```
# Importing the required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
from scipy.stats import norm
from scipy.stats import ttest_ind # this test for independent samples
from scipy.stats import shapiro # Shapiro - Wilk's test for normality
from scipy.stats import levene # Levene's test for checking the variances to be same
from scipy.stats import f_oneway # One way anova test
from scipy.stats import chi2_contingency  #chisquare test of independence
from scipy.stats import kstest# kstest for the non-parametric test
```

```
# loading the data
! gdown '1j66CvhrpM7AelV5ypRqXf5eXE7SHrRwg'
df = pd.read_csv('/content/bike_sharing.txt')
df.head()
```

```
Downloading...
From: https://drive.google.com/uc?id=1j66CvhrpM7AelV5ypRqXf5eXE7SHrRwg
To: /content/bike_sharing.txt
100% 648k/648k [00:00<00:00, 106MB/s]
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0

Insights :- There are 10886 number of rows and 12 columns.

```
# Checking the number of rows and columns
df.shape
```

```
(10886, 12)
```

```
# checking the quantiles of the data
df.describe()
```

	season	holiday	workingday	weather	temp	atem
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.00000
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.65508
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.47460
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.76000
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.66500
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.24000
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.06000
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.45500

Insights :- There are no null values in the data.

```
# checking the null values in the data
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
```

Insights:- No duplicate data is present in the dataset.

```
#Checking the duplicate data in the dataset.
df[df.duplicated()]
```

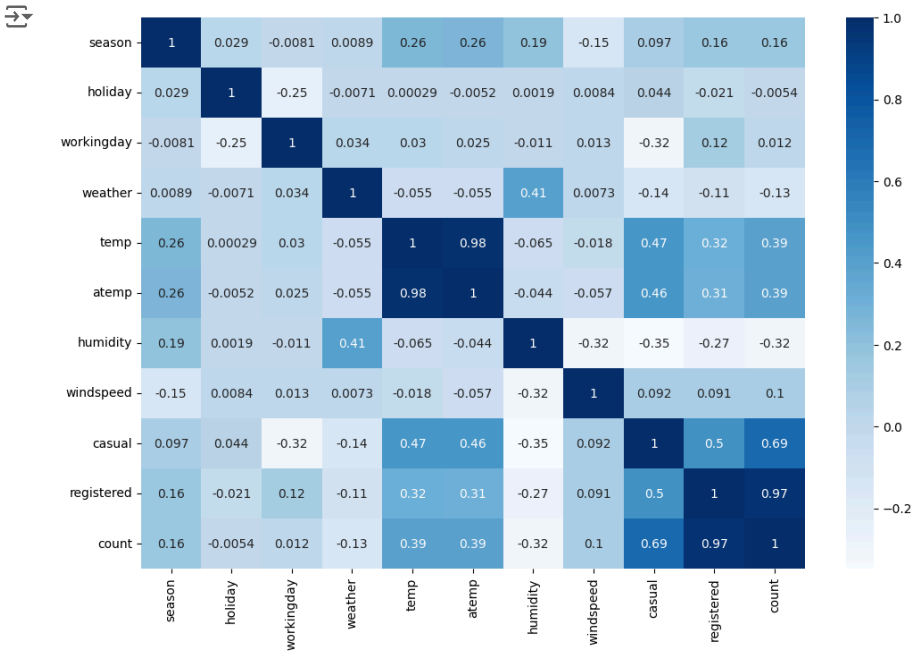
	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	ca

```
# writing a function to check the number of unqiue values in the data for categorical columns.
def dist_check(df,col_name):
  print('Unique values:',df[col_name].unique())
  print('Value counts: ')
  print(df[col_name].value_counts())
```

```
col_list = ['workingday','holiday','weather','season']

for col in col_list:
  print(col,':-')
  dist_check(df,col)
  print('\n')
```

```
workingday :-
Unique values: [0 1]
Value counts:
workingday
1    7412
0    3474
Name: count, dtype: int64


holiday :-
Unique values: [0 1]
Value counts:
holiday
0    10575
1      311
Name: count, dtype: int64


weather :-
Unique values: [1 2 3 4]
Value counts:
weather
1    7192
2    2834
3     859
4       1
Name: count, dtype: int64


season :-
Unique values: [1 2 3 4]
Value counts:
season
4    2734
2    2733
3    2733
1    2686
Name: count, dtype: int64
```

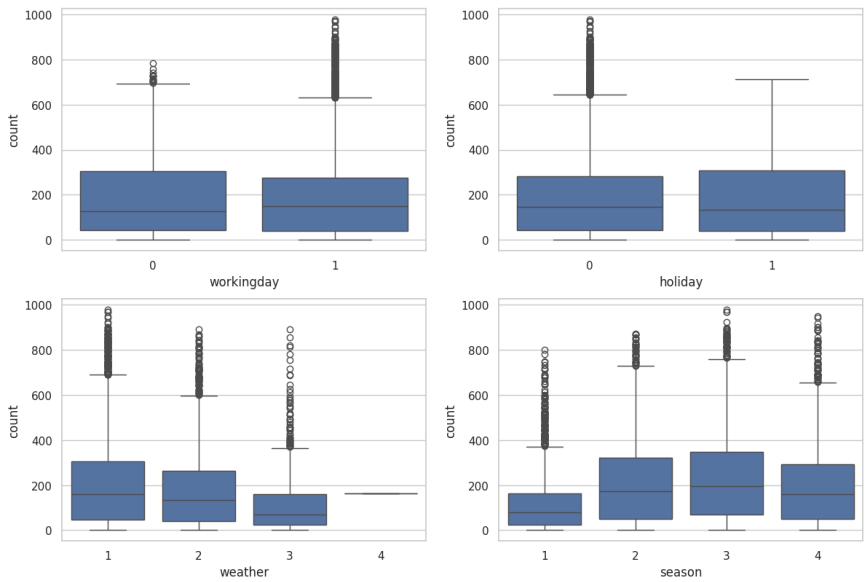
```
# Plotting a heatmap for all the columns with dtypes = int64 and float64
plt.figure(figsize=(12,8))
sns.heatmap(df[df.select_dtypes(include=['int64','float64']).columns].corr(),annot=True,cmap='Blues')
plt.show()
```




```
# Droping the unwanted rows from the dataset
df = df.drop(columns =['casual','registered','atemp'])
```

Insights :- There are outliers in all the 4 plotted columns :- weather,workingday,holiday,season.

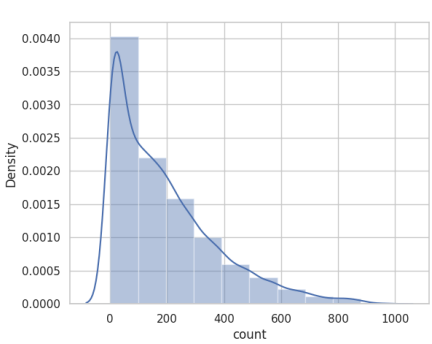
```
# Outliers Detection using Boxplots :-
sns.set(style = 'whitegrid')
fig = plt.figure(figsize =(8,25))
fig.subplots_adjust(right=1.5)

for plot in range(1,len(col_list)+1):
  plt.subplot(5,2,plot)
  sns.boxplot(x = df[col_list[plot-1]], y = df['count'])

plt.show()
```

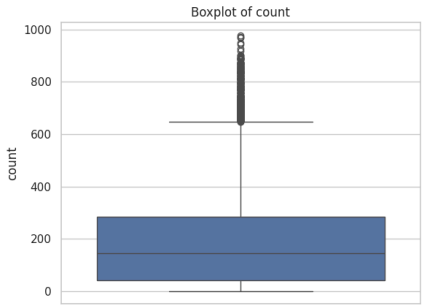



```
# outlier detection method - Local Outlier Factor detection method.

# Checking distribution of 'count' column -
plt.figure(figsize=(14,5))

# Histogram
plt.subplot(1,2,1)
sns.distplot(df['count'],bins = 10)

# Boxplot
plt.subplot(1,2,2)
sns.boxplot(y = df['count'])
plt.title('Boxplot of count')

plt.show()
```

```
<ipython-input-26-851a44d7f103>:8: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

```
sns.distplot(df['count'],bins = 10)
```





Aggregating the total number of bike rides based on given factors.

```
# 1. workingday -
pd.DataFrame(df.groupby('workingday')['count'].describe())
```

workingday	count	mean	std	min	25%	50%	75%	max
0	3474.0	188.506621	173.724015	1.0	44.0	128.0	304.0	783.0
1	7412.0	193.011873	184.513659	1.0	41.0	151.0	277.0	977.0

```
# 2. Holiday
pd.DataFrame(df.groupby('holiday')['count'].describe())
```

holiday	count	mean	std	min	25%	50%	75%	max
0	10575.0	191.741655	181.513131	1.0	43.0	145.0	283.0	977.0
1	311.0	185.877814	168.300531	1.0	38.5	133.0	308.0	712.0

```
# 3. Season
pd.DataFrame(df.groupby('season')['count'].describe())
```

season	count	mean	std	min	25%	50%	75%	max
1	2686.0	116.343261	125.273974	1.0	24.0	78.0	164.0	801.0
2	2733.0	215.251372	192.007843	1.0	49.0	172.0	321.0	873.0
3	2733.0	234.417124	197.151001	1.0	68.0	195.0	347.0	977.0
4	2734.0	198.988296	177.622409	1.0	51.0	161.0	294.0	948.0

```
# 4. Weather
pd.DataFrame(df.groupby('weather')['count'].describe())
```

weather	count	mean	std	min	25%	50%	75%	max
1	7192.0	205.236791	187.959566	1.0	48.0	161.0	305.0	977.0
2	2834.0	178.955540	168.366413	1.0	41.0	134.0	264.0	890.0
3	859.0	118.846333	138.581297	1.0	23.0	71.0	161.0	891.0
4	1.0	164.000000	NaN	164.0	164.0	164.0	164.0	164.0

Hypothesis Testing

Weekdays vs weekends

```
weekday = df[df['workingday'] == 1]['count'].sample(2999)
weekend = df[df['workingday'] == 0]['count'].sample(2999)

print('The sample standard deviation of the bike rides on weekdays is:',round(weekday.std(),2))
print('The sample standard deviation of the bike rides on weekends is:',round(weekend.std(),2))
```

```
The sample standard deviation of the bike rides on weekdays is: 183.99
The sample standard deviation of the bike rides on weekends is: 173.73
```

Significance level = 0.05

```
alpha = 0.05

def result(p_value,alpha):
  if p_value < alpha:
    print('Reject the null hypothesis')
  else:
    print('Fail to reject the null hypothesis')
```

```
test_stat, p_value = ttest_ind(weekday,weekend,alternative='less')
print('The p-value is:',p_value)
result(p_value,alpha)
```

```
The p-value is: 0.8616892552405073
Fail to reject the null hypothesis
```

Holiday vs Regular

```
holiday = df[df['holiday'] == 1]['count'].sample(299)
regular = df[df['holiday'] == 0]['count'].sample(299)

print('The sample standard deviation of the bike rides on holidays is:',round(holiday.std(),2))
print('The sample standard deviation of the bike rides on regular is:',round(regular.std(),2))
```

```
The sample standard deviation of the bike rides on holidays is: 168.88
The sample standard deviation of the bike rides on regular is: 198.39
```

```
test_stat, p_value = ttest_ind(regular,holiday,alternative='less')
print('The p-value is:',p_value)
result(p_value,alpha)
```

```
The p-value is: 0.8581946645160745
Fail to reject the null hypothesis
```

Weather dependency

```
df = df[~(df['weather'] == 4)]

w1 = df[df['weather'] == 1]['count'].sample(750)
w2 = df[df['weather'] == 2]['count'].sample(750)
w3 = df[df['weather'] == 3]['count'].sample(750)
```

```
test_stat, p_value = f_oneway(w1,w2,w3)
print('The p-value is:',p_value)
result(p_value,alpha)
```

```
The p-value is: 1.864951902150935e-26
Reject the null hypothesis
```

Season's dependancy

```
s1 = df[df['season'] == 1]['count'].sample(2600)
s2 = df[df['season'] == 2]['count'].sample(2600)
s3 = df[df['season'] == 3]['count'].sample(2600)
s4 = df[df['season'] == 4]['count'].sample(2600)
```

```
test_stat, p_value = f_oneway(w1,w2,w3)
print('The p-value is:',p_value)
result(p_value,alpha)
```

```
The p-value is: 1.864951902150935e-26
Reject the null hypothesis
```

Insights from testing :-

1. The no. of bikes rented on weekdays is comparatively higher than on weekends.
2. The no. of bikes rented on regular is comparatively higher than on holidays.
3. The demand of the bicycle on rent differs under the weather conditions.

Recommendations :-

1. The demand of bikes on rent are usaually higher during Weekdays.
2. The demand of bikes on rent are usaually higher during Regular days.
3. The chances of person renting a bike are usually higher during season 3.
4. The chances of person renting a bike are usually higher during weather 1.

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