Bike Demand Project

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

Store Sharing Project!

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.

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.

Sütun Açıklamaları

timestamp - verileri gruplandırmak için zaman verisi sütunu

cnt - yeni bisiklet paylaşımları sayısı

t1 - C cinsinden gerçek sıcaklık

t2 - C cinsinden hissedilen sıcaklık

hum - yüzde cinsinden nem oranı

wind_speed - km/h cinsinden rüzgar hızı

weather_code - hava durumu kategorisi

is_holiday - boolean alanı - 1 tatil / 0 tatil değil

is_weekend - boolean alanı - gün hafta sonuysa 1

season - kategori alanı meteorolojik mevsimler: 0-ilkbahar; 1-yaz; 2-sonbahar; 3-kış.

"weather_code" kategori açıklaması:

1 = Açık; çoğunlukla açık ancak pus/sis/sis parçaları/yakınlarda sis

2 = dağınık bulutlar / az bulut

3 = Parçalı bulutlar

4 = Bulutlu

7 = Yağmur/hafif Yağmur duşu/Hafif yağmur

10 = gök gürültülü fırtınalı yağmur

26 = kar yağışı

94 = Donan Sis

Import Libraries, Loading the Dataset and Initial Exploration

- Load the dataset, display first few rows, check the structure of the dataset.

- Inspect the data types and missing values using df.info()
- Get basic statistics for numerical columns with df.describe()

```
# ilgili kütüphaneleri yükleyiniz.

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('store_sharing.csv')

# ilk 5 satırı gösteriniz.

df.head()
```

```
             timestamp  cnt   t1   t2    hum  wind_speed  weather_code
\
0  2015-01-04 00:00:00  182  3.0  2.0   93.0         6.0           3.0

1  2015-01-04 01:00:00  138  3.0  2.5   93.0         5.0           1.0

2  2015-01-04 02:00:00  134  2.5  2.5   96.5         0.0           1.0

3  2015-01-04 03:00:00   72  2.0  2.0  100.0         0.0           1.0

4  2015-01-04 04:00:00   47  2.0  0.0   93.0         6.5           1.0


   is_holiday  is_weekend  season
0         0.0         1.0     3.0
1         0.0         1.0     3.0
2         0.0         1.0     3.0
3         0.0         1.0     3.0
4         0.0         1.0     3.0
```

```
# Verisetinin yapısına göz atınız.

df.info
```

```
<bound method DataFrame.info of                 timestamp   cnt   t1
t2    hum  wind_speed  weather_code  \
0      2015-01-04 00:00:00   182  3.0  2.0   93.0         6.0
3.0
1      2015-01-04 01:00:00   138  3.0  2.5   93.0         5.0
1.0
2      2015-01-04 02:00:00   134  2.5  2.5   96.5         0.0
1.0
3      2015-01-04 03:00:00    72  2.0  2.0  100.0         0.0
```

```
1.0
4      2015-01-04 04:00:00    47  2.0  0.0   93.0          6.5
1.0
...                    ...   ...  ...  ...    ...          ...
...
17409  2017-01-03 19:00:00  1042  5.0  1.0   81.0         19.0
3.0
17410  2017-01-03 20:00:00   541  5.0  1.0   81.0         21.0
4.0
17411  2017-01-03 21:00:00   337  5.5  1.5   78.5         24.0
4.0
17412  2017-01-03 22:00:00   224  5.5  1.5   76.0         23.0
4.0
17413  2017-01-03 23:00:00   139  5.0  1.0   76.0         22.0
2.0

       is_holiday  is_weekend  season
0             0.0         1.0     3.0
1             0.0         1.0     3.0
2             0.0         1.0     3.0
3             0.0         1.0     3.0
4             0.0         1.0     3.0
...           ...         ...     ...
17409         0.0         0.0     3.0
17410         0.0         0.0     3.0
17411         0.0         0.0     3.0
17412         0.0         0.0     3.0
17413         0.0         0.0     3.0

[17414 rows x 10 columns]>
```

```
# 17414 satır ve 10 sütun bulunmaktadır. Bu satırların her biri bir
bisiklet kiralama süreci ile ilgilidir.
```

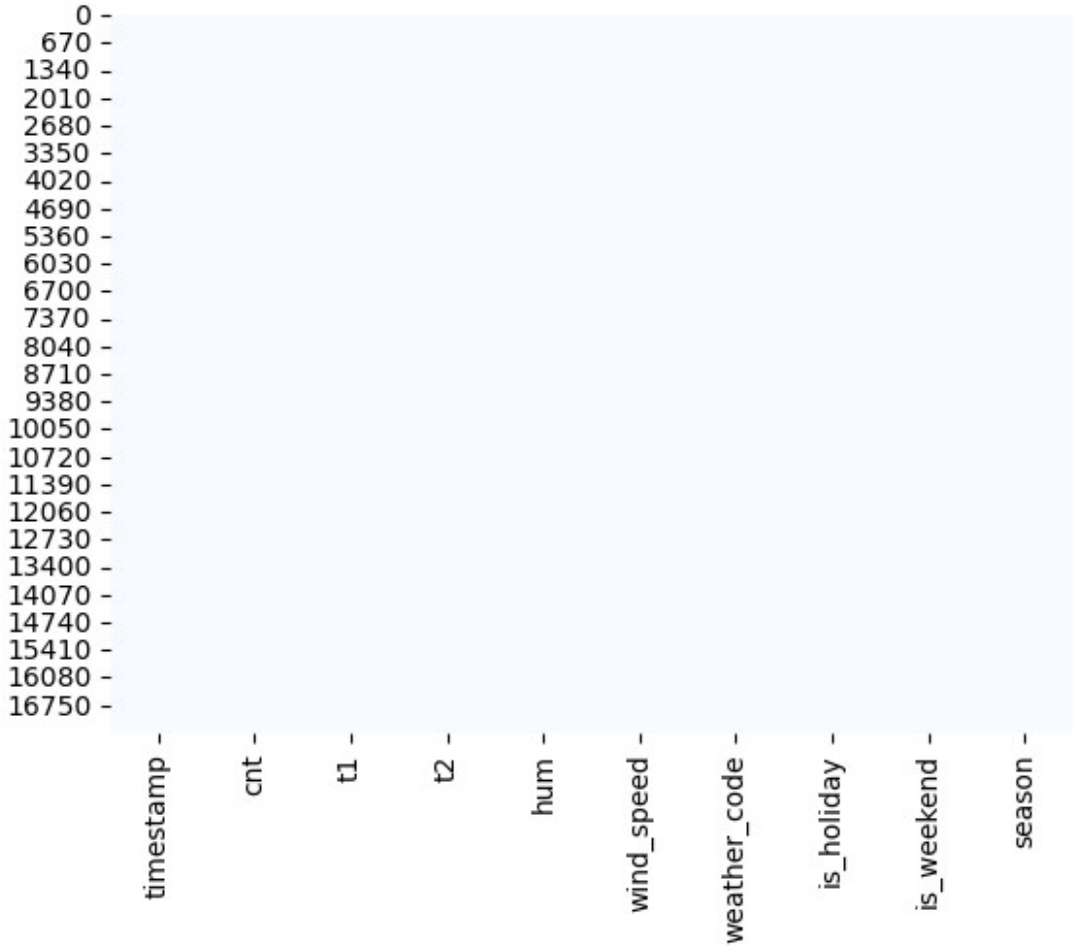
```
df.describe().T
```

```
                count         mean          std   min    25%    50%
75%  \
cnt           17414.0  1143.101642  1085.108068   0.0  257.0  844.0
1671.75
t1            17414.0    12.468091     5.571818  -1.5    8.0   12.5
16.00
t2            17414.0    11.520836     6.615145  -6.0    6.0   12.5
16.00
hum           17414.0    72.324954    14.313186  20.5   63.0   74.5
83.00
wind_speed    17414.0    15.913063     7.894570   0.0   10.0   15.0
20.50
weather_code  17414.0     2.722752     2.341163   1.0    1.0    2.0
3.00
```

```
is_holiday    17414.0     0.022051     0.146854   0.0    0.0    0.0
0.00
is_weekend    17414.0     0.285403     0.451619   0.0    0.0    0.0
1.00
season        17414.0     1.492075     1.118911   0.0    0.0    1.0
2.00

                  max
cnt            7860.0
t1               34.0
t2               34.0
hum             100.0
wind_speed       56.5
weather_code     26.0
is_holiday        1.0
is_weekend        1.0
season            3.0

df.isnull().sum

<bound method DataFrame.sum of        timestamp    cnt     t1     t2
hum  wind_speed  weather_code  \
0          False  False  False  False  False       False          False

1          False  False  False  False  False       False          False

2          False  False  False  False  False       False          False

3          False  False  False  False  False       False          False

4          False  False  False  False  False       False          False

...          ...    ...    ...    ...    ...         ...            ...

17409      False  False  False  False  False       False          False

17410      False  False  False  False  False       False          False

17411      False  False  False  False  False       False          False

17412      False  False  False  False  False       False          False

17413      False  False  False  False  False       False          False


       is_holiday  is_weekend  season
0           False       False   False
1           False       False   False
2           False       False   False
3           False       False   False
4           False       False   False
```

```
...           ...         ...     ...
17409       False       False   False
17410       False       False   False
17411       False       False   False
17412       False       False   False
17413       False       False   False

[17414 rows x 10 columns]>
```

```
sns.heatmap(df.isnull(),cmap = 'Blues',cbar = False);
```




```
# Veri setinde "Null" yani boş değer bulunmamaktadır.

# Verisetinin data tiplerini inceleyiniz.

df.dtypes
```

```
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
```

timestamp - verileri gruplandırmak için zaman verisi sütunu

cnt - yeni bisiklet paylaşımları sayısı

t1 - C cinsinden gerçek sıcaklık

t2 - C cinsinden hissedilen sıcaklık

hum - yüzde cinsinden nem oranı

wind_speed - km/h cinsinden rüzgar hızı

weather_code - hava durumu kategorisi

is_holiday - boolean alanı - 1 tatil / 0 tatil değil

is_weekend - boolean alanı - gün hafta sonuysa 1

season - kategori alanı meteorolojik mevsimler: 0-ilkbahar; 1-yaz; 2-sonbahar; 3-kış.

"*weather_code*" kategori açıklaması:

1 = Açık; çoğunlukla açık ancak pus/sis/sis parçaları/yakınlarda sis

2 = dağınık bulutlar / az bulut

3 = Parçalı bulutlar

4 = Bulutlu

7 = Yağmur/hafif Yağmur duşu/Hafif yağmur

10 = gök gürültülü fırtınalı yağmur

26 = kar yağışı

94 = Donan Sis

timestamp object : timestamp olması lazım : düzeltilecek cnt int64 : yeni bisiklet paylaşımları sayısı tamsayı, doğru t1 float64 : Celsius yani derece cinsinden sıcaklık ondalıklı değer, doğru t2

float64 : Celsius yani derece cinsinden hissedilen sıcaklık yani ondalıklı değer, doğru hum float64 : Yüzde cinsinden nem oranı, doğru wind_speed float64 : Km/h cinsinden rüzgar hızı, doğru weather_code float64 : Hava Durumu kategorisi, kategorik olması lazım, doğru is_holiday float64 : Tatil mi değil mi kategorisi, kategorik olması lazım, doğru is_weekend float64 : Hafta sonu mu hafta içi mi kategorik olması lazım, doğru season float64 : Mevsimler, kategorik olması lazım, doğru dtype: object

Data Cleaning:

- Handle missing values.
- Check for duplicates and remove them if found.
- Standardize column names (if necessary) for consistent naming conventions.
- Validate data types and convert columns to appropriate types if needed.
- Look at the data type of each variable, transform timestamp in type, and set it as index.
- Make feature engineering. Extract new columns (day of the week, day of the month, hour, month, season, year etc.)

Handle missing values : Eksik değer bulunmamaktadır.

```
df.duplicated().sum
```

```
<bound method Series.sum of 0        False
1        False
2        False
3        False
4        False
         ...
17409    False
17410    False
17411    False
17412    False
17413    False
Length: 17414, dtype: bool>
```

```
# Duplicated değer bulunmamaktadır.
```

```
df.columns
```

```
Index(['timestamp', 'cnt', 't1', 't2', 'hum', 'wind_speed',
'weather_code',
       'is_holiday', 'is_weekend', 'season'],
      dtype='object')
```

```
df.columns = df.columns.str.strip()

df.columns = df.columns.str.strip().str.lower().str.replace(' ',
'_').str.replace(r'[^\w]', '', regex=True)
df.columns

Index(['timestamp', 'cnt', 't1', 't2', 'hum', 'wind_speed',
'weather_code',
       'is_holiday', 'is_weekend', 'season'],
      dtype='object')

df = df.rename(columns={'t1': 'temperature_1', 't2': 'temperature_2'})
df.head(3)

             timestamp  cnt  temperature_1  temperature_2   hum
wind_speed  \
0  2015-01-04 00:00:00  182            3.0            2.0  93.0
6.0
1  2015-01-04 01:00:00  138            3.0            2.5  93.0
5.0
2  2015-01-04 02:00:00  134            2.5            2.5  96.5
0.0

   weather_code  is_holiday  is_weekend  season
0           3.0         0.0         1.0     3.0
1           1.0         0.0         1.0     3.0
2           1.0         0.0         1.0     3.0
```

Analysis Goal

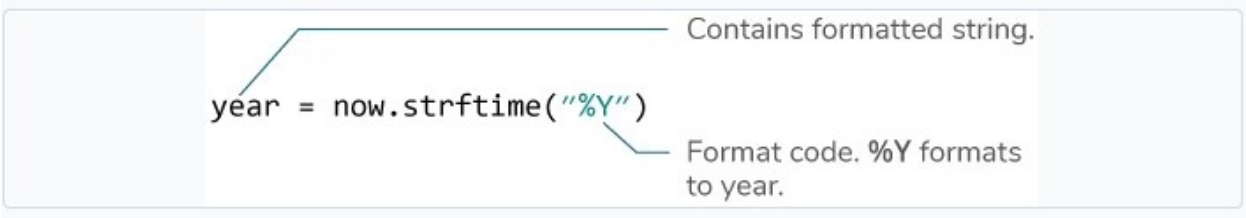
Look at the data type of each variable, transform timestamp in type, and set it as index.

```
# Datatipleri kontrol edilmiştir.

# timestamp sütununu datetime veri tipine çevirme işlemi:
df['timestamp'] = pd.to_datetime(df['timestamp'])
df.dtypes

timestamp        datetime64[ns]
cnt                       int64
temperature_1           float64
temperature_2           float64
hum                     float64
wind_speed              float64
weather_code            float64
is_holiday              float64
is_weekend              float64
season                  float64
dtype: object
```

```
# timestamp sütununu indeks olarak ayarlayın
df = df.set_index('timestamp')
df.head(2)
```

```
                     cnt  temperature_1  temperature_2   hum
wind_speed  \
timestamp

2015-01-04 00:00:00  182            3.0            2.0  93.0
6.0
2015-01-04 01:00:00  138            3.0            2.5  93.0
5.0

                     weather_code  is_holiday  is_weekend  season
timestamp
2015-01-04 00:00:00           3.0         0.0         1.0     3.0
2015-01-04 01:00:00           1.0         0.0         1.0     3.0
```

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




```
df['year'] = df.index.year

df['month'] = df.index.month

df['day'] = df.index.day

df['hour'] = df.index.hour

df['day_of_week'] = df.index.dayofweek

df['day_name'] = df.index.day_name()

df['day_of_month'] = df.index.day

df['week_number'] = df.index.isocalendar().week

df['day_of_year'] = df.index.dayofyear

def get_season(month):
    if 3 <= month <= 5:
        return 'Spring'
```

```
    elif 6 <= month <= 8:
        return 'Summer'
    elif 9 <= month <= 11:
        return 'Autumn'
    else:
        return 'Winter'

df['season'] = df['month'].apply(get_season)

df['temp_diff'] = df['temperature_1'] - df['temperature_2']

df.head()
print(df.dtypes)
```

```
cnt                int64
temperature_1    float64
temperature_2    float64
hum              float64
wind_speed       float64
weather_code     float64
is_holiday       float64
is_weekend       float64
season            object
year               int32
month              int32
day                int32
hour               int32
day_of_week        int32
day_name          object
day_of_month       int32
week_number       UInt32
day_of_year        int32
temp_diff        float64
dtype: object
```

```
df.head()
```

```
                     cnt  temperature_1  temperature_2    hum
wind_speed  \
timestamp

2015-01-04 00:00:00  182            3.0            2.0   93.0
6.0
2015-01-04 01:00:00  138            3.0            2.5   93.0
5.0
2015-01-04 02:00:00  134            2.5            2.5   96.5
0.0
2015-01-04 03:00:00   72            2.0            2.0  100.0
0.0
2015-01-04 04:00:00   47            2.0            0.0   93.0
```

```
6.5

                     weather_code  is_holiday  is_weekend  season
year  \
timestamp

2015-01-04 00:00:00           3.0         0.0         1.0  Winter
2015
2015-01-04 01:00:00           1.0         0.0         1.0  Winter
2015
2015-01-04 02:00:00           1.0         0.0         1.0  Winter
2015
2015-01-04 03:00:00           1.0         0.0         1.0  Winter
2015
2015-01-04 04:00:00           1.0         0.0         1.0  Winter
2015

                     month  day  hour  day_of_week day_name
day_of_month  \
timestamp

2015-01-04 00:00:00      1    4     0            6   Sunday
4
2015-01-04 01:00:00      1    4     1            6   Sunday
4
2015-01-04 02:00:00      1    4     2            6   Sunday
4
2015-01-04 03:00:00      1    4     3            6   Sunday
4
2015-01-04 04:00:00      1    4     4            6   Sunday
4

                     week_number  day_of_year  temp_diff
timestamp
2015-01-04 00:00:00            1            4        1.0
2015-01-04 01:00:00            1            4        0.5
2015-01-04 02:00:00            1            4        0.0
2015-01-04 03:00:00            1            4        0.0
2015-01-04 04:00:00            1            4        2.0
```

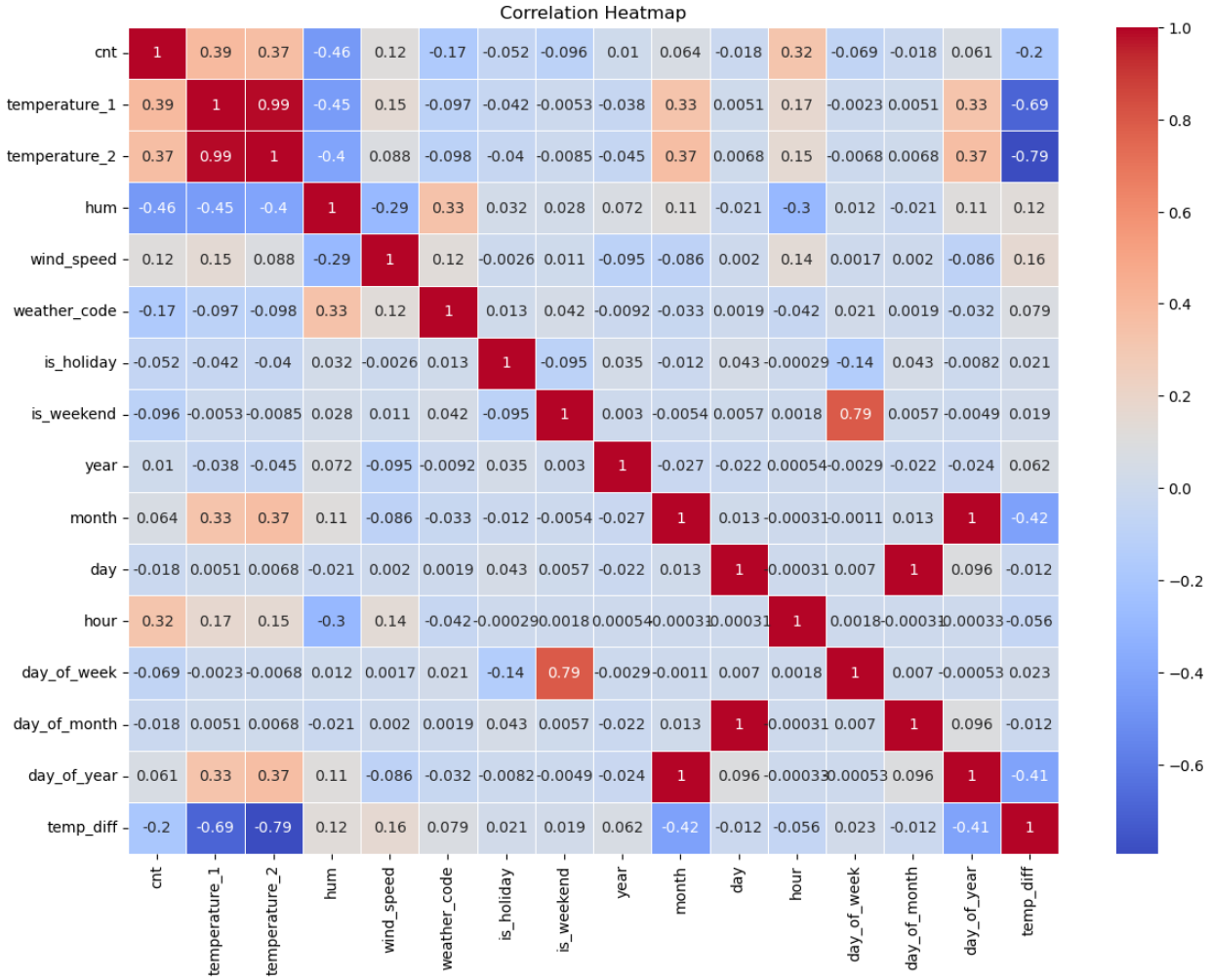
Visualize the correlation with a heatmap

```
df.columns

Index(['cnt', 'temperature_1', 'temperature_2', 'hum', 'wind_speed',
       'weather_code', 'is_holiday', 'is_weekend', 'season', 'year',
'month',
       'day', 'hour', 'day_of_week', 'day_name', 'day_of_month',
'week_number',
```

```
       'day_of_year', 'temp_diff'],
      dtype='object')
```

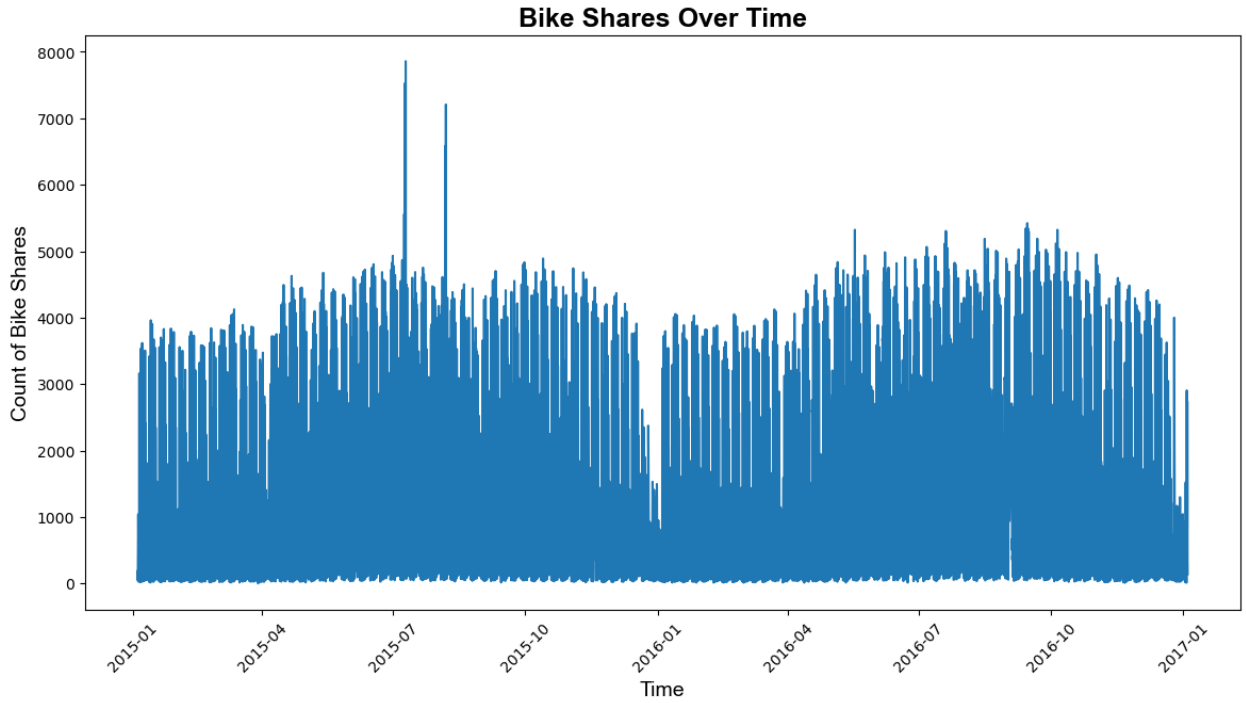
Visualize the correlation of the target variable and the other features with barplot

```
# Select only numeric columns
numeric_df = df.select_dtypes(include=[float, int])


# Calculate the correlation matrix
corr_matrix = numeric_df.corr()

# Create the heatmap
plt.figure(figsize=(14, 10))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```



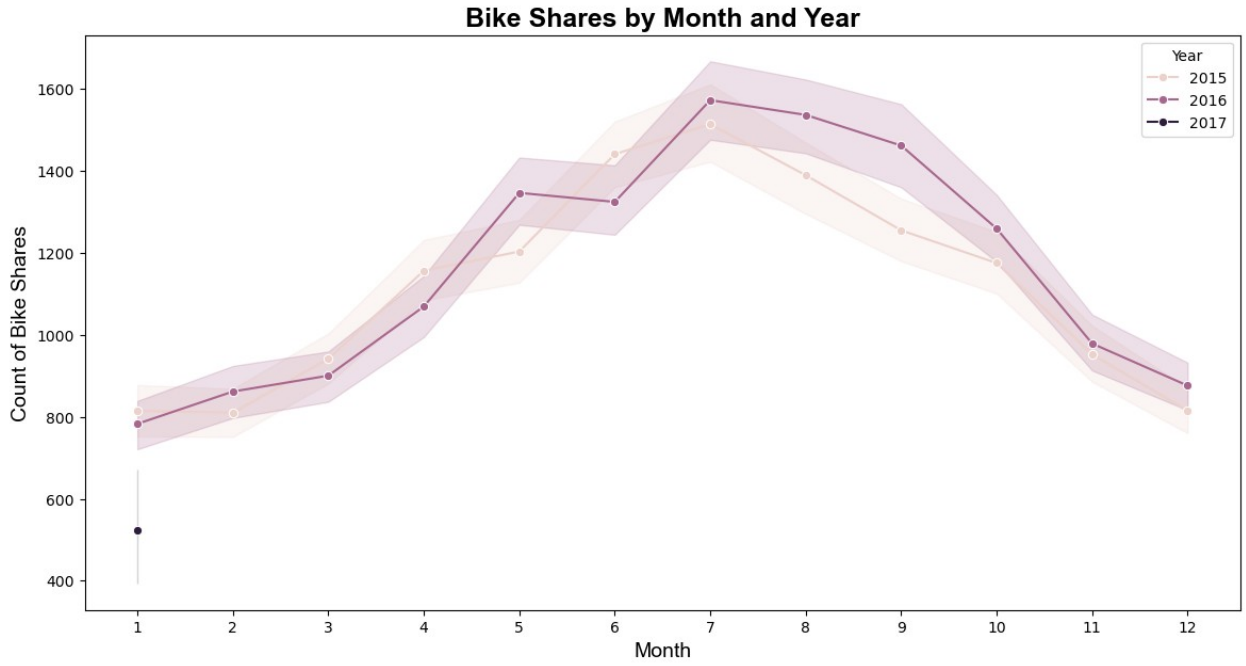
Plot bike shares over time use lineplot.

```
# Create the line plot
plt.figure(figsize=(14, 7))
sns.lineplot(data=df, x=df.index, y='cnt')
plt.title('Bike Shares Over Time', fontsize=18, fontweight='bold',
fontname='Arial')
plt.xlabel('Time', fontsize=14, fontname='Arial')
plt.ylabel('Count of Bike Shares', fontsize=14, fontname='Arial')
plt.xticks(rotation=45)
plt.show()
```

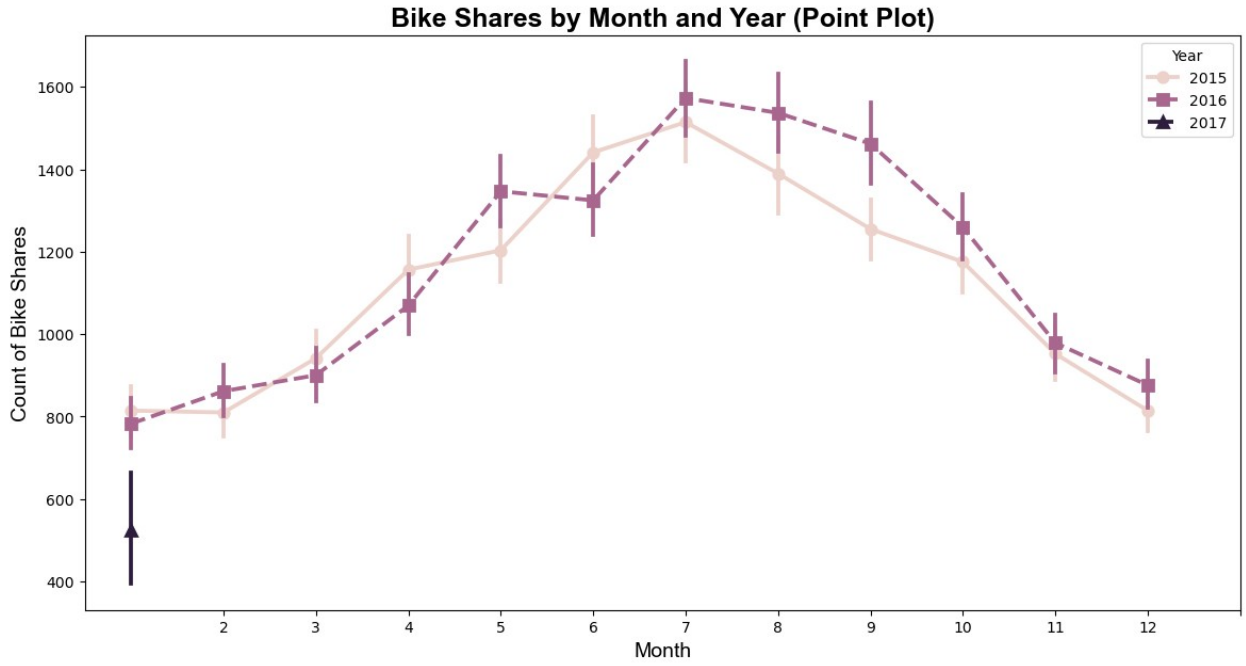


Plot bike shares by months and year_of_month (use lineplot, pointplot, barplot).

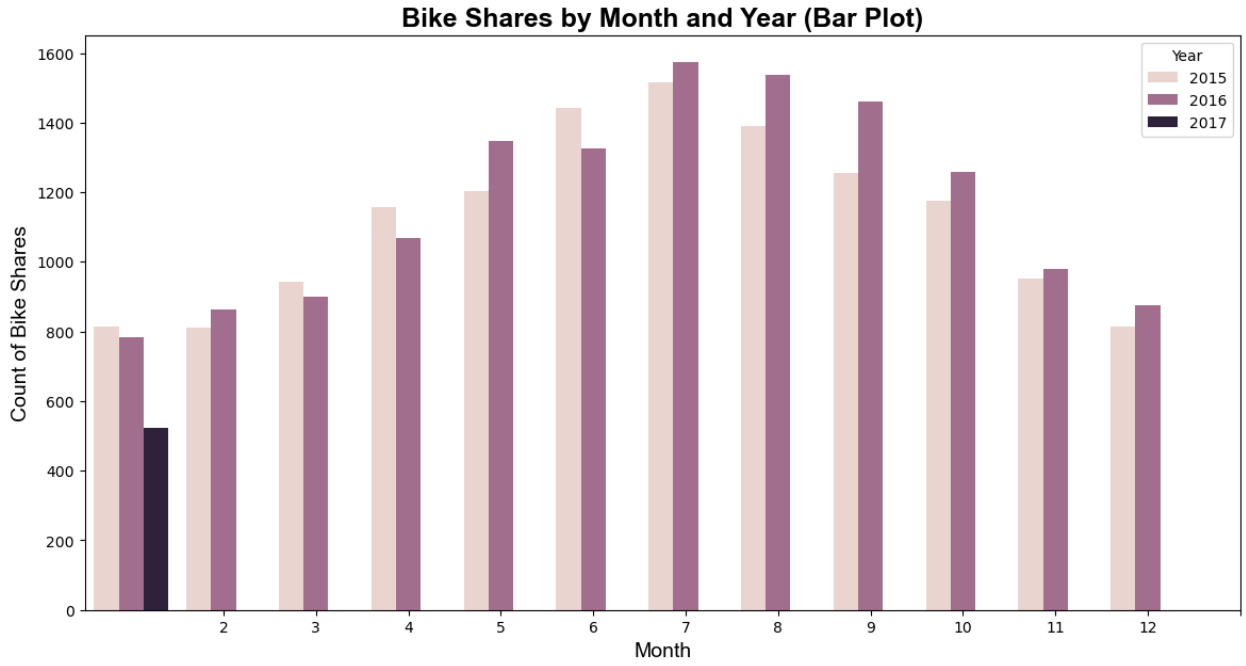
```
plt.figure(figsize=(14, 7))
sns.lineplot(data=df, x='month', y='cnt', hue='year', marker='o')
plt.title('Bike Shares by Month and Year', fontsize=18,
fontweight='bold', fontname='Arial')
plt.xlabel('Month', fontsize=14, fontname='Arial')
plt.ylabel('Count of Bike Shares', fontsize=14, fontname='Arial')
plt.xticks(range(1, 13))
plt.legend(title='Year')
plt.show()
```




```
plt.figure(figsize=(14, 7))
sns.pointplot(data=df, x='month', y='cnt', hue='year', markers=['o',
's', '^', 'D'], linestyles=['-', '--', '-.', ':'])
plt.title('Bike Shares by Month and Year (Point Plot)', fontsize=18,
fontweight='bold', fontname='Arial')
plt.xlabel('Month', fontsize=14, fontname='Arial')
plt.ylabel('Count of Bike Shares', fontsize=14, fontname='Arial')
plt.xticks(range(1, 13))
plt.legend(title='Year')
plt.show()
```

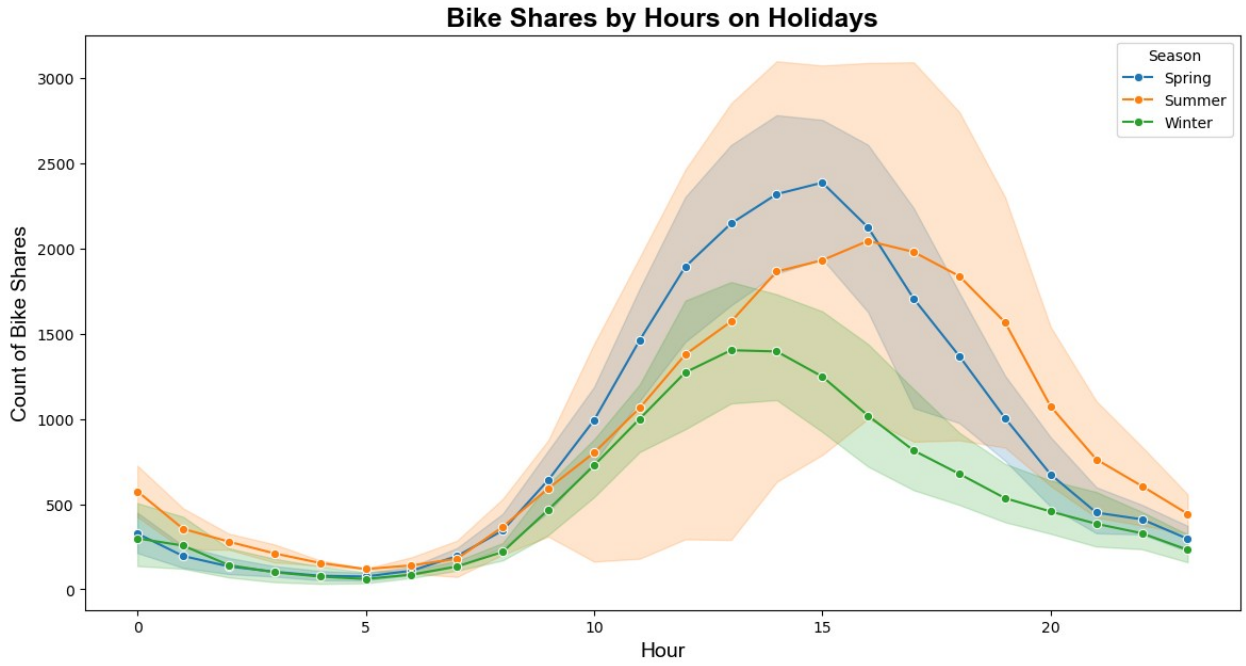



```
plt.figure(figsize=(14, 7))
sns.barplot(data=df, x='month', y='cnt', hue='year', ci=None)
plt.title('Bike Shares by Month and Year (Bar Plot)', fontsize=18,
fontweight='bold', fontname='Arial')
plt.xlabel('Month', fontsize=14, fontname='Arial')
plt.ylabel('Count of Bike Shares', fontsize=14, fontname='Arial')
plt.xticks(range(1, 13))
plt.legend(title='Year')
plt.show()
```

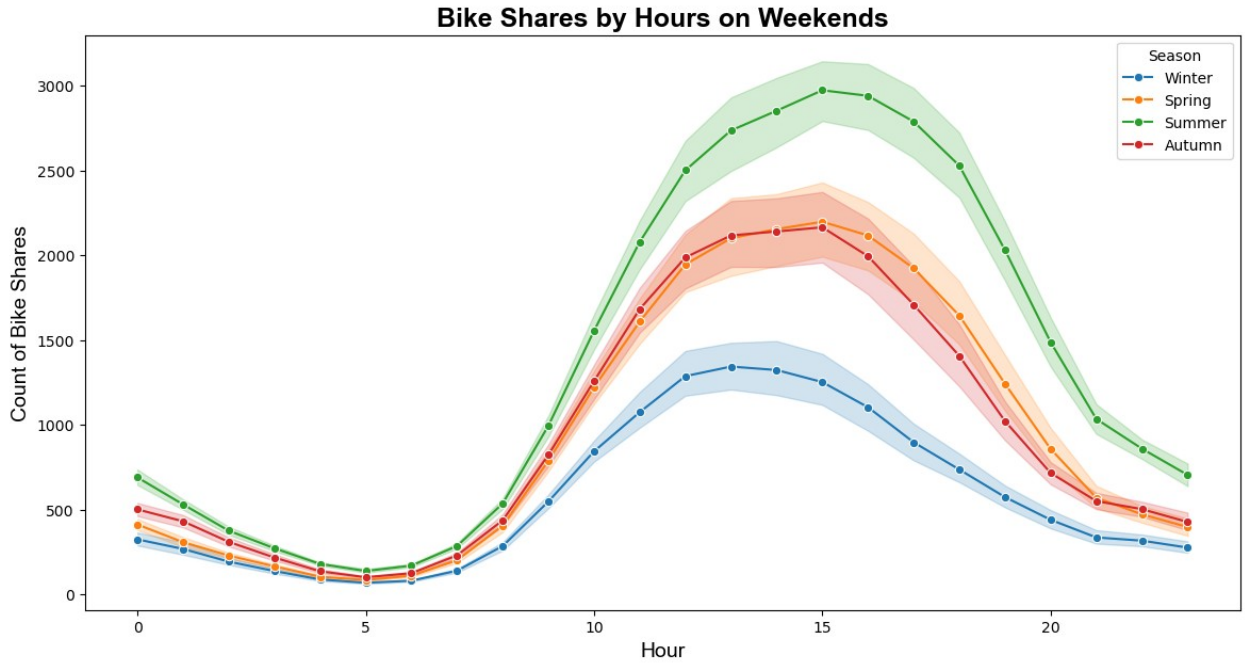


Plot bike shares by hours on (holidays, weekend, season).

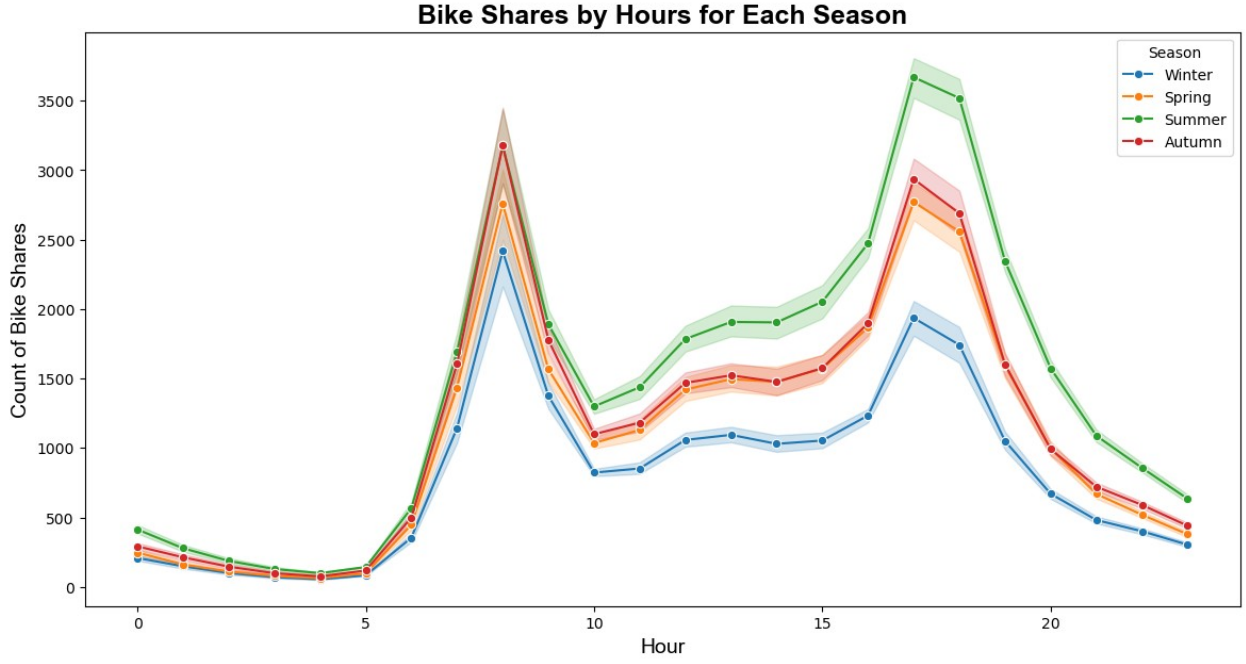
```
plt.figure(figsize=(14, 7))
sns.lineplot(data=df[df['is_holiday'] == 1], x='hour', y='cnt',
hue='season', marker='o')
plt.title('Bike Shares by Hours on Holidays', fontsize=18,
fontweight='bold', fontname='Arial')
plt.xlabel('Hour', fontsize=14, fontname='Arial')
plt.ylabel('Count of Bike Shares', fontsize=14, fontname='Arial')
plt.legend(title='Season')
plt.show()
```




```
plt.figure(figsize=(14, 7))
sns.lineplot(data=df[df['is_weekend'] == 1], x='hour', y='cnt',
hue='season', marker='o')
plt.title('Bike Shares by Hours on Weekends', fontsize=18,
fontweight='bold', fontname='Arial')
plt.xlabel('Hour', fontsize=14, fontname='Arial')
plt.ylabel('Count of Bike Shares', fontsize=14, fontname='Arial')
plt.legend(title='Season')
plt.show()
```



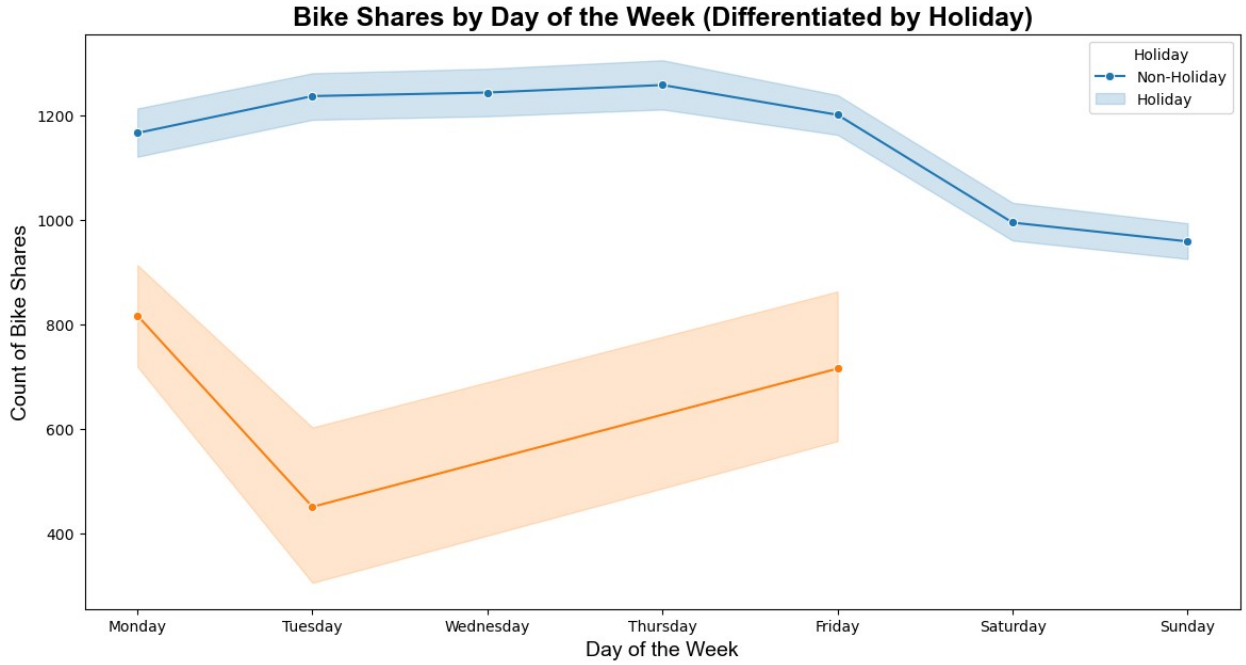

```
plt.figure(figsize=(14, 7))
sns.lineplot(data=df, x='hour', y='cnt', hue='season', marker='o')
plt.title('Bike Shares by Hours for Each Season', fontsize=18,
fontweight='bold', fontname='Arial')
plt.xlabel('Hour', fontsize=14, fontname='Arial')
plt.ylabel('Count of Bike Shares', fontsize=14, fontname='Arial')
plt.legend(title='Season')
plt.show()
```



Plot bike shares by day of week.

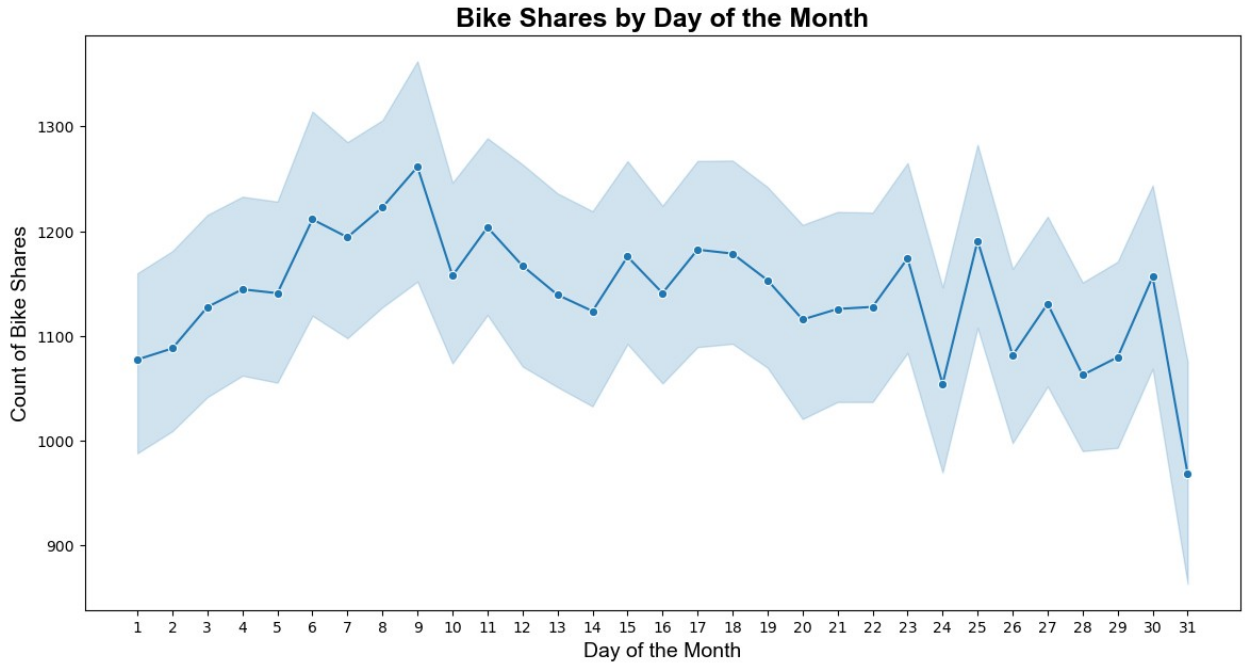
- You may want to see whether it is a holiday or not

```
plt.figure(figsize=(14, 7))
sns.lineplot(data=df, x='day_of_week', y='cnt', hue='is_holiday',
marker='o')
plt.title('Bike Shares by Day of the Week (Differentiated by
Holiday)', fontsize=18, fontweight='bold', fontname='Arial')
plt.xlabel('Day of the Week', fontsize=14, fontname='Arial')
plt.ylabel('Count of Bike Shares', fontsize=14, fontname='Arial')
plt.xticks(ticks=range(7), labels=['Monday', 'Tuesday', 'Wednesday',
'Thursday', 'Friday', 'Saturday', 'Sunday'])
plt.legend(title='Holiday', labels=['Non-Holiday', 'Holiday'])
plt.show()
```



Plot bike shares by day of month

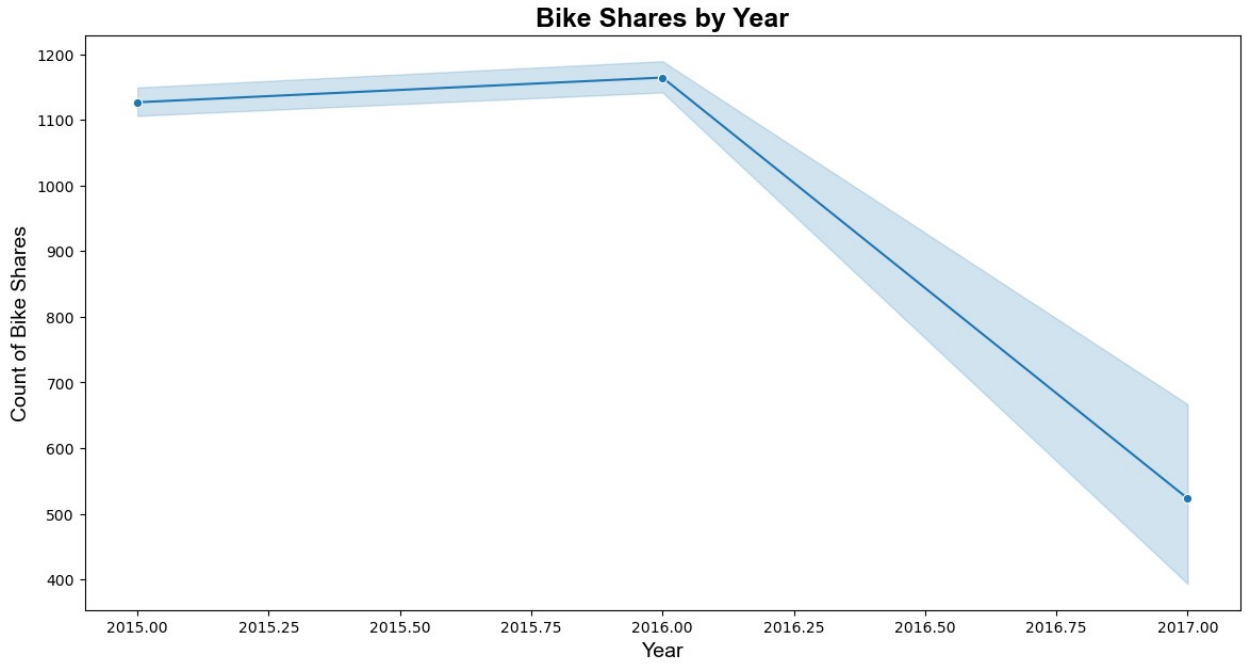
```
plt.figure(figsize=(14, 7))
sns.lineplot(data=df, x='day_of_month', y='cnt', marker='o')
plt.title('Bike Shares by Day of the Month', fontsize=18,
fontweight='bold', fontname='Arial')
plt.xlabel('Day of the Month', fontsize=14, fontname='Arial')
plt.ylabel('Count of Bike Shares', fontsize=14, fontname='Arial')
plt.xticks(range(1, 32))
plt.show()
```



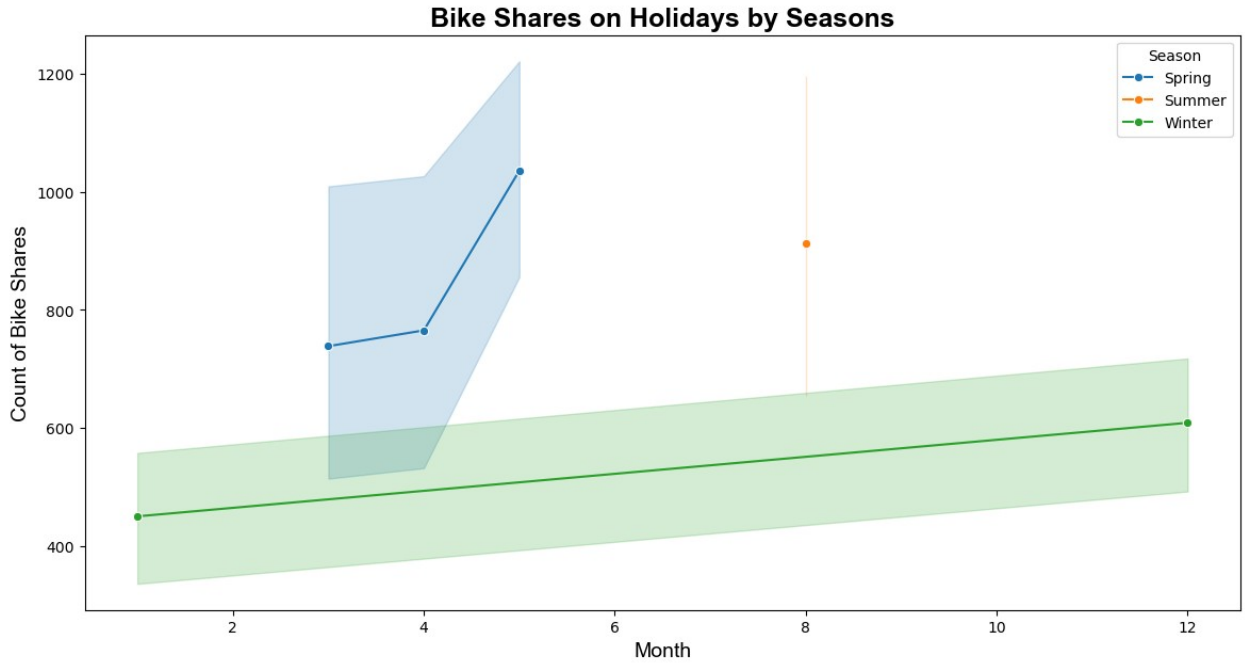
Plot bike shares by year

- Plot bike shares on holidays by seasons

```
plt.figure(figsize=(14, 7))
sns.lineplot(data=df, x='year', y='cnt', marker='o')
plt.title('Bike Shares by Year', fontsize=18, fontweight='bold',
fontname='Arial')
plt.xlabel('Year', fontsize=14, fontname='Arial')
plt.ylabel('Count of Bike Shares', fontsize=14, fontname='Arial')
plt.show()
```

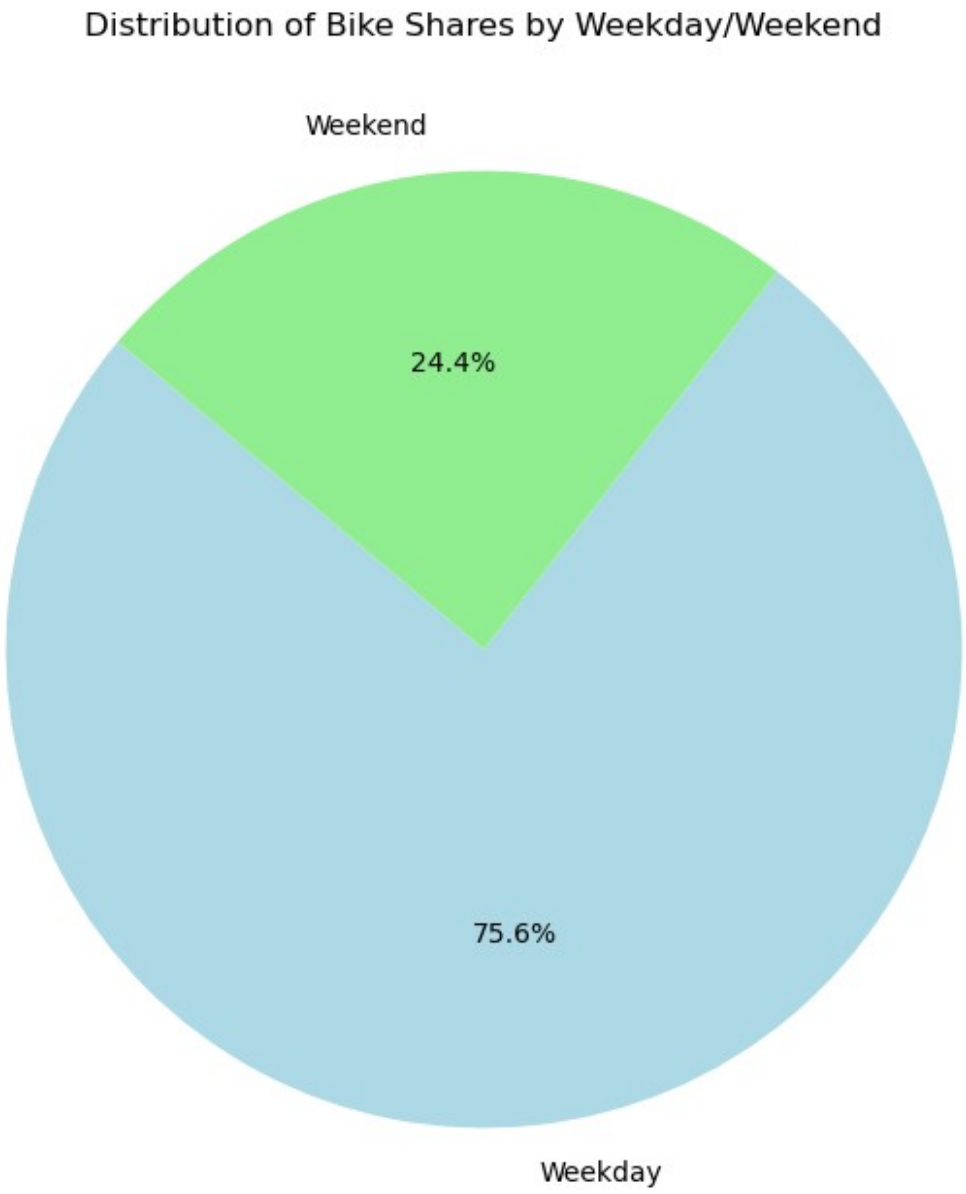



```
plt.figure(figsize=(14, 7))
sns.lineplot(data=df[df['is_holiday'] == 1], x='month', y='cnt',
hue='season', marker='o')
plt.title('Bike Shares on Holidays by Seasons', fontsize=18,
fontweight='bold', fontname='Arial')
plt.xlabel('Month', fontsize=14, fontname='Arial')
plt.ylabel('Count of Bike Shares', fontsize=14, fontname='Arial')
plt.legend(title='Season')
plt.show()
```

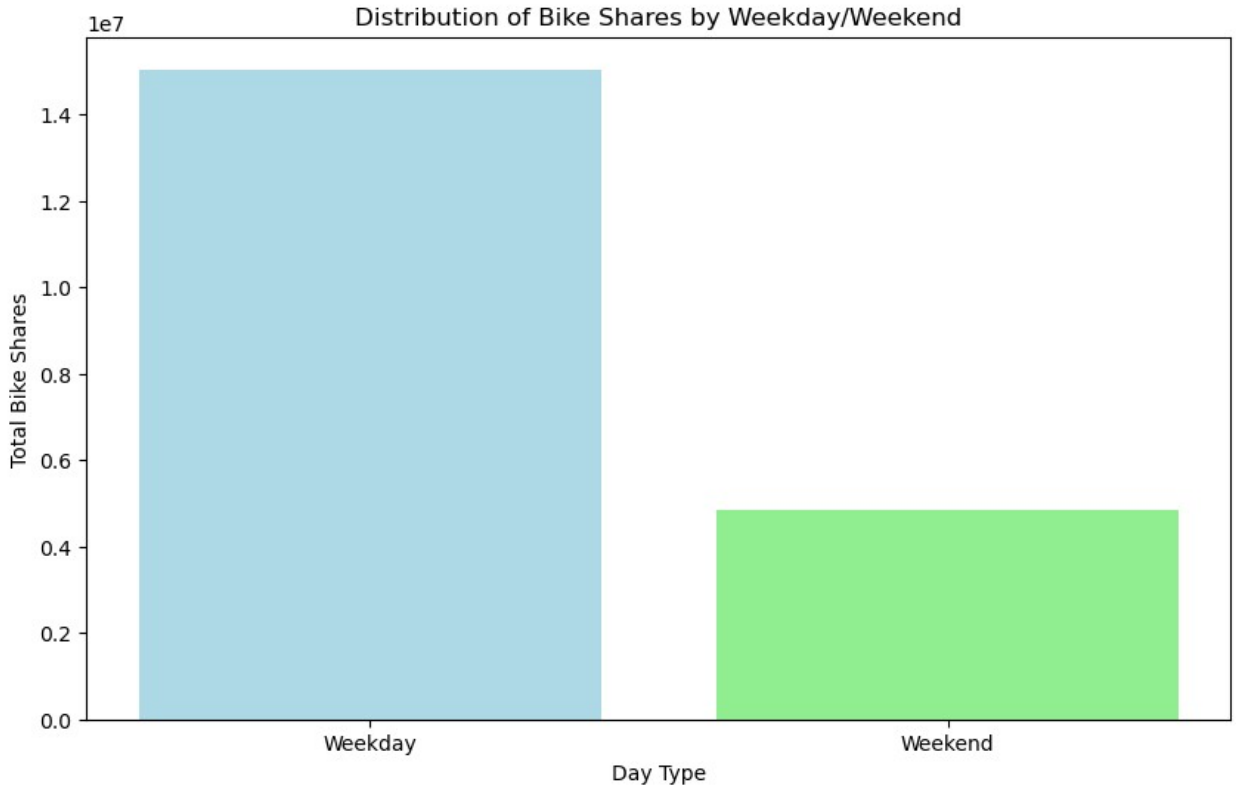


Visualize the distribution of bike shares by weekday/weekend with piechart and barplot

```
# Group by 'is_weekend' and calculate the total bike shares
weekend_distribution = df.groupby('is_weekend')['cnt'].sum()

# Create the pie chart
plt.figure(figsize=(8, 8))
plt.pie(weekend_distribution, labels=['Weekday', 'Weekend'],
autopct='%1.1f%%', startangle=140, colors=['lightblue', 'lightgreen'])
plt.title('Distribution of Bike Shares by Weekday/Weekend')
plt.show()
```

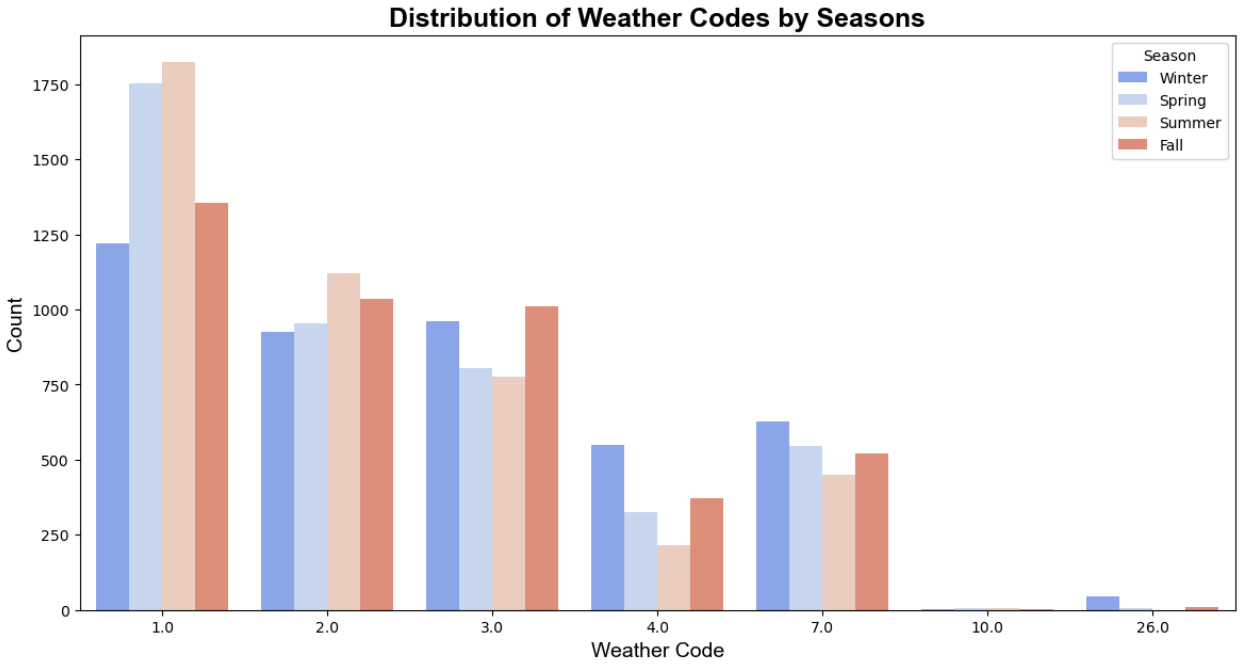



```
plt.figure(figsize=(10, 6))
plt.bar(weekend_distribution.index, weekend_distribution.values,
color=['lightblue', 'lightgreen'])
plt.xticks(ticks=[0, 1], labels=['Weekday', 'Weekend'])
plt.xlabel('Day Type')
plt.ylabel('Total Bike Shares')
plt.title('Distribution of Bike Shares by Weekday/Weekend')
plt.show()
```



Plot the distribution of weather code by seasons

```
# Define the season column if it's not already defined
def get_season(month):
    if month in [12, 1, 2]:
        return 'Winter'
    elif month in [3, 4, 5]:
        return 'Spring'
    elif month in [6, 7, 8]:
        return 'Summer'
    elif month in [9, 10, 11]:
        return 'Fall'

# Assuming 'month' column is present in your DataFrame
df['season'] = df['month'].apply(get_season)

plt.figure(figsize=(14, 7))
sns.countplot(data=df, x='weather_code', hue='season',
palette='coolwarm')
plt.title('Distribution of Weather Codes by Seasons', fontsize=18,
fontweight='bold', fontname='Arial')
plt.xlabel('Weather Code', fontsize=14, fontname='Arial')
plt.ylabel('Count', fontsize=14, fontname='Arial')
plt.legend(title='Season')
plt.show()
```



Feel free to include any additional analyses.

```
# Thank you very much....
```

Conclusions

Bike Demand Project

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