- 🔹 SQL-style queries using Spark SQL
- 🔹 Temporary and permanent views
- 🔹 Parquet file handling
- 🔹 Nested JSON parsing
- 🔹 Conditional logic (`when`, `otherwise`)
- 🔹 `explode`, `arrays`, and `structs`

📂 Dataset: `sales_data.json` (nested JSON)

```
from pyspark.sql import Row

data = [
    Row(OrderID=101, Customer="Ali", Items=[{"Product":"Laptop", "Qty":1},
{"Product":"Mouse", "Qty":2}], Region="Asia", Amount=1200.0),
    Row(OrderID=102, Customer="Zara", Items=[{"Product":"Tablet", "Qty":1}],
Region="Europe", Amount=650.0),
    Row(OrderID=103, Customer="Mohan", Items=[{"Product":"Phone", "Qty":2},
{"Product":"Charger", "Qty":1}], Region="Asia", Amount=890.0),
    Row(OrderID=104, Customer="Sara", Items=[{"Product":"Desk", "Qty":1}],
Region="US", Amount=450.0)
]

df_sales = spark.createDataFrame(data)
df_sales.show(truncate=False)
```

🧪 PySpark Exercises – Set 4 (SQL, JSON, Advanced Functions)

🔹 Working with JSON & Nested Fields

1. Flatten the `Items` array using `explode()` to create one row per product.
2. Count total quantity sold per product.
3. Count number of orders per region.

🔹 Using `when` and `otherwise`

4. Create a new column `HighValueOrder`:
    - "Yes" if Amount > 1000
    - "No" otherwise
5. Add a column `ShippingZone`:
    - Asia → "Zone A", Europe → "Zone B", US → "Zone C"

🔹 Temporary & Permanent Views

6. Register `df_sales` as a temporary view named `sales_view`.
7. Write a SQL query to:

- Count orders by `Region`
- Find average amount per region

8. Create a permanent view using `saveAsTable()`.

SQL Queries via Spark

```
spark.sql("SELECT Region, COUNT(*) as OrderCount FROM sales_view GROUP BY Region").show()
```

9. Use SQL to filter all orders with more than 1 item.
10. Use SQL to extract customer names where Amount > 800.

Saving as Parquet and Reading Again

11. Save the exploded product-level DataFrame as a partitioned Parquet file by `Region`.
12. Read the parquet back and perform a group-by on `Product`.
