**Watermark and Window**

To make things clearer, let's walk through a specific example with tables and sample data. We’ll create a delivery\_orders table, insert some data with timestamps, and then run a query using watermarks and windows.

**Step 1: Create the delivery\_orders Table**

This table will hold the order data, including order ID, delivery time, and event timestamp.

CREATE TABLE delivery\_orders (

order\_id STRING,

delivery\_time\_minutes INT,

event\_time TIMESTAMP(3),

WATERMARK FOR event\_time AS event\_time - INTERVAL '5' MINUTE

) WITH (

'connector' = 'kafka',

'topic' = 'delivery\_orders',

'format' = 'json',

'properties.bootstrap.servers' = 'kafka-server:9092'

);

**Sample Data for delivery\_orders**



| **order\_id** | **delivery\_time\_minutes** | **event\_time** |
| --- | --- | --- |
| O1 | 30 | 2023-10-29 09:01:00 |
| O2 | 45 | 2023-10-29 09:06:00 |
| O3 | 35 | 2023-10-29 09:08:00 |
| O4 | 25 | 2023-10-29 09:17:00 |
| O5 | 40 | 2023-10-29 09:18:00 |
| O6 | 50 | 2023-10-29 09:20:00 |
| O7 | 20 | 2023-10-29 09:29:00 |
| O8 | 45 | 2023-10-29 09:33:00 |

**Step 2: Define the Query with a 15-Minute Tumbling Window**

Here’s a query that calculates the average delivery time for every 15-minute window:

SELECT

TUMBLE\_START(event\_time, INTERVAL '15' MINUTE) AS window\_start,

TUMBLE\_END(event\_time, INTERVAL '15' MINUTE) AS window\_end,

AVG(delivery\_time\_minutes) AS avg\_delivery\_time

FROM delivery\_orders

GROUP BY TUMBLE(event\_time, INTERVAL '15' MINUTE);

**Step 3: Execution and Results**

**Explanation of Tumbling Windows (15-Minute Intervals)**

Based on our sample data and the query above, let’s go through each 15-minute window and calculate the average delivery time:

1. **Window 09:00 to 09:15**
   * Orders: O1, O2, O3
   * Average Delivery Time = (30 + 45 + 35) / 3 = 36.67
2. **Window 09:15 to 09:30**
   * Orders: O4, O5, O6
   * Average Delivery Time = (25 + 40 + 50) / 3 = 38.33
3. **Window 09:30 to 09:45**
   * Orders: O7, O8



* + Average Delivery Time = (20 + 45) / 2 = 32.5



They **should not**, based on event time alone:

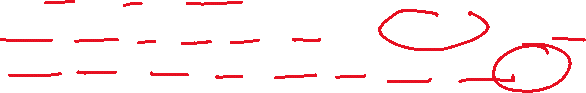
* O7 has event\_time = 09:29:00 → should belong to **09:15–09:30**



* O8 has event\_time = 09:33:00 → correctly belongs to **09:30–09:45**



However, in your output, both O7 and O8 appear in the **09:30–09:45** window. This suggests **O7 was too late for its correct window** due to the watermark logic.



**Understanding Watermark Impact**

Your watermark is defined as:

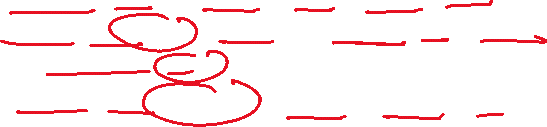
WATERMARK FOR event\_time AS event\_time - INTERVAL '5' MINUTE



This means:



* A window will **close** when the **watermark passes the window end**.
* An event with event\_time = 09:29:00 is considered on time **until the watermark reaches 09:34:00** (09:29 + 5 minutes).



BUT—Flink only holds the **09:15–09:30** window **open until the watermark reaches 09:30:00**.



So if:

* O7 arrives **after the watermark has passed 09:30:00**
* even though its event\_time = 09:29:00,



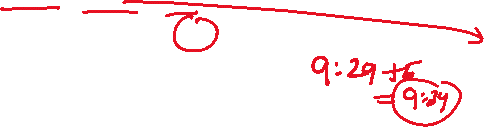
* it is considered **late for the 09:15–09:30 window**



And unless you have **allowedLateness** configured, late events are **dropped or routed to the next window** (or trigger a side output in advanced cases).



Flink therefore **assigns O7 to the next available window (09:30–09:45)**—which is also where O8 correctly belongs.

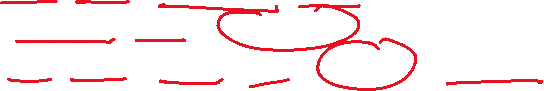


**Final Explanation**

O7 and O8 both appear in the **09:30–09:45 window** **not because of event time**, but because:



1. O7 arrived **late** (after watermark passed 09:30:00)
2. It **missed** its intended window (09:15–09:30)



1. It was **included** in the next open window (09:30–09:45) by Flink's windowing engine



**How to Prevent This**

To allow O7 to still go into its correct window even if it arrives late:

* Use allowedLateness (in Table API or SQL supported via configuration)

**Query Output Table**

| **window\_start** | **window\_end** | **avg\_delivery\_time** |
| --- | --- | --- |
| 2023-10-29 09:00:00 | 2023-10-29 09:15:00 | 36.67 |
| 2023-10-29 09:15:00 | 2023-10-29 09:30:00 | 38.33 |
| 2023-10-29 09:30:00 | 2023-10-29 09:45:00 | 32.5 |

**Key Points Illustrated**

* **Watermark**: The watermark allows late events up to 5 minutes. If an order arrives up to 5 minutes late, it will still be considered in the corresponding window.
* **Tumbling Windows**: Each 15-minute segment is processed independently, providing rolling averages for delivery times over time.