In [493…]
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import chi2_contingency, ttest_ind,f_oneway, kruskal, shapiro, ttest
```

In [494…]
```
df=pd.read_csv("bike_sharing.csv")
```

In [495…]
```
df.head()
```

Out[495]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	regis
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 [496…]
```
df.shape
```

Out[496]:
```
(10886, 12)
```

In [497…]
```
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 [498…]
```
df.describe()
```

Out[498]:

	season	holiday	workingday	weather	temp	atemp	hu
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000	10886.0
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.8
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.2
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	0.0
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	47.0
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	62.0
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	77.0
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	100.0

In [499…
```
df.dtypes
```

Out[499]:
```
datetime       object
season          int64
holiday         int64
workingday      int64
weather         int64
temp          float64
atemp         float64
humidity        int64
windspeed     float64
casual          int64
registered      int64
count           int64
dtype: object
```

In [500…
```
df.isna().sum()
```

Out[500]:
```
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
```

In [501…
```
df.nunique()
```

Out[501]:
```
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 [502…
```
df.duplicated().sum()
```

Out[502]:
```
0
```

In [503…
```
df["season"].value_counts()
```

Out[503]:
```
4    2734
2    2733
3    2733
1    2686
Name: season, dtype: int64
```

In [504…
```
df["holiday"].value_counts()
```

Out[504]:
```
0    10575
1      311
Name: holiday, dtype: int64
```

In [505…
```
df["workingday"].value_counts()
```

Out[505]:
```
1    7412
0    3474
Name: workingday, dtype: int64
```

In [506…
```
df["weather"].value_counts()
```

Out[506]:
```
1    7192
2    2834
3     859
4       1
Name: weather, dtype: int64
```

In [507…
```
df["temp"].value_counts()
```

Out[507]:
```
14.76    467
26.24    453
28.70    427
13.94    413
18.86    406
22.14    403
25.42    403
16.40    400
22.96    395
27.06    394
24.60    390
12.30    385
21.32    362
17.22    356
13.12    356
29.52    353
10.66    332
18.04    328
20.50    327
30.34    299
9.84     294
15.58    255
9.02     248
31.16    242
8.20     229
27.88    224
23.78    203
32.80    202
11.48    181
19.68    170
6.56     146
33.62    130
5.74     107
7.38     106
31.98     98
34.44     80
35.26     76
4.92      60
36.90     46
4.10      44
37.72     34
36.08     23
3.28      11
0.82       7
38.54      7
39.36      6
2.46       5
1.64       2
41.00      1
Name: temp, dtype: int64
```

In [508…
```
#converting datetime column into date & time format

df["datetime"]=pd.to_datetime(df["datetime"])
df["datetime"].dtype
```

Out[508]:
```
dtype('<M8[ns]')
```

In [509…
```
#segregating year and month from datetime column

df["year"]=df["datetime"].dt.year
df["month"]=df["datetime"].dt.month
```

In [510…
```
df.head()
```

Out[510]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	regis
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 [511…
```
#converting numeric to categoric

df["season"] = df["season"].replace({1: "spring", 2: "summer", 3: "fall", 4: "winter"
df["holiday"] = df["holiday"].replace({1: "Yes", 0: "No"})
df["workingday"] = df["workingday"].replace({1: "Yes" , 0: "No"})
df["weather"] = df["weather"].replace({1: "Clear",2: "Mist",3: "Light_Rain" ,4: "Heav
df["month"] = df["month"].replace({1: "January", 2: "February", 3: "March", 4: "April
                                   7: "July", 8: "August",9: "September", 10: "October", 11
```

In [512…
```
cat_cols=["season", "holiday", "workingday", "weather", "month"]
for i in cat_cols:
    df[i]=df[i].astype("category")
```

In [513…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 14 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   datetime    10886 non-null  datetime64[ns]
 1   season      10886 non-null  category
 2   holiday     10886 non-null  category
 3   workingday  10886 non-null  category
 4   weather     10886 non-null  category
 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
 12  year        10886 non-null  int64
 13  month       10886 non-null  category
dtypes: category(5), datetime64[ns](1), float64(3), int64(5)
memory usage: 819.7 KB
```

In [514…
```
#Dataset is pretty clean and we don't have any missing values.
```

In [515…
```
#Now we will do univariate and bi-variate analysis to see about the distribution of d
```

Univariate Analysis

univariate analysis for qualitative attributes

In [516…
```
df_new=df[["season", "holiday", "workingday", "weather"]].melt()
df_new.groupby(['variable', 'value'])[['value']].count() / len(df)*100
```

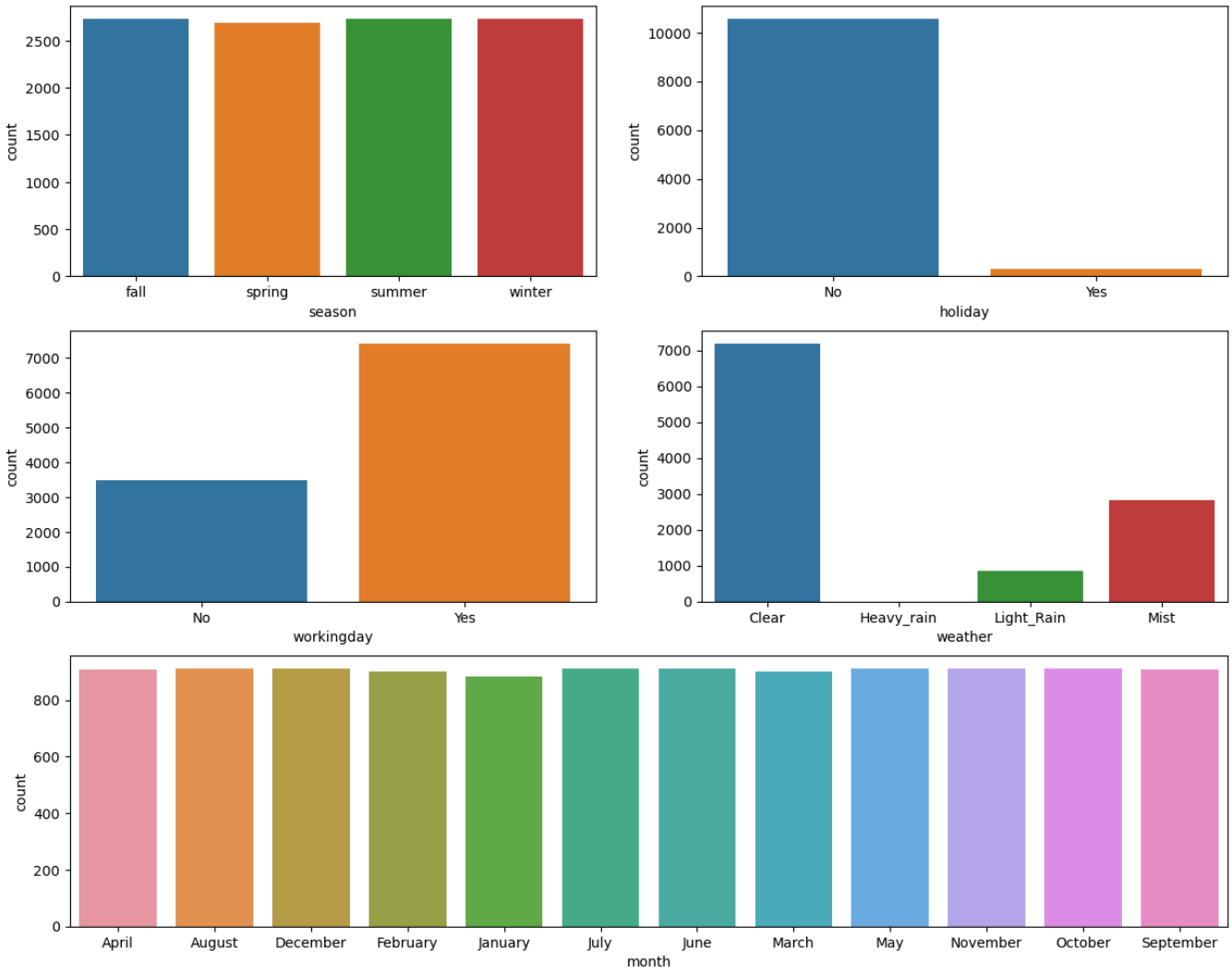
Out[516]:

		value
variable	**value**	
holiday	**No**	97.143120
	Yes	2.856880
season	**fall**	25.105640
	spring	24.673893
	summer	25.105640
	winter	25.114826
weather	**Clear**	66.066507
	Heavy_rain	0.009186
	Light_Rain	7.890869
	Mist	26.033437
workingday	**No**	31.912548
	Yes	68.087452

In [517…
```
cat_cols1=["season", "holiday", "workingday", "weather"]
```

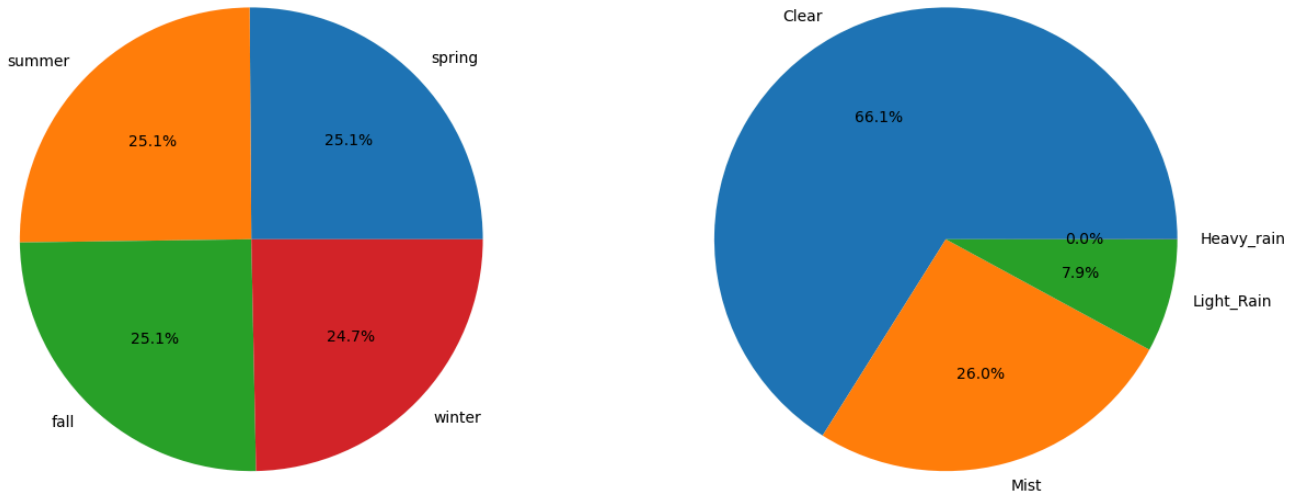
In [518…
```
fig = plt.figure(figsize=(15,12))
for i,col in enumerate(cat_cols1,1):
    plt.subplot(3,2,i)
    sns.countplot(x=col,data=df)
plt.subplot(3,1,3)
sns.countplot(x="month",data=df)
fig.suptitle('Univariate Analysis/Qualitative',fontsize=20)
plt.show()
```

Univariate Analysis/Qualitative



In [519…

```
fig = plt.figure(figsize=(15,12))
plt.subplot(1,2,1)
plt.pie(df["season"].value_counts(),labels=df["season"].unique(),autopct='%1.1f%%')
plt.subplot(1,2,2)
plt.pie(df["weather"].value_counts(),labels=df["weather"].unique(),autopct='%1.1f%%')
plt.show()
```

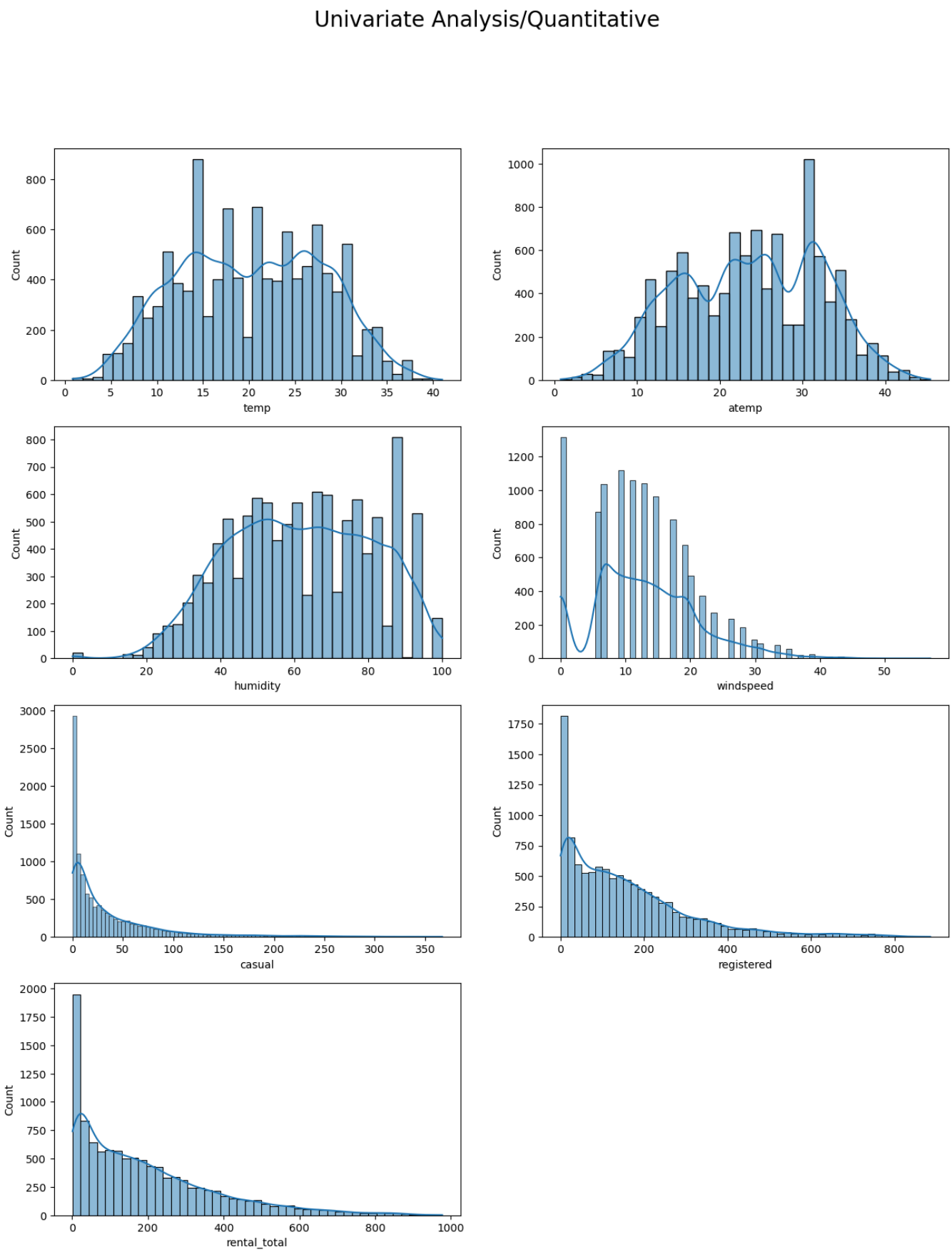


univariate analysis for quantitative attributes

In [520…
```
df.rename(columns={"count" : "rental_total"}, inplace=True)
```

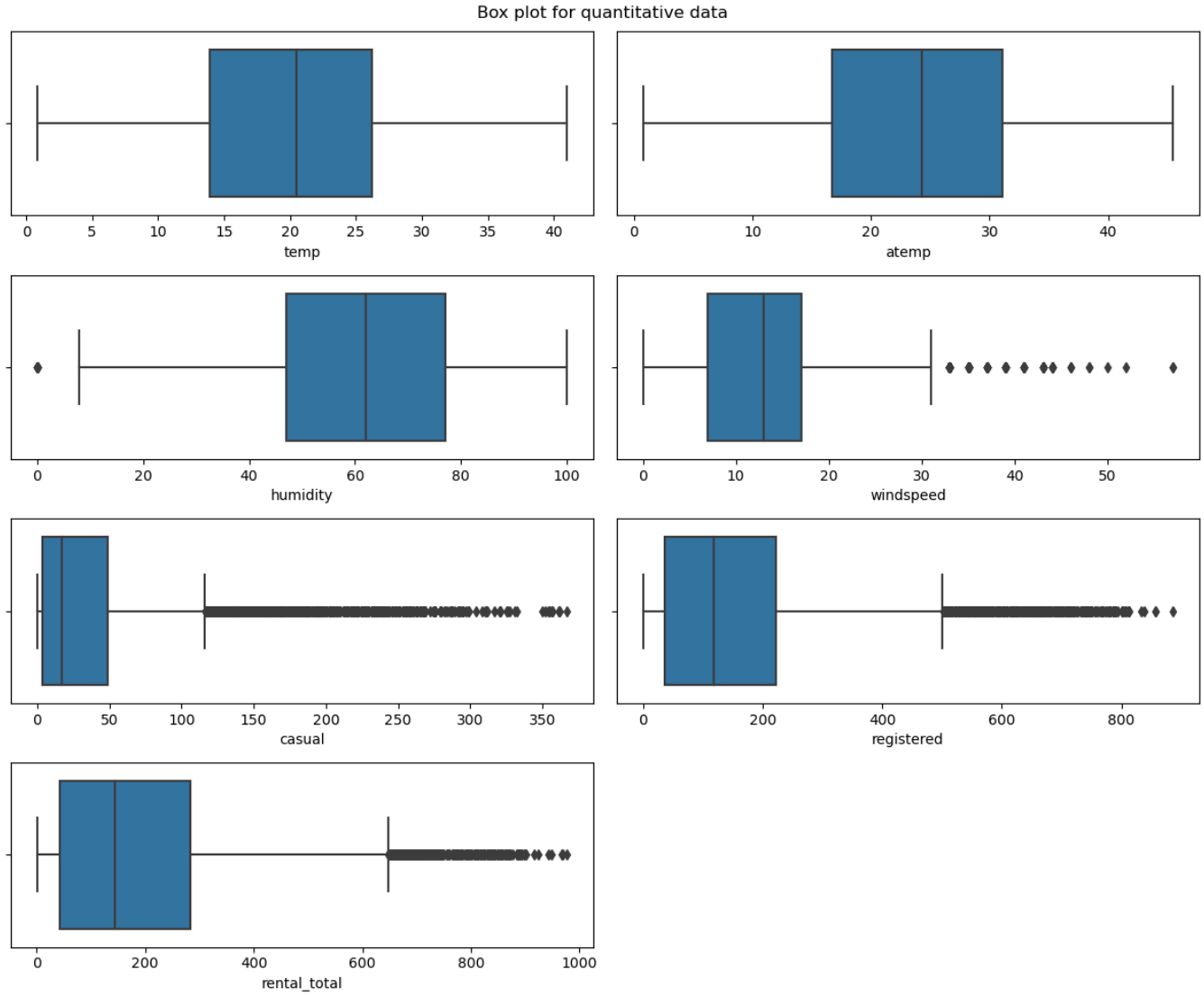
In [521…
```
num_cols=["temp", "atemp","humidity","windspeed","casual","registered","rental_total"
```

In [522…
```
fig = plt.figure(figsize=(15,18))
for i,col in enumerate(num_cols,1):
    plt.subplot(4,2,i)
    sns.histplot(x=col,data=df,kde=True)
fig.suptitle("Univariate Analysis/Quantitative",fontsize=20)
plt.show()
```



In [523…
```
#boxplot for outlier detetction
```

In [524…
```
fig = plt.figure(figsize=(12,10))
for i,col in enumerate(num_cols,1):
    plt.subplot(4,2,i)
    sns.boxplot(x=col,data=df)
fig.suptitle("Box plot for quantitative data")
plt.tight_layout()
plt.show()
```

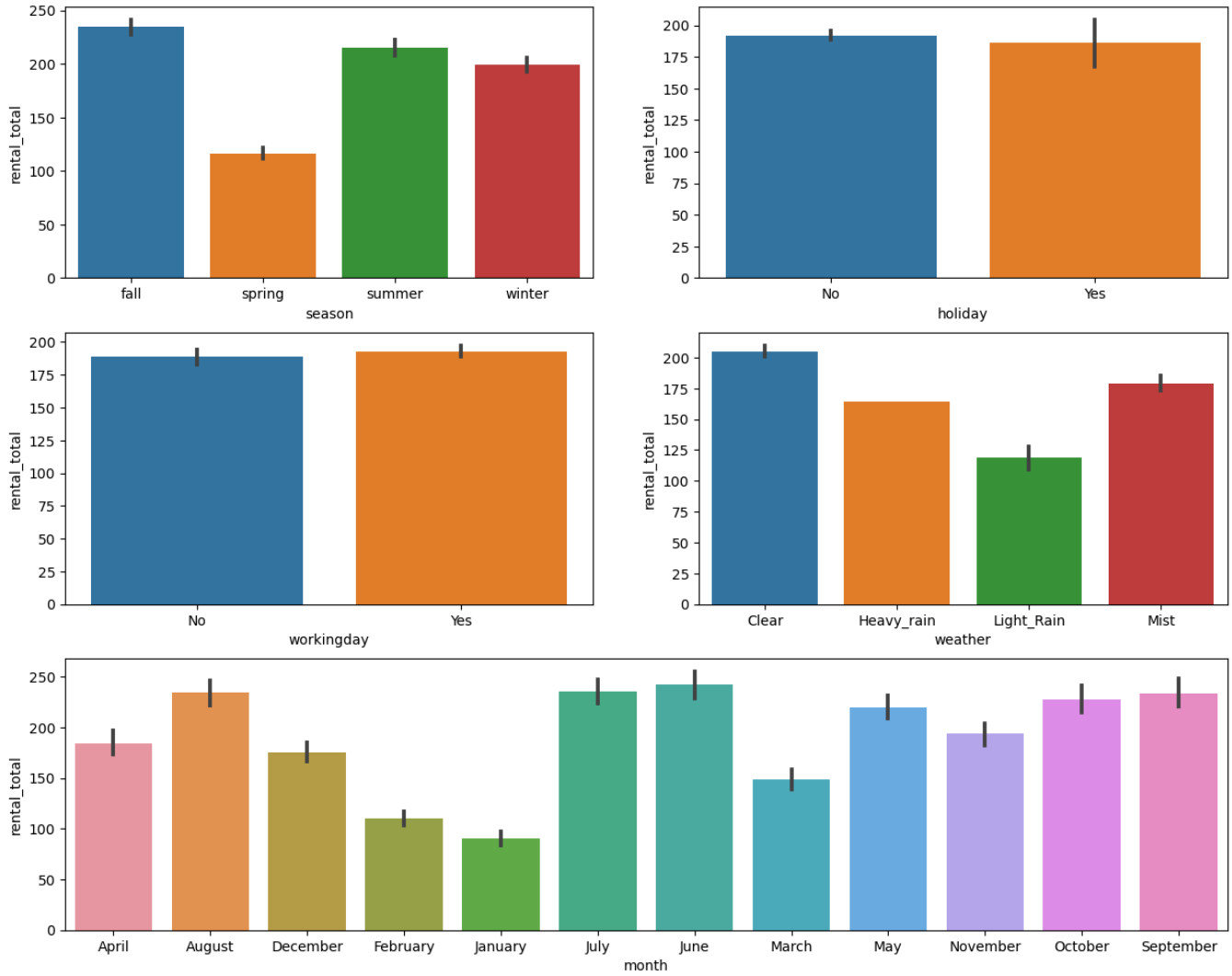


From above boxplot we can see that windspeed,casual,registered and rentat_total have high amount of outliers,humidity have very few otlier and temp & atemp have no visible outlier.

Bivariate Analysis

In [525…
```
fig = plt.figure(figsize=(15,12)).suptitle('BIivariate Analysis/Qualitative',fontsize
for i,col in enumerate(cat_cols1,1):
    plt.subplot(3,2,i)
    sns.barplot(x=col,y="rental_total", data=df)
plt.subplot(3,1,3)
sns.barplot(x="month",y="rental_total", data=df)
plt.tight_layout
plt.show()
```

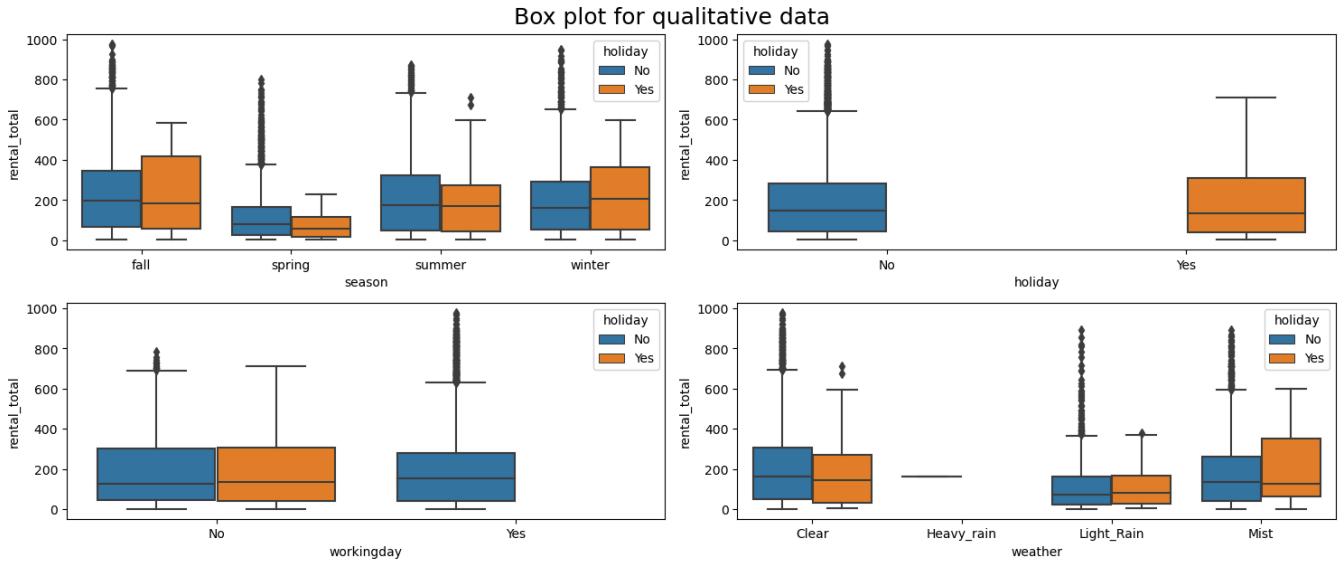
BIivariate Analysis/Qualitative



Multivariate Analysis

In [526…

```
fig = plt.figure(figsize=(15,12)).suptitle("Box plot for qualitative data",fontsize=1
for i,col in enumerate(cat_cols1,1):
    plt.subplot(4,2,i)
    sns.boxplot(x=col,y="rental_total",hue="holiday", data=df)
plt.tight_layout()
plt.show()
```



Relationship between the Dependent and Independent Variables

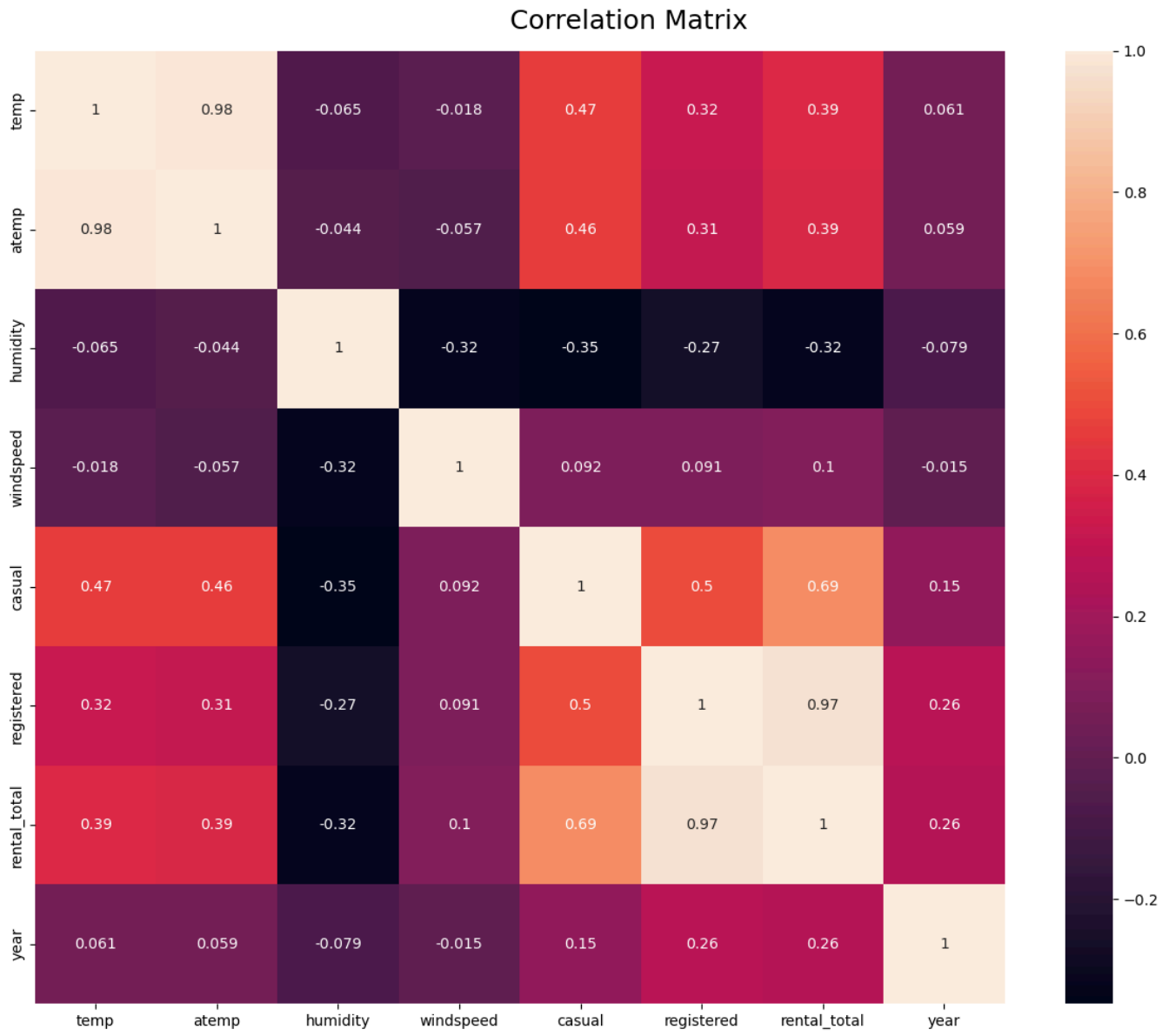
In [527…
```
cr=df.corr(numeric_only=True)
```

In [528…
```
cr
```

Out[528]:

	temp	atemp	humidity	windspeed	casual	registered	rental_total	yea
temp	1.000000	0.984948	-0.064949	-0.017852	0.467097	0.318571	0.394454	0.06122
atemp	0.984948	1.000000	-0.043536	-0.057473	0.462067	0.314635	0.389784	0.05854
humidity	-0.064949	-0.043536	1.000000	-0.318607	-0.348187	-0.265458	-0.317371	-0.07860
windspeed	-0.017852	-0.057473	-0.318607	1.000000	0.092276	0.091052	0.101369	-0.01522
casual	0.467097	0.462067	-0.348187	0.092276	1.000000	0.497250	0.690414	0.14524
registered	0.318571	0.314635	-0.265458	0.091052	0.497250	1.000000	0.970948	0.26426
rental_total	0.394454	0.389784	-0.317371	0.101369	0.690414	0.970948	1.000000	0.26040
year	0.061226	0.058540	-0.078606	-0.015221	0.145241	0.264265	0.260403	1.00000

In [529…
```
fig = plt.figure(figsize=(12,10)).suptitle("Correlation Matrix",fontsize=18)
sns.heatmap(df.corr(numeric_only=True),annot=True)
plt.tight_layout()
plt.show()
```



Insights

- Temperature and ambient temperature show positive correlation with total rental count.
- Humidity shows negative correlation with total count.
- windspeed shows positive but very weak correlation with total count.
- Casual and registered user both shows very strong positive correlation with total count.

Hypothesis Testing

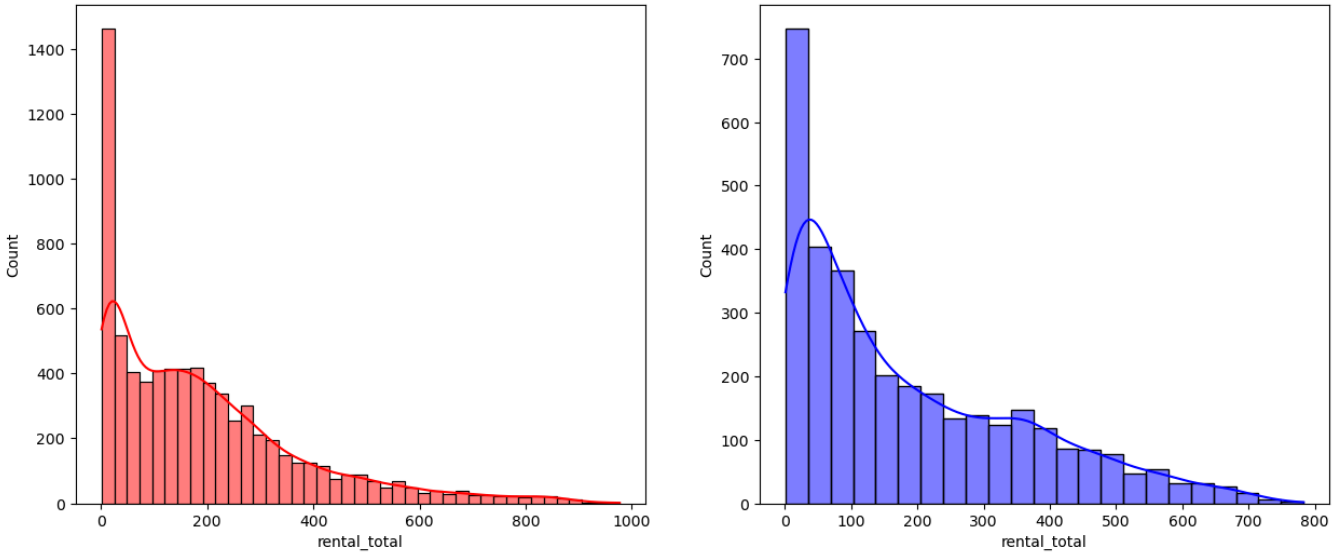
Bike rental on week day and weekends

Step 1: Setup null and alternate hypothesis

- Null Hypothesis(Ho):Number of bike rental is same on weekdays and weekend
- Alternate Hypothesis(Ha):Number of bike rental is different on weekdays and weekend

Step 2: Determine the type of distribution

In [530…
```
fig = plt.figure(figsize=(15,6))
plt.subplot(1,2,1)
sns.histplot(x="rental_total", data=df[df["workingday"]=="Yes"],kde=True,color="r")
plt.subplot(1,2,2)
sns.histplot(x="rental_total", data=df[df["workingday"]=="No"],kde=True,color="b")
plt.tight_layout
plt.show()
```



Clearly we can see data is not normally distributed so we will use t-test here and as we have two sample both numeric, so we will use two_sample t-test i.e.ttest_ind.

Step 3:Determine p-value and set significance level(alpha)

Here we will take alpha as 0.05

In [531…
```
working_day=df[df["workingday"]=="Yes"]["rental_total"]
nonworking_day=data=df[df["workingday"]=="No"]["rental_total"]
```

In [532…
```
t_stat, p_value = ttest_ind(working_day,nonworking_day)
print("p_value:",p_value)

alpha = 0.05

if p_value < alpha:
    print("Reject H0")
    print("Number of bike rental is different on weekdays and weekend")

else:
    print("Fail to reject H0")
    print("Number of bike rental is same on weekdays and weekend")
```

```
p_value: 0.22644804226361348
Fail to reject H0
Number of bike rental is same on weekdays and weekend
```

Conclusion :As p value is high and we cannot reject null hypothesis, we conclude that demand of bikes is same for both weekdays and weekends.

Demand of bicycles for different weather condition

Step 1: Setup null and alternate hypothesis

- Null Hypothesis(Ho):Number of bike rental is same in all weather condition

- Alternate Hypothesis(Ha):Number of bike rental is different in different weather condition

Step 2: Determine the type of distribution

Step 3 :Determine p-value and set significance level(alpha)

Here we will take alpha as 0.05

Step 4:Compare p-value with significance level(alpha)

Here we will use Anova as we have numeric vs categorical data having more than 2 categories

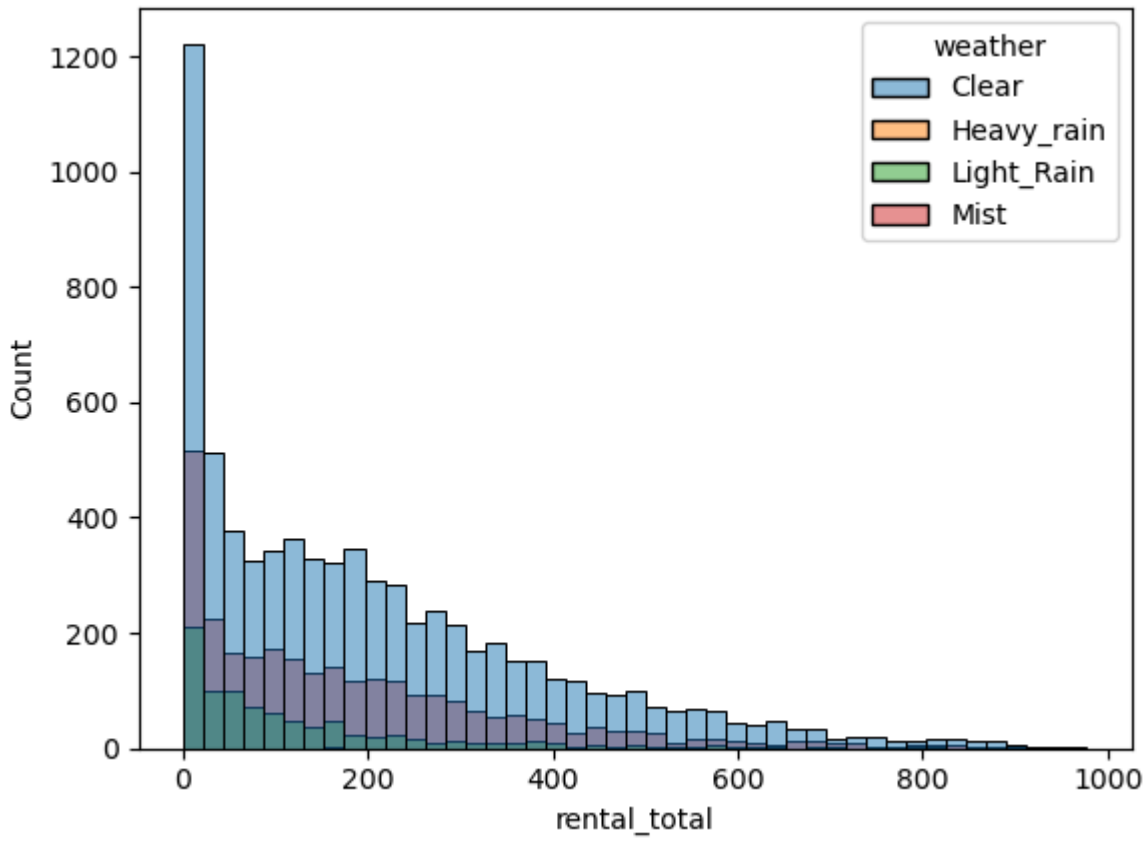
In [533…
```
df.groupby("weather")["rental_total"].mean()
```

Out[533]:
```
weather
Clear         205.236791
Heavy_rain    164.000000
Light_Rain    118.846333
Mist          178.955540
Name: rental_total, dtype: float64
```

In [534…
```
Clear=df[df["weather"]=="Clear"]["rental_total"]
Heavy_rain=df[df["weather"]=="Heavy_rain"]["rental_total"]
Light_Rain=df[df["weather"]=="Light_Rain"]["rental_total"]
Mist=df[df["weather"]=="Mist"]["rental_total"]
```

In [535…
```
sns.histplot(x="rental_total",data=df,hue="weather")
```

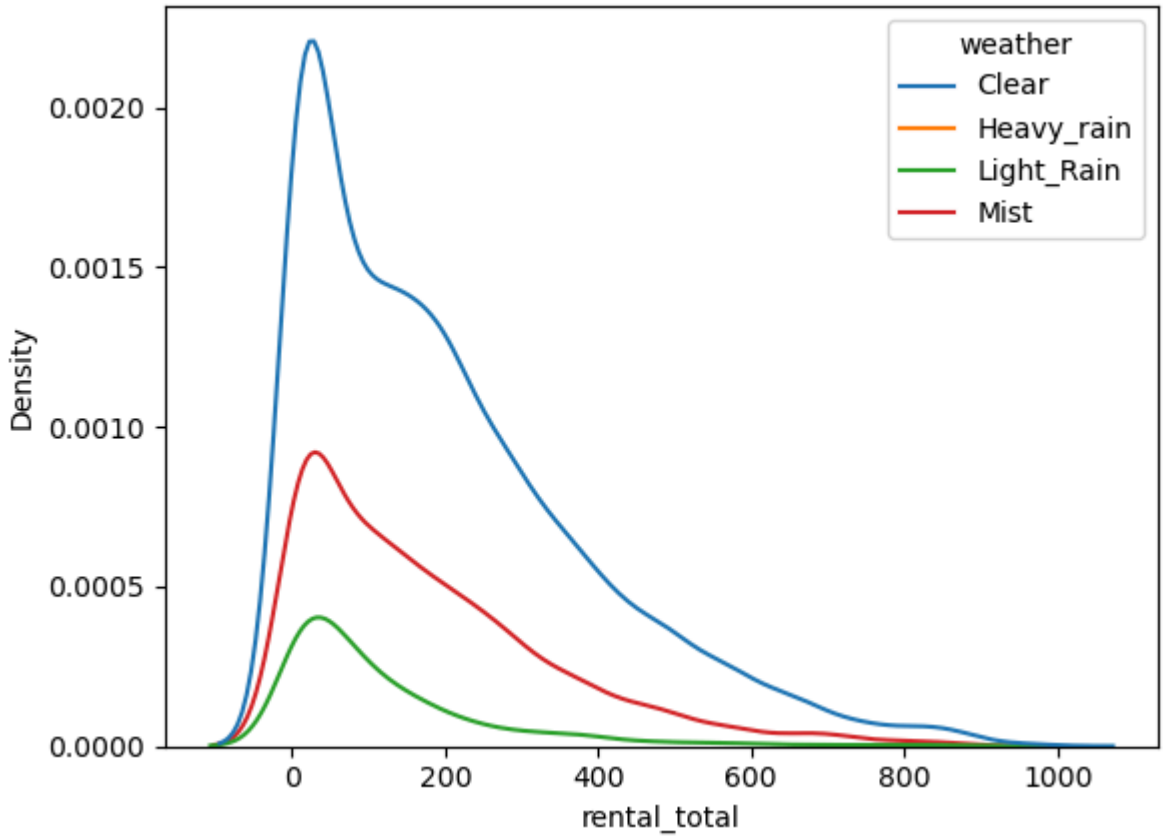
Out[535]:
```
<Axes: xlabel='rental_total', ylabel='Count'>
```



In [536…
```
sns.kdeplot(x="rental_total",data=df,hue="weather")
```

```
/var/folders/63/h28070vs2zx_1xl7yyzsblth0000gn/T/ipykernel_7736/1919156447.py:1: User
Warning: Dataset has 0 variance; skipping density estimate. Pass `warn_singular=False
` to disable this warning.
  sns.kdeplot(x="rental_total",data=df,hue="weather")
```

Out[536]: <Axes: xlabel='rental_total', ylabel='Density'>



Data clearly doesn't seem having normal distribution,it is right skewed still we will do shapiro test

Shapiro Test

In [537…
```
Clear_subset=Clear.sample(100)
Light_Rain_subset=Light_Rain.sample(100)
Mist_subset=Mist.sample(100)
```

In [538…
```
test_stat, p_value = shapiro(Clear_subset)

if p_value < 0.05:
  print("Reject H0")
  print("Not Gaussian")
else:
  print("Fail to reject H0")
  print("Follows Gaussian")
```

```
Reject H0
Not Gaussian
```

In [539…
```
test_stat, p_value = shapiro(Light_Rain_subset)

if p_value < 0.05:
  print("Reject H0")
  print("Not Gaussian")
else:
  print("Fail to reject H0")
  print("Follows Gaussian")
```

```
Reject H0
Not Gaussian
```

In [540…
```
test_stat, p_value = shapiro(Mist_subset)

if p_value < 0.05:
```

```
  print("Reject H0")
  print("Not Gaussian")
else:
  print("Fail to reject H0")
  print("Follows Gaussian")
```

```
Reject H0
Not Gaussian
```

Data of none of the weather condition follows gaussian distribution.

Levene Test

In [541…

```
#H0: Variances are equal
#Ha: Variances are not equal

levene_stat, p_value = levene(Clear, Light_Rain, Mist)

if p_value < 0.05:
  print("Reject H0")
  print("Variance are not equal")
else:
  print("Fail to reject H0")
  print("Variance are equal")
```

```
Reject H0
Variance are not equal
```

Variance among all three weather is also not same

We can't use Anova here as two of our assumptions is not fulfilling the required creteria.
So we will use kruskal walis test here.

Kruskal Walis Test

In [542…

```
k_stat, p_value =  kruskal(Clear, Light_Rain, Mist)
print(p_value)
if p_value < 0.05:
  print("Reject H0")
else:
  print("Fail to reject H0")
  print("All groups have same mean")
```

```
3.1220661786485616e-45
Reject H0
```

As p_value is smaller than alpha we can reject null hypothesis.
So Number of bike rental is different in different weather condition.

Demand of bicycles for different seasons

Step 1: Setup null and alternate hypothesis

- Null Hypothesis(Ho):Number of bike rental is same in all seasons
- Alternate Hypothesis(Ha):Number of bike rental is different in different season

Step 2: Determine the type of distribution

Step 3 :Determine p-value and set significance level(alpha)

Here we will take alpha as 0.05

Step 4:Compare p-value with significance level(alpha)

Here also we will use Anova as we have numeric vs categorical data having more than 2 categories

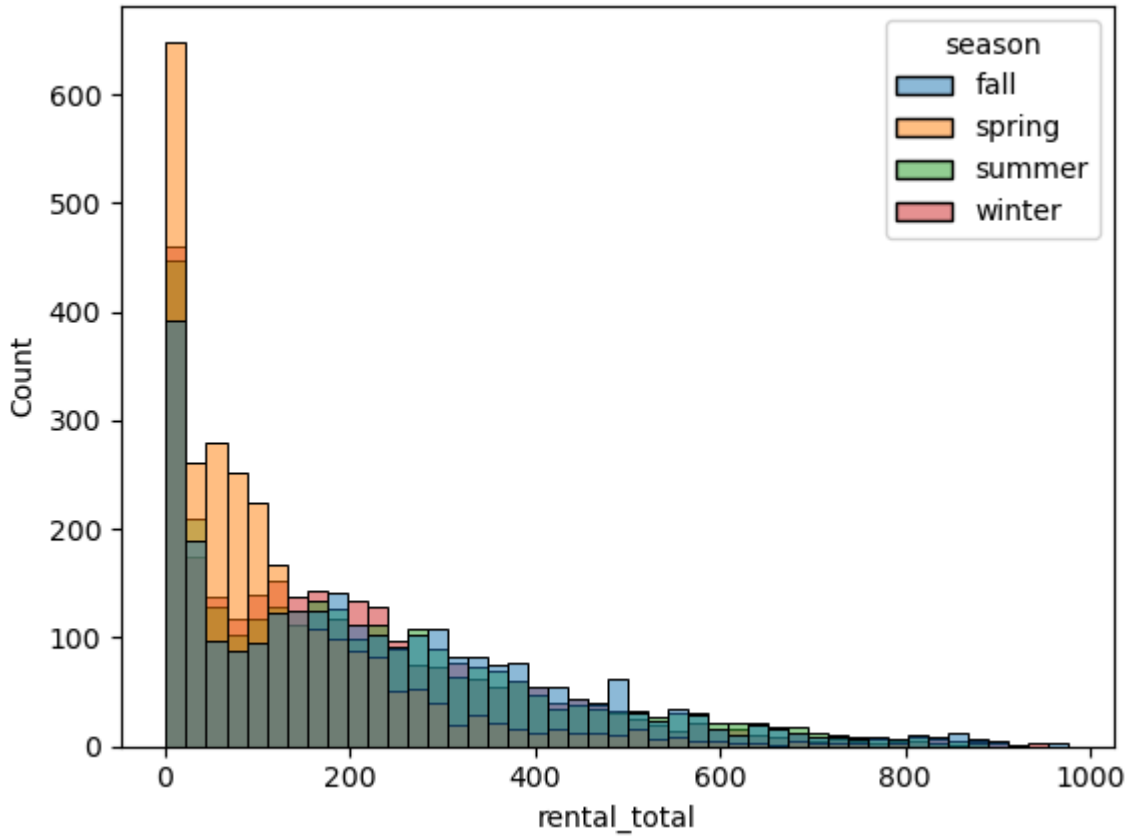
In [543…
```
df.groupby("season")["rental_total"].mean()
```

Out[543]:
```
season
fall      234.417124
spring    116.343261
summer    215.251372
winter    198.988296
Name: rental_total, dtype: float64
```

In [544…
```
Fall=df[df["season"]=="fall"]["rental_total"]
Spring=df[df["season"]=="spring"]["rental_total"]
Summer=df[df["season"]=="summer"]["rental_total"]
Winter=df[df["season"]=="winter"]["rental_total"]
```

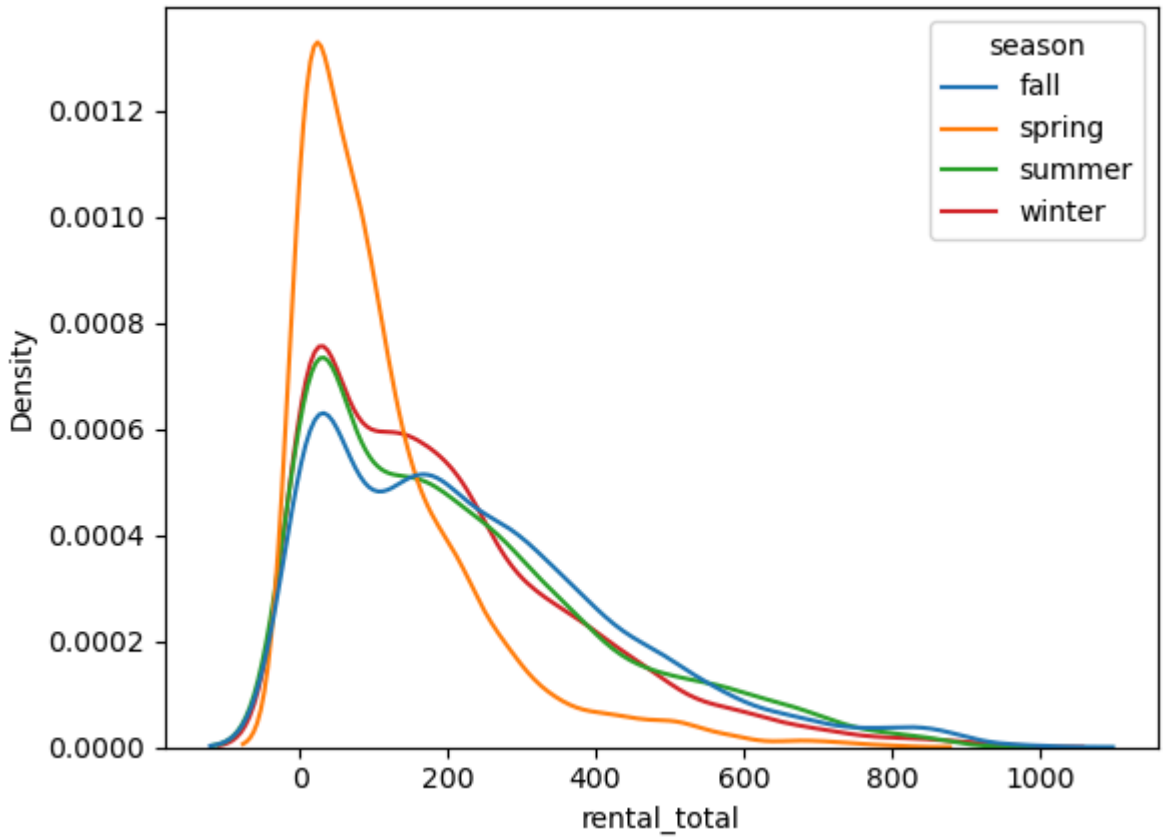
In [545…
```
sns.histplot(x="rental_total",data=df,hue="season")
```

Out[545]:
```
<Axes: xlabel='rental_total', ylabel='Count'>
```



In [546…
```
sns.kdeplot(x="rental_total",data=df,hue="season")
```

Out[546]:
```
<Axes: xlabel='rental_total', ylabel='Density'>
```



Data clearly doesn't seem having normal distribution,it is right skewed still we will do shapiro test

Shapiro Test

In [547…
```
Fall_subset=Fall.sample(100)
Spring_subset=Spring.sample(100)
Summer_subset=Summer.sample(100)
Winter_subset=Winter.sample(100)
```

In [548…
```
test_stat, p_value = shapiro(Fall_subset)

if p_value < 0.05:
  print("Reject H0")
  print("Not Gaussian")
else:
  print("Fail to reject H0")
  print("Follows Gaussian")
```

```
Reject H0
Not Gaussian
```

In [549…
```
test_stat, p_value = shapiro(Spring_subset)

if p_value < 0.05:
  print("Reject H0")
  print("Not Gaussian")
else:
  print("Fail to reject H0")
  print("Follows Gaussian")
```

```
Reject H0
Not Gaussian
```

In [550…
```
test_stat, p_value = shapiro(Summer_subset)

if p_value < 0.05:
  print("Reject H0")
```

```
  print("Not Gaussian")
else:
  print("Fail to reject H0")
  print("Follows Gaussian")
```

```
Reject H0
Not Gaussian
```

In [551…

```
test_stat, p_value = shapiro(Winter_subset)

if p_value < 0.05:
  print("Reject H0")
  print("Not Gaussian")
else:
  print("Fail to reject H0")
  print("Follows Gaussian")
```

```
Reject H0
Not Gaussian
```

Again shapiro test shows that none of the season follows gaussian distribution.

Levene Test

In [552…

```
#H0: Variances are equal
#Ha: Variances are not equal

levene_stat, p_value = levene(Fall,Spring,Summer,Winter)

if p_value < 0.05:
  print("Reject H0")
  print("Variance are not equal")
else:
  print("Fail to reject H0")
  print("Variance are equal")
```

```
Reject H0
Variance are not equal
```

Variance among all four season is not same so we can't use anova here but let's do it anyways and see the difference between Anova and Kruskal Walis Test

Anova

In [553…

```
#H0: All groups have same mean
#Ha: One or more group have different mean

f_stats, p_value = f_oneway(Fall,Spring,Summer,Winter)
print(p_value)
if p_value < 0.05:
  print("Reject H0")
else:
  print("Fail to reject H0")
  print("All groups have same mean")
```

```
6.164843386499654e-149
Reject H0
```

Kruskal Walis Test

In [554…

```
k_stat, p_value =  kruskal(Fall,Spring,Summer,Winter)
print(p_value)
if p_value < 0.05:
```

```
  print("Reject H0")
else:
  print("Fail to reject H0")
  print("All groups have same mean")
```

```
2.479008372608633e-151
Reject H0
```

As p_value is smaller than alpha we can reject null hypothesis.
So Number of bike rental is different in different season.

Effect of Weather conditions during different Seasons

Step 1: Setup null and alternate hypothesis

- Null Hypothesis(Ho):Weather condition and Season are independent of each other
- Alternate Hypothesis(Ha):Weather condition and Season are dependent on each other

Step 2: Determine the type of distribution

Step 3 :Determine p-value and set significance level(alpha)

Here we will take alpha as 0.05

Step 4:Compare p-value with significance level(alpha)

Here we will use chi2_contingency as we have categorical vs categorical data having more than 2 categories

In [555…
```
Chi_tab=pd.crosstab(df["weather"],df["season"])
chi_tab
```

Out[555]:

season	Fall	Spring	Summer	Winter
weather				
Clear	1930	1759	1801	1702
Heavy_rain	0	1	0	0
Light_Rain	199	211	224	225
Mist	604	715	708	807

Chi_square Test

In [556…
```
chi_stat, p_value, df , exp_freq = chi2_contingency(Chi_tab)
print(chi_stat)
print(p_value)
print(df)
print(exp_freq)

if p_value < alpha:
  print("Reject H0")
  print("Season has impact on weather condition")
else:
  print("Fail to reject")
  print("Season has no impact on weather condition")
```

```
49.15865559689363
1.5499250736864862e-07
9
[[1.80559765e+03 1.77454639e+03 1.80559765e+03 1.80625831e+03]
 [2.51056403e-01 2.46738931e-01 2.51056403e-01 2.51148264e-01]
 [2.15657450e+02 2.11948742e+02 2.15657450e+02 2.15736359e+02]
 [7.11493845e+02 6.99258130e+02 7.11493845e+02 7.11754180e+02]]
Reject H0
Season has impact on weather condition
```

As p_value is smaller than alpha we can reject null hypothesis.

So Season has impact on weather condition.

Insights

- Seasons have good impact on rental bikes with highest rental in fall season followed by summer and winter.
- There is no much difference in rental during normal weekdays and holidays.
- Weather condition also have significant influence in total rentals with clear weather having highest rentals and light rain having lowest.
- Also time of the year i.e. different months have different demand of rentals with june to september having highest rentals.
- Humidity,Temperature and Windspeed also have significant impact on total rentals.
- Registered users have more probability of renting bikes as compared to casual and new users.

Recommendations

- Marketing team should focus on promoting rental according to seasons giving more discounts during off seasons like winter.
- Company should focus on promoting bike rental during holidays by giving offers and discounts.
- company should try to set pricing according to weather condition and also introduce variable pricing during peak weather conditions.
- Also they should focus on improving customer riding comfort and making their journey seamless.
- Time dependent pricing can be a good option.
- As most of the customers are registered,yulu should provide offers tailored to them like coupons, vouchers ,membership plan benefits.
- For new users yulu can provide free rides upto certain numbers,this will encourage customers to try yulu bikes.
- yulu should try to provide monthly,quaterly,half-yearly and yearly subscriptions for new users with different discounts percentages on different plans they purchase.
- Location for picking up of the bikes should be made easily accessible.
- Social media promotions and digital marketing can be very effective in today's time.