DESCRIPTION

Table: Queue

+-------------+---------+

| Column Name | Type |

+-------------+---------+

| person_id | int |

| person_name | varchar |

| weight | int |

| turn | int |

+-------------+---------+

person_id column contains unique values.

This table has the information about all people waiting for a bus.

The person_id and turn columns will contain all numbers from 1 to n, where n is the number of rows in the table.

turn determines the order of which the people will board the bus, where turn=1 denotes the first person to board and turn=n denotes the last person to board.

weight is the weight of the person in kilograms.

There is a queue of people waiting to board a bus. However, the bus has a weight limit of 1000 **kilograms**, so there may be some people who cannot board.

Write a solution to find the person_name of the **last person** that can fit on the bus without exceeding the weight limit. The test cases are generated such that the first person does not exceed the weight limit.

Note that *only one* person can board the bus at any given turn.

The result format is in the following example.

Example 1:

Input:

Queue table:

+-----------+-------------+--------+------+

| person_id | person_name | weight | turn |

+-----------+-------------+--------+------+

| 5 | Alice | 250 | 1 |

| 4 | Bob | 175 | 5 |

| 3 | Alex | 350 | 2 |

| 6 | John Cena | 400 | 3 |

| 1 | Winston | 500 | 6 |

| 2 | Marie | 200 | 4 |

+-----------+-------------+--------+------+

Output:

+-------------+

| person_name |

+-------------+

| John Cena |

+-------------+

Explanation: The folowing table is ordered by the turn for simplicity.

+------+----+-----------+--------+--------------+

| Turn | ID | Name | Weight | Total Weight |

+------+----+-----------+--------+--------------+

| 1 | 5 | Alice | 250 | 250 |

| 2 | 3 | Alex | 350 | 600 |

| 3 | 6 | John Cena | 400 | 1000 | (last person to board)

| 4 | 2 | Marie | 200 | 1200 | (cannot board)

| 5 | 4 | Bob | 175 | ___ |

| 6 | 1 | Winston | 500 | ___ |

+------+----+-----------+--------+--------------+

SOLUTION

MySQL:

- Calculate accumulative weight using the window function SUM(weight) OVER(ORDER BY turn) and add this query in CTE

- Filter passengers where total_weight <= 1000
- Get the last person order by total_weight in a descending order and limit 1

```
WITH running_total AS (
    SELECT person_name, SUM(weight) OVER(ORDER BY turn) AS total_weight
    FROM Queue
    GROUP BY person_name)
SELECT person_name
FROM running_total
WHERE total_weight <= 1000
ORDER BY total_weight DESC
LIMIT 1;
```

PostgreSQL:

- Self-join Queue as q1 and q2 where each row is paired with all rows that have a higher or equal turn value
- Group by q1.person_name and calculate accumulate weight using SUM(q2.weight)
- Filter groups whose accumulative weight is not greater than 1000 using the having clause
- Order by SUM(q2.weight) in a descending order and limit 1 to select the last person

```
SELECT q1.person_name
FROM Queue q1
JOIN Queue q2
ON q1.turn >= q2.turn
GROUP BY 1
HAVING SUM(q2.weight) <= 1000
ORDER BY SUM(q2.weight) DESC
LIMIT 1;
```