LAB 06: Advanced Transforms — PySpark & SQL

Duration: ~40 min | **Day:** 2 | **After module:** M06: Advanced Transforms | **Difficulty:** Intermediate-Advanced

---

Scenario

> *"Build analytical reports using window functions, CTEs, complex joins, and array operations. Create Gold-layer tables ready for BI consumption."*

Objectives

After completing this lab you will be able to: - Apply window functions (`ROW_NUMBER` , `SUM OVER`) - Write multi-step CTEs for complex analytics - Use correlated subqueries for filtering by aggregates - Flatten arrays with `explode()` - Create Gold tables using CTAS - Parse JSON strings with `from_json` - Segment data with `CASE WHEN` - Use higher-order functions (`transform` , `filter`)

Prerequisites

- Cluster running and attached to notebook
- Setup cell recreates Bronze tables if needed

Tasks Overview

Open **`LAB_06_code.ipynb`** and complete the `# TODO` cells.

Task	What to do	Key concept
Task 1	Rank Products by Revenue	`Window.orderBy(desc())`, `row_number()`
Task 2	Running Total	`SUM() OVER (PARTITION BY ... ORDER BY ... ROWS ...)`
Task 3	Multi-step CTE	`WITH daily_sales AS (...), ranked_days AS (...)`
Task 4	Correlated Subquery	`HAVING SUM() > (SELECT AVG(...))`
Task 5	Explode Array Column	`explode(col("array_col"))`
Task 6	CTAS — Create Gold Table	`CREATE TABLE ... AS SELECT ...`
Task 7	JSON Processing	`from_json()`, `schema_of_json()`, struct field access
Task 8	CASE WHEN — Segmentation	VIP (>500), Regular (100–500), Low (<100)
Task 9	Higher-Order Functions	`transform(array, x -> ...)`, `filter(array, x -> ...)`

Detailed Hints

Task 1: Window Functions

- `Window.orderBy(desc("total_revenue"))` for descending order
- `row_number().over(window_spec)` assigns unique sequential ranks

Task 2: Running Total

- `PARTITION BY customer_id` — reset per customer
- `ORDER BY order_datetime` — accumulate in chronological order
- `ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW` — standard running total

Task 3: CTE

- First CTE: `SUM(total_amount)` and `COUNT(*)` grouped by date
- Second CTE: `ROW_NUMBER() OVER (ORDER BY daily_revenue DESC)`

Task 5: Explode

- `explode(col("categories")).alias("category")` — one row per array element

Task 6: CTAS

- `CREATE TABLE {gold_table} AS SELECT ...`

Task 7: JSON

- Define schema with `StructType` matching JSON keys
- `from_json(col("json_col"), schema)` parses the string
- Access fields: `col("parsed.field_name")`

Task 8: CASE WHEN

- Thresholds: >500 = VIP, >=100 = Regular, else Low

Task 9: Higher-Order Functions

- `transform(array, x -> expression)` applies to each element
- `filter(array, x -> condition)` keeps matching elements

Summary

In this lab you: - Applied window functions (row_number, running SUM) - Written multi-step CTEs - Used subqueries to filter by aggregate conditions - Flattened arrays with explode() - Created Gold tables using CTAS - Parsed JSON strings with `from_json` - Used `CASE WHEN` for customer segmentation - Applied higher-order functions (`transform` , `filter`) on arrays

> **Exam Tip:** Know the difference: `ROW_NUMBER()` always gives unique sequential numbers. `RANK()` gives the same number for ties (with gaps). `DENSE_RANK()` gives same number for ties (no gaps).

> **What's next:** Day 3 starts with LAB 07 — building a Lakeflow Declarative Pipeline with Medallion architecture.