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.

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()

In [5]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
warnings.warn("this will not show")
```

In [6]:
```
df= pd.read_csv('store_sharing.csv')
df.head()
```

Out[6]:

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

In [7]:
```
df1 = df.copy()
```

In [8]:
```
df.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
```

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
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 [10]:
```
df.duplicated().sum()
```

Out[10]: 0

In [11]:
```
from skimpy import skim
skim(df)
```

skimpy summary

Data Summary

dataframe	Values
Number of rows	17414
Number of columns	10

Data Types

Column Type	Count
float64	8
string	1
int32	1

number

column_name	NA	NA %	mean	sd	p0
cnt	0	0	1143	1085	0
t1	0	0	12.47	5.572	-1.5
t2	0	0	11.52	6.615	-6
hum	0	0	72.32	14.31	20.5
wind_speed	0	0	15.91	7.895	0
weather_code	0	0	2.723	2.341	1
is_holiday	0	0	0.02205	0.1469	0
is_weekend	0	0	0.2854	0.4516	0
season	0	0	1.492	1.119	0

string

column_name	NA	NA %	words per row
timestamp	0	0	

End

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.)

In [13]:
```
data_columns = {
    'timestamp': 'Datetime',
    'cnt': 'Ride_Count',
    't1': 'Actual_Temperature',
    't2': 'Feels_Like_Temperature',
    'hum': 'Humidity_Percent',
    'wind_speed': 'Wind_Speed_Km/H',
    'weather_code': 'Weather_Code',
    'is_holiday': 'Is_Holiday',
    'is_weekend': 'Is_Weekend',
    'season': 'Season_Code'
}

df.rename(columns=data_columns, inplace=True)

df.head(2)
```

Out[13]:

	Datetime	Ride_Count	Actual_Temperature	Feels_Like_Temperature	Humidi
0	2015-01-04 00:00:00	182	3.0	2.0	
1	2015-01-04 01:00:00	138	3.0	2.5	

In [14]:
```
df.Weather_Code.value_counts()
```

Out[14]:
```
Weather_Code
1.0     6150
2.0     4034
3.0     3551
7.0     2141
4.0     1464
26.0      60
10.0      14
Name: count, dtype: int64
```

In [15]:
```
Weather_Code = {
    1.0: '1 : Clear / Haze / Fog',
    2.0: '2 : Scattered Clouds / Few Clouds',
    3.0: '3 : Broken Clouds',
    4.0: '4 : Cloudy',
    7.0: '7 : Rain / Light Rain Shower',
    10.0: '10 : Rain with Thunderstorm',
    26.0: '26 : Snowfall',
    94.0: '94 : Freezing Fog'
}
```

```
df['Weather_Description'] = df['Weather_Code'].map(Weather_Code)

df.head(2)
```

Out[15]:

	Datetime	Ride_Count	Actual_Temperature	Feels_Like_Temperature	Humidit
0	2015-01-04 00:00:00	182	3.0	2.0	
1	2015-01-04 01:00:00	138	3.0	2.5	

In [16]:

```
season_descriptions = {
    0.0: 'Spring',
    1.0: 'Summer',
    2.0: 'Fall',
    3.0: 'Winter'
}
df['season_description'] = df['Season_Code'].map(season_descriptions)

df.head(2)
```

Out[16]:

	Datetime	Ride_Count	Actual_Temperature	Feels_Like_Temperature	Humidit
0	2015-01-04 00:00:00	182	3.0	2.0	
1	2015-01-04 01:00:00	138	3.0	2.5	

Analysis Goal

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

In [19]:

```
df['Datetime'] = pd.to_datetime(df['Datetime'])
```

In [20]:

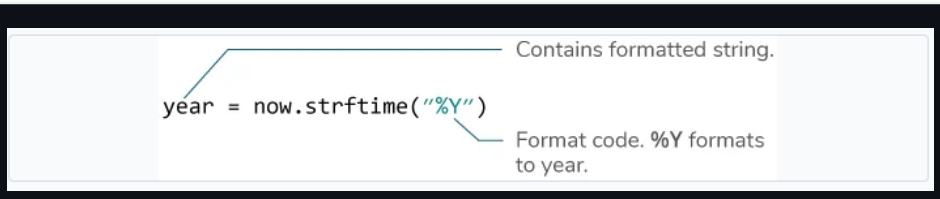
```
df['Is_Holiday'] = df['Is_Holiday'].astype(int)
df['Is_Weekend'] = df['Is_Weekend'].astype(int)
df['Weather_Code'] = df['Weather_Code'].astype(int)
df['Season_Code'] = df['Season_Code'].astype(int)

df.head(2)
```

Out[20]:

	Datetime	Ride_Count	Actual_Temperature	Feels_Like_Temperature	Hum

0	2015-01-04 00:00:00	182	3.0	2.0
1	2015-01-04 01:00:00	138	3.0	2.5

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



In [23]:
```
df['Year'] = df['Datetime'].dt.year
df['Month'] = df['Datetime'].dt.month
df['Day'] = df['Datetime'].dt.day
df['Hour'] = df['Datetime'].dt.hour

df.head(2)
```

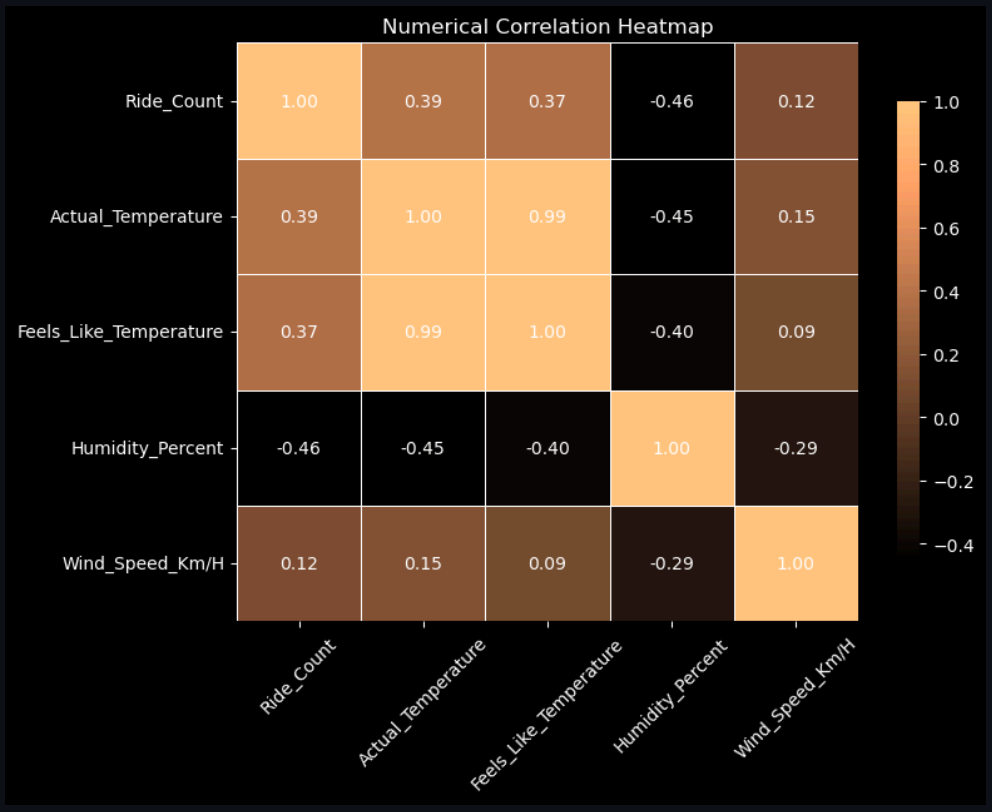
Out[23]:

	Datetime	Ride_Count	Actual_Temperature	Feels_Like_Temperature	Hu
0	2015-01-04 00:00:00	182	3.0	2.0	
1	2015-01-04 01:00:00	138	3.0	2.5	

Visualize the correlation with a heatmap

In [25]:
```
numerical_columns = ['Ride_Count', 'Actual_Temperature', 'Feels_L
correlation_matrix = df[numerical_columns].corr()

plt.style.use('dark_background')
fig = plt.figure(figsize=(8, 6))

sns.heatmap(
    correlation_matrix,
    annot=True,
    cmap="copper",
    fmt=".2f",
    linewidths=0.5,
    cbar_kws={'shrink': 0.8},
    annot_kws={"color": "white"}
)
plt.title("Numerical Correlation Heatmap", color="white")
```

```
plt.xticks(rotation=45)
plt.yticks(rotation=0)
plt.show()
```



🌡️ "Feels Like" ve Gerçek Sıcaklık Arasındaki İlişki

📊 **"Feels Like" sıcaklığı** ile **gerçek sıcaklık ("Actual Temperature")** arasında neredeyse **%100'e yakın doğrusal bir ilişki** olduğu gözlemlenmiştir.

📉 Diğer tüm değişkenlerin kendi aralarındaki **ikili (pairwise) korelasyonları** incelendiğinde ise **düşük korelasyon değerleri** tespit edilmiştir.

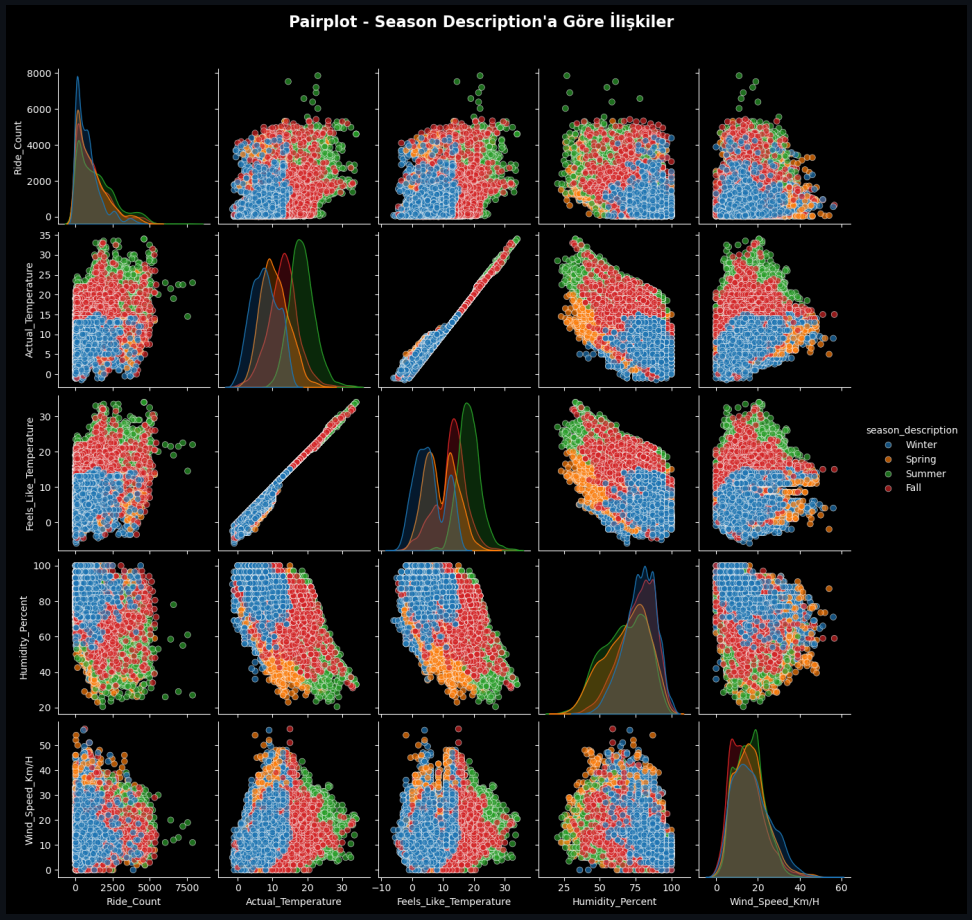
🔗 Daha fazla bilgi için: Everything You Need to Know About Interpreting Correlations

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

In [28]:

```
# colors = ['#A1C6EA', '#FF5C8D', '#F7C8A3', '#F9E6A0']  # Açık

# Pairplot
plt.style.use('dark_background')

numerical_columns = ['Ride_Count', 'Actual_Temperature', 'Feels_

sns.pairplot(df[numerical_columns], hue="season_description", pa
```

```
# Başlık
plt.suptitle("Pairplot - Season Description'a Göre İlişkiler", s

# Gösterim
plt.show()
```



🎯 "Ride Count" Hedef Değişken Olarak Belirlendi

📌 **"Ride Count"**, veride yer alan diğer değişkenlerle olan ilişkileri dikkate alınarak **hedef değişken (target variable)** olarak belirlenmiştir.

🌦️ Özellikle **yağış, rüzgar, zaman ve hava sıcaklığı** gibi faktörlerle **açıklanabilen bir sonuç değişkeni** olduğu görülmüştür.

📊 Bu değişkenlerin "Ride Count" üzerindeki etkisini daha iyi anlamak için korelasyon ve regresyon analizleri yapılabilir.

In [30]:

```
import plotly.graph_objects as go

def create_correlation_plot(df):
    numerical_columns = ['Ride_Count', 'Actual_Temperature', 'Fe
                         'Humidity_Percent', 'Wind_Speed_Km/H']

    correlations = df[numerical_columns].corr()['Ride_Count'].so

    correlations = correlations.drop('Ride_Count')
```

```
    fig = go.Figure(go.Bar(
        x=correlations,
        y=correlations.index,
        orientation='h',
        marker=dict(
            color=['blue' if val >= 0 else 'red' for val in corr
            line=dict(color='white', width=1)
        )
    ))

    fig.update_layout(
        title='Feature Correlations with Ride Count',
        xaxis_title='Correlation with Ride_Count',
        yaxis_title='Features',
        plot_bgcolor='black',
        paper_bgcolor='black',
        font=dict(color='white'),
        xaxis=dict(showgrid=True, gridcolor='gray'),
        yaxis=dict(showgrid=True, gridcolor='gray')
    )

    fig.show()
```

In [31]:
```
create_correlation_plot(df)
```

📊 "Ride Count" Değişkeninin Diğer Değişkenlerle İlişkisi

📌 **"Ride Count"** değişkeninin diğer değişkenlerle olan ilişkisi, **interaktif bir bar grafiği (barplot)** kullanılarak görselleştirilmiştir.

🔍 **Analiz Sonuçları:**

- 🌡️ **Sıcaklık** ile **pozitif** ilişki
- 💨 **Rüzgar** ile **pozitif** ilişki
- 💧 **Nem** ile **negatif** ilişki

📈 Bu bulgular, hava koşullarının **sürüş sayısı ("Ride Count")** üzerindeki etkisini anlamak açısından önemli içgörüler sunmaktadır.

Plot bike shares over time use lineplot.

In [34]:
```
data_yearly2 = df.groupby('Datetime')['Ride_Count'].mean().rese
```

In [35]:
```
import plotly.express as px

fig = px.line(
    data_yearly2
```

```
    x="Datetime",
    y="Ride_Count",
    title="Bike Shares Over Time",
    labels={"Datetime": "Time", "Ride_Count": "Ride Count"},
    template="plotly_dark",
)

fig.update_traces(
    line=dict(color='blue', width=0.7)
)

fig.update_layout(
    title_font=dict(size=16, family="Arial", color="white"),
    title_x=0.5,
    title_y=0.97,
    xaxis_title_font=dict(size=12, family="Arial"),
    yaxis_title_font=dict(size=12, family="Arial"),
    xaxis=dict(tickangle=45),
    margin=dict(t=50, l=50, r=50, b=50),
    width=1200,
    height=700,
)

fig.show()
```

📈 Zirve Noktaları ve Genel Değişim Eğilimi

📌 **Temmuz 2015'ten sonra, iki ayrı tarihte zirve (peak) yaptığı** gözlemlenmiştir.

📊 **Genel Eğilim:**

- 🔺 **İki belirgin zirve noktası** tespit edilmiştir.
- 🔄 **Yüksek frekanslı değişimler** gözlemlenmektedir, bu da veride ani dalgalanmalar olduğunu göstermektedir.

🗓️ Bu değişimlerin arkasındaki olası faktörleri anlamak için zaman serisi analizi yapılabilir.

In [37]:

```
df['year_month'] = df['Datetime'].dt.to_period('M')
monthly_shares = df.groupby('year_month')['Ride_Count'].sum().r

monthly_shares['year_month'] = monthly_shares['year_month'].as

fig = px.line(
    monthly_shares,
    x='year_month',
    y='Ride_Count',
    title="Sum of Bike Shares by Year-Month (Line Plot)",
    labels={"year_month": "Year-Month", "Ride_Count": "Bike Sha
    markers=True,
    template="plotly_dark",
)
fig.update_layout(
```

```
    title_font=dict(size=16, family="Arial", color="white"),
    title_x=0.5,
    title_y=0.97,
    xaxis_title_font=dict(size=12, family="Arial"),
    yaxis_title_font=dict(size=12, family="Arial"),
    xaxis=dict(tickangle=45),
    margin=dict(t=50, l=50, r=50, b=50),
    width=1200,
    height=700,
)

fig.show()
```

🚴 Yıl ve Ay Bazında Bisiklet Paylaşım Değişimi

📌 **Toplam bisiklet paylaşımlarının** yıl ve ay bazındaki değişimi, **interaktif bir çizgi grafiği (line plot)** ile görselleştirilmiştir.

📊 **Analiz Bulguları:**

- 📆 **Zaman içerisindeki trendler** net bir şekilde gözlemlenebilir.
- 📈 Mevsimsel dalgalanmalar ve dönemsel artış/azalışlar detaylı incelenebilir.

🔍 Daha derinlemesine analiz için grafik üzerindeki interaktif özellikler kullanılabilir.

In [39]:

```
data_yearly = df.groupby('Year')['Ride_Count'].mean().reset_ind


fig = px.pie(
    data_yearly,
    values='Ride_Count',
    names='Year',
    title="Yearly Bike Shares Distribution",
    template="plotly_dark",
)


fig.update_traces(
    pull=[0.0, 0.0, 0.1],
    marker=dict(
        colors=['#FF5733', '#33FF57', '#3357FF', '#FF33A1'],
        line=dict(color='#1E1E1E', width=2)
    ),
    textinfo='percent+label',
)

fig.update_layout(
    title_font=dict(size=16, family="Arial", color="white"),
    legend_title=dict(font=dict(size=12, family="Arial", color=
    width=800,
    height=600,
    margin=dict(t=50, l=50, r=50, b=50),
```

```
)

fig.show()
```

🏆 Yıllık Bisiklet Paylaşımlarının Dağılımı

📌 **Yıllık bisiklet paylaşımlarının dağılımı**, **interaktif bir pasta grafiği (pie chart)** ile görselleştirilmiştir.

📊 **Analiz Bulguları:**

- 📅 **Her yılın toplam paylaşıma katkısı** net bir şekilde görülebilir.
- 🎯 **Payların yüzdesel dağılımı**, belirli yıllarda artış veya azalış olup olmadığını anlamaya yardımcı olur.

🔍 Grafik üzerindeki interaktif özellikler sayesinde detaylar daha iyi incelenebilir.

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

In [42]:

```
data_monthly = df.groupby('Month')['Ride_Count'].mean().reset

fig = px.line(
    data_monthly,
    x='Month',
    y='Ride_Count',
    title="Monthly Bike Shares",
    labels={"Month": "Month", "Ride_Count": "Total Bike Share
    markers=True,
    template="plotly_dark",
)

fig.update_layout(
    title_font=dict(size=16, family="Arial", color="white"),
    xaxis_title_font=dict(size=12, family="Arial"),
    yaxis_title_font=dict(size=12, family="Arial"),
    margin=dict(t=50, l=50, r=50, b=50),
    legend_title=dict(font=dict(size=12, family="Arial")),
    width=1000,
    height=600,
    xaxis=dict(showgrid=True, gridwidth=1, gridcolor="gray"),
    yaxis=dict(showgrid=True, gridwidth=1, gridcolor="gray"),
)

fig.show()
```

📅 Aylara Göre Ortalama Bisiklet Paylaşımı

📌 **Aylara göre ortalama bisiklet paylaşımı** verisi, **interaktif bir çizgi grafiği (line plot)** ile görselleştirilmiştir.

📊 **Analiz Bulguları:**

- 🌱 **Bahar aylarında artış başlamakta**,
- 🌞 **Yaz aylarında zirveye ulaşmakta**,
- 🍂 **Sonbaharla birlikte düşüşe geçmektedir.**

🔍 Sezonluk değişimler bisiklet kullanım alışkanlıklarını anlamak için önemli ipuçları sunmaktadır.

In [44]:

```
data_seasonal = df.groupby('season_description')['Ride_Count'

# Tab10 renk paletinden 4 renk
colors = ['#ff7f0e', '#2ca02c', '#ffdb58', '#ffffff']  # tab1

# Create bar plot
fig = px.bar(
    data_seasonal,
    x='season_description',
    y='Ride_Count',
    title="Seasonal Bike Shares",
    labels={"season_description": "Season", "Ride_Count": "To
    template="plotly_dark",
    color='season_description',
    color_discrete_sequence=colors  # Tab10 renkleri ile özel
)

# Update layout
fig.update_layout(
    title_font=dict(size=16, family="Arial", color="white"),
    title_x=0.5,
    xaxis_title_font=dict(size=12, family="Arial"),
    yaxis_title_font=dict(size=12, family="Arial"),
    margin=dict(t=50, l=50, r=50, b=50),
    legend_title=dict(font=dict(size=12, family="Arial", colo
    width=1200,
    height=600,
    xaxis=dict(showgrid=True, gridwidth=1, gridcolor="gray"),
    yaxis=dict(showgrid=True, gridwidth=1, gridcolor="gray"),
)

fig.show()
```

🌎 Mevsimlere Göre Ortalama Bisiklet Paylaşımları

📌 **Mevsimlere göre ortalama bisiklet paylaşımları**, **interaktif bir sütun grafiği (bar chart)** ile görselleştirilmiştir.

📊 **Analiz Bulguları:**

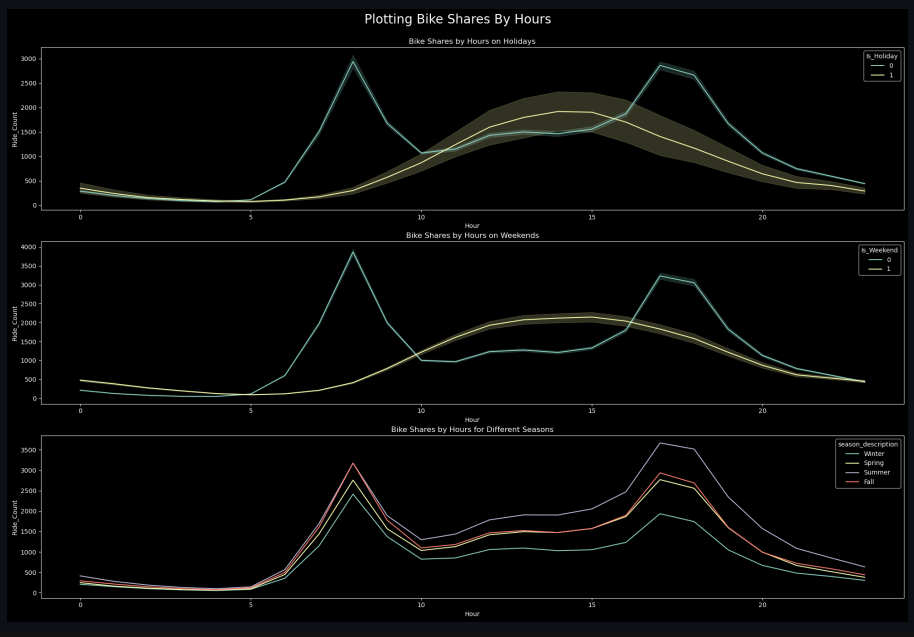
- 🌸 **İlkbahar** ile birlikte bisiklet kullanımı artmaya başlar.
- 🌞 **Yaz aylarında en yüksek seviyeye ulaşır.**
- 🍁 **Sonbaharda azalmaya başlar,**
- ❄️ **Kış aylarında ise en düşük seviyeye iner.**

📈 **Bu grafik, yukarıdaki çizgi grafiğini de desteklemektedir** ve bisiklet kullanımındaki mevsimsel trendleri net bir şekilde ortaya koymaktadır.

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

In [47]:

```
plt.style.use('dark_background')

fig = plt.figure(figsize=(20, 15))
fig.suptitle("Plotting Bike Shares By Hours", y=0.9, fontsiz

plt.subplot(3, 1, 1)
sns.lineplot(x=df.Hour, y=df.Ride_Count, hue=df.Is_Holiday)
plt.title("Bike Shares by Hours on Holidays", color='white')

plt.subplot(3, 1, 2)
sns.lineplot(x=df.Hour, y=df.Ride_Count, hue=df.Is_Weekend)
plt.title("Bike Shares by Hours on Weekends", color='white')

plt.subplot(3, 1, 3)
sns.lineplot(x=df.Hour, y=df.Ride_Count, hue=df.season_descr
plt.title("Bike Shares by Hours for Different Seasons", colo

plt.tight_layout()
plt.subplots_adjust(top=0.85)
plt.show()
```



🚴 Bisiklet Kullanım Alışkanlıkları

📌 **İnsanlar hafta içi ve tatil olmayan günlerde genellikle sabah ve akşam saatlerinde bisiklet kullanmayı tercih etmişlerdir.**

📊 **Analiz Bulguları:**

- ⏰ **Sabah ve akşam saatlerindeki yoğunluk**, insanların işe gidiş-geliş saatlerinde bisiklet kullandığını göstermektedir.
- 🗓️ **Hafta sonları ve tatil günlerinde**, bisiklet kullanım sayısında belirgin bir **artış** gözlemlenmektedir.

🔍 Bu durum, bisikletin sadece bir ulaşım aracı olarak değil, aynı zamanda bir **rekreasyon aracı** olarak da kullanıldığını göstermektedir.

In [49]:

```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

fig = make_subplots(rows=3, cols=1, subplot_titles=[
    "Bike Shares by Hours on Holidays",
    "Bike Shares by Hours on Weekends",
    "Bike Shares by Hours for Different Seasons"
])


for holiday in df['Is_Holiday'].unique():
    subset = df[df['Is_Holiday'] == holiday]
    fig.add_trace(
        go.Scatter(
            x=subset['Hour'],
            y=subset['Ride_Count'],
            mode='lines',
            name=f"Holiday: {holiday}",
            line=dict(width=2)
        ),
        row=1, col=1
    )


for weekend in df['Is_Weekend'].unique():
    subset = df[df['Is_Weekend'] == weekend]
    fig.add_trace(
        go.Scatter(
            x=subset['Hour'],
            y=subset['Ride_Count'],
            mode='lines',
            name=f"Weekend: {weekend}",
            line=dict(width=2)
        ),
        row=2, col=1
    )


for season in df['season_description'].unique():
```

```
    subset = df[df['season_description'] == season]
    fig.add_trace(
        go.Scatter(
            x=subset['Hour'],
            y=subset['Ride_Count'],
            mode='lines',
            name=f"Season: {season}",
            line=dict(width=2)
        ),
        row=3, col=1
    )


fig.update_layout(
    height=900,
    width=1200,
    title_text="Plotting Bike Shares By Hours",
    title_font_size=20,
    template='plotly_dark',
    showlegend=True
)

fig.update_xaxes(title_text="Hour", row=1, col=1)
fig.update_xaxes(title_text="Hour", row=2, col=1)
fig.update_xaxes(title_text="Hour", row=3, col=1)

fig.update_yaxes(title_text="Ride Count", row=1, col=1)
fig.update_yaxes(title_text="Ride Count", row=2, col=1)
fig.update_yaxes(title_text="Ride Count", row=3, col=1)


fig.show()
```

🔍 İnteraktif Grafik

📌 **Yukarıdaki grafiğin interaktif versiyonunu** inceleyebilirsiniz.

💻 Bu sürüm, veriyi daha detaylı incelemenizi ve farklı parametrelerle etkileşime girmenizi sağlar.

Plot bike shares by day of week.

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

In [52]:

```
def create_interactive_bike_share_by_day(df):
    df['Day_of_Week'] = df['Datetime'].dt.strftime('%A')

    bike_share_by_day = df.groupby('Day_of_Week')['Ride_Co

    bike_share_by_day['Day_of_Week'] = pd.Categorical(
        bike_share_by_day['Day_of_Week'],
        categories=['Monday', 'Tuesday', 'Wednesday', 'Thu
        ordered=True
    )

    bike_share_by_day = bike_share_by_day.sort_values('Day
```

```
    fig = px.bar(
        bike_share_by_day,
        x='Day_of_Week',
        y='Ride_Count',
        title='Bike Share by Day of the Week',
        labels={'Day_of_Week': 'Day of the Week', 'Ride_Co
        color='Ride_Count',
        color_continuous_scale='Viridis',
        template='plotly_dark',
        width=800,
        height=500
    )

    fig.update_layout(
        title_x=0.5,
        title_font=dict(size=16, family="Arial", color="wh
    )

    fig.show()

create_interactive_bike_share_by_day(df)
```

📅 Haftanın Günlerine Göre Ortalama Bisiklet Paylaşımları

📌 **Bu grafikte, haftanın günlerine göre ortalama bisiklet paylaşımları** gösterilmiştir.

📊 **Analiz Bulguları:**

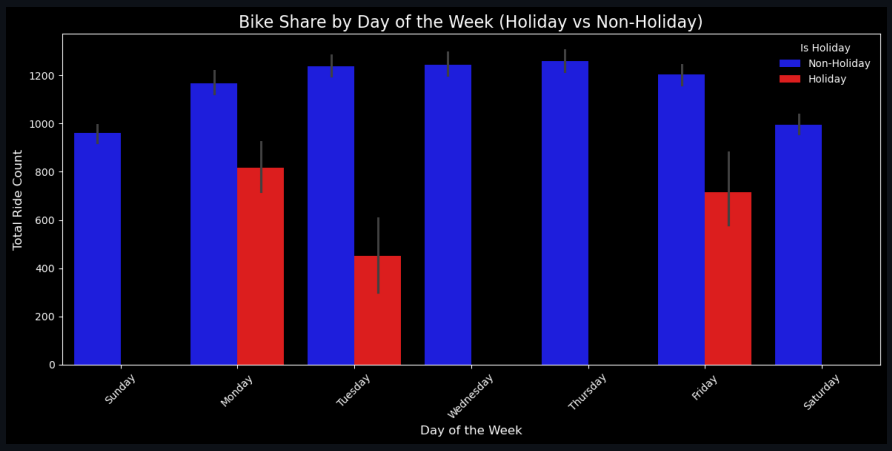
- 📈 **En yüksek kullanım, haftanın orta günlerine** denk gelmektedir.
- 🚲 Bu durum, **okul ve iş yerleri için ulaşımda bisikletin tercih edildiği** çıkarımını desteklemektedir.

🔍 Bu eğilim, bisikletin **günlük ulaşım aracı olarak yaygın kullanımını** vurgulamaktadır.

In [54]:

```
plt.style.use('dark_background')

plt.figure(figsize=(12, 6))


sns.barplot(
    data=df,
    x='Day_of_Week', y='Ride_Count', hue='Is_Holiday',
    palette={0: '#0000FF', 1: '#FF0000'}
)

plt.title('Bike Share by Day of the Week (Holiday vs Non-H
plt.xlabel('Day of the Week', fontsize=12, color='white')
plt.ylabel('Total Ride Count', fontsize=12, color='white')

handles, labels = plt.gca().get_legend_handles_labels()
plt.legend(handles, ['Non-Holiday', 'Holiday'], title='Is
```

```
plt.xticks(rotation=45, color='white')
plt.tight_layout()

# Göster
plt.show()
```



📊 Tatil ve Tatil Olmayan Günlerde Haftanın Günlerine Göre Bisiklet Paylaşım Kullanımı

📌 **Grafik, haftanın günlerine göre bisiklet paylaşım kullanımını, tatil (Holiday)** ve **tatil olmayan (Non-Holiday)** günler arasında karşılaştırmaktadır.

📊 **Analiz Bulguları:**

- 🚴 **Tatil olmayan günlerde**, bisiklet kullanımı genellikle daha yüksektir ve **hafta içi günlerinde (özellikle Salı ve Çarşamba)** zirve yapmaktadır.
- 🛑 **Tatil günlerinde**, kullanım belirgin şekilde düşmektedir, özellikle hafta içi günlerinde.
- 📉 **Hafta sonu kullanımda azalma** gözlemlenmektedir.

🔍 **Bu durum, tatil günlerinde ulaşım ihtiyaçlarının azalmasından veya farklı aktivitelerin tercih edilmesinden kaynaklanabilir.**

Plot bike shares by day of month

In [57]:

```
facet_data = df.groupby(['Day'])['Ride_Count'].mean().res

fig = px.line(
    facet_data,
    x='Day',
    y='Ride_Count',
    title='Bike Shares by Day of the Month',
    labels={'Day': 'Day of the Month', 'Ride_Count': 'Ave
```

```
    markers=True,
    line_shape='linear',
    template='plotly_dark',
    width=800,
    height=500
)


fig.update_xaxes(tickmode='array', tickvals=list(range(1,


fig.update_layout(
    title_font_size=16,
    title_font_family='Arial',
    xaxis_title='Day of the Month',
    yaxis_title='Average Ride Count',
)

fig.update_layout(
        title_x=0.5,
        title_font=dict(size=16, family="Arial", color="w
    )

fig.show()
```

📆 Ay Bazında Bisiklet Kullanım Eğilimleri

1️⃣ Genel Eğilim:

- 🏁 Ayın başında (1-10. günler arasında), **bisiklet kullanımı artış eğilimindedir**.
- 📈 **En yüksek ortalama kullanım**, yaklaşık 9. gün civarında gözlemlenmektedir.

2️⃣ Orta Bölüm (11-20. Günler):

- 🔄 Bu günler arasında, **dalgalı bir trend** görülmekte ve kullanım sayılarında belirgin bir **azalma veya artış** gözlemlenmemektedir.

3️⃣ Son Bölüm (21-31. Günler):

- 📉 Ayın sonlarına doğru, **ortalama kullanımda bir düşüş eğilimi** dikkat çekmektedir.
- 🛑 **31. günde**, oldukça **düşük bir değer** gözlemlenmiştir.

🔍 **Bu eğilimler**, ayın başında iş ve okul gibi rutin aktivitelerin etkisiyle daha fazla kullanım, orta dönemde ise istikrarsızlık ve ay sonuna doğru ise dinlenme ve tatil dönemleriyle azalacak şekilde şekillenmiş olabilir.

Plot bike shares by year

- Plot bike shares on holidays by seasons

In [59]:

```
yearly_data = round(df.groupby('Year')['Ride_Count'].mea

fig1 = px.bar(yearly_data, x='Year', y='Ride_Count',
              title="Average Bike Shares by Year",
              labels={"Year": "Year", "Ride_Count": "Ave
              color='Ride_Count', color_continuous_scale
              text='Ride_Count')

fig1.update_layout(
    paper_bgcolor="black",
    font=dict(color="white"),
    title_font=dict(color="white"),
    xaxis_title="Year",
    yaxis_title="Average Ride Count",
    title={"x": 0.5},
    margin=dict(t=50, b=50, l=50, r=50)
)

holiday_data = df[df['Is_Holiday'] == 1]
seasonal_data = holiday_data.groupby('season_description

fig2 = px.bar(seasonal_data, x='season_description', y=
              title="Bike Shares on Holidays by Seasons'
              labels={"season_description": "Season", "
              color='Ride_Count', color_continuous_scale
              text='Ride_Count')

fig2.update_layout(
    paper_bgcolor="black",
    plot_bgcolor="black",
    title_font=dict(color="white"),
    xaxis_title="Season",
    yaxis_title="Ride Count",
    title={"x": 0.5},
    margin=dict(t=50, b=50, l=50, r=50)
)

fig1.show()
fig2.show()
```

📊 Ortalama Bisiklet Paylaşımları Yıllık Bazda (Üstteki Grafik)

📌 **2015 Yılı:** Ortalama bisiklet kullanımı 1126.78 olarak kaydedilmiştir.

📌 **2016 Yılı:** Küçük bir artışla, 1164.45'e ulaşmıştır.

📌 **2017 Yılı: 523.33** ile dramatik bir düşüş yaşanmıştır.

🔍 **Yorum:** Bu düşüş, 2017 yılında kullanılan veri setinin eksikliği, operasyonel zorluklar ya da dışsal faktörlerden

(örneğin, hava durumu veya altyapı değişiklikleri) kaynaklanmış olabilir.

🚴 Tatil Günlerinde Bisiklet Kullanımı Mevsimlere Göre (Alttaki Grafik)

📌 **İlkbahar (Spring):** Tatil günlerinde bisiklet kullanımı 171,611 ile en yüksek seviyededir.
📌 **Yaz (Summer):** Bu değer 43,815'e düşerek en düşük kullanım seviyesini temsil etmektedir.
📌 **Kış (Winter):** 80,072 kullanımlık orta düzey bir performans görülmüştür.

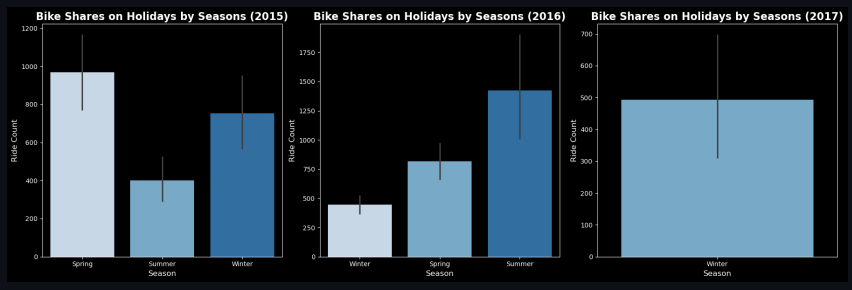
🔍 **Yorum:**

- 🌸 İlkbaharda bisiklet kullanımı için en uygun koşulların bulunması (ılıman hava, tatil aktiviteleri) bu yüksek seviyeyi açıklayabilir.
- 🌞 Yaz aylarındaki düşüş, aşırı sıcaklıklar veya tatil aktivitelerinin bisiklet kullanımını olumsuz etkilemesiyle ilgili olabilir.

In [60]:

```
holiday_data_2015 = df[(df['Year'] == 2015) & (df['Is_H
holiday_data_2016 = df[(df['Year'] == 2016) & (df['Is_H
holiday_data_2017 = df[(df['Year'] == 2017) & (df['Is_H

fig, axes = plt.subplots(1, 3, figsize=(18, 6))

# 2015 grafiği
sns.barplot(data=holiday_data_2015, x='season_descriptio
axes[0].set_title("Bike Shares on Holidays by Seasons (2
axes[0].set_xlabel("Season", fontsize=12)
axes[0].set_ylabel("Ride Count", fontsize=12)
axes[0].tick_params(axis='x', labelsize=10)
axes[0].tick_params(axis='y', labelsize=10)

# 2016 grafiği
sns.barplot(data=holiday_data_2016, x='season_descriptio
axes[1].set_title("Bike Shares on Holidays by Seasons (2
axes[1].set_xlabel("Season", fontsize=12)
axes[1].set_ylabel("Ride Count", fontsize=12)
axes[1].tick_params(axis='x', labelsize=10)
axes[1].tick_params(axis='y', labelsize=10)

# 2017 grafiği
sns.barplot(data=holiday_data_2017, x='season_descriptio
axes[2].set_title("Bike Shares on Holidays by Seasons (2
axes[2].set_xlabel("Season", fontsize=12)
axes[2].set_ylabel("Ride Count", fontsize=12)
axes[2].tick_params(axis='x', labelsize=10)
axes[2].tick_params(axis='y', labelsize=10)

plt.tight_layout()
```

```
plt.show()
```



🌱 Mevsimsel Bisiklet Kullanım Eğilimleri

1 2015 Yılı:

📌 **İlkbahar tatil günleri**, bisiklet kullanımı açısından **en yoğun sezon** olmuştur. Bu trend, yaz mevsiminde devam etmemiştir.

2 2016 Yılı:

📌 **Yaz sezonundaki ani artış**, bisiklet paylaşım sistemlerinin tatil günlerinde **yoğun olarak kullanıldığını** veya **yaz dönemine özgü etkinliklerin** etkili olduğunu gösterebilir.

3 2017 Yılı:

📌 **Eksik veriler**, mevsimsel karşılaştırmaları sınırlasa da, **kış sezonunda nispeten stabil** bir kullanım gözlemlenmiştir.

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

In [114...

```
weekend_data = df.groupby('Is_Weekend')['Ride_Count'].
weekend_data['Is_Weekend'] = weekend_data['Is_Weekend'

# Subplot oluşturma
fig = make_subplots(
    rows=1, cols=2,
    subplot_titles=('Bike Shares by Weekday/Weekend',
    specs=[[{"type": "pie"}, {"type": "bar"}]]
)

# Pie chart (ilk grafik)
fig.add_trace(
    go.Pie(
        labels=['Weekday', 'Weekend'],
        values=weekend_data['Ride_Count'],
        textinfo='percent+label',
        marker=dict(colors=['#FF0000', '#0000FF']),
```

```
            name='Distribution'
        ),
        row=1, col=1
)

# Bar chart (ikinci grafik)
fig.add_trace(
        go.Bar(
            x=weekend_data['Is_Weekend'],
            y=weekend_data['Ride_Count'],
            marker_color=['#FF0000', '#0000FF'],
            name='Counts'
        ),
        row=1, col=2
)

# Grafik düzenleme
fig.update_layout(
    title_text='Bike Shares: Weekday vs. Weekend',
    plot_bgcolor='black',
    paper_bgcolor='black',
    font=dict(color='white'),
    title_font=dict(size=18, color='white'),
    showlegend=False
)
fig.update_layout(
            title_x=0.5,
            title_font=dict(size=20, family="Arial", color
        )

fig.show()
```

🚴 Bisiklet Paylaşım Sistemi: Haftaiçi ve Haftasonu Kullanım İstatistikleri

📊 **Grafikte Görülen Veriler:**

1 Haftaiçi Kullanım:

- **%75.6**'lık büyük bir kısmı haftaiçi (weekday) gerçekleşmektedir.
- Çubuk grafikte, haftaiçi kullanımın yaklaşık **12-13 milyon civarında** olduğu gözlemlenmektedir.

2 Haftasonu Kullanım:

- **%24.4**'lük kısım ise haftasonu (weekend) gerçekleşmektedir.
- Haftasonu kullanımının ise yaklaşık **4-5 milyon civarında** olduğu tespit edilmiştir.

🔍 **Yorum:**

- Bu dağılım, bisiklet paylaşım sisteminin **öncelikle**

iş/okul gibi **haftaiçi aktiviteleri** için bir ulaşım aracı olarak kullanıldığını göstermektedir.

- Haftasonu kullanımının görece düşük olması, sistemin **rekreasyonel amaçlardan ziyade günlük ulaşım ihtiyaçları** için tercih edildiğine işaret etmektedir.

Plot the distribution of weather code by seasons

In [64]:
```
df['season_description'] = df['season_description'].r

# Plotly ile interaktif histogram
fig = px.histogram(
    df,
    x='season_description',
    color='Weather_Description',
    barmode='group',
    color_discrete_sequence=px.colors.qualitative.Set
    title="Distribution of Weather Codes by Seasons"
)

# Grafik düzenleme
fig.update_layout(
    xaxis_title="Season",
    yaxis_title="Count",
    legend_title_text="Weather Code",
    plot_bgcolor='black',
    paper_bgcolor='black',
    font=dict(color='white'),
    title_font=dict(size=18, color='white'),
    legend=dict(x=1.05, y=1)
)

fig.update_layout(
        title_x=0.5,
        title_font=dict(size=20, family="Arial", colo
    )

fig.show()
```

🌦️ Mevsimlere Göre Hava Durumu Kodu Dağılımı

🔍 Öne Çıkan Noktalar:

1️⃣ Açık/Sisli/Puslu Hava (Kod 1):

- **Tüm mevsimlerde en yüksek frekansa sahip** olup, özellikle **ilkbahar** ve **kış mevsimlerinde** yaklaşık **1500 kez** gözlemlenmiştir.

2️⃣ Parçalı Bulutlu Hava (Kod 2):

- **Tüm mevsimlerde nispeten istikrarlı** bir dağılım göstermekte ve **yaklaşık 800-1000** arasında seyretmektedir.

3️⃣ **Kapalı Hava (Kod 4) ve Yağmurlu/Hafif Yağışlı Durumlar (Kod 7):**

- Diğer hava koşullarına göre daha **az sıklıkta** gözlemlenmiştir.

4️⃣ **Kar Yağışı (Kod 26) ve Gök Gürültülü Yağmur (Kod 10):**

- **En az gözlemlenen** hava durumlarıdır.

🌍 Yorum:

- Bu dağılım, bölgenin genel olarak **açık ve parçalı bulutlu** bir iklime sahip olduğunu ve **aşırı hava olaylarının** ise daha nadir gerçekleştiğini göstermektedir.

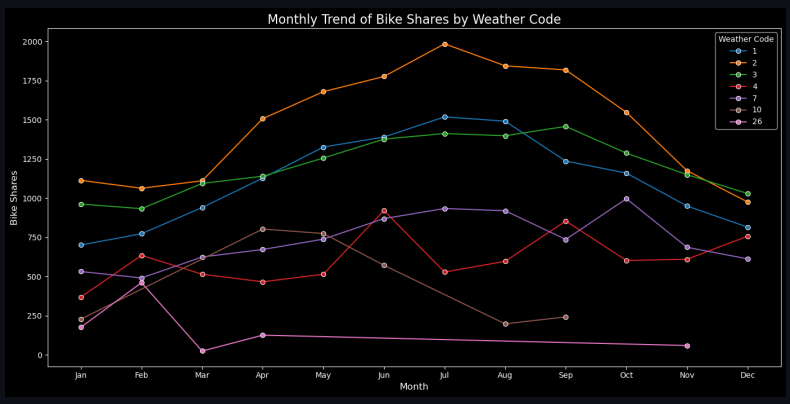
Feel free to include any additional analyses.

In [66]:

```
fig = px.line(
    df,
    x='Month',
    y='Ride_Count',
    color='Weather_Code',
    markers=True,
    title="Monthly Trend of Bike Shares by Weather (
    color_discrete_sequence=px.colors.qualitative.S
)

fig.update_layout(
    xaxis=dict(
        tickmode='array',
        tickvals=list(range(1, 13)),
        ticktext=["Jan", "Feb", "Mar", "Apr", "May"
    ),
    xaxis_title="Month",
    yaxis_title="Bike Shares",
    legend_title_text="Weather Code",
    plot_bgcolor='black',
    paper_bgcolor='black',
    font=dict(color='white'),
    title_font=dict(size=18, color='white')
)
fig.update_layout(
        title_x=0.5,
        title_font=dict(size=16, family="Arial", co
    )
fig.show()
```

In [67]:

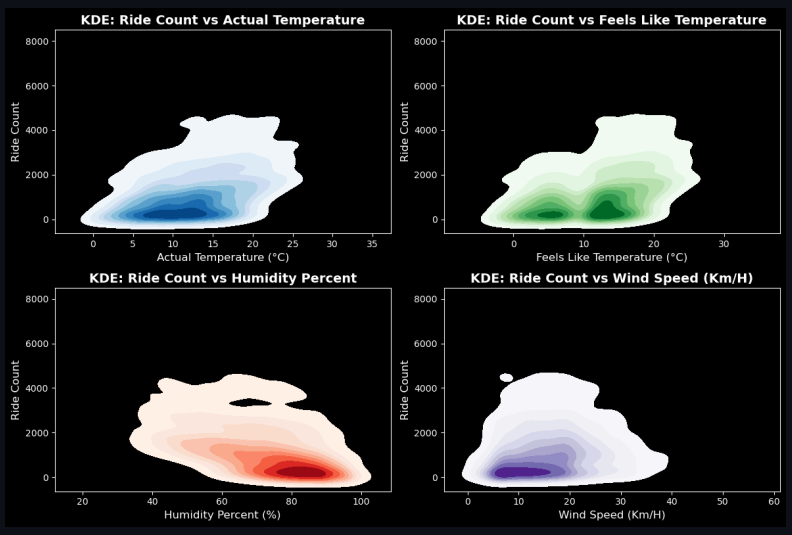
```
plt.figure(figsize=(14, 7))

plt.style.use('dark_background')


sns.lineplot(data=df, x='Month', y='Ride_Count', hu

plt.title("Monthly Trend of Bike Shares by Weather
plt.xlabel("Month", fontsize=12)
plt.ylabel("Bike Shares", fontsize=12)
plt.xticks(ticks=range(1, 13), labels=["Jan", "Feb"

plt.legend(title="Weather Code", loc='upper right')


plt.tight_layout()

plt.show()
```



In [68]:

```
plt.style.use('dark_background')

plt.figure(figsize=(12, 8))

plt.subplot(2, 2, 1)
sns.kdeplot(x=df['Actual_Temperature'], y=df['Ride_
plt.title('KDE: Ride Count vs Actual Temperature',
plt.xlabel('Actual Temperature (°C)', fontsize=12,
plt.ylabel('Ride Count', fontsize=12, color='white'

plt.subplot(2, 2, 2)
sns.kdeplot(x=df['Feels_Like_Temperature'], y=df['R
plt.title('KDE: Ride Count vs Feels Like Temperatur
plt.xlabel('Feels Like Temperature (°C)', fontsize=
plt.ylabel('Ride Count', fontsize=12, color='white'

plt.subplot(2, 2, 3)
sns.kdeplot(x=df['Humidity_Percent'], y=df['Ride_Co
plt.title('KDE: Ride Count vs Humidity Percent', fo
plt.xlabel('Humidity Percent (%)', fontsize=12, col
plt.ylabel('Ride Count', fontsize=12, color='white'

plt.subplot(2, 2, 4)
sns.kdeplot(x=df['Wind_Speed_Km/H'], y=df['Ride_Cou
plt.title('KDE: Ride Count vs Wind Speed (Km/H)', f
plt.xlabel('Wind Speed (Km/H)', fontsize=12, color=
plt.ylabel('Ride Count', fontsize=12, color='white'
```

```
plt.tight_layout()


plt.show()
```



Conclusions

📊 Sonuçlar

- "Bisiklet Talebi Görselleştirme Projesi," görüselleştirme teknikleri ve veri analizi kullanarak bisiklet paylaşım talebini etkileyen faktörleri etkili bir şekilde incelemektedir. Projeden çıkarılan temel bulgular şunlardır:

🌱 Mevsimsel ve Zamansal Desenler:

- Analiz, bisiklet talebinde belirgin zamansal eğilimler göstermiştir; belirli aylar ve mevsimlerde daha yüksek kullanım gözlemlenmiştir. Yaz aylarında bisiklet kullanımı daha yoğun olup bu durum, hava koşullarının elverişliliği ile uyumludur.
- Hafta içi ve hafta sonu talep desenleri, hafta içi iş günleri ve hafta sonlarındaki eğlence faaliyetlerini yansıtarak farklılıklar göstermektedir.

🌦️ Hava Koşullarının Etkisi:

- Hava koşulları, özellikle sıcaklık, nem ve yağış, bisiklet kullanımını önemli ölçüde etkilemektedir.

Daha sıcak havalar, daha yüksek taleple pozitif bir korelasyon gösterirken, ağır yağmur gibi olumsuz hava koşulları kullanımda belirgin bir düşüşe yol açmaktadır.

🚲 Kullanıcı Davranışının Analizi:

- Kullanıcılar arasındaki kullanım desenleri farklılıklar göstermektedir. Kullanıcılar, hafta içi talebine büyük ölçüde katkı sağlarken, bu durum büyük olasılıkla günlük iş güzergahlarından kaynaklanmaktadır. Buna karşın, kullanıcılar hafta sonları ve tatil günlerinde daha az aktif olma eğilimi göstermektedir.

🎉 Tatiller ve Özel Etkinliklerin Etkisi:

- Veriler, tatiller ve hafta sonu sırasında bisiklet kullanımının azalış gösterdiğini ortaya koymuştur. Bu durum, tatil günlerinde ulaşım ihtiyaçlarının azalmasından veya farklı aktivitelerin tercih edilmesinden kaynaklanabilir.