In [51]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
```

In [41]:
```python
df = pd.read_csv("https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/428/original/bike_sharing.csv?1642089089
df
```

Out[41]:

	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

In [3]:
```python
df.info()
```

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

In [4]:
```python
df.describe()
```

Out[4]:

	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

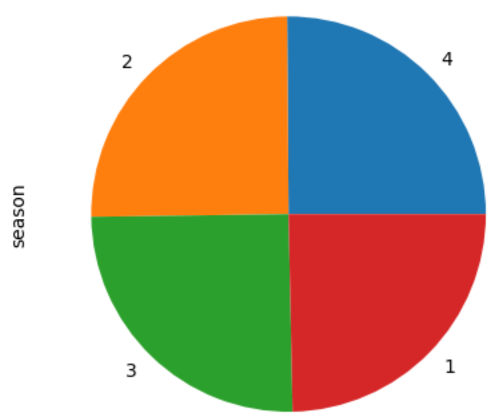
In [5]:
```
df.nunique()
```

Out[5]:
```
datetime      10886
season            4
holiday           2
workingday        2
weather           4
temp             49
atemp            60
humidity         89
windspeed        28
casual          309
registered      731
count           822
dtype: int64
```

In [6]:
```
df['season'].value_counts()
```

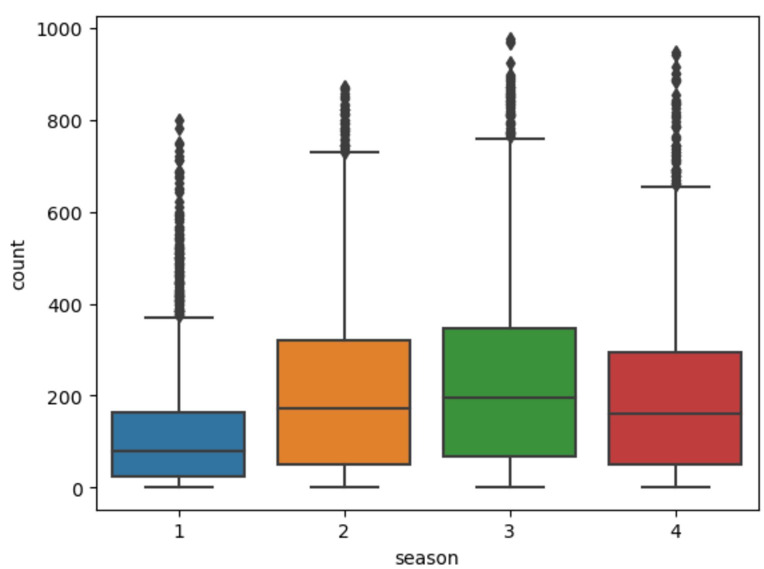
Out[6]:
```
4    2734
2    2733
3    2733
1    2686
Name: season, dtype: int64
```

In [7]:
```
df['season'].value_counts(normalize=True).plot(kind='pie')
plt.show()
```



In [8]:
```
sns.boxplot(data = df, x = 'season', y = 'count')
```

Out[8]: <AxesSubplot:xlabel='season', ylabel='count'>

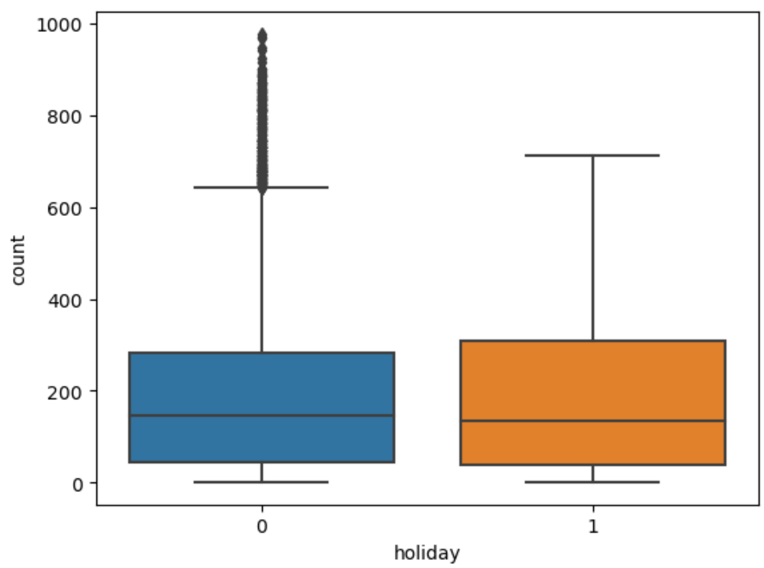


In [9]:
```python
df.groupby('season')['count'].mean()
```

Out[9]:
```
season
1    116.343261
2    215.251372
3    234.417124
4    198.988296
Name: count, dtype: float64
```

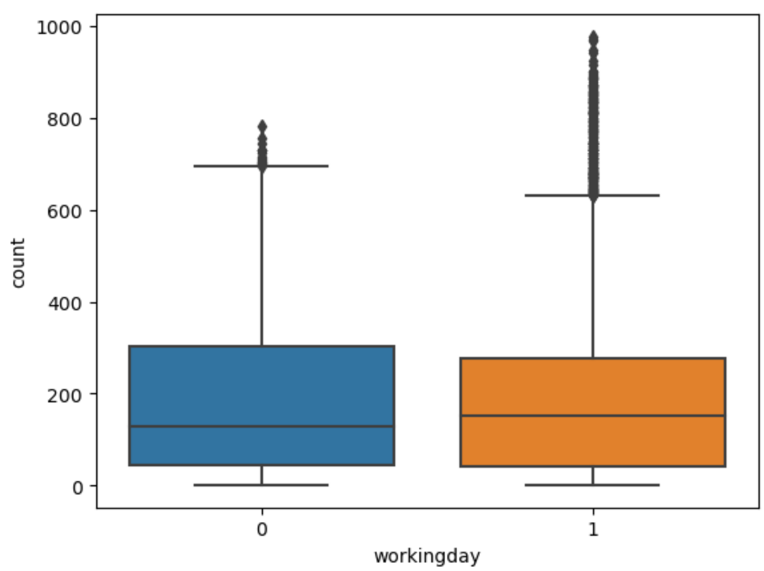
In [10]:
```python
sns.boxplot(data = df, x = 'holiday', y = 'count')
```

Out[10]: <AxesSubplot:xlabel='holiday', ylabel='count'>

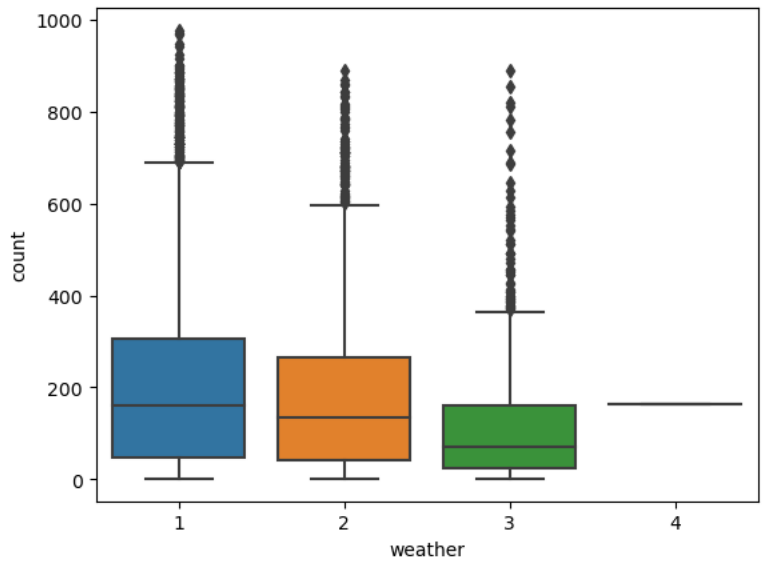


In [11]:
```python
sns.boxplot(data = df, x = 'workingday', y = 'count')
```

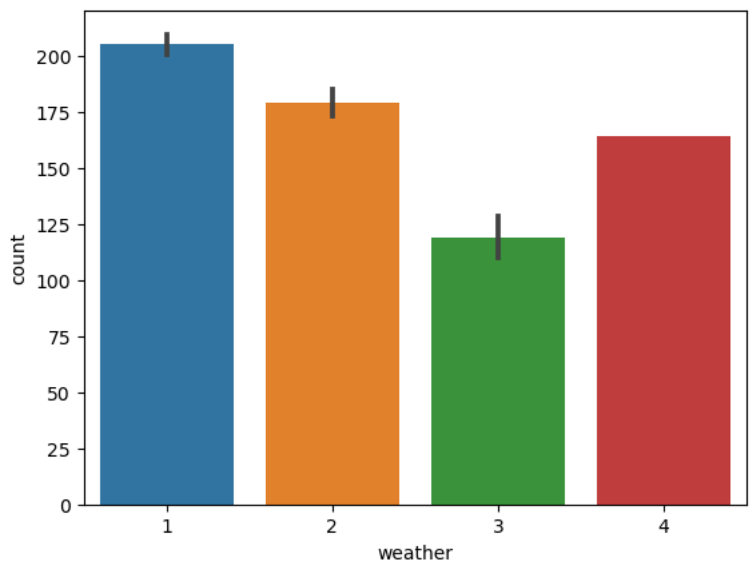
Out[11]: <AxesSubplot:xlabel='workingday', ylabel='count'>



In [12]:
```
sns.boxplot(data = df, x = 'weather', y = 'count')
```
Out[12]: <AxesSubplot:xlabel='weather', ylabel='count'>

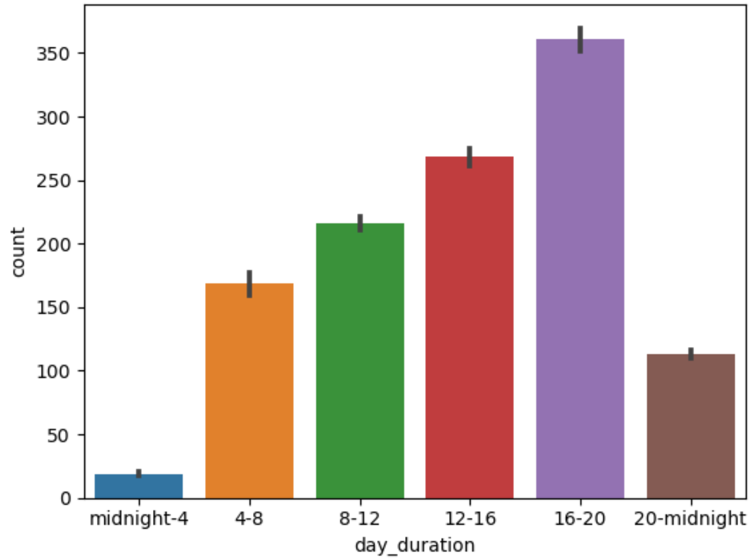


In [13]:
```
sns.barplot(data = df, x = 'weather', y = 'count')
```
Out[13]: <AxesSubplot:xlabel='weather', ylabel='count'>



In [14]:
```
df[['date','time']] = df['datetime'].str.split(' ',expand=True)
```
In [15]:
```
df[['hour','minute','sec']] = df['time'].str.split(':',expand=True)
```
In [16]:
```
df.drop(['datetime','date','minute','sec','time'],axis='columns',inplace=True)
```
In [17]:
```
df['hour']=df['hour'].astype(int)
df['hour'] = df ['hour'].replace ([00],24)
```
In [18]:
```
df['day_duration']=pd.cut(x=df['hour'],include_lowest=True, bins=[1,4,8,12,16,20,24],labels=['midnight-4','4-8','8-12','12-16
df.drop('hour',inplace=True,axis='columns')
```

In [19]:
```
sns.barplot(data = df, x = 'day_duration', y = 'count')
```
Out[19]: <AxesSubplot:xlabel='day_duration', ylabel='count'>

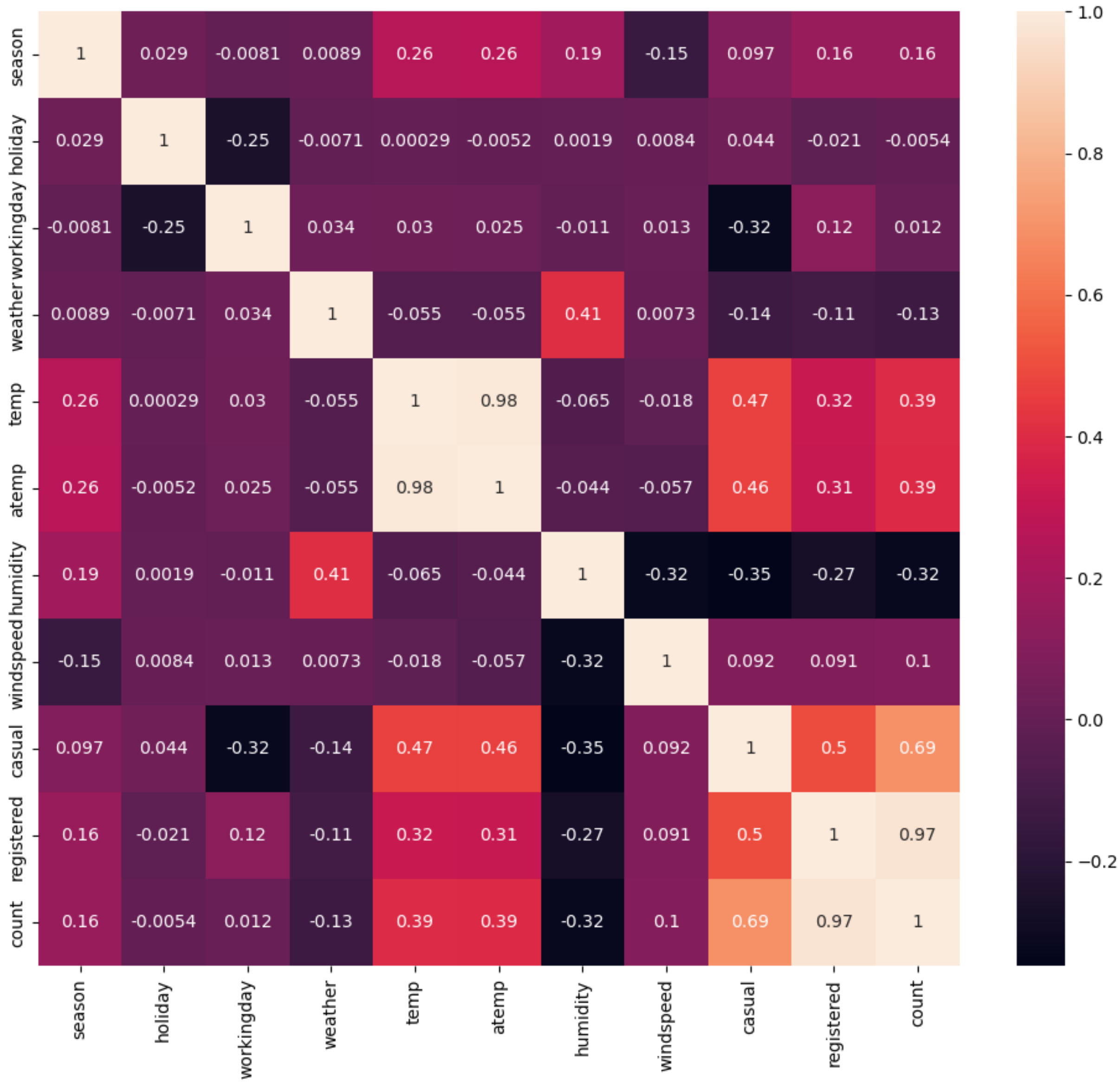


In [20]:
```
df.corr()
```
Out[20]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
season	1.000000	0.029368	-0.008126	0.008879	0.258689	0.264744	0.190610	-0.147121	0.096758	0.164011	0.163439
holiday	0.029368	1.000000	-0.250491	-0.007074	0.000295	-0.005215	0.001929	0.008409	0.043799	-0.020956	-0.005393
workingday	-0.008126	-0.250491	1.000000	0.033772	0.029966	0.024660	-0.010880	0.013373	-0.319111	0.119460	0.011594
weather	0.008879	-0.007074	0.033772	1.000000	-0.055035	-0.055376	0.406244	0.007261	-0.135918	-0.109340	-0.128655
temp	0.258689	0.000295	0.029966	-0.055035	1.000000	0.984948	-0.064949	-0.017852	0.467097	0.318571	0.394454
atemp	0.264744	-0.005215	0.024660	-0.055376	0.984948	1.000000	-0.043536	-0.057473	0.462067	0.314635	0.389784
humidity	0.190610	0.001929	-0.010880	0.406244	-0.064949	-0.043536	1.000000	-0.318607	-0.348187	-0.265458	-0.317371
windspeed	-0.147121	0.008409	0.013373	0.007261	-0.017852	-0.057473	-0.318607	1.000000	0.092276	0.091052	0.101369
casual	0.096758	0.043799	-0.319111	-0.135918	0.467097	0.462067	-0.348187	0.092276	1.000000	0.497250	0.690414
registered	0.164011	-0.020956	0.119460	-0.109340	0.318571	0.314635	-0.265458	0.091052	0.497250	1.000000	0.970948
count	0.163439	-0.005393	0.011594	-0.128655	0.394454	0.389784	-0.317371	0.101369	0.690414	0.970948	1.000000

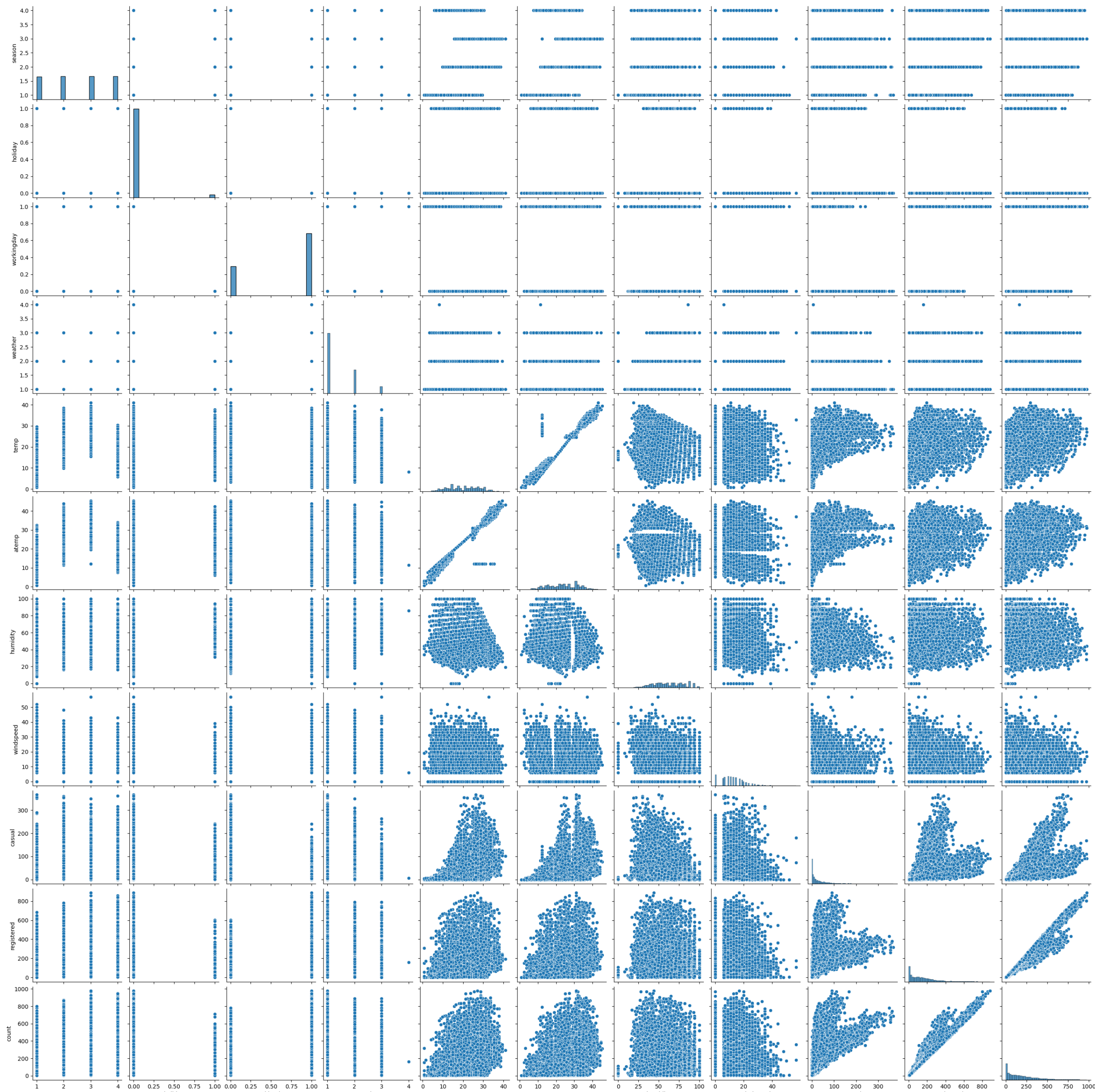
In [21]:

```
plt.figure(figsize=(12,10))
sns.heatmap(data = df.corr(), annot = True)
plt.show()
```

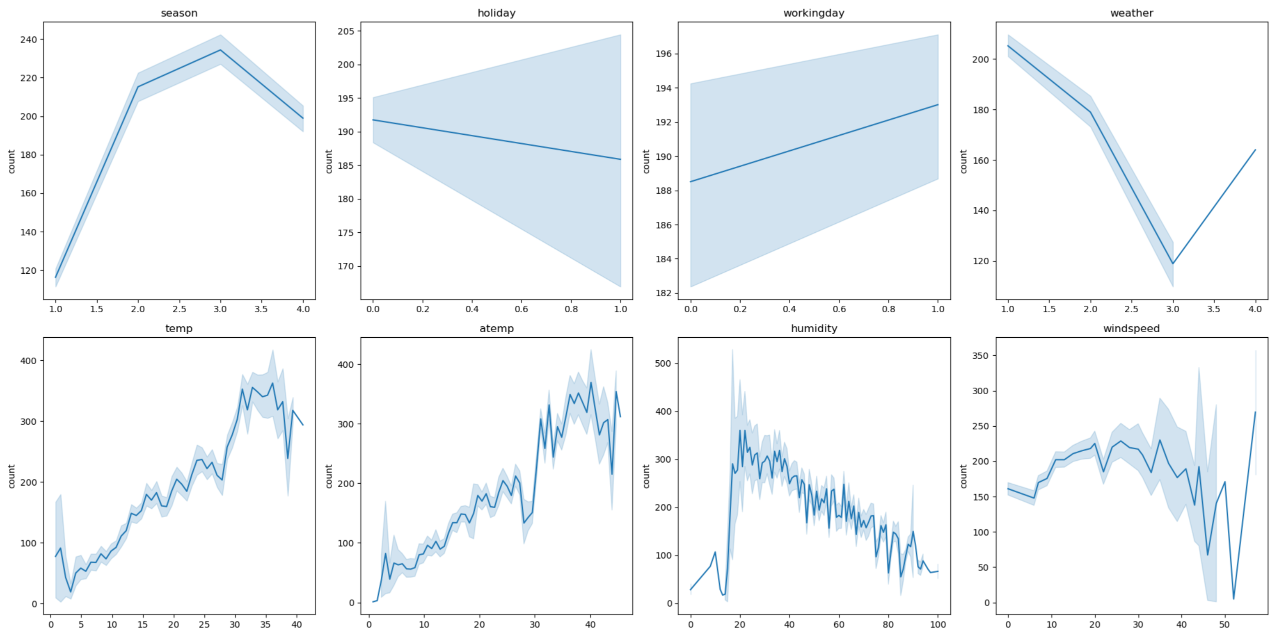


In [22]:
```
sns.pairplot(data = df)
```

Out[22]: <seaborn.axisgrid.PairGrid at 0x2237b4a8e20>



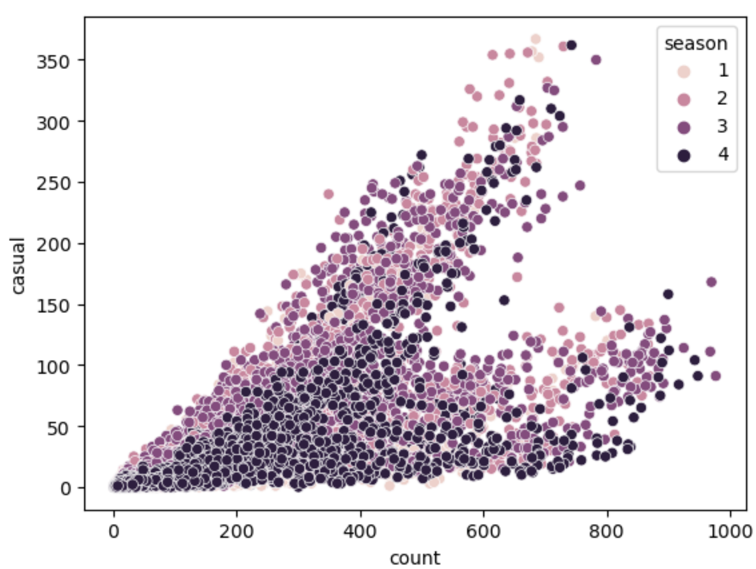
In [23]:
```
fig, axes = plt.subplots(nrows=2, ncols=4, figsize=(20,10))

categorical_features=['season', 'holiday', 'workingday','weather','temp','atemp','humidity','windspeed']

for i, ax in enumerate(axes.flatten()):
    sns.lineplot(x=categorical_features[i], y='count', data=df, ax=ax)
    ax.set_xlabel('')
    ax.set_ylabel('count')
    ax.set_title(categorical_features[i])

fig.subplots_adjust(hspace=0.5, wspace=0.3)
plt.tight_layout()
plt.show()
```



We can clearly see that count is linearly related to workingday, temp, atemp and inversely related to humidity, holiday, windspeed.

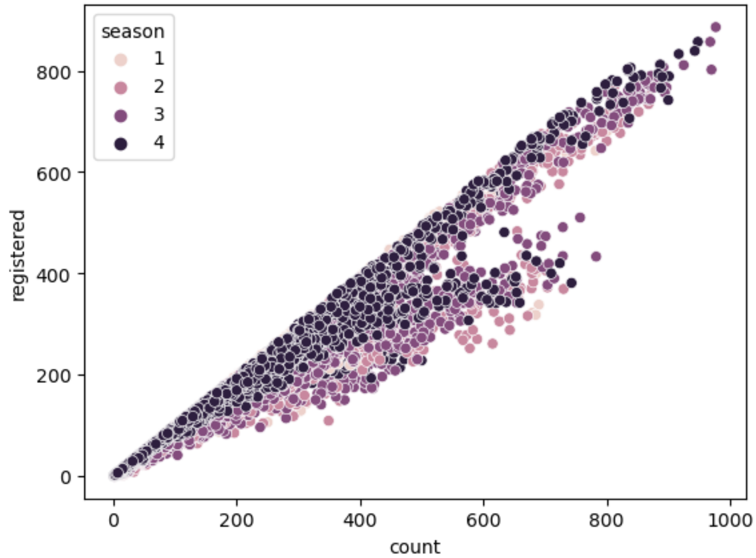
In [24]:
```
sns.scatterplot(data = df, x = 'count', y = 'casual', hue = 'season')
```

Out[24]: <AxesSubplot:xlabel='count', ylabel='casual'>



It is a clear case of heteroscedasticity, where the range is extremely large. Also, due to which we can only conclude that not large but moderate amoung of users tend to take service in fall and winter.

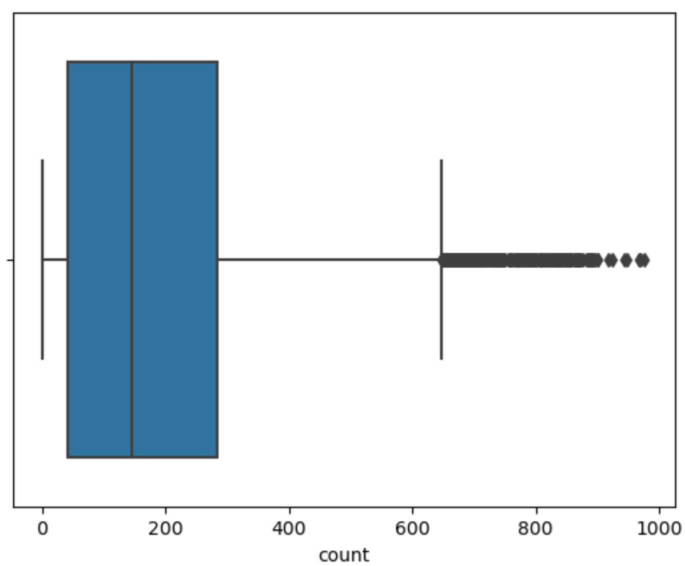
In [25]:
```
sns.scatterplot(data = df, x = 'count', y = 'registered', hue = 'season')
```
Out[25]: `<AxesSubplot:xlabel='count', ylabel='registered'>`



We can clearly see that it is a condition of homoscedasticity, where the random variables have the same finite variance.

With which we can also have an impression that the count of registered users has a positive response in each season, having said that there is a clear overlapping of two seasons complemented by the hue of 3 and 4 respectively.

In [52]:
```
sns.boxplot(df['count'])
```
```
D:\games\Anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```
Out[52]: `<AxesSubplot:xlabel='count'>`



In [53]:
```
q1 = df['count'].quantile(0.25)
q3 = df['count'].quantile(0.75)
iqr = q3 - q1
outlier = df[(df['count']<q1-1.5*iqr) | (df['count']>q3+1.5*iqr)]
outlier.shape
```
Out[53]: (300, 13)

Significance level used is 0.05

Chi-squared test for checking the significance of seasons over weather.

We are checking if weather and season has a relation.

Null Hypothesis: Weather proportion is same across all season

Alternate Hypothesis: Weather proportions is different across different seasons

In [27]:
```
data_table = pd.crosstab(df['season'], df['weather'])
print("Observed values:")
data_table
```

```
Observed values:
```

Out[27]:

weather season	1	2	3	4
1	1759	715	211	1
2	1801	708	224	0
3	1930	604	199	0
4	1702	807	225	0

In [28]:
```
from scipy import stats

val = stats.chi2_contingency(data_table)
expected_values = val[3]
expected_values
```

Out[28]:
```
array([[1.77454639e+03, 6.99258130e+02, 2.11948742e+02, 2.46738931e-01],
       [1.80559765e+03, 7.11493845e+02, 2.15657450e+02, 2.51056403e-01],
       [1.80559765e+03, 7.11493845e+02, 2.15657450e+02, 2.51056403e-01],
       [1.80625831e+03, 7.11754180e+02, 2.15736359e+02, 2.51148264e-01]])
```

In [29]:
```
nrows, ncols = 4, 4
dof = (nrows-1)*(ncols-1)
print("degrees of freedom: ", dof)
alpha = 0.05


chi_sqr = sum([(o-e)**2/e for o, e in zip(data_table.values, expected_values)])
chi_sqr_statistic = chi_sqr[0] + chi_sqr[1]
print("chi-square test statistic: ", chi_sqr_statistic)

critical_val = stats.chi2.ppf(q=1-alpha, df=dof)
print(f"critical value: {critical_val}")

p_val = 1-stats.chi2.cdf(x=chi_sqr_statistic, df=dof)
print(f"p-value: {p_val}")

if p_val <= alpha:
    print("\nSince p-value is less than alpha 0.05, we reject the Null Hypothesis. \nMeans : Weather is dependent on the seas
else:
    print("Since p-value is greater than the alpha 0.05, we fail to reject the Null Hypothesis")
```

```
degrees of freedom:  9
chi-square test statistic:  44.09441248632364
critical value: 16.918977604620448
p-value: 1.3560001579371317e-06

Since p-value is less than alpha 0.05, we reject the Null Hypothesis.
Means : Weather is dependent on the season.
```

Normality test for count.

Shapiro test

Null Hypothesis: count follows normal distribution

Alternative Hypothesis: count doesn't follow normal distribution

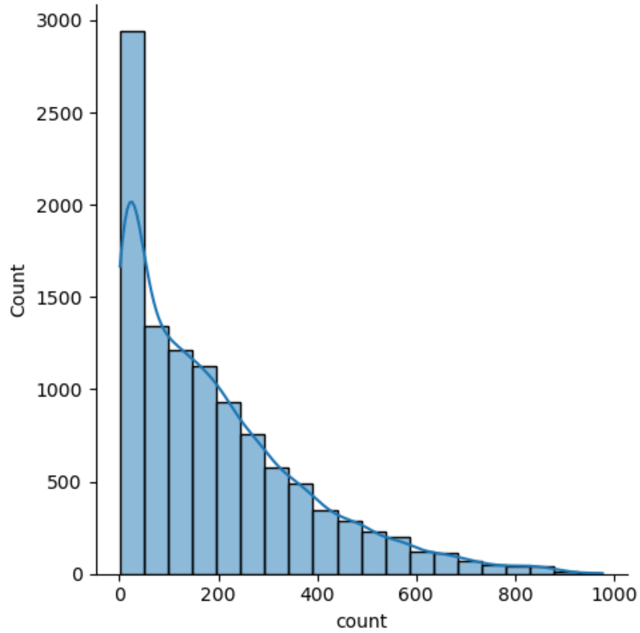
In [44]:
```
from scipy.stats import shapiro

w, p_value = shapiro(df['count'])
print('The p-value is', p_value)
```

```
The p-value is 0.0
```

Cannot conclude anything from Shapiro test, we will continue our analysis.

In [31]:
```
sns.displot(df['count'], bins=20, kde=True)
```

Out[31]: <seaborn.axisgrid.FacetGrid at 0x22306f58ac0>

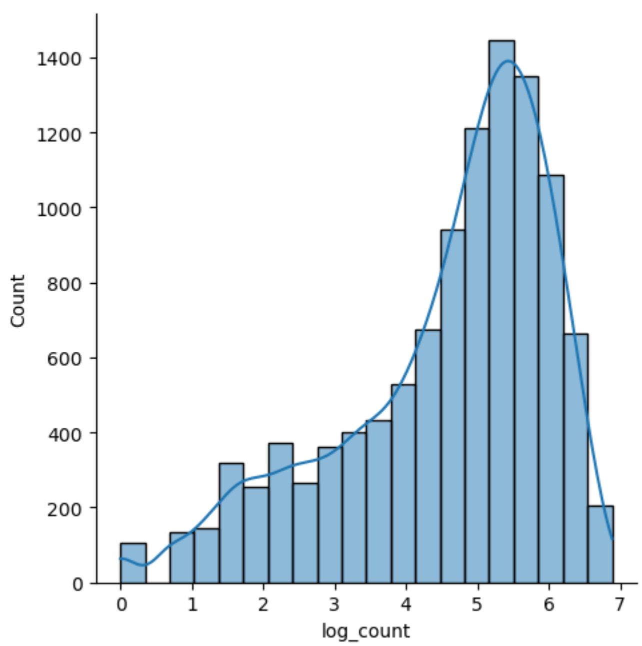


It is evident from the graph, that the data is not gausian.

In [47]:
```
df['log_count']=np.log(df['count'])
```

In [48]:
```
sns.displot(df['log_count'], bins = 20, kde = True)
```

Out[48]: <seaborn.axisgrid.FacetGrid at 0x22306ca8850>

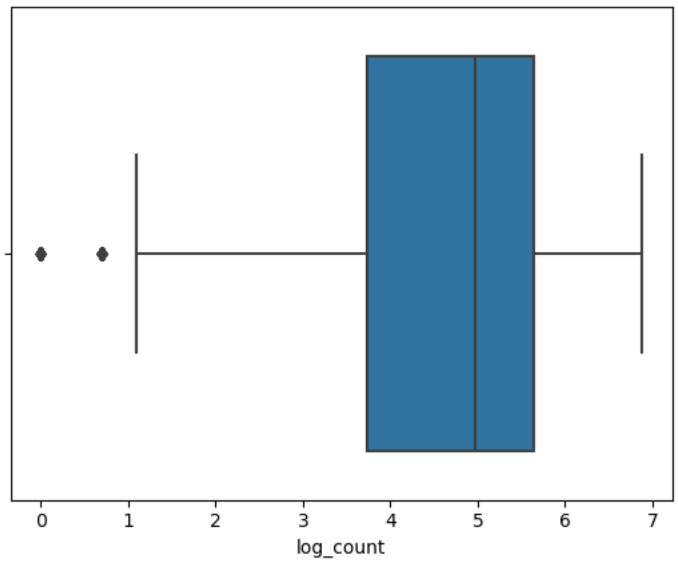


Even after taking log, we can not say that the data has a susceptive distribution, to work with.

In [49]:
```
sns.boxplot(df['log_count'])
```

```
D:\games\Anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit key
word will result in an error or misinterpretation.
  warnings.warn(
```

Out[49]:
```
<AxesSubplot:xlabel='log_count'>
```



After taking log though, it is clear that the reduction of outlier is quite significant.

We are checking if there is an equality of variances.

Levene’s test

Null hypothesis : All the count variances are equal

Alternative hypothesis : At least one variance is different from the rest

In [34]:
```
df.groupby('season')['log_count'].describe()
```

Out[34]:

season	count	mean	std	min	25%	50%	75%	max
1	2686.0	3.984206	1.539737	0.0	3.178054	4.356709	5.099866	6.685861
2	2733.0	4.703267	1.462172	0.0	3.891820	5.147494	5.771441	6.771936
3	2733.0	4.860311	1.378662	0.0	4.219508	5.273000	5.849325	6.884487
4	2734.0	4.652650	1.421134	0.0	3.931826	5.081404	5.683580	6.854355

In [35]:
```
from scipy.stats import levene

alpha = 0.05

statistic, p_value = levene(
 df[df['season']==1]['log_count'].sample(2686),
 df[df['season']==2]['log_count'].sample(2686),
 df[df['season']==3]['log_count'].sample(2686),
 df[df['season']==4]['log_count'].sample(2686)
 )
print('The p-value is ',p_value)

if p_val <= alpha:
    print("\nSince p-value is less than alpha 0.05, we reject the Null Hypothesis. \nMeans : At least one variance is differe
else:
    print("Since p-value is greater than the alpha 0.05, we fail to reject the Null Hypothesis")
```

```
The p-value is  1.562855150355551e-06

Since p-value is less than alpha 0.05, we reject the Null Hypothesis.
Means : At least one variance is different.
```

We are checking if the count changes with the change in season.

ANOVA test

Null Hypothesis : Count in each season is same

Alternative Hypothesis : Count in each season is not the same

In [36]:
```
from scipy.stats import f_oneway

alpha = 0.5

test_stat, p_value = f_oneway(
 df[df['season']==1]['log_count'].sample(2686),
 df[df['season']==2]['log_count'].sample(2686),
 df[df['season']==3]['log_count'].sample(2686),
 df[df['season']==4]['log_count'].sample(2686))
print('The p-value is', p_value)

if p_val <= alpha:
    print("\nSince p-value is less than alpha 0.05, we reject the Null Hypothesis. \nMeans : Count in each season is not the
else:
    print("Since p-value is greater than the alpha 0.05, we fail to reject the Null Hypothesis")
```

```
The p-value is 3.449826047976868e-120

Since p-value is less than alpha 0.05, we reject the Null Hypothesis.
Means : Count in each season is not the same.
```

Cheking for a statistical difference between working and non working days.

T test

Null Hypothesis: Count on a weekday is equal to count on a weekend

Alternate Hypothesis: Count on weekday is not equal to count on the weekend

In [37]:
```
df.groupby('workingday')['log_count'].describe()
```

Out[37]:

workingday	count	mean	std	min	25%	50%	75%	max
0	3474.0	4.591984	1.381237	0.0	3.784190	4.85203	5.717028	6.663133
1	7412.0	4.534084	1.536713	0.0	3.713572	5.01728	5.624018	6.884487

In [38]:
```
alpha = 0.05

working_day = df[df['workingday']==1]['log_count'].sample(3474)
non_working_day = df[df['workingday']==0]['log_count'].sample(3474)

t_static, p_value = stats.ttest_ind(working_day, non_working_day, alternative = 'two-sided')
print("Test statistic = {} , P value = {} ".format(t_static, p_value))

if p_val <= alpha:
    print("\nSince p-value is less than alpha 0.05, we reject the Null Hypothesis. \nMeans : There is a statistical differenc
else:
    print("Since p-value is greater than the alpha 0.05, we fail to reject the Null Hypothesis")
```

```
Test statistic = -1.4519333631679476 , P value = 0.14656529993675302

Since p-value is less than alpha 0.05, we reject the Null Hypothesis.
Means : There is a statistical difference between the count of working days vs non working days.
```

In [ ]: