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

1. About Yulu

Yulu is India's leading micro-mobility service provider, which offers unique vehicles for the daily commute. Starting off as a mission to eliminate traffic congestion in India, Yulu provides the safest commute solution through a user-friendly mobile app to enable shared, solo and sustainable commuting.

Yulu zones are located at all the appropriate locations (including metro stations, bus stands, office spaces, residential areas, corporate offices, etc) to make those first and last miles smooth, affordable, and convenient!

Yulu has recently suffered considerable dips in its revenues. They have contracted a consulting company to understand the factors on which the demand for these shared electric cycles depends. Specifically, they want to understand the factors affecting the demand for these shared electric cycles in the Indian market.

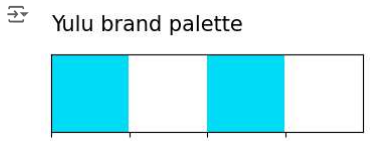
Problem statement

Understainding demand side Economics. Factors affecting the demand.

```
Start coding or generate with AI.
```

2. Yulu Brand Pallet

```
sns.palplot(['#00DCF7', '#FFFFFF', '#00DCF7', '#FFFFFF'])

plt.title("Yulu brand palette ",loc='left',fontsize=15,y=1.2)
plt.show()
```



Define Problem Statement and perform Exploratory Data Analysis (10 points)

Definition of problem (as per given problem statement with additional views)

3. Importing the Dataset

Observations on shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required) , missing value detection, statistical summary.

```
!wget "https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/original/bike_sharing.csv?1642089089" -O bike.csv
```

```
--2024-05-27 12:15:41--  https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/original/bike_sharing.csv?1642089089
Resolving d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)... 13.224.9.24, 13.224.9.181, 13.224.9.103, ...
Connecting to d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)|13.224.9.24|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 648353 (633K) [text/plain]
Saving to: 'bike.csv'
```

```
bike.csv            100%[===================>] 633.16K  --.-KB/s    in 0.03s

2024-05-27 12:15:41 (20.9 MB/s) - 'bike.csv' saved [648353/648353]
```

```
# loading the data in pandas dataframe
df = pd.read_csv("bike.csv")
```

```
#viewing the data
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

4. Cleaning the Data and Understandings it.

```
df.shape
```

```
(10886, 12)
```

There are 10886 rows and 12 columns

```
#checking for null values in all the column
df.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
```

There is no null values in the data set.

```
# Understanding the data type
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
```

The date time column need to be converted to datetime

```
# Lets get the unique values
df.nunique()
```

```
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
```

Season, Holiday, Weather, Working Day has less unique values.

Temp / Atemp , Humidity, Windspeed is categorical data but it has high uniques values. for better analysis we can divide that into 5 categories for better analysis.

Cheching weather two Categorical Columns are Statistically Significant or Not?

1. Season / Weather / tem/ atemp / humidity / windspeed

Here we will use Chi Square Independent to test weather the two categorical variables are independent or not

```
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

```
from scipy.stats import chi2_contingency
```

```
# cate_season = ['season','holiday','weather','temp', 'atemp', 'humidity','windspeed']
cate_season = ['season','temp','weather','atemp', 'humidity','windspeed']
# a = len(cate_season)
for i in cate_season :
  # print(i)
  for j in cate_season :
    # print(j)
    if i != j :

      gender_product = pd.crosstab(index=df[i],columns=df[j])
      chi_stat, p_value, dof, exp_freq = chi2_contingency(gender_product) # chi_stat, p_value, df, expected value

      # print("chi_stat:",chi_stat)
      # print("p_value:",p_value)
      # print("df:",df)
      # print("exp_freq:",exp_freq)

      alpha = 0.05

      if p_value < alpha:
          # print("Reject H0")
          print(f"{i} impacts {j}")
      else:
          # print("Fail to reject H0")
          print(f"{i} does not impact {j}")
```

```
season impacts temp
season impacts weather
season impacts atemp
season impacts humidity
season impacts windspeed
temp impacts season
temp impacts weather
temp impacts atemp
temp impacts humidity
temp impacts windspeed
weather impacts season
weather impacts temp
weather impacts atemp
weather impacts humidity
weather impacts windspeed
atemp impacts season
atemp impacts temp
atemp impacts weather
atemp impacts humidity
atemp impacts windspeed
humidity impacts season
humidity impacts temp
humidity impacts weather
humidity impacts atemp
humidity impacts windspeed
windspeed impacts season
windspeed impacts temp
windspeed impacts weather
windspeed impacts atemp
windspeed impacts humidity
```

All these features Season, Temp, Weather, Atemp, Temp, Humidity, Windspeed are correated. Let us now analyse other set of features like holiday and working day

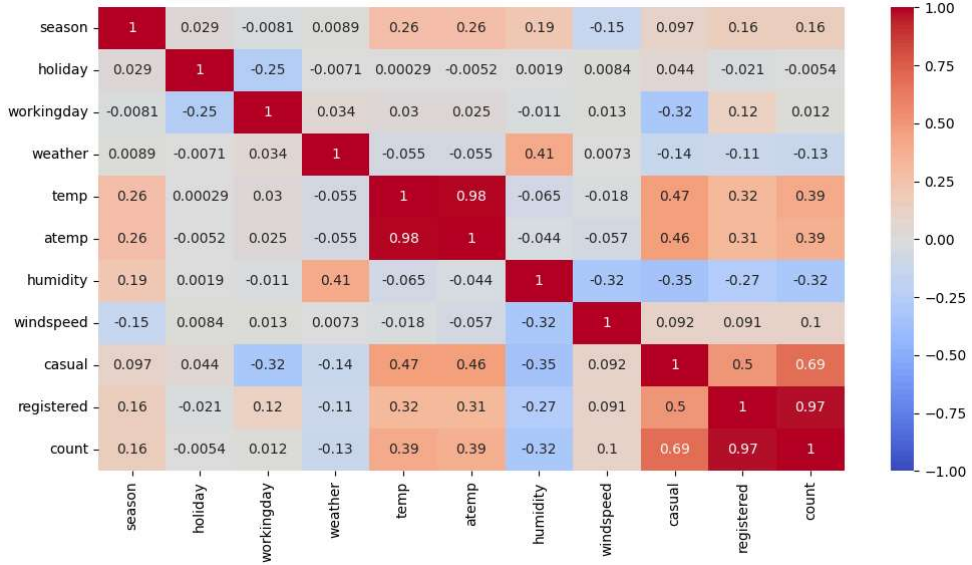
2. Holiday/workingday

```
cate_season = ['holiday','workingday']
# a = len(cate_season)
for i in cate_season :
  # print(i)
  for j in cate_season :
    # print(j)
    if i != j :

      gender_product = pd.crosstab(index=df[i],columns=df[j])
      chi_stat, p_value, dof, exp_freq = chi2_contingency(gender_product) # chi_stat, p_value, df, expected value

      # print("chi_stat:",chi_stat)
      # print("p_value:",p_value)
      # print("df:",df)
      # print("exp_freq:",exp_freq)

      alpha = 0.05

      if p_value < alpha:
          # print("Reject H0")
          print(f"{i} impacts {j}")
      else:
          # print("Fail to reject H0")
          print(f"{i} does not impact {j}")
```

```
holiday impacts workingday
workingday impacts holiday
```

```
df_wodatetime = df.drop(columns = "datetime")
# corr_mat = df_wodatetime.corr()

plt.figure(figsize=(12, 6))
sns.heatmap(df_wodatetime.corr(), annot=True, cmap='coolwarm', vmin=-1, vmax=1)
```

```
<Axes: >
```



Since Temp and ATemp has high positive correated we can drop one of the columns. Moreover count and registered are also higly correlated means most of the traffic is from Registered Users. So, we can seprately analyse the casual users and Registered Users. Since we are analysing demand side so here we can analyse when the casual users are taking the ride and here we can push them to registered users, this will derive more organic growth.

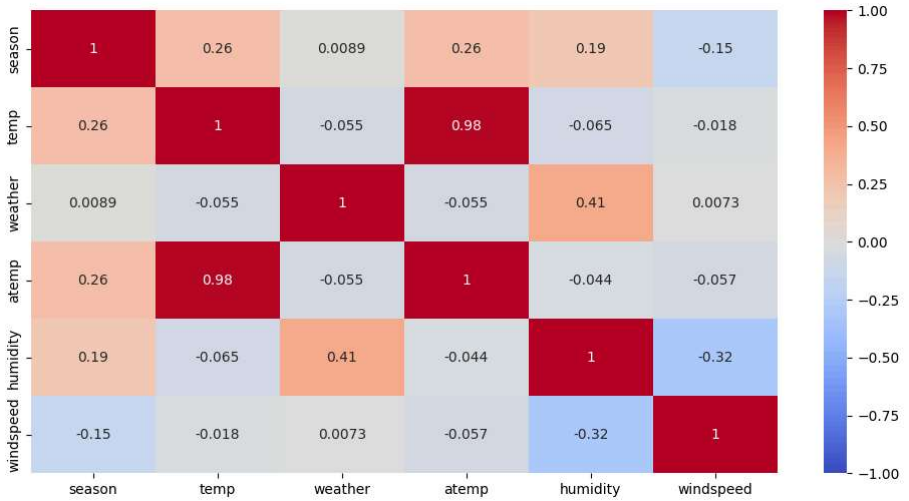
Lets also check the Specific correlation with respect to weather features:-

Seasons are Spring, Summer, Fall and Winter

season: season (1: spring, 2: summer, 3: fall, 4: winter)

```
cate_season = ['season','temp','weather','atemp', 'humidity','windspeed']
df_season = df_wodatetime[cate_season]
plt.figure(figsize=(12, 6))
sns.heatmap(df_season.corr(), annot=True, cmap='coolwarm', vmin=-1, vmax=1)
```

```
<Axes: >
```



Since Temp and ATemp are highly positively correlated we can drop on of the column. Atem is the temperature that is felt and temp is the recorded. So, its better to keep temp as it would better help in Analysis as its more accurate.

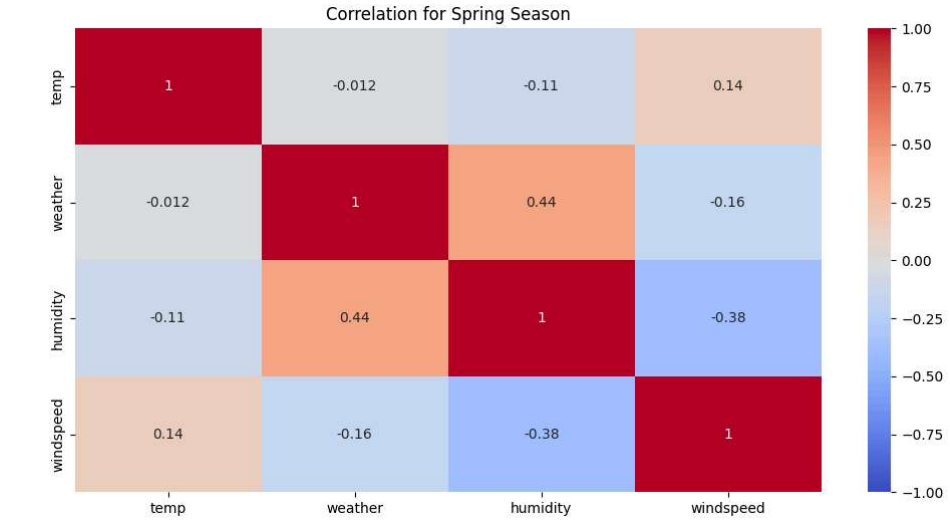
Lets analyse all these features with respect to each season.

Seasons are Spring, Summer, Fall and Winter

season: season (1: spring, 2: summer, 3: fall, 4: winter)

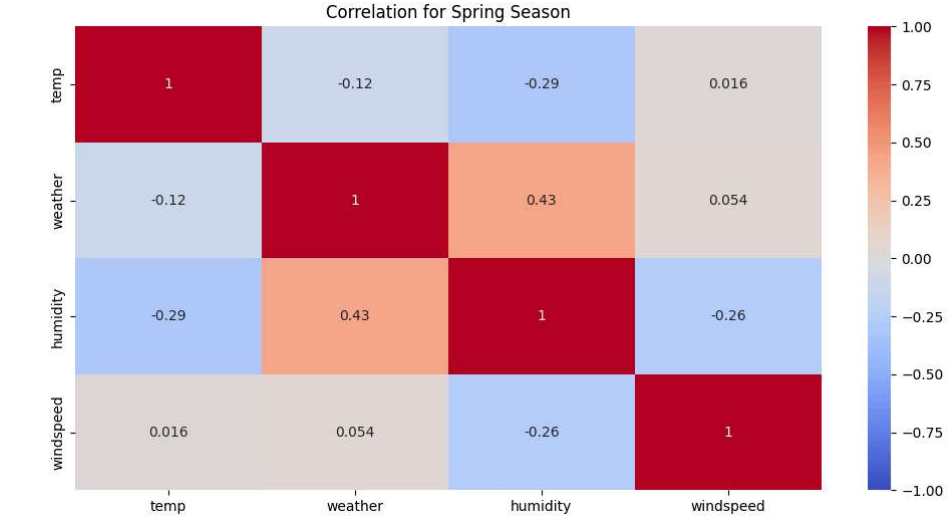
```
cate_season = ['temp','weather', 'humidity','windspeed']
df_season = df_wodatetime[cate_season]
plt.figure(figsize=(12, 6))
plt.title("Correlation for Spring Season")
sns.heatmap(df_season[df["season"] == 1].corr(), annot=True, cmap='coolwarm', vmin=-1, vmax=1)
```

```
<Axes: title={'center': 'Correlation for Spring Season'}>
```



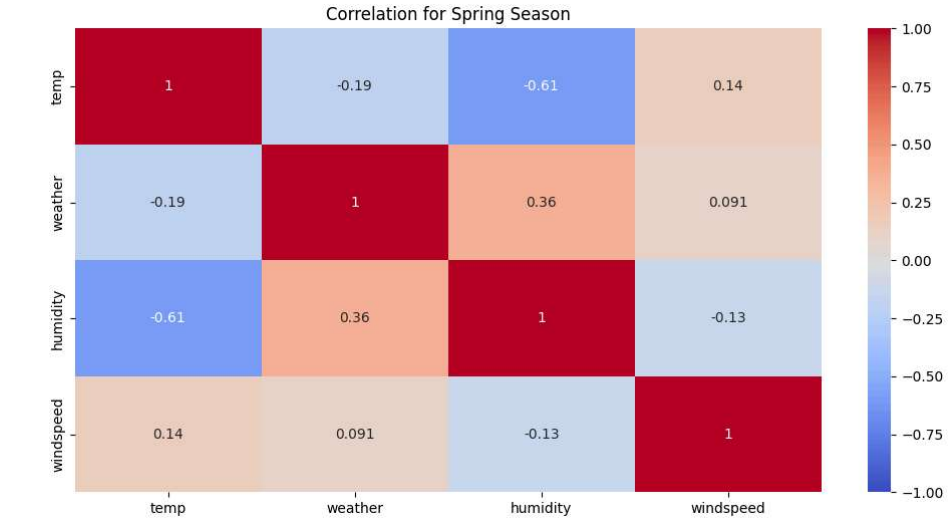

```
cate_season = ['temp','weather', 'humidity','windspeed']
df_season = df_wodatetime[cate_season]
plt.figure(figsize=(12, 6))
plt.title("Correlation for Spring Season")
sns.heatmap(df_season[df["season"] == 2].corr(), annot=True, cmap='coolwarm', vmin=-1, vmax=1)
```

```
<Axes: title={'center': 'Correlation for Spring Season'}>
```



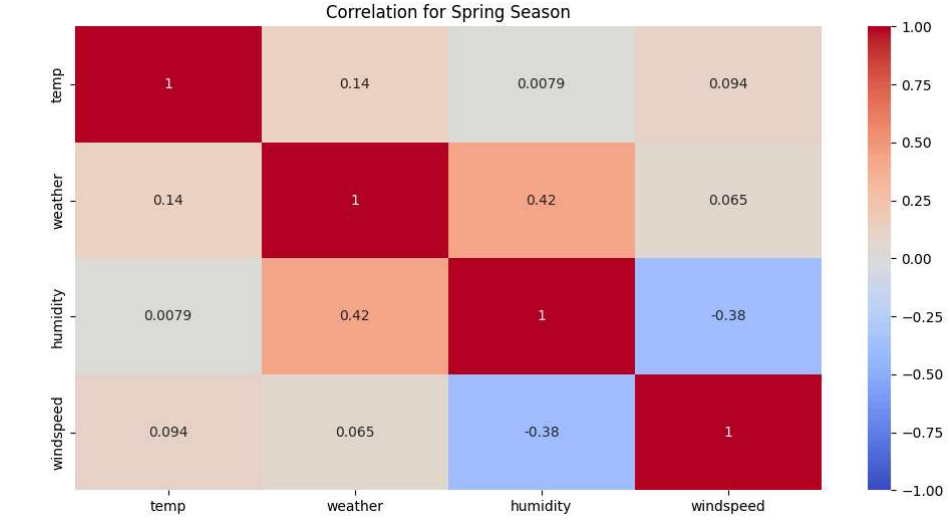

```
cate_season = ['temp','weather', 'humidity','windspeed']
df_season = df_wodatetime[cate_season]
plt.figure(figsize=(12, 6))
plt.title("Correlation for Spring Season")
sns.heatmap(df_season[df["season"] == 3].corr(), annot=True, cmap='coolwarm', vmin=-1, vmax=1)
```

```
<Axes: title={'center': 'Correlation for Spring Season'}>
```




```
cate_season = ['temp','weather', 'humidity','windspeed']
df_season = df_wodatetime[cate_season]
plt.figure(figsize=(12, 6))
plt.title("Correlation for Spring Season")
sns.heatmap(df_season[df["season"] == 4].corr(), annot=True, cmap='coolwarm', vmin=-1, vmax=1)
```

```
<Axes: title={'center': 'Correlation for Spring Season'}>
```




```
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

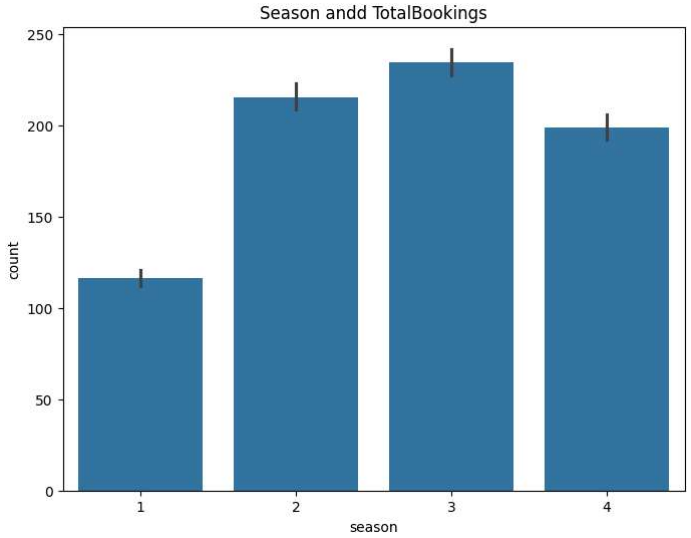
Next steps: Generate code with df View recommended plots

```
#Uniques values in each column
categorical = ["season","holiday","workingday", "weather", "temp", "atemp","humidity","windspeed"]

for i in categorical :
  print(f"{i} :-> {df[i].unique()}")
```

```
season :-> [1 2 3 4]
holiday :-> [0 1]
workingday :-> [0 1]
weather :-> [1 2 3 4]
temp :-> [ 9.84  9.02  8.2  13.12 15.58 14.76 17.22 18.86 18.04 16.4  13.94 12.3
 10.66  6.56  5.74  7.38  4.92 11.48  4.1   3.28  2.46 21.32 22.96 23.78
 24.6  19.68 22.14 20.5  27.06 26.24 25.42 27.88 28.7  30.34 31.16 29.52
 33.62 35.26 36.9  32.8  31.98 34.44 36.08 37.72 38.54  1.64  0.82 39.36
 41.  ]
atemp :-> [14.395 13.635 12.88  17.425 19.695 16.665 21.21  22.725 21.97  20.455
 11.365 10.605  9.85   8.335  6.82   5.305  6.06   9.09  12.12   7.575
 15.91   3.03   3.79   4.545 15.15  18.18  25.    26.515 27.275 29.545
 23.485 25.76  31.06  30.305 24.24  18.94  31.82  32.575 33.335 28.79
 34.85  35.605 37.12  40.15  41.665 40.91  39.395 34.09  28.03  36.365
 37.88  42.425 43.94  38.635  1.515  0.76   2.275 43.18  44.695 45.455]
humidity :-> [ 81  80  75  86  76  77  72  82  88  87  94 100  71  66  57  46  42  39
  44  47  50  43  40  35  30  32  64  69  55  59  63  68  74  51  56  52
  49  48  37  33  28  38  36  93  29  53  34  54  41  45  92  62  58  61
  60  65  70  27  25  26  31  73  21  24  23  22  19  15  67  10   8  12
  14  13  17  16  18  20  85   0  83  84  78  79  89  97  90  96  91]
windspeed :-> [ 0.      6.0032 16.9979 19.0012 19.9995 12.998  15.0013  8.9981 11.0014
 22.0028 30.0026 23.9994 27.9993 26.0027  7.0015 32.9975 36.9974 31.0009
 35.0008 39.0007 43.9989 40.9973 51.9987 46.0022 50.0021 43.0006 56.9969
 47.9988]
```

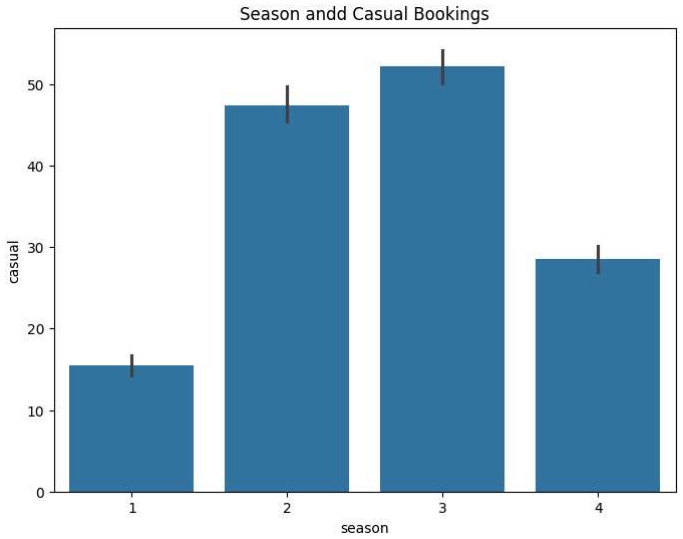
```
categorical = ["season","holiday","workingday", "weather", "temp", "atemp","humidity","windspeed"]
plt.figure(figsize = (8,6))
plt.title("Season andd TotalBookings")
sns.barplot(x = df["season"],y = df["count"])
# season: season (1: spring, 2: summer, 3: fall, 4: winter)
```

```
<Axes: title={'center': 'Season andd TotalBookings'}, xlabel='season', ylabel='count'>
```




```
categorical = ["season","holiday","workingday", "weather", "temp", "atemp","humidity","windspeed"]
plt.figure(figsize = (8,6))
plt.title("Season andd Casual Bookings")
sns.barplot(x = df["season"],y = df["casual"])
# season: season (1: spring, 2: summer, 3: fall, 4: winter)
```

```
<Axes: title={'center': 'Season andd Casual Bookings'}, xlabel='season', ylabel='casual'>
```



Double-click (or enter) to edit

Understanding Column Profiling

datetime: datetime

season: season (1: spring, 2: summer, 3: fall, 4: winter)

holiday: whether day is a holiday or not (extracted from http://dchr.dc.gov/page/holiday-schedule)

workingday: if day is neither weekend nor holiday is 1, otherwise is 0.

weather: 1: Clear, Few clouds, partly cloudy, partly cloudy 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds 4: Heavy Rain + Ice Pallets + 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

```
df['temp_labeled'] = pd.qcut(df['temp'], q=5, labels=['T1', 'T2', 'T3', 'T4','T5'])
df['atemp_labeled'] = pd.qcut(df['atemp'], q=5, labels=['AT1', 'AT2', 'AT3', 'AT4','AT5'])
df['humidity_labeled'] = pd.qcut(df['temp'], q=3, labels=['H1', 'H2', 'H3'])
df["windspeed_labled"] =  pd.qcut(df['windspeed'], q=5, labels=['W1', 'W2', 'W3', 'W4','W5'])
df['datetime'] = pd.to_datetime(df['datetime'])
df['year'] = df['datetime'].dt.year
df['month'] = df['datetime'].dt.month
df['dayofweek'] = df['datetime'].dt.dayofweek
df['hour'] = df['datetime'].dt.hour
```

```
df_final_r = df[["year","month","dayofweek", "hour","season", "holiday","workingday", "weather","temp_labeled", "humidity_labeled", "windspeed_labled", "registered", "casual"]]
df_final_t = df[["year","month","dayofweek","hour", "season", "holiday", "workingday","weather","temp_labeled", "humidity_labeled", "windspeed_labled", "registered", "casual", "count"]]
```

```
df_final_t.head()
```

	year	month	dayofweek	hour	season	holiday	workingday	weather	temp_labeled	humidity_labeled	windspeed_labled	registered	casual	count
0	2011	1	5	0	1	0	0	1	T1	H1	W1	13	3	16
1	2011	1	5	1	1	0	0	1	T1	H1	W1	32	8	40
2	2011	1	5	2	1	0	0	1	T1	H1	W1	27	5	32
3	2011	1	5	3	1	0	0	1	T1	H1	W1	10	3	13
4	2011	1	5	4	1	0	0	1	T1	H1	W1	1	0	1

Next steps: Generate code with df_final_t View recommended plots

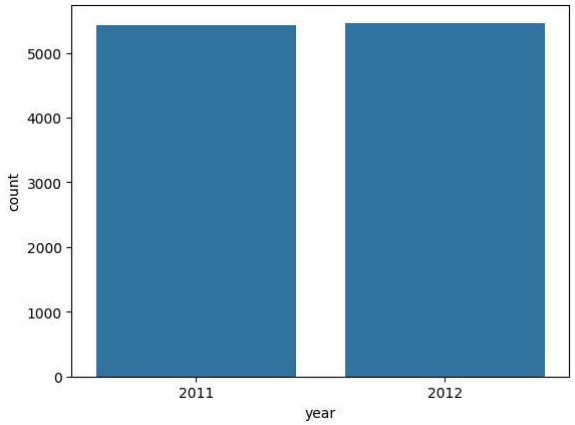
```
orignaldf = df
```

5. Analysing the Data

Univariate Analysis (distribution plots of all the continuous variable(s) barplots/countplots of all the categorical variables)

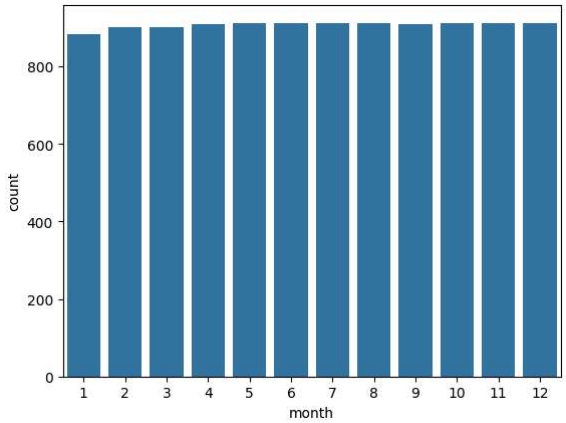
```
sns.countplot(x=df_final_t["year"], data=df_final_t)
```

```
<Axes: xlabel='year', ylabel='count'>
```



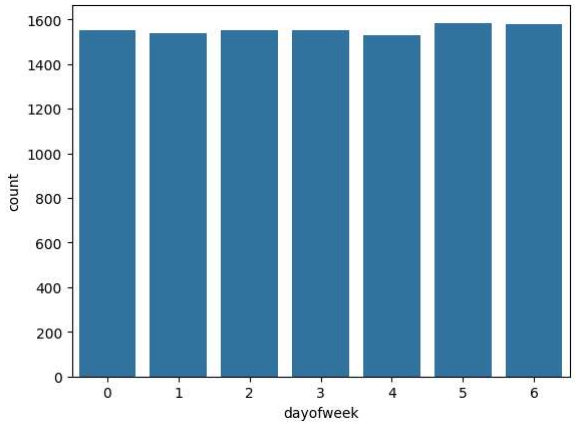

```
sns.countplot(x=df_final_t["month"], data=df_final_t)
```

```
<Axes: xlabel='month', ylabel='count'>
```



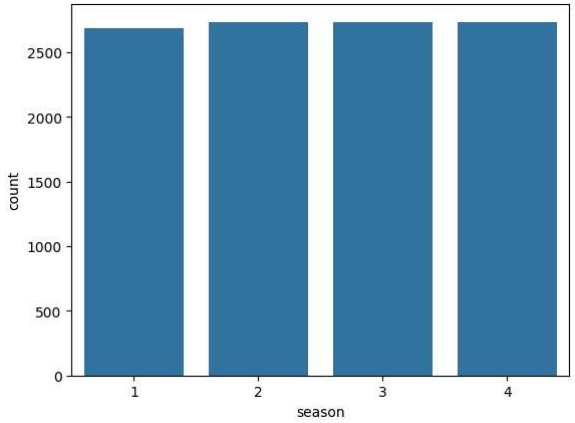

```
sns.countplot(x=df_final_r["dayofweek"], data=df_final_r)
```

```
<Axes: xlabel='dayofweek', ylabel='count'>
```



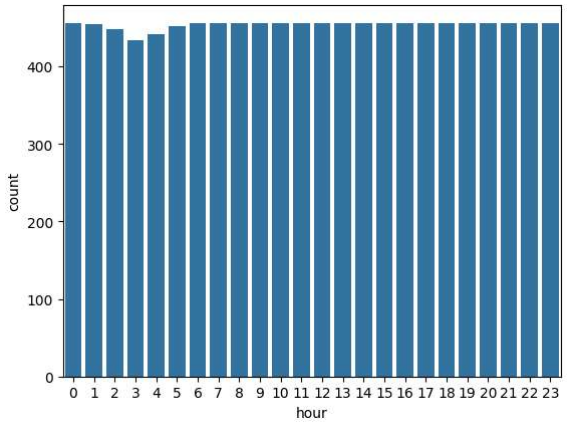

```
# ["year","month","dayofweek", "season", "holiday", "weather","temp_labeled", "humidity_labeled", "windspeed_labled", "count"]
sns.countplot(x=df_final_r["season"], data=df_final_r)
```

```
<Axes: xlabel='season', ylabel='count'>
```



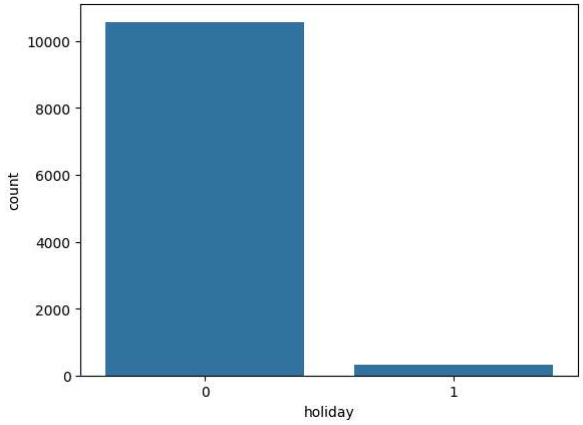

```
sns.countplot(x=df_final_t["hour"], data=df_final_t)
```

```
<Axes: xlabel='hour', ylabel='count'>
```




```
# ["year","month","dayofweek", "season", "holiday", "weather","temp_labeled", "humidity_labeled", "windspeed_labled", "count"]
sns.countplot(x=df_final_r["holiday"], data=df_final_r)
```

```
<Axes: xlabel='holiday', ylabel='count'>
```




```
(df_final_r.groupby(["weather"])["registered"].sum()/df_final_r.groupby(["weather"])["registered"].sum().sum())*100
```

```
weather
1    70.048679
2    24.797959
3     5.144032
4     0.009331
Name: registered, dtype: float64
```

Lets check for percentage of Casual users vis a vis Registered Users

```
(df_final_r.groupby(["weather"])["casual"].sum()/ df_final_r.groupby(["weather"])["registered"].sum()) *100
```

```
weather
1    24.440149
2    20.777112
3    17.200882
4     3.797468
dtype: float64
```

(weather: 1: Clear, Few clouds, partly cloudy, partly cloudy 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog)

In weather 1 i.e 24 percent of people are Casual riders. In weather 2 i.e 21 percent of people are Casual riders. In weather 3 i.e 17 percent of people are Casual riders. In weather 4 i.e 3 percent of people are Casual riders.

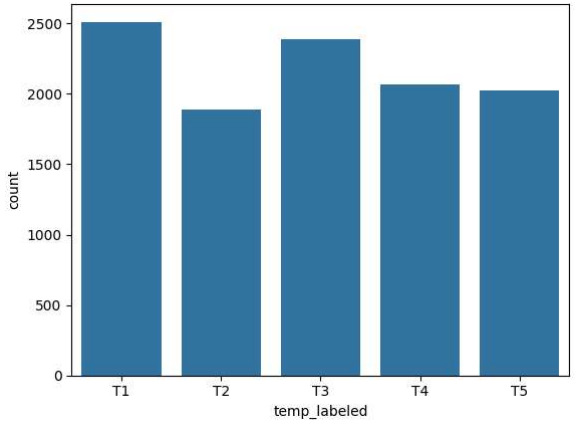
Giving discount to first time users on 2 - 3 rides espically during Weather 1 and Weather 2 would substantially increase these are potential customers as casual rides are 44 percent during this time. So, this would increase the organic users. Hence taking care of the scenario on the demand side.

```
df_final_t.groupby(["weather"])["count"].sum()
```

```
weather
1    1476063
2     507160
3     102089
4        164
Name: count, dtype: int64
```

```
# ["year","month","dayofweek", "season", "holiday", "weather","temp_labeled", "humidity_labeled", "windspeed_labled", "count"]
sns.countplot(x=df_final_r["temp_labeled"], data=df_final_r)
```

```
<Axes: xlabel='temp_labeled', ylabel='count'>
```

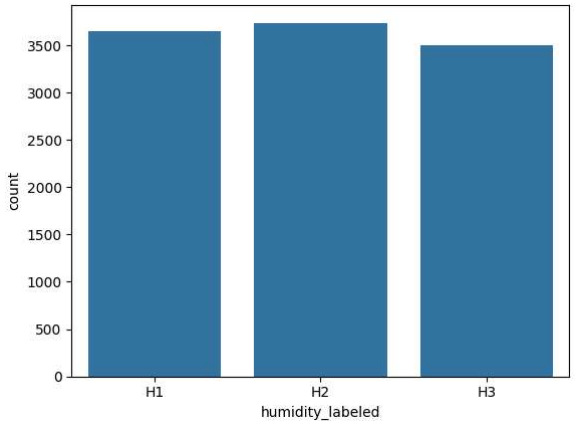



```
df_final_t.groupby(["temp_labeled"])["count"].sum()
```

```
temp_labeled
T1    242058
T2    310253
T3    445453
T4    463821
T5    623891
Name: count, dtype: int64
```

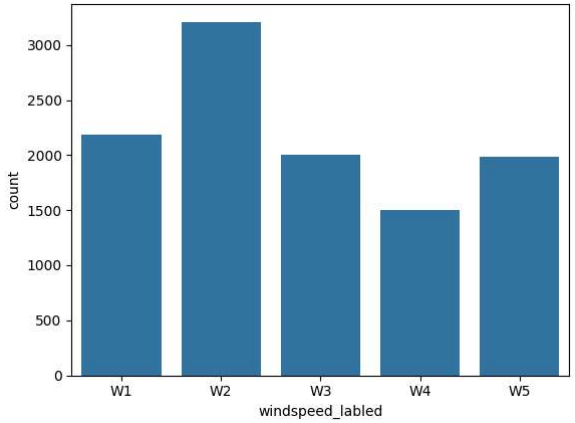
```
# ["year","month","dayofweek", "season", "holiday", "weather","temp_labeled", "humidity_labeled", "windspeed_labled", "count"]
sns.countplot(x=df_final_r["humidity_labeled"], data=df_final_r)
```

```
<Axes: xlabel='humidity_labeled', ylabel='count'>
```




```
# ["year","month","dayofweek", "season", "holiday", "weather","temp_labeled", "humidity_labeled", "windspeed_labled", "count"]
sns.countplot(x=df_final_r["windspeed_labled"], data=df_final_r)
```

```
<Axes: xlabel='windspeed_labled', ylabel='count'>
```




```
df.head()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	temp_labeled	atemp_labeled	humidity_labeled	windspeed_labled	year	month	dayofweek	hour
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16	T1	AT1	H1	W1	2011	1	5	0
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40	T1	AT1	H1	W1	2011	1	5	1
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32	T1	AT1	H1	W1	2011	1	5	2
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13	T1	AT1	H1	W1	2011	1	5	3
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1	T1	AT1	H1	W1	2011	1	5	4

Next steps: Generate code with df View recommended plots

```
temprange = df.groupby(["temp_labeled"]).agg({'temp' : ['min','max']})
windspeedrange = df.groupby(["windspeed_labled"]).agg({'windspeed' : ['min','max']})
humidityrange = df.groupby(["humidity_labeled"]).agg({'humidity' : ['min','max']})


print(temprange, windspeedrange,humidityrange)
```

```
                temp
                 min    max
    temp_labeled
    T1            0.82  13.12
    T2           13.94  17.22
    T3           18.04  22.96
    T4           23.78  27.88
    T5           28.70  41.00                  windspeed
                          min      max
    windspeed_labled
    W1                 0.0000   6.0032
    W2                 7.0015  11.0014
    W3                12.9980  15.0013
    W4                16.9979  19.0012
    W5                19.9995  56.9969                  humidity
                         min  max
    humidity_labeled
    H1                     0  100
    H2                     0  100
    H3                    17  100
```

Temperature ranges are as follows:-

T1 : (0.82-13.12)

T2 : (13.94,17.22)

T3 : (18.04 - 22.96)

T4: (23.78 - 27.88)

T5 : (28.70 - 41.00)

Windspeed ranges are as follows:-

W1 : (0.0000 , 6.0032)

W2 : (7.0015 , 11.0014)

W3 : (12.9980 , 15.0013)

W4 : (16.9979 , 19.0012)

W5 : (19.9995 , 56.9969)

Humidity ranges are as follows:-

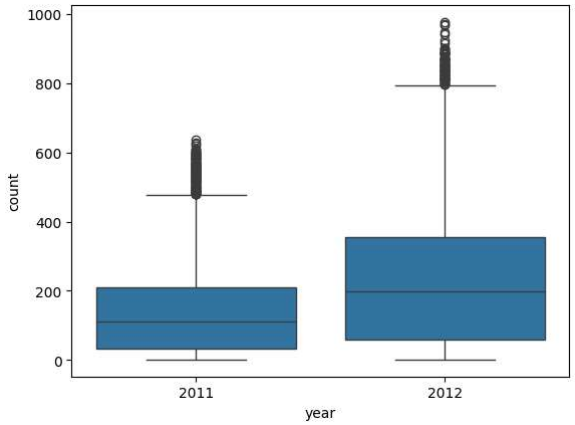
H1 : (21, 100) H2 : (0, 100) H3 : (0, 100)

Bivariate Analysis (Relationships between important variables such as workday and count, season and count, weather and count.

1. Increase/ Decrease in riders yearly
2. Classification of total riders as per Month, Day of the week
3. Effect of weather, temp, humidity, windspeed on total riders.

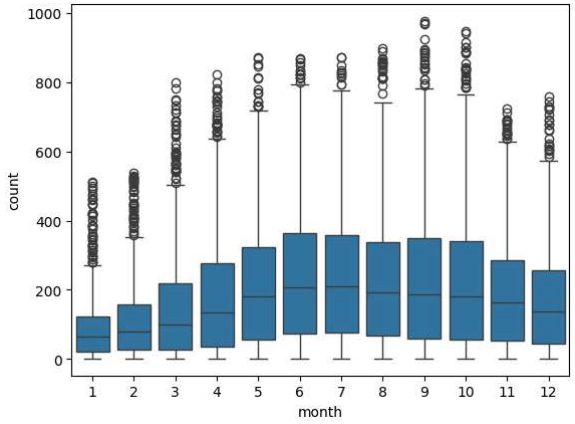
```
# ["year","month","dayofweek", "season", "holiday", "weather","temp_labeled", "humidity_labeled", "windspeed_labled", "count"]
sns.boxplot(x='year', y='count', data=df_final_t)
```

```
<Axes: xlabel='year', ylabel='count'>
```




```
# ["year","month","dayofweek", "season", "holiday", "weather","temp_labeled", "humidity_labeled", "windspeed_labled", "count"]
sns.boxplot(x='month', y='count', data=df_final_t)
```

```
<Axes: xlabel='month', ylabel='count'>
```



The number of rides are higher from April to October i.e 70 percent

January and March has lots of outlierish data.

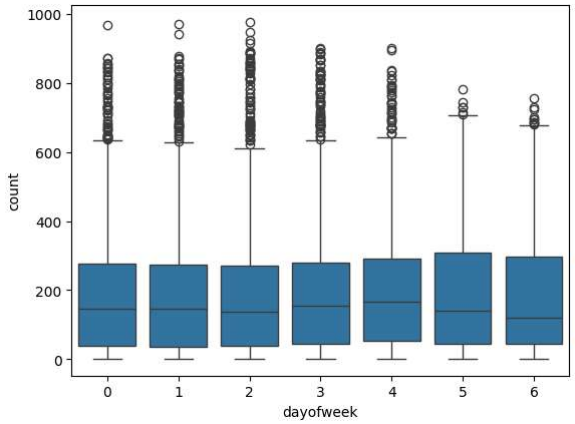
```
# popmonth = df_final_t[[(df_final_t["month"] == 5) & (df_final_t["month"] == 6) & (df_final_t["month"] == 7) & (df_final_t["month"] == 8) & (df_final_t["month"] == 9) & (df_final_t["month"] == 9) ]]
(df_final_t.groupby(["month"])["count"].sum()/df_final_t.groupby(["month"])["count"].sum().sum() ) *100
```

```
month
1      3.830492
2      4.752536
3      6.401464
4      8.027040
5      9.597185
6     10.584298
7     10.291032
8     10.238238
9     10.190911
10     9.946602
11     8.460419
12     7.679781
Name: count, dtype: float64
```

```
# ["year","month","dayofweek", "season", "holiday", "weather","temp labeled", "humidity labeled", "windspeed labled", "count"]
```

```
sns.boxplot(x='dayofweek', y='count', data=df_final_t)
```

```
<Axes: xlabel='dayofweek', ylabel='count'>
```



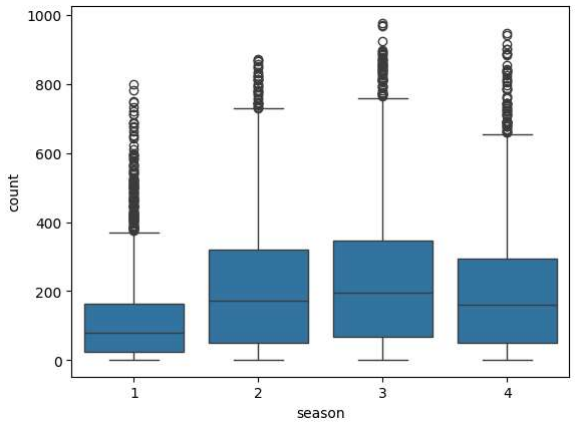
Fridays and Wednesdays are the days when rides are take the most. Though the differnece is not much. The least dip could be seen on Sunday followed by Tuesday. Yulu can offer higher discount on Sunday and Tuesday to push the demand on Sunday as no of bikes avaliable remains same.

```
(df_final_t.groupby(["dayofweek"])["count"].sum()/df_final_t.groupby(["dayofweek"])["count"].sum().sum() ) *100
```

```
dayofweek
0    14.159645
1    14.000880
2    14.012436
3    14.692137
4    14.505274
5    14.937501
6    13.692126
Name: count, dtype: float64
```

```
# ["year","month","dayofweek", "season", "holiday", "weather","temp_labeled", "humidity_labeled", "windspeed_labled", "count"]
sns.boxplot(x='season', y='count', data=df_final_t)
```

```
<Axes: xlabel='season', ylabel='count'>
```



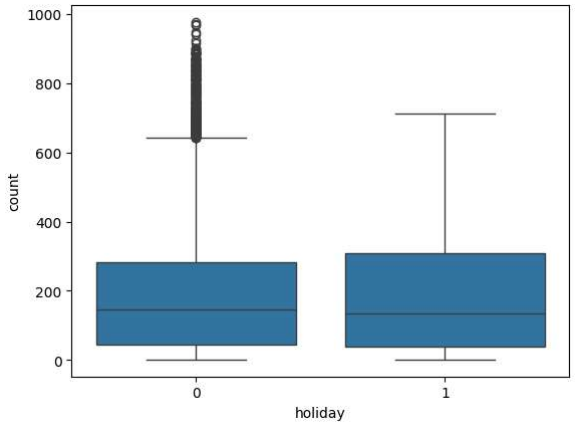
60 percent of rides are from season 2 and 3. There are lots of outliers during season 1 and season 4. We can offer discounts during season 1, this would boost the demand as number of rides unused would decrease.

```
(df_final_t.groupby(["season"])["count"].sum()/df_final_t.groupby(["season"])["count"].sum().sum())*100
```

```
season
1    14.984493
2    28.208524
3    30.720181
4    26.086802
Name: count, dtype: float64
```

```
# ["year","month","dayofweek", "season", "holiday", "weather","temp_labeled", "humidity_labeled", "windspeed_labled", "count"]
sns.boxplot(x='holiday', y='count', data=df_final_t)
```

```
<Axes: xlabel='holiday', ylabel='count'>
```



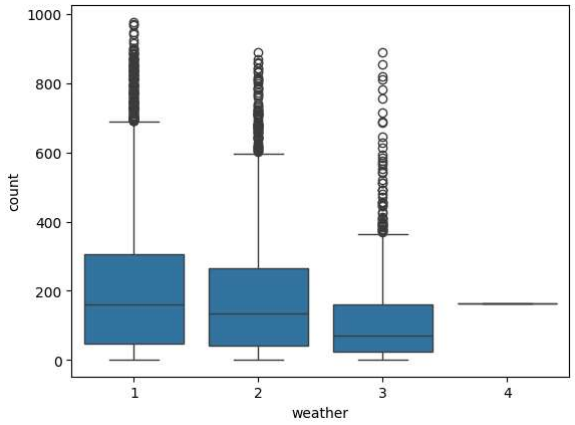
97 percent of bikes are booked when there is no holiday.

```
(df_final_t.groupby(["holiday"])["count"].sum()/df_final_t.groupby(["holiday"])["count"].sum().sum())*100
```

```
holiday
0    97.228067
1     2.771933
Name: count, dtype: float64
```

```
# ["year","month","dayofweek", "season", "holiday", "weather","temp_labeled", "humidity_labeled", "windspeed_labled", "count"]
sns.boxplot(x='weather', y='count', data=df_final_t)
```

```
<Axes: xlabel='weather', ylabel='count'>
```



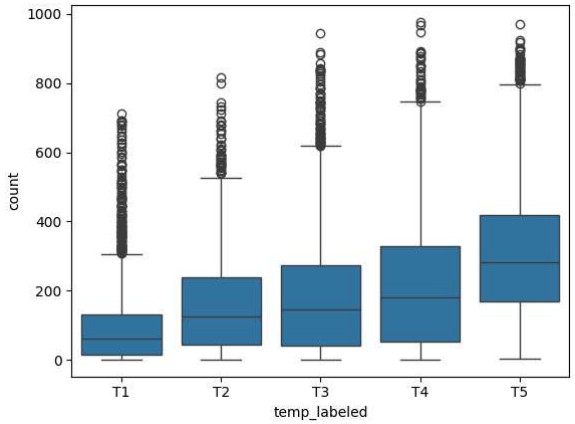
Almost 95 percent rides comes in Weather 1 and Weather 2.

```
(df_final_t.groupby(["weather"])["count"].sum()/df_final_t.groupby(["weather"])["count"].sum().sum())*100
```

```
weather
1    70.778230
2    24.318669
3     4.895237
4     0.007864
Name: count, dtype: float64
```

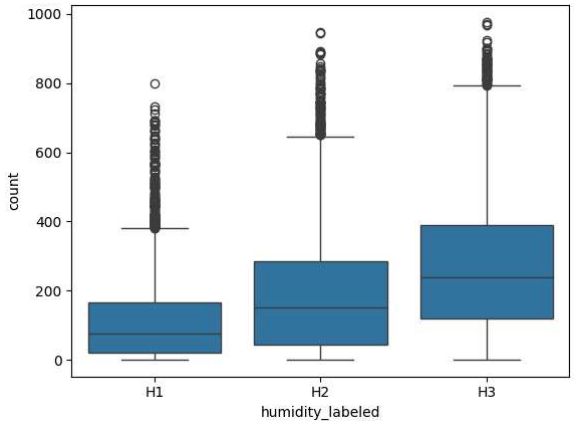
```
# ["year","month","dayofweek", "season", "holiday", "weather","temp_labeled", "humidity_labeled", "windspeed_labled", "count"]
sns.boxplot(x='temp_labeled', y='count', data=df_final_t)
```

```
<Axes: xlabel='temp_labeled', ylabel='count'>
```



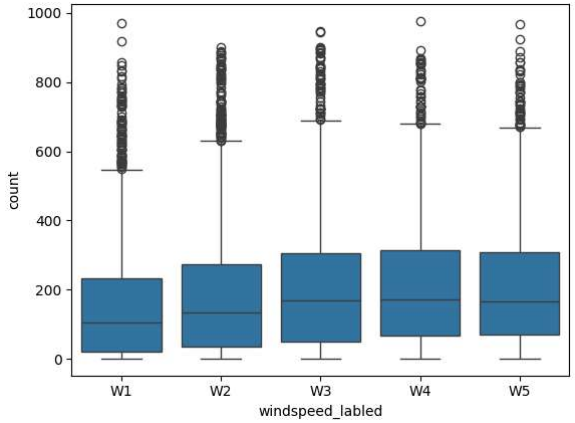

```
# ["year","month","dayofweek", "season", "holiday", "weather","temp_labeled", "humidity_labeled", "windspeed_labled", "count"]
sns.boxplot(x='humidity_labeled', y='count', data=df_final_t)
```

```
<Axes: xlabel='humidity_labeled', ylabel='count'>
```




```
# ["year","month","dayofweek", "season", "holiday", "weather","temp_labeled", "humidity_labeled", "windspeed_labled", "count"]
sns.boxplot(x='windspeed_labled', y='count', data=df_final_t)
```

```
<Axes: xlabel='windspeed_labled', ylabel='count'>
```



Let us conduct ANNOVA test and t test.

Set a significance level (alpha)

```
import scipy.stats as stats

# # Sample data
# data = {
#     'Column1': [1, 2, 3, 4, 5],
#     'Column2': [6, 7, 8, 9, 10],
#     'Column3': [11, 12, 13, 14, 15],
#     'Column4': [16, 17, 18, 19, 20],
#     'Column5': [21, 22, 23, 24, 25],
#     'Column6': [26, 27, 28, 29, 30]
# }
# df = pd.DataFrame(data)

# Set the confidence level (e.g., 95%)
confidence_level = 0.95

# Calculate the confidence intervals for each column
confidence_intervals = {}
for column in df_final_t.columns:
    mean = np.mean(df_final_t[column])
    std_err = stats.sem(df_final_t[column])
    interval = stats.t.interval(confidence_level, df=len(df_final_t[column])-1, loc=mean, scale=std_err)
    confidence_intervals[column] = interval

# Display the confidence intervals
for column, interval in confidence_intervals.items():
    print(f"Column '{column}' Confidence Interval: {interval}")
```

```
Column 'year' Confidence Interval: (0.49253511816548673, 0.5113230482868373)
Column 'month' Confidence Interval: (5.456785340989594, 5.586205656622017)
Column 'dayofweek' Confidence Interval: (2.9763023270769366, 3.0516234491494094)
Column 'hour' Confidence Interval: (11.411683801584656, 11.671542360458334)
Column 'season' Confidence Interval: (1.4856442018459477, 1.527583774191635)
Column 'holiday' Confidence Interval: (0.025438876877411944, 0.031698731059387614)
Column 'workingday' Confidence Interval: (0.6721166889346516, 0.6896323465237354)
Column 'weather' Confidence Interval: (0.4065192811550954, 0.4303353945752004)
Column 'temp_labeled' Confidence Interval: (1.8995898242975648, 1.953064961666058)
Column 'humidity_labeled' Confidence Interval: (0.9710007625578786, 1.0014409056398064)
Column 'windspeed_labled' Confidence Interval: (1.7803591046159257, 1.8325381946675579)
Column 'registered' Confidence Interval: (152.714575530128, 158.38977868629678)
Column 'casual' Confidence Interval: (35.083336989644835, 36.96057261902685)
Column 'count' Confidence Interval: (188.17093356334237, 194.9773302617541)
```

```
df = df_final_t
```

```
df.head()
```

	year	month	dayofweek	hour	season	holiday	workingday	weather	temp_labeled	humidity_labeled	windspeed_labled	registered	casual	count
0	2011	1	5	0	1	0	0	1	T1	H1	W1	13	3	16
1	2011	1	5	1	1	0	0	1	T1	H1	W1	32	8	40
2	2011	1	5	2	1	0	0	1	T1	H1	W1	27	5	32
3	2011	1	5	3	1	0	0	1	T1	H1	W1	10	3	13
4	2011	1	5	4	1	0	0	1	T1	H1	W1	1	0	1

Next steps: Generate code with df View recommended plots

```
from scipy.stats import ttest_ind
# H0: Two means are same
t_stat, p_value = ttest_ind(df[df["year"]== 2011 ]["count"], df[df["year"]== 2012 ]["count"])
print("p_value:",p_value)
if p_value < 0.05:
    print("Reject H0")
    print('This means that we have enough evidence to conclude that the means of the two income product groups are statistically different from each other.')
else:
    print("Fail to reject H0")
    print('This means that we do not have enough evidence to conclude that the means of the two income product groups are statistically different from each other.')
```

```
p_value: 3.2420142331759836e-168
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups are statistically different from each other.
```

Performing ANNOVA test on Months

```
#a = ["year","month","dayofweek", "season", "holiday", "weather","temp_labeled", "humidity_labeled", "windspeed_labled"]
#

from scipy.stats import f_oneway # Numeric Vs categorical for many categories
from scipy.stats import ttest_ind # Numeric Vs categorical
f_stats, p_value = f_oneway(df[df["month"]== 1 ]["count"], df[df["month"]== 2 ]["count"],df[df["month"]== 3 ]["count"],df[df["month"]== 4 ]["count"],df[df["month"]== 5 ]["count"],df[df["month"]== 6 ]["count"],df[df["month"]== 7 ]["count"],df[df["mont

print("test statistic:",f_stats)
print("p_value:",p_value)

if p_value < 0.05:
    print("Reject H0")
    print("Atleast one group have different mean")

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

```
test statistic: 78.48339105291323
p_value: 3.9670124592025475e-171
Reject H0
Atleast one group have different mean
```

Performing T test on Months

```
# H0: Two means are same
for i in range(1,12):
  # for j in range(i,13):
    j = i+1
    t_stat, p_value = ttest_ind(df[df["month"]== i ]["count"], df[df["month"]== j ]["count"])
    # print("p_value:",p_value)
    if p_value < 0.05:
        print("Reject H0")
        print(f'This means that we have enough evidence to conclude that the means of the two income product groups month{i} and month{j} are statistically different from each other.')
    else:
        print("Fail to reject H0")
        print(f'This means that we do not have enough evidence to conclude that the means of the two income product groups month{i} and month{j} are statistically different from each other.')
```

```
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups month1 and month2 are statistically different from each other.
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups month2 and month3 are statistically different from each other.
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups month3 and month4 are statistically different from each other.
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups month4 and month5 are statistically different from each other.
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups month5 and month6 are statistically different from each other.
Fail to reject H0
This means that we do not have enough evidence to conclude that the means of the two income product groups month6 and month7 are statistically different from each other.
Fail to reject H0
This means that we do not have enough evidence to conclude that the means of the two income product groups month7 and month8 are statistically different from each other.
Fail to reject H0
This means that we do not have enough evidence to conclude that the means of the two income product groups month8 and month9 are statistically different from each other.
Fail to reject H0
This means that we do not have enough evidence to conclude that the means of the two income product groups month9 and month10 are statistically different from each other.
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups month10 and month11 are statistically different from each other.
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups month11 and month12 are statistically different from each other.
```

```
#a = ["year","month","dayofweek", "season", "holiday", "weather","temp_labeled", "humidity_labeled", "windspeed_labled"]
#

from scipy.stats import f_oneway # Numeric Vs categorical for many categories
from scipy.stats import ttest_ind # Numeric Vs categorical
f_stats, p_value = f_oneway(df[df["weather"]== 1 ]["count"], df[df["weather"]== 2 ]["count"],df[df["weather"]== 3 ]["count"],df[df["weather"]== 4 ]["count"])

print("test statistic:",f_stats)
print("p_value:",p_value)

if p_value < 0.05:
    print("Reject H0")
    print("Atleast one group have different mean")

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

```
test statistic: 65.53024112793271
p_value: 5.482069475935669e-42
Reject H0
Atleast one group have different mean
```

Performing ANNOVA Test on Weather

```
# H0: Two means are same
for i in range(1,4):
  # for j in range(i,13):
    j = i+1
    t_stat, p_value = ttest_ind(df[df["weather"]== i ]["count"], df[df["weather"]== j ]["count"])
    # print("p_value:",p_value)
    if p_value < 0.05:
        print("Reject H0")
        print(f'This means that we have enough evidence to conclude that the means of the two income product groups season{i} and season{j} are statistically different from each other.')
    else:
        print("Fail to reject H0")
        print(f'This means that we do not have enough evidence to conclude that the means of the two income product groups season{i} and season{j} are statistically different from each other.')
```

```
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups season1 and season2 are statistically different from each other.
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups season2 and season3 are statistically different from each other.
Fail to reject H0
This means that we do not have enough evidence to conclude that the means of the two income product groups season3 and season4 are statistically different from each other.
```

Performing T test on Holiday

Performing ANNOVA Test on Season

```
#a = ["year","month","dayofweek", "season", "holiday", "weather","temp_labeled", "humidity_labeled", "windspeed_labled"]

from scipy.stats import f_oneway # Numeric Vs categorical for many categories
from scipy.stats import ttest_ind # Numeric Vs categorical
f_stats, p_value = f_oneway(df[df["season"]== 1 ]["count"], df[df["season"]== 2 ]["count"],df[df["season"]== 3 ]["count"],df[df["season"]== 4 ]["count"])

print("test statistic:",f_stats)
print("p_value:",p_value)

if p_value < 0.05:
    print("Reject H0")
    print("Atleast one group have different mean")
    for i in range(1,4):
  # for j in range(i,13):
      j = i+1
      t_stat, p_value = ttest_ind(df[df["season"]== i ]["count"], df[df["season"]== j ]["count"])
      # print("p_value:",p_value)
      if p_value < 0.05:
          print("Reject H0")
          print(f'This means that we have enough evidence to conclude that the means of the two income product groups season{i} and season{j} are  statistically different from each other.')
      else:
          print("Fail to reject H0")
          print(f'This means that we do not have enough evidence to conclude that the means of the two income product groups season{i} and season{j} are statistically different from each other.')

else:
    print("Fail to reject H0")
```

```
test statistic: 236.94671081032106
p_value: 6.164843386499654e-149
Reject H0
Atleast one group have different mean
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups season1 and season2 are  statistically different from each other.
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups season2 and season3 are  statistically different from each other.
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups season3 and season4 are  statistically different from each other.
```

Performing ANNOVA Test on Temperature

```
df.head()
```

	year	month	dayofweek	hour	season	holiday	workingday	weather	temp_labeled	humidity_labeled	windspeed_labled	registered	casual	count
0	2011	1	5	0	1	0	0	1	T1	H1	W1	13	3	16
1	2011	1	5	1	1	0	0	1	T1	H1	W1	32	8	40
2	2011	1	5	2	1	0	0	1	T1	H1	W1	27	5	32
3	2011	1	5	3	1	0	0	1	T1	H1	W1	10	3	13
4	2011	1	5	4	1	0	0	1	T1	H1	W1	1	0	1

Next steps: Generate code with df View recommended plots

```
#a = ["year","month","dayofweek", "season", "holiday", "weather","temp_labeled", "humidity_labeled", "windspeed_labled"]

from scipy.stats import f_oneway # Numeric Vs categorical for many categories
from scipy.stats import ttest_ind # Numeric Vs categorical
f_stats, p_value = f_oneway(df[df["temp_labeled"] == 'T1' ]["count"], df[df["temp_labeled"] == 'T2' ]["count"], df[df["temp_labeled"]== 'T3' ]["count"],df[df["temp_labeled"]== 'T4']["count"],df[df["temp_la

print("test statistic:",f_stats)
print("p_value:",p_value)
# p_value = 0.005
if p_value < 0.05:
    print("Reject H0")
    print("Atleast one group have different mean")
    for i in range(1,5):
  # for j in range(i,13):
      j = i+1
      t_stat, p_value = ttest_ind(df[df["temp_labeled"]== f'T{i}' ]["count"], df[df["temp_labeled"]== f'T{j}' ]["count"])
      # print("p_value:",p_value)
      if p_value < 0.05:
          print("Reject H0")
          print(f'This means that we have enough evidence to conclude that the means of the two income product groups temp{i} and temp{j} are not statistically different from each other.')
      else:
```

```
            print("Fail to reject H0")
            print(f'This means that we do not have enough evidence to conclude that the means of the two income product groups temp{i} and temp{j} are statistically different from each other.')

else:
    print("Fail to reject H0")
    print("Atleast one group have same mean")
```

```
test statistic: 485.31640145205574
p_value: 0.0
Reject H0
Atleast one group have different mean
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups temp1 and temp2 are not statistically different from each other.
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups temp2 and temp3 are not statistically different from each other.
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups temp3 and temp4 are not statistically different from each other.
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups temp4 and temp5 are not statistically different from each other.
```

Here we can conclude that the temprature and feeling of temperature is affecting the count of rides booked

```
#a = ["year","month","dayofweek", "season", "holiday", "weather","temp_labeled", "humidity_labeled", "windspeed_labled"]

from scipy.stats import f_oneway # Numeric Vs categorical for many categories
from scipy.stats import ttest_ind # Numeric Vs categorical
# f_stats, p_value = f_oneway(df[df["temp"] == 'T1' ]["count"], df[df["temp"] == 'T2' ]["count"], df[df["temp"]== 'T3' ]["count"],df[df["temp"]== 'T4']["count"],df[df["temp"]== 'T5' ]["count"])

print("test statistic:",f_stats)
print("p_value:",p_value)
p_value = 0.005
if p_value < 0.05:
    print("Reject H0")
    print("Atleast one group have different mean")
    for i in range(1,6):
  # for j in range(i,13):

      j = i+1

      while j < 6 :
        t_stat, p_value = ttest_ind(df[df["windspeed_labled"]== f'W{i}' ]["count"], df[df["windspeed_labled"]== f'W{j}' ]["count"])
        # print("p_value:",p_value)
        if p_value < 0.05:
            print("Reject H0")
            print(f'This means that we have enough evidence to conclude that the means of the two income product groups Windspeed{i} and Windspeed{j} are  statistically different from each other.')
        else:
            print("Fail to reject H0")
            print(f'This means that we do not have enough evidence to conclude that the means of the two income product groups Windspeed{i} and Windspeed{j} are statistically different from each other.')
        j = j+1
else:
    print("Fail to reject H0")
    print("Atleast one group have same mean")
```

```
test statistic: 485.31640145205574
p_value: 1.4912678454626984e-41
Reject H0
Atleast one group have different mean
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups Windspeed1 and Windspeed2 are  statistically different from each other.
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups Windspeed1 and Windspeed3 are  statistically different from each other.
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups Windspeed1 and Windspeed4 are  statistically different from each other.
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups Windspeed1 and Windspeed5 are  statistically different from each other.
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups Windspeed2 and Windspeed3 are  statistically different from each other.
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups Windspeed2 and Windspeed4 are  statistically different from each other.
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups Windspeed2 and Windspeed5 are  statistically different from each other.
Fail to reject H0
This means that we do not have enough evidence to conclude that the means of the two income product groups Windspeed3 and Windspeed4 are statistically different from each other.
Fail to reject H0
This means that we do not have enough evidence to conclude that the means of the two income product groups Windspeed3 and Windspeed5 are statistically different from each other.
Fail to reject H0
This means that we do not have enough evidence to conclude that the means of the two income product groups Windspeed4 and Windspeed5 are statistically different from each other.
```

Windspeed 3, Windspeed 4 and Windspeed 5 could be statistically similar. The ride books during Windspeed 3 could be similar to rides booked during Windspeed 4 and 5 and via versa.

```
a = df.groupby(["windspeed_labled"])["count"].sum()
a.head()
```

```
windspeed_labled
W1    340464
W2    586141
W3    413355
W4    324437
W5    421079
Name: count, dtype: int64
```

If the historical data over the year suggest that there are high bookings during Windspeed 3, then we can conclude higher bookings during windspeed 4 and 5.

As opposed if we found that there are lower bookings on year on year data during Windspeed 3 then we can add additional discounts to boost ride booking during windspeed 4 and 5.

Conducting Chi test to find how much a feature contribute to the Target

6. Try establishing a relation between the dependent and independent variable (Dependent “Count” & Independent: Workingday, Weather, Season etc)

```
a = ["year","month","dayofweek", "season", "holiday", "weather","temp_labeled", "humidity_labeled", "windspeed_labled"]
from sklearn.feature_selection import chi2
from sklearn.preprocessing import LabelEncoder

# Sample DataFrame with categorical features
# data = {'Feature1': ['A', 'B', 'A', 'B', 'A'],
#         'Feature2': ['X', 'Y', 'X', 'Y', 'X'],
#         'Target': [1, 0, 1, 0, 1]}
# df = pd.DataFrame(data)

# Encoding categorical variables
le = LabelEncoder()
df_new = df
for i in a :
  df_new[i] = le.fit_transform(df[i])


# Chi-squared test
X = df_new[["year","month","dayofweek", "season", "holiday", "weather","temp_labeled", "humidity_labeled", "windspeed_labled"]]
y = df['count']

chi_scores = chi2(X, y)
chi2_series = pd.Series(chi_scores[0], index=X.columns)
print(chi2_series.sort_values())
```

```
humidity_labeled    1486.802413
month               2586.109439
temp_labeled        2692.160795
dtype: float64
<ipython-input-71-6073231f3370>:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_new[i] = le.fit_transform(df[i])
<ipython-input-71-6073231f3370>:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
  df_new[i] = le.fit_transform(df[i])
<ipython-input-71-6073231f3370>:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_new[i] = le.fit_transform(df[i])
<ipython-input-71-6073231f3370>:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_new[i] = le.fit_transform(df[i])
<ipython-input-71-6073231f3370>:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_new[i] = le.fit_transform(df[i])
<ipython-input-71-6073231f3370>:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_new[i] = le.fit_transform(df[i])
<ipython-input-71-6073231f3370>:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_new[i] = le.fit_transform(df[i])
<ipython-input-71-6073231f3370>:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_new[i] = le.fit_transform(df[i])
```

We can conclude that Temperature, Month, Humidity, Windspeed has high correlation to the number of Bikes Booked.

Lets analyse the correation between weather and season by Chi- Square Test of Independence for each Categorical Variables.

```
import scipy.stats as stats
a = ["year","month","dayofweek", "season", "holiday", "weather","temp_labeled", "humidity_labeled", "windspeed_labled"]
b = ["year","month","dayofweek", "season", "holiday", "weather","temp_labeled", "humidity_labeled", "windspeed_labled"]


for i in a :
  for j in b:
    if i!=j:
      # Create a contingency table
      contingency_table = pd.crosstab(df[i], df[j])
      # Perform the Chi-Squared test
      chi2, p, dof, expected = stats.chi2_contingency(contingency_table)

      # Display the test results
      # print("\nChi-Squared Test Results:")
      # print(f"Chi-Squared: {chi2}")
      # print(f"P-value: {p}")
      # print(f"Degrees of Freedom: {dof}")
      # print("Expected Frequencies:")
      # print(pd.DataFrame(expected, index=contingency_table.index, columns=contingency_table.columns))

      if p_value < 0.05:
        print("Reject H0")
        print(f"This suggests that there is significant association between {i} and {j} at the 5% significance level.")
      else:
        print("Fail to reject H0")
        print(f"This suggests that there is no significant association between {i} and {j} at the 5% significance level.")
```

```
This suggests that there is no significant association between weather and windspeed_labled at the 5% significance level.
Fail to reject H0
This suggests that there is no significant association between temp_labeled and year at the 5% significance level.
Fail to reject H0
This suggests that there is no significant association between temp_labeled and month at the 5% significance level.
Fail to reject H0
This suggests that there is no significant association between temp_labeled and dayofweek at the 5% significance level.
Fail to reject H0
This suggests that there is no significant association between temp_labeled and season at the 5% significance level.
Fail to reject H0
This suggests that there is no significant association between temp_labeled and holiday at the 5% significance level.
Fail to reject H0
This suggests that there is no significant association between temp_labeled and weather at the 5% significance level.
Fail to reject H0
This suggests that there is no significant association between temp_labeled and humidity_labeled at the 5% significance level.
Fail to reject H0
This suggests that there is no significant association between temp_labeled and windspeed_labled at the 5% significance level.
Fail to reject H0
This suggests that there is no significant association between humidity_labeled and year at the 5% significance level.
Fail to reject H0
This suggests that there is no significant association between humidity_labeled and month at the 5% significance level.
Fail to reject H0
This suggests that there is no significant association between humidity_labeled and dayofweek at the 5% significance level.
Fail to reject H0
This suggests that there is no significant association between humidity_labeled and season at the 5% significance level.
Fail to reject H0
This suggests that there is no significant association between humidity_labeled and holiday at the 5% significance level.
Fail to reject H0
This suggests that there is no significant association between humidity_labeled and weather at the 5% significance level.
Fail to reject H0
This suggests that there is no significant association between humidity_labeled and temp_labeled at the 5% significance level.
Fail to reject H0
This suggests that there is no significant association between humidity_labeled and windspeed_labled at the 5% significance level.
Fail to reject H0
This suggests that there is no significant association between windspeed_labled and year at the 5% significance level.
Fail to reject H0
This suggests that there is no significant association between windspeed_labled and month at the 5% significance level.
Fail to reject H0
This suggests that there is no significant association between windspeed_labled and dayofweek at the 5% significance level.
Fail to reject H0
This suggests that there is no significant association between windspeed_labled and season at the 5% significance level.
Fail to reject H0
This suggests that there is no significant association between windspeed_labled and holiday at the 5% significance level.
Fail to reject H0
This suggests that there is no significant association between windspeed_labled and weather at the 5% significance level.
Fail to reject H0
This suggests that there is no significant association between windspeed_labled and temp_labeled at the 5% significance level.
Fail to reject H0
This suggests that there is no significant association between windspeed_labled and humidity_labeled at the 5% significance level.
```

We can conclude that there is no significant assocication between any of the categorical variables.

Illustrate the insights based on EDA

8.

Set up Null Hypothesis (H0)

State the Alternate hypothesis (H1)

7. Select an appropriate test to check whether:

7.1. Working Day has effect on number of electric cycles rented

7.2. No. of cycles rented similar or different in different seasons

7.3. No. of cycles rented similar or different in different weather

7.4. Weather is dependent on season (check between 2 predictor variable)

7.1. Working Day has effect on number of electric cycles rented

(2 sample T - Test)

```
df.head()
```

	year	month	dayofweek	hour	season	holiday	workingday	weather	temp_labeled	humidity_labeled	windspeed_labled	registered	casual	count
0	0	0	5	0	0	0	0	0	0	0	0	13	3	16
1	0	0	5	1	0	0	0	0	0	0	0	32	8	40
2	0	0	5	2	0	0	0	0	0	0	0	27	5	32
3	0	0	5	3	0	0	0	0	0	0	0	10	3	13
4	0	0	5	4	0	0	0	0	0	0	0	1	0	1

Next steps: Generate code with df View recommended plots

```
df_final_t.head()
```

	year	month	dayofweek	hour	season	holiday	workingday	weather	temp_labeled	humidity_labeled	windspeed_labled	registered	casual	count
0	0	0	5	0	0	0	0	0	0	0	0	13	3	16
1	0	0	5	1	0	0	0	0	0	0	0	32	8	40
2	0	0	5	2	0	0	0	0	0	0	0	27	5	32
3	0	0	5	3	0	0	0	0	0	0	0	10	3	13
4	0	0	5	4	0	0	0	0	0	0	0	1	0	1

Next steps: Generate code with df_final_t View recommended plots

```
working_day_rentals = df_final_t[df_final_t["workingday"] == 1]["count"]

non_working_day_rentals = df_final_t[df_final_t["workingday"] == 0]["count"]
t_stat, p_value = stats.ttest_ind(working_day_rentals, non_working_day_rentals)
print(p_value)

if p_value < 0.05:
  print("Reject H0")
  print(f"This suggests that there is significant association between Working Day and Bikes Booked at the 5% significance level.")
else:
  print("Fail to reject H0")
  print(f"This suggests that there is no significant association between  Working Day and Bikes Booked  at the 5% significance level.")
```

```
0.22644804226361348
Fail to reject H0
This suggests that there is no significant association between  Working Day and Bikes Booked  at the 5% significance level.
```

```
(df_final_t.groupby(["workingday"])["count"].sum() / df_final_t.groupby(["workingday"])["count"].sum().sum())*100
```

```
workingday
0    31.40156
1    68.59844
Name: count, dtype: float64
```

69 percent bikes are booked on working day but there is no significant association between Working Day and Bikes Booked at the 5% significance level.

7.2. No. of cycles rented similar or different in different seasons

(ANNOVA)

```
df_final_t["season"].unique()
```

```
array([0, 1, 2, 3])
```

```
#a = ["year","month","dayofweek", "season", "holiday", "weather","temp_labeled", "humidity_labeled", "windspeed_labled"]

from scipy.stats import f_oneway # Numeric Vs categorical for many categories
from scipy.stats import ttest_ind # Numeric Vs categorical
f_stats, p_value = f_oneway(df_final_t[df_final_t["season"]== 0 ]["count"], df_final_t[df_final_t["season"]== 1 ]["count"],df_final_t[df_final_t["season"]== 2 ]["count"],df_final_t[df_final_t["season"]== 3 ]["count"])

print("test statistic:",f_stats)
print("p_value:",p_value)

if p_value < 0.05:
    print("Reject H0")
    print("Atleast one group have different mean")
    for i in range(0,4):
  # for j in range(i,13):
      j = i+1
      t_stat, p_value = ttest_ind(df_final_t[df_final_t["season"]== i ]["count"], df_final_t[df_final_t["season"]== j ]["count"])
      # print("p_value:",p_value)
      if p_value < 0.05:
          print("Reject H0")
          print(f'This means that we have enough evidence to conclude that the means of the two income product groups season{i} and season{j} are  statistically different from each other.')
      else:
          print("Fail to reject H0")
          print(f'This means that we do not have enough evidence to conclude that the means of the two income product groups season{i} and season{j} are statistically different from each other.')

else:
    print("Fail to reject H0")
```

```
test statistic: 236.94671081032106
p_value: 6.164843386499654e-149
Reject H0
Atleast one group have different mean
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups season0 and season1 are  statistically different from each other.
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups season1 and season2 are  statistically different from each other.
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups season2 and season3 are  statistically different from each other.
Fail to reject H0
This means that we do not have enough evidence to conclude that the means of the two income product groups season3 and season4 are statistically different from each other.
```

```
similarseason_rentals = df_final_t[(df_final_t["season"] == 3) & (df_final_t["season"] == 4)]["count"]

differnetseason_rentals = df_final_t[(df_final_t["season"] == 0) & (df_final_t["season"] == 1) & (df_final_t["season"] == 2)]["count"]
t_stat, p_value = stats.ttest_ind(working_day_rentals, non_working_day_rentals)
print(p_value)

if p_value < 0.05:
  print("Reject H0")
  print(f"This suggests that there is significant association between Similar Seasong and Bikes Booked at the 5% significance level.")
else:
  print("Fail to reject H0")
  print(f"This suggests that there is no significant association between  Different Season, Similar Season and Bikes Booked  at the 5% significance level.")
```

```
0.22644804226361348
Fail to reject H0
This suggests that there is no significant association between  Different Season, Similar Season and Bikes Booked  at the 5% significance level.
```

We can conclude that there is relation between Season3 and Season4

Double-click (or enter) to edit

```
Start coding or generate with AI.
```

7.3. No. of cycles rented similar or different in different weather

(Chi Square Test)

```
#a = ["year","month","dayofweek", "season", "holiday", "weather","temp_labeled", "humidity_labeled", "windspeed_labled"]

from scipy.stats import f_oneway # Numeric Vs categorical for many categories
from scipy.stats import ttest_ind # Numeric Vs categorical
f_stats, p_value = f_oneway(df_final_t[df_final_t["weather"]== 0 ]["count"], df_final_t[df_final_t["weather"]== 1 ]["count"],df_final_t[df_final_t["weather"]== 2 ]["count"],df_final_t[df_final_t["weather"]== 3 ]["count"])

print("test statistic:",f_stats)
print("p_value:",p_value)

if p_value < 0.05:
    print("Reject H0")
    print("Atleast one group have different mean")
    for i in range(0,4):
  # for j in range(i,13):
      j = i+1
      t_stat, p_value = ttest_ind(df_final_t[df_final_t["weather"]== i ]["count"], df_final_t[df_final_t["weather"]== j ]["count"])
      # print("p_value:",p_value)
      if p_value < 0.05:
          print("Reject H0")
          print(f'This means that we have enough evidence to conclude that the means of the two income product groups weather{i} and weather{j} are  statistically different from each other.')
      else:
          print("Fail to reject H0")
          print(f'This means that we do not have enough evidence to conclude that the means of the two income product groups weather{i} and weather{j} are statistically different from each other.')

else:
    print("Fail to reject H0")
```

```
test statistic: 65.53024112793271
p_value: 5.482069475935669e-42
Reject H0
Atleast one group have different mean
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups weather0 and weather1 are  statistically different from each other.
Reject H0
This means that we have enough evidence to conclude that the means of the two income product groups weather1 and weather2 are  statistically different from each other.
Fail to reject H0
This means that we do not have enough evidence to conclude that the means of the two income product groups weather2 and weather3 are statistically different from each other.
Fail to reject H0
This means that we do not have enough evidence to conclude that the means of the two income product groups weather3 and weather4 are statistically different from each other.
```

Weather 2,3 and 4 are statistically similar

7.4. Weather is dependent on season (check between 2 predictor variable)

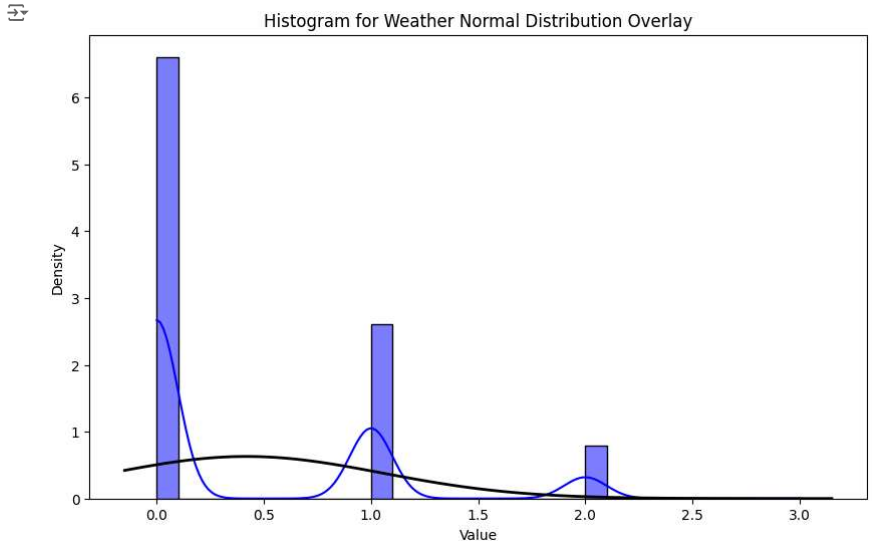
Start coding or generate with AI.

Start coding or generate with AI.

9.

Check assumptions of the test (Normality, Equal Variance). You can check it using Histogram, Q-Q plot or statistical methods like levene's test, Shapiro-wilk test (optional) Please continue doing the analysis even If some assumptions fail (levene's test or Shapiro-wilk test) but double check using visual analysis and report wherever necessary

```
# df = pd.DataFrame(data, columns=['value'])
from scipy.stats import norm
# # Set Seaborn style
# sns.set(style="whitegrid")

# Plot the histogram with density plot
plt.figure(figsize=(10, 6))
sns.histplot(df['weather'], kde=True, bins=30, color='blue', stat='density')

# Overlay a normal distribution curve
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = norm.pdf(x, df['weather'].mean(), df['weather'].std())
plt.plot(x, p, 'k', linewidth=2)

# Add titles and labels
plt.title('Histogram for Weather Normal Distribution Overlay')
plt.xlabel('Value')
plt.ylabel('Density')

# Show the plot
plt.show()
```

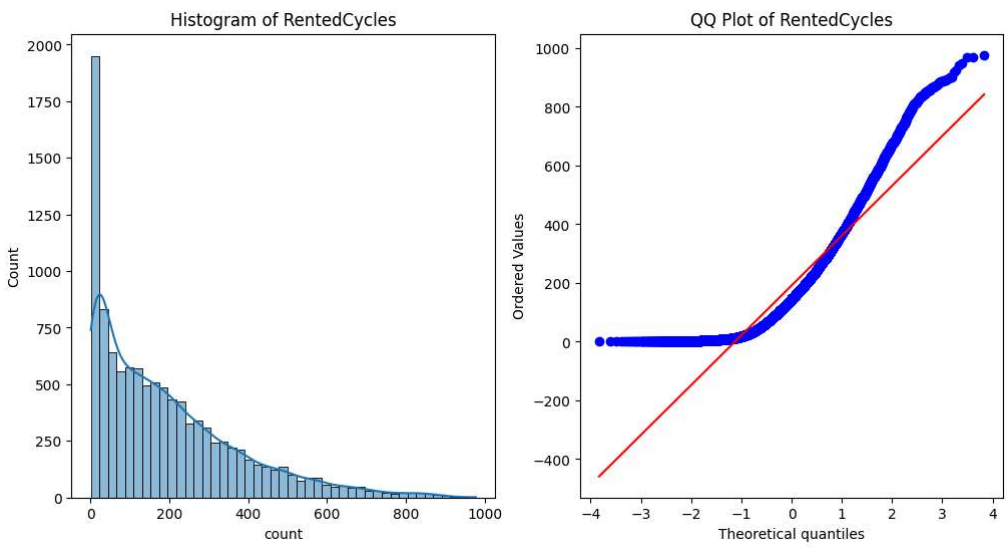


CHecking with Shapiro Wilk test for Normal Distribution

```
import pandas as pd
import scipy.stats as stats
from scipy.stats import shapiro

a = ["year","month","dayofweek", "season", "holiday", "weather","temp_labeled", "humidity_labeled", "windspeed_labled","count"]
sample_df = df_final_t.sample(n=1000, random_state=42)

for i in a :
    stat, p_value = shapiro(sample_df[i])

    # Output the results
    # print(f"Test Statistic: {stat}")
    # print(f"p-value: {p_value}")

    # Interpretation
    alpha = 0.05
    if p_value > alpha:
        print(f"Fail to reject the null hypothesis (H0): Data is normally distributed for {i}")
    else:
        print(f"Reject the null hypothesis (H0): Data is not normally distributed for {i}")
```

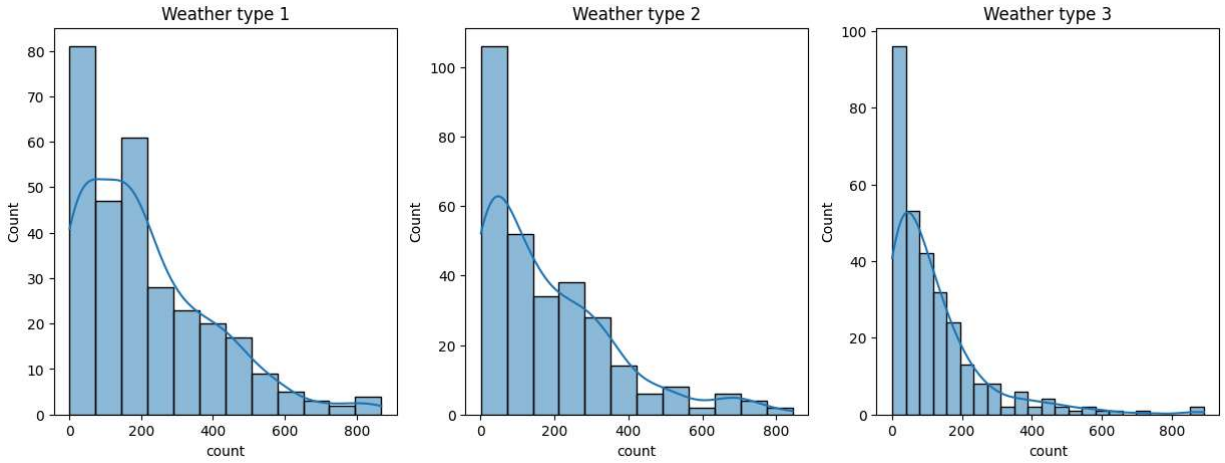
```
Reject the null hypothesis (H0): Data is not normally distributed for year
Reject the null hypothesis (H0): Data is not normally distributed for month
Reject the null hypothesis (H0): Data is not normally distributed for dayofweek
Reject the null hypothesis (H0): Data is not normally distributed for season
Reject the null hypothesis (H0): Data is not normally distributed for holiday
Reject the null hypothesis (H0): Data is not normally distributed for weather
Reject the null hypothesis (H0): Data is not normally distributed for temp_labeled
Reject the null hypothesis (H0): Data is not normally distributed for humidity_labeled
Reject the null hypothesis (H0): Data is not normally distributed for windspeed_labled
Reject the null hypothesis (H0): Data is not normally distributed for count
```

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

# Plot a histogram and a QQ plot
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
sns.histplot(df['count'], kde=True)
plt.title('Histogram of RentedCycles')

plt.subplot(1, 2, 2)
stats.probplot(df['count'], dist="norm", plot=plt)
plt.title('QQ Plot of RentedCycles')

plt.show()
```




```
sample_1 = df_final_t[df_final_t['weather'] == 0].sample(300)['count']
sample_2 = df_final_t[df_final_t['weather'] == 1].sample(300)['count']
sample_3 = df_final_t[df_final_t['weather'] == 2].sample(300)['count']
```

```
fig = plt.figure(figsize=(15,5))
ax2 = fig.add_subplot(131)
sns.histplot(sample_1,kde=True,ax=ax2)
ax2.set_title("Weather type 1")
ax = fig.add_subplot(132)
sns.histplot(sample_2,kde=True,ax=ax)
ax.set_title("Weather type 2")
ax3 = fig.add_subplot(133)
sns.histplot(sample_3,kde=True,ax=ax3)
ax3.set_title("Weather type 3")
plt.show()
```

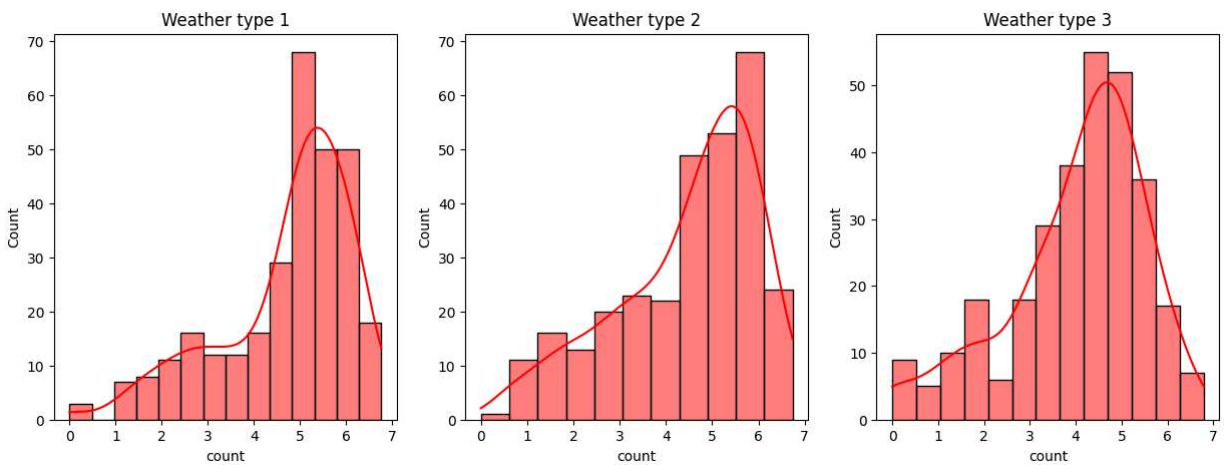


We see that none of the weather type distributions follow normal distribution.

Hence we apply log transformation to make the distributions near to normal.

```
log_1 = np.log(sample_1)
log_2 = np.log(sample_2)
log_3 = np.log(sample_3)
```

```
fig = plt.figure(figsize=(15,5))
ax2 = fig.add_subplot(131)
sns.histplot(log_1,kde=True,ax=ax2,color='r')
ax2.set_title("Weather type 1")
ax = fig.add_subplot(132)
sns.histplot(log_2,kde=True,ax=ax,color='r')
ax.set_title("Weather type 2")
ax3 = fig.add_subplot(133)
sns.histplot(log_3,kde=True,ax=ax3,color='r')
ax3.set_title("Weather type 3")
plt.show()
```



We have satified the conditions wherein samples must be independent and the groups have equal sample size. We should now check for the homogenity of variance and the normality of the distribution.

We perform Levene's test to check the homogenity of variances

H0 : The variance is equal across all groups Ha : The variance is not equal across the groups

```
stats.levene(log_1,log_2,log_3,center='median')
```

```
LeveneResult(statistic=1.4716667353099293, pvalue=0.2300962663971127)
```

- Since pvalue is not less than 0.05, we fail to reject null hypothesis.
- This means we do not have sufficient evidence to say that variance across different weather's is significantly different thus making the assumption of homogenity of variances true.

Shapiro-Wilk test to check the normality of the distribution

```
'{:.20f}'.format(shapiro(log_1)[1])
```

```
'0.00000000000014121487'
```

From the above output, we see that the p value is far less than 0.05, Hence we reject the null hypothesis. We have sufficient evidence to say that the weather type 1 sample data does not come from normal distribution.

```
'{:.20f}'.format(shapiro(log_2)[1])
```

```
'0.00000000002752700325'
```

From the above output, we see that the p value is far less than 0.05, Hence we reject the null hypothesis. We have sufficient evidence to say that the weather type 2 sample data does not come from normal distribution.

Checking the normality of weather type 3 samples

We select the level of signifiance as 5% and the null and alternate hypothesis is as follows:

H0 : The weather type 3 samples are normally distributed Ha: The weather type 3 samples are not normally distributed.

```
'{:.20f}'.format(shapiro(log_3)[1])
```

```
'0.00000000053056103955'
```

- From the above output, we see that the p value is far less than 0.05, Hence we reject the null hypothesis.
- We have sufficient evidence to say that the **weather type 3 sample data** does not come from normal distribu

```
from scipy.stats import f_oneway
f_oneway(log_1,log_2,log_3)
```

```
F_onewayResult(statistic=16.417407362197125, pvalue=9.940496778293128e-08)
```

Conclusion

The f test statistic value is 13.35 and the corresponding pvalue is 0.00000193497406718207. Since the pvalue is lesser than 0.05 we reject the null hypothesis.

This means we have sufficient evidence to say that there is a difference in number of bikes rented among the 3 different types of weather conditions.

10. Inference from the analysis

Conclusion Based on P value

Final Conclusion

Some of the variables that influence electric cycles demand are:

- Weather : Electric cycles are most sought on weather type 1 followed by 2 and 3 (We omit 4 since we do not have much data)

> We even performed ANNOVA test to verify that there is a significant difference amongst Yulu bikes being rented across different weather types.

- Season : Electric cycles are most sought on Fall season followed by Winter, Summer and Spring.

> We have verified the statistical significance here as well with the help of ANNOVA to check if there is difference in Yulu bikes being rented across different seasons.

- Fridays and Wednesdays are the days when rides are take the most. Though the differnece is not much. The least dip could be seen on Sunday followed by Tuesday. Yulu can offer higher discount on Sunday and Tuesday to push the demand on Sunday as no of bikes avaliable remains same.
- 60 percent of rides are from season 2 and 3. There are lots of outliers during season 1 and season 4. We can offer discounts during season 1, this would boost the demand as number of rides unused would decrease.
- 97 percent of bikes are booked when there is no holiday.