Apache Spark 2 using SQL - Windowing Functions

1️⃣ What are Windowing Functions in Spark SQL?

A **window function** performs a calculation **across a set of rows** that are somehow related to the current row (like a sliding frame) — without collapsing the rows into one like a GROUP BY would.
Common uses:

- Ranking (e.g., **ROW_NUMBER, RANK, DENSE_RANK**)
- Running totals (SUM over a window)
- Moving averages
- Lead/Lag comparisons

2️⃣ Spark 2 Setup in Scala

In Spark 2, you can use window functions via:

```
import org.apache.spark.sql.expressions.Window
import org.apache.spark.sql.functions._
```

3️⃣ Example Dataset

Let's take a **sales dataset**:

```
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.expressions.Window
import org.apache.spark.sql.functions._

val spark = SparkSession.builder()
  .appName("Spark Window Functions Example")
  .master("local[*]")
  .getOrCreate()

spark.sparkContext.setLogLevel("ERROR")
import spark.implicits._
val salesDF = Seq(
  ("North", "2025-08-01", "ProductA", 1000),
  ("North", "2025-08-02", "ProductB", 1500),
```

```
  ("North", "2025-08-03", "ProductA", 800),
  ("South", "2025-08-01", "ProductA", 1200),
  ("South", "2025-08-02", "ProductB", 1000),
  ("South", "2025-08-03", "ProductA", 900)
).toDF("region", "date", "product", "amount")
```

4 Window Spec

We define **how** the window is partitioned and ordered:

```
val windowSpec = Window
  .partitionBy("region")  // group rows by region
  .orderBy("date")        // sort rows in each region by date
  .rowsBetween(Window.unboundedPreceding, Window.currentRow) // cumulative
```

5 Examples of Window Functions

a) Cumulative Sum of Sales

```
val withCumulative = salesDF.withColumn(
  "cumulative_amount",
  sum("amount").over(windowSpec)
)
withCumulative.show()
```

b) Ranking Products by Date per Region

```
val rankedDF = salesDF.withColumn(
  "rank_in_region",
  rank().over(windowSpec)
)
rankedDF.show()
```

c) Lead and Lag

```
val withLeadLag = salesDF
  .withColumn("prev_amount", lag("amount", 1).over(windowSpec))
  .withColumn("next_amount", lead("amount", 1).over(windowSpec))
withLeadLag.show()
```

d) Average over Last 2 Days

```
val movingAvgSpec = Window
  .partitionBy("region")
  .orderBy("date")
  .rowsBetween(-1, 0) // previous row and current row
val movingAvgDF = salesDF.withColumn(
  "2day_avg",
  avg("amount").over(movingAvgSpec)
)
movingAvgDF.show()
```

6 Using SQL Directly

You can also register the DataFrame as a temp view and use SQL syntax:

```
salesDF.createOrReplaceTempView("sales")
spark.sql("""
  SELECT region, date, product, amount,
        SUM(amount) OVER (PARTITION BY region ORDER BY date
                    ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS cumulative_amount
  FROM sales
""").show()
```