**Comparing GROUP BY and OVER in Flink SQL**

**Sample Data: orders Table**

| **user\_id** | **event\_time** | **order\_id** | **amount** |
| --- | --- | --- | --- |
| A | 2024-02-21 10:01:00 | 101 | 50 |
| B | 2024-02-21 10:03:00 | 102 | 30 |
| A | 2024-02-21 10:05:00 | 103 | 20 |
| A | 2024-02-21 10:12:00 | 104 | 40 |
| B | 2024-02-21 10:15:00 | 105 | 25 |

**Example 1: GROUP BY with Tumbling Window (10-minute)**

SELECT window\_start, window\_end, user\_id, COUNT(\*) AS order\_count, SUM(amount) AS total\_amount

FROM TABLE(

TUMBLE(TABLE orders, DESCRIPTOR(event\_time), INTERVAL '10' MINUTES)

)

GROUP BY window\_start, window\_end, user\_id;

**Result:**

| **window\_start** | **window\_end** | **user\_id** | **order\_count** | **total\_amount** |
| --- | --- | --- | --- | --- |
| 2024-02-21 10:00:00 | 2024-02-21 10:10:00 | A | 2 | 70 |
| 2024-02-21 10:00:00 | 2024-02-21 10:10:00 | B | 1 | 30 |
| 2024-02-21 10:10:00 | 2024-02-21 10:20:00 | A | 1 | 40 |
| 2024-02-21 10:10:00 | 2024-02-21 10:20:00 | B | 1 | 25 |

**Key Takeaways:**

* Groups orders into **fixed 10-minute windows**.
* Results are emitted **once per window**.

**Example 2: OVER for Running Sum in Last 10 Minutes**

SELECT user\_id, event\_time, order\_id,

SUM(amount) OVER (

PARTITION BY user\_id

ORDER BY event\_time

RANGE BETWEEN INTERVAL '10' MINUTES PRECEDING AND CURRENT ROW

) AS running\_total

FROM orders;

**Result:**

| **user\_id** | **event\_time** | **order\_id** | **running\_total** |
| --- | --- | --- | --- |
| A | 2024-02-21 10:01:00 | 101 | 50 |
| B | 2024-02-21 10:03:00 | 102 | 30 |
| A | 2024-02-21 10:05:00 | 103 | 70 |
| A | 2024-02-21 10:12:00 | 104 | 40 |
| B | 2024-02-21 10:15:00 | 105 | 25 |

**Key Takeaways:**

* Calculates a **running total** of amount for each user\_id over the last **10 minutes**.
* Updates the result **continuously for each row**.

**Comparison Table**

| **Feature** | **GROUP BY (Tumbling Window)** | **OVER (Window Function)** |
| --- | --- | --- |
| **Aggregation** | Aggregates rows in a time window | Row-level running aggregation |
| **Result Emission** | At the end of the window | Continuously updates per row |
| **Use Case** | Periodic reporting (e.g., orders per 10 min) | Real-time tracking (e.g., running sum per user) |
| **Latency** | Delayed until window closes | Low latency |