

Databricks Lakehouse Portfolio

JDE Aerospace Data Platform on Azure Databricks

John Paul (JP) Castro

Senior Data Platform Architect

25+ Years | Aerospace, Entertainment, Telecommunications

jpcenterprises.com | github.com/johnpaul-castro

johnpaulcastro@gmail.com | (818) 943-5159

Built on Databricks 14-Day Trial | May 2026

Unity Catalog | Lakeflow Declarative Pipelines | Serverless SQL

Contents

1. Architecture Overview
2. Bronze Layer: Raw Ingestion
3. Silver Layer: Cleansed + Validated
4. Gold Layer: Business Models
5. Lakeflow Declarative Pipeline (DLT)
6. Workflow Orchestration
7. Unity Catalog RBAC
8. Data Quality Framework
9. AI/BI Dashboards
10. Genie Spaces
11. Query Profiling
12. Data Ingestion Connectors

1. Architecture Overview

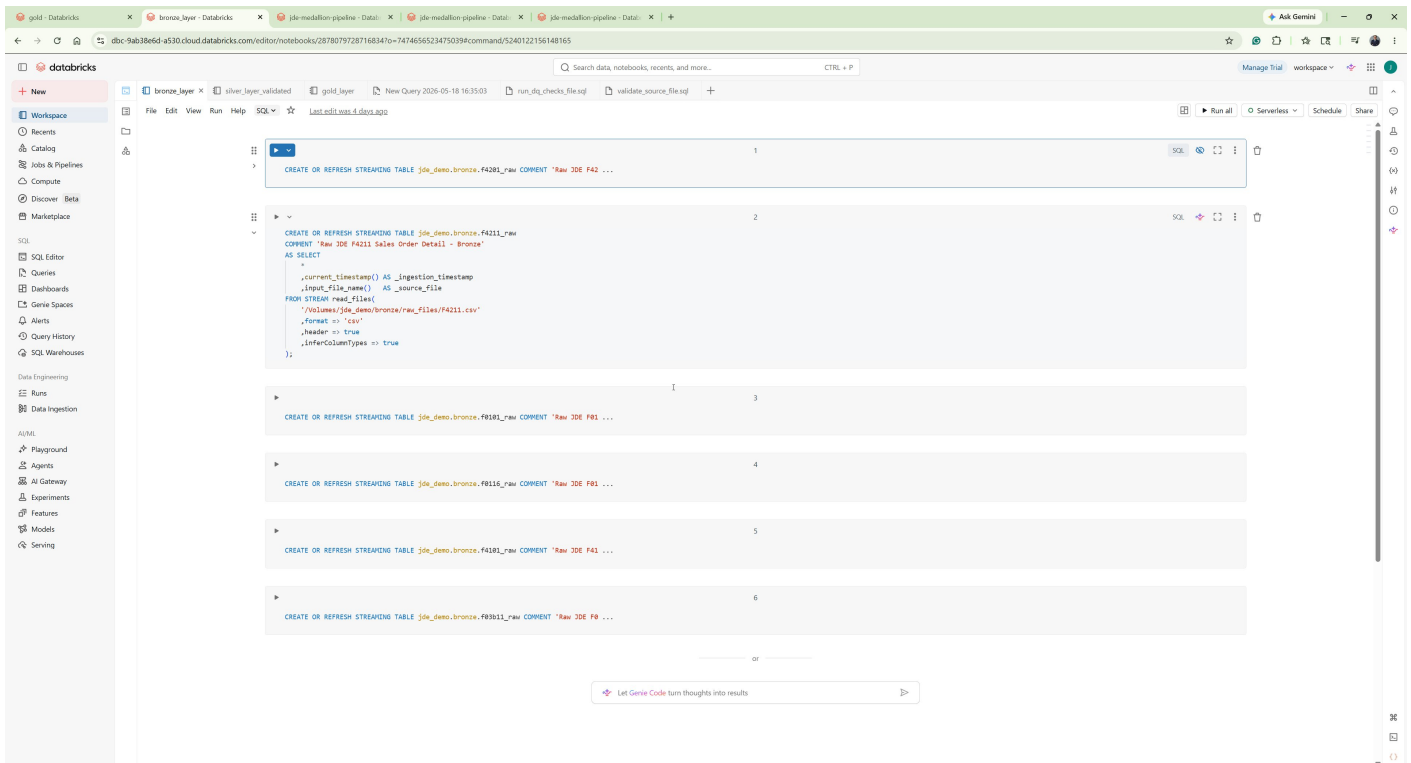
This portfolio demonstrates a complete Databricks Lakehouse implementation built on a 14-day trial workspace. The platform ingests JDE (JD Edwards) ERP data from six source tables representing a realistic aerospace distribution environment, processes it through a medallion architecture (Bronze, Silver, Gold), and delivers analytics through AI/BI dashboards and Genie Spaces.

- **6 JDE source tables:** F4201 (Sales Order Header), F4211 (Sales Order Detail), F0101 (Address Book), F0116 (Phone/Address), F4101 (Item Master), F03B11 (Accounts Receivable)
- **Medallion architecture:** Bronze (raw streaming), Silver (cleansed + validated), Gold (business models + DQ)
- **22 pipeline tables:** 6 Bronze streaming tables, 6 Silver materialized views, 10 Gold materialized views/streaming tables
- **Infrastructure:** Unity Catalog, Serverless compute, Lakeflow Declarative Pipelines, Databricks Workflows
- **Governance:** 4 role-based groups with schema-level GRANT permissions, column sensitivity tags
- **Analytics:** 4-tab AI/BI dashboard, Genie Space with natural language queries, DQ monitoring

2. Bronze Layer: Raw Ingestion

The Bronze layer uses CREATE OR REFRESH STREAMING TABLE statements to ingest CSV files from a Unity Catalog Volume. Each table reads from /Volumes/jde_demo/bronze/raw_files/ using STREAM read_files() with CSV format, header detection, and type inference. Metadata columns (_ingestion_timestamp, _source_file) are added automatically.

Bronze Layer Notebook



Bronze layer notebook: 6 streaming table definitions (F4201, F4211, F0101, F0116, F4101, F03B11) using CREATE OR REFRESH STREAMING TABLE with STREAM read_files(). F4211 expanded showing CSV format, header/inferColumnTypes options.

Bronze Raw Data: F4211 (Sales Order Detail)

The screenshot shows a Databricks workspace with a SQL query executed. The query is: `SELECT * FROM jde_demo.bronze.f4211_raw LIMIT 10`. The result is a table with 10 rows and 25 columns. The columns are: `r3_SDDOCO`, `r3_SDDCTO`, `i2_SDKCOO`, `i2_SSDUND`, `r3_SDSANB`, `r3_SDSHAN`, `r3_SSDITM`, `r3_SSDITM`, `r3_SSDITM`, `r3_SDDSC1`, `r3_SDMNCU`, `r3_SDUOM`, `i2_SDSOOS`, `i2_SDSHPN`, `i2_SSDPRC`, `i2_SSDADP`, `i2_SSONTR`, `i2_SSDLTR`, `r3_SSDUMU`, and `r3_prescd_data`. The data in the table consists of various alphanumeric codes and numbers.

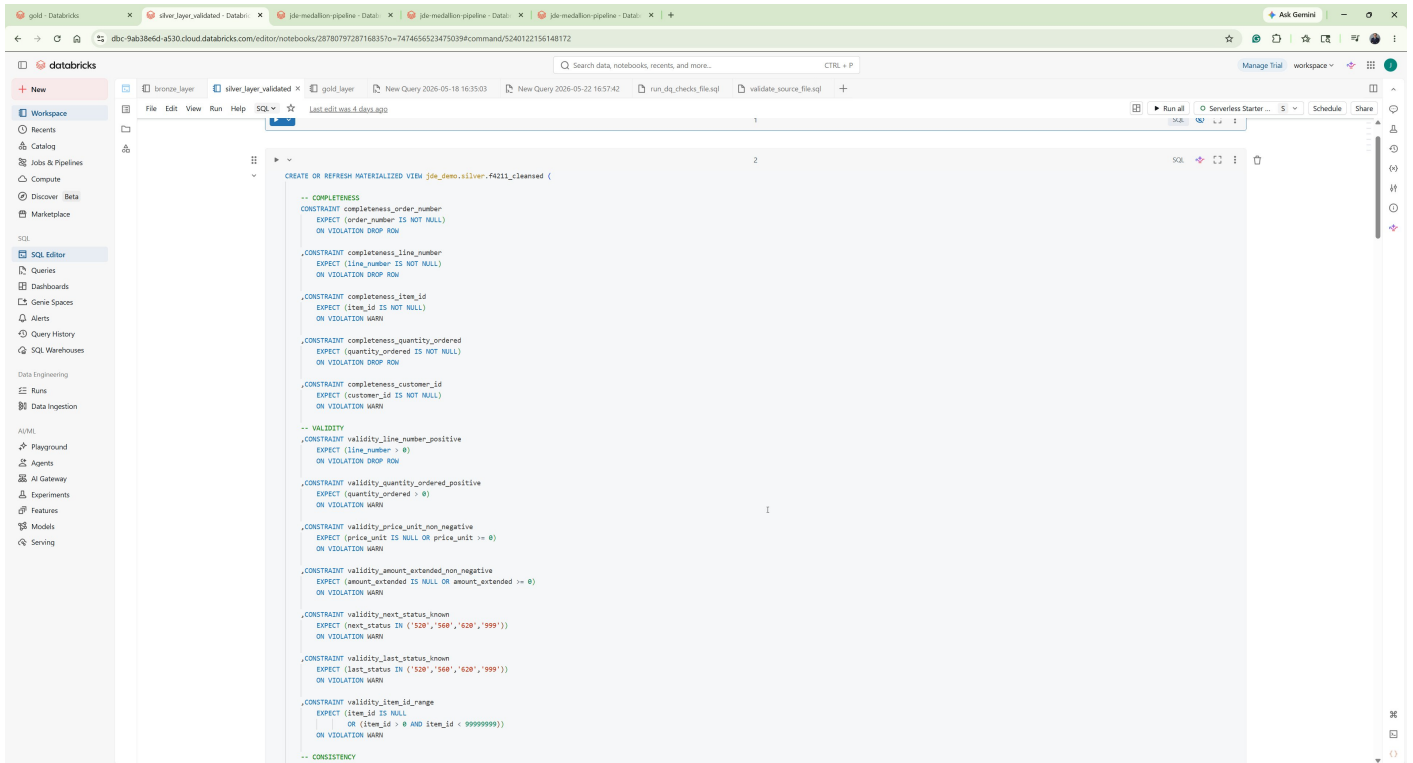
| | r3_SDDOCO | r3_SDDCTO | i2_SDKCOO | i2_SSDUND | r3_SDSANB | r3_SDSHAN | r3_SSDITM | r3_SSDITM | r3_SDDSC1 | r3_SDMNCU | r3_SDUOM | i2_SDSOOS | i2_SDSHPN | i2_SSDPRC | i2_SSDADP | i2_SSONTR | i2_SSDLTR | r3_SSDUMU | r3_prescd_data | |
|----|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-------------|--------------|----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|----------------|--------|
| 1 | 500001 | SO | | 1 | 1 | 10003 | 10003 | 100002 | PAR8-100002 | ITEM DESC:1 | LAX | EX | 2 | 2 | 12.6 | 25.2 | 999 | 620 | 126104 | NAILED |
| 2 | 500002 | SO | | 1 | 1 | 10006 | 10006 | 100003 | PAR8-100003 | ITEM DESC:2 | ATL | KT | 0 | 3 | 14.7 | 44.1 | 560 | 520 | 126103 | NAILED |
| 3 | 500003 | SO | | 1 | 1 | 10009 | 10009 | 100004 | PAR8-100004 | ITEM DESC:3 | DFW | PR | 4 | 4 | 16.8 | 67.2 | 620 | 560 | 126102 | NAILED |
| 4 | 500004 | SO | | 1 | 1 | 10012 | 10012 | 100005 | PAR8-100005 | ITEM DESC:4 | LAX | PK | 5 | 5 | 18.9 | 94.5 | 999 | 620 | 126101 | NAILED |
| 5 | 500005 | SO | | 1 | 1 | 10015 | 10015 | 100006 | PAR8-100006 | ITEM DESC:5 | ATL | EA | 0 | 6 | 21 | 126 | 560 | 520 | 126100 | NAILED |
| 6 | 500006 | SO | | 1 | 1 | 10018 | 10018 | 100007 | PAR8-100007 | ITEM DESC:6 | DFW | BK | 7 | 7 | 23.1 | 161.7 | 620 | 560 | 126099 | NAILED |
| 7 | 500007 | SO | | 1 | 1 | 10021 | 10021 | 100008 | PAR8-100008 | ITEM DESC:7 | LAX | KT | 8 | 8 | 25.2 | 201.6 | 999 | 620 | 126088 | NAILED |
| 8 | 500008 | SO | | 1 | 1 | 10024 | 10024 | 100009 | PAR8-100009 | ITEM DESC:8 | ATL | PR | 0 | 9 | 27.3 | 245.7 | 560 | 520 | 126097 | NAILED |
| 9 | 500009 | SO | | 1 | 1 | 10027 | 10027 | 100010 | PAR8-100010 | ITEM DESC:9 | DFW | PK | 10 | 10 | 29.4 | 294 | 620 | 560 | 126096 | NAILED |
| 10 | 500010 | SO | | 1 | 1 | 10030 | 10030 | 100011 | PAR8-100011 | ITEM DESC:10 | LAX | EA | 11 | 11 | 31.5 | 346.5 | 999 | 620 | 126095 | NAILED |

Raw JDE data in Bronze: original column names (SDDOCO, SDDCTO, SDKCOO, SDSHAN, SDITM, SDDSC1) preserved exactly as exported from JD Edwards. 10 rows, 0.40s runtime.

3. Silver Layer: Cleansed + Validated

The Silver layer transforms raw JDE column names into business-friendly names, applies DLT Expectations for data quality enforcement, and adds column-level sensitivity tags for governance. Critical fields use ON VIOLATION DROP ROW while non-critical fields use ON VIOLATION WARN.

DLT Expectations: F4211 Cleansed



Silver layer DLT Expectations for F4211: COMPLETENESS constraints (order_number, line_number NOT NULL with DROP ROW; item_id, customer_id with WARN), VALIDITY constraints (positive values, known status codes IN ('520','560','620','999'), item_id range checks.

Silver Cleansed Data: F4211

gold - databricks | New Query 2026-05-22 16:57 | jde-medallion-pipeline - Data | jde-medallion-pipeline - Data | jde-medallion-pipeline - Data | +

dbcf-9ab38bed-af30.cloud.databricks.com/editor/queries/395767207155305370~747465652475039

databricks | Search data, notebooks, recents, and more... | Manage Trial | workspace | Save*

Run selected (1000) | Last row (1/10) | workspace | default | New SQL editor: CN | Last edit was now

```

1 SELECT * FROM jde_demo_bronze.f4211_raw LIMIT 10;
2
3 | SELECT * FROM jde_demo_silver.f4211_cleansed LIMIT 10;
4
5

```

Add parameter | I

| order_number | order_type | order_company | line_number | customer_id | ship_to_id | item_id | item_number | item_description | business_unit | unit_of_measure | quantity_ordered | shipment_number | price_unit | amount_extended | |
|--------------|------------|---------------|-------------|-------------|------------|---------|-------------|------------------|---------------|-----------------|------------------|-----------------|------------|-----------------|-------|
| 1 | 500001 | SO | 1.0 | 10003 | 10003 | 100002 | PARF-100002 | ITEM DESC 1 | LAX | BX | 2.0000 | 2 | 12.0000 | 25.20 | 999.0 |
| 2 | 500002 | SO | 1.0 | 10006 | 10006 | 100003 | PARF-100003 | ITEM DESC 2 | ATL | KT | 0.0000 | 3 | 14.7000 | 44.10 | 560.0 |
| 3 | 500003 | SO | 1.0 | 10009 | 10009 | 100004 | PARF-100004 | ITEM DESC 3 | DFW | PR | 4.0000 | 4 | 16.8000 | 67.20 | 620.0 |
| 4 | 500004 | SO | 1.0 | 10012 | 10012 | 100005 | PARF-100005 | ITEM DESC 4 | LAX | PK | 5.0000 | 5 | 18.9000 | 94.50 | 999.0 |
| 5 | 500005 | SO | 1.0 | 10015 | 10015 | 100006 | PARF-100006 | ITEM DESC 5 | ATL | EA | 0.0000 | 6 | 21.0000 | 126.00 | 560.0 |
| 6 | 500006 | SO | 1.0 | 10018 | 10018 | 100007 | PARF-100007 | ITEM DESC 6 | DFW | BX | 7.0000 | 7 | 23.1000 | 161.70 | 620.0 |
| 7 | 500007 | SO | 1.0 | 10021 | 10021 | 100008 | PARF-100008 | ITEM DESC 7 | LAX | KT | 8.0000 | 8 | 25.2000 | 201.60 | 999.0 |
| 8 | 500008 | SO | 1.0 | 10024 | 10024 | 100009 | PARF-100009 | ITEM DESC 8 | ATL | PR | 9.0000 | 9 | 27.3000 | 243.70 | 560.0 |
| 9 | 500009 | SO | 1.0 | 10027 | 10027 | 100010 | PARF-100010 | ITEM DESC 9 | DFW | PK | 10.0000 | 10 | 29.4000 | 294.00 | 620.0 |
| 10 | 500010 | SO | 1.0 | 10030 | 10030 | 100011 | PARF-100011 | ITEM DESC 10 | LAX | EA | 11.0000 | 11 | 31.5000 | 346.50 | 999.0 |

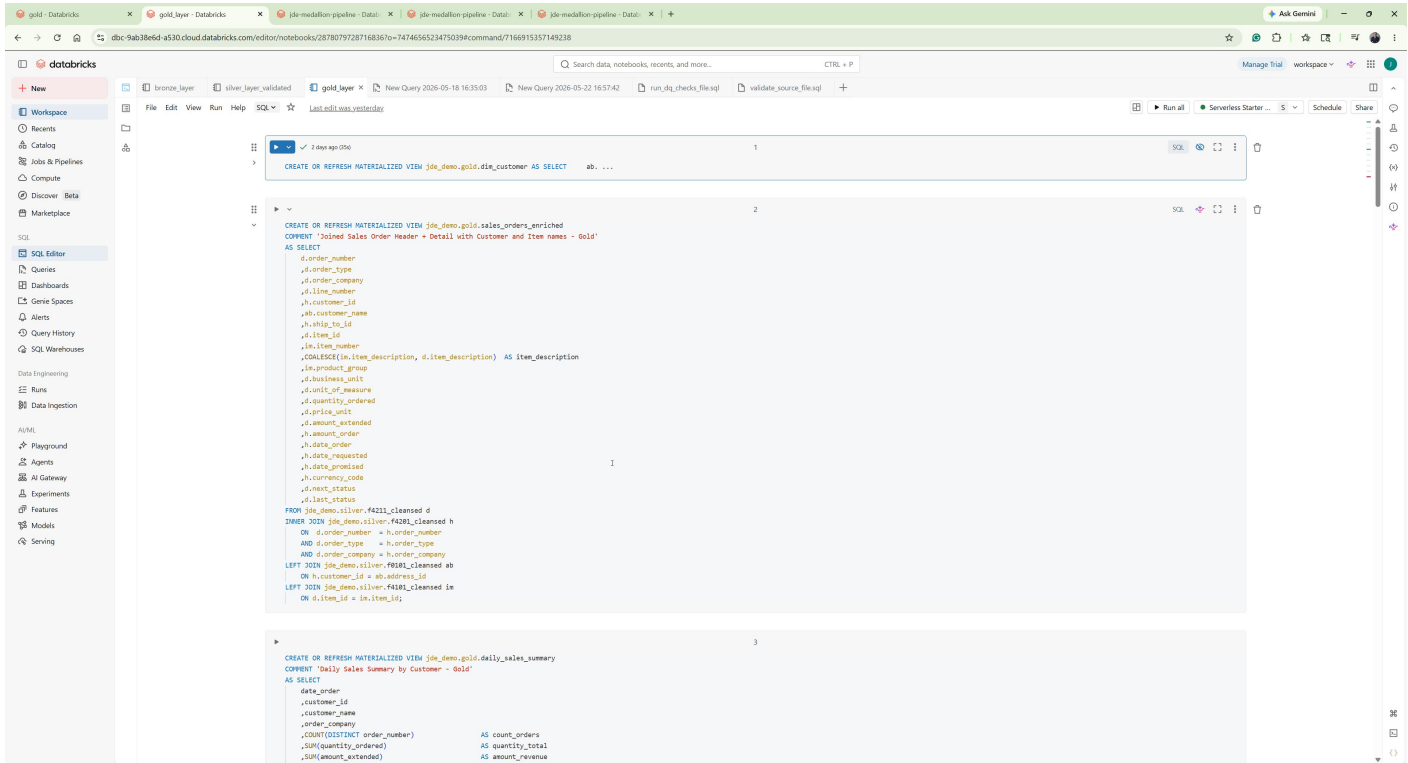
10 rows | 10.81s runtime | See performance (1) | Optimize | Refreshed now

Silver cleansed data: JDE codes transformed to business names (SDDOCO to order_number, SDSHAN to customer_id, SDITM to item_id, SDDSC1 to item_description, SDMCU to business_unit). 10 rows, 10.81s runtime.

4. Gold Layer: Business Models

The Gold layer builds business-ready analytical models: denormalized sales orders, daily sales summaries, AR aging analysis, customer dimensions (including SCD Type 2), and data quality monitoring views.

Gold Layer Notebook: Materialized Views



Gold layer notebook: sales_orders_enriched (INNER JOIN F4211 to F4201 on order_number/type/company, LEFT JOIN F0101 for customer names, LEFT JOIN F4101 for item details, COALESCE for item_description). daily_sales_summary with COUNT(DISTINCT), SUM aggregations by customer and date.

AR Aging Analysis (Materialized View SQL)

```

CREATE OR REPLACE MATERIALIZED VIEW jdb_demo.gold.ar_aging_analysis AS
SELECT
  ar.document_number,
  ar.customer_id,
  ab.customer_name,
  ar.date_invoiced,
  ar.date_due,
  ar.date_g,
  ar.amount_gross,
  ar.amount_open,
  ar.payment_status,
  DATEDIFF(current_date(), ar.date_due) AS days_past_due,
  CASE
    WHEN ar.date_due >= current_date() THEN 'Current'
    WHEN DATEDIFF(current_date(), ar.date_due) BETWEEN 1 AND 30 THEN '1-30 Days'
    WHEN DATEDIFF(current_date(), ar.date_due) BETWEEN 31 AND 60 THEN '31-60 Days'
    WHEN DATEDIFF(current_date(), ar.date_due) BETWEEN 61 AND 90 THEN '61-90 Days'
    ELSE '90+ Days'
  END AS aging_bucket,
  CASE
    WHEN ar.date_due >= current_date() THEN 1
    WHEN DATEDIFF(current_date(), ar.date_due) BETWEEN 1 AND 30 THEN 2
    WHEN DATEDIFF(current_date(), ar.date_due) BETWEEN 31 AND 60 THEN 3
    WHEN DATEDIFF(current_date(), ar.date_due) BETWEEN 61 AND 90 THEN 4
    ELSE 5
  END AS aging_bucket_sort
FROM jdb_demo.silver.f03b11_cleansed ar
LEFT JOIN jdb_demo.silver.f0101_cleansed ab
  ON ar.customer_id = ab.address_id
WHERE ar.amount_open > 0

```

Gold layer: `ar_aging_analysis` materialized view with `DATEDIFF` aging buckets (Current, 1-30, 31-60, 61-90, 90+ Days), `aging_bucket_sort` for ordering, joining `silver.f03b11_cleansed` to `silver.f0101_cleansed`. `WHERE amount_open > 0`. Successfully executed in 36.42s.

Sales Orders Enriched: Query Results

| id | order_number | order_type | order_company | item_number | customer_id | customer_name | ship_to_id | item_id | item_number | item_description | product_group | business_unit | unit_of_measure | quantity_ordered | price_unit | amount |
|----|--------------|------------|---------------|-------------|-------------|---------------------|------------|---------|-------------|--------------------|---------------|---------------|-----------------|------------------|------------|--------|
| 1 | 500531 | SO | 1.0 | 10144 | 10144 | Pilatus Aircraft | 10144 | 100332 | PART-100332 | WASHER FLAT 533 | FASTEN | DFW | BX | 32.0000 | 75.6000 | |
| 2 | 500099 | SO | 1.0 | 10093 | 10093 | Bell Textron | 10093 | 100700 | PART-100700 | BOLT HEX 701 | FASTEN | DFW | PK | 50.0000 | 218.4000 | |
| 3 | 5000651 | SO | 1.0 | 10054 | 10054 | Textron | 10054 | 100652 | PART-100652 | WASHER FLAT 653 | FASTEN | DFW | BX | 2.0000 | 117.6000 | |
| 4 | 500016 | SO | 1.0 | 10099 | 10099 | Hexcel | 10099 | 100617 | PART-100617 | FILTER OIL 618 | SEAL | LAX | BX | 17.0000 | 44.5000 | |
| 5 | 500085 | SO | 1.0 | 10054 | 10054 | Textron | 10054 | 100986 | PART-100986 | GCKET ENGINE 907 | BEARIN | LAX | EA | 30.0000 | 189.9000 | |
| 6 | 500095 | SO | 1.0 | 10072 | 10072 | Amentum | 10072 | 100992 | PART-100992 | WASHER FLAT 913 | FASTEN | LAX | BX | 42.0000 | 201.6000 | |
| 7 | 500078 | SO | 1.0 | 10033 | 10033 | Airbus | 10033 | 100979 | PART-100979 | CONNECTOR ELEC 980 | ELECT | DFW | PR | 29.0000 | 174.3000 | |
| 8 | 500029 | SO | 1.0 | 10090 | 10090 | Gulfstream | 10090 | 100330 | PART-100330 | BOLT HEX 331 | BEARIN | ATL | PK | 0.0000 | 71.4000 | |
| 9 | 500362 | SO | 1.0 | 10084 | 10084 | Booz Allen Hamilton | 10084 | 100363 | PART-100363 | BRACKET ASSY 364 | ELECT | ATL | KT | 0.0000 | 140.7000 | |
| 10 | 500548 | SO | 1.0 | 10045 | 10045 | GE Aerospace | 10045 | 100549 | PART-100549 | CONNECTOR ELEC 550 | SEAL | ATL | PR | 0.0000 | 113.3000 | |

Gold: `sales_orders_enriched` query results showing denormalized data with customer names (Pilatus Aircraft, Bell Textron, Hexcel, Amentum, Airbus, Gulfstream, Booz Allen Hamilton, GE Aerospace), product groups (FASTEN, SEAL, BEARIN, ELECT), business units (DFW, LAX, ATL).

AR Aging Analysis: Query Results

The screenshot shows a Databricks SQL Editor window with a query and its results. The query is as follows:

```

1 SELECT * FROM jdc_demo.bronze.f4211_raw LIMIT 10;
2
3 SELECT * FROM jdc_demo.silver.f4211_cleaned LIMIT 10;
4
5 SELECT * FROM jdc_demo.gold.sales_orders_enriched LIMIT 10;
6
7 SELECT * FROM jdc_demo.gold.ar_aging_analysis LIMIT 10;
8
9 SELECT * FROM jdc_demo.gold.dim_customer_scd2 LIMIT 10;
10
11 SELECT * FROM jdc_demo.gold.dg_summary_dashboard LIMIT 10;
12
13 SELECT * FROM jdc_demo.gold.dg_row_count_reconciliation LIMIT 10;
14
15 SELECT * FROM jdc_demo.gold.dg_referential_integrity LIMIT 10;
16
17

```

The results table shows 5 rows of data:

| id | child_table | R_relationship | r ₁ count_total_rows | r ₂ count_orphans | percent_orphan | result |
|----|-------------|------------------------------------|---------------------------------|------------------------------|----------------|--------|
| 1 | F4201 | customer_id -> F1011.address_id | 1000 | 0 | 0.00 | PASS |
| 2 | F4211 | customer_id -> F1011.address_id | 1000 | 0 | 0.00 | PASS |
| 3 | F4211 | order_number -> F4201.order_number | 1000 | 0 | 0.00 | PASS |
| 4 | F4211 | item_id -> F4101.item_id | 1000 | 0 | 0.00 | PASS |
| 5 | F03B11 | customer_id -> F1011.address_id | 1000 | 0 | 0.00 | PASS |

Gold: *ar_aging_analysis* results for Boeing (customer_id 10021) showing invoice dates, due dates, GL dates, gross/open amounts, payment_status (H=Hold, O=Open), days_past_due calculations, and aging_bucket assignments (Current, 1-30 Days).

Customer Dimension (SCD Type 2)

The screenshot shows a Databricks workspace with a query result and a visualization. The query is:

```

Which products have the most orders? Reviewable

```

The results show that all products have equal order frequency. The top 15 products by revenue are:

| Product | Revenue |
|--------------------|---------|
| BOLT HEX 801 | \$10.0K |
| BOLT HEX 101 | \$10.0K |
| BOLT HEX 901 | \$10.0K |
| BOLT HEX 2 | \$10.0K |
| BOLT HEX 701 | \$10.0K |
| BOLT HEX 001 | \$10.0K |
| BOLT HEX 201 | \$10.0K |
| BOLT HEX 401 | \$10.0K |
| BOLT HEX 501 | \$10.0K |
| BOLT HEX 601 | \$10.0K |
| BOLT HEX 201 | \$10.0K |
| CONNECTOR ELEC 900 | \$10.0K |
| CONNECTOR ELEC 700 | \$10.0K |
| CONNECTOR ELEC 1 | \$10.0K |
| CONNECTOR ELEC 400 | \$10.0K |
| CONNECTOR ELEC 200 | \$10.0K |

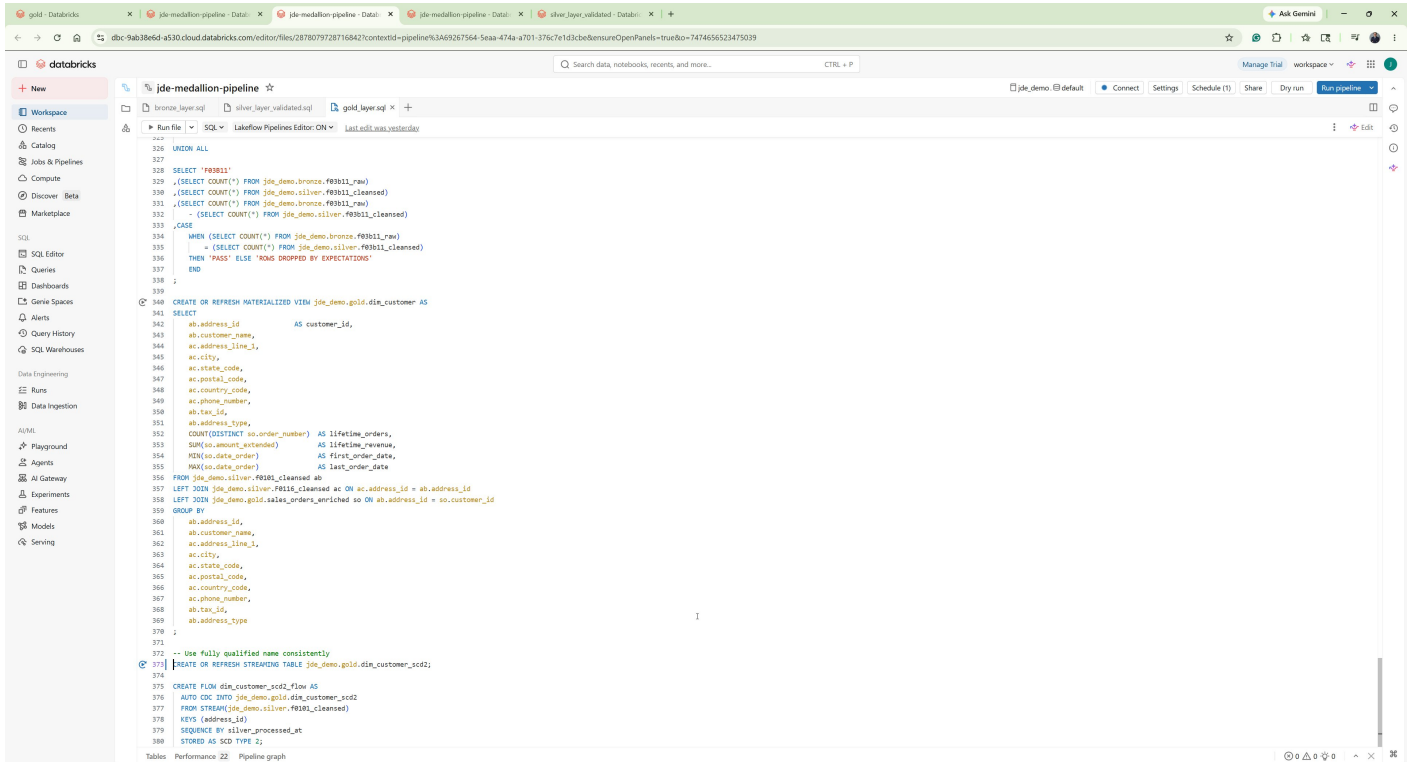
The top products are primarily BOLT HEX items from the FASTEN group and CONNECTOR ELEC items from the ELECT group, each generating approximately \$10,900 in revenue.

Product Group Performance:

| Product Group | Total Revenue |
|---------------|---------------|
| FASTEN | \$873.0K |
| BEARIN | \$646.3K |
| ELECT | \$831.1K |
| SEAL | \$555.0K |

Gold: *dim_customer_scd2* with surrogate *address_key*, *address_type* (V=Vendor, E=Employee, C=Customer), customer names (Ducommun, Lockheed Martin, Kaman Aerospace, SpaceX, Curtiss-Wright, RTX), *_START_AT/_END_AT* timestamps for SCD Type 2 history tracking.

SCD Type 2 Pipeline Code



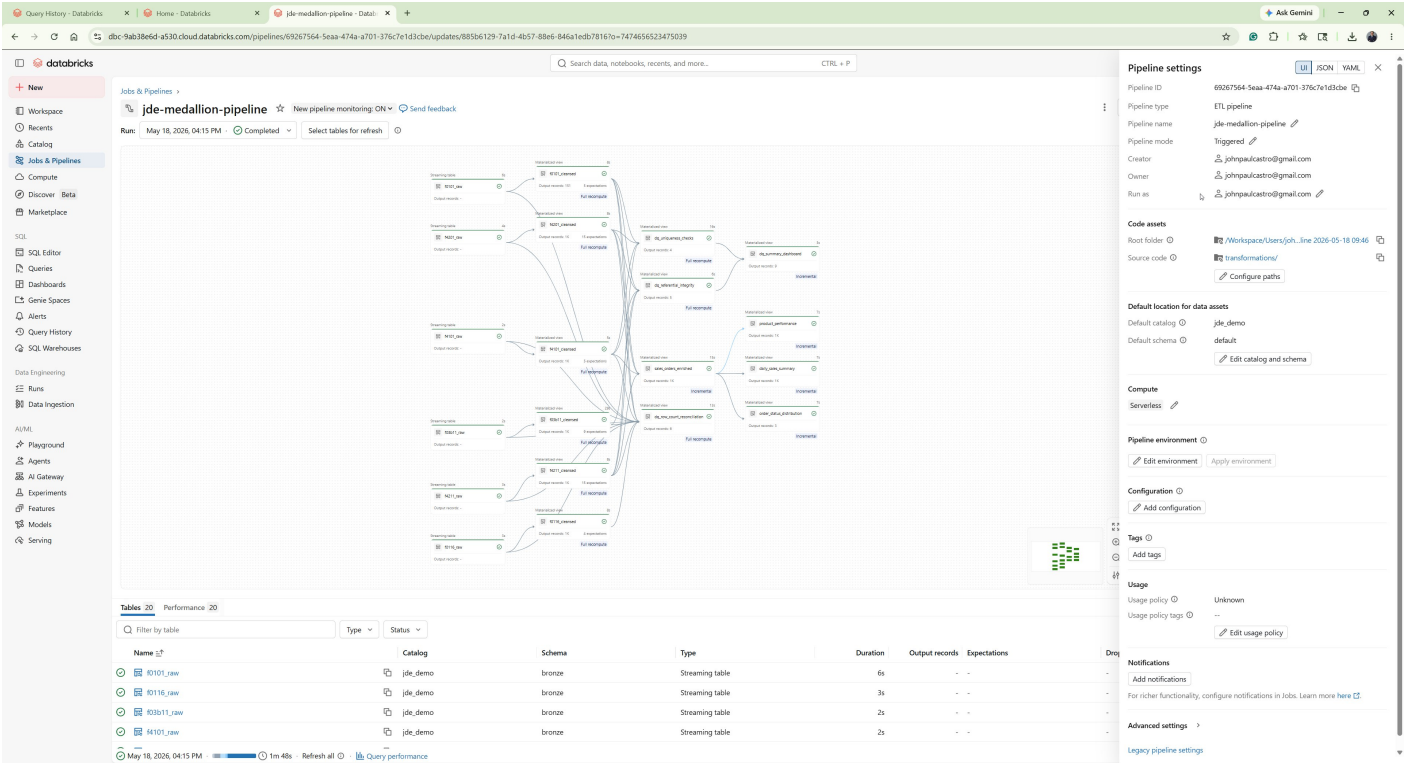
```
326 UNDO ALL
327
328 SELECT 'F03011'
329 ,(SELECT COUNT(*) FROM jde_demo.bronze.f03011_raw)
330 ,(SELECT COUNT(*) FROM jde_demo.silver.f03011_cleansed)
331 ,(SELECT COUNT(*) FROM jde_demo.bronze.f03011_raw)
332 ,(SELECT COUNT(*) FROM jde_demo.silver.f03011_cleansed)
333 ;
334
335 CASE
336 WHEN (SELECT COUNT(*) FROM jde_demo.bronze.f03011_raw)
337     = (SELECT COUNT(*) FROM jde_demo.silver.f03011_cleansed)
338 THEN PAUSE ELSE ROWS DROPPED BY EXPECTATIONS
339 END
340 ;
341
342 CREATE OR REFRESH MATERIALIZED VIEW jde_demo.gold.dim_customer AS
343 SELECT
344     ab.address_id           AS customer_id,
345     ab.customer_name,
346     ac.address_line_1,
347     ac.city,
348     ac.state_code,
349     ac.postal_code,
350     ac.country_code,
351     ac.phone_number,
352     ab.tax_id,
353     ab.address_type,
354     COUNT(DISTINCT so.order_number) AS lifetime_orders,
355     SUM(so.amount_extended) AS lifetime_revenue,
356     MIN(so.data_order) AS first_order_date,
357     MAX(so.data_order) AS last_order_date
358 FROM jde_demo.silver.f0101_cleansed ab
359 LEFT JOIN jde_demo.silver.f0101_cleansed ac ON ac.address_id = ab.address_id
360 LEFT JOIN jde_demo.gold.sales_orders_enriched so ON ab.address_id = so.customer_id
361 GROUP BY
362     ab.address_id,
363     ab.customer_name,
364     ac.address_line_1,
365     ac.city,
366     ac.state_code,
367     ac.postal_code,
368     ac.country_code,
369     ac.phone_number,
370     ab.tax_id,
371     ab.address_type
372 ;
373
374 -- Use fully qualified name consistently
375 CREATE OR REFRESH STREAMING TABLE jde_demo.gold.dim_customer_scd2;
376
377 CREATE FLOW dim_customer_scd2_flow AS
378 AUTO CDC INTO jde_demo.gold.dim_customer_scd2
379 FROM STREAM(jde_demo.silver.f0101_cleansed)
380 KEYS (address_id)
381 SEQUENCE BY silver_processed_at
382 STORED AS SCD TYPE 2;
```

Pipeline editor: *dim_customer* materialized view (*lifetime_orders*, *lifetime_revenue*, *first/last_order_date*), *dim_customer_scd2* streaming table with *CREATE FLOW* using *AUTO CDC INTO dim_customer_scd2*, *FROM STREAM(silver.f0101_cleansed)*, *KEYS(address_id)*, *SEQUENCE BY silver_processed_at*, *STORED AS SCD TYPE 2*.

5. Lakeflow Declarative Pipeline (DLT)

The jde-medallion-pipeline is an ETL pipeline running on Serverless compute in Triggered mode. It processes 22 tables across Bronze, Silver, and Gold layers in a single declarative pipeline with full lineage tracking. Pipeline code is organized in three SQL files: bronze_layer.sql, silver_layer_validated.sql, and gold_layer.sql.

Pipeline DAG: May 18 Run (Settings View)



Pipeline settings

- Pipeline ID: 60267564-5eaa-474a-a701-376c7e1d3db6
- Pipeline type: ETL pipeline
- Pipeline name: jde-medallion-pipeline
- Pipeline mode: Triggered
- Creator: jshnpaujcastro@gmail.com
- Owner: jshnpaujcastro@gmail.com
- Run as: jshnpaujcastro@gmail.com

Code assets

- Root folder: /Workspace/Users/jsh.../line-2026-05-18-0946
- Source code: transformations/

Default location for data assets

- Default catalog: jde_demo
- Default schema: default

Compute

- Serverless

Pipeline environment

- Edit environment | Apply environment

Configuration

- Add configuration

Tags

- Add tags

Usage

- Usage policy: Unknown
- Edit usage policy

Notifications

- Add notifications

Advanced settings

- Legacy pipeline settings

| Name | Catalog | Schema | Type | Duration | Output records | Expectations | Drop |
|------------|----------|--------|-----------------|----------|----------------|--------------|------|
| f0101_raw | jde_demo | bronze | Streaming table | 6s | - | - | - |
| f0116_raw | jde_demo | bronze | Streaming table | 3s | - | - | - |
| f03b11_raw | jde_demo | bronze | Streaming table | 2s | - | - | - |
| f4101_raw | jde_demo | bronze | Streaming table | 2s | - | - | - |

jde-medallion-pipeline: Completed run (May 18, 1m 48s), 20 tables. Pipeline Settings showing ETL type, Triggered mode, Serverless compute, jde_demo catalog. Table list: f0101_raw (6s), f0116_raw (3s), f03b11_raw (2s), f4101_raw (2s) as streaming tables.

Pipeline DAG: May 21 Run (Details View)

Pipeline details

- Pipeline ID: 69267564-5aaa-474a-a701-376c7e1d3cb4
- Pipeline type: ETL pipeline
- Creator: johnpaulcastro@gmail.com
- Owner: johnpaulcastro@gmail.com
- Run as: johnpaulcastro@gmail.com
- Source code: 1 folder, Open in Editor

Run details

- Pipeline run ID: 9672445-fbc5-40d6-bd4-623642364450
- Run status: Completed
- Run type: Refresh all
- Start time: May 21, 2026, 01:14 PM
- End time: May 21, 2026, 01:16 PM
- Total duration: 1m 12s

Compute

- Serverless
- View details, Logs

Usage

- Usage policy: Unknown
- Usage policy tags: --

| Name | Catalog | Schema | Type | Duration | Output records | Expectations | Dropped | Warnings | Failed |
|------------------------|----------|--------|-----------------|----------|----------------|--------------|---------|----------|--------|
| 0101_raw | jde_demo | bronze | Streaming table | 2s | - | - | - | - | - |
| ... (20 more rows) ... | | | | | | | | | |

jde-medallion-pipeline: Completed Refresh All run (May 21, 1m 12s), 22 tables. Run details showing Pipeline run ID, Serverless compute, source code in transformations/ folder, full DAG visualization with all Bronze/Silver/Gold nodes.

6. Workflow Orchestration

The `jde-daily-pipeline` workflow orchestrates the full data pipeline as a 3-task DAG: `validate_source_file` (SQL) validates Bronze data exists, `run_pipeline` (Pipeline) triggers the DLT pipeline, and `run_dq_checks` (SQL) runs data quality checks. Each task depends on the previous one succeeding. Performance optimization is enabled.

Workflow DAG: Overview

The screenshot displays the Databricks Jobs & Pipelines interface. The main area shows a Directed Acyclic Graph (DAG) with three tasks: `validate_source_file`, `run_pipeline`, and `run_dq_checks`. The `run_pipeline` task is selected, and its details are shown on the right. The job ID is 241118969980051, and the cluster is running on a Serverless Starter Warehouse. Performance optimization is enabled.

jde-daily-pipeline: 3-task DAG (`validate_source_file` > `run_pipeline` > `run_dq_checks`). Cluster running, 7 upstream tables, performance optimized ON. Job ID visible, Serverless Starter Warehouse.

Task 1: `validate_source_file`

The screenshot shows the configuration for the 'validate_source_file' task in a Databricks pipeline. The task is configured with the following details:

- Task name:** validate_source_file
- Type:** SQL
- SQL task:** File
- Source:** Workspace
- Path:** /Workspace/Users/johnpaulcastro@gmail.com/validate_source_file.sql
- SQL warehouse:** Serverless Starter Warehouse (S)
- Dependencies:** Select task dependencies...

The right-hand panel shows job details for the pipeline, including the Job ID (24113896980051), Creator (johnpaulcastro@gmail.com), and various configuration options like Schedules & Triggers, Job parameters, Compute, Tags, Job notifications, and Permissions.

Task 1 detail: SQL type, File source from /Workspace/Users/johnpaulcastro@gmail.com/validate_source_file.sql, Serverless Starter Warehouse (S), no upstream dependencies.

Task 2: run_pipeline

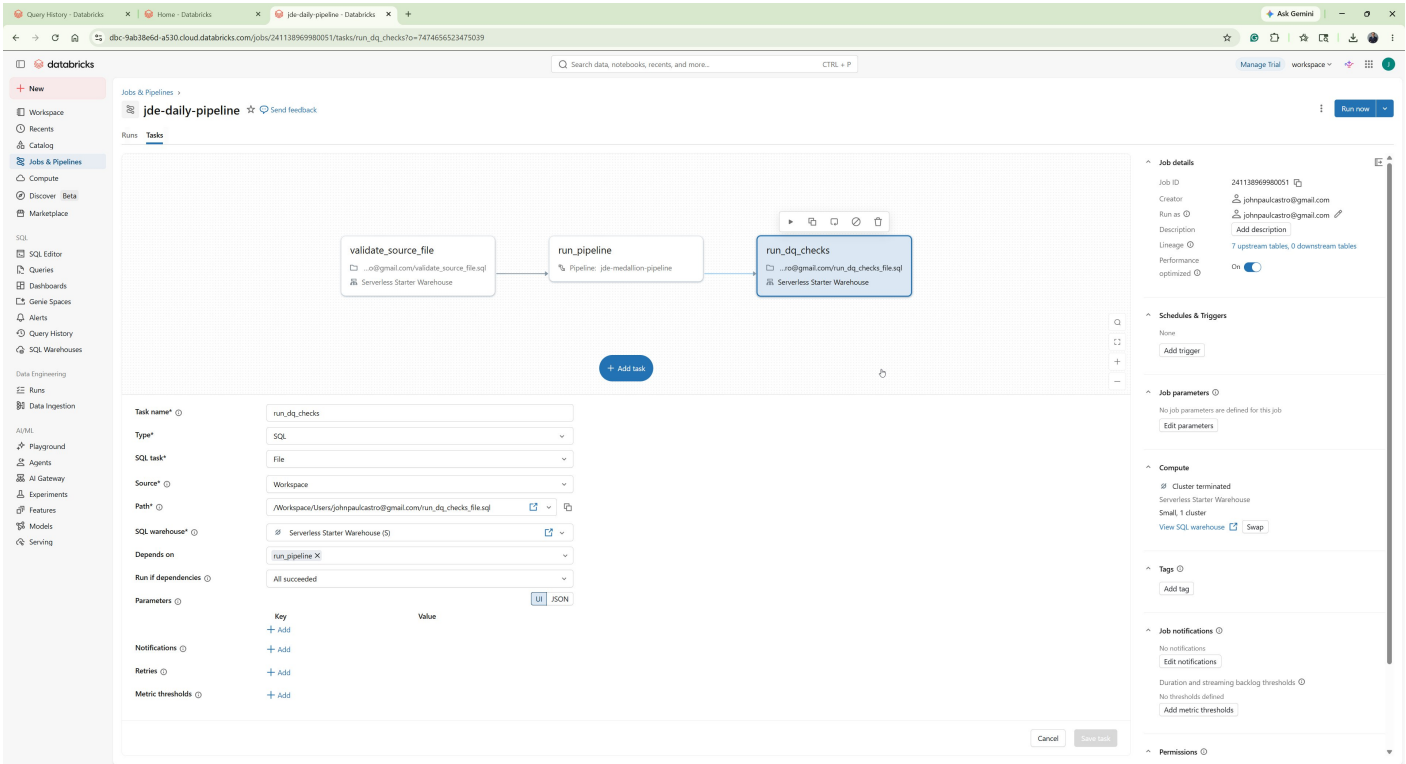
The screenshot shows the configuration for the 'run_pipeline' task in a Databricks pipeline. The task is configured with the following details:

- Task name:** run_pipeline
- Type:** Pipeline
- Pipeline:** jde-medallion-pipeline
- Dependencies:** validate_source_file X
- Run if dependencies:** All succeeded

The right-hand panel shows job details for the pipeline, including the Job ID (24113896980051), Creator (johnpaulcastro@gmail.com), and various configuration options like Schedules & Triggers, Job parameters, Compute, Tags, Job notifications, and Permissions.

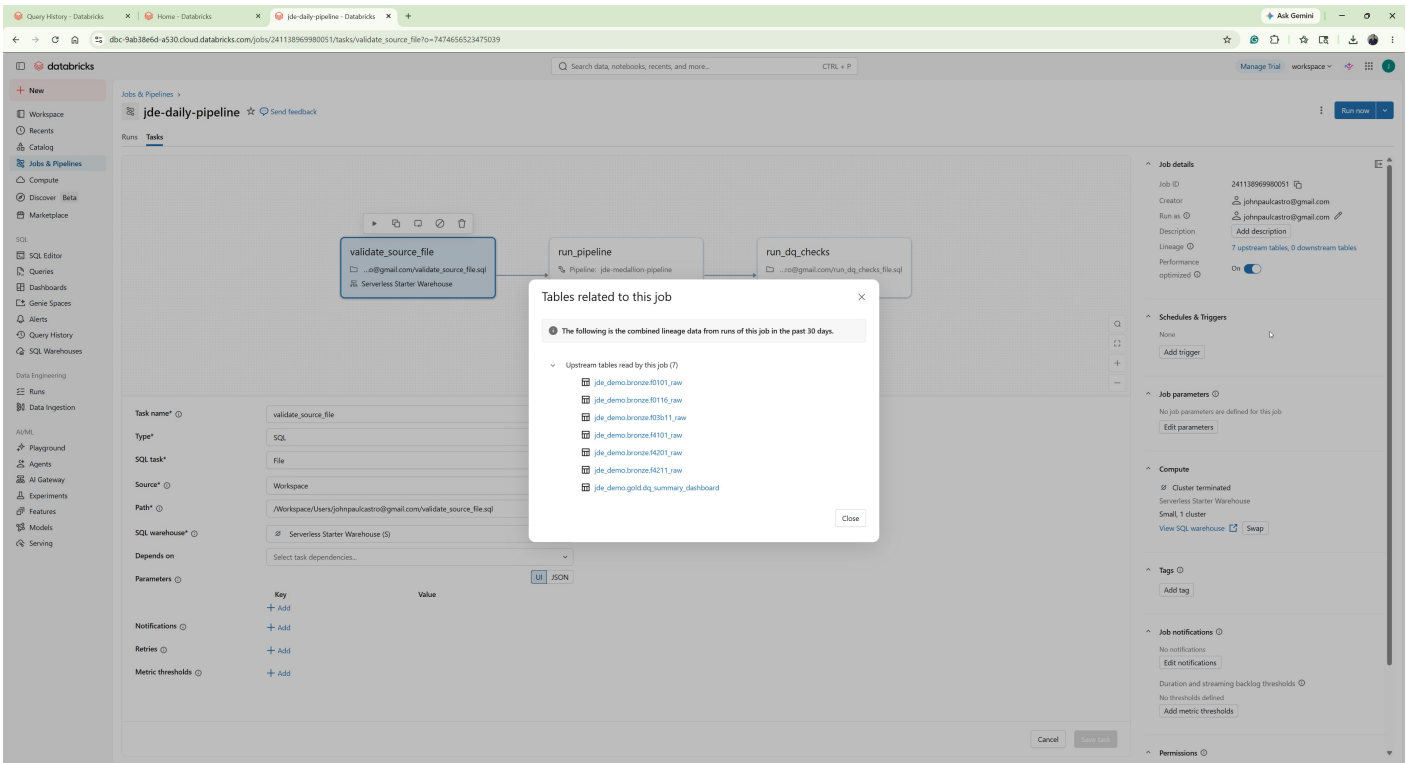
Task 2 detail: Pipeline type, triggers jde-medallion-pipeline, depends on validate_source_file (All succeeded). Option to trigger full refresh on pipeline.

Task 3: run_dq_checks



Task 3 detail: SQL type, File source from run_dq_checks_file.sql, depends on run_pipeline (All succeeded). Final step validates data quality after pipeline completion.

Workflow Lineage: Upstream Tables

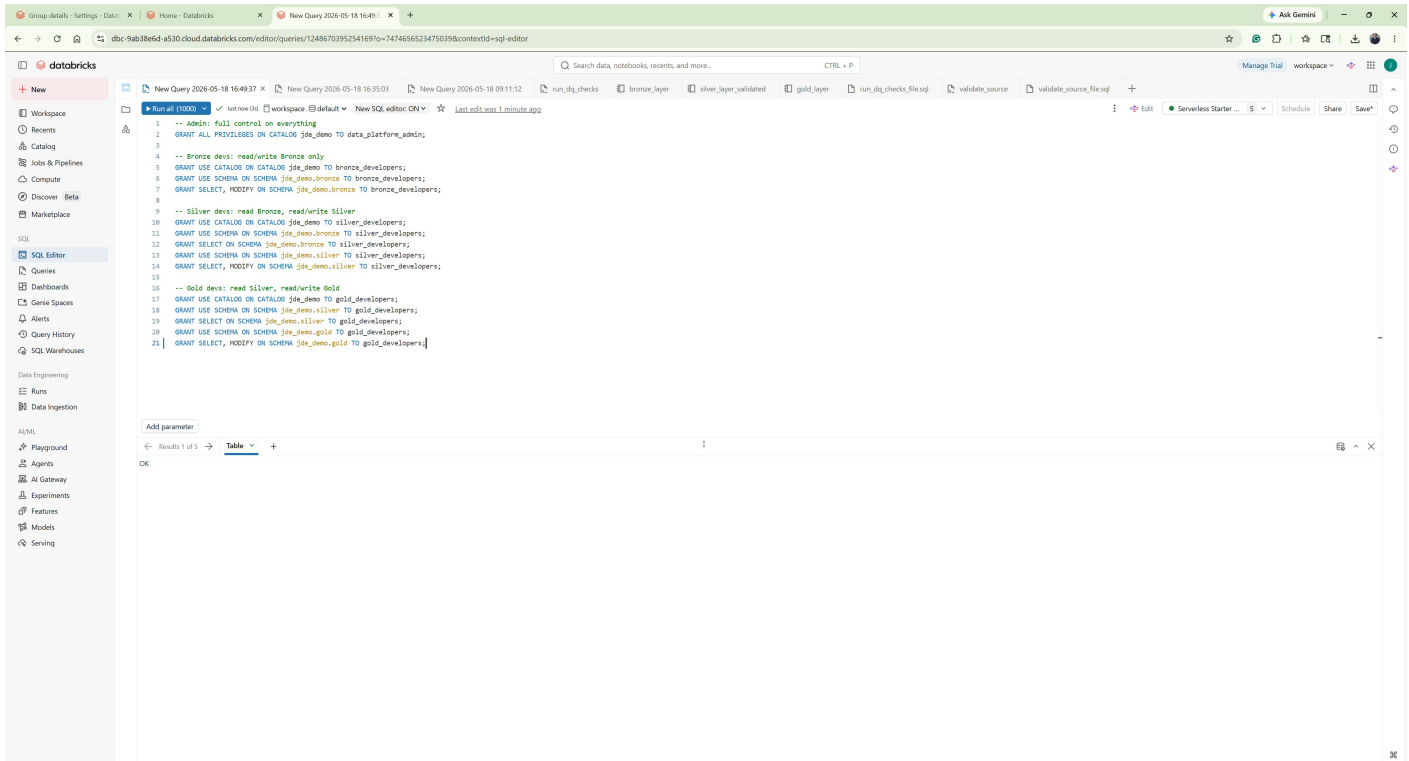


Tables related to this job: 7 upstream tables from combined lineage over 30 days. jde_demo.bronze (f0101_raw, f0116_raw, f03b11_raw, f4101_raw, f4201_raw, f4211_raw) plus jde_demo.gold.dq_summary_dashboard.

7. Unity Catalog RBAC

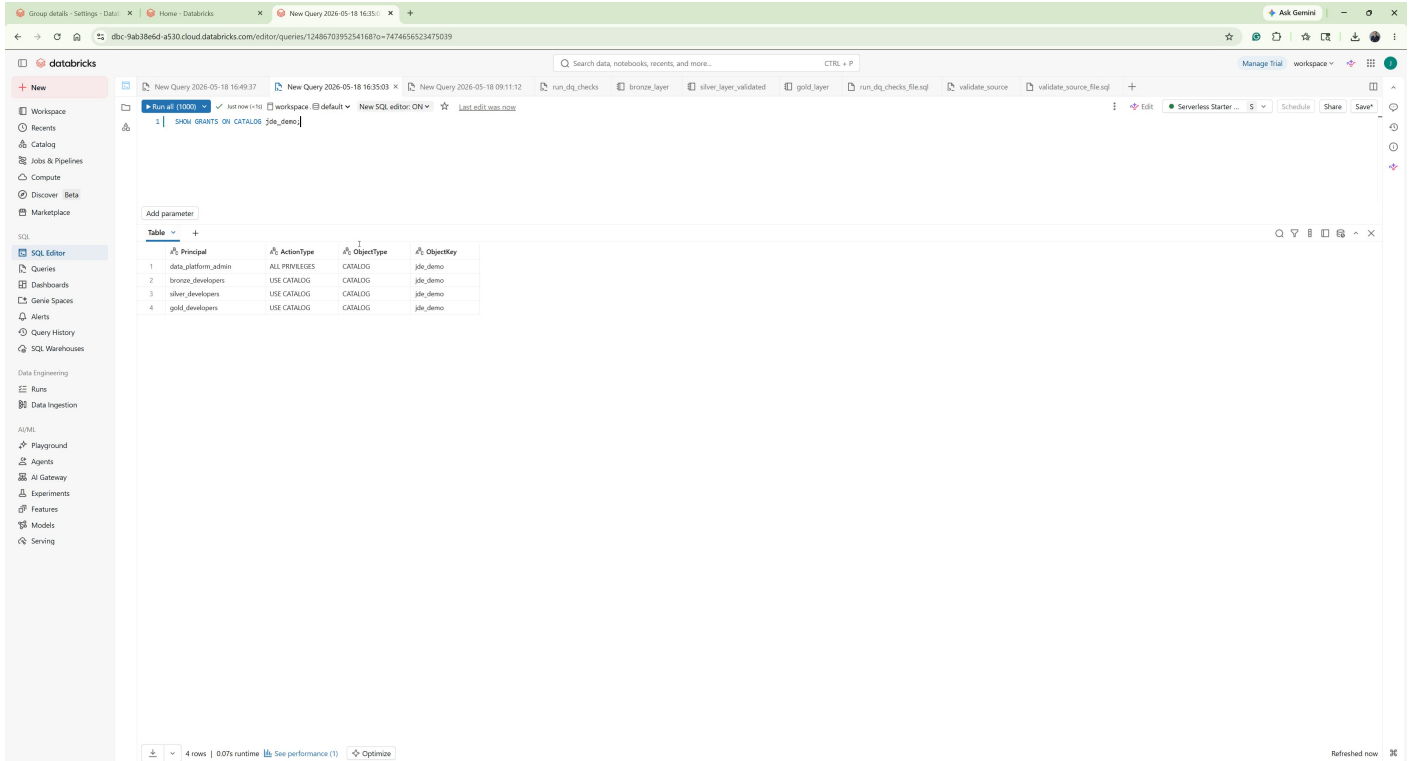
Role-based access control is implemented through four Unity Catalog groups with schema-level permissions following a progressive access model: Bronze developers can only modify Bronze, Silver developers can read Bronze and modify Silver, Gold developers can read Silver and modify Gold, and data_platform_admin has full control.

GRANT Statements: Full RBAC Script



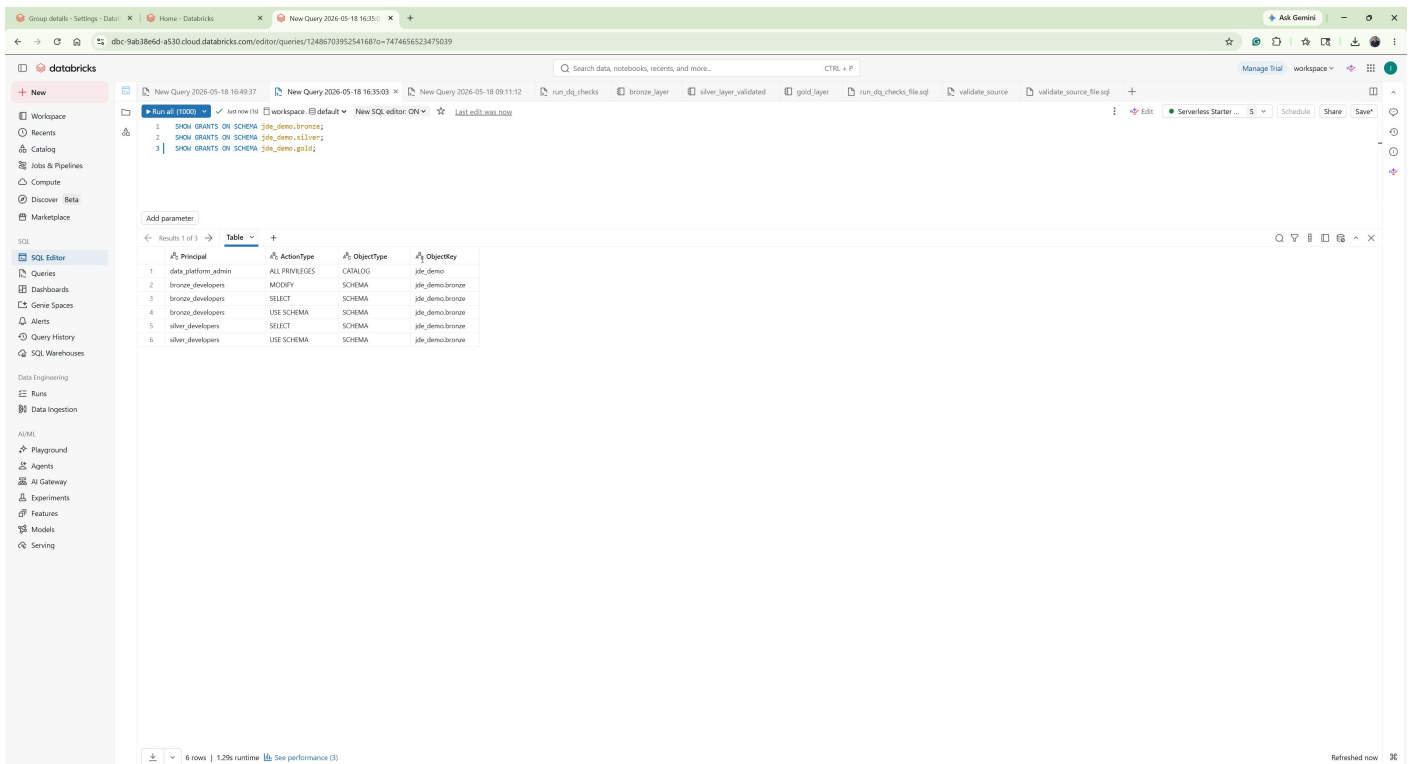
RBAC GRANT script: data_platform_admin gets ALL PRIVILEGES ON CATALOG. Bronze devs get read/write Bronze only. Silver devs get read Bronze + read/write Silver. Gold devs get read Silver + read/write Gold. Result: OK.

SHOW GRANTS ON CATALOG: jde_demo



Catalog-level grants: `data_platform_admin` has `ALL PRIVILEGES`, `bronze_developers`, `silver_developers`, and `gold_developers` each have `USE CATALOG` on `jde_demo`.

SHOW GRANTS ON SCHEMA: Bronze



Bronze schema grants: `bronze_developers` have `MODIFY/SELECT/USE SCHEMA` on `jde_demo.bronze`. `silver_developers` have `SELECT/USE SCHEMA` (read-only downstream access).

SHOW GRANTS ON SCHEMA: Silver

The screenshot shows the Databricks SQL Editor interface. The query results are displayed in a table with the following data:

| Principal | ActionType | ObjectType | ObjectKey |
|---------------------|----------------|------------|-----------------|
| data_platform_admin | ALL PRIVILEGES | CATALOG | jde_demo |
| silver_developers | MODIFY | SCHEMA | jde_demo.silver |
| silver_developers | SELECT | SCHEMA | jde_demo.silver |
| silver_developers | USE SCHEMA | SCHEMA | jde_demo.silver |
| gold_developers | SELECT | SCHEMA | jde_demo.silver |
| gold_developers | USE SCHEMA | SCHEMA | jde_demo.silver |

Silver schema grants: *silver_developers* have **MODIFY/SELECT/USE SCHEMA** on *jde_demo.silver*. *gold_developers* have **SELECT/USE SCHEMA** on *jde_demo.silver* (read-only downstream).

SHOW GRANTS ON SCHEMA: Gold

The screenshot shows the Databricks SQL Editor interface. The query results are displayed in a table with the following data:

| Principal | ActionType | ObjectType | ObjectKey |
|---------------------|----------------|------------|---------------|
| data_platform_admin | ALL PRIVILEGES | CATALOG | jde_demo |
| gold_developers | MODIFY | SCHEMA | jde_demo.gold |
| gold_developers | SELECT | SCHEMA | jde_demo.gold |
| gold_developers | USE SCHEMA | SCHEMA | jde_demo.gold |

Gold schema grants: *gold_developers* have **MODIFY/SELECT/USE SCHEMA** on *jde_demo.gold*. *data_platform_admin* retains **ALL PRIVILEGES** at the catalog level.

Group Details: data_platform_admin

The screenshot shows the Databricks interface for the 'data_platform_admin' group. The left sidebar contains navigation options like 'New', 'Workspace', 'Recents', 'Catalog', 'Jobs & Pipelines', 'Compute', 'Discover Beta', 'Marketplace', 'SQL', 'SQL Editor', 'Queries', 'Dashboards', 'Genie Spaces', 'Alerts', 'Query History', 'SQL Warehouses', 'Data Engineering', 'Runs', 'Data Ingestion', 'AI/ML', 'Playground', 'Agents', 'AI Gateway', 'Experiments', 'Features', 'Models', and 'Serving'. The main content area is titled 'Group details' and includes a search bar, a 'Delete' button, and a search input field containing 'johnpaulcastro@gmail.com'. Below this, a table lists group members with columns for 'Name' and 'Email/ID'. One member is listed: 'johnpaulcastro@gmail.com' with the same email/ID. A pagination control at the bottom right shows '< Previous', 'Next >', and '20 / page'.

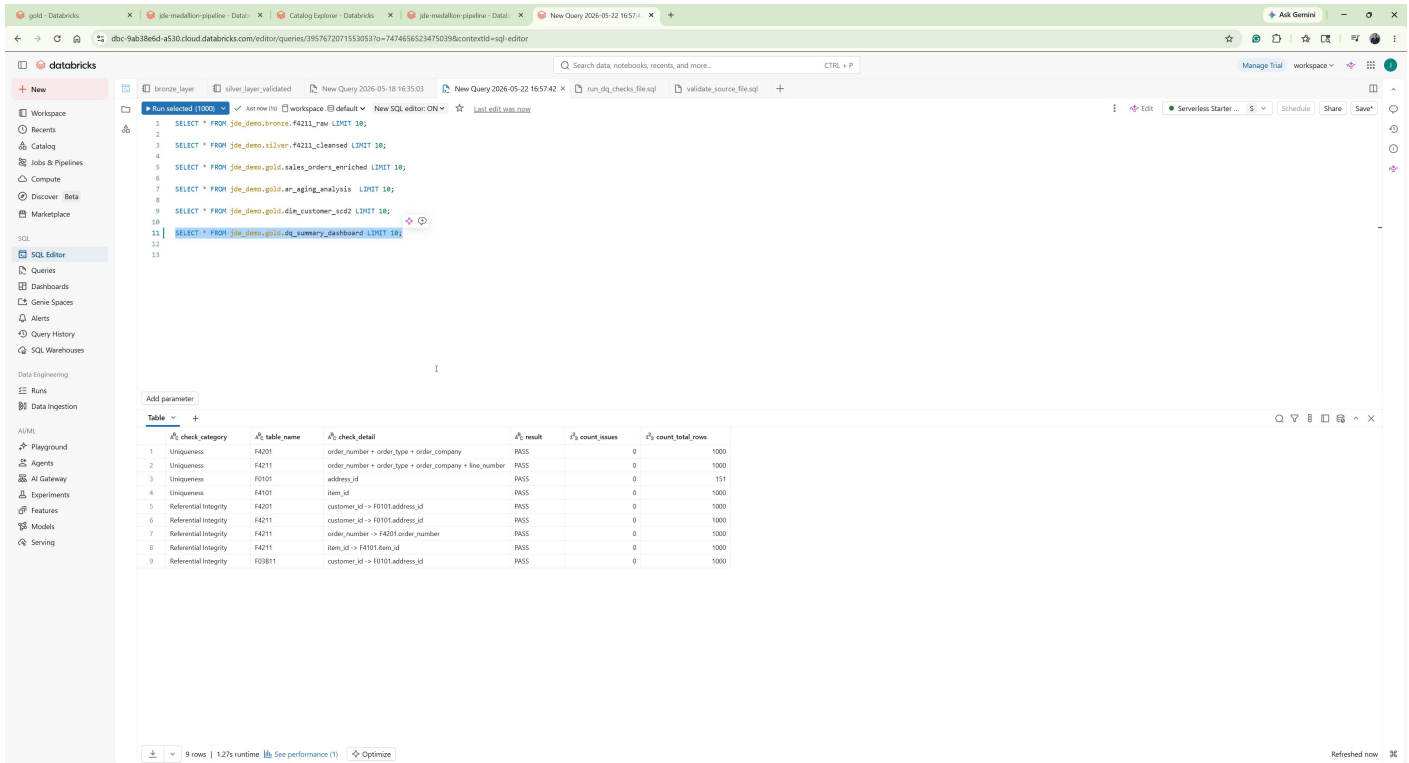
| Name | Email/ID |
|--------------------------|--------------------------|
| johnpaulcastro@gmail.com | johnpaulcastro@gmail.com |

Identity and Access settings: data_platform_admin group membership showing johnpaulcastro@gmail.com. Workspace admin > Identity and access > Groups path.

8. Data Quality Framework

The data quality framework runs as the final task in the orchestration workflow. It includes three check categories: Uniqueness (composite key validation), Referential Integrity (foreign key validation across tables), and Row Count Reconciliation (Bronze vs Silver count matching to detect dropped rows from DLT Expectations).

DQ Summary Dashboard



dq_summary_dashboard: 9 checks total (4 Uniqueness + 5 Referential Integrity), all PASS, 0 count_issues across all tables (F4201, F4211, F0101, F4101, F03B11). Check details show composite key patterns and FK relationships.

Referential Integrity Checks

```

1 SELECT * FROM $joe_demo.bronze.f4211_raw L1M1T 18;
2 SELECT * FROM $joe_demo.silver.f4211_cleaned L1M1T 18;
3 SELECT * FROM $joe_demo.gold.sales_orders_enriched L1M1T 18;
4 SELECT * FROM $joe_demo.gold.ar_aging_analysis L1M1T 18;
5
6
7

```

| | document_number | customer_id | customer_name | date_invoice | date_due | date_gl | amount_gross | amount_open | payment_status | days_past_due | aging_bucket | aging_bucket_sort |
|----|-----------------|-------------|---------------|--------------|------------|------------|--------------|-------------|----------------|---------------|--------------|-------------------|
| 1 | 900007 | 10021 | Boeing | 2025-04-08 | 2025-05-22 | 2025-04-08 | 107.00 | 107.00 | H | -1 | Current | 1 |
| 2 | 900196 | 10021 | Boeing | 2025-12-30 | 2026-05-31 | 2025-12-30 | 206.00 | 206.00 | O | -10 | Current | 1 |
| 3 | 900206 | 10021 | Boeing | 2025-09-21 | 2026-06-10 | 2025-09-21 | 306.00 | 306.00 | O | -20 | Current | 1 |
| 4 | 900306 | 10021 | Boeing | 2025-06-13 | 2026-05-21 | 2025-06-13 | 406.00 | 406.00 | O | 0 | Current | 1 |
| 5 | 900341 | 10021 | Boeing | 2025-05-09 | 2026-05-26 | 2025-05-09 | 441.00 | 441.00 | O | -5 | Current | 1 |
| 6 | 900390 | 10021 | Boeing | 2026-03-21 | 2026-05-15 | 2026-03-21 | 490.00 | 490.00 | O | 6 | 1-30 Days | 2 |
| 7 | 900490 | 10021 | Boeing | 2025-12-11 | 2026-05-25 | 2025-12-11 | 560.00 | 560.00 | O | -4 | Current | 1 |
| 8 | 900590 | 10021 | Boeing | 2025-09-02 | 2026-06-04 | 2025-09-02 | 690.00 | 690.00 | O | -14 | Current | 1 |
| 9 | 900675 | 10021 | Boeing | 2025-06-09 | 2026-05-30 | 2025-06-09 | 775.00 | 775.00 | H | -9 | Current | 1 |
| 10 | 900774 | 10021 | Boeing | 2026-03-02 | 2026-06-08 | 2026-03-02 | 874.00 | 874.00 | O | -18 | Current | 1 |

dq_referential_integrity: 5 FK checks all PASS with 0 orphans. F4201 customer_id to F0101.address_id, F4211 customer_id to F0101.address_id, F4211 order_number to F4201.order_number, F4211 item_id to F4101.item_id, F03B11 customer_id to F0101.address_id.

Row Count Reconciliation

```

13 SELECT * FROM $joe_demo.gold.dq_row_count_reconciliation L1M1T 18;
14
15

```

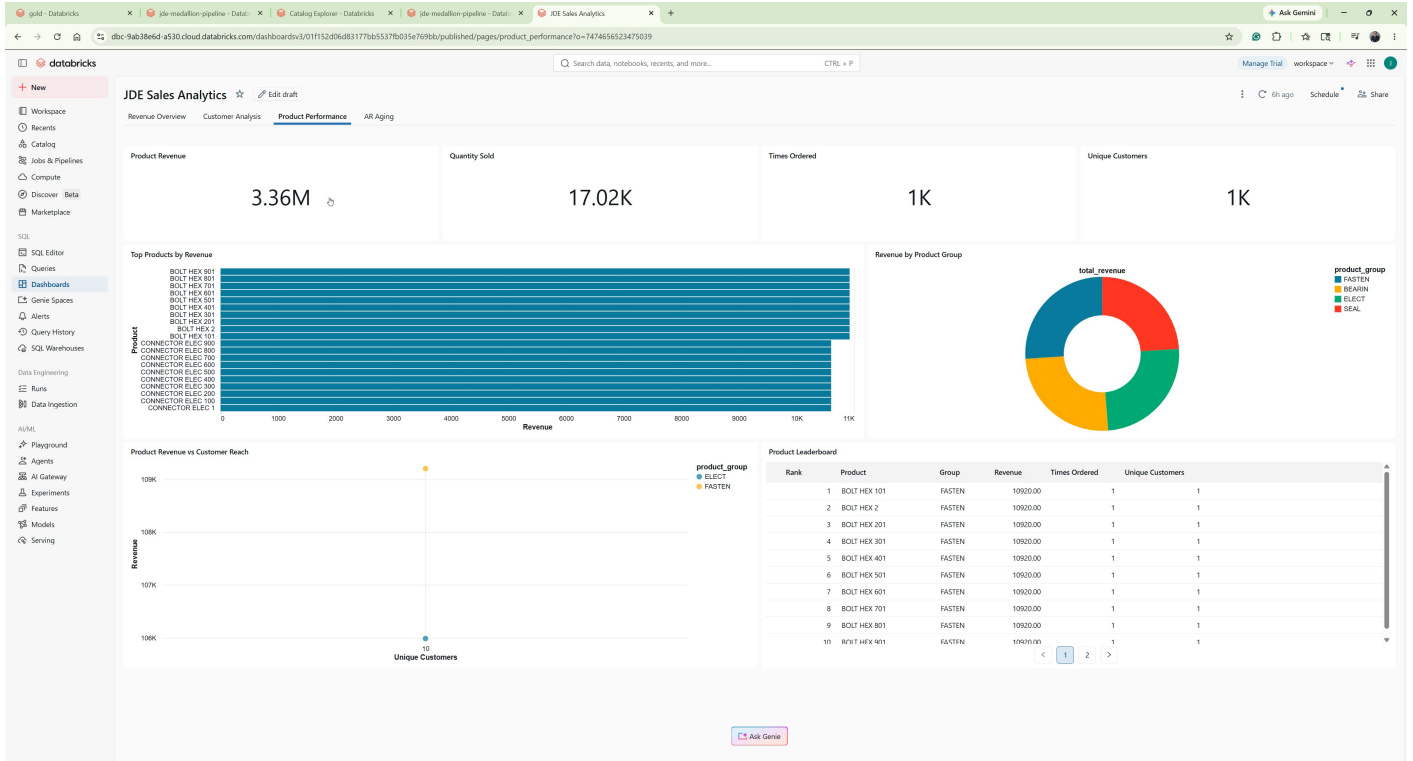
| | table_name | count_bronze | count_silver | count_dropped | result |
|---|------------|--------------|--------------|---------------|--------|
| 1 | F4201 | 1000 | 1000 | 0 | PASS |
| 2 | F4211 | 1000 | 1000 | 0 | PASS |
| 3 | F0101 | 151 | 151 | 0 | PASS |
| 4 | F0116 | 1000 | 1000 | 0 | PASS |
| 5 | F4101 | 1000 | 1000 | 0 | PASS |
| 6 | F03B11 | 1000 | 1000 | 0 | PASS |

dq_row_count_reconciliation: All 6 tables PASS. Bronze vs Silver counts match perfectly (F4201: 1000/1000, F4211: 1000/1000, F0101: 151/151, F0116: 1000/1000, F4101: 1000/1000, F03B11: 1000/1000). Zero rows dropped by DLT Expectations.

9. AI/BI Dashboards

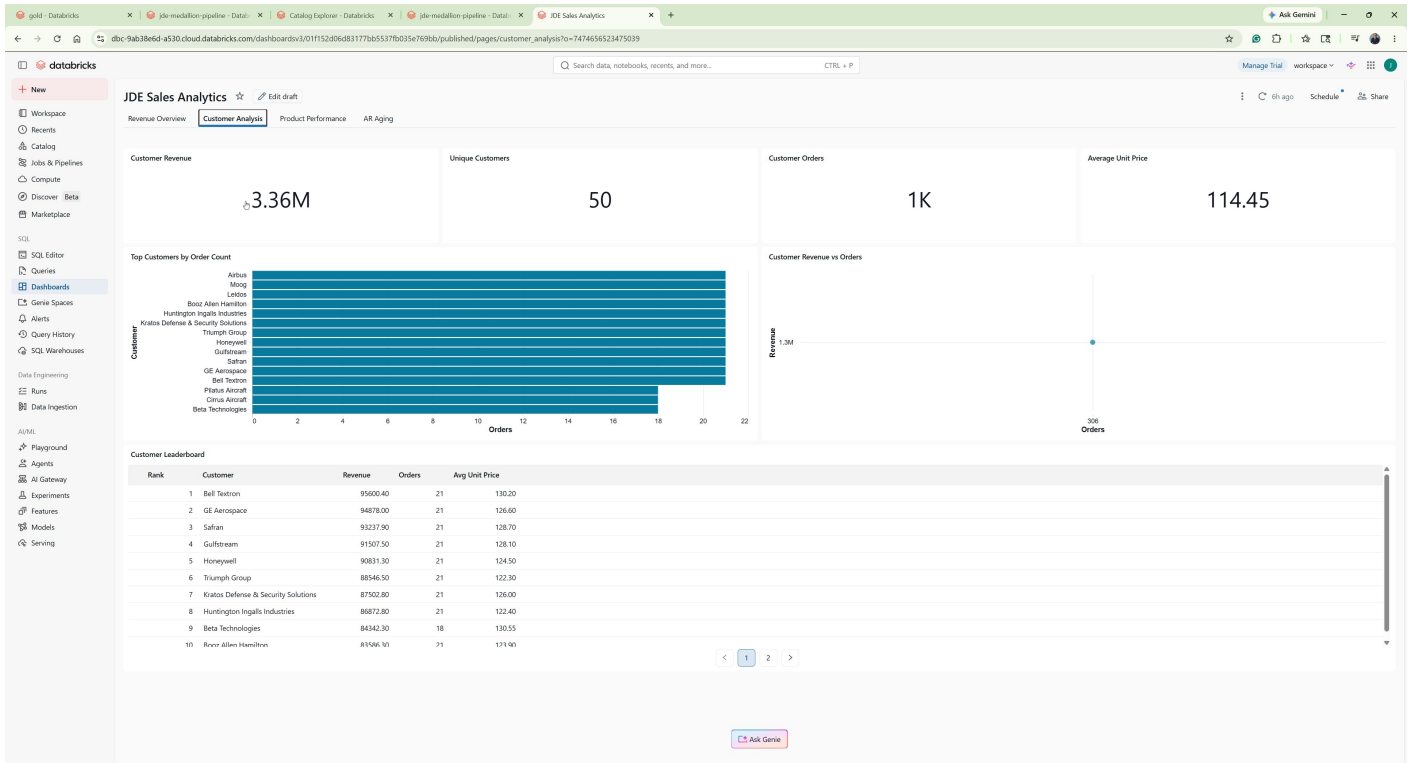
The JDE Sales Analytics dashboard is a 4-tab AI/BI dashboard built with Genie Code. It provides Revenue Overview, Customer Analysis, Product Performance, and AR Aging views with KPI cards, bar charts, donut charts, scatter plots, and detailed data tables.

Dashboard: Product Performance



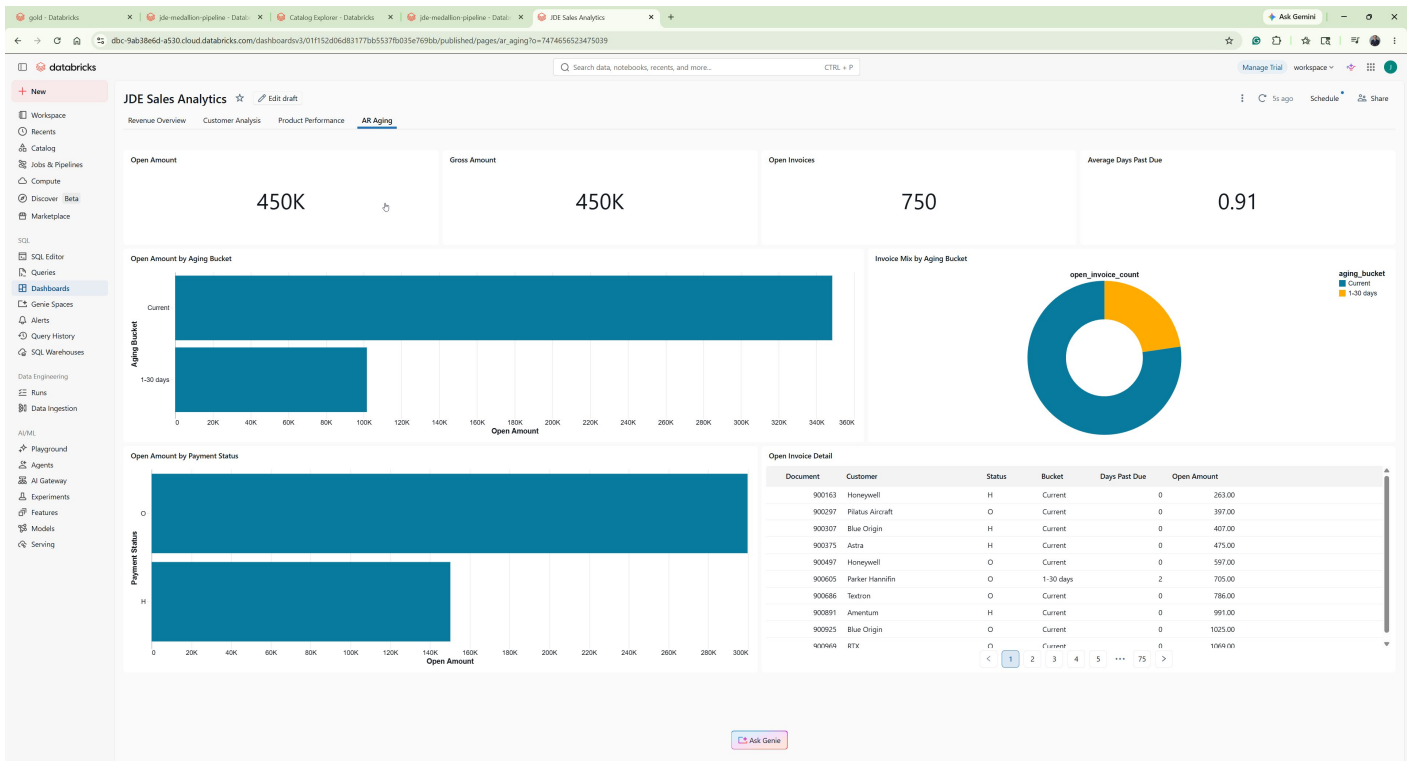
Product Performance tab: \$3.36M product revenue, 17.02K quantity sold, 1K times ordered, 1K unique customers. Top Products by Revenue bar chart, Revenue by Product Group donut (FASTEN, BEARIN, ELECT, SEAL), Product Leaderboard table.

Dashboard: Customer Analysis



Customer Analysis tab: \$3.36M revenue, 50 unique customers, 1K orders, \$114.45 avg unit price. Top Customers by Order Count (Airbus, Moog, Leidos leading), Customer Revenue vs Orders scatter, Customer Leaderboard (Bell Textron #1 \$95,600, GE Aerospace #2 \$94,878, Safran #3 \$93,237).

Dashboard: AR Aging



AR Aging tab: \$450K open amount, \$450K gross amount, 750 open invoices, 0.91 avg days past due. Open Amount by Aging Bucket (Current dominant), Invoice Mix by Aging Bucket donut, Open Amount by Payment Status, Open Invoice Detail (Honeywell, Blue Origin, Astra, RTX).

10. Genie Spaces

The JDE Aerospace Sales Analytics Genie Space enables natural language queries against the Gold layer. It includes pre-built SQL queries (Top 10 Customers by Revenue, Top Products by Revenue) and role-based sharing aligned with the Unity Catalog RBAC model.

Genie Space: Natural Language Query

The screenshot shows the Databricks workspace interface. At the top, there are several tabs for different queries. The active query is titled "New Query 2026-05-22 16:57:42". The SQL editor contains the following query:

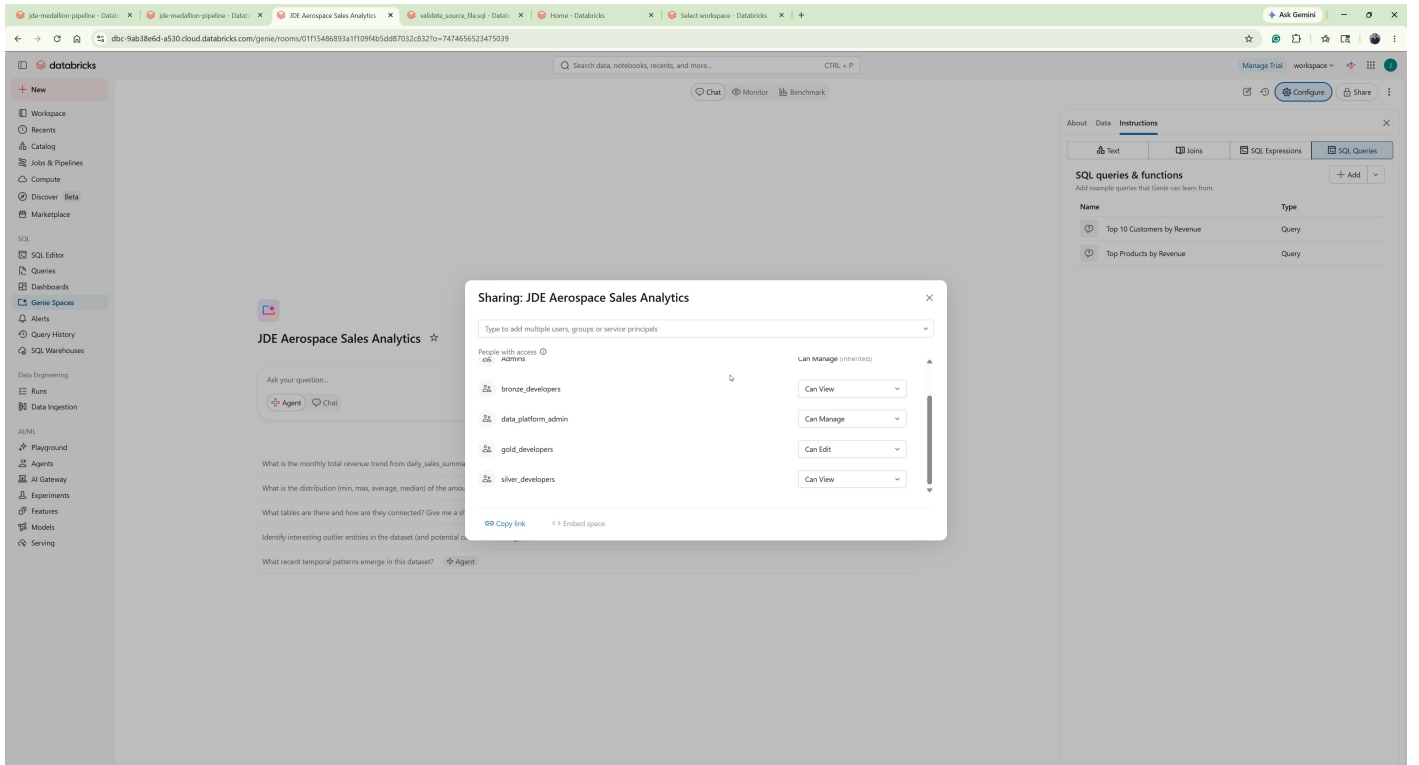
```
1 SELECT * FROM $db_demo.bronze.f4211_raw LIMIT 10;  
2  
3 SELECT * FROM $db_demo.silver.f4211_cleaned LIMIT 10;  
4  
5 SELECT * FROM $db_demo.gold.sales_orders_enriched LIMIT 10;  
6  
7 SELECT * FROM $db_demo.gold.ar_uping_analysis LIMIT 10;  
8  
9 SELECT * FROM $db_demo.gold.dsp_customer_scdd LIMIT 10;
```

Below the query editor, a table of results is displayed. The table has 15 columns and 10 rows of data. The columns are: address_id, address_key, address_type, customer_name, description, tax_id, sic_code, credit_message, credit_rating, date_updated_jstn, updated_by_user, silver_processed_at, _START_AT, and _END_AT.

| address_id | address_key | address_type | customer_name | description | tax_id | sic_code | credit_message | credit_rating | date_updated_jstn | updated_by_user | silver_processed_at | _START_AT | _END_AT |
|------------|-------------|--------------|------------------|------------------|------------|----------|----------------|---------------|------------------------------|-----------------|------------------------------|------------------------------|---------|
| 10001 | 10001.0 | V | Ducommun | Ducommun | 7800000017 | 3611 | | USD | 2026-05-21T17:15:13.112+0000 | PDJMO | 2026-05-21T17:15:13.112+0000 | 2026-05-21T17:15:13.112+0000 | 3611 |
| 10002 | 10002.0 | E | John Paul Castro | John Paul Castro | 7800000034 | 3611 | | USD | 2026-05-21T17:15:13.112+0000 | PDJMO | 2026-05-21T17:15:13.112+0000 | 2026-05-21T17:15:13.112+0000 | 3611 |
| 10003 | 10003.0 | C | Lockheed Martin | Lockheed Martin | 7800000051 | 3611 | | USD | 2026-05-21T17:15:13.112+0000 | PDJMO | 2026-05-21T17:15:13.112+0000 | 2026-05-21T17:15:13.112+0000 | 3611 |
| 10004 | 10004.0 | V | Kaman Aerospace | Kaman Aerospace | 7800000068 | 3611 | | USD | 2026-05-21T17:15:13.112+0000 | PDJMO | 2026-05-21T17:15:13.112+0000 | 2026-05-21T17:15:13.112+0000 | 3611 |
| 10005 | 10005.0 | E | Sarah Chen | Sarah Chen | 7800000085 | 3611 | 1 | USD | 2026-05-21T17:15:13.112+0000 | PDJMO | 2026-05-21T17:15:13.112+0000 | 2026-05-21T17:15:13.112+0000 | 3611 |
| 10006 | 10006.0 | C | RTX | RTX | 7800000102 | 3611 | | USD | 2026-05-21T17:15:13.112+0000 | PDJMO | 2026-05-21T17:15:13.112+0000 | 2026-05-21T17:15:13.112+0000 | 3611 |
| 10007 | 10007.0 | V | Triumph Group | Triumph Group | 7800000119 | 3611 | | USD | 2026-05-21T17:15:13.112+0000 | PDJMO | 2026-05-21T17:15:13.112+0000 | 2026-05-21T17:15:13.112+0000 | 3611 |
| 10008 | 10008.0 | E | Robert Torres | Robert Torres | 7800000136 | 3611 | | USD | 2026-05-21T17:15:13.112+0000 | PDJMO | 2026-05-21T17:15:13.112+0000 | 2026-05-21T17:15:13.112+0000 | 3611 |
| 10009 | 10009.0 | C | SpaceX | SpaceX | 7800000153 | 3611 | | USD | 2026-05-21T17:15:13.112+0000 | PDJMO | 2026-05-21T17:15:13.112+0000 | 2026-05-21T17:15:13.112+0000 | 3611 |
| 10010 | 10010.0 | V | Curios-Wright | Curios-Wright | 7800000170 | 3611 | 1 | USD | 2026-05-21T17:15:13.112+0000 | PDJMO | 2026-05-21T17:15:13.112+0000 | 2026-05-21T17:15:13.112+0000 | 3611 |

Genie Space answering 'Which products have the most orders?' with natural language response, Top 15 Products by Revenue bar chart (BOLT HEX and CONNECTOR ELEC items at ~\$10.9K each), Revenue by Product Group chart. Configure panel showing General Instructions.

Genie Space: Role-Based Sharing

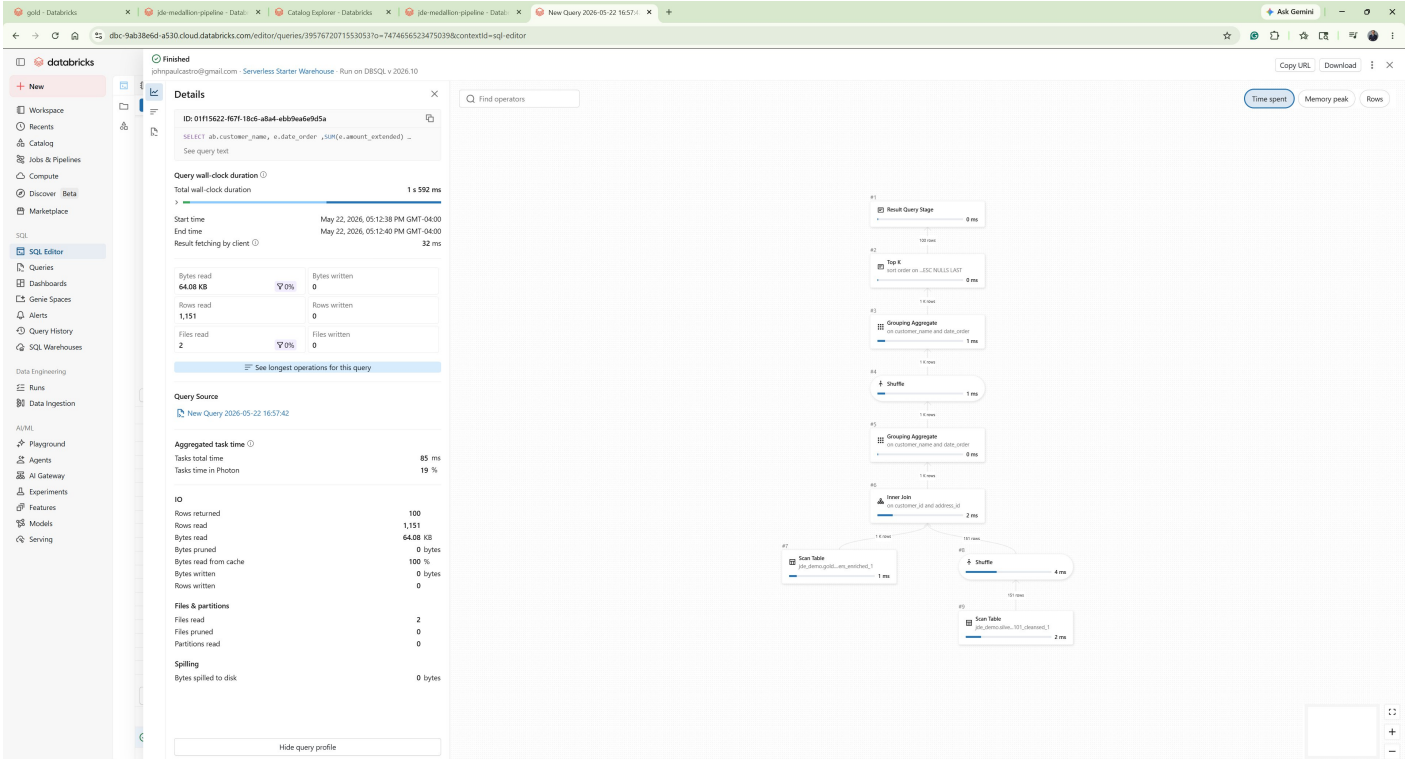


Genie Space sharing: Admins (Can Manage, inherited), bronze_developers (Can View), data_platform_admin (Can Manage), gold_developers (Can Edit), silver_developers (Can View). SQL Queries tab showing pre-built 'Top 10 Customers by Revenue' and 'Top Products by Revenue'.

11. Query Profiling

Query profiling demonstrates Databricks Serverless SQL performance with Photon engine acceleration. The execution plan shows scan, join, aggregate, and sort stages with timing breakdowns and I/O statistics.

Query Profile: Gold Layer Join

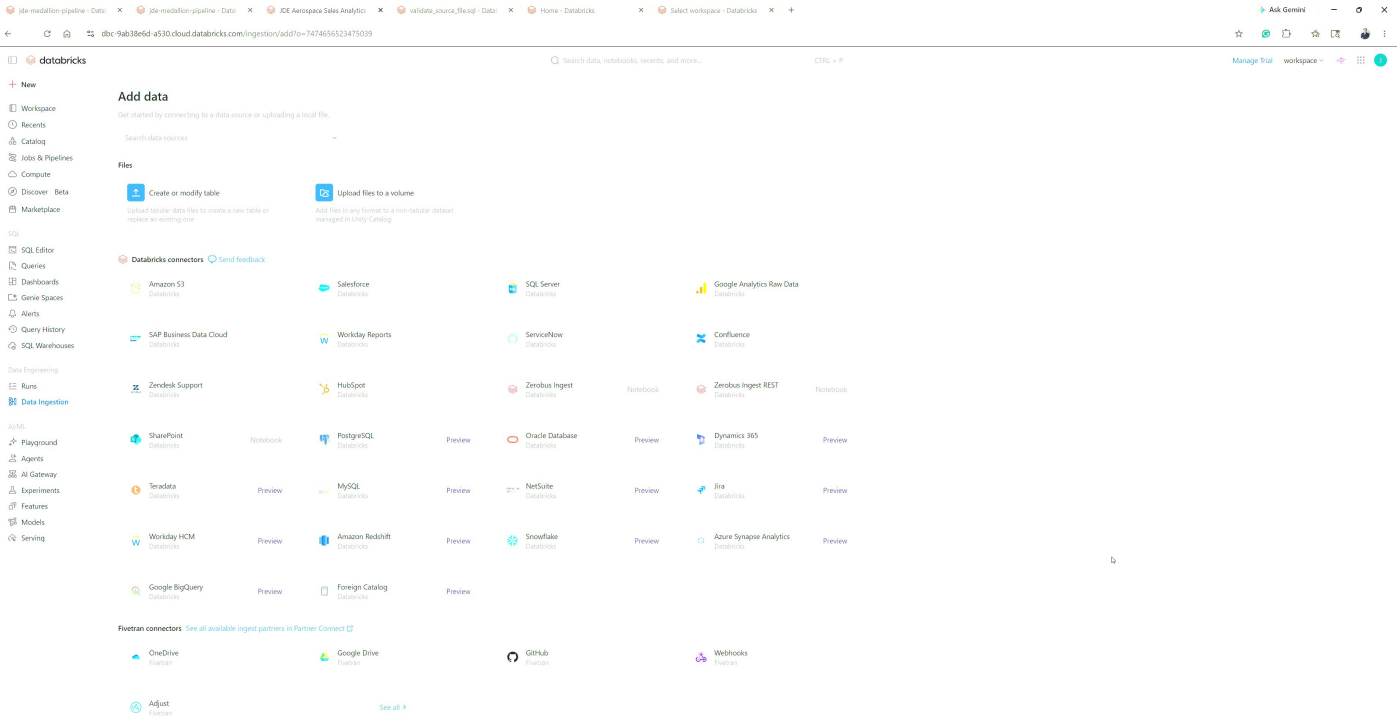


Query profile: 1.592s wall-clock duration, 19% Photon execution. Execution plan showing Scan Table (gold.sales_orders_enriched 1ms) > Inner Join on customer_id/address_id (2ms) > Grouping Aggregate (1ms) > Shuffle (4ms) > Top K sort. 1,151 rows read, 64.08 KB, 100% bytes from cache, 0 bytes spilled to disk.

12. Data Ingestion Connectors

Databricks supports native connectors to enterprise data sources including SQL Server, Salesforce, Oracle, PostgreSQL, SAP, Workday, and more. Fivetran connectors extend coverage to GitHub, Google Drive, OneDrive, and webhook-based sources. This portfolio used CSV file upload via Unity Catalog Volumes, but a production deployment would use Lakeflow Connect with the SQL Server connector for CDC-based real-time ingestion.

Add Data: Available Connectors



Data Ingestion page: Databricks connectors (Amazon S3, Salesforce, SQL Server, Google Analytics, SAP, Workday, ServiceNow, Confluence, HubSpot, PostgreSQL, Oracle, Dynamics 365, Teradata, MySQL, NetSuite, Jira, Snowflake, Azure Synapse, BigQuery, Foreign Catalog). Fivetran connectors (OneDrive, Google Drive, GitHub, Webhooks).

This portfolio was built on a Databricks 14-day trial workspace (May 2026) using real JDE ERP data patterns from 25+ years of aerospace distribution experience. Every screenshot represents live, executed work on a running Databricks environment. The platform demonstrates hands-on proficiency with Unity Catalog, Lakeflow Declarative Pipelines, Serverless SQL, Databricks Workflows, AI/BI Dashboards, Genie Spaces, and enterprise RBAC governance.

John Paul (JP) Castro | jpcenterprises.com | johnpaulcastro@gmail.com