In [38]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Problem Statement: Factors on which the demand for the shared electric cycles depends.

Specifically, the company want to understand the factors affecting the demand and user counts for these shared electric cycles in the Indian market.

Which variables are significant in predicting the demand for shared electric cycles in the Indian market?

How well these variables describe the electric cycle demands

In [3]:
```
df=pd.read_csv("/Users/praneetcb/Documents/yulu.csv") #Importing a
```

In [4]:
```
df.head()
```

Out[4]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	ca
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	

In [ ]:
```
#Analysing the basic metrics and identifying variable and data type
```

In [ ]:
```
#Let's Understand what these number represent in the above columns:

#season: 1: spring, 2: summer, 3: fall, 4: winter
#workingday: if day is neither weekend nor holiday is 1, otherwise
#weather:#1: Clear, Few clouds, partly cloudy, partly cloudy
        #2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds,
        #3: Light Snow, Light Rain + Thunderstorm + Scattered cloud
        #4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow +
#count: count of total rental bikes including both casual and regis
```

In [5]:
```
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 [13]:
```
df.isna().sum()/len(df)*100 #Checking Null values (no missing value
```

Out[13]:
```
datetime      0.0
season        0.0
holiday       0.0
workingday    0.0
weather       0.0
temp          0.0
atemp         0.0
humidity      0.0
windspeed     0.0
casual        0.0
registered    0.0
count         0.0
dtype: float64
```

In [21]:
```
df.dtypes
```

Out[21]:
```
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 [22]:
```
# We see that datetime is in object type, Let's convert into dateti
df['datetime']=pd.to_datetime(df['datetime'])
```

In [29]:
```
df.datetime.dtypes
```

Out[29]:
```
dtype('<M8[ns]')
```

In [36]:
```
df.nunique()
```

Out[36]:
```
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 [241]:
```
df['weather'].value_counts()
```

Out[241]:
```
weather
1    7192
2    2834
3     859
4       1
Name: count, dtype: int64
```

In [243]:
```
df['workingday'].value_counts()
```

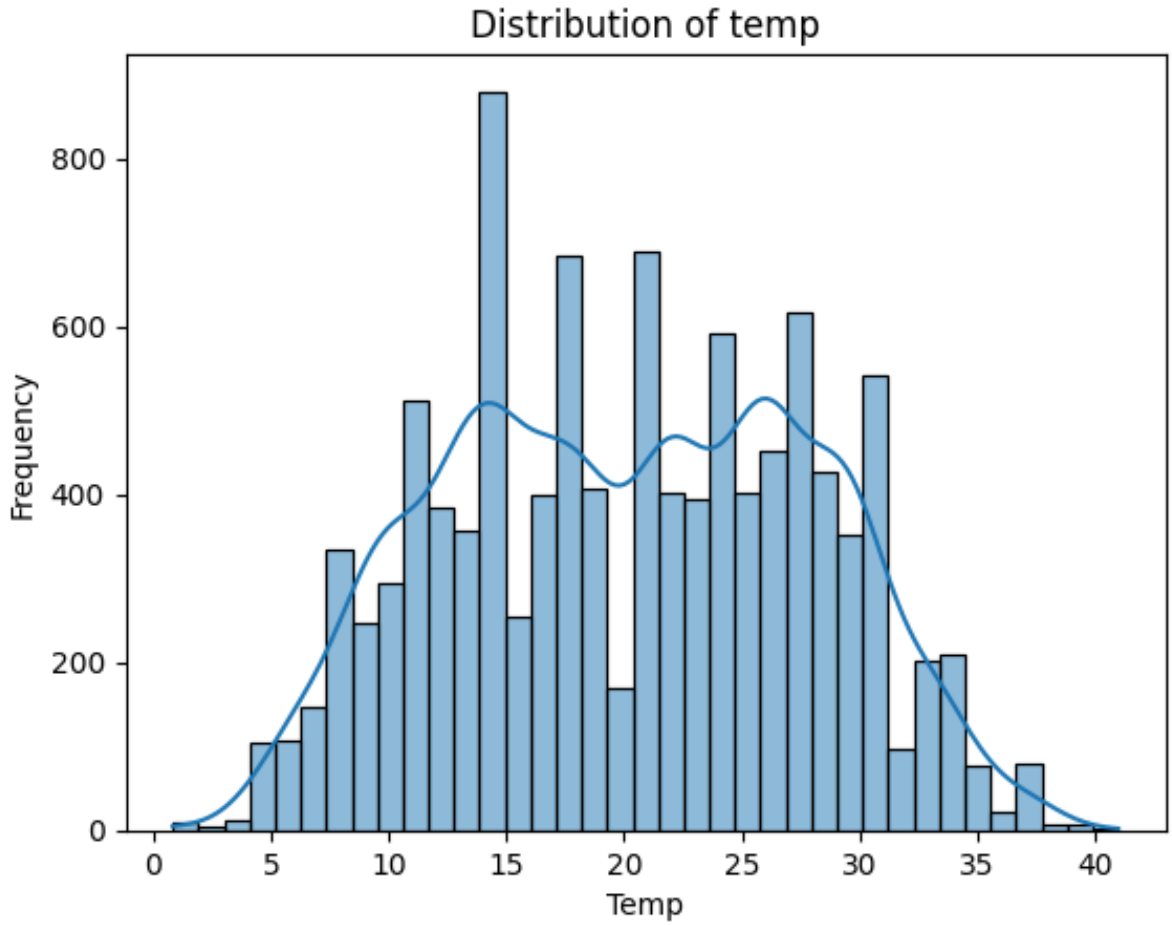
Out[243]:
```
workingday
1    7412
0    3474
Name: count, dtype: int64
```

In [242]:
```
df['season'].value_counts()
```

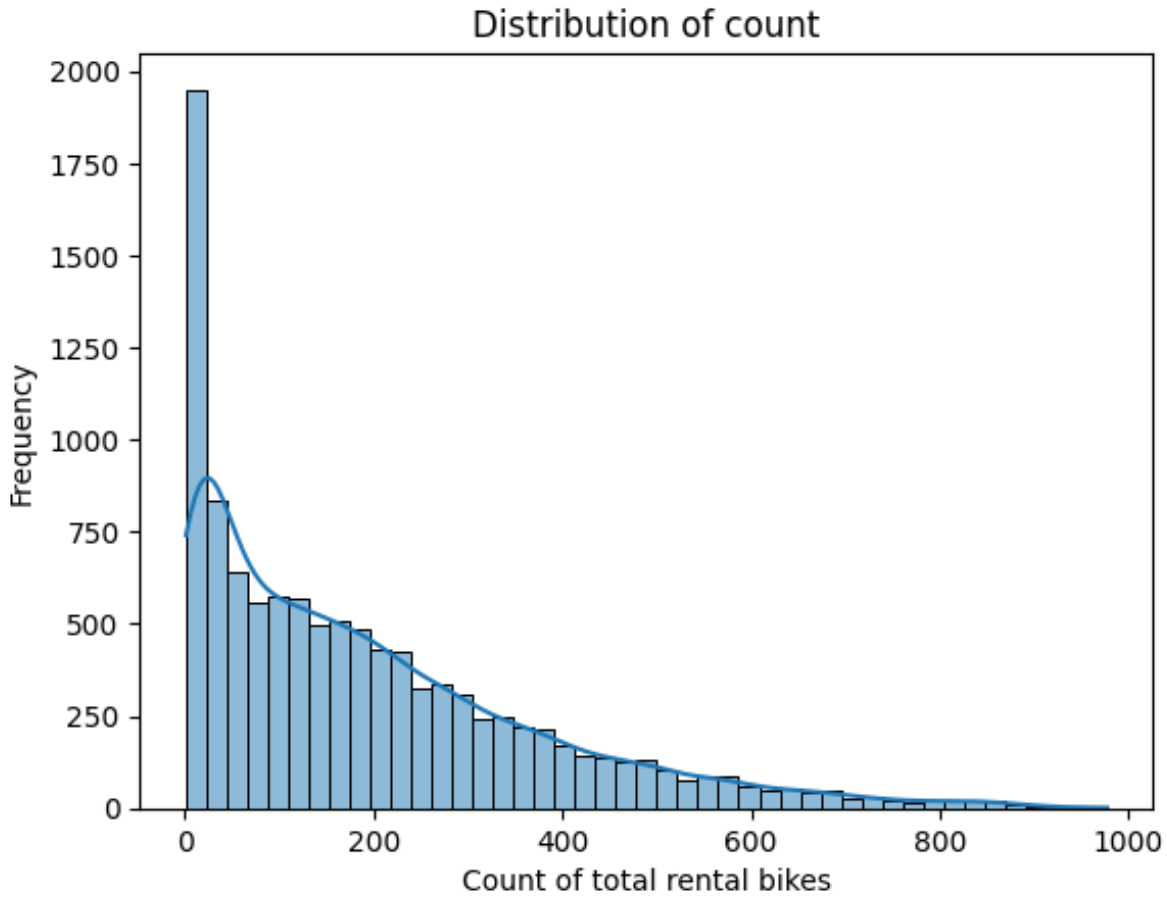
Out[242]:
```
season
4    2734
2    2733
3    2733
1    2686
Name: count, dtype: int64
```

Univarite Analysis

In [47]:
```
# Continuous Variables - Distribution Plots

#To find the distribution of temperature
sns.histplot(data=df, x='temp', kde=True)
plt.title('Distribution of temp')
plt.xlabel('Temp')
plt.ylabel('Frequency')
plt.show()
```

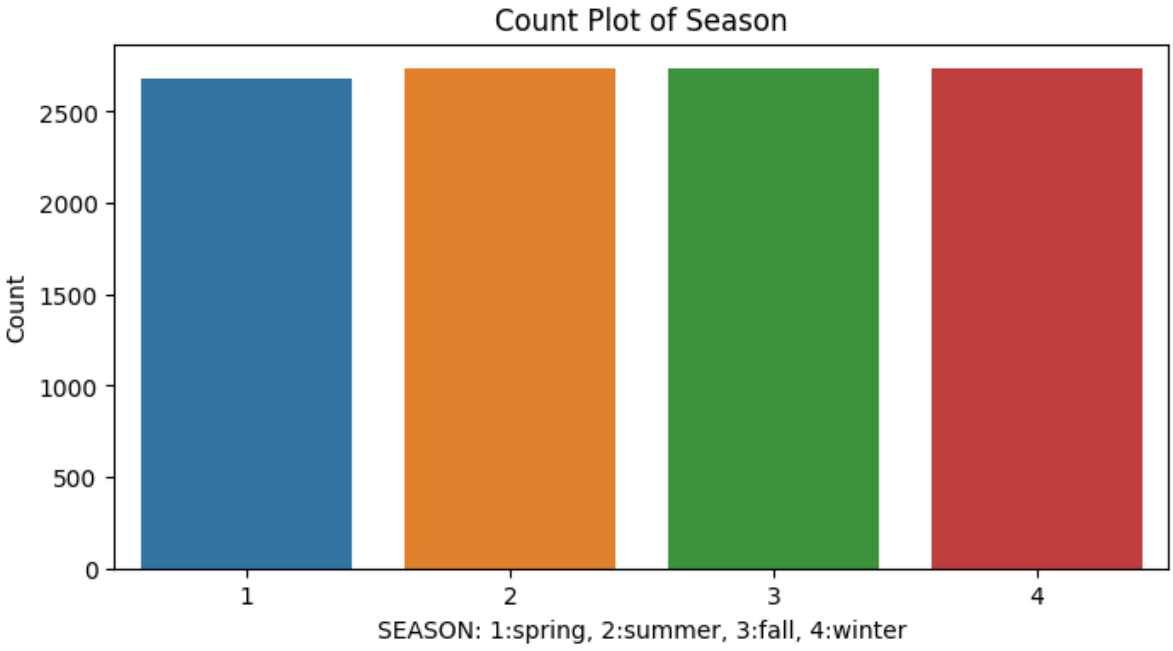


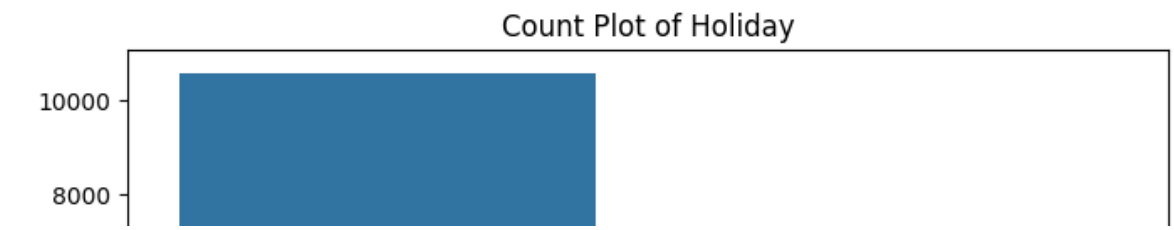
In [48]:
```
#To find the distribution of Count/number of total rental bikes

sns.histplot(data=df, x='count', kde=True)
plt.title('Distribution of count')
plt.xlabel('Count of total rental bikes')
plt.ylabel('Frequency')
plt.show()
```

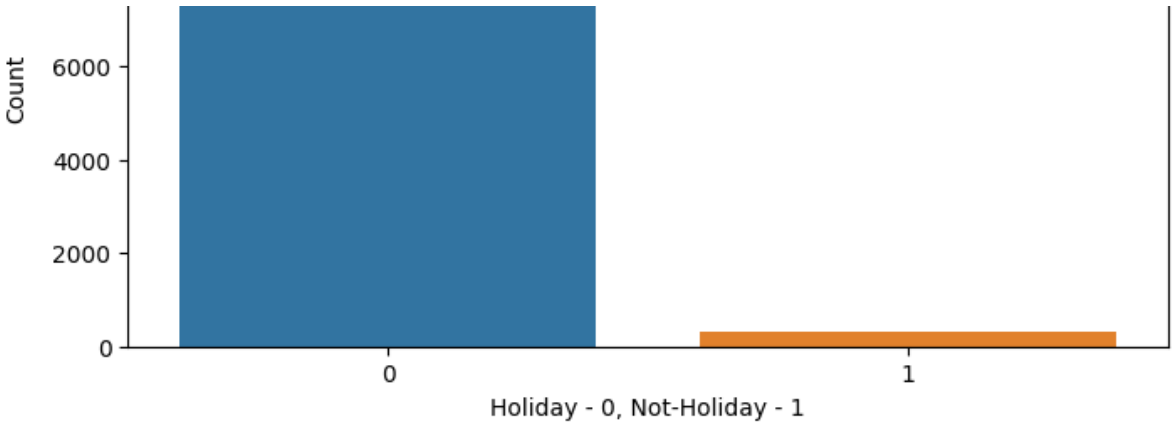


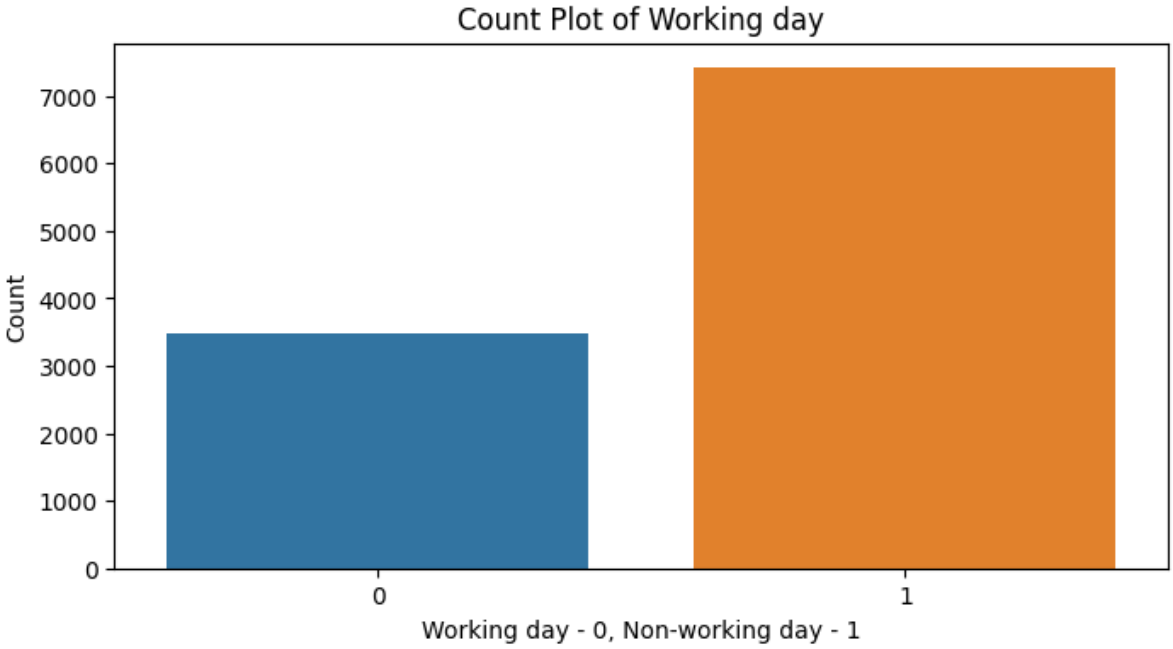
In [64]:

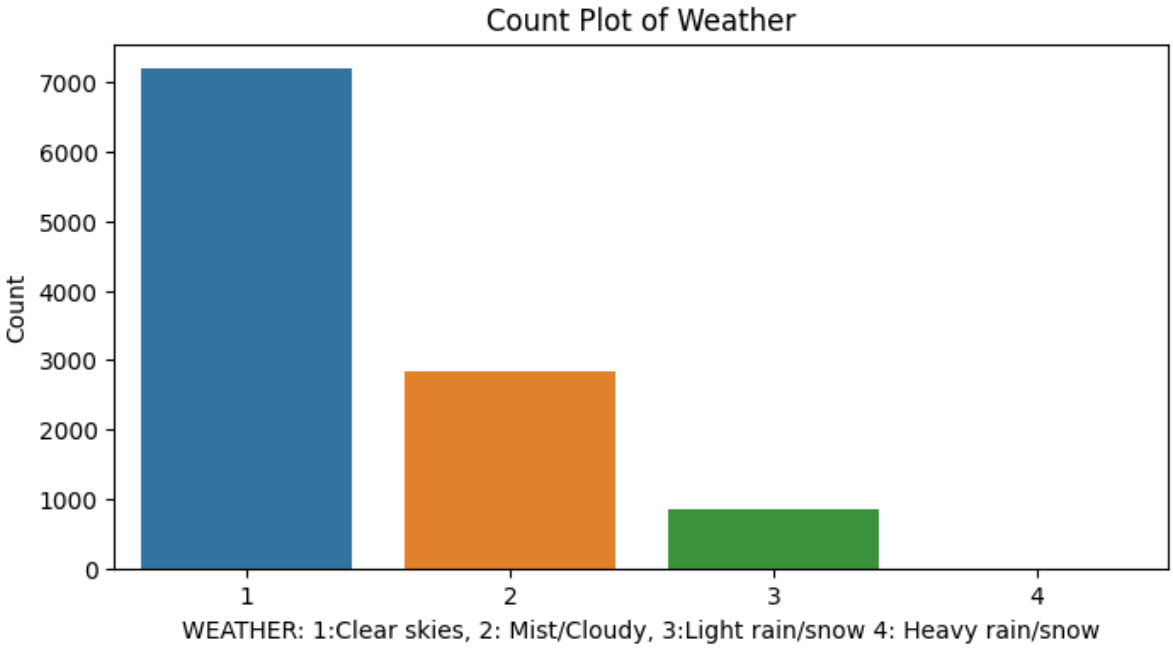
```
# Categorical Variables - Bar Plots/Count Plots

# Countplot for all the categorical_vars = 'Season', 'Holiday', 'Wo

plt.figure(figsize=(8, 4))
sns.countplot(data=df, x='season')
plt.title('Count Plot of Season')
plt.xlabel('SEASON: 1:spring, 2:summer, 3:fall, 4:winter')
plt.ylabel('Count')
plt.show()

plt.figure(figsize=(8, 4))
sns.countplot(data=df, x='holiday')
plt.title('Count Plot of Holiday')
plt.xlabel('Holiday - 0, Not-Holiday - 1')
plt.ylabel('Count')
plt.show()

plt.figure(figsize=(8, 4))
sns.countplot(data=df, x='workingday')
plt.title('Count Plot of Working day')
plt.xlabel('Working day - 0, Non-working day - 1')
plt.ylabel('Count')
plt.show()

plt.figure(figsize=(8, 4))
sns.countplot(data=df, x='weather')
plt.title('Count Plot of Weather')
plt.xlabel('WEATHER: 1:Clear skies, 2: Mist/Cloudy, 3:Light rain/sn
plt.ylabel('Count')
plt.show()
```





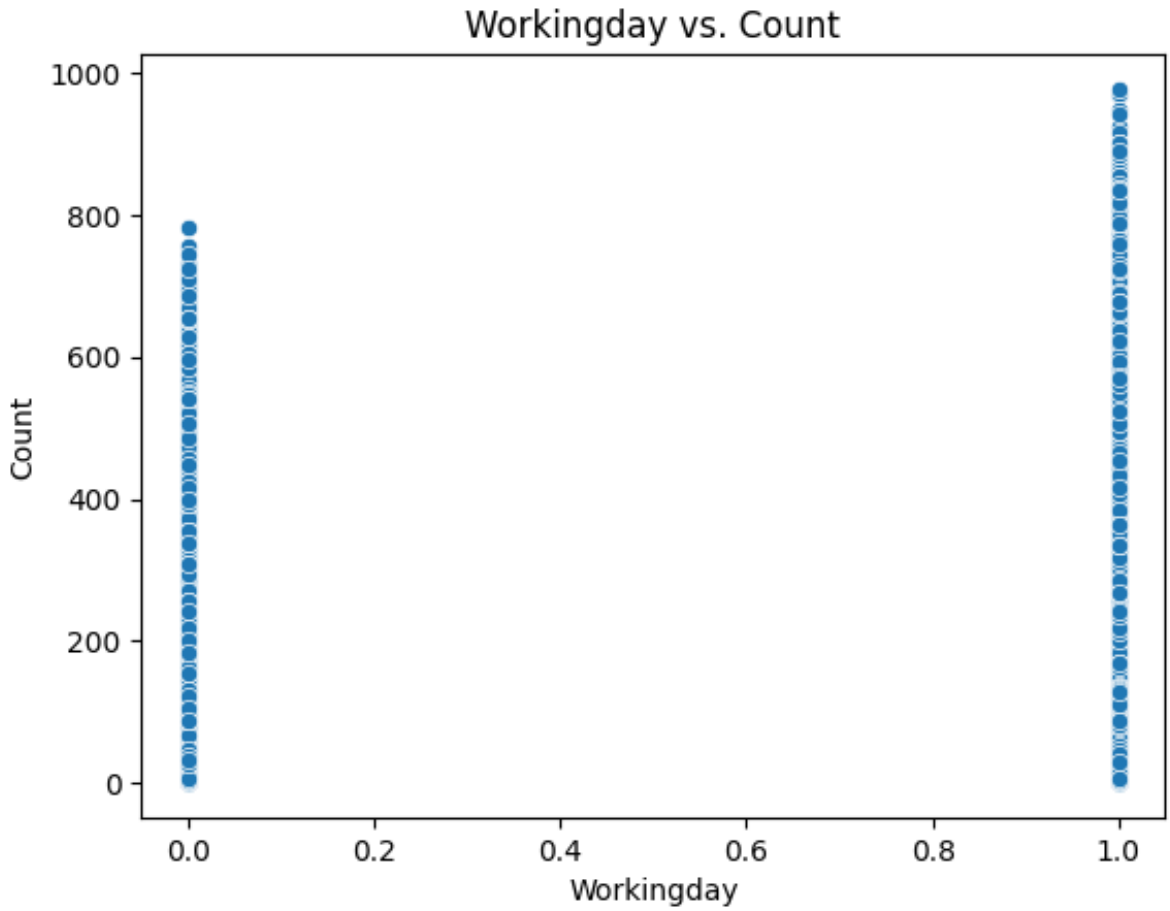




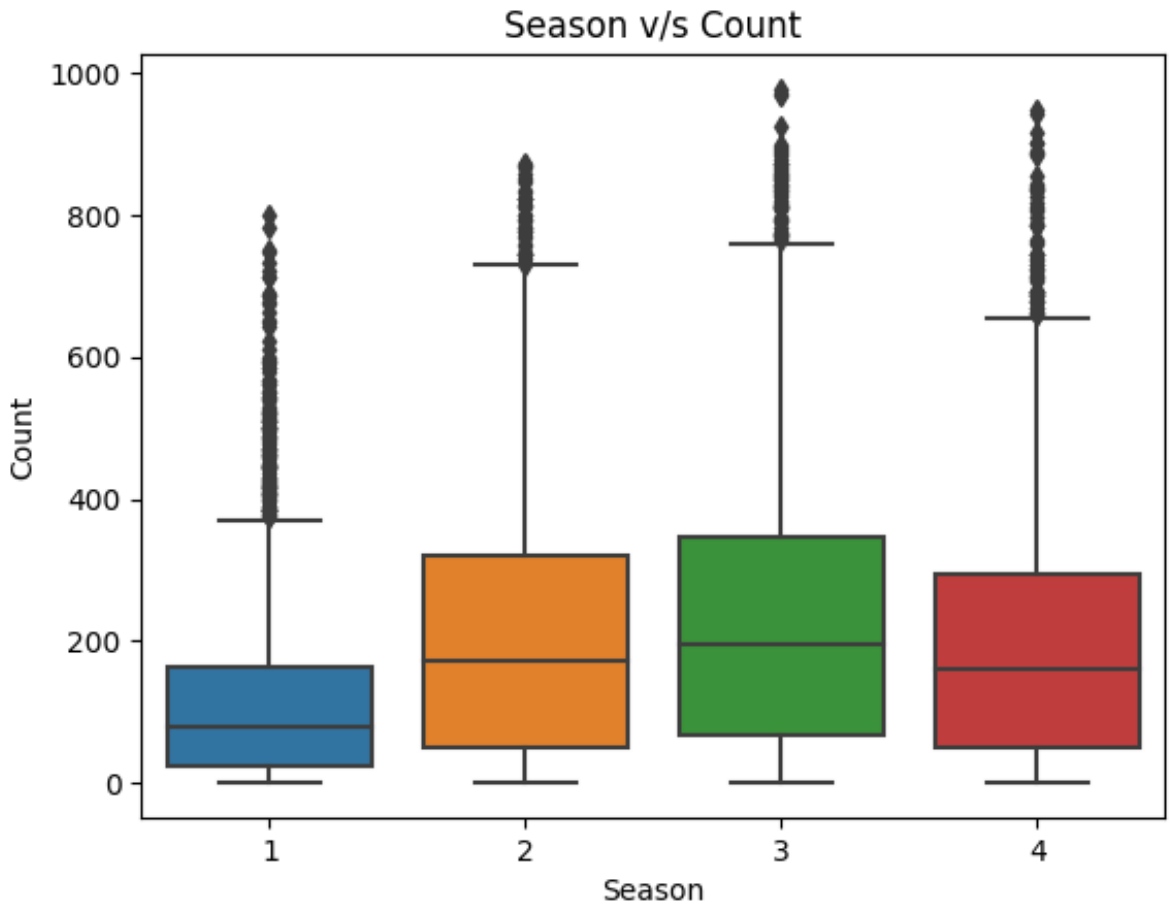


Bivarite Analysis

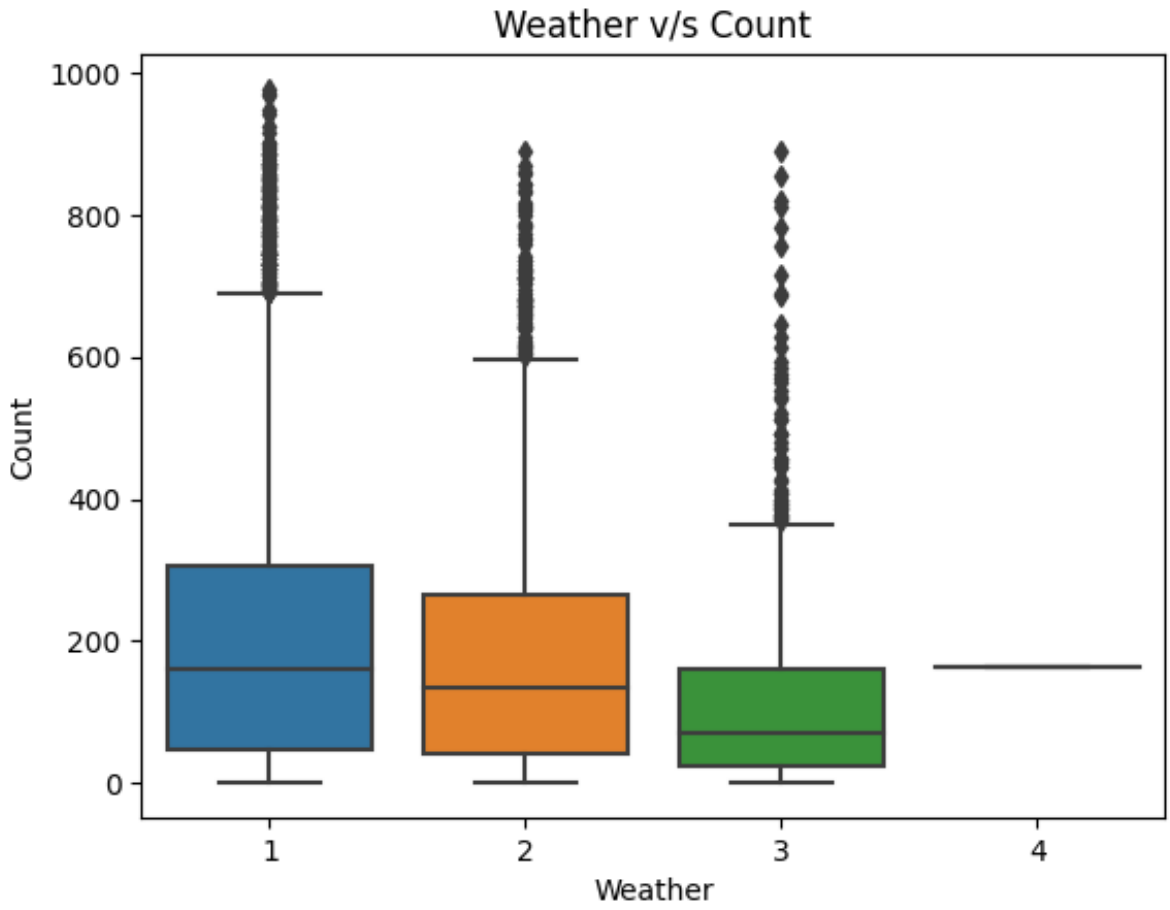
In [66]:
```
# Scatter Plot: Workingday vs Count

sns.scatterplot(data=df, x='workingday', y='count')
plt.title('Workingday vs. Count')
plt.xlabel('Workingday')
plt.ylabel('Count')
plt.show()
```



In [69]:
```
#Boxplot: Season v/s Count

sns.boxplot(data=df, x='season', y='count')
plt.title('Season v/s Count')
plt.xlabel('Season')
plt.ylabel('Count')
plt.show()
```



In [71]:
```
#Boxplot: Weather v/s Count

sns.boxplot(data=df, x='weather', y='count')
plt.title('Weather v/s Count')
plt.xlabel('Weather')
plt.ylabel('Count')
plt.show()
```



Statistical Tests

T-Test

In [ ]:
```
# Let's perform some Statistical Tests to check how factors effect
```

In [73]:
```
# Importing essential libraries
from scipy.stats import ttest_ind
```

In [ ]:
```
# Test to check if Working Day has effect on number of cycles rente

# Ho: Working day has no effect on the number of cycles rented
# Ha: Working day has effect on the number of cycles rented

# We can perform 2 sample T-test
```

In [79]:
```
working_day=df[df['workingday']==1]['count']
nonworking_day=df[df['workingday']==0]['count']

t_stat,p_value=ttest_ind(working_day,nonworking_day)

print('2-Sample t-test: Workingday vs. Non-workingday')
print('t-statistic:', t_stat)
print('p-value:', p_value)
```

```
2-Sample t-test: Workingday vs. Non-workingday
t-statistic: 1.2096277376026694
p-value: 0.22644804226361348
```

In [80]:
```
# Set a significance level
alpha=0.05

if p_value>alpha:
    print('We Fail to Reject the Null Hypothesis')
else:
    print('We Reject the Null Hypothesis')
```

```
We Fail to Reject the Null Hypothesis
```

Working day has no effect on the number of cycles rented

ANOVA tests

In [ ]:
```
# Check test Assumptions on Season
```

In [188]:
```
# To meet the asssumptions of ANOVA test we can perfom Shapiro Wilk
# To check the normality and homogenity of variance
from scipy.stats import shapiro, levene

#Shapiro_test
#Ho: Data follows normal distribution
#Ha: Data does not follow normal distribution

#levene_test
# Ho: The variances of the groups being compared are equal.
# Ha: The variances of the groups being compared are not equal.
```

In [239]:
```
# Extract data for each season
data = [df[df['season'] == season]['count'] for season in seasons]
seasons=df['season'].unique()

# Shapiro-Wilk Test for Normality
for season, count_data in zip(seasons, data):
    sh, p = shapiro(count_data)
    print(f"Shapiro-Wilk Test - Season {season}")
    print("p-value:", p)
    print()
```

```
Shapiro-Wilk Test - Season 1
p-value: 0.0

Shapiro-Wilk Test - Season 2
p-value: 6.039093315091269e-39

Shapiro-Wilk Test - Season 3
p-value: 1.043458045587339e-36

Shapiro-Wilk Test - Season 4
p-value: 1.1301682309549298e-39
```

In [ ]:
```
# p_values are less than 0.05 hence we reject the null hypothesis -
```

In [236]:
```
# Levene's Test for check Homogeneity of Variance

lev, p_val = levene(*data)
print("Levene's Test - Homogeneity of Variance")
print("p-value:", p_val)
```

```
Levene's Test - Homogeneity of Variance
p-value: 1.0147116860043298e-118
```

In [ ]:
```
# p_value is less than 0.05 so The variances of the groups being co
```

In [98]:
```
# Importing essential libraries
from scipy.stats import f_oneway,kruskal
```

In [ ]:
```
# Test to check if Season has effect on number of cycles rented

# Ho: Season has no effect on the numebr of cycles rented
# Ha: Season has effect on the number of cycles rented

# We can perform ANNOVA test here as there are two or more groups
```

In [94]:
```
seasons=df['season'].unique()
s1=df[df['season']==seasons[0]]['count']
s2=df[df['season']==seasons[1]]['count']
s3=df[df['season']==seasons[2]]['count']
s4=df[df['season']==seasons[3]]['count']
```

In [96]:
```
f_stats,p_value=f_oneway(s1,s2,s3,s4)

print('ANOVA: Season v/s Count')
print('f-statistic:', f_stats)
print('p-value:', p_value)
```

```
ANOVA: Season v/s Count
f-statistic: 236.94671081032106
p-value: 6.164843386499654e-149
```

In [110]:
```
# Set a significance level
alpha=0.05

if p_value>alpha:
    print('We Fail to Reject the Null Hypothesis')
else:
    print('We Reject the Null Hypothesis')
```

```
We Reject the Null Hypothesis
```

In [112]:
```
#Lets also validate the test with Kruskal's test as assumptions of

f_stats,p_value=kruskal(s1,s2,s3,s4)

# Set a significance level
alpha=0.05

if p_value>alpha:
    print('We Fail to Reject the Null Hypothesis')
else:
    print('We Reject the Null Hypothesis')
```

```
We Reject the Null Hypothesis
```

Season has effect on the number of cycles rented

In [ ]:
```
# Check test Assumptions on Weather
```

In [240]:
```
# Shapiro_Wilkin test for all weathers

weathers=df['season'].unique()
w1=df[df['weather']==weathers[0]]['count']
w2=df[df['weather']==weathers[1]]['count']
w3=df[df['weather']==weathers[2]]['count']
w4=df[df['weather']==weathers[3]]['count']
```

In [194]:
```
shapiro(w1)
```

Out[194]:
```
ShapiroResult(statistic=0.8909230828285217, pvalue=0.0)
```

In [195]:
```
shapiro(w2)
```

Out[195]:
```
ShapiroResult(statistic=0.8767687082290649, pvalue=9.781063280987223e-43)
```

In [196]:
```
shapiro(w3)
```

Out[196]:
```
ShapiroResult(statistic=0.7674332857131958, pvalue=3.876090133422781e-33)
```

In [ ]:
```
# p_values are less than 0.05 hence we reject the null hypothesis -
```

In [233]:
```
datab = [df[df['weather'] == weather]['count'] for weather in weath
```

In [238]:
```
lev, p_val = levene(*datab)
print("Levene's Test - Homogeneity of Variance")
print("p-value:", p_val)
```

```
Levene's Test - Homogeneity of Variance
p-value: 3.504937946833238e-35
```

In [ ]:
```
# p_value is less than 0.05 so The variances of the groups being co
```

In [ ]:
```
# Test to check if Weather has effect on number of cycles rented

# Ho: Weather has no effect on the numebr of cycles rented
# Ha: Weather has effect on the number of cycles rented

# We can perform ANOVA test here as there are two or more groups
```

In [134]:
```
weathers=df['season'].unique()
w1=df[df['weather']==weathers[0]]['count']
w2=df[df['weather']==weathers[1]]['count']
w3=df[df['weather']==weathers[2]]['count']
w4=df[df['weather']==weathers[3]]['count']
```

In [135]:
```
f_stats,p_value=f_oneway(w1,w2,w3,w4)

print('ANOVA: Weather v/s Count')
print('f-statistic:', f_stats)
print('p-value:', p_value)
```

```
ANOVA: Weather v/s Count
f-statistic: 65.53024112793271
p-value: 5.482069475935669e-42
```

In [138]:
```
# Set a significance level
alpha=0.05

if p_value>alpha:
    print('We Fail to Reject the Null Hypothesis')
else:
    print('We Reject the Null Hypothesis')
```

```
We Reject the Null Hypothesis
```

In [140]:
```
#Lets also validate the test with Kruskal's test as assumptions of

f_stats,p_value=kruskal(w1,w2,w3,w4)

# Set a significance level
alpha=0.05

if p_value>alpha:
    print('We Fail to Reject the Null Hypothesis')
else:
    print('We Reject the Null Hypothesis')
```

```
We Reject the Null Hypothesis
```

Weather has effect on the number of cycles rented

ChiSquare Test

In [143]:
```
# importing essential library
from scipy.stats import chi2_contingency
```

In [ ]:
```
# Test to check if Weather is dependent on season

# Ho: Weather and Season are independent variable and has no associ
# Ha: Weather and Season are dependent variable and has some associ

# We can perform ChiSquare Test to check dependancy
```

In [146]:
```
contingency_table=pd.crosstab(df['season'],df['weather'])
```

In [147]:
```
contingency_table
```

Out[147]:

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

In [152]:
```
chi2, pvalue, _, _ = chi2_contingency(contingency_table)

# Print the results
print("Chi-square Test: Weather and Season")
print("Chi-square:", chi2)
print("p-value:", p_value)
```

```
Chi-square Test: Weather and Season
Chi-square: 49.158655596893624
p-value: 1.549925073686492e-07
```

In [153]:
```
# Set a significance level
alpha=0.05

if pvalue>alpha:
    print('We Fail to Reject the Null Hypothesis')
else:
    print('We Reject the Null Hypothesis')
```

```
We Reject the Null Hypothesis
```

Weather and Season are dependent variable and has some association

In [ ]:
```

```

From the datasets we can conclude several key points:

T-Test Working day has no effect on the number of cycles rented.

ANOVA Test:

a) Season has effect on the number of cycles rented.

b) Weather has effect on the number of cycles rented.

Chi2-Test: Weather and Season are dependent variable and has some association

In [ ]:

In [ ]: