Yulu case study

Q1. Define the Problem Statement, Import the required Libraries and perform Exploratory Data Analysis.

In [567…
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [568…
```
df=pd.read_csv('https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/original/bike_sharing.csv?1642089089')
```

In [569…
```
df.head(10)
```

Out[569…

	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	0.0000	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0000	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0000	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0000	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0000	0	1	1
5	2011-01-01 05:00:00	1	0	0	2	9.84	12.880	75	6.0032	0	1	1
6	2011-01-01 06:00:00	1	0	0	1	9.02	13.635	80	0.0000	2	0	2
7	2011-01-01 07:00:00	1	0	0	1	8.20	12.880	86	0.0000	1	2	3
8	2011-01-01 08:00:00	1	0	0	1	9.84	14.395	75	0.0000	1	7	8
9	2011-01-01 09:00:00	1	0	0	1	13.12	17.425	76	0.0000	8	6	14

shape of data

In [570…
```
df.shape
```

Out[570…
```
(10886, 12)
```

In [571…
```
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
```

statastical summary

In [572…
```
df.describe()
```

Out[572...

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	r
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000	10886.000000	10886.000000	10886.000000	108
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.886460	12.799395	36.021955	1
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.245033	8.164537	49.960477	1
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	0.000000	0.000000	0.000000	
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	47.000000	7.001500	4.000000	
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	62.000000	12.998000	17.000000	1
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	77.000000	16.997900	49.000000	2
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	100.000000	56.996900	367.000000	8

counting uniques of each column

In [573...

```
print(f"the number of unique season is {df['season'].nunique()}")
print(f"the unique seasons are {df['season'].unique()}")
```

```
the number of unique season is 4
the unique seasons are [1 2 3 4]
```

In [574...

```
df["holiday"].unique()
```

Out[574...

```
array([0, 1])
```

In [575...

```
df["workingday"].unique()
```

Out[575...

```
array([0, 1])
```

In [576...

```
df["weather"].unique()
```

Out[576...

```
array([1, 2, 3, 4])
```

checking of nullvalues

In [577...

```
df.isnull().sum()
```

Out[577...

	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

observations

- no null values are found

checking of duplicates

In [578...

```
df.duplicated()
```

Out[578…

	0
0	False
1	False
2	False
3	False
4	False
...	...
10881	False
10882	False
10883	False
10884	False
10885	False

10886 rows × 1 columns

dtype: bool

Non graphical analysis

In [579…

```
df
```

Out[579…

	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	0.0000	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0000	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0000	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0000	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0000	0	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...
10881	2012-12-19 19:00:00	4	0	1	1	15.58	19.695	50	26.0027	7	329	336
10882	2012-12-19 20:00:00	4	0	1	1	14.76	17.425	57	15.0013	10	231	241
10883	2012-12-19 21:00:00	4	0	1	1	13.94	15.910	61	15.0013	4	164	168
10884	2012-12-19 22:00:00	4	0	1	1	13.94	17.425	61	6.0032	12	117	129
10885	2012-12-19 23:00:00	4	0	1	1	13.12	16.665	66	8.9981	4	84	88

10886 rows × 12 columns

In [580…

```
df.groupby("season")["count"].sum()
```

Out[580…

	count
season	
1	312498
2	588282
3	640662
4	544034

dtype: int64

In [581…

```
df.groupby("holiday")["count"].sum()
```

Out[581…

	count
holiday	
0	2027668
1	57808

dtype: int64

In [582…
```
df.groupby("workingday")["count"].sum()
```

Out[582…

	count
workingday	
0	654872
1	1430604

dtype: int64

In [583…
```
df.groupby("weather")["count"].sum()
```

Out[583…

	count
weather	
1	1476063
2	507160
3	102089
4	164

dtype: int64

In [584…
```
df.groupby("season")["count"].sum()
```

Out[584…

	count
season	
1	312498
2	588282
3	640662
4	544034

dtype: int64

In [585…
```
df["casual"].sum()
```

Out[585…
```
392135
```

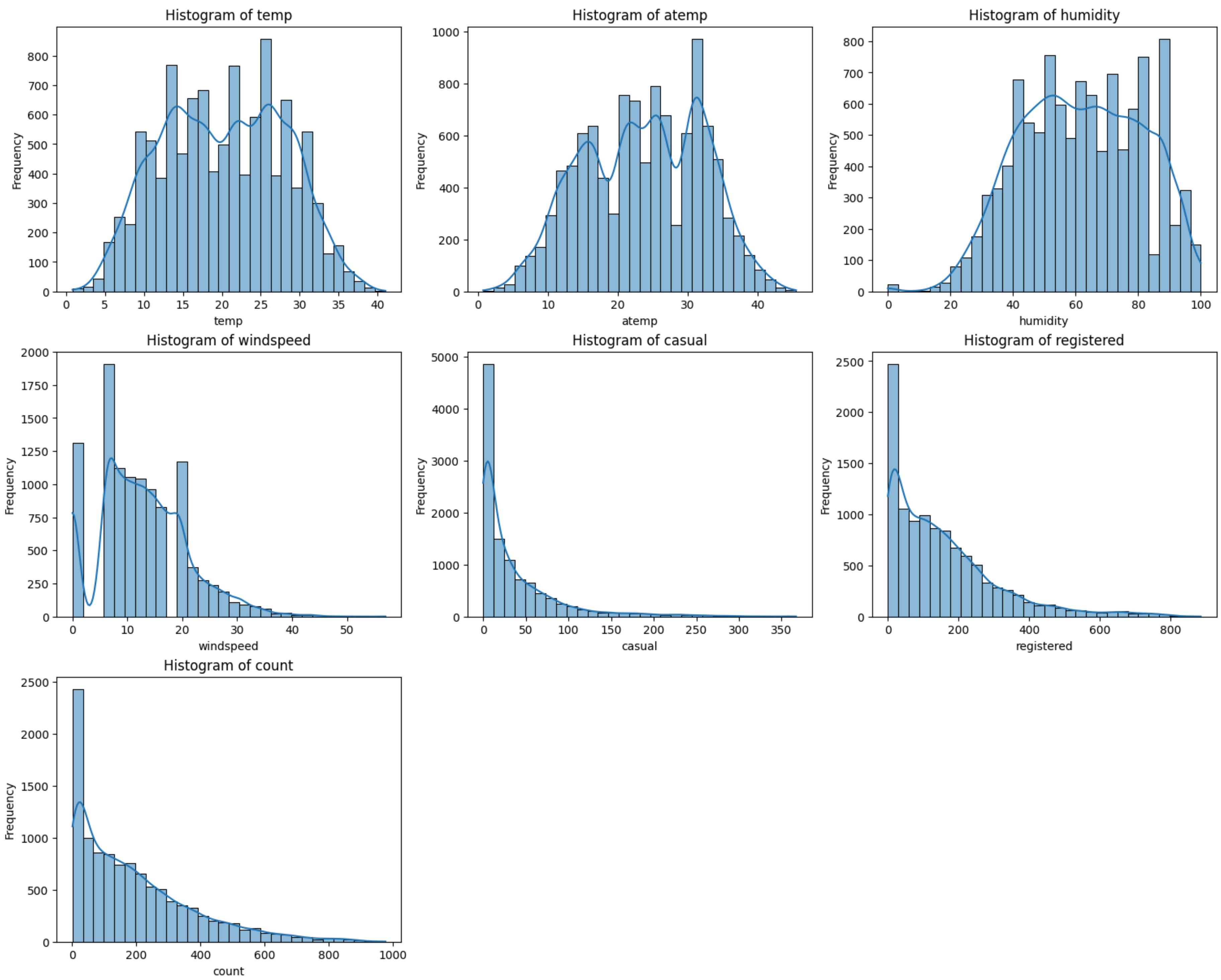
In [586…
```
df["registered"].sum()
```

Out[586…
```
1693341
```

univariate analysis

In [587…

```
import seaborn as sns
import matplotlib.pyplot as plt
columns = ["temp", "atemp", "humidity", "windspeed", "casual", "registered", "count"]
n_rows = 3
n_cols = 3
fig, axes = plt.subplots(n_rows, n_cols, figsize=(15, 12), constrained_layout=True)
axes = axes.flatten()
for i, col in enumerate(columns):
    sns.histplot(df[col], kde=True, ax=axes[i], bins=30)
    axes[i].set_title(f"Histogram of {col}")
    axes[i].set_xlabel(col)
    axes[i].set_ylabel("Frequency")

for j in range(len(columns), len(axes)):
    fig.delaxes(axes[j])

plt.show()
```



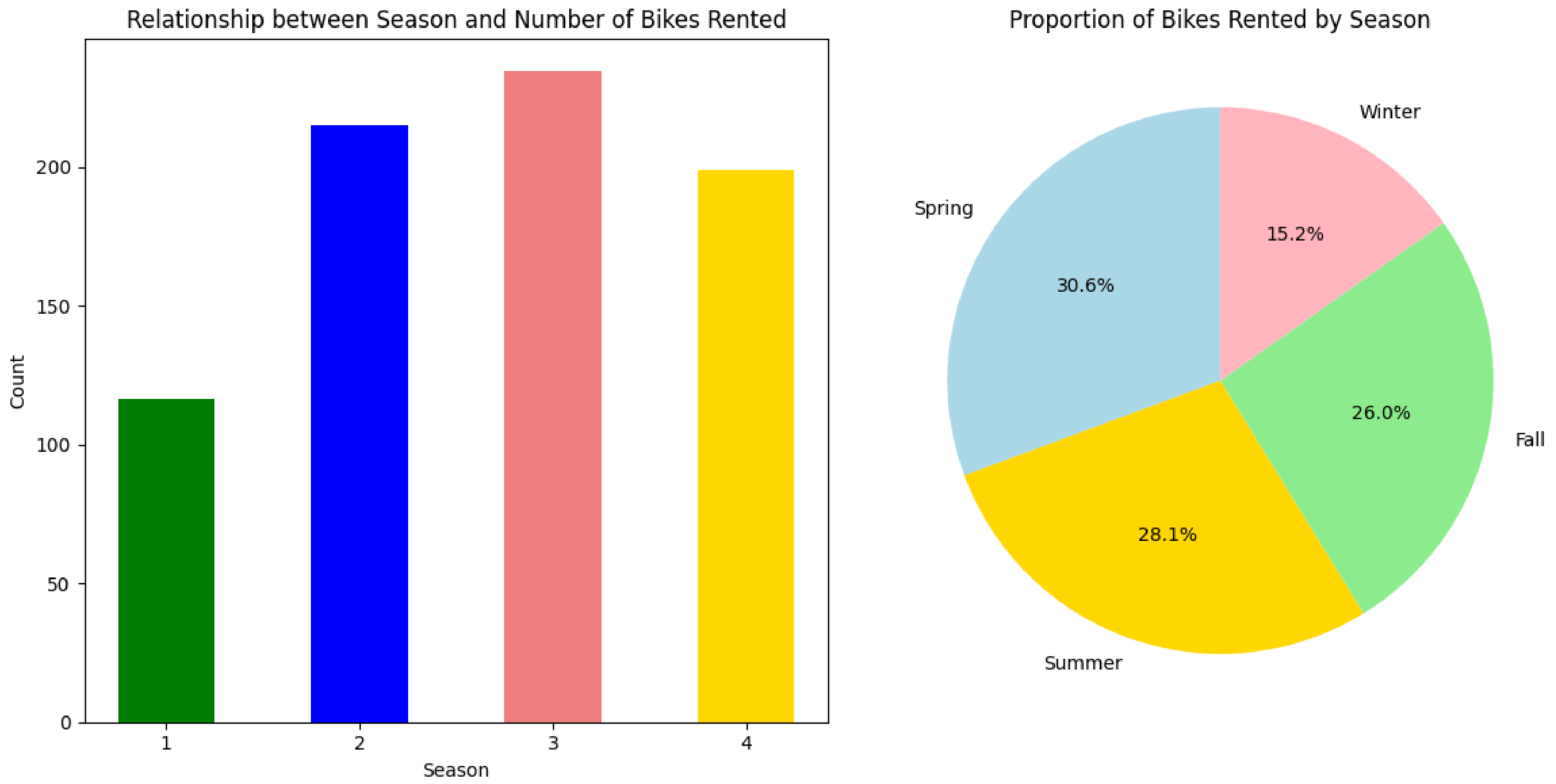
Bivariate analysis

relationship between season and number of bikes rented

In [588…

```
import matplotlib.pyplot as plt

# Assuming `seasons` is already calculated
seasons = df.groupby("season")["count"].mean().sort_values(ascending=False)

# Create a figure with two subplots
fig, ax = plt.subplots(1, 2, figsize=(12, 6))

# Bar chart
ax[0].bar(x=seasons.index, height=seasons.values, color=["lightcoral","blue","gold","green"], width=0.5)
ax[0].set_xticks(seasons.index)
ax[0].set_xlabel("Season")
ax[0].set_ylabel("Count")
ax[0].set_title("Relationship between Season and Number of Bikes Rented")

# Pie chart
ax[1].pie(
```

```
    seasons.values,
    labels=["Spring", "Summer", "Fall", "Winter"],
    autopct="%1.1f%%",
    startangle=90,
    colors=["lightblue", "gold", "lightgreen", "lightpink"]
)
ax[1].set_title("Proportion of Bikes Rented by Season")

# Show the plots
plt.tight_layout()
plt.show()

# Print season data
print(seasons)
```




```
season
3    234.417124
2    215.251372
4    198.988296
1    116.343261
Name: count, dtype: float64
```

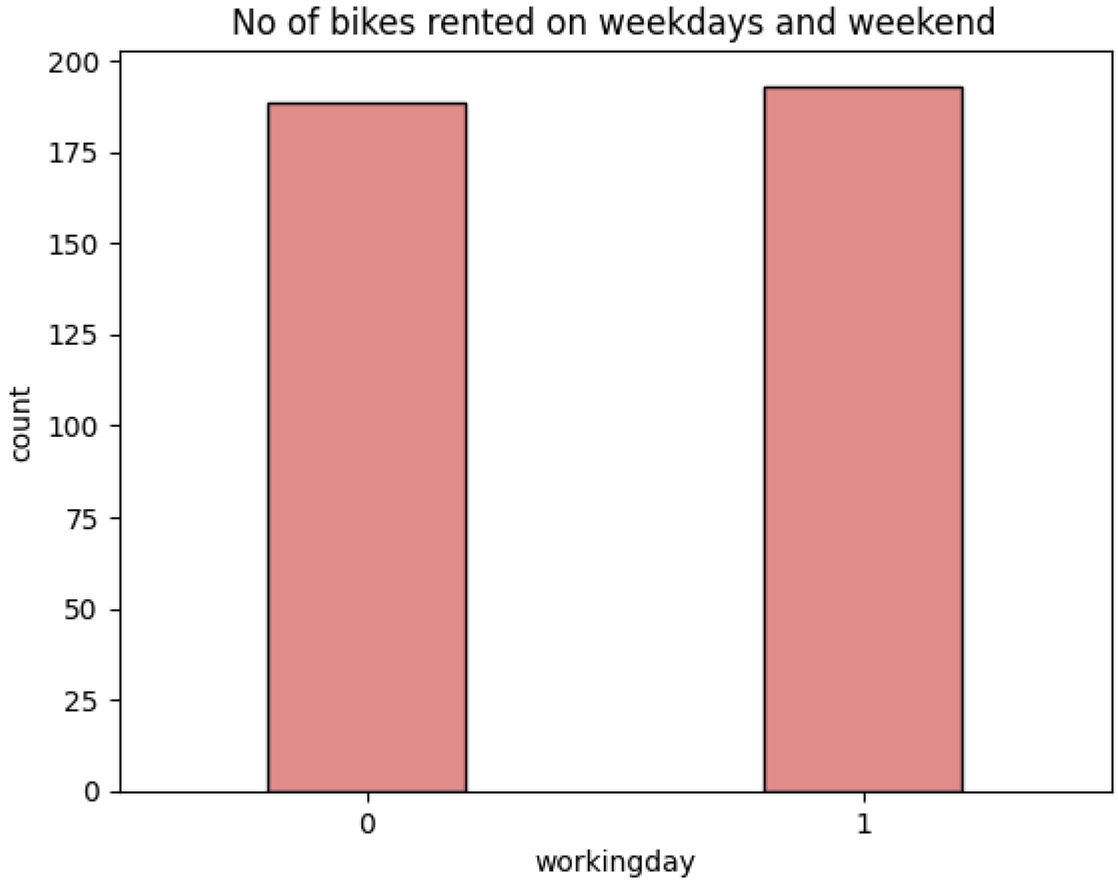
observations

- we can see that fall(season 3) has the highsest average of bikes rented.
- summer and winter has 2nd and 3rd highest average with spring being the least.

No of bikes rented on weekdays and weekend

In [589…

```
workingday = df.groupby("workingday")["count"].mean()
sns.barplot(x=workingday.index, y=workingday.values,color="lightcoral",width=0.4,
            edgecolor="black")
plt.xlabel("workingday")
plt.ylabel("count")
plt.title("No of bikes rented on weekdays and weekend")
plt.show()
print(workingday)
```




```
workingday
0    188.506621
1    193.011873
Name: count, dtype: float64
```

observations:

- the avg number of bikes rented during working and non working days doesnt show much difference.

realationship between holiday and no of bikes rented.

In [590…

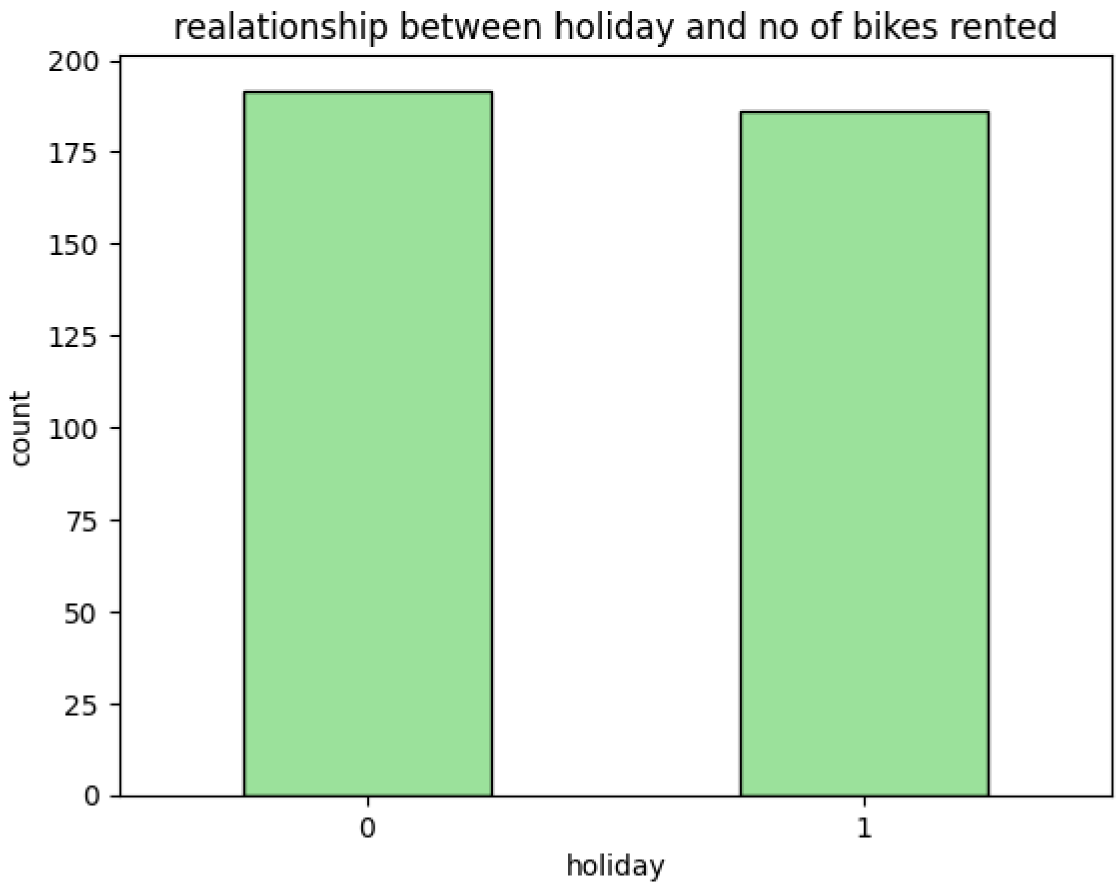
```
df["holiday"].unique()
```

Out[590…

```
array([0, 1])
```

In [591…

```
holidays=df.groupby("holiday")["count"].mean()
sns.barplot(x=holidays.index,y=holidays.values,color="lightgreen",width=0.5,edgecolor="black")
plt.xlabel("holiday")
plt.ylabel("count")
plt.title("realationship between holiday and no of bikes rented")
plt.show()
print(holidays)
```




```
holiday
0    191.741655
1    185.877814
Name: count, dtype: float64
```

observations

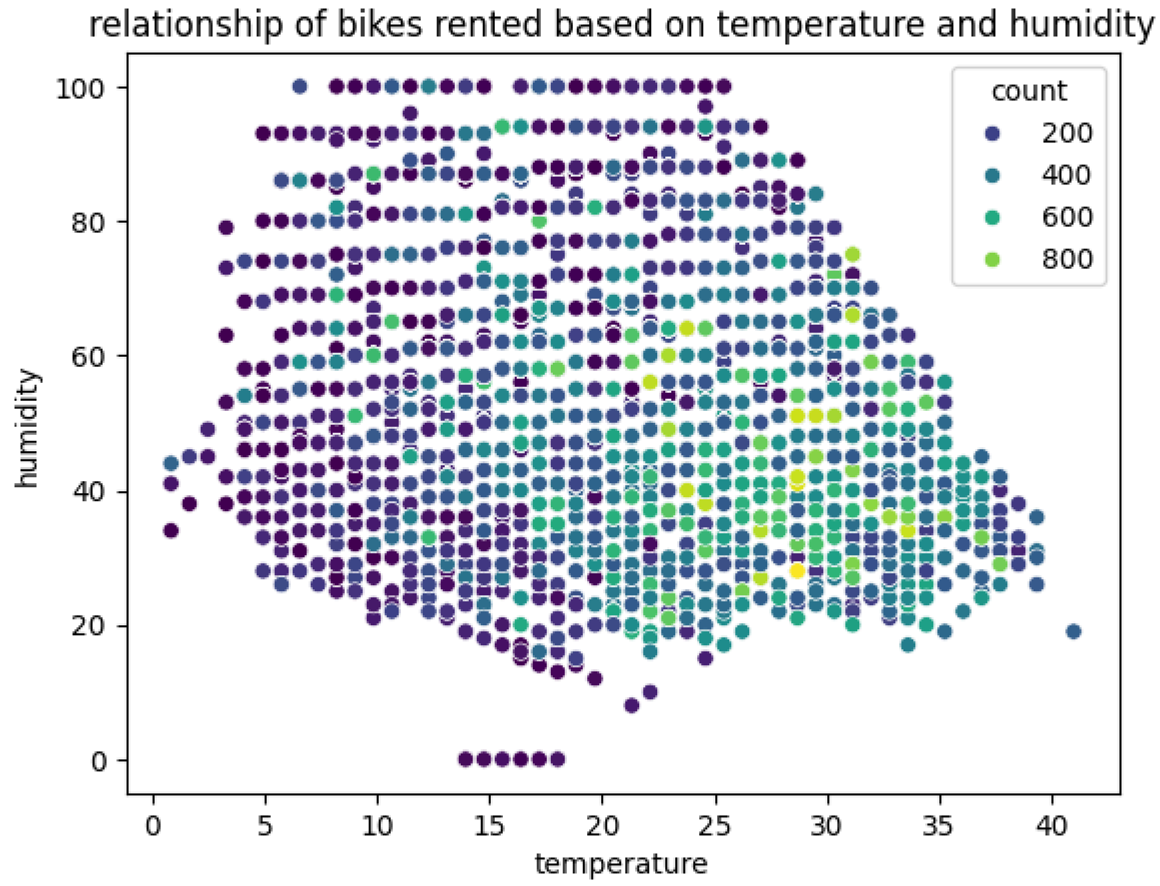
- the avaerage number of bycycles rented in holidays and working days doesnt show much of the difference.

multivariate analysis

relationship of bikes rented based on temperature and humidity

In [592…

```
sns.scatterplot(x=df["temp"],y=df["humidity"],hue=df["count"],palette="viridis") # Removed the extra bracket from "temperature
plt.xlabel("temperature")
plt.ylabel("humidity")
plt.title("relationship of bikes rented based on temperature and humidity")
plt.show()
```



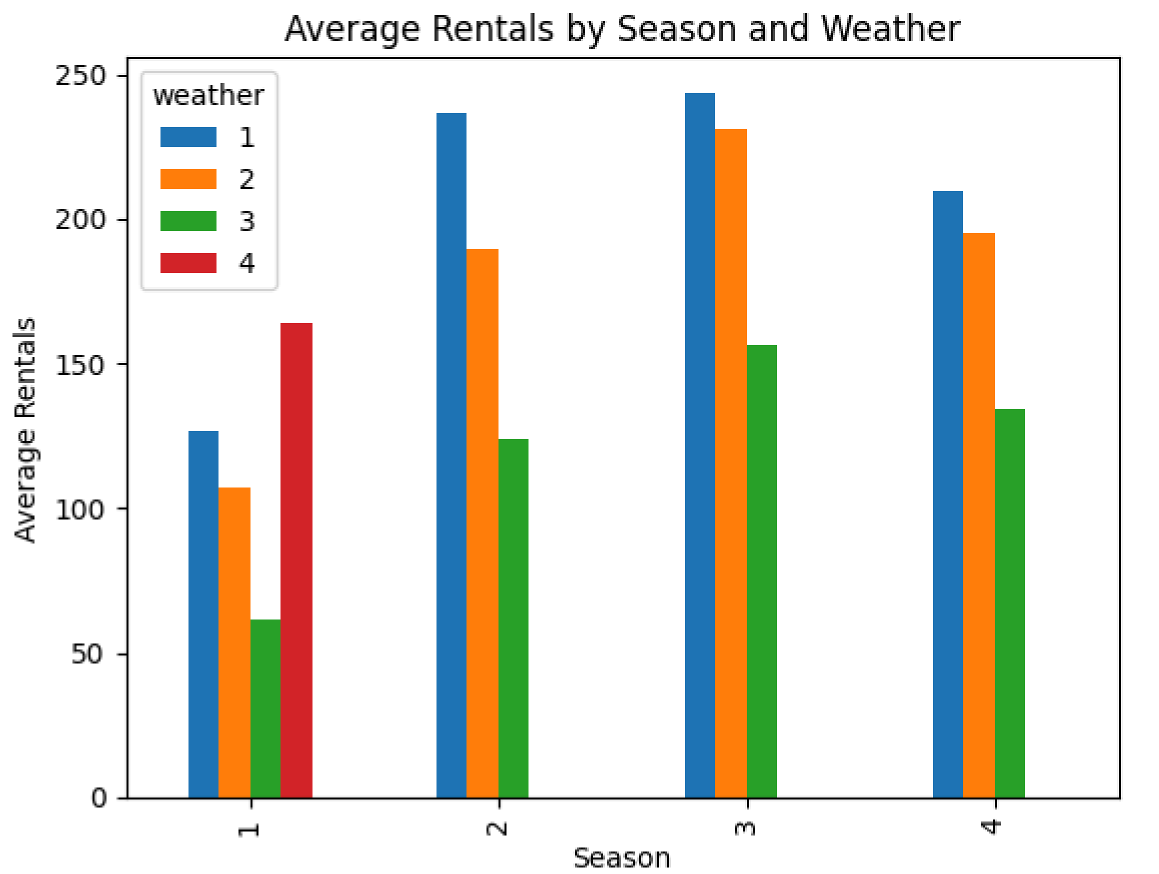
observations

- higher bike rentals(green points) seem to occur within specific temperature and humidity , posssibly in moderate temperature and humidity.
- lower bike rentals(purple points) are scattered more uniformly across ectreme temoerature and humidity values.

In [593…

```
plt.figure(figsize=(10, 6))
df.groupby(['season', 'weather'])['count'].mean().unstack().plot(kind='bar')
plt.title('Average Rentals by Season and Weather')
plt.xlabel('Season')
plt.ylabel('Average Rentals')
plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```



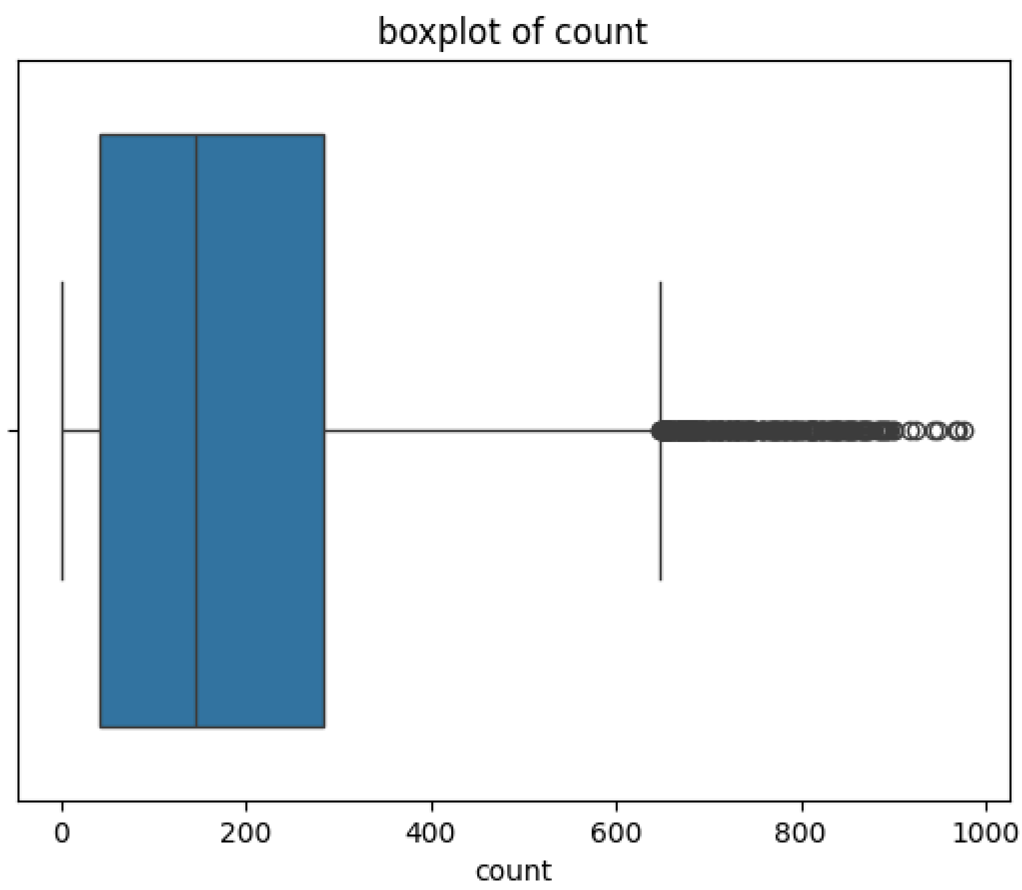
observations

- season(3) fall shows the highest average especially under weather1(clear weather).
- weather 1 consistently leads to higher averahe across all seasons.
- rentals during weather(2)i,e moderate conditions show stronger performance in season 2 and 3.

detecting outliers

In [594…

```
sns.boxplot(x=df["count"])
plt.title("boxplot of count")
plt.show()
```

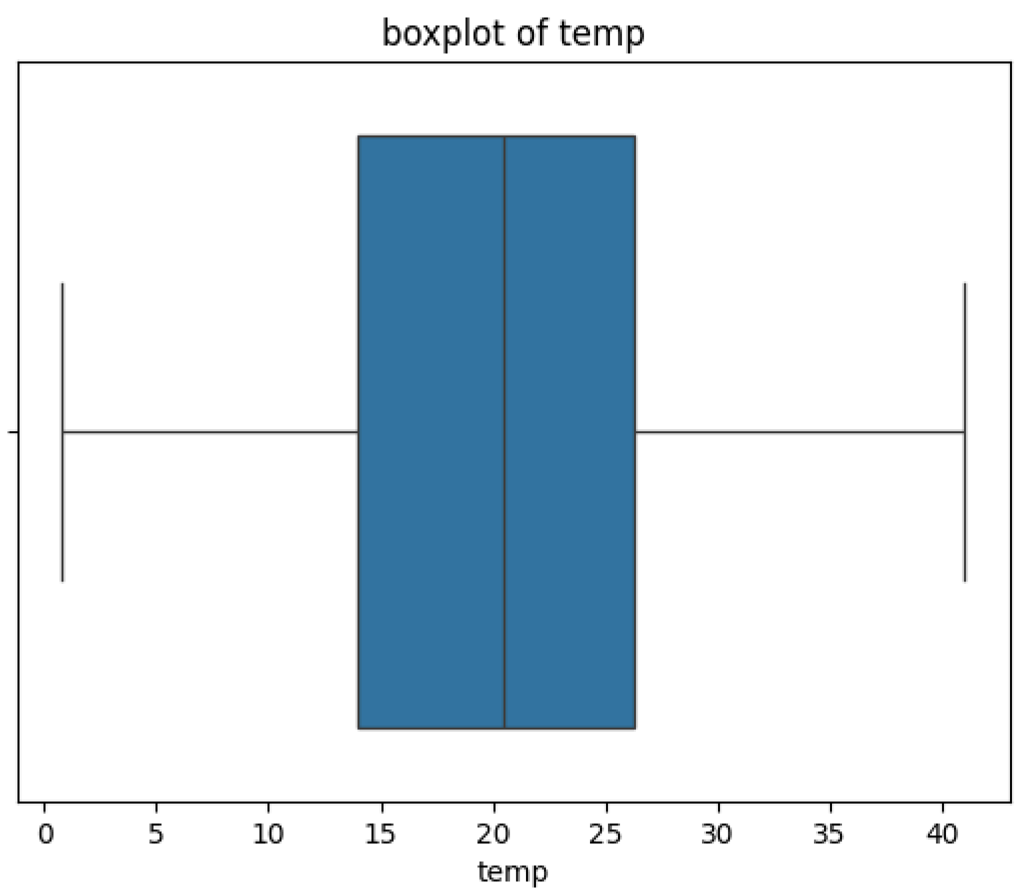


observations:

- most counts of bicycle rides lies upto the range of 300.
- we can see significant outliers after 600.

In [595…

```
sns.boxplot(x=df["temp"])
plt.title("boxplot of temp")
plt.show()
```



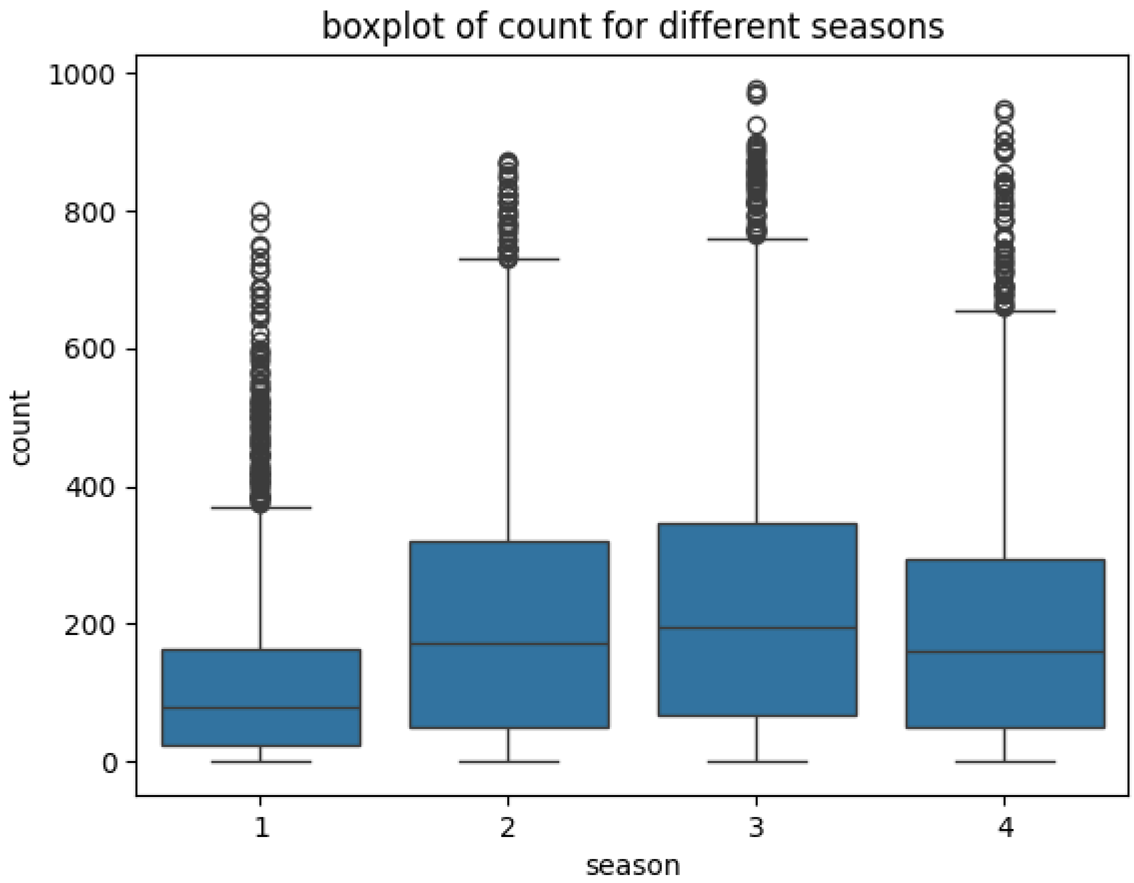
observations

- the temperature mostly lies between 15 to 25 with median temperature being 20.

- this shows most bicycle rides are taken during normaltemperature conditions.

In [596…

```
sns.boxplot(x=df['season'], y=df['count'])
plt.title("boxplot of count for different seasons")
plt.show()
```

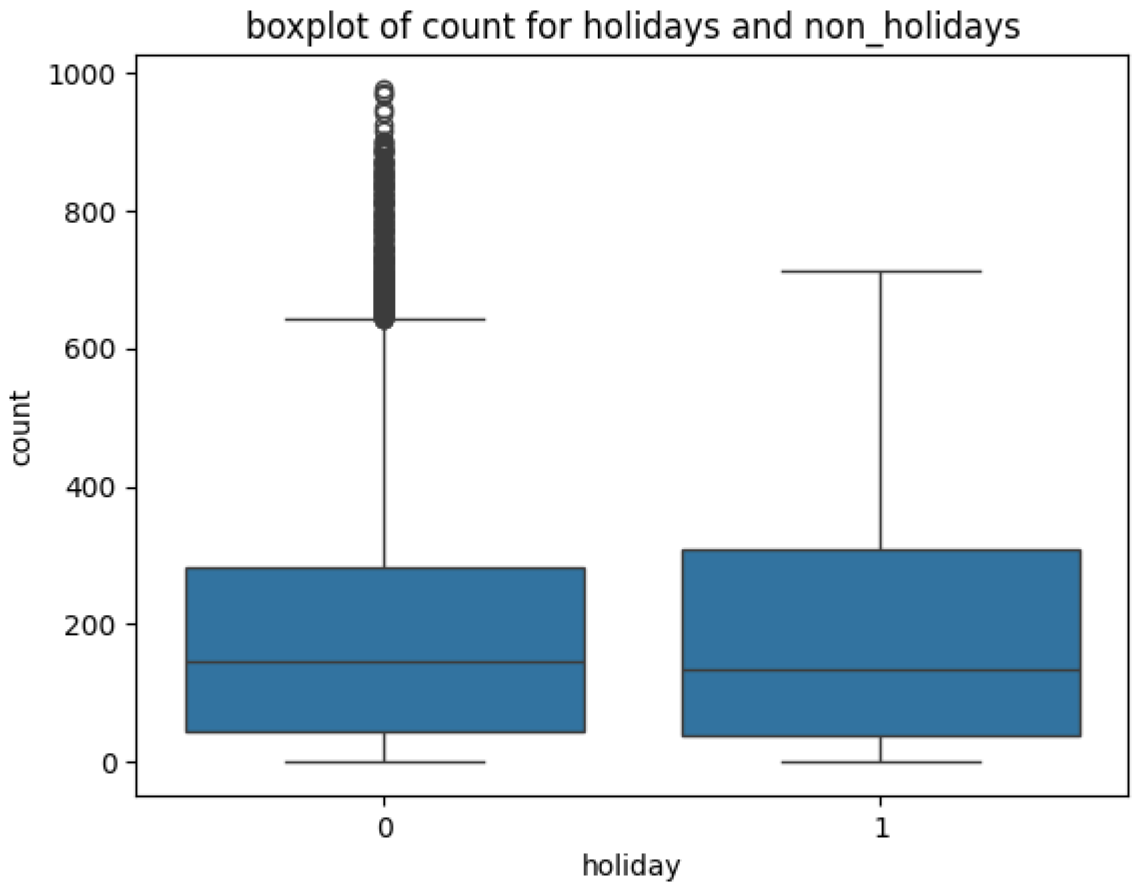


observations

- (1: spring, 2: summer, 3: fall, 4: winter)
- season 1 Median count is the lowest compared to other seasons, indicating that the activity (or metric represented by count) is relatively low during this time.
- The IQR is narrow, meaning there is less variability in the counts during this season.
- The median of summer is higher than in Spring, suggesting an increase in activity.
- the iqr is more in season 2 which shows there is some variablity in counts during this season
- fall has the highest median count,implying it is the peak season for the activity.
- high value outliers are also found during fall season showing exceptionally high activity
- in winter the median is slightly lower than fall season,but better than spring and summer.

In [597…

```
sns.boxplot(x=df['holiday'], y=df['count'])
plt.title("boxplot of count for holidays and non_holidays")
plt.show()
```



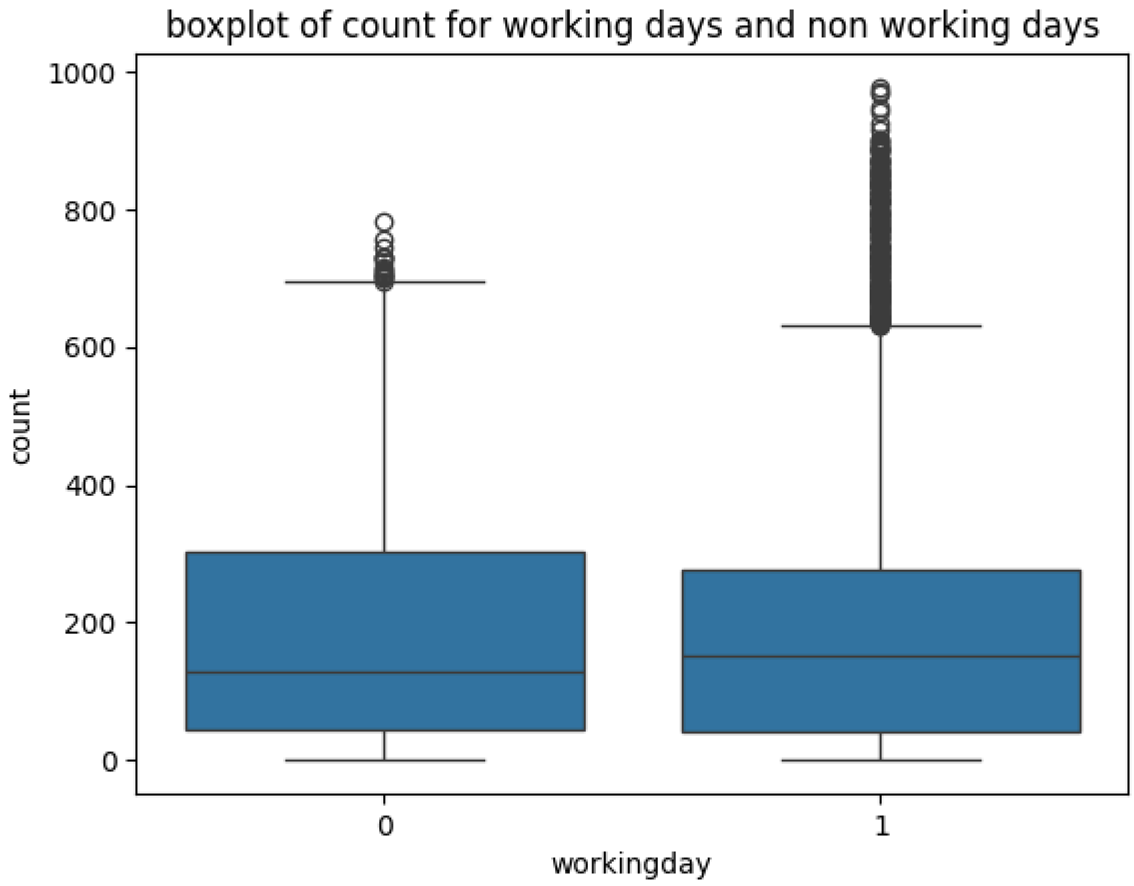
observations

- the median for both holidays and non holidays looks more or less same indicating the activity on both holidyas and non holidays more or less same.

boxplot of working days and counts

In [598…

```
sns.boxplot(x=df['workingday'], y=df['count'])
plt.title("boxplot of count for working days and non working days")
plt.show()
```

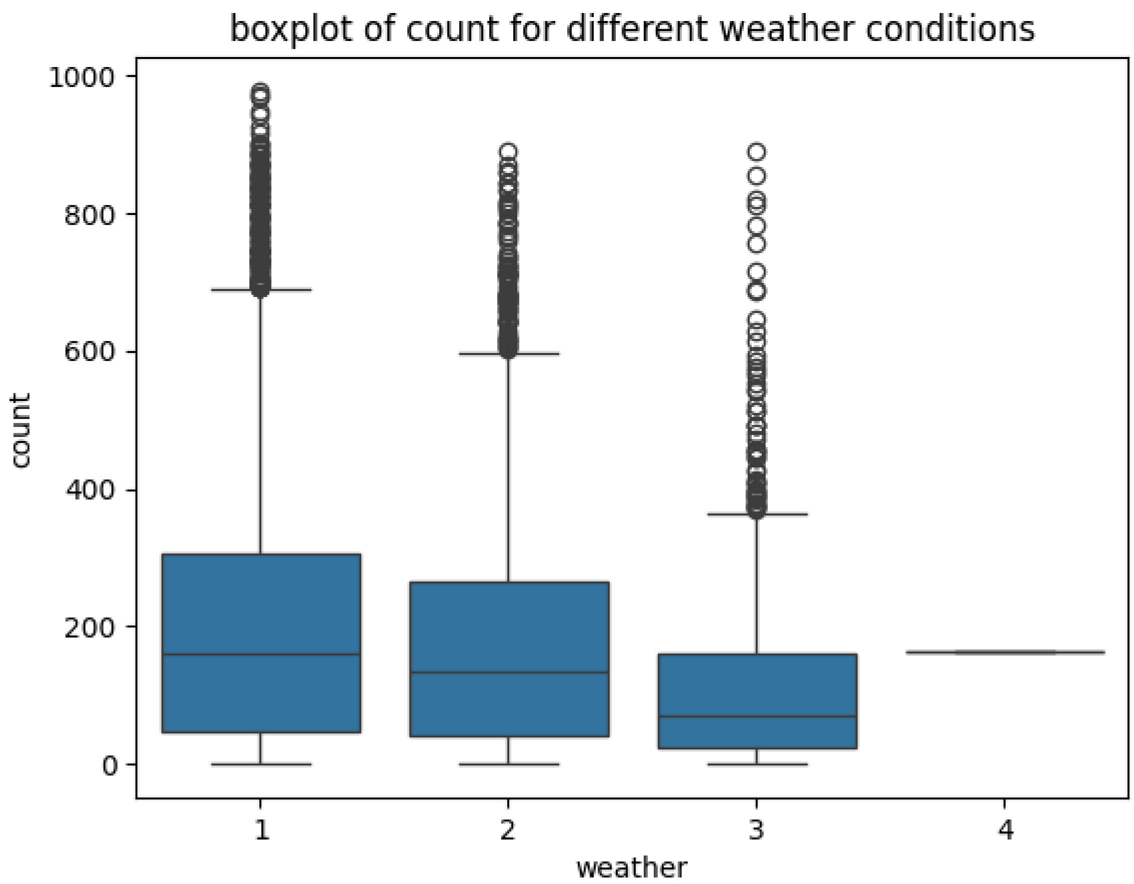


observations

- the median of both working and non working days looks same,however working days shows high amount of outliers showing extensive activity some times.

In [599…

```
sns.boxplot(x=df['weather'], y=df['count'])
plt.title("boxplot of count for different weather conditions")
plt.show()
```



observation

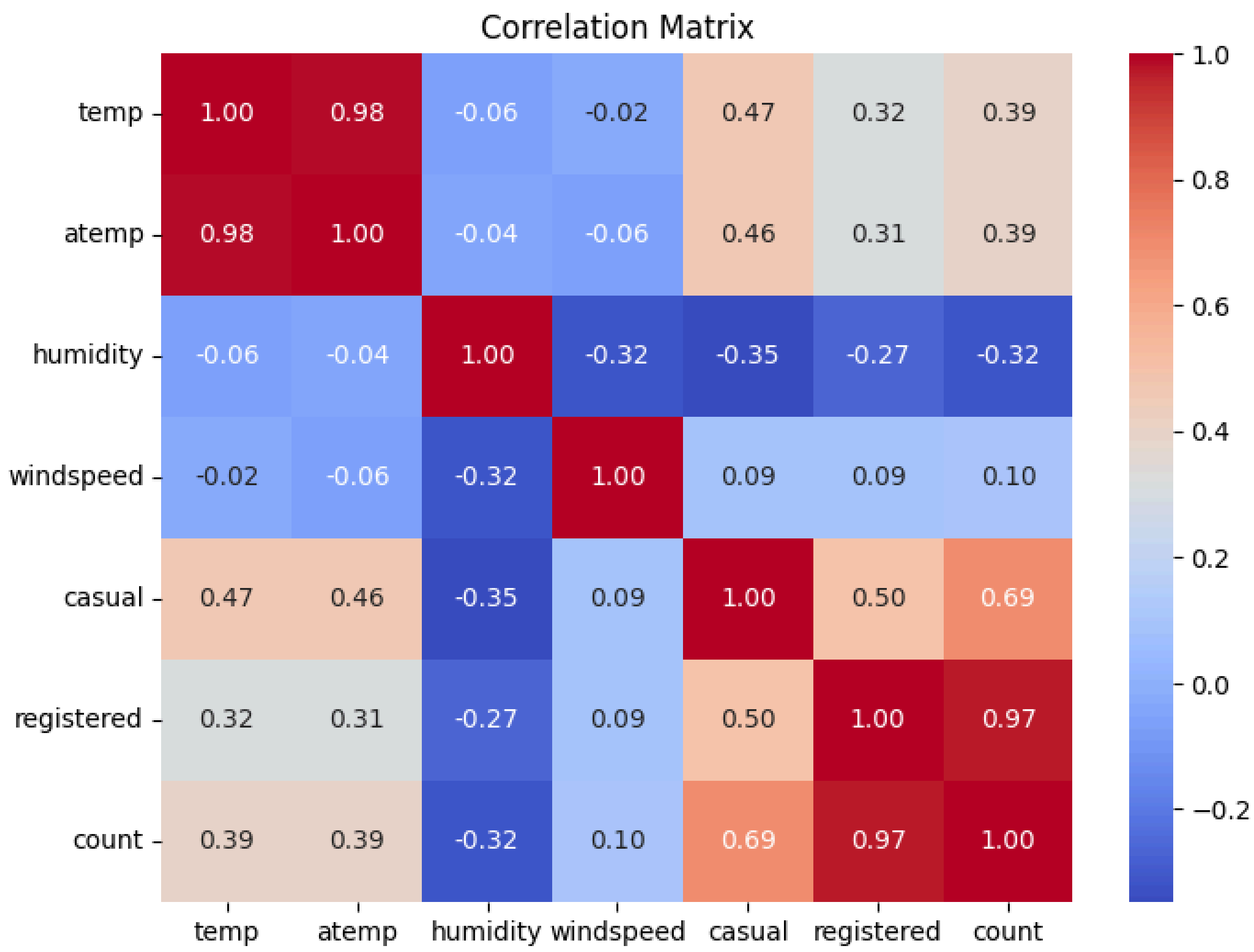
- the median and iqr is highest in 1(clear weather) which shows there is significant usage of bikes during clear weather.
- huge amount of outliers are also found during clear weathwer showing high activity.

In [599…

Q2. Relationship between the Dependent and Independent Variables.

In [600…

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

data = {
    'temp': df["temp"],
    'atemp': df["atemp"],
    'humidity': df["humidity"],
    'windspeed': df["windspeed"],
    'casual': df["casual"],
    'registered': df["registered"],
    'count':df["count"]
}
cor = pd.DataFrame(data)
correlation_matrix = cor.corr()
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap="coolwarm", cbar=True)
plt.title("Correlation Matrix")
plt.show()
```



observations

- registered and count (~0.97): Suggests the total count of rentals is heavily influenced by registered users.
- casual and count (~0.69): Casual users also have a notable but weaker impact on total rentals.
- humidity and casual/count (around -0.32): As humidity increases, the number of rentals decreases.
- Variables like windspeed and temp have almost no correlation with most other variables, indicating limited influence.

3. Check if there is significant difference in bike rides between weekdays and weekends

- null = there is no significant difference between bike rides during weekdays and weekeends
- alternate = there is a significant difference between bike rides during weekdays and weekeends
- we are using ttest for this purpose with significnce level of 5%.

In [601…

```
# the significance level for all test is 0.05
alpha=0.05
```

In [602…

```
# we use ttest for this purpose
working=df[df["workingday"]==1]["count"].sample(1000)
non_working=df[df["workingday"]==0]["count"].sample(1000)
alpha=0.05
```

In [603…
```
from scipy.stats import ttest_ind
tstat,pvalue=ttest_ind(working,non_working,alternative="two-sided")
tstat,pvalue
```

Out[603… (1.0259222175870046, 0.30505243711222996)

In [604…
```
if pvalue<=alpha:
    print("we reject the null hypothesis")
    print(" there is significant difference between bike rides during weekdays and weekeends")
else:
    print("we accept the null hypothesis")
    print("there is no significant difference between bike rides during weekdays and weekeends")
```

```
we accept the null hypothesis
there is no significant difference between bike rides during weekdays and weekeends
```

now we will check weather the no of bike rides greater during week days

- **null: no of bike rides between weakdays and weekends are same**
- **alternate: no of bike rides in weekdays is greater than weekends**

In [605…
```
tstat,pvalue=ttest_ind(working,non_working,alternative="greater")
tstat,pvalue
```

Out[605… (1.0259222175870046, 0.15252621855611498)

In [606…
```
if pvalue<=alpha:
    print("we reject the null hypothesis")
    print("no of bike rides in weekdays is greater than weekends")
else:
    print("we fail to reject hypothesis")
    print("no of bike rides between weakdays and weekends are mostly same")
```

```
we fail to reject hypothesis
no of bike rides between weakdays and weekends are mostly same
```

observations

- The result indicates that there is no statistically significant difference in the number of bike rides during weekdays and weekends.
- This suggests that the bike ride count is fairly consistent regardless of whether it is a working day or not.

recommendations

- Since the bike usage is consistent across weekdays and weekends, operational resources can be evenly distributed throughout the week.
- invest in consistent marketing campaigns for both weekdays and weekends to attract more riders without a specific focus on one.
- explore weather public transport and other factors affect the cosnsitency of bike usage throughout the week

04) Check if the demand of bikes on rent is the same for different Weather conditions?

- **null: the demand for bikes is same for different weather conditions.**
- **alternate: the demand for bikes for different weather changes with different weather conditions**
- we will use one way annova for this purpose

In [607…
```
df["weather"].unique()
```

Out[607… array([1, 2, 3, 4])

In [608…
```
#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
```

In [609…
```
df["weather"].value_counts()
```

Out[609…

weather	count
1	7192
2	2834
3	859
4	1

dtype: int64

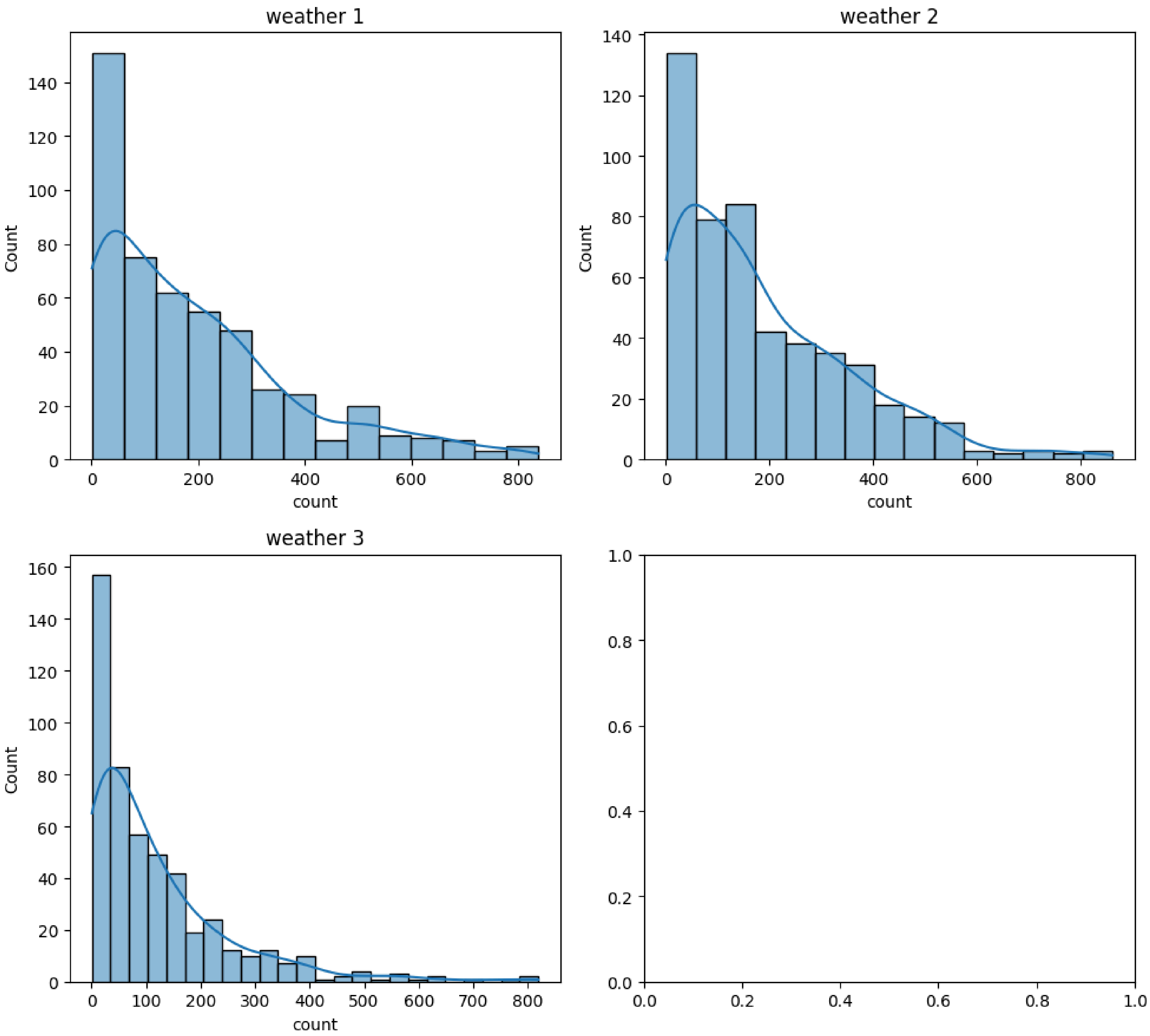
In [610…

```
weather1=df[df["weather"]==1]['count'].sample(500)
weather2=df[df["weather"]==2]['count'].sample(500)
weather3=df[df["weather"]==3]['count'].sample(500)
weather4=df[df["weather"]==4]['count'].sample(1)
```

- checking the assumptions of test.

In [611…

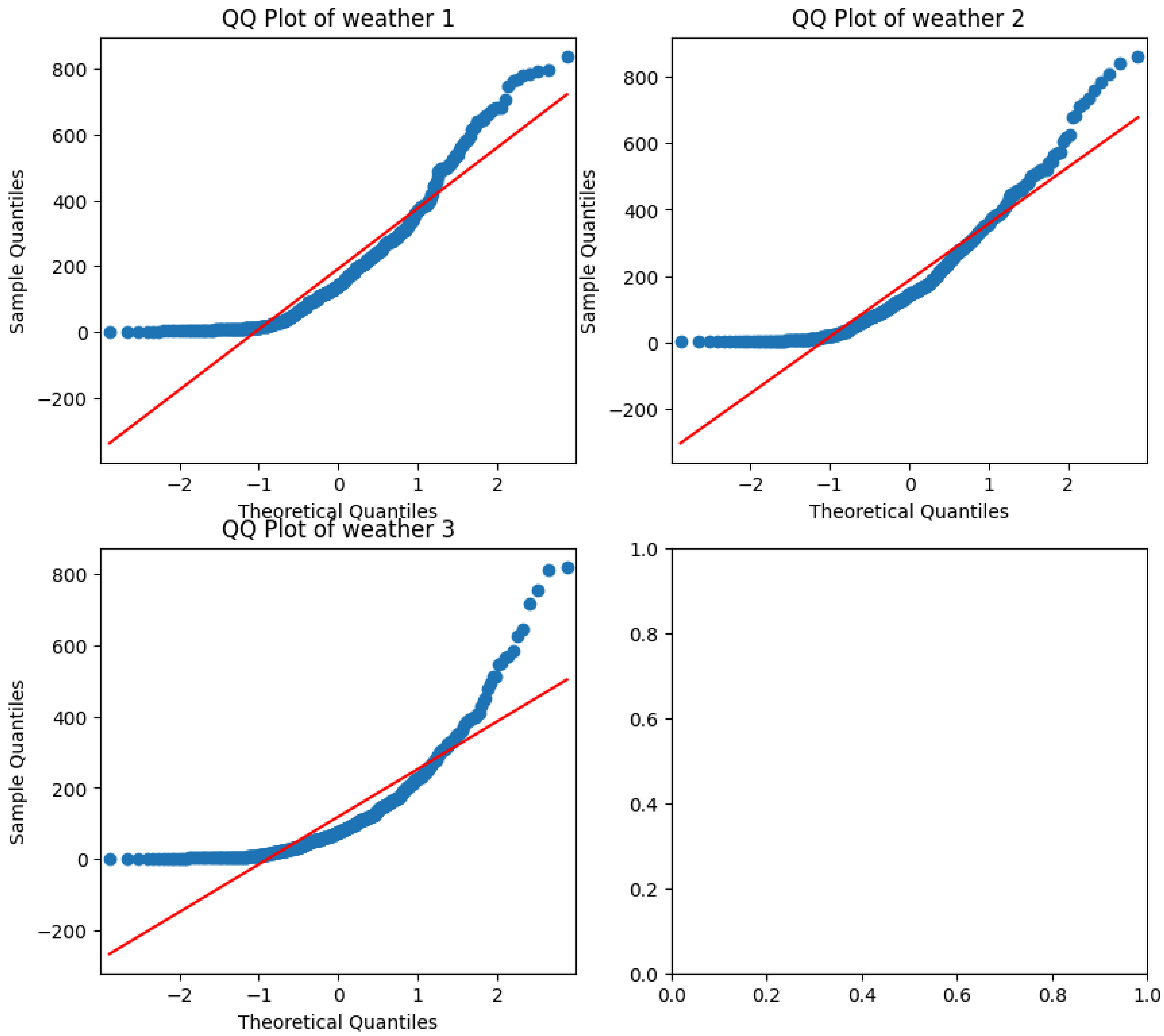
```
import statsmodels.api as sm
all_weathers=[weather1,weather2,weather3]
n_rows, n_cols = 2,2
fig, axes = plt.subplots(n_rows, n_cols, figsize=(10, 9))
axes = axes.flatten()
for idx, data in enumerate(all_weathers):
    sns.histplot(data, kde=True, ax=axes[idx])
    axes[idx].set_title(f"weather {idx+1}")
plt.tight_layout()
```



In [612…

```
import statsmodels.api as sm # import the statsmodels library

all_weather = [weather1,weather2,weather3]
n_rows, n_cols = 2, 2
fig, axes = plt.subplots(n_rows, n_cols, figsize=(10, 9))
```

```
axes = axes.flatten()

for i in range(len(all_weather)):
    sm.qqplot(all_weather[i], line="s", ax=axes[i])
    axes[i].set_title(f"QQ Plot of weather {i + 1}")
```



shapiro_wilk normalacy test

In [613…

```
#null:dist is normal
#alternate:dist is not normal
from scipy.stats import shapiro
for i in all_weathers:
    stat,pvalue=shapiro(i)
    print(f"stat:{stat},pvalue:{pvalue}")

if pvalue<alpha:
  print("the dist is not normal")
else:
  print("the dist is normal")
```

```
stat:0.870959445287386,pvalue:6.285021522731182e-20
stat:0.8880083775753156,pvalue:1.3496833223580858e-18
stat:0.7832749426856567,pvalue:3.160425489077075e-25
the dist is not normal
```

checking of equality of varience

In [614…

```
from scipy.stats import levene
lstat,pvalue=levene(weather1,weather2,weather3)
lstat,pvalue

if pvalue<alpha:
    print("the varience is not equal")
else:
    print("the varience is equal")
```

```
the varience is not equal
```

hence the assumptions of normality and varience fail we use kw test

In [615…
```
from scipy.stats import kruskal
stat,pvalue=kruskal(weather1,weather2,weather3)
print(f"k_stat:{stat},p_value:{pvalue}")

if pvalue<alpha:
    print("we reject the null hypothesis")
    print("the demand for bikes is not the same for different weather conditions")
else:
    print("we fail to reject null hypothesis")
    print("the demand for bikes is the same for different weather conditions")
```

```
k_stat:58.548962162972586,p_value:1.9331003779032836e-13
we reject the null hypothesis
the demand for bikes is not the same for different weather conditions
```

observations

- the p value is very low hence we reject null hypo with confidence.
- this shows weather plays significant role in the usage of bycicles.
- The variability in demand across weather conditions highlights the importance of weather as a critical factor influencing bike usage.

recommendations

- during favorable weather conditions increase the number of bikes available to ensure efficient catering to demand.
- take feedback or organise feedback sessions from customers regarding how they percieve bike rides during bad weather.
- offer incentives and discounts for using the bycicles even in the less favorable weather conditions.
- Plan for potential surge in bike usage on clear days by making the availablity of bikes/bycicles at popular locations.

5) Check if the demand of bikes on rent is the same for different Seasons?

In [616…
```
df["season"].unique() #season: season (1: spring, 2: summer, 3: fall, 4: winter)
```

Out[616…
```
array([1, 2, 3, 4])
```

In [617…
```
spring=df[df["season"]==1]["count"].sample(1000)
summer=df[df["season"]==2]["count"].sample(1000)
fall=df[df["season"]==3]["count"].sample(1000)
winter=df[df["season"]==4]["count"].sample(1000)
```

assumptions

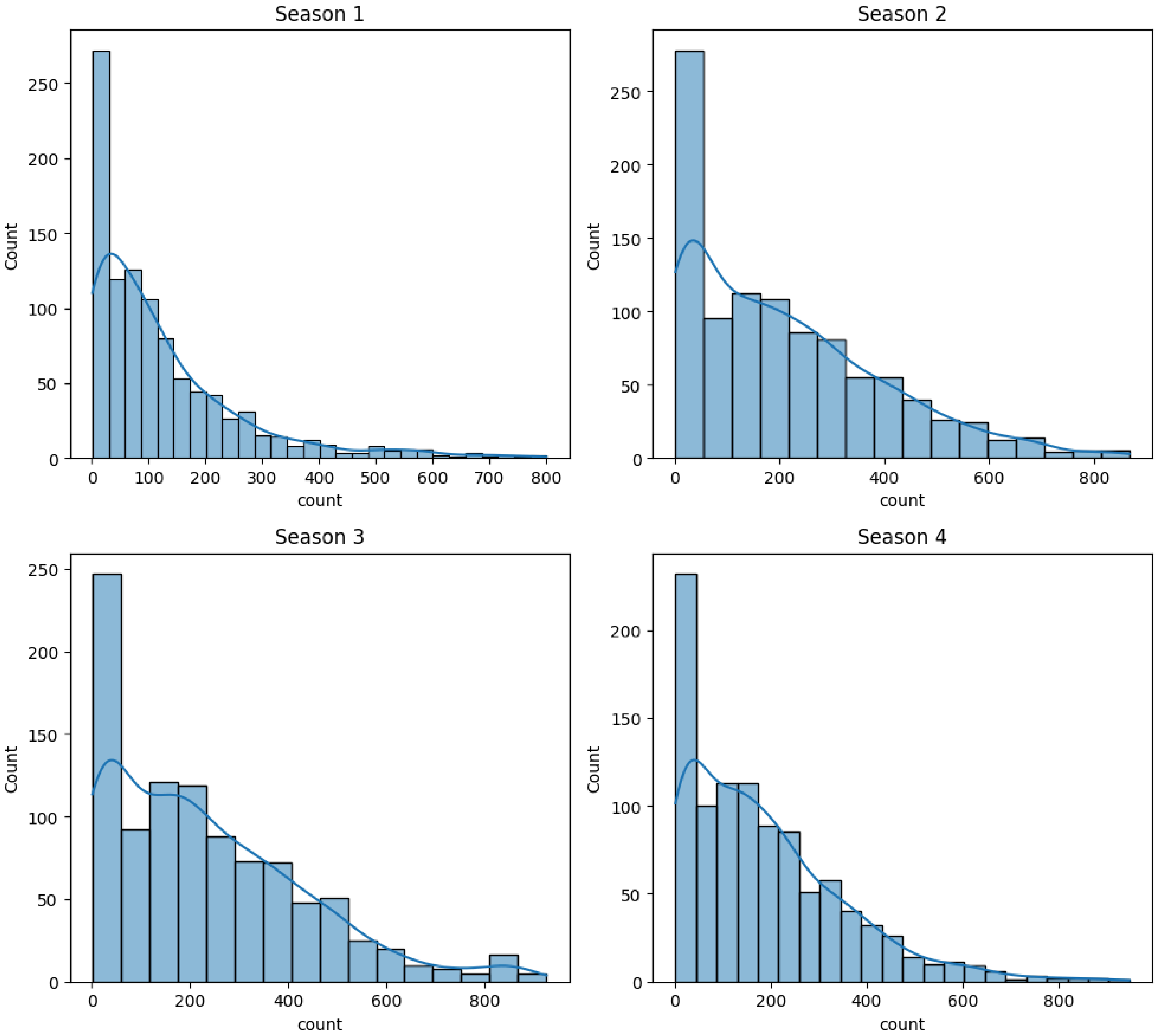
- null hypothesis= demand of bikes on rent is the same for different Seasons
- alternate hypothesis= demand of bikes on rent is not the same for different Seasons

test assumptions

1. data must be normal
2. its should be independent
3. equality of varience

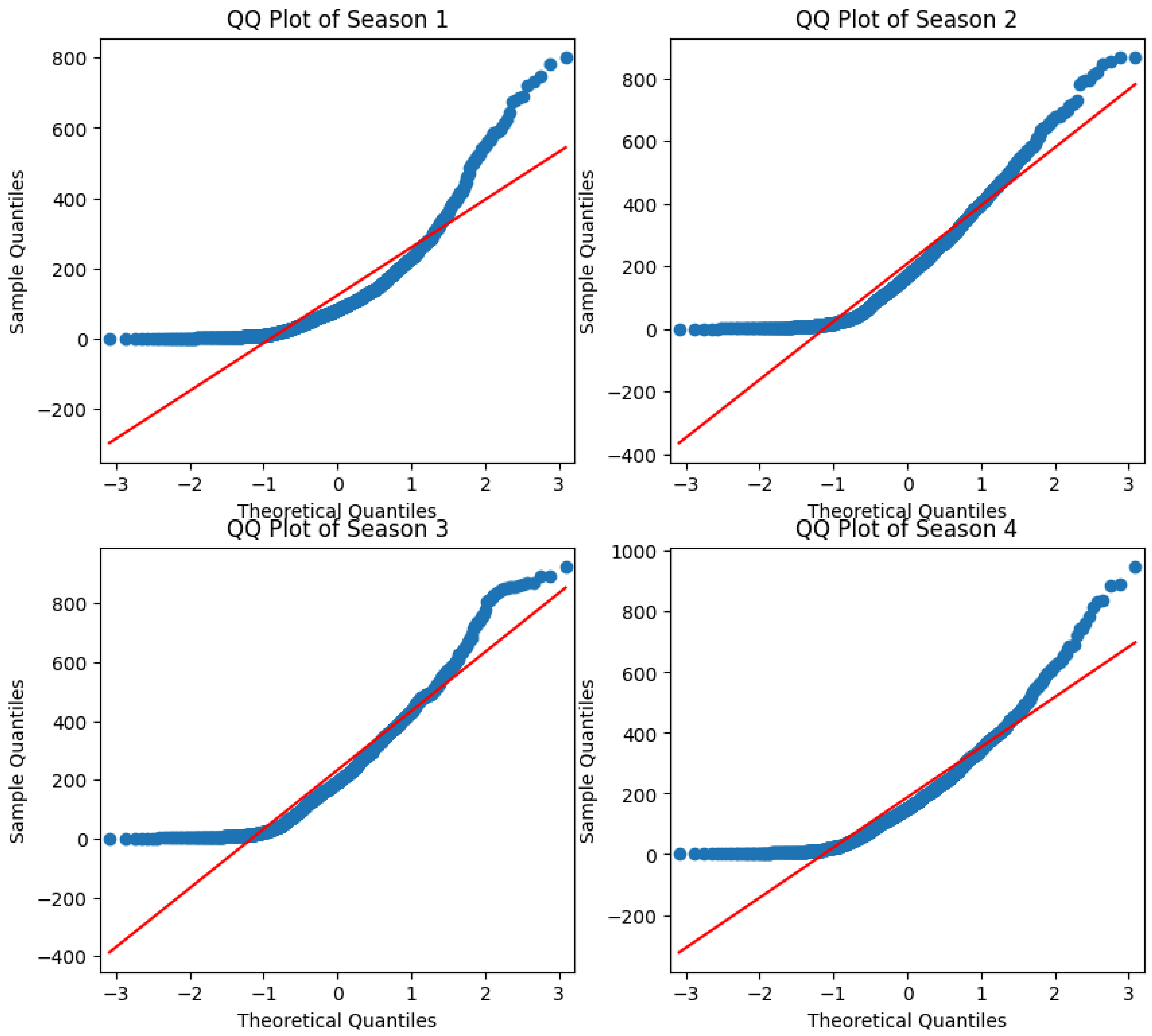
normality test

In [618…
```
import statsmodels.api as sm
all_seasons=[spring,summer,fall,winter]
n_rows, n_cols = 2,2
fig, axes = plt.subplots(n_rows, n_cols, figsize=(10, 9))
axes = axes.flatten()
for idx, data in enumerate(all_seasons):
    sns.histplot(data, kde=True, ax=axes[idx])
    axes[idx].set_title(f"Season {idx+1}")
plt.tight_layout()
```



In [619…

```
import statsmodels.api as sm # import the statsmodels library

all_seasons = [spring,summer,fall,winter]
n_rows, n_cols = 2, 2
fig, axes = plt.subplots(n_rows, n_cols, figsize=(10, 9))
axes = axes.flatten()

for i in range(len(all_seasons)):
    sm.qqplot(all_seasons[i], line="s", ax=axes[i])
    axes[i].set_title(f"QQ Plot of Season {i + 1}")
```



shapiro wilk test of normalacy

In [620…

```
from scipy.stats import shapiro
for i in all_seasons:
    stat,pvalue=shapiro(i)
    print(f"stat:{stat},pvalue:{pvalue}")

if pvalue<alpha:
  print("the dist is not normal")
else:
  print("the dist is normal")
```

```
stat:0.7952564791694602,pvalue:1.1296245961015156e-33
stat:0.9056356608770569,pvalue:2.270698301570405e-24
stat:0.9104476205382142,pvalue:8.464922461760708e-24
stat:0.8940965306940302,pvalue:1.177695575848917e-25
the dist is not normal
```

as we can see the underlying distribution of data is not normal

levenes test for equality of varience

In [621…

```
# equality of varience test
from scipy.stats import levene
stat,pvalue=levene(spring,summer,fall,winter)
stat,pvalue

if pvalue<=alpha:
    print("the varience is not equal")
else:
    print("the varience is equal")
```

```
the varience is not equal
```

so we will go ahead with kw test

In [622…
```
from scipy.stats import kruskal
stat,pvalue=kruskal(spring,summer,fall,winter)
stat,pvalue

if pvalue<alpha:
    print("we reject the null hypothesis")
    print("demand of bikes on rent is not the same for different Seasons")
else:
    print("we accept the null hypothesis")
    print("demand of bikes on rent is the same for different Seasons")
```

```
we reject the null hypothesis
demand of bikes on rent is not the same for different Seasons
```

however we will also try to do it with annova even though its not relaiable if dist is not normal

In [623…
```
from scipy.stats import f_oneway
stat,pvalue=f_oneway(spring,summer,fall,winter)
stat,pvalue
if pvalue<alpha:
    print("we reject the null hypothesis")
    print("demand of bikes on rent is not the same for different Seasons")
else:
    print("we accept the null hypothesis")
    print("demand of bikes on rent is the same for different Seasons")
```

```
we reject the null hypothesis
demand of bikes on rent is not the same for different Seasons
```

observations

- From the earlier boxplot, demand appeared highest in summer and lowest in winter. These patterns align with typical outdoor activity trends.
- The rejection of the null hypothesis shows that the demand for bikes varies significantly across different seasons.\
- Seasonality significantly impacts bike rentals, with certain seasons (e.g., summer or spring) likely encouraging outdoor recreation and travel, while adverse seasons (e.g., winter or rainy periods) see reduced demand.

recommendations

- increase the bicycle availablity during peak seasons to cater demand.
- provide people with incentives and discounts during less demand season to sustain demand.
- provide season appropriate gears along with bikes to enhance customer loyality and demand.

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

we use chi_square test for this purpose

setting of null and alternate hypothesis

In [624…
```
h0="the weather conditions are independent of season"
hA="the weather conditions are significantly dependent on season"
```

In [625…
```
# contigency table
contingency_table=pd.crosstab(df["season"],df["weather"])
contingency_table
```

Out[625…

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

In [626…
```
from scipy.stats import chi2_contingency
stat,pvalue,dof,expected=chi2_contingency(contingency_table)
stat,pvalue
```

Out[626…
```
(49.158655596893624, 1.549925073686492e-07)
```

In [627…
```
if pvalue<alpha:
    print("we reject the null hypothesis")
    print(hA)
else:
    print("we fail to reject null hypothesis")
    print(h0)
```

```
we reject the null hypothesis
the weather conditions are significantly dependent on season
```

observations

we can see from the above that the weather conditions are significantly dependent on season

- The rejection of the null hypothesis suggests that weather conditions are significantly dependent on the season.
- The demand for bike rentals may fluctuate based on these seasonal weather conditions, such as lower demand during cold or rainy weather and higher demand during sunny or mild weather.

recommendations

- Ensure bike availability aligns with the weather for each season (e.g., increasing bike availability during dry, sunny months).
- Develop and promote season-specific biking gear such as rain covers for bikes, weather-appropriate clothing options etc.

In [627…

General insights

- Average commute and usage: the average usage during workdays and holidays shows very slight difference, in other words both are nearly same.
- seasonwise bike rentals: the average usage of bikes is significant during the fall months followed by summer,winter,spring.
- impact of weather: there is a significant impact of weather on usage of bike rentals, adverse weather conditions leads to decrease in the bike rentals, it can be seen people tend to rent bike significantly on clear weather conditions.
- registration and casual users: the service is significantly used by the users who are registered when compared to the customers who are casual users.

Recommendations:

- **optimise bike management acc to weather**:since weather has a clear impact on bike rental usage, consider adjusting the bikes based on seasonal weather forecasts. For instance you could allocate more bikes to locations that tend to see higher demand during clear weather days and reduce bike availability during bad weather (e.g., rain, snow).
- This would ensure that bikes are available when demand is high and prevent over-supply during poor weather conditions
- **enhance user experience of registered users**: by providing better discounts, push notifications when new bikes available, and provide bike reservation options to satisfy customer,this would in turn improve customer retension.
- **attract casual users:** attract casual users to register by providing better incentives such as one free ride after registration, better discounts,and future dicount promises through coupons etc.
- **Seasonal Promotions:** Given that bike rentals peak in the fall and summer, consider running seasonal promotions or discounts during the off-peak months (winter and spring) to maintain consistent rental rates throughout the year.
- **weather forecasting:**Add a weather forecasting feature to the app that alerts users about ideal biking conditions, such as clear skies or mild temperatures. You could send push notifications or reminders to registered users on days when the weather is perfect for biking, helping them make the most of good riding days. By focusing on weather trends that encourage biking, you can boost rentals on those optimal days.

In [ ]: