**Assignment 1: Initial Data Exploration**

1. Load the dataset using read\_csv.
2. Display the first 10 rows using head().
3. Display the last 5 rows using tail().
4. Print the shape and column names of the dataset.
5. Use info() and describe() to summarize the dataset.

**Assignment 2: Column Exploration and Filtering**

1. Find the unique values in the "category" and "sex" columns.
2. Count the frequency of each unique value in the "sex" column.
3. Filter rows where "category" is "Athletics" and the "sex" is "Female".
4. Filter rows where the year is between 1990 and 2010 (inclusive).

**Assignment 3: Logical Conditions**

1. Use a condition to filter rows where either:
   * The "category" is "Swimming" and the "sex" is "Male".
   * Or the "category" is "Gymnastics".
2. Save the filtered DataFrame to a new variable.

**Assignment 4: Slicing and Subsetting**

1. Use df.filter to create a DataFrame with only the "sex" and "category" columns.
2. Use loc to extract all rows and columns from "category" to "sex".

**Assignment 5: Date Manipulations**

1. Identify if there are any columns representing dates.
2. If yes, convert them into datetime format.
3. Filter rows based on a specific range of years or dates (use .between()).

**Assignment 6: Missing Data**

1. Check for missing data using isnull().sum().
2. Sort the columns based on the count of missing values using .sort\_values().
3. Create a shallow copy of the dataset and drop rows with missing values.

**Assignment 7: Combining Concepts**

1. Filter rows where:
   * "category" is "Athletics"
   * "year" is between 2000 and 2016.
2. Display the shape of the filtered dataset.
3. Extract a subset with columns from "category" to "sex".
4. Count unique values in the "category" column.

**Assignment 8: Descriptive Statistics**

1. Use describe(include='all') to analyze all columns.
2. Compare the summary statistics for subsets:
   * Rows where "sex" is "Male".
   * Rows where "sex" is "Female".

**Assignment 9: Advanced Filtering**

1. Create a filter to find rows where "category" contains "Track" or "Field".
2. Use .loc to update a column value for the filtered rows.
3. Count how many rows were updated.

**Assignment 10: Comprehensive Analysis**

1. Load the dataset and perform the initial exploration (head, info, shape, etc.).
2. Filter rows where "category" is "Swimming" and the "year" is greater than 2010.
3. Check for missing data in the filtered dataset.
4. Extract and display the count of unique values in the "category" column.
5. Save this filtered DataFrame to a new CSV file.

**15 Intermediate Assignments for Practice**

1. **Data Cleaning and Basic Analysis**
   * Load the olympics.csv file.
   * Display the first 10 rows and the last 5 rows using head() and tail().
   * Check the shape, column names, and basic statistics of the dataset using shape, columns, and describe().
   * Identify and display any null values in the dataset.
2. **Exploring and Filtering Data**
   * Load both CSV files.
   * Filter the olympics DataFrame to show all female athletes older than 25 who participated in the Summer Olympics.
   * From the filtered data, display only the Name, Age, and Team columns.
3. **Sorting and Conditional Selection**
   * Sort the bios DataFrame by born\_date in ascending order and height\_cm in descending order.
   * Identify athletes born after 1980 and have a height\_cm greater than 180. Display their names and heights.
4. **Joining and Combining Data**
   * Merge the two datasets on NOC to find the region and birth country of athletes in the olympics dataset.
   * Filter the combined DataFrame to display only athletes from Japan (NOC = "JPN") and calculate their average age and height.
5. **Creating and Renaming Columns**
   * Add a new column to the olympics DataFrame, BMI, calculated as Weight / (Height/100)\*\*2.
   * Rename the Sport column to Discipline and save the updated DataFrame to a new CSV file.
6. **Groupby and Aggregation**
   * Group the olympics data by Team and calculate the average age of athletes for each team.
   * Sort the results to find the top 10 teams with the youngest average age.
7. **Loc and Iloc Practice**
   * Using loc, select all rows for athletes who competed in the 2008 Olympics and their corresponding Sport and Event.
   * Using iloc, select the first 100 rows and only the Name, Team, and Year columns.
8. **Handling Dates and Null Values**
   * Convert the born\_date in bios to datetime format.
   * Identify rows where born\_date or died\_date is null and fill them with the string "Unknown".
9. **Sample and Value Counts**
   * Take a random sample of 50 athletes from the olympics DataFrame.
   * Use value\_counts() to find the most common sports in the sampled data.
10. **Rank and Sorting**
    * Rank athletes in the olympics dataset based on their weight.
    * Add a new column weight\_rank to store these ranks, and sort the data based on the rank.
11. **Advanced Filtering and Querying**
    * Identify athletes who participated in both Summer and Winter Olympics using conditional filters.
    * From these athletes, display their Name, Team, and Year.
12. **Top and Bottom Values**
    * Find the top 5 tallest athletes and the bottom 5 shortest athletes from the bios DataFrame.
    * Display their name, height\_cm, and born\_country.
13. **Handling Columns and Saving Data**
    * Drop the City column from the olympics dataset.
    * Save the updated DataFrame with the removed column to a new CSV file.
14. **Calculations and Unique Values**
    * Calculate the total number of unique events in the olympics dataset.
    * Add a column experience\_level based on age:
      + "Young" for athletes under 25, "Experienced" for 25–35, and "Veteran" for those above 35.
15. **Using Both DataFrames Together**
    * Combine bios and olympics datasets using merge to find athletes' birth regions and their participation in events.
    * Group the combined data by born\_country and find the average height and weight of athletes for each country.

Here are 15 assignments of intermediate difficulty for your practice, each covering multiple topics:

Assignment 1:

Read the Olympics CSV file and create a new DataFrame with only the columns "Name", "Age", "Height", and "Weight". Use the head() method to display the first 5 rows. Then, use the describe() method to get summary statistics for the new DataFrame.

Assignment 2:

Use the loc[] method to select rows from the Olympics DataFrame where the "Age" is greater than 30 and the "Sport" is "Swimming". Then, use the value\_counts() method to count the number of unique values in the "Event" column for the selected rows.

Assignment 3:

Create a new column "BMI" in the Olympics DataFrame by calculating the body mass index (BMI) using the "Height" and "Weight" columns. Then, use the sort\_values() method to sort the DataFrame by the "BMI" column in descending order.

Assignment 4:

Use the groupby() function to group the Olympics DataFrame by the "NOC" column and calculate the mean "Height" and "Weight" for each group. Then, use the rank() function to rank the groups by their mean "Height".

Assignment 5:

Read the Bios CSV file and create a new DataFrame with only the columns "name", "born\_date", and "height\_cm". Use the head() method to display the first 5 rows. Then, use the info() method to get information about the new DataFrame.

Assignment 6:

Use the loc[] method to select rows from the Bios DataFrame where the "born\_country" is "USA" and the "height\_cm" is greater than 180. Then, use the describe() method to get summary statistics for the selected rows.

Assignment 7:

Create a new column "age\_at\_death" in the Bios DataFrame by calculating the age at death using the "born\_date" and "died\_date" columns. Then, use the value\_counts() method to count the number of unique values in the "age\_at\_death" column.

Assignment 8:

Use the groupby() function to group the Bios DataFrame by the "born\_region" column and calculate the mean "height\_cm" and "weight\_kg" for each group. Then, use the sort\_values() method to sort the groups by their mean "height\_cm" in descending order.

Assignment 9:

Use the loc[] method to select rows from the Olympics DataFrame where the "Year" is 2012 and the "Season" is "Summer". Then, use the describe() method to get summary statistics for the selected rows.

Assignment 10:

Create a new column "weight\_category" in the Olympics DataFrame by categorizing the "Weight" column into different categories (e.g. lightweight, middleweight, heavyweight). Then, use the value\_counts() method to count the number of unique values in the "weight\_category" column.

Assignment 11:

Use the groupby() function to group the Olympics DataFrame by the "Sport" column and calculate the mean "Height" and "Weight" for each group. Then, use the rank() function to rank the groups by their mean "Height".

Assignment 12:

Read the Bios CSV file and create a new DataFrame with only the columns "name", "born\_date", and "died\_date". Use the head() method to display the first 5 rows. Then, use the info() method to get information about the new DataFrame.

Assignment 13:

Use the loc[] method to select rows from the Bios DataFrame where the "born\_country" is "China" and the "height\_cm" is greater than 170. Then, use the describe() method to get summary statistics for the selected rows.

Assignment 14:

Create a new column "life\_expectancy" in the Bios DataFrame by calculating the life expectancy using the "born\_date" and "died\_date" columns. Then, use the value\_counts() method to count the number of unique values in the "life\_expectancy" column.

Assignment 15:

Use the groupby() function to group the Bios DataFrame by the "born\_region" column and calculate the mean "height\_cm" and "weight\_kg" for each group. Then, use the sort\_values() method to sort the groups by their mean "height\_cm" in descending order.

1. **Data Cleaning and Basic Analysis**
   * Load the olympics.csv file.
   * Display the first 10 rows and the last 5 rows using head() and tail().
   * Check the shape, column names, and basic statistics of the dataset using shape, columns, and describe().
   * Identify and display any null values in the dataset.