YULU CASE STUDY

Import the dataset and do usual data analysis steps like checking the structure & characteristics of the dataset

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import ttest_ind, f_oneway, chi2_contingency
```

In [2]:
```
df=pd.read_csv(r'\Users\Home\Downloads\yulu_data.csv')
df
```

Out[2]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0000	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

1. Defining Problem Statement and Analysing basic metrics

Yulu is India's leading micro-mobility service provider, which offers unique vehicles for the daily commute through a user-friendly mobile app to enable shared, solo and sustainable commuting.Yulu zones are located at all the appropriate locations (including metro stations, bus stands, office spaces, residential areas, corporate offices, etc) to make those first and last miles smooth, affordable, and convenient!

Yulu has recently suffered considerable dips in its revenues. This case study aims to understand the factors on which the demand for these shared electric cycles depends in the Indian market.

The present case study aims at analyzing the data by setting appropriate hypothesis and using different types of hypothesis testing to establish a relationship between different variables (dependent and independent) in order to find out:

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

In [3]:
```
df.shape  #shape of data
```

Out[3]:
```
(10886, 12)
```

In [4]:
```
df.info()  #data type of all attributes
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   datetime    10886 non-null  object
 1   season      10886 non-null  int64
 2   holiday     10886 non-null  int64
 3   workingday  10886 non-null  int64
 4   weather     10886 non-null  int64
 5   temp        10886 non-null  float64
 6   atemp       10886 non-null  float64
 7   humidity    10886 non-null  int64
 8   windspeed   10886 non-null  float64
 9   casual      10886 non-null  int64
 10  registered  10886 non-null  int64
 11  count       10886 non-null  int64
dtypes: float64(3), int64(8), object(1)
memory usage: 1020.7+ KB
```

In [5]:
```
df.describe(include='all')  #statistical summary
```

Out[5]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
count	10886	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000
unique	10886	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
top	2011-01-01 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
freq	1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
mean	NaN	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.886460	12.799395	36.021955	155.552177	191.574132
std	NaN	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.245033	8.164537	49.960477	151.039033	181.144454
min	NaN	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	NaN	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	47.000000	7.001500	4.000000	36.000000	42.000000
50%	NaN	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	62.000000	12.998000	17.000000	118.000000	145.000000
75%	NaN	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	77.000000	16.997900	49.000000	222.000000	284.000000
max	NaN	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	100.000000	56.996900	367.000000	886.000000	977.000000

In [6]:
```
df.isna().sum()  #checking for null values
```

Out[6]:
```
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
```

Non-Graphical Analysis: Value counts and unique attributes

In [7]:
```
df.datetime.value_counts()
```

Out[7]:
```
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: datetime, Length: 10886, dtype: int64
```

In [8]:
```
df.season.value_counts()
```

Out[8]:
```
4    2734
2    2733
3    2733
1    2686
Name: season, dtype: int64
```

In [9]:
```
df.holiday.value_counts()
```

Out[9]:
```
0    10575
1      311
Name: holiday, dtype: int64
```

In [10]:
```
df.workingday.value_counts()
```

Out[10]:
```
1    7412
0    3474
Name: workingday, dtype: int64
```

In [11]:
```
df.weather.value_counts()
```

Out[11]:
```
1    7192
2    2834
3     859
4       1
Name: weather, dtype: int64
```

In [12]:
```
df.temp.value_counts()
```

Out[12]:

14.76	467
26.24	453
28.70	427
13.94	413
18.86	406
22.14	403
25.42	403
16.40	400
22.96	395
27.06	394
24.60	390
12.30	385
21.32	362
17.22	356
13.12	356
29.52	353
10.66	332
18.04	328
20.50	327
30.34	299
9.84	294
15.58	255
9.02	248
31.16	242
8.20	229
27.88	224
23.78	203
32.80	202
11.48	181
19.68	170
6.56	146
33.62	130
5.74	107
7.38	106
31.98	98
34.44	80
35.26	76
4.92	60
36.90	46
4.10	44
37.72	34
36.08	23
3.28	11
0.82	7
38.54	7

```
39.36      6
2.46       5
1.64       2
41.00      1
Name: temp, dtype: int64
```

In [13]:
```
df.atemp.value_counts()
```

Out[13]:

31.060	671
25.760	423
22.725	406
20.455	400
26.515	395
16.665	381
25.000	365
33.335	364
21.210	356
30.305	350
15.150	338
21.970	328
24.240	327
17.425	314
31.820	299
34.850	283
27.275	282
32.575	272
11.365	271
14.395	269
29.545	257
19.695	255
15.910	254
12.880	247
13.635	237
34.090	224
12.120	195
28.790	175
23.485	170
10.605	166
35.605	159
9.850	127
18.180	123
36.365	123
37.120	118
9.090	107
37.880	97
28.030	80
7.575	75
38.635	74
6.060	73
39.395	67
6.820	63
8.335	63
18.940	45

```
40.150     45
40.910     39
5.305      25
42.425     24
41.665     23
3.790      16
4.545      11
3.030       7
43.940      7
2.275       7
43.180      7
44.695      3
0.760       2
1.515       1
45.455      1
Name: atemp, dtype: int64
```

In [14]:
```
df.humidity.value_counts()
```

Out[14]:
```
88    368
94    324
83    316
87    289
70    259
     ...
8       1
10      1
97      1
96      1
91      1
Name: humidity, Length: 89, dtype: int64
```

In [15]:
```
df.windspeed.value_counts()
```

Out[15]:
```
0.0000     1313
8.9981     1120
11.0014    1057
12.9980    1042
7.0015     1034
15.0013     961
6.0032      872
16.9979     824
19.0012     676
19.9995     492
22.0028     372
23.9994     274
26.0027     235
27.9993     187
30.0026     111
31.0009      89
32.9975      80
35.0008      58
39.0007      27
36.9974      22
43.0006      12
40.9973      11
43.9989       8
46.0022       3
56.9969       2
47.9988       2
51.9987       1
50.0021       1
Name: windspeed, dtype: int64
```

In [16]:
```
df.casual.value_counts()
```

Out[16]:
```
0      986
1      667
2      487
3      438
4      354
      ...
332      1
361      1
356      1
331      1
304      1
Name: casual, Length: 309, dtype: int64
```

In [17]:
```
df.registered.value_counts()
```

Out[17]:
```
3      195
4      190
5      177
6      155
2      150
      ...
570      1
422      1
678      1
565      1
636      1
Name: registered, Length: 731, dtype: int64
```

In [18]:
```
df['count'].value_counts()
```

Out[18]:
```
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
```

Visual Analysis - Univariate & Bivariate

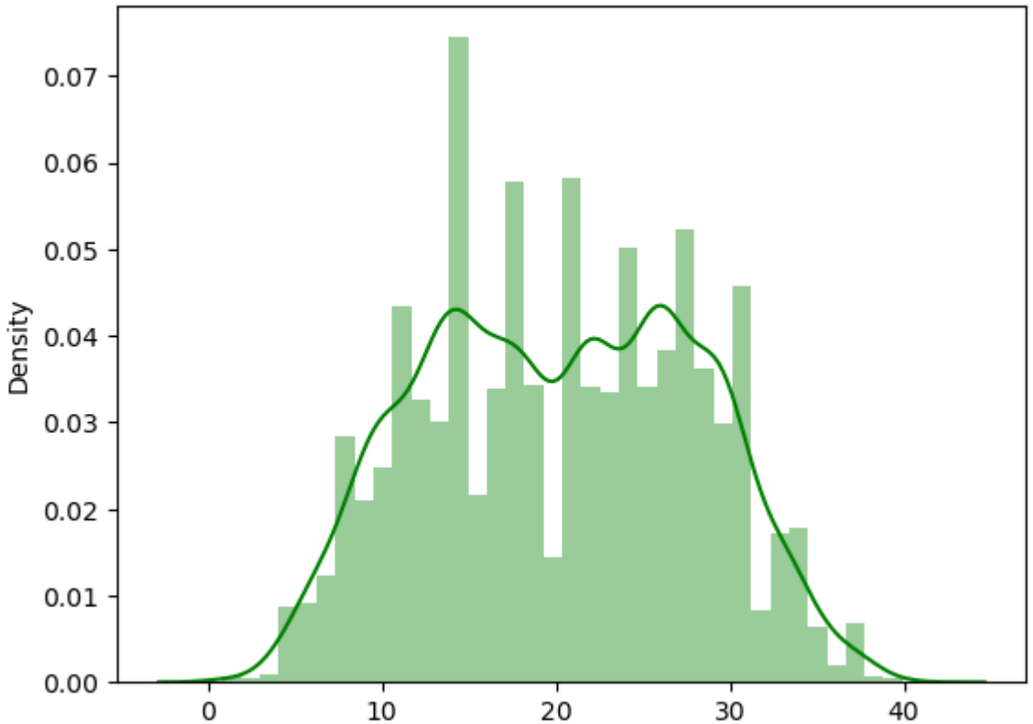
Univariate Analysis

For continuous variable(s)

In [19]:
```
sns.distplot(x=df['temp'],color="Green")
```

```
C:\Users\Home\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[19]:
```
<AxesSubplot:ylabel='Density'>
```

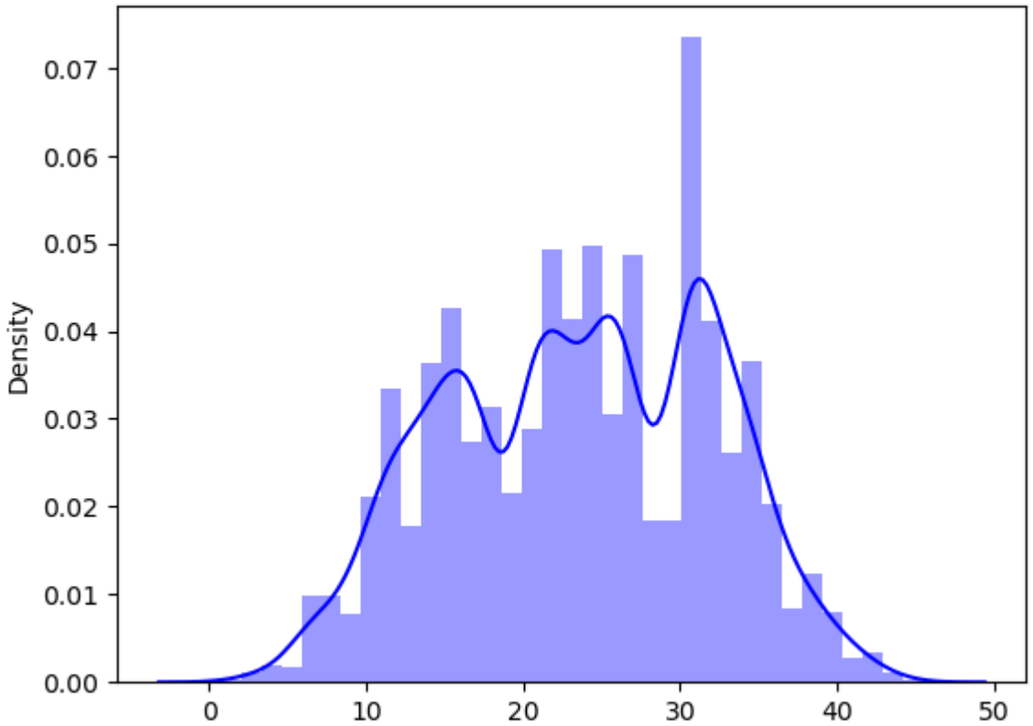


From the above, we can observe that the data recorded of temperature is gaussian.

In [20]:
```
sns.distplot(x=df['atemp'],color="Blue")
```

```
C:\Users\Home\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[20]:
```
<AxesSubplot:ylabel='Density'>
```

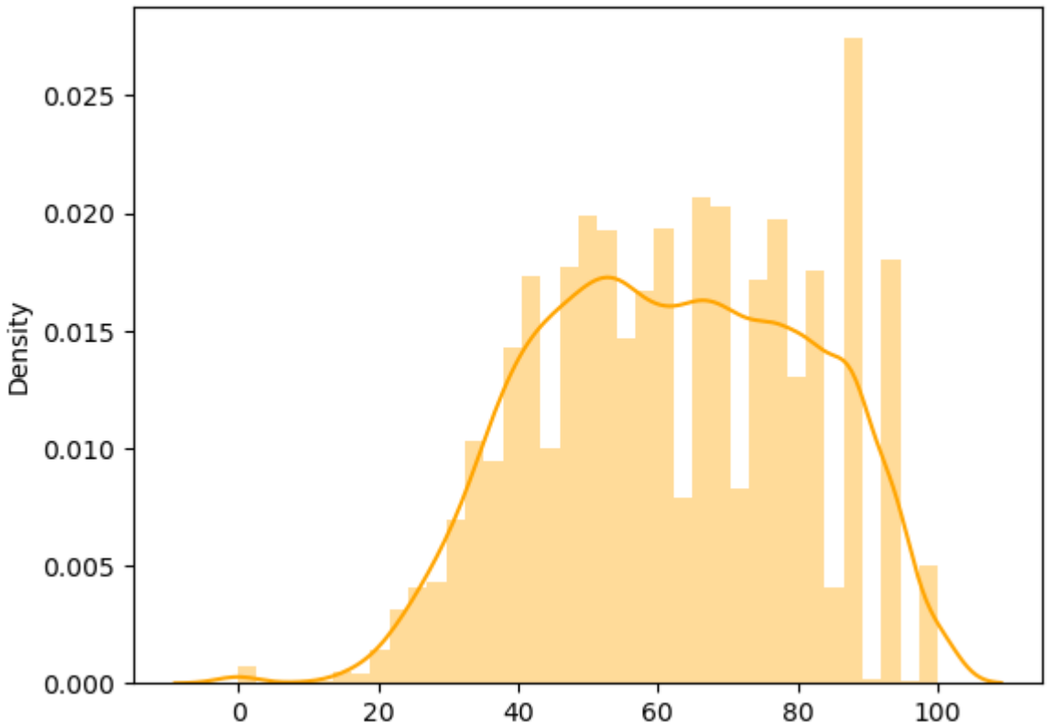


From the above, we can observe that the data recorded of feeling temperature is gaussian.

In [21]:
```
sns.distplot(x=df['humidity'],color="Orange")
```

```
C:\Users\Home\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future ver
sion. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[21]:
```
<AxesSubplot:ylabel='Density'>
```

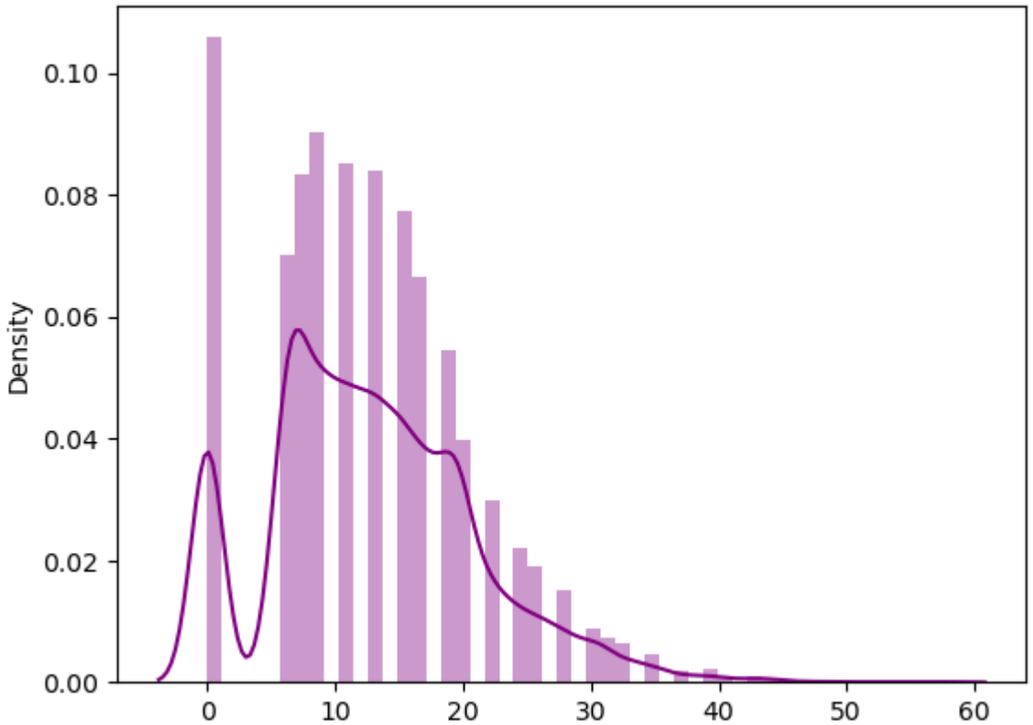


From the above, we can observe that the data recorded of humidity is gaussian.

In [22]:
```
sns.distplot(x=df['windspeed'],color="Purple")
```

```
C:\Users\Home\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[22]:
```
<AxesSubplot:ylabel='Density'>
```

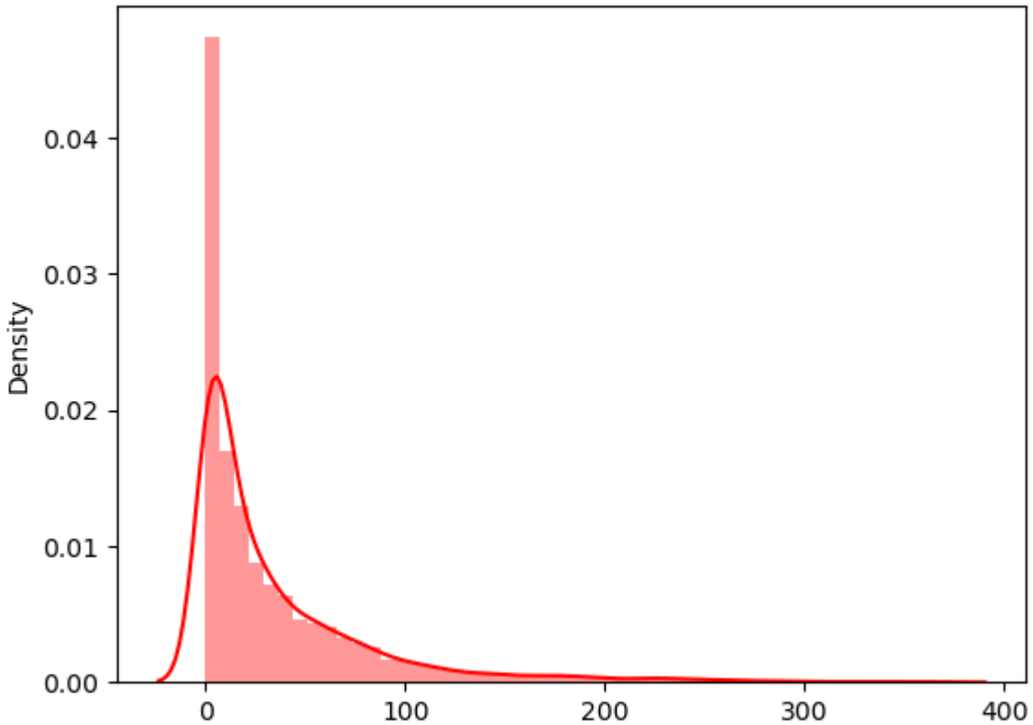


From the above, we can observe that the data recorded of windspeed is right skewed.

In [23]:
```
sns.distplot(x=df['casual'],color="Red")
```

```
C:\Users\Home\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future ver
sion. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[23]:
```
<AxesSubplot:ylabel='Density'>
```

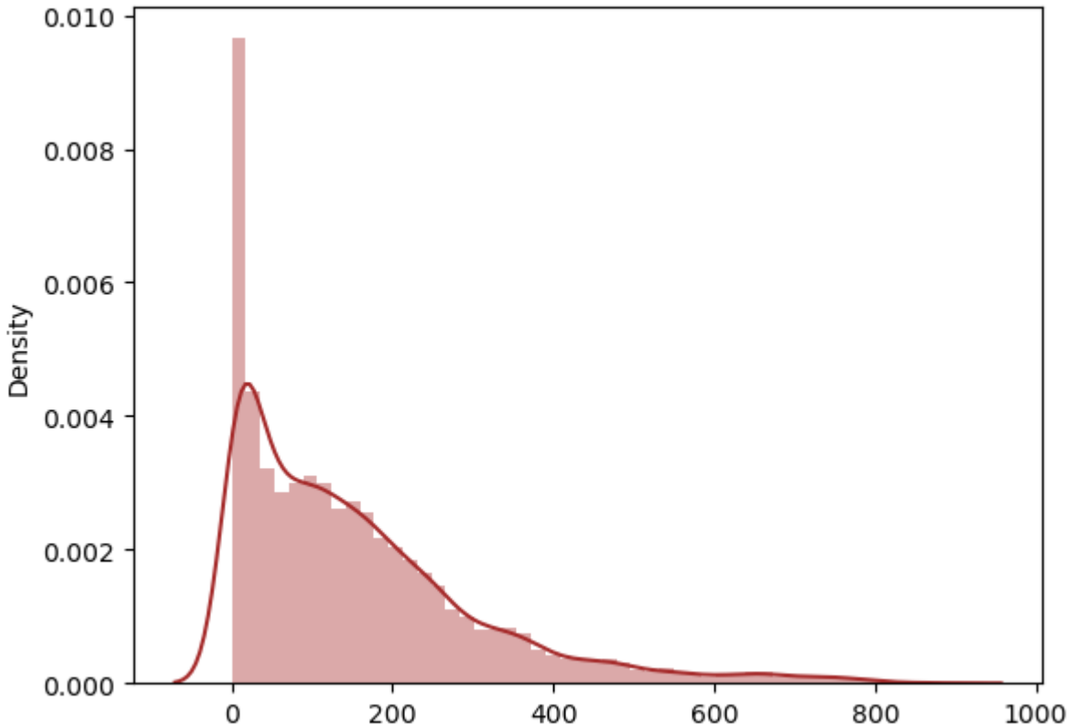


From the above, we can observe that the data recorded of casual users is right skewed.

In [24]:
```
sns.distplot(x=df['registered'],color="Brown")
```

```
C:\Users\Home\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future ver
sion. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[24]:
```
<AxesSubplot:ylabel='Density'>
```

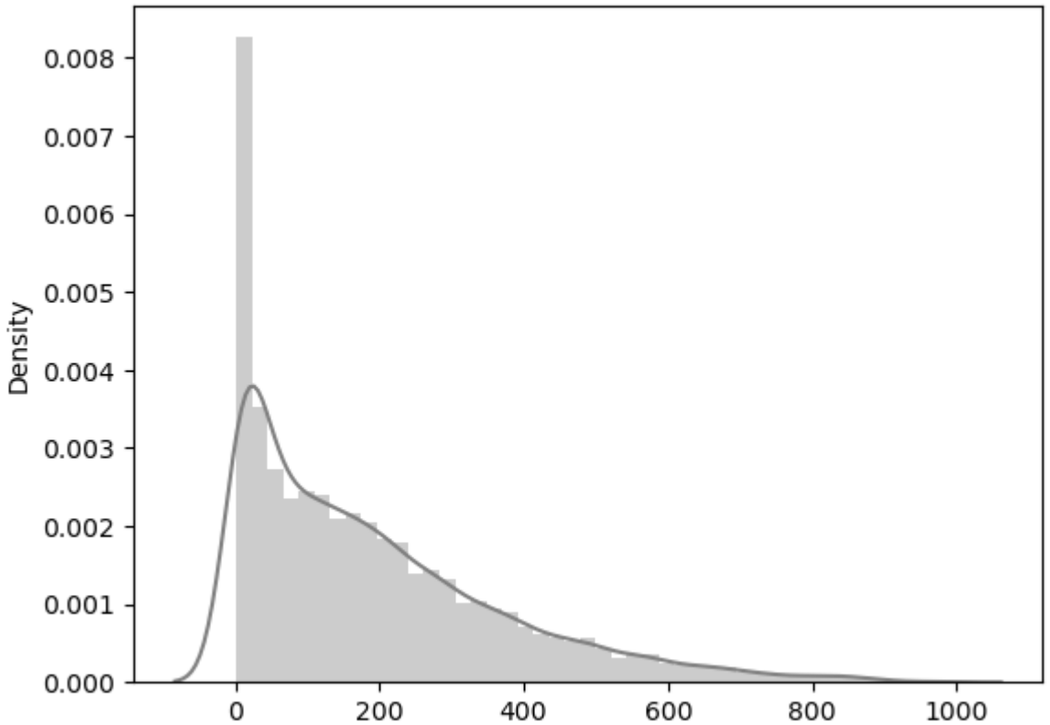


From the above, we can observe that the data recorded of registered users is right skewed.

In [25]:
```
sns.distplot(x=df['count'],color="Grey")
```

```
C:\Users\Home\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future ver
sion. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[25]:
```
<AxesSubplot:ylabel='Density'>
```

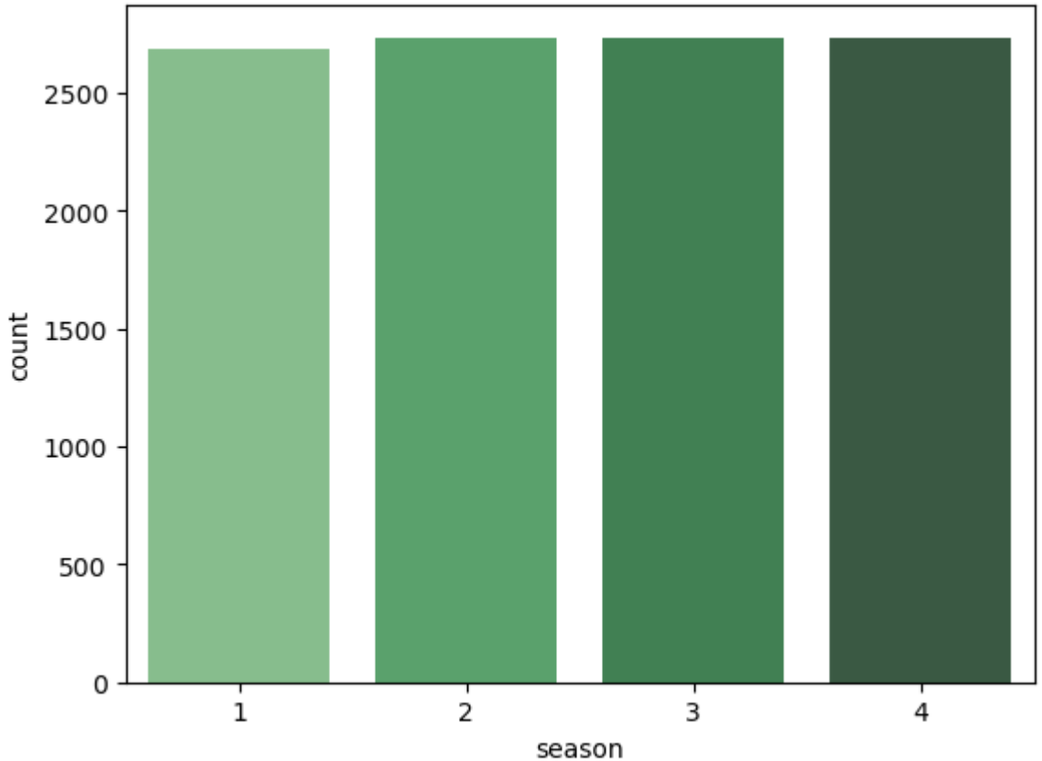


From the above, we can observe that the data recorded of count of electric cycles rented is right skewed.

For Categorical Variable(s)

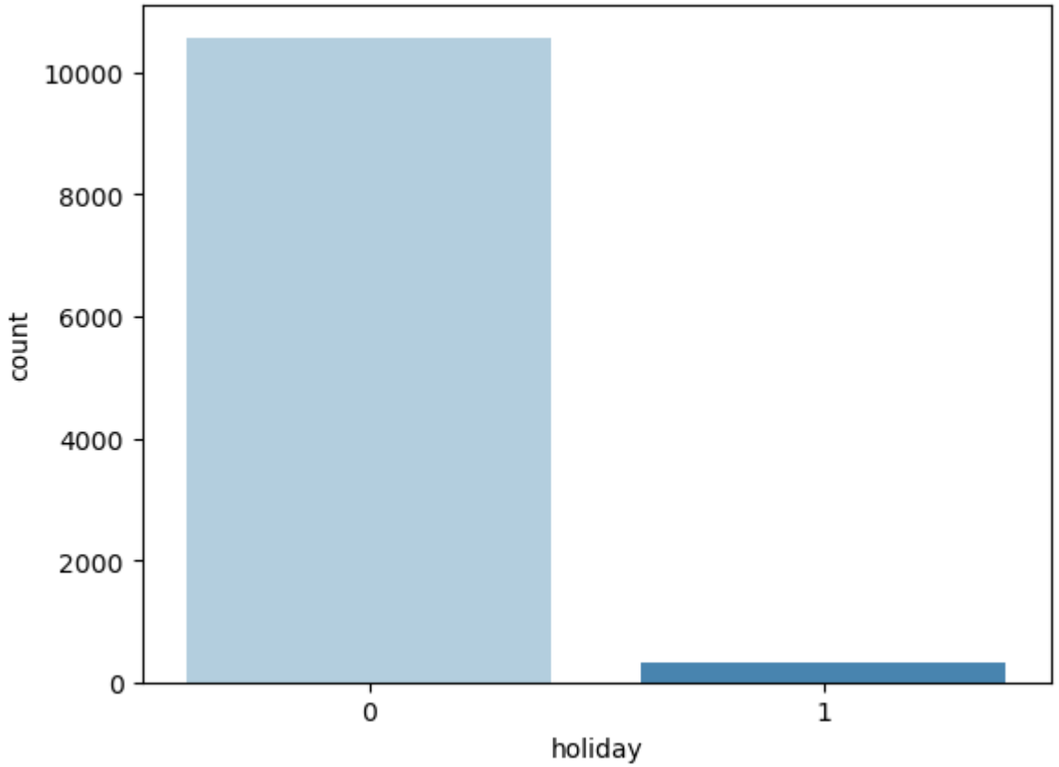
In [26]:
```
sns.countplot(x=df['season'],palette="Greens_d")
```

Out[26]:
```
<AxesSubplot:xlabel='season', ylabel='count'>
```



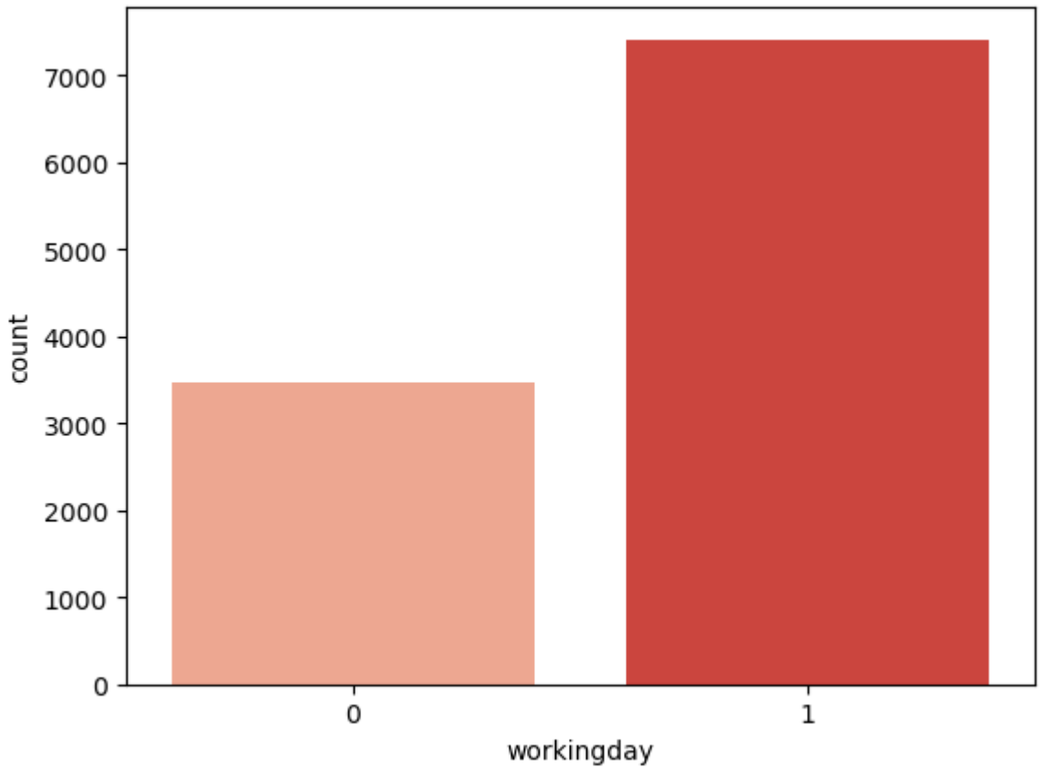
In [27]:
```
sns.countplot(x=df['holiday'], palette="Blues")
```

Out[27]:
```
<AxesSubplot:xlabel='holiday', ylabel='count'>
```



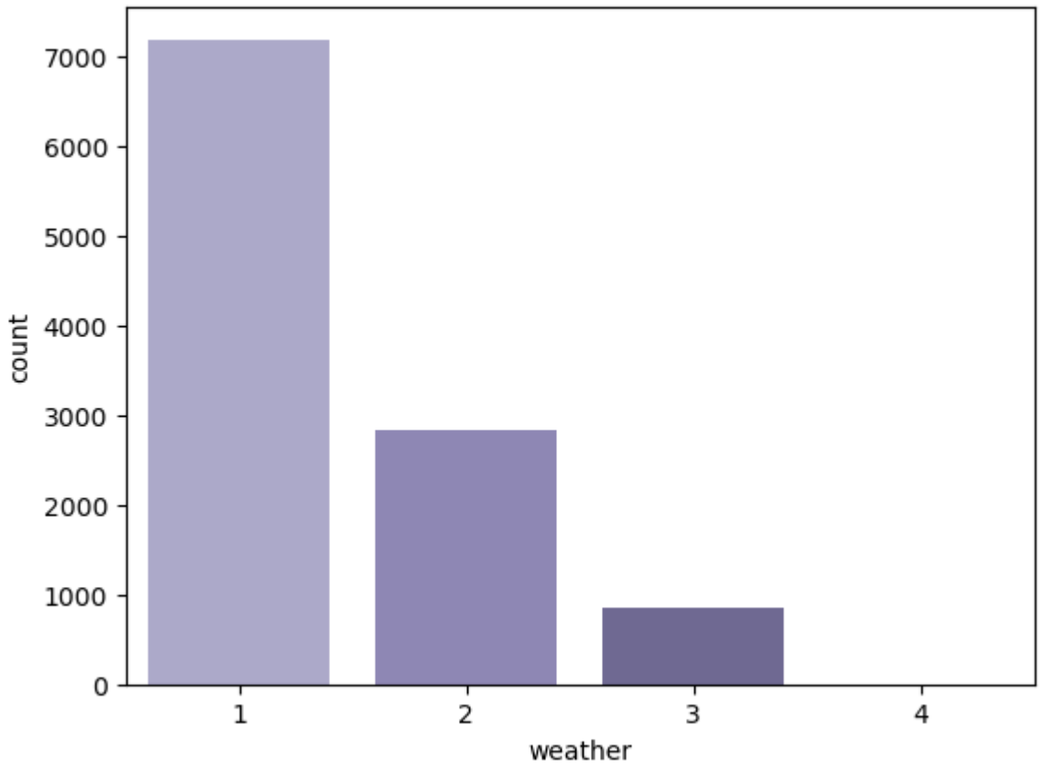
In [28]:
```
sns.countplot(x=df['workingday'], palette="Reds")
```

Out[28]:
```
<AxesSubplot:xlabel='workingday', ylabel='count'>
```

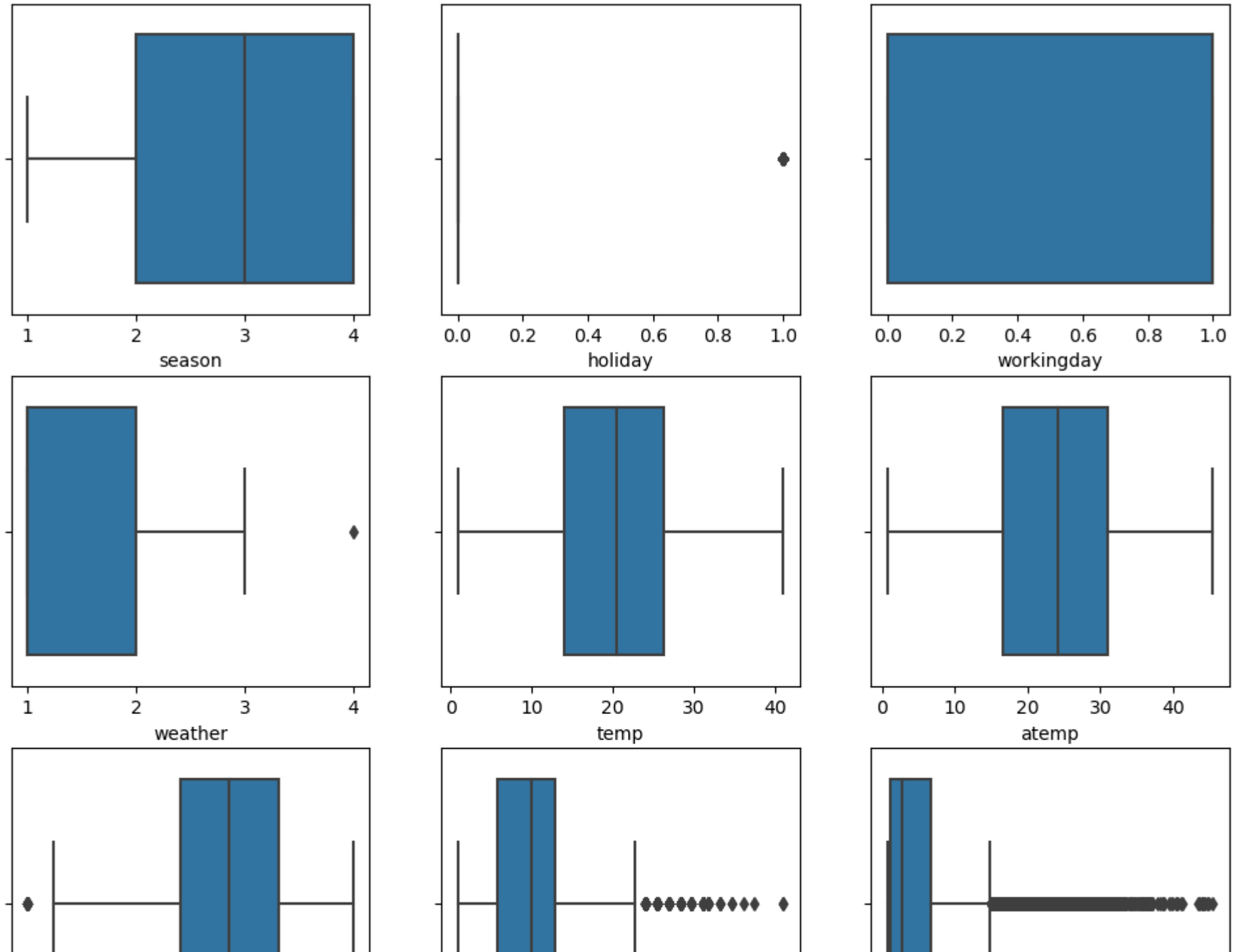


In [29]:
```
sns.countplot(x=df['weather'], palette="Purples_d")
```

Out[29]:
```
<AxesSubplot:xlabel='weather', ylabel='count'>
```



In [30]:
```
#outlier detection
fig, axis = plt.subplots(nrows=4, ncols=3, figsize=(12, 10))
fig.subplots_adjust(top=1.2)
sns.boxplot(data=df, x='season', orient='h', ax=axis[0,0])
sns.boxplot(data=df, x='holiday', orient='h', ax=axis[0,1])
sns.boxplot(data=df, x='workingday', orient='h', ax=axis[0,2])
sns.boxplot(data=df, x='weather', orient='h', ax=axis[1,0])
sns.boxplot(data=df, x='temp', orient='h', ax=axis[1,1])
sns.boxplot(data=df, x='atemp', orient='h', ax=axis[1,2])
sns.boxplot(data=df, x='humidity', orient='h', ax=axis[2,0])
sns.boxplot(data=df, x='windspeed', orient='h', ax=axis[2,1])
sns.boxplot(data=df, x='casual', orient='h', ax=axis[2,2])
sns.boxplot(data=df, x='registered', orient='h', ax=axis[3,0])
sns.boxplot(data=df, x='count', orient='h', ax=axis[3,1])
plt.show()
```

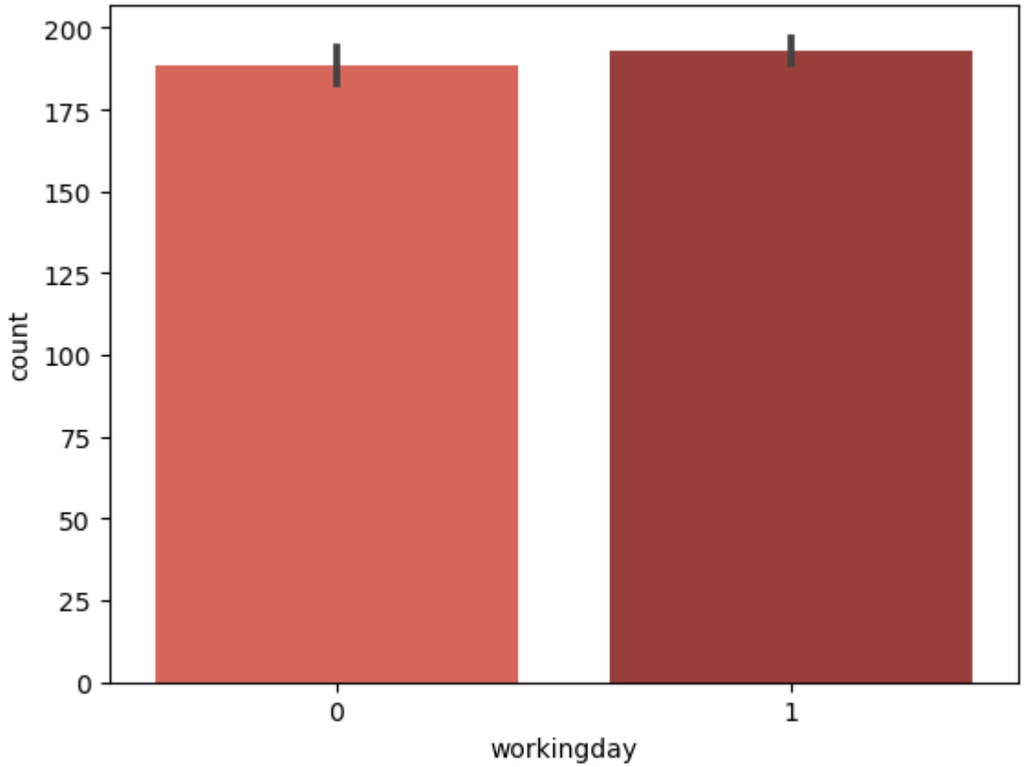

1
2
3
4
season
0.0
0.2
0.4
0.6
0.8
1.0
holiday
0.0
0.2
0.4
0.6
0.8
1.0
workingday
1
2
3
4
weather
0
10
20
30
40
temp
0
10
20
30
40
atemp

From the above, we can observe that there are high number of outliers in variables such as windspeed, casual, registered and count.

Bivariate Analysis

In [31]:
```
sns.barplot(data=df, x='workingday', y='count', palette = 'Reds_d')
```

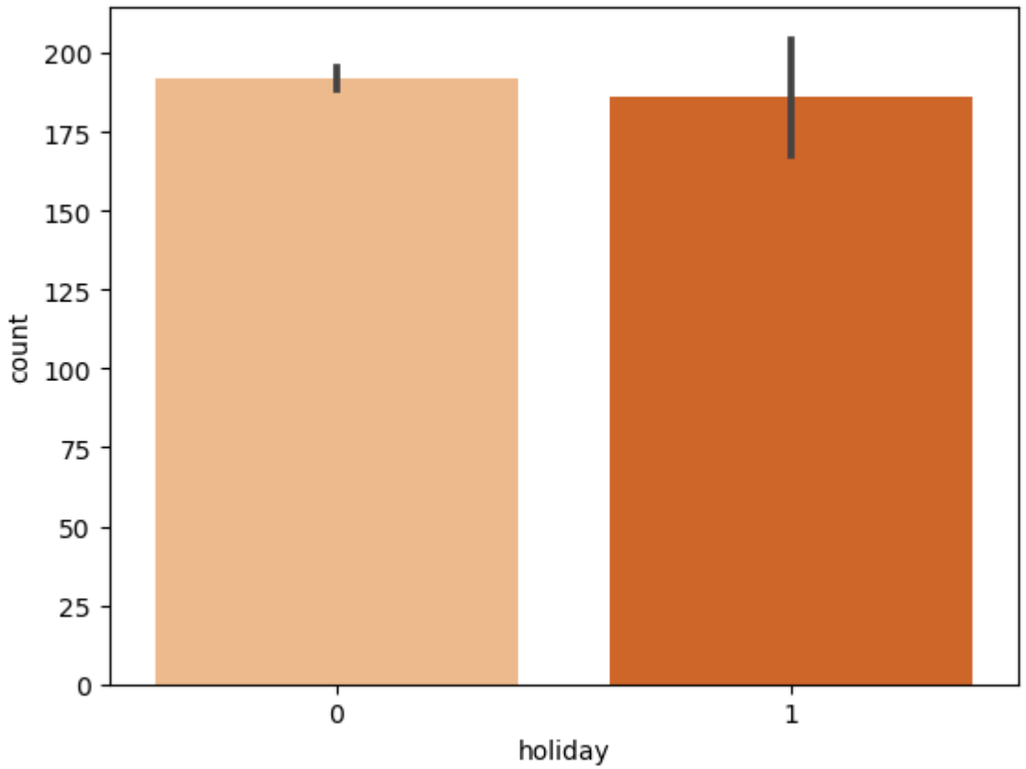
Out[31]:
```
<AxesSubplot:xlabel='workingday', ylabel='count'>
```



From the above, we can observe that there is very slight variation in the count depending on whether it's a workingday or not. However, the significance of such variation can only be analysed after conducting hypothesis testing.

In [32]:
```
sns.barplot(data=df, x='holiday', y='count', palette = 'Oranges')
```

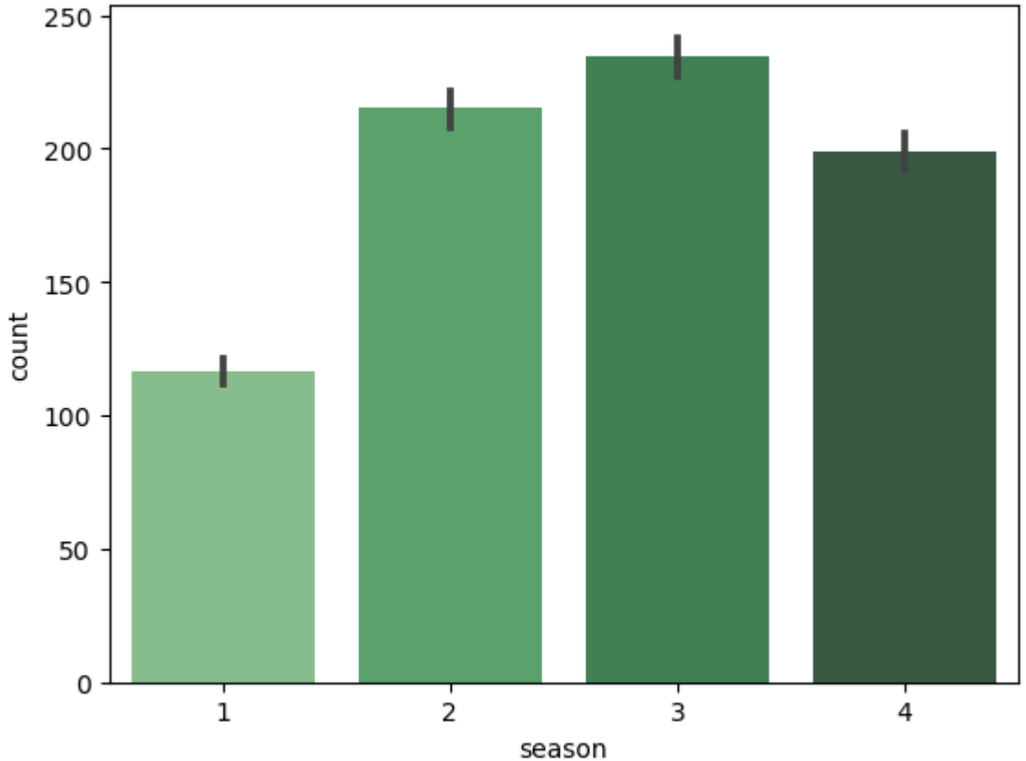
Out[32]:
```
<AxesSubplot:xlabel='holiday', ylabel='count'>
```



From the above, we can observe that there is very slight variation in the count depending on whether it's a holiday or not. However, the significance of such variation can only be analysed after conducting hypothesis testing.

In [33]:
```
sns.barplot(data=df, x='season', y='count', palette = 'Greens_d')
```

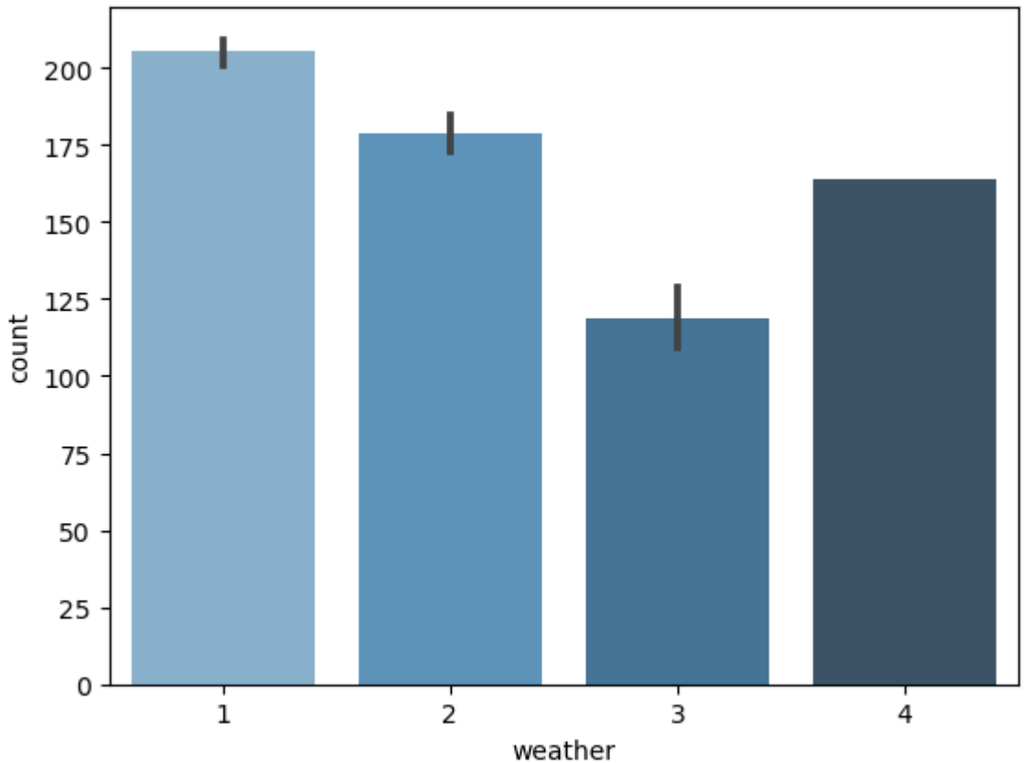
Out[33]:
```
<AxesSubplot:xlabel='season', ylabel='count'>
```



From the above, we can observe that there is variation in the count depending on season. However, the significance of such variation can only be analysed after conducting hypothesis testing.

In [34]:
```
sns.barplot(data=df, x='weather', y='count', palette = 'Blues_d')
```

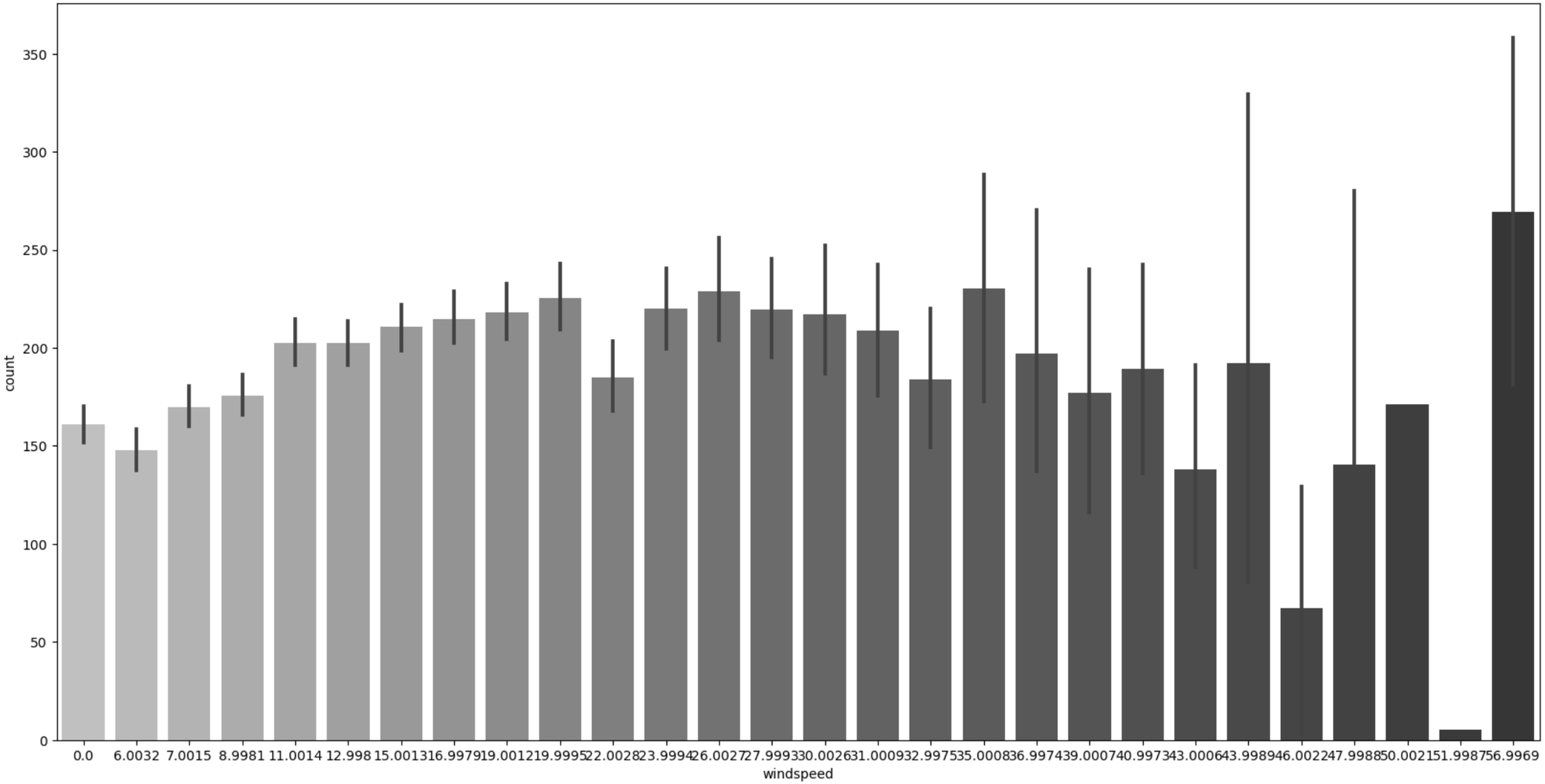
Out[34]:
```
<AxesSubplot:xlabel='weather', ylabel='count'>
```



From the above, we can observe that there is high variation in the count depending on weather. However, the significance of such variation can only be analysed after conducting hypothesis testing.

In [35]:
```
plt.figure(figsize=(20,10))
sns.barplot(data=df, x='windspeed', y='count', palette = 'Greys_d')
```

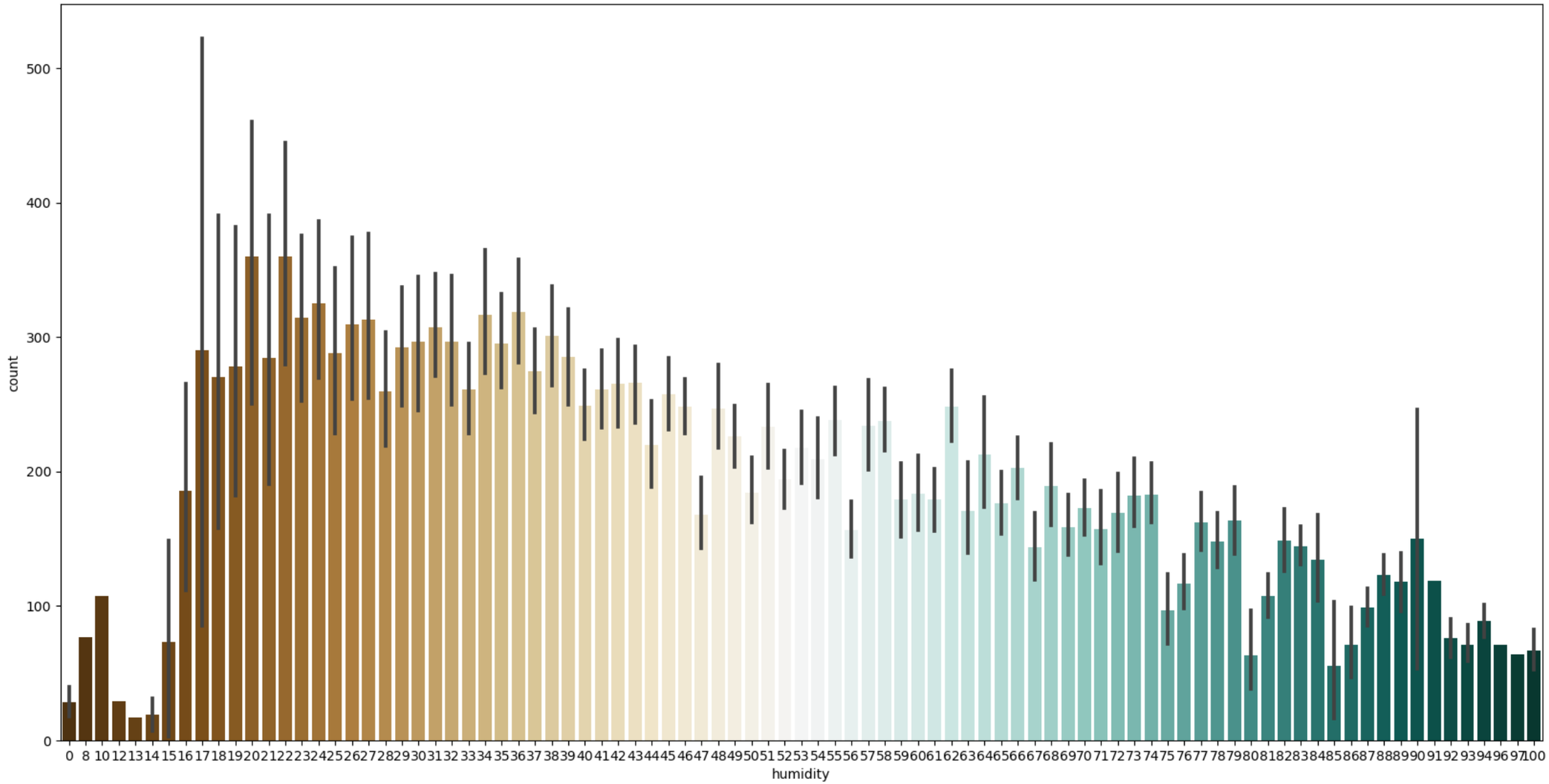
Out[35]:
```
<AxesSubplot:xlabel='windspeed', ylabel='count'>
```



From the above, we can observe that there is high variation in the count depending on windspeed. However, the significance of such variation can only be analysed after conducting hypothesis testing.

In [36]:
```
plt.figure(figsize=(20,10))
sns.barplot(data=df, x='humidity', y='count', palette = 'BrBG')
```

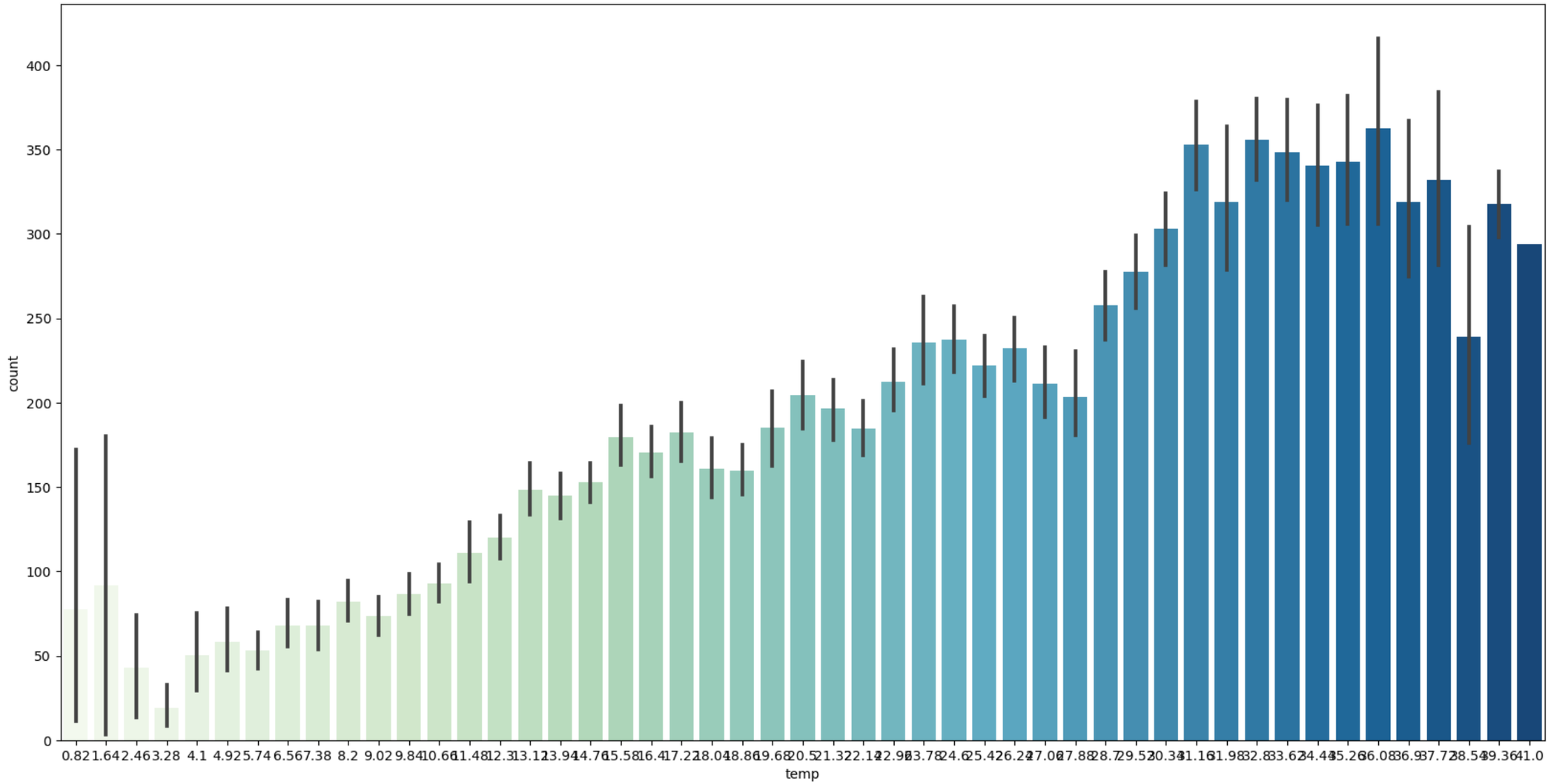
Out[36]: <AxesSubplot:xlabel='humidity', ylabel='count'>



From the above, we can observe that there is high variation in the count depending on humidity. However, the significance of such variation can only be analysed after conducting hypothesis testing.

In [37]:
```
plt.figure(figsize=(20,10))
sns.barplot(data=df, x='temp', y='count', palette = 'GnBu')
```

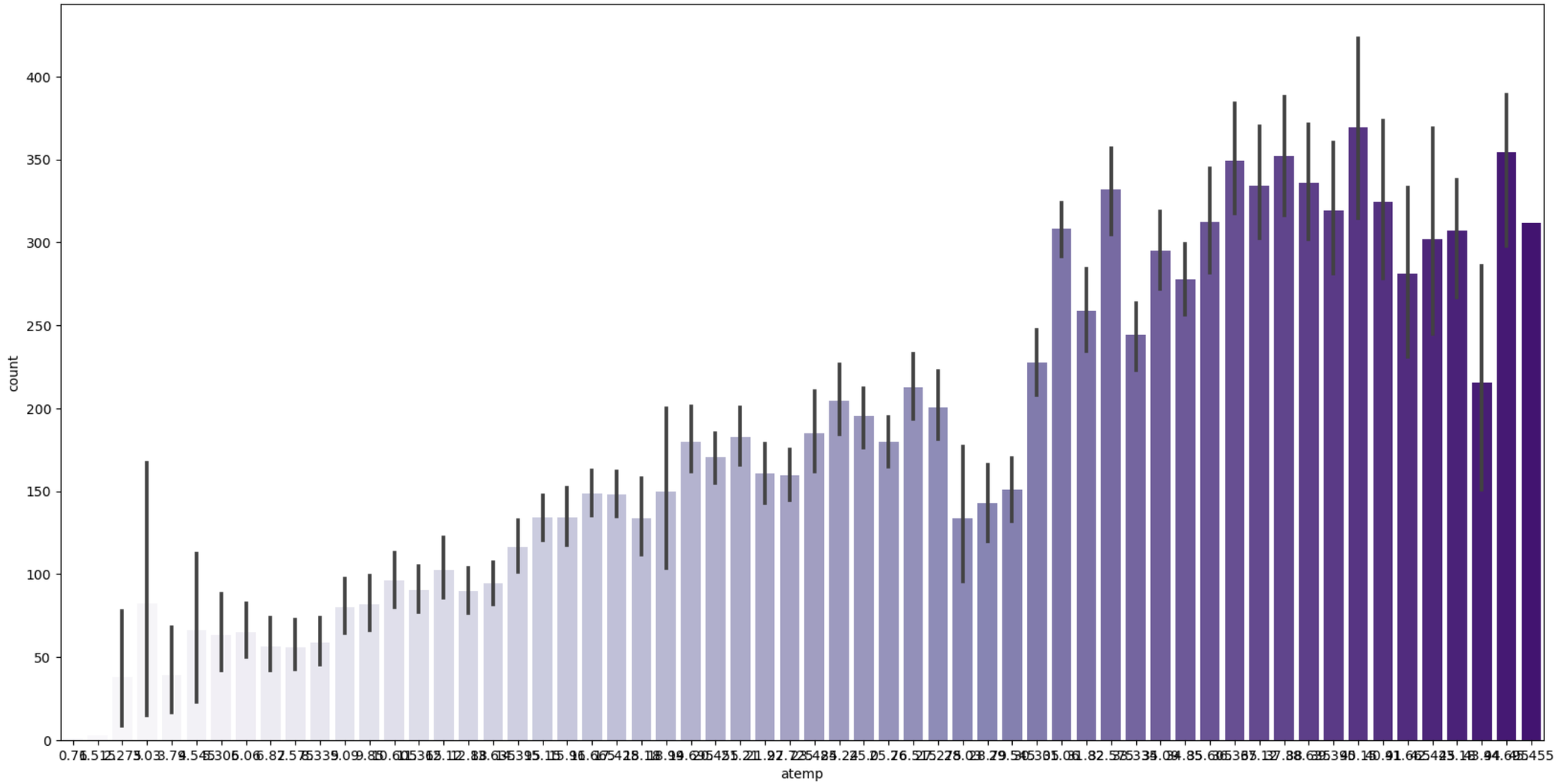
Out[37]: <AxesSubplot:xlabel='temp', ylabel='count'>



From the above, we can observe that there is high variation in the count depending on temperature. However, the significance of such variation can only be analysed after conducting hypothesis testing.

In [38]:
```
plt.figure(figsize=(20,10))
sns.barplot(data=df, x='atemp', y='count', palette = 'Purples')
```

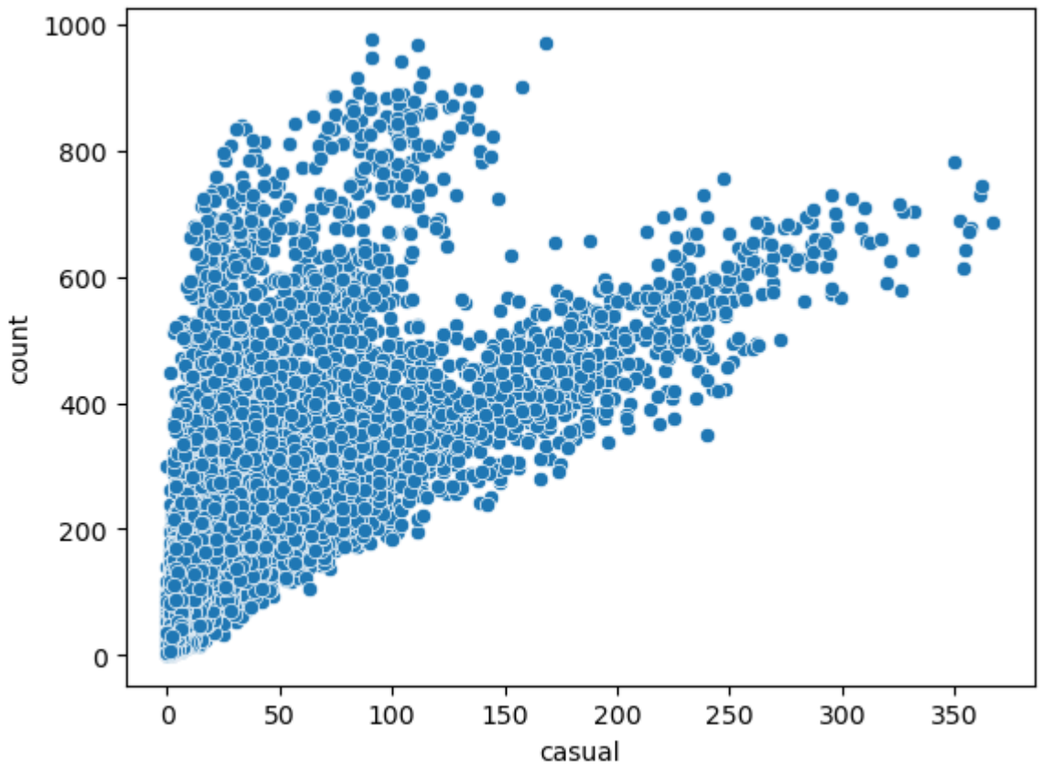
Out[38]: <AxesSubplot:xlabel='atemp', ylabel='count'>



From the above, we can observe that there is high variation in the count depending on feeling temperature. However, the significance of such variation can only be analysed after conducting hypothesis testing.

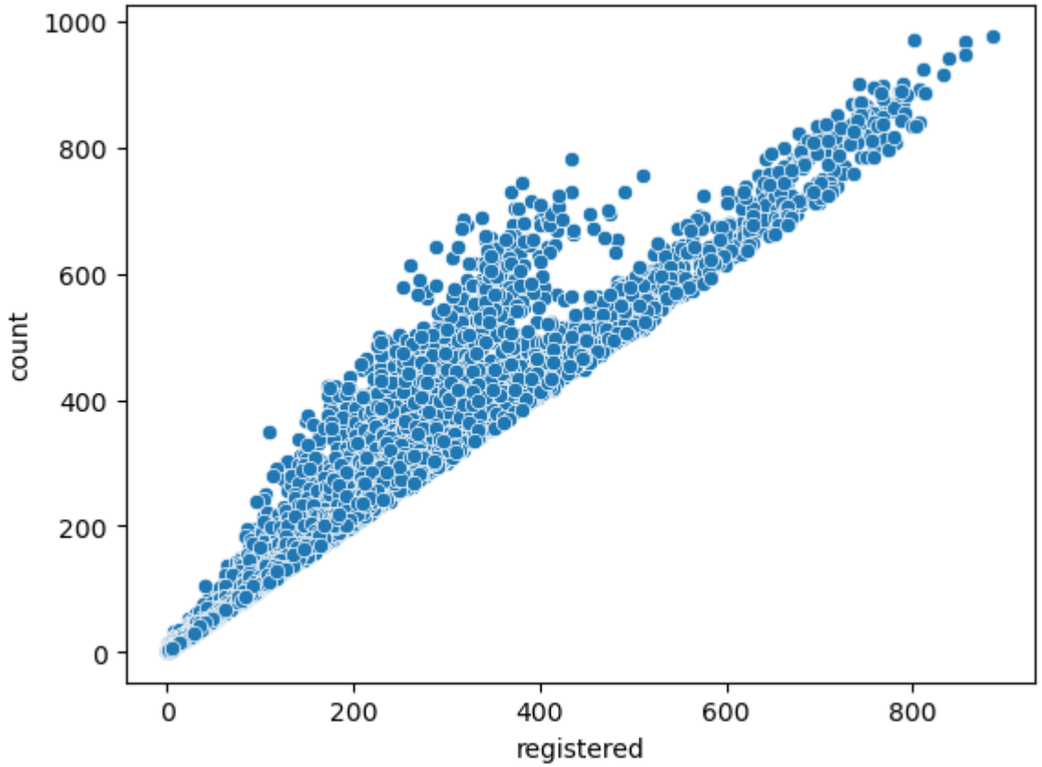
In [39]:
```
sns.scatterplot(data=df, x='casual', y='count')
```

Out[39]: <AxesSubplot:xlabel='casual', ylabel='count'>



In [40]:
```
sns.scatterplot(data=df, x='registered', y='count')
```

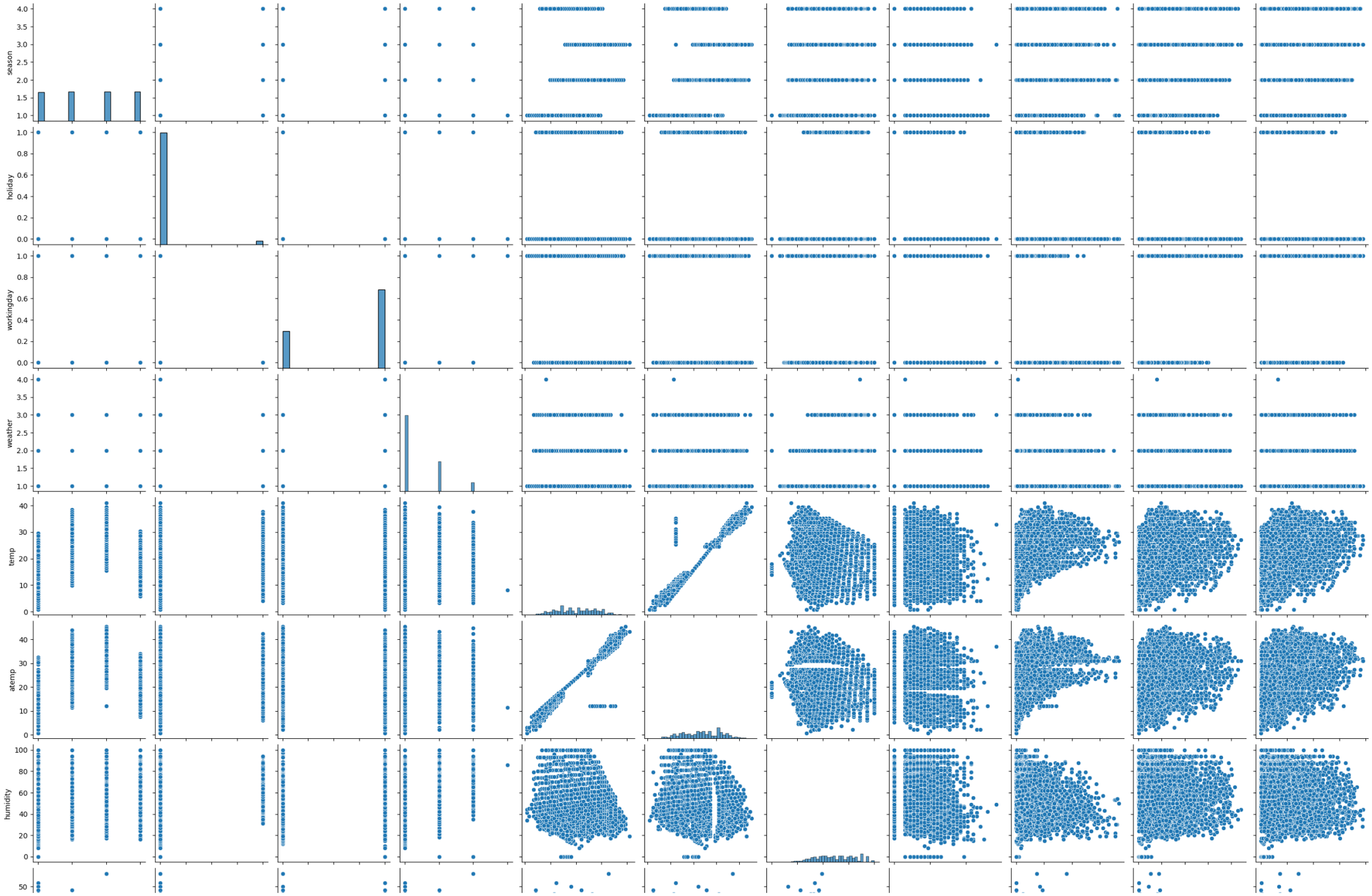
Out[40]: <AxesSubplot:xlabel='registered', ylabel='count'>



For correlation: Heatmaps, Pairplots

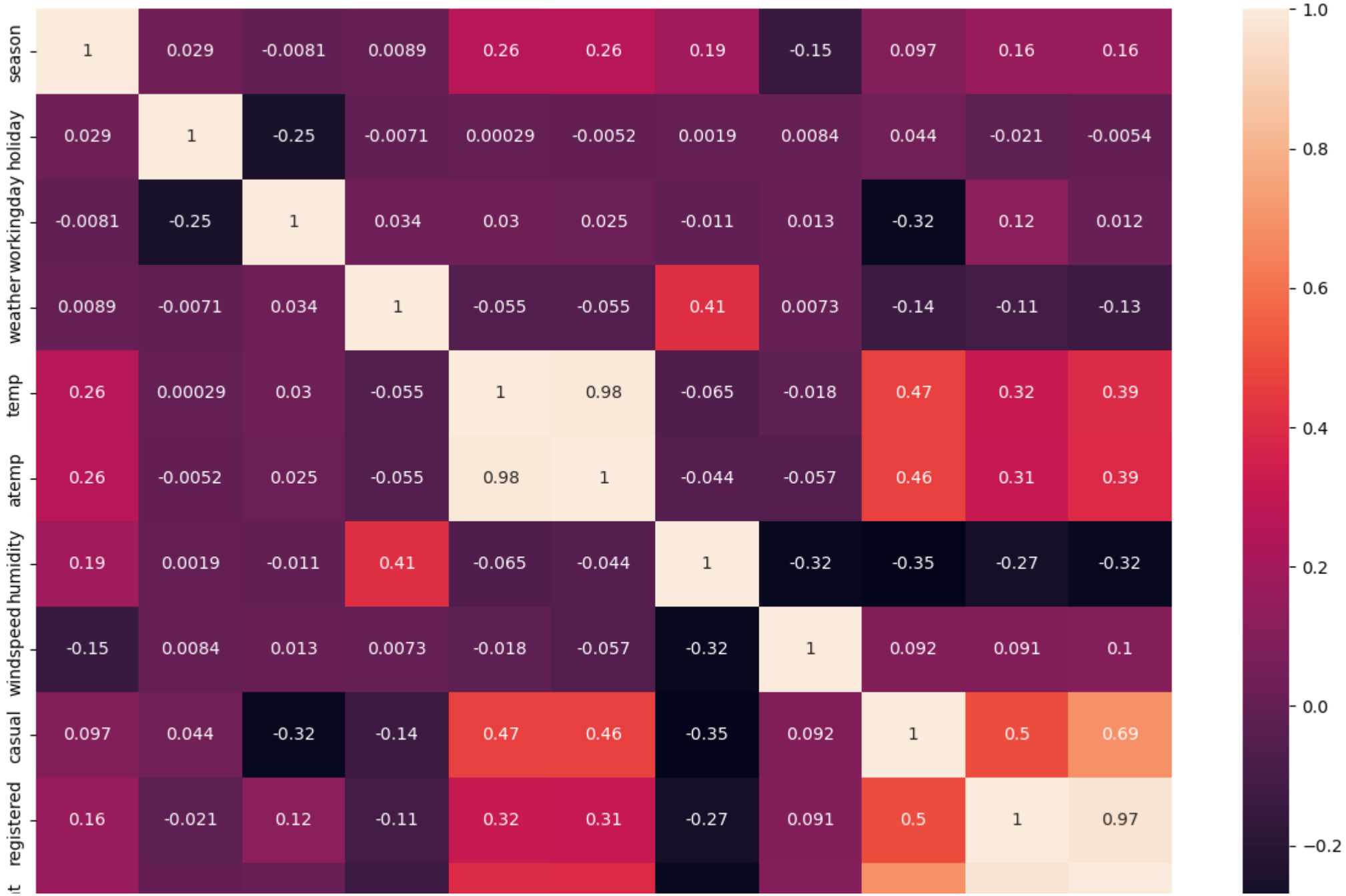
In [41]:
```
sns.pairplot(df)
```

Out[41]:
```
<seaborn.axisgrid.PairGrid at 0x194069ded60>
```


season
holiday
workingday
weather
temp
atemp
humidity

In [42]:
```
plt.figure(figsize=(15,10))
sns.heatmap(df.corr(), annot = True)
```

Out[42]:
```
<AxesSubplot:>
```


season 1 0.029 -0.0081 0.0089 0.26 0.26 0.19 -0.15 0.097 0.16 0.16
holiday 0.029 1 -0.25 -0.0071 0.00029 -0.0052 0.0019 0.0084 0.044 -0.021 -0.0054
workingday -0.0081 -0.25 1 0.034 0.03 0.025 -0.011 0.013 -0.32 0.12 0.012
weather 0.0089 -0.0071 0.034 1 -0.055 -0.055 0.41 0.0073 -0.14 -0.11 -0.13
temp 0.26 0.00029 0.03 -0.055 1 0.98 -0.065 -0.018 0.47 0.32 0.39
atemp 0.26 -0.0052 0.025 -0.055 0.98 1 -0.044 -0.057 0.46 0.31 0.39
humidity 0.19 0.0019 -0.011 0.41 -0.065 -0.044 1 -0.32 -0.35 -0.27 -0.32
windspeed -0.15 0.0084 0.013 0.0073 -0.018 -0.057 -0.32 1 0.092 0.091 0.1
casual 0.097 0.044 -0.32 -0.14 0.47 0.46 -0.35 0.092 1 0.5 0.69
registered 0.16 -0.021 0.12 -0.11 0.32 0.31 -0.27 0.091 0.5 1 0.97
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

Business Insights based on Non- Graphical and Visual Analysis

A. Comments on range of attributes, outliers of various attributes

The dataset analyses 10886 customers of rental bikes across 4 different seasons and weathers on working day as well as holiday. Some of the rental customers are casual, while others are registered. The dataset also has a record of humidity, windspeed, temperature and feeling temperature.

After analysing the data, we can observe that there are high number of outliers in variables such as windspeed, casual, registered and count.

B. Comments on the distribution of the variables and relationship between them (Univariate and Bivariate plots)

After univariate and bivariate analysis of the data, we observe that:

i. There is very slight variation in the count depending on whether it's a workingday or not.

ii. There is very slight variation in the count depending on whether it's a holiday or not.

iii. There is variation in the count depending on season.

iv. There is high variation in the count depending on weather.

v. There is high variation in the count depending on windspeed.

vi. There is high variation in the count depending on humidity.

vii. There is high variation in the count depending on temperature.

viii.There is high variation in the count depending on feeling temperature.

However, the significance of such variation can only be analysed after conducting hypothesis testing.

2. Hypothesis Testing

Sample T-Test

In [43]:
```
#H0 = holiday has no effect on count
#Ha = holiday has effect on count
a = df[df['holiday']==0]['count']
b = df[df['holiday']==1]['count']
ttest_ind(a, b)
```

Out[43]:
```
Ttest_indResult(statistic=0.5626388963477119, pvalue=0.5736923883271103)
```

Taking significance value as 0.05, we observe that p_value > significance value i.e. 0.57 > 0.05. Hence, we fail to reject H0 in this case. Thus, holiday has no effect on count of electric cycles rented.

In [44]:
```
pd.crosstab(df['count'],df['holiday'], normalize="index")
```

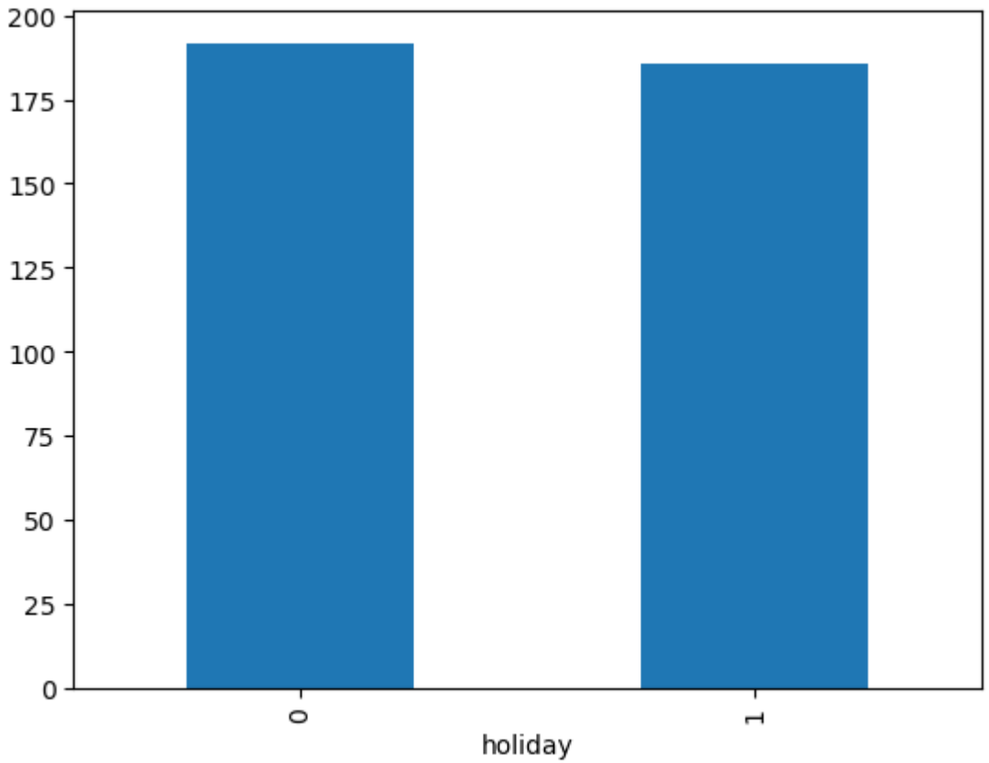
Out[44]:

holiday	0	1
count		
1	0.990476	0.009524
2	0.992424	0.007576
3	0.972222	0.027778
4	0.953020	0.046980
5	0.994083	0.005917
...	...	...
943	1.000000	0.000000
948	1.000000	0.000000
968	1.000000	0.000000
970	1.000000	0.000000
977	1.000000	0.000000

822 rows × 2 columns

In [45]:
```
df.groupby('holiday')['count'].mean().plot.bar()
```

Out[45]:
```
<AxesSubplot:xlabel='holiday'>
```



In [46]:
```
#H0 = workingday has no effect on count
#Ha = workingday has effect on count
a = df[df['workingday']==0]['count']
b = df[df['workingday']==1]['count']
ttest_ind(a, b)
```

Out[46]:
```
Ttest_indResult(statistic=-1.2096277376026694, pvalue=0.22644804226361348)
```

Taking significance value as 0.05, we observe that p_value > significance value i.e. 0.22 > 0.05. Hence, we fail to reject H0 in this case. Thus, working day has no effect on count of electric cycles rented.

In [47]:
```
pd.crosstab(df['count'],df['workingday'], normalize="index")
```

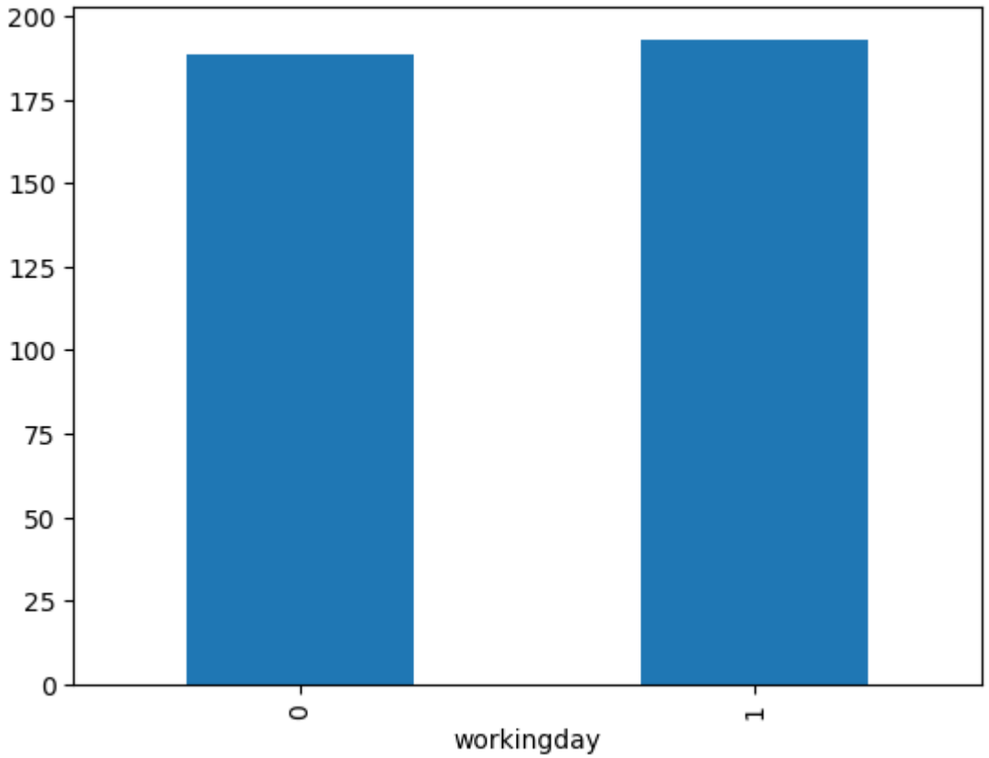
Out[47]:

workingday	0	1
count		
1	0.200000	0.800000
2	0.189394	0.810606
3	0.187500	0.812500
4	0.181208	0.818792
5	0.207101	0.792899
...	...	...
943	0.000000	1.000000
948	0.000000	1.000000
968	0.000000	1.000000
970	0.000000	1.000000
977	0.000000	1.000000

822 rows × 2 columns

In [48]:
```
df.groupby('workingday')['count'].mean().plot.bar()
```

Out[48]:
```
<AxesSubplot:xlabel='workingday'>
```



ANNOVA

In [49]:
```
# H0: Weather has no effect on count
# Ha: Weather has effect on count
f_oneway(df['weather'], df['count'])
```

Out[49]:
```
F_onewayResult(statistic=11995.867126320749, pvalue=0.0)
```

Taking significance value as 0.05, we observe that p_value < significance value i.e. 0.0 < 0.05. Hence, we reject H0 in this case. Thus, weather has an effect on count of electric cycles rented.

In [50]:
```
pd.crosstab(df['count'],df['weather'], normalize="index")
```

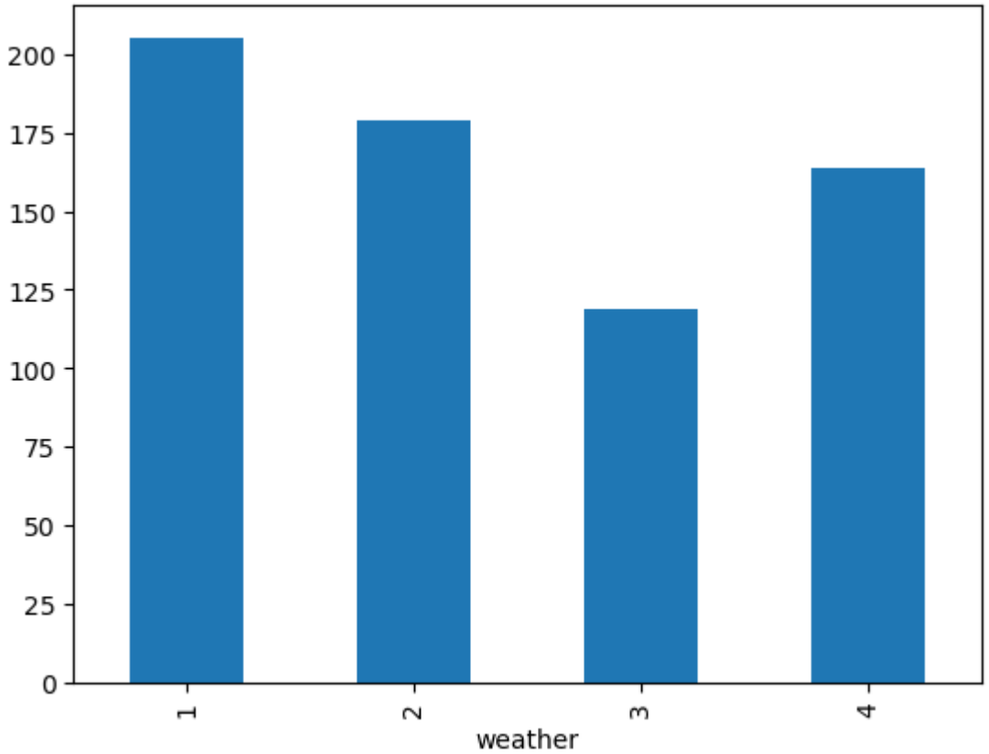
Out[50]:

weather	1	2	3	4
count				
1	0.628571	0.238095	0.133333	0.0
2	0.598485	0.257576	0.143939	0.0
3	0.576389	0.243056	0.180556	0.0
4	0.651007	0.248322	0.100671	0.0
5	0.609467	0.260355	0.130178	0.0
...	...	...	...	...
943	1.000000	0.000000	0.000000	0.0
948	1.000000	0.000000	0.000000	0.0
968	1.000000	0.000000	0.000000	0.0
970	1.000000	0.000000	0.000000	0.0
977	1.000000	0.000000	0.000000	0.0

822 rows × 4 columns

In [51]:
```
df.groupby('weather')['count'].mean().plot.bar()
```

Out[51]:
```
<AxesSubplot:xlabel='weather'>
```



In [52]:
```
# H0: season has no effect on count
# Ha: season has effect on count
f_oneway(df['season'], df['count'])
```

Out[52]:
```
F_onewayResult(statistic=11858.659616383344, pvalue=0.0)
```

Taking significance value as 0.05, we observe that p_value < significance value i.e. 0.0 < 0.05. Hence, we reject H0 in this case. Thus, season has an effect on count of electric cycles rented.

In [53]:
```
pd.crosstab(df['count'],df['season'], normalize="index")
```

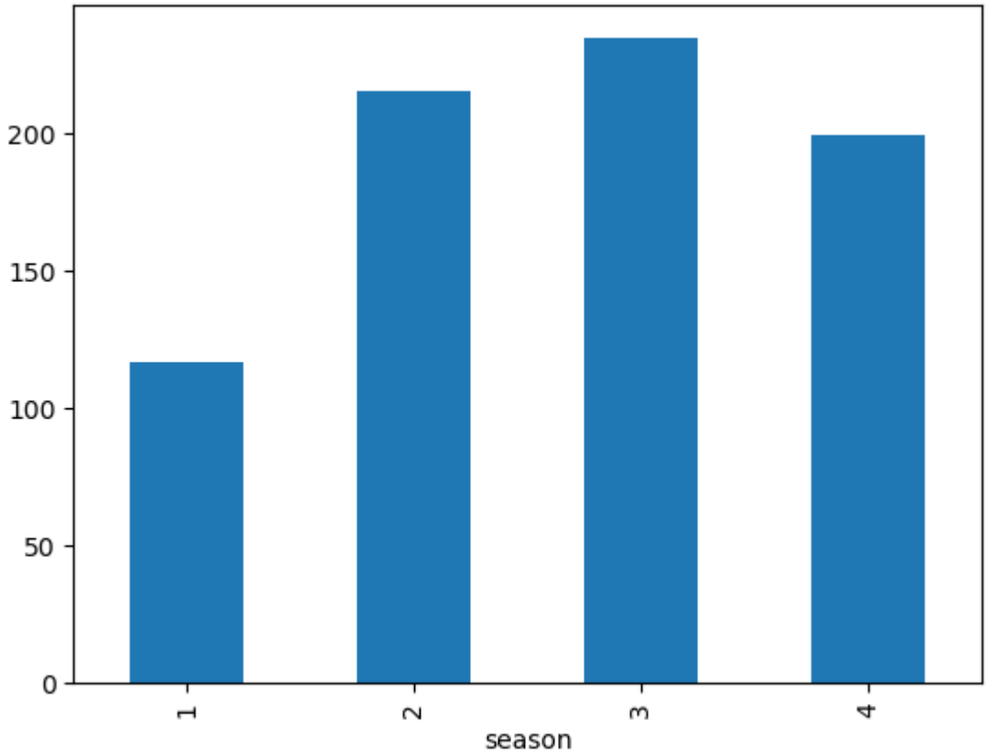
Out[53]:

season	1	2	3	4
count				
1	0.723810	0.152381	0.038095	0.085714
2	0.583333	0.189394	0.060606	0.166667
3	0.527778	0.166667	0.097222	0.208333
4	0.348993	0.255034	0.187919	0.208054
5	0.236686	0.266272	0.236686	0.260355
...	...	...	...	...
943	0.000000	0.000000	0.000000	1.000000
948	0.000000	0.000000	0.000000	1.000000
968	0.000000	0.000000	1.000000	0.000000
970	0.000000	0.000000	1.000000	0.000000
977	0.000000	0.000000	1.000000	0.000000

822 rows × 4 columns

In [54]:
```
df.groupby('season')['count'].mean().plot.bar()
```

Out[54]:
```
<AxesSubplot:xlabel='season'>
```



In [55]:
```
# H0: windspeed has no effect on count
# Ha: windspeed has effect on count
f_oneway(df['windspeed'], df['count'])
```

Out[55]:
```
F_onewayResult(statistic=10581.547195265242, pvalue=0.0)
```

Taking significance value as 0.05, we observe that p_value < significance value i.e. 0.0 < 0.05. Hence, we reject H0 in this case. Thus, windspeed has an effect on count of electric cycles rented.

In [56]:
```
pd.crosstab(df['count'],df['windspeed'], normalize="index")
```

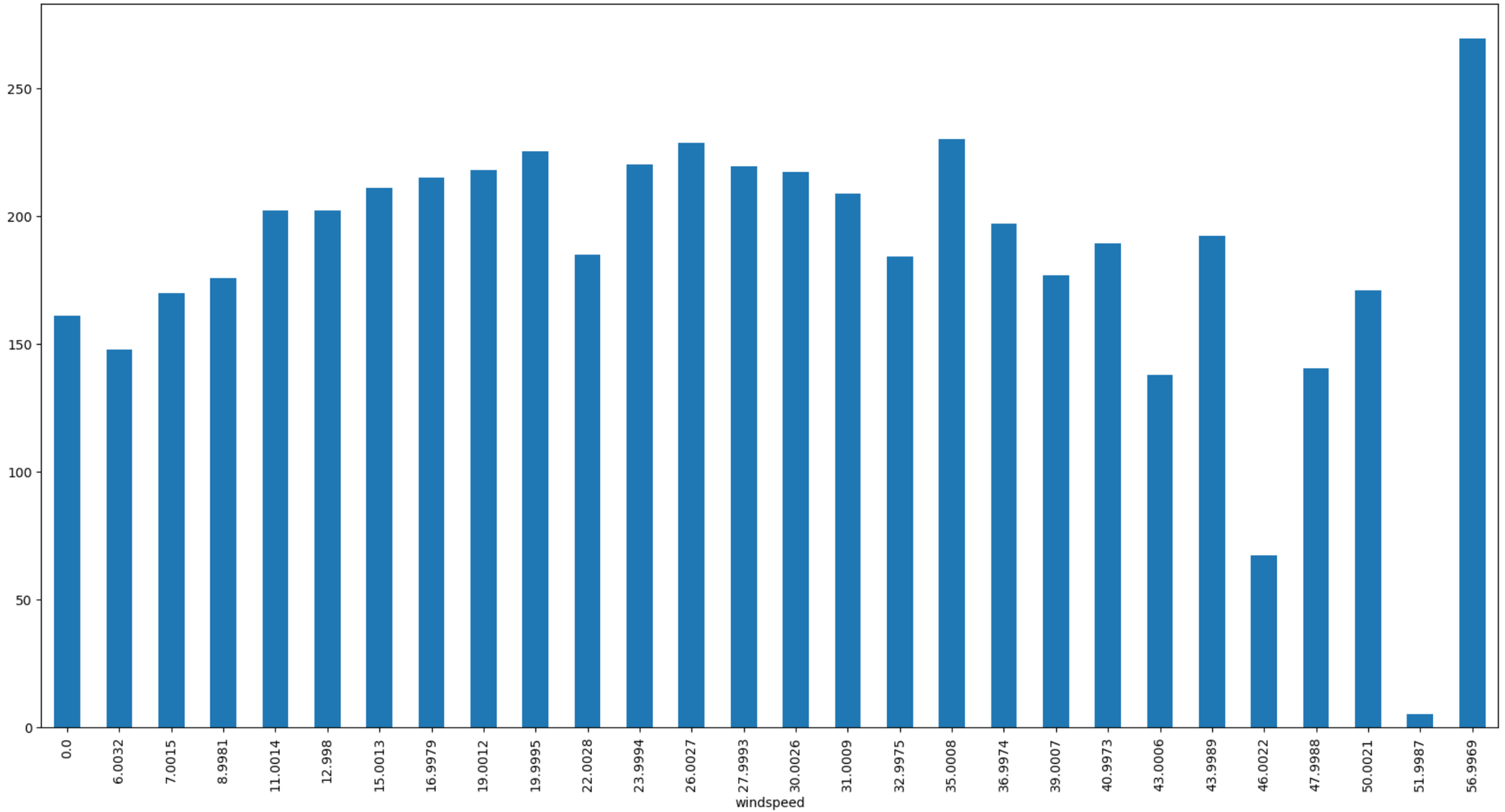
Out[56]:

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	46.0022	47.9988	50.002
count																			
1	0.085714	0.114286	0.076190	0.095238	0.104762	0.104762	0.076190	0.066667	0.095238	0.047619	...	0.0	0.009524	0.0	0.000000	0.0	0.000000	0.009524	0.
2	0.166667	0.083333	0.075758	0.128788	0.053030	0.098485	0.037879	0.068182	0.083333	0.068182	...	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.000000	0.
3	0.201389	0.104167	0.055556	0.097222	0.083333	0.118056	0.055556	0.062500	0.041667	0.048611	...	0.0	0.000000	0.0	0.000000	0.0	0.006944	0.000000	0.
4	0.174497	0.107383	0.100671	0.127517	0.087248	0.087248	0.073826	0.060403	0.080537	0.020134	...	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.000000	0.
5	0.142012	0.100592	0.100592	0.153846	0.124260	0.106509	0.088757	0.053254	0.029586	0.035503	...	0.0	0.000000	0.0	0.005917	0.0	0.000000	0.000000	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
943	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	...	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.000000	0.
948	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	...	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.000000	0.
968	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	...	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.000000	0.
970	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.000000	0.
977	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	...	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.000000	0.

822 rows × 28 columns

In [57]:
```
plt.figure(figsize=(20,10))
df.groupby('windspeed')['count'].mean().plot.bar()
```

Out[57]:
```
<AxesSubplot:xlabel='windspeed'>
```



In [58]:
```
# H0: humidity has no effect on count
# Ha: humidity has effect on count
f_oneway(df['humidity'], df['count'])
```

Out[58]:
```
F_onewayResult(statistic=5517.484417344751, pvalue=0.0)
```

Taking significance value as 0.05, we observe that p_value < significance value i.e. 0.0 < 0.05. Hence, we reject H0 in this case. Thus, humidity has an effect on count of electric cycles rented.

In [59]:
```
pd.crosstab(df['count'],df['humidity'], normalize="index")
```

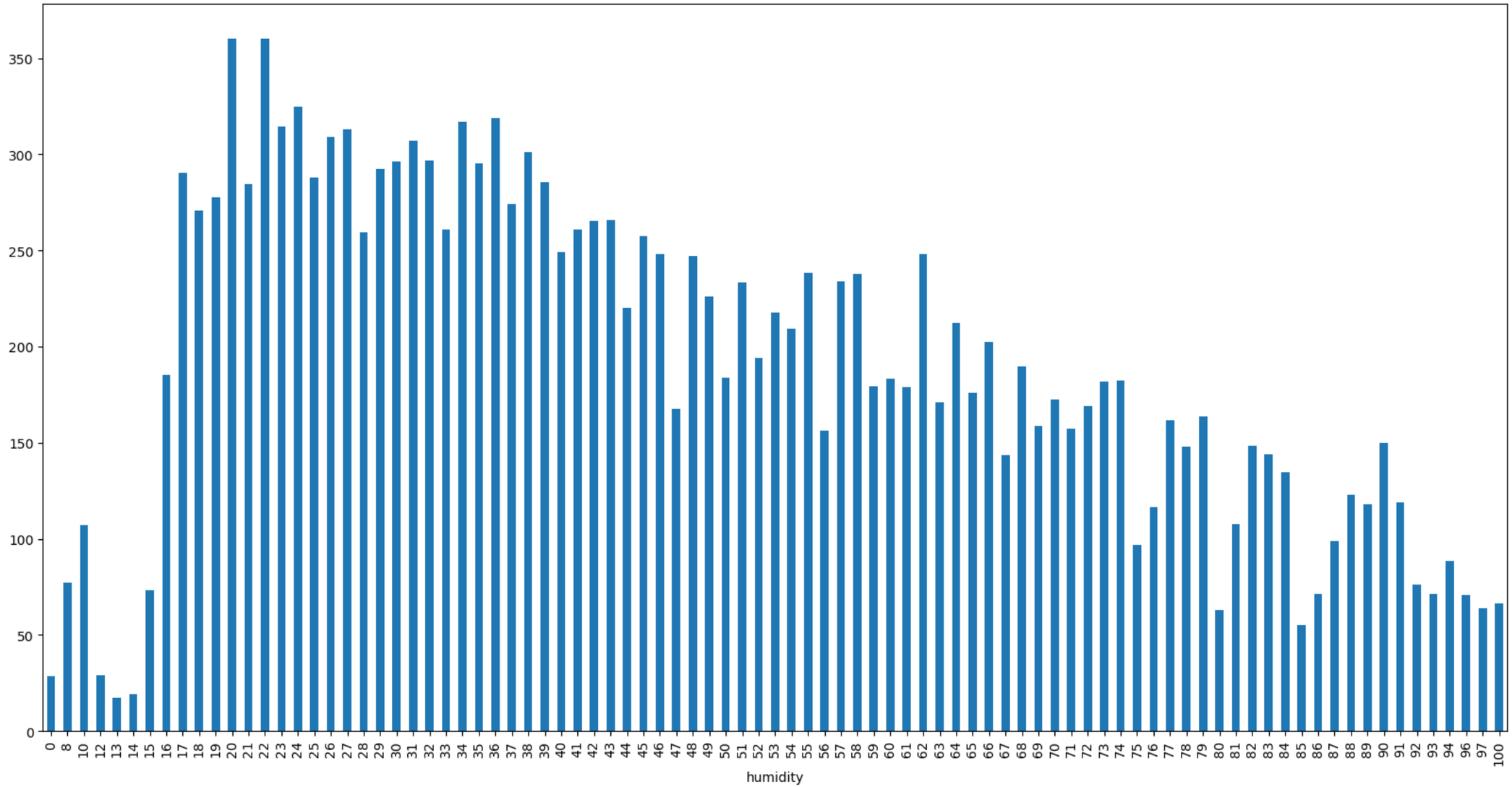
Out[59]:

humidity	0	8	10	12	13	14	15	16	17	18	...	88	89	90	91	92	93	94	96	97	100
count																					
1	0.009524	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	...	0.019048	0.009524	0.0	0.0	0.0	0.066667	0.028571	0.0	0.0	0.019048
2	0.007576	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	...	0.053030	0.007576	0.0	0.0	0.0	0.090909	0.060606	0.0	0.0	0.015152
3	0.013889	0.0	0.0	0.0	0.0	0.0	0.013889	0.0	0.0	0.0	...	0.027778	0.006944	0.0	0.0	0.0	0.090278	0.090278	0.0	0.0	0.048611
4	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	...	0.040268	0.020134	0.0	0.0	0.0	0.046980	0.093960	0.0	0.0	0.046980
5	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	...	0.088757	0.029586	0.0	0.0	0.0	0.029586	0.082840	0.0	0.0	0.017751
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
943	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	...	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.000000
948	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	...	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.000000
968	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	...	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.000000
970	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	...	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.000000
977	0.000000	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	...	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.000000	0.0	0.0	0.000000

822 rows × 89 columns

In [60]:
```
plt.figure(figsize=(20,10))
df.groupby('humidity')['count'].mean().plot.bar()
```

Out[60]:
```
<AxesSubplot:xlabel='humidity'>
```



In [61]:
```
# H0: temperature has no effect on count
# Ha: temperature has effect on count
f_oneway(df['temp'], df['count'])
```

Out[61]:
```
F_onewayResult(statistic=9721.865326646866, pvalue=0.0)
```

Taking significance value as 0.05, we observe that p_value < significance value i.e. 0.0 < 0.05. Hence, we reject H0 in this case. Thus, temperature has an effect on count of electric cycles rented.

In [62]:
```
pd.crosstab(df['count'],df['temp'], normalize="index")
```

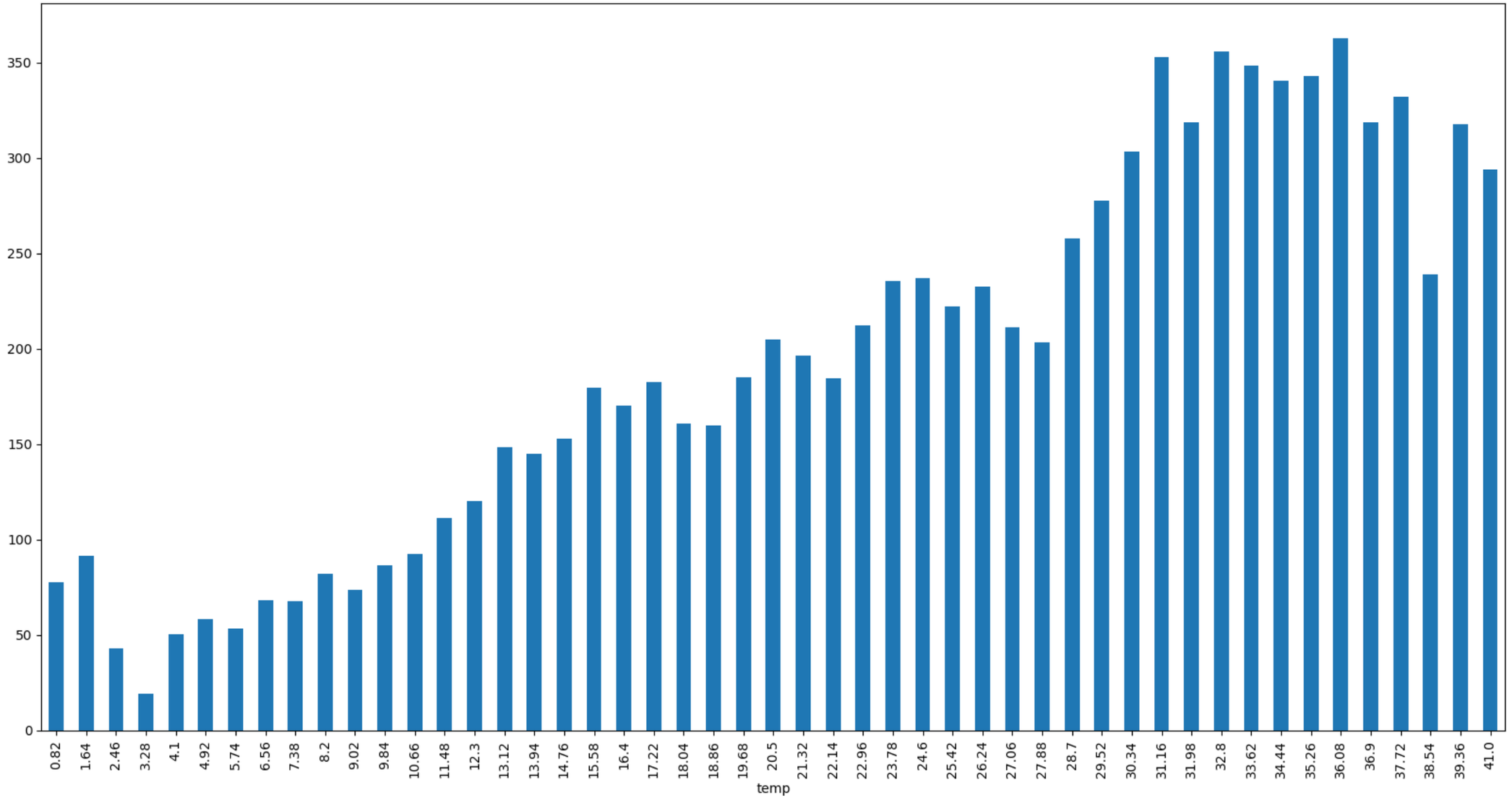
Out[62]:

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
count																					
1	0.019048	0.000000	0.000000	0.019048	0.038095	0.047619	0.047619	0.038095	0.057143	0.104762	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.007576	0.000000	0.007576	0.000000	0.007576	0.015152	0.022727	0.060606	0.037879	0.053030	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.000000	0.006944	0.000000	0.006944	0.027778	0.013889	0.055556	0.027778	0.041667	0.034722	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.000000	0.000000	0.000000	0.006711	0.006711	0.020134	0.026846	0.020134	0.006711	0.060403	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.000000	0.000000	0.000000	0.000000	0.005917	0.005917	0.017751	0.035503	0.005917	0.041420	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
943	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
948	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
968	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
970	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
977	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

822 rows × 49 columns

In [63]:
```
plt.figure(figsize=(20,10))
df.groupby('temp')['count'].mean().plot.bar()
```

Out[63]:
```
<AxesSubplot:xlabel='temp'>
```



In [64]:
```
# H0: feeling temperature has no effect on count
# Ha: feeling temperature has effect on count
f_oneway(df['atemp'], df['count'])
```

Out[64]:
```
F_onewayResult(statistic=9334.018696726596, pvalue=0.0)
```

Taking significance value as 0.05, we observe that p_value < significance value i.e. 0.0 < 0.05. Hence, we reject H0 in this case. Thus, feeling temperature has an effect on count of electric cycles rented.

In [65]:
```
pd.crosstab(df['count'],df['atemp'], normalize="index")
```

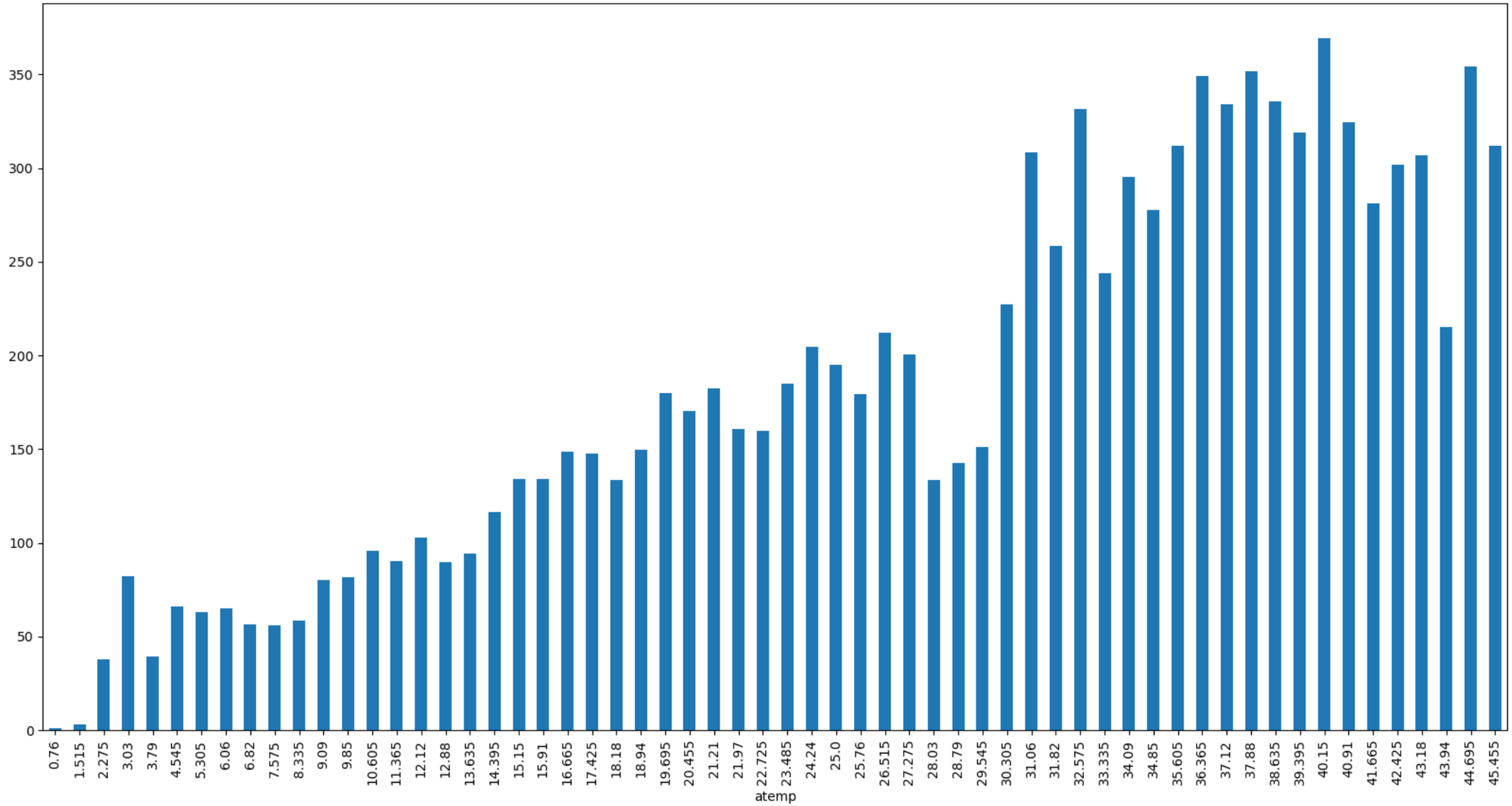
Out[65]:

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	44.695	45.455
count																					
1	0.019048	0.000000	0.009524	0.0	0.000000	0.028571	0.000000	0.038095	0.028571	0.057143	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.000000	0.000000	0.007576	0.0	0.015152	0.000000	0.000000	0.015152	0.022727	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.000000	0.006944	0.000000	0.0	0.013889	0.000000	0.006944	0.027778	0.013889	0.034722	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.000000	0.000000	0.000000	0.0	0.006711	0.000000	0.000000	0.013423	0.013423	0.020134	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.011834	0.005917	0.029586	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
943	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
948	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
968	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
970	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
977	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

822 rows × 60 columns

In [66]:
```
plt.figure(figsize=(20,10))
df.groupby('atemp')['count'].mean().plot.bar()
```

Out[66]:
```
<AxesSubplot:xlabel='atemp'>
```



Chi-square test

In [67]:
```
# H0: weather is dependent on season
# Ha: weather is not dependent on season
```

```
chi_stat, p_value, dof, expected = chi2_contingency(df['weather'], df['season'])
print(p_value)
```

```
1.0
```

Taking significance value as 0.05, we observe that p_value > significance value i.e. 1.0 > 0.05. Hence, we fail to reject H0 in this case. Thus, weather is dependent on season.

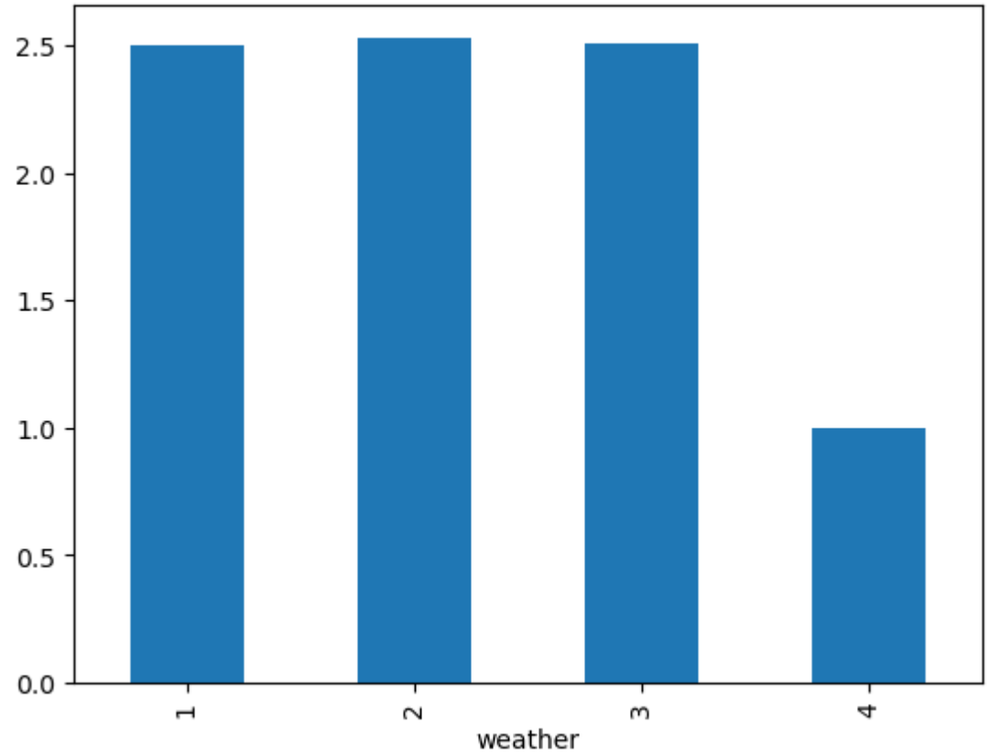
In [68]:
```
pd.crosstab(df['weather'],df['season'], normalize="index")
```

Out[68]:

season	1	2	3	4
weather				
1	0.244577	0.250417	0.268354	0.236652
2	0.252294	0.249824	0.213126	0.284757
3	0.245634	0.260768	0.231665	0.261932
4	1.000000	0.000000	0.000000	0.000000

In [69]:
```
df.groupby('weather')['season'].mean().plot.bar()
```

Out[69]:
```
<AxesSubplot:xlabel='weather'>
```



Conclusion

A. Which variables are significant in predicting the demand for shared electric cycles in the Indian market? After non-graphical and graphical analysis of the data and conducting hypothesis testing, we can conclude that:

i. Weather, season, windspeed, humidity, temperature and feeling temperature have a significant impact on the number of electric cycles rented.

ii. Holiday and working day have no significant impact on the number of electric cycles rented.

iii. Weather is dependent on season.

B. How well those variables describe the electric cycle demands?

We can observe from the analysis done in the present case study that there is high variance in the count of electric cycles rented based on weather, season, windspeed, humidity and temperature.

In [ ]: