**Create External Read & Write tables using File format(Delta, Parquet, JSON & CSV) & Supporting DML Operations**

**Notebook: Create External Read & Write tables (Delta, Parquet, JSON, CSV) & DML demo**

**1) Mount ADLS Gen2 (optional — choose method you prefer)**

****

**2) Create PySpark DataFrame from sample CSV(s), clean, and write out in 4 formats**

# Databricks Python cell

from pyspark.sql import SparkSession

from pyspark.sql.types import StructType, StructField, StringType

# Define schema

schema = StructType([

    StructField("employee\_id", StringType(), True),

    StructField("employee\_name", StringType(), True),

    StructField("Pincode", StringType(), True),

    StructField("Contact\_No", StringType(), True),

    StructField("State", StringType(), True),

    StructField("Country", StringType(), True)

])

# Sample data

data = [

    ("1001", "John Smith", "560001", "9876543210", "Karnataka", "India"),

    ("1002", "Priya Sharma", "400001", "9876500001", "Maharashtra", "India"),

    ("1003", "Michael Johnson", "94102", "9876500002", "California", "USA"),

    ("1004", "Emma Brown", "2000", "9876500003", "NSW", "Australia"),

    ("1005", "David Wilson", "10115", "9876500004", "Berlin", "Germany")

]

# Create DataFrame

df = spark.createDataFrame(data, schema)

# Path where Step 2 will read the CSV

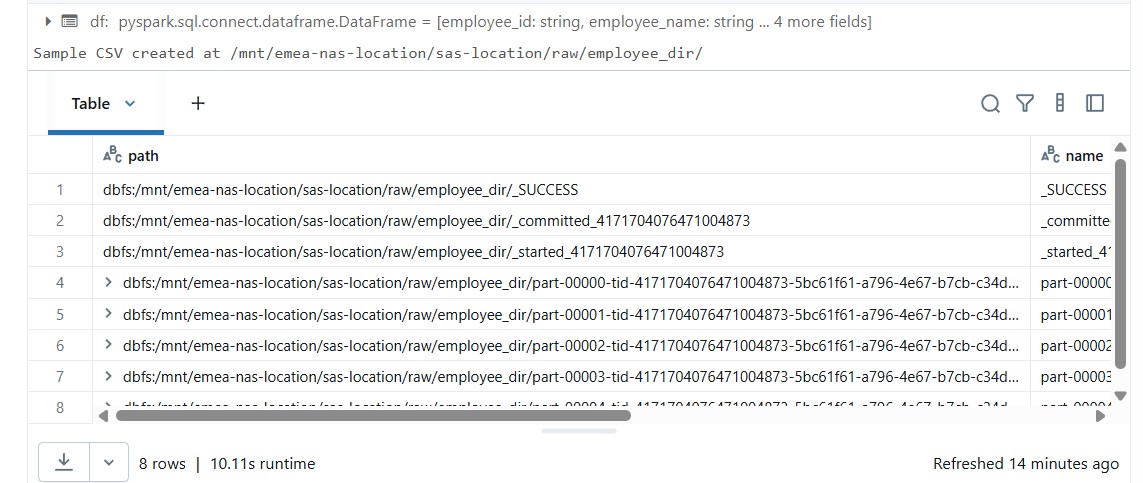
output\_path = "/mnt/emea-nas-location/sas-location/raw/employee\_dir/"

# Write CSV file (overwrite if exists)

df.write.mode("overwrite").option("header", "true").csv(output\_path)

print(f"Sample CSV created at {output\_path}")

display(dbutils.fs.ls(output\_path))

****

# Python / PySpark cell

from pyspark.sql.types import StructType, StructField, StringType

from pyspark.sql.functions import trim, regexp\_replace, col, lit

schema = StructType([

StructField('employee\_id', StringType(), True),

StructField('employee\_name', StringType(), True),

StructField('Pincode', StringType(), True),

StructField('Contact\_No', StringType(), True),

StructField('State', StringType(), True),

StructField('Country', StringType(), True),

])

# Adjust the input path to the folder containing input CSV(s)

input\_path = "/mnt/emea-nas-location/sas-location/raw/employee\_dir/\*.csv"

df = spark.read.format('csv') \

.option("header", "true") \

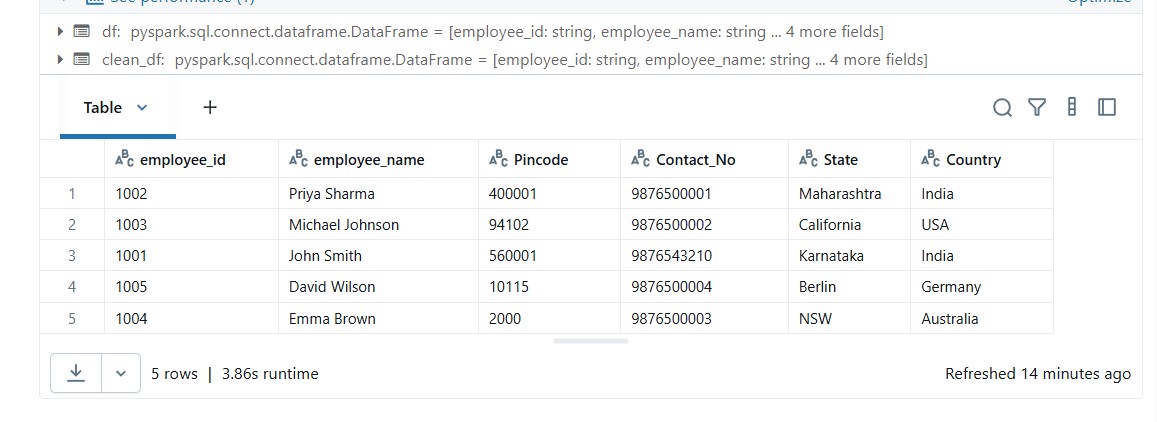
.schema(schema) \

.load(input\_path)

# Clean data: trim & remove quotes

clean\_df = df.select([trim(regexp\_replace(col(c), r'[\"\\\']', "")).alias(c) for c in df.columns])

display(clean\_df.limit(10))



# Write out to Parquet, Delta, CSV, JSON (partitioned by Country)

base\_out = "/mnt/emea-nas-location/sas-location/raw/employee\_{}"

# Parquet

clean\_df.write.format("parquet").partitionBy("Country").mode("overwrite")\

.save(base\_out.format("parquet"))

# Delta

clean\_df.write.format("delta").partitionBy("Country").mode("overwrite")\

.save(base\_out.format("delta"))

# CSV (note: when writing partitioned CSV, subfolders created)

clean\_df.write.format("csv").option("header","true").partitionBy("Country").mode("overwrite")\

.save(base\_out.format("csv"))

# JSON

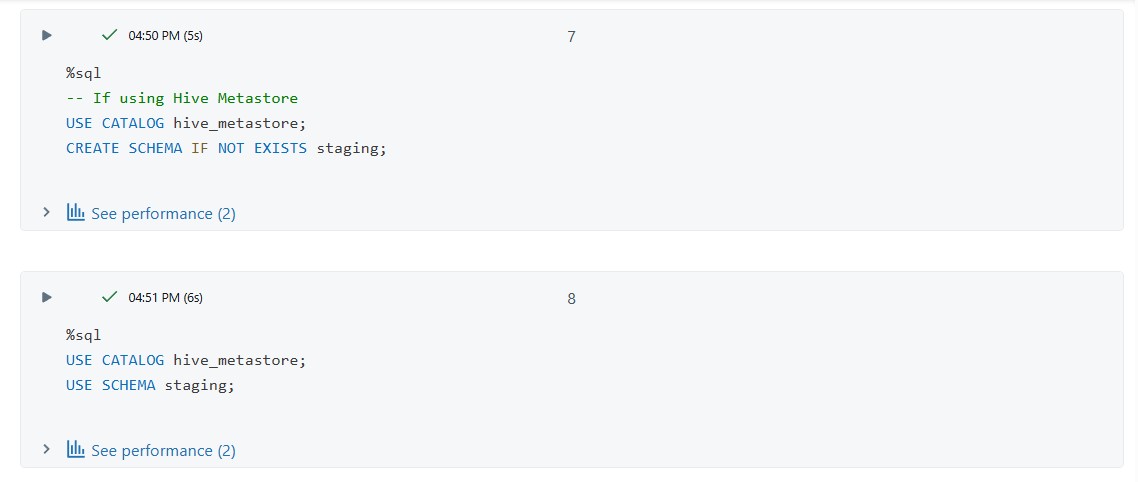
clean\_df.write.format("json").partitionBy("Country").mode("overwrite")\

.save(base\_out.format("json"))

print("Writes complete. Check the folders under:", base\_out.format("\*"))



**3) Use Hive Metastore (or Unity Catalog) — choose the right catalog/schema**

****

**4) CREATE EXTERNAL TABLE (Delta) — using data source (recommended for Delta)**

****

**5) CREATE EXTERNAL TABLE (Parquet) — USING data source**

****

**6) CREATE EXTERNAL TABLE (CSV)**

****

**7) CREATE EXTERNAL TABLE (JSON)**

****

**8) Verify reads (quick SELECT checks)**

-- Try selects

SELECT count(\*) FROM EMPLOYEE\_DELTA\_EXT;

SELECT \* FROM EMPLOYEE\_PARQUET\_EXT LIMIT 10;

SELECT \* FROM EMPLOYEE\_CSV\_EXT LIMIT 10;

SELECT \* FROM EMPLOYEE\_JSON\_EXT LIMIT 10;

