Part 3: Data Cleaning and Preparation

In [1]:
```
# Step 1: Load and Combine CitiBike Data Files
# This code snippet combines multiple CitiBike data files into a single DataFrame.
import pandas as pd
import glob

# List all CSV files in the folder
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)
```

```
C:\Users\Asus\AppData\Local\Temp\ipykernel_13600\274476337.py:13: 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)
```

Display the few rows

In [2]:
```
print(all_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          6364.10

             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
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: Check for Duplicates and Missing Values

Clean the data by removing duplicates and handling missing values.

In [3]:
```
# Check for duplicates
duplicates = all_data[all_data.duplicated()]
print(f"Number of duplicates: {len(duplicates)}")

# Drop duplicates if any
all_data = all_data.drop_duplicates()

# Check for missing values
missing_values = all_data.isnull().sum()
print("Missing values per column:")
print(missing_values)
```

```
Number of duplicates: 0
Missing values per column:
ride_id                 0
rideable_type           0
started_at              0
ended_at                0
start_station_name      0
start_station_id        0
end_station_name     9289
end_station_id       9289
start_lat               0
start_lng               0
end_lat              6614
end_lng              6614
member_casual           0
dtype: int64
```

Part 4: Fetch Weather Data using an API

Step 1: Set Up NOAA API Request with Comments

Add comments to each API parameter to provide context

In [4]:
```
import requests

# NOAA API token
token = "IjlRiTQPicwBnfVgePvsmZGuSVzbRxZa"
headers = {"token": token}

# API endpoint and parameters with comments
url = "https://www.ncei.noaa.gov/access/services/data/v1"
params = {
    "dataset": "daily-summaries",  # NOAA dataset for daily weather summaries
    "stations": "USW00094728",  # Station ID (e.g., LaGuardia Airport)
    "startDate": "2022-01-01",  # Start date for weather data
    "endDate": "2022-12-31",  # End date for weather data
    "dataTypes": "TMAX,TMIN,PRCP",  # Weather data types: max temp, min temp, and precipitation
    "format": "csv"  # Format for the returned data
}

# Request weather data
response = requests.get(url, headers=headers, params=params)

# Save to a CSV file
with open("weather_2022.csv", "wb") as f:
    f.write(response.content)
```

Part 5: Merge CitiBike and Weather Data

Step 1: Prepare and Merge Data on Date

In [5]:
```
# Load weather data
weather = pd.read_csv("weather_2022.csv")

# Convert 'DATE' in weather and 'started_at' in all_data to datetime format
weather['DATE'] = pd.to_datetime(weather['DATE'])
all_data['start_date'] = pd.to_datetime(all_data['started_at']).dt.normalize()

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

Part 6: Preliminary Observations

Step 1: Explore Key Trends

Analyze temperature trends and peak trip times to make informed decisions about visualization.

In [6]:
```
# Observing average temperature
avg_temp = merged_data[['TMAX', 'TMIN']].mean()
print(f"Average Max Temp: {avg_temp['TMAX']}, Average Min Temp: {avg_temp['TMIN']}")

# Finding peak hours for trips
merged_data['hour'] = pd.to_datetime(merged_data['started_at']).dt.hour
peak_times = merged_data['hour'].value_counts().head(5)
print("Top 5 peak hours for trips:")
print(peak_times)
```

```
Average Max Temp: 35.87037033419927, Average Min Temp: -40.15350118170986
Top 5 peak hours for trips:
hour
17    91275
16    87762
15    83357
18    83284
14    75830
Name: count, dtype: int64
```

Part 7: Performance Tips for Large Datasets

Step 1: Optimize Data for Efficient Processing

In [7]:
```
# Filter Columns: Keep only relevant columns before merging
all_data = all_data[['ride_id', 'started_at', 'start_date']]
```

In [8]:
```
# Adjust Data Types: Use memory-efficient data types
all_data['ride_id'] = all_data['ride_id'].astype('str')
```

In [ ]:
```

```