Problem Statement and perform Exploratory Data Analysis

- Which variables are significant in predicting the demand for shared electric cycles in the Indian market?
- How well those variables describe the electric cycle demands ?

In [ ]:
```
# downloading data to working directory
!wget https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/original/bike_sharing.csv?1642089089
```

```
--2023-04-06 01:42:28--  https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/original/bike_sharing.csv?164208
9089
Resolving d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)... 13.35.153.13, 13.35.153.17, 13.35.153.227, ...
Connecting to d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)|13.35.153.13|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 648353 (633K) [text/plain]
Saving to: ‘bike_sharing.csv?1642089089’

bike_sharing.csv?16 100%[===================>] 633.16K  --.-KB/s    in 0.02s

2023-04-06 01:42:29 (38.3 MB/s) - ‘bike_sharing.csv?1642089089’ saved [648353/648353]
```

In [ ]:
```
#importing libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.special import comb
from scipy.stats import binom
import seaborn as sns
from statsmodels.distributions.empirical_distribution import ECDF  # empirical CDF\n",
from scipy.stats import norm,poisson,expon ## norm --> 'Normal' or \"Gaussian' "
from scipy.stats import ttest_ind,ttest_ind_from_stats,ttest_1samp,levene,shapiro,t,f_oneway,f,chi2_contingency,chi2
```

In [ ]:
```
# assigning data to object
df=pd.read_csv("/content/bike_sharing.csv?1642089089")
```

In [ ]:
```
#Exploring first five rows of data set
df.head()
```

Out[ ]:

	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

In [ ]:

```
#Exploring last five rows of data set
df.tail()
```

Out[ ]:

	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

Shape of data, data types of all the attributes:

In [ ]:

```
# Checking dataset shape
df.shape
```

Out[ ]:

```
(10886, 12)
```

In [ ]:

```
# Length of dataset
len(df)
```

Out[ ]:

```
10886
```

In [ ]:

```
# Checking dataset datatypes
df.dtypes
```

Out[ ]:
```
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
```

In [ ]:
```
#converting datetime column to datetime format
df['datetime'] = pd.to_datetime(df['datetime'])
```

In [ ]:
```
# Checking dataset datatypes
df.dtypes
```

Out[ ]:
```
datetime      datetime64[ns]
season                 int64
holiday                int64
workingday             int64
weather                int64
temp                 float64
atemp                float64
humidity               int64
windspeed            float64
casual                 int64
registered             int64
count                  int64
dtype: object
```

In [ ]:
```
# information about the data
# column names, datatypes, non-null values, memory usage
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  int64         
 2   holiday     10886 non-null  int64         
 3   workingday  10886 non-null  int64         
 4   weather     10886 non-null  int64         
 5   temp        10886 non-null  float64       
 6   atemp       10886 non-null  float64       
 7   humidity    10886 non-null  int64         
 8   windspeed   10886 non-null  float64       
 9   casual      10886 non-null  int64         
 10  registered  10886 non-null  int64         
 11  count       10886 non-null  int64         
dtypes: datetime64[ns](1), float64(3), int64(8)
memory usage: 1020.7 KB
```

Observations:

- Yulu Business Case Study dataset having 10886 rows and 12 columns.
- In this data set datetime column having unique values.
- Columns like season, holiday, workingday and weather are the categorical variable where temp,atemp,humidity, windspeed, casual, registered,and count are falling under continuous variables category.

In [ ]:
```
# Adding two more columns while extracting date and time from datetime column.
df['date'] = pd.to_datetime(df['datetime']).dt.date
df['time'] = pd.to_datetime(df['datetime']).dt.time
df["month"] = pd.to_datetime(df["date"]).dt.month
df["year"] = pd.to_datetime(df["date"]).dt.year
```

In [ ]:
```
df.head(5)
```

Out[ ]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count	date	time	month	year
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16	2011-01-01	00:00:00	1	2011
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40	2011-01-01	01:00:00	1	2011
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32	2011-01-01	02:00:00	1	2011
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13	2011-01-01	03:00:00	1	2011
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1	2011-01-01	04:00:00	1	2011

In [ ]:

```
# Checking number of nunique values in our dataset
for i in df.columns:
 print(i,":",df[i].nunique())
```

```
datetime : 10886
season : 4
holiday : 2
workingday : 2
weather : 4
temp : 49
atemp : 60
humidity : 89
windspeed : 28
casual : 309
registered : 731
count : 822
date : 456
time : 24
month : 12
year : 2
```

In [ ]:
```
df.isna().sum()
```

Out[ ]:
```
datetime      0
season        0
holiday       0
workingday    0
weather       0
temp          0
atemp         0
humidity      0
windspeed     0
casual        0
registered    0
count         0
date          0
time          0
month         0
year          0
dtype: int64
```

Observations:

- Season are categorised in 4 parts i.e 1: spring, 2: summer, 3: fall, 4: winter
- Weather has been categorised in 4 parts i.e

  1: Clear, Few clouds, partly cloudy partly cloudy.

  2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist.

  3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds.

  4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog.
- In holiday column, 1 stand for the day which is holiday and 0 for the day which is not holiday.
- Working day categorised in 1 and 0 where 1 indicates day is neither weekend nor holiday and 0 indicates day is weekend or holiday.

nunique,missing value detection and statistical summary:

In [ ]:
```
# Checking number of unique values in our dataset
for i in df.columns:
  print(i,":",df[i].nunique())
```

```
datetime : 10886
season : 4
holiday : 2
workingday : 2
weather : 4
temp : 49
atemp : 60
humidity : 89
windspeed : 28
casual : 309
registered : 731
count : 822
date : 456
time : 24
month : 12
year : 2
temp_segment : 4
atemp_segment : 4
humidity_segment : 3
windspeed_segment : 4
```

In [109…
```
# season: season (1: spring, 2: summer, 3: fall, 4: winter)
# checking days under unique season
season=df.groupby('season')['date'].nunique()
j = 1
for i in ['spring','summer','Fall','Winter']:
 print(f'days in {i} season: {season[j]}')
 j+=1
```

```
days in spring season: 114
days in summer season: 114
days in Fall season: 114
days in Winter season: 114
```

In [110…
```
# holiday: whether day is a holiday or not
# 0 indicates normal days
# 1 indicates holidays.
# checking days under unique holiday
holiday=df.groupby('holiday')['date'].nunique()
j = 0
```

```
for i in ['Normal Days','Holidays']:
 print(f'number of {i}: {holiday[j]}')
 j+=1
```

```
number of Normal Days: 443
number of Holidays: 13
```

In [111…

```
#working day: if day is neither weekend nor holiday is 1, otherwise is 0.
# checking days under unique workingday
workingday=df.groupby('workingday')['date'].nunique()
j = 0
for i in ['non Working Days','Working Days']:
 print(f'number of {i}: {workingday[j]}')
 j+=1
```

```
number of non Working Days: 145
number of Working Days: 311
```

In [ ]:

```
# weather:
# 1: Clear, Few clouds, partly cloudy, partly cloudy
# 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
# 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light  Rain + Scattered clouds
# 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
# checking days under unique weather
df.groupby('weather')['date'].nunique().to_frame().T
```

Out[ ]:

weather	1	2	3	4
date	434	346	187	1

In [ ]:

```
# temp: temperature in Celsius
# checking days under unique measured temperature
df.groupby('temp')['date'].nunique().to_frame().T
```

Out[ ]:

temp	0.82	1.64	2.46	3.28	4.10	4.92	5.74	6.56	7.38	8.20	...	32.80	33.62	34.44	35.26	36.08	36.90	37.72	38.54	39.36	41.00
date	1	1	2	5	12	21	32	39	36	65	...	79	61	39	32	17	20	11	5	2	1

1 rows × 49 columns

In [ ]:
```
# atemp: feeling temperature in Celsius
#  checking days under unique feeling temperature
df.groupby('atemp')['date'].nunique().to_frame().T
```

Out[ ]:

atemp	0.760	1.515	2.275	3.030	3.790	4.545	5.305	6.060	6.820	7.575	...	38.635	39.395	40.150	40.910	41.665	42.425	43.180	43.940	44.6
date	1	1	2	4	6	4	13	22	24	27	...	33	33	22	19	9	11	3	5	

1 rows × 60 columns

In [ ]:
```
# humidity: humidity
#  checking days under unique humidity
df.groupby('humidity')['date'].nunique().to_frame().T
```

Out[ ]:

humidity	0	8	10	12	13	14	15	16	17	18	...	88	89	90	91	92	93	94	96	97	100
date	1	1	1	1	1	1	2	3	4	5	...	105	46	4	1	2	57	83	1	1	40

1 rows × 89 columns

In [ ]:
```
# windspeed: wind speed
#  checking days under unique windspeed
df.groupby('windspeed')['date'].nunique().to_frame().T
```

Out[ ]:

windspeed	0.0000	6.0032	7.0015	8.9981	11.0014	12.9980	15.0013	16.9979	19.0012	19.9995	...	36.9974	39.0007	40.9973	43.0006	43.9989
date	321	323	362	378	381	377	367	341	304	250	...	16	20	9	8	6

1 rows × 28 columns

In [ ]:
```
# statistical summary
df.describe(include=["int","float"])
```

Out[ ]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	1088
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.886460	12.799395	36.021955	155.552177	19
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.245033	8.164537	49.960477	151.039033	18
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	0.000000	0.000000	0.000000	0.000000	
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	47.000000	7.001500	4.000000	36.000000	4
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	62.000000	12.998000	17.000000	118.000000	14
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	77.000000	16.997900	49.000000	222.000000	28
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	100.000000	56.996900	367.000000	886.000000	97

In [ ]:

```python
# statistical summary
df.describe(include="object")
```

Out[ ]:

	date	time
count	10886	10886
unique	456	24
top	2011-01-01	12:00:00
freq	24	456

In [ ]:

```python
# null values analysis
df.isna().sum()
```

Out[ ]:
```
datetime      0
season        0
holiday       0
workingday    0
weather       0
temp          0
atemp         0
humidity      0
windspeed     0
casual        0
registered    0
count         0
date          0
time          0
month         0
year          0
dtype: int64
```

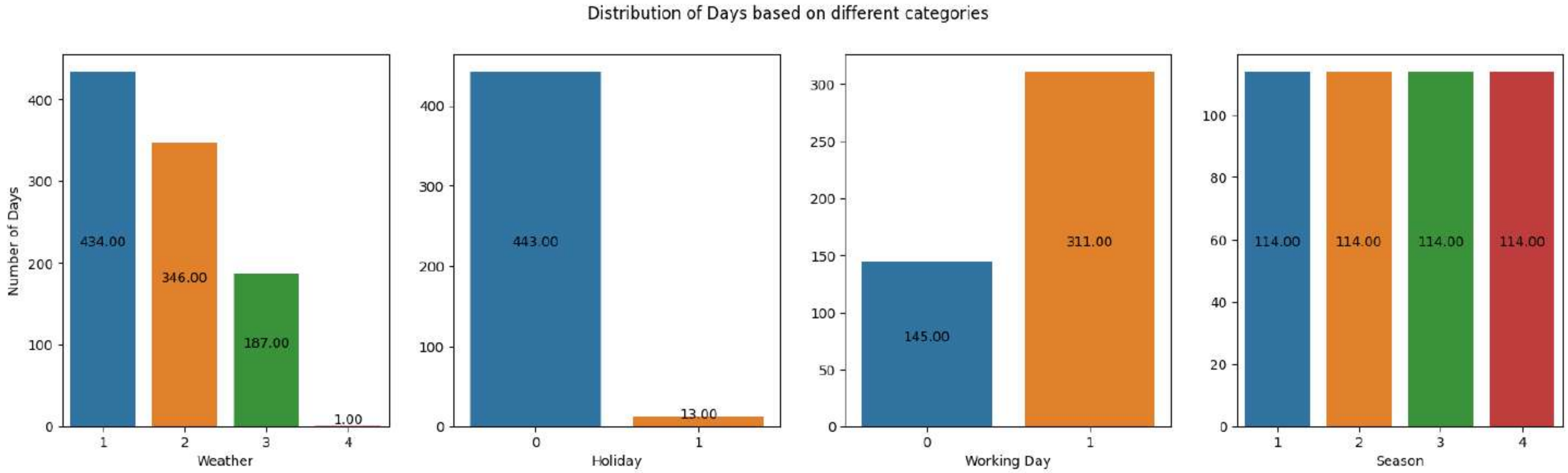
Observations:

- Total number of unique date has been found in data sets are 456.
- Data has been captured in 24 unique time slots.
- No Null values are present in the Data set and Neither duplicate row has been found.
- Avg.temperature has been captured is 20.23 degree , minimum is 0.82 degree and maximum is 41 degree centigrade.
- Highest temperature experienced by user is approximate 46 degree centigrade.
- Avg. windspeed in our data set is 12.8 m/s and maximum wind speed is 57 m/s.
- Avg. humidity is arround 62%.
- Highest number achieved is 367 in count of casual users.
- Highest number achieved is 886 in count of registered users.
- Highest number achieved is 997 in count of total users.

Univariate Analysis:

In [ ]:
```
## Distribution of categorical variables
fig = plt.figure(figsize=(20,5))
plt.subplot(1, 4, 1)
ax = sns.barplot(data=df.groupby('weather')['date'].nunique().reset_index(),y='date',x='weather')
for p in ax.patches:
```

```
  ax.annotate("{:,.2f}".format(p.get_height()),
 (p.get_x() + p.get_width()/2, p.get_height()/2),ha = 'center', va = 'bottom')
plt.xlabel('Weather')
plt.ylabel('Number of Days')
plt.subplot(1,4, 2)
ax = sns.barplot(data=df.groupby('holiday')['date'].nunique().reset_index(),y='date',x='holiday')
for p in ax.patches:
  ax.annotate("{:,.2f}".format(p.get_height()),
 (p.get_x() + p.get_width()/2, p.get_height()/2),ha = 'center', va = 'bottom')
plt.xlabel('Holiday')
plt.ylabel('')
plt.subplot(1,4, 3)
ax = sns.barplot(data=df.groupby('workingday')['date'].nunique().reset_index(),y='date',x='workingday')
for p in ax.patches:
  ax.annotate("{:,.2f}".format(p.get_height()),
 (p.get_x() + p.get_width()/2, p.get_height()/2),ha = 'center', va = 'bottom')
plt.xlabel('Working Day')
plt.ylabel('')
plt.subplot(1,4, 4)
ax = sns.barplot(data=df.groupby('season')['date'].nunique().reset_index(),y='date',x='season')
for p in ax.patches:
  ax.annotate("{:,.2f}".format(p.get_height()),
 (p.get_x() + p.get_width()/2, p.get_height()/2),ha = 'center', va = 'bottom')
plt.xlabel('Season')
plt.ylabel('')
plt.suptitle("Distribution of Days based on different categories")
plt.show()
```

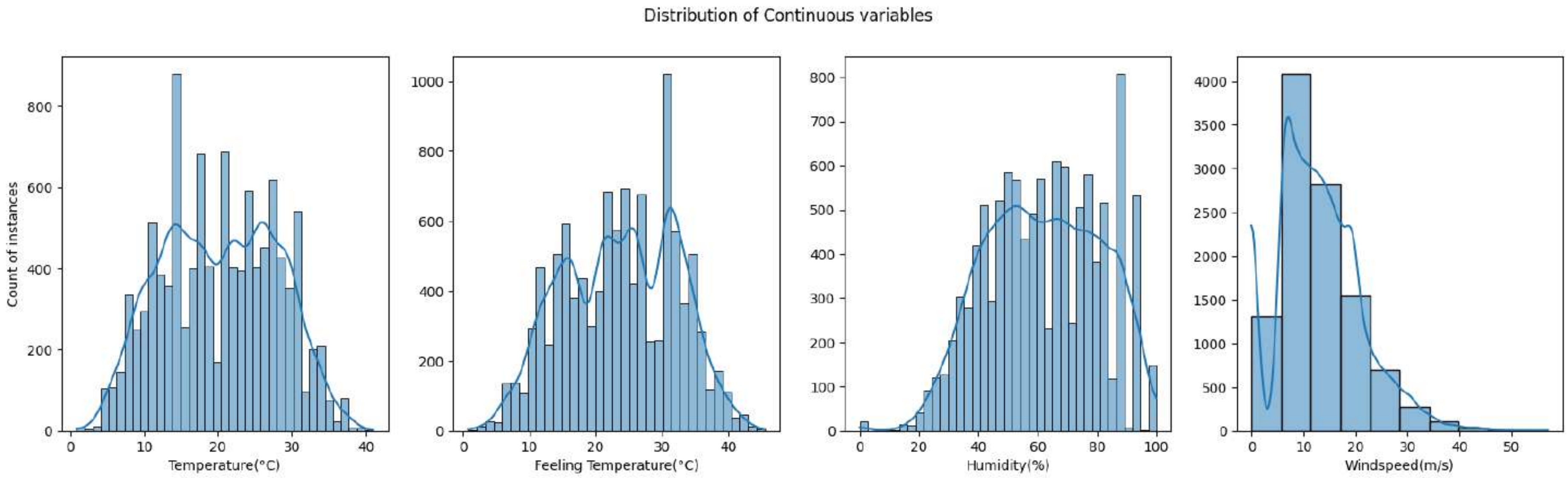


Observations:

- Extreme weather conditions has been observed on one day only.
- 97% of the days are normal days.
- 68% of the days are working days.
- Every season having equal number of days.

In [ ]:
```
# Distribution of Continuous variables
fig = plt.figure(figsize=(20,5))
# Distribution of temperature
plt.subplot(1, 4, 1)
ax = sns.histplot(df,x='temp',kde=True)
plt.xlabel('Temperature(°C)')
plt.ylabel('Count of instances')
# Distribution of feeling temperature
plt.subplot(1,4,2)
ax = sns.histplot(df,x='atemp',kde=True)
plt.xlabel('Feeling Temperature(°C)')
plt.ylabel("")
# Distribution of Humidity
plt.subplot(1,4,3)
sns.histplot(df,x='humidity',kde=True)
```

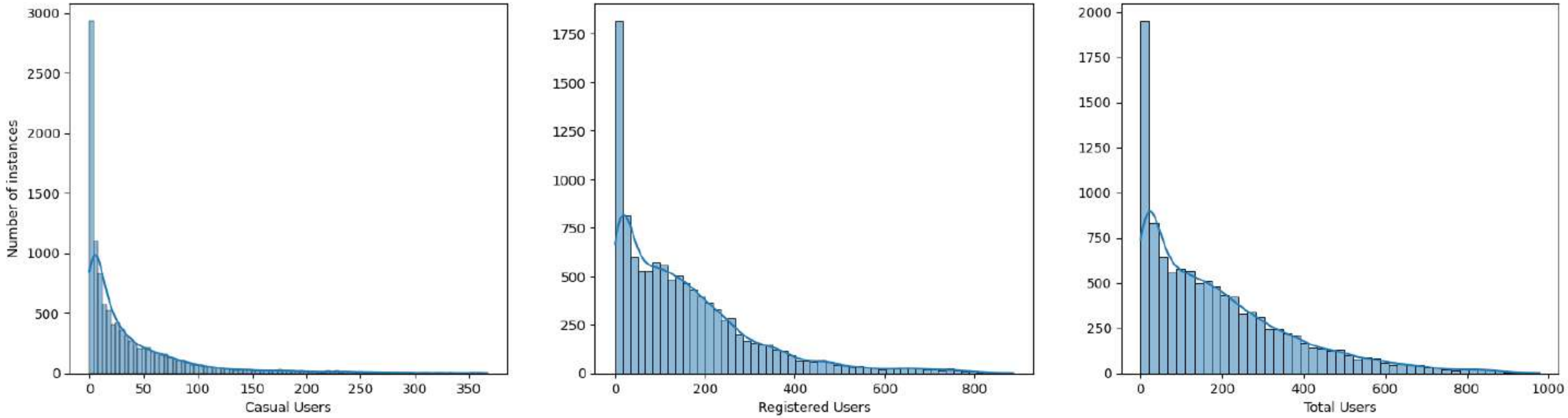
```
plt.xlabel('Humidity(%)')
plt.ylabel("")
# Distribution of WindSpeed
plt.subplot(1,4,4)
sns.histplot(df,x='windspeed',kde=True,bins = 10)
plt.xlabel('Windspeed(m/s)')
plt.ylabel("")
plt.suptitle("Distribution of Continuous variables")
plt.show()
```



In [ ]:

```
# Distribution of count, registered and casual rentals
fig = plt.figure(figsize=(20,5))
# distribution of casual users
plt.subplot(1, 3, 1)
sns.histplot(df['casual'],kde=True)
plt.xlabel('Casual Users')
plt.ylabel('Number of instances')
# distribution of Registered users
plt.subplot(1, 3, 2)
sns.histplot(df['registered'],kde=True)
plt.xlabel('Registered Users')
plt.ylabel('')
# distribution of Total users
plt.subplot(1, 3, 3)
sns.histplot(df['count'],kde=True)
```

```
plt.xlabel('Total Users')
plt.ylabel('')
plt.show()
```



Observations:

- User counts data has been found right skewed in nature.

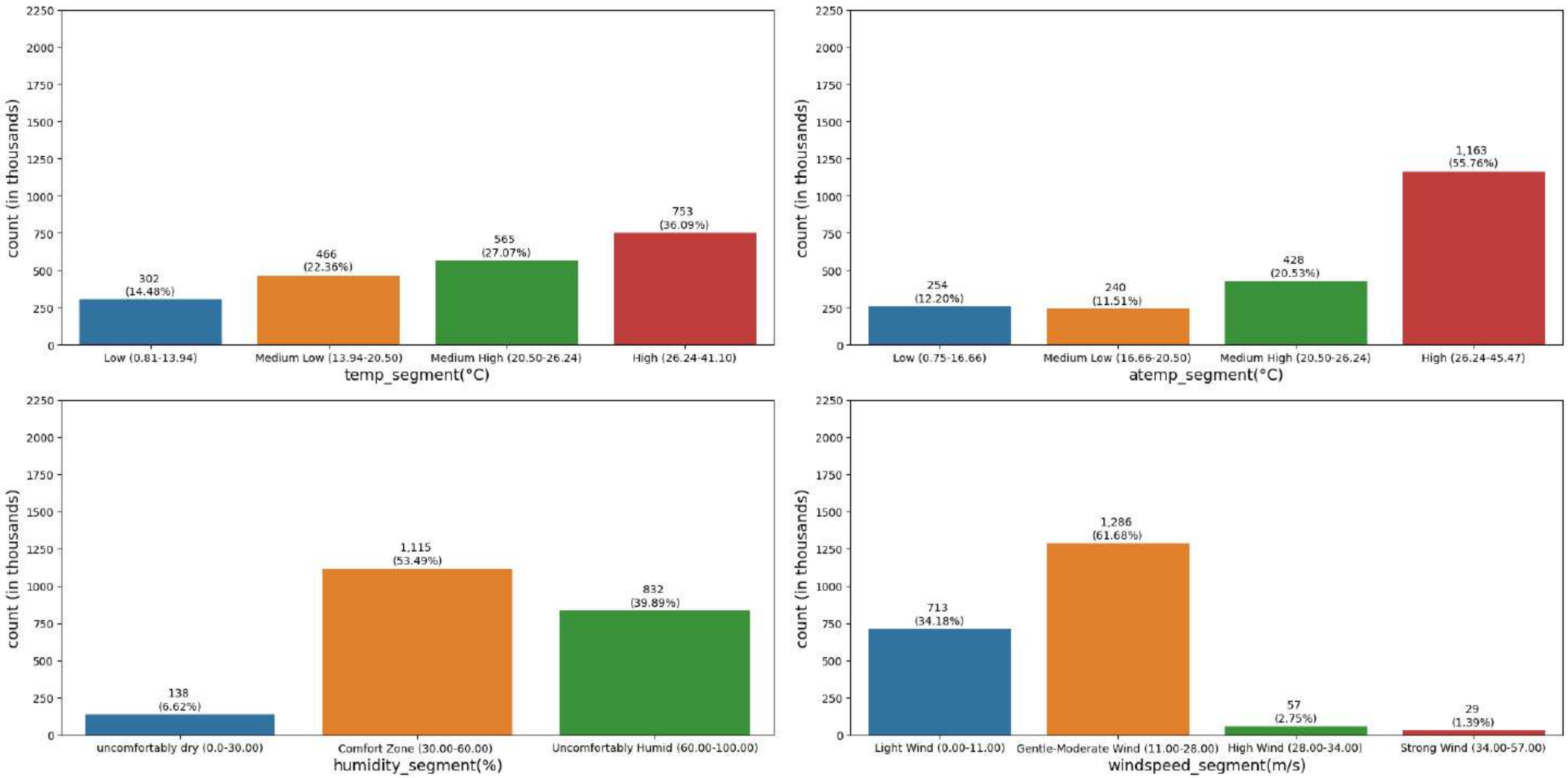
Relationship between Dependent and independent variables:

In [ ]:
```
# defining category based on the temp,atemp,humidity and wind speed in different segment.
bins=[0.81, 13.94, 20.50, 26.24, 41.10]
labels=["Low (0.81-13.94)","Medium Low (13.94-20.50)","Medium High (20.50-26.24)","High (26.24-41.10)"]
df["temp_segment"]=pd.cut(x=df["temp"], bins=bins, labels=labels, include_lowest=True)
bins=[0.75, 16.66, 20.50, 26.24, 45.456]
labels=["Low (0.75-16.66)","Medium Low (16.66-20.50)","Medium High (20.50-26.24)","High (26.24-45.47)"]
df["atemp_segment"]=pd.cut(x=df["atemp"], bins=bins, labels=labels, include_lowest=True)
bins=[0.00, 30.00, 60.00, 100.00]
labels=["uncomfortably dry (0.0-30.00)","Comfort Zone (30.00-60.00) ","Uncomfortably Humid (60.00-100.00)"]
df["humidity_segment"]=pd.cut(x=df["humidity"], bins=bins, labels=labels, include_lowest=True)
bins=[0.00, 11.00, 28.00, 34.00, 57.00]
labels=["Light Wind (0.00-11.00)","Gentle-Moderate Wind (11.00-28.00)","High Wind (28.00-34.00)","Strong Wind (34.00-57.00)"]
df["windspeed_segment"]=pd.cut(x=df["windspeed"], bins=bins, labels=labels, include_lowest=True)
```

In [ ]:
```
#  Analysis for total count
ts=df.groupby(["temp_segment"])["count"].sum().reset_index()
ts["count"]=ts["count"]/1000
ats=df.groupby(["atemp_segment"])["count"].sum().reset_index()
ats["count"]=ats["count"]/1000
hs=df.groupby(["humidity_segment"])["count"].sum().reset_index()
hs["count"]=hs["count"]/1000
ws=df.groupby(["windspeed_segment"])["count"].sum().reset_index()
ws["count"]=ws["count"]/1000

total_sum=(df["count"].sum())/1000
fig = plt.figure(figsize=(20,10))
plt.subplot(2, 2, 1)
ax = sns.barplot(data=ts,x='temp_segment',y='count',orient='v')
plt.yticks(np.arange(0, 2500, 250))
plt.xlabel('temp_segment(°C)',fontsize=14)
plt.ylabel('count (in thousands)',fontsize=14)
for p in ax.patches:
  ax.annotate("{:,.0f} \n({:,.2f}%)".format(p.get_height(), p.get_height() * 100/total_sum),
  (p.get_x() + p.get_width()/2, p.get_height()+20),ha = 'center', va = 'bottom')
plt.subplot(2, 2, 2)
ax = sns.barplot(data=ats,x='atemp_segment',y='count',orient='v')
plt.yticks(np.arange(0, 2500, 250))
plt.xlabel('atemp_segment(°C)',fontsize=14)
plt.ylabel('count (in thousands)',fontsize=14)
for p in ax.patches:
  ax.annotate("{:,.0f} \n({:,.2f}%)".format(p.get_height(), p.get_height() * 100/total_sum),
  (p.get_x() + p.get_width()/2, p.get_height()+20),ha = 'center', va = 'bottom')
plt.subplot(2, 2, 3)
ax = sns.barplot(data=hs,x='humidity_segment',y='count',orient='v')
plt.yticks(np.arange(0, 2500, 250))
plt.xlabel('humidity_segment(%)',fontsize=14)
plt.ylabel('count (in thousands)',fontsize=14)
for p in ax.patches:
  ax.annotate("{:,.0f} \n({:,.2f}%)".format(p.get_height(), p.get_height() * 100/total_sum),
  (p.get_x() + p.get_width()/2, p.get_height()+20),ha = 'center', va = 'bottom')
plt.subplot(2, 2, 4)
ax = sns.barplot(data=ws,x='windspeed_segment',y='count',orient='v')
plt.yticks(np.arange(0, 2500, 250))
plt.xlabel('windspeed_segment(m/s)',fontsize=14)
plt.ylabel('count (in thousands)',fontsize=14)
for p in ax.patches:
  ax.annotate("{:,.0f} \n({:,.2f}%)".format(p.get_height(), p.get_height() * 100/total_sum),
```

```
    (p.get_x() + p.get_width()/2, p.get_height()+20),ha = 'center', va = 'bottom')
plt.tight_layout()
plt.show()
```



Observations:

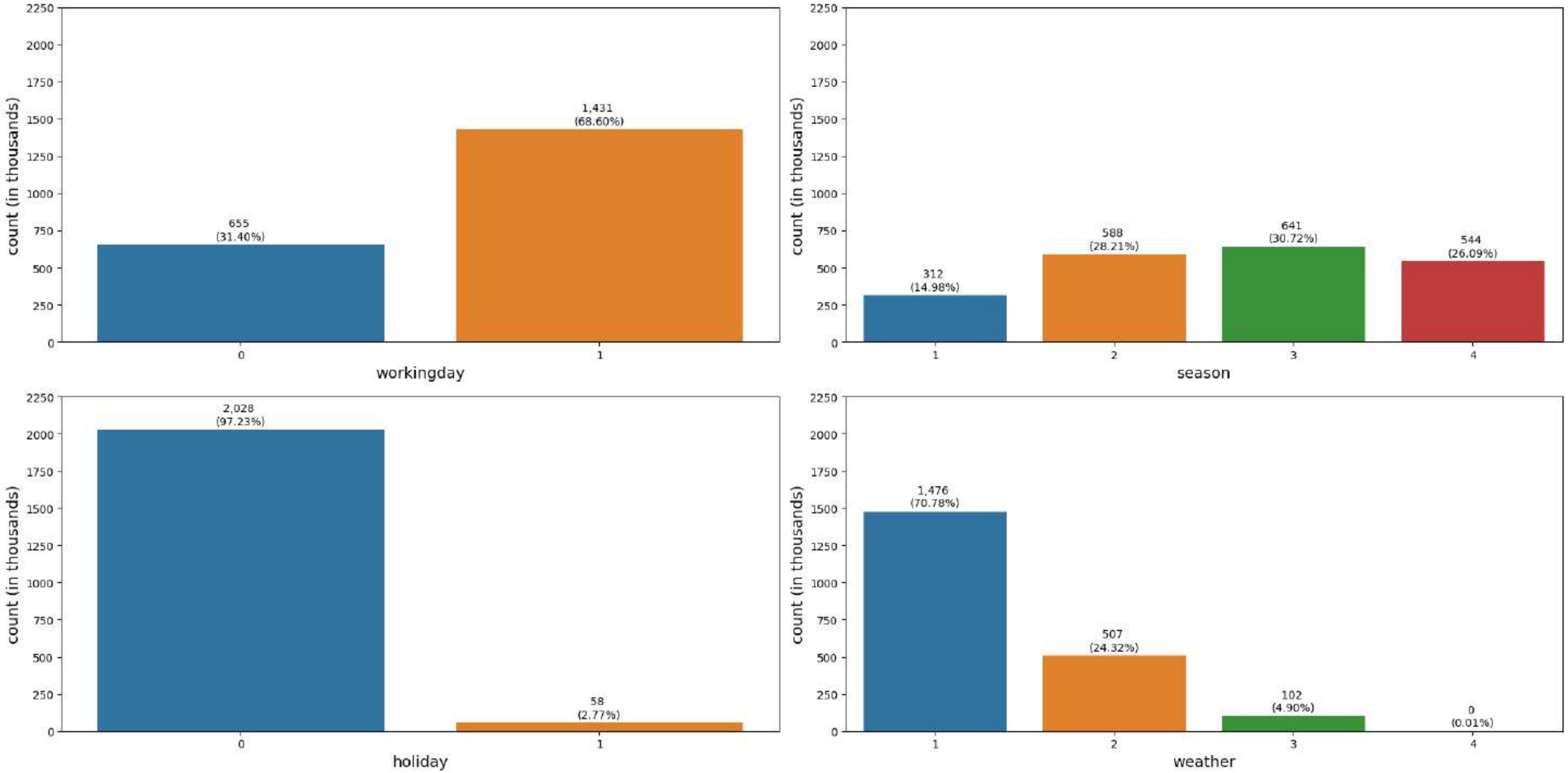
- Highest total users count has been observed when temperature in range from above 20°C-41°C.
- More then 50% of the total users noticed, when humidity level in the range of 30%-60% which is a comfort zone for human being.
- Maximum (62%) of the total users count has been obseved when windspeed is in Gentle to moderate(11-28).

In [ ]:

```
wd=df.groupby(["workingday"])["count"].sum().reset_index()
wd["count"]=wd["count"]/1000
```

```
s=df.groupby(["season"])["count"].sum().reset_index()
s["count"]=s["count"]/1000
hd=df.groupby(["holiday"])["count"].sum().reset_index()
hd["count"]=hd["count"]/1000
w=df.groupby(["weather"])["count"].sum().reset_index()
w["count"]=w["count"]/1000

total_sum=(df["count"].sum())/1000
fig = plt.figure(figsize=(20,10))
plt.subplot(2, 2, 1)
ax = sns.barplot(data=wd,x='workingday',y='count',orient='v')
plt.yticks(np.arange(0, 2500, 250))
plt.xlabel('workingday',fontsize=14)
plt.ylabel('count (in thousands)',fontsize=14)
for p in ax.patches:
  ax.annotate("{:,.0f} \n({:,.2f}%)".format(p.get_height(), p.get_height() * 100/total_sum),
  (p.get_x() + p.get_width()/2, p.get_height()+20),ha = 'center', va = 'bottom')
plt.subplot(2, 2, 2)
ax = sns.barplot(data=s,x='season',y='count',orient='v')
plt.yticks(np.arange(0, 2500, 250))
plt.xlabel('season',fontsize=14)
plt.ylabel('count (in thousands)',fontsize=14)
for p in ax.patches:
  ax.annotate("{:,.0f} \n({:,.2f}%)".format(p.get_height(), p.get_height() * 100/total_sum),
  (p.get_x() + p.get_width()/2, p.get_height()+20),ha = 'center', va = 'bottom')
plt.subplot(2, 2, 3)
ax = sns.barplot(data=hd,x='holiday',y='count',orient='v')
plt.yticks(np.arange(0, 2500, 250))
plt.xlabel('holiday',fontsize=14)
plt.ylabel('count (in thousands)',fontsize=14)
for p in ax.patches:
  ax.annotate("{:,.0f} \n({:,.2f}%)".format(p.get_height(), p.get_height() * 100/total_sum),
  (p.get_x() + p.get_width()/2, p.get_height()+20),ha = 'center', va = 'bottom')
plt.subplot(2, 2, 4)
ax = sns.barplot(data=w,x='weather',y='count',orient='v')
plt.yticks(np.arange(0, 2500, 250))
plt.xlabel('weather',fontsize=14)
plt.ylabel('count (in thousands)',fontsize=14)
for p in ax.patches:
  ax.annotate("{:,.0f} \n({:,.2f}%)".format(p.get_height(), p.get_height() * 100/total_sum),
  (p.get_x() + p.get_width()/2, p.get_height()+20),ha = 'center', va = 'bottom')
plt.tight_layout()
plt.show()
```



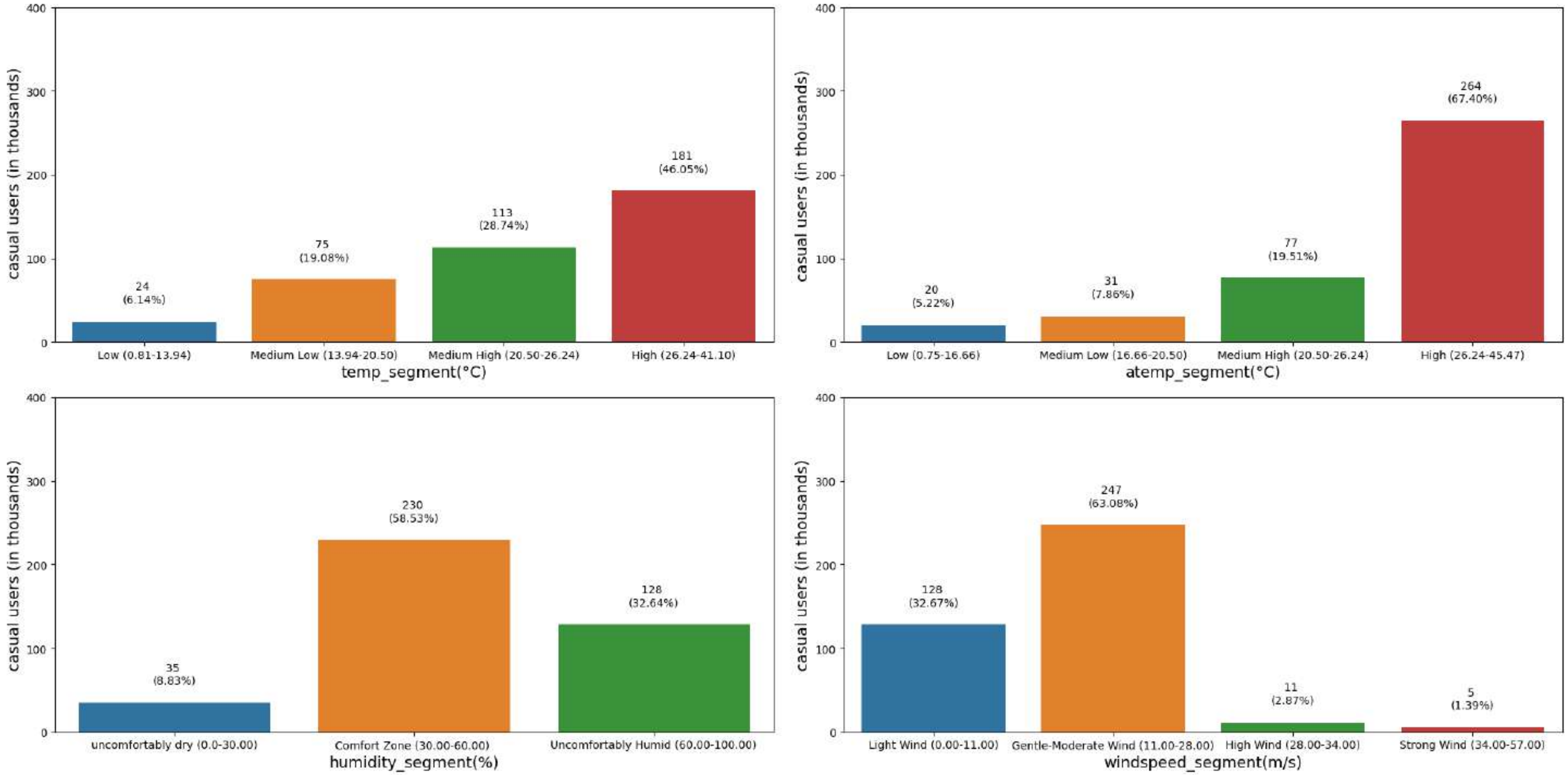
Observations:

- Total users counts are 2086 thousands.
- 71% of the total count observed when weather is found clear.
- Total users count are more or less equally distributed across the all seasons.
- 97% of total users count are belongs to the normal days and only 68% instances belongs to working days.

In [ ]:
```
#  Analysis for total casual users
ts=df.groupby(["temp_segment"])["casual"].sum().reset_index()
ts["casual"]=ts["casual"]/1000
ats=df.groupby(["atemp_segment"])["casual"].sum().reset_index()
ats["casual"]=ats["casual"]/1000
```

```
hs=df.groupby(["humidity_segment"])["casual"].sum().reset_index()
hs["casual"]=hs["casual"]/1000
ws=df.groupby(["windspeed_segment"])["casual"].sum().reset_index()
ws["casual"]=ws["casual"]/1000

total_sum=(df["casual"].sum())/1000
fig = plt.figure(figsize=(20,10))
plt.subplot(2, 2, 1)
ax = sns.barplot(data=ts,x='temp_segment',y='casual',orient='v')
plt.yticks(np.arange(0, 500, 100))
plt.xlabel('temp_segment(°C)',fontsize=14)
plt.ylabel(' casual users (in thousands)',fontsize=14)
for p in ax.patches:
  ax.annotate("{:,.0f} \n({:,.2f}%)".format(p.get_height(), p.get_height() * 100/total_sum),
  (p.get_x() + p.get_width()/2, p.get_height()+20),ha = 'center', va = 'bottom')
plt.subplot(2, 2, 2)
ax = sns.barplot(data=ats,x='atemp_segment',y='casual',orient='v')
plt.yticks(np.arange(0, 500, 100))
plt.xlabel('atemp_segment(°C)',fontsize=14)
plt.ylabel('casual users (in thousands)',fontsize=14)
for p in ax.patches:
  ax.annotate("{:,.0f} \n({:,.2f}%)".format(p.get_height(), p.get_height() * 100/total_sum),
  (p.get_x() + p.get_width()/2, p.get_height()+20),ha = 'center', va = 'bottom')
plt.subplot(2, 2, 3)
ax = sns.barplot(data=hs,x='humidity_segment',y='casual',orient='v')
plt.yticks(np.arange(0, 500, 100))
plt.xlabel('humidity_segment(%)',fontsize=14)
plt.ylabel('casual users (in thousands)',fontsize=14)
for p in ax.patches:
  ax.annotate("{:,.0f} \n({:,.2f}%)".format(p.get_height(), p.get_height() * 100/total_sum),
  (p.get_x() + p.get_width()/2, p.get_height()+20),ha = 'center', va = 'bottom')
plt.subplot(2, 2, 4)
ax = sns.barplot(data=ws,x='windspeed_segment',y='casual',orient='v')
plt.yticks(np.arange(0, 500, 100))
plt.xlabel('windspeed_segment(m/s)',fontsize=14)
plt.ylabel('casual users (in thousands)',fontsize=14)
for p in ax.patches:
  ax.annotate("{:,.0f} \n({:,.2f}%)".format(p.get_height(), p.get_height() * 100/total_sum),
  (p.get_x() + p.get_width()/2, p.get_height()+20),ha = 'center', va = 'bottom')
plt.tight_layout()
plt.show()
```



In [ ]:
```
total_sum=(df["casual"].sum())/1000
total_sum
```

Out[ ]:
```
392.135
```

Observations:

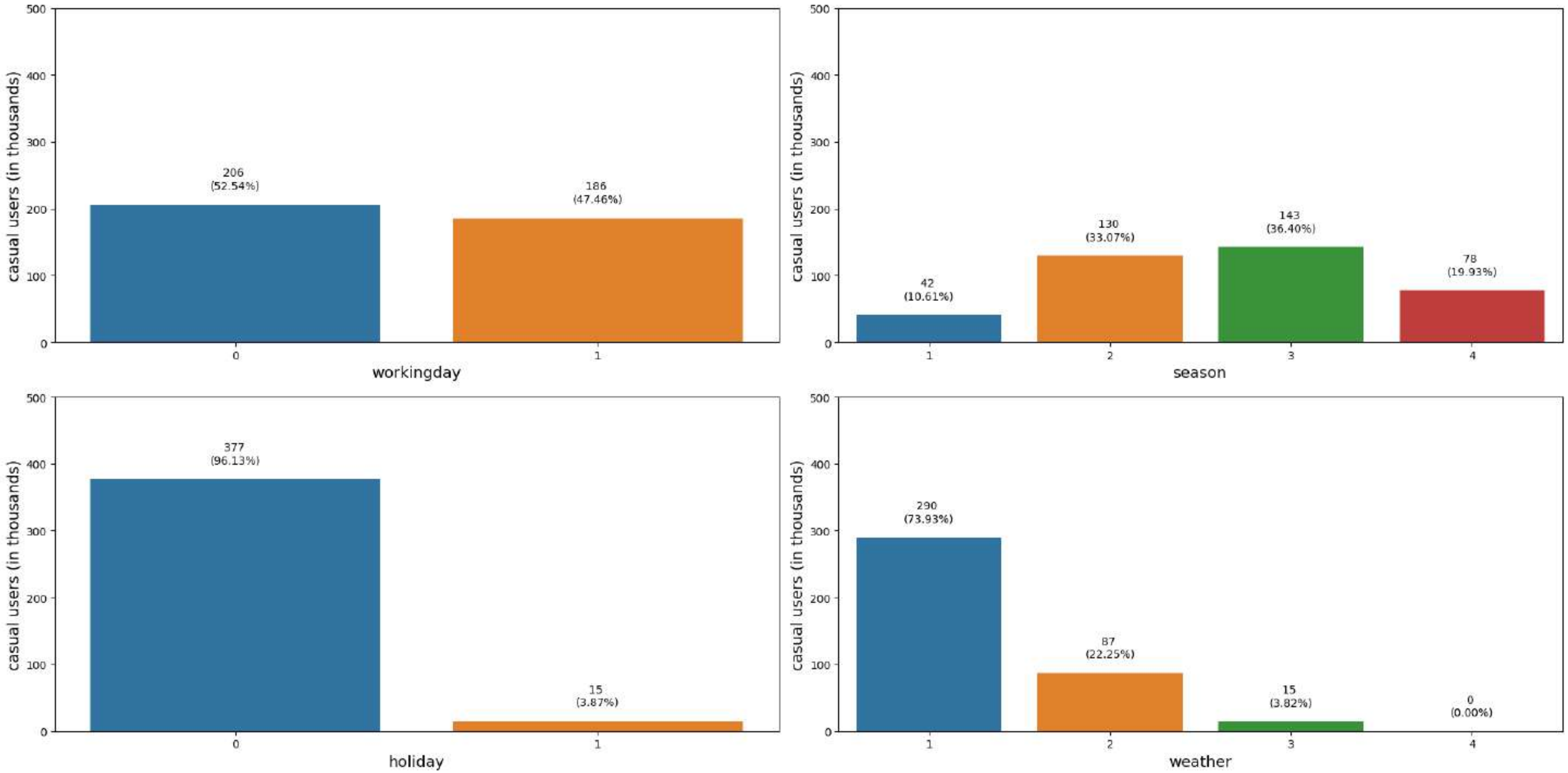
- Approximate 19% of the total users count belongs to casual category.
- Total users count under casual category is 393 thousands.
- Highest total users count has been observed when temperature in range from above 20°C-41°C.
- More then 58% of the total users noticed, when humidity level in the range of 30%-60% which is a comfort zone for human being.

- Maximum (63%) of the total users count has been obseved when windspeed is in Gentle to moderate(11-28).

In [ ]:
```
wd=df.groupby(["workingday"])["casual"].sum().reset_index()
wd["casual"]=wd["casual"]/1000
s=df.groupby(["season"])["casual"].sum().reset_index()
s["casual"]=s["casual"]/1000
hd=df.groupby(["holiday"])["casual"].sum().reset_index()
hd["casual"]=hd["casual"]/1000
w=df.groupby(["weather"])["casual"].sum().reset_index()
w["casual"]=w["casual"]/1000

total_sum=(df["casual"].sum())/1000
fig = plt.figure(figsize=(20,10))
plt.subplot(2, 2, 1)
ax = sns.barplot(data=wd,x='workingday',y='casual',orient='v')
plt.yticks(np.arange(0, 600, 100))
plt.xlabel('workingday',fontsize=14)
plt.ylabel('casual users (in thousands)',fontsize=14)
for p in ax.patches:
  ax.annotate("{:,.0f} \n({:,.2f}%)".format(p.get_height(), p.get_height() * 100/total_sum),
  (p.get_x() + p.get_width()/2, p.get_height()+20),ha = 'center', va = 'bottom')
plt.subplot(2, 2, 2)
ax = sns.barplot(data=s,x='season',y='casual',orient='v')
plt.yticks(np.arange(0, 600, 100))
plt.xlabel('season',fontsize=14)
plt.ylabel('casual users (in thousands)',fontsize=14)
for p in ax.patches:
  ax.annotate("{:,.0f} \n({:,.2f}%)".format(p.get_height(), p.get_height() * 100/total_sum),
  (p.get_x() + p.get_width()/2, p.get_height()+20),ha = 'center', va = 'bottom')
plt.subplot(2, 2, 3)
ax = sns.barplot(data=hd,x='holiday',y='casual',orient='v')
plt.yticks(np.arange(0, 600, 100))
plt.xlabel('holiday',fontsize=14)
plt.ylabel('casual users (in thousands)',fontsize=14)
for p in ax.patches:
  ax.annotate("{:,.0f} \n({:,.2f}%)".format(p.get_height(), p.get_height() * 100/total_sum),
  (p.get_x() + p.get_width()/2, p.get_height()+20),ha = 'center', va = 'bottom')
plt.subplot(2, 2, 4)
ax = sns.barplot(data=w,x='weather',y='casual',orient='v')
plt.yticks(np.arange(0, 600, 100))
plt.xlabel('weather',fontsize=14)
plt.ylabel('casual users (in thousands)',fontsize=14)
for p in ax.patches:
```

```
  ax.annotate("{:,.0f} \n({:,.2f}%)".format(p.get_height(), p.get_height() * 100/total_sum),
  (p.get_x() + p.get_width()/2, p.get_height()+20),ha = 'center', va = 'bottom')
plt.tight_layout()
plt.show()
```



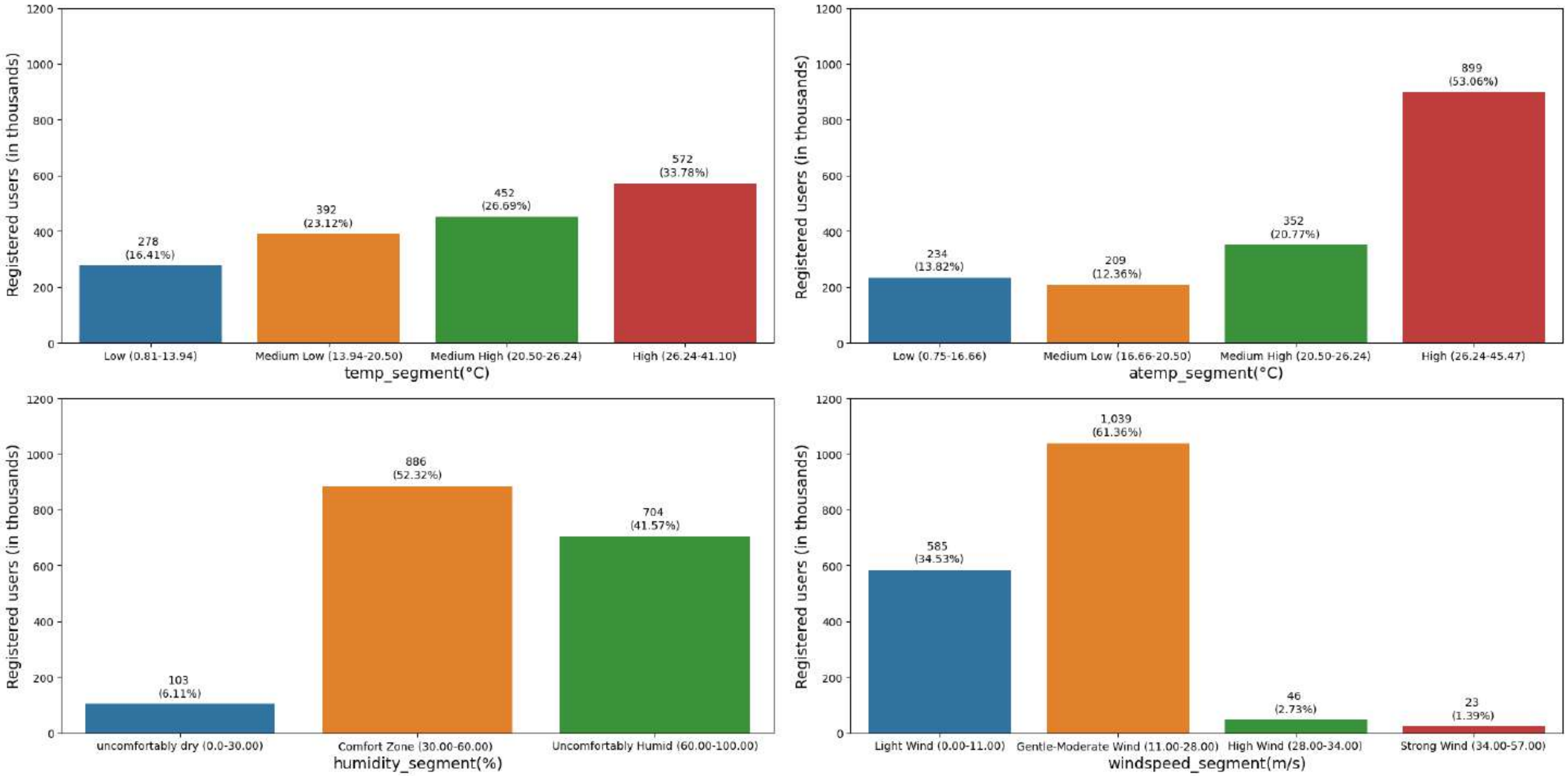
Observations:

- Total users counts for casual category are 393 thousands.
- 74% of the total count observed when weather is found clear.
- 70% of the total users count belongs to summer and fall season.
- 96% of total users count are belongs to the normal days.
- Casual user counts almost equally distributed in workingdays.

In [ ]:
```
#  Analysis for total Registered users
ts=df.groupby(["temp_segment"])["registered"].sum().reset_index()
ts["registered"]=ts["registered"]/1000
ats=df.groupby(["atemp_segment"])["registered"].sum().reset_index()
ats["registered"]=ats["registered"]/1000
hs=df.groupby(["humidity_segment"])["registered"].sum().reset_index()
hs["registered"]=hs["registered"]/1000
ws=df.groupby(["windspeed_segment"])["registered"].sum().reset_index()
ws["registered"]=ws["registered"]/1000

total_sum=(df["registered"].sum())/1000
fig = plt.figure(figsize=(20,10))
plt.subplot(2, 2, 1)
ax = sns.barplot(data=ts,x='temp_segment',y='registered',orient='v')
plt.yticks(np.arange(0, 1400, 200))
plt.xlabel('temp_segment(°C)',fontsize=14)
plt.ylabel(' Registered users (in thousands)',fontsize=14)
for p in ax.patches:
  ax.annotate("{:,.0f} \n({:,.2f}%)".format(p.get_height(), p.get_height() * 100/total_sum),
  (p.get_x() + p.get_width()/2, p.get_height()+20),ha = 'center', va = 'bottom')
plt.subplot(2, 2, 2)
ax = sns.barplot(data=ats,x='atemp_segment',y='registered',orient='v')
plt.yticks(np.arange(0, 1400, 200))
plt.xlabel('atemp_segment(°C)',fontsize=14)
plt.ylabel('Registered users (in thousands)',fontsize=14)
for p in ax.patches:
  ax.annotate("{:,.0f} \n({:,.2f}%)".format(p.get_height(), p.get_height() * 100/total_sum),
  (p.get_x() + p.get_width()/2, p.get_height()+20),ha = 'center', va = 'bottom')
plt.subplot(2, 2, 3)
ax = sns.barplot(data=hs,x='humidity_segment',y='registered',orient='v')
plt.yticks(np.arange(0, 1400, 200))
plt.xlabel('humidity_segment(%)',fontsize=14)
plt.ylabel('Registered users (in thousands)',fontsize=14)
for p in ax.patches:
  ax.annotate("{:,.0f} \n({:,.2f}%)".format(p.get_height(), p.get_height() * 100/total_sum),
  (p.get_x() + p.get_width()/2, p.get_height()+20),ha = 'center', va = 'bottom')
plt.subplot(2, 2, 4)
ax = sns.barplot(data=ws,x='windspeed_segment',y='registered',orient='v')
plt.yticks(np.arange(0, 1400, 200))
plt.xlabel('windspeed_segment(m/s)',fontsize=14)
plt.ylabel('Registered users (in thousands)',fontsize=14)
for p in ax.patches:
```

```
  ax.annotate("{:,.0f} \n({:,.2f}%)".format(p.get_height(), p.get_height() * 100/total_sum),
  (p.get_x() + p.get_width()/2, p.get_height()+20),ha = 'center', va = 'bottom')
plt.tight_layout()
plt.show()
```



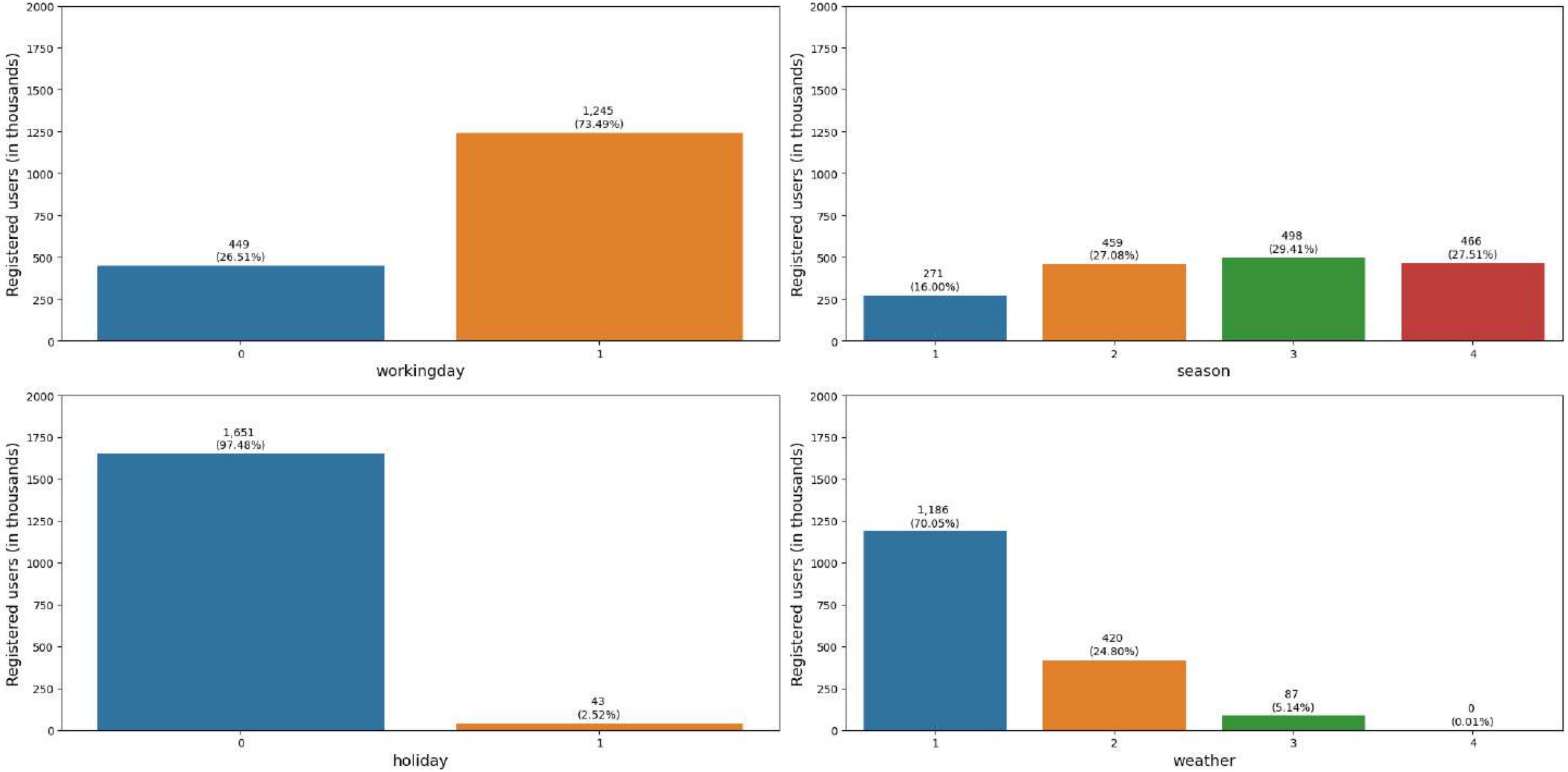
Observations:

- Approximately 81% of total users are belongs to registered category.
- Highest user counts occurs when humidity level in comfort zone and when windspeed is gentle to moderate zone.

In [ ]:
```
wd=df.groupby(["workingday"])["registered"].sum().reset_index()
wd["registered"]=wd["registered"]/1000
s=df.groupby(["season"])["registered"].sum().reset_index()
```

```
s["registered"]=s["registered"]/1000
hd=df.groupby(["holiday"])["registered"].sum().reset_index()
hd["registered"]=hd["registered"]/1000
w=df.groupby(["weather"])["registered"].sum().reset_index()
w["registered"]=w["registered"]/1000

total_sum=(df["registered"].sum())/1000
fig = plt.figure(figsize=(20,10))
plt.subplot(2, 2, 1)
ax = sns.barplot(data=wd,x='workingday',y='registered',orient='v')
plt.yticks(np.arange(0, 2200, 250))
plt.xlabel('workingday',fontsize=14)
plt.ylabel('Registered users (in thousands)',fontsize=14)
for p in ax.patches:
  ax.annotate("{:,.0f} \n({:,.2f}%)".format(p.get_height(), p.get_height() * 100/total_sum),
  (p.get_x() + p.get_width()/2, p.get_height()+20),ha = 'center', va = 'bottom')
plt.subplot(2, 2, 2)
ax = sns.barplot(data=s,x='season',y='registered',orient='v')
plt.yticks(np.arange(0, 2200, 250))
plt.xlabel('season',fontsize=14)
plt.ylabel('Registered users (in thousands)',fontsize=14)
for p in ax.patches:
  ax.annotate("{:,.0f} \n({:,.2f}%)".format(p.get_height(), p.get_height() * 100/total_sum),
  (p.get_x() + p.get_width()/2, p.get_height()+20),ha = 'center', va = 'bottom')
plt.subplot(2, 2, 3)
ax = sns.barplot(data=hd,x='holiday',y='registered',orient='v')
plt.yticks(np.arange(0, 2200, 250))
plt.xlabel('holiday',fontsize=14)
plt.ylabel('Registered users (in thousands)',fontsize=14)
for p in ax.patches:
  ax.annotate("{:,.0f} \n({:,.2f}%)".format(p.get_height(), p.get_height() * 100/total_sum),
  (p.get_x() + p.get_width()/2, p.get_height()+20),ha = 'center', va = 'bottom')
plt.subplot(2, 2, 4)
ax = sns.barplot(data=w,x='weather',y='registered',orient='v')
plt.yticks(np.arange(0, 2200, 250))
plt.xlabel('weather',fontsize=14)
plt.ylabel('Registered users (in thousands)',fontsize=14)
for p in ax.patches:
  ax.annotate("{:,.0f} \n({:,.2f}%)".format(p.get_height(), p.get_height() * 100/total_sum),
  (p.get_x() + p.get_width()/2, p.get_height()+20),ha = 'center', va = 'bottom')
plt.tight_layout()
plt.show()
```

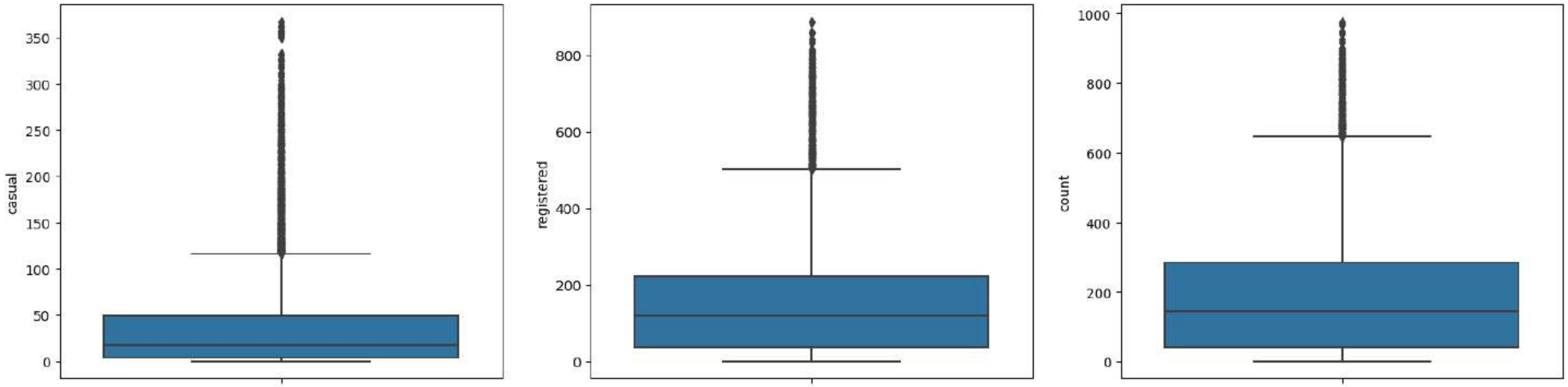


Observations:

- Total users counts for registerd category are 1693 thousands.
- 70% of the total count observed when weather is found clear.
- Total users count are more or less equally distributed across the all seasons.
- 98% of total users count are belongs to the normal days and only 74% instances belongs to working days.

Outlier Detection:

In [ ]:
```
# outlier detection
fig = plt.figure(figsize=(20,5))
plt.subplot(1, 3, 1)
sns.boxplot(data=df,y='casual')
plt.subplot(1, 3, 2)
sns.boxplot(data=df,y='registered')
plt.subplot(1, 3, 3)
sns.boxplot(data=df,y='count')
plt.show()
```

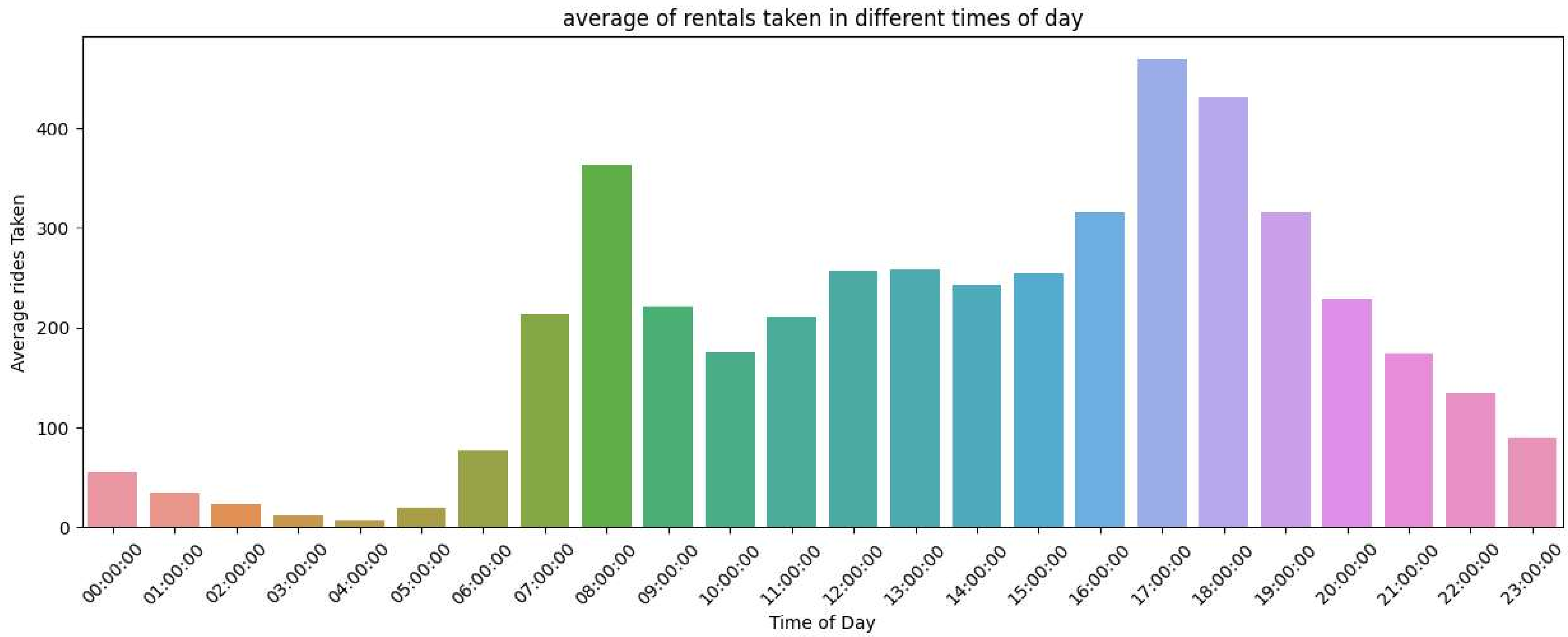


Observations:

- Outlier has been detected in all category of total users counts.

Bi-variate Analysis:

In [ ]:
```
# distribution of average total rentals based on the time of the day
data = df.groupby('time')['count'].mean().reset_index()
fig = plt.figure(figsize=(15,5))
sns.barplot(data=data,x='time', y='count')
plt.xticks(rotation = 45)
plt.xlabel('Time of Day')
plt.ylabel('Average rides Taken')
plt.title('average of rentals taken in different times of day')
plt.show()
```

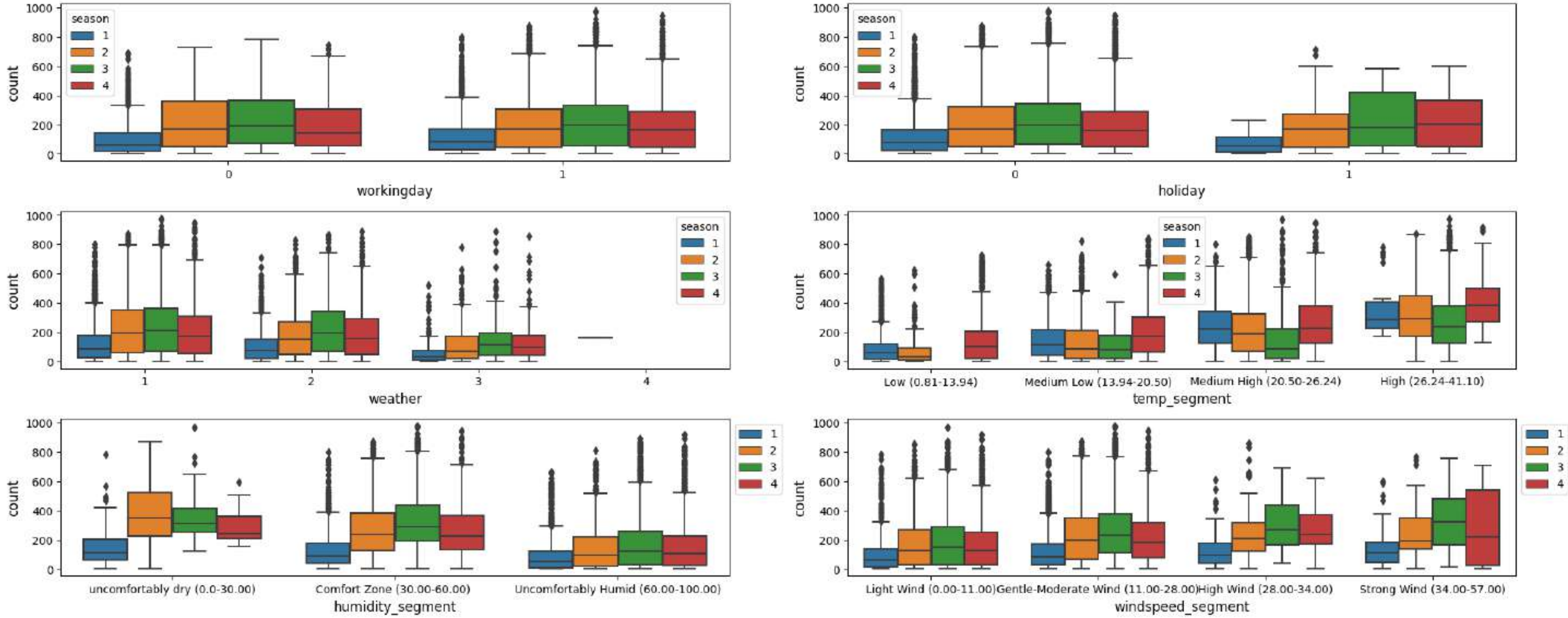


Observations:

- Least number of rentals were taken at 4:00 am.
- Maximum nmumber of rentals were taken at 5:00 pm in the evening.
- There are high number of rentals in morning around 8 am and 4-7 pm in the evening(Office timings).
- It signifies that people are using rentals for office commute.

In [ ]:
```
# before outlier removal
fig=plt.figure(figsize=(20,8))
plt.subplot(3,2,1)
sns.boxplot(data=df, x="workingday", y="count", hue="season")
plt.ylabel("count",fontsize=12)
plt.xlabel("workingday",fontsize=12)
plt.subplot(3,2,2)
sns.boxplot(data=df, x="holiday", y="count", hue="season")
plt.ylabel("count",fontsize=12)
```

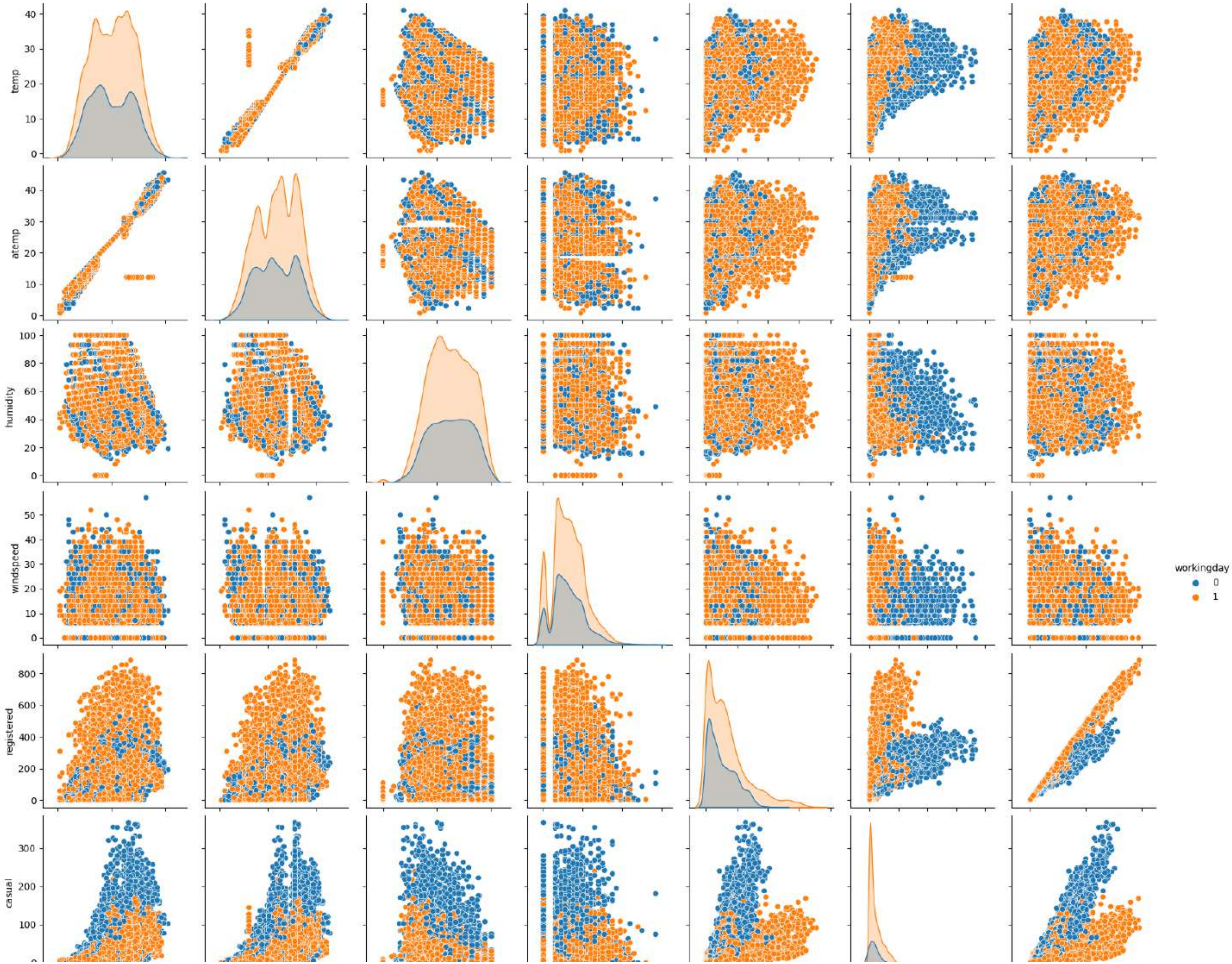
```
plt.xlabel("holiday",fontsize=12)
plt.subplot(3,2,3)
sns.boxplot(data=df, x="weather", y="count", hue="season")
plt.ylabel("count",fontsize=12)
plt.xlabel("weather",fontsize=12)
plt.subplot(3,2,4)
sns.boxplot(data=df, x="temp_segment", y="count", hue="season")
plt.ylabel("count",fontsize=12)
plt.xlabel("temp_segment",fontsize=12)
plt.subplot(3,2,5)
sns.boxplot(data=df, x="humidity_segment", y="count", hue="season")
plt.ylabel("count",fontsize=12)
plt.xlabel("humidity_segment",fontsize=12)
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.subplot(3,2,6)
sns.boxplot(data=df, x="windspeed_segment", y="count", hue="season")
plt.ylabel("count",fontsize=12)
plt.xlabel("windspeed_segment",fontsize=12)
plt.legend(bbox_to_anchor=(1.0, 1.0))
plt.tight_layout()
plt.show()
```

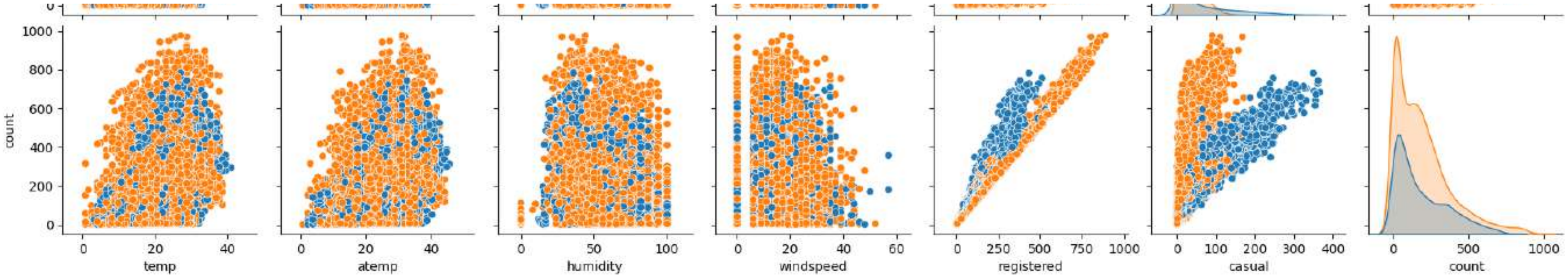


Observations:

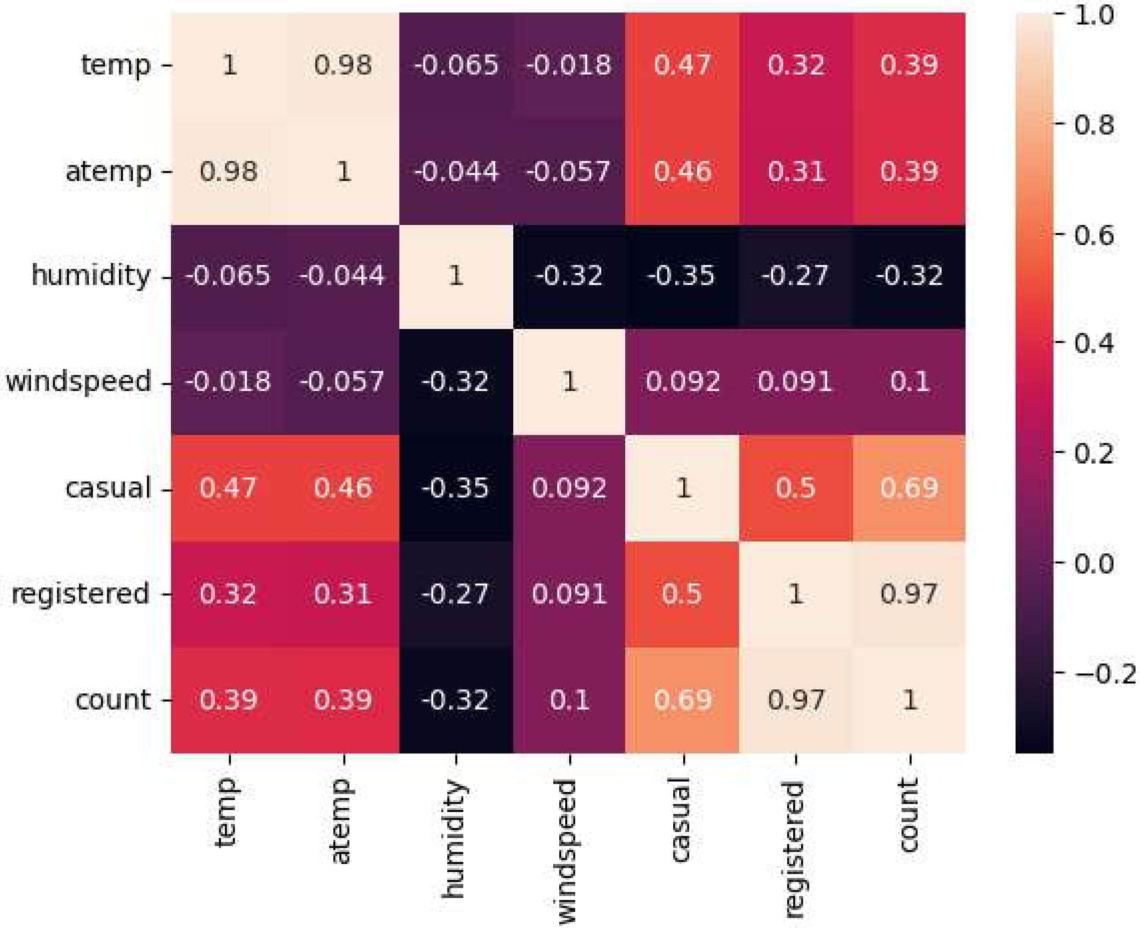
- As the season changes, the change has been observed across humidity level, in windspeed level and temperature level as well.

In [ ]:
```
sns.pairplot(df[['workingday','temp','atemp',
       'humidity', 'windspeed','registered','casual','count']],hue="workingday")
plt.show()
```


temp
atemp
humidity
windspeed
registered
casual
workingday
0
1



In [ ]:
```python
heat_map=sns.heatmap(df[['temp','atemp','humidity','windspeed','casual','registered','count']].corr(), annot=True)
```

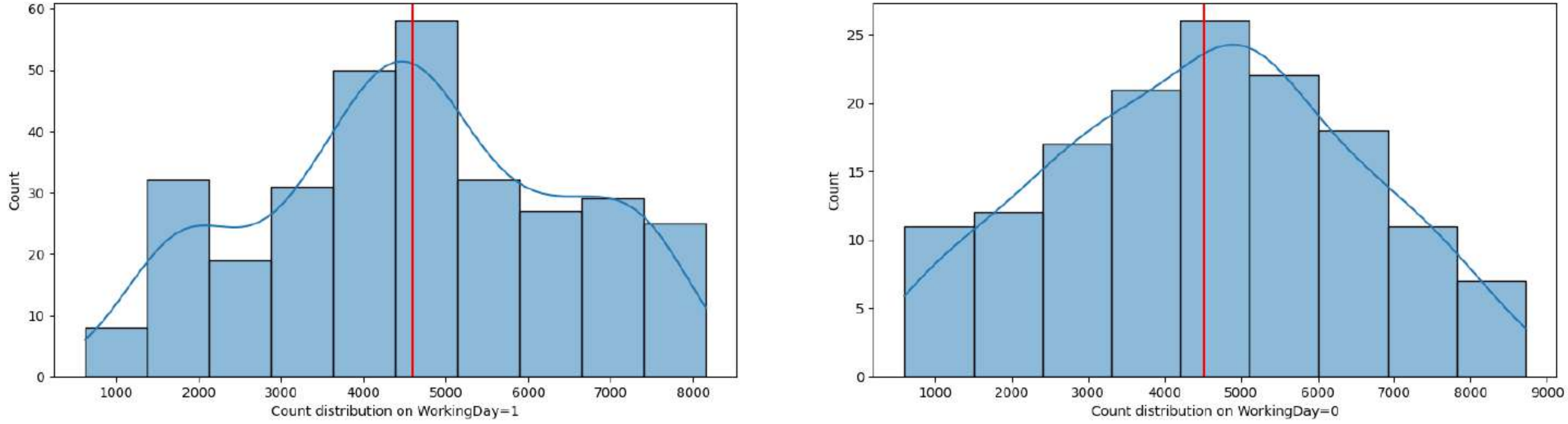


Observations:

1. Measured temperature and feeling temperature are highly correlated.
2. There is a decent correlation between casual riders and temperature.
3. Negative correlation coffecient shows the inverse relation between two variables.

Hypothesis Testing

In [ ]:
```
# visual representation for normality test
df_wd_1=df[df["workingday"]==1].groupby(df["date"])["count"].sum().reset_index()
df_wd_0=df[df["workingday"]==0].groupby(df["date"])["count"].sum().reset_index()
fig = plt.figure(figsize=(20,5))
plt.subplot(1, 2, 1)
sns.histplot(df_wd_1["count"],kde = True)
plt.axvline((df_wd_1["count"]).mean(),color="r")
plt.xlabel('Count distribution on WorkingDay=1',fontsize=10)
plt.subplot(1, 2, 2)
sns.histplot(df_wd_0["count"],kde = True)
plt.axvline((df_wd_0["count"]).mean(),color="r")
plt.xlabel('Count distribution on WorkingDay=0',fontsize=10)
plt.show()
```



Assumptions under t-test

1. Independence: The observations in one sample are independent of the observations in the other sample.
2. Normality: Both samples are approximately normally distributed.
3. Homogeneity of Variances: Both samples have approximately the same variance.
4. Random Sampling: Both samples were obtained using a random sampling method.

Validation of assumptions:

- Variance test using Levene Test
- shapiro test using to check normality of data

In [ ]:
```
# Validation of assumptions:
# Assumptions 1: sample df_wd_1 and df_wd_0 are independent sample.
# Assumptions 2:len(df_wd_1)>50,len(df_wd_0)>50 and
# Ho: sample following Gaussian distribution
# Ha: sample not following Gaussian distribution.
df_wd_1=df[df["workingday"]==1].groupby(df["date"])["count"].sum()
df_wd_0=df[df["workingday"]==0].groupby(df["date"])["count"].sum()
print("sample size df_wd_1 :",len(df_wd_1))
print("sample size df_wd_0:",len(df_wd_0))
alpha=0.05
shapiro_stat,p_value=shapiro(df_wd_1)
print("alpha:",0.05)
print("p_value:",p_value)
if p_value<alpha:
  print("Reject Null Hypothesis: sample not following Gaussian distribution")
else:
  print('Accept Null Hypothesis: sample following Gaussian distribution')
shapiro_stat,p_value=shapiro(df_wd_0)
print("alpha:",0.05)
print("p_value:",p_value)
if p_value<alpha:
  print("Reject Null Hypothesis: sample not following Gaussian distribution")
else:
  print('Accept Null Hypothesis: sample following Gaussian distribution')
# Assumption 3:
# H0:variance of the sample is same.
# Ha: variances of the sample is not same.
```

```
alpha=0.05
levene_stat,p_value=levene(df_wd_1,df_wd_0)
print("alpha:",0.05)
print("p_value:",p_value)
if p_value<alpha:
  print("Reject Null Hypothesis: Variance of the input datasets is not same")
else:
  print('Accept Null Hypothesis: Variance of the input datasets is Same/Close')
# Assumption 4: samples are random
```

```
sample size df_wd_1 : 311
sample size df_wd_0: 145
alpha: 0.05
p_value: 2.0652621969929896e-05
Reject Null Hypothesis: sample not following Gaussian distribution
alpha: 0.05
p_value: 0.0804058164358139
Accept Null Hypothesis: sample following Gaussian distribution
alpha: 0.05
p_value: 0.28003858261286085
Accept Null Hypothesis: Variance of the input datasets is Same/Close
```

Observations:

- For both sample, size of data is greater than 50.
- Sample data df_dw_1 **not** following Gaussian distribution.
- Sample data df_dw_0 is following Gaussian distribution.
- Variance of the both sample data is statistically **same**.

NOTE: Even our one or two Assumptions are failed still we will continue our analysis as per mentioned test in Problem Statement.

Hypothesis testing (2-Sample t-test):

- Hypothesis testing on Working Day, whether it has an effect on number of electric cycles rented or not
- **Null hypothesis**,Ho:mu_0=mu_1
- **Alternate hypothesis**,H1:mu_0<>mu_1

In [ ]:
```
# 2- Sample T-Test to check if Working Day has an effect on the number of electric cycles rented
df_wd_1=df[df["workingday"]==1].groupby(df["date"])["count"].sum()
df_wd_0=df[df["workingday"]==0].groupby(df["date"])["count"].sum()
def two_sample_t_test(CL):
  alpha=1-(CL/100) # significance level(alpha)
  t_critical_upper=t.ppf(1-(alpha/2),df=len(df_wd_1)+len(df_wd_0)-1)
  t_critical_lower=t.ppf(alpha/2,df=len(df_wd_1)+len(df_wd_0)-1)
  t_stat,p_value=ttest_ind(df_wd_1,df_wd_0,alternative="two-sided")
  print("Alpha:",alpha)
  print("p value:",p_value)
  print("t_critical_upper:",t_critical_upper)
  print("t_critical_lower:",t_critical_lower)
  print("t statistics:",t_stat)
  if p_value<alpha:
    print("Reject Null Hypothesis: Working Day has an effect on the number of electric cycles rented")
  else:
    print('Accept Null Hypothesis: Working Day has no effect on the number of electric cycles rented')
```

In [ ]:
```
# With confidence level of 95%
two_sample_t_test(95)
```

```
Alpha: 0.050000000000000044
p value: 0.656696335987859
t_critical_upper: 1.9651914282465222
t_critical_lower: -1.9651914282465222
t statistics: 0.44477221614881995
Accept Null Hypothesis: Working Day has no effect on the number of electric cycles rented
```

Observations:

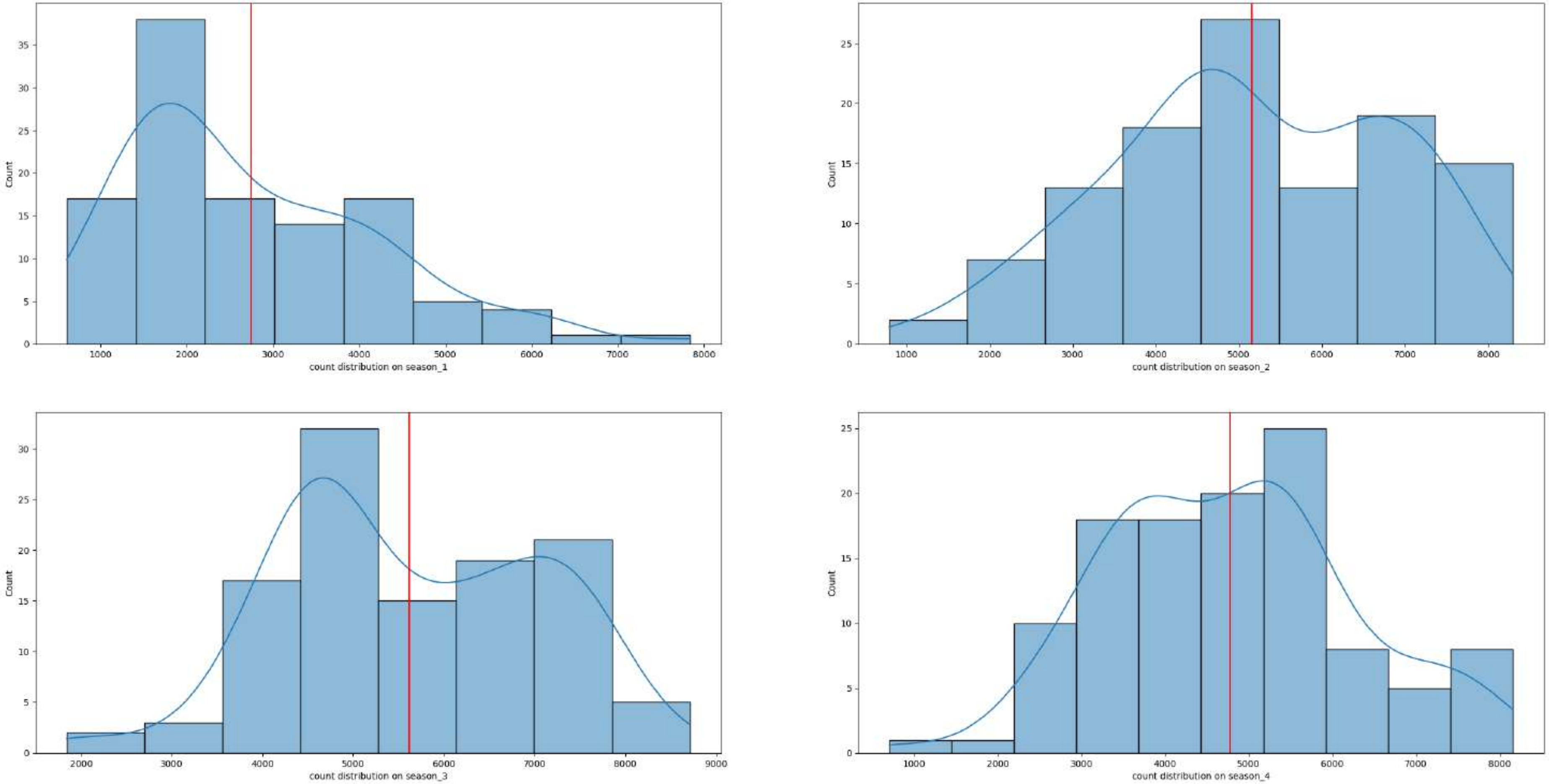
- Assumption-2(Normality): only 1 data set out of 2 follows normal distribution.
- Assumption-3(Variance): data sets pass the variance test.
- The calculated p-value : 0.6566
- which is greater than the significance level.
- Null Hypothesis is accepted.
- working day has no effect on rentals

Analysis of Seasonal rented cycles

(Part-1_Season) (ANOVA test)

In [ ]:

```
# visual representation for normality test
df_s_1=df[df["season"]==1].groupby(df["date"])["count"].sum().reset_index()
df_s_2=df[df["season"]==2].groupby(df["date"])["count"].sum().reset_index()
df_s_3=df[df["season"]==3].groupby(df["date"])["count"].sum().reset_index()
df_s_4=df[df["season"]==4].groupby(df["date"])["count"].sum().reset_index()
fig = plt.figure(figsize=(30,15))
plt.subplot(2, 2, 1)
sns.histplot(df_s_1["count"],kde = True)
plt.axvline((df_s_1["count"]).mean(),color="r")
plt.xlabel('count distribution on season_1',fontsize=10)
plt.subplot(2, 2, 2)
sns.histplot(df_s_2["count"],kde = True)
plt.axvline((df_s_2["count"]).mean(),color="r")
plt.xlabel('count distribution on season_2',fontsize=10)
plt.subplot(2, 2, 3)
sns.histplot(df_s_3["count"],kde = True)
plt.axvline((df_s_3["count"]).mean(),color="r")
plt.xlabel('count distribution on season_3',fontsize=10)
plt.subplot(2, 2, 4)
sns.histplot(df_s_4["count"],kde = True)
plt.axvline((df_s_4["count"]).mean(),color="r")
plt.xlabel('count distribution on season_4',fontsize=10)
plt.show()
```



Assumptions under ANOVA test:

1. The drawn samples are independent.
2. The responses for each factor level have a normal population distribution.
3. The varience of different samples are same.

Validation of assumptions:

- Variance test using Levene Test
- shapiro test using to check normality of data

In [ ]:
```
# Validation of assumptions:
# Assumptions 1: samples df_s_1,df_s_2,df_s_3 and df_s_4 are independent sample.
# Assumptions 2:The responses for each factor level have a normal population distribution.
# Null Hypothesis,Ho: sample is following normal population distribution
# Alternate Hypothesis,Ha: sample is not following normal population distribution.
df_s_1=df[df["season"]==1].groupby(df["date"])["count"].sum()
df_s_2=df[df["season"]==2].groupby(df["date"])["count"].sum()
df_s_3=df[df["season"]==3].groupby(df["date"])["count"].sum()
df_s_4=df[df["season"]==4].groupby(df["date"])["count"].sum()
print("sample size df_s_1 :",len(df_s_1))
print("sample size df_s_2:",len(df_s_2))
print("sample size df_s_3 :",len(df_s_3))
print("sample size df_s_4:",len(df_s_4))
alpha=0.05
shapiro_stat,p_value=shapiro(df_s_1)
print("alpha:",0.05)
print("p_value_df_s_1:",p_value)
if p_value<alpha:
  print("Reject Null Hypothesis: sample not following normal population distribution")
else:
  print('Accept Null Hypothesis: sample following normal population distribution')
shapiro_stat,p_value=shapiro(df_s_2)
print("alpha:",0.05)
print("p_value_df_s_2:",p_value)
if p_value<alpha:
  print("Reject Null Hypothesis: sample not following normal population distribution")
else:
  print('Accept Null Hypothesis: sample following normal population distribution')
shapiro_stat,p_value=shapiro(df_s_3)
print("alpha:",0.05)
print("p_value_df_s_3:",p_value)
if p_value<alpha:
  print("Reject Null Hypothesis: sample not following normal population distribution")
else:
  print('Accept Null Hypothesis: sample following normal population distribution')
shapiro_stat,p_value=shapiro(df_s_4)
print("alpha:",0.05)
print("p_value_df_s_4:",p_value)
if p_value<alpha:
  print("Reject Null Hypothesis: sample not following normal population distribution")
else:
  print('Accept Null Hypothesis: sample following normal population distribution')
```

```
# Assumption 3: The varience of different samples are same
# Null Hypothesis,Ho:variance of the samples are same.
# Alternate Hypothesis,Ha: variances of the samples are not same.
alpha=0.05
levene_stat,p_value=levene(df_s_1,df_s_2,df_s_3,df_s_4)
print("alpha:",0.05)
print("p_value_levene:",p_value)
if p_value<alpha:
  print("Reject Null Hypothesis: Variance of the input datasets is not same")
else:
  print('Accept Null Hypothesis: Variance of the input datasets is Same/Close')
```

```
sample size df_s_1 : 114
sample size df_s_2: 114
sample size df_s_3 : 114
sample size df_s_4: 114
alpha: 0.05
p_value_df_s_1: 1.4321534763439558e-05
Reject Null Hypothesis: sample not following normal population distribution
alpha: 0.05
p_value_df_s_2: 0.032791439443826675
Reject Null Hypothesis: sample not following normal population distribution
alpha: 0.05
p_value_df_s_3: 0.003765953006222844
Reject Null Hypothesis: sample not following normal population distribution
alpha: 0.05
p_value_df_s_4: 0.17639805376529694
Accept Null Hypothesis: sample following normal population distribution
alpha: 0.05
p_value_levene: 0.21194448921499898
Accept Null Hypothesis: Variance of the input datasets is Same/Close
```

NOTE: Even our one or two Assumptions are failed still we will continue our analysis as per mentioned test in Problem Statement.

Hypothesis testing (ANOVA Test):

- Hypothesis testing on ANOVA to check if No. of cycles rented is similar or different in different season
- **Null hypothesis**,Ho:Avg.number of total rented cycles in different season is same (mu_1=mu_2=mu_3=mu_4)
- **Alternate hypothesis**,Ha:At least in one of season, the avg. number of total rented cycles is different.

In [112…

```
# Null hypothesis testing on ANNOVA to check if No. of cycles rented is similar or different in different season
# Null hypothesis,Ho:Avg.number of total rented bike in different season is same (mu_1=mu_2=mu_3=mu_4)
# Alternate hypothesis,Ha:At least in one of season,the avg. number of total rented bike is different.
df_s_1=df[df["season"]==1].groupby(df["date"])["count"].sum()
df_s_2=df[df["season"]==2].groupby(df["date"])["count"].sum()
df_s_3=df[df["season"]==3].groupby(df["date"])["count"].sum()
df_s_4=df[df["season"]==4].groupby(df["date"])["count"].sum()
def Anova_test_s(CL):
  alpha=1-(CL/100) # significance level(alpha)
  number_of_sample=df["season"].nunique()
  dfn=number_of_sample-1
  N=len(df_s_1)+len(df_s_2)+len(df_s_3)+len(df_s_4)
  dfd=N-number_of_sample
  f_critical=f.ppf(1-alpha,dfn,dfd)
  f_stat,p_value=f_oneway(df_s_1,df_s_2,df_s_3,df_s_4)
  print("Alpha:",alpha)
  print("p value:",p_value)
  print("f_critical:",f_critical)
  print("f statistics:",f_stat)
  if p_value<alpha:
    print("Reject Null Hypothesis: Season has an effect on the number of electric cycles rented")
  else:
    print('Accept Null Hypothesis: Season has no effect on the number of electric cycles rented')
```

In [113…

```
# With confidence level of 95%
Anova_test_s(95)
```

```
Alpha: 0.050000000000000044
p value: 1.506580502991204e-41
f_critical: 2.6246364471959573
f statistics: 80.0504789788067
Reject Null Hypothesis: Season has an effect on the number of electric cycles rented
```

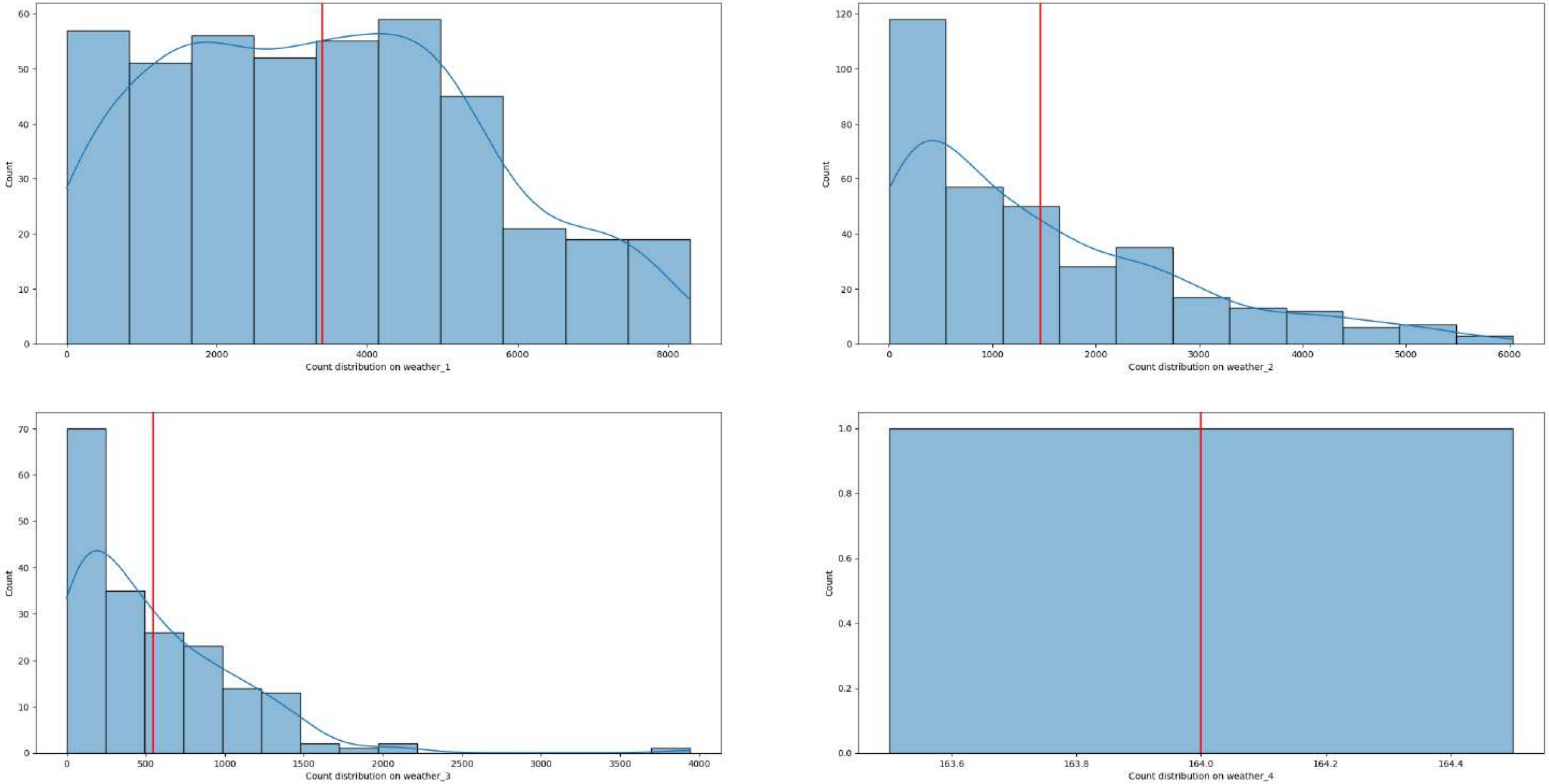
Observations:

- Assumption-2(Normality): only 1 data set out of 4 follows normal distribution.
- Assumption-3(Variance): data sets pass the variance test.
- The calculated p-value : 1.506580502991204e-41 which is less than the significance level(0.05).
- Null Hypothesis is rejected.
- Season has an effect on the number of electric cycles rented

Analysis of weather-wise rented cycles

(Part-2_Weather) (ANOVA test)

In [ ]:
```
# visual representation for normality test
df_w_1=df[df["weather"]==1].groupby(df["date"])["count"].sum().reset_index()
df_w_2=df[df["weather"]==2].groupby(df["date"])["count"].sum().reset_index()
df_w_3=df[df["weather"]==3].groupby(df["date"])["count"].sum().reset_index()
df_w_4=df[df["weather"]==4].groupby(df["date"])["count"].sum().reset_index()
fig = plt.figure(figsize=(30,15))
plt.subplot(2, 2, 1)
sns.histplot(df_w_1["count"],kde = True)
plt.axvline((df_w_1["count"]).mean(),color="r")
plt.xlabel('Count distribution on weather_1',fontsize=10)
plt.subplot(2, 2, 2)
sns.histplot(df_w_2["count"],kde = True)
plt.axvline((df_w_2["count"]).mean(),color="r")
plt.xlabel('Count distribution on weather_2',fontsize=10)
plt.subplot(2, 2, 3)
sns.histplot(df_w_3["count"],kde = True)
plt.axvline((df_w_3["count"]).mean(),color="r")
plt.xlabel('Count distribution on weather_3',fontsize=10)
plt.subplot(2, 2, 4)
sns.histplot(df_w_4["count"],kde = True)
plt.axvline((df_w_4["count"]).mean(),color="r")
plt.xlabel('Count distribution on weather_4',fontsize=10)
plt.show()
```



Assumptions under ANOVA test:

1. The drawn samples are independent.
2. The responses for each factor level have a normal population distribution.
3. The varience of different samples are same.

Validation of assumptions:

- Variance test using Levene Test
- shapiro test using to check normality of data

In [ ]:
```
# Validation of assumptions:
# Assumptions 1: samples df_w_1,df_w_2,df_w_3 and df_w_4 are independent sample.
# Assumptions 2:The responses for each factor level have a normal population distribution.
# Null Hypothesis,Ho: sample is following normal population distribution
# Alternate Hypothesis,Ha: sample is not following normal population distribution.
df_w_1=df[df["weather"]==1].groupby(df["date"])["count"].sum()
df_w_2=df[df["weather"]==2].groupby(df["date"])["count"].sum()
df_w_3=df[df["weather"]==3].groupby(df["date"])["count"].sum()
df_w_4=df[df["weather"]==4].groupby(df["date"])["count"].sum()
print("sample size df_w_1 :",len(df_w_1))
print("sample size df_w_2:",len(df_w_2))
print("sample size df_w_3 :",len(df_w_3))
print("sample size df_w_4:",len(df_w_4))
alpha=0.05
shapiro_stat,p_value=shapiro(df_w_1)
print("alpha:",0.05)
print("p_value_df_w_1:",p_value)
if p_value<alpha:
  print("Reject Null Hypothesis: sample not following normal population distribution")
else:
  print('Accept Null Hypothesis: sample following normal population distribution')
shapiro_stat,p_value=shapiro(df_w_2)
print("alpha:",0.05)
print("p_value_df_w_2:",p_value)
if p_value<alpha:
  print("Reject Null Hypothesis: sample not following normal population distribution")
else:
  print('Accept Null Hypothesis: sample following normal population distribution')
shapiro_stat,p_value=shapiro(df_w_3)
print("alpha:",0.05)
print("p_value_df_w_3:",p_value)
if p_value<alpha:
  print("Reject Null Hypothesis: sample not following normal population distribution")
else:
  print('Accept Null Hypothesis: sample following normal population distribution')
# shapiro_stat,p_value=shapiro(df_w_4)
# shapiro_test fails,because data must be at least length 3
# Assumption 3: The varience of different samples are same
# Null Hypothesis,Ho:variance of the samples are same.
# Alternate Hypothesis,Ha: variances of the samples are not same.
alpha=0.05
levene_stat,p_value=levene(df_w_1,df_w_2,df_w_3)
```

```
print("alpha:",0.05)
print("p_value_levene:",p_value)
if p_value<alpha:
  print("Reject Null Hypothesis: Variance of the input datasets is not same")
else:
  print('Accept Null Hypothesis: Variance of the input datasets is Same/Close')
```

```
sample size df_w_1 : 434
sample size df_w_2: 346
sample size df_w_3 : 187
sample size df_w_4: 1
alpha: 0.05
p_value_df_w_1: 1.1694455537281101e-07
Reject Null Hypothesis: sample not following normal population distribution
alpha: 0.05
p_value_df_w_2: 6.997944805825062e-16
Reject Null Hypothesis: sample not following normal population distribution
alpha: 0.05
p_value_df_w_3: 3.36847439047841e-13
Reject Null Hypothesis: sample not following normal population distribution
alpha: 0.05
p_value_levene: 6.545722170352777e-54
Reject Null Hypothesis: Variance of the input datasets is not same
```

NOTE: Even our one or two Assumptions are failed still we will continue our analysis as per mentioned test in Problem Statement.

Hypothesis testing (ANOVA Test):

- Null hypothesis testing on ANOVA to check if No. of cycles rented is similar or different in different weather
- **Null hypothesis**,Ho:Avg.number of total rented bike in different weather is same (mu_1=mu_2=mu_3=mu_4)
- **Alternate hypothesis**,Ha:At least in one of weather, the avg. number of total rented cycles is different.

In [116…

```
# Null hypothesis testing on ANNOVA to check if No. of cycles rented is similar or different in different weather
# Null hypothesis,Ho:Avg.number of total rented cycles in different weather is same (mu_1=mu_2=mu_3=mu_4)
# Alternate hypothesis,Ha:At least in one of weather,the avg. number of total rented cycles is different.
df_w_1=df[df["weather"]==1].groupby(df["date"])["count"].sum()
df_w_2=df[df["weather"]==2].groupby(df["date"])["count"].sum()
df_w_3=df[df["weather"]==3].groupby(df["date"])["count"].sum()
def Anova_test_w(CL):
```

```
  alpha=1-(CL/100) # significance level(alpha)
  number_of_sample=df["weather"].nunique()
  dfn=number_of_sample-1
  N=len(df_w_1)+len(df_w_2)+len(df_w_3)
  dfd=N-number_of_sample
  f_critical=f.ppf(1-alpha,dfn,dfd)
  f_stat,p_value=f_oneway(df_w_1,df_w_2,df_w_3)
  print("Alpha:",alpha)
  print("p value:",p_value)
  print("f_critical:",f_critical)
  print("f statistics:",f_stat)
  if p_value<alpha:
    print("Reject Null Hypothesis: weather has an effect on the number of electric cycles rented")
  else:
    print('Accept Null Hypothesis: weather has no effect on the number of electric cycles rented')
```

In [117…

```
# With confidence level of 95%
Anova_test_w(95)
```

```
Alpha: 0.050000000000000044
p value: 1.0951526874744494e-86
f_critical: 2.614146048649389
f statistics: 244.7555835815733
Reject Null Hypothesis: weather has an effect on the number of electric cycles rented
```

Observations:

- Assumption-2(Normality): Not any data set out of 3 follows normal distribution.
- Assumption-3(Variance): data sets also fail in the the variance test.
- The calculated p-value : 1.0951526874744494e-86 which is less than the significance level(0.05).
- Null Hypothesis is rejected.
- Weather has an effect on the number of electric cycles rented

Analysis of weather vs seasonal rented cycles

Assumptions under Chi-Square test:

1. Both variables are categorical.

2. All observations are independent.
3. Cells in the contingency table are mutually exclusive.
4. Expected value of cells should be 5 or greater in at least 80% of cells.

Validation of assumptions:

In [ ]:
```
# Assumption 1: Both variables are categorical.
# season and weather both are categorical variable.
# Assumption 2: All observations are independent.
# data under all observations are independent.
# Assumption 3: Cells in the contingency table are mutually exclusive.
# Cells in the contingency table are mutually exclusive
# Assumption 4: Expected value of cells should be 5 or greater in at least 80% of cells.
observed=pd.crosstab(index=df["season"], columns=df[df["weather"]!=4]["weather"])
print(observed)
# it has been noticed that under weather category(4) number of observations for all category of season(1,2,3,4) is less than 5
# which violates Assumption 4.There we will drop that column from observed table.
```

```
weather     1    2    3
season
1        1759  715  211
2        1801  708  224
3        1930  604  199
4        1702  807  225
```

Hypothesis testing (Chi-Square test):

- **Null Hypothesis,H_0:** The two categorical variables season and weather are independent of each other.
- **Alternate Hypothesis,Ha:** The two categorical variables season and weather are dependent on each other.

In [ ]:
```
# creating contingency table for Observed values.
observed=pd.crosstab(index=df["season"], columns=df[df["weather"]!=4]["weather"])
def chi2_test(CL):
  alpha=1-(CL/100) # significance level(alpha)
  chi_stat, p_value, dof, expected = chi2_contingency(observed)
  chi2_critical=chi2.ppf(1-alpha,df=dof)
  print("Alpha:",alpha)
  print("p value:",p_value)
  print("degree of freedom:",dof)
```

```
    print("chi statistics:",chi_stat)
    print("chi2 critical:",chi2_critical)
    print("observed:",observed)
    print("expected:",expected)
    if p_value<alpha:
      print("Reject Null Hypothesis: The two categorical variables season and weather are dependent on each other")
    else:
      print('Accept Null Hypothesis: The two categorical variables season and weather are independent of each other')
```

In [ ]:

```
# With confidence level of 95%
chi2_test(95)
```

```
Alpha: 0.050000000000000044
p value: 2.8260014509929343e-08
degree of freedom: 6
chi statistics: 46.10145731073249
chi2 critical: 12.591587243743977
observed: weather     1    2    3
season
1          1759  715  211
2          1801  708  224
3          1930  604  199
4          1702  807  225
expected: [[1774.04869086  699.06201194  211.8892972 ]
 [1805.76352779  711.55920992  215.67726229]
 [1805.76352779  711.55920992  215.67726229]
 [1806.42425356  711.81956821  215.75617823]]
Reject Null Hypothesis: The two categorical variables season and weather are dependent on each other
```

Observations:

- weather = 4 is dropped because of only 1 data point
- selected sample data set satisfying all assumptions.
- The calculated p-value : 2.8260014509929343e-08 which is less than the significance level(0.05).
- Null Hypothesis is rejected.
- The two categorical variables season and weather are dependent on each other

Business Insights:

1. Users are more happy to preferred rental bikes, whenever humidity level is in comfort zone(30-60%).
2. Users are not happy to preferred rental bikes when windspeed is greater than the 28 m/s.
3. Summer and Fall season see high number of rental bikes used by Users.
4. On holidays and non working days casual customers are ready to rent more bikes.
5. Under bad weather conditions the number of rentals decreases.
6. Under low temperature conditions also the number of rentals decreases.
7. Average rental is high in the office hours in morning(08:00 AM) and evening (04:00-07:00 PM).
8. Based on 2 sample t-test: the working Day has no effect on rentals.
9. Based on ANOVA test: Weather and Seasons having direct dependency on rentals.
10. Based on chi2 test: Weather and Seasons having direct dependency on each other.In every season there are different type of weather.

Recommendations:

1. They must focus on technological advancement in rental bike in order to control availabilty of bikes as per weather condition.
2. Right forecasting about weather condition help us to building healthy customer relationship.
3. Increase the availability of bikes in peak hours.
4. Maintenance of the bikes should be done in late hours.
5. Adopt more features while manufacturing of bikes such as higher speed, more convenience, effortless driving, and variable motor power per road conditions just to attract more customer sharing purposes even in high wind and strong wind condition.