EDA & Data Visualization Project

Yeniliklerden ilk siz haberdar olmak istiyorsanız lütfen bizi takip etmeyi unutmayın

YouTube | Instagram | Facebook | Telegram | Watsapp | Linkedin |

WELCOME!

Bike Demand Visualization Project

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.

About Dataset:

The bike-sharing system is a new generation of traditional bike rentals where the whole process from membership, rental and return back has become automatic. Through these systems, the user is able to easily rent a bike from a particular position and return back to another position. where the whole process is from membership, rental, and return.

Currently, there are about over ***500 bike-sharing programs around the world*** which are composed of over ***500 thousand bicycles.*** Today, there exists great interest in these systems due to their important role in traffic, environmental, and health issues.

The bike-sharing rental process is highly correlated to the environmental and seasonal settings. For instance, weather
conditions, precipitation, day of the week, season, hour of the day, etc. can affect the rental behaviors.

There have been many online sources regarding bike-sharing datasets one of which is at the UCI archive

The dataset is related to the two-year historical log corresponding to the years, between January 2015 and January 2017.

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.

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

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

In [3]:
```
import matplotlib.ticker as ticker
```

2. Read Dataset

In [4]:
```
bike_shares = pd.read_csv("store_sharing.csv")
```

In [5]:
```
bike_shares.head()
```

Out[5]:

	timestamp	cnt	t1	t2	hum	wind_speed	weather_code	is_holiday	is_weekend	seas
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	

In [6]:
```
bike_shares.shape
```

Out[6]:
```
(17414, 10)
```

In [7]:
```
bike_shares.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17414 entries, 0 to 17413
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   timestamp     17414 non-null  object 
 1   cnt           17414 non-null  int64  
 2   t1            17414 non-null  float64
 3   t2            17414 non-null  float64
 4   hum           17414 non-null  float64
 5   wind_speed    17414 non-null  float64
 6   weather_code  17414 non-null  float64
 7   is_holiday    17414 non-null  float64
 8   is_weekend    17414 non-null  float64
 9   season        17414 non-null  float64
dtypes: float64(8), int64(1), object(1)
memory usage: 1.3+ MB
```

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

In [8]:
```
bike_shares.isnull().sum()
```

Out[8]:
```
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 [9]:
```
bike_shares.duplicated()
```

Out[9]:
```
0        False
1        False
2        False
3        False
4        False
         ...  
17409    False
17410    False
17411    False
17412    False
17413    False
Length: 17414, dtype: bool
```

In [10]:
```
bike_shares.duplicated().sum()
```

Out[10]:
```
0
```

#################### ***Fortunetly there isn't any dublicated and missing values.***

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

In [11]:
```
bike_shares.describe().T
```

Out[11]:

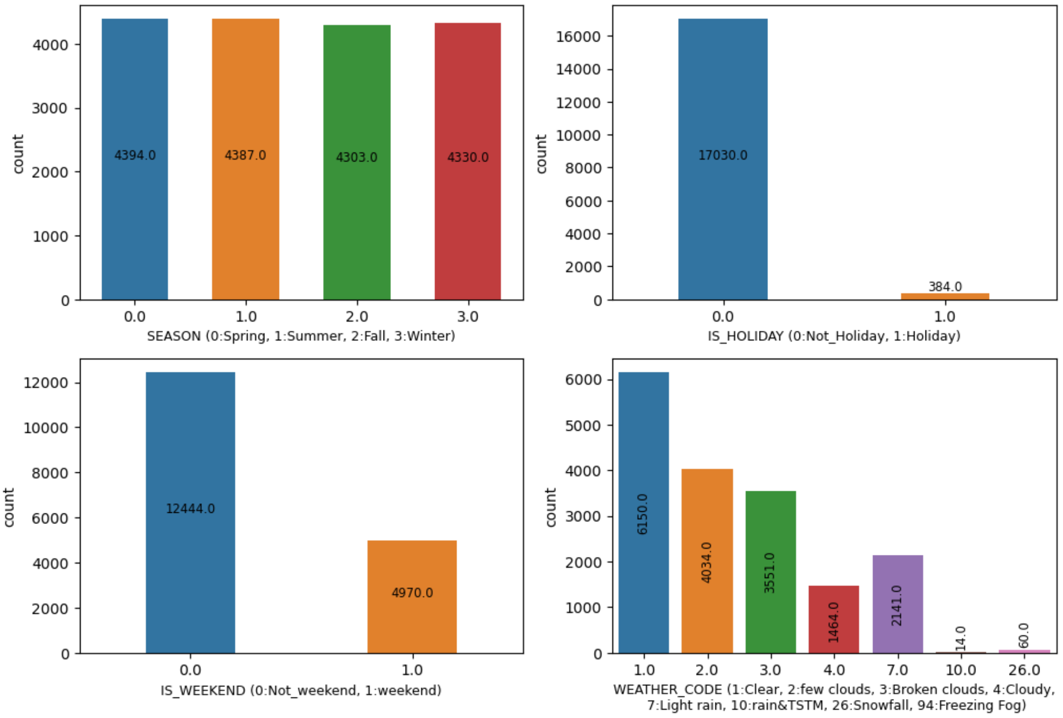
	count	mean	std	min	25%	50%	75%	max
cnt	17414.0	1143.101642	1085.108068	0.0	257.0	844.0	1671.75	7860.0
t1	17414.0	12.468091	5.571818	-1.5	8.0	12.5	16.00	34.0
t2	17414.0	11.520836	6.615145	-6.0	6.0	12.5	16.00	34.0
hum	17414.0	72.324954	14.313186	20.5	63.0	74.5	83.00	100.0
wind_speed	17414.0	15.913063	7.894570	0.0	10.0	15.0	20.50	56.5
weather_code	17414.0	2.722752	2.341163	1.0	1.0	2.0	3.00	26.0
is_holiday	17414.0	0.022051	0.146854	0.0	0.0	0.0	0.00	1.0
is_weekend	17414.0	0.285403	0.451619	0.0	0.0	0.0	1.00	1.0
season	17414.0	1.492075	1.118911	0.0	0.0	1.0	2.00	3.0

In [100…
```
sns.color_palette()
```

Out[100]:

In [11]:
```
fig,ax=plt.subplots(2,2,figsize=(12,8))

sns.countplot(data=bike_shares,x="season",ax=ax[0,0],width=0.6)
ax[0,0].set_xlabel("SEASON (0:Spring, 1:Summer, 2:Fall, 3:Winter)",fontsiz

for p in ax[0,0].patches:

    ax[0,0].annotate(p.get_height(),((p.get_x() + p.get_width()/2) ,
            (p.get_height()/2)),
            ha="center",fontsize="small")
```

```
sns.countplot(data=bike_shares,x="is_holiday",ax=ax[0,1],width=0.4)
ax[0,1].set_xlabel("IS_HOLIDAY (0:Not_Holiday, 1:Holiday)",fontsize=9)

for p in ax[0,1].patches:
    if p.get_height() > 500 :
        ax[0,1].annotate(p.get_height(),((p.get_x() + p.get_width()/2) ,
            (p.get_height()/2)),
            ha="center",fontsize="small")
    elif  p.get_height() != 0 :
        ax[0,1].annotate(p.get_height(),((p.get_x() + p.get_width()/2) ,
            (p.get_height()+150)),
            ha="center",fontsize="small")


sns.countplot(data=bike_shares,x="is_weekend",width=0.4,ax=ax[1,0])
ax[1,0].set_xlabel("IS_WEEKEND (0:Not_weekend, 1:weekend)",fontsize=9)

for p in ax[1,0].patches:
    if p.get_height() > 0 :
        ax[1,0].annotate(p.get_height(),((p.get_x() + p.get_width()/2) ,
            (p.get_height()/2)),
            ha="center",fontsize="small")

sns.countplot(data=bike_shares,x="weather_code",ax=ax[1,1])
ax[1,1].set_xlabel("WEATHER_CODE (1:Clear, 2:few clouds, 3:Broken clouds, 
                  fontsize=9)

for p in ax[1,1].patches:
    if p.get_height() > 1000 :
        ax[1,1].annotate(p.get_height(),((p.get_x() + p.get_width()/2) ,
            (p.get_height()/2)),
            ha="center",va="center",rotation="vertical",fontsize="small",)
    elif p.get_height() !=0 :
        ax[1,1].annotate(p.get_height(),((p.get_x() + p.get_width()/2) ,
            (p.get_height()+350)),
            ha="center",va="center",rotation="vertical",fontsize="small",)

plt.tight_layout
plt.show;
```



- Although there is not much seasonal variation, there is a higher demand in spring and summer compared to others.
- The demand is higher on working days instead of holidays and weekends, which leads to the interpretation that people prefer this method of transport while commuting to work,
- Generally and naturally, clear weather conditions were preferred, but it seems that there is more demand in unusual light rainy weather than in cloudy weather,

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

In [12]:
```
bike_shares.dtypes
```

Out[12]:
```
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 [6]:
```
bike_shares.timestamp=pd.to_datetime(bike_shares.timestamp,errors="coerce
bike_shares.timestamp
```

Out[6]:
```
0       2015-01-04 00:00:00
1       2015-01-04 01:00:00
2       2015-01-04 02:00:00
3       2015-01-04 03:00:00
4       2015-01-04 04:00:00
                ...
17409   2017-01-03 19:00:00
17410   2017-01-03 20:00:00
17411   2017-01-03 21:00:00
17412   2017-01-03 22:00:00
17413   2017-01-03 23:00:00
Name: timestamp, Length: 17414, dtype: datetime64[ns]
```

6. Make feature engineering. Extract new columns (day of the week, day of the month, hour, month, season, year etc.)

In [7]:
```
bike_shares["day_of_the_week"] = bike_shares["timestamp"].dt.day_name()
bike_shares["day_of_the_month"] = bike_shares["timestamp"].dt.day
bike_shares["hour_of_the_day"] = bike_shares["timestamp"].dt.strftime("%
bike_shares["month"] = bike_shares["timestamp"].dt.month_name()
bike_shares["year"] = bike_shares["timestamp"].dt.year
```

In [8]:
```
bike_shares["year_month"]=bike_shares["timestamp"].dt.strftime("%Y-%m")
```

In [9]:
```
bike_shares.to_csv("bike_shares.csv")
```

In [10]:
```
bike_shares_new=pd.read_csv("bike_shares.csv",index_col="timestamp")
```

In [11]:
```
bike_shares_new.drop("Unnamed: 0",inplace=True,axis=1)
```

In [12]:
```
bike_shares_new.sample(5)
```

Out[12]:

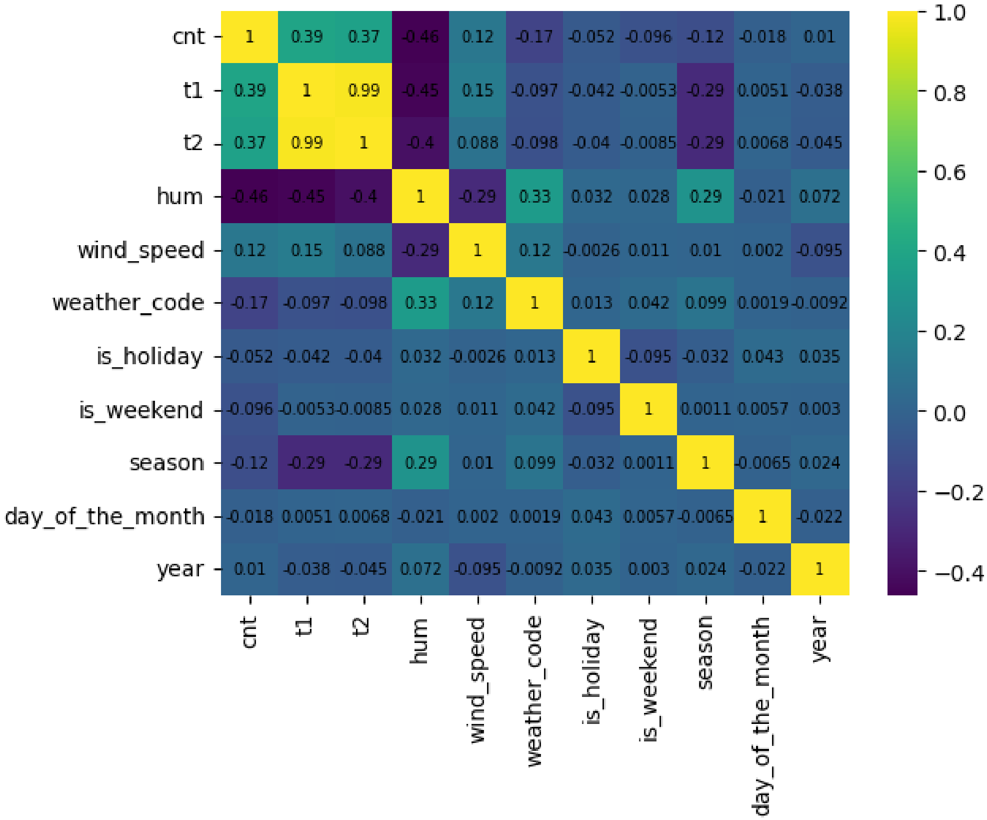
	cnt	t1	t2	hum	wind_speed	weather_code	is_holiday	is_weekenc
timestamp								
2016-05-01 03:00:00	154	6.0	5.5	76.0	5.0	1.0	0.0	1.(
2015-08-04 16:00:00	2297	21.5	21.0	45.0	25.0	3.0	0.0	0.(
2015-05-31 20:00:00	821	13.5	13.5	42.5	27.5	1.0	0.0	1.(
2016-11-30 18:00:00	3092	5.0	2.0	63.5	15.0	1.0	0.0	0.(
2016-06-20 21:00:00	845	20.0	20.0	64.0	18.5	1.0	0.0	0.(

7. Visualize the correlation with a heatmap

In [20]:
```
sns.heatmap(data=bike_shares_new.corr(),annot=True,annot_kws={"size":7
```

Out[20]:

```
<Axes: >
```



- t1 (actual temperature) and t2 (perceived temperature) have the highest positive correlation with the number of bike shares.
  - Hum (humidity) has the highest but negative correlation, so humidity is the most important criterion that negatively affects demand.
  - The t1 and t2 correlations are in the same direction and close to each other, so one of them can be preferred.

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

In [21]:
```
corr_matrix=bike_shares_new.corr()
corr_matrix
```

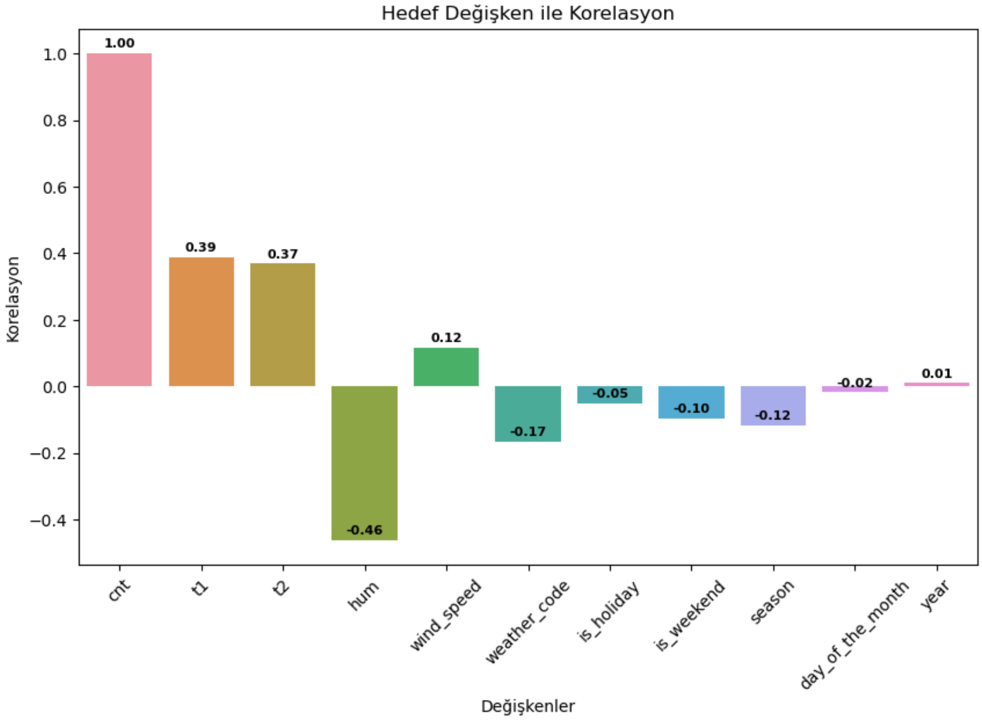
Out[21]:

	cnt	t1	t2	hum	wind_speed	weathe
cnt	1.000000	0.388798	0.369035	-0.462901	0.116295	-0.
t1	0.388798	1.000000	0.988344	-0.447781	0.145471	-0.
t2	0.369035	0.988344	1.000000	-0.403495	0.088409	-0.
hum	-0.462901	-0.447781	-0.403495	1.000000	-0.287789	0.
wind_speed	0.116295	0.145471	0.088409	-0.287789	1.000000	0.
weather_code	-0.166633	-0.097114	-0.098385	0.334750	0.124803	1.
is_holiday	-0.051698	-0.042233	-0.040051	0.032068	-0.002606	0.
is_weekend	-0.096499	-0.005342	-0.008510	0.028098	0.011479	0.
season	-0.116180	-0.285851	-0.285900	0.290381	0.010305	0.
day_of_the_month	-0.017887	0.005072	0.006791	-0.020868	0.002040	0.
year	0.010046	-0.037959	-0.044972	0.072443	-0.094739	-0.

- We prefer to visualise the correlation between our target variable (cnt), i.e. bike shares, and other variables as follows

In [22]:

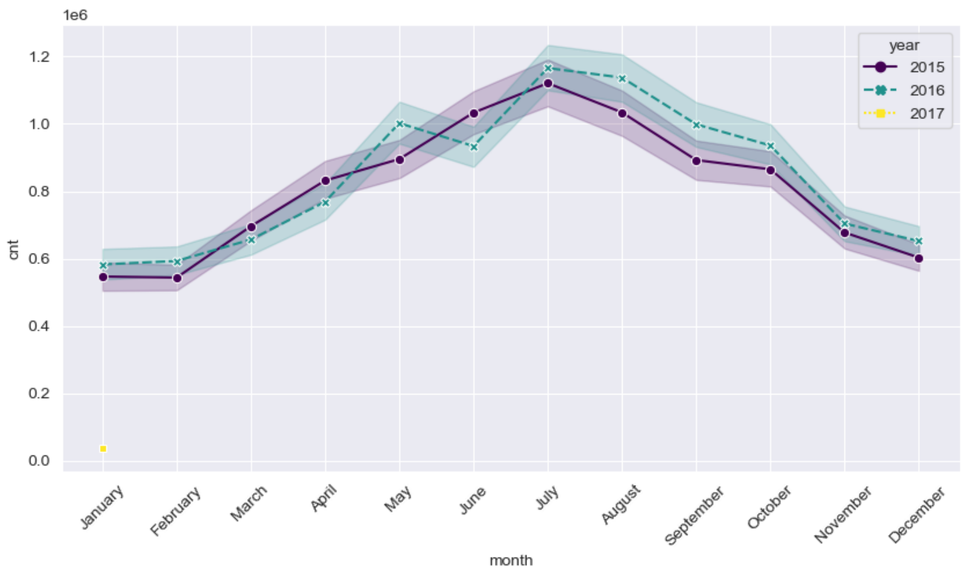
```
plt.figure(figsize=(10, 6))

bar_plot=sns.barplot(x=corr_matrix["cnt"].index, y=corr_matrix["cnt"]
plt.xticks(rotation=45)
plt.title('Hedef Değişken ile Korelasyon')
plt.xlabel('Değişkenler')
plt.ylabel('Korelasyon')

for index, value in enumerate(corr_matrix["cnt"]):

    bar_plot.text(index, value + 0.01, f'{value:.2f}', ha='center', v

plt.show()
```



- t1 (actual temperature) and t2 (perceived temperature) have the highest positive correlation with the number of bike shares.
  - Hum (humidity) has the highest but negative correlation, so humidity is the most important criterion that negatively affects demand.
  - The t1 and t2 correlations are in the same direction and close to each other, so one of them can be preferred.

9. Plot bike shares over time use lineplot.

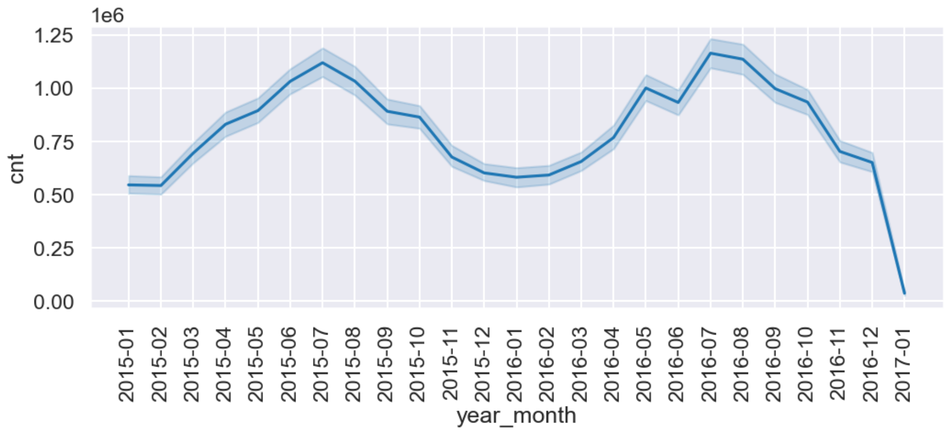
In [23]:
```
plt.figure(figsize=(10, 5))
sns.set_style("darkgrid")
sns.lineplot(data=bike_shares_new,x="month",y="cnt",hue="year",styl

plt.xticks(rotation=45);
```



- As we can see from the graph, it is clear that the demand (bicycle sharing) increases with the spring months, peaks in July and then decreases.
- Since the data for 2017 is only for January, we cannot see the graph very much.

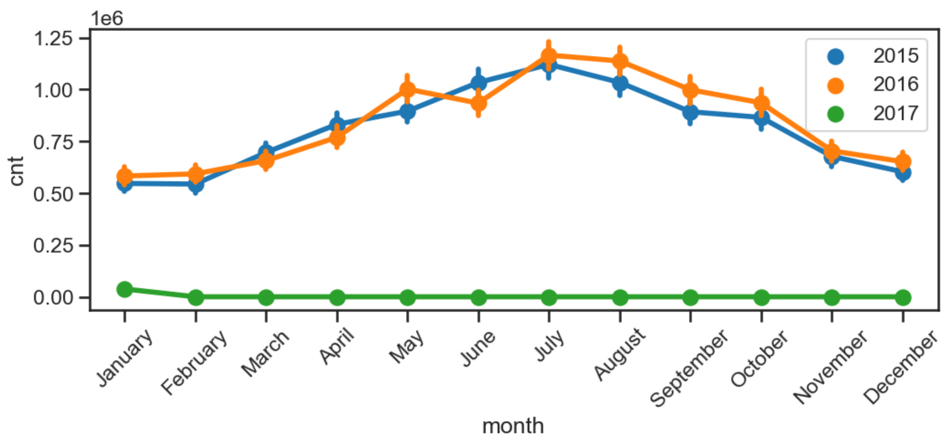
10. Plot bike shares by months and year_of_month (use lineplot, pointplot, barplot).

In [51]:
```
plt.figure(figsize=(12,4))
sns.lineplot(data=bike_shares_new, x="year_month", y="cnt", estima
plt.xticks(rotation=90);
plt.yscale("linear")
```

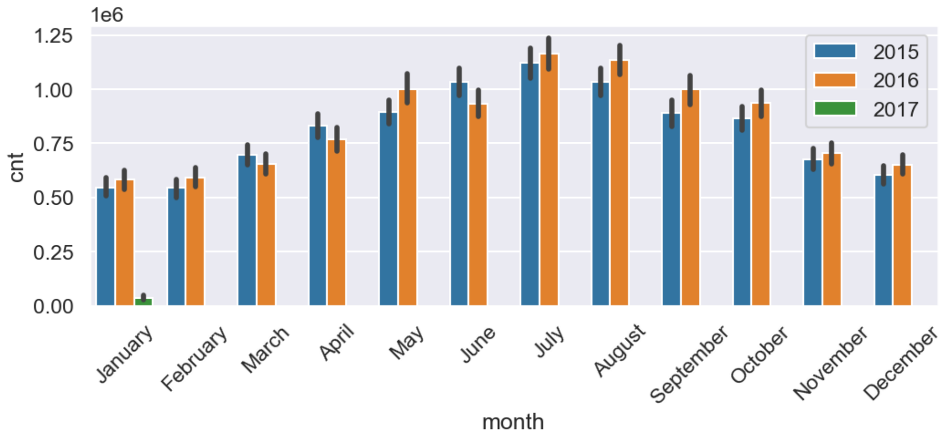


- Although we can see the pattern we saw in the previous graph here, we can say that there is a sharp decline in January 2017.

In [61]:
```
plt.figure(figsize=(12,4))

sns.set_style("ticks")

sns.pointplot(data=bike_shares_new, x="month", y="cnt", hue="year"

plt.legend(loc="upper right")

plt.xticks(rotation=45)
plt.show();
```

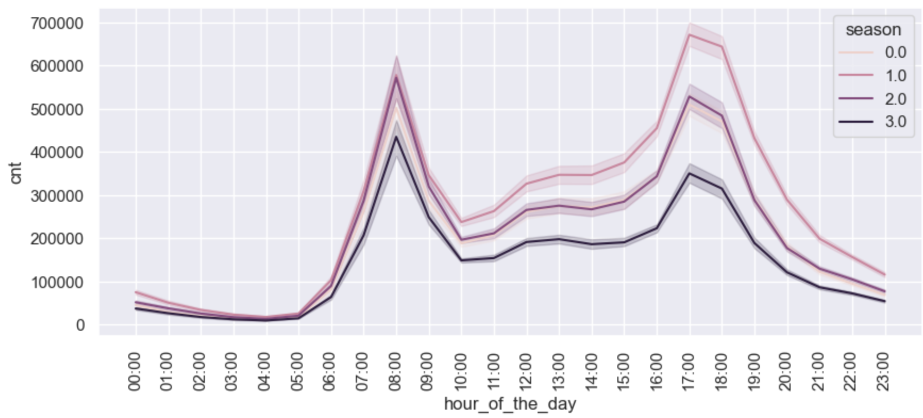


In [56]:
```
plt.figure(figsize=(12,4))

sns.barplot(data=bike_shares_new, x="month", y="cnt", hue="year",

plt.xticks(rotation=45)

plt.legend(loc="upper right")

plt.show();
```



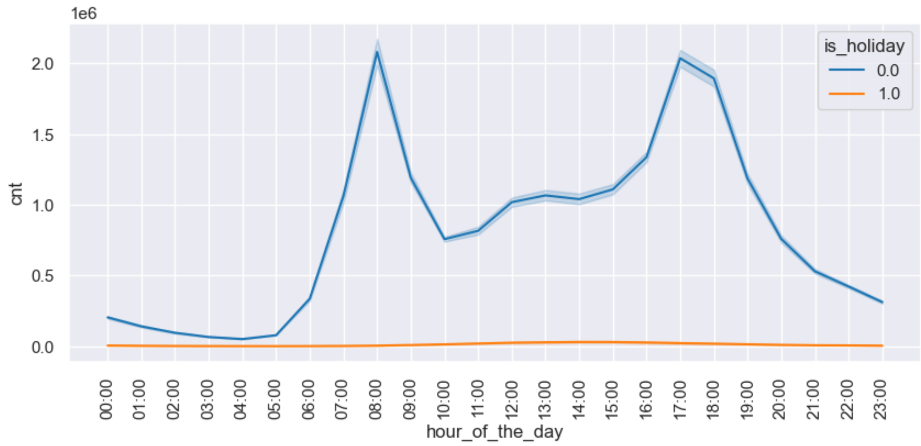
11. Plot bike shares by hours on (holidays, weekend, season).

In [78]:
```
plt.figure(figsize=(10,4))
sns.set_style("darkgrid")
sns.lineplot(data=bike_shares_new,x="hour_of_the_day",y="cnt",hue
plt.xticks(rotation=90);
```



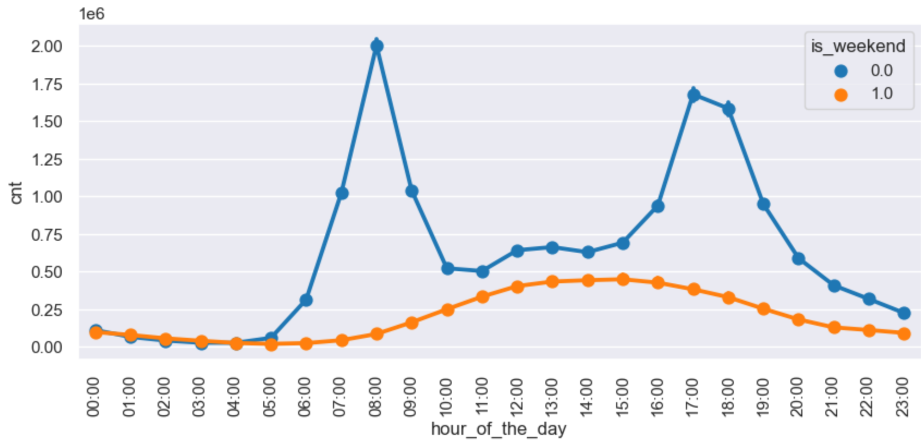
- It is observed that the daily change does not change much seasonally and follows similar patterns, but the numerical effect of the seasonal change shifts the graph.

In [77]:
```
sns.set_style("darkgrid")
plt.figure(figsize=(10,4))
sns.lineplot(data=bike_shares_new,x="hour_of_the_day",y="cnt",hue
plt.xticks(rotation=90);
```



- Although it is not clearly seen that there is no demand on holidays, it is also seen that it does not show any change during the day.

In [80]:
```
sns.set_style("darkgrid")
plt.figure(figsize=(10,4))
sns.pointplot(data=bike_shares_new,x="hour_of_the_day",y="cnt",hu
plt.xticks(rotation=90);
```



- Although weekends show changes during the day, it is also seen through this graph that the main demand is on working days.
- On working days, it is seen that there is a fluctuation at the beginning and end of working hours.

In [151…
```
bike_shares_new[(bike_shares_new["is_holiday"]==1) & (bike_share
```

Out[151]:

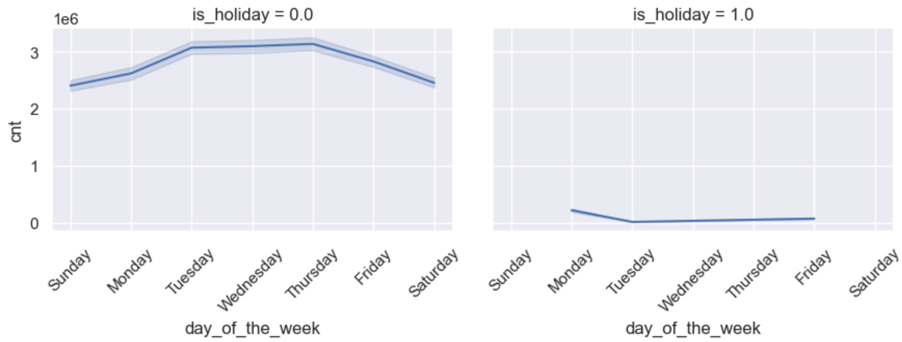
	cnt	t1	t2	hum	wind_speed	weather_code	is_holiday	is_weel
timestamp								

12. Plot bike shares by day of week.

- You may want to see whether it is a holiday or not

In [161…
```
plt.figure(figsize=(12,6))
g=sns.FacetGrid(data=bike_shares_new,col="is_holiday",aspect=1.
g.map(sns.lineplot, "day_of_the_week", "cnt",estimator=sum)

g.set_xticklabels(rotation=45);
```

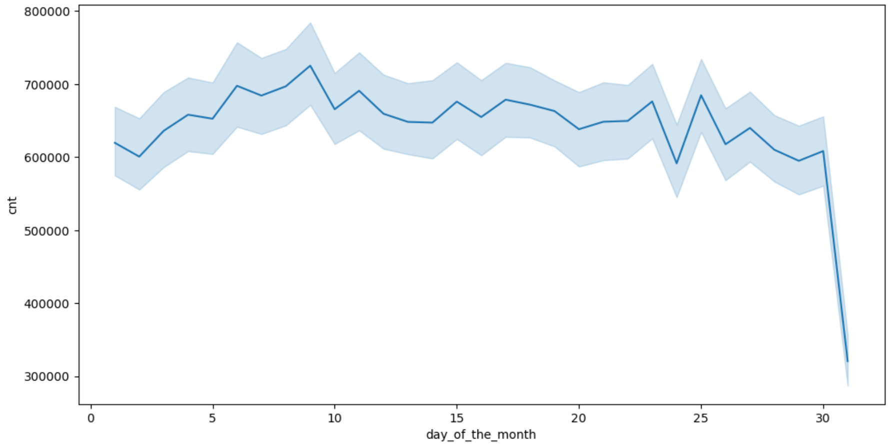
<Figure size 1200x600 with 0 Axes>



- During non-holiday periods, demand is high and approximately constant on Tuesdays, Wednesdays and Thursdays.
- On holiday days, demand is lost sharply.

13. Plot bike shares by day of month

In [15]:
```
plt.figure (figsize=(12,6))
sns.lineplot(data=bike_shares_new,x="day_of_the_month",y="cnt'
plt.show()
```

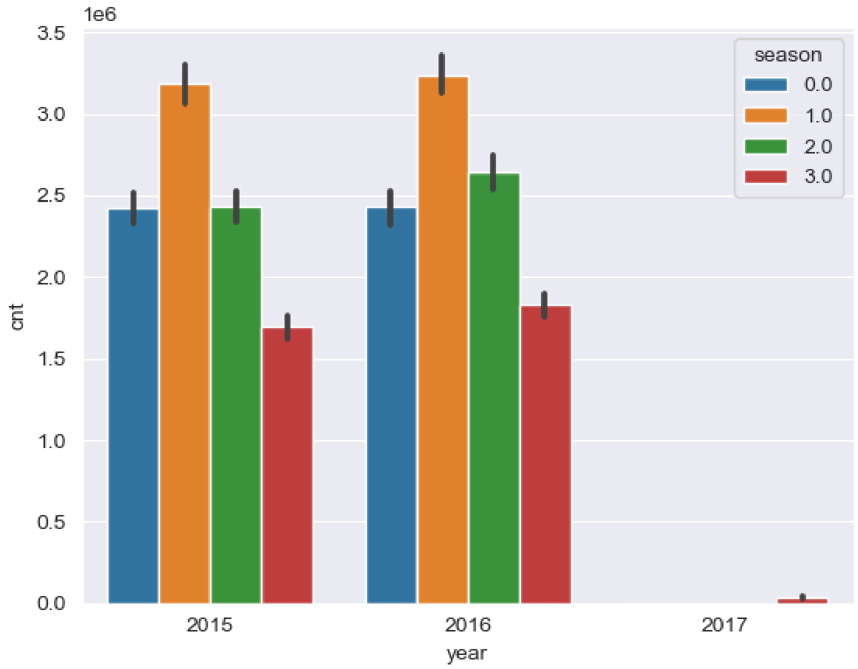


- At the end of the month, it is observed that the amount of demand decreased sharply.

14. Plot bike shares by year

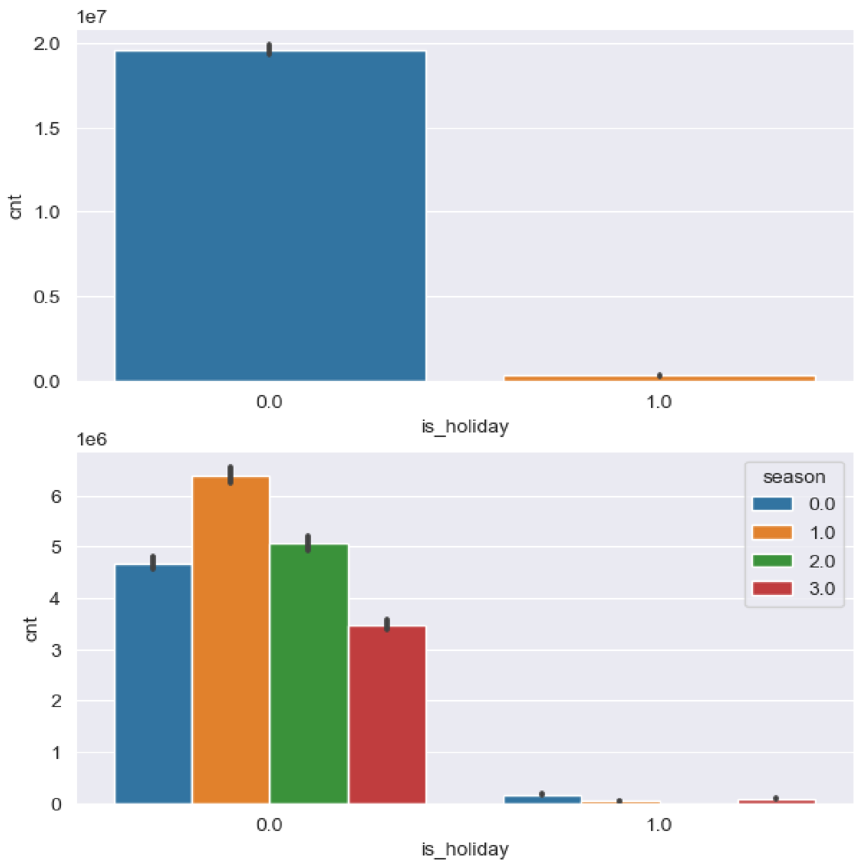
- Plot bike shares on holidays by seasons

In [48]:
```
sns.barplot(data=bike_shares_new, x="year", y="cnt", hue="se
```



- In the graph above, it is possible to see the seasonal change on a yearly basis

In [47]:
```
fig, ax = plt.subplots(2,1, figsize=(7,7))

sns.barplot(data=bike_shares_new, x = "is_holiday", y="cnt",
sns.barplot(data=bike_shares_new, x="is_holiday", y="cnt", h

plt.show();
```



Season : 0-spring ; 1-summer; 2-fall; 3-winter.

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

In [49]:
```
df_pie = bike_shares_new.groupby(['is_weekend'])["cnt"].sum
df_pie
```

Out[49]:

	is_weekend	cnt
0	0.0	15048216
1	1.0	4857756

In [54]:
```
df_pie["is_weekend"].replace({0:"weekday",1:"weekend"},inpl
```

In [55]:
```
df_pie
```

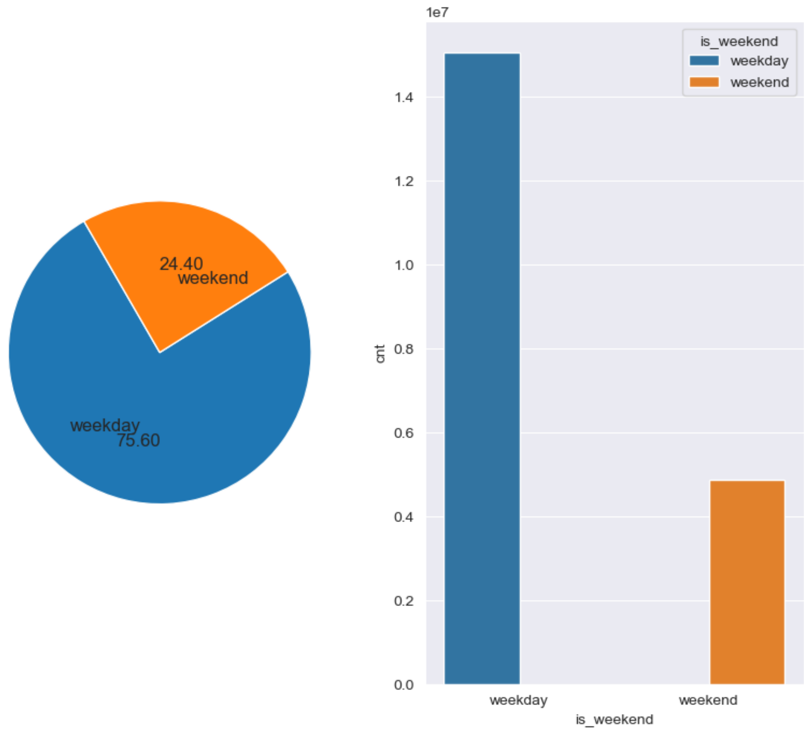
Out[55]:

	is_weekend	cnt
0	weekday	15048216
1	weekend	4857756

In [57]:
```
df_pie.dtypes
```

Out[57]:
```
is_weekend    object
cnt            int64
dtype: object
```

In [82]:
```
fig, ax = plt.subplots(1,2,figsize=(10, 8))

ax[0].pie(df_pie["cnt"], labels =df_pie["is_weekend"], labe
        startangle=120,textprops={"fontsize":12})
sns.barplot(x=df_pie["is_weekend"],y=df_pie["cnt"],data=df_

plt.show()
```



- We have seen the rates on a weekend and weekday basis

Conclusions

- As a result, it is seen that the demand for the transport method we define as a bicycle sharing system, especially for the most interesting and obvious insight for the examined city, is intense for commuting to work and is significantly affected by weather conditions.

EDA & Data Visualization Project

Yeniliklerden ilk siz haberdar olmak istiyorsanız lütfen bizi takip etmeyi unutmayın YouTube | Instagram | Facebook | Telegram | Watsapp | Linkedin |