```
#importing the Libraries for EDA
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from scipy.stats import norm,ttest_1samp,ttest_ind,chisquare,chi2,f_oneway,chi2_contingency

from statsmodels.graphics.gofplots import qqplot


#downloading the Data
! gdown https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/original/bike_sharing.csv?1642089089
```

```
Downloading...
From: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/original/bike_sharing.csv?1642089089
To: /content/bike_sharing.csv?1642089089
100% 648k/648k [00:00<00:00, 8.04MB/s]
```

```
#Reading the Data and converting into Dataframe
data=pd.read_csv("bike_sharing.csv?1642089089")
df=pd.DataFrame(data)


#head function is used to get the top n rows bu default we will get 5 rows
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

```
#Tail function is used to get the bottom n rows bu default we will get bottom 5 rows
df.tail()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
10881	2012-12-19 19:00:00	4	0	1	1	15.58	19.695	50	26.0027	7	329	336
10882	2012-12-19 20:00:00	4	0	1	1	14.76	17.425	57	15.0013	10	231	241
10883	2012-12-19 21:00:00	4	0	1	1	13.94	15.910	61	15.0013	4	164	168
10884	2012-12-19 22:00:00	4	0	1	1	13.94	17.425	61	6.0032	12	117	129
10885	2012-12-19 23:00:00	4	0	1	1	13.12	16.665	66	8.9981	4	84	88

```
#Statical summary about Data
df.describe()
```

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	re
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000	10886.000000	10886.000000	10886.000000	1088
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.886460	12.799395	36.021955	15
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.245033	8.164537	49.960477	15
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	0.000000	0.000000	0.000000	
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	47.000000	7.001500	4.000000	3
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	62.000000	12.998000	17.000000	11
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	77.000000	16.997900	49.000000	22
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	100.000000	56.996900	367.000000	88

```
#INFO function is used to get information about Rows and Columns
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
```

```
#ISNA function is uswed to get the Bollen  output if is there missing values
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
```

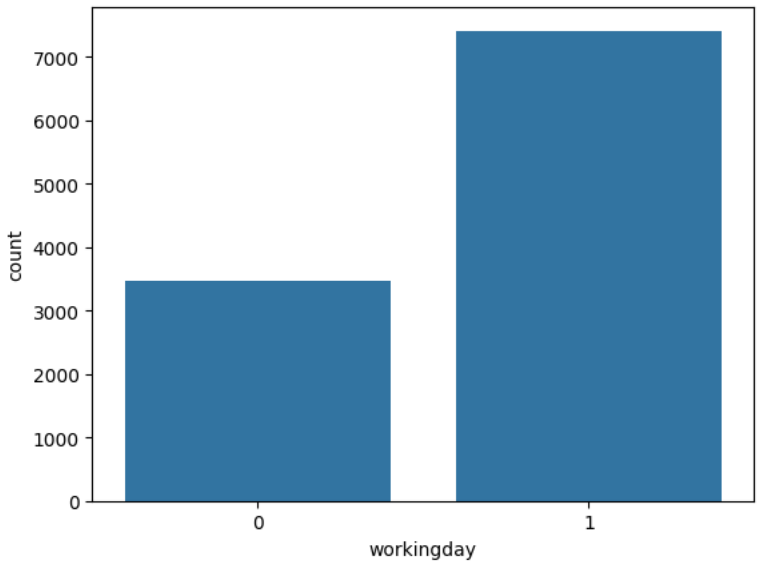
```
#Dtypes will retrive the datatypes of each columns
df.dtypes
```

```
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
```

```
#Here converting Object to  datatime of datetime column
df["datetime"]=pd.to_datetime(df["datetime"])
```

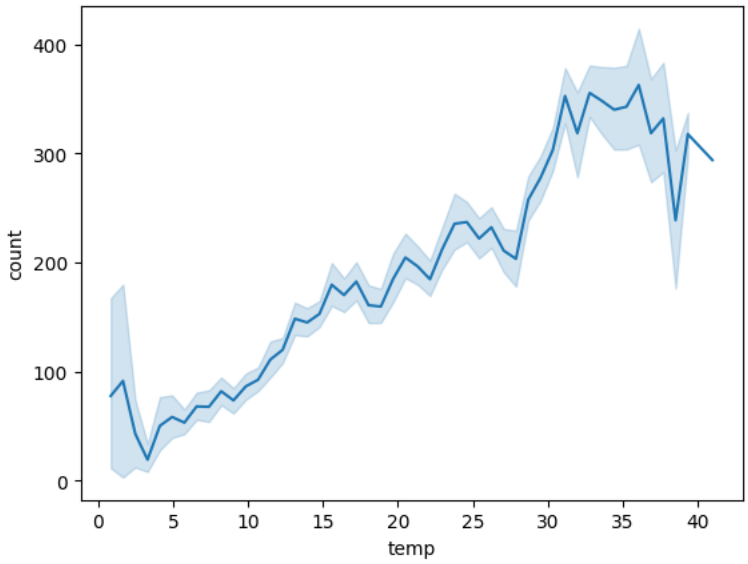
```
#plot the countplot for workingday
sns.countplot(x='workingday',data=df)
```

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



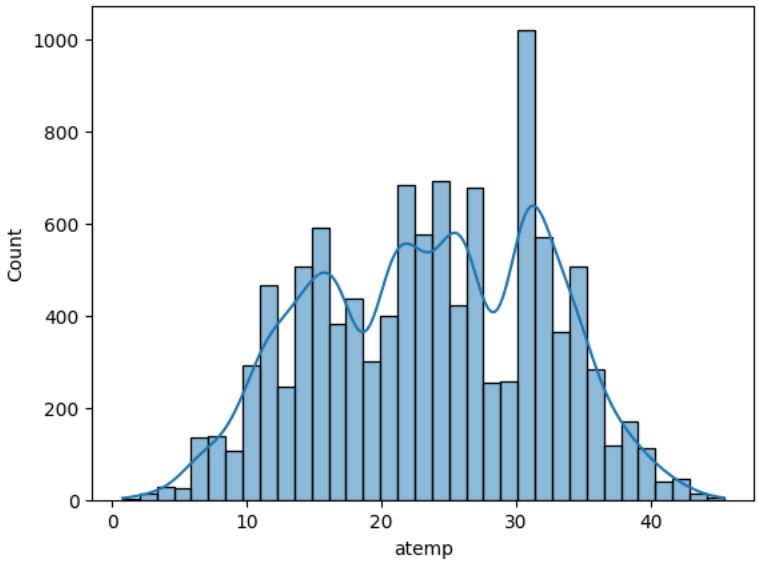

```
# plot lineplot for temp and count
sns.lineplot(x="temp", y="count", data=df)
```

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



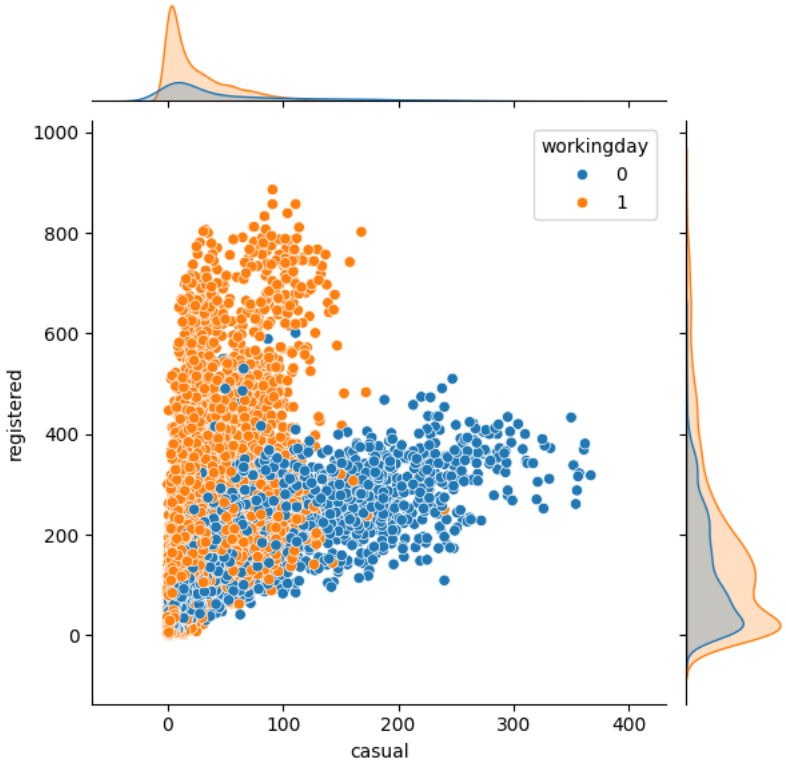

```
#plotting Histogra for temperaturte
sns.histplot(x="atemp",data=df,kde=True)
```

```
<Axes: xlabel='atemp', ylabel='Count'>
```



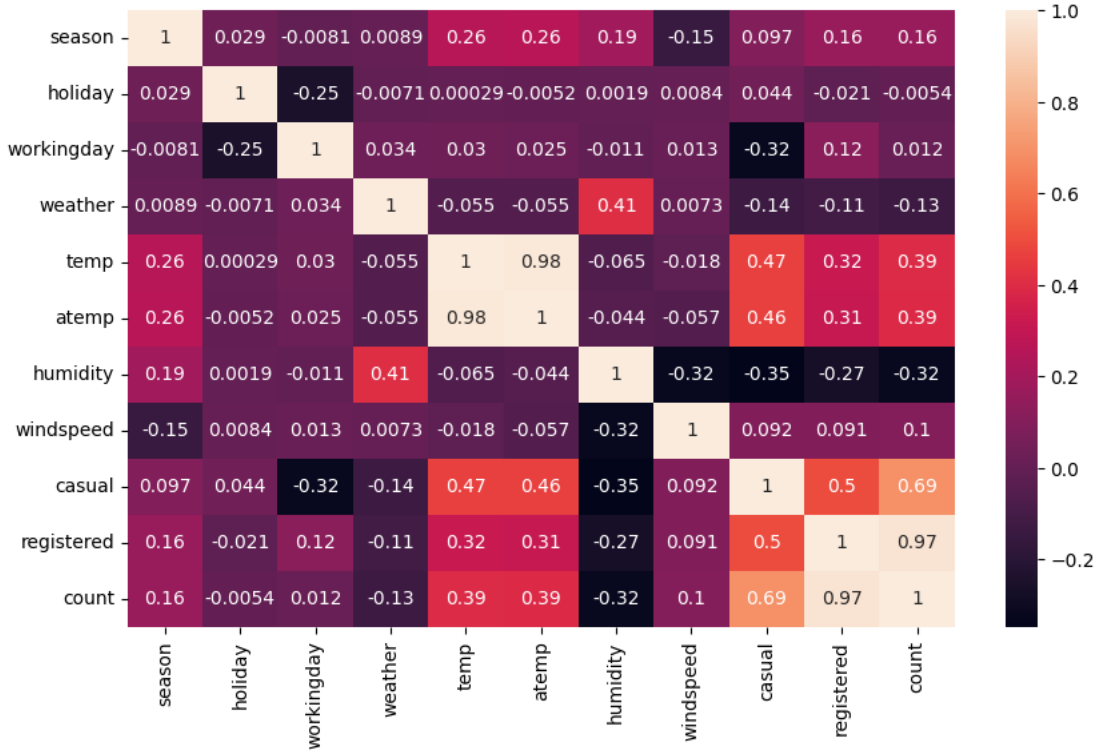

```
sns.jointplot(data=df,x="casual",y="registered",hue="workingday")
```

```
<seaborn.axisgrid.JointGrid at 0x7b0739650520>
```



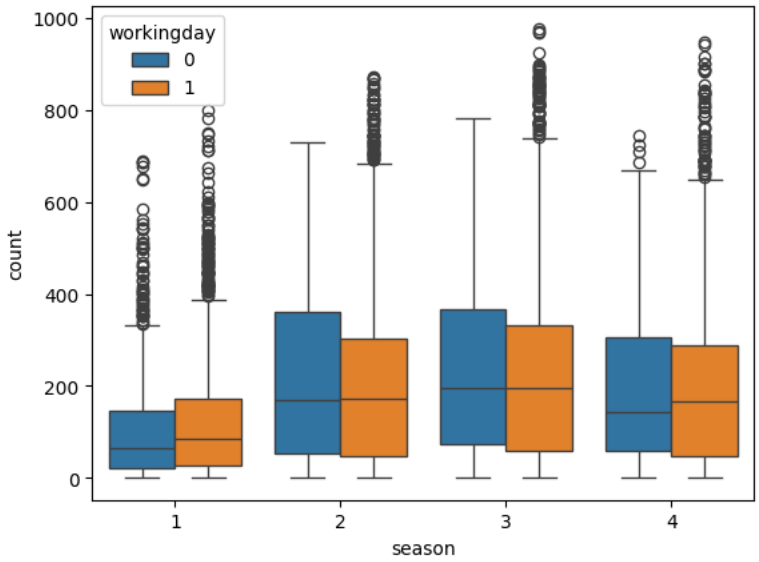

```
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(),annot=True)
plt.show()
```

```
<ipython-input-15-6e3f148dfcfc>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future ver
  sns.heatmap(df.corr(),annot=True)
```




```
sns.boxplot(data=df,x="season",y="count",hue="workingday")
```

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



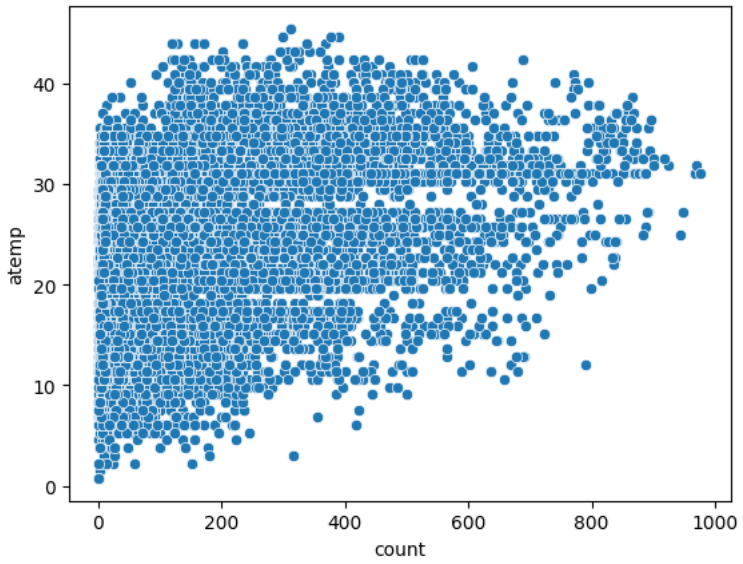

```
df.groupby(["season","workingday"])["count","casual"].agg(["sum","count"])
```

```
<ipython-input-17-d980b5b5f184>:1: FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of keys) will be depr
  df.groupby(["season","workingday"])["count","casual"].agg(["sum","count"])
```

		count		casual	
		sum	count	sum	count
season	workingday				
1	0	90714	858	22836	858
	1	221784	1828	18769	1828
2	0	187062	840	67089	840
	1	401220	1893	62583	1893
3	0	206478	888	73037	888
	1	434184	1845	69681	1845
4	0	170618	888	43075	888
	1	373416	1846	35065	1846

```
sns.scatterplot(data=df,x="count",y="atemp")
```

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



Hypothesis Testing

2- Sample T-Test to check if Working Day has an effect on the number of electric cycles rented

```
# H0 : null hypothsis  : There is no statistically significant difference in the number of electric cycles rented on working days and nc
# Ha : Alternative hypothsis : There is a statistically significant difference in the number of electric cycles rented on working days a


#2- Sample T-Test to check if Working Day has an effect on the number of electric cycles rented
#considering the siginficant value is 5%.

working_days = df[df['workingday'] == 1]['count']
non_working_days = df[df['workingday'] == 0]['count']

ttest, p_value = ttest_ind(working_days, non_working_days)
print(p_value)
if p_value < 0.05:
  print("Reject Null hypothesis")
  print("There is a statistically significant difference in the number of electric cycles rented on working days and non-working days.")
else:
  print("Fail to reject Null  hypothesis")
  print("There is no statistically significant difference in the number of electric cycles rented on working days and non-working days.'


    0.22644804226361348
    Fail to reject Null  hypothesis
    There is no statistically significant difference in the number of electric cycles rented on working days and non-working days.
```

From the above statistical values we can conclude that working day or non-working days are independent on the for number of electric cycles rented.

ANNOVA Test : ANNOVA to check if No. of cycles rented is similar or different in different weather and season

```
df.head(2)
```

	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

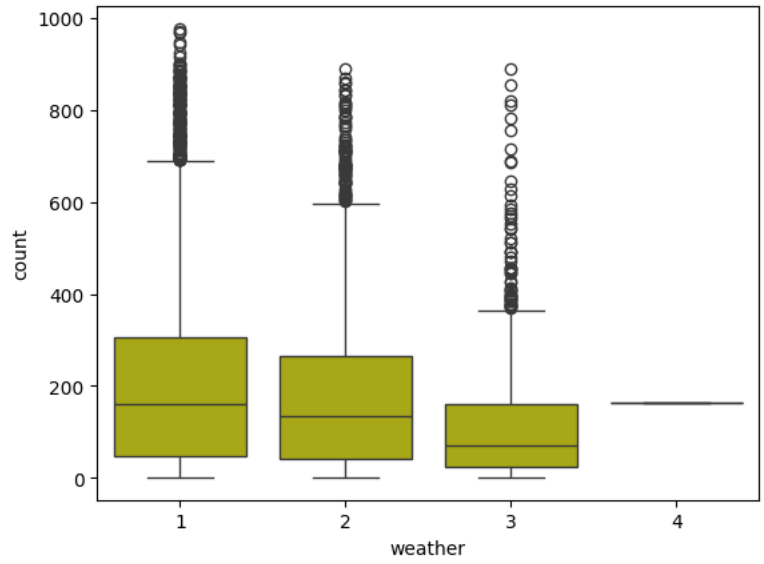
Weather Testing

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

```
1    7192
2    2834
3     859
4       1
Name: weather, dtype: int64
```

from the Above data we can neglect the weather equals to 4 due to the less data we have for weather 4

```
#check the medains for weather using plot

sns.boxplot(x="weather", y="count", data=df,color="y")
plt.show()
```



Weather Medians are different from each others

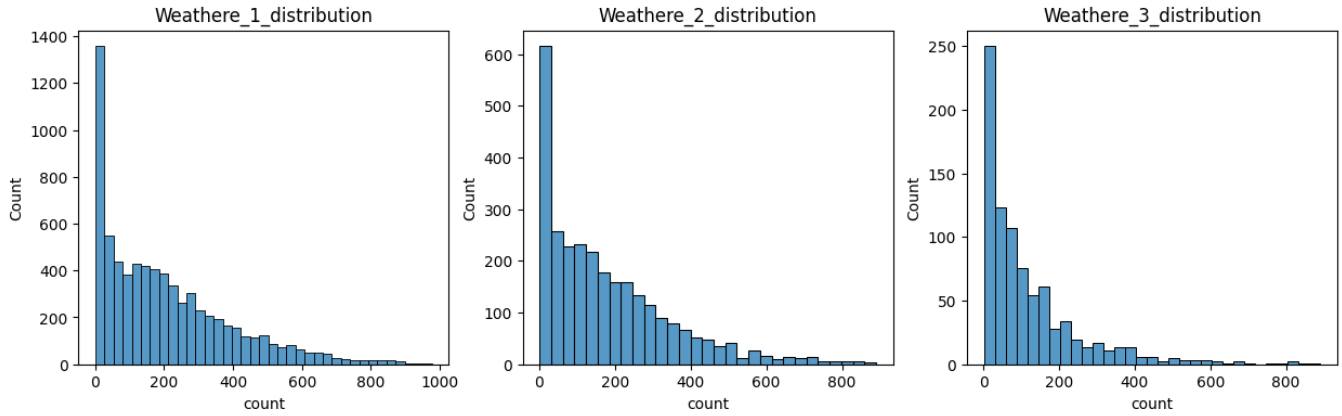
for performing the statisical test for the weathere we have the more than 2 categories so we will go with ANNOVA Test.

before going with ANNOVA test, we nned to check the Assumptions of Annova

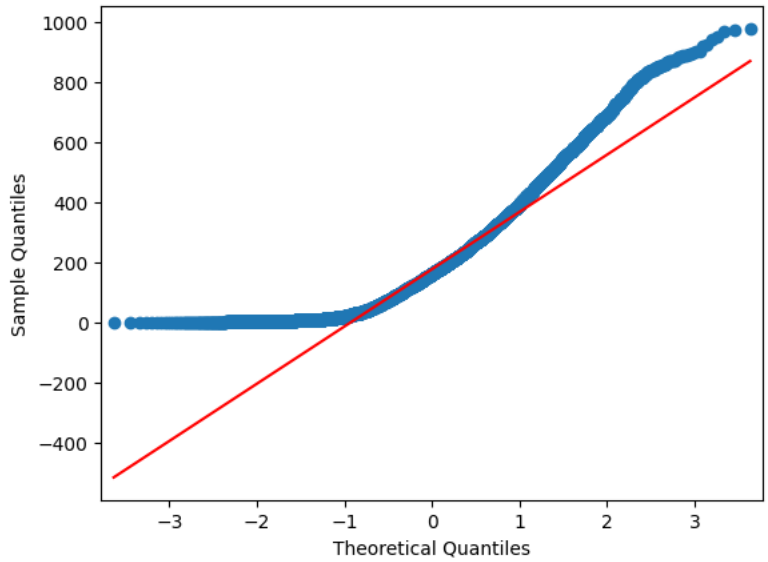
```
w1=df[df["weather"]==1]["count"]
w2=df[df["weather"]==2]["count"]
w3=df[df["weather"]==3]["count"]
```

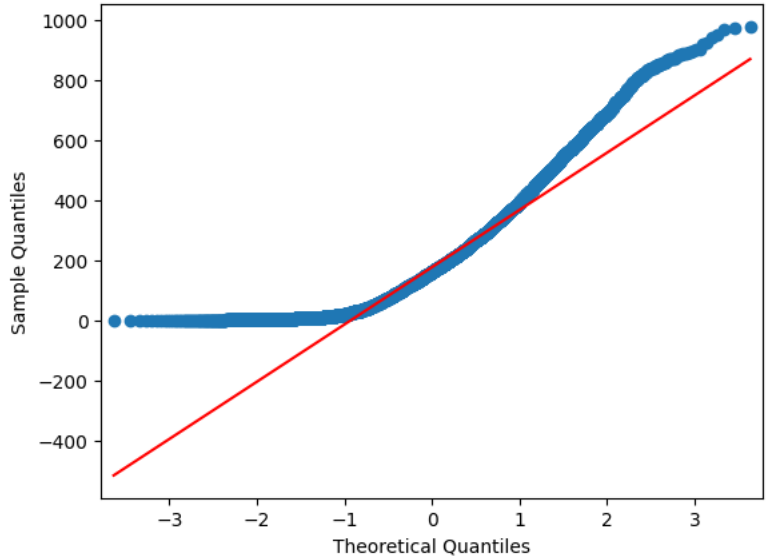
```
#Assumption 1 : wheather the data is following normal distribution or not
```

```
plt.figure(figsize=(15,4))

plt.subplot(1,3,1)
plt.title("Weathere_1_distribution")
sns.histplot(w1)

plt.subplot(1,3,2)
plt.title("Weathere_2_distribution")
sns.histplot(w2)

plt.subplot(1,3,3)
plt.title("Weathere_3_distribution")
sns.histplot(w3)

plt.show()
```




```
#by using the QQPLOT also we can conclude that wheather data is following linear or Non-linear .
qqplot(w1,line='q')
```






```
# Assumption 2: Homogeneity of variances
# Levene's test
from scipy.stats import levene
stat, p = levene(w1, w2, w3)
print(p)
if p < 0.05:
  print("Reject Null hypothesis")
  print("The variances are not equal.")
else:
  print("Fail to reject Null  hypothesis")
  print("The variances are equal.")
```

```
    6.198278710731511e-36
    Reject Null hypothesis
    The variances are not equal.
```

Assumption 1 if the data is not a normal distribution the ANNOVA Test got failed

Assumptions 2 ANNOVA test is got failed because of the variances are not equal.

Assumption 3 Independence of observations This assumption is met if the data are collected independently of each other.

```
from scipy.stats import kruskal
stats,p_value=kruskal(w1, w2, w3)
print(p_value)
if p_value < 0.05:
  print("Reject Null hypothesis")
  print("There is a statistically significant difference in the number of electric cycles rented in different weather conditions.")
else:
  print("Fail to reject Null  hypothesis")
  print("There is no statistically significant difference in the number of electric cycles rented in different weather conditions.")
```

```
    3.122066178659941e-45
    Reject Null hypothesis
```

```
There is a statistically significant difference in the number of electric cycles rented in different weather conditions.
```

```
W1_sample=w1.sample(800)
W2_sample=w2.sample(800)
W3_sample=w3.sample(800)
```

```
# Perform the ANOVA test
#f_stat, p_value = f_oneway(w1, w2, w3)
f_stat, p_value = f_oneway(W1_sample, W2_sample, W3_sample)
print(p_value)
if p_value < 0.05:
  print("Reject Null hypothesis")
  print("There is a statistically significant difference in the number of electric cycles rented in different weather conditions.")
else:
  print("Fail to reject Null  hypothesis")
  print("There is no statistically significant difference in the number of electric cycles rented in different weather conditions.")
```

```
5.827000003331273e-23
Reject Null hypothesis
There is a statistically significant difference in the number of electric cycles rented in different weather conditions.
```

From the above statisticall value we can conclude that number of electric cycles rented are depending on the weather conditions.

SEASON TEST

```
df["season"].value_counts()
```

```
4    2734
2    2733
3    2733
1    2686
Name: season, dtype: int64
```

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

```
season
1    312498
2    588282
3    640662
4    544034
Name: count, dtype: int64
```

As we have more than 2 categories we'll go with the ANNOVA test.

so we need check the Assumptions of Annova test to perform this statistical test.

```
S1=df[df["season"]==1]["count"]
S2=df[df["season"]==2]["count"]
S3=df[df["season"]==3]["count"]
S4=df[df["season"]==4]["count"]
```

```
# Assumption 1: Homogeneity of variances
# Levene's test
from scipy.stats import levene
stat, p = levene(S1, S2, S3,S4)
print(p)
if p < 0.05:
  print("Reject Null hypothesis")
  print("The variances are not equal.")
else:
  print("Fail to reject Null  hypothesis")
  print("The variances are equal.")
```

```
#checking Medians for seasons
sns.boxplot(data=df,x="season",y="count")
```

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

Hence the Seasons variance is diffrenet from each others, Hence one of the ANNOVA Assumptions got failed

Assumption 2 checking the Normal distribution for seasons

```
plt.figure(figsize=(15,10))
```

```
plt.figure(figsize=(15,10))

plt.subplot(2,2,1)
plt.title("Season_1_distribution")
sns.histplot(S1)

plt.subplot(2,2,2)
plt.title("Season_2_distribution")
sns.histplot(S2)

plt.subplot(2,2,3)
plt.title("Season_3_distribution")
sns.histplot(S3)

plt.subplot(2,2,4)
plt.title("Season_4_distribution")
sns.histplot(S4)


plt.show()
```

```
#ANNOVA to check if No. of cycles rented is similar or different season

from scipy.stats import f_oneway
season_1=S1.sample(2000)
season_2=S2.sample(2000)
season_3=S3.sample(2000)
season_4=S4.sample(2000)
f_value,p_value=f_oneway(season_1,season_2,season_3,season_4)
print(p_value)
if p_value < 0.05:
  print("Reject Null hypothesis")
  print("There is a statistically significant difference in the number of electric cycles rented in different seasons.")
else:
  print("Fail to reject Null  hypothesis")
  print("There is no statistically significant difference in the number of electric cycles rented in different seasons.")
```

```
1.565633649607611e-113
Reject Null hypothesis
There is a statistically significant difference in the number of electric cycles rented in different seasons.
```

Therefore the average number of rental bikes is statistically different for different seasons

Chi-square test to check if Weather is dependent on the season

```
# Chi-square test to check if Weather is dependent on the season

df_cross = pd.crosstab(df['season'], df['weather'])
chi2, p, dof, expected = chi2_contingency(df_cross)
print(p)
if p < 0.05:
  print("Reject Null hypothesis")
  print("Weather is dependent on the season")
else:
  print("Fail to reject Null  hypothesis")
  print("Weather is not dependent on the season")
```

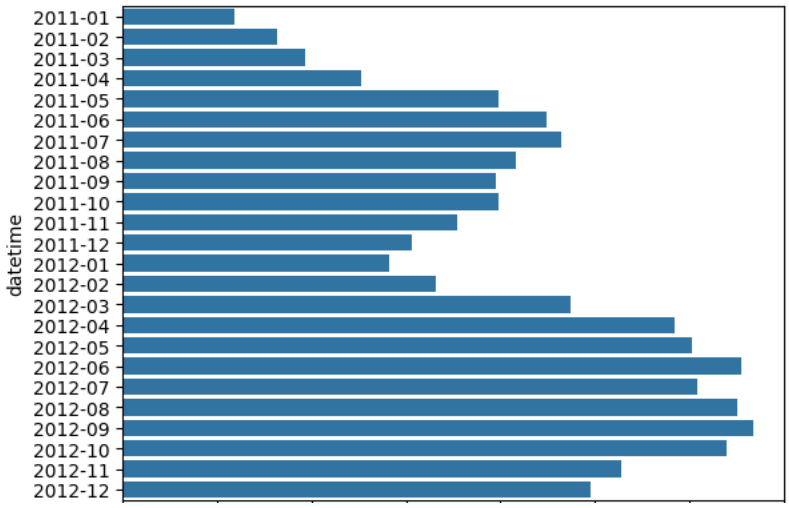
```
1.549925073686492e-07
Reject Null hypothesis
Weather is dependent on the season
```

INSIGHTS

```
# getting the only date from the datetime
import datetime as dt
df["datetime"]=pd.to_datetime(df["datetime"])
df["date"] = df["datetime"].dt.strftime("%Y-%m-%d")
```

```
monthly_count=df.groupby(df["datetime"].dt.strftime("%Y-%m"))["count"].aggregate("sum").reset_index()
#plt.figure(figsize=(6,15))
sns.barplot(data=monthly_count,y="datetime",x="count")
```

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




```
#getting the specified temp column to get the count of rental bikes

df['new_temp'] = df['temp'].apply(lambda x: 1 if x < 18 else 2 if x >= 18 and x < 28 else 3)


hour_count=df.groupby(df["datetime"].dt.strftime("%H"))["count"].aggregate("sum").reset_index().sort_values("count")
hour_count
```

-->Tha data is given from Timestamp(2011-01-01 00:00.00) f0 Timestamp(2012-12-19 23:00:00/) The total tima penod for which the data is given is 718 days 23:00:00"

-->compare to all seasons summer and Rainy season has more count spring has less count because of the Festival season customers may be not in that town 1:spring, 2: summer, 3: Rainy, 4: winter

-->Workingdays has more registraion count compare to the holidays...

-->When the weather is having Clear or Few clouds then the more customers are booking the bikes and most of the retal bikes are bokked in the

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.