Documentation: Ranking Scores in a DataFrame

This documentation provides a detailed explanation of how to rank scores from a DataFrame using the Pandas library in Python. The ranking follows specific rules to handle ties and ensure that the ranks are in consecutive order without any gaps.

Problem Statement

You are given a table named Scores with the following schema:

- **id (int):** The primary key of the table. It is a unique identifier for each row.
- **score (decimal):** A floating-point value representing the score of a game. It has two decimal places.

The task is to find the rank of each score in descending order. The ranking should adhere to the following rules:

1. **Descending Order:** Scores should be ranked from the highest to the lowest.
2. **Handling Ties:** If there is a tie between two scores (i.e., two or more scores are the same), they should share the same rank.
3. **Consecutive Ranks:** After a tie, the next rank number should be the next consecutive integer. There should be no gaps in the ranking sequence.

Input

A DataFrame named scores that contains two columns:

- **id:** A unique identifier for each score entry.
- **score:** The score to be ranked.

Example Input:

Id	score
1	3.50
2	3.65
3	4.00
4	3.85
5	4.00
6	3.65

Output

The function should return a DataFrame with two columns:

- **score:** The original score values sorted in descending order.
- **rank:** The rank of each score based on the rules defined above.

Example Output:

score	rank
4.00	1
4.00	1
3.85	2
3.65	3
3.65	3
3.50	4

Approach

1. **Sorting the DataFrame:**
   - The initial step involves sorting the DataFrame by the score column in descending order. This ensures that the highest scores appear first in the DataFrame.

2. Assigning Ranks to Scores:

- After sorting, the next step is to assign ranks to the sorted scores. To achieve this, the rank method from Pandas is used.
- The rank function provides several methods for ranking. In this problem, the method='dense' parameter is used. This method assigns the same rank to tied scores and ensures that the next rank is incremented by one. For instance, if two scores are tied at rank 1, the next unique score will receive rank 2.
- The ascending=False parameter ensures that ranks are assigned in descending order of scores.

3. Selecting and Returning the Required Columns:

- The final step is to select the required columns (score and rank) from the DataFrame and return them. The DataFrame should be sorted by score in descending order to match the output format.

Key Considerations

- **Handling Ties:** The method='dense' ranking method is crucial for ensuring that there are no gaps in the ranking sequence after a tie. Other methods like min, max, or average might produce different ranking sequences that do not fulfill the problem requirements.

- **Performance:** Sorting and ranking operations are generally efficient in Pandas, but for very large datasets, performance considerations may arise. Optimization techniques or alternative libraries may be considered for handling large-scale data.

- **Output Format:** The output must strictly adhere to the format with two columns (score and rank) and be sorted by score in descending order. This format is crucial for correctly interpreting the results in the context of the problem.

Edge Cases

- **All Scores are the Same:** If all scores in the DataFrame are identical, all entries should have the same rank (rank 1).

- **All Scores are Unique:** If all scores are distinct, each score should receive a unique rank from 1 to N (where N is the number of scores).

- **Empty DataFrame:** If the input DataFrame is empty, the output should also be an empty DataFrame with the required columns (score and rank).

By following this approach, you can effectively rank the scores in a DataFrame according to the specified rules, ensuring accurate and meaningful ranking results.