**Rank Scores**

**SELECT**

**Score,**

**DENSE\_RANK() OVER (ORDER BY Score DESC) as 'Rank'**

**FROM**

**Scores**

**ORDER BY**

**Score DESC;**

**Explaination**

1. **FROM Scores**: This specifies that we're retrieving data from a table called "Scores".
2. **SELECT Score**: We're selecting the "Score" column from this table.
3. **DENSE\_RANK() OVER (ORDER BY Score DESC) as 'Rank'**: This is the key part:
   * DENSE\_RANK() is a window function that assigns a rank to each distinct value in a result set
   * OVER (ORDER BY Score DESC) specifies how we want the ranking to be calculated:
     + Scores are ordered in descending order (highest first)
     + Equal scores receive the same rank
     + Unlike regular RANK(), DENSE\_RANK() doesn't skip numbers after ties
   * as 'Rank' names this calculation as "Rank" in the output
4. **ORDER BY Score DESC**: This final clause sorts the output table by scores in descending order (highest to lowest).

**How the ranking works:**

* When two or more rows have the same score (like the two 4.00 scores in your example), they all get the same rank (1).
* After a tie, the next distinct value gets the next consecutive rank. So after the 4.00 scores (rank 1), 3.85 gets rank 2.
* This ensures there are no "holes" in the ranking, which is what the problem requires.

For your sample data, here's how the ranks are assigned:

* 4.00 → Rank 1 (highest score)
* 4.00 → Rank 1 (tied with previous)
* 3.85 → Rank 2 (next distinct score)
* 3.65 → Rank 3 (next distinct score)
* 3.65 → Rank 3 (tied with previous)
* 3.50 → Rank 4 (next distinct score)

This is different from regular RANK() which would leave gaps after ties (it would produce 1,1,3,4,4,6 instead of 1,1,2,3,3,4).