```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.graphics.gofplots import qqplot
from scipy.stats import ttest_ind, chisquare, shapiro,levene,chi2_contingency, kruskal
```

```
! gdown https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/original/bike_sharing.csv
```

```
Downloading...
From: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/original/bike_sharing.csv
To: /content/bike_sharing.csv
100% 648k/648k [00:00<00:00, 15.3MB/s]
```

```
df = pd.read_csv('bike_sharing.csv')
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
	2011-01-01											

Next steps: Generate code with df View recommended plots

```
df.shape
```

```
(10886, 12)
```

```
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
```

```
#As we can see here in our dataset there are zero null values
```

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

```
df.describe()
```

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	cas
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000	10886.000000	10886.000000	10886.000
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.886460	12.799395	36.021
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.245033	8.164537	49.960
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	0.000000	0.000000	0.000
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	47.000000	7.001500	4.000
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	62.000000	12.998000	17.000
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	77.000000	16.997900	49.000
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	100.000000	56.996900	367.000

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

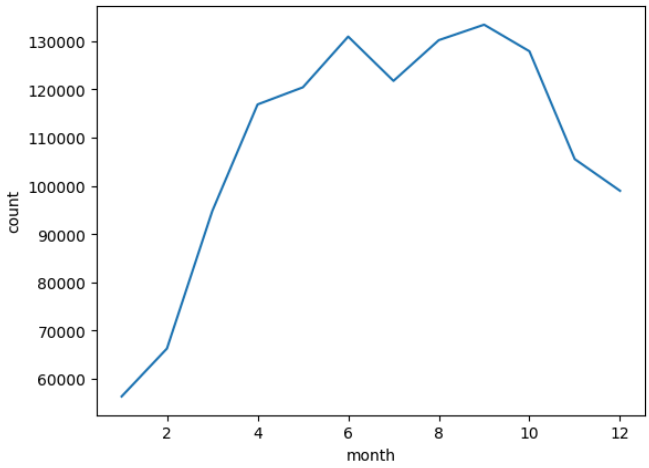
```
d_data=df[["datetime","count"]]
d_data["year"]=d_data["datetime"].dt.year
d_data["month"]=d_data["datetime"].dt.month
d_data=pd.DataFrame(d_data.groupby(["year","month"])["count"].sum()).reset_index()
recent_data=d_data[d_data["year"]==2012]
recent_data["per_increase"]=recent_data["count"].pct_change()*100
recent_data
```

```
<ipython-input-10-d9f81759967b>:2: 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-vie
  d_data["year"]=d_data["datetime"].dt.year
<ipython-input-10-d9f81759967b>:3: 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-vie
  d_data["month"]=d_data["datetime"].dt.month
<ipython-input-10-d9f81759967b>:6: 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-vie
  recent_data["per_increase"]=recent_data["count"].pct_change()*100
```

	year	month	count	per_increase
12	2012	1	56332	NaN
13	2012	2	66269	17.640062
14	2012	3	94766	43.002007
15	2012	4	116885	23.340650
16	2012	5	120434	3.036318
17	2012	6	130957	8.737566
18	2012	7	121769	-7.016043
19	2012	8	130220	6.940190
20	2012	9	133425	2.461219
21	2012	10	127912	-4.131909
22	2012	11	105551	-17.481550
23	2012	12	98977	-6.228269

Next steps: Generate code with `recent_data` View recommended plots

```
sns.lineplot(data=recent_data, x="month",y="count")
plt.show()
```




```
#From the above graph we can see that count of yulu vehicle is dropped in 12 month of 2012.
```

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

```
continous_data=df[["temp","atemp","humidity",   "windspeed","count"]]
continous_data.head()
```

	temp	atemp	humidity	windspeed	count
0	9.84	14.395	81	0.0	16
1	9.02	13.635	80	0.0	40
2	9.02	13.635	80	0.0	32
3	9.84	14.395	75	0.0	13
4	9.84	14.395	75	0.0	1

Next steps: Generate code with continous_data View recommended plots

```
discrete_data=df[["season","holiday","workingday","weather","casual","registered","count"]]
discrete_data.head()
```

	season	holiday	workingday	weather	casual	registered	count
0	1	0	0	1	3	13	16
1	1	0	0	1	8	32	40
2	1	0	0	1	5	27	32
3	1	0	0	1	3	10	13
4	1	0	0	1	0	1	1

Next steps: Generate code with discrete_data View recommended plots

Analysis on Discrete Data

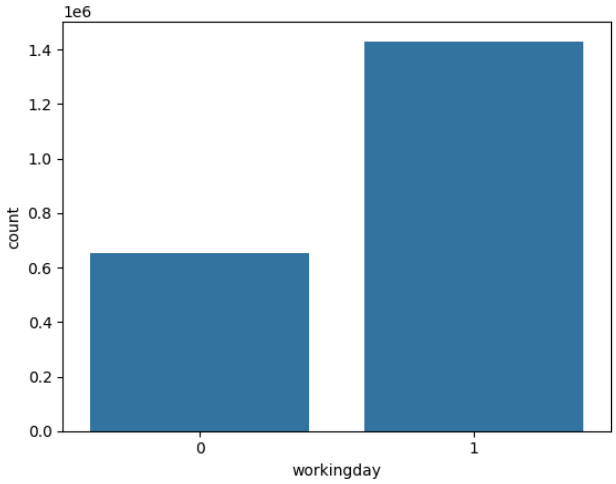
Working day affect on count

```
work_day=pd.DataFrame(discrete_data.groupby("workingday")["count"].sum()).reset_index()
work_day
```

	workingday	count
0	0	654872
1	1	1430604

Next steps: Generate code with work_day View recommended plots

```
sns.barplot(data=work_day,x="workingday",y="count")
plt.show()
```




```
#As we can see on overall working day have more bikes rented count.
pd.DataFrame(discrete_data.groupby("workingday")["count"].mean()).reset_index()
```

	workingday	count
0	0	188.506621
1	1	193.011873

```
#Now lets test the Hypothesis for is Working day have more count of bike or not.
# Ho=avg count of rented bike on working day and Non-working day are same
# Ha=avg count of rented bike on working day is less that  Non-working day
# With 5% significance level


work_sampl=discrete_data[discrete_data["workingday"]==1]["count"]
Non_work_sampl=discrete_data[discrete_data["workingday"]==0]["count"]
ttest_ind(work_sampl,Non_work_sampl,alternative="greater")
```

```
TtestResult(statistic=1.2096277376026694, pvalue=0.11322402113180674, df=10884.0)
```

```
#Result
#Test statistic= 1.2096277376026694
#P-value= 0.11322402113180674
# Hence fail to reject null Hypothesis:- Hence we can not say that working day has more rented bikes than Non-working day.


#Insight:- Working day and non_working day has same number of bikes on rent.
```

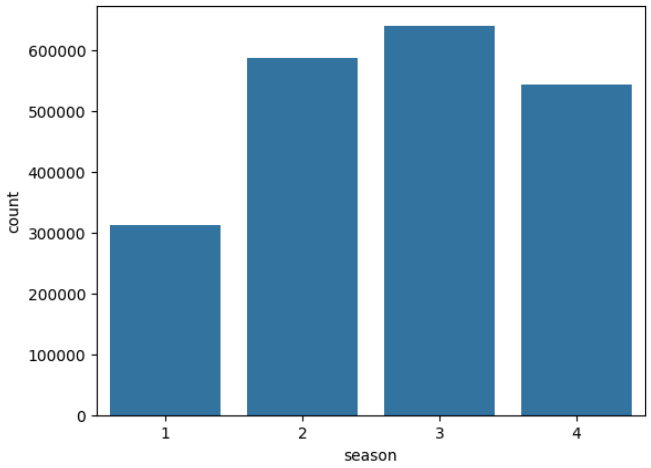
Number of cycles rented similar or different in different seasons

```
seasonal_data=pd.DataFrame(discrete_data.groupby("season")["count"].sum()).reset_index()
seasonal_data.head()
```

	season	count
0	1	312498
1	2	588282
2	3	640662
3	4	544034

Next steps: Generate code with seasonal_data | View recommended plots

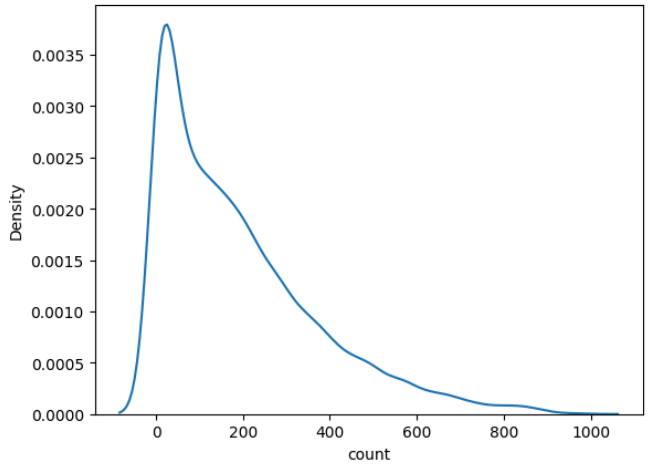
```
sns.barplot(data=seasonal_data,x="season",y="count")
plt.show()
```



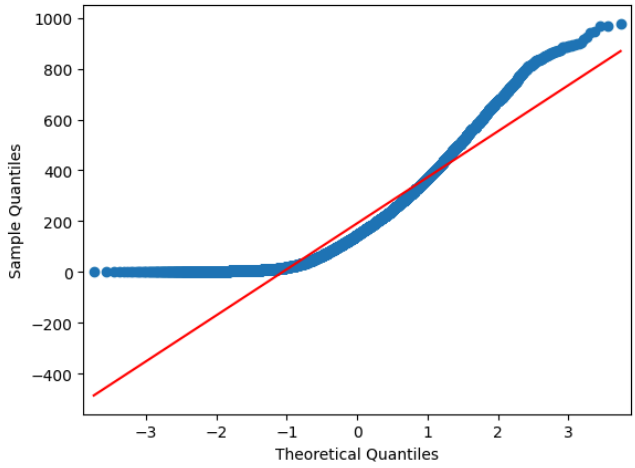

```
# As we can see in above plot there is change in avg rented bike  over different season
pd.DataFrame(discrete_data.groupby("season")["count"].mean()).reset_index()
```

	season	count
0	1	116.343261
1	2	215.251372
2	3	234.417124
3	4	198.988296

```
#Now lets do hypothesis testing with the help of ANOVA test to check the dependecny of season on count and to see if different season
#Hypothesis Test:-If different season have same average number booking or different average number booking.
# Ho: Is different season have same average number of booking (U1=U2=U3=U4)
# Ha: Is different season have different average number of booking
# Significance level=5%


# from scipy.stats in


#Anova test Assumtions
# 1)Normality
# 2)same varaince across all season
# 3)data are independent


# Normality
# Distribution is not looking Gaussian
sns.kdeplot(data=df,x="count")
plt.show()
```

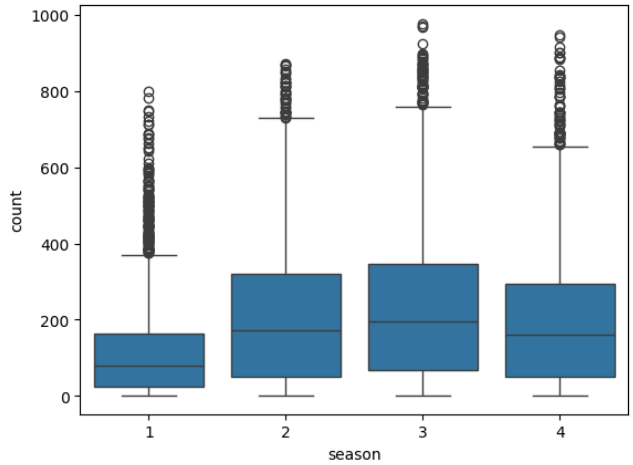



```
#Q-Q plot confirms data is not normal
qqplot(df["count"],line="s")
plt.show()
```




```
#Variance Test of different seasons
```

```
#we can observe the is change in variance with in group
sns.boxplot(data=discrete_data,y="count",x="season")
plt.show()
```




```
#There is drastically change in avg bike rented in season 1 to other season
pd.DataFrame(discrete_data.groupby("season")["count"].var()).reset_index()
```

	season	count
0	1	15693.568534
1	2	36867.011826
2	3	38868.517013
3	4	31549.720317

```
# Ho:Varaince are same in all groups
# Ha:Atleast one of the Varaince are not same
# Significant level= 5%


season1=discrete_data[discrete_data["season"]==1]["count"]
season2=discrete_data[discrete_data["season"]==2]["count"]
season3=discrete_data[discrete_data["season"]==3]["count"]
season4=discrete_data[discrete_data["season"]==4]["count"]


levene(season1,season2,season3,season4)
```

```
LeveneResult(statistic=187.7706624026276, pvalue=1.0147116860043298e-118)
```

```
#Result of Test
# Test statistic=187.7706624026276
# pvalue=1.0147116860043298e-118
# We can see there is differenece in varaince of seasons
# So we can't perform Anova test we have to per Kruskal Test


# To cheack the dependence of Season on count we perform statically test called Kruskal test
#Hypothesis Test :- Is different season have same avg number of  rented bike or different ?
#Test Name:-T-test for independent sample
#Ho=Is different season have same avg number of  rented bike (U1=U2=U3=U4)
#Ha=Is different season have different avg number of  rented bike
#significance level=5%


kruskal(season1,season2,season3,season4)
```

```
KruskalResult(statistic=699.6668548181988, pvalue=2.479008372608633e-151)
```

```
# Result Of Test
# Test statistic=699.6668548181988
# pvalue=2.479008372608633e-151 :-Very Small there is difference is variance in various seasons.


# Insights:
# We can say count of rented variable is dependent or correlated with Seasons
```

No. of cycles rented similar or different in different weather

```
discrete_data.head()
```

	season	holiday	workingday	weather	casual	registered	count
0	1	0	0	1	3	13	16
1	1	0	0	1	8	32	40
2	1	0	0	1	5	27	32
3	1	0	0	1	3	10	13
4	1	0	0	1	0	1	1

Next steps: Generate code with `discrete_data` View recommended plots

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

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

```
#We have very few count on weather 4
weather_data=pd.DataFrame(discrete_data.groupby("weather")["count"].sum()).reset_index()
weather_data.head()
```

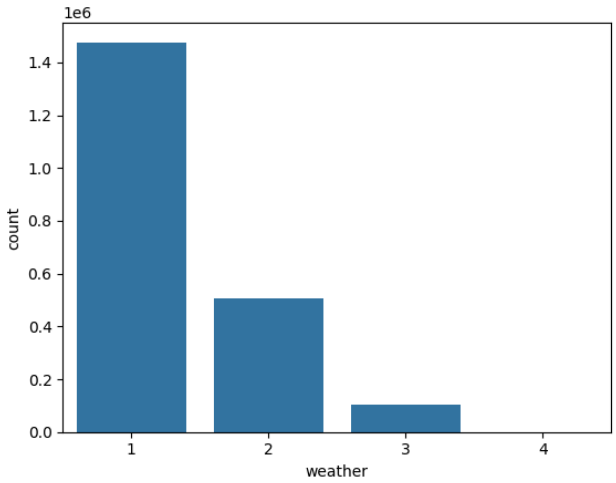
	weather	count
0	1	1476063
1	2	507160
2	3	102089
3	4	164

Next steps: Generate code with `weather_data` View recommended plots

```
#See we have only one data point of weather 4
#So we can do analysys with one data point against good data this will create tomuch variability in analysys
discrete_data[discrete_data["weather"]==4]
```

	season	holiday	workingday	weather	casual	registered	count
5631	1	0	1	4	6	158	164

```
#As we can observe each season have diffent count or rented bikes
sns.barplot(data=weather_data,x="weather",y="count")
plt.show()
```



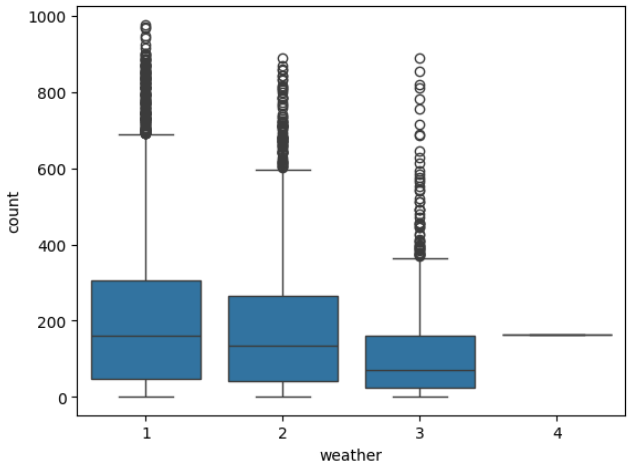

```
#We can see different weather have different avg count of rented bikes
pd.DataFrame(discrete_data.groupby("weather")["count"].mean()).reset_index()
```

	weather	count
0	1	205.236791
1	2	178.955540
2	3	118.846333
3	4	164.000000

```
# To cheack the Corelation of Weather on count we perform statically test called Annova test
#Hypothesis Test :- Is different weathere have same avg no booking or different avg no booking?
#Test Name:-One way Anova
#Ho=Is different weather have same avg no bookinge (U1=U2=U3=U4)
#Ha=Is different weather have different avg no bookinge
#significance level=5%
```

As we know data is not normally distribute we are perform Variability test

```
#Boxplt show there is variance within the different weather
sns.boxplot(data=discrete_data,y="count",x="weather")
plt.show()
```




```
#As there is difference in variance in different weather data
pd.DataFrame(discrete_data.groupby("weather")["count"].var()).reset_index()
```

	weather	count
0	1	35328.798463
1	2	28347.248993
2	3	19204.775893
3	4	NaN

```
weather1=discrete_data[discrete_data["weather"]==1]["count"]
weather2=discrete_data[discrete_data["weather"]==2]["count"]
weather3=discrete_data[discrete_data["weather"]==3]["count"]
weather4=discrete_data[discrete_data["weather"]==4]["count"]
```

```
#We are not considering weather4 data
# Ho:Varaince are same in all groups
# Ha:Atleast one of the Varaince are not same
# Significant level= 5%


levene(weather1,weather2,weather3)
```

```
     LeveneResult(statistic=81.67574924435011, pvalue=6.198278710731511e-36)
```

```
#Result of Test
# Test statistic=81.67574924435011
# pvalue=6.198278710731511e-36
# We can see there is differenece in varaince of weathere
# So we can't perform Anova test we have to per Kruskal Test


# To cheack the dependence of weeather on count we perform statically test called Kruskal test

#Hypothesis Test :- Is different weather have same avg number of  rented bike or different ?
#Test Name:-One-Way Anova Test
#Ho=Is different weather have same avg number of  rented bike (U1=U2=U3=U4)
#Ha=Is different weather have different avg number of  rented bike
#significance level=5%


kruskal(weather1,weather2,weather3)
```

```
     KruskalResult(statistic=204.95566833068537, pvalue=3.122066178659941e-45)
```

```
#Result Of Test
#Test statistic=204.95566833068537
#pvalue=3.122066178659941e-45 :-Very Small there is difference is avg count  of rented bike in various weather.


# Insights:
# We can say count of rented variable is dependent or correlated with weather
```

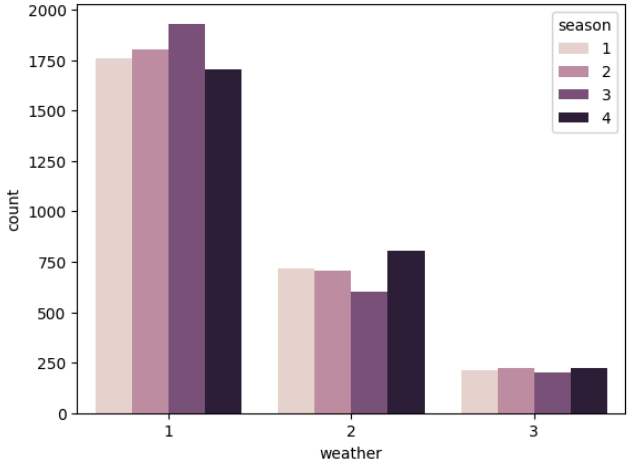
Weather is dependent on season

```
#we are removing weather 4 from here just because one data point make negative impact on analysis
discrete_data.head()
discrete_data=discrete_data[discrete_data["weather"]!=4]
discrete_data.head()
```

	season	holiday	workingday	weather	casual	registered	count
0	1	0	0	1	3	13	16
1	1	0	0	1	8	32	40
2	1	0	0	1	5	27	32
3	1	0	0	1	3	10	13
4	1	0	0	1	0	1	1

Next steps: Generate code with `discrete_data` View recommended plots

```
#As we are are looking we can see there is some change in  in proption season in weather 1 and 2
sns.countplot(data=discrete_data,x="weather",hue="season")
plt.show()
```




```
a=pd.crosstab(discrete_data["season"],discrete_data["weather"],margins=True)
a
```

weather	1	2	3	All
season				
1	1759	715	211	2685
2	1801	708	224	2733
3	1930	604	199	2733
4	1702	807	225	2734
All	7192	2834	859	10885

Next steps: Generate code with a View recommended plots

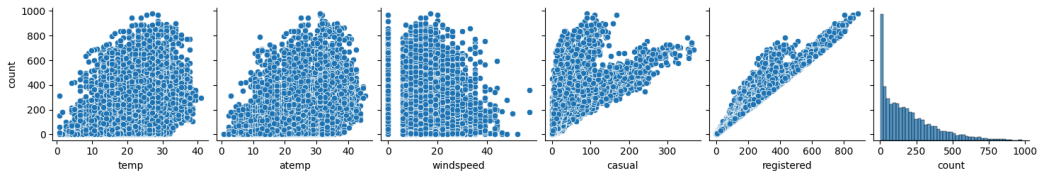
```
# To cheack the corealation of weather ans season we have to perform indendance chisquare test

#Hypothesis Test :-Is weather is correlated with season
#Test Name:-One-Way Anova Test
#Ho=weather are independent of season
#Ha=weather are dependent of season
#significance level=5%


chi2_contingency(a)
```

```
Chi2ContingencyResult(statistic=46.10145731073249, pvalue=6.664576536706683e-06, dof=12, expected_freq=array([[ 1774.04869086,   699.06201194,   211.8892972 ,
2685.        ],
       [ 1805.76352779,   711.55920992,   215.67726229,  2733.        ],
       [ 1805.76352779,   711.55920992,   215.67726229,  2733.        ],
       [ 1806.42425356,   711.81956821,   215.75617823,  2734.        ],
       [ 7192.        ,  2834.        ,   859.        , 10885.        ]]))
```

```
#Result Of Test
#Test statistic=46.10145731073249
#pvalue=6.664576536706683e-06 :- weather are dependent on season


# Insights:
# We can say Weather and Season are closely realted to each other
```

Analysis on Continous Freatures

```
continous_data=df[["temp","atemp","windspeed","casual","registered","count"]]


#As we can see there is not much relationship between  continous feature and count of bike rented
sns.pairplot(continous_data,y_vars=["count"])
plt.show()
```




```
corr=continous_data.corr(method = 'pearson').loc["count":]
corr
```

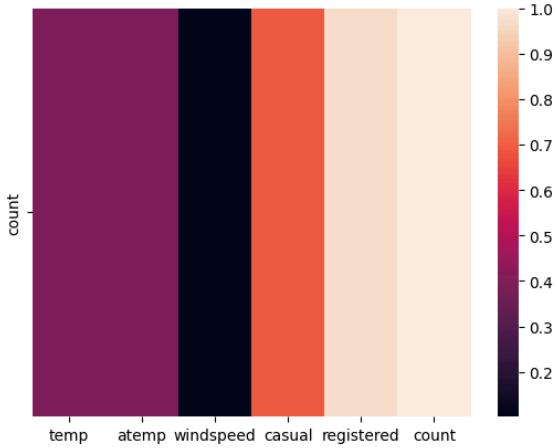
	temp	atemp	windspeed	casual	registered	count
count	0.394454	0.389784	0.101369	0.690414	0.970948	1.0

```
sns.heatmap(corr)
plt.show
```

matplotlib.pyplot.show
def show(*args, **kwargs)

/usr/local/lib/python3.10/dist-packages/matplotlib/pyplot.py
Display all open figures.

Parameters
----
block : bool, optional



.