Delta Lake on Databricks

# GCS Authentication Configuration

spark.conf.set("spark.hadoop.google.cloud.auth.service.account.enable", "true")  
spark.conf.set("spark.hadoop.fs.gs.auth.service.account.email", "databricks-compute@calcium-field-455700-p2.iam.gserviceaccount.com")  
spark.conf.set("spark.hadoop.fs.gs.project.id", "calcium-field-455700-p2")  
spark.conf.set("spark.hadoop.fs.gs.auth.service.account.private.key", "[PRIVATE KEY OMITTED FOR SECURITY]")  
spark.conf.set("spark.hadoop.fs.gs.auth.service.account.private.key.id", "cc792d2f8ef8022756e4bdf8e1f3c2105aa4ec1c")

# Step 1: Read Data from Orders.csv

df = spark.read.csv("gs://heart\_raw-bronze/Orders.csv", header=True, inferSchema=True)  
df.show(25)

# Step 2: Add Ingestion Timestamp Column

from pyspark.sql.functions import current\_timestamp  
df = df.withColumn("ingestion\_timestamp", current\_timestamp())  
df.show(25)

# Step 3: Write to Delta Format (Initial Load)

df.write.format("delta").mode("overwrite").save("gs://heart\_raw-bronze/delta/sales")

# Step 4: Upsert Using Delta Merge (Full Update)

from delta.tables import DeltaTable  
  
existing\_data = DeltaTable.forPath(spark, "gs://heart\_raw-bronze/delta/sales")  
existing\_data.alias("existing").merge(  
 df.alias("new"),  
 "existing.OrderID = new.OrderID"  
).whenMatchedUpdateAll() .whenNotMatchedInsertAll() .execute()

# Validate the Delta Table

spark.read.format("delta").load("gs://heart\_raw-bronze/delta/sales").show()

# Update Subset of Columns (Only Quantity & TotalPrice)

from delta.tables import DeltaTable  
  
delta\_table = DeltaTable.forPath(spark, "gs://heart\_raw-bronze/delta/sales")  
delta\_table.alias("target").merge(  
 df.alias("source"),  
 "target.OrderID = source.OrderID"  
).whenMatchedUpdate(  
 set={  
 "Quantity": "source.Quantity",  
 "TotalPrice": "source.TotalPrice"  
 }  
).whenNotMatchedInsertAll() .execute()

# Enable Delta Table History

DeltaTable.forPath(spark, "gs://heart\_raw-bronze/delta/sales").history().show()

# Read Specific Version or Timestamp

# Version 0  
spark.read.format("delta") .option("versionAsOf", 0) .load("gs://heart\_raw-bronze/delta/sales") .show()  
  
# Timestamp after initial save  
spark.read.format("delta") .option("timestampAsOf", "2025-05-26T22:41:00") .load("gs://heart\_raw-bronze/delta/sales") .show()

# Simulate Day 2 Incremental Load

from pyspark.sql import Row  
from pyspark.sql.functions import current\_timestamp  
  
day2\_data = [  
 Row(OrderID=2, CustomerID=1002, OrderDate="2025-01-02", ProductID=2, ProductName="Smartphone", Quantity=2, UnitPrice=800, TotalPrice=1600),  
 Row(OrderID=11, CustomerID=1004, OrderDate="2025-02-10", ProductID=3, ProductName="Headphones", Quantity=2, UnitPrice=100, TotalPrice=200)  
]  
df\_day2 = spark.createDataFrame(day2\_data)  
df\_day2 = df\_day2.withColumn("ingestion\_timestamp", current\_timestamp())  
df\_day2.show()

# MERGE Day 2 Data into Delta Table

delta\_table = DeltaTable.forPath(spark, "gs://heart\_raw-bronze/delta/sales")  
delta\_table.alias("target").merge(  
 df\_day2.alias("source"),  
 "target.OrderID = source.OrderID"  
).whenMatchedUpdateAll() .whenNotMatchedInsertAll() .execute()

# Check Delta Table History After Merge

DeltaTable.forPath(spark, "gs://heart\_raw-bronze/delta/sales").history().show()

# Time Travel to Version 0 and 1

spark.read.format("delta").option("versionAsOf", 0).load("gs://heart\_raw-bronze/delta/sales").show()  
spark.read.format("delta").option("versionAsOf", 1).load("gs://heart\_raw-bronze/delta/sales").show()

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.