In [2]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import ttest_ind, ttest_rel
from scipy.stats import expon
from scipy.stats import poisson
from scipy.stats import chisquare, chi2_contingency
from scipy.stats import stats
from scipy.stats import f_oneway
```

Solution 1

In [120…
```python
DF= pd.read_csv("https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/42
DF.head()
```

Out[120]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	reg
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	

In [38]:
```python
DF.shape
```

Out[38]:
```
(10886, 12)
```

In [7]:
```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 [8]:

```
DF.describe()
```

Out[8]:

	season	holiday	workingday	weather	temp	atemp	humidi
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000	10886.0000
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.8864
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.2450
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	0.0000
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	47.0000
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	62.0000
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	77.0000
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	100.0000

In [9]:

```
# Null values

DF.isnull().sum()
```

Out[9]:

```
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
```

In [40]:

```
# Duplicate values
DF.duplicated()
```

Out[40]:
```
0        False
1        False
2        False
3        False
4        False
         ...
10881    False
10882    False
10883    False
10884    False
10885    False
Length: 10886, dtype: bool
```

In [49]:
```
DF.dtypes
```

Out[49]:
```
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 [50]:
```
# Categorical and numerical column saggrigation

cat_cols = list(DF.dtypes[DF.dtypes == "object"].index)
num_cols = list(DF.dtypes[DF.dtypes != "object"].index)
```

In [51]:
```
# Calling Categorical column
DF[cat_cols]
```

Out[51]:

	datetime
0	2011-01-01 00:00:00
1	2011-01-01 01:00:00
2	2011-01-01 02:00:00
3	2011-01-01 03:00:00
4	2011-01-01 04:00:00
...	...
10881	2012-12-19 19:00:00
10882	2012-12-19 20:00:00
10883	2012-12-19 21:00:00
10884	2012-12-19 22:00:00
10885	2012-12-19 23:00:00

10886 rows × 1 columns

In [52]:

```
# Calling Numerical Numerical
DF[num_cols]
```

Out[52]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered
0	1	0	0	1	9.84	14.395	81	0.0000	3	13
1	1	0	0	1	9.02	13.635	80	0.0000	8	32
2	1	0	0	1	9.02	13.635	80	0.0000	5	27
3	1	0	0	1	9.84	14.395	75	0.0000	3	10
4	1	0	0	1	9.84	14.395	75	0.0000	0	1
...	...	...	...	...	...	...	...	...	...	..
10881	4	0	1	1	15.58	19.695	50	26.0027	7	329
10882	4	0	1	1	14.76	17.425	57	15.0013	10	231
10883	4	0	1	1	13.94	15.910	61	15.0013	4	164
10884	4	0	1	1	13.94	17.425	61	6.0032	12	117
10885	4	0	1	1	13.12	16.665	66	8.9981	4	84

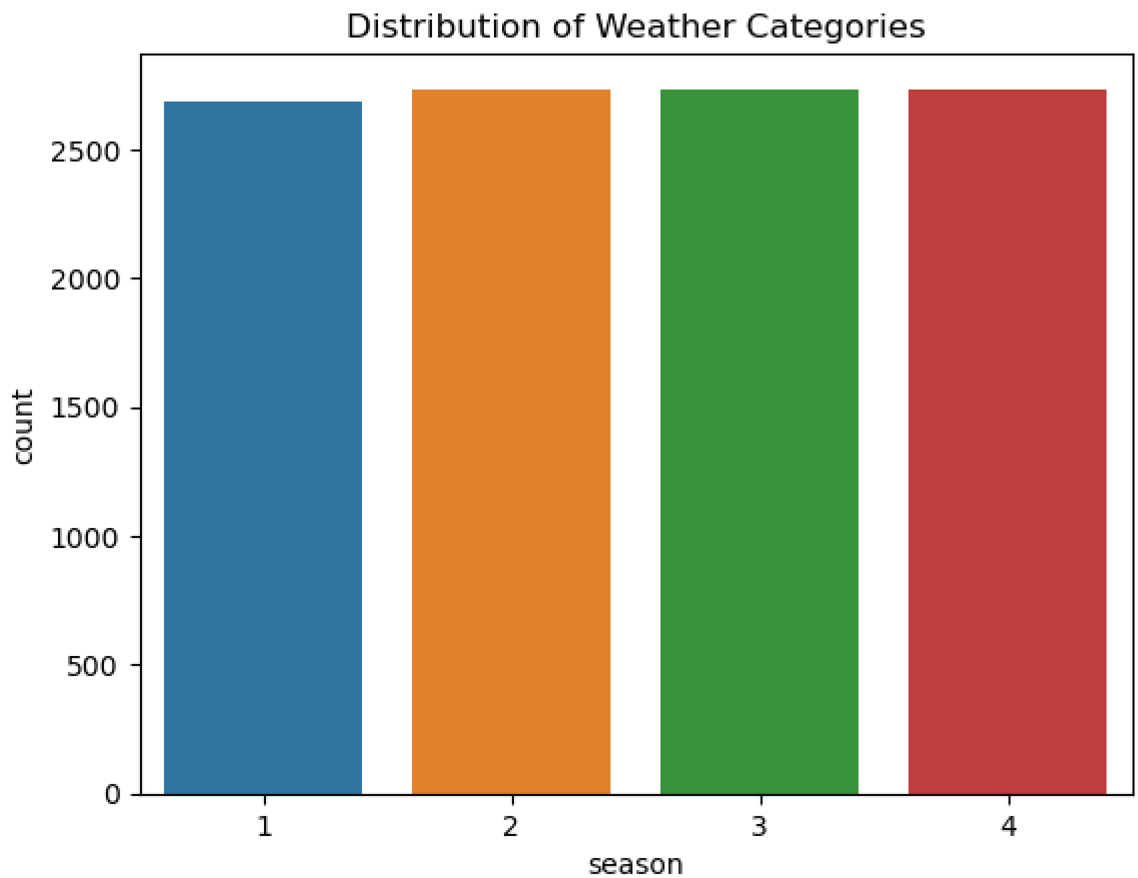
10886 rows × 11 columns

In [41]:

```
# (1: spring, 2: summer, 3: fall, 4: winter)

DF["season"].value_counts()
```

Out[41]:
```
4    2734
2    2733
3    2733
1    2686
Name: season, dtype: int64
```

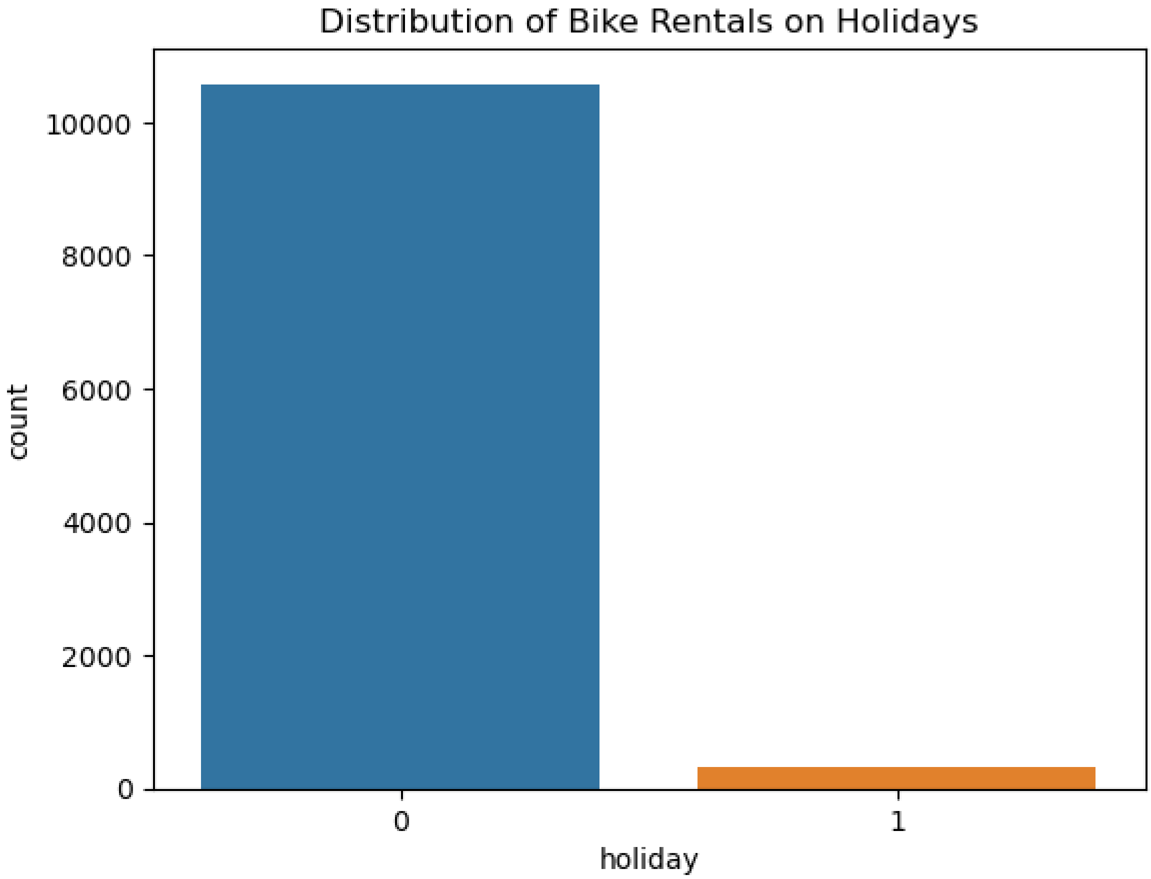
In [40]:
```
# Countplot showing weather categoroies
sns.countplot(x='season', data=DF)
plt.title("Distribution of Weather Categories")
plt.show()
```



In [42]:
```
# if day is neither weekend nor holiday is 1, otherwise is 0

DF["holiday"].value_counts()
```

Out[42]:
```
0    10575
1      311
Name: holiday, dtype: int64
```

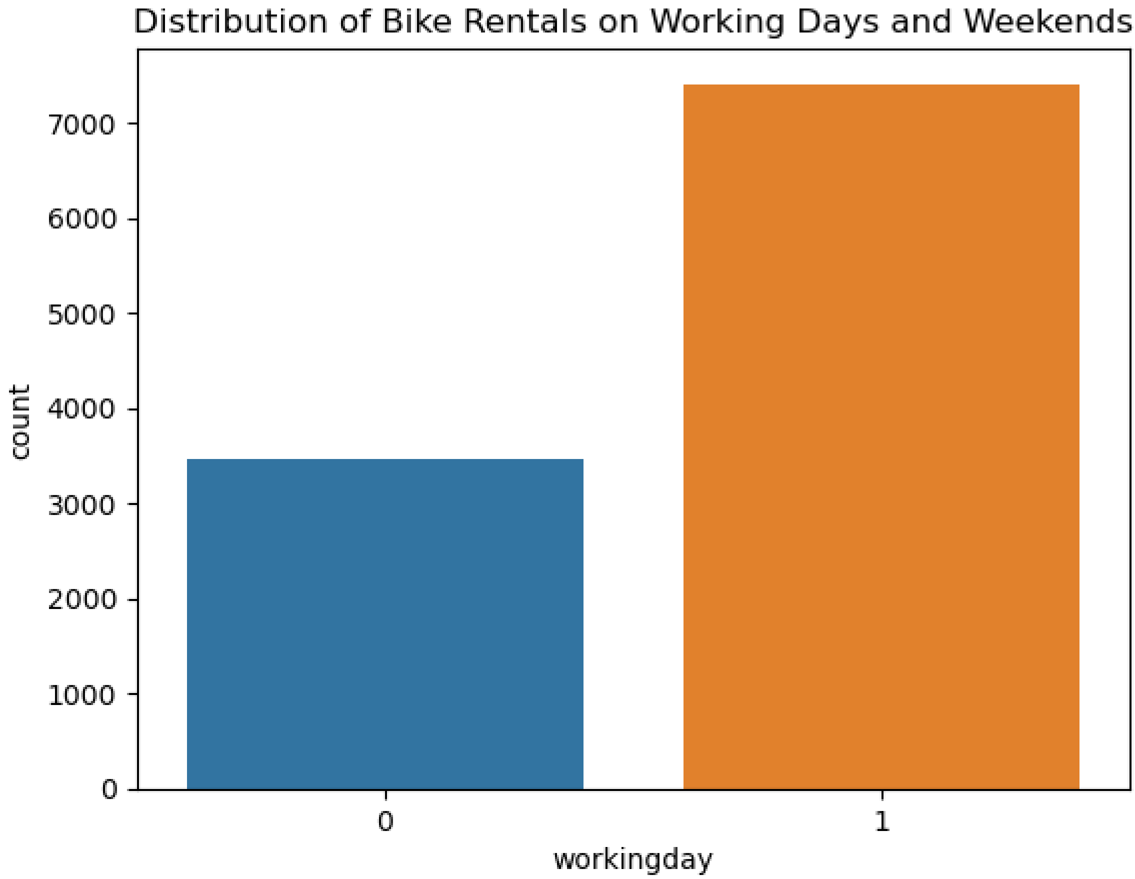
In [43]:
```
# count of holiday
# 0 represents "not a holiday," and 1 represents "holiday."
sns.countplot(data=DF, x= "holiday")
plt.title("Distribution of Bike Rentals on Holidays")
plt.show()
```



In [43]:
```
DF["workingday"].value_counts()
```

Out[43]:
```
1    7412
0    3474
Name: workingday, dtype: int64
```

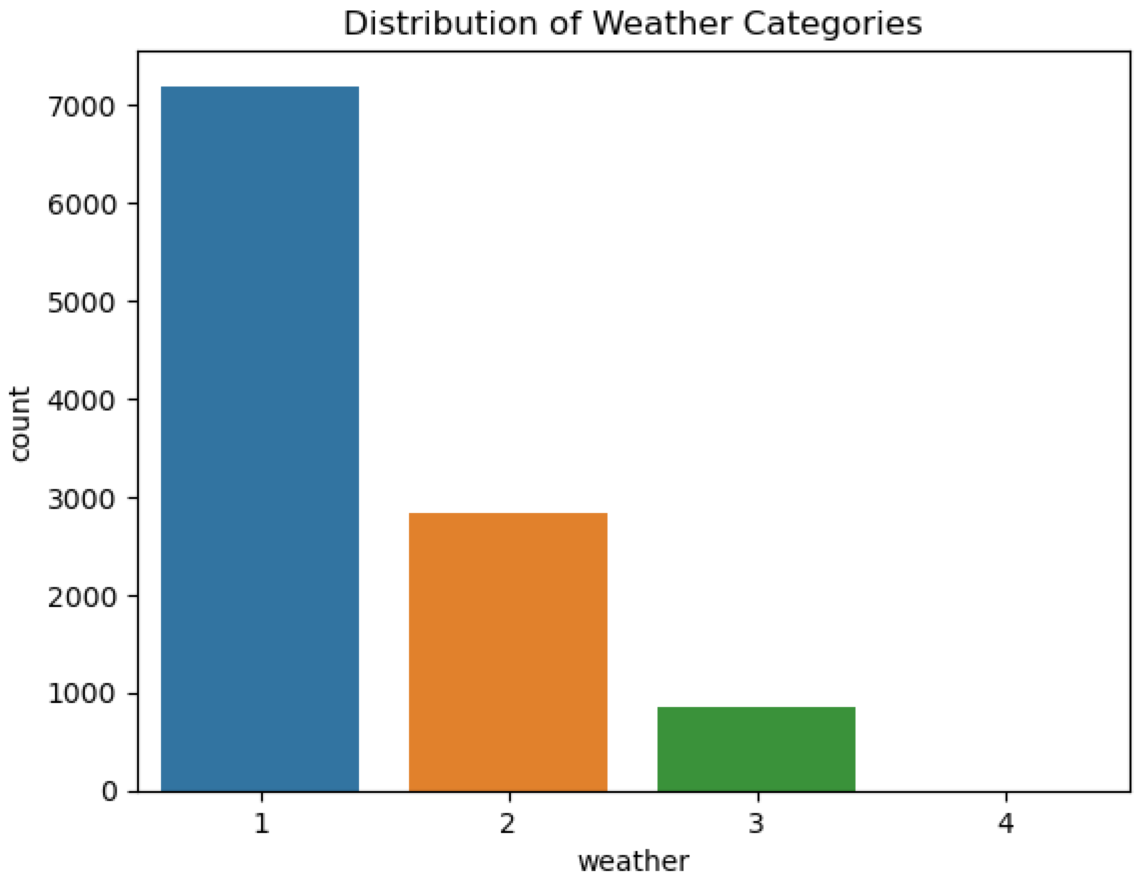
In [41]:
```
#1 represents "working day," and 0 represents "non-working day"
sns.countplot(data=DF, x="workingday")
plt.title("Distribution of Bike Rentals on Working Days and Weekends")
plt.show()
```



In [44]:
```
# 1:clear, 2: Mist, 3:  Light Snow, 4:Heavy Rain + Ice Pallets
DF["weather"].value_counts()
```

Out[44]:
```
1    7192
2    2834
3     859
4       1
Name: weather, dtype: int64
```

In [23]:
```
# Create a countplot for the weather categories
sns.countplot(x='weather', data=DF)
plt.title("Distribution of Weather Categories")
plt.show()
```



In [47]:

```
# count of casual users like not registerd users
DF["casual"].value_counts()
```

Out[47]:

```
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 [63]:

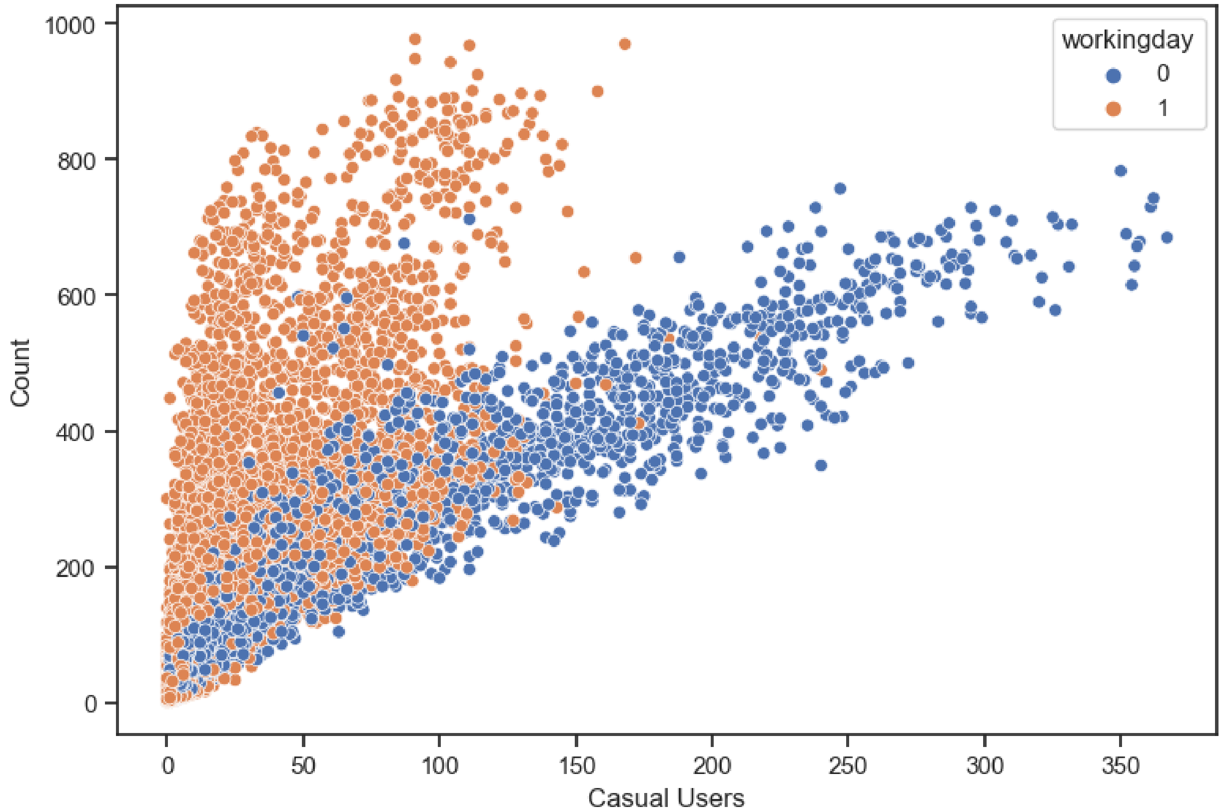
```
DF["registered"].value_counts()
```

Out[63]:

```
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
```

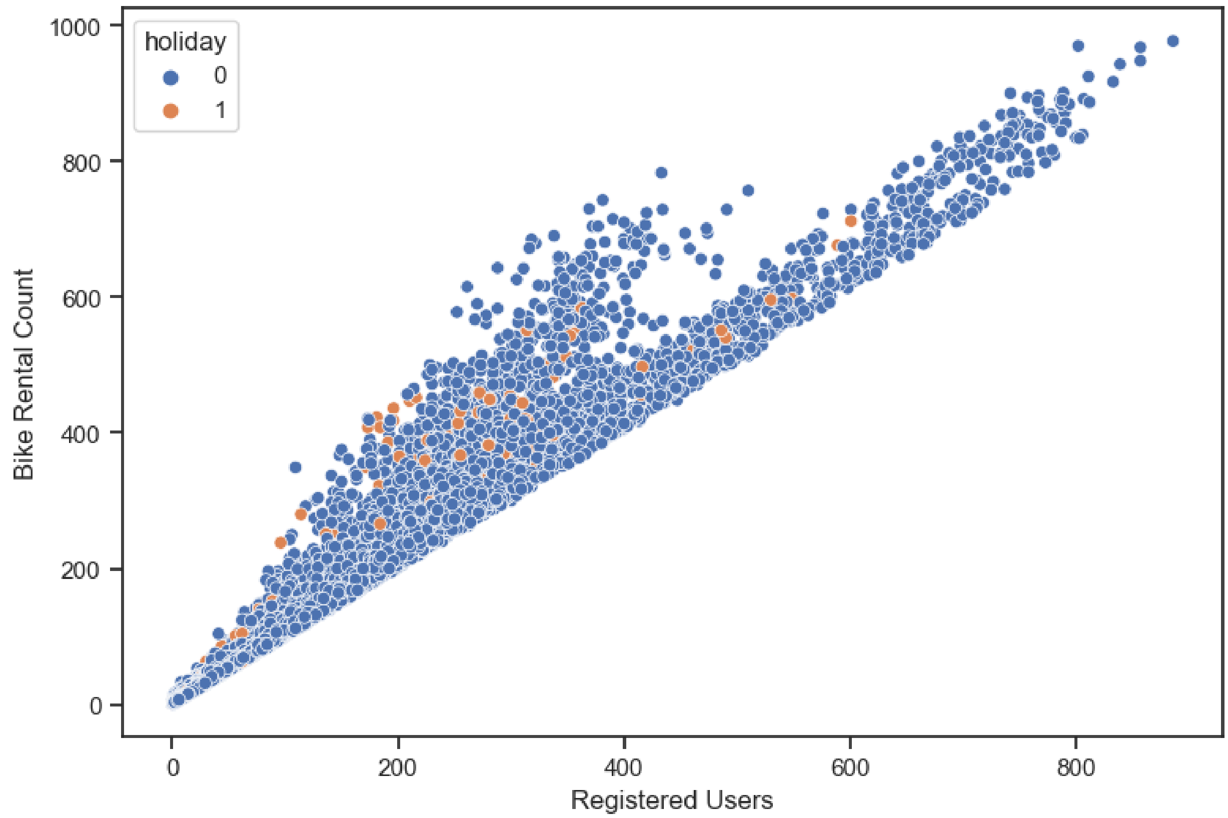
In [118…

```
plt.figure(figsize=(9, 6))
sns.scatterplot(data=DF, x= "casual", y= "count", hue="workingday" )
plt.xlabel("Casual Users")
plt.ylabel("Count")
plt.show()
```



In [114…

```
plt.figure(figsize=(9, 6))
sns.scatterplot(data=DF, x="registered", y="count", hue="holiday")
plt.xlabel("Registered Users")
plt.ylabel("Bike Rental Count")
plt.show()
```

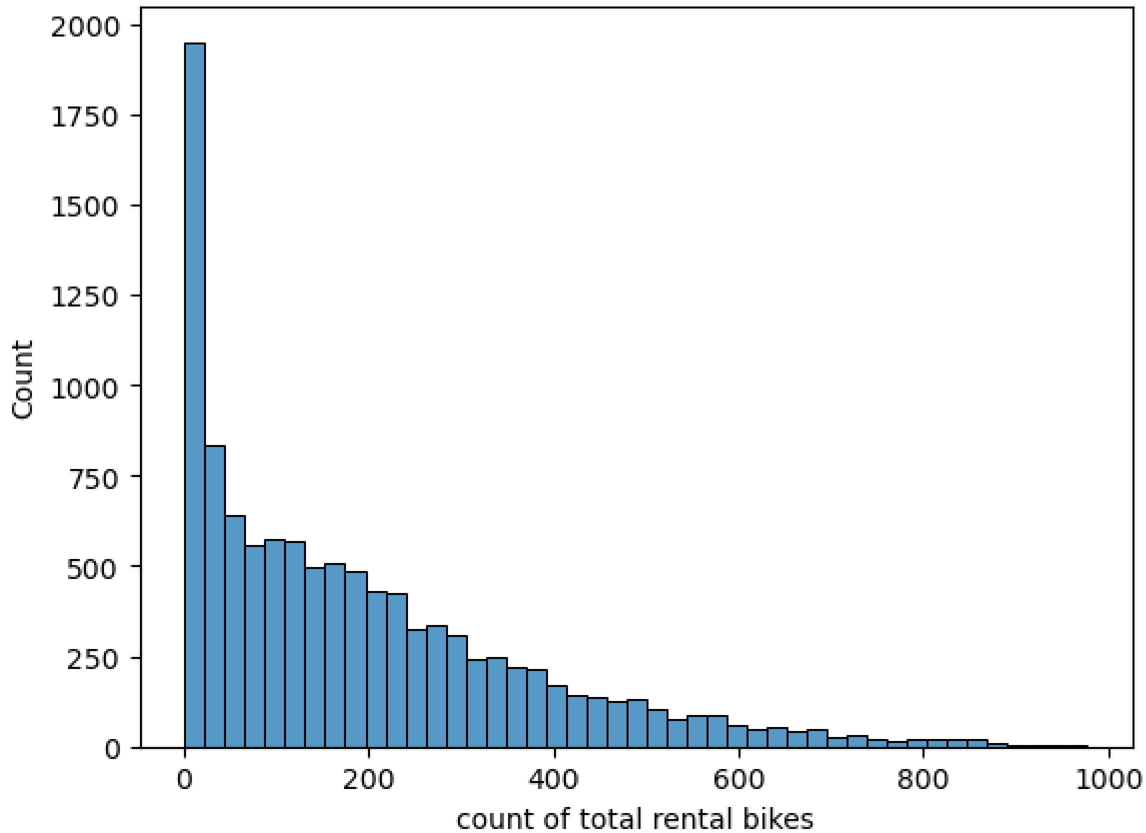


In [ ]:

In [49]:
```
#count of total rental bikes including both casual and registered
DF["count"].value_counts()
```

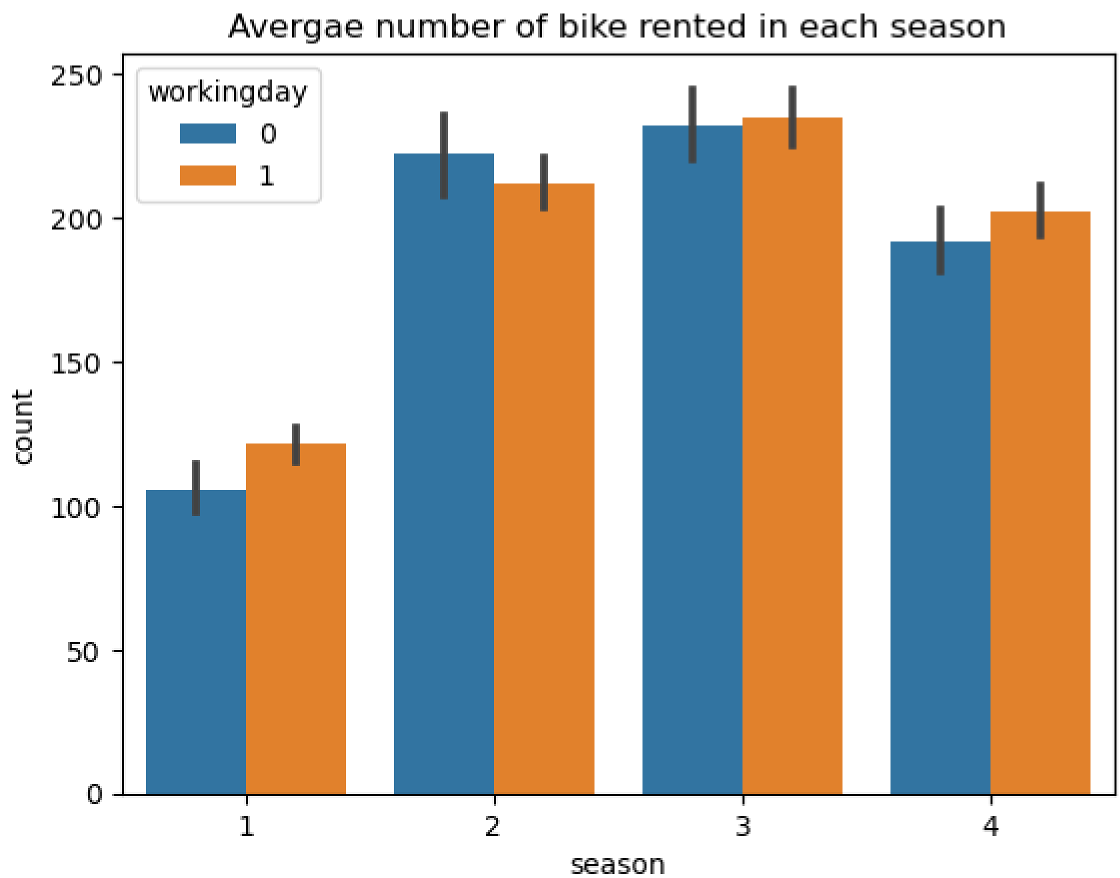
Out[49]:
```
5      169
4      149
3      144
6      135
2      132
      ...
801      1
629      1
825      1
589      1
636      1
Name: count, Length: 822, dtype: int64
```

In [26]:
```
sns.histplot(data=DF, x="count")
plt.xlabel("count of total rental bikes")
plt.show()
```

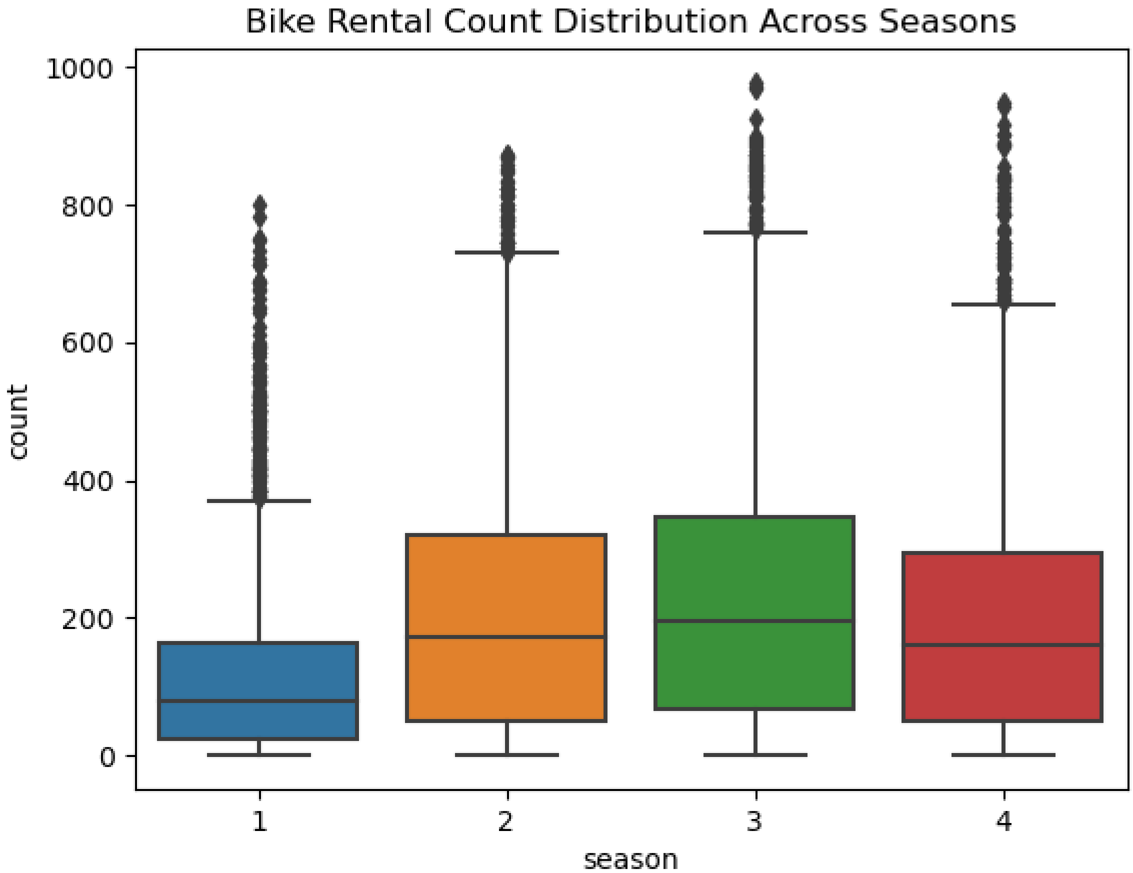


conclusion: People prefer to rent bikes more during clear weather.

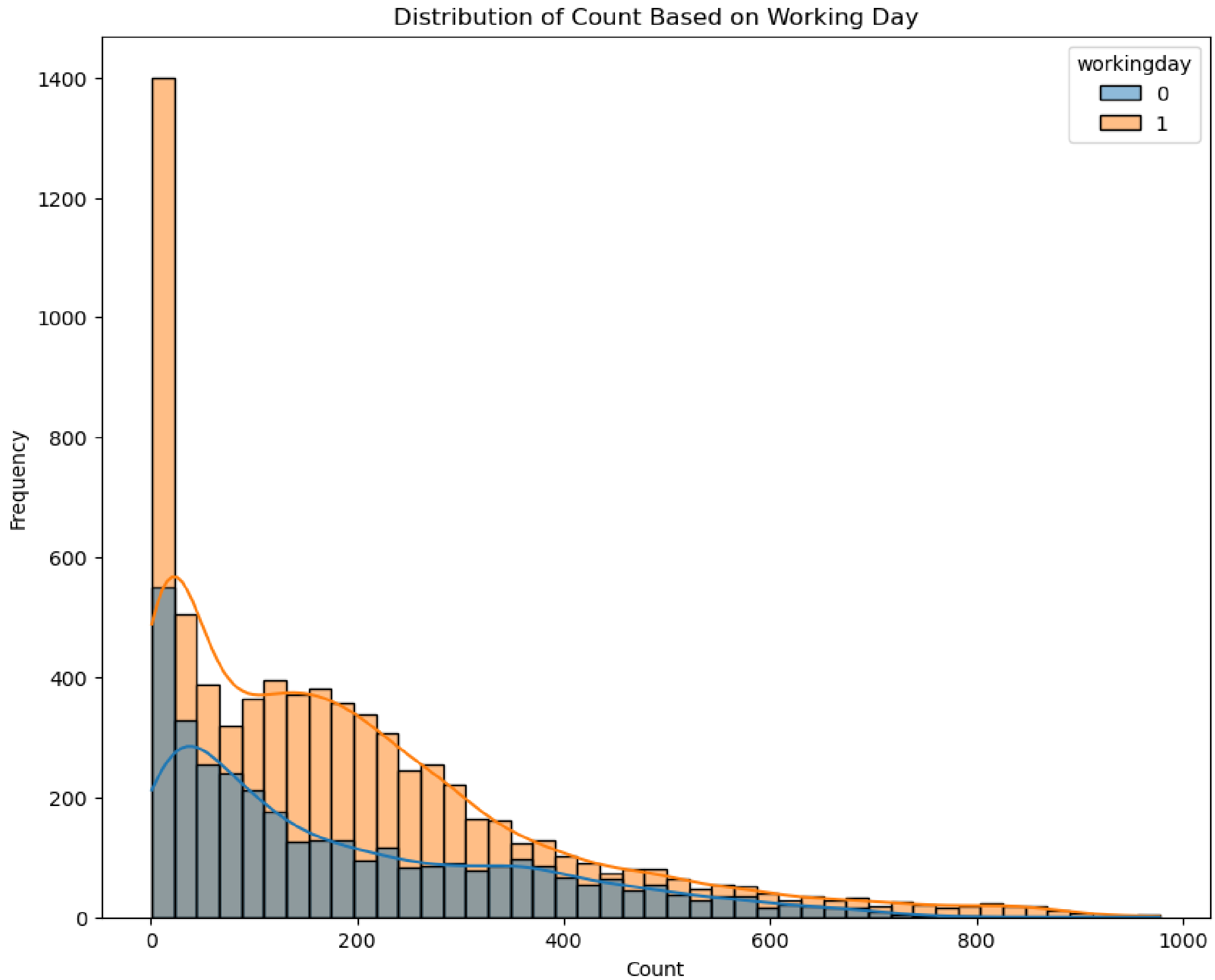
In [24]:
```
# barplot shows relationship between a categorical variable (season in this case) and
# Useful for comparing the means or central tendencies of different categories.
# season wise count of rental bike that is
# (1: spring(march-april), 2: summer(may-June), 3: fall(sep and oct), 4: winter(dec to
sns.barplot(data=DF, x= "season", y="count", hue='workingday')
plt.title("Avergae number of bike rented in each season")
plt.show()
```



In [22]:
```
# Boxplot to visualize the distribution of 'count' across different 'season' values.
# The boxplot displays the median, quartiles, and potential outliers in the data.`
sns.boxplot(data= DF, x="season", y="count")
plt.title("Bike Rental Count Distribution Across Seasons")
plt.show()
```

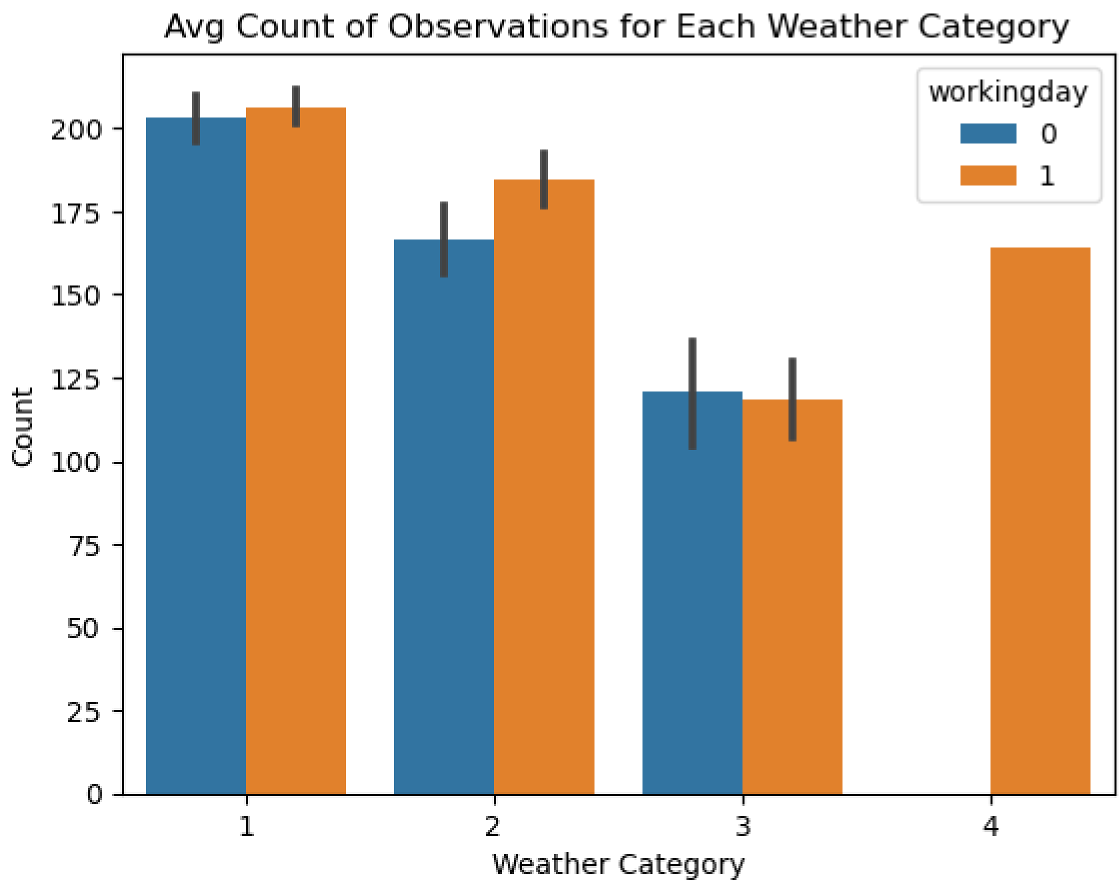


In [21]:
```
# histogram typically depict the distribution of a single variable.
# Additionally, hue is generally used to color the data points based on another catego
plt.figure(figsize=(10, 8))
sns.histplot(data=DF, x="count", hue="workingday",kde=True)
plt.xlabel("Count")
plt.ylabel("Frequency")
plt.title("Distribution of Count Based on Working Day")
plt.show()
```



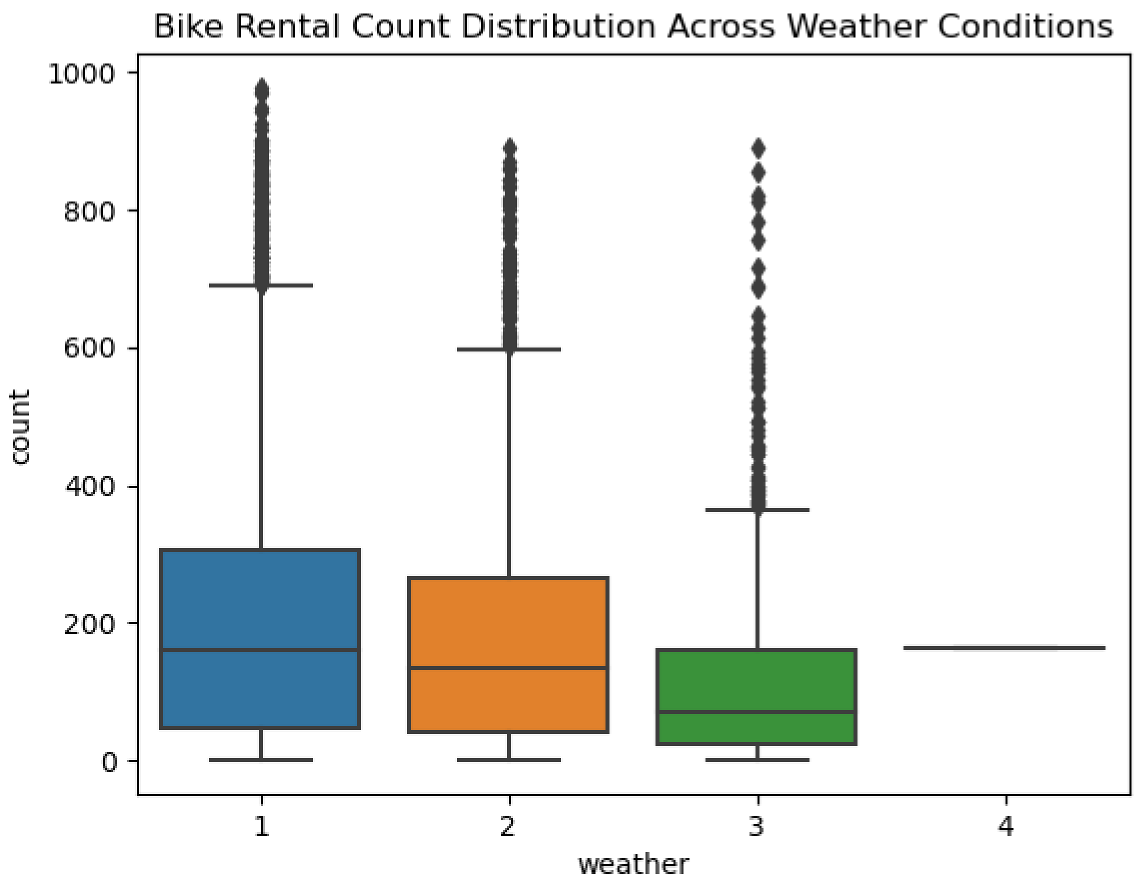
In [25]:

```
# weather wise count of rental bike 1:clear, 2: Mist, 3:  Light Snow, 4:Heavy Rain + I
# hourly count of rented bikes
#barplot by default calculate average
sns.barplot(data=DF, x= "weather", y="count", hue='workingday', estimator="mean")
# Set the title
plt.title("Avg Count of Observations for Each Weather Category")
plt.xlabel("Weather Category")
plt.ylabel("Count")
plt.show()
```



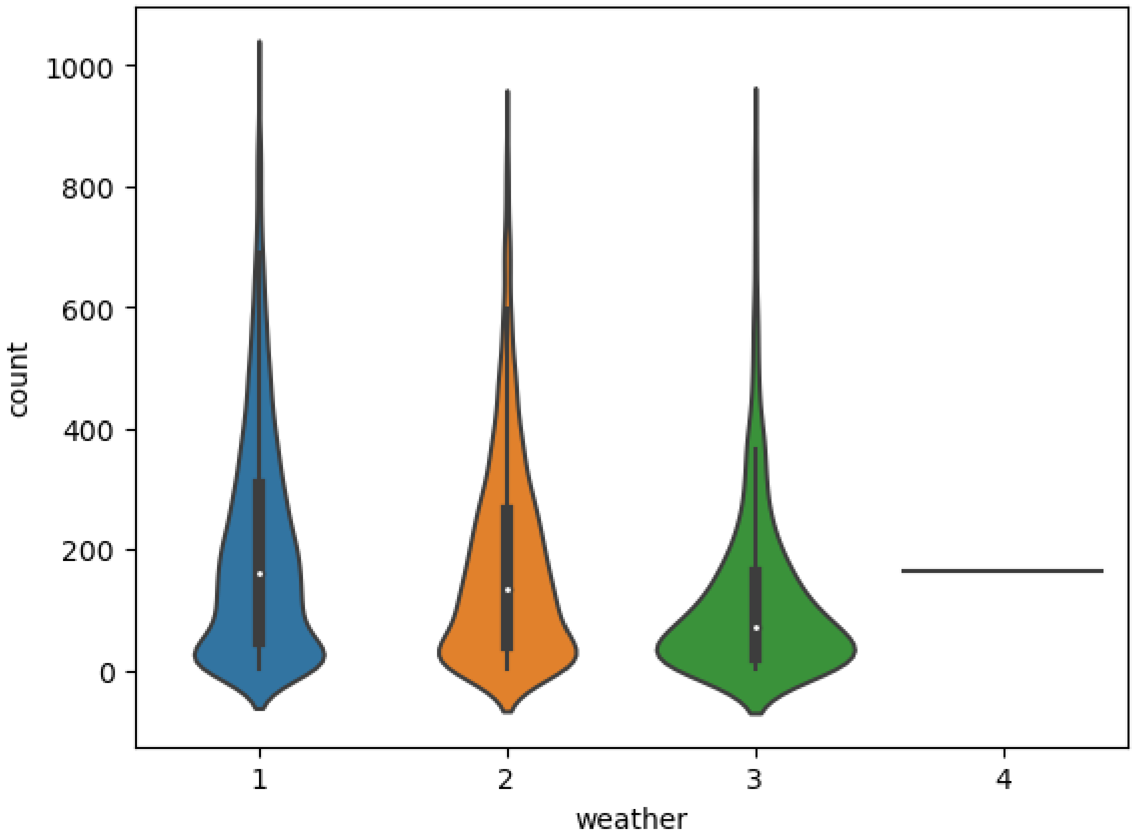
In [27]:

```
# Create a boxplot for rental counts on different weather conditions
sns.boxplot(x='weather', y='count', data=DF)
plt.title("Bike Rental Count Distribution Across Weather Conditions")
plt.show()
```



In [ ]:

In [13]:
```
# Violinplot to visualize the distribution of 'count' across different 'weather' value
sns.violinplot(data=DF, x='weather', y='count')
plt.show()
```

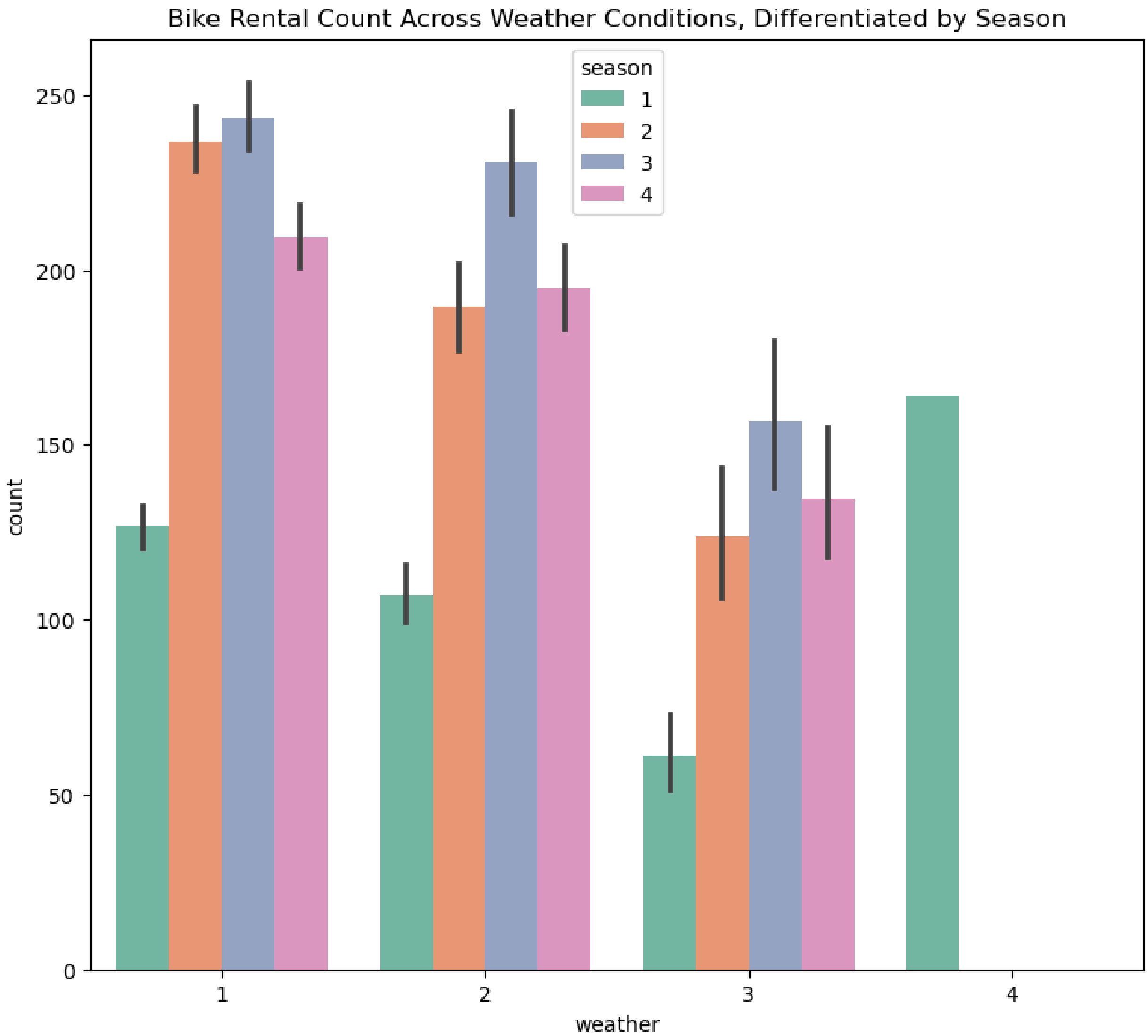


In [ ]:
```
Anova Test For seasons and count
Ho: season does not impact on no of bike rented
Ha: season have impact on number of bike rented
```

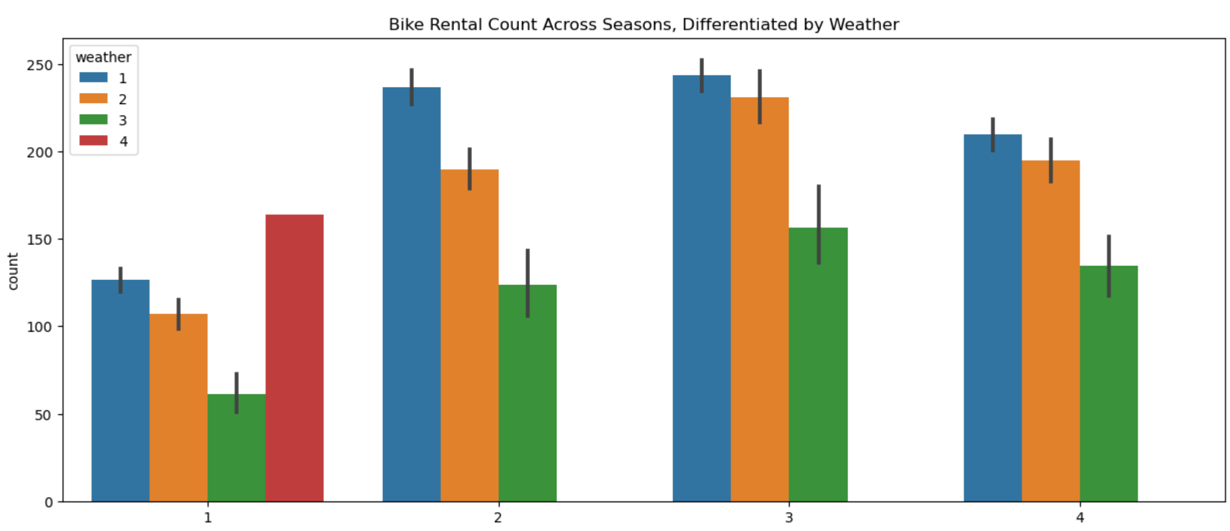
In [12]:
```
#Holiday and Working day (T-Test)
Ho: Avg no bike rented on hoilday is same as avg no of bike rented on Weekday
Ha: Avg is not same (Muh < Muw)
```

In [ ]:
```
Anova Test For seasons and count
Ho: season does not impact on no of bike rented
Ha: season have impact on number of bike rented
```

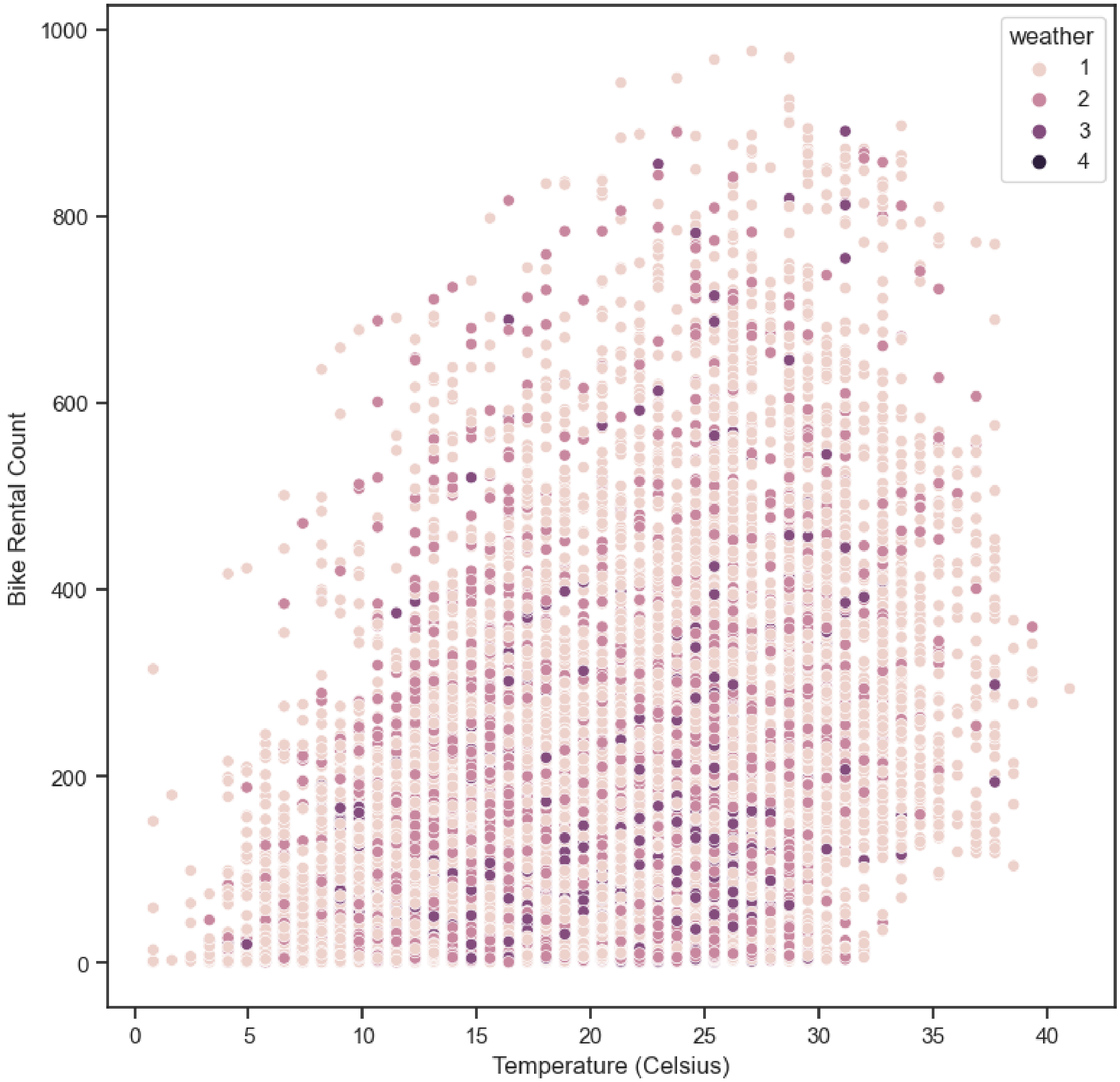
In [51]:
```
plt.figure(figsize=(9,8))
sns.barplot(data=DF, x="weather", y="count", hue="season", palette="Set2")

plt.title("Bike Rental Count Across Weather Conditions, Differentiated by Season")
plt.show()
```



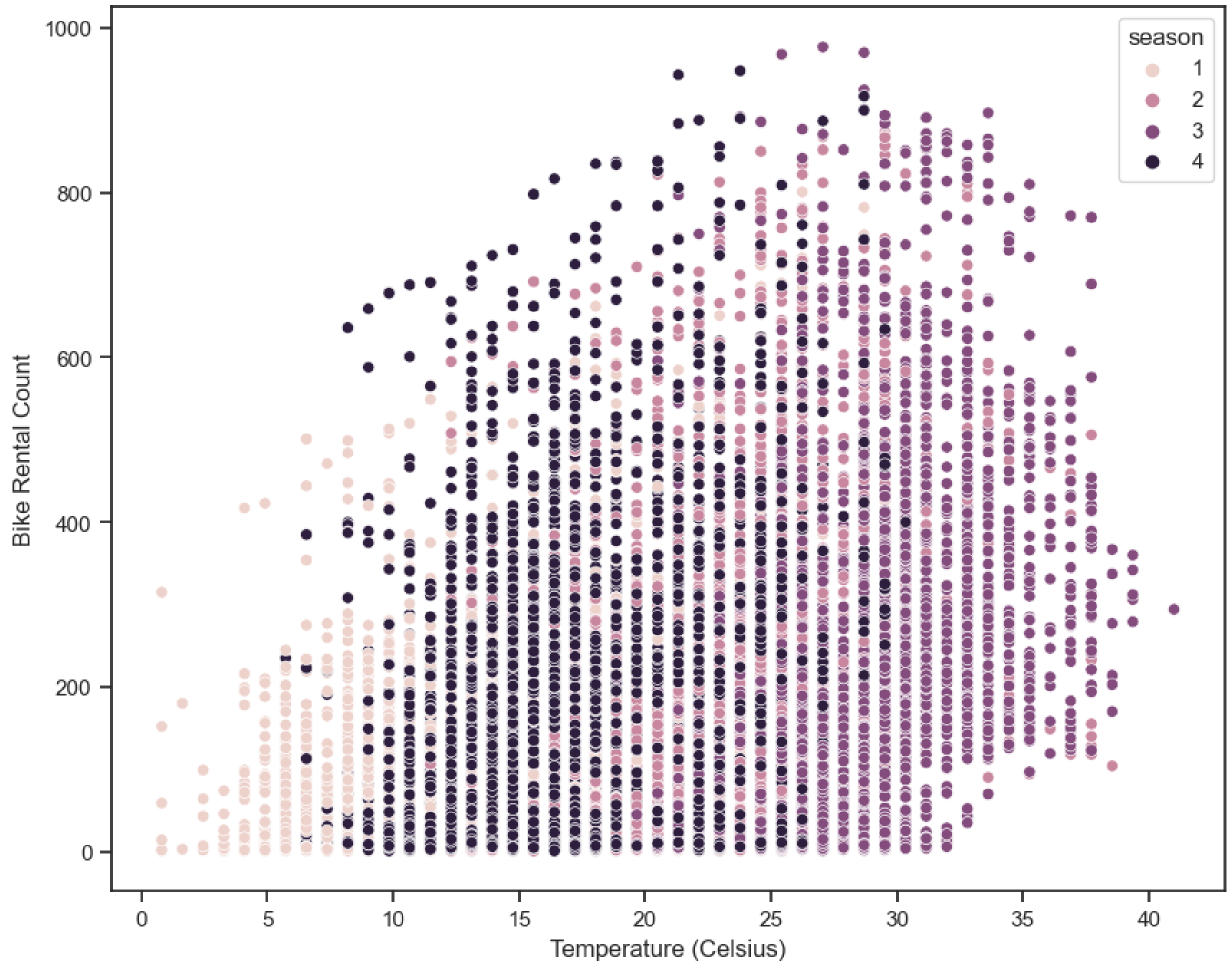
In [31]:
```
plt.figure(figsize=(15, 6))
sns.barplot(data=DF, x="season", y="count", hue="weather")
plt.title("Bike Rental Count Across Seasons, Differentiated by Weather")
plt.show()
```



In [54]:
```
# This histogram shows that most preperred weather is 1 which is clear and temprature
plt.figure(figsize=(9, 9))
sns.scatterplot(data=DF, x="temp", y="count", hue="weather")
plt.xlabel("Temperature (Celsius)")
plt.ylabel("Bike Rental Count")
plt.show()
```



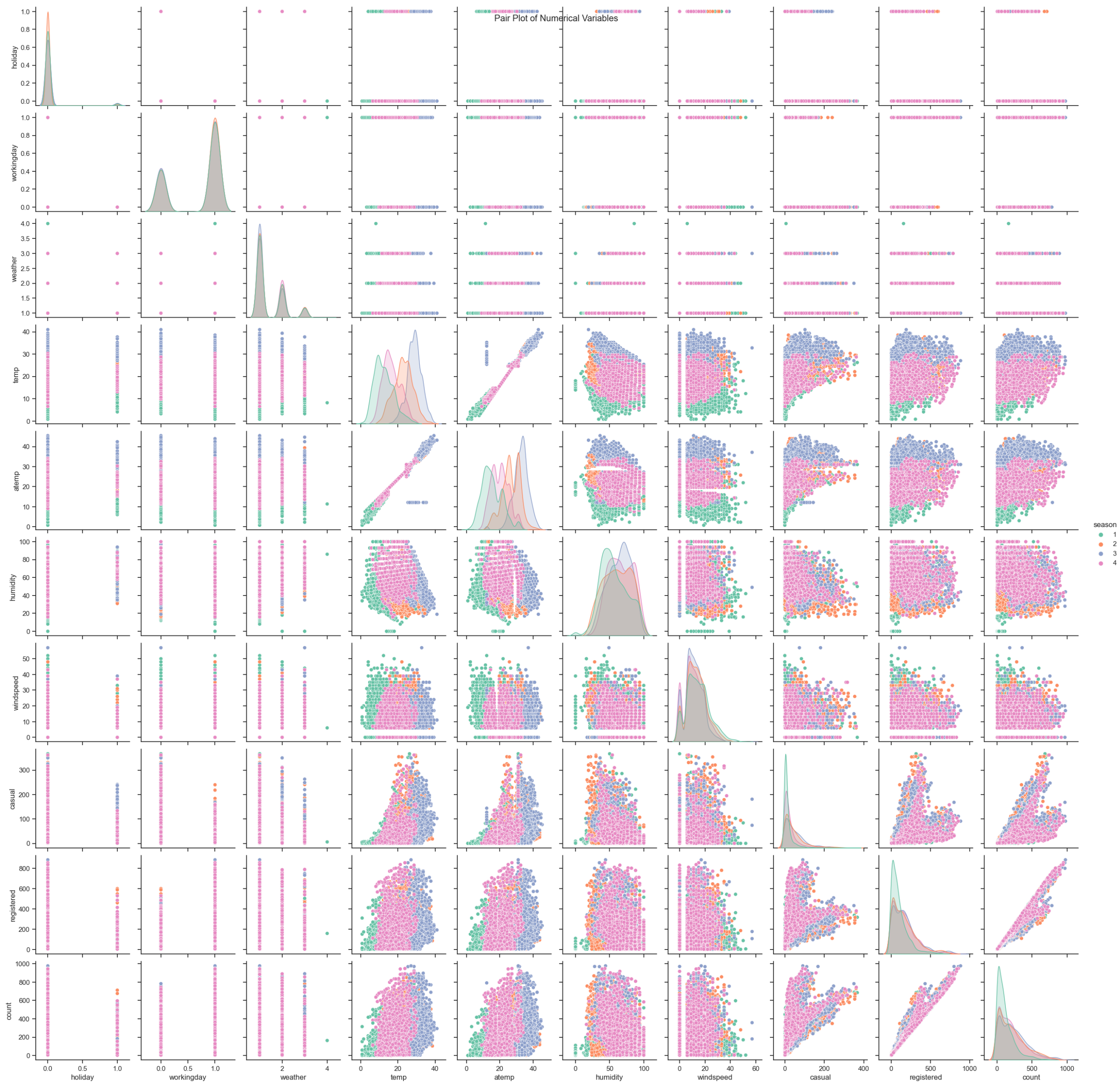
In [53]:
```
# this histogram shows that count is high when temp 12 to 33 cel and season is 3, and
plt.figure(figsize=(10, 8))
sns.set(style="ticks")
sns.scatterplot(data=DF, x="temp", y="count", hue="season")
plt.xlabel("Temperature (Celsius)")
plt.ylabel("Bike Rental Count")
plt.show()
```



In [54]:

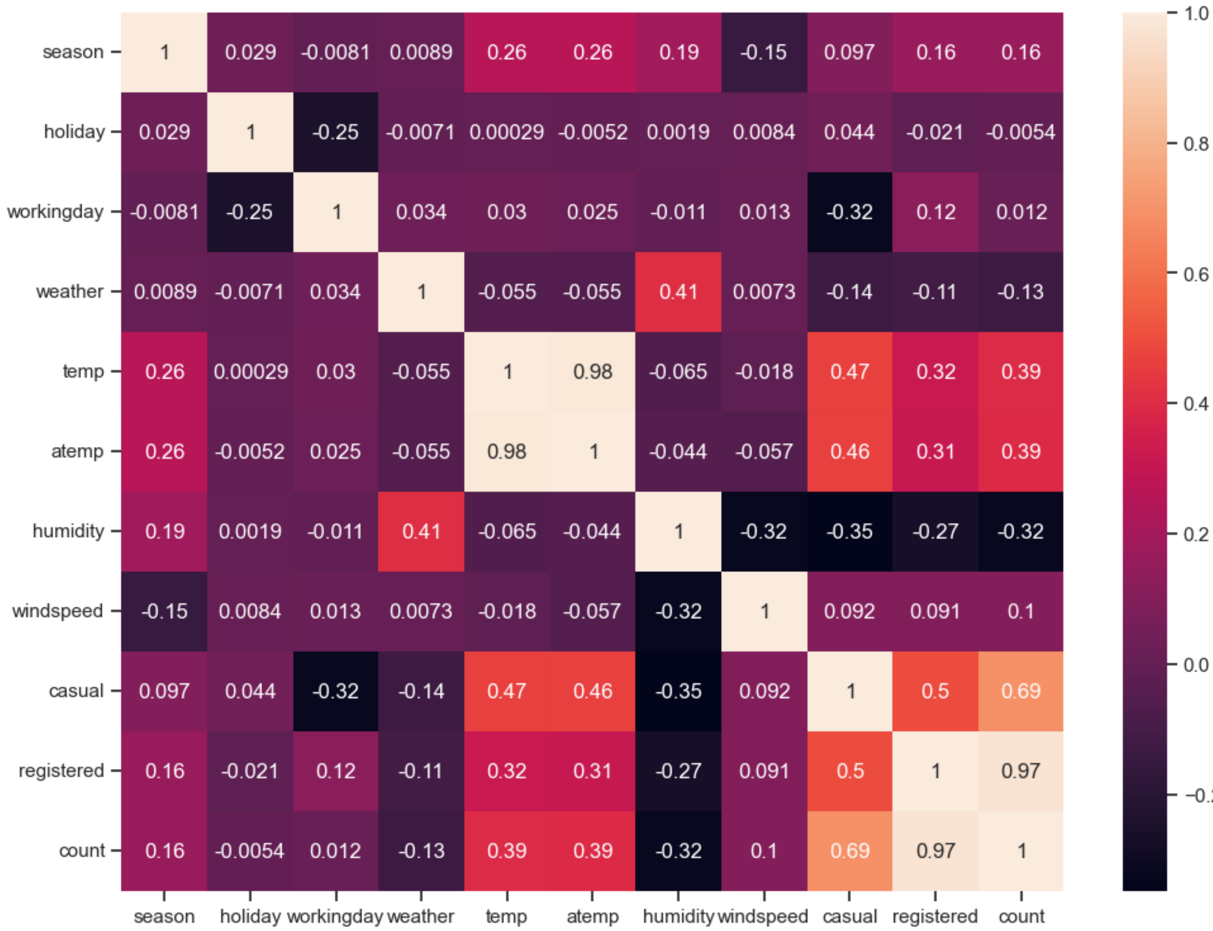
```
# Pairplot to visualize relationships between numerical variables
plt.figure(figsize=(12, 6))
sns.set(style="ticks")
sns.pairplot(data= DF[num_cols], diag_kind="kde", hue="season", palette="Set2")

plt.suptitle("Pair Plot of Numerical Variables")
plt.show()
```

<Figure size 1200x600 with 0 Axes>



In [62]:
```python
# Heatmap to visualize correlation between numerical variables

plt.figure(figsize=(12, 9))
sns.heatmap(DF.corr(),annot=True)
plt.show()
```

```
C:\Users\harsh\AppData\Local\Temp\ipykernel_8556\3708558687.py:2: FutureWarning: The
default value of numeric_only in DataFrame.corr is deprecated. In a future version, i
t will default to False. Select only valid columns or specify the value of numeric_on
ly to silence this warning.
  sns.heatmap(DF.corr(),annot=True)
```



In [ ]:

```
# Boxplot to visualize the distribution of 'count' across different 'season' values
sns.boxplot(data=DF,x='season', y='count')
plt.show()
```

Solution-2

T-Test

2- Sample T-Test to check if Working Day has an effect on the number of electric cycles rented

Null Hypothesis (H0): There is no significant difference in the mean number of electric cycles rented between working days and non-working days.

Alternative Hypothesis (H1): There is a significant difference in the mean number of electric cycles rented between working days non-working days.

In [63]:

```
workingday_count= DF[DF["workingday"] == 1]["count"]
Non_workingday_count= DF[DF["workingday"] == 0]["count"]
```

In [67]:

```
t_stat, p_value= ttest_ind(workingday_count, Non_workingday_count)
t_stat, p_value
```

Out[67]:
```
(1.2096277376026694, 0.22644804226361348)
```

In [69]:
```
alpha=0.05
```

In [70]:
```
if(p_value<alpha):
    print("Reject Ho")
else:
    print("Fail to reject Ho means Ho is passed")
```

```
Fail to reject Ho means Ho is passed
```

Conclusion: There is no significant difference in the mean number of electric cycles rented between working days and non-working days.

Solution 2(b)

ANNOVA to check if No. of cycles rented is similar or different in different 1. weather 2. season (10 points)

In [ ]:
```
Ho: weather and season does not impact the Number of cycle rented
Ha:  weather and season have significance impact the Number of cycle rented
```

In [83]:
```
#weather_group = [DF["count"]DF["weather"]==i] for i in DF["weather"].unique()
#season_group = [DF["count"]DF["season"]==i] for i in DF["season"].unique()
```

In [80]:
```
#season_group= DF.groupby("season")["count"].apply(list)
#weather_group = DF.groupby("weather")["count"].apply(list)
```

In [85]:
```
season_group
```

Out[85]:
```
season
1    [16, 40, 32, 13, 1, 1, 2, 3, 8, 14, 36, 56, 84...
2    [6, 4, 7, 4, 3, 12, 28, 95, 206, 173, 75, 89, ...
3    [68, 31, 13, 11, 6, 30, 108, 243, 492, 260, 17...
4    [130, 58, 67, 25, 8, 5, 19, 36, 67, 129, 121, ...
Name: count, dtype: object
```

In [86]:
```
weather_group
```

Out[86]:
```
weather
1    [16, 40, 32, 13, 1, 2, 3, 8, 14, 36, 56, 84, 6...
2    [1, 94, 106, 110, 93, 67, 36, 34, 28, 39, 17, ...
3    [35, 37, 2, 8, 59, 74, 76, 5, 7, 1, 15, 20, 95...
4                                                [164]
Name: count, dtype: object
```

In [ ]:
```
weather_groups = [DF['count'][(DF['season'] == season) & (DF['weather'] == weather)].t
                  for season in DF['season'].unique() for weather in DF['weather'].uni
```

In [95]:
```
# Filter out empty groups
weather_groups = [group for group in weather_groups if len(group) > 0]
```

In [97]:
```
# Perform two-way ANOVA if there are valid groups
if len(weather_groups) > 1:
    test_stat, p_value = f_oneway(*weather_groups)
```

In [98]:
```
# Print the result
print("Test Statistic:", test_stat)
print("P-Value:", p_value)
```

```
Test Statistic: 78.52062887533677
P-Value: 7.760264814517954e-186
```

In [100…
```
# Define significance level
alpha=.05
```

In [101…
```
# Check if the result is significant at a 0.05 significance level

if(p_value<alpha):
    print("Reject Ho")
else:
    print("Fail to reject Ho means Ho is passed")
```

```
Reject Ho
```

Conclusion: weather and season have significance impact the Number of cycle rented

Solution 2(c)

In [ ]:
```
Q. Chi-square test to check if Weather is dependent on the season
```

In [103…
```
#Ho: weather is not dependent on season
#Ha: weather is dependent on season
```

In [106…
```
# create a contingency table
contingency_table= pd.crosstab(DF["season"], DF["weather"])
contingency_table
```

Out[106]:

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 [107…
```
# perform chi_square contingency test
test_stat, p_value, dof, exp_freq = chi2_contingency(contingency_table)
test_stat, p_value, dof, exp_freq
```

Out[107]:
```
(49.158655596893624,
 1.549925073686492e-07,
 9,
 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 [ ]:
```
# Describe significance level
alpha= 0.05
```

In [108…
```
# Check if the result is significant at a 0.05 significance level

if(p_value < alpha):
    print("Reject Ho")
else:
    print("Fail to Reject Ho")
```

```
Reject Ho
```

Conclusion of chi-sqaure Test: weather is dependent on season

Solution 3 (Recommendation by Analysis)

1. There is no significant difference in the mean number of electric cycles rented between working days and non-working days:
   ```
   So whatever compaign you are running to increase the sale it should be run no matter
   whether it is workingday or weekend.
   ```
2. weather and season have significance impact the Number of cycle rented:
   ```
   Need to practice all the marketing straegies
   that can help to improve the bike renting during bad weather in any season.
   ```
3. Weather is dependent on season.
4. People prefer to rent bikes more during clear weather:
   ```
   So try to capture more number of customer during clear weather
   ```
   start with reasonable price and offers and referral codes and better customer support, easy booking and advertisement.
5. Season 1 (spring) which has lowest number of bike rent.
6. In Season 3 & 2, which is fall & summer, had the highest number of bikes is rented by users.

7. Tempratue is also a factor: user prefer to bike rent when temprature between 12 to 35 (celsius) in seaon 3 & 4.
8. Registered users book bike mostly when there is no holiday.

```
In [ ]:
```