SQL Sessionization Challenge

Important Scenario Based Question

"A new session begins whenever the gap between consecutive events for the same user **exceeds 30 minutes** . Assign a Session_ID to each row."

📊 Input Data: User Events with Timestamps

User_ID	Event_Time
A	2024-01-01 10:00:00
A	2024-01-01 10:20:00
A	2024-01-01 11:05:00
A	2024-01-01 11:25:00
A	2024-01-01 12:10:00
B	2024-01-02 09:00:00
B	2024-01-02 09:15:00
B	2024-01-02 10:10:00
B	2024-01-02 10:20:00
B	2024-01-02 11:00:00

The Core Pattern

Cumulative SUM() OVER Pattern

1 Calculate Time Gaps

Use LAG() to find minutes since last event for each user

2 Flag New Sessions

Mark rows where gap > 30 minutes or it's the first event

3 Cumulative SUM

Use SUM() OVER to create sequential session IDs

Raw Events User timestamps → **LAG() Function** Calculate gaps → **CASE WHEN** Flag new sessions → **SUM() OVER** Create IDs → **Session IDs** Final output

🎯 Visual Example: How Session IDs Are Created

New Session Flags	Cumulative SUM	Final Session IDs
1	1	**1**
0	1+0 = 1	**1**
1	1+0+1 = 2	**2**
0	1+0+1+0 = 2	**2**
1	1+0+1+0+1 = 3	**3**

User Session Identification

SQL Server 2012+

```
WITH EventAnalysis AS (
    SELECT User_ID, Event_Time,
    DATEDIFF(
        MINUTE,
        LAG(Event_Time) OVER (PARTITION BY User_ID ORDER BY
Event_Time),
        Event_Time
    ) AS gap_minutes
    FROM user_events
),
SessionStarts AS (
    SELECT User_ID, Event_Time, gap_minutes,

    CASE
        WHEN gap_minutes IS NULL THEN 1
        WHEN gap_minutes > 30 THEN 1
        ELSE 0
    END AS session_start_flag

    FROM EventAnalysis
)
SELECT User_ID, Event_Time, gap_minutes, session_start_flag,

    SUM(session_start_flag) OVER (
        PARTITION BY User_ID ORDER BY Event_Time
    ) AS Session_ID

FROM SessionStarts
ORDER BY User_ID, Event_Time;
```

Sessionization Output

📊 Final Output: User Events with Session IDs

User_ID	Event_Time	Gap (minutes)	Session Logic	Session_ID
A	2024-01-01 10:00:00	First Event	First event → Start Session 1	1
A	2024-01-01 10:20:00	20	Gap ≤ 30 → Same Session	1
A	2024-01-01 11:05:00	45	Gap > 30 → New Session	2
A	2024-01-01 11:25:00	20	Gap ≤ 30 → Same Session	2
A	2024-01-01 12:10:00	45	Gap > 30 → New Session	3
B	2024-01-02 09:00:00	First Event	First event → Start Session 1	1
B	2024-01-02 09:15:00	15	Gap ≤ 30 → Same Session	1
B	2024-01-02 10:10:00	55	Gap > 30 → New Session	2
B	2024-01-02 10:20:00	10	Gap ≤ 30 → Same Session	2
B	2024-01-02 11:00:00	40	Gap > 30 → New Session	3

Master Time-Based Analysis

Essential for User Behavior Analytics

The Sessionization Pattern

Crucial for web analytics, app usage tracking, and customer journey mapping

`LAG() + CASE + SUM() OVER`

✅ Web Analytics: "User sessions on website"
✅ App Usage: "Active sessions in mobile app"
✅ E-commerce: "Shopping cart abandonment"
✅ Support Tickets: "Customer interaction sessions"
✅ IoT Devices: "Device connectivity sessions"

🔁 **REPOST so others can level up too**

💡 Pro Tip: This pattern is frequently asked in **Data Analyst** and **Product Analyst** interviews!