Lab 4 — Window Functions (Ranking, Deduplication, Running Totals)

This lab introduces beginners to window functions—one of the most powerful Spark transformation tools. The goal is to make learners comfortable with ranking, deduplication, and running totals.

Learning Objective

Learners will understand how window functions work in Spark and how they help analyze data across partitions without collapsing rows.

Learning Outcomes

- Use `Window.partitionBy()` and `Window.orderBy()`
- Apply ranking functions: `row_number`, `rank`, `dense_rank`
- Perform deduplication using window functions
- Compute running totals per customer

Dataset Requirement

Use **orders_raw.csv** from previous labs:

```
/Volumes/workspace/default/test/orders_raw.csv
```

Step-by-Step Lab Instructions

Step 1 — Read the Orders Dataset

```
orders_path = '/Volumes/workspace/default/test/orders_raw.csv'
df =
spark.read.option('header','true').option('inferSchema','true').csv(orders_path)
display(df)
```

Step 2 — Define Window Specification

```
from pyspark.sql.window import Window
from pyspark.sql import functions as F

window_spec = Window.partitionBy('customer_id').orderBy('order_value')
```

Step 3 — Calculate Order Value

```
df = df.withColumn('order_value', F.col('quantity') * F.col('price'))
display(df)
```

Step 4 — Ranking (row_number, rank, dense_rank)

```
ranked_df = df.withColumn('row_num', F.row_number().over(window_spec))
             .withColumn('rank', F.rank().over(window_spec))
             .withColumn('dense_rank', F.dense_rank().over(window_spec))

display(ranked_df)
```

Step 5 — Deduplication Using Window Functions

Keep the first order (lowest order_value) per customer.

```
dedup_df = ranked_df.filter(F.col('row_num') == 1)
display(dedup_df)
```

Step 6 — Running Total (Cumulative Sum per Customer)

```
total_window = Window.partitionBy('customer_id').orderBy('order_value')
                           .rowsBetween(Window.unboundedPreceding,
Window.currentRow)

running_total_df = df.withColumn('running_total',
F.sum('order_value').over(total_window))
display(running_total_df)
```

Step 7 — Write Output to Silver Layer

```
silver_path = '/Volumes/workspace/default/test/window_output'
running_total_df.write.format('delta').mode('overwrite').option('overwriteSchema','true').save(si
```

Post-Lab Practice

1. Create a window function that computes the highest order value per customer.
2. Use `lag()` to compare each order with the previous one.
3. Use `lead()` to compare with the next order.

End of Lab 4