Yulu

Column Profiling:

● datetime: datetime
● season: season (1: spring, 2: summer, 3: fall, 4: winter)
● holiday : whether day is a holiday or not
● workingday : if day is neither weekend nor holiday is 1, otherwise is 0.
● weather:
  o 1: Clear, Few clouds, partly cloudy
  o 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
  o 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
  o 4: Heavy Rain + Ice Pellets + 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

The company wants to know:

● Which variables are significant in predicting the demand for shared electric cycles in the Indian market?
● How well those variables describe the electric cycle demands.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import chi2_contingency
from statsmodels.graphics.gofplots import qqplot
from scipy.stats import skew
from scipy.stats import shapiro
from scipy.stats import f_oneway
from scipy.stats import levene
from scipy.stats import Kruskal

data =
pd.read_csv('https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/original/bike
_sharing.csv?1642089089')
```

data.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

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

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

i. Hint: You can use .shape, .info(), .describe()

```
data.shape
```

```
(10886, 12)
```

```
data.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
```

```
data.describe()
```

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.886460	12.799395	36.021955	155.552177	191.574132
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.245033	8.164537	49.960477	151.039033	181.144454
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	47.000000	7.001500	4.000000	36.000000	42.000000
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	62.000000	12.998000	17.000000	118.000000	145.000000
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	77.000000	16.997900	49.000000	222.000000	284.000000
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	100.000000	56.996900	367.000000	886.000000	977.000000

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

i. Hint: You can use .isnull() or .isna()

```
data.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
```

c. Identify and remove duplicate records.

i. Hint: You can use .duplicated()

```
data.duplicated().sum()
```

```
0
```

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

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

data = data.astype({
    'season':'object',
    'holiday':'object',
    'workingday':'object',
    'weather':'object'
})

numer = data.select_dtypes(include = np.number)
numer.head()
```

	temp	atemp	humidity	windspeed	casual	registered	count
0	9.84	14.395	81	0.0	3	13	16
1	9.02	13.635	80	0.0	8	32	40
2	9.02	13.635	80	0.0	5	27	32
3	9.84	14.395	75	0.0	3	10	13
4	9.84	14.395	75	0.0	0	1	1

cat = data.select_dtypes(include = object)
cat.head()

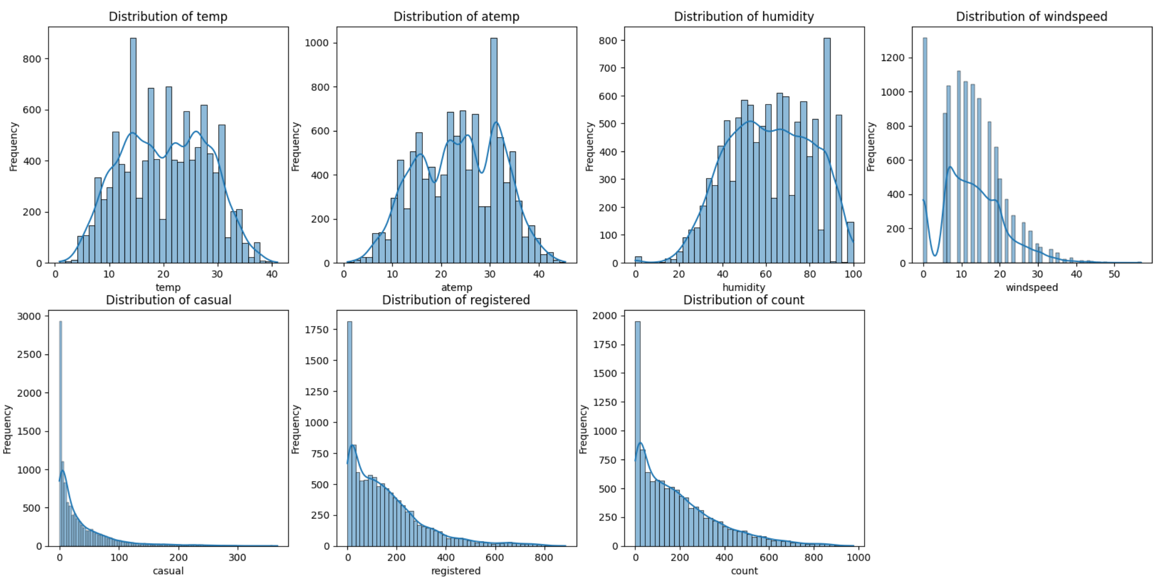
	season	holiday	workingday	weather
0	1	0	0	1
1	1	0	0	1
2	1	0	0	1
3	1	0	0	1
4	1	0	0	1

```
data.dtypes
```

```
datetime      datetime64[ns]
season                object
holiday               object
workingday            object
weather               object
temp                 float64
atemp                float64
humidity               int64
windspeed            float64
casual                 int64
registered             int64
count                  int64
dtype: object
```

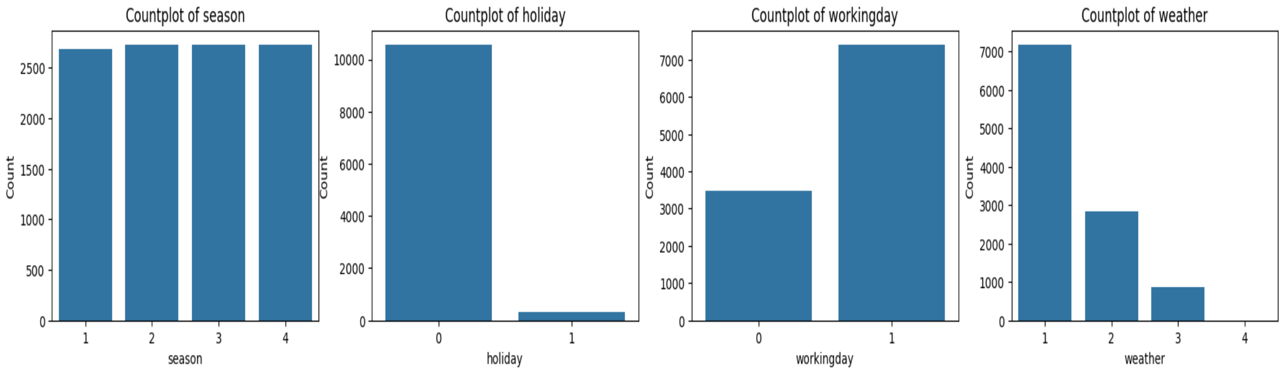
i. For Numerical features use Histogram, Distplot, etc.

```
fig = plt.figure(figsize = (20, 20))
for i,feature in enumerate(numer):
    plt.subplot(4,4,i+1)
    sns.histplot(data[feature], kde=True)
    plt.title(f'Distribution of {feature}')
    plt.xlabel(feature)
    plt.ylabel('Frequency')
plt.show()
```



ii. Hint: For Categorical features use Countplot, Pie Chart, etc.

```
fig = plt.figure(figsize = (22,16))
for i,feature in enumerate(cat):
  plt.subplot(4,4,i+1)
  sns.countplot(x=data[feature])
  plt.title(f'Countplot of {feature}')
  plt.xlabel(feature)
  plt.ylabel('Count')
plt.show()
```

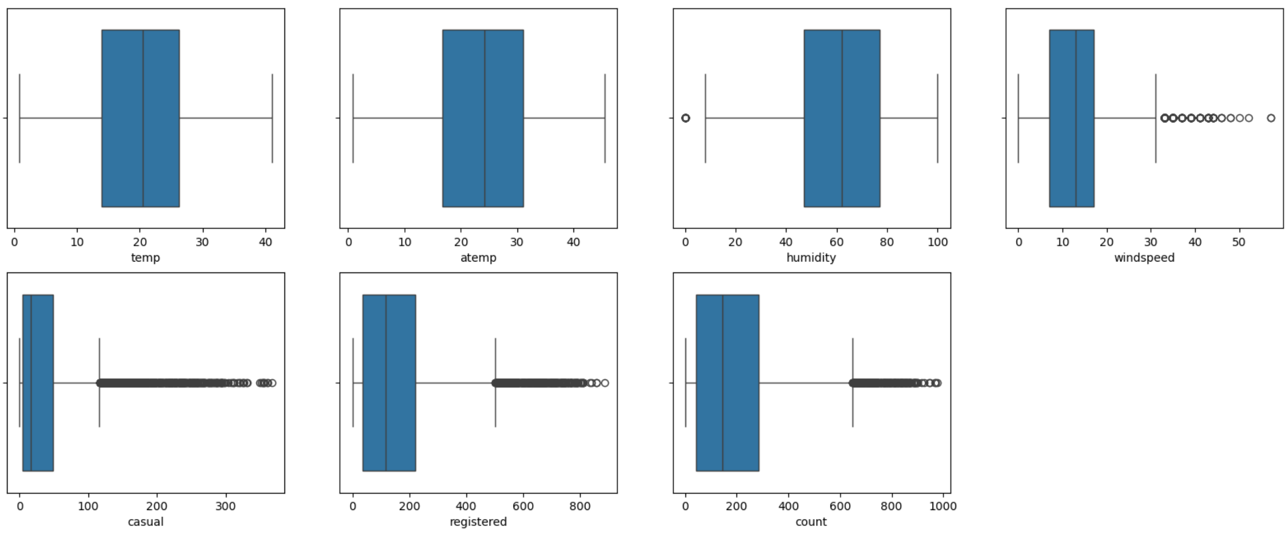


e. Check for Outliers and deal with them accordingly.

Hint:

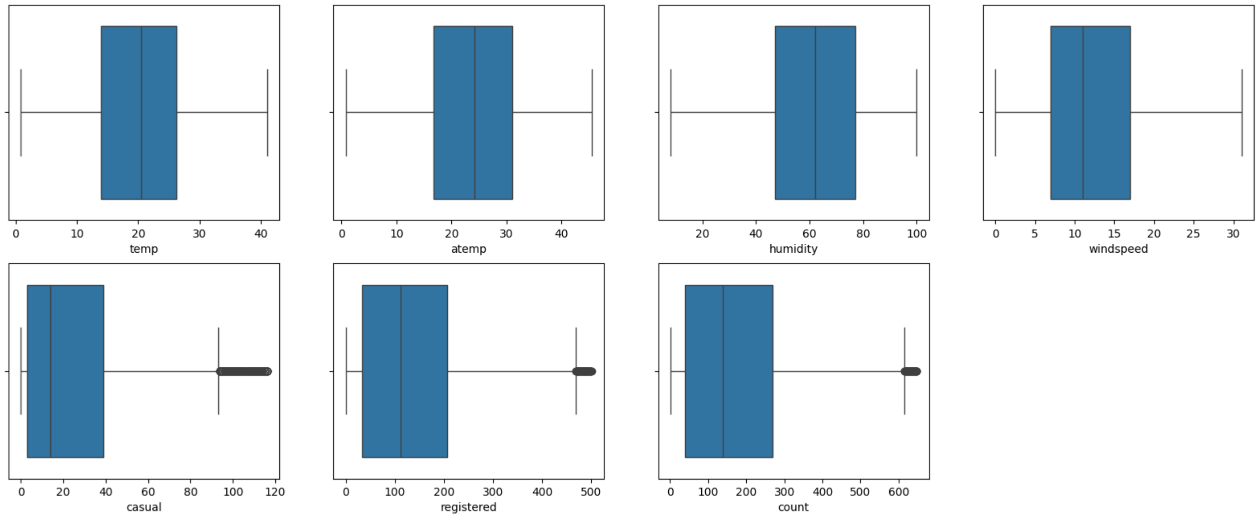
i. You can use Boxplot, Interquartile Range (IQR)

```
fig = plt.figure(figsize=(20,16))
for i,col in enumerate(numer):
  plt.subplot(4,4,i+1)
  sns.boxplot(x = numer[col])
plt.show()
```



ii. Remove/Clip existing outliers as necessary.

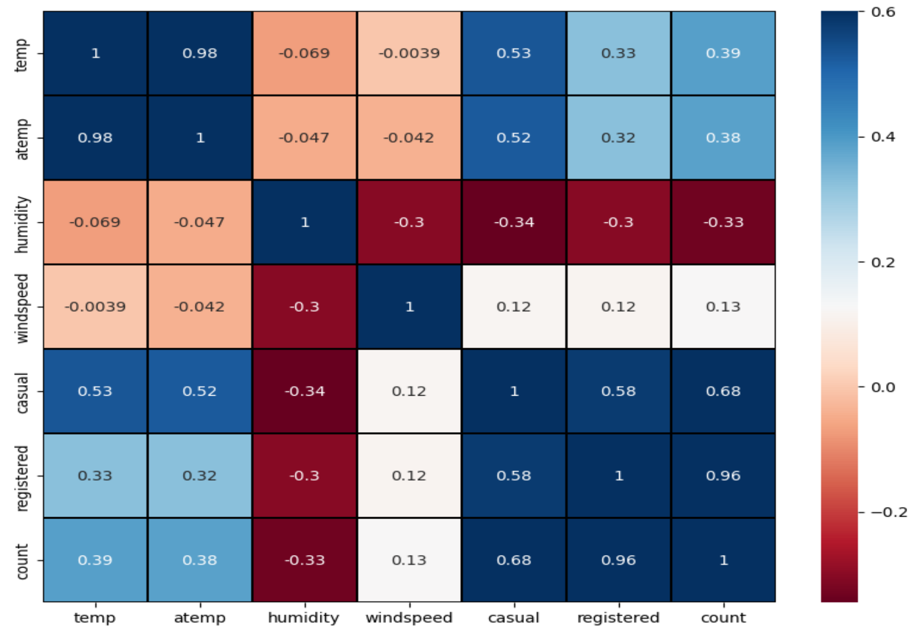
```
Q1 = numer.quantile(0.25)
Q3 = numer.quantile(0.75)
IQR = Q3-Q1
print(IQR)
numer_iqr = numer[~((numer < (Q1 - 1.5 * IQR))|(numer > (Q3 + 1.5 * IQR)))]
```



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

i. Plot a Correlation Heatmap and draw insights.

```
plt.figure(figsize=(10,8))
corre = numer_iqr.corr()
sns.heatmap(corre, annot = True,linecolor="black", linewidths=0.01,cmap= 'RdBu',square=True, vmax = .6)
plt.show()
```



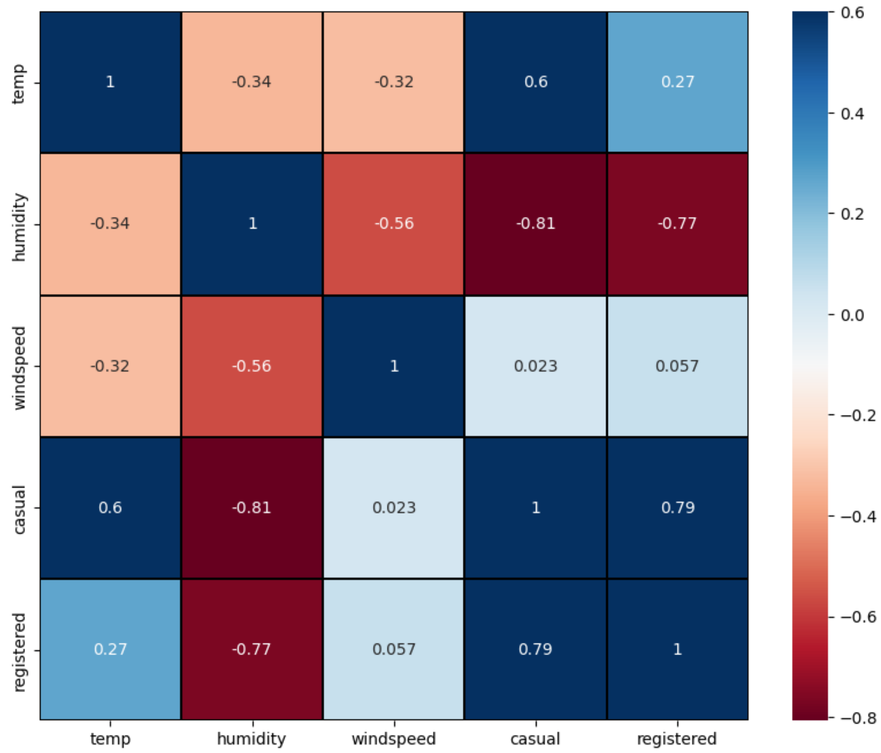
ii. Remove the highly correlated variables, if any

```
threshold = 0.8
high_corr_value = set()
for i in range(len(corre.columns)):
  for j in range(i):
    if abs(corre.iloc[i,j]) > threshold:
      colname_i = corre.columns[i]
      colname_j = corre.columns[j]
      high_corr_value.add((colname_i,colname_j))
print(high_corr_value)
```

```
{('atemp', 'temp'), ('count', 'registered')}
```

```
col_drop = set()
for col1,col2 in high_corr_value:
  col_drop.add(col1)
data_red = corre.drop(columns = col_drop)
```

```
plt.figure(figsize=(10,8))
corre = data_red.corr()
sns.heatmap(corre, annot = True,linecolor="black", linewidths=0.01,cmap= 'RdBu',square=True, vmax
= .6)
plt.show()
```



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

Null Hypothesis (H0):
There is no significant difference between the number of bike rides on weekdays and weekends

Alternative Hypothesis (H1) :
There is significant difference between the number of bike rides on weekdays and weekends

```
from scipy.stats import ttest_ind
alpha = 0.05
t_stat,p_val =
ttest_ind(data[data['workingday']==0]['count'],data[data['workingday']==1]['count'])
print(f'Test Statistics: {t_stat}, P_value: {p_val}')

if p_val <= alpha:
  print('Reject Null Hypothesis : There is significant difference between the number of bike rides
on weekdays and weekends')
else:
  print('Accept Null Hypothesis : There is no significant difference between the number of bike
rides on weekdays and weekends')
```

```
Test Statistics: -1.2096277376026694, P_value: 0.22644804226361348
Accept Null Hypothesis : There is no significant difference between the number of bike rides on weekdays and weekends
```

There is no statistically significant difference between the number of bike rides on weekdays and weekends. This means that the average number of bike rides on weekdays is not significantly different from the average number of bike rides on weekends
Yulu should try promoting daily office commuters to use bike as an alternative work travel mode.

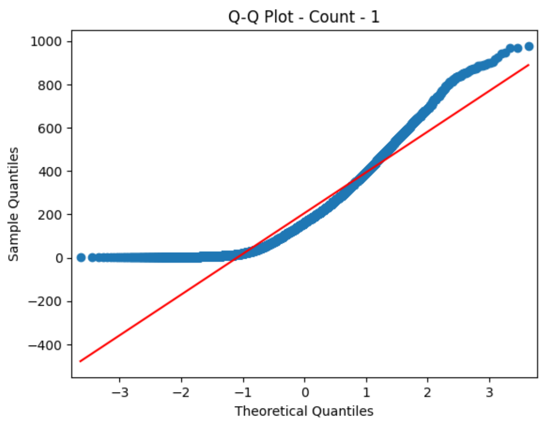
4. Check if the demand of bicycles on rent is the same for different Weather conditions?

Null Hypothesis (H0):
Demand for bicycles on rent is same for different weather conditions

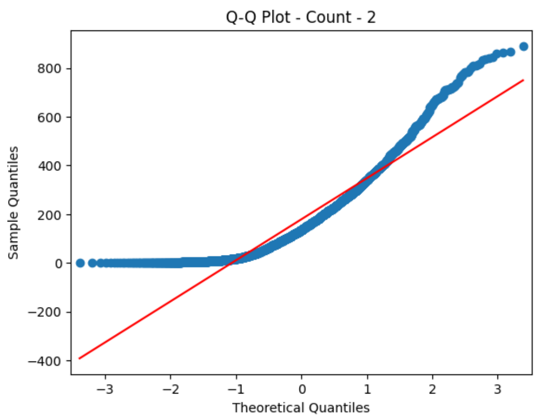
Alternative Hypothesis (H1):
Demand for bicycles on rent is the different for different Weather conditions

```
condition = data['weather'].unique()

def plot_qqplot(data,title):
  plt.figure(figsize=(12, 10))
  qqplot(data, line='s')
  plt.title(f'Q-Q Plot - {title}')
  plt.show()
def skew_kurtosis(data,title):
  return data.skew(),data.kurt()
def shapiro_test(data):
    stat, p = shapiro(data)
```

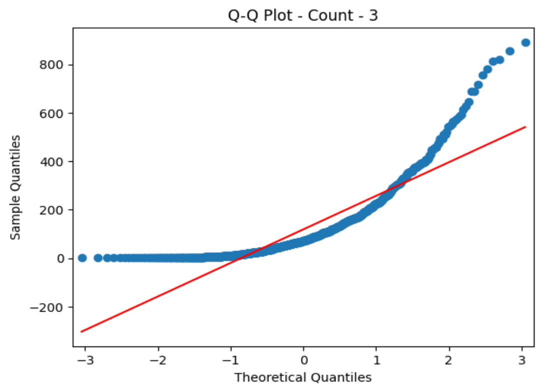
```
    return stat, p

for i,cond in enumerate(condition):
  data_by_weather = data[data['weather']==cond]['count']
  plot_qqplot(data_by_weather, f'Count - {cond}')
  skew,kurt = skew_kurtosis(data_by_weather, f'Count - {cond}')
  print(f'Skewness: {skew}, Kurtosis: {kurt}')
  if len(data_by_weather) >= 3:
    stat, p = shapiro(data_by_weather)
    print(f'{cond} - Shapiro-Wilk Test: Statistic={stat}, p-value={p}')
  else:
    print(f'{cond} - Not enough data for Shapiro-Wilk Test (need at least 3 data points)')
```



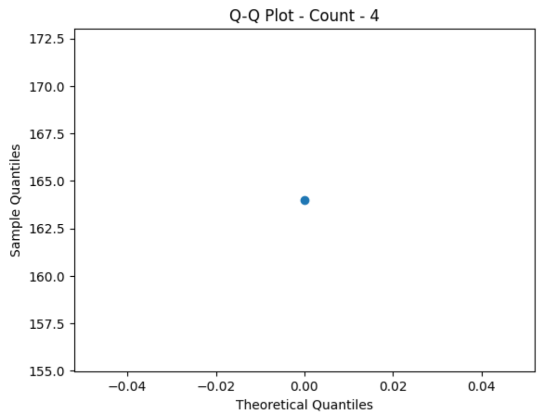
Skewness: 1.1398572666918205, Kurtosis: 0.964719852310354
1 - Shapiro-Wilk Test: Statistic=0.8909230828285217, p-value=0.0



Skewness: 1.294444423357868, Kurtosis: 1.5884304891319174
2 - Shapiro-Wilk Test: Statistic=0.8767687082290649, p-value=9.781063280987223e-43



Skewness: 2.1871371080456594, Kurtosis: 6.003053730759276
3 - Shapiro-Wilk Test: Statistic=0.7674332857131958, p-value=3.876090133422781e-33



Skewness: nan, Kurtosis: nan
4 - Not enough data for Shapiro-Wilk Test (need at least 3 data points)

```
data_by_weather = [data[data['weather']==cond]['count'] for cond in condition]
lstat,lpval = levene(*data_by_weather)
print(f'Levene Test Statistics: {lstat}, P_value: {lpval}')
```

NOTE:
The data provided doesn’t meet any of the conditions like Normality, equalance Variable.
So in this case we could have gone for Kruskal-Wallis Test. But as per the statement we go for one-way ANOVA.

```
fstat,pval = f_oneway(*data_by_weather)
print(f'One_way ANOVA Test Statistics: {fstat}, P_value: {pval}')

if pval <= alpha:
  print('Reject Null Hypothesis : Demand for bicycles on rent is the different for different Weather conditions')
else:
  print('Accept Null Hypothesis : Demand for bicycles on rent is same for different weather conditions')
```

```
Levene Test Statistics: 54.85106195954556, P_value: 3.5049379468332238e-35
One_way ANOVA Test Statistics: 65.53024112793271, P_value: 5.482069475935669e-42
Reject Null Hypothesis : Demand for bicycles on rent is the different for different Weather conditions
```

Inferences & conclusions
The demand for bicycles on rent varies significantly across different seasons. This variability is statistically significant, meaning that different seasons see different levels of bicycle rental activity.

Recommendations
Spring and Summer: Likely higher demand. Increase inventory to ensure enough bikes are available. Consider increasing staff to handle higher rental volumes.

Fall: Moderate demand. Maintain a balanced inventory. Monitor trends closely and adjust as needed.

Winter: Lower demand. Reduce inventory to avoid excess bikes. Focus on maintenance and preparing for the next high-demand season.

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

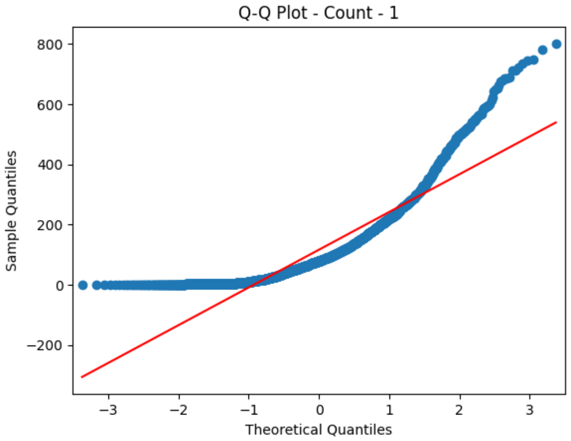
Null Hypothesis(H0) :
Demand for bicycle on rent is same for different season

Alternative Hypothesis(H1) :
Demand for bicycle on rend is different for different season

```
season = data['season'].unique()
season
```

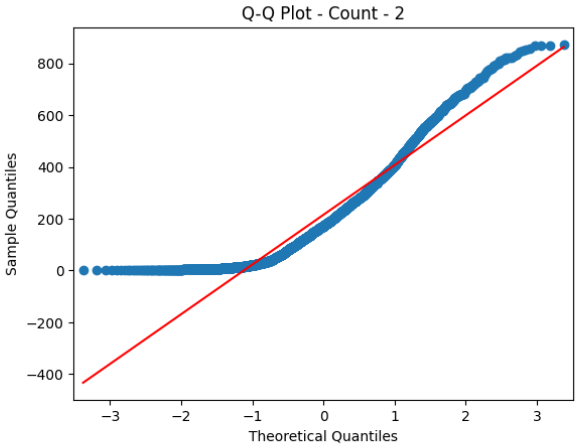
```
array([1, 2, 3, 4], dtype=object)
```

```
def skew_kurtosis(data,title):
  return data.skew(),data.kurt()
def shapiro_test(data):
    stat, p = shapiro(data)
    return stat,p

for i,sea in enumerate(season):
  data_by_season = data[data['season']==sea]['count']
  plot_qqplot(data_by_season, f'Count - {sea}')
  skew,kurt = skew_kurtosis(data_by_season, f'Count - {sea}')
  print(f'Skewness: {skew}, Kurtosis: {kurt}')
  stat, p = shapiro(data_by_season)
  print(f'{sea} - Shapiro-Wilk Test: Statistic={stat}, p-value={p}')
```



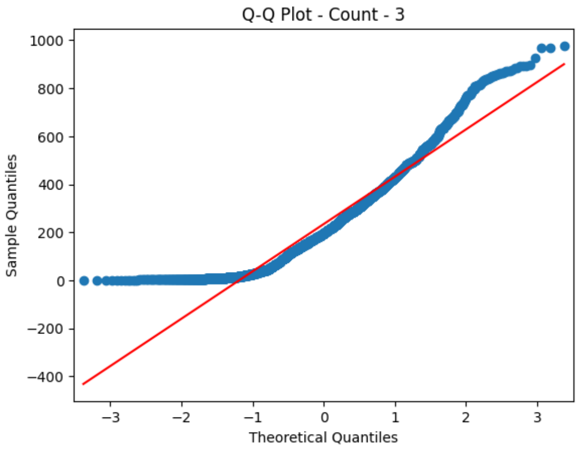
Skewness: 1.8880559001782309, Kurtosis: 4.31475739331681

1 - Shapiro-Wilk Test: Statistic=0.8087388873100281, p-value=0.0



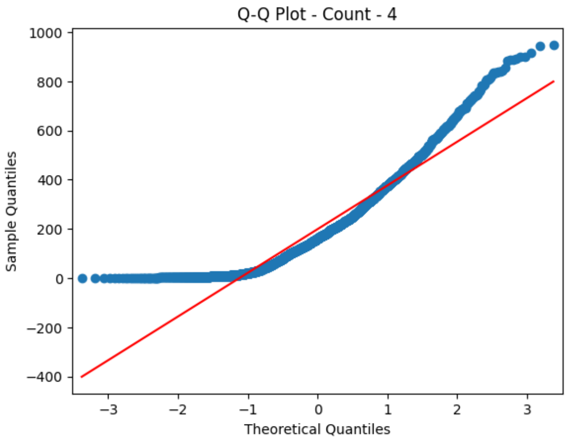
Skewness: 1.0032642267278118, Kurtosis: 0.42521337827415717

2 - Shapiro-Wilk Test: Statistic=0.900481641292572, p-value=6.039093315091269e-39



Skewness: 0.9914946474772749, Kurtosis: 0.6993825795653992

3 - Shapiro-Wilk Test: Statistic=0.9148160815238953, p-value=1.043458045587339e-36



Skewness: 1.172117329762622, Kurtosis: 1.2734853552995302
4 - Shapiro-Wilk Test: Statistic=0.8954644799232483, p-value=1.1301682309549298e-39

```
data_by_season = [data[data['season']==sea]['count'] for sea in season]
lstat,lpval = levene(*data_by_season)
print(f'Levene Test Statistics: {lstat}, P_value: {lpval}')
```

```
Levene Test Statistics: 187.7706624026276, P_value: 1.0147116860043298e-118
```

NOTE:
The data provided doesn't meet any of the conditions like Normality, equalance Variable.
So in this case we could have gone for Kruskal-Wallis Test. But as per the statement we go for one-way ANOVA.

```
alpha = 0.05
fstat,pval = f_oneway(*data_by_season)
print(f'One_way ANOVA Test Statistics: {fstat}, P_value: {pval}')

if pval < alpha:
  print('Reject Null Hypothesis : Demand for bicycles on rent is the different for different Weather conditions')
else:
  print('Accept Null Hypothesis : Demand for bicycles on rent is same for different weather conditions')
```

```
One_way ANOVA Test Statistics: 236.94671081032106, P_value: 6.164843386499654e-149
Reject Null Hypothesis : Demand for bicycles on rent is the different for different Weather conditions
```

Inferences & conclusions

The demand for bicycles on rent varies significantly across different seasons. This variability is statistically significant, meaning that different seasons see different levels of bicycle rental activity.

Recommendations

Spring and Summer: Likely higher demand. Increase inventory to ensure enough bikes are available. Consider increasing staff to handle higher rental volumes.

Fall: Moderate demand. Maintain a balanced inventory. Monitor trends closely and adjust as needed.

Winter: Lower demand. Reduce inventory to avoid excess bikes. Focus on maintenance and preparing for the next high-demand season.

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

Null Hypothesis (H0):
Weather conditions are same during different seasons

Alternative Hypothesis (H1):
Weather conditions are significantly different during different Seasons

```
val = pd.crosstab(data['season'],data['weather'])
val
```

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

```
chi2_stat, p_val, dof, expected = chi2_contingency(val)
print(f'Chi-Square Test Statistics: {chi2_stat}, P_value: {p_val}')
```

```
Chi-Square Test Statistics: 49.158655596893624, P_value: 1.549925073686492e-07
```

```
alpha = 0.05
if p_val < alpha:
  print('Reject Null Hypothesis : Weather conditions are significantly different during different Seasons')
else:
  print('Accept Null Hypothesis : Weather conditions are same during different seasons')
```

```
Reject Null Hypothesis : Weather conditions are significantly different during different Seasons
```

Resource Allocation: Allocate resources such as bikes and staff based on the expected weather conditions for each season.

Maintenance Scheduling: Plan bike maintenance during seasons with expected lower demand (e.g., winter) to ensure all bikes are in top condition for the peak seasons.

Weather-Based Promotions: Offer weather-specific promotions to encourage bike rentals regardless of weather conditions.

Dynamic Pricing: Implement dynamic pricing strategies to adjust rental rates based on the current and forecasted weather conditions.

The analysis indicates significant variations in weather conditions across different seasons. By understanding these patterns, the bike rental service can strategically plan inventory, marketing, and operational activities to align with expected weather conditions. Implementing these recommendations will help optimize service delivery, ensure customer satisfaction, and maintain steady demand throughout the year.