Project 3: Olympic Athletes

You've just been hired by an ambitious sports analytics startup that wants to build the most comprehensive database of Olympic athletes and their performances in history. Your job is to extract detailed biographical information and event results for athletes from Olympedia.org — a large but poorly structured website with no direct API access.

Your boss has directed you to scrape and process the data directly from the athlete profile pages (example: Jean-François Blanchy).

As a passionate data scientist, you're tasked with building a clean, structured dataset from the site's messy tables. This will involve careful HTML parsing, thoughtful handling of inconsistent data, and integrating performance results with biographical data for each athlete.

For this project you will do the following:

- Scrape and prepare two datasets: athlete biographical data and athlete event results.
- Tidy and standardize both datasets.
- Integrate both datasets using a unique athlete identifier.
- Perform exploratory analysis on the resulting data.

Part 1: Data Scraping and Preparation

Step 1: Scrape athlete biographical data

You'll start by writing a function to extract biographical details from each athlete's profile page.

Instructions:

- Use the requests library to download each athlete's profile page.
- Parse the HTML using BeautifulSoup.
- Locate the table with class biodata.
- Use pd.read_html() to convert this table into a Pandas DataFrame.
- Transpose the DataFrame so each row corresponds to one athlete.
- Add a new column called athlete_id with the athlete's numeric ID from the URL.
- Return the resulting DataFrame.

Step 2: Scrape athlete performance results

Next, extract competition results from each athlete's page.

Instructions:

- Locate the table with the class table.

- Use pd.read_html() to convert this table into a DataFrame.
- Add a column for athlete_id.
- Note that the table mixes rows for Games, NOC, and individual events.
- Use forward-filling (ffill()) to propagate values for columns like Games and Discipline down to their event rows.
- Rename columns:
    - Discipline (Sport) / Event → Event
    - NOC / Team → Team
- Drop any unnecessary columns.
- Return a tidy DataFrame with each row representing a competition result for one athlete.

Step 3: Automate scraping for all athletes
Now, build a loop to iterate through athlete IDs (from 1 to 149225).

Instructions:

- For each athlete ID:
    - Request the athlete's page.
    - If the page exists (status_code == 200), scrape both biographical data and event results using the functions you wrote.
    - Append the biographical data to a master DataFrame called bios.
    - Append the event results to a master DataFrame called results.
    - Handle any errors by logging the athlete ID to an errors list.
- Every 1,000 athletes save both bios and results DataFrames to CSV files as intermediate backups.
- At the end of the loop, save the final bios.csv, results.csv, and a text file with any failed athlete IDs.

Part 2: Data Cleaning

Now that you've scraped the raw athlete bios and result data, it's time to tidy it into a clean, structured format ready for analysis.

Step 1: Clean athlete biographical data

- One of the columns contains bullet characters (•) in place of spaces. Replace any instances of • with a regular space.
- Split Measurements Column into Height and Weight.
- Extract City, Region, and Country of Birth
- Extract Date of Birth and Death
- Any other cleaning procedure you deem necessary

Step 2: Clean athlete performance results

- Split Year and Type from Games. The Games column has strings like 1988 Summer Olympics or 2002 Winter Olympics. Split this into two new columns:
    - Year (integer)
    - Season (string)
- Split Out Tied Column from Pos. The Pos (Position) column sometimes includes a =, meaning a tied position, e.g. =2 or =1. Create a new logical (boolean) column: Tied — True if position includes =, else False.
- Clean up the Pos column by removing the =
- Make Non-Numeric Pos Values NaN. If the Pos column contains non-numeric values (e.g. 'DNF', 'DNS', empty string), convert those to NaN. Ensure Pos is numeric after this operation
- Any other cleaning procedure you deem necessary.

Step 3: Merge the datasets

Merge the bios and results dataframes using athlete_id as the key. Make the merge such that we only retain athletes who have results.

Part 3: Analysis

With your data cleaned, it's time to **explore its distributions and relationships**. You are free to take this in any direction you deem fit. Here are possible things you can try:

- Medal Distribution by Country
- Gender Breakdown by Season
- Average Athlete Height & Weight by Discipline