LAB 06: Advanced Transforms – PySpark & SQL

Duration: ~40 min
Day: 3
After module: M06: Advanced Transforms
Difficulty: Intermediate-Advanced

Scenario

> *"The RetailHub business team needs analytical reports: top products by revenue, customer spending trends, and category performance over time. You'll build these using window functions, CTEs, explode, and CTAS – both in PySpark and SQL."*

Objectives

After completing this lab you will be able to: - Use PySpark window functions (ROW_NUMBER, RANK, SUM OVER) - Write SQL CTEs (Common Table Expressions)
- Flatten nested JSON with `explode()` - Create tables using `CREATE TABLE AS SELECT` (CTAS) - Compare PySpark and SQL approaches side by side

---

Part 1: Window Functions (~15 min)

Task 1: Rank Products by Revenue (PySpark)

Use `Window.partitionBy("category").orderBy(desc("revenue"))` with `row_number()` to rank products within each category.

Task 2: Running Totals (SQL)

Write a SQL query with `SUM(amount) OVER (PARTITION BY customer_id ORDER BY order_date)` to compute a running total per customer.

<screen = Display output showing customer orders with a running_total column that accumulates per customer>

> **Exam Tip:** Window functions don't reduce rows (unlike GROUP BY). They add computed columns based on a "window" of related rows.

Part 2: CTE and Subqueries (~10 min)

Task 3: Multi-step CTE

Write a SQL query with multiple CTEs: 1. CTE `daily_sales` – aggregate orders by date 2. CTE `ranked_days` – rank days by total revenue 3. Final SELECT – top 5 days by revenue

Task 4: Correlated Subquery

Find customers whose total spending exceeds the average spending of all customers.

Part 3: Explode & JSON (~10 min)

Task 5: Explode Array Column

Given an orders table with an `items` array column, use `explode()` to create one row per item.

Task 6: Parse Nested JSON

Use `from_json()` to parse a JSON string column, then access nested fields.

Part 4: CTAS – Persist Results (~5 min)

Task 7: Create Gold Tables with CTAS

Create Gold-layer summary tables using `CREATE TABLE AS SELECT`: - `gold.top_products` – top 10 products by revenue - `gold.customer_segments` – customers segmented by spending

<screen = Catalog Explorer showing the new gold schema with top_products and customer_segments tables>

Summary

In this lab you: - Applied window functions (ROW_NUMBER, RANK, running SUM) in PySpark and SQL - Wrote multi-step CTEs for complex analytics - Flattened arrays with explode() - Created Gold tables with CTAS

> **What's next:** LAB 07 - Build a full Medallion pipeline in Lakeflow.