NAME: NERELLA VENKATA RADHAKRISHNA

ID: 190031187

SKILL-4

1.Write about Seaborn Library in Python

A.Seaborn is a library that uses Matplotlib underneath to plot graphs. It will be used to visualize random distributions.

It supports the following plots:

```
->Distribution Plots

->Matrix Plots

->Regression Plots

->Time Series Plots

->Categorical Plots
```

To Install [Python]: pip install seaborn

To Install [Jupiter]: !pip install seaborn

To Import: import seaborn as sns

In [2]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings("ignore")

pd.options.display.float_format = '{:.5f}'.format
pd.options.display.max_columns = None
pd.options.display.max_rows = None
np.random.seed(100)
```

In [3]:
```
day = pd.read_csv('day.csv')
hour = pd.read_csv('hour.csv')
day.head()
```

Out[3]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	winds
0	1	2011-01-01	1	0	1	0	6	0	2	0.34417	0.36362	0.80583	0.1
1	2	2011-01-02	1	0	1	0	0	0	2	0.36348	0.35374	0.69609	0.2
2	3	2011-01-03	1	0	1	0	1	1	1	0.19636	0.18940	0.43727	0.2
3	4	2011-01-04	1	0	1	0	2	1	1	0.20000	0.21212	0.59044	0.1
4	5	2011-01-05	1	0	1	0	3	1	1	0.22696	0.22927	0.43696	0.1

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

Out[4]:

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	wi
0	1	2011-01-01	1	0	1	0	0	6	0	1	0.24000	0.28790	0.81000	
1	2	2011-01-01	1	0	1	1	0	6	0	1	0.22000	0.27270	0.80000	
2	3	2011-01-01	1	0	1	2	0	6	0	1	0.22000	0.27270	0.80000	
3	4	2011-01-01	1	0	1	3	0	6	0	1	0.24000	0.28790	0.75000	
4	5	2011-01-01	1	0	1	4	0	6	0	1	0.24000	0.28790	0.75000	

In [5]:
```
day['temp'] = day['temp']*41
hour['temp'] = hour['temp']*41

day['atemp'] = day['atemp']*50
hour['atemp'] = hour['atemp']*50

day['hum'] = day['hum']*100
hour['hum'] = hour['hum']*100

day['windspeed'] = day['windspeed']*67
hour['windspeed'] = hour['windspeed']*67
```

In [6]:
```
hour.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17379 entries, 0 to 17378
Data columns (total 17 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   instant     17379 non-null  int64  
 1   dteday      17379 non-null  object 
 2   season      17379 non-null  int64  
 3   yr          17379 non-null  int64  
 4   mnth        17379 non-null  int64  
 5   hr          17379 non-null  int64  
 6   holiday     17379 non-null  int64  
 7   weekday     17379 non-null  int64  
 8   workingday  17379 non-null  int64  
 9   weathersit  17379 non-null  int64  
 10  temp        17379 non-null  float64
 11  atemp       17379 non-null  float64
 12  hum         17379 non-null  float64
 13  windspeed   17379 non-null  float64
 14  casual      17379 non-null  int64  
 15  registered  17379 non-null  int64  
 16  cnt         17379 non-null  int64  
dtypes: float64(4), int64(12), object(1)
memory usage: 2.3+ MB
```

In [7]:
```
day.isna().sum()
```

Out[7]:
```
instant       0
dteday        0
season        0
yr            0
mnth          0
holiday       0
weekday       0
workingday    0
weathersit    0
temp          0
atemp         0
hum           0
windspeed     0
casual        0
registered    0
cnt           0
dtype: int64
```

In [8]:
```
hour.isna().sum()
```

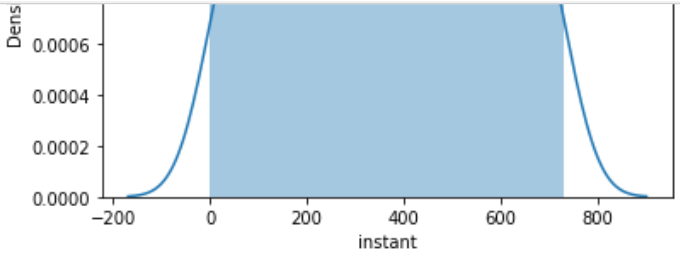
Out[8]:
```
instant       0
dteday        0
season        0
yr            0
mnth          0
hr            0
holiday       0
weekday       0
workingday    0
weathersit    0
temp          0
atemp         0
hum           0
windspeed     0
casual        0
registered    0
cnt           0
dtype: int64
```

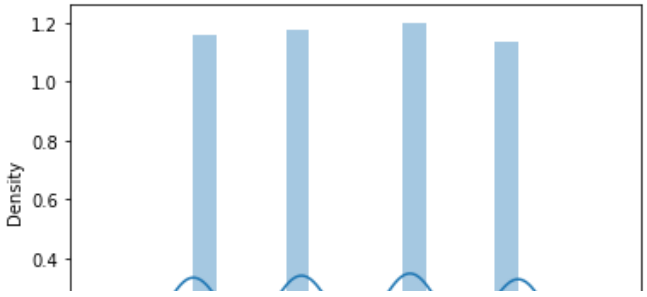
In [9]:
```
day.describe().T
```

Out[9]:

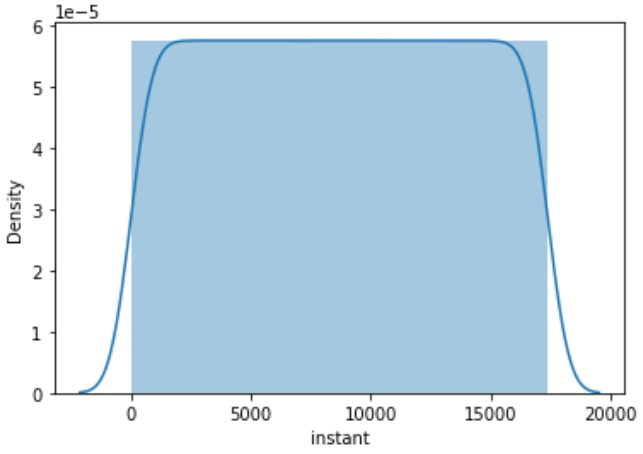
	count	mean	std	min	25%	50%	75%	max
instant	731.00000	366.00000	211.16581	1.00000	183.50000	366.00000	548.50000	731.00000
season	731.00000	2.49658	1.11081	1.00000	2.00000	3.00000	3.00000	4.00000
yr	731.00000	0.50068	0.50034	0.00000	0.00000	1.00000	1.00000	1.00000
mnth	731.00000	6.51984	3.45191	1.00000	4.00000	7.00000	10.00000	12.00000
holiday	731.00000	0.02873	0.16715	0.00000	0.00000	0.00000	0.00000	1.00000
weekday	731.00000	2.99726	2.00479	0.00000	1.00000	3.00000	5.00000	6.00000
workingday	731.00000	0.68399	0.46523	0.00000	0.00000	1.00000	1.00000	1.00000
weathersit	731.00000	1.39535	0.54489	1.00000	1.00000	1.00000	2.00000	3.00000
temp	731.00000	20.31078	7.50509	2.42435	13.82042	20.43165	26.87208	35.32835
atemp	731.00000	23.71770	8.14806	3.95348	16.89213	24.33665	30.43010	42.04480
hum	731.00000	62.78941	14.24291	0.00000	52.00000	62.66670	73.02085	97.25000
windspeed	731.00000	12.76258	5.19236	1.50024	9.04165	12.12533	15.62537	34.00002
casual	731.00000	848.17647	686.62249	2.00000	315.50000	713.00000	1096.00000	3410.00000
registered	731.00000	3656.17237	1560.25638	20.00000	2497.00000	3662.00000	4776.50000	6946.00000
cnt	731.00000	4504.34884	1937.21145	22.00000	3152.00000	4548.00000	5956.00000	8714.00000

In [10]:
```
#a) Distplot
for i in day.select_dtypes(include='int'):
    sns.distplot(day[i])
    plt.show()
```

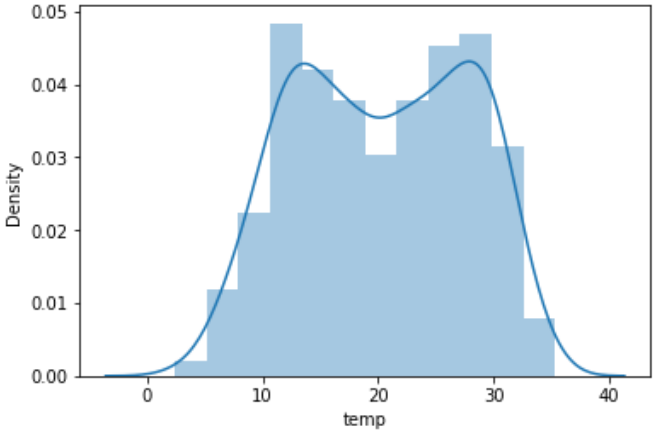


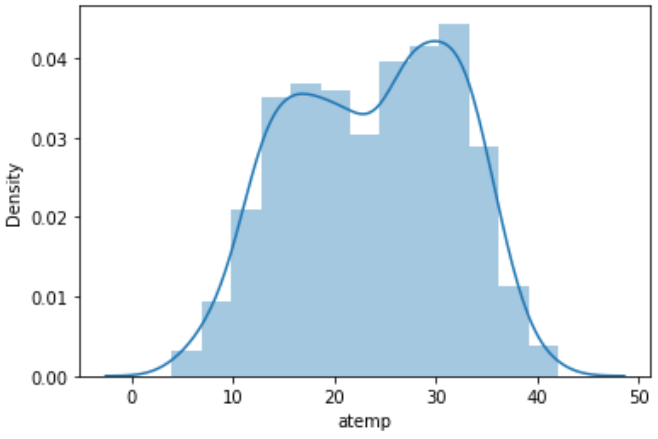


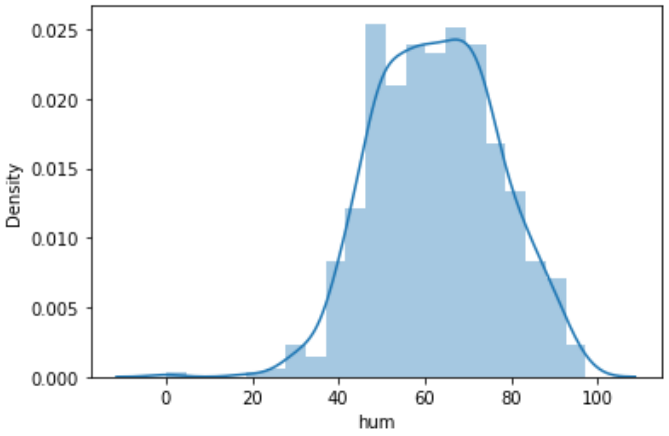
In [11]:
```
#a) Distplot
for i in hour.select_dtypes(include='int'):
    sns.distplot(hour[i])
    plt.show()
```

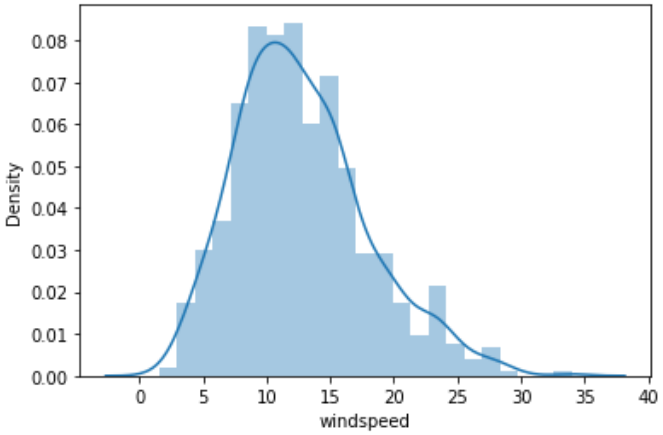


In [14]:
```
#a) Distplot
for i in day.select_dtypes(include='float'):
    sns.distplot(day[i])
    plt.show()
```

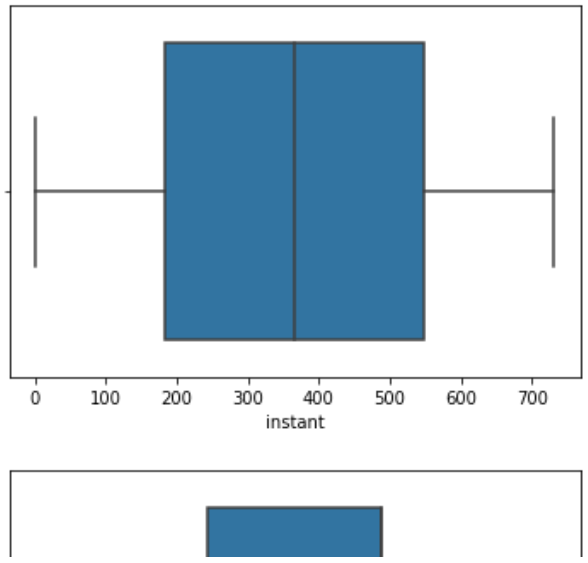




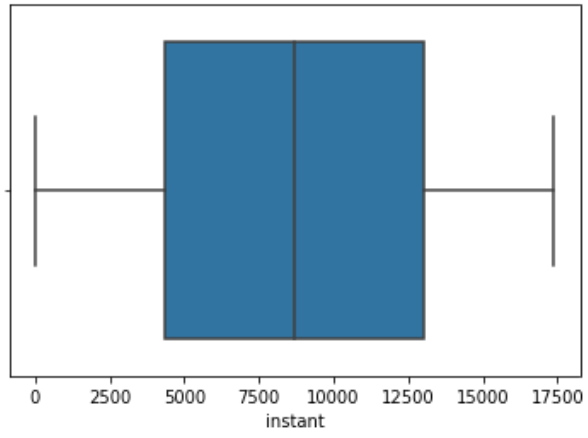

0.025
0.020
0.015
0.010
0.005
0.000
Density
0
20
40
60
80
100
hum


0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
Density
0
5
10
15
20
25
30
35
40
windspeed

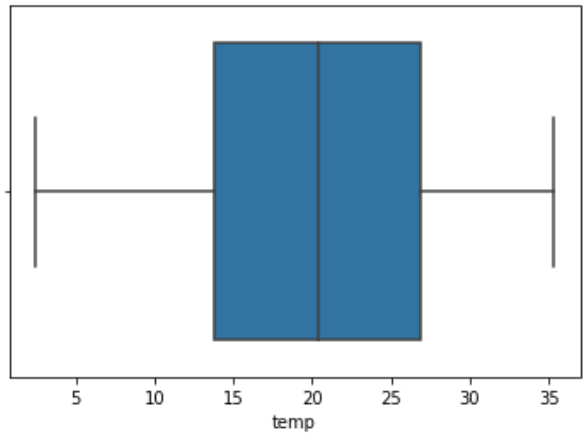
In [28]:
```
#b) Boxplot
for i in day.select_dtypes(include='int'):
    sns.boxplot(day[i])
    plt.show()
```

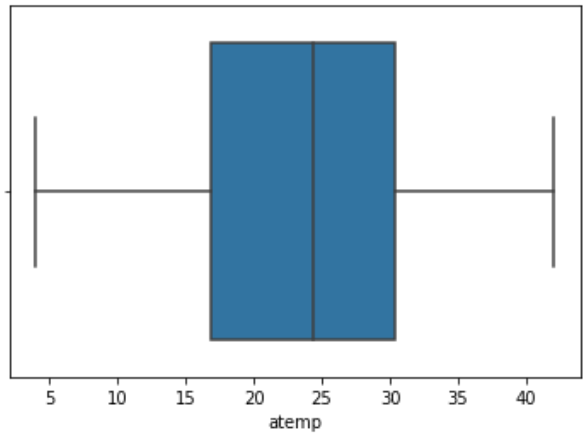


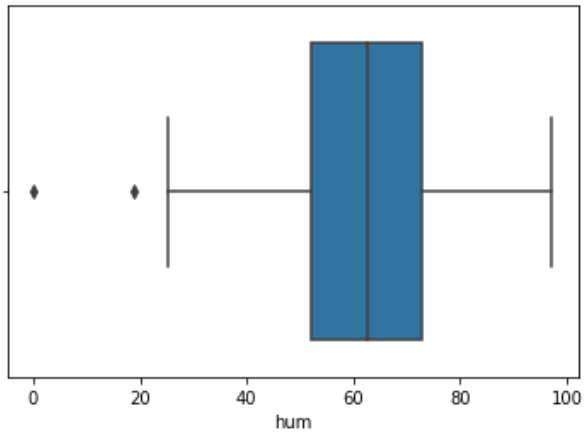
In [29]:
```
#b) Boxplot
for i in hour.select_dtypes(include='int'):
    sns.boxplot(hour[i])
    plt.show()
```

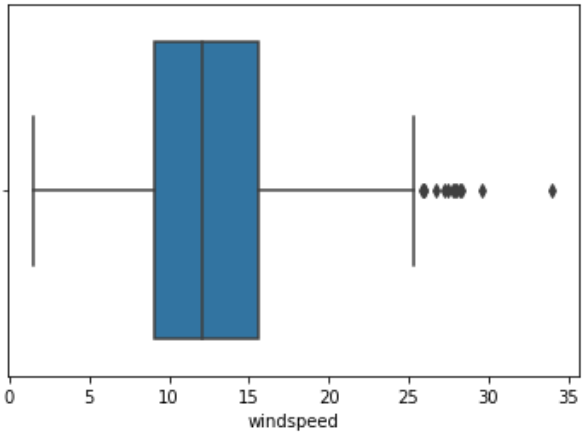


In [15]:
```
#b) Boxplot
for i in day.select_dtypes(include='float'):
    sns.boxplot(day[i])
    plt.show()
```



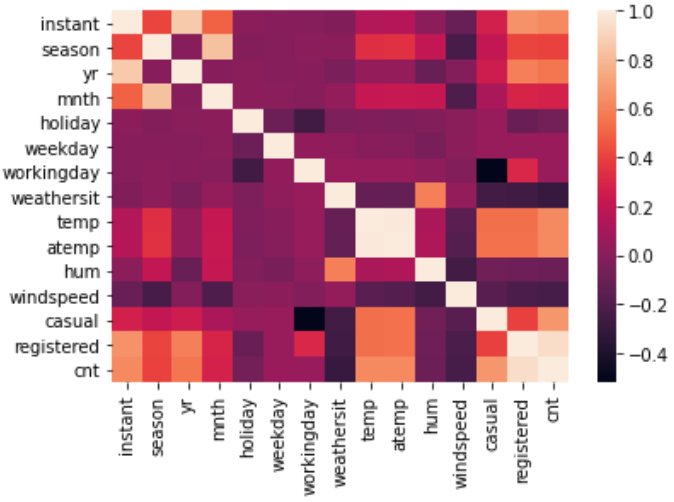



0
20
40
60
80
100
hum


0
5
10
15
20
25
30
35
windspeed

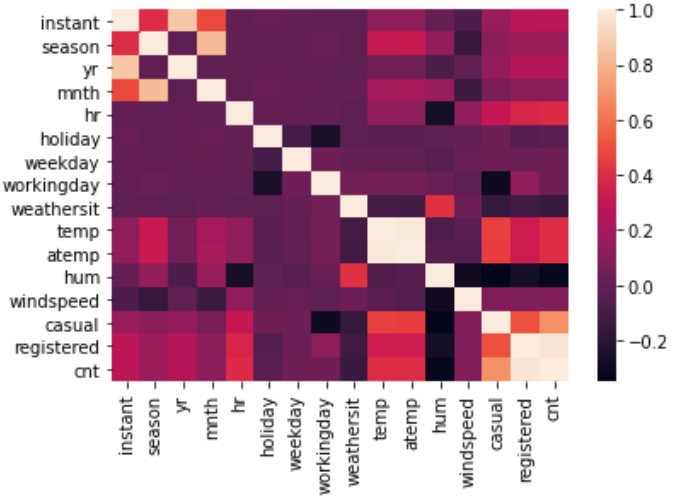
In [16]:
```
#c) HeatMap
sns.heatmap(day.corr())
```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x7f326233fba8>



In [17]:
```
#c) HeatMap
sns.heatmap(hour.corr())
```

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x7f326acca1d0>



In [18]:
```
day.corr()['cnt']
```

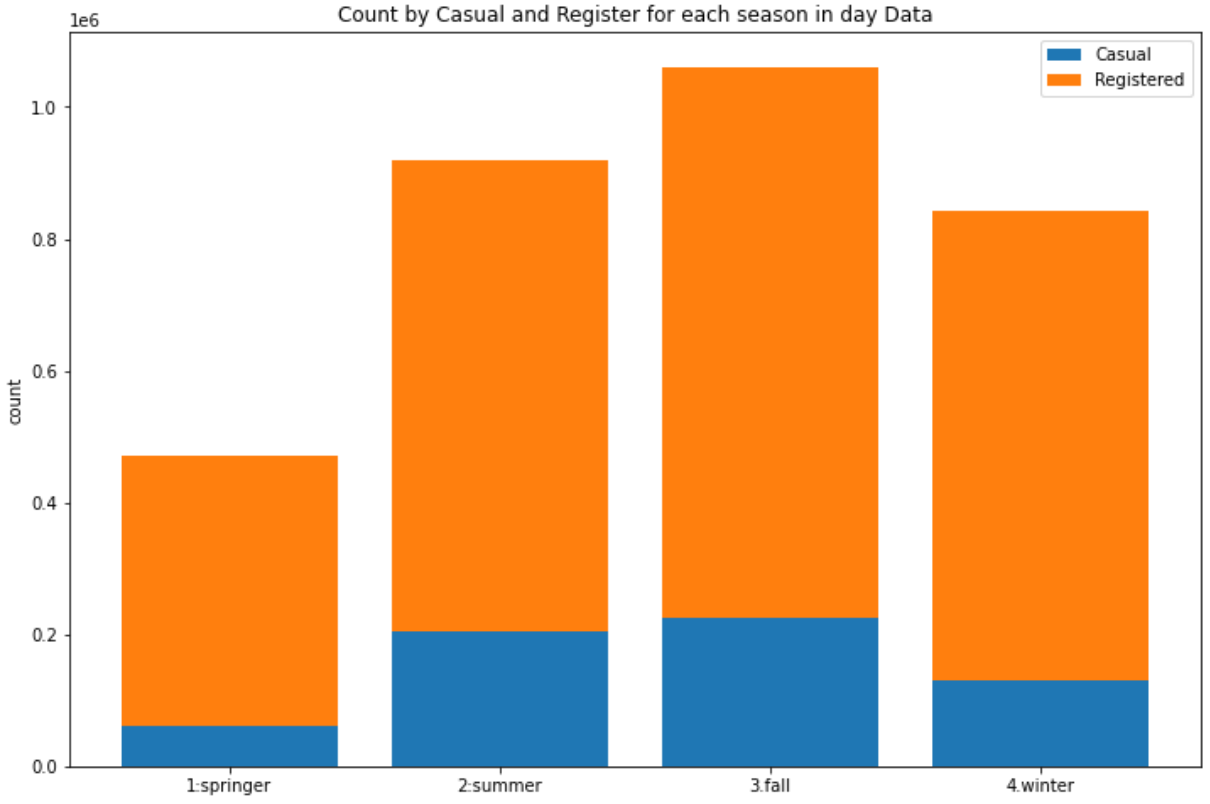
Out[18]:
```
instant       0.62883
season        0.40610
yr            0.56671
mnth          0.27998
holiday      -0.06835
weekday       0.06744
workingday    0.06116
weathersit   -0.29739
temp          0.62749
atemp         0.63107
hum          -0.10066
windspeed    -0.23454
casual        0.67280
registered    0.94552
cnt           1.00000
Name: cnt, dtype: float64
```

In [19]:
```
hour.corr()['cnt']
```

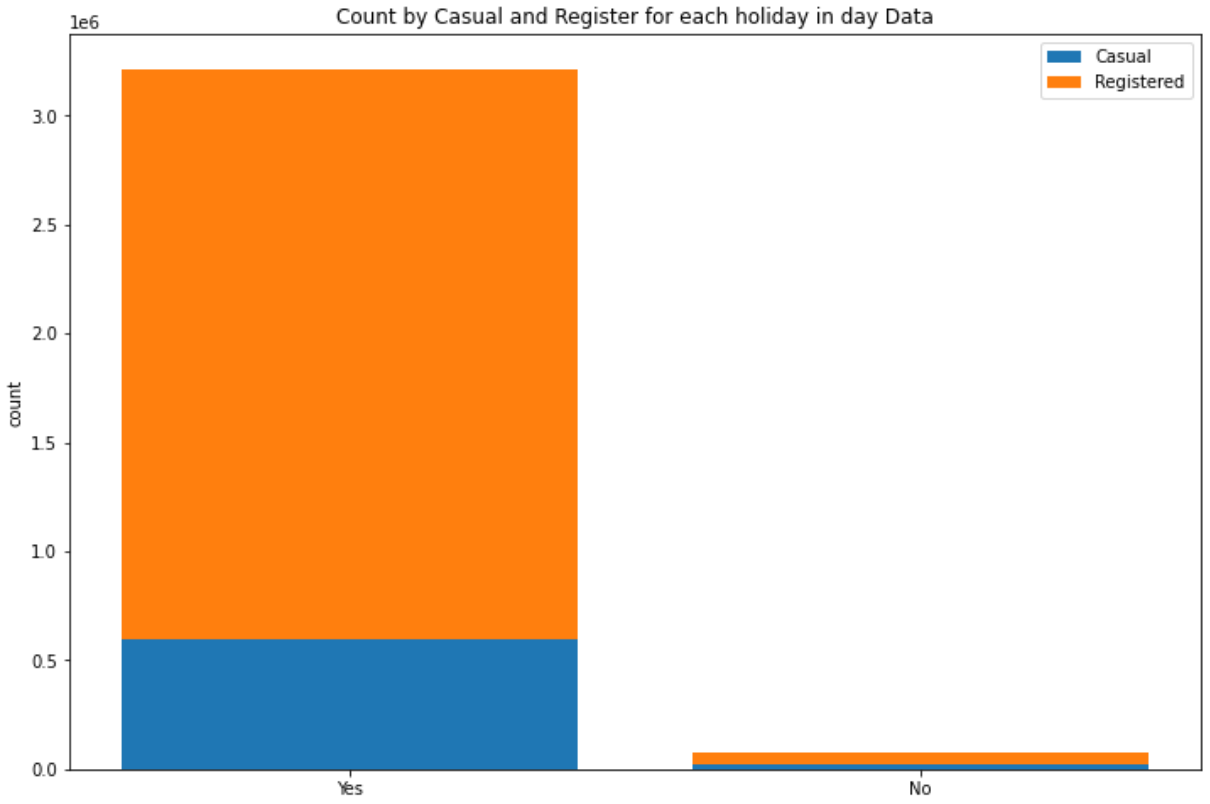
Out[19]:
```
instant       0.27838
season        0.17806
yr            0.25049
mnth          0.12064
hr            0.39407
holiday      -0.03093
weekday       0.02690
workingday    0.03028
weathersit   -0.14243
temp          0.40477
atemp         0.40093
hum          -0.32291
windspeed     0.09323
casual        0.69456
registered    0.97215
cnt           1.00000
Name: cnt, dtype: float64
```

In [20]:
```
#d) Stacked bar chart
def get_df_name(df):
    name = [x for x in globals() if globals()[x] is df][0]
    return name

def plot_stack_bar_chart(data, col, name):
    plt.figure(figsize=(12,8))
    #Data Here I taken related to Casual People
    p1 = plt.bar(data[col].unique(),
                 data.groupby([col])['casual'].sum())

    #Data Here I taken related to Registered People
    p2 = plt.bar(data[col].unique(),
                 data.groupby([col])['registered'].sum(),
                  bottom = data.groupby([col])['casual'].sum())

    #ylabel is used to set label to the Y - Axis
    plt.ylabel('count')

    #To give title at the top of the chart
    plt.title("Count by Casual and Register for each {} in {} Data".format(col, get_df_name(da

    #The names which need to be generated on the X - Axis
    plt.xticks(data[col].unique(), name)

    #To display the legend i.e., which is on the top-right side corner of the chart
    plt.legend((p1[0], p2[0]), ('Casual', 'Registered'))
    plt.show()
```

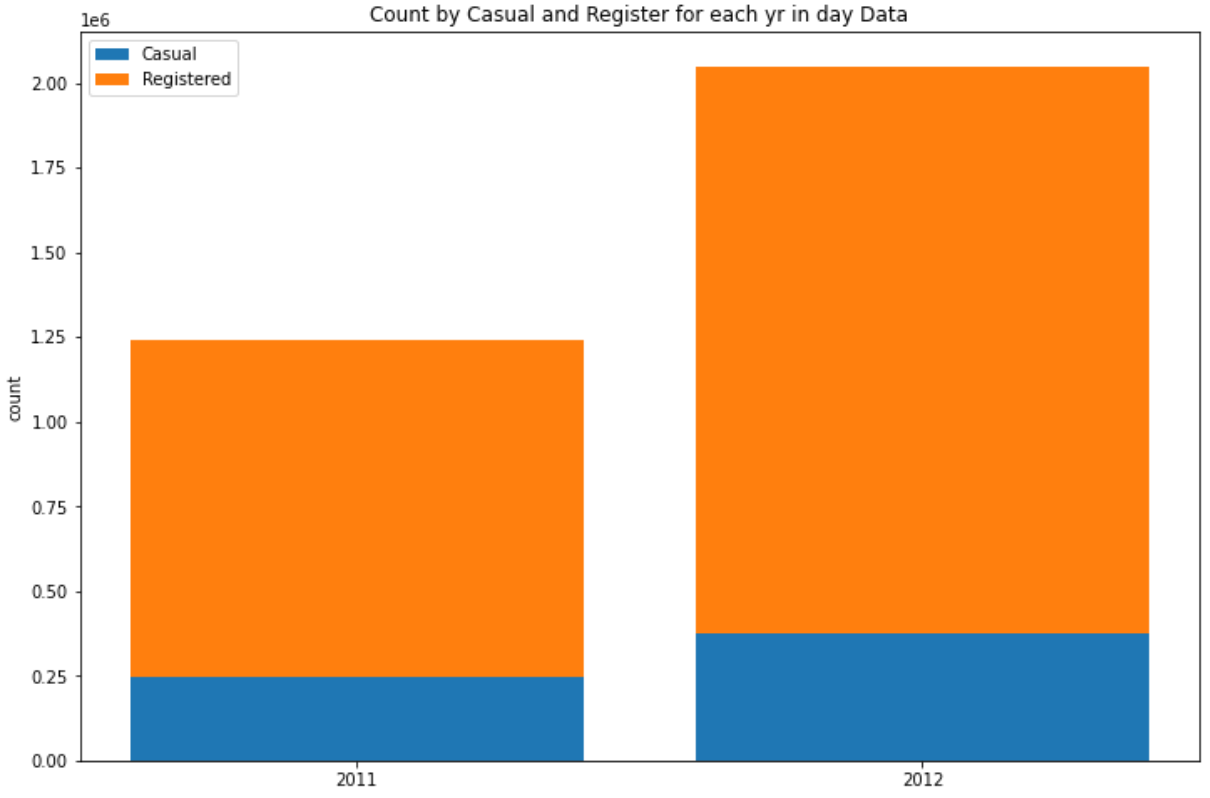
In [21]:
```
#Using this function def. I have given three paramenter i.e., (1)Day, (2)Season & (3)springer,
plot_stack_bar_chart(day, 'season', ('1:springer', '2:summer', '3.fall', '4.winter'))
```



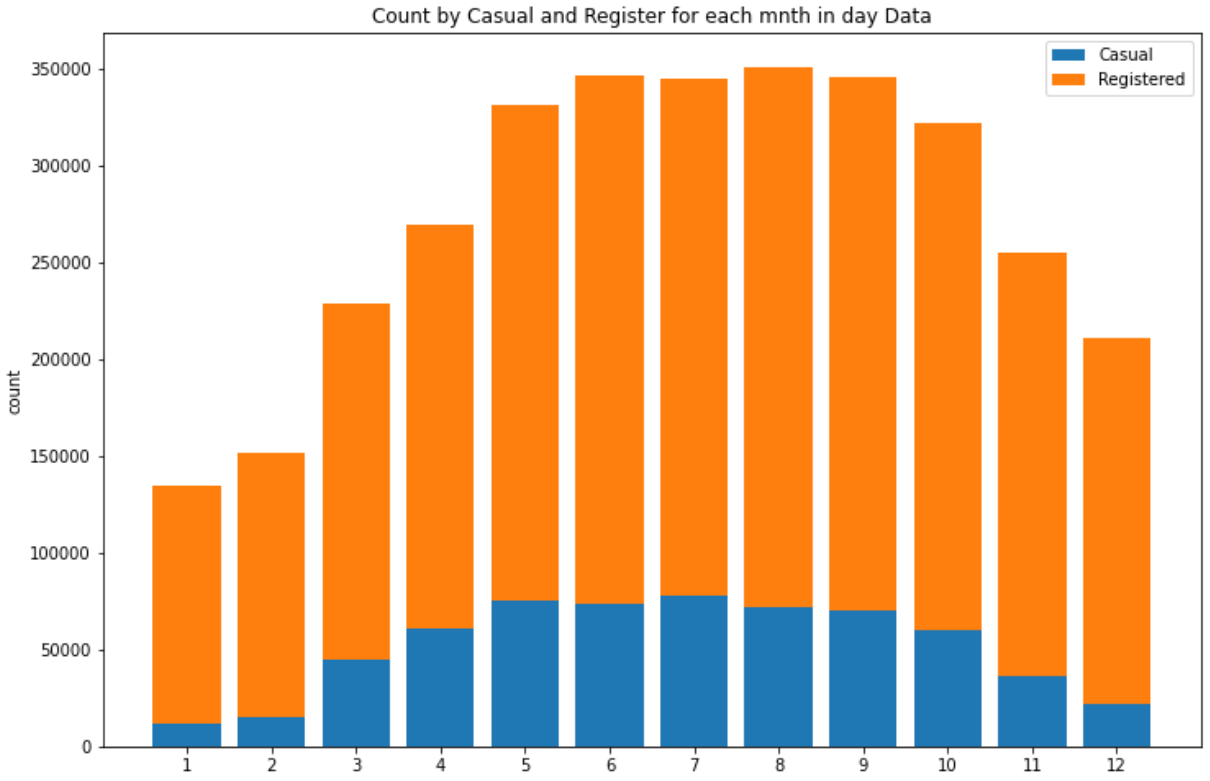
In [22]:
```
plot_stack_bar_chart(day, 'holiday', ('Yes', 'No'))
```



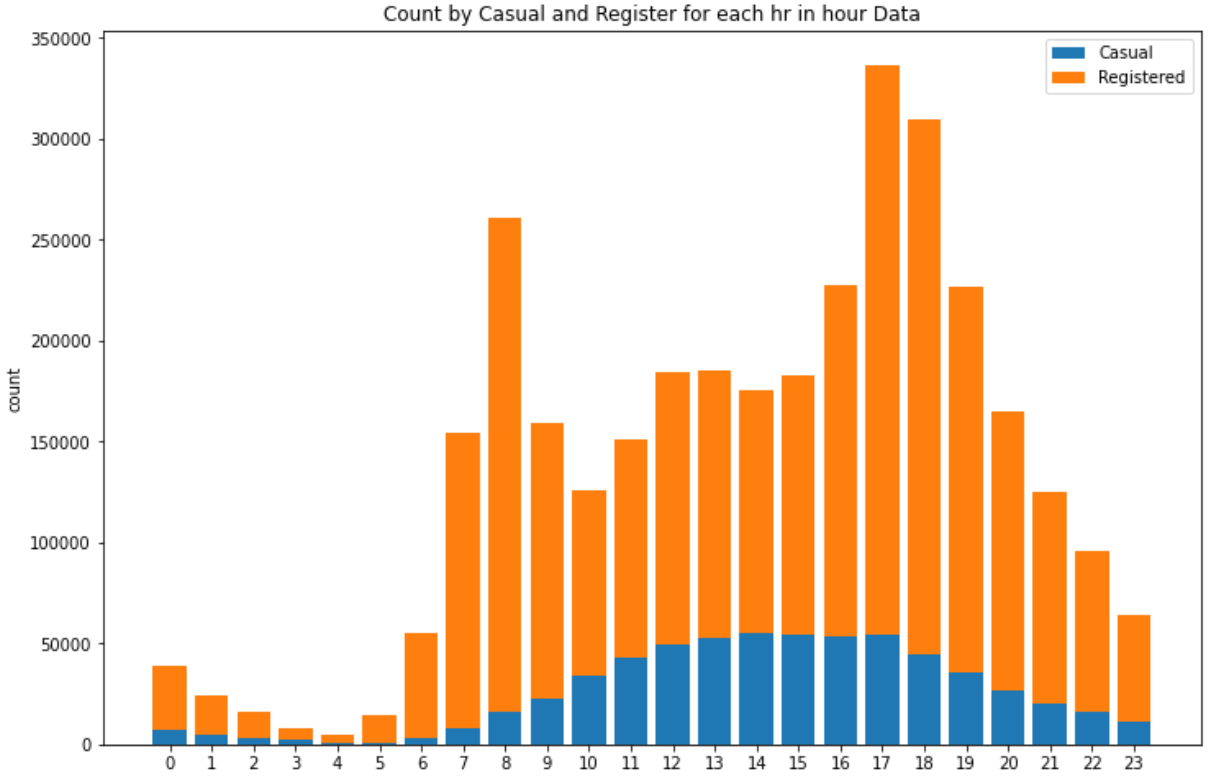
In [23]:
```
plot_stack_bar_chart(day, 'yr', ('2011', '2012'))
```



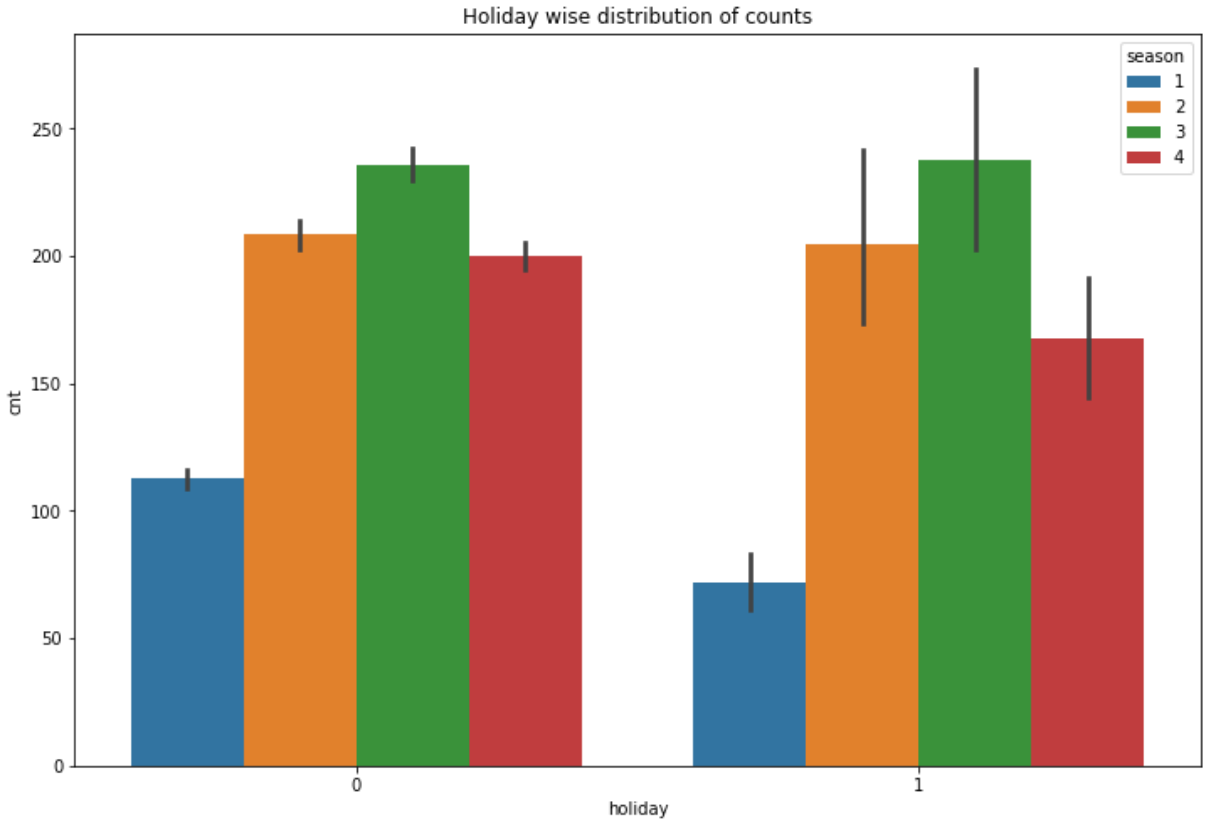
In [24]:
```
plot_stack_bar_chart(day, 'mnth', [str(i) for i in day['mnth'].unique()])
```



In [25]:
```
plot_stack_bar_chart(hour, 'hr', [str(i) for i in hour['hr'].unique()])
```



In [26]:
```
#e) Barplot
plt.figure(figsize=(12, 8))
sns.barplot(x = hour['holiday'], y = hour['cnt'], hue = hour['season'])
plt.title('Holiday wise distribution of counts')
plt.show()
```



In [27]:
```
#e) Barplot
plt.figure(figsize=(12, 8))
sns.barplot(x = day['mnth'], y = day['registered'], hue = day['season'])
plt.title('Month wise distribution of registered')
plt.show()
```