Lab 3 — Basic Join Transformation (Beginner)

** This lab introduces learners to the concept of joining datasets using Spark DataFrames in Databricks. It is intentionally simple and beginner-friendly. Learners will join two small datasets: ***customers* and *orders*. They will explore different join types, perform a basic inner join, and write the joined output to the silver layer.

Learning Objective

Learners will understand how to combine data from multiple tables using joins and how joins help enrich datasets in real-world pipelines.

Learning Outcomes

After completing this lab, learners will be able to:

- Load two datasets from Volumes.
- Understand primary/foreign key relationships.
- Perform an **inner join** using a shared key (`customer_id`).
- Explore `left` and `right` joins.
- Create a unified enriched dataset containing both customer and order information.
- Write the joined output as a Delta table.

Dataset Description

You must upload the following two CSV files into the same location as previous labs:

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

customers_raw.csv (example structure)

- customer_id
- name
- email
- city

orders_raw.csv (created earlier)

- order_id
- customer_id

- product
- quantity
- price

Step-by-Step Lab Instructions

Step 1 — Load Both Datasets

```
customers_path = '/Volumes/workspace/default/test/customers_raw.csv'
orders_path = '/Volumes/workspace/default/test/orders_raw.csv'

df_customers =
spark.read.option('header','true').option('inferSchema','true').csv(customers_path)
df_orders =
spark.read.option('header','true').option('inferSchema','true').csv(orders_path)

display(df_customers)
display(df_orders)
```

Trainer: Explain the relationship between both datasets using `customer_id`.

Step 2 — Perform a Basic Inner Join

The inner join returns only matching `customer_id` records from both datasets.

```
from pyspark.sql import functions as F

joined_df = df_orders.join(df_customers, on='customer_id', how='inner')

display(joined_df)
joined_df.printSchema()
```

Trainer: Explain that inner join is the most common join because it keeps only valid matches.

Step 3 — Add a Derived Field (Optional but Helpful)

We calculate order value to enrich the dataset.

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

display(joined_df.limit(10))
```

Step 4 — Demonstrate Left Join (Trainer Demo Only)

```
left_join_df = df_customers.join(df_orders, on='customer_id', how='left')
display(left_join_df)
```

Trainer: Explain how left join keeps all customers even if they have no orders.

Step 5 — Demonstrate Right Join (Trainer Demo Only)

```
right_join_df = df_customers.join(df_orders, on='customer_id', how='right')
display(right_join_df)
```

Trainer: Explain how right join keeps all orders even if customer details are missing.

Step 6 — Write Final Joined Dataset to Silver

```
silver_path = '/Volumes/workspace/default/test/customer_orders_joined'

joined_df.write.format('delta').mode('overwrite').option('overwriteSchema','true').save(silver_pa
```

Trainer: Emphasize why an enriched joined table is valuable for analytics.

Step 7 — Verify the Output

```
spark.read.format('delta').load(silver_path).show()
```

Always verify output after writing.

Post-Lab Practice

1. Try a full outer join and see which records appear.
2. Add a column `order_category` based on price ranges.
3. Group the joined dataset by city and calculate total revenue per city.

End of Lab 3 — Basic Join Transformation (Beginner)