INDIAN DATA CLUB

CODE BASICS

databricks

14 DAYS

AI CHALLENGE

DAY 07

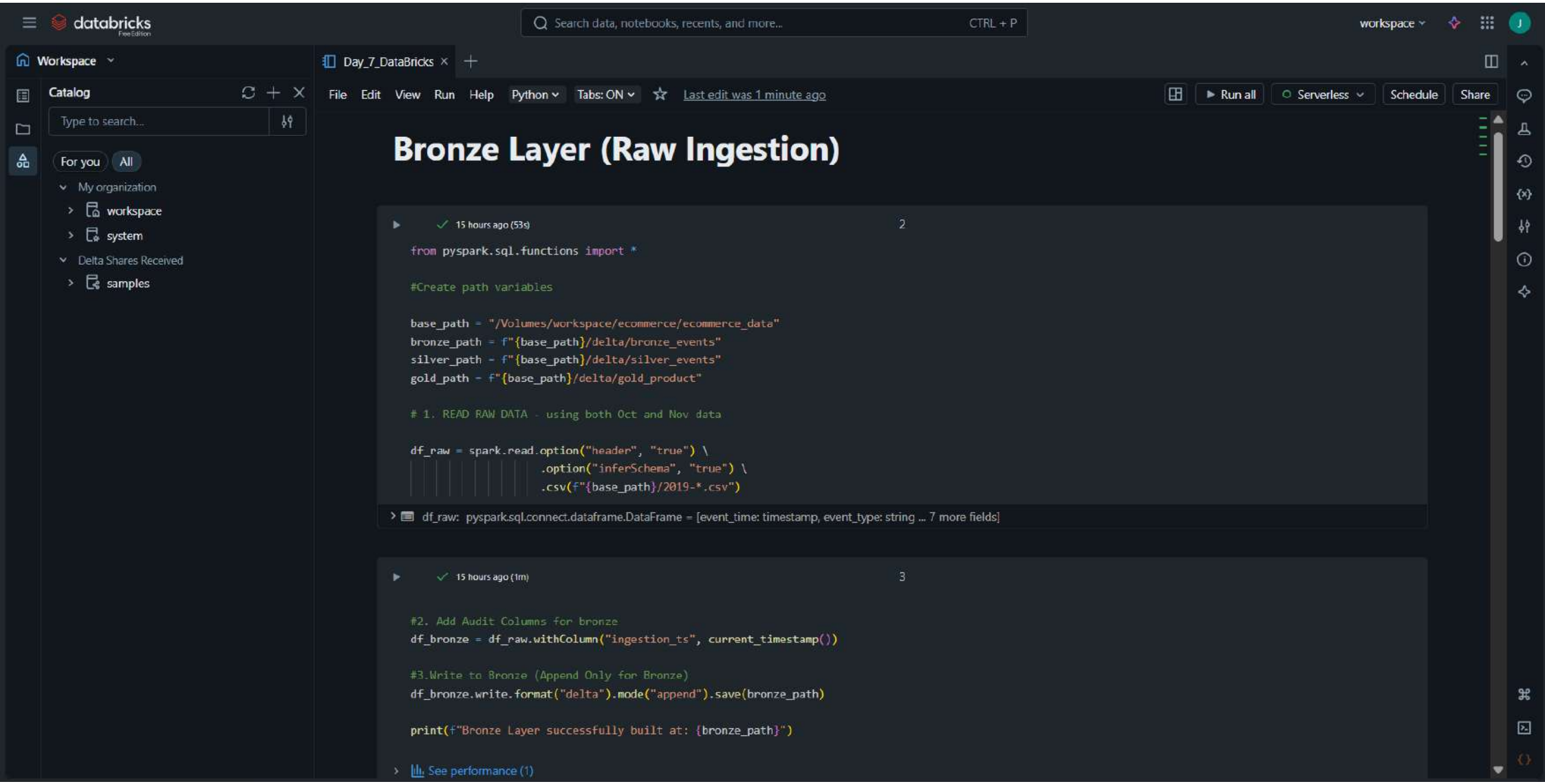
Topic:

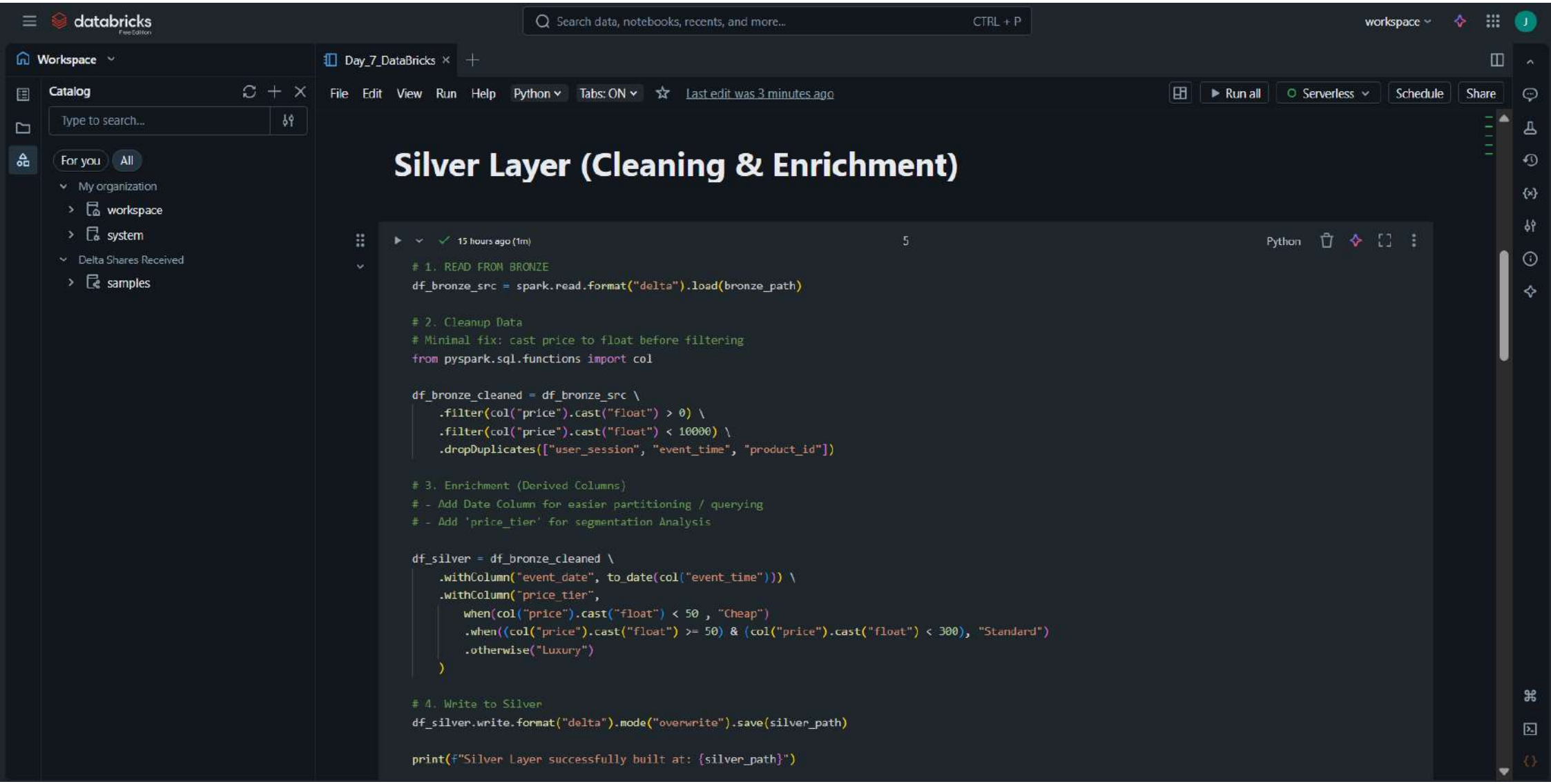
Workflows & Job Orchestration

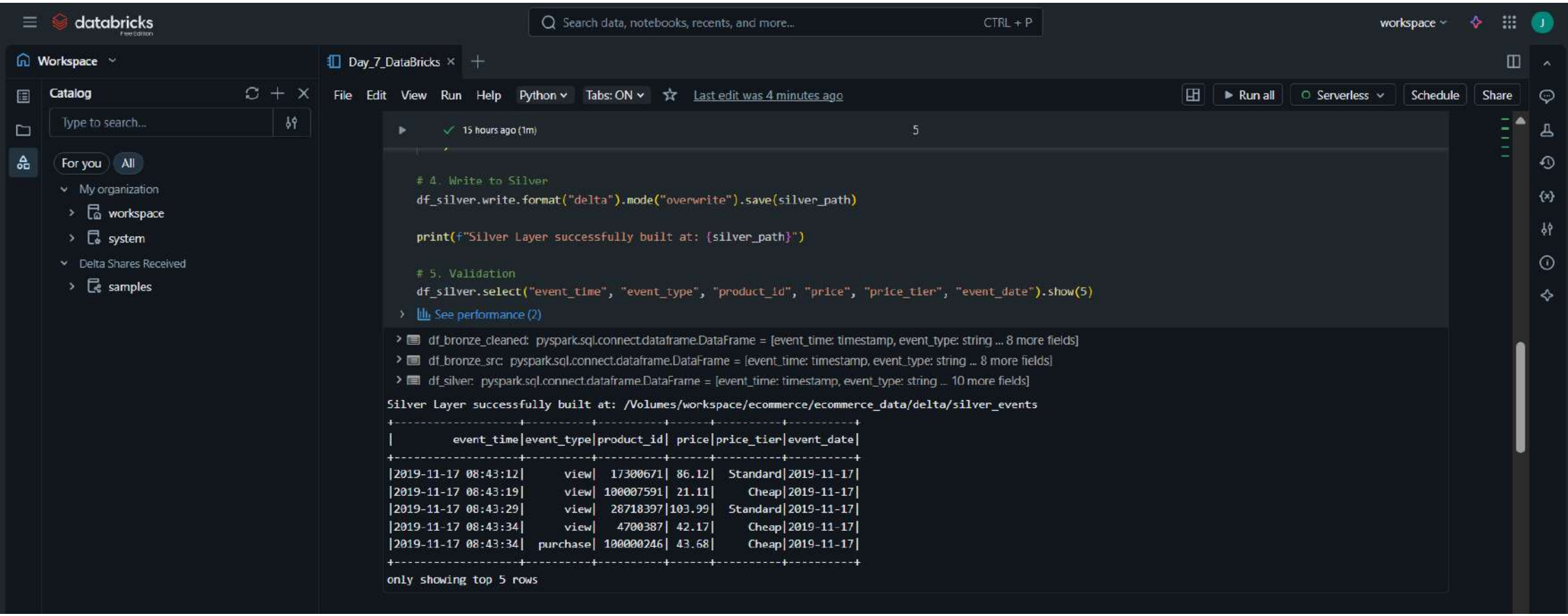
Challenge:

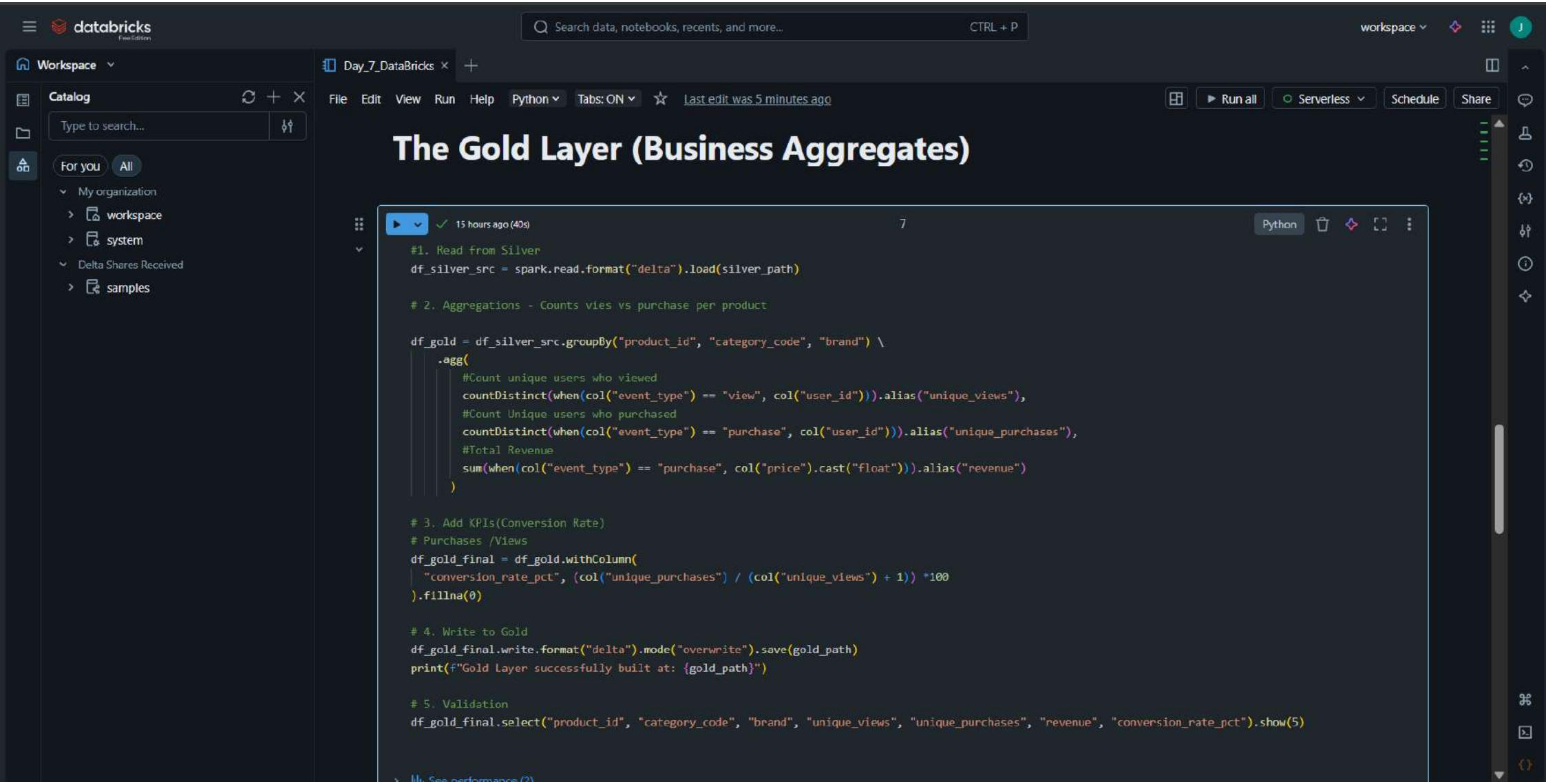
1. Add parameter widgets to notebooks
2. Create multi-task job (Bronze→Silver→Gold)
3. Set up dependencies
4. Schedule execution

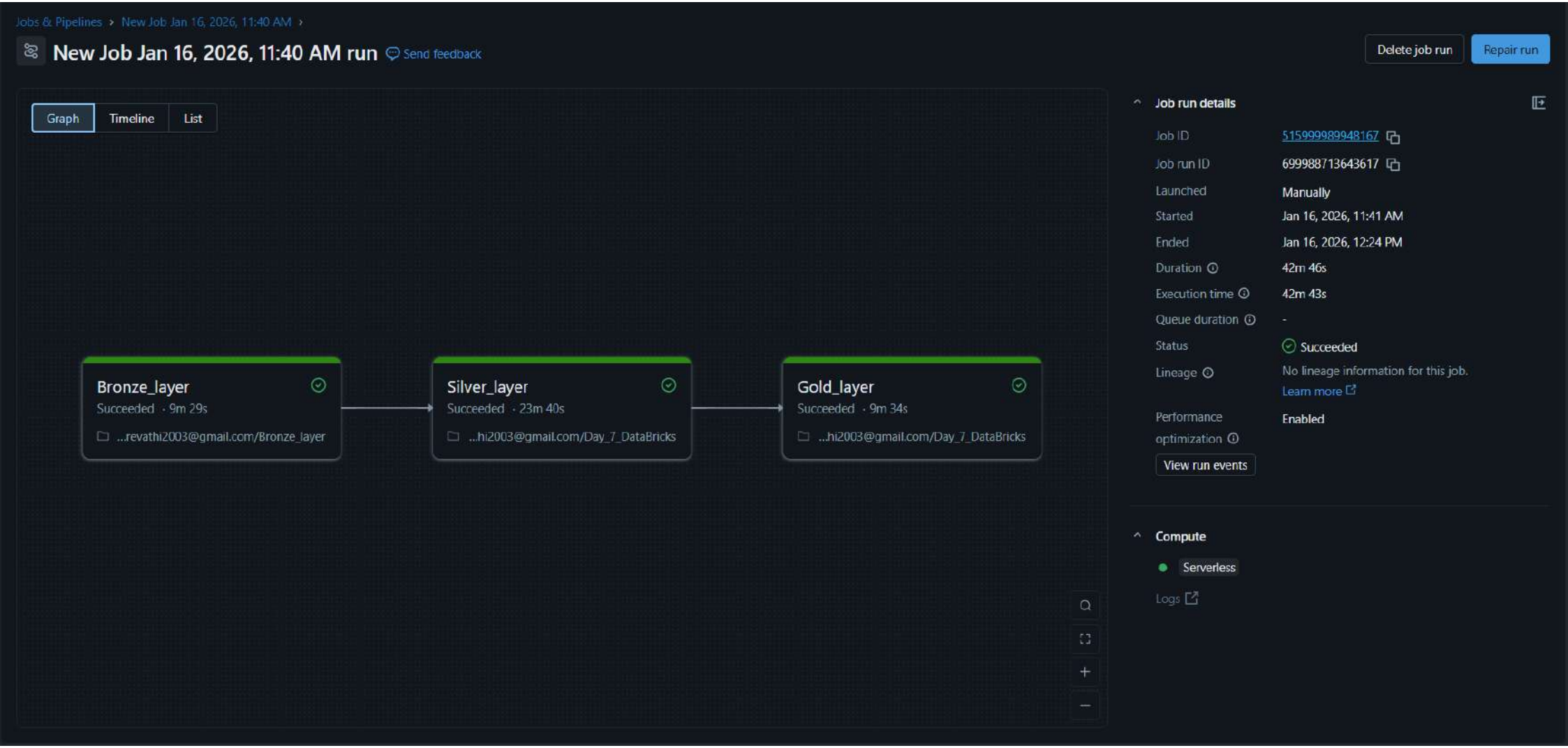
#DatabricksWithIDC

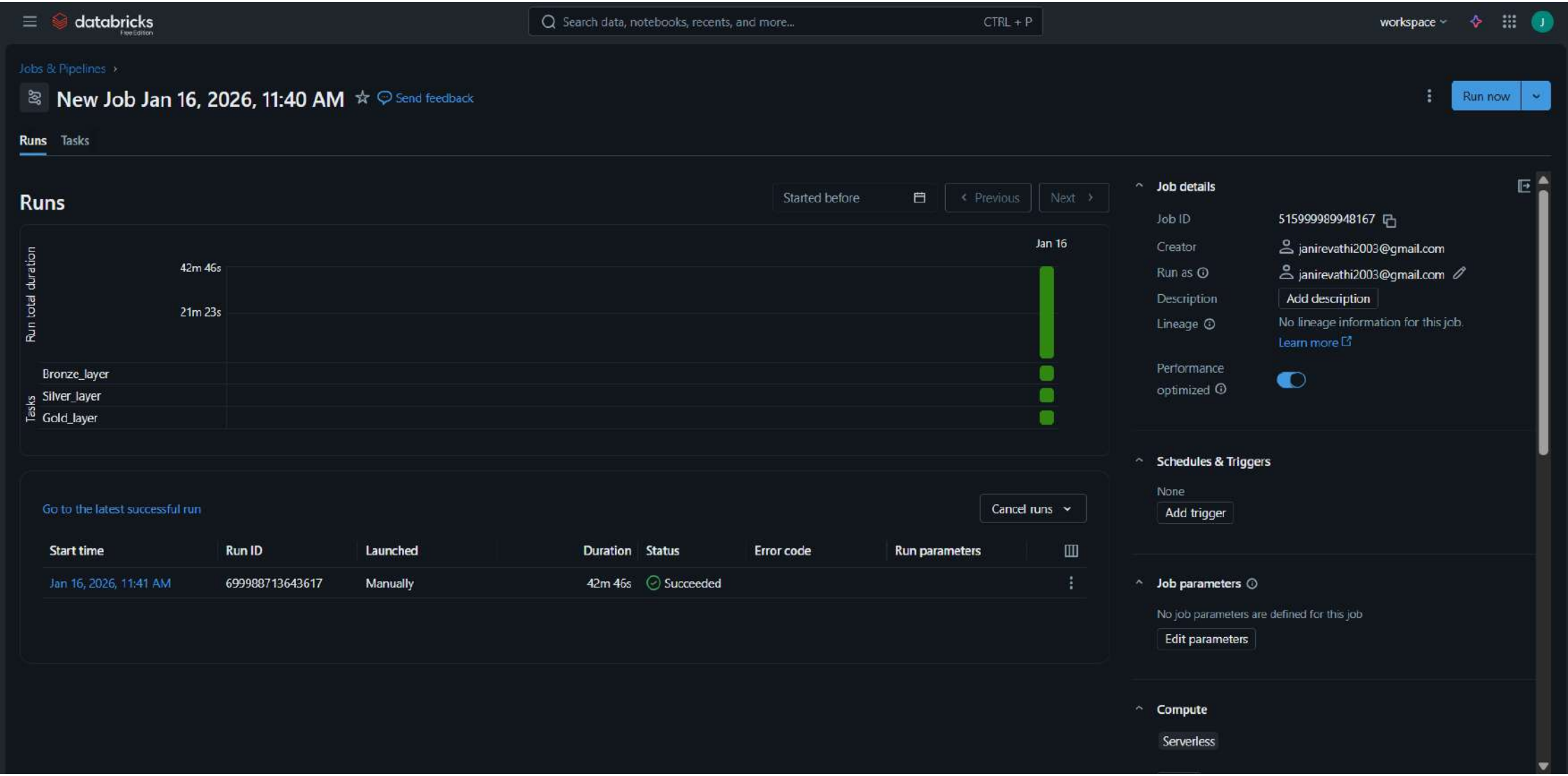

databricks
Free Edition
Search data, notebooks, recents, and more...
CTRL + P
workspace
Workspace
Day_7_DataBricks
Catalog
Type to search...
For you
All
My organization
workspace
system
Delta Shares Received
samples
File Edit View Run Help Python Tabs: ON
Last edit was 1 minute ago
Run all
Serverless
Schedule
Share
Bronze Layer (Raw Ingestion)
15 hours ago (53s)
2
from pyspark.sql.functions import *
#Create path variables
base_path = "/Volumes/workspace/ecommerce/ecommerce_data"
bronze_path = f"{base_path}/delta/bronze_events"
silver_path = f"{base_path}/delta/silver_events"
gold_path = f"{base_path}/delta/gold_product"
# 1. READ RAW DATA - using both Oct and Nov data
df_raw = spark.read.option("header", "true") \
.option("inferSchema", "true") \
.csv(f"{base_path}/2019-*.csv")
df_raw: pyspark.sql.connect.dataframe.DataFrame = [event_time: timestamp, event_type: string ... 7 more fields]
15 hours ago (1m)
3
#2. Add Audit Columns for bronze
df_bronze = df_raw.withColumn("ingestion_ts", current_timestamp())
#3.Write to Bronze (Append Only for Bronze)
df_bronze.write.format("delta").mode("append").save(bronze_path)
print(f"Bronze Layer successfully built at: {bronze_path}")
See performance (1)


databricks
Search data, notebooks, recents, and more... CTRL + P
workspace
Workspace
Day_7_DataBricks
Catalog
Type to search...
For you
All
My organization
workspace
system
Delta Shares Received
samples
File Edit View Run Help Python Tabs: ON
Last edit was 3 minutes ago
Run all
Serverless
Schedule
Share
Silver Layer (Cleaning & Enrichment)
15 hours ago (1m)
5
Python
# 1. READ FROM BRONZE
df_bronze_src = spark.read.format("delta").load(bronze_path)
# 2. Cleanup Data
# Minimal fix: cast price to float before filtering
from pyspark.sql.functions import col
df_bronze_cleaned = df_bronze_src \
.filter(col("price").cast("float") > 0) \
.filter(col("price").cast("float") < 10000) \
.dropDuplicates(["user_session", "event_time", "product_id"])
# 3. Enrichment (Derived Columns)
# - Add Date Column for easier partitioning / querying
# - Add 'price_tier' for segmentation Analysis
df_silver = df_bronze_cleaned \
.withColumn("event_date", to_date(col("event_time"))) \
.withColumn("price_tier",
when(col("price").cast("float") < 50 , "Cheap")
.when((col("price").cast("float") >= 50) & (col("price").cast("float") < 300), "Standard")
.otherwise("Luxury")
)
# 4. Write to Silver
df_silver.write.format("delta").mode("overwrite").save(silver_path)
print(f"Silver Layer successfully built at: {silver_path}")


databricks
Search data, notebooks, recents, and more...
CTRL + P
workspace
Workspace
Day_7_DataBricks
Catalog
Type to search...
For you
All
My organization
workspace
system
Delta Shares Received
samples
File Edit View Run Help Python Tabs: ON
Last edit was 4 minutes ago
Run all
Serverless
Schedule
Share
15 hours ago (1m)
5
# 4. Write to Silver
df_silver.write.format("delta").mode("overwrite").save(silver_path)
print(f"Silver Layer successfully built at: {silver_path}")
# 5. Validation
df_silver.select("event_time", "event_type", "product_id", "price", "price_tier", "event_date").show(5)
See performance (2)
df_bronze_cleaned: pyspark.sql.connect.dataframe.DataFrame = [event_time: timestamp, event_type: string ... 8 more fields]
df_bronze_src: pyspark.sql.connect.dataframe.DataFrame = [event_time: timestamp, event_type: string ... 8 more fields]
df_silver: pyspark.sql.connect.dataframe.DataFrame = [event_time: timestamp, event_type: string ... 10 more fields]
Silver Layer successfully built at: /Volumes/workspace/ecommerce/ecommerce_data/delta/silver_events
+-------------------+----------+----------+------+----------+----------+
| event_time|event_type|product_id| price|price_tier|event_date|
+-------------------+----------+----------+------+----------+----------+
|2019-11-17 08:43:12| view| 17300671| 86.12| Standard|2019-11-17|
|2019-11-17 08:43:19| view| 100007591| 21.11| Cheap|2019-11-17|
|2019-11-17 08:43:29| view| 28718397|103.99| Standard|2019-11-17|
|2019-11-17 08:43:34| view| 4700387| 42.17| Cheap|2019-11-17|
|2019-11-17 08:43:34| purchase| 100000246| 43.68| Cheap|2019-11-17|
+-------------------+----------+----------+------+----------+----------+
only showing top 5 rows


databricks
Search data, notebooks, recents, and more...
CTRL + P
workspace
Workspace
Day_7_DataBricks
Catalog
Type to search...
For you
All
My organization
workspace
system
Delta Shares Received
samples
File Edit View Run Help Python Tabs: ON
Last edit was 5 minutes ago
Run all
Serverless
Schedule
Share
The Gold Layer (Business Aggregates)
15 hours ago (40s)
7
Python
#1. Read from Silver
df_silver_src = spark.read.format("delta").load(silver_path)
# 2. Aggregations - Counts vies vs purchase per product
df_gold = df_silver_src.groupBy("product_id", "category_code", "brand") \
.agg(
#Count unique users who viewed
countDistinct(when(col("event_type") == "view", col("user_id"))).alias("unique_views"),
#Count Unique users who purchased
countDistinct(when(col("event_type") == "purchase", col("user_id"))).alias("unique_purchases"),
#Total Revenue
sum(when(col("event_type") == "purchase", col("price").cast("float"))).alias("revenue")
)
# 3. Add KPIs(Conversion Rate)
# Purchases /Views
df_gold_final = df_gold.withColumn(
"conversion_rate_pct", (col("unique_purchases") / (col("unique_views") + 1)) *100
).fillna(0)
# 4. Write to Gold
df_gold_final.write.format("delta").mode("overwrite").save(gold_path)
print(f"Gold Layer successfully built at: {gold_path}")
# 5. Validation
df_gold_final.select("product_id", "category_code", "brand", "unique_views", "unique_purchases", "revenue", "conversion_rate_pct").show(5)


Jobs & Pipelines › New Job Jan 16, 2026, 11:40 AM ›
New Job Jan 16, 2026, 11:40 AM run
Send feedback
Delete job run
Repair run
Graph
Timeline
List
Bronze_layer
Succeeded · 9m 29s
...revathi2003@gmail.com/Bronze_layer
Silver_layer
Succeeded · 23m 40s
...hi2003@gmail.com/Day_7_DataBricks
Gold_layer
Succeeded · 9m 34s
...hi2003@gmail.com/Day_7_DataBricks
Job run details
Job ID
515999989948167
Job run ID
699988713643617
Launched
Manually
Started
Jan 16, 2026, 11:41 AM
Ended
Jan 16, 2026, 12:24 PM
Duration
42m 46s
Execution time
42m 43s
Queue duration
-
Status
Succeeded
Lineage
No lineage information for this job.
Learn more
Performance optimization
Enabled
View run events
Compute
Serverless
Logs


databricks
Free Edition
Search data, notebooks, recents, and more...
CTRL + P
workspace
Jobs & Pipelines
New Job Jan 16, 2026, 11:40 AM
Send feedback
Run now
Runs
Tasks
Runs
Started before
Previous
Next
Jan 16
Run total duration
42m 46s
21m 23s
Tasks
Bronze_layer
Silver_layer
Gold_layer
Go to the latest successful run
Cancel runs
Start time
Run ID
Launched
Duration
Status
Error code
Run parameters
Jan 16, 2026, 11:41 AM
699988713643617
Manually
42m 46s
Succeeded
Job details
Job ID
515999989948167
Creator
janirevathi2003@gmail.com
Run as
janirevathi2003@gmail.com
Description
Add description
Lineage
No lineage information for this job.
Learn more
Performance optimized
Schedules & Triggers
None
Add trigger
Job parameters
No job parameters are defined for this job
Edit parameters
Compute
Serverless