Step 2: Import Libraries and Read Dataset

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import requests

# File paths for CitiBike data
file_paths = [
    r"C:\Users\Asus\Music\Citibike\202201-citibike-tripdata\202201-citibike-tripdata_1.csv",
    r"C:\Users\Asus\Music\Citibike\202201-citibike-tripdata\202201-citibike-tripdata_2.csv"
]

# Combine all files into one DataFrame
all_data = pd.concat([pd.read_csv(file) for file in file_paths], ignore_index=True)

# Display the first few rows
print(all_data.head())
```

```
C:\Users\Asus\AppData\Local\Temp\ipykernel_6748\185710368.py:12: DtypeWarning: Columns (5,7) have mixed types. Specify dtype option on import or set low_memory=False.
  all_data = pd.concat([pd.read_csv(file) for file in file_paths], ignore_index=True)
            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  95893AB04CED4AB8  electric_bike  2022-01-26 10:52:43.096
3  F853B50772137378   classic_bike  2022-01-03 08:35:48.247
4  7590ADAF834797B4B   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          6364.10

           end_station_name end_station_id  start_lat  start_lng  \
0  Mt Morris Park W & W 120 St         7685.14  40.802117 -73.946181
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
0  40.804038 -73.945925        member
1  40.688489 -73.991160        member
2  40.745168 -73.986831        member
3  40.745168 -73.986831        member
4  40.745168 -73.986831        member
```

Step 2: Fetch and Load Weather Data

In [2]:
```
# NOAA API token and parameters
token = "lJLRiTQPlcwBnfVgePvsmZGuSVzbRxZa"
headers = {"token": token}
url = "https://www.ncei.noaa.gov/access/services/data/v1"
params = {
    "dataset": "daily-summaries",
    "stations": "USW00094728",  # LaGuardia Airport
    "startDate": "2022-01-01",
    "endDate": "2022-12-31",
    "dataTypes": "TMAX,TMIN,PRCP",
    "format": "csv"
}

# Fetch data and save to a CSV file
response = requests.get(url, headers=headers, params=params)
with open("weather_2022.csv", "wb") as f:
    f.write(response.content)

# Load weather data
weather = pd.read_csv("weather_2022.csv")

# Display first few rows of weather data
print(weather.head())
```

```
      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
```

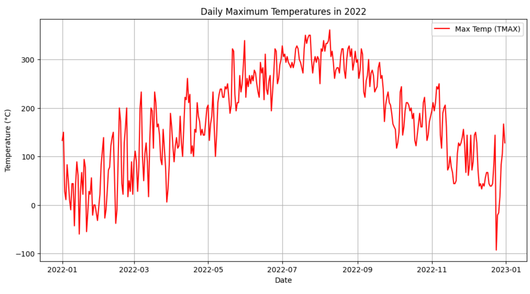
Step 3: Merge CitiBike and Weather Data

In [3]:
```
# Ensure columns are in datetime format
weather['DATE'] = pd.to_datetime(weather['DATE'])
all_data['start_date'] = pd.to_datetime(all_data['started_at']).dt.normalize()

# Merge the datasets
merged_data = pd.merge(
    all_data,
    weather,
    left_on='start_date',
    right_on='DATE',
    how='left'
)

# Display first few rows of merged data
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  95893AB04CED4AB8  electric_bike  2022-01-26 10:52:43.096
3  F853B50772137378   classic_bike  2022-01-03 08:35:48.247
4  7590ADAF834797B4B   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          6364.10

           end_station_name end_station_id  start_lat  start_lng  \
0  Mt Morris Park W & W 120 St         7685.14  40.802117 -73.946181
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
```

Step 3: Create a Line Plot of Temperatures

In [4]:
```
# Plot temperatures (TMAX)
plt.figure(figsize=(12, 6))
plt.plot(weather['DATE'], weather['TMAX'], label='Max Temp (TMAX)', color='red')
plt.title('Daily Maximum Temperatures in 2022')
plt.xlabel('Date')
plt.ylabel('Temperature (°C)')
plt.legend()
plt.grid()
plt.show()
```



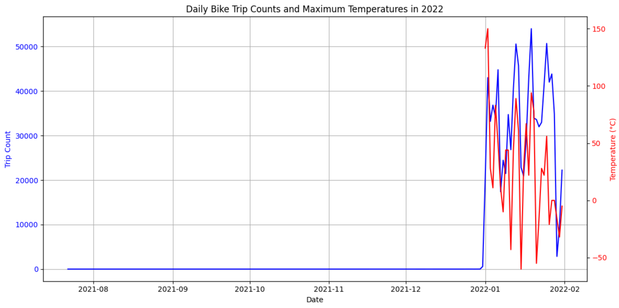
Step 4: Add Trip Counts per Day

In [5]:
```
# Create a column with trip counts per day
trip_counts = all_data.groupby('start_date').size().reset_index(name='trip_count')

# Merge trip counts with weather data
trip_weather_data = pd.merge(
    trip_counts,
    weather,
    left_on='start_date',
    right_on='DATE',
    how='left'
)

# Display the resulting DataFrame
print(trip_weather_data.head())
```

```
  start_date  trip_count STATION DATE  PRCP  TMAX  TMIN
0 2021-07-22           1     NaN  NaT   NaN   NaN   NaN
1 2021-09-13           1     NaN  NaT   NaN   NaN   NaN
2 2021-11-07           1     NaN  NaT   NaN   NaN   NaN
3 2021-11-09           1     NaN  NaT   NaN   NaN   NaN
4 2021-11-13           1     NaN  NaT   NaN   NaN   NaN
```

Step 5: Create Dual-Axis Line Chart

In [6]:
```
fig, ax1 = plt.subplots(figsize=(12, 6))

# Plot trip counts
ax1.plot(trip_weather_data['start_date'], trip_weather_data['trip_count'], color='blue', label='Trip Count')
ax1.set_xlabel('Date')
ax1.set_ylabel('Trip Count', color='blue')
ax1.tick_params(axis='y', labelcolor='blue')

# Create a twin y-axis for temperature
ax2 = ax1.twinx()
ax2.plot(trip_weather_data['start_date'], trip_weather_data['TMAX'], color='red', label='Max Temp (TMAX)')
ax2.set_ylabel('Temperature (°C)', color='red')
ax2.tick_params(axis='y', labelcolor='red')

# Add title and grid
plt.title('Daily Bike Trip Counts and Maximum Temperatures in 2022')
ax1.grid()
fig.tight_layout()
plt.show()
```



Step 8: Explanation in Markdown Cell

To create the dual-axis chart, I used the Object-Oriented paradigm of Matplotlib. The ax1 object plots the bike trip counts on the primary y-axis, while the ax2 object plots the temperatures on a secondary y-axis. Both y-axes share the same x-axis (date). This approach allows us to compare two datasets with different scales effectively.

In [ ]: