💻 Yulu Hypothesis Testing- Business Case Study

```
!wget "https://drive.google.com/uc?export=download&id=1EwC2r1pVVjWMPknzalYcPZuekpJ_DmOJ" -O yulu_data.csv
```

```
--2025-01-27 10:19:56--  https://drive.google.com/uc?export=download&id=1EwC2r1pVVjWMPknzalYcPZuekpJ_DmOJ
Resolving drive.google.com (drive.google.com)... 74.125.132.100, 74.125.132.102, 74.125.132.139, ...
Connecting to drive.google.com (drive.google.com)|74.125.132.100|:443... connected.
HTTP request sent, awaiting response... 303 See Other
Location: https://drive.usercontent.google.com/download?id=1EwC2r1pVVjWMPknzalYcPZuekpJ_DmOJ&export=download [following]
--2025-01-27 10:19:56--  https://drive.usercontent.google.com/download?id=1EwC2r1pVVjWMPknzalYcPZuekpJ_DmOJ&export=download
Resolving drive.usercontent.google.com (drive.usercontent.google.com)... 209.85.200.132, 2607:f8b0:4001:c16::84
Connecting to drive.usercontent.google.com (drive.usercontent.google.com)|209.85.200.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 648353 (633K) [application/octet-stream]
Saving to: 'yulu_data.csv'

yulu_data.csv       100%[===================>] 633.16K  --.-KB/s    in 0.02s

2025-01-27 10:19:58 (38.3 MB/s) - 'yulu_data.csv' saved [648353/648353]
```

1. Introduction

❓ What is Yulu Business Case Study?

- Yulu, India's pioneering micro-mobility service provider, has embarked on a mission to revolutionize daily commutes by offering unique, sustainable transportation solutions. However, recent revenue setbacks have prompted Yulu to seek the expertise of a consulting company to delve into the factors influencing the demand for their shared electric cycle specifically in the Indian market.

🎯 Objective:

- Strategic Expansion: Yulu's decision to enter the Indian market is a strategic move to expand its global footprint. Understanding the demand factors in this new market is essential to tailor their services and strategies accordingly.
- Revenue Recovery: Yulu's recent revenue decline is a pressing concern. By analyzing the factors affecting demand for shared electric cycles in the Indian market, they can make informed adjustments to regain profitability.

📚 About Data:

📃 Features of the dataset:

- datetime: datetime
- season: season (1: spring, 2: summer, 3: fall, 4: winter)
- holiday : whether day is a holiday or not
- workingday : if day is neither weekend nor holiday is 1, otherwise is 0.
- weather:

1: Clear, Few clouds, 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 Pellets + 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 registere

2.Exploratory Data Analysis

```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
import warnings
warnings.filterwarnings('ignore')
from scipy import stats
from scipy.stats import norm
```

```
from scipy.stats import ttest_ind
from scipy.stats import f_oneway
from scipy.stats import shapiro
from scipy.stats import levene
from scipy.stats import kruskal
from scipy.stats import chi2_contingency
```

```
# loading the dataset
df = pd.read_csv('yulu_data.csv')
```

```
#to view full data
df
```

	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	0.0000	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0000	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0000	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0000	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0000	0	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...
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

10886 rows × 12 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
#to view columns

df.columns
#df.keys()== df.columns
```

```
Index(['datetime', 'season', 'holiday', 'workingday', 'weather', 'temp',
       'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count'],
      dtype='object')
```

```
#view first 5 rows/records
df.head(5)
#view first 5 rows/records default=5
#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

```
#view last 5 rows/records,deafault=5
df.tail()
#view last 5 rows/records
#df.tail(5)
```

	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

```
#To get index of dataframe
df.index
```

```
RangeIndex(start=0, stop=10886, step=1)
```

```
#To get shape information

df.shape

#10886 rows and 12 columns
```

```
(10886, 12)
```

```
# to get dimensional detail of dataframe

df.ndim

#2D
```

```
2
```

```
#Datatype

print(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
```

```
# Convert columns to categorical types and datetime to Datetime
df['season'] = df['season'].astype('category')
df['holiday'] = df['holiday'].astype('category')
df['workingday'] = df['workingday'].astype('category')
df['weather'] = df['weather'].astype('category')
df['datetime']=pd.to_datetime(df['datetime'])
```

```
df.dtypes
```

	0
datetime	datetime64[ns]
season	category
holiday	category
workingday	category
weather	category
temp	float64
atemp	float64
humidity	int64
windspeed	float64
casual	int64
registered	int64
count	int64

dtype: object

```
# to get complete information of each column of dataframe like counts,datatype,memory usage.
#Note: For missing value in each column data type will be object

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  datetime64[ns]
 1   season      10886 non-null  category
 2   holiday     10886 non-null  category
 3   workingday  10886 non-null  category
 4   weather     10886 non-null  category
 5   temp        10886 non-null  float64
 6   atemp       10886 non-null  float64
 7   humidity    10886 non-null  int64
 8   windspeed   10886 non-null  float64
 9   casual      10886 non-null  int64
 10  registered  10886 non-null  int64
 11  count       10886 non-null  int64
dtypes: category(4), datetime64[ns](1), float64(3), int64(4)
memory usage: 723.7 KB
```

🔍 Insights

From the above details it is clear that given dataframe is of dimension 2D with 10886 rows and 12 columns.

Also we can also observe that there are no missing values for any columns .

📝 Statistical Summary

```
#for column with datatype as int, df.describe() will give statistical information like count,mean,min,max,std detail for that column.
```

```
df.describe()
```

	datetime	temp	atemp	humidity	windspeed	casual	registered	count
count	10886	10886.00000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000
mean	2011-12-27 05:56:22.399411968	20.23086	23.655084	61.886460	12.799395	36.021955	155.552177	191.574132
min	2011-01-01 00:00:00	0.82000	0.760000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	2011-07-02 07:15:00	13.94000	16.665000	47.000000	7.001500	4.000000	36.000000	42.000000
50%	2012-01-01 20:30:00	20.50000	24.240000	62.000000	12.998000	17.000000	118.000000	145.000000
75%	2012-07-01 12:45:00	26.24000	31.060000	77.000000	16.997900	49.000000	222.000000	284.000000
max	2012-12-19 23:00:00	41.00000	45.455000	100.000000	56.996900	367.000000	886.000000	977.000000
std	NaN	7.79159	8.474601	19.245033	8.164537	49.960477	151.039033	181.144454

```
#Statistical Summary: Generate a statistical summary of the dataset.
```

```
df.describe(include='all')
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual
count	10886	10886.0	10886.0	10886.0	10886.0	10886.00000	10886.000000	10886.000000	10886.000000	10886.000000
unique	NaN	4.0	2.0	2.0	4.0	NaN	NaN	NaN	NaN	NaN
top	NaN	4.0	0.0	1.0	1.0	NaN	NaN	NaN	NaN	NaN
freq	NaN	2734.0	10575.0	7412.0	7192.0	NaN	NaN	NaN	NaN	NaN
mean	2011-12-27 05:56:22.399411968	NaN	NaN	NaN	NaN	20.23086	23.655084	61.886460	12.799395	36.021955
min	2011-01-01 00:00:00	NaN	NaN	NaN	NaN	0.82000	0.760000	0.000000	0.000000	0.000000
25%	2011-07-02 07:15:00	NaN	NaN	NaN	NaN	13.94000	16.665000	47.000000	7.001500	4.000000
50%	2012-01-01 20:30:00	NaN	NaN	NaN	NaN	20.50000	24.240000	62.000000	12.998000	17.000000
75%	2012-07-01 12:45:00	NaN	NaN	NaN	NaN	26.24000	31.060000	77.000000	16.997900	49.000000
max	2012-12-19 23:00:00	NaN	NaN	NaN	NaN	41.00000	45.455000	100.000000	56.996900	367.000000
std	NaN	NaN	NaN	NaN	NaN	7.79159	8.474601	19.245033	8.164537	49.960477

🔍 Insights:

1) Datetime:

- Count: 10,886 entries
- Mean: 2011-12-27 05:56:22.399411968
- Min: 2011-01-01 00:00:00
- 25%: 2011-07-02 07:15:00
- 50%: 2012-01-01 20:30:00
- 75%: 2012-07-01 12:45:00
- Max: 2012-12-19 23:00:00

2) Season:

- Count: 10,886
- Unique: 4 (spring, summer, fall, winter)
- Top: 4 (winter)
- Frequency: 2,734

3) Holiday:

- Count: 10,886
- Unique: 2 (0: not a holiday, 1: holiday)
- Top: 0 (not a holiday)
- Frequency: 10,575

4) Working Day:

- Count: 10,886
- Unique: 2 (0: not a working day, 1: working day)
- Top: 1 (working day)
- Frequency: 7,412

5) Weather:

- Count: 10,886
- Unique: 4 (1: Clear, Few clouds, 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 Pellets + Thunderstorm + Mist, Snow + Fog)
- Top: 1 (Clear, Few clouds, partly cloudy)
- Frequency: 7,192

6) Temperature (temp):

- Count: 10,886
- Mean: 20.23°C
- Min: 0.82°C
- 025%: 13.94°C
- 50%: 20.50°C
- 75%: 26.24°C
- Max: 41.00°C
- Std: 7.79°C

7) Feels Like Temperature (atemp): -Count: 10,886

- Mean: 23.66°C
- Min: 0.76°C
- 25%: 16.67°C
- 50%: 24.24°C
- 75%: 31.06°C
- Max: 45.46°C
- Std: 8.47°C

8) Humidity:

- Count: 10,886
- Mean: 61.89%
- Min: 0%

- 25%: 47%
- 50%: 62%
- 75%: 77%
- Max: 100%
- Std: 19.25%

9) Windspeed:

- Count: 10,886
- Mean: 12.80 km/h
- Min: 0 km/h
- 25%: 7.00 km/h
- 50%: 13.00 km/h
- 75%: 17.00 km/h
- Max: 57.00 km/h
- Std: 8.16 km/h

10) Casual Users:

- Count: 10,886
- Mean: 36.02
- Min: 0
- 25%: 4
- 50%: 17
- 75%: 49
- Max: 367
- Std: 49.96

11) Registered Users:

- Count: 10,886
- Mean: 155.55
- Min: 0
- 25%: 36
- 50%: 118
- 75%: 222
- Max: 886
- Std: 151.04

12) Total Count of Bike Rentals:

- Count: 10,886
- Mean: 191.57
- Min: 1
- 25%: 42
- 50%: 145
- 75%: 284
- Max: 977
- Std: 181.14

👥 Duplicate Detection

```
df.duplicated().value_counts()
```

	count
False	10886

dtype: int64

🔍 Insights There are no duplicate entries in the dataset

🧑‍💻 Missing Value Analysis

```
missing_values=df.isnull().sum()
missing_values
```

	0
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

🔍 Insights There is no missing values for all columns.

📍 For Non-graphical Analysis:

✅ Sanity Check for columns

```
# checking the unique values for columns
for i in df.columns:
  print('Unique Values in',i,'column are :-')
  print(df[i].unique())
  print('-'*70)
```

```
Unique Values in datetime column are :-
<DatetimeArray>
['2011-01-01 00:00:00', '2011-01-01 01:00:00', '2011-01-01 02:00:00',
 '2011-01-01 03:00:00', '2011-01-01 04:00:00', '2011-01-01 05:00:00',
 '2011-01-01 06:00:00', '2011-01-01 07:00:00', '2011-01-01 08:00:00',
 '2011-01-01 09:00:00',
 ...
 '2012-12-19 14:00:00', '2012-12-19 15:00:00', '2012-12-19 16:00:00',
 '2012-12-19 17:00:00', '2012-12-19 18:00:00', '2012-12-19 19:00:00',
 '2012-12-19 20:00:00', '2012-12-19 21:00:00', '2012-12-19 22:00:00',
 '2012-12-19 23:00:00']
Length: 10886, dtype: datetime64[ns]
----------------------------------------------------------------------
Unique Values in season column are :-
[1, 2, 3, 4]
Categories (4, int64): [1, 2, 3, 4]
----------------------------------------------------------------------
Unique Values in holiday column are :-
[0, 1]
Categories (2, int64): [0, 1]
----------------------------------------------------------------------
Unique Values in workingday column are :-
[0, 1]
Categories (2, int64): [0, 1]
----------------------------------------------------------------------
Unique Values in weather column are :-
[1, 2, 3, 4]
Categories (4, int64): [1, 2, 3, 4]
----------------------------------------------------------------------
Unique Values in temp column are :-
[ 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.  ]
----------------------------------------------------------------------
Unique Values in atemp column are :-
[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]
----------------------------------------------------------------------
Unique Values in humidity column are :-
[ 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]
----------------------------------------------------------------------
Unique Values in windspeed column are :-
[ 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]

----------------------------------------------------------------------
```

```
 749 623 713 746 651 686 690 679 685 648 560 503 521 554 541 721 801 561
 573 589 729 618 494 757 800 684 744 759 822 698 490 536 655 643 626 615
 567 617 632 646 692 704 624 656 610 738 671 678 660 658 635 681 616 522
 673 781 775 576 677 748 776 557 743 666 813 504 627 706 641 575 639 769
 680 546 717 710 458 622 705 630 732 770 439 779 659 602 478 733 650 873
 846 474 634 852 868 745 812 669 642 730 672 645 694 493 668 647 702 665
 834 850 790 415 724 869 700 793 723 534 831 613 653 857 719 867 823 403
 693 603 583 542 614 580 811 795 747 581 722 689 849 872 631 649 819 674
 830 814 633 825 629 835 667 755 794 661 772 657 771 777 837 891 652 739
 865 767 741 469 605 858 843 640 737 862 810 577 818 854 682 851 848 897
 832 791 654 856 839 725 863 808 792 696 701 871 968 750 970 877 925 977
 758 884 766 894 715 783 683 842 774 797 886 892 784 687 809 917 901 887
 785 900 761 806 507 948 844 798 827 670 637 619 592 943 838 817 888 890
 788 588 606 608 691 711 663 731 708 609 688 636]
----------------------------------------------------------------------
```

```
# to understand the diversity of data in each specified column.
for i in df.columns:
    print('Unique Values in',i,'column are :-')
    print(df[i].nunique())
    print('-'*70)
```

```
Unique Values in datetime column are :-
10886
----------------------------------------------------------------------
Unique Values in season column are :-
4
----------------------------------------------------------------------
Unique Values in holiday column are :-
2
----------------------------------------------------------------------
Unique Values in workingday column are :-
2
----------------------------------------------------------------------
Unique Values in weather column are :-
4
----------------------------------------------------------------------
Unique Values in temp column are :-
49
----------------------------------------------------------------------
Unique Values in atemp column are :-
60
----------------------------------------------------------------------
Unique Values in humidity column are :-
89
----------------------------------------------------------------------
Unique Values in windspeed column are :-
28
----------------------------------------------------------------------
Unique Values in casual column are :-
309
----------------------------------------------------------------------
Unique Values in registered column are :-
731
----------------------------------------------------------------------
Unique Values in count column are :-
822
----------------------------------------------------------------------
```

```
for i in df.columns:
  print('Value count in',i,'column are :-')
  print(df[i].value_counts())
  print('-'*70)
```

```
Value count in datetime column are :-
datetime
2011-01-01 00:00:00    1
2012-05-01 21:00:00    1
2012-05-01 13:00:00    1
2012-05-01 14:00:00    1
2012-05-01 15:00:00    1
                      ..
2011-09-02 04:00:00    1
2011-09-02 05:00:00    1
2011-09-02 06:00:00    1
2011-09-02 07:00:00    1
2012-12-19 23:00:00    1
Name: count, Length: 10886, dtype: int64
----------------------------------------------------------------------
Value count in season column are :-
season
4    2734
2    2733
3    2733
1    2686
Name: count, dtype: int64
----------------------------------------------------------------------
Value count in holiday column are :-
holiday
0    10575
1      311
Name: count, dtype: int64
----------------------------------------------------------------------
Value count in workingday column are :-
workingday
1    7412
0    3474
Name: count, dtype: int64
----------------------------------------------------------------------
```

```
304      1
Name: count, Length: 309, dtype: int64
----------------------------------------------------------------------
Value count in registered column are :-
registered
3      195
4      190
5      177
6      155
2      150
      ...
570      1
422      1
678      1
565      1
636      1
Name: count, Length: 731, dtype: int64
----------------------------------------------------------------------
Value count in count column are :-
count
5      169
4      149
3      144
6      135
2      132
      ...
801      1
629      1
825      1
589      1
636      1
Name: count, Length: 822, dtype: int64
----------------------------------------------------------------------
```

🔍 Insights:

1) Datetime: 10,886 unique values

2) Season: 4 unique values

3) Holiday: 2 unique values

4) Working Day: 2 unique values

5) Weather: 4 unique values

6) Temperature (temp): 49 unique values

7) Feels Like Temperature (atemp): 60 unique values

8) Humidity: 89 unique values

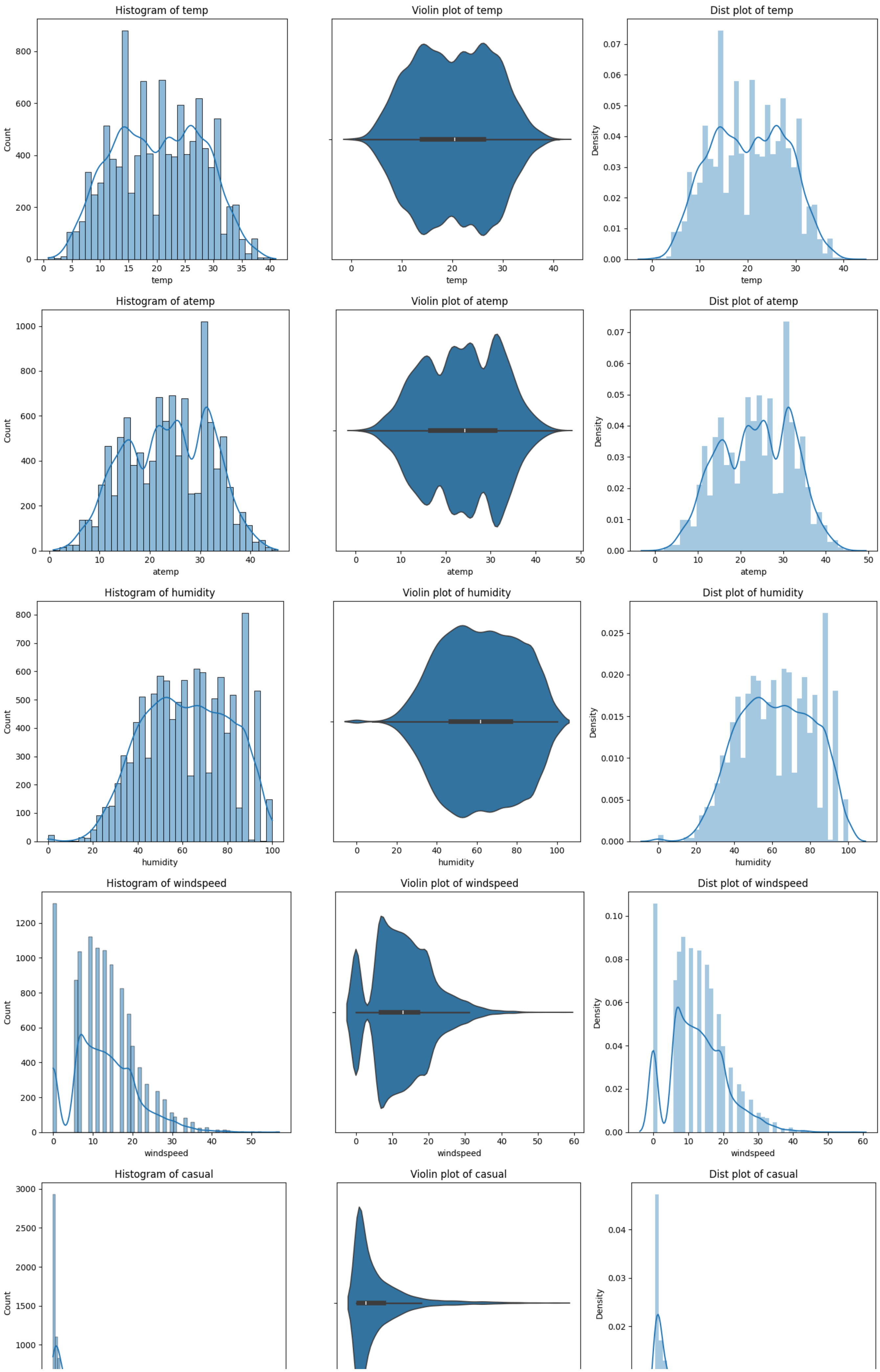
9) Windspeed: 28 unique values

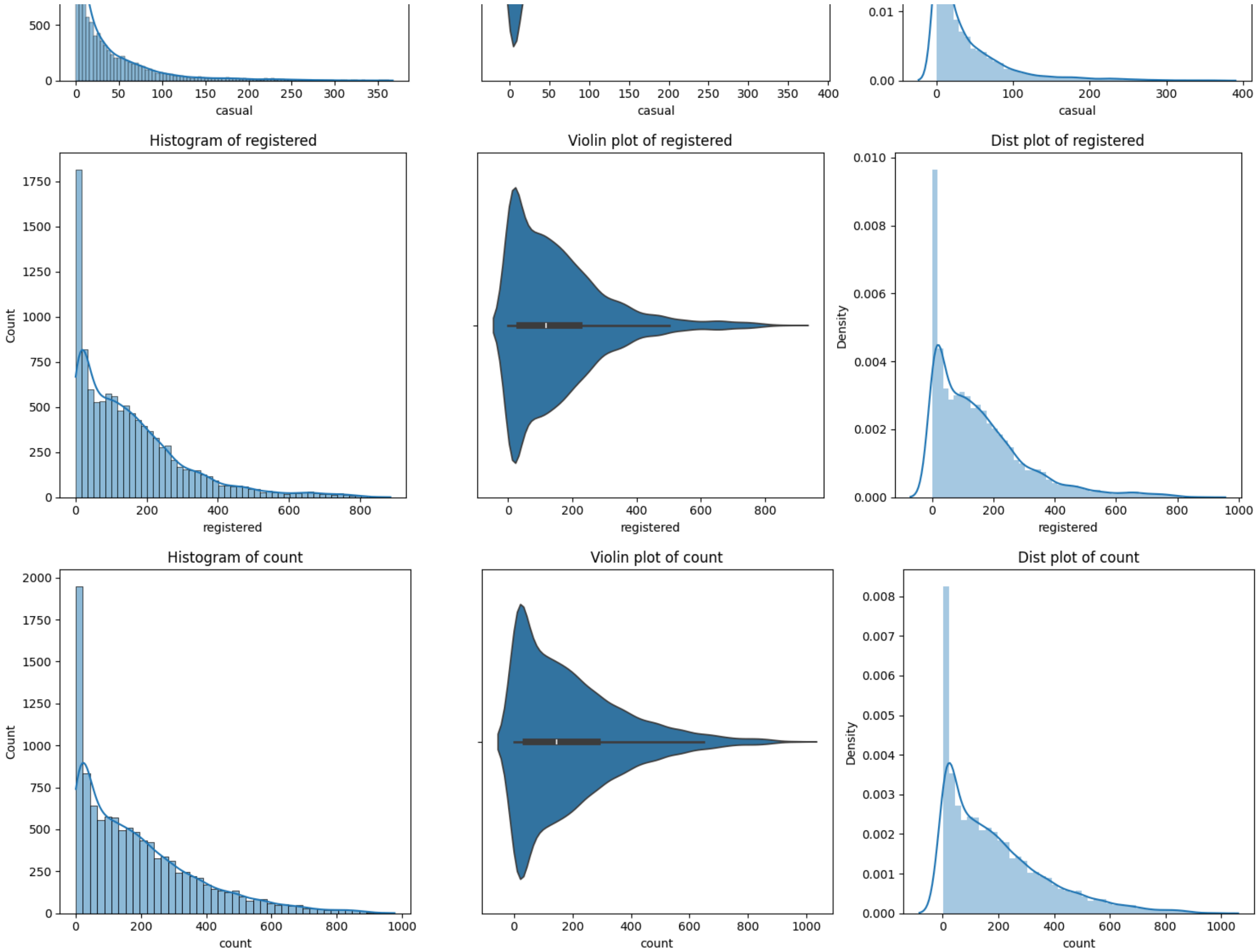
10) Casual Users: 309 unique values

11) Registered Users: 731 unique values

12) Total Count of Bike Rentals: 822 unique values

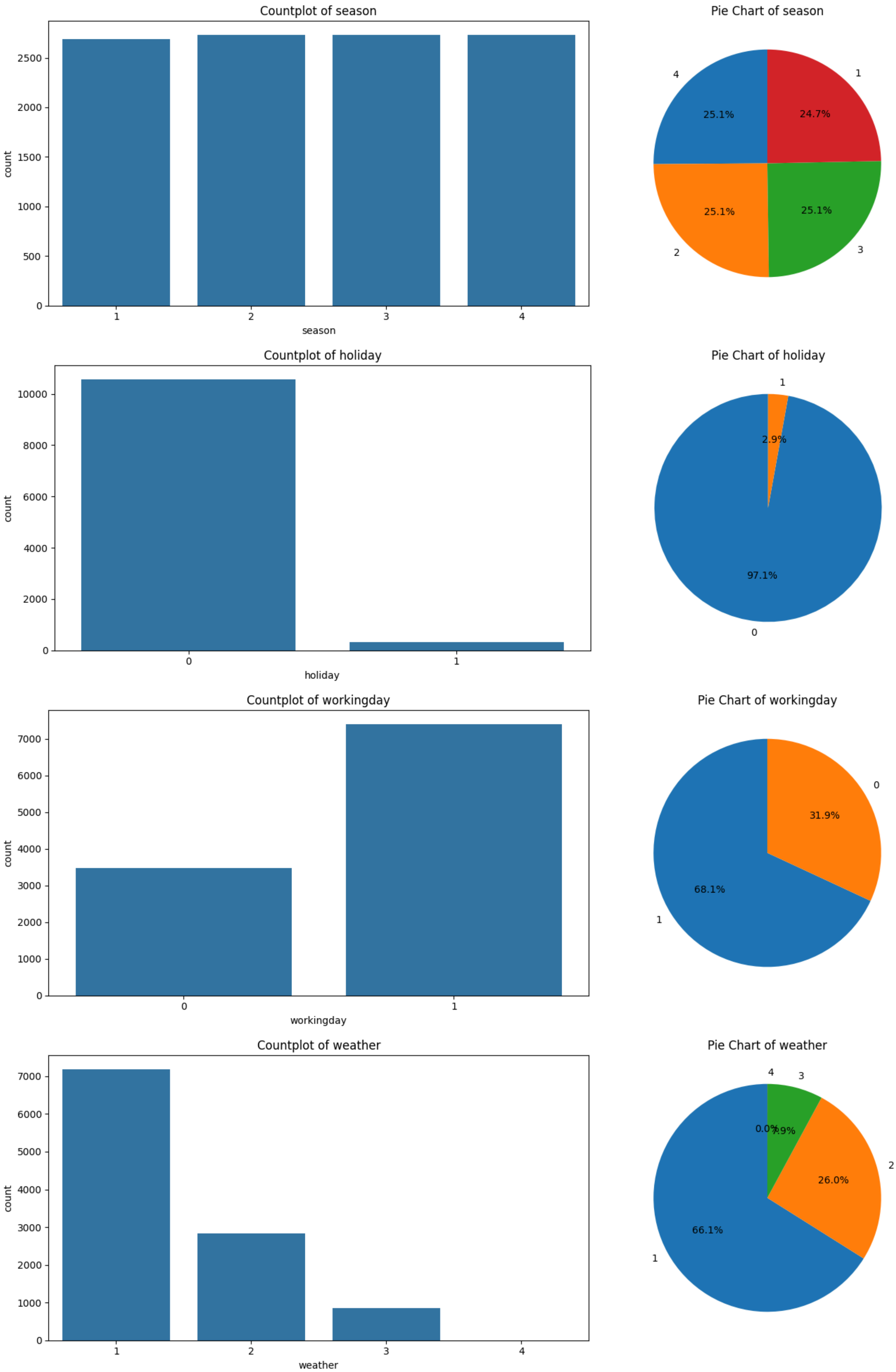
```python
# List of continuous columns
continuous_columns = ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count']

# Visual Analysis - For continuous variable(s): Distplot, violinplot, histogram for univariate analysis
for column in continuous_columns:
    plt.figure(figsize=(15, 5))

    # Histogram
    plt.subplot(1, 3, 1)
    sns.histplot(df[column].dropna(), kde=True)
    plt.title(f'Histogram of {column}')
    plt.tight_layout()

    # Violin plot
    plt.subplot(1, 3, 2)
    sns.violinplot(x=df[column].dropna())
    plt.title(f'Violin plot of {column}')
    plt.tight_layout()

    # Dist plot
    plt.subplot(1, 3, 3)
    sns.distplot(df[column].dropna())
    plt.title(f'Dist plot of {column}')
    plt.tight_layout()

    plt.show()
```


Histogram of temp
Violin plot of temp
Dist plot of temp
Histogram of atemp
Violin plot of atemp
Dist plot of atemp
Histogram of humidity
Violin plot of humidity
Dist plot of humidity
Histogram of windspeed
Violin plot of windspeed
Dist plot of windspeed
Histogram of casual
Violin plot of casual
Dist plot of casual
Count
Density
temp
atemp
humidity
windspeed



casual
casual
casual
Histogram of registered
Violin plot of registered
Dist plot of registered
Count
Density
registered
registered
registered
Histogram of count
Violin plot of count
Dist plot of count
Count
Density
count
count
count

```
# List of categorical columns
categorical_columns = ['season', 'holiday', 'workingday', 'weather']

# Function to plot countplot and pie chart for categorical columns
def plot_categorical_feature(column):
    plt.figure(figsize=(15, 5))

    # Countplot
    plt.subplot(1, 2, 1)
    sns.countplot(x=column, data=df)
    plt.title(f'Countplot of {column}')
    plt.tight_layout()

    # Pie Chart
    plt.subplot(1, 2, 2)
    df[column].value_counts().plot.pie(autopct='%1.1f%%', startangle=90)
    plt.title(f'Pie Chart of {column}')
    plt.ylabel('')
    plt.tight_layout()

    plt.show()

# Plotting for categorical columns
for column in categorical_columns:
    plot_categorical_feature(column)
```


Countplot of season
2500
2000
1500
1000
500
0
count
1
2
3
4
season
Pie Chart of season
4
1
25.1%
24.7%
25.1%
25.1%
2
3
Countplot of holiday
10000
8000
6000
4000
2000
0
count
0
1
holiday
Pie Chart of holiday
1
2.9%
97.1%
0
Countplot of workingday
7000
6000
5000
4000
3000
2000
1000
0
count
0
1
workingday
Pie Chart of workingday
0
31.9%
68.1%
1
Countplot of weather
7000
6000
5000
4000
3000
2000
1000
0
count
1
2
3
4
weather
Pie Chart of weather
4
3
0.0%
7.9%
2
26.0%
66.1%
1

```
#created a copy of your DataFrame df and assigned it to df_new; any changes made to df_new wont affetct to df.
df_new=df.copy()
df_new
```

	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	0.0000	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0000	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0000	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0000	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0000	0	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...
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

10886 rows × 12 columns

Next steps: Generate code with df_new | View recommended plots | New interactive sheet

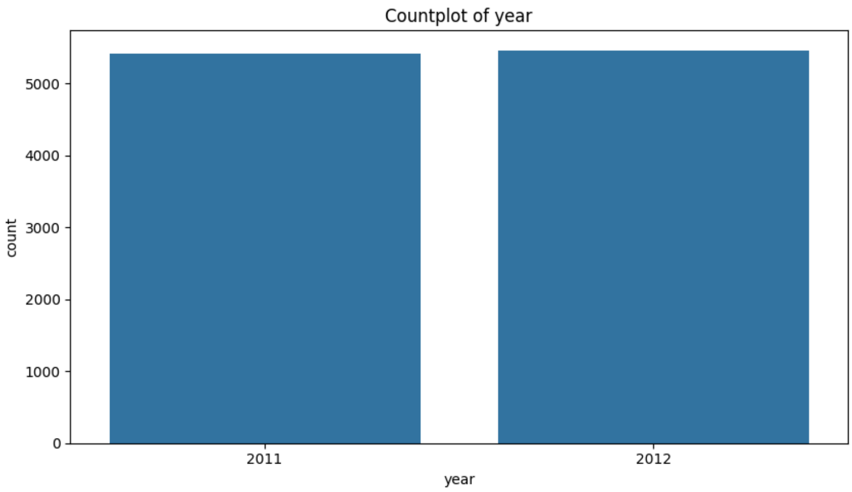
```
df_new.dtypes
```

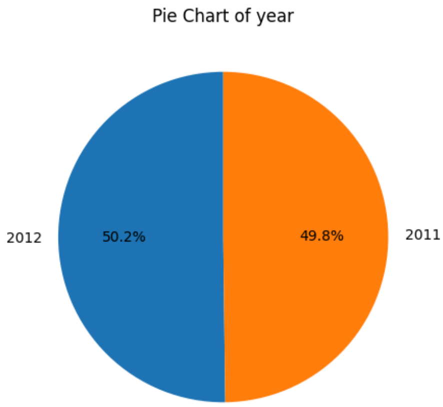
	0
datetime	datetime64[ns]
season	category
holiday	category
workingday	category
weather	category
temp	float64
atemp	float64
humidity	int64
windspeed	float64
casual	int64
registered	int64
count	int64

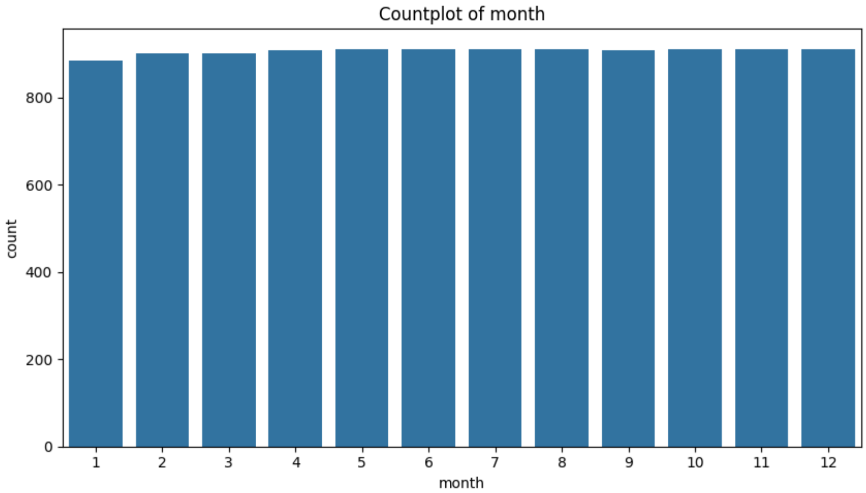
dtype: object

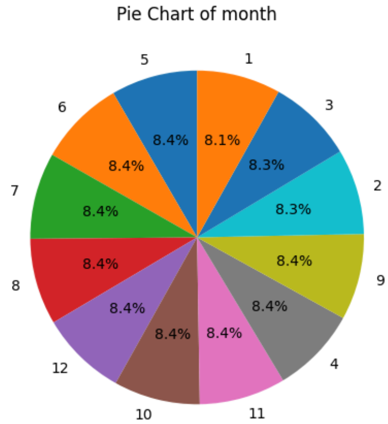
```
# Extracting categorical features from datetime and plotting
df_new['year'] = df_new['datetime'].dt.year
df_new['month'] = df_new['datetime'].dt.month
df_new['day'] = df_new['datetime'].dt.day
df_new['hour'] = df_new['datetime'].dt.hour

# Function to plot countplot and pie chart
def plot_categorical_feature(column):
    plt.figure(figsize=(15, 5))

    # Countplot
    plt.subplot(1, 2, 1)
    sns.countplot(x=column, data=df_new)
    plt.title(f'Countplot of {column}')
    plt.tight_layout()

    # Pie Chart
```

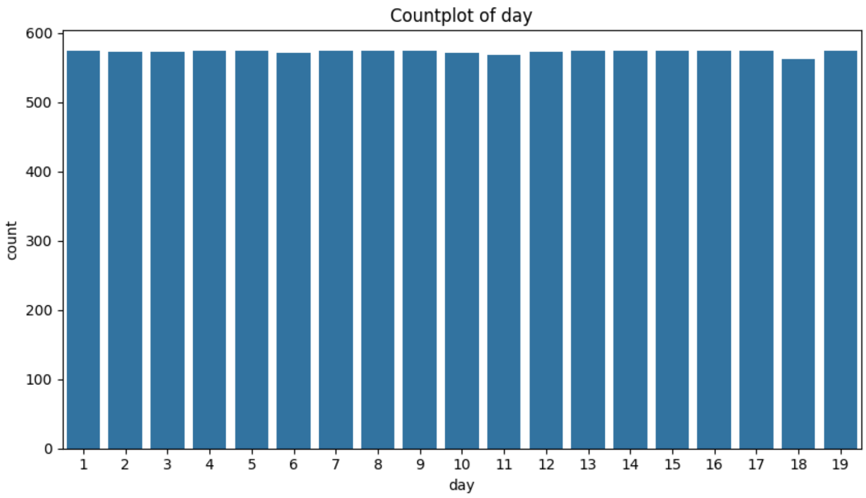
```
    plt.subplot(1, 2, 2)
    df_new[column].value_counts().plot.pie(autopct='%1.1f%%', startangle=90)
    plt.title(f'Pie Chart of {column}')
    plt.ylabel('')
    plt.tight_layout()

    plt.show()

# Plotting for year, month, day, and hour
plot_categorical_feature('year')
plot_categorical_feature('month')
plot_categorical_feature('day')
plot_categorical_feature('hour')
```

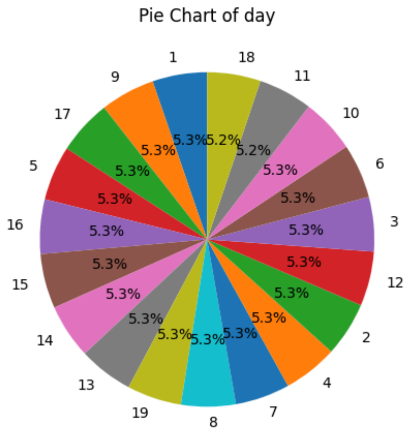

Countplot of year
count
0
1000
2000
3000
4000
5000
2011
2012
year

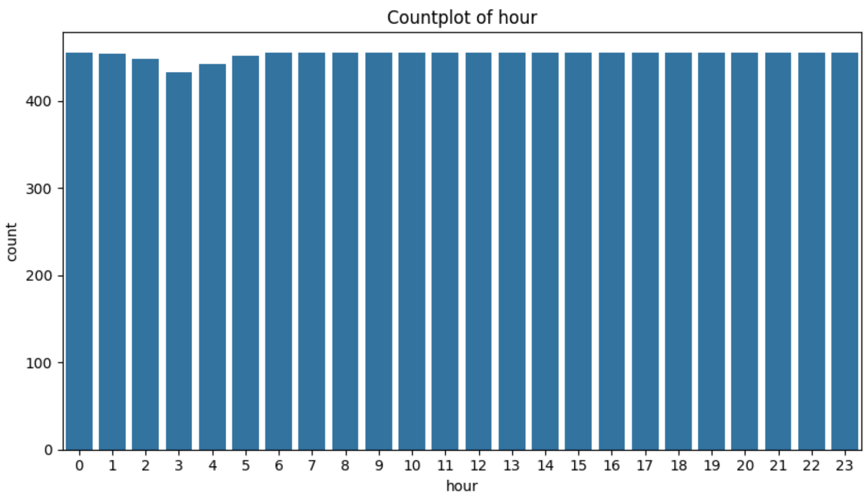

Pie Chart of year
2012
50.2%
49.8%
2011

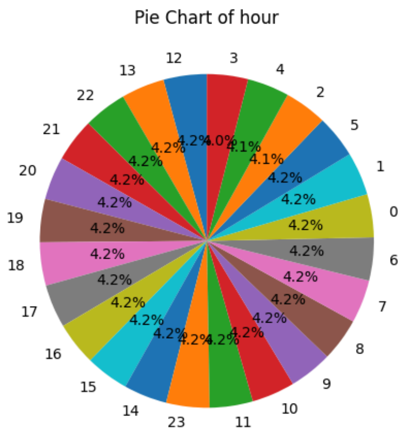

Countplot of month
count
0
200
400
600
800
1 2 3 4 5 6 7 8 9 10 11 12
month


Pie Chart of month
5 1 3 2 9 4 11 10 12 8 7 6
8.4% 8.1% 8.3% 8.3% 8.4% 8.4% 8.4% 8.4% 8.4% 8.4% 8.4% 8.4%


Countplot of day
count
0
100
200
300
400
500
600
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
day


Pie Chart of day
1 18 11 10 6 3 12 2 4 7 8 19 13 14 15 16 5 17 9
5.3% 5.2% 5.2% 5.3%


Countplot of hour
count
0
100
200
300
400
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
hour


Pie Chart of hour
12 3 4 2 5 1 0 6 7 8 9 10 11 23 14 15 16 17 18 19 20 21 22 13
4.2% 4.0% 4.1% 4.1% 4.2%

Business Insights based on Non-Graphical and Visual Analysis

```
# Comments on the range of attributes
print("Comments on the range of attributes:")
for column in df.columns:
    if pd.api.types.is_categorical_dtype(df[column]):
        df[column] = df[column].astype('category').cat.as_ordered()

    print(f"{column}: {df[column].min()} to {df[column].max()}")
```

```
Comments on the range of attributes:
datetime: 2011-01-01 00:00:00 to 2012-12-19 23:00:00
season: 1 to 4
holiday: 0 to 1
workingday: 0 to 1
weather: 1 to 4
temp: 0.82 to 41.0
atemp: 0.76 to 45.455
humidity: 0 to 100
windspeed: 0.0 to 56.9969
casual: 0 to 367
registered: 0 to 886
count: 1 to 977
```

Insights:

1) Datetime:

- Range: 2011-01-01 00:00:00 to 2012-12-19 23:00:00
- Comment: The dataset spans nearly two years, providing a comprehensive view of bike rental patterns over different seasons and weather conditions.

2) Season:

- Range: 1 to 4 (1: Spring, 2: Summer, 3: Fall, 4: Winter)
- Comment: The dataset includes all four seasons, allowing for analysis of seasonal trends in bike rentals.

3) Holiday:

- Range: 0 to 1 (0: Non-holiday, 1: Holiday)
- Comment: The binary nature of this variable helps in distinguishing bike rental patterns on holidays versus non-holidays.

4) Workingday:

- Range: 0 to 1 (0: Weekend or Holiday, 1: Working Day)
- Comment: This variable is useful for analyzing differences in bike rentals between working days and weekends/holidays.

5) Weather:

- Range: 1 to 4 (1: Clear, Few clouds, 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 Pellets + Thunderstorm + Mist, Snow + Fog)
- Comment: The weather conditions are categorized into four distinct types, which can be used to study the impact of weather on bike rentals.

6) Temp (Temperature in Celsius):

- Range: 0.82 to 41.0
- Comment: The temperature range covers a wide spectrum, from very cold to very hot, which can influence bike rental behavior.

7) Atemp (Feeling Temperature in Celsius):

- Range: 0.76 to 45.455
- Comment: The apparent temperature range is slightly broader than the actual temperature, reflecting how weather conditions are perceived by users.

8) Humidity:

- Range: 0 to 100
- Comment: The full range of humidity levels is represented, which can affect the comfort level of bike riders.

9) Windspeed:

- Range: 0.0 to 56.9969
- Comment: The wind speed varies significantly, which can impact the ease of riding and thus the number of rentals.

10) Casual (Count of Casual Users):

- Range: 0 to 367
- Comment: The number of casual users varies widely, indicating different levels of spontaneous bike rentals.

11) Registered (Count of Registered Users):

- Range: 0 to 886
- Comment: The number of registered users also varies, showing the extent of regular bike usage.

12) Count (Total Rental Bikes):

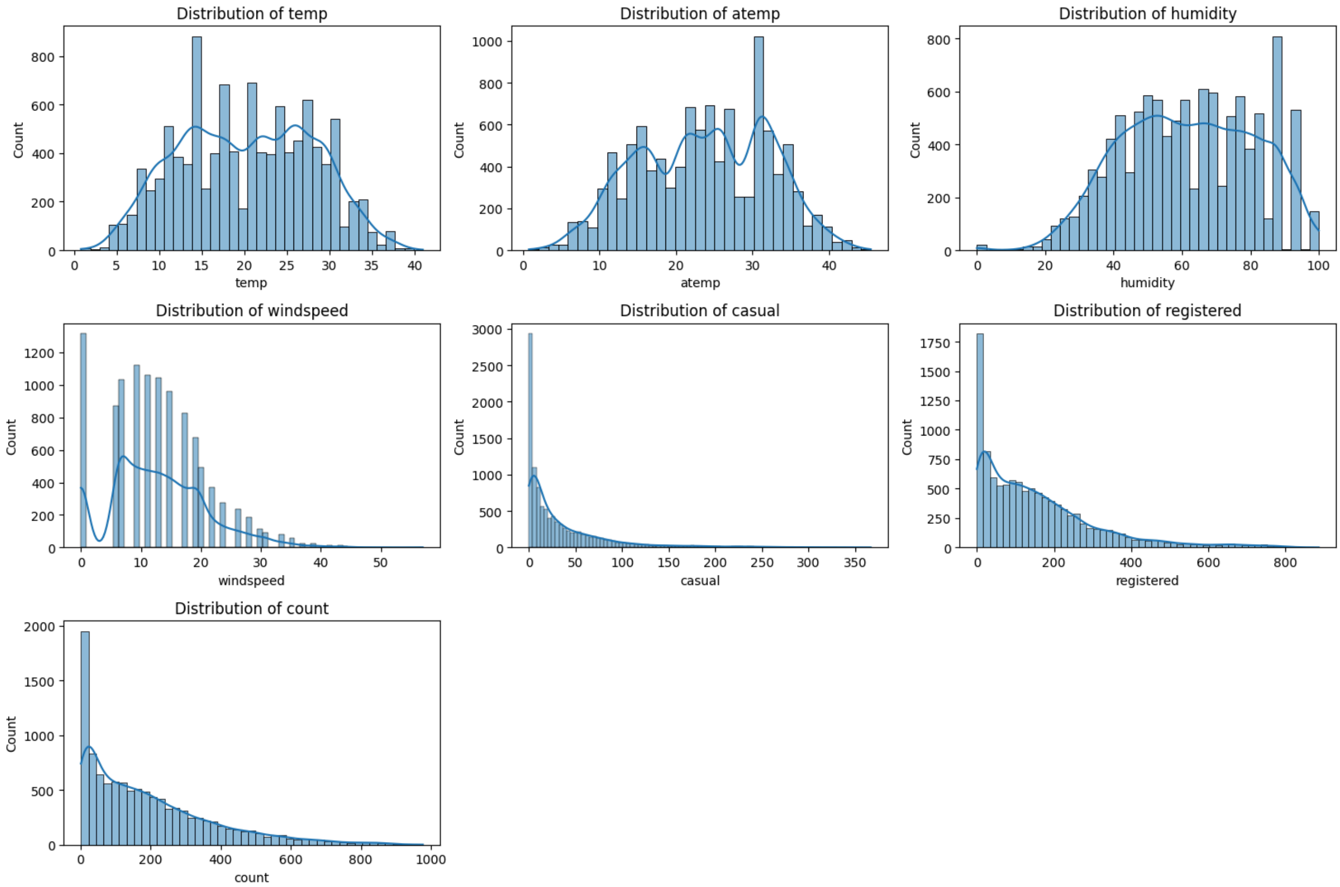
- Range: 1 to 977
- Comment: The total count of bike rentals ranges from 1 to 977, reflecting the overall demand for bike rentals.

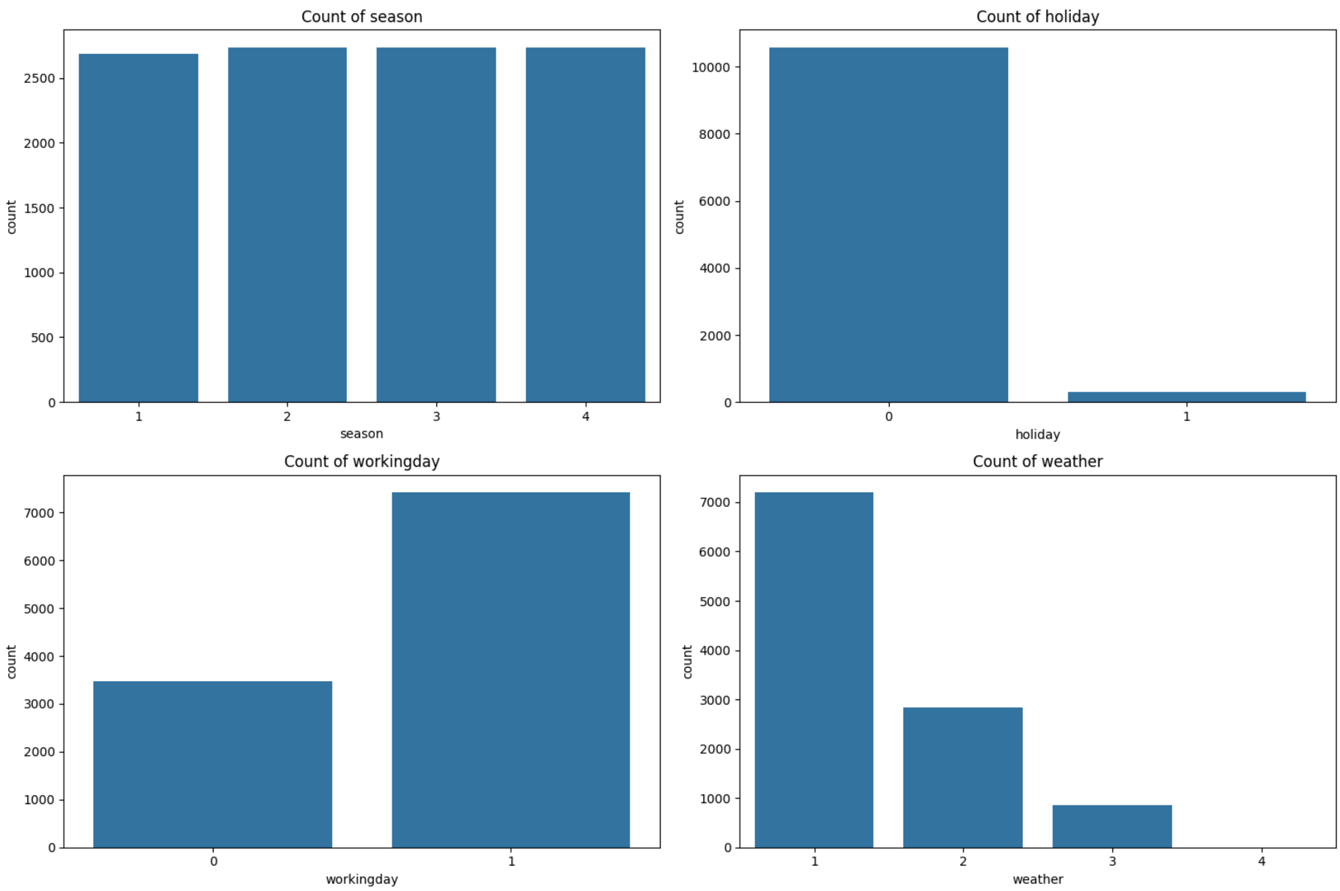
```
# Comments on the distribution of the variables and relationship between them
def comments_on_distribution(df):
    comments = []

    for column in df.columns:
        comments.append(f"{column}:")
        if pd.api.types.is_numeric_dtype(df[column]):
            comments.append(f" - Distribution: The {column} distribution spans from {df[column].min()} to {df[column].max()}.")
        else:
            comments.append(f" - Distribution: The {column} distribution includes categories: {df[column].unique().tolist()}.")

        # Relationship with other variables
        comments.append(" - Relationship with Other Variables:")
        for other_column in df.columns:
            if column != other_column:
                comments.append(f"   - {other_column}: Relationship analysis between {column} and {other_column}.")

        comments.append("\n")

    return "\n".join(comments)

# Print the comments on distribution and relationships
print(comments_on_distribution(df))
```

```
datetime:
 - Distribution: The datetime distribution includes categories: [Timestamp('2011-01-01 00:00:00'), Timestamp('2011-01-01 01:00:0
 - Relationship with Other Variables:
   - season: Relationship analysis between datetime and season.
   - holiday: Relationship analysis between datetime and holiday.
   - workingday: Relationship analysis between datetime and workingday.
   - weather: Relationship analysis between datetime and weather.
   - temp: Relationship analysis between datetime and temp.
   - atemp: Relationship analysis between datetime and atemp.
   - humidity: Relationship analysis between datetime and humidity.
   - windspeed: Relationship analysis between datetime and windspeed.
   - casual: Relationship analysis between datetime and casual.
   - registered: Relationship analysis between datetime and registered.
   - count: Relationship analysis between datetime and count.


season:
 - Distribution: The season distribution includes categories: [1, 2, 3, 4].
 - Relationship with Other Variables:
   - datetime: Relationship analysis between season and datetime.
   - holiday: Relationship analysis between season and holiday.
   - workingday: Relationship analysis between season and workingday.
   - weather: Relationship analysis between season and weather.
   - temp: Relationship analysis between season and temp.
   - atemp: Relationship analysis between season and atemp.
   - humidity: Relationship analysis between season and humidity.
   - windspeed: Relationship analysis between season and windspeed.
   - casual: Relationship analysis between season and casual.
   - registered: Relationship analysis between season and registered.
   - count: Relationship analysis between season and count.


holiday:
 - Distribution: The holiday distribution includes categories: [0, 1].
 - Relationship with Other Variables:
   - datetime: Relationship analysis between holiday and datetime.
   - season: Relationship analysis between holiday and season.
   - workingday: Relationship analysis between holiday and workingday.
   - weather: Relationship analysis between holiday and weather.
   - temp: Relationship analysis between holiday and temp.
   - atemp: Relationship analysis between holiday and atemp.
```

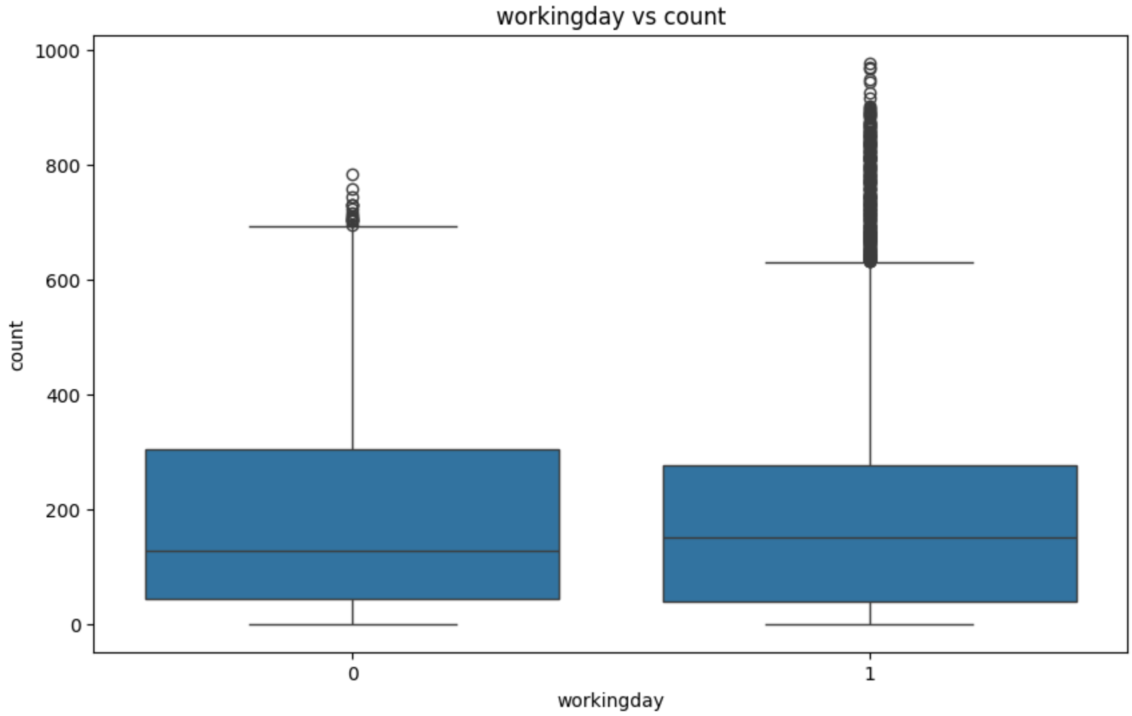
```
        - humidity: Relationship analysis between holiday and humidity.
        - windspeed: Relationship analysis between holiday and windspeed.
        - casual: Relationship analysis between holiday and casual.
        - registered: Relationship analysis between holiday and registered.
        - count: Relationship analysis between holiday and count.


     workingday:
      - Distribution: The workingday distribution includes categories: [0, 1].
      - Relationship with Other Variables:
        - datetime: Relationship analysis between workingday and datetime.
        - season: Relationship analysis between workingday and season.
        - holiday: Relationship analysis between workingday and holiday.
        - weather: Relationship analysis between workingday and weather.
        - temp: Relationship analysis between workingday and temp.
```

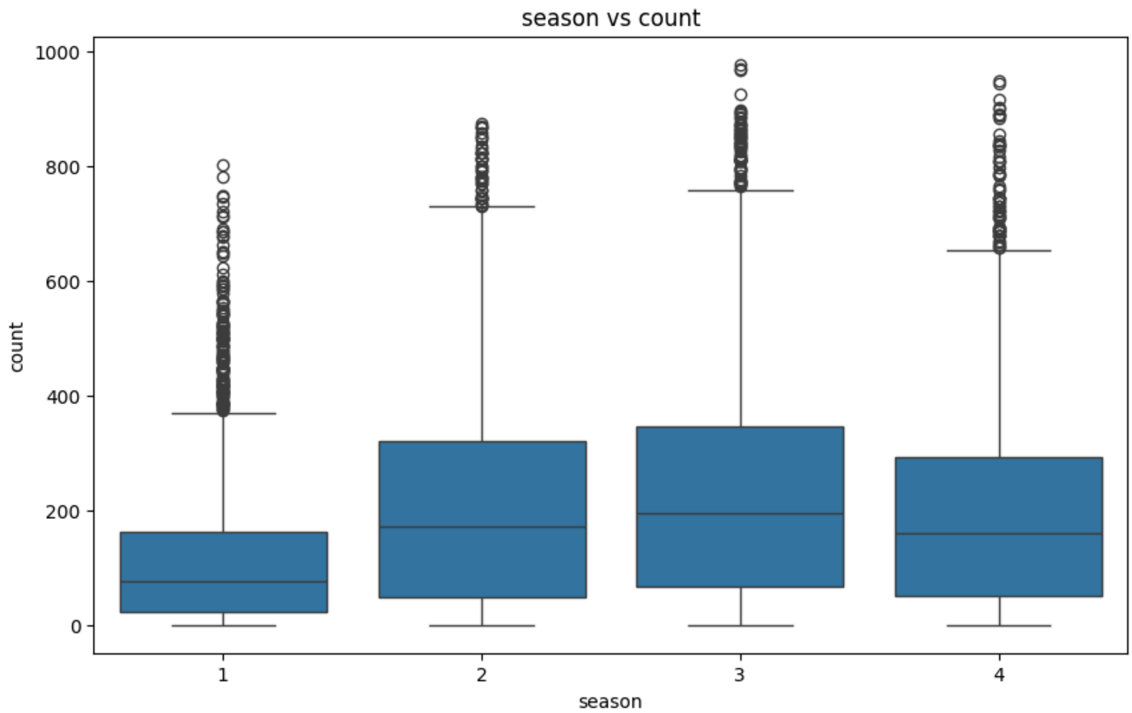
```
#Univariate Analysis (distribution plots of all the continuous variable(s) barplots/countplots of all the categorical variables)

# Univariate Analysis for Continuous Variables
continuous_vars = ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count']

plt.figure(figsize=(15, 10))
for i, column in enumerate(continuous_vars, 1):
    plt.subplot(3, 3, i)
    sns.histplot(df[column], kde=True)
    plt.title(f'Distribution of {column}')
plt.tight_layout()
plt.show()

# Univariate Analysis for Categorical Variables
categorical_vars = ['season', 'holiday', 'workingday', 'weather']

plt.figure(figsize=(15, 10))
for i, column in enumerate(categorical_vars, 1):
    plt.subplot(2, 2, i)
    sns.countplot(x=column, data=df)
    plt.title(f'Count of {column}')
plt.tight_layout()
plt.show()
```

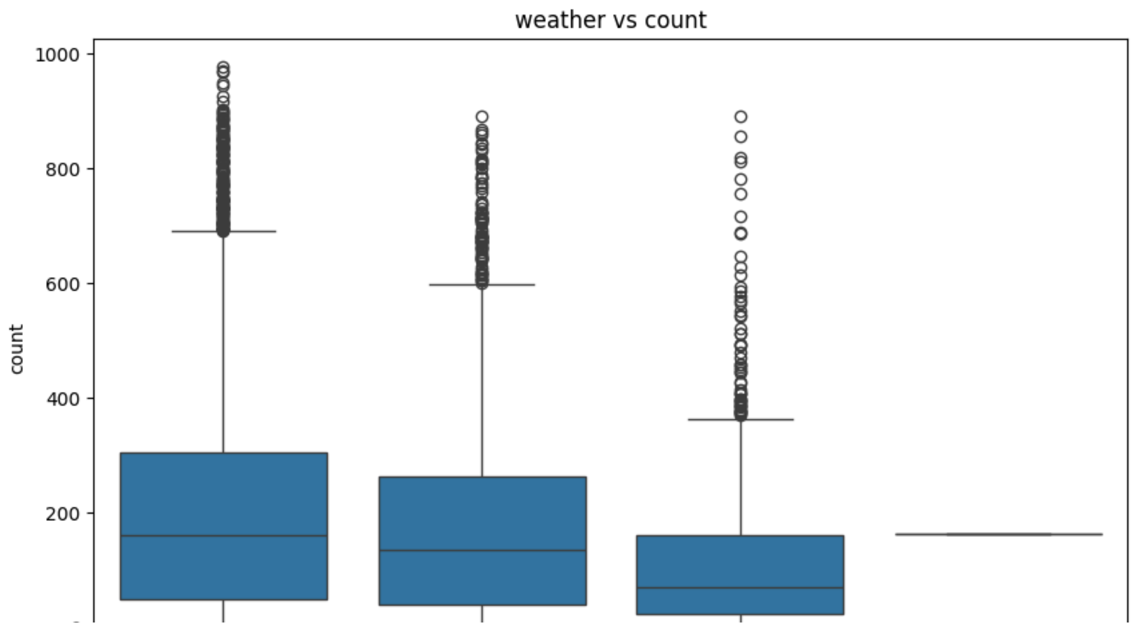

Distribution of temp
Count
temp
Distribution of atemp
Count
atemp
Distribution of humidity
Count
humidity
Distribution of windspeed
Count
windspeed
Distribution of casual
Count
casual
Distribution of registered
Count
registered
Distribution of count
Count
count

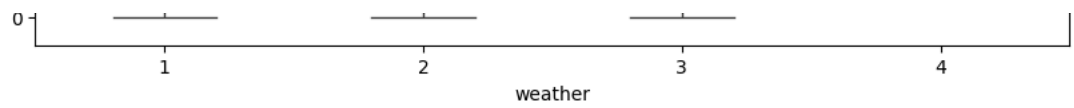

Count of season
count
season
Count of holiday
count
holiday
Count of workingday
count
workingday
Count of weather
count
weather

```
#Bivariate Analysis (Relationships between important variables such as workday and count, season and count, weather and count.
# Function to plot bivariate distributions
def plot_bivariate(x, y):
    plt.figure(figsize=(10, 6))
    sns.boxplot(x=x, y=y, data=df)
    plt.title(f'{x} vs {y}')
    plt.xlabel(x)
    plt.ylabel(y)
    plt.show()

# Plotting Bivariate Distributions for important variables
plot_bivariate('workingday', 'count')
plot_bivariate('season', 'count')
plot_bivariate('weather', 'count')
```


workingday vs count
count
workingday
0
1
0
200
400
600
800
1000


season vs count
count
season
1
2
3
4
0
200
400
600
800
1000


weather vs count
count
200
400
600
800
1000




```
# Comments for each univariate and bivariate plot
def generate_comments(df):
    comments = []

    # Univariate comments
    for column in df.columns:
        comments.append(f"Univariate Plot ({column}):")
        if pd.api.types.is_numeric_dtype(df[column]):
            comments.append(f" - The {column} distribution spans from {df[column].min()} to {df[column].max()}.")
        else:
            comments.append(f" - The {column} distribution includes categories: {df[column].unique().tolist()}.")
        comments.append("\n")

    # Bivariate comments
    for i in range(len(df.columns)):
        for j in range(i + 1, len(df.columns)):
            x = df.columns[i]
            y = df.columns[j]
            comments.append(f"Bivariate Plot ({x} vs {y}):")
            comments.append(f" - Relationship analysis between {x} and {y}.")
            comments.append("\n")

    return "\n".join(comments)



# Print the comments on distribution and relationships
print(generate_comments(df))
```

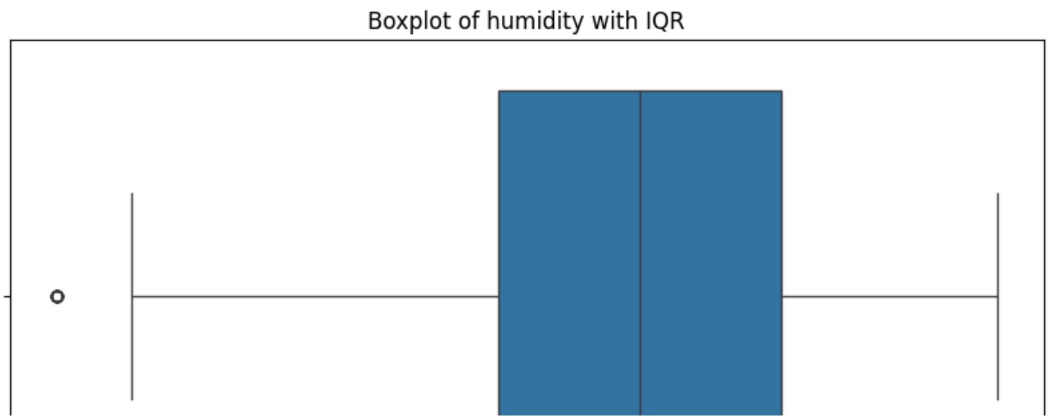
```
Univariate Plot (datetime):
 - The datetime distribution includes categories: [Timestamp('2011-01-01 00:00:00'), Timestamp('2011-01-01 01:00:00'), Timestamp('2011-01-01 02:00:00'), Timestamp('2011-01-01 0
imestamp('2011-05-07 04:00:00'), Timestamp('2011-05-07 05:00:00'), Timestamp('2011-05-07 06:00:00'), Timestamp('2011-05-07 07:00:00'), Timestamp('2011-05-07 08:00:00'), Tim
11 13:00:00'), Timestamp('2011-09-11 14:00:00'), Timestamp('2011-09-11 15:00:00'), Timestamp('2011-09-11 16:00:00'), Timestamp('2011-09-11 17:00:00'), Timestamp('2011-09-11
estamp('2012-01-16 01:00:00'), Timestamp('2012-01-16 02:00:00'), Timestamp('2012-01-16 03:00:00'), Timestamp('2012-01-16 04:00:00'), Timestamp('2012-01-16 05:00:00'), Times
 13:00:00'), Timestamp('2012-06-01 14:00:00'), Timestamp('2012-06-01 15:00:00'), Timestamp('2012-06-01 16:00:00'), Timestamp('2012-06-01 17:00:00'), Timestamp('2012-06-01 18
tamp('2012-10-05 21:00:00'), Timestamp('2012-10-05 22:00:00'), Timestamp('2012-10-05 23:00:00'), Timestamp('2012-10-06 00:00:00'), Timestamp('2012-10-06 01:00:00'), Timesta


Univariate Plot (season):
 - The season distribution includes categories: [1, 2, 3, 4].


Univariate Plot (holiday):
 - The holiday distribution includes categories: [0, 1].


Univariate Plot (workingday):
 - The workingday distribution includes categories: [0, 1].


Univariate Plot (weather):
 - The weather distribution includes categories: [1, 2, 3, 4].


Univariate Plot (temp):
 - The temp distribution spans from 0.82 to 41.0.


Univariate Plot (atemp):
 - The atemp distribution spans from 0.76 to 45.455.


Univariate Plot (humidity):
 - The humidity distribution spans from 0 to 100.


Univariate Plot (windspeed):
 - The windspeed distribution spans from 0.0 to 56.9969.


Univariate Plot (casual):
 - The casual distribution spans from 0 to 367.


Univariate Plot (registered):
 - The registered distribution spans from 0 to 886.


Univariate Plot (count):
 - The count distribution spans from 1 to 977.


Bivariate Plot (datetime vs season):
 - Relationship analysis between datetime and season.


Bivariate Plot (datetime vs holiday):
 - Relationship analysis between datetime and holiday.


Bivariate Plot (datetime vs workingday):
 - Relationship analysis between datetime and workingday.


Bivariate Plot (datetime vs weather):
 - Relationship analysis between datetime and weather.
```

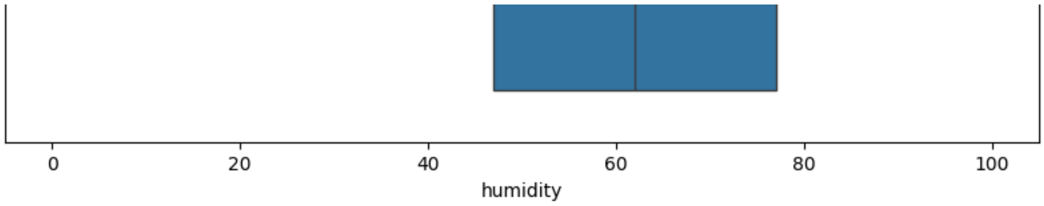
```
Bivariate Plot (registered vs count):
 - Relationship analysis between registered and count.
```

2.Check for Outliers and deal with them accordingly

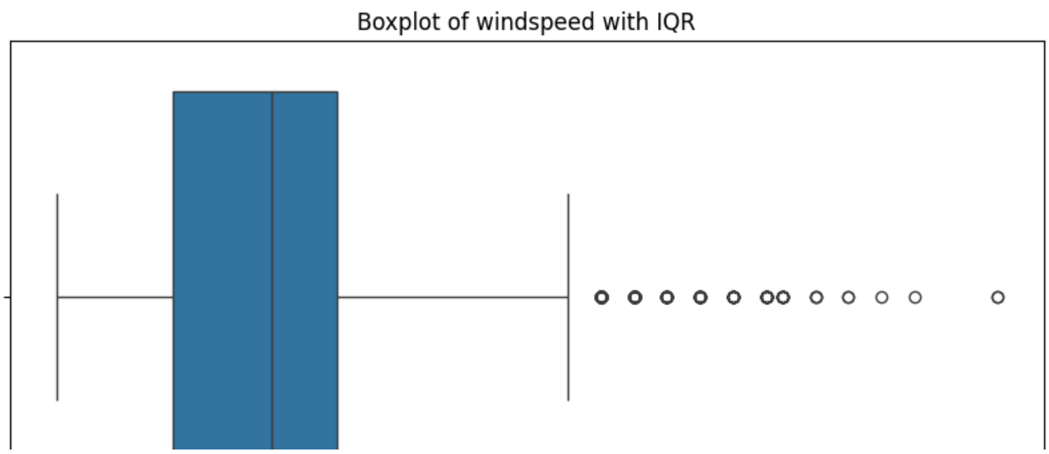
```
# List of continuous columns
continuous_columns = ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count']

#i)You can use Boxplot, Interquartile Range (IQR

# IQR Method to find outliers and plot boxplots
for column in continuous_columns:
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    print(column)
    print("Q1: ",Q1)
    print("Q3: ",Q3)
    print("IQR: ",IQR)
    print("Lower Bound: ", lower_bound)
    print("Upper Bound: ", upper_bound)
    outliers = df[(df[column] < lower_bound) | (df[column] > upper_bound)]
    print(f"Outliers in {column} (IQR Method): ")
    if outliers.empty:
      print("No,Outliers Detected for ", column , "\n")
    else:
      print(outliers)
      plt.figure(figsize=(10, 5))
      sns.boxplot(x=df[column])
      plt.title(f'Boxplot of {column} with IQR')
      plt.show()

      df = df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]

      # Print the new dataframe without outliers
      print("Dataframe without outliers for", column)
      print(df)
```

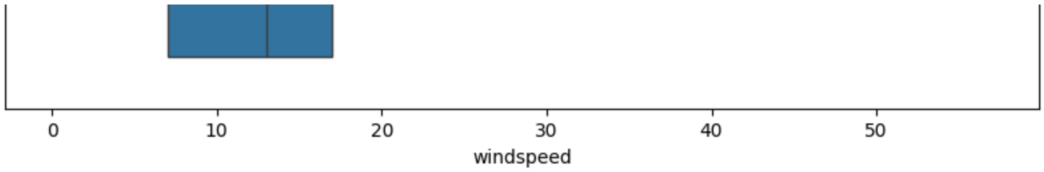
```
temp
Q1:  13.94
Q3:  26.24
IQR:  12.299999999999999
Lower Bound:  -4.51
Upper Bound:  44.69
Outliers in temp (IQR Method):
No,Outliers Detected for  temp

atemp
Q1:  16.665
Q3:  31.06
IQR:  14.395
Lower Bound:  -4.927500000000002
Upper Bound:  52.6525
Outliers in atemp (IQR Method):
No,Outliers Detected for  atemp

humidity
Q1:  47.0
Q3:  77.0
IQR:  30.0
Lower Bound:  2.0
Upper Bound:  122.0
Outliers in humidity (IQR Method):
                datetime season holiday workingday weather   temp   atemp  \
1091 2011-03-10 00:00:00      1       0          1       3  13.94  15.910
1092 2011-03-10 01:00:00      1       0          1       3  13.94  15.910
1093 2011-03-10 02:00:00      1       0          1       3  13.94  15.910
1094 2011-03-10 05:00:00      1       0          1       3  14.76  17.425
1095 2011-03-10 06:00:00      1       0          1       3  14.76  16.665
1096 2011-03-10 07:00:00      1       0          1       3  15.58  19.695
1097 2011-03-10 08:00:00      1       0          1       3  15.58  19.695
1098 2011-03-10 09:00:00      1       0          1       3  16.40  20.455
1099 2011-03-10 10:00:00      1       0          1       3  16.40  20.455
1100 2011-03-10 11:00:00      1       0          1       3  16.40  20.455
1101 2011-03-10 12:00:00      1       0          1       3  17.22  21.210
1102 2011-03-10 13:00:00      1       0          1       3  17.22  21.210
1103 2011-03-10 14:00:00      1       0          1       3  18.04  21.970
1104 2011-03-10 15:00:00      1       0          1       3  18.04  21.970
1105 2011-03-10 16:00:00      1       0          1       3  17.22  21.210
1106 2011-03-10 17:00:00      1       0          1       2  18.04  21.970
1107 2011-03-10 18:00:00      1       0          1       3  18.04  21.970
1108 2011-03-10 19:00:00      1       0          1       3  18.04  21.970
1109 2011-03-10 20:00:00      1       0          1       3  14.76  16.665
1110 2011-03-10 21:00:00      1       0          1       3  14.76  17.425
1111 2011-03-10 22:00:00      1       0          1       2  13.94  16.665
1112 2011-03-10 23:00:00      1       0          1       3  13.94  17.425

      humidity  windspeed  casual  registered  count
1091         0    16.9979       3           0      3
1092         0    16.9979       0           2      2
1093         0    16.9979       0           1      1
1094         0    12.9980       1           2      3
1095         0    22.0028       0          12     12
1096         0    15.0013       1          36     37
1097         0    19.0012       1          43     44
1098         0    15.0013       1          23     24
1099         0    11.0014       0          17     17
1100         0    16.9979       6           5     11
1101         0    15.0013       4          30     34
1102         0    15.0013       1          11     12
1103         0    19.9995       0          12     12
1104         0    15.0013       3          11     14
1105         0    16.9979       1          20     21
1106         0    26.0027       2         109    111
1107         0    23.9994       2          80     82
1108         0    39.0007       5          51     56
1109         0    22.0028       9          29     38
1110         0    15.0013       1          27     28
1111         0     8.9981       4          30     34
1112         0     6.0032       1          26     27
```





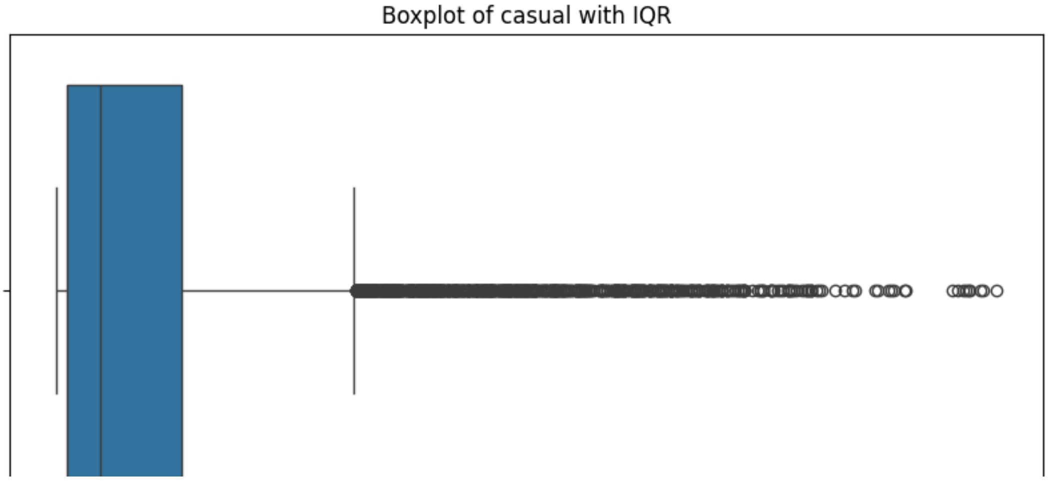

```
Dataframe without outliers for humidity
                 datetime season holiday workingday weather   temp   atemp  \
0     2011-01-01 00:00:00      1       0          0       1   9.84  14.395
1     2011-01-01 01:00:00      1       0          0       1   9.02  13.635
2     2011-01-01 02:00:00      1       0          0       1   9.02  13.635
3     2011-01-01 03:00:00      1       0          0       1   9.84  14.395
4     2011-01-01 04:00:00      1       0          0       1   9.84  14.395
...                   ...    ...     ...        ...     ...    ...     ...
10881 2012-12-19 19:00:00      4       0          1       1  15.58  19.695
10882 2012-12-19 20:00:00      4       0          1       1  14.76  17.425
10883 2012-12-19 21:00:00      4       0          1       1  13.94  15.910
10884 2012-12-19 22:00:00      4       0          1       1  13.94  17.425
10885 2012-12-19 23:00:00      4       0          1       1  13.12  16.665

       humidity  windspeed  casual  registered  count
0            81     0.0000       3          13     16
1            80     0.0000       8          32     40
2            80     0.0000       5          27     32
3            75     0.0000       3          10     13
4            75     0.0000       0           1      1
...         ...        ...     ...         ...    ...
10881        50    26.0027       7         329    336
10882        57    15.0013      10         231    241
10883        61    15.0013       4         164    168
10884        61     6.0032      12         117    129
10885        66     8.9981       4          84     88

[10864 rows x 12 columns]
windspeed
Q1:  7.0015
Q3:  16.9979
IQR:  9.996400000000001
Lower Bound:  -7.993100000000002
Upper Bound:  31.992500000000003
Outliers in windspeed (IQR Method):
                 datetime season holiday workingday weather   temp   atemp  \
175   2011-01-08 14:00:00      1       0          0       1   8.20   8.335
178   2011-01-08 17:00:00      1       0          0       1   6.56   6.060
194   2011-01-09 09:00:00      1       0          0       1   4.92   3.790
196   2011-01-09 11:00:00      1       0          0       1   6.56   6.060
265   2011-01-12 12:00:00      1       0          1       1   8.20   7.575
...                   ...    ...     ...        ...     ...    ...     ...
10013 2012-11-02 14:00:00      4       0          1       2  16.40  20.455
10154 2012-11-08 12:00:00      4       0          1       1  16.40  20.455
10263 2012-11-13 01:00:00      4       0          1       3  18.04  21.970
10540 2012-12-05 14:00:00      4       0          1       1  19.68  23.485
10853 2012-12-18 15:00:00      4       0          1       1  18.86  22.725

       humidity  windspeed  casual  registered  count
175          32    32.9975      12          83     95
178          37    36.9974       5          64     69
194          46    35.0008       0          19     19
196          40    35.0008       2          47     49
265          47    39.0007       3          52     55
...         ...        ...     ...         ...    ...
10013        40    32.9975      63         199    262
10154        24    32.9975      33         202    235
10263        88    43.0006       0           5      5
10540        33    32.9975      39         179    218
10853        44    32.9975      28         218    246

[226 rows x 12 columns]
```

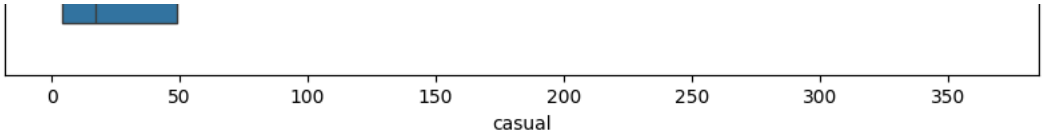





```
Dataframe without outliers for windspeed
                 datetime season holiday workingday weather   temp   atemp  \
0     2011-01-01 00:00:00      1       0          0       1   9.84  14.395
1     2011-01-01 01:00:00      1       0          0       1   9.02  13.635
2     2011-01-01 02:00:00      1       0          0       1   9.02  13.635
3     2011-01-01 03:00:00      1       0          0       1   9.84  14.395
4     2011-01-01 04:00:00      1       0          0       1   9.84  14.395
...                   ...    ...     ...        ...     ...    ...     ...
10881 2012-12-19 19:00:00      4       0          1       1  15.58  19.695
10882 2012-12-19 20:00:00      4       0          1       1  14.76  17.425
10883 2012-12-19 21:00:00      4       0          1       1  13.94  15.910
10884 2012-12-19 22:00:00      4       0          1       1  13.94  17.425
10885 2012-12-19 23:00:00      4       0          1       1  13.12  16.665

       humidity  windspeed  casual  registered  count
0            81     0.0000       3          13     16
1            80     0.0000       8          32     40
2            80     0.0000       5          27     32
3            75     0.0000       3          10     13
4            75     0.0000       0           1      1
...         ...        ...     ...         ...    ...
10881        50    26.0027       7         329    336
10882        57    15.0013      10         231    241
10883        61    15.0013       4         164    168
10884        61     6.0032      12         117    129
10885        66     8.9981       4          84     88

[10638 rows x 12 columns]
casual
Q1:  4.0
Q3:  49.0
IQR:  45.0
Lower Bound:  -63.5
Upper Bound:  116.5
Outliers in casual (IQR Method):
                 datetime season holiday workingday weather   temp   atemp  \
1173  2011-03-13 14:00:00      1       0          0       1  18.86  22.725
1174  2011-03-13 15:00:00      1       0          0       1  19.68  23.485
1175  2011-03-13 16:00:00      1       0          0       1  18.86  22.725
1311  2011-03-19 12:00:00      1       0          0       1  18.86  22.725
1312  2011-03-19 13:00:00      1       0          0       1  20.50  24.240
...                   ...    ...     ...        ...     ...    ...     ...
10610 2012-12-08 12:00:00      4       0          0       2  16.40  20.455
10611 2012-12-08 13:00:00      4       0          0       2  16.40  20.455
10612 2012-12-08 14:00:00      4       0          0       2  16.40  20.455
10613 2012-12-08 15:00:00      4       0          0       1  17.22  21.210
10614 2012-12-08 16:00:00      4       0          0       1  17.22  21.210

       humidity  windspeed  casual  registered  count
1173         41    19.9995     144         106    250
1174         39    22.0028     149         155    304
1175         41    26.0027     124         132    256
1311         33    27.9993     126         141    267
1312         34    31.0009     174         127    301
...         ...        ...     ...         ...    ...
10610        87    15.0013     122         364    486
10611        87    11.0014     148         399    547
10612        87    19.0012     164         378    542
10613        82    11.0014     167         374    541
10614        82    11.0014     139         368    507

[737 rows x 12 columns]
```

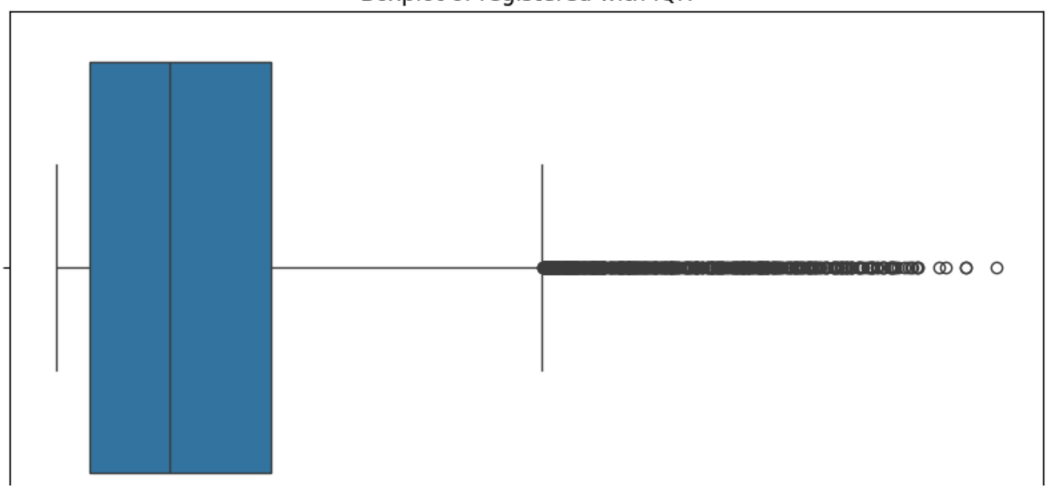
Boxplot of casual with IQR

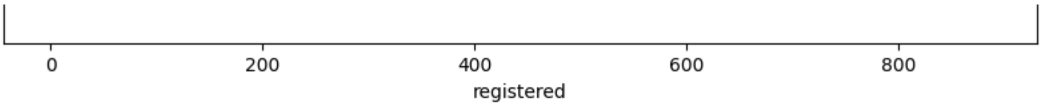





```
Dataframe without outliers for casual
                  datetime season holiday workingday weather   temp   atemp  \
0      2011-01-01 00:00:00      1       0          0       1   9.84  14.395
1      2011-01-01 01:00:00      1       0          0       1   9.02  13.635
2      2011-01-01 02:00:00      1       0          0       1   9.02  13.635
3      2011-01-01 03:00:00      1       0          0       1   9.84  14.395
4      2011-01-01 04:00:00      1       0          0       1   9.84  14.395
...                    ...    ...     ...        ...     ...    ...     ...
10881 2012-12-19 19:00:00      4       0          1       1  15.58  19.695
10882 2012-12-19 20:00:00      4       0          1       1  14.76  17.425
10883 2012-12-19 21:00:00      4       0          1       1  13.94  15.910
10884 2012-12-19 22:00:00      4       0          1       1  13.94  17.425
10885 2012-12-19 23:00:00      4       0          1       1  13.12  16.665

       humidity  windspeed  casual  registered  count
0            81     0.0000       3          13     16
1            80     0.0000       8          32     40
2            80     0.0000       5          27     32
3            75     0.0000       3          10     13
4            75     0.0000       0           1      1
...         ...        ...     ...         ...    ...
10881        50    26.0027       7         329    336
10882        57    15.0013      10         231    241
10883        61    15.0013       4         164    168
10884        61     6.0032      12         117    129
10885        66     8.9981       4          84     88

[9901 rows x 12 columns]
registered
Q1:  31.0
Q3:  202.0
IQR:  171.0
Lower Bound:  -225.5
Upper Bound:  458.5
Outliers in registered (IQR Method):
                  datetime season holiday workingday weather   temp   atemp  \
1819  2011-05-02 17:00:00      2       0          1       1  27.06  31.060
1843  2011-05-03 17:00:00      2       0          1       1  28.70  32.575
1844  2011-05-03 18:00:00      2       0          1       1  28.70  32.575
1891  2011-05-05 17:00:00      2       0          1       1  22.96  26.515
1915  2011-05-06 17:00:00      2       0          1       1  23.78  27.275
...                    ...    ...     ...        ...     ...    ...     ...
10855 2012-12-18 17:00:00      4       0          1       1  16.40  20.455
10856 2012-12-18 18:00:00      4       0          1       1  15.58  19.695
10870 2012-12-19 08:00:00      4       0          1       1   9.84  12.880
10879 2012-12-19 17:00:00      4       0          1       1  16.40  20.455
10880 2012-12-19 18:00:00      4       0          1       1  15.58  19.695

       humidity  windspeed  casual  registered  count
1819         65    12.9980      65         472    537
1843         54    31.0009      53         464    517
1844         48    27.9993      59         485    544
1891         26    26.0027      66         467    533
1915         40    23.9994      83         470    553
...         ...        ...     ...         ...    ...
10855        47    30.0026      39         533    572
10856        46    22.0028      13         512    525
10870        87     7.0015      13         665    678
10879        50    26.0027      26         536    562
10880        50    23.9994      23         546    569

[518 rows x 12 columns]
```

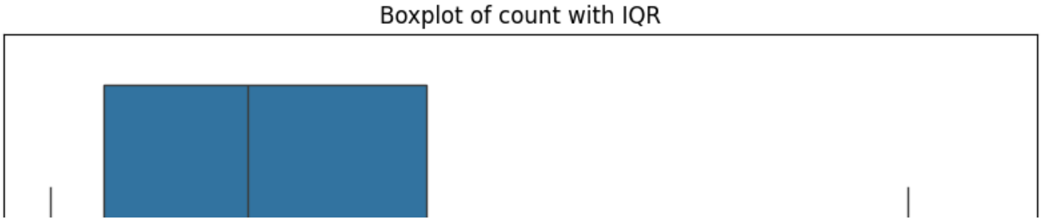
Boxplot of registered with IQR

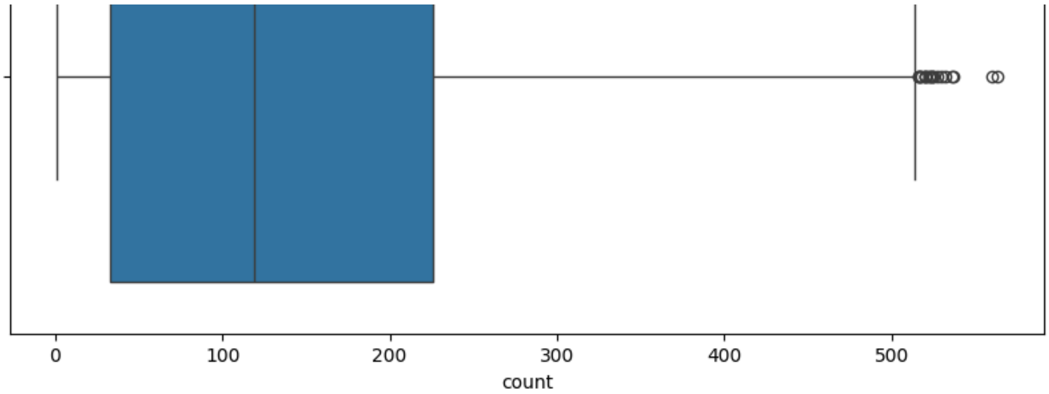





```
Dataframe without outliers for registered
                  datetime season holiday workingday weather   temp   atemp  \
0      2011-01-01 00:00:00      1       0          0       1   9.84  14.395
1      2011-01-01 01:00:00      1       0          0       1   9.02  13.635
2      2011-01-01 02:00:00      1       0          0       1   9.02  13.635
3      2011-01-01 03:00:00      1       0          0       1   9.84  14.395
4      2011-01-01 04:00:00      1       0          0       1   9.84  14.395
...                    ...    ...     ...        ...     ...    ...     ...
10881 2012-12-19 19:00:00      4       0          1       1  15.58  19.695
10882 2012-12-19 20:00:00      4       0          1       1  14.76  17.425
10883 2012-12-19 21:00:00      4       0          1       1  13.94  15.910
10884 2012-12-19 22:00:00      4       0          1       1  13.94  17.425
10885 2012-12-19 23:00:00      4       0          1       1  13.12  16.665

       humidity  windspeed  casual  registered  count
0            81     0.0000       3          13     16
1            80     0.0000       8          32     40
2            80     0.0000       5          27     32
3            75     0.0000       3          10     13
4            75     0.0000       0           1      1
...         ...        ...     ...         ...    ...
10881        50    26.0027       7         329    336
10882        57    15.0013      10         231    241
10883        61    15.0013       4         164    168
10884        61     6.0032      12         117    129
10885        66     8.9981       4          84     88

[9383 rows x 12 columns]
count
Q1:  33.0
Q3:  226.0
IQR:  193.0
Lower Bound:  -256.5
Upper Bound:  515.5
Outliers in count (IQR Method):
                  datetime season holiday workingday weather   temp   atemp  \
1820  2011-05-02 18:00:00      2       0          1       2  26.24  30.305
1892  2011-05-05 18:00:00      2       0          1       1  22.96  26.515
1988  2011-05-09 18:00:00      2       0          1       1  25.42  31.060
2276  2011-06-02 18:00:00      2       0          1       1  30.34  32.575
2467  2011-06-10 17:00:00      2       0          1       1  33.62  38.635
3140  2011-07-19 18:00:00      3       0          1       1  32.80  40.150
3236  2011-08-04 18:00:00      3       0          1       1  28.70  33.335
3355  2011-08-09 17:00:00      3       0          1       1  32.80  37.880
3427  2011-08-12 17:00:00      3       0          1       1  33.62  35.605
4145  2011-10-04 18:00:00      4       0          1       1  22.14  25.760
7642  2012-05-17 19:00:00      2       0          1       1  26.24  31.060
7879  2012-06-08 16:00:00      2       0          1       1  31.16  33.335
8215  2012-07-03 16:00:00      3       0          1       2  36.90  38.635
8289  2012-07-06 18:00:00      3       0          1       1  36.90  40.150
8578  2012-07-18 19:00:00      3       0          1       2  31.16  35.605
9319  2012-09-11 16:00:00      3       0          1       1  28.70  31.820
9391  2012-09-14 16:00:00      3       0          1       2  29.52  33.335
9799  2012-10-12 16:00:00      4       0          1       1  18.86  22.725
10466 2012-12-02 12:00:00      4       0          0       2  13.94  16.665

       humidity  windspeed  casual  registered  count
1820         65    12.9980      68         450    518
1892         26    19.0012      64         456    520
1988         38    19.9995      66         458    524
2276         22    26.0027      74         451    525
2467         52    15.0013      63         454    517
3140         66     8.9981      80         450    530
3236         74    19.9995      88         435    523
3355         55    11.0014      79         453    532
3427         26     0.0000      93         423    516
4145         64    19.0012      68         456    524
7642         31    12.9980      86         451    537
7879         29    19.9995     111         412    523
8215         24    12.9980      90         437    527
8289         33    16.9979     106         454    560
8578         62     7.0015      78         438    516
9319         28     8.9981      99         437    536
9391         51     0.0000     109         454    563
9799         41    23.9994     108         412    520
10466        81    11.0014     111         409    520
```

Boxplot of count with IQR






```
Dataframe without outliers for count
                 datetime season holiday workingday weather   temp   atemp  \
0     2011-01-01 00:00:00      1       0          0       1   9.84  14.395
1     2011-01-01 01:00:00      1       0          0       1   9.02  13.635
2     2011-01-01 02:00:00      1       0          0       1   9.02  13.635
3     2011-01-01 03:00:00      1       0          0       1   9.84  14.395
4     2011-01-01 04:00:00      1       0          0       1   9.84  14.395
...                   ...    ...     ...        ...     ...    ...     ...
10881 2012-12-19 19:00:00      4       0          1       1  15.58  19.695
10882 2012-12-19 20:00:00      4       0          1       1  14.76  17.425
10883 2012-12-19 21:00:00      4       0          1       1  13.94  15.910
10884 2012-12-19 22:00:00      4       0          1       1  13.94  17.425
10885 2012-12-19 23:00:00      4       0          1       1  13.12  16.665

       humidity  windspeed  casual  registered  count
0            81     0.0000       3          13     16
1            80     0.0000       8          32     40
2            80     0.0000       5          27     32
3            75     0.0000       3          10     13
4            75     0.0000       0           1      1
...         ...        ...     ...         ...    ...
10881        50    26.0027       7         329    336
10882        57    15.0013      10         231    241
10883        61    15.0013       4         164    168
10884        61     6.0032      12         117    129
10885        66     8.9981       4          84     88

[9364 rows x 12 columns]
```

Insights:

1) Temperature (temp):

- Q1 (25th percentile): 13.94
- Q3 (75th percentile): 26.24
- IQR (Interquartile Range): 12.30
- Lower Bound: -4.51
- Upper Bound: 44.69
- Outliers: No outliers detected.

2) Feels Like Temperature (atemp):

- Q1 (25th percentile): 16.665
- Q3 (75th percentile): 31.06
- IQR (Interquartile Range): 14.395
- Lower Bound: -4.93
- Upper Bound: 52.65
- Outliers: No outliers detected.

3) Humidity:

- Q1 (25th percentile): 47.0
- Q3 (75th percentile): 77.0
- IQR (Interquartile Range): 30.0
- Lower Bound: 2.0
- Upper Bound: 122.0
- Outliers: Detected (humidity = 0).

4) Casual:

- Q1 (25th percentile): 4.0
- Q3 (75th percentile): 49.0
- IQR (Interquartile Range): 45.0
- Lower Bound: -63.5
- Upper Bound: 116.5
- Outliers: Detected
- Min Outlier :117
- Max Outlier: 367

5) Registered:

- Q1 (25th percentile): 31.0
- Q3 (75th percentile): 202.0
- IQR (Interquartile Range): 171.0
- Lower Bound: -225.5
- Upper Bound: 458.5
- Outliers: Detected
- Min Outlier :459
- Max Outlier: 886

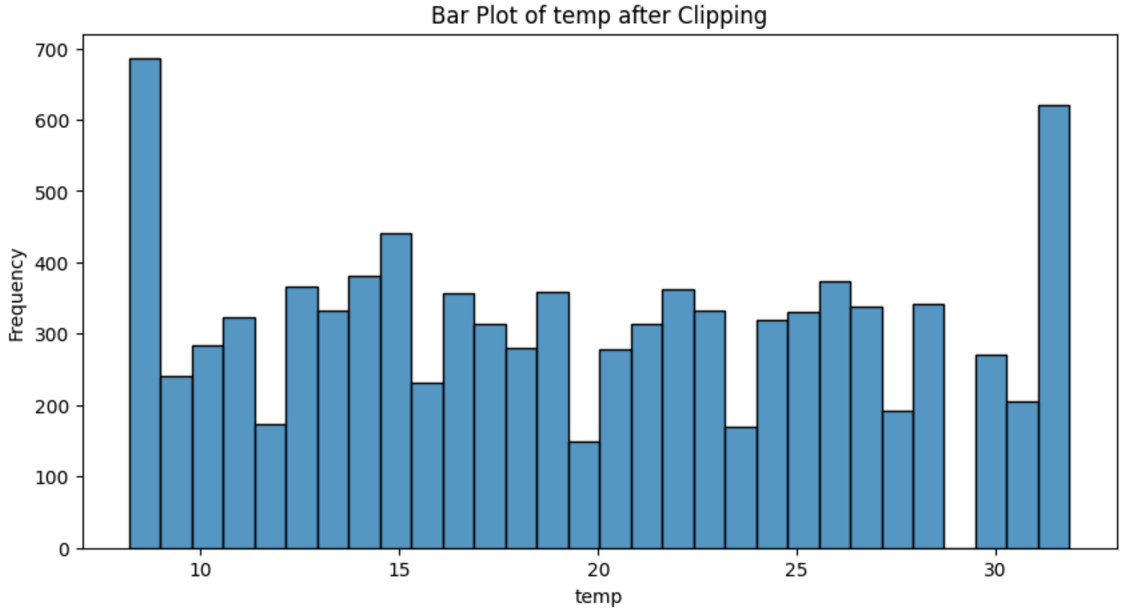
6) Count:

- Q1 (25th percentile): 33.0
- Q3 (75th percentile): 226.0
- IQR (Interquartile Range): 193.0
- Lower Bound: -256.5
- Upper Bound: 515.5
- Outliers: Detected
- Min Outlier :516
- Max Outlier: 977

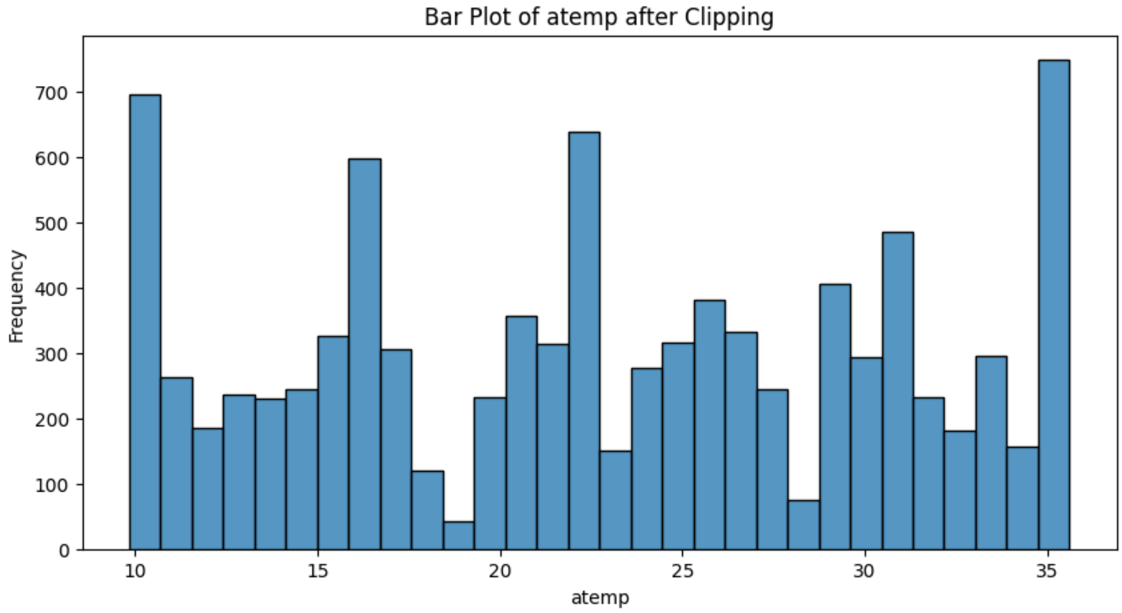
```
#b) Remove/clip the data between the 5 percentile and 95 percentile

# Clip the data between the 5th percentile and 95th percentile
for column in continuous_columns:
  lower_bound = np.percentile(df[column], 5)
  upper_bound = np.percentile(df[column], 95)
  print(column,"lower bound :",lower_bound)
  print(column,"upper bound :",upper_bound)
```

```
df[column] = np.clip(df[column], lower_bound, upper_bound)
# Plot the results as bar plots
plt.figure(figsize=(10, 5))
sns.histplot(df[column], bins=30, kde=False)
plt.title(f'Bar Plot of {column} after Clipping')
plt.xlabel(column)
plt.ylabel('Frequency')
plt.show()
```

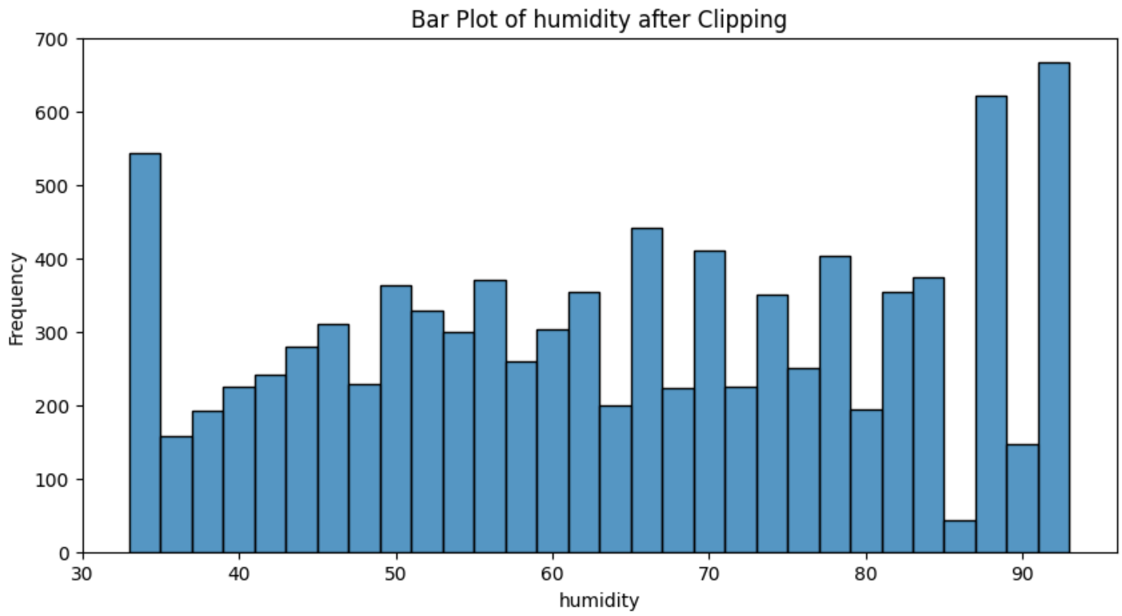
```
temp lower bound : 8.2
temp upper bound : 31.857000000000298
```




```
atemp lower bound : 9.85
atemp upper bound : 35.605
```

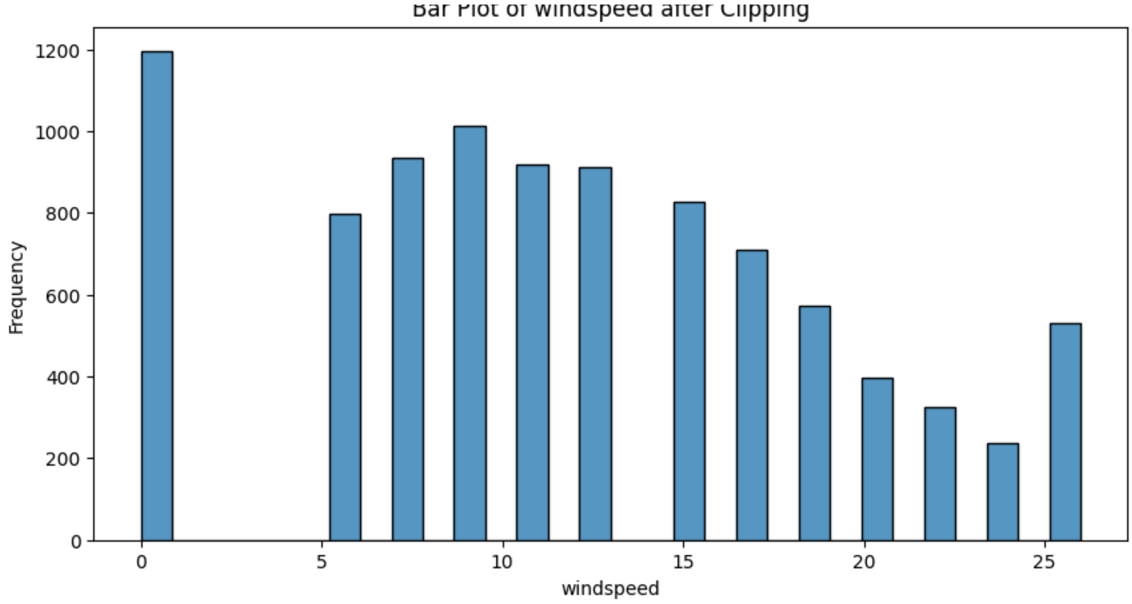



```
humidity lower bound : 33.0
humidity upper bound : 93.0
```



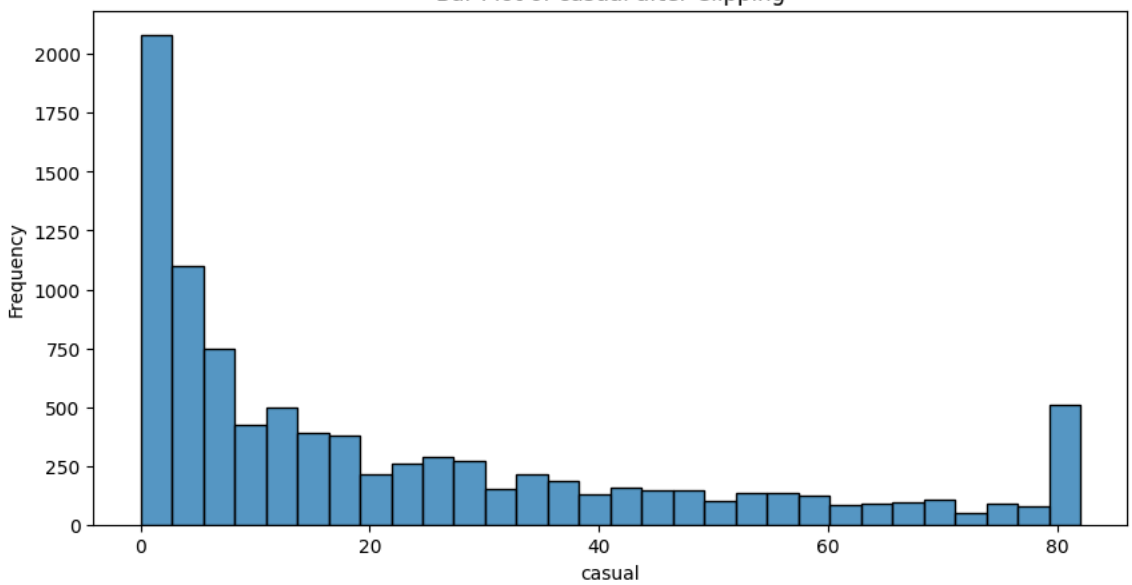

```
windspeed lower bound : 0.0
windspeed upper bound : 26.0027
```

Bar Plot of windspeed after Clipping



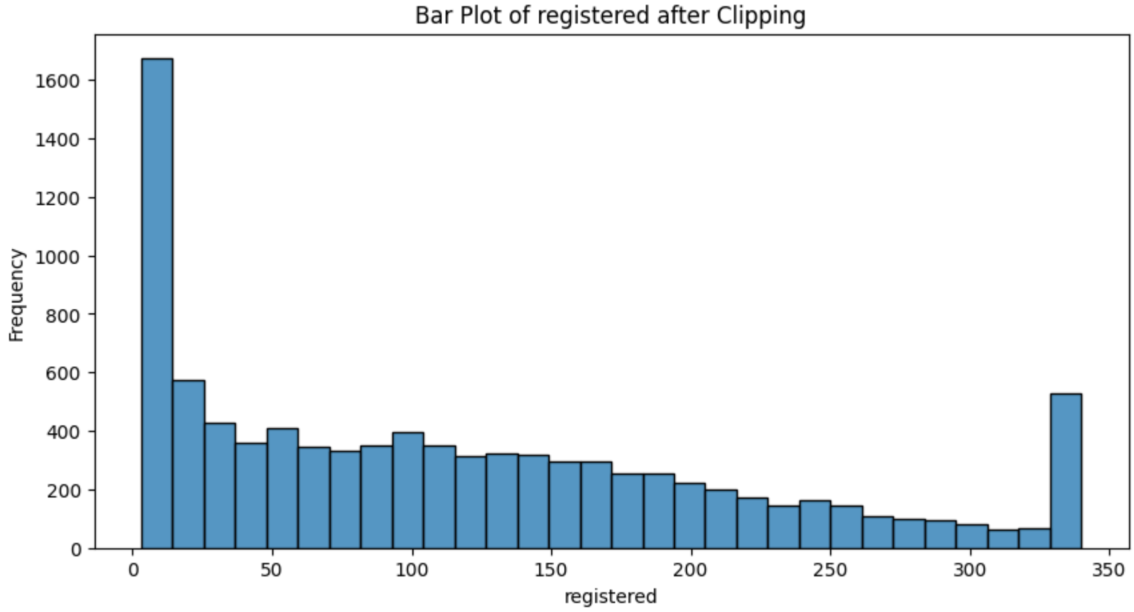

```
casual lower bound : 0.0
casual upper bound : 82.0
```

Bar Plot of casual after Clipping



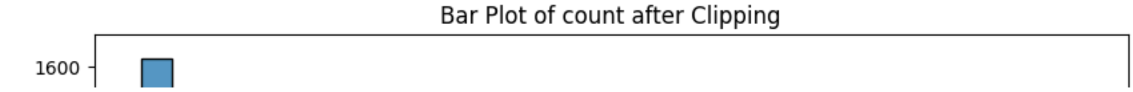

```
registered lower bound : 3.0
registered upper bound : 340.0
```

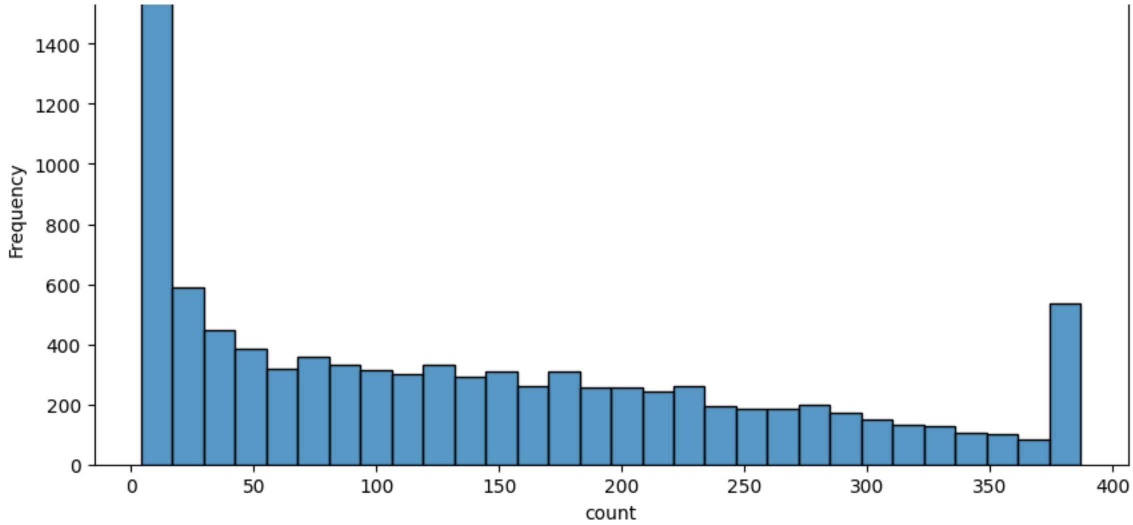
Bar Plot of registered after Clipping




```
count lower bound : 4.0
count upper bound : 386.85000000000036
```

Bar Plot of count after Clipping





Frequency
1400
1200
1000
800
600
400
200
0
0
50
100
150
200
250
300
350
400
count

Insights:

1) Temperature (temp):

- Lower Bound: 8.2
- Upper Bound: 31.86

2) Feels Like Temperature (atemp):

- Lower Bound: 9.85
- Upper Bound: 35.61

3) Humidity:

- Lower Bound: 33.0
- Upper Bound: 93.0

4) Windspeed:

- Lower Bound: 0.0
- Upper Bound: 26.00

5) Casual Users:

- Lower Bound: 0.0
- Upper Bound: 82.0

6) Registered Users:

- Lower Bound: 3.0
- Upper Bound: 340.0

7) Total Count of Bike Rentals:

- Lower Bound: 4.0
- Upper Bound: 386.85

Key Takeaways:

- Clipping: By clipping the data to the 5th and 95th percentiles, extreme outliers are removed, resulting in a more robust dataset.
- Temperature and Feels Like Temperature: Both have similar ranges, indicating consistency in temperature measurements.
- Humidity and Windspeed: The ranges are reasonable, ensuring that extreme weather conditions are excluded.
- User Counts: The ranges for casual and registered users are adjusted to exclude extreme values, providing a more accurate representation of typical usage.

2) Try establishing a Relationship between the Dependent and Independent Variables.

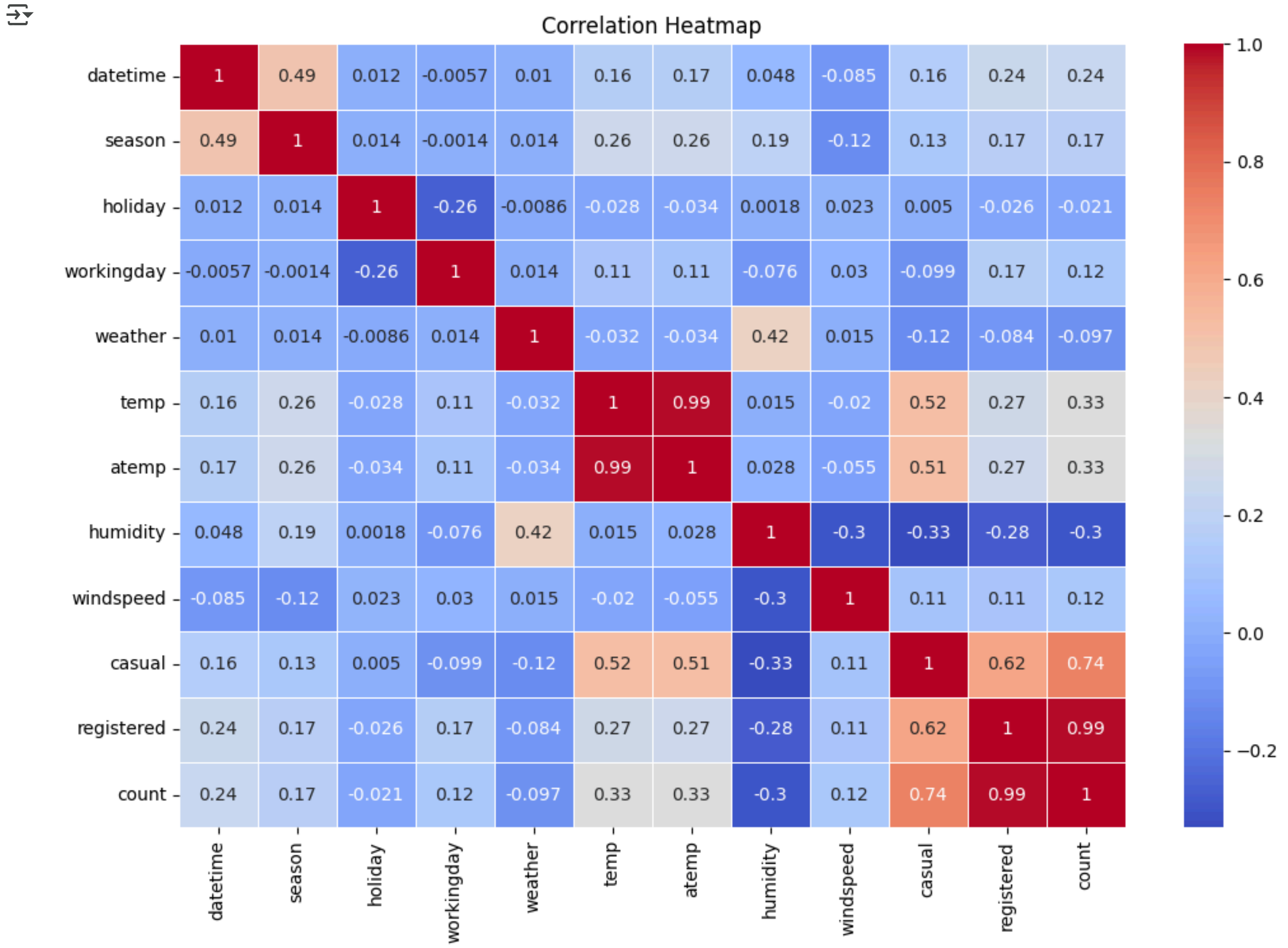
i) Plot a Correlation Heatmap and draw insights.

1) Dependent Variable: count

2) Independent Variable: Rest all the columns except count column.

- These independent variables are used to predict the dependent variable count, which represents the total number of bike rentals.

```
# Plot a Correlation Heatmap
plt.figure(figsize=(12, 8))
corr_matrix = df.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```



corr_matrix

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	c
datetime	1.000000	0.486804	0.011762	-0.005692	0.010030	0.164694	0.166334	0.047986	-0.084638	0.155053	0.243758	0.23
season	0.486804	1.000000	0.014068	-0.001357	0.014253	0.257204	0.262606	0.193775	-0.124598	0.130271	0.168432	0.17
holiday	0.011762	0.014068	1.000000	-0.259111	-0.008569	-0.027760	-0.033904	0.001847	0.023293	0.005036	-0.025708	-0.02
workingday	-0.005692	-0.001357	-0.259111	1.000000	0.014410	0.112995	0.107831	-0.076467	0.029549	-0.099197	0.168065	0.12
weather	0.010030	0.014253	-0.008569	0.014410	1.000000	-0.031536	-0.034417	0.420144	0.014547	-0.121163	-0.084188	-0.09
temp	0.164694	0.257204	-0.027760	0.112995	-0.031536	1.000000	0.986494	0.014628	-0.019651	0.516386	0.267147	0.33
atemp	0.166334	0.262606	-0.033904	0.107831	-0.034417	0.986494	1.000000	0.027937	-0.055269	0.512240	0.265006	0.33
humidity	0.047986	0.193775	0.001847	-0.076467	0.420144	0.014628	0.027937	1.000000	-0.302822	-0.330837	-0.276900	-0.30
windspeed	-0.084638	-0.124598	0.023293	0.029549	0.014547	-0.019651	-0.055269	-0.302822	1.000000	0.109845	0.112844	0.11
casual	0.155053	0.130271	0.005036	-0.099197	-0.121163	0.516386	0.512240	-0.330837	0.109845	1.000000	0.620073	0.73
registered	0.243758	0.168432	-0.025708	0.168065	-0.084188	0.267147	0.265006	-0.276900	0.112844	0.620073	1.000000	0.98
count	0.239823	0.170973	0.020837	0.120117	0.097003	0.333205	0.330567	0.304364	0.118874	0.737062	0.985459	1.00

Next steps: Generate code with `corr_matrix` | View recommended plots | New interactive sheet

```
# Generate insights from the correlation matrix
def generate_insights(corr_matrix):
    weak_negative_correlations = []
    weak_positive_correlations = []
    moderate_positive_correlations = []
    moderate_negative_correlations = []
    strong_positive_correlations = []
    strong_negative_correlations = []

    for col in corr_matrix.columns:
        for row in corr_matrix.index:
            if col != row:
                correlation = corr_matrix.loc[row, col]
```

```
            if -0.3 <= correlation <= 0:
                weak_negative_correlations.append(f"Weak Negative correlation between {row} and {col}: {correlation:4f}")
            elif 0 < correlation <= 0.3:
                weak_positive_correlations.append(f"Weak positive correlation between {row} and {col}: {correlation:4f}")
            elif 0.3 < correlation <= 0.7:
                moderate_positive_correlations.append(f"Moderate positive correlation between {row} and {col}: {correlation:4f}")
            elif -0.7 <= correlation < -0.3:
                moderate_negative_correlations.append(f"Moderate negative correlation between {row} and {col}: {correlation:4f}")
            elif correlation > 0.7:
                strong_positive_correlations.append(f"Strong positive correlation between {row} and {col}: {correlation:4f}")
            elif correlation < -0.7:
                strong_negative_correlations.append(f"Strong negative correlation between {row} and {col}: {correlation:4f}")

    return weak_negative_correlations,weak_positive_correlations, moderate_positive_correlations, moderate_negative_correlations, strong

# Generate and print insights
weak_negative_correlations,weak_positive_correlations, moderate_positive_correlations, moderate_negative_correlations, strong_positive_c

print("🔍 INSIGHTS:\n")

print("Weak Positive Correlations:")
if weak_positive_correlations:
    for insight in weak_positive_correlations:
        print(insight)
else:
    print("There is no weak positive correlation")

print("Weak Negative Correlations:")
if weak_negative_correlations:
    for insight in weak_negative_correlations:
        print(insight)
else:
    print("There is no weak negative correlation")
print("\nModerate Positive Correlations:")
if moderate_positive_correlations:
    for insight in moderate_positive_correlations:
        print(insight)
else:
    print("There is no moderate positive correlation")

print("\nModerate Negative Correlations:")
if moderate_negative_correlations:
    for insight in moderate_negative_correlations:
        print(insight)
else:
    print("There is no moderate negative correlation")

print("\nStrong Positive Correlations:")
if strong_positive_correlations:
    for insight in strong_positive_correlations:
        print(insight)
else:
    print("There is no strong positive correlation")

print("\nStrong Negative Correlations:")
if strong_negative_correlations:
    for insight in strong_negative_correlations:
        print(insight)
else:
    print("There is no strong negative correlation")
```

INSIGHTS:

Weak Positive Correlations:
Weak positive correlation between holiday and datetime: 0.011762
Weak positive correlation between weather and datetime: 0.010030
Weak positive correlation between temp and datetime: 0.164694
Weak positive correlation between atemp and datetime: 0.166334
Weak positive correlation between humidity and datetime: 0.047986
Weak positive correlation between casual and datetime: 0.155053
Weak positive correlation between registered and datetime: 0.243758
Weak positive correlation between count and datetime: 0.239823
Weak positive correlation between holiday and season: 0.014068
Weak positive correlation between weather and season: 0.014253
Weak positive correlation between temp and season: 0.257204
Weak positive correlation between atemp and season: 0.262606
Weak positive correlation between humidity and season: 0.193775
Weak positive correlation between casual and season: 0.130271
Weak positive correlation between registered and season: 0.168432
Weak positive correlation between count and season: 0.170973
Weak positive correlation between datetime and holiday: 0.011762
Weak positive correlation between season and holiday: 0.014068
Weak positive correlation between humidity and holiday: 0.001847
Weak positive correlation between windspeed and holiday: 0.023293
Weak positive correlation between casual and holiday: 0.005036
Weak positive correlation between weather and workingday: 0.014410
Weak positive correlation between temp and workingday: 0.112995
Weak positive correlation between atemp and workingday: 0.107831
Weak positive correlation between windspeed and workingday: 0.029549
Weak positive correlation between registered and workingday: 0.168065
Weak positive correlation between count and workingday: 0.120117
Weak positive correlation between datetime and weather: 0.010030

```
Moderate Positive Correlations:
Moderate positive correlation between season and datetime: 0.486804
Moderate positive correlation between datetime and season: 0.486804
Moderate positive correlation between humidity and weather: 0.420144
Moderate positive correlation between casual and temp: 0.516386
Moderate positive correlation between count and temp: 0.333205
Moderate positive correlation between casual and atemp: 0.512240
Moderate positive correlation between count and atemp: 0.330567
Moderate positive correlation between weather and humidity: 0.420144
Moderate positive correlation between temp and casual: 0.516386
Moderate positive correlation between atemp and casual: 0.512240
Moderate positive correlation between registered and casual: 0.620073
Moderate positive correlation between casual and registered: 0.620073
Moderate positive correlation between temp and count: 0.333205
Moderate positive correlation between atemp and count: 0.330567

Moderate Negative Correlations:
Moderate negative correlation between windspeed and humidity: -0.302822
Moderate negative correlation between casual and humidity: -0.330837
Moderate negative correlation between count and humidity: -0.304364
Moderate negative correlation between humidity and windspeed: -0.302822
Moderate negative correlation between humidity and casual: -0.330837
Moderate negative correlation between humidity and count: -0.304364

Strong Positive Correlations:
Strong positive correlation between atemp and temp: 0.986494
Strong positive correlation between temp and atemp: 0.986494
Strong positive correlation between count and casual: 0.737062
Strong positive correlation between count and registered: 0.985459
Strong positive correlation between casual and count: 0.737062
Strong positive correlation between registered and count: 0.985459

Strong Negative Correlations:
There is no strong negative correlation
```

Insights:

1) Count and Registered: There is a strong positive correlation between count and registered, indicating that the number of registered users significantly influences the total count of bike rentals.

2) Count and Casual: There is also a positive correlation between count and casual, suggesting that casual users contribute to the total count of bike rentals.

3) Temperature (temp) and Feels Like Temperature (atemp): There is a very strong positive correlation between temp and atemp, which is expected as they both measure temperature.

4) Count and Temperature (temp): There is a moderate positive correlation between count and temp, indicating that higher temperatures may lead to an increase in bike rentals.

5) Count and Windspeed: There is a weak positive correlation between count and windspeed, suggesting that higher wind speeds might slightly reduce the number of bike rentals.

Key Takeaways:

- High Correlation: temp and atemp are highly correlated (0.99), indicating redundancy.
- Strong Predictors: registered and casual are strong predictors of count.
- Moderate Predictors: temp, atemp, and humidity show moderate correlations with count.

```
# Comments on the distribution of the variables and relationship between them
comments = """
1. Distribution of Variables:
   - The 'datetime' variable represents timestamps and is not suitable for histogram analysis.
   - The 'season' variable has four distinct categories (1: spring, 2: summer, 3: fall, 4: winter).
   - The 'holiday' and 'workingday' variables are binary (0 or 1).
   - The 'weather' variable has four distinct categories (1 to 4).
   - The 'temp', 'atemp', 'humidity', and 'windspeed' variables are continuous and show varying distributions.
   - The 'casual', 'registered', and 'count' variables represent the number of bike rentals and show right-skewed distributions.

2. Relationship Between Variables:
   - The correlation matrix shows the relationships between the variables.
   - There is a strong positive correlation between 'temp' and 'atemp'.
   - The 'count' variable has a positive correlation with 'temp' and 'atemp', indicating that bike rentals increase with higher temperatu
   - The 'count' variable has a negative correlation with 'humidity' and 'windspeed', indicating that bike rentals decrease with higher h
   - The 'casual' and 'registered' variables have a strong positive correlation with the 'count' variable, as they contribute to the tota
"""

print(comments)
```

```
1. Distribution of Variables:
```

```
   - The 'datetime' variable represents timestamps and is not suitable for histogram analysis.
   - The 'season' variable has four distinct categories (1: spring, 2: summer, 3: fall, 4: winter).
   - The 'holiday' and 'workingday' variables are binary (0 or 1).
   - The 'weather' variable has four distinct categories (1 to 4).
   - The 'temp', 'atemp', 'humidity', and 'windspeed' variables are continuous and show varying distributions.
   - The 'casual', 'registered', and 'count' variables represent the number of bike rentals and show right-skewed distributions.

2. Relationship Between Variables:
   - The correlation matrix shows the relationships between the variables.
   - There is a strong positive correlation between 'temp' and 'atemp'.
   - The 'count' variable has a positive correlation with 'temp' and 'atemp', indicating that bike rentals increase with higher temp
   - The 'count' variable has a negative correlation with 'humidity' and 'windspeed', indicating that bike rentals decrease with hig
   - The 'casual' and 'registered' variables have a strong positive correlation with the 'count' variable, as they contribute to the
```

ii) Remove the highly correlated variables, if any

```
# Remove highly correlated variables (threshold > 0.8)
threshold = 0.8
highly_correlated_vars = set()
for i in range(len(corr_matrix.columns)):
    for j in range(i):
        if abs(corr_matrix.iloc[i, j]) > threshold:
            colname = corr_matrix.columns[i]
            highly_correlated_vars.add(colname)

# Drop highly correlated variables from the dataset
df_reduced = df.drop(columns=highly_correlated_vars)

# Display the remaining columns
print("Remaining columns after removing highly correlated variables:")
print(df_reduced.columns)
```

```
Remaining columns after removing highly correlated variables:
Index(['datetime', 'season', 'holiday', 'workingday', 'weather', 'temp',
       'humidity', 'windspeed', 'casual', 'registered'],
      dtype='object')
```

Insights:

- After removing the highly correlated variables(i.e. temp and atemp) , the remaining columns in your dataset are:

1) datetime: The date and time of the bike rental.

2) season: The season in which the bike rental occurred.

3) holiday: Whether the day is a holiday or not.

4) workingday: Whether the day is a working day.

5) weather: The weather condition.

6) temp: The temperature in Celsius.

7) humidity: The humidity level.

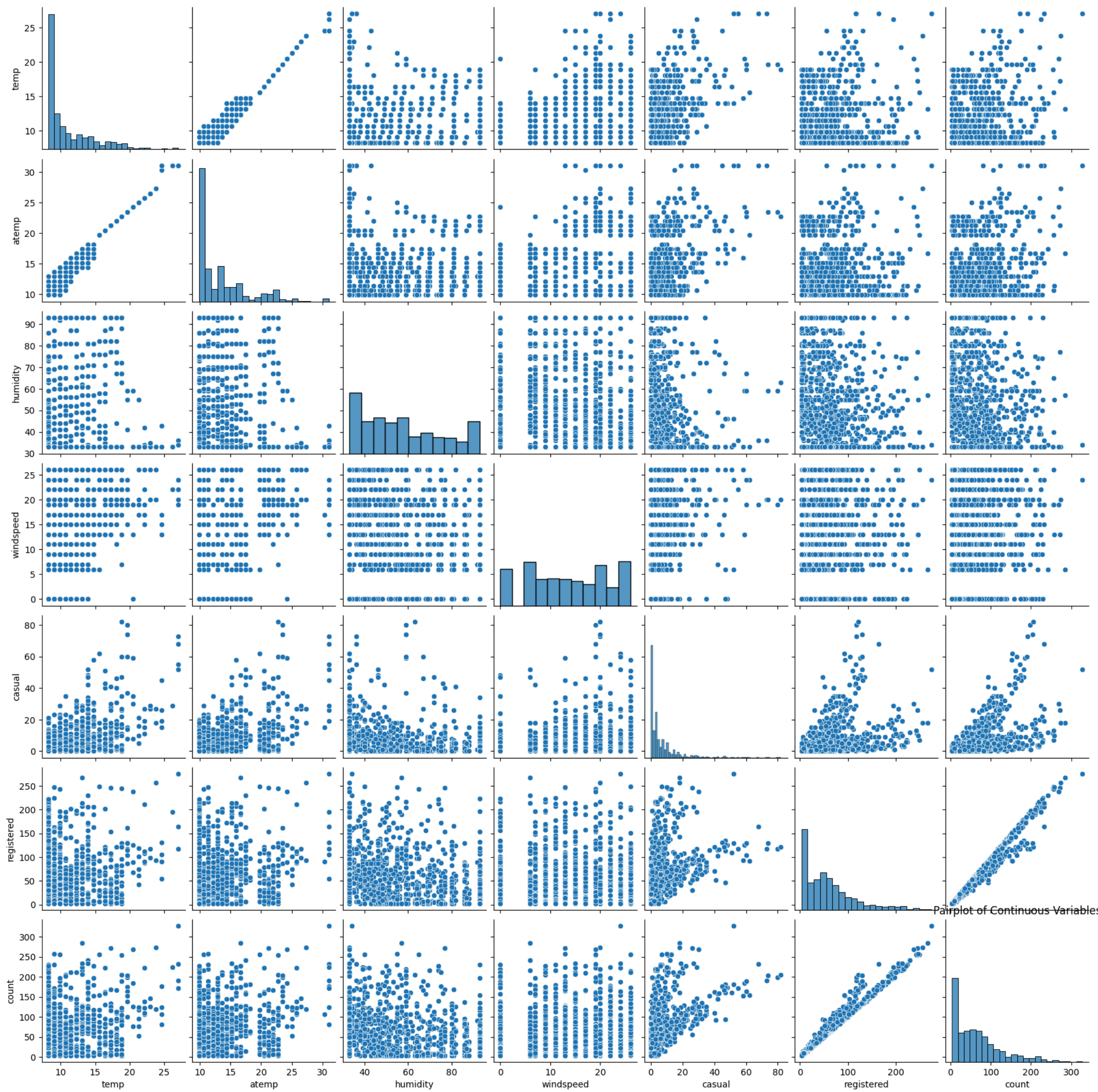
8) windspeed: The wind speed.

9) casual: The count of casual users.

10) registered: The count of registered users.

- These columns will be used as independent variables to predict the dependent variable count.

```
# Plot the pairplot for first 1000 rows for clear visibility
df_1000=df.head(1000)
sns.pairplot(df_1000[continuous_columns])
plt.title('Pairplot of Continuous Variables')
plt.show()
```


Pairplot of Continuous Variables
temp
atemp
humidity
windspeed
casual
registered
count

3)Check if there any significant difference between the no. of bike rides on Weekdays and Weekends?

- Way1: using day_of_week column which is extracted using datetime column.

```
# Extract day of the week from 'datetime' column
df['day_of_week'] = df['datetime'].dt.day_name()
```

+ Code + Text

```
# Separate the data into weekdays and weekends from day_of_week
weekdays = df[df['day_of_week'].isin(['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday'])]['count']
weekends = df[df['day_of_week'].isin(['Saturday', 'Sunday'])]['count']
```

```
#a)Formulate Null Hypothesis (H0) and Alternate Hypothesis (H1)

#H0: There is no significance difference between the no. of bike rides on Weekdays and Weekends.
#H1: There is significance difference between the no. of bike rides on Weekdays and Weekends.
```

```
#b. Select an appropriate test - 2- Sample Independent T-test

# Perform 2-Sample Independent T-test
t_stat, p_value = stats.ttest_ind(weekdays, weekends)

#d) Print test statistics and p-value
print(f"T-statistic: {t_stat}")
print(f"P-value: {p_value}")
```

```
T-statistic: 11.311456806771902
P-value: 1.7880911847932131e-29
```

```
#c) set a significance level
alpha=0.05 #5% as recommended

#e) Decide whether to accept or reject the Null Hypothesis
if p_value < 0.05:
    print("Reject the Null Hypothesis (H0). There is a significant difference between the number of bike rides on weekdays and weekends.
else:
    print("Fail to reject the Null Hypothesis (H0). There is no significant difference between the number of bike rides on weekdays and
```

```
Reject the Null Hypothesis (H0). There is a significant difference between the number of bike rides on weekdays and weekends.
```

- Way 2 ; using working days column

```
# Separate the data into weekdays and weekends using the 'workingday' column
weekdays_1 = df[df['workingday'] == 1]['count']
weekends_0 = df[df['workingday'] == 0]['count']


# Perform 2-Sample Independent T-test
t_stat, p_value = stats.ttest_ind(weekdays_1, weekends_0)

# Print test statistics and p-value
print(f"T-statistic: {t_stat}")
print(f"P-value: {p_value}")
# Decide whether to accept or reject the Null Hypothesis
if p_value < 0.05:
    print("Reject the Null Hypothesis (H0). There is a significant difference between the number of bike rides on weekdays and weekends.
else:
    print("Fail to reject the Null Hypothesis (H0). There is no significant difference between the number of bike rides on weekdays and
```

```
T-statistic: 11.706987666880087
P-value: 1.9429461229901098e-31
Reject the Null Hypothesis (H0). There is a significant difference between the number of bike rides on weekdays and weekends.
```

Draw Inferences & Conclusions:

- If the p-value is less than or equal to 0.05, reject the Null Hypothesis. This means there is a significant difference in the number of bike rides between weekdays and weekends.

- If the p-value is greater than 0.05, fail to reject the Null Hypothesis. This means there is no significant difference in the number of bike rides between weekdays and weekends.

Recommendations:

- If significant difference: Consider adjusting bike availability or pricing strategies based on the day of the week to optimize usage.
- If no significant difference: No changes needed based on the day of the week. Focus on other factors to optimize bike usage.

Final Recommendation:

- As p-value is less than 0.05 hence,This means there is a significant difference in the number of bike rides between weekdays and weekends.
- Consider adjusting bike availability or pricing strategies based on the day of the week to optimize usage.

4)Check if the demand of bicycles on rent is the same for different Weather conditions?

```
# a)Formulate Null Hypothesis (H0) and Alternate Hypothesis (H1)

# H0: The demand for bicycles on rent is the same for different weather conditions.
# H1: The demand for bicycles on rent is not the same for different weather conditions.
```

```
#b)  Select an appropriate test -One-way ANOVA test; Count(N) and Weather (C)
```

- Check assumptions of the test:

i) Normality:

- Use Histogram, Q-Q Plot, Skewness & Kurtosis
- Shapiro-Wilk’s test ii. Equality Variance Hint:

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

#1: Clear, Few clouds, 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 Pellets + Thunderstorm + Mist, Snow + Fog
```

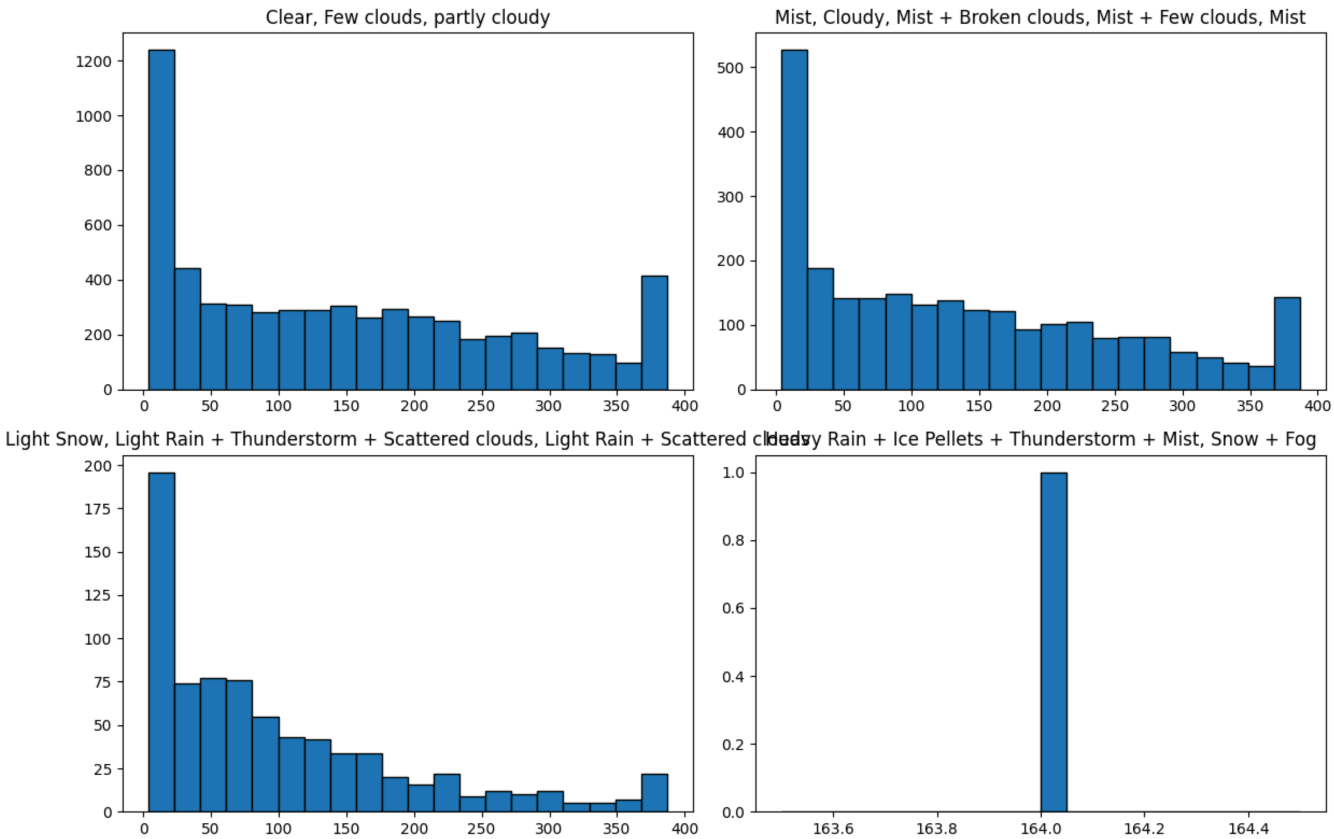
weather	count
1	6059
2	2533
3	771
4	1

```
# Filter the data for each weather condition
Clear_Few_clouds_partly_cloudy = df[df['weather'] == 1]['count']
Mist_Cloudy_Mist_Broken_clouds_Mist_Few_clouds_Mist = df[df['weather'] == 2]['count']
Light_Snow_Light_Rain_Thunderstorm_Scattered_clouds_Light_Rain_Scatter = df[df['weather'] == 3]['count']
Heavy_Rain_Ice_Pellets_Thunderstorm_Mist_Snow_Fog = df[df['weather'] == 4]['count']
```

```
# Display the number of bike rides for each weather condition
print(f"Number of bike rides in Clear, Few clouds, partly cloudy weather: {Clear_Few_clouds_partly_cloudy.sum()}")
print(f"Number of bike rides in Mist, Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist weather: {Mist_Cloudy_Mist_Broken_clouds_Mis
print(f"Number of bike rides in Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds weather: {Light_
print(f"Number of bike rides in Heavy Rain + Ice Pellets + Thunderstorm + Mist, Snow + Fog weather: {Heavy_Rain_Ice_Pellets_Thunderstorm
```

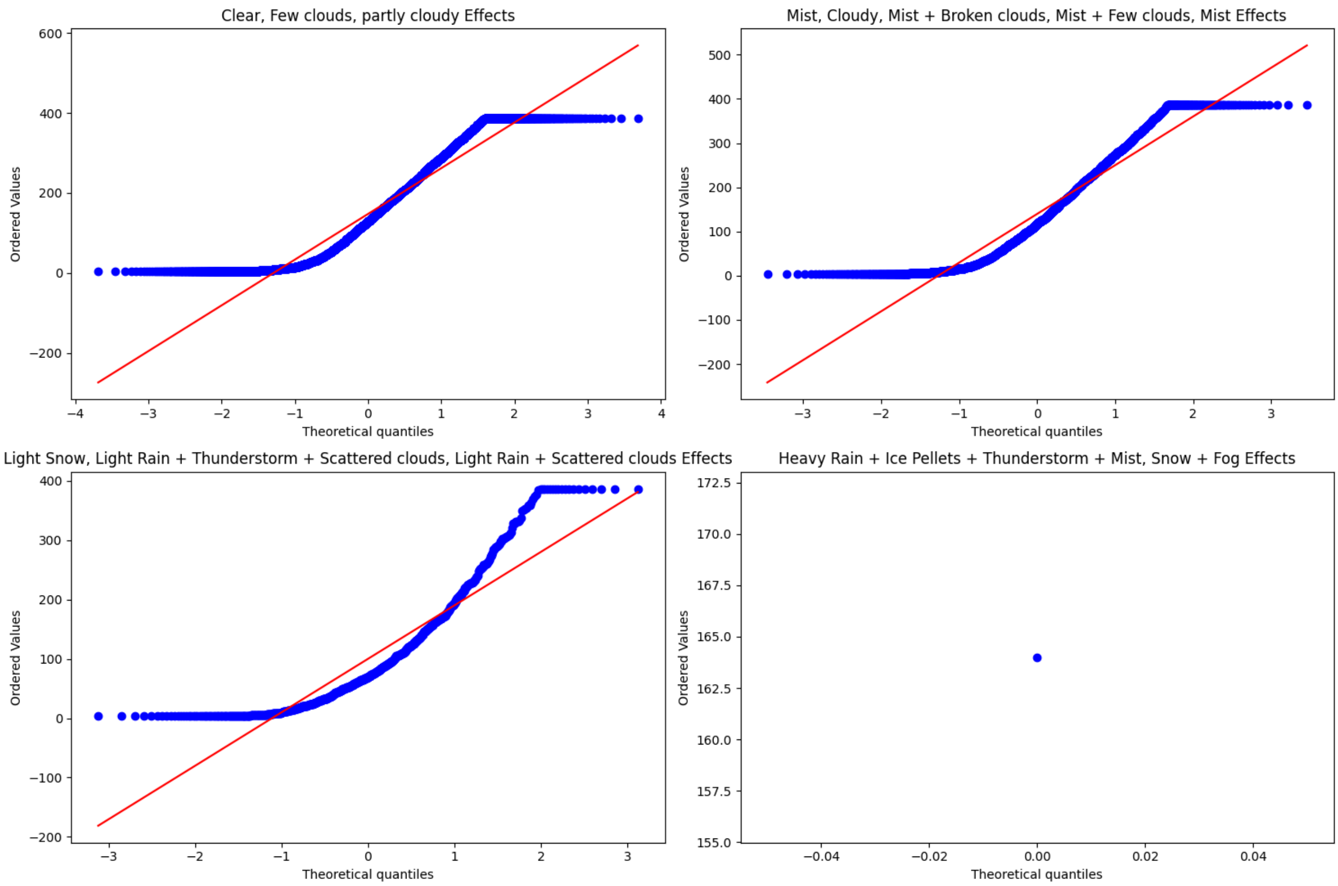
```
Number of bike rides in Clear, Few clouds, partly cloudy weather: 893760.05
Number of bike rides in Mist, Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist weather: 353831.30000000005
Number of bike rides in Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds weather: 77158.3
Number of bike rides in Heavy Rain + Ice Pellets + Thunderstorm + Mist, Snow + Fog weather: 164.0
```

```
# Check assumptions of the test

# Normality
# 1. Histogram
plt.figure(figsize=(12, 8))
plt.subplot(2, 2, 1)
plt.hist(Clear_Few_clouds_partly_cloudy, bins=20, edgecolor='k')
plt.title('Clear, Few clouds, partly cloudy')

plt.subplot(2, 2, 2)
plt.hist(Mist_Cloudy_Mist_Broken_clouds_Mist_Few_clouds_Mist, bins=20, edgecolor='k')
plt.title('Mist, Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist')

plt.subplot(2, 2, 3)
plt.hist(Light_Snow_Light_Rain_Thunderstorm_Scattered_clouds_Light_Rain_Scatter, bins=20, edgecolor='k')
plt.title('Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds')

plt.subplot(2, 2, 4)
plt.hist(Heavy_Rain_Ice_Pellets_Thunderstorm_Mist_Snow_Fog, bins=20, edgecolor='k')
plt.title('Heavy Rain + Ice Pellets + Thunderstorm + Mist, Snow + Fog')

plt.tight_layout()
plt.show()
```




```
# 2. Q-Q Plot
fig, axs = plt.subplots(2, 2, figsize=(15, 10))
effects = [
    'Clear, Few clouds, partly cloudy',
    'Mist, Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist',
    'Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds',
    'Heavy Rain + Ice Pellets + Thunderstorm + Mist, Snow + Fog'
]

data = [
    Clear_Few_clouds_partly_cloudy,
    Mist_Cloudy_Mist_Broken_clouds_Mist_Few_clouds_Mist,
```

```
    Light_Snow_Light_Rain_Thunderstorm_Scattered_clouds_Light_Rain_Scatter,
    Heavy_Rain_Ice_Pellets_Thunderstorm_Mist_Snow_Fog
]

for i in range(len(data)):
    row = i // 2
    col = i % 2
    stats.probplot(data[i], dist="norm", plot=axs[row][col])
    axs[row][col].set_title(f'{effects[i]} Effects')

plt.tight_layout()
plt.show()
```




```
# Calculate skewness and kurtosis for each weather condition
conditions = {
    'Clear, Few clouds, partly cloudy': Clear_Few_clouds_partly_cloudy,
    'Mist, Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist': Mist_Cloudy_Mist_Broken_clouds_Mist_Few_clouds_Mist,
    'Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds': Light_Snow_Light_Rain_Thunderstorm_Scatte
    'Heavy Rain + Ice Pellets + Thunderstorm + Mist, Snow + Fog': Heavy_Rain_Ice_Pellets_Thunderstorm_Mist_Snow_Fog
}

for condition, data in conditions.items():
    skewness_value = skew(data)
    kurtosis_value = kurtosis(data)
    print(f"Weather: {condition}")

    # Determine the type of distribution based on skewness value
    if skewness_value > 0:
        distribution_type = "Right Skewed"
    elif skewness_value < 0:
        distribution_type = "Left Skewed"
    elif skewness_value == 0:
        distribution_type = "Normally Distributed"
    else:
        distribution_type = "None of the above"

    print(f"Skewness Value: {skewness_value}")
    print(f"Distribution Type: {distribution_type}\n")

    # Determine the type of distribution based on kurtosis value
```

```
if kurtosis_value == 3:
    distribution_type = "Mesokurtic"
elif kurtosis_value > 3:
    distribution_type = "Leptokurtic"
else:
    distribution_type = "Platykurtic"

print(f"Kurtosis Value: {kurtosis_value}")
print(f"Distribution Type: {distribution_type}\n")
```

```
Weather: Clear, Few clouds, partly cloudy
Skewness Value: 0.5140460403843464
Distribution Type: Right Skewed

Kurtosis Value: -0.9034868640899352
Distribution Type: Platykurtic

Weather: Mist, Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
Skewness Value: 0.6150531093080647
Distribution Type: Right Skewed

Kurtosis Value: -0.7221685696541469
Distribution Type: Platykurtic

Weather: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
Skewness Value: 1.2617398125925274
Distribution Type: Right Skewed

Kurtosis Value: 0.9596133177411619
Distribution Type: Platykurtic

Weather: Heavy Rain + Ice Pellets + Thunderstorm + Mist, Snow + Fog
Skewness Value: nan
Distribution Type: None of the above

Kurtosis Value: nan
Distribution Type: Platykurtic
```

Insight:

1) Weather: Clear, Few clouds, partly cloudy

- Skewness Value: 0.5140 (Right Skewed)
- Kurtosis Value: -0.9035 (Platykurtic)

The data for clear and partly cloudy weather is right-skewed, indicating a longer tail on the right side. The platykurtic kurtosis value suggests that the data has lighter tails and a flatter peak compared to a normal distribution.

2) Weather: Mist, Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist

- Skewness Value: 0.6151 (Right Skewed)
- Kurtosis Value: -0.7222 (Platykurtic)

The data for misty and cloudy weather is right-skewed, indicating a longer tail on the right side. The platykurtic kurtosis value suggests lighter tails and a flatter peak.

3) Weather: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds

- Skewness Value: 1.2617 (Right Skewed)
- Kurtosis Value: 0.9596 (Platykurtic)

The data for light snow and rainy weather with thunderstorms is right-skewed, indicating a longer tail on the right side. The platykurtic kurtosis value suggests lighter tails and a flatter peak.

4) Weather: Heavy Rain + Ice Pellets + Thunderstorm + Mist, Snow + Fog

- Skewness Value: NaN (Not available)
- Kurtosis Value: NaN (Not available)

The data for heavy rain, ice pellets, thunderstorms, mist, snow, and fog is not available for skewness and kurtosis analysis, possibly due to insufficient data points.

Summary

- Across most weather conditions, the data shows a right-skewed distribution, indicating that higher demand values are less frequent but present. The platykurtic nature of the data suggests that the distributions have lighter tails and flatter peaks compared to a normal distribution.

```
# 3. Shapiro-Wilk's test
data = [
    Clear_Few_clouds_partly_cloudy,
    Mist_Cloudy_Mist_Broken_clouds_Mist_Few_clouds_Mist,
    Light_Snow_Light_Rain_Thunderstorm_Scattered_clouds_Light_Rain_Scatter,
    Heavy_Rain_Ice_Pellets_Thunderstorm_Mist_Snow_Fog
]
p_values = []
for condition in data:
    if len(condition) >= 3:
        _, p_value = shapiro(condition)
        p_values.append(p_value)
    else:
        p_values.append(float('nan'))

print(f"Normality p-values: a={p_values[0]}, b={p_values[1]}, c={p_values[2]}, d={p_values[3]}")

if all(p_value > 0.05 for p_value in p_values):
    print("The data is normally distributed.")
else:
  print("The data is not normally distributed.")
```

```
Normality p-values: a=7.153720675041487e-50, b=7.210863280598551e-36, c=5.818817243737285e-26, d=nan
The data is not normally distributed.
```

```
#ii) Equality of Variance using Levene's test
levene_statistic, levene_p_value = levene(
    Clear_Few_clouds_partly_cloudy,
    Mist_Cloudy_Mist_Broken_clouds_Mist_Few_clouds_Mist,
    Light_Snow_Light_Rain_Thunderstorm_Scattered_clouds_Light_Rain_Scatter,
    Heavy_Rain_Ice_Pellets_Thunderstorm_Mist_Snow_Fog
)

print(f"Levene's test p-value: {levene_p_value}")

if levene_p_value < 0.05:
    print("The variances are not equal.")
else:
    print("The variances are equal.")
```

```
Levene's test p-value: 1.979310876597833e-27
The variances are not equal.
```

Insights:

- From above Shapiro-Wilk's test it is concluded the data is not norn=mally distributed.
- From Levene's test it is also concluded that variance are not equal.
- Hence, will perform Alternate One-Way Anova Test : Kruskal Wallis Test.

```
# Perform Kruskal-Wallis test
stat, anova_p_value = kruskal(
    Clear_Few_clouds_partly_cloudy,
    Mist_Cloudy_Mist_Broken_clouds_Mist_Few_clouds_Mist,
    Light_Snow_Light_Rain_Thunderstorm_Scattered_clouds_Light_Rain_Scatter,
    Heavy_Rain_Ice_Pellets_Thunderstorm_Mist_Snow_Fog
)

# Print the test statistic and p-value
print("Test statistic:", stat)
print("P-value:", anova_p_value)
```

```
Test statistic: 103.70589947326987
P-value: 2.480393865912849e-22
```

```
#e) # Decide whether to accept or reject the Null Hypothesis
alpha = 0.05

if anova_p_value < alpha:
    print("Reject the Null Hypothesis (H0). The demand for bicycles on rent is not the same for different weather conditions.")
else:
    print("Fail to reject the Null Hypothesis (H0). The demand for bicycles on rent is the same for different weather conditions.")

# Draw inferences & conclusions from the analysis and provide recommendations
if anova_p_value < alpha:
    print("Inference: The analysis suggests that there is a significant difference in the demand for bicycles on rent across different w
    print("Recommendation: Consider adjusting bike availability or pricing strategies based on weather conditions to optimize usage.")
```

```
else:
    print("Inference: The analysis suggests that there is no significant difference in the demand for bicycles on rent across different
    print("Recommendation: No changes needed based on weather conditions. Focus on other factors to optimize bike usage.")
```

```
Reject the Null Hypothesis (H0). The demand for bicycles on rent is not the same for different weather conditions.
Inference: The analysis suggests that there is a significant difference in the demand for bicycles on rent across different weather
Recommendation: Consider adjusting bike availability or pricing strategies based on weather conditions to optimize usage.
```

5) Check if the demand of bicycles on rent is the same for different Seasons?

```
# a)Formulate Null Hypothesis (H0) and Alternate Hypothesis (H1)

# H0: The demand for bicycles on rent is the same for different seasons .
# H1: The demand for bicycles on rent is not the same for different seasons.


#b)  Select an appropriate test -One-way ANOVA test; Count(N) and Seasons(C)

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

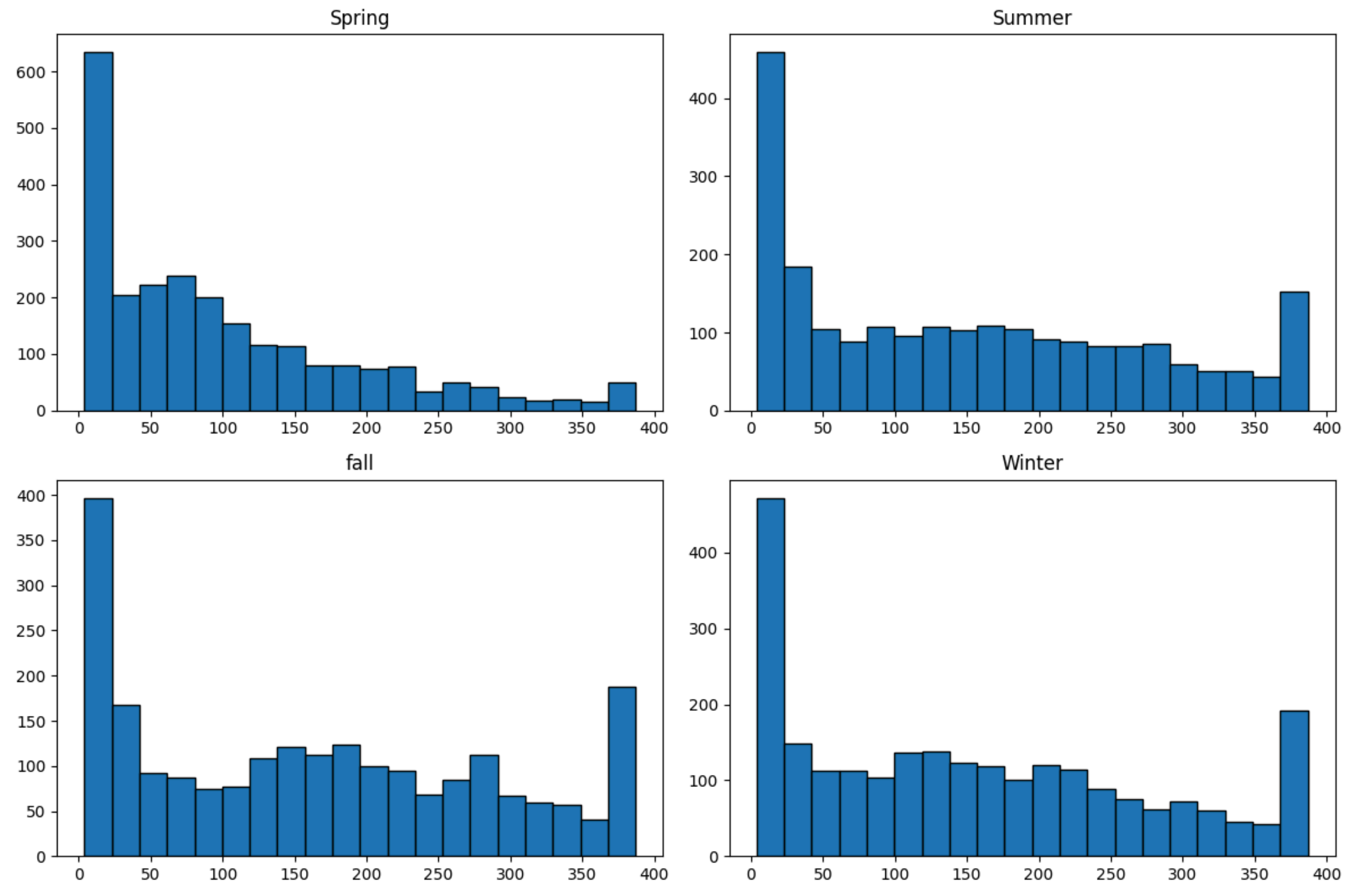
season	count
1	2448
4	2437
2	2249
3	2230

dtype: int64

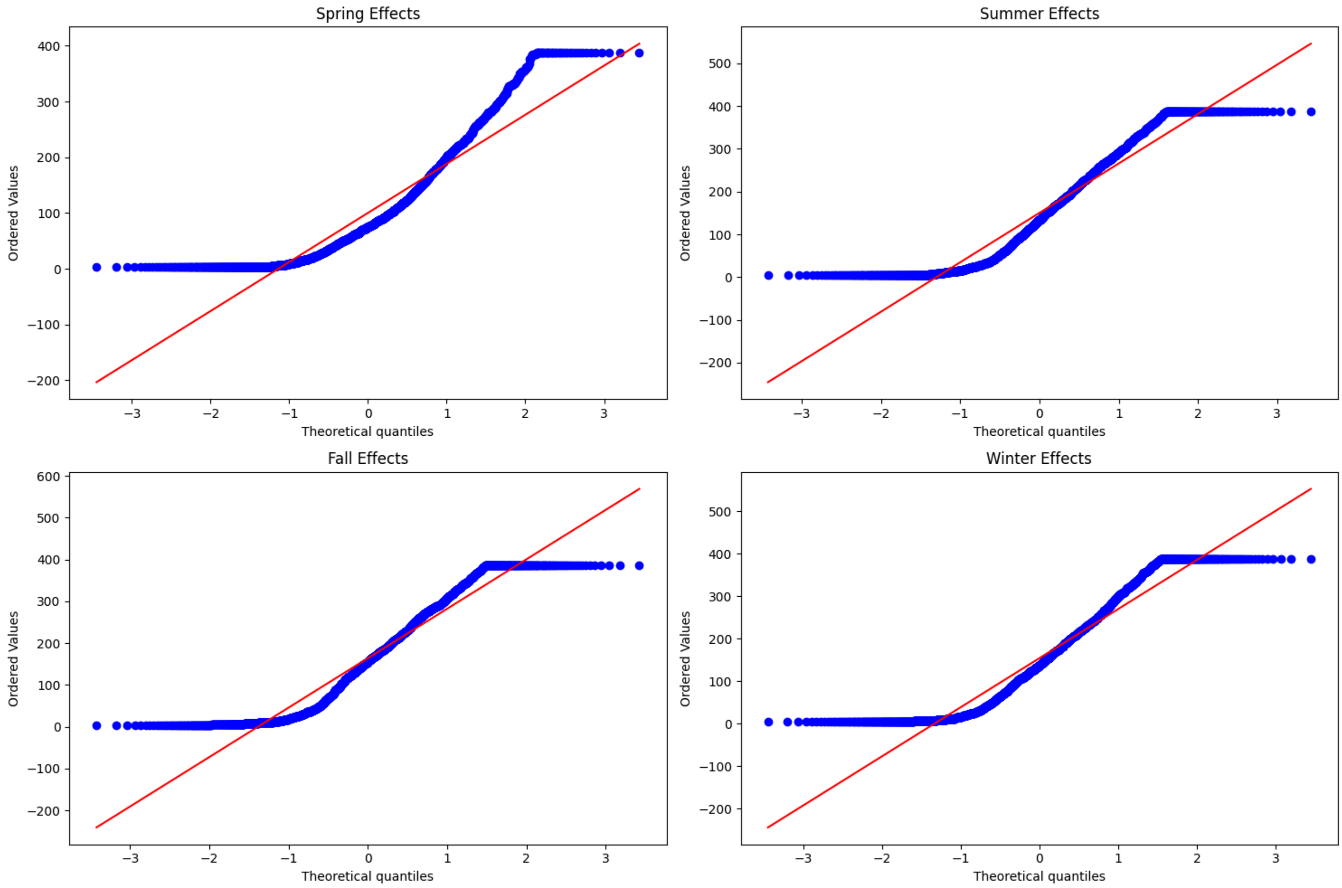
```
#1: spring, 2: summer, 3: fall, 4: winter
# Filter the data for each seasons
Spring = df[df['season'] == 1]['count']
Summer = df[df['season'] == 2]['count']
Fall = df[df['season'] == 3]['count']
Winter = df[df['season'] == 4]['count']


# Display the number of bike rides for each weather condition
print(f"Number of bike rides in Spring: {Spring.sum()}")
print(f"Number of bike rides in Summer: {Summer.sum()}")
print(f"Number of bike rides in Fall: {Fall.sum()}")
print(f"Number of bike rides in Winter: {Winter.sum()}")
```

```
Number of bike rides in Spring: 245194.0
Number of bike rides in Summer: 338008.55000000005
Number of bike rides in Fall: 365761.75000000006
Number of bike rides in Winter: 375949.35000000003
```

```
# Check assumptions of the test

# Normality
# 1. Histogram
plt.figure(figsize=(12, 8))
plt.subplot(2, 2, 1)
plt.hist(Spring, bins=20, edgecolor='k')
plt.title('Spring')

plt.subplot(2, 2, 2)
plt.hist(Summer, bins=20, edgecolor='k')
plt.title('Summer')

plt.subplot(2, 2, 3)
plt.hist(Fall, bins=20, edgecolor='k')
plt.title('fall')

plt.subplot(2, 2, 4)
plt.hist(Winter, bins=20, edgecolor='k')
plt.title('Winter')

plt.tight_layout()
plt.show()
```




```
# 2. Q-Q Plot
fig, axs = plt.subplots(2, 2, figsize=(15, 10))
effects = [
    'Spring','Summer','Fall','Winter'
]

data = [
    Spring,
    Summer,
    Fall,
    Winter
]

for i in range(len(data)):
    row = i // 2
    col = i % 2
    stats.probplot(data[i], dist="norm", plot=axs[row][col])
    axs[row][col].set_title(f'{effects[i]} Effects')

plt.tight_layout()
plt.show()
```




```
# Skewness and Kurtosis
conditions = {
    'Spring': Spring,
    'Summer': Summer,
    'Fall': Fall,
    'Winter': Winter
}

for condition, data in conditions.items():
    skewness_value = skew(data)
    kurtosis_value = kurtosis(data)
    print(f"Season: {condition}")

    # Determine the type of distribution based on skewness value
    if skewness_value > 0:
        distribution_type = "Right Skewed"
    elif skewness_value < 0:
        distribution_type = "Left Skewed"
    elif skewness_value == 0:
        distribution_type = "Normally Distributed"
    else:
        distribution_type = "None of the above"

    print(f"Skewness Value: {skewness_value}")
    print(f"Distribution Type: {distribution_type}\n")

    # Determine the type of distribution based on kurtosis value
    if kurtosis_value == 3:
        distribution_type = "Mesokurtic"
    elif kurtosis_value > 3:
        distribution_type = "Leptokurtic"
    else:
        distribution_type = "Platykurtic"

    print(f"Kurtosis Value: {kurtosis_value}")
    print(f"Distribution Type: {distribution_type}\n")
```

```
Season: Spring
Skewness Value: 1.158767369707755
Distribution Type: Right Skewed
```

```
Kurtosis Value: 0.7539553318570027
Distribution Type: Platykurtic

Season: Summer
Skewness Value: 0.4597486522208333
Distribution Type: Right Skewed

Kurtosis Value: -0.990984889090408
Distribution Type: Platykurtic

Season: Fall
Skewness Value: 0.3068921208989542
Distribution Type: Right Skewed

Kurtosis Value: -1.116778863354376
Distribution Type: Platykurtic

Season: Winter
Skewness Value: 0.4549545626330315
Distribution Type: Right Skewed

Kurtosis Value: -0.9399714032476618
Distribution Type: Platykurtic
```

Insight:

1) Season: Spring

- Skewness Value: 1.1588 (Right Skewed)
- Kurtosis Value: 0.7540 (Platykurtic)

The data for Spring is right-skewed, indicating a longer tail on the right side. The platykurtic kurtosis value suggests that the data has lighter tails and a flatter peak compared to a normal distribution.

2) Season: Summer

- Skewness Value: 0.4597 (Right Skewed)
- Kurtosis Value: -0.9910 (Platykurtic)

The data for Summer is slightly right-skewed, indicating a slight asymmetry with a longer tail on the right. The platykurtic kurtosis value indicates lighter tails and a flatter peak.

3) Season: Fall

- Skewness Value: 0.3069 (Right Skewed)
- Kurtosis Value: -1.1168 (Platykurtic)

The data for Fall is also slightly right-skewed, with a longer tail on the right. The platykurtic kurtosis value suggests lighter tails and a flatter peak.

4) Season: Winter

- Skewness Value: 0.4550 (Right Skewed)
- Kurtosis Value: -0.9400 (Platykurtic)

The data for Winter is slightly right-skewed, indicating a longer tail on the right. The platykurtic kurtosis value suggests lighter tails and a flatter peak.

Summary

- Across all seasons, the data shows a right-skewed distribution, indicating that higher demand values are less frequent but present. The platykurtic nature of the data across all seasons suggests that the distributions have lighter tails and flatter peaks compared to a normal distribution.

```
# 3. Shapiro-Wilk’s test
data = [
    Spring,
    Summer,
    Fall,
    Winter
]

p_values = []
for condition in data:
    if len(condition) >= 3:
        _, p_value = shapiro(condition)
        p_values.append(p_value)
    else:
```

```
        p_values.append(float('nan'))

print(f"Normality p-values: a={p_values[0]}, b={p_values[1]}, c={p_values[2]}, d={p_values[3]}")

if all(p_value > 0.05 for p_value in p_values):
    print("The data is normally distributed.")
else:
  print("The data is not normally distributed.")
```

```
Normality p-values: a=1.1580238815809688e-40, b=1.2143743148502336e-33, c=8.885223348001803e-32, d=9.89248636410002e-34
The data is not normally distributed.
```

```
#ii) Equality of Variance using Levene's test
levene_statistic, levene_p_value = levene(
    Spring,
    Summer,
    Fall,
    Winter
)

print(f"Levene's test p-value: {levene_p_value}")

if levene_p_value < 0.05:
    print("The variances are not equal.")
else:
    print("The variances are equal.")
```

```
Levene's test p-value: 5.214587709652295e-90
The variances are not equal.
```

Insights:

- From above Shapiro-Wilk's test it is concluded the data is not norn=mally distributed.
- From Levene's test it is also concluded that variance are not equal.
- Hence, will perform Alternate One-Way Anova Test : Kruskal Wallis Test.

```
# Perform Kruskal-Wallis test
stat, anova_p_value = kruskal(
    Spring,
    Summer,
    Fall,
    Winter
)

# Print the test statistic and p-value
print("Test statistic:", stat)
print("P-value:", anova_p_value)
```

```
Test statistic: 398.9125399706125
P-value: 3.8080803362077674e-86
```

```
#e) # Decide whether to accept or reject the Null Hypothesis
alpha = 0.05

if anova_p_value < alpha:
    print("Reject the Null Hypothesis (H0). The demand for bicycles on rent is not the same for different seasons.")
else:
    print("Fail to reject the Null Hypothesis (H0). The demand for bicycles on rent is the same for different seasons.")

# Draw inferences & conclusions from the analysis and provide recommendations
if anova_p_value < alpha:
    print("Inference: The analysis suggests that there is a significant difference in the demand for bicycles on rent across different s
    print("Recommendation: Consider adjusting bike availability or pricing strategies based on seasons to optimize usage.")
else:
    print("Inference: The analysis suggests that there is no significant difference in the demand for bicycles on rent across different
    print("Recommendation: No changes needed based on seasons. Focus on other factors to optimize bike usage.")
```

```
Reject the Null Hypothesis (H0). The demand for bicycles on rent is not the same for different seasons.
Inference: The analysis suggests that there is a significant difference in the demand for bicycles on rent across different seasons
Recommendation: Consider adjusting bike availability or pricing strategies based on seasons to optimize usage.
```

6) Check if the Weather conditions are significantly different during different Seasons

```
#a. Formulate Null Hypothesis (H0) and Alternate Hypothesis (H1)

#H0: Weather conditions are not significantly different during different Seasons.
#H1: Weather conditions are significantly different during different Seasons


#b) Select an appropriate test --- Chi-square test;Weather(C) and Seasons(N)


# Create a Contingency Table against 'Weather' & 'Season' columns
contingency_table = pd.crosstab(df['weather'], df['season'])

# Perform Chi-square test
chi2_stat, p_value, dof, expected = chi2_contingency(contingency_table)

contingency_table
```

season weather	1	2	3	4
1	1583	1436	1557	1483
2	680	610	500	743
3	184	203	173	211
4	1	0	0	0

Insights from Seasonal and Weather Data

1) Seasonal Demand for Bicycle Rentals

a) Season 1 (Spring):

- Weather 1 (Clear, Few clouds, partly cloudy): 1583 rentals
- Weather 2 (Mist, Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist): 680 rentals
- Weather 3 (Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds): 184 rentals
- Weather 4 (Heavy Rain + Ice Pellets + Thunderstorm + Mist, Snow + Fog): 1 rental

b) Season 2 (Summer):

- Weather 1 (Clear, Few clouds, partly cloudy): 1436 rentals
- Weather 2 (Mist, Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist): 610 rentals
- Weather 3 (Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds): 203 rentals
- Weather 4 (Heavy Rain + Ice Pellets + Thunderstorm + Mist, Snow + Fog): 0 rentals

c) Season 3 (Fall):

- Weather 1 (Clear, Few clouds, partly cloudy): 1557 rentals
- Weather 2 (Mist, Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist): 500 rentals
- Weather 3 (Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds): 173 rentals
- Weather 4 (Heavy Rain + Ice Pellets + Thunderstorm + Mist, Snow + Fog): 0 rentals

d) Season 4 (Winter):

- Weather 1 (Clear, Few clouds, partly cloudy): 1483 rentals
- Weather 2 (Mist, Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist): 743 rentals
- Weather 3 (Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds): 211 rentals
- Weather 4 (Heavy Rain + Ice Pellets + Thunderstorm + Mist, Snow + Fog): 0 rentals

Summary

- Clear, Few clouds, partly cloudy weather consistently shows the highest demand for bicycle rentals across all seasons.
- Mist, Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist weather shows moderate demand, with the highest in Winter (Season 4).
- Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds weather shows lower demand, with the highest in Winter (Season 4).
- Heavy Rain + Ice Pellets + Thunderstorm + Mist, Snow + Fog weather shows very low to no demand, with only 1 rental in Spring (Season 1).

```
# Print the test statistics and p-value
print(f"Chi-square test statistic: {chi2_stat}")
print(f"P-value: {p_value}")
```

```
Chi-square test statistic: 50.87276261778956
P-value: 7.37899576712981e-08
```

```python
# Set significance level
alpha = 0.05

# Decide whether to accept or reject the Null Hypothesis
if p_value < alpha:
    print("Reject the Null Hypothesis (H0). Weather conditions are significantly different during different seasons.")
else:
    print("Fail to reject the Null Hypothesis (H0). Weather conditions are not significantly different during different seasons.")

# Draw inferences & conclusions from the analysis and provide recommendations
if p_value < alpha:
    print("Inference: The analysis suggests that weather conditions are significantly different during different seasons.")
    print("Recommendation: Consider adjusting operational strategies based on seasonal weather patterns to optimize performance.")
else:
    print("Inference: The analysis suggests that weather conditions are not significantly different during different seasons.")
    print("Recommendation: No changes needed based on seasonal weather patterns. Focus on other factors to optimize performance.")
```

```
Reject the Null Hypothesis (H0). Weather conditions are significantly different during different seasons.
Inference: The analysis suggests that weather conditions are significantly different during different seasons.
Recommendation: Consider adjusting operational strategies based on seasonal weather patterns to optimize performance.
```

Overall Recommendations:

1) Bike Rides on Weekdays vs. Weekends:

- Analysis: We performed a 2-Sample Independent T-test to check if there is a significant difference in the number of bike rides between weekdays and weekends.
- Recommendation: As there is significant difference, consider adjusting bike availability or pricing strategies based on the day of the week to optimize usage.

2) Weather Conditions and Bike Rides

- Analysis: We checked the normality of the data using the Shapiro-Wilk test and performed the Kruskal-Wallis test to see if the demand for bicycles on rent is the same across different weather conditions.
- Recommendation: As there is a significant difference, consider adjusting bike availability or pricing strategies based on weather conditions to optimize usage.

3) Weather Conditions Across Seasons:

- Analysis: We performed a Chi-square test to check if weather conditions are significantly different during different seasons.
- Recommendation: As weather conditions are significantly different across seasons, consider adjusting operational strategies based on seasonal weather patterns to optimize performance.

4) Seasons and Bike Rides:

- Analysis: We checked the normality of the data using the Shapiro-Wilk test and performed the Kruskal-Wallis test to see if the demand for bicycles on rent is the same across different seasons.
- Recommendation: As there is a significant difference, consider adjusting bike availability or pricing strategies based on seasons to optimize usage.

4) Marketing Strategies

- Implement targeted marketing strategies based on the analysis to attract more users and increase bike rentals.

5) Optimize Inventory Management

- High Demand Periods: Ensure a higher inventory of bicycles during clear, few clouds, and partly cloudy weather conditions, as these show the highest demand across all seasons.
- Moderate Demand Periods: Maintain a moderate inventory during misty and cloudy weather conditions, especially in winter when demand is relatively higher.

- Low Demand Periods: Reduce inventory during light snow, light rain, and thunderstorm conditions, as demand is generally lower.

6) Targeted Marketing Campaigns

- Seasonal Promotions: Run promotions and discounts during spring and summer to capitalize on the higher demand during these seasons.
- Weather-Based Offers: Offer special deals on days with clear and partly cloudy weather to attract more customers.

7) Customer Engagement

- Weather Updates: Provide real-time weather updates and recommendations to customers, encouraging them to rent bicycles on favorable weather days.
- Loyalty Programs: Implement loyalty programs to reward frequent renters, especially during high-demand periods.

8) Operational Adjustments

- Staffing: Adjust staffing levels based on expected demand. Increase staff during high-demand periods and reduce during low-demand periods to optimize operational efficiency.
- Maintenance: Schedule regular maintenance during low-demand periods to ensure bicycles are in top condition during peak times.

9) Data-Driven Decisions

- Continuous Monitoring: Regularly monitor rental data and weather patterns to adjust strategies dynamically.
- Feedback Collection: Collect customer feedback to understand preferences and improve service offerings.

10) Safety Measures

- Weather Alerts: Provide safety alerts and guidelines for riding in different weather conditions to ensure customer safety.
- Equipment: Offer weather-appropriate gear, such as raincoats and helmets, to enhance the riding experience.