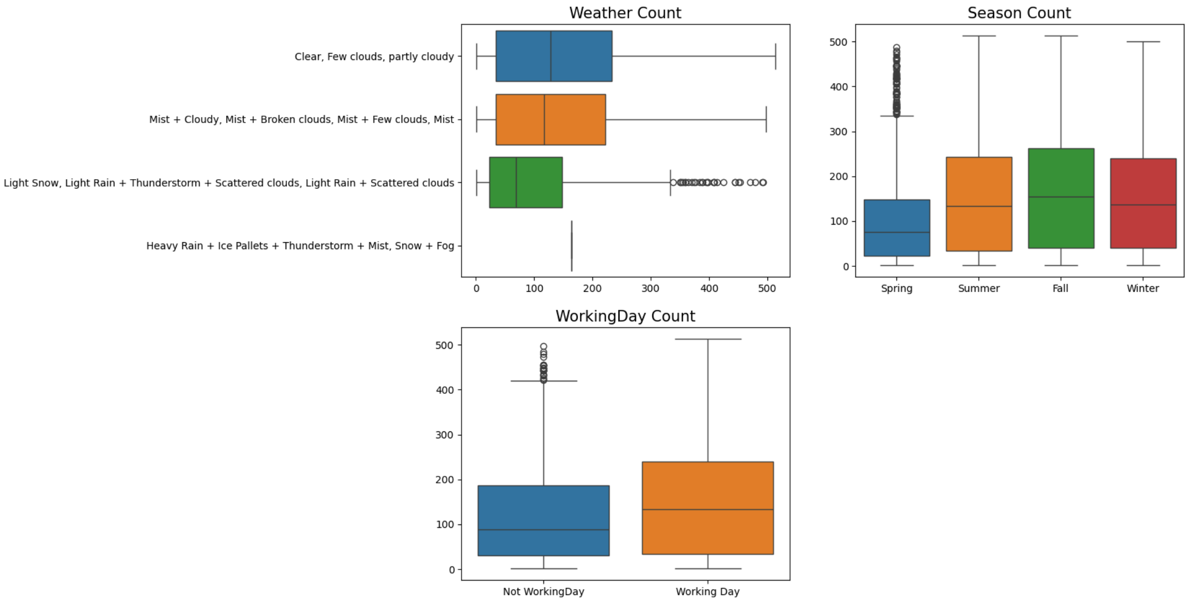


Insight -

- Season has a noticeable impact on humidity and casual users, indicating that different seasons likely bring variations in humidity and casual bike rentals.
- There's a clear distinction between holidays and working days, with a negative correlation, meaning that holidays and working days are mutually exclusive.
- Weather and Humidity: These two variables are strongly positively correlated, suggesting that bad weather conditions are often associated with higher humidity.
- Casual Users: Casual user counts decrease on working days and increase with windspeed. Higher humidity tends to decrease casual users.

Bivariate Analysis

In [ ]:
```
#BoxPlot
plt.figure(figsize=(20,10))
plt.subplot(2,3,1)
sns.boxplot(y='weather_category', x='count',hue='weather_category', data=df)
plt.title('Weather Count', fontsize=15)
plt.xlabel('')
plt.ylabel('')
plt.subplot(2,3,2)
sns.boxplot(x='season_category', y='count',hue='season_category', data=df)
plt.title('Season Count', fontsize=15)
plt.xlabel('')
plt.ylabel('')
plt.subplot(2,3,4)
sns.boxplot(x='workingday_category', y='count',hue='workingday_category', data=df)
plt.title('WorkingDay Count', fontsize=15)
plt.xlabel('')
plt.ylabel('')
plt.show()
```

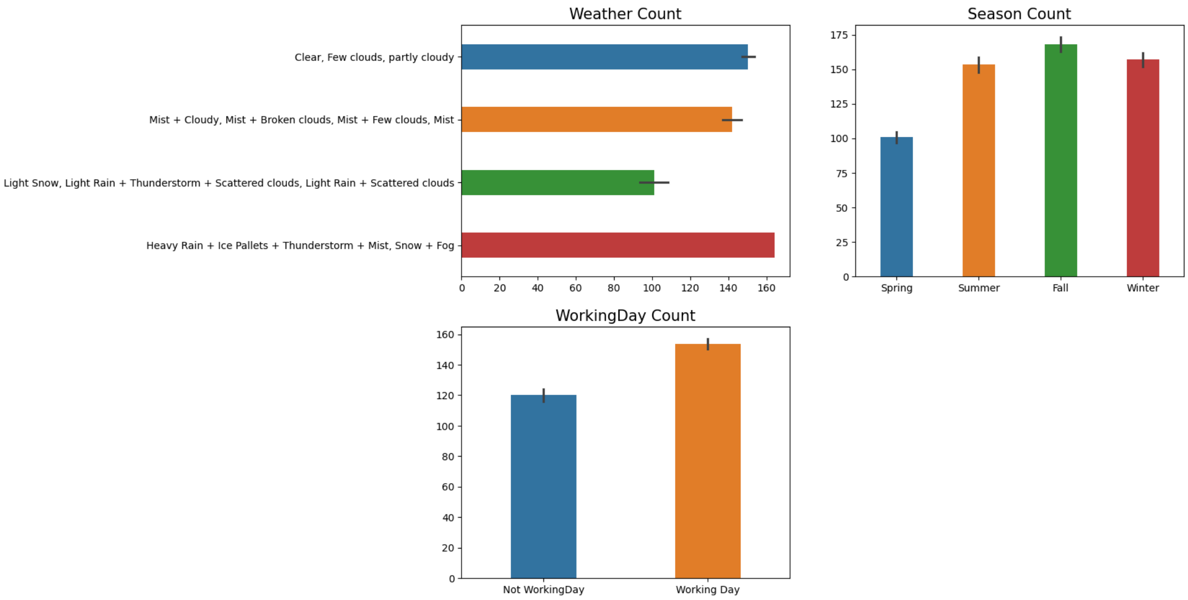


Insight :

- When we analyse weather and count columns we see that the count of cycles rented is different for various weather conditions.
- For season and count columns the count of cycles rented are also different for different seasons.

In [ ]:
```
#BarPlot
plt.figure(figsize=(20,10))
plt.subplot(2,3,1)
sns.barplot(y='weather_category', x='count',hue='weather_category', data=df,width=0
plt.title('Weather Count', fontsize=15)
plt.xlabel('')
plt.ylabel('')
plt.subplot(2,3,2)
sns.barplot(x='season_category', y='count',hue='season_category', data=df,width=0.4
plt.title('Season Count', fontsize=15)
plt.xlabel('')
plt.ylabel('')
plt.subplot(2,3,4)
sns.barplot(x='workingday_category', y='count',hue='workingday_category', data=df,w
plt.title('WorkingDay Count', fontsize=15)
plt.xlabel('')
```

```
plt.ylabel('')
plt.show()
```



In [ ]:
```
#Converting datatype of datetime column using pandas
df["datetime"]=pd.to_datetime(df["datetime"])
```

In [ ]:
```
df["dayofweek"]=df["datetime"].dt.dayofweek
```

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

STEP 1:

What should be the null and alternate hypothesis?

Null Hypothesis (H0):

There is no significant difference between the no. of bike rides on Weekdays and Weekends.

Alternative Hypothesis (H1):

There is a significant difference between the no. of bike rides on Weekdays and Weekends.

STEP 2:

> #### What is the Test it follows?

- Two-Sided

STEP 3:

We perform Two Sample ttest independent and calculate the P-Value

In [ ]:
```
df_weekdays=df[df["dayofweek"]<5]["count"]
df_weekends=df[df["dayofweek"]>=5]["count"]
```

In [ ]:
```
t_stats, pvalue = ttest_ind(df_weekdays, df_weekends, alternative ='two-sided')
print(f"P-Value is {pvalue}")
```

```
P-Value is 2.2048306477213035e-31
```

STEP 4:

We defined $\alpha = 0.05$ for confidence level 95%

In [ ]:
```
alpha=0.05
if pvalue<alpha:
  print("We reject H0")
  print("There is a significant difference between the no. of bike rides on Weekday
else:
  print("We fail to reject H0")
  print("There is no significant difference between the no. of bike rides on Weekda
```

```
We reject H0
There is a significant difference between the no. of bike rides on Weekdays and We
ekends
```

Insight :

- There is a difference in number of cycles rented on Weekdays and Weekends.
- Most Customers preferred bicycles on weekdays than weekends may be due to office work on weekdays.

2-- Check if the demand of bicycles on rent is the same for different Weather conditions?

STEP 1:

What should be the null and alternate hypothesis?

Null Hypothesis (H0):

There is no significant difference between the demand of bicycles on rent for different weather conditions.

Alternative Hypothesis (H1):

There is a significant difference between the demand of bicycles on rent for different weather conditions.

STEP 2:

> **What is the Distribution it follows?**

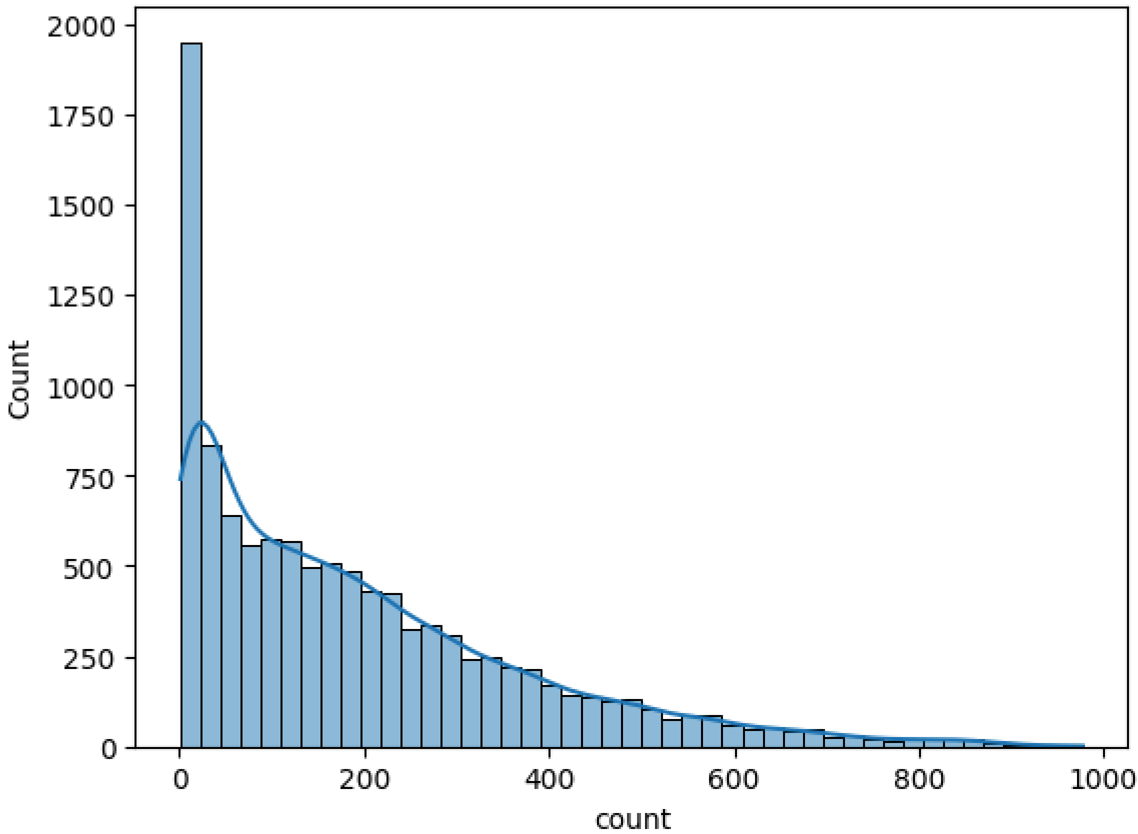
- One-Way Anova

STEP 3:

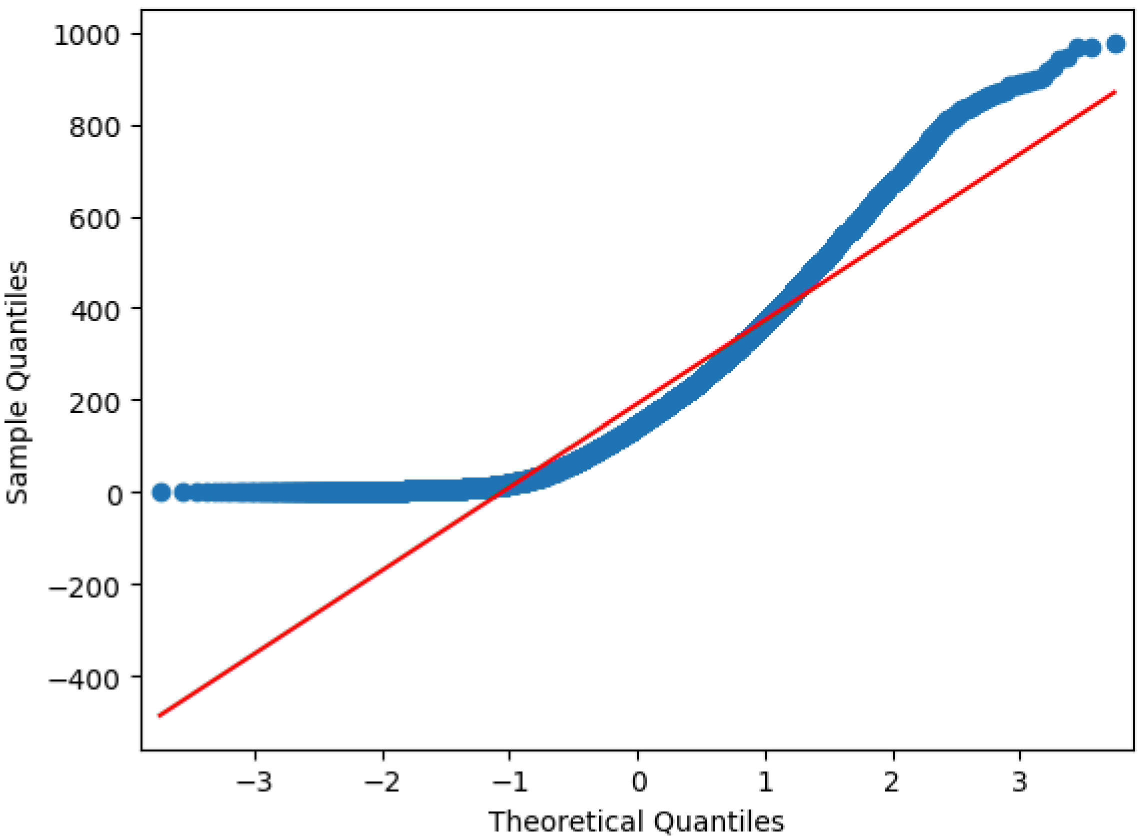
Checking assumptions of the test

1) Normality-

In [ ]:
```
sns.histplot(df["count"],kde=True)
plt.show()
```



In [ ]:
```
qqplot(df["count"], line="s")
plt.show()
```



In [ ]:
```
Skewness=df["count"].skew()
print(f"Skewness : {Skewness}")
```

```
Skewness : 0.7752150951429889
```

In [ ]:
```
Kurtosis=df["count"].kurt()
print(f"Kurtosis : {Kurtosis}")
```

```
Kurtosis : -0.23573570229188157
```

Shapiro Test

In [ ]:
```
# H0: Data is Gaussian
# Ha: Data is not Gaussian
test_stat, pvalue = shapiro(df["count"].head(2000))
print(f"P-Value is {pvalue}")
```

```
P-Value is 2.614797430798058e-38
```

In [ ]:
```
alpha=0.05
if pvalue < alpha:
    print("Reject H0")
    print("Data is not Gaussian")
else:
    print("Fail to reject H0")
    print("Data is Gaussian")
```

```
Reject H0
Data is not Gaussian
```

Levene Test

In [ ]:
```
weather1 = df[df["weather"]==1]["count"]
weather2 = df[df["weather"]==2]["count"]
weather3 = df[df["weather"]==3]["count"]
weather4 = df[df["weather"]==4]["count"]
```

In [ ]:
```
# H0: Variances are equal
# Ha: Variances are not equal
levene_stat1, pvalue = levene(weather1, weather2,weather3,weather4)
if pvalue < 0.05:
  print("Variances are not equal")
else:
  print("Variances are equal")
print(f"P-Value is {pvalue}")
```

```
Variances are not equal
P-Value is 1.842537422004331e-25
```

--- Since,We can see that data does not follow Assumptions of One Way ANOVA, We will need to perform Kruskal-Wallis Test in Order to make Conclusions

Kruskal-Wallis Test

In [ ]:
```
# Null Hypothesis (H0): There is no significant difference between the demand of bi
# Alternative Hypothesis (H1): There is a significant difference between the demand
stat, pvalue = kruskal(weather1,weather2,weather3,weather4)
print("test statistic:",stat)
print("P-Value:",pvalue)
if pvalue < 0.05:
   print("Reject H0")
   print("There is a significant difference between the demand of bicycles on rent
else:
   print("Fail to reject H0")
   print("There is no significant difference between the demand of bicycles on rent
```

```
test statistic: 104.06612180303777
P-Value: 2.0750961454445037e-22
Reject H0
There is a significant difference between the demand of bicycles on rent for diffe
rent weather conditions
```

Insight :

- Bicycle rental demand can fluctuate based on weather conditions, with higher demand during favorable weather and lower demand during adverse conditions.
- The demand for bicycles may vary not just due to weather but also because of other factors like local events or changes in consumer preferences.

3-- Check if the demand of bicycles on rent is the same for different Seasons?

STEP 1:

What should be the null and alternate hypothesis?

Null Hypothesis (H0):

There is no significant difference between the demand of bicycles on rent for different seasons.

Alternative Hypothesis (H1):

There is a significant difference between the demand of bicycles on rent for different seasons.

STEP 2:

> **What is the Distribution it follows?**

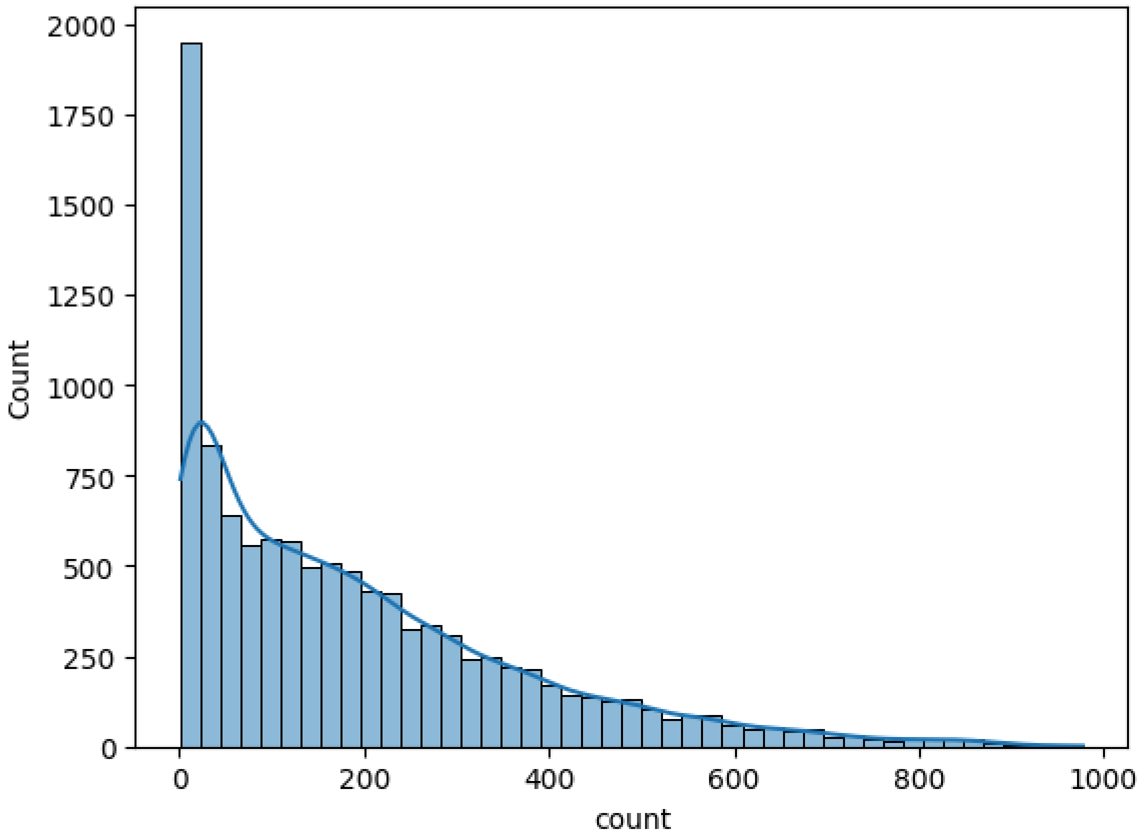
- One-Way Anova

STEP 3:

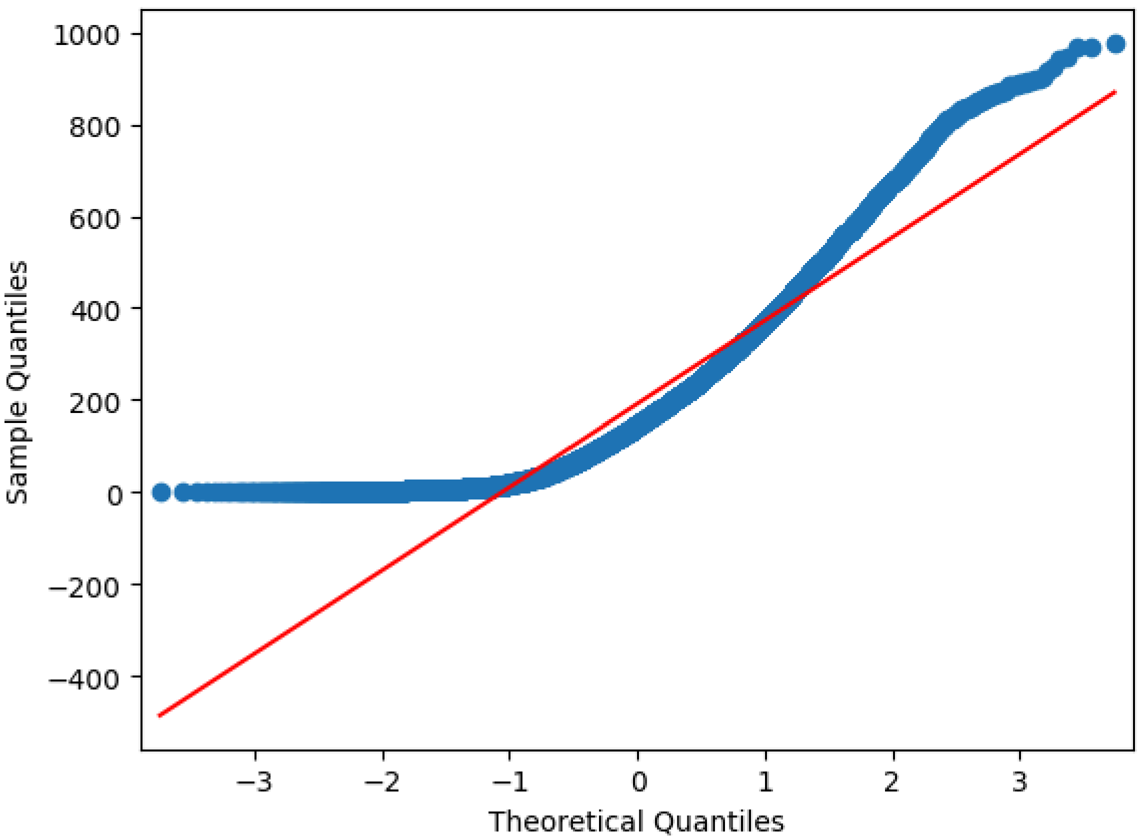
Checking assumptions of the test

1) Normality-

In [ ]:
```
sns.histplot(df["count"],kde=True)
plt.show()
```



In [ ]:
```
qqplot(df["count"], line="s")
plt.show()
```



In [ ]:
```
Skewness=df["count"].skew()
print(f"Skewness : {Skewness}")
```

```
Skewness : 0.7752150951429889
```

In [ ]:
```
Kurtosis=df["count"].kurt()
print(f"Kurtosis : {Kurtosis}")
```

```
Kurtosis : -0.23573570229188157
```

Shapiro Test

In [ ]:
```
# H0: Data is Gaussian
# Ha: Data is not Gaussian
test_stat, pvalue = shapiro(df["count"].head(2000))
print(f"P-Value is {pvalue}")
```

```
P-Value is 7.549915866089249e-41
```

In [ ]:
```
alpha=0.05
if pvalue < alpha:
    print("Reject H0")
    print("Data is not Gaussian")
else:
    print("Fail to reject H0")
    print("Data is Gaussian")
```

```
Reject H0
Data is not Gaussian
```

Levene Test

In [ ]:
```
season1 = df[df["season"]==1]["count"]
season2 = df[df["season"]==2]["count"]
season3 = df[df["season"]==3]["count"]
season4 = df[df["season"]==4]["count"]
```

In [ ]:
```
# H0: Variances are equal
# Ha: Variances are not equal
levene_stat1, pvalue = levene(season1,season2,season3,season4)
if pvalue < 0.05:
  print("Variances are not equal")
else:
  print("Variances are equal")
print(f"P-Value is {pvalue}")
```

```
Variances are not equal
P-Value is 1.4155817904060813e-85
```

--- Since,We can see that data does not follow Assumptions of One Way ANOVA, We will need to perform Kruskal-Wallis Test in Order to make Conclusions

Kruskal-Wallis Test

Null Hypothesis (H0):

There is no significant difference between the demand of bicycles on rent for different seasons.

Alternative Hypothesis (H1):

There is a significant difference between the demand of bicycles on rent for different seasons.

In [ ]:
```
stat, pvalue = kruskal(season1,season2,season3,season4)
print("test statistic:",stat)
print("P-Value:",pvalue)
if pvalue < 0.05:
   print("We Reject H0")
   print("There is a significant difference between the demand of bicycles on rent
else:
   print("We Fail to reject H0")
   print("There is no significant difference between the demand of bicycles on rent
```

```
test statistic: 402.4201174781308
P-Value: 6.621202982238916e-87
We Reject H0
There is a significant difference between the demand of bicycles on rent for diffe
rent seasons
```

Insight :

- Bicycle rental demand typically varies across seasons, with peak and off-peak periods influenced by weather and activity patterns.
- Demand for bicycle rentals can experience fluctuations not solely tied to seasons, but also influenced by events, promotions, or economic factors.

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

STEP 1:

What should be the null and alternate hypothesis?

Null Hypothesis (H0):

There is no significant difference between weather conditions during different seasons.

Alternative Hypothesis (H1):

There is a significant difference between weather conditions during different seasons.

STEP 2:

> **What is the Distribution it follows?**

- Chi-Squared Test

In [ ]:
```
df_chiq=pd.crosstab(df["weather"],df["season"])
df_chiq
```

Out[ ]:

season	1	2	3	4
weather				
1	1759	1801	1930	1702
2	715	708	604	807
3	211	224	199	225
4	1	0	0	0

STEP 3:

We perform Chi-Squared Test and calculate the P-Value

In [ ]:
```
stat,pvalue,dof,exp_freq=chi2_contingency(df_chiq)
print(f"P-Value is {pvalue}")
```

```
P-Value is 1.5499250736864862e-07
```

STEP 4:

We defined $\alpha = 0.05$ for confidence level 95%

In [ ]:
```
alpha=0.05
if pvalue<alpha:
  print("We reject H0")
  print("There is a significant difference between weather conditions during differ
else:
  print("We fail to reject H0")
  print("There is no significant difference between weather conditions during diffe
```

```
We reject H0
There is a significant difference between weather conditions during different seas
ons
```

Insight :

- Weather can sometimes remain consistent across seasons due to local factors or the impact of climate change, leading to deviations from typical seasonal patterns.

Recommendations :

- Prioritize services and features tailored to the needs of Registered users, but continue to monitor and address the needs of Casual users to increase overall user base.
- Adjust bicycle availability based on weather forecasts to optimize resource allocation.
- Stock a higher number of bicycles during clear and cloudy weather conditions to meet increased demand during these times.
- Ensure an adequate supply of bicycles during weekdays, as many office workers rely on them for commuting.
- Launch targeted ad campaigns to boost bicycle rentals during weekends and holidays.