Yulu: Hypothesis Testing

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

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

```
[4] # Showing dataset structure
    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 | New interactive sheet

```
[6] df.shape
```

```
(10886, 12)
```

```
[13] # Showing Characteristics of dataset
     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
```

```
[14] # Showing Stattistical Summary
     df.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.

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

	0
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.

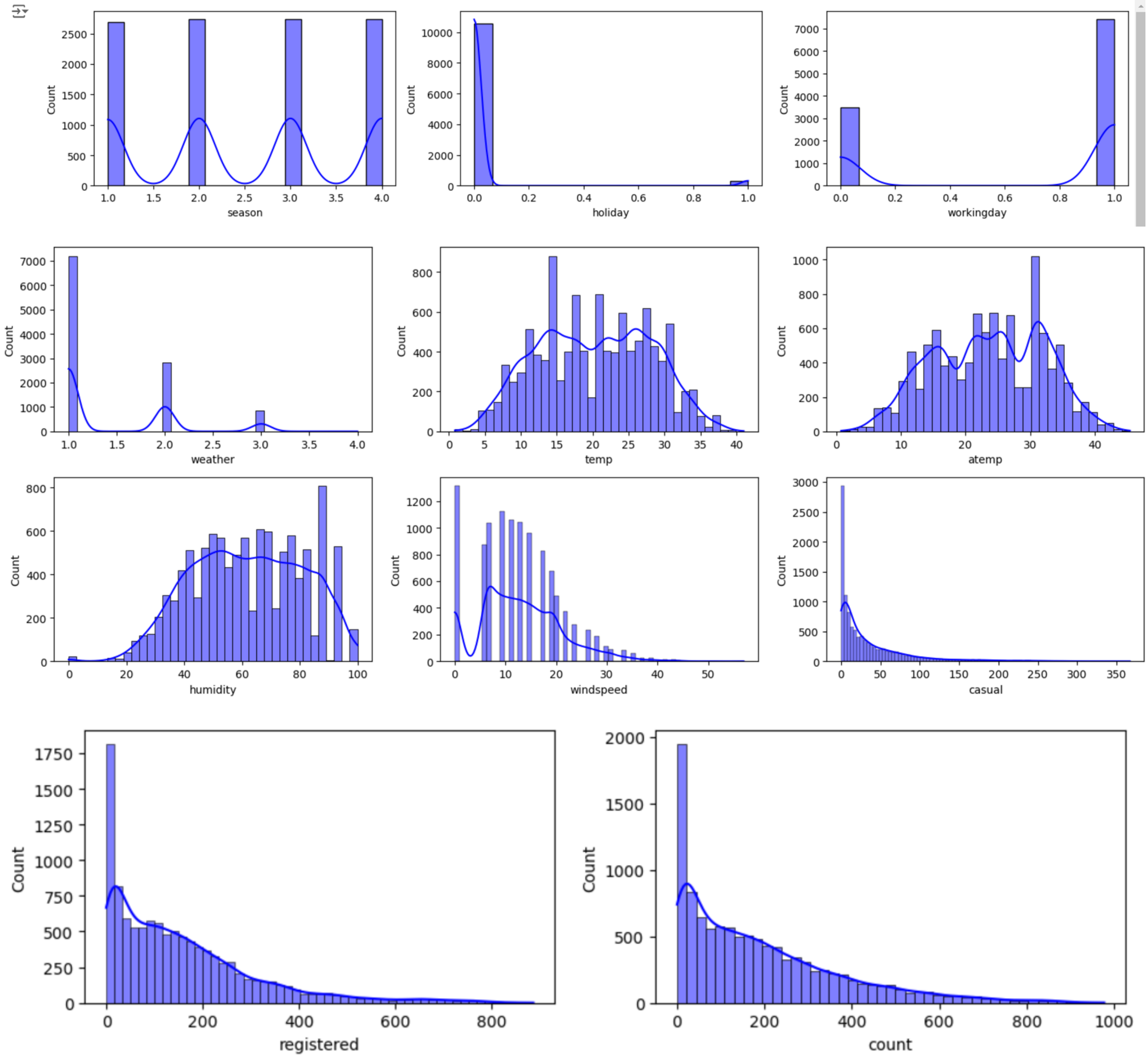
```
[22] # Identifying Duplicate records
     df.duplicated().sum()
```

```
0
```

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

```
# Distribution of Numerical Variables
num_col = df.select_dtypes(include='number').columns[:11]
fig, axes = plt.subplots(nrows=4, ncols=3, figsize=(15, 12))
axes = axes.flatten()
for i, col in enumerate(num_col):
    sbn.histplot(df[col], ax=axes[i], kde=True, color='blue')

for j in range(len(num_col), len(axes)):
    fig.delaxes(axes[j])
plt.tight_layout()
plt.show()
```



e. Check for Outliers and deal with them accordingly.

```
# Checking for Outliers
num_col=df.select_dtypes(include='number').columns
outliers={}
for col in num_col:
  Q1=df[col].quantile(0.25)
  Q3=df[col].quantile(0.75)
  IQR=Q3-Q1
  outliers[col]=df[(df[col]<(Q1-1.5*IQR))|(df[col]>(Q3+1.5*IQR))][col]
print(outliers)
```

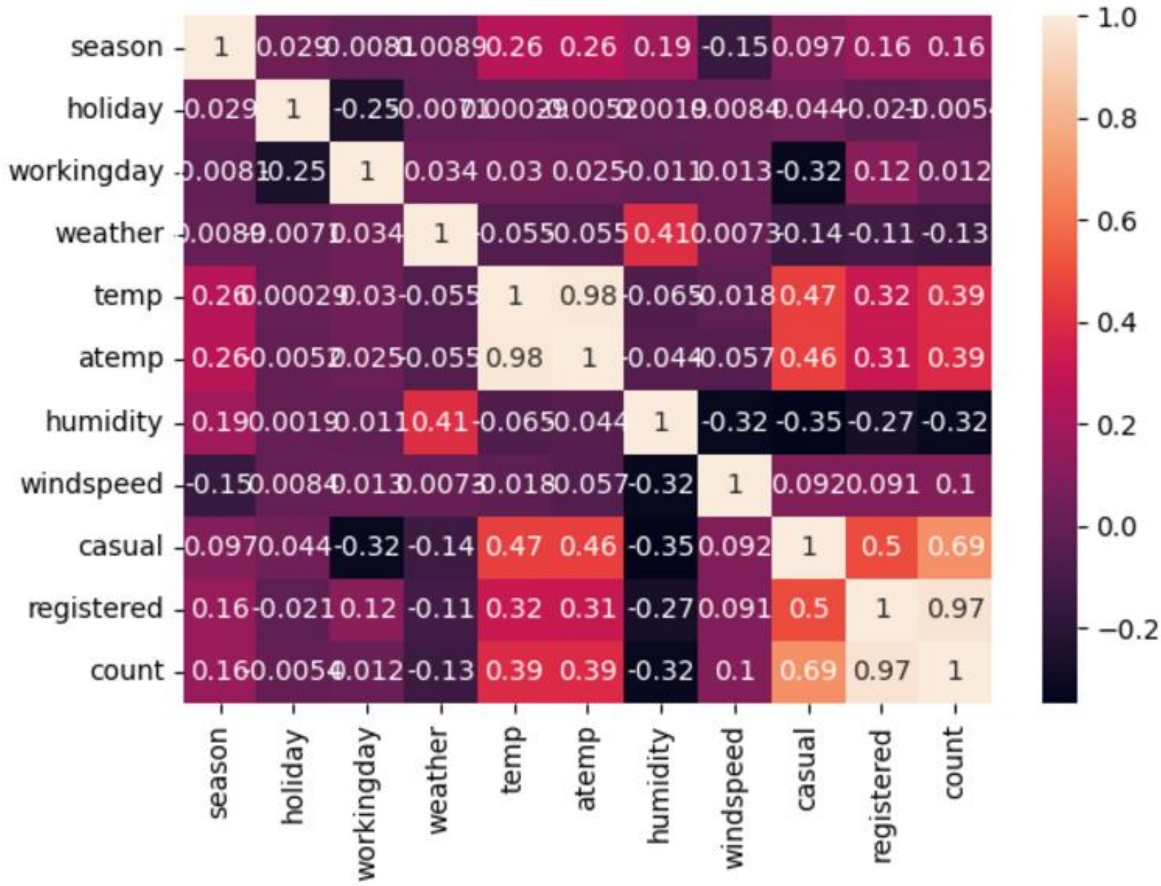
```
{'season': Series([], Name: season, dtype: int64), 'holiday': 372      1
373      1
374      1
375      1
376      1
        ..
10257    1
10258    1
10259    1
10260    1
10261    1
Name: holiday, Length: 311, dtype: int64, 'workingday': Series([], Name: workingda
Name: weather, dtype: int64, 'temp': Series([], Name: temp, dtype: float64), 'aten
1092    0
1093    0
1094    0
1095    0
1096    0
1097    0
1098    0
1099    0
1100    0
1101    0
1102    0
1103    0
1104    0
1105    0
1106    0
1107    0
1108    0
1109    0
1110    0
1111    0
1112    0
```

```
Name: humidity, dtype: int64, 'windspeed': 175      32.9975
178      36.9974
194      35.0008
196      35.0008
265      39.0007
          ...
10013    32.9975
10154    32.9975
10263    43.0006
10540    32.9975
10853    32.9975
Name: windspeed, Length: 227, dtype: float64, 'casual': 1173     144
1174     149
1175     124
1311     126
1312     174
        ...
10610    122
10611    148
10612    164
10613    167
10614    139

Name: casual, Length: 749, dtype: int64, 'registered': 1987     539
2011     532
2059     540
2179     521
2371     516
        ...
10855    533
10856    512
10870    665
10879    536
10880    546
Name: registered, Length: 423, dtype: int64, 'count': 6611     712
6634     676
6635     734
6649     662
6658     782
        ...
10678    724
10702    688
10726    679
10846    662
10870    678
Name: count, Length: 300, dtype: int64}
```

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

```
# Establishing relationship between Dependent & Independent variables
df_num = df.select_dtypes(include=np.number)
sbn.heatmap(df_num.corr(), annot=True)
plt.tight_layout()
plt.show()
```



- Casual users are increased during holidays.
- Registered users during Fall and winter.
- Casual users are proportional to registered users.
- Humidity increases during light rain as well as in heavy rain.
- Count increases as temp increase.
- Wind speed increases in spring and summer.
- Registered uses increase with increase in temperature.

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

a. Formulate Null Hypothesis (H0) and Alternate Hypothesis (H1)

H0: Mean of average riders in weekdays and weekend are equal.
H1: Mean of average riders in weekdays and weekends are not equal.

b. Select an appropriate test

```
[12] # Checking for significant difference between the no. of bike rides on Weekdays and weekends
     df_weekdays=df.loc[df.workingday==1,'count']
     df_weekends=df.loc[df.workingday==0,'count']

     import scipy.stats as stats
     stats.ttest_ind(df_weekdays,df_weekends)

     TtestResult(statistic=1.2096277376026694, pvalue=0.22644804226361348, df=10884.0)
```

c. Set a significance level

Alpha=5%

d. Calculate test Statistics / p-value

Test stat=1.2096, p-value=0.226

e. Decide whether to accept or reject the Null Hypothesis.

As p-value > Alpha (5%), hence we fail to reject null hypothesis(H0)

f. Draw inferences & conclusions from the analysis and provide recommendations.

Conclusion: Mean of average riders on weekdays and weekend are approximately equal.

Recommendation:

- Offer monthly ride packages for office commuters.
- Offer group ride promotions for families and friends on weekends
- Lower prices during off peak hours on weekdays.
- Slightly increase prices during weekends peak time.

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

a. Formulate Null Hypothesis (H0) and Alternate Hypothesis (H1)

H0: Mean of all weather conditions are equal.
H1: At least one weather condition mean is different from others.

b. Select an appropriate test –

```
[16] # Checking if the demand of bicycles is the same for different Weather conditions.
df_1=df.loc[df.weather==1,'count']
df_2=df.loc[df.weather==2,'count']
df_3=df.loc[df.weather==3,'count']
df_4=df.loc[df.weather==4,'count']

import scipy.stats as stats
stats.f_oneway(df_1,df_2,df_3,df_4)
```

```
F_onewayResult(statistic=65.53024112793271, pvalue=5.482069475935669e-42)
```

c. Check assumptions of the test

Normality:

H0: Sample data is gaussian

H1: Sample data is not gaussian

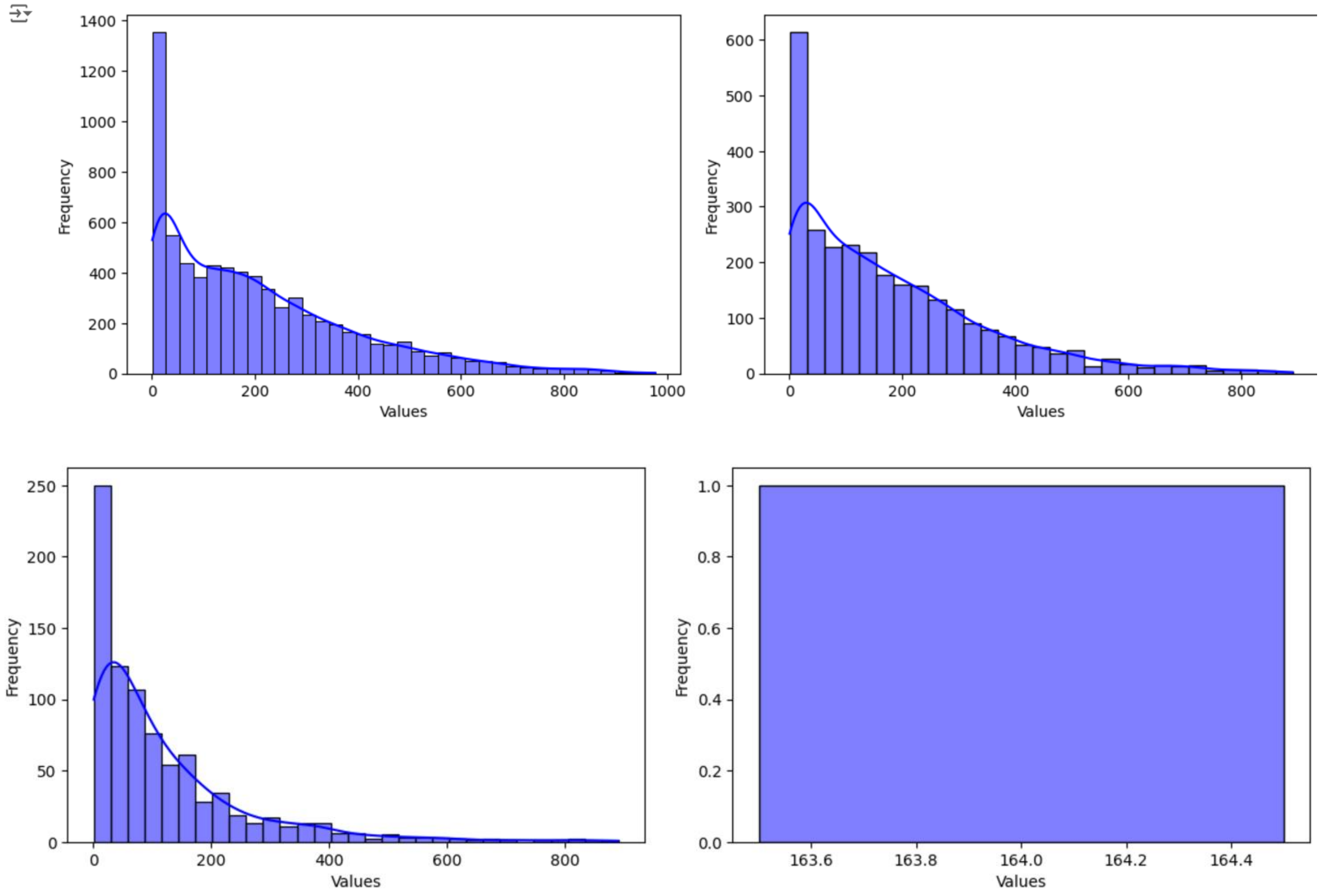
As sample data has p_value < Alpha (5%)

Hence, we conclude that data is not gaussian.

```
[37] # Checking Normality for all weather groups
from scipy.stats import shapiro
df1_sample= df_1.sample(n=4000, random_state=42)
df2_sample=df_1.sample(n=4000, random_state=42)
df3_sample=df_1.sample(n=4000, random_state=42)
df4_sample=df_1.sample(n=4000, random_state=42)
stats, p_value_1=shapiro(df1_sample)
stats, p_value_2=shapiro(df2_sample)
stats, p_value_3=shapiro(df3_sample)
stats, p_value_4=shapiro(df4_sample)
print(stats, p_value_1)
print(stats, p_value_2)
print(stats, p_value_3)
print(stats, p_value_4)
```

```
0.8921085931130791 1.6062128408387876e-46
0.8921085931130791 1.6062128408387876e-46
0.8921085931130791 1.6062128408387876e-46
0.8921085931130791 1.6062128408387876e-46
```

```python
# Checking normality for all 4 weather conditions
import matplotlib.pyplot as plt
dataframes = [df_1, df_2, df_3, df_4]
fig, axes = plt.subplots(2, 2, figsize=(12, 8))
for i, ax in enumerate(axes.flatten()):
        sbn.histplot(dataframes[i], ax=ax, kde=True, color='blue')
        ax.set_xlabel('Values')
        ax.set_ylabel('Frequency')
plt.tight_layout()
plt.show()
```



Equality of Variance:

H0: Variance is equal

H1: Variance is not equal

As p_value < Alpha (5%), hence we conclude that variance is not equal.

```python
[50] # Checking for Equal Variance
from scipy.stats import levene
stats, p_value=levene(df_1,df_2,df_3,df_4)
print(stats, p_value)
```

```
54.85106195954556 3.504937946833238e-35
```

d. **Set a significance level and calculate the test Statistics / p-value.**

Alpha = 5%

Stats=54.85106195954556, P_Value=3.504937946833238e-35

e. **Decide whether to accept or reject the Null Hypothesis.**

As P_value is less than alpha (5%) hence we reject null hypothesis.

f. **Draw inferences & conclusions from the analysis and provide recommendations.**

Inferences and Conclusions:

- Mean count of bikes in at least one weather condition is significantly different from others, it suggests weather conditions impact bike availability.

Recommendation:

- Increase bike availability during high demand weather condition.
- Provide discounts during low demand weather condition.
- During heavy rain, move unused bikes to area with indoor docking stations.
- Promote safety feature like slip reissuance tyre.
- Suggest safe routes based on weather conditions.

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

a. Formulate Null Hypothesis (H0) and Alternate Hypothesis (H1)

H0: Mean of no. of bikes for all four seasons are equal.

H1: Mean of no. of bikes for at least one season is different to rest others.

b. Select an appropriate test –

```
# Checking for the demand of bikes for different Seasons.
import scipy.stats as stats

df_summer=df.loc[df.season==1,'count']
df_winter=df.loc[df.season==2,'count']
df_fall=df.loc[df.season==3,'count']
df_spring=df.loc[df.season==4,'count']

stats_result, p_value = stats.f_oneway(df_summer, df_winter, df_fall, df_spring)
print(stats_result, p_value)
```

```
236.94671081032106 6.164843386499654e-149
```

c. Check assumptions of the test-

Normality:

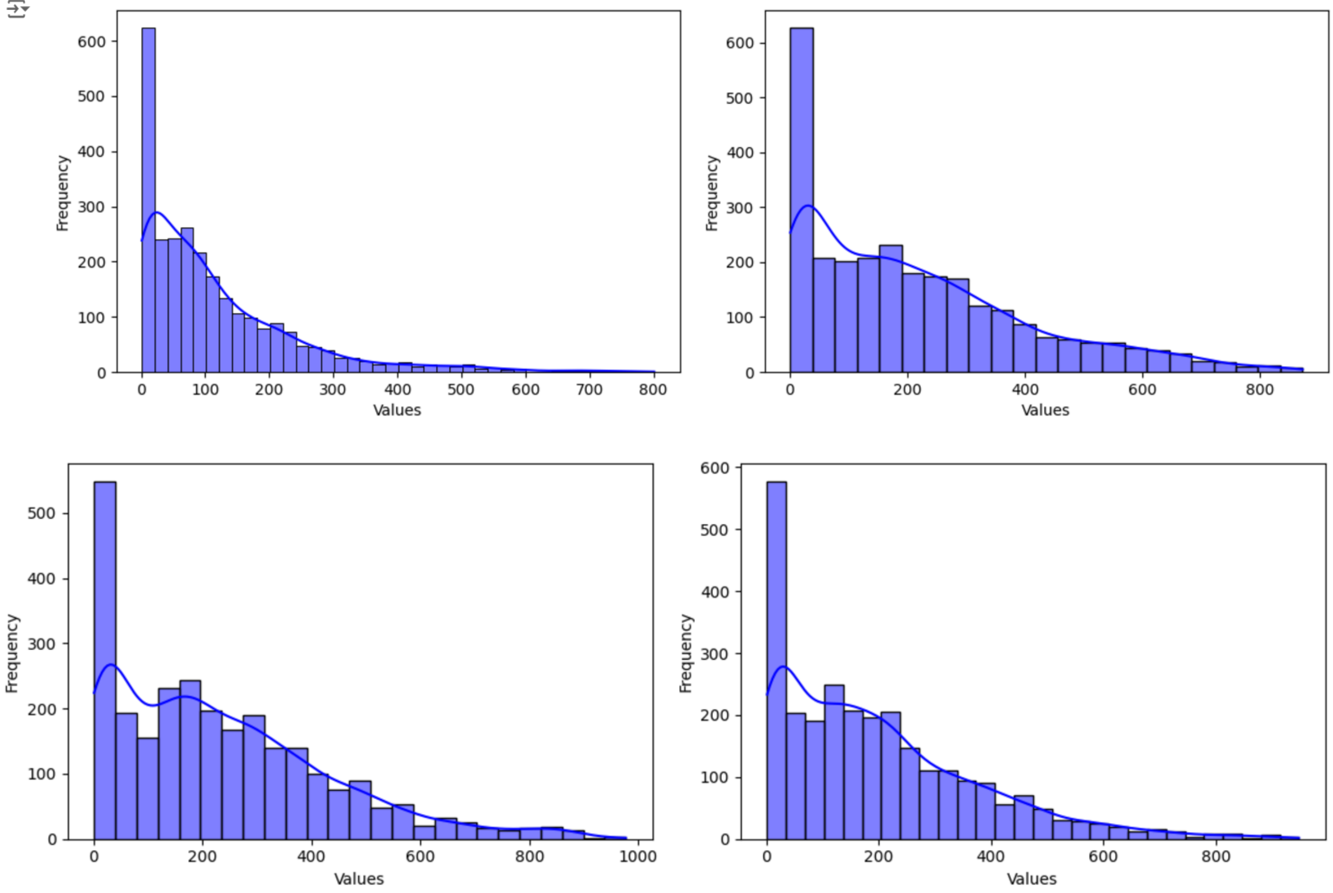
As the p_value < Alpha (5%), hence reject null hypothesis.

```
[56] # Checking for normality of count of bikes for different seasons
stats_summer, p_value_summer = shapiro(df_summer)
stats_winter, p_value_winter = shapiro(df_winter)
stats_fall, p_value_fall = shapiro(df_fall)
stats_spring, p_value_spring = shapiro(df_spring)

print(stats_summer, p_value_summer)
print(stats_winter, p_value_winter)
print(stats_fall, p_value_fall)
print(stats_spring, p_value_spring)
```

```
0.8087378401253588 8.749584618867662e-49
0.9004818080893252 6.039374406270491e-39
0.9148166372899196 1.043680518918597e-36
0.8954637482095505 1.1299244409282836e-39
```

[57]
```python
# Checking normality for count of bikes for all seasons
import matplotlib.pyplot as plt
dataframes = [df_summer, df_winter, df_fall, df_spring]
fig, axes = plt.subplots(2, 2, figsize=(12, 8))
for i, ax in enumerate(axes.flatten()):
        sbn.histplot(dataframes[i], ax=ax, kde=True, color='blue')
        ax.set_xlabel('Values')
        ax.set_ylabel('Frequency')
plt.tight_layout()
plt.show()
```



Variance Equality:

H0: Variance is equal

H1: Variance is not equal

As p_value < Alpha (5%), hence we conclude that variance is not equal.

[58]
```python
# Checking for Equal Variance
from scipy.stats import levene
stats, p_value=levene(df_summer,df_winter,df_fall,df_spring)
print(stats, p_value)
```

```
187.7706624026276 1.0147116860043298e-118
```

d. Set a significance level and calculate the test Statistics / p-value.

Alpha=5%

Stats=236.94671081032106, P_Value=6.164843386499654e-149

e. Decide whether to accept or reject the Null Hypothesis.

As p_value is less than alpha (5%), hence we reject null hypothesis.

f. Draw inferences & conclusions from the analysis and provide recommendations.

Inferences and Conclusions:

- Mean count of bikes in at least one season condition is significantly different from others, it suggests season impact bike availability.

Recommendation:

- Increase number of bikes in popular location in parks and tourist spot.
- Reduce count of bikes in less used locations to save operational cost.
- Attract more riders with promotional offers.
- Promotes morning and afternoon rides to avoid cold nights.
- Offer handlebars with grip warmers or gloves for comfort.
- Beat the Heat with Evening Rides.
- Higher prices during peak seasons to maximize revenue.
- Lower prices in off-peak seasons to attract new users.

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

a. Formulate Null Hypothesis (H0) and Alternate Hypothesis (H1).

H0: There are no association between weather conditions and seasons.
H1: Weather conditions and seasons are dependent on each other.

b. Select an appropriate test –

```
[4] # Checking weather conditions during different seasons.
    dfw_1=df[df.season==1].weather.value_counts()
    dfw_2=df[df.season==2].weather.value_counts()
    dfw_3=df[df.season==3].weather.value_counts()
    dfw_4=df[df.season==4].weather.value_counts()

    contingency_table = pd.DataFrame({'season_1': dfw_1, 'season_2': dfw_2, 'season_3': dfw_3, 'season_4': dfw_4}).fillna(0).astype(int)


    from scipy.stats import chi2_contingency
    stats, p_value, dof, expected = chi2_contingency(contingency_table)
    print(stats, p_value)
```

```
49.15865559689363 1.5499250736864862e-07
```

c. Create a Contingency Table against 'Weather' & 'Season' columns.

```
# Creating Contingency Table against ‘Weather’ & ‘Season’ columns.
season_mapping = {1: "Spring(1)", 2: "Summer(2)", 3: "Fall(3)", 4: "Winter(4)"}
df['season_name'] = df['season'].map(season_mapping)

contingency_table = pd.crosstab(df['weather'], columns=df['season_name'])
print(contingency_table)
```

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

d. Set a significance level and calculate the test Statistics / p-value.

Alpha=5%

Stats=49.15865559689363, P_Value=1.5499250736864862e-07

e. Decide whether to accept or reject the Null Hypothesis.

As p_value < alpha, hence we reject null hypothesis.

f. Draw inferences & conclusions from the analysis and provide recommendations.

Inferences & Conclusions:
Weather conditions significantly depend upon seasons.

- Weather condition 1 dominates across all seasons, with its highest frequency in Fall.
- Weather condition 2 shows higher occurrences in Winter.
- Weather condition 3 has a relatively uniform distribution.
- Weather condition 4 is rare but observed in spring.

Recommendations:

- Deploy more bikes with favourable weather like fall and summer.
- Reduce fleet availability during adverse conditions (Ex. in winter)
- Get users updated with real time weather conditions like **rain** and **cold.**
- Provide facilities like raincoats, umbrellas, or warm gear for rent in seasons with more adverse weather conditions (Ex. winter, rain)
- Create seasonal marketing strategy, highlighting the joy of riding in **Spring** and **Fall**, when weather is most favourable.
- Educate users on safety measures during adverse weather conditions (Ex. slippery roads in monsoon) .
- Implement dynamic pricing to reflect demand during weather changes.