**Big-Data Pipeline on Databricks**

**Task:** Load Dataset → Flatten JSON → Write External Parquet Table

**Table of Contents**

1. Objective
2. Dataset & Tools
3. Step-by-Step Implementation  
    3.1 Upload JSON to DBFS  
    3.2 Read & Flatten JSON  
    3.3 Persist as Parquet  
    3.4 Register External Table
4. Results & Verification
5. Conclusion

**1 Objective**

Demonstrate an end-to-end data-engineering pipeline in Databricks:

* Ingest a raw (nested) JSON dataset into **DBFS**
* Flatten nested structures with **PySpark**
* Persist the cleansed data as **Parquet** and register it as an **external table** for downstream analytics

**2 Dataset & Tools**

**Dataset**

* Sample nested JSON: people.json  
  <https://raw.githubusercontent.com/apache/spark/master/examples/src/main/resources/people.json>

**Platform / Libraries**

* Databricks Community / Azure Databricks
* Apache Spark 3.x (PySpark)
* DBFS (Databricks File System)

**3 Step-by-Step Implementation**

**3.1 Upload JSON into DBFS**

**A. Copy from a public URL**

dbutils.fs.cp(

"https://raw.githubusercontent.com/apache/spark/master/examples/src/main/resources/people.json",

"dbfs:/FileStore/nested/people.json")

**B. Upload via the UI**  
*Data ► Add Data ► Upload File ► Target Location = /FileStore/nested/*

**3.2 Read & Flatten Nested JSON**

from pyspark.sql.functions import col, explode

# Read raw JSON

raw = (spark.read

.option("multiLine", True)

.json("dbfs:/FileStore/nested/people.json"))

# Flatten struct + explode array

flat = (raw

.select("id", "name",

col("address.city").alias("city"),

col("address.state").alias("state"),

explode("orders").alias("order"))

.select("id", "name", "city", "state",

col("order.id").alias("order\_id"),

col("order.amount").alias("order\_amount")))

flat.show(5)

**3.3 Persist as Parquet**

target = "dbfs:/mnt/flattened/people\_parquet"

flat.write.mode("overwrite").parquet(target)

**3.4 Register External Table**

spark.sql(f"""

CREATE TABLE IF NOT EXISTS people\_flattened

USING PARQUET

OPTIONS (path '{target}')

""")

# Quick sanity query

spark.sql("SELECT \* FROM people\_flattened LIMIT 10").display()

**4 Results & Verification**

The external table **people\_flattened** is now queryable:

SELECT city,

COUNT(\*) AS orders

FROM people\_flattened

GROUP BY city

ORDER BY orders DESC;

**5 Conclusion**

The ingestion → transformation → persistence pipeline was successfully completed.  
Nested JSON data was flattened, stored in an optimized Parquet format, and exposed as an external table for analytics within Databricks.