🧪Hands-On Exercise: Understanding Partitioning in Databricks / Delta Lake

🕖Learning Objectives

By the end of this lab, you will:

- Understand what partitioning means in Delta Lake.
- Create a CSV file with sample sales data.
- Load this CSV as a DataFrame.
- Write the DataFrame as a partitioned Delta table.
- Observe how Delta automatically creates sub-folders per partition.
- Query the data and understand partition pruning.

📁Step 1: Create Sample CSV Data

Create a CSV file named `sales_data.csv` using the below data:

```
order_id,country,year,amount
1,India,2024,5000
2,India,2023,4500
3,USA,2024,7000
4,USA,2023,6800
5,UK,2024,6500
6,UK,2023,6200
7,India,2024,5500
8,USA,2024,7100
9,UK,2023,6400
10,India,2023,4800
11,USA,2024,7300
12,UK,2024,6600
13,India,2024,5900
14,USA,2023,6900
15,India,2023,4300
```

Upload this file to DBFS (using **Add Data → Upload File**).

🧪 Step 2: Load CSV into a PySpark DataFrame

```
# Reading the uploaded CSV file
df_raw = spark.read.format("csv")
    .option("header", "true")
    .option("inferSchema", "true")
    .load("/FileStore/sales_data.csv")

df_raw.show()
```

Explanation:

- `header=true` → the first row contains column names.
- `inferSchema=true` → Spark detects correct data types.

🧪 Step 3: Write Data Without Partitioning (for comparison)

```
df_raw.write
    .format("delta")
    .mode("overwrite")
    .save("/mnt/delta/sales_unpartitioned")
```

This will create a single folder with parquet files.

🧪 Step 4: Write the Same Data With Partitioning

Partitioning by **country**:

```
df_raw.write
    .format("delta")
    .partitionBy("country")
    .mode("overwrite")
    .save("/mnt/delta/sales_partitioned")
```

Delta will automatically create sub-folders:

```
/mnt/delta/sales_partitioned/
    country=India/
    country=USA/
    country=UK/
```

🧪 Step 5: Read the Partitioned Table

```
df_part = spark.read.format("delta").load("/mnt/delta/sales_partitioned")
df_part.show()
```

🧪 Step 6: Query & Observe Partition Pruning

Query 1: Filter by Partition Column

```
df_part.where("country = 'India'").show()
```

Spark reads **only** the folder:

```
country=India/
```

This is called **Partition Pruning**.

🧪 Step 7: Query on Non-Partition Column (No Pruning)

```
df_part.where("year = 2023").show()
```

Since we did **not** partition by `year`, Spark must scan all folders.

Important

- Partitioning divides data into subdirectories based on the column values.

• Spark reads only required folders during queries.
• Improves speed, reduces cost, and optimizes cluster performance.
• Choose columns with **low or medium cardinality** that appear frequently in filters.
