Step 2: Import Libraries and Load Data

In [1]:
```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load data
bike_data = pd.read_csv('merged_citibike_weather.csv')  # Replace with the actual file path
weather_data = pd.read_csv('weather_2022.csv')  # Replace with the actual file path

# Display the first few rows to ensure the data loaded correctly
print(bike_data.head())
print(weather_data.head())
```

```
C:\Users\Asus\AppData\Local\Temp\ipykernel_18704\2188774527.py:6: DtypeWarning: Columns (5,7) have mixed types. Specify dtype option on import or set low_memory=False.
  bike_data = pd.read_csv('merged_citibike_weather.csv')  # Replace with the actual file path
           ride_id  rideable_type               started_at  \
0  BFD29218AB271154  electric_bike  2022-01-21 13:13:43.392
1  7C953F2FD7BE1302   classic_bike  2022-01-10 11:30:54.162
2  95893ABD40CED4B8  electric_bike  2022-01-26 10:52:43.096
3  F853B50772137378   classic_bike  2022-01-03 08:35:48.247
4  7590ADF834797B4B   classic_bike  2022-01-22 14:14:23.043

                  ended_at       start_station_name start_station_id  \
0  2022-01-21 13:22:31.463  West End Ave & W 107 St          7650.05
1  2022-01-10 11:41:43.422             4 Ave & 3 St          4028.04
2  2022-01-26 11:06:35.227          1 Ave & E 62 St          6753.08
3  2022-01-03 09:10:50.475          2 Ave & E 96 St          7338.02
4  2022-01-22 14:34:57.474          6 Ave & W 34 St           6364.1

              end_station_name end_station_id  start_lat  start_lng  \
0  Mt Morris Park W & W 120 St        7685.14  40.802117 -73.968181
1      Boerum Pl\t& Pacific St        4488.09  40.673746 -73.985649
2              5 Ave & E 29 St        6248.06  40.761227 -73.960940
3              5 Ave & E 29 St        6248.06  40.783964 -73.947167
4              5 Ave & E 29 St        6248.06  40.749640 -73.988050

     end_lat    end_lng member_casual  start_date      STATION        DATE  \
0  40.804038 -73.945925        member  2022-01-21  USW00094728  2022-01-21
1  40.688489 -73.991160        member  2022-01-10  USW00094728  2022-01-10
2  40.745168 -73.986831        member  2022-01-26  USW00094728  2022-01-26
3  40.745168 -73.986831        member  2022-01-03  USW00094728  2022-01-03
4  40.745168 -73.986831        member  2022-01-22  USW00094728  2022-01-22

   PRCP  TMAX   TMIN
0   0.0 -55.0  -99.0
1   0.0  44.0  -43.0
2   0.0 -21.0  -66.0
3   0.0  28.0  -55.0
4   0.0 -16.0 -105.0
       STATION        DATE  PRCP  TMAX  TMIN
0  USW00094728  2022-01-01   201   133   100
1  USW00094728  2022-01-02    10   150    28
2  USW00094728  2022-01-03     0    28   -55
3  USW00094728  2022-01-04     0    11   -71
4  USW00094728  2022-01-05    58    83    -5
```

Step 3: Set a Theme and Color Palette

In [2]:
```python
sns.set_theme(style="whitegrid", palette="muted")
```

Step 4: Bar Chart of Top 20 Starting Station Frequencies

In [5]:
```python
top_stations = bike_data['start_station_name'].value_counts().head(20)
```

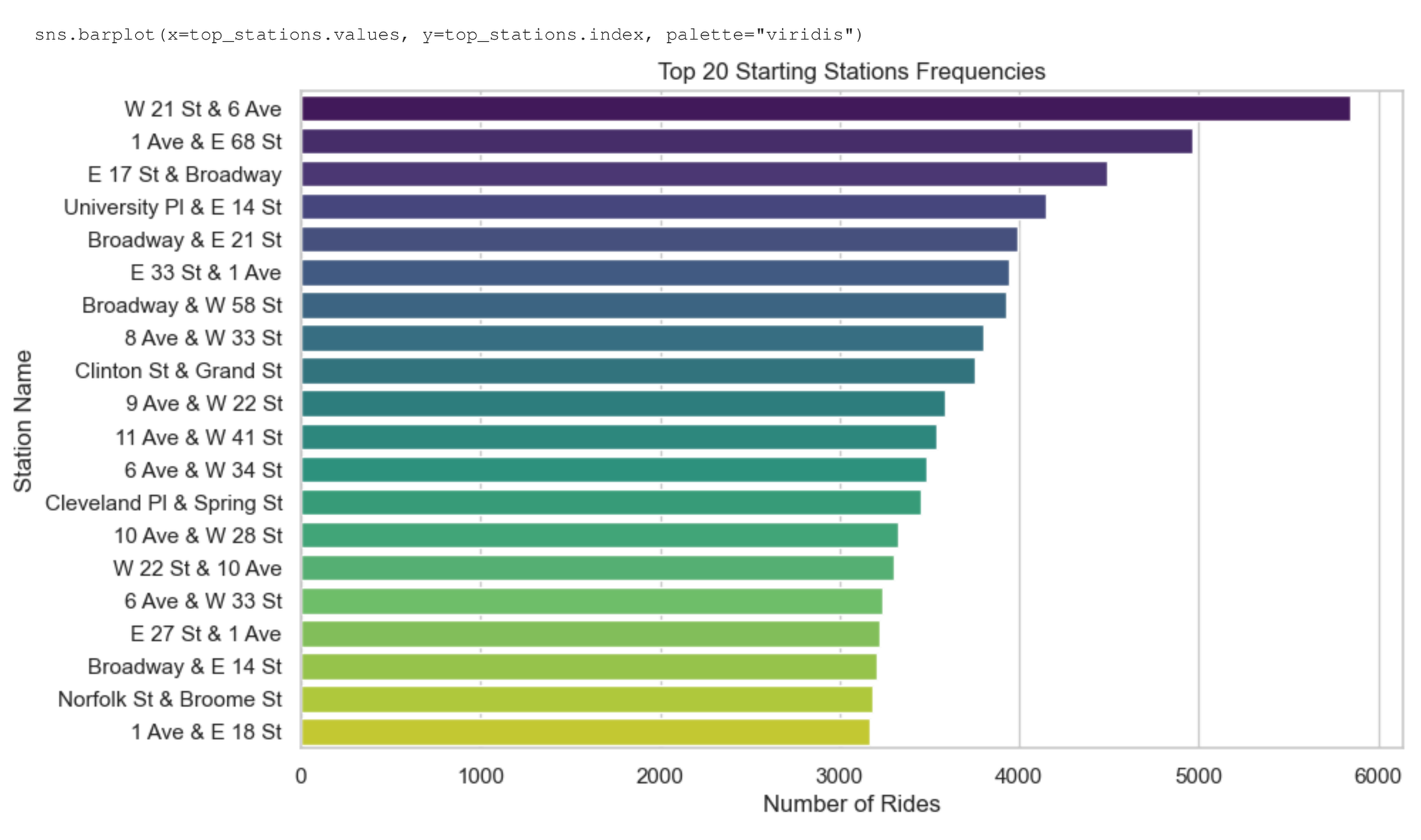
In [6]:
```python
print(top_stations)
```

```
start_station_name
W 21 St & 6 Ave            5846
1 Ave & E 68 St            4964
E 17 St & Broadway         4490
University Pl & E 14 St    4154
Broadway & E 21 St         3990
E 33 St & 1 Ave            3948
Broadway & W 58 St         3931
8 Ave & W 33 St            3799
Clinton St & Grand St      3758
9 Ave & W 22 St            3587
11 Ave & W 41 St           3544
6 Ave & W 34 St            3484
Cleveland Pl & Spring St   3451
10 Ave & W 28 St           3325
W 22 St & 10 Ave           3305
6 Ave & W 33 St            3243
E 27 St & 1 Ave            3221
Broadway & E 14 St         3206
Norfolk St & Broome St     3181
1 Ave & E 18 St            3168
Name: count, dtype: int64
```

Create a bar chart:

In [7]:
```python
plt.figure(figsize=(10, 6))
sns.barplot(x=top_stations.values, y=top_stations.index, palette="viridis")
plt.title("Top 20 Starting Stations Frequencies")
plt.xlabel("Number of Rides")
plt.ylabel("Station Name")
plt.show()
```

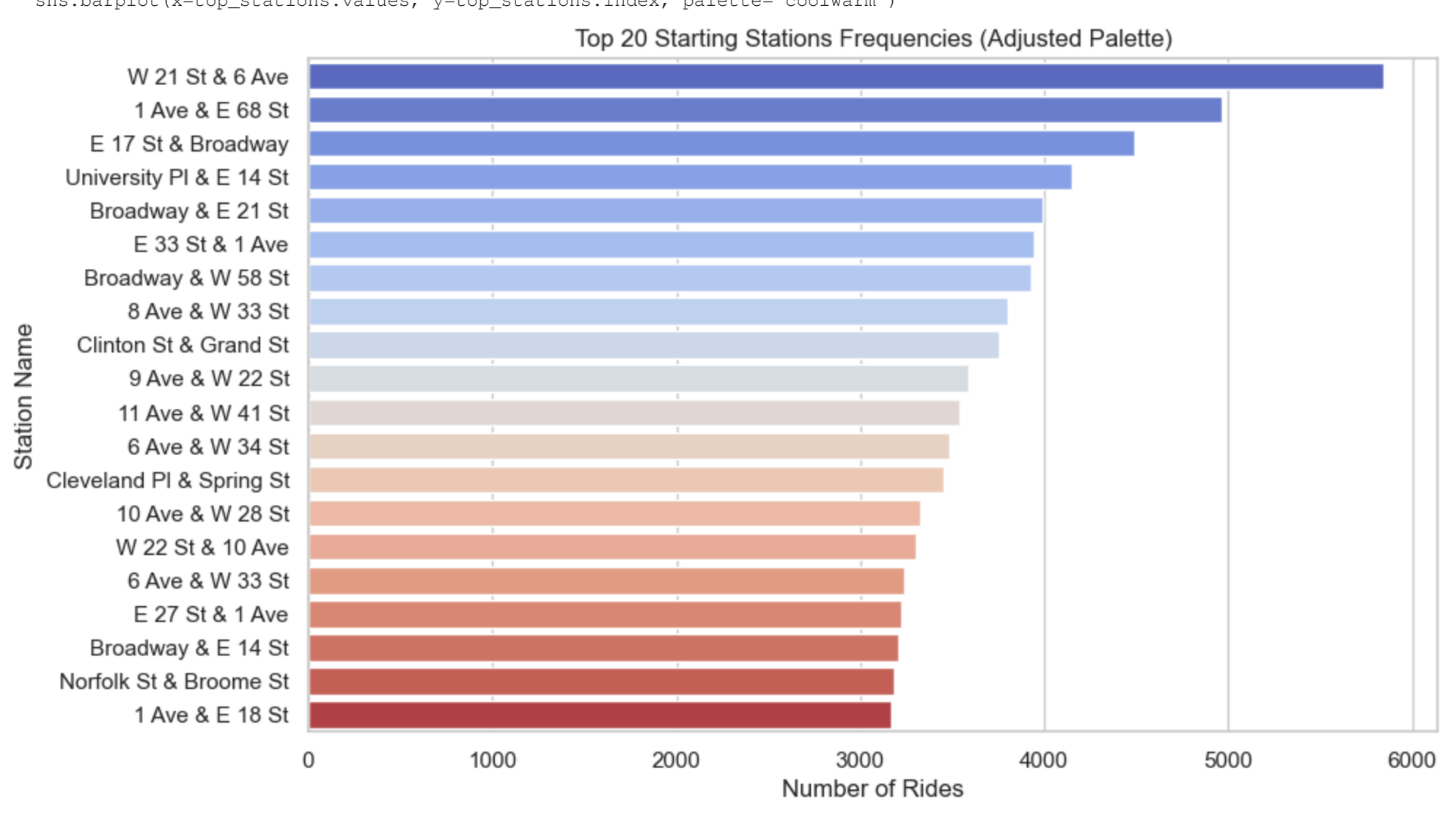
```
C:\Users\Asus\AppData\Local\Temp\ipykernel_18704\3239719016.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=top_stations.values, y=top_stations.index, palette="viridis")
```



Adjust the palette

In [8]:
```python
plt.figure(figsize=(10, 6))
sns.barplot(x=top_stations.values, y=top_stations.index, palette="coolwarm")
plt.title("Top 20 Starting Stations Frequencies (Adjusted Palette)")
plt.xlabel("Number of Rides")
plt.ylabel("Station Name")
plt.show()
```

```
C:\Users\Asus\AppData\Local\Temp\ipykernel_18704\224794803.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=top_stations.values, y=top_stations.index, palette="coolwarm")
```



Step 5: Recreate Dual-Axis Line Plot

In [10]:
```python
print(bike_data.columns)
print(weather_data.columns)
```

```
Index(['ride_id', 'rideable_type', 'started_at', 'ended_at',
       'start_station_name', 'start_station_id', 'end_station_name',
       'end_station_id', 'start_lat', 'start_lng', 'end_lat', 'end_lng',
       'member_casual', 'start_date', 'STATION', 'DATE', 'PRCP', 'TMAX',
       'TMIN'],
      dtype='object')
Index(['STATION', 'DATE', 'PRCP', 'TMAX', 'TMIN'], dtype='object')
```

In [11]:
```python
# Rename date columns to a common name ('date')
bike_data.rename(columns={'start_date': 'date'}, inplace=True)
weather_data.rename(columns={'DATE': 'date'}, inplace=True)
```

In [12]:
```python
merged_data = pd.merge(bike_data, weather_data, on='date')
```

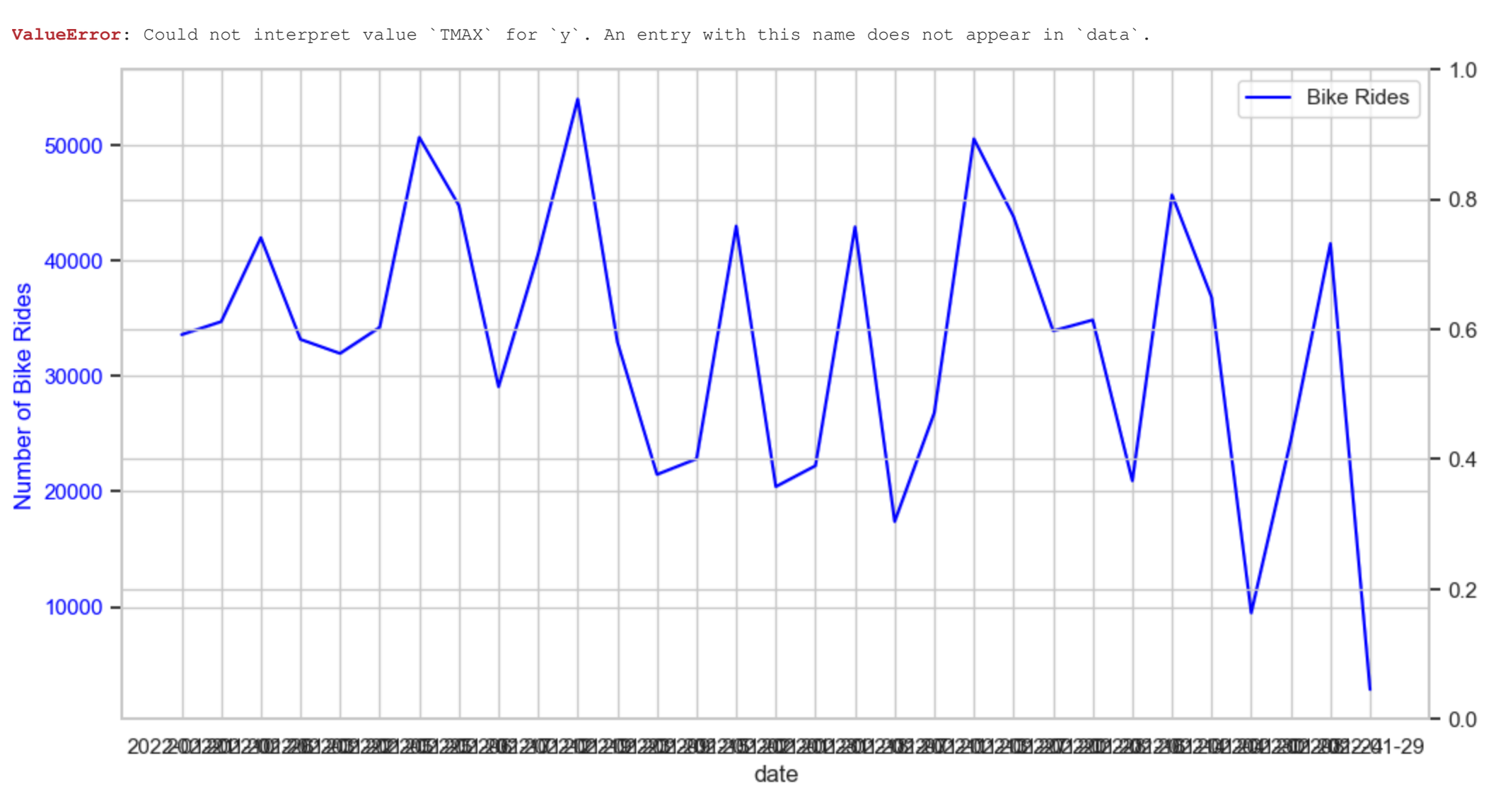
Create the dual-axis line plot:

In [17]:
```python
print(merged_data.columns)
```

```
Index(['ride_id', 'rideable_type', 'started_at', 'ended_at',
       'start_station_name', 'start_station_id', 'end_station_name',
       'end_station_id', 'start_lat', 'start_lng', 'end_lat', 'end_lng',
       'member_casual', 'date', 'STATION_x', 'DATE', 'PRCP_x', 'TMAX_x',
       'TMIN_x', 'STATION_y', 'PRCP_y', 'TMAX_y', 'TMIN_y', 'ride_count'],
      dtype='object')
```

In [19]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Group by date and count the rides (assuming ride_id is unique for each ride)
merged_data['ride_count'] = merged_data.groupby('date')['ride_id'].transform('count')

# Now, plot the data
fig, ax1 = plt.subplots(figsize=(12, 6))

# Plot the bike rides count
sns.lineplot(data=merged_data, x='date', y='ride_count', ax=ax1, label="Bike Rides", color='blue')
ax1.set_ylabel('Number of Bike Rides', color='blue')
ax1.tick_params(axis='y', labelcolor='blue')

# Create a second y-axis for the temperature data
ax2 = ax1.twinx()
sns.lineplot(data=merged_data, x='date', y='TMAX', ax=ax2, label="Temperature", color='red')  # Assuming TMAX is the column for temperature
ax2.set_ylabel('Temperature (°C)', color='red')
ax2.tick_params(axis='y', labelcolor='red')

# Add title and legend
plt.title("Bike Rides and Temperature Trends")
fig.legend(loc="upper left", bbox_to_anchor=(0.1, 0.9))

# Show the plot
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[19], line 18
     16 # Create a second y-axis for the temperature data
     17 ax2 = ax1.twinx()
---> 18 sns.lineplot(data=merged_data, x='date', y='TMAX', ax=ax2, label="Temperature", color='red')  # Assuming TMAX is the column for temperature
     19 ax2.set_ylabel('Temperature (°C)', color='red')
     20 ax2.tick_params(axis='y', labelcolor='red')

File ~\anaconda3\New folder\envs\20th_century\Lib\site-packages\seaborn\relational.py:485, in lineplot(data, x, y, hue, size, style, units, weights, palette, hue_order, hue_norm, sizes, size_order, size_norm, dashes, markers, style_order, estimator, errorbar, n_boot, seed, orient, sort, err_style, err_kws, legend, ci, ax, **kwargs)
    471 def lineplot(
    472     data=None, *,
    473     x=None, y=None, hue=None, size=None, style=None, units=None, weights=None,
   (...)
    481
    482     # Handle deprecation of ci parameter
    483     errorbar = _deprecate_ci(errorbar, ci)
--> 485     p = _LinePlotter(
    486         data=data,
    487         variables=dict(
    488             x=x, y=y, hue=hue, size=size, style=style, units=units, weight=weights
    489         ),
    490         estimator=estimator, n_boot=n_boot, seed=seed, errorbar=errorbar,
    491         sort=sort, orient=orient, err_style=err_style, err_kws=err_kws,
    492         legend=legend,
    493     )
    495     p.map_hue(palette=palette, order=hue_order, norm=hue_norm)
    496     p.map_size(sizes=sizes, order=size_order, norm=size_norm)

File ~\anaconda3\New folder\envs\20th_century\Lib\site-packages\seaborn\relational.py:216, in _LinePlotter.__init__(self, data, variables, estimator, n_boot, seed, errorbar, sort, orient, err_style, err_kws, legend)
    202 def __init__(
    203     self, *,
    204     data=None, variables={},
   (...)
    210     # the kind of plot to draw, but for the time being we need to set
    211     # this information so the SizeMapping can use it
    212     self._default_size_range = (
    213         np.r_[.5, 2] * mpl.rcParams["lines.linewidth"]
    214     )
--> 216     super().__init__(data=data, variables=variables)
    218     self.estimator = estimator
    219     self.errorbar = errorbar

File ~\anaconda3\New folder\envs\20th_century\Lib\site-packages\seaborn\_base.py:634, in VectorPlotter.__init__(self, data, variables)
    629 # var_ordered is relevant only for categorical axis variables, and may
    630 # be better handled by an internal axis information object that tracks
    631 # such information and is set up by the scale_* methods. The analogous
    632 # information for numeric axes would be information about log scales.
    633 self._var_ordered = {"x": False, "y": False}  # alt., used DefaultDict
--> 634 self.assign_variables(data, variables)
    636 # TODO Lots of tests assume that these are called to initialize the
    637 # mappings to default values on class initialization. I'd prefer to
    638 # move away from that and only have a mapping when explicitly called.
    639 for var in ["hue", "size", "style"]:

File ~\anaconda3\New folder\envs\20th_century\Lib\site-packages\seaborn\_base.py:679, in VectorPlotter.assign_variables(self, data, variables)
    674 else:
    675     # When dealing with long-form input, use the newer PlotData
    676     # object (internal but introduced for the objects interface)
    677     # to centralize / standardize data consumption logic.
    678     self.input_format = "long"
--> 679     plot_data = PlotData(data, variables)
    680     frame = plot_data.frame
    681     names = plot_data.names

File ~\anaconda3\New folder\envs\20th_century\Lib\site-packages\seaborn\_core\data.py:58, in PlotData.__init__(self, data, variables)
     51 def __init__(
     52     self,
     53     data: DataSource,
     54     variables: dict[str, VariableSpec],
     55 ):
     57     data = handle_data_source(data)
---> 58     frame, names, ids = self._assign_variables(data, variables)
     60     self.frame = frame
     61     self.names = names

File ~\anaconda3\New folder\envs\20th_century\Lib\site-packages\seaborn\_core\data.py:232, in PlotData._assign_variables(self, data, variables)
    230     else:
    231         err += "An entry with this name does not appear in `data`."
--> 232     raise ValueError(err)
    234 else:
    235
    236     # Otherwise, assume the value somehow represents data
    237
    238     # Ignore empty data structures
    239     if isinstance(val, Sized) and len(val) == 0:

ValueError: Could not interpret value `TMAX` for `y`. An entry with this name does not appear in `data`.
```



In [20]:
```python
print(merged_data.columns)
```

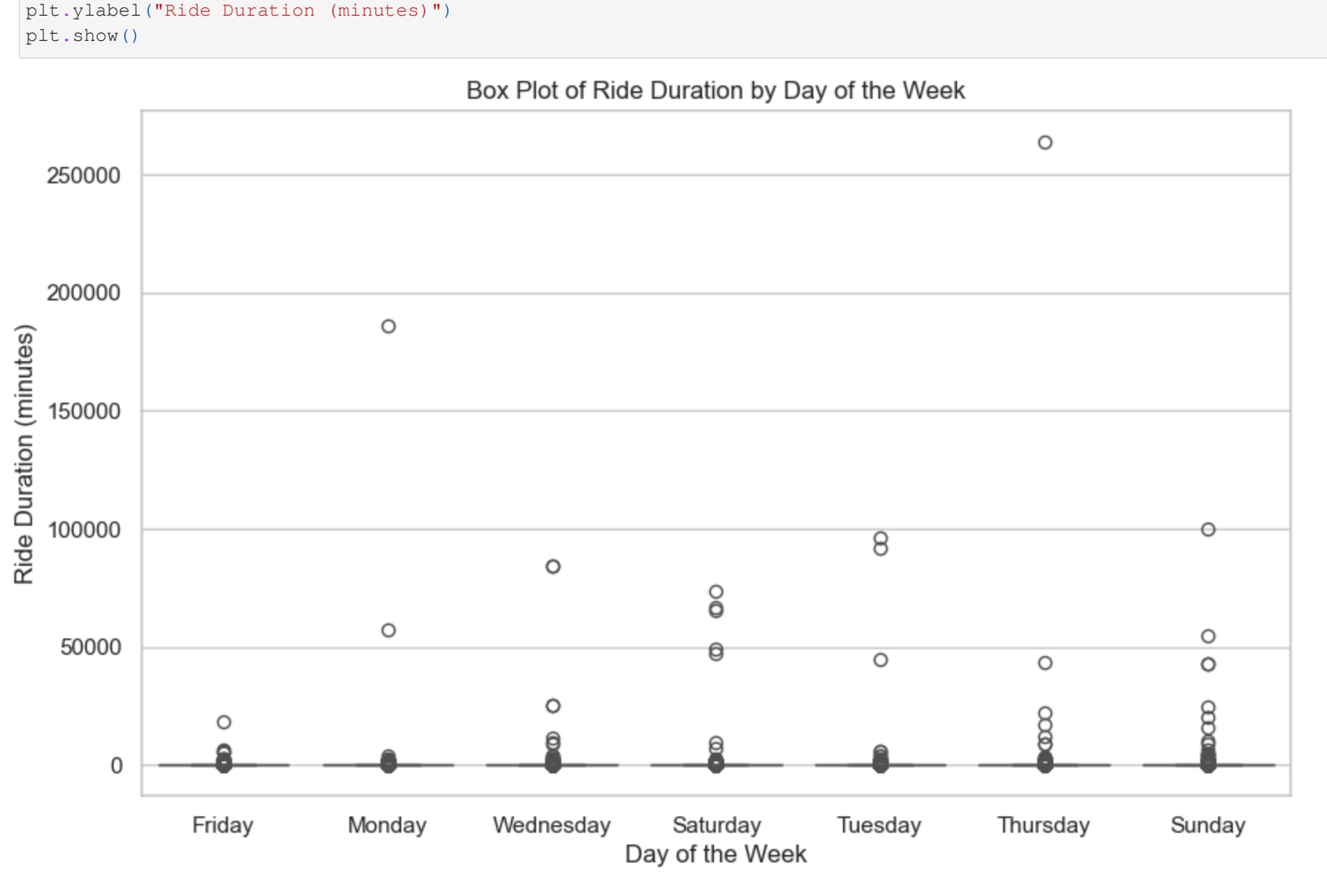
```
Index(['ride_id', 'rideable_type', 'started_at', 'ended_at',
       'start_station_name', 'start_station_id', 'end_station_name',
       'end_station_id', 'start_lat', 'start_lng', 'end_lat', 'end_lng',
       'member_casual', 'date', 'STATION_x', 'DATE', 'PRCP_x', 'TMAX_x',
       'TMIN_x', 'STATION_y', 'PRCP_y', 'TMAX_y', 'TMIN_y', 'ride_count'],
      dtype='object')
```

In [23]:
```python
print(merged_data.head())
```

```
           ride_id  rideable_type               started_at  \
0  BFD29218AB271154  electric_bike  2022-01-21 13:13:43.392
1  7C953F2FD7BE1302   classic_bike  2022-01-10 11:30:54.162
2  95893ABD40CED4B8  electric_bike  2022-01-26 10:52:43.096
3  F853B50772137378   classic_bike  2022-01-03 08:35:48.247
4  7590ADF834797B4B   classic_bike  2022-01-22 14:14:23.043

                  ended_at       start_station_name start_station_id  \
0  2022-01-21 13:22:31.463  West End Ave & W 107 St          7650.05
1  2022-01-10 11:41:43.422             4 Ave & 3 St          4028.04
2  2022-01-26 11:06:35.227          1 Ave & E 62 St          6753.08
3  2022-01-03 09:10:50.475          2 Ave & E 96 St          7338.02
4  2022-01-22 14:34:57.474          6 Ave & W 34 St           6364.1

              end_station_name end_station_id  start_lat  start_lng  ...  \
0  Mt Morris Park W & W 120 St        7685.14  40.802117 -73.968181  ...
1      Boerum Pl\t& Pacific St        4488.09  40.673746 -73.985649  ...
2              5 Ave & E 29 St        6248.06  40.761227 -73.960940  ...
3              5 Ave & E 29 St        6248.06  40.783964 -73.947167  ...
4              5 Ave & E 29 St        6248.06  40.749640 -73.988050  ...

     STATION_x        DATE PRCP_x TMAX_x TMIN_x    STATION_y  PRCP_y  TMAX_y  \
0  USW00094728  2022-01-21    0.0  -55.0  -99.0  USW00094728       0     -55
1  USW00094728  2022-01-10    0.0   44.0  -43.0  USW00094728       0      44
2  USW00094728  2022-01-26    0.0  -21.0  -66.0  USW00094728       0     -21
3  USW00094728  2022-01-03    0.0   28.0  -55.0  USW00094728       0      28
4  USW00094728  2022-01-22    0.0  -16.0 -105.0  USW00094728       0     -16

   TMIN_y ride_count
0     -99      33598
1     -43      34727
2     -66      42005
3     -55      33189
4    -105      31969

[5 rows x 24 columns]
```

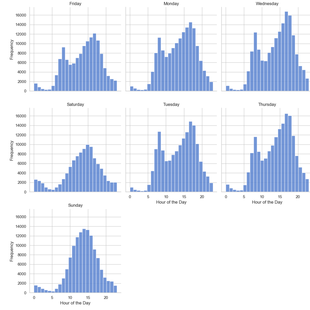
Step 6: Box Plot of a Categorical Variable

In [26]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Ensure date columns are in datetime format
bike_data['started_at'] = pd.to_datetime(bike_data['started_at'])
bike_data['ended_at'] = pd.to_datetime(bike_data['ended_at'])

# Create a 'day_of_week' column based on the 'started_at' column
bike_data['day_of_week'] = bike_data['started_at'].dt.day_name()

# Calculate the ride duration in minutes
bike_data['ride_duration'] = (bike_data['ended_at'] - bike_data['started_at']).dt.total_seconds() / 60

# Plotting the box plot for ride duration by day of the week
plt.figure(figsize=(10, 6))
sns.boxplot(x='day_of_week', y='ride_duration', data=bike_data)
plt.title("Box Plot of Ride Duration by Day of the Week")
plt.xlabel("Day of the Week")
plt.ylabel("Ride Duration (minutes)")
plt.show()
```



Step 7: Create a FacetGrid

In [28]:
```python
# Ensure that 'started_at' is in datetime format
bike_data['started_at'] = pd.to_datetime(bike_data['started_at'])

# Extract the hour from the 'started_at' column
bike_data['hour'] = bike_data['started_at'].dt.hour

# Now you can use FacetGrid to plot the histogram
import seaborn as sns
import matplotlib.pyplot as plt

g = sns.FacetGrid(bike_data, col="day_of_week", col_wrap=3, height=4)
g.map(sns.histplot, "hour", bins=24)
g.set_titles("{col_name}")
g.set_axis_labels("Hour of the Day", "Frequency")
plt.show()
```


Friday
Monday
Wednesday
Pickups
0
2000
4000
6000
8000
10000
12000
14000
16000

Saturday
Tuesday
Thursday
Pickups
0
2000
4000
6000
8000
10000
12000
14000
16000
0
5
10
15
20
Hour of the Day

Sunday
Pickups
0
2000
4000
6000
8000
10000
12000
14000
16000
0
5
10
15
20
Hour of the Day