Assume you're given a table on Walmart user transactions. Based on their most recent transaction date, write a query that retrieve the users along with the number of products they bought.

Output the user's most recent transaction date, user ID, and the number of products, sorted in chronological order by the transaction date.

*Starting from November 10th, 2022, the official solution was updated, and the expected output of transaction date, number of users, and number of products was changed to the current expected output.*

**user\_transactions Table:**

| **Column Name** | **Type** |
| --- | --- |
| product\_id | integer |
| user\_id | integer |
| spend | decimal |
| transaction\_date | timestamp |

**user\_transactions Example Input:**

| **product\_id** | **user\_id** | **spend** | **transaction\_date** |
| --- | --- | --- | --- |
| 3673 | 123 | 68.90 | 07/08/2022 12:00:00 |
| 9623 | 123 | 274.10 | 07/08/2022 12:00:00 |
| 1467 | 115 | 19.90 | 07/08/2022 12:00:00 |
| 2513 | 159 | 25.00 | 07/08/2022 12:00:00 |
| 1452 | 159 | 74.50 | 07/10/2022 12:00:00 |

**Example Output:**

| **transaction\_date** | **user\_id** | **purchase\_count** |
| --- | --- | --- |
| 07/08/2022 12:00:00 | 115 | 1 |
| 07/08/2022 12:00:000 | 123 | 2 |
| 07/10/2022 12:00:00 | 159 | 1 |

The dataset you are querying against may have different input & output - **this is just an example**!

**Solution**

**Step 1: Rank transactions by date for each user**

To determine the latest transaction date for each user, we use the RANK() window function.

By partitioning the data by user ID and ordering it by transaction date in descending order, we assign a rank to each transaction date. The rank 1 represents the latest transaction date.

SELECT

transaction\_date,

user\_id,

product\_id,

RANK() OVER (

PARTITION BY user\_id

ORDER BY transaction\_date DESC) AS transaction\_rank

FROM user\_transactions;

**Step 2: Filter records for the latest transactions only**

In this step, we filter the records to include only the transactions with the highest rank (rank 1). This ensures that we retain only the latest transaction for each user.

WITH latest\_transactions\_cte AS (

SELECT

transaction\_date,

user\_id,

product\_id,

RANK() OVER (

PARTITION BY user\_id

ORDER BY transaction\_date DESC) AS transaction\_rank

FROM user\_transactions)

SELECT \*

FROM latest\_transactions\_cte

WHERE transaction\_rank = 1

**Step 3: Retrieve transaction details and count the number of products purchased**

In this step, we select the transaction date, user ID, and product ID from the filtered records. Additionally, we include a count of the product IDs to determine the number of products purchased in each transaction.

WITH latest\_transactions\_cte AS (

SELECT

transaction\_date,

user\_id,

product\_id,

RANK() OVER (

PARTITION BY user\_id

ORDER BY transaction\_date DESC) AS transaction\_rank

FROM user\_transactions)

SELECT

transaction\_date,

user\_id,

COUNT(product\_id) AS purchase\_count

FROM latest\_transactions\_cte

WHERE transaction\_rank = 1

GROUP BY transaction\_date, user\_id

ORDER BY transaction\_date;