WELCOME!

Welcome to "Bike Demand Visualization Project" which is the capstone project of Data Visualization Lessons . As you know recently, free or affordable access to bicycles has been provided for short-distance trips in an urban area as an alternative to motorized public transport or private vehicles. Thus, it is aimed to reduce traffic congestion, noise and air pollution.

The aim of this project is to reveal the current patterns in the data by showing the historical data of London bike shares with visualization tools.

This will allow us to X-ray the data as part of the EDA process before setting up a machine learning model.

In [1]:

```
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

Table of Contents:

Goal

---> To determine the factors that increase or decrease bike shares <--

Determines

Features

- timestamp - timestamp field for grouping the data
- cnt - the count of a new bike shares
- t1 - real temperature in C
- t2 - temperature in C “feels like”
- hum - humidity in percentage
- wind_speed - wind speed in km/h
- weather_code - category of the weather
- is_holiday - boolean field - 1 holiday / 0 non holiday
- is_weekend - boolean field - 1 if the day is weekend
- season - category field meteorological seasons: 0-spring ; 1-summer; 2-fall; 3-winter.

"weather_code" category description:

- 1 = Clear ; mostly clear but have some values with haze/fog/patches of fog/ fog in vicinity
- 2 = scattered clouds / few clouds
- 3 = Broken clouds
- 4 = Cloudy
- 7 = Rain/ light Rain shower/ Light rain
- 10 = rain with thunderstorm
- 26 = snowfall
- 94 = Freezing Fog

Initially, the task of discovering data will be waiting for you as always. Recognize features, detect missing values, outliers etc. Review the data from various angles in different time breakdowns. For example, visualize the distribution of bike shares by day of the week. With this graph, you will be able to easily observe and make inferences how people's behavior changes daily. Likewise, you can make hourly, monthly, seasonally etc. analyzes. In addition, you can analyze correlation of variables with a heatmap.

Data Preprocessing

1. Import Libraries

In [2]:

```
import numpy as np
import pandas as pd

import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline

#import warnings ;
#warnings.filterwarnings("ignore")
```

2. Read Dataset

In [3]:

```
df = pd.read_csv(r"C:\Users\EmincanY\Desktop\GitHub\EDA\Data_Visulization\DataVis-CapStoneP
df.head()
```

Out[3]:

	timestamp	cnt	t1	t2	hum	wind_speed	weather_code	is_holiday	is_weekend	season
0	2015-01-04 00:00:00	182	3.0	2.0	93.0	6.0	3.0	0.0	1.0	3.0
1	2015-01-04 01:00:00	138	3.0	2.5	93.0	5.0	1.0	0.0	1.0	3.0
2	2015-01-04 02:00:00	134	2.5	2.5	96.5	0.0	1.0	0.0	1.0	3.0
3	2015-01-04 03:00:00	72	2.0	2.0	100.0	0.0	1.0	0.0	1.0	3.0
4	2015-01-04 04:00:00	47	2.0	0.0	93.0	6.5	1.0	0.0	1.0	3.0

3. Check missing values and if there are any duplicate rows or not.

In [4]:

```
df.isnull().sum() # There is no any nan value.
```

Out[4]:

```
timestamp       0
cnt             0
t1              0
t2              0
hum             0
wind_speed      0
weather_code    0
is_holiday      0
is_weekend      0
season          0
dtype: int64
```

In [5]:

```
df.drop_duplicates() # There is no duplicate rows. No one dropped.
```

Out[5]:

	timestamp	cnt	t1	t2	hum	wind_speed	weather_code	is_holiday	is_weekend	s
0	2015-01-04 00:00:00	182	3.0	2.0	93.0	6.0	3.0	0.0	1.0	
1	2015-01-04 01:00:00	138	3.0	2.5	93.0	5.0	1.0	0.0	1.0	
2	2015-01-04 02:00:00	134	2.5	2.5	96.5	0.0	1.0	0.0	1.0	
3	2015-01-04 03:00:00	72	2.0	2.0	100.0	0.0	1.0	0.0	1.0	
4	2015-01-04 04:00:00	47	2.0	0.0	93.0	6.5	1.0	0.0	1.0	
...	...	...	...	...	...	...	...	...	...	
17409	2017-01-03 19:00:00	1042	5.0	1.0	81.0	19.0	3.0	0.0	0.0	
17410	2017-01-03 20:00:00	541	5.0	1.0	81.0	21.0	4.0	0.0	0.0	
17411	2017-01-03 21:00:00	337	5.5	1.5	78.5	24.0	4.0	0.0	0.0	
17412	2017-01-03 22:00:00	224	5.5	1.5	76.0	23.0	4.0	0.0	0.0	
17413	2017-01-03 23:00:00	139	5.0	1.0	76.0	22.0	2.0	0.0	0.0	

17414 rows × 10 columns

Feature Engineering

1. Look at the data type of each variable, transform timestamp in type, and set it as index.

In [6]:

```
df.dtypes
```

Out[6]:

```
timestamp        object
cnt               int64
t1              float64
t2              float64
hum             float64
wind_speed      float64
weather_code    float64
is_holiday      float64
is_weekend      float64
season          float64
dtype: object
```

In [7]:

```
df.head()
```

Out[7]:

	timestamp	cnt	t1	t2	hum	wind_speed	weather_code	is_holiday	is_weekend	season
0	2015-01-04 00:00:00	182	3.0	2.0	93.0	6.0	3.0	0.0	1.0	3.0
1	2015-01-04 01:00:00	138	3.0	2.5	93.0	5.0	1.0	0.0	1.0	3.0
2	2015-01-04 02:00:00	134	2.5	2.5	96.5	0.0	1.0	0.0	1.0	3.0
3	2015-01-04 03:00:00	72	2.0	2.0	100.0	0.0	1.0	0.0	1.0	3.0
4	2015-01-04 04:00:00	47	2.0	0.0	93.0	6.5	1.0	0.0	1.0	3.0

In [8]:

```
df["timestamp"] = df.timestamp.astype("datetime64")  #df['timestamp'] = pd.to_datetime(df['
df.dtypes
```

Out[8]:

```
timestamp       datetime64[ns]
cnt                      int64
t1                     float64
t2                     float64
hum                    float64
wind_speed             float64
weather_code           float64
is_holiday             float64
is_weekend             float64
season                 float64
dtype: object
```

In [9]:

```
df.set_index("timestamp" , inplace = True ) # We set new index with timestamp column.
```

2. Add season_name Feature

In [10]:

```
mapping = {0.0 : "Spring" , 1.0 : "Summer" , 2.0 : "Fall" , 3.0 : "Winter"}
df["season_name"] = df.season.map(mapping) # We created season_name column to better graphs
```

In [11]:

```
df.head()
```

Out[11]:

timestamp	cnt	t1	t2	hum	wind_speed	weather_code	is_holiday	is_weekend	season	s
2015-01-04 00:00:00	182	3.0	2.0	93.0	6.0	3.0	0.0	1.0	3.0	
2015-01-04 01:00:00	138	3.0	2.5	93.0	5.0	1.0	0.0	1.0	3.0	
2015-01-04 02:00:00	134	2.5	2.5	96.5	0.0	1.0	0.0	1.0	3.0	
2015-01-04 03:00:00	72	2.0	2.0	100.0	0.0	1.0	0.0	1.0	3.0	
2015-01-04 04:00:00	47	2.0	0.0	93.0	6.5	1.0	0.0	1.0	3.0	

3. Extract new columns (day of the week, day of the month, hour, month, season, year etc.)

In [12]:

```
df["year"] = df.index.year
df["month"] = df.index.month
df["day"] = df.index.day
df["hour"] = df.index.hour
```

In [13]:

```
df['dayofweek_num'] = df.index.dayofweek
df['dayofweek_name'] = df.index.day_name()
df["dayofmonth_num"] = df.index.strftime("%d")
```

In [14]:

```
df.head()
```

Out[14]:

timestamp	cnt	t1	t2	hum	wind_speed	weather_code	is_holiday	is_weekend	season	s
2015-01-04 00:00:00	182	3.0	2.0	93.0	6.0	3.0	0.0	1.0	3.0	
2015-01-04 01:00:00	138	3.0	2.5	93.0	5.0	1.0	0.0	1.0	3.0	
2015-01-04 02:00:00	134	2.5	2.5	96.5	0.0	1.0	0.0	1.0	3.0	
2015-01-04 03:00:00	72	2.0	2.0	100.0	0.0	1.0	0.0	1.0	3.0	
2015-01-04 04:00:00	47	2.0	0.0	93.0	6.5	1.0	0.0	1.0	3.0	

In [15]:

```
df.dayofmonth_num = df.dayofmonth_num.astype("int64")
```

Data Visulization

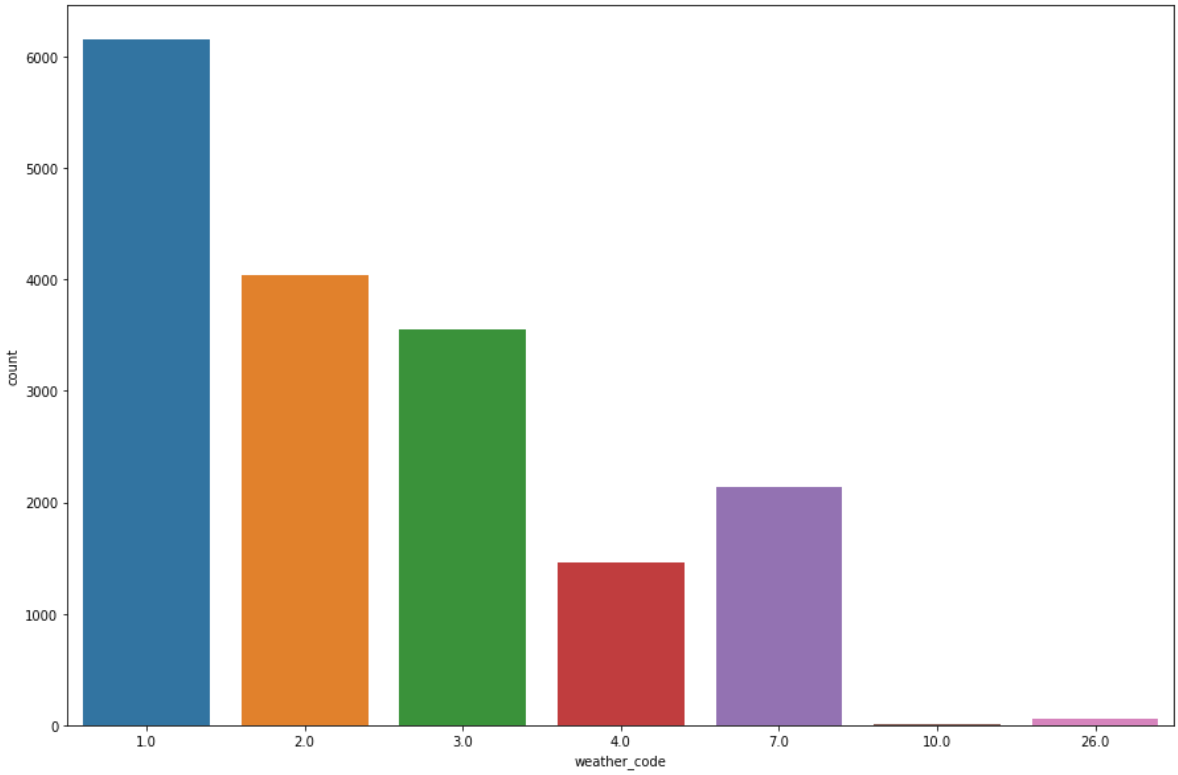
1. Weather Counts

- 1 = Clear ; mostly clear but have some values with haze/fog/patches of fog/ fog in vicinity
- 2 = scattered clouds / few clouds
- 3 = Broken clouds
- 4 = Cloudy
- 7 = Rain/ light Rain shower/ Light rain

- 10 = rain with thunderstorm
- 26 = snowfall
- 94 = Freezing Fog

In [16]:

```
plt.figure(figsize=(15,10))
sns.countplot(x='weather_code',data=df);
```



2. Plot the distribution of various discrete features on (Season, holiday, weekend and weathercode)

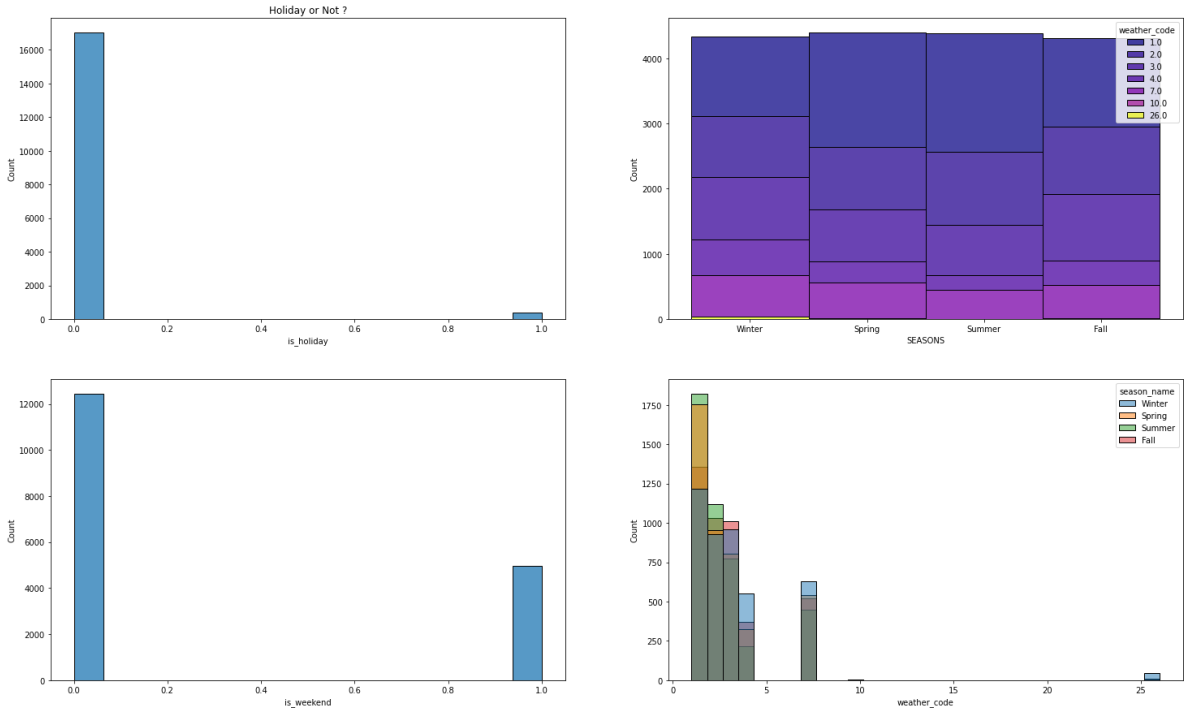
In [17]:

```
fig, ax = plt.subplots(2,2 , figsize = (25,15))


a = sns.histplot(ax = ax[0][0] , data = df , x = "is_holiday" )
a.set_title("Holiday or Not ?") # Değişkene atayım set yap.

b = sns.histplot(data = df , x = "season_name" , ax = ax[0][1] , hue = "weather_code", pale
b.set_xlabel("SEASONS")

#c = sns.histplot(x = df["is_weekend"] , ax = ax[1][0] , hue = df["weather_code"] )
c = sns.histplot( x = df["is_weekend"] , ax = ax[1][0])
d = sns.histplot(x = df["weather_code"] , ax = ax[1][1] , bins = 30 , hue = df["season_name
```



3. Visualize the correlation with a heatmap

In [18]:

```
df.head()
```

Out[18]:

timestamp	cnt	t1	t2	hum	wind_speed	weather_code	is_holiday	is_weekend	season	s
2015-01-04 00:00:00	182	3.0	2.0	93.0	6.0	3.0	0.0	1.0	3.0	
2015-01-04 01:00:00	138	3.0	2.5	93.0	5.0	1.0	0.0	1.0	3.0	
2015-01-04 02:00:00	134	2.5	2.5	96.5	0.0	1.0	0.0	1.0	3.0	
2015-01-04 03:00:00	72	2.0	2.0	100.0	0.0	1.0	0.0	1.0	3.0	
2015-01-04 04:00:00	47	2.0	0.0	93.0	6.5	1.0	0.0	1.0	3.0	

In [19]:

```
df.corr()
```

Out[19]:

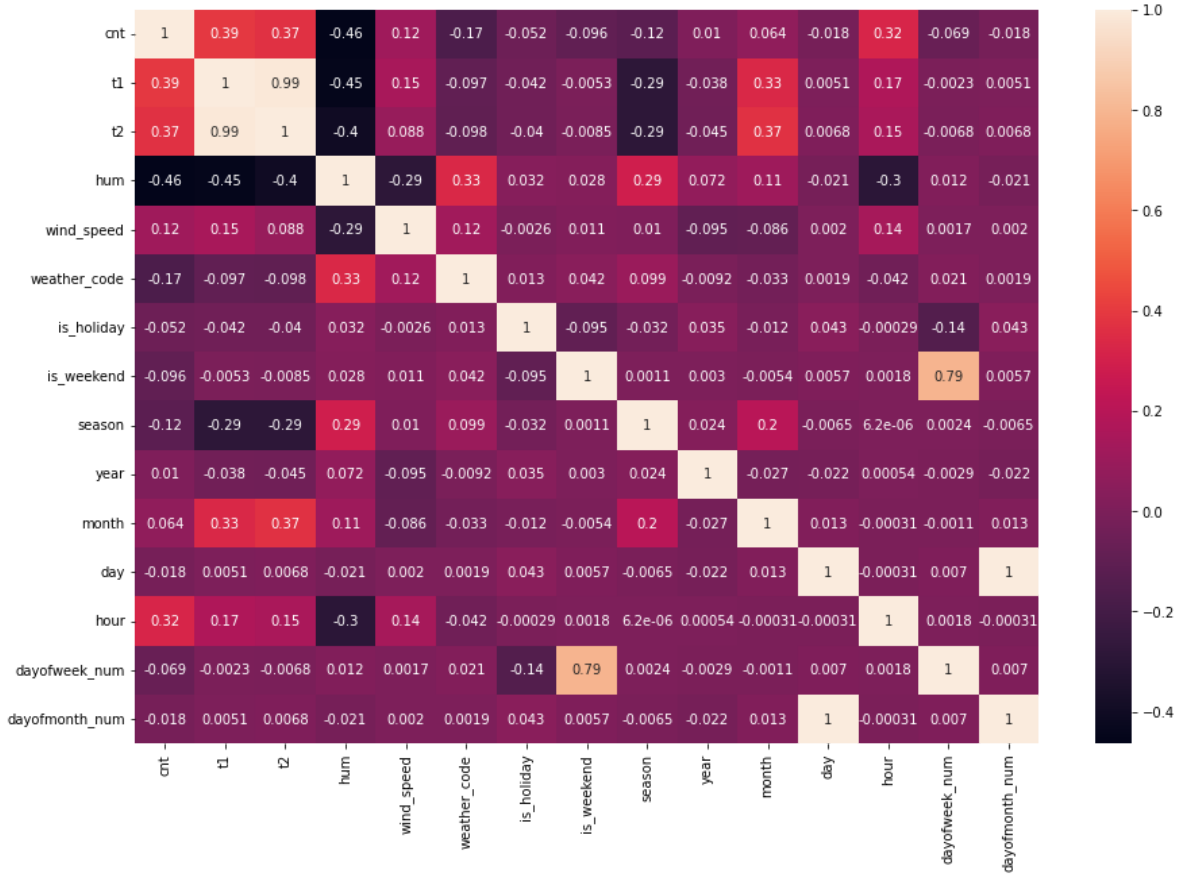
	cnt	t1	t2	hum	wind_speed	weather_code	is_holi
cnt	1.000000	0.388798	0.369035	-0.462901	0.116295	-0.166633	-0.051
t1	0.388798	1.000000	0.988344	-0.447781	0.145471	-0.097114	-0.042
t2	0.369035	0.988344	1.000000	-0.403495	0.088409	-0.098385	-0.040
hum	-0.462901	-0.447781	-0.403495	1.000000	-0.287789	0.334750	0.032
wind_speed	0.116295	0.145471	0.088409	-0.287789	1.000000	0.124803	-0.002
weather_code	-0.166633	-0.097114	-0.098385	0.334750	0.124803	1.000000	0.012
is_holiday	-0.051698	-0.042233	-0.040051	0.032068	-0.002606	0.012939	1.000
is_weekend	-0.096499	-0.005342	-0.008510	0.028098	0.011479	0.042362	-0.094
season	-0.116180	-0.285851	-0.285900	0.290381	0.010305	0.098976	-0.032
year	0.010046	-0.037959	-0.044972	0.072443	-0.094739	-0.009234	0.034
month	0.063757	0.332712	0.368366	0.113149	-0.086383	-0.033253	-0.011
day	-0.017887	0.005072	0.006791	-0.020868	0.002040	0.001904	0.042
hour	0.324423	0.168708	0.153956	-0.295653	0.141792	-0.041786	-0.000
dayofweek_num	-0.068688	-0.002317	-0.006824	0.011556	0.001708	0.020619	-0.144
dayofmonth_num	-0.017887	0.005072	0.006791	-0.020868	0.002040	0.001904	0.042

In [20]:

```
from matplotlib.pyplot import annotate


plt.figure(figsize= (15,10))

sns.heatmap(df.corr() , annot = True);
```



4. Visualize the correlation of the target variable and the other features with barplot

In [21]:

```
df.corr()
```

Out[21]:

	cnt	t1	t2	hum	wind_speed	weather_code	is_holi
cnt	1.000000	0.388798	0.369035	-0.462901	0.116295	-0.166633	-0.051
t1	0.388798	1.000000	0.988344	-0.447781	0.145471	-0.097114	-0.042
t2	0.369035	0.988344	1.000000	-0.403495	0.088409	-0.098385	-0.040
hum	-0.462901	-0.447781	-0.403495	1.000000	-0.287789	0.334750	0.032
wind_speed	0.116295	0.145471	0.088409	-0.287789	1.000000	0.124803	-0.002
weather_code	-0.166633	-0.097114	-0.098385	0.334750	0.124803	1.000000	0.012
is_holiday	-0.051698	-0.042233	-0.040051	0.032068	-0.002606	0.012939	1.000
is_weekend	-0.096499	-0.005342	-0.008510	0.028098	0.011479	0.042362	-0.094
season	-0.116180	-0.285851	-0.285900	0.290381	0.010305	0.098976	-0.032
year	0.010046	-0.037959	-0.044972	0.072443	-0.094739	-0.009234	0.034
month	0.063757	0.332712	0.368366	0.113149	-0.086383	-0.033253	-0.011
day	-0.017887	0.005072	0.006791	-0.020868	0.002040	0.001904	0.042
hour	0.324423	0.168708	0.153956	-0.295653	0.141792	-0.041786	-0.000
dayofweek_num	-0.068688	-0.002317	-0.006824	0.011556	0.001708	0.020619	-0.144
dayofmonth_num	-0.017887	0.005072	0.006791	-0.020868	0.002040	0.001904	0.042

In [22]:

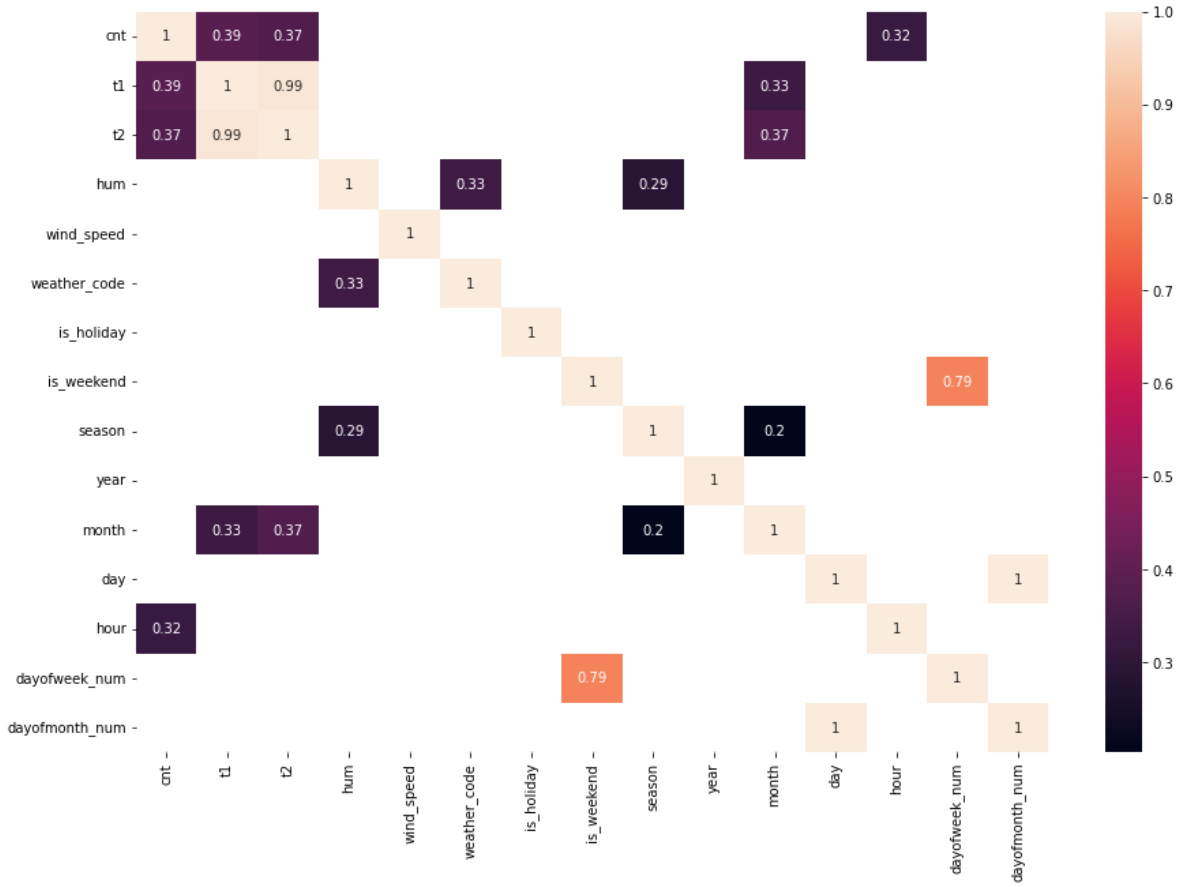
```
df.corr()[df.corr() > 0.2]
```

Out[22]:

	cnt	t1	t2	hum	wind_speed	weather_code	is_holida
cnt	1.000000	0.388798	0.369035	NaN	NaN	NaN	NaI
t1	0.388798	1.000000	0.988344	NaN	NaN	NaN	NaI
t2	0.369035	0.988344	1.000000	NaN	NaN	NaN	NaI
hum	NaN	NaN	NaN	1.000000	NaN	0.33475	NaI
wind_speed	NaN	NaN	NaN	NaN	1.0	NaN	NaI
weather_code	NaN	NaN	NaN	0.334750	NaN	1.00000	NaI
is_holiday	NaN	NaN	NaN	NaN	NaN	NaN	1.
is_weekend	NaN	NaN	NaN	NaN	NaN	NaN	NaI
season	NaN	NaN	NaN	0.290381	NaN	NaN	NaI
year	NaN	NaN	NaN	NaN	NaN	NaN	NaI
month	NaN	0.332712	0.368366	NaN	NaN	NaN	NaI
day	NaN	NaN	NaN	NaN	NaN	NaN	NaI
hour	0.324423	NaN	NaN	NaN	NaN	NaN	NaI
dayofweek_num	NaN	NaN	NaN	NaN	NaN	NaN	NaI
dayofmonth_num	NaN	NaN	NaN	NaN	NaN	NaN	NaI

In [23]:

```
plt.figure(figsize=(15,10))
sns.heatmap(df.corr()[df.corr() > 0.2] , annot = True);
```



In [24]:

```
df.head()
```

Out[24]:

timestamp	cnt	t1	t2	hum	wind_speed	weather_code	is_holiday	is_weekend	season	s
2015-01-04 00:00:00	182	3.0	2.0	93.0	6.0	3.0	0.0	1.0	3.0	
2015-01-04 01:00:00	138	3.0	2.5	93.0	5.0	1.0	0.0	1.0	3.0	
2015-01-04 02:00:00	134	2.5	2.5	96.5	0.0	1.0	0.0	1.0	3.0	
2015-01-04 03:00:00	72	2.0	2.0	100.0	0.0	1.0	0.0	1.0	3.0	
2015-01-04 04:00:00	47	2.0	0.0	93.0	6.5	1.0	0.0	1.0	3.0	

In [25]:

```
df.corr().cnt.index # This will be x axis.
```

Out[25]:

```
Index(['cnt', 't1', 't2', 'hum', 'wind_speed', 'weather_code', 'is_holiday',
       'is_weekend', 'season', 'year', 'month', 'day', 'hour', 'dayofweek_nu
m',
       'dayofmonth_num'],
      dtype='object')
```

In [26]:

```
df.corr()["cnt"]# This will be y axis.
```

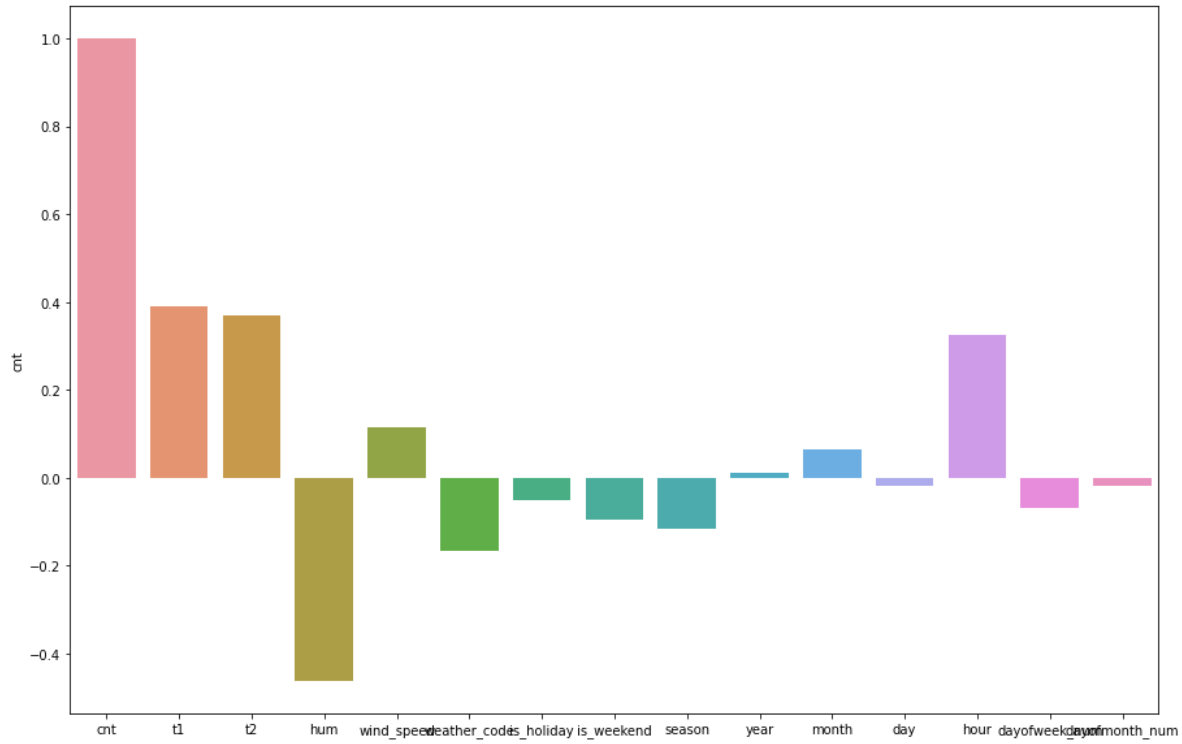
Out[26]:

```
cnt              1.000000
t1               0.388798
t2               0.369035
hum             -0.462901
wind_speed       0.116295
weather_code    -0.166633
is_holiday      -0.051698
is_weekend      -0.096499
season          -0.116180
year             0.010046
month            0.063757
day             -0.017887
hour             0.324423
dayofweek_num   -0.068688
dayofmonth_num  -0.017887
Name: cnt, dtype: float64
```

In [27]:

```
plt.figure(figsize=(15,10))

sns.barplot(x = df.corr().cnt.index , y = df.corr().cnt );
```



5. Plot bike shares over time use lineplot.

In [28]:

```
df.head()
```

Out[28]:

timestamp	cnt	t1	t2	hum	wind_speed	weather_code	is_holiday	is_weekend	season	s
2015-01-04 00:00:00	182	3.0	2.0	93.0	6.0	3.0	0.0	1.0	3.0	
2015-01-04 01:00:00	138	3.0	2.5	93.0	5.0	1.0	0.0	1.0	3.0	
2015-01-04 02:00:00	134	2.5	2.5	96.5	0.0	1.0	0.0	1.0	3.0	
2015-01-04 03:00:00	72	2.0	2.0	100.0	0.0	1.0	0.0	1.0	3.0	
2015-01-04 04:00:00	47	2.0	0.0	93.0	6.5	1.0	0.0	1.0	3.0	

In [29]:

```
df.index.year
```

Out[29]:

```
Int64Index([2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015,
            ...
            2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017],
           dtype='int64', name='timestamp', length=17414)
```

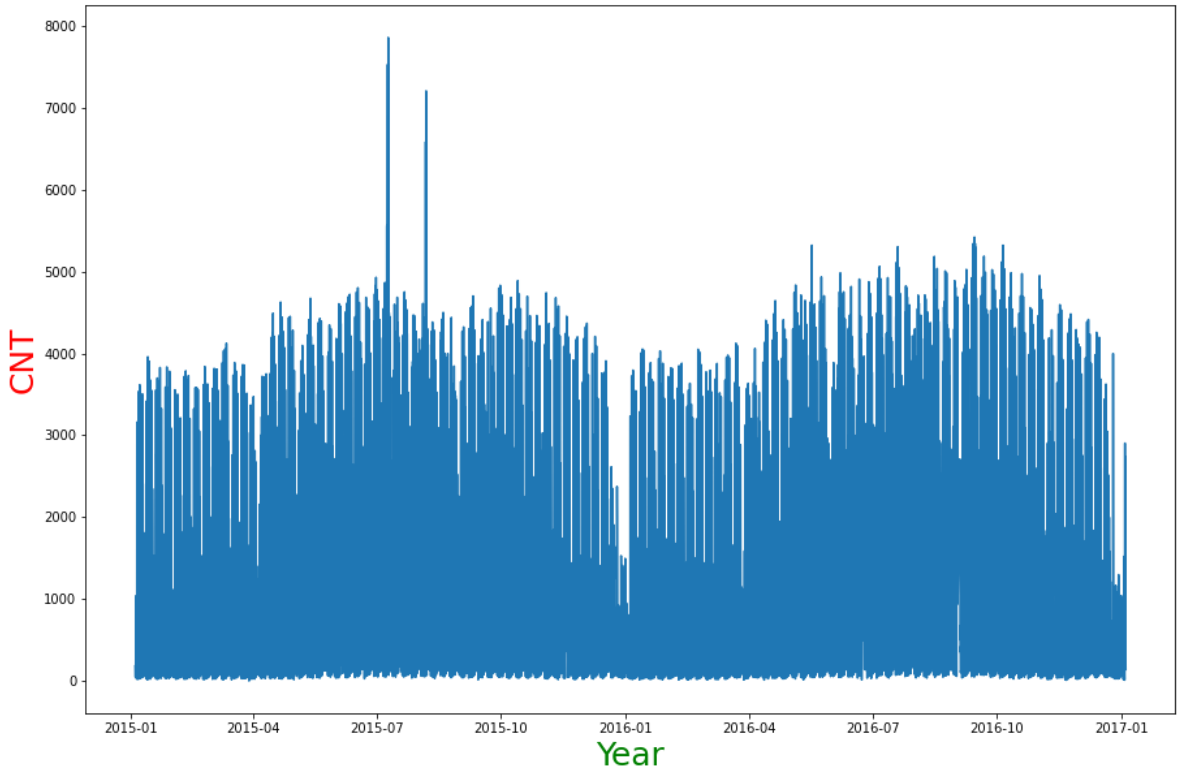
In [30]:

```
from turtle import color


plt.figure(figsize=(15,10));

sns.lineplot(x = df.index , y =df["cnt"]);

plt.xlabel("Year" , fontsize = 25 , color = "green");
plt.ylabel("CNT" , fontsize = 25 , color = "red");
```



6. Plot bike shares by months and year_of_month (use lineplot, pointplot, barplot).

In [31]:

```
df.head()
```

Out[31]:

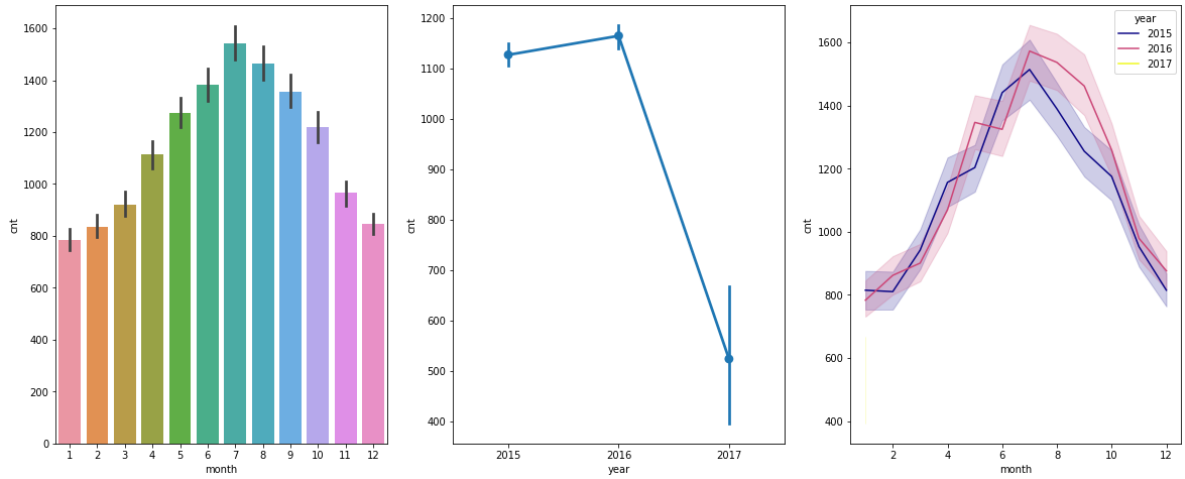
timestamp	cnt	t1	t2	hum	wind_speed	weather_code	is_holiday	is_weekend	season	s
2015-01-04 00:00:00	182	3.0	2.0	93.0	6.0	3.0	0.0	1.0	3.0	
2015-01-04 01:00:00	138	3.0	2.5	93.0	5.0	1.0	0.0	1.0	3.0	
2015-01-04 02:00:00	134	2.5	2.5	96.5	0.0	1.0	0.0	1.0	3.0	
2015-01-04 03:00:00	72	2.0	2.0	100.0	0.0	1.0	0.0	1.0	3.0	
2015-01-04 04:00:00	47	2.0	0.0	93.0	6.5	1.0	0.0	1.0	3.0	

In [32]:

```
from matplotlib.pyplot import xlabel, ylabel


fig, ax = plt.subplots(1,3 , figsize = (20,8))

#sns.lineplot( x = df["month"] , y = df["cnt"], ax = ax[0][0]);
#sns.pointplot( x = df["month"] , y = df["cnt"], ax = ax[0][1]);
sns.barplot( x = df["month"] , y = df["cnt"], ax = ax[0]);
sns.pointplot( x = df["year"] , y = df["cnt"], ax = ax[1]);
sns.lineplot( x = df["month"] , y = df["cnt"] , hue = df["year"] , palette = "plasma" , ax
#sns.lineplot( x = df["year"] , y = df["cnt"], ax = ax[1][2]);
```



7. Plot bike shares by hours on (holidays, weekend, season).

In [33]:

```
df.head()
```

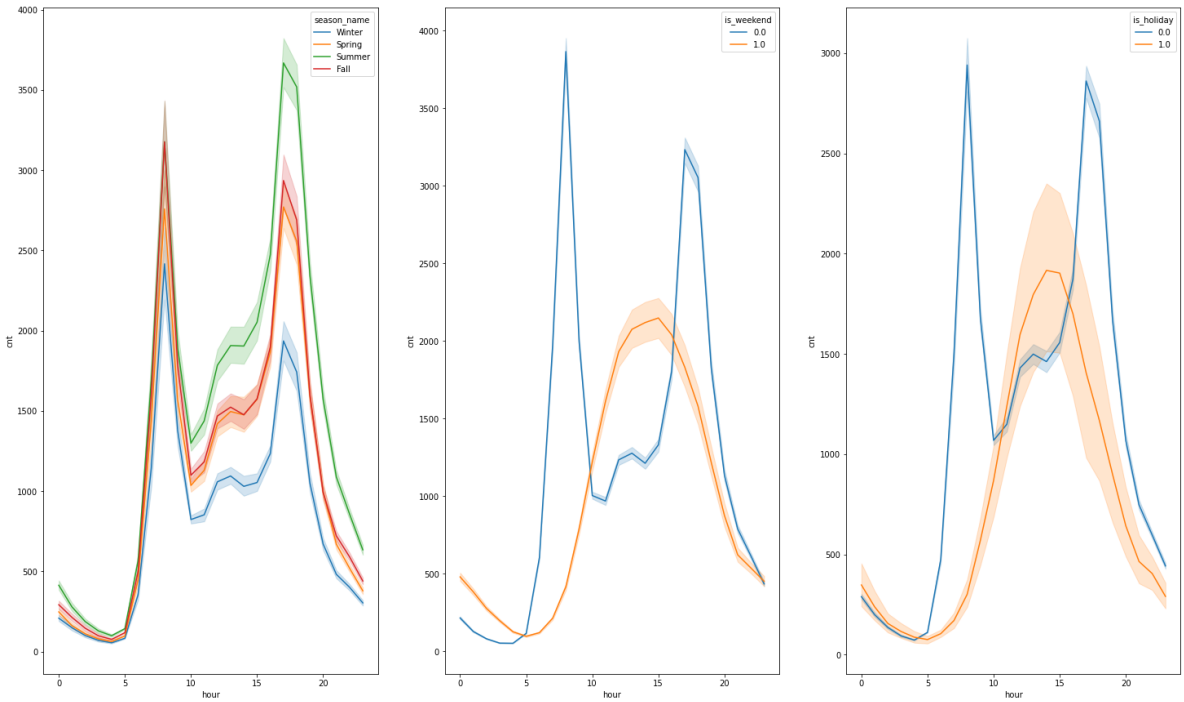
Out[33]:

timestamp	cnt	t1	t2	hum	wind_speed	weather_code	is_holiday	is_weekend	season	s
2015-01-04 00:00:00	182	3.0	2.0	93.0	6.0	3.0	0.0	1.0	3.0	
2015-01-04 01:00:00	138	3.0	2.5	93.0	5.0	1.0	0.0	1.0	3.0	
2015-01-04 02:00:00	134	2.5	2.5	96.5	0.0	1.0	0.0	1.0	3.0	
2015-01-04 03:00:00	72	2.0	2.0	100.0	0.0	1.0	0.0	1.0	3.0	
2015-01-04 04:00:00	47	2.0	0.0	93.0	6.5	1.0	0.0	1.0	3.0	

In [34]:

```
fig, ax = plt.subplots(1,3 , figsize = (25,15))

sns.lineplot(x = df.hour , y = df.cnt , hue = df.season_name , ax = ax[0]);
sns.lineplot(x = df.hour , y = df.cnt , hue = df["is_weekend"] , ax = ax[1]);
sns.lineplot(x = df.hour , y = df.cnt , hue = df["is_holiday"] ,ax = ax[2]);
```



In [35]:

```
#fig, ax = plt.subplots(1,3,figsize = (20,14))

#sns.lineplot( x = df["is_holiday"] , y = df["cnt"] , ax = ax[0]);
#sns.lineplot( x = df["is_weekend"] , y = df["cnt"] , ax = ax[1]);
#sns.lineplot( x = df["season_name"] , y = df["cnt"], ax = ax[2]);
```

8. Plot bike shares by day of week.
   - You may want to see whether it is a holiday or not

In [36]:

```
df.head()
```

Out[36]:

timestamp	cnt	t1	t2	hum	wind_speed	weather_code	is_holiday	is_weekend	season	s
2015-01-04 00:00:00	182	3.0	2.0	93.0	6.0	3.0	0.0	1.0	3.0	
2015-01-04 01:00:00	138	3.0	2.5	93.0	5.0	1.0	0.0	1.0	3.0	
2015-01-04 02:00:00	134	2.5	2.5	96.5	0.0	1.0	0.0	1.0	3.0	
2015-01-04 03:00:00	72	2.0	2.0	100.0	0.0	1.0	0.0	1.0	3.0	
2015-01-04 04:00:00	47	2.0	0.0	93.0	6.5	1.0	0.0	1.0	3.0	

In [37]:

```
df.groupby("is_holiday")["cnt"].count()
```

Out[37]:

```
is_holiday
0.0    17030
1.0      384
Name: cnt, dtype: int64
```

In [38]:

```
df.groupby("is_holiday")["cnt"].mean()
```

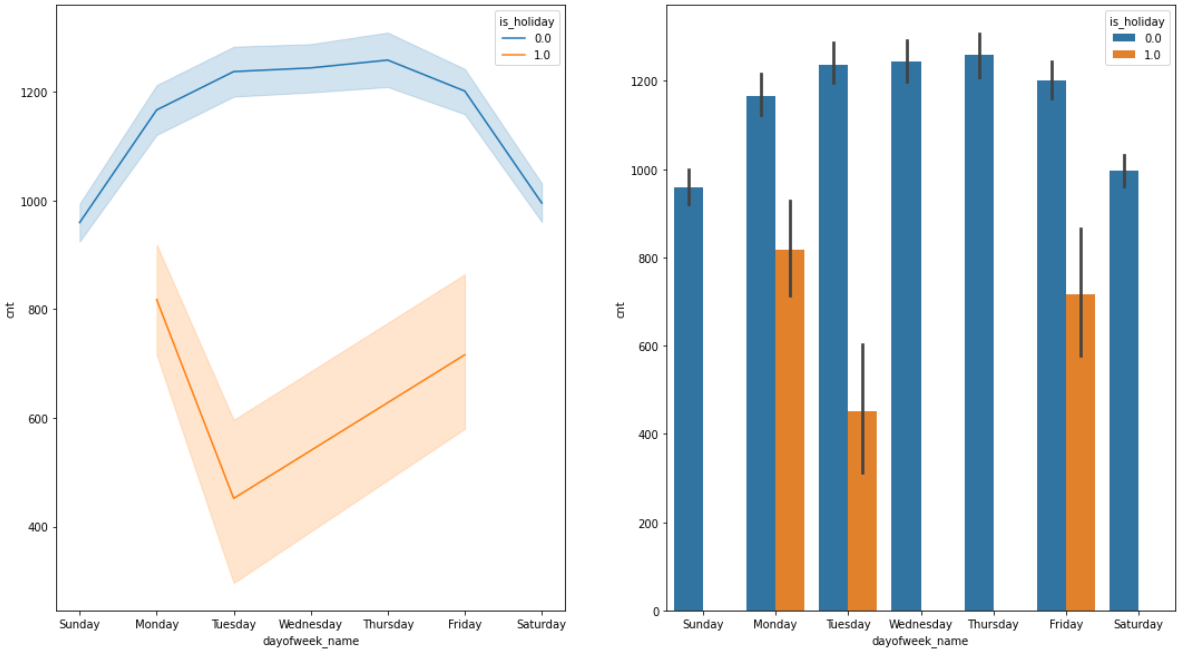
Out[38]:

```
is_holiday
0.0    1151.525191
1.0     769.526042
Name: cnt, dtype: float64
```

In [39]:

```
fig, ax = plt.subplots(1,2,figsize = (18,10))

sns.lineplot(x= df["dayofweek_name"] , y= df["cnt"] , hue = df["is_holiday"] , ax = ax[0]);
sns.barplot(x = "dayofweek_name" , y = "cnt" , data = df , hue = "is_holiday" , ax = ax[1])
```

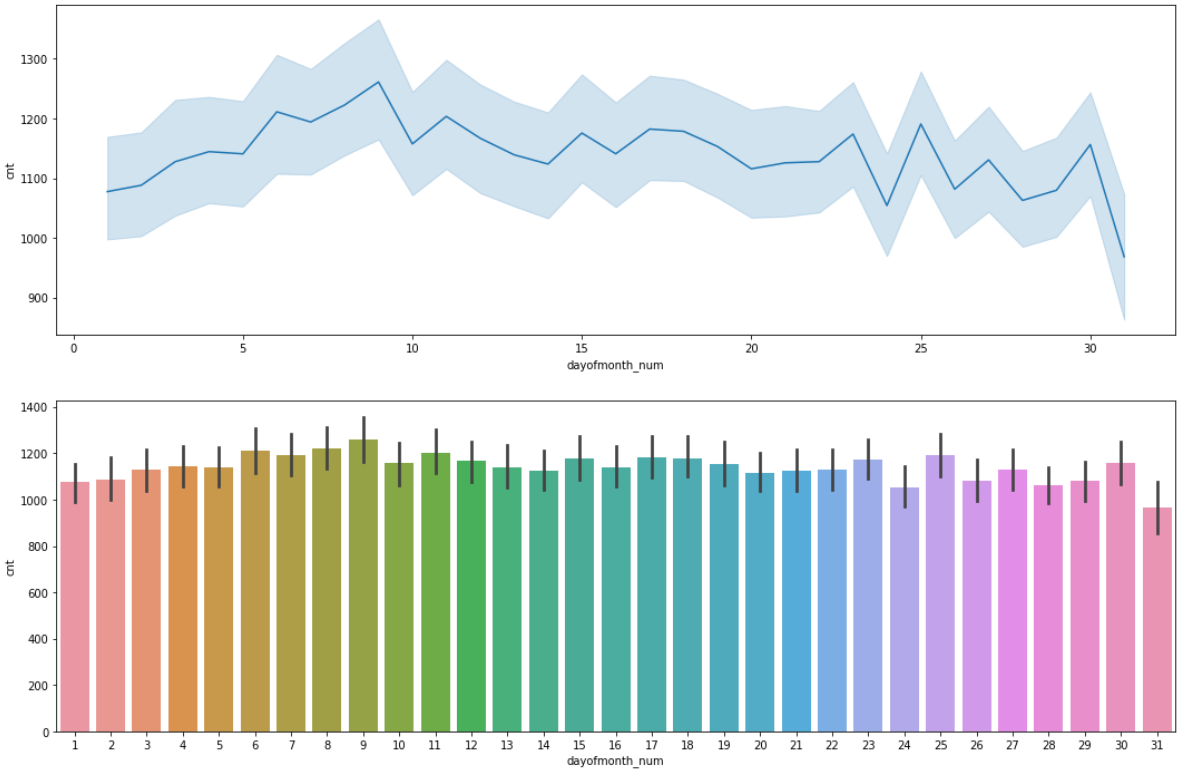


9. Plot bike shares by day of month

In [40]:

```
fig, ax = plt.subplots(2,1,figsize = (18,12))


sns.lineplot(x = df["dayofmonth_num"] , y = df["cnt"] , ax = ax[0]);
sns.barplot(x = df["dayofmonth_num"] , y = df.cnt, ax = ax[1] ) ;
```



10. Plot bike shares by year
    - Plot bike shares on holidays by seasons

In [41]:

```
df[df["is_holiday"] == 1]
```

Out[41]:

timestamp	cnt	t1	t2	hum	wind_speed	weather_code	is_holiday	is_weekend	season	s
2015-04-03 00:00:00	279	8.0	6.0	82.0	10.0	7.0	1.0	0.0	0.0	
2015-04-03 01:00:00	174	8.0	5.5	79.0	14.0	7.0	1.0	0.0	0.0	
2015-04-03 02:00:00	89	7.5	5.5	84.5	12.0	7.0	1.0	0.0	0.0	
2015-04-03 03:00:00	61	7.0	5.0	87.0	11.0	7.0	1.0	0.0	0.0	
2015-04-03 04:00:00	46	7.0	6.0	93.0	6.0	7.0	1.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...
2017-01-02 19:00:00	433	3.0	0.0	81.0	11.0	1.0	1.0	0.0	3.0	
2017-01-02 20:00:00	334	3.0	0.0	75.0	13.0	1.0	1.0	0.0	3.0	
2017-01-02 21:00:00	233	2.5	-0.5	78.0	11.0	1.0	1.0	0.0	3.0	
2017-01-02 22:00:00	201	2.0	-1.0	81.0	10.0	1.0	1.0	0.0	3.0	
2017-01-02 23:00:00	145	1.0	-2.0	93.0	10.0	1.0	1.0	0.0	3.0	

384 rows × 17 columns

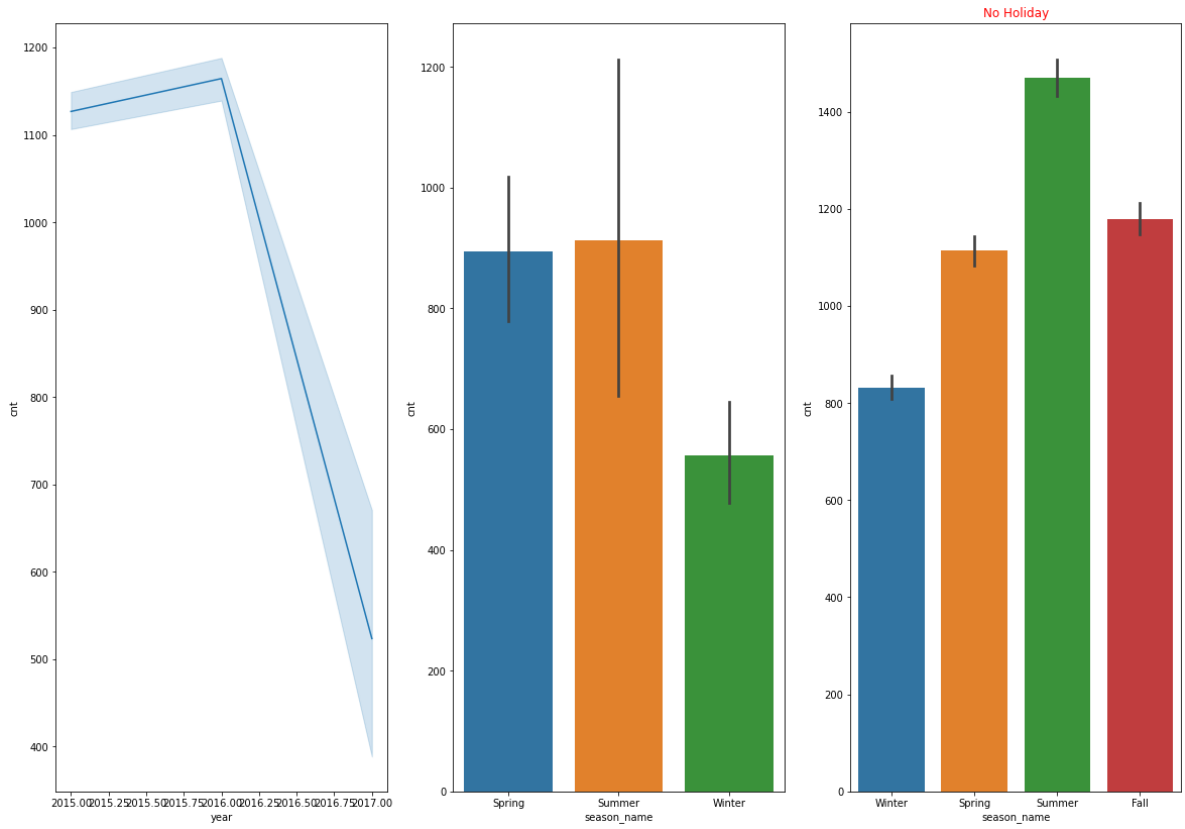
In [42]:

```
from turtle import title


fig, ax = plt.subplots(1,3, figsize = (20,14))

sns.lineplot(x = df.year , y = df.cnt , ax = ax[0]) ;
sns.barplot(x = df[df["is_holiday"] == 1]["season_name"] , y = df[df["is_holiday"] == 1].cn
plt.title("No Holiday" , color="red")

sns.barplot(x = df[df["is_holiday"] == 0]["season_name"] , y = df[df["is_holiday"] == 0].cn
```



11. Visualize the distribution of bike shares by weekday/weekend with piechart and barplot

In [43]:

```
df.groupby("is_weekend").cnt.mean()
```

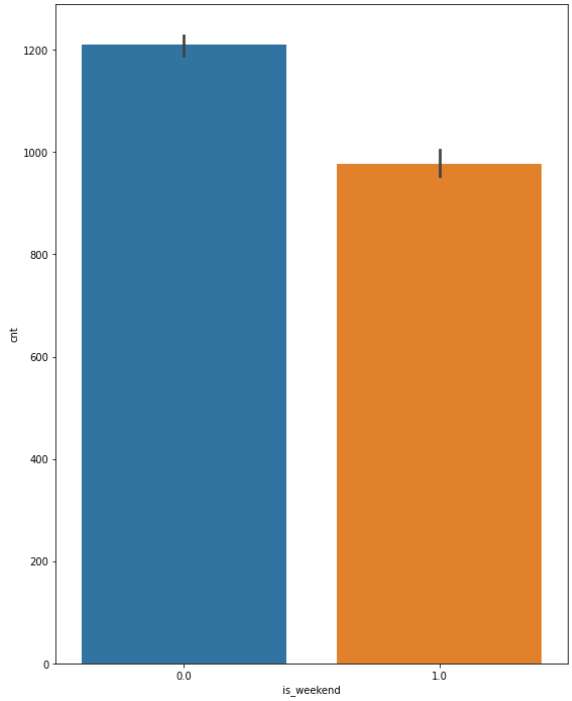
Out[43]:

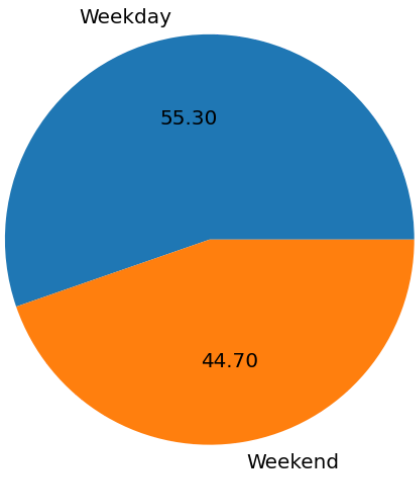
```
is_weekend
0.0    1209.274831
1.0     977.415694
Name: cnt, dtype: float64
```

In [44]:

```
fig, ax = plt.subplots(1,2, figsize = (20,12))

sns.barplot(x = df["is_weekend"] , y = df["cnt"] , ax = ax[0]);
plt.pie(x = df.groupby("is_weekend").cnt.mean(), labels = ["Weekday" , "Weekend"] , autopct
```





12. Plot the distribution of weather code by seasons

In [45]:

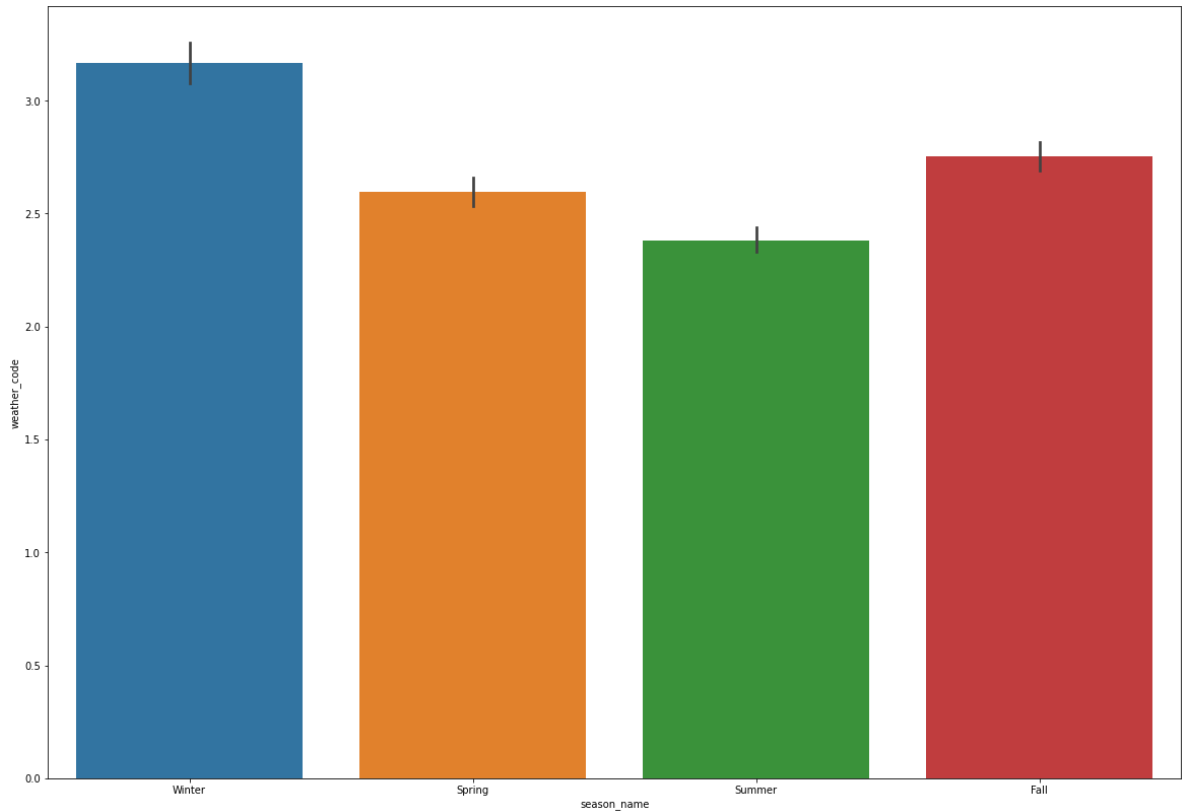
```
df.head()
```

Out[45]:

timestamp	cnt	t1	t2	hum	wind_speed	weather_code	is_holiday	is_weekend	season	s
2015-01-04 00:00:00	182	3.0	2.0	93.0	6.0	3.0	0.0	1.0	3.0	
2015-01-04 01:00:00	138	3.0	2.5	93.0	5.0	1.0	0.0	1.0	3.0	
2015-01-04 02:00:00	134	2.5	2.5	96.5	0.0	1.0	0.0	1.0	3.0	
2015-01-04 03:00:00	72	2.0	2.0	100.0	0.0	1.0	0.0	1.0	3.0	
2015-01-04 04:00:00	47	2.0	0.0	93.0	6.5	1.0	0.0	1.0	3.0	

In [46]:

```
plt.figure(figsize=(20,14))
sns.barplot(x = df["season_name"] , y = df.weather_code ) ;
```



In [47]:

```
# Additional

df.weather_code_mapped = df.weather_code.map({
    1:'Clear',
    2:'Few clouds',
    3:'Broken clouds',
    4:'Cloudy',
    7:'Light Rain shower',
    10:'Thunderstorm',
    26:'Snowfall',
    94:'Freezing Fog'
})
```

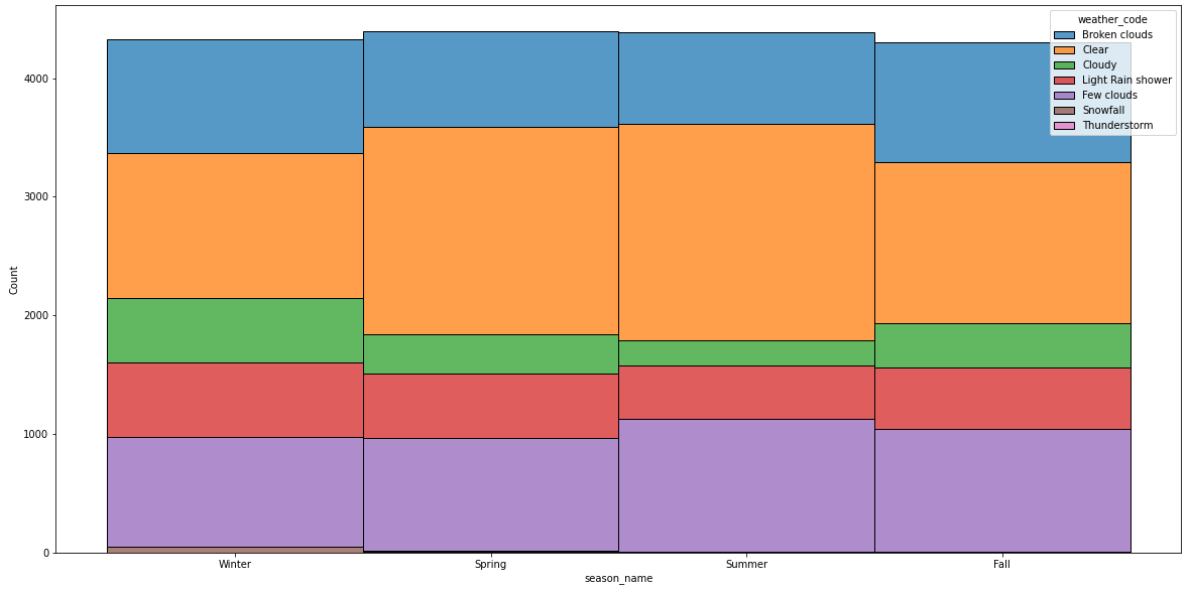
```
C:\Users\EmincanY\AppData\Local\Temp\ipykernel_12080\3093734476.py:3: UserWa
rning: Pandas doesn't allow columns to be created via a new attribute name -
see https://pandas.pydata.org/pandas-docs/stable/indexing.html#attribute-acc
ess (https://pandas.pydata.org/pandas-docs/stable/indexing.html#attribute-ac
cess)
  df.weather_code_mapped = df.weather_code.map({
```

In [48]:

```
# Additional

plt.figure(figsize=(20, 10))

sns.histplot(binwidth=0.5, x=df.season_name, hue=df.weather_code_mapped, data=df, stat="cou
```



Conclusions

- There was totally clear data. No null and no duplicate rows.
- Too many days weather is clear. Freezing Fog never happened. Snowfall sometimes happened. Rain with thunderstorms happened the least.
- As we expect the spring and summer have too many clear days. In the fall and winter have got more Scattered and Broken clouds.

- Bike shares have positive relations with Temperature, Feeling Temperature, Hour and Wind Speed.
- Bike shares have negative relations with humidity, weather status and season.
- As a normally too strong positive correlation between temperature and feeling temperature.
- As a normally positive correlation between Temperature/Feeling Temperature and months.
- As a normally , positive correlation between the hum and seasons.
- Humidity and weather status have positive realations.
- Humidity and wind speed have negative realations.
- Bike shares is declining but peaked in the summer of 2015
- As a normally , In summers more bike shares happens.
- In all seasons. Early 8-9a.m and 17-18p.m too many bikes shares.
- In weekdays also same eary 8-9a.m and 17.18p.m.
- But in the weekends too many bikes shares around at 14-15p.m and have normal distribution.
- On a holiday, less bike is shared than normal.
- In a normal day(not holiday) summer have too many bike shares. Winter have lest.
- During the holidays, not happened any bike shares in Fall. (This is strange) Winter still lest. Summer and spring almost the same bike shares.
- Seasons have normal weather status. Hard to beautiful ==> Winter>Fall>Spring>Summer
- That's all.

Thanks for Attending!..