Assignment 1: Pivot Table and Groupby (Etsy)

Create a pivot table to show the average product price by seller name and tags. Then, use the groupby function to group the data by seller name and calculate the total views and favorites for each seller.

Assignment 1: Pivot Table and Groupby (Bios)

Create a pivot table to show the average height and weight by born\_country and born\_region. Then, use the groupby function to group the data by born\_country and calculate the total number of athletes for each country.

Assignment 2: Crosstab and Apply (Bios)

Create a crosstab to show the frequency of each born\_country by born\_region. Then, use the apply function to apply a lambda function that calculates the average height and weight for each born\_region.

Assignment 3: Datetime and Groupby (Bios)

Convert the born\_date column to datetime format and extract the year and month. Then, use the groupby function to group the data by year and month, and calculate the total number of athletes born in each year and month.

Assignment 4: Map and Pivot Table (Bios)

Create a map to map the born\_country to the corresponding NOC. Then, create a pivot table to show the average height and weight by NOC and born\_region.

Assignment 5: Groupby and Apply (Bios)

Group the data by born\_city and apply a lambda function that calculates the average height and weight for each city. Then, use the apply function to apply a lambda function that calculates the total number of athletes for each city.

Assignment 6: Crosstab and Datetime (Bios)

Create a crosstab to show the frequency of each born\_country by decade (e.g. 1900-1909, 1910-1919, etc.). Then, use the datetime format to extract the century and create a crosstab to show the frequency of each born\_country by century.

Assignment 7: Pivot Table and Map (Bios)

Create a pivot table to show the average height and weight by born\_country and born\_region. Then, use the map function to map the born\_country to the corresponding NOC, and create a pivot table to show the average height and weight by NOC and born\_region.

Assignment 8: Groupby and Crosstab (Bios)

Group the data by born\_region and create a crosstab to show the frequency of each born\_country by born\_region. Then, use the groupby function to group the data by born\_country and create a crosstab to show the frequency of each born\_region by born\_country.

Assignment 9: Datetime and Apply (Bios)

Convert the died\_date column to datetime format and extract the year and month. Then, use the apply function to apply a lambda function that calculates the average age at death for each year and month.

Assignment 10: Pivot Table and Groupby (Bios)

Create a pivot table to show the average height and weight by born\_country and born\_city. Then, use the groupby function to group the data by born\_country and calculate the total number of athletes for each country, and create a pivot table to show the average height and weight by born\_country and born\_city.

I hope these assignments help you practice the topics you mentioned!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.

Here’s how your additional topics can be integrated into the existing 15 assignments to make them more comprehensive:

15 Intermediate Assignments for Practice with Added Topics

1. Data Cleaning and Basic Analysis
   * Use dropna() to remove rows with null values and analyze the effect on the dataset.
   * Replace null values with the mean of the respective columns using fillna().
   * Use reset\_index() to reassign a clean index after dropping duplicates in the dataset.
2. Exploring and Filtering Data
   * Add a column Age\_Group using the apply() function:
     + "Young" for Age < 25, "Mid" for 25–35, "Senior" for > 35.
   * Use groupby() to aggregate the data and calculate the mean age of athletes grouped by their team.
   * Drop duplicates in the Name column and filter data again.
3. Sorting and Conditional Selection
   * After sorting, use rank() to rank the athletes by their height within each team.
   * Use string operations to convert names to uppercase (str.upper()) and display them for the filtered athletes.
4. Joining and Combining Data
   * Handle null values in the combined dataset using bfill or ffill.
   * Add a datettime column showing the date of birth for athletes and extract the year, month, and day using dt.year, dt.month, and dt.day.
5. Creating and Renaming Columns
   * Use the rename() function to rename multiple columns simultaneously.
   * Create a column Category using map():
     + Map "Summer" to "Hot Events" and "Winter" to "Cold Events."
   * Replace all instances of "Discipline" with "Activity" using replace().
6. Groupby and Aggregation
   * Add a new column Median\_Height by grouping athletes by their teams and calculating the median height.
   * Use a pivot table to analyze the average age and weight of athletes grouped by Discipline.
7. Loc and Iloc Practice
   * Using iloc, filter all rows where the index is divisible by 5.
   * Add a multi-column index using set\_index() with Year and Discipline.
8. Handling Dates and Null Values
   * Use interpolate() to fill missing born\_date values.
   * Extract the day names from born\_date using .dt.day\_name() and create a new column called Born\_Day.
9. Advanced Filtering and Querying
   * After filtering, drop duplicate rows using drop\_duplicates().
   * Apply string operations to filter athletes with names starting with specific letters (str.startswith('A')).
10. Top and Bottom Values
    * Use advanced indexing techniques to extract top and bottom rows based on height.
    * Add a column Height\_Category to classify athletes as "Short," "Medium," or "Tall" using the apply() function.
11. Handling Columns and Saving Data
    * After dropping the City column, reset the index to start from 0.
    * Use string operations to convert column names to lowercase before saving the file.
12. Calculations and Unique Values
    * Replace invalid or missing values in the age column using the median with fillna().
    * Add an Event\_Count column using groupby() and aggregation to count the number of events each athlete participated in.
13. Using Both DataFrames Together
    * After combining, create a pivot table to analyze average height and weight by Discipline and born\_country.
    * Use crosstab to find the frequency distribution of athletes by Discipline and Age\_Group.

Each assignment now includes a mix of basic, intermediate, and advanced topics to reinforce understanding of the concepts while exploring diverse functionalities of pandas. Let me know if you need any further refinements!

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.