Scenario: Building a Smart Data Aggregator

You have been hired by a startup working on a **Smart Data Aggregator tool**. The goal of the tool is to manage and analyze large sets of user data efficiently. The startup focuses on different types of collections (List, Tuple, Set, and Dictionary) to handle various tasks such as real-time analytics, tracking data, and reporting. Your task is to develop different modules for the aggregator using Python.

Part 1: User Data Processing with Lists

You are provided with user information in the form of a list of tuples. Each tuple represents a user with the format: (user_id, user_name, age, country). The list can contain more than 100 records, and you are required to:

Write a function to:

- Filter out users older than 30 from specific countries (‘USA’, ‘Canada’).
- Extract their names into a new list.

Implement a function that:

- Sorts the original list of tuples by age and returns the top 10 oldest users.
- checks if there are any users with duplicate names in the list. If duplicates are found, output those names.

Part 2: Immutable Data Management with Tuples

You need to handle transaction data from the aggregator's analytics module. This data is stored in the form of tuples since it needs to remain immutable. Each transaction is represented as a tuple: (transaction_id, user_id, amount, timestamp).

Write a function that:

- Takes a list of transactions (tuples) and finds the total number of unique users involved in transactions.
- Ensures the integrity of the tuples by avoiding any changes to the original data.

Implement a function that:

- Identifies and returns the transaction with the highest amount without altering the list of tuples.
- receives a list of tuples and returns two separate lists: one containing all the transaction_ids and the other containing all user_ids. What challenges might arise if the tuple size is inconsistent?

Part 3: Unique Data Handling with Sets

The Smart Data Aggregator also manages sets of unique user IDs who visited certain pages. You have three sets, each representing user IDs of visitors to pages A, B, and C.

Write a function that:

- Finds the users who visited both Page A and Page B.
- Finds users who visited either Page A or Page C, but not both.

Implement a function that:

- Updates the set for Page A with new user IDs.
- Removes a list of user IDs from the set for Page B.

Part 4: Data Aggregation with Dictionaries

The aggregator collects user feedback stored in a dictionary. The dictionary uses the user_id as keys, and the values are nested dictionaries with feedback details: {'rating': int, 'comments': str}.

Write a function that:

- Filters out users who rated 4 or higher and stores their user_id and rating in a new dictionary.
- sort the dictionary of user feedback by rating in descending order and return the top 5 users.

Implement a function that:

- Combines feedback from multiple dictionaries. If a user is present in more than one dictionary, update their rating to the highest one and append their comments.
- Use dictionary comprehension to create a dictionary of user_id and rating for all users whose rating is greater than 3.

Note:

Plagiarism will be detected very easily so do your own assignment.