Databricks Lakehouse Project
Medallion Pipeline (Bronze / Silver / Gold)

PySpark • Delta Lake • Unity Catalog • Jobs Orchestration • Star Schema (Gold)

Portfolio summary

End-to-end ingestion, cleansing, and analytics publishing pipeline over the Iowa Liquor Sales dataset.
Silver applies strict data-quality rules; Gold publishes a star schema for BI/KPIs.
Designed for repeatable runs (MERGE-based loads) and operationalized with Jobs + orchestrator.

Highlights

- Medallion architecture with layer-by-layer jobs and end-to-end orchestrator (scheduled twice daily).
- Gold model: dim_time, dim_store, dim_item (Type I) + fact_sales partitioned by (year, month).
- Idempotency & dedup strategy documented (MERGE / analytics-only dedup view).

Andres Olguin

Data Engineer — Python / SQL / PySpark / Databricks

github.com/Andy47UTN/databricks-medallion-lakehouse-pipeline

Portfolio version • 2026-02-15

Tabla de contenido

1) Versionamiento

- Versión: 1.0
- Fecha: 2025-10-25
- Autor: Andres Olguin
- Versión inicial para entrega del Proyecto Final

2) Introducción

Solución de ingesta, depuración, modelado y publicación analítica sobre Databricks/Delta Lake para el dataset de ventas de licores de Iowa. Se utiliza arquitectura medalla (Bronze/Silver/Gold), modelo estrella en Gold y automatización con Jobs y orquestador programado. El foco es asegurar calidad, trazabilidad y performance para responder consultas de negocio.

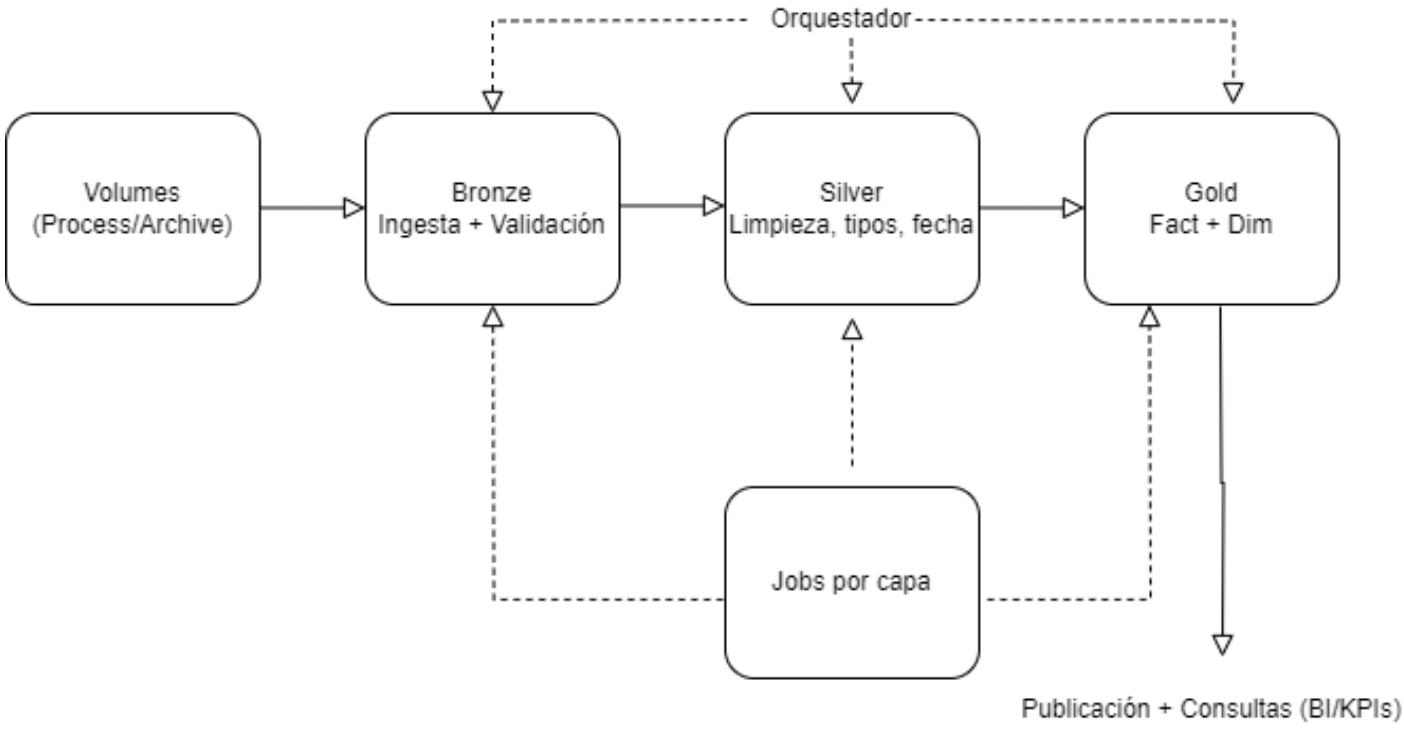
3) Objetivo general

Implementar un pipeline productizable que permita analizar ventas (tendencias, ranking por tienda/condado, categorías y rentabilidad) con datos confiables y procesos repetibles.

4) Objetivos específicos

1. Estandarizar la ingesta y el archivado de archivos con control de *naming* (_yyyyMMdd).
2. Construir Silver con reglas de calidad estrictas (anti-desplazamiento, tipificación, fechas válidas).
3. Publicar Gold con dimensiones Tipo I y fact particionada por year, month.
4. Exponer consultas de negocio (tendencias, ranking, crecimiento por categoría, margen bruto).
5. Automatizar la ejecución con Jobs por capa y orquestador con triggers L-V 06:00/18:00.

5) Diagrama de Arquitectura



6) Diagrama lógico de la solución (modelo estrella)

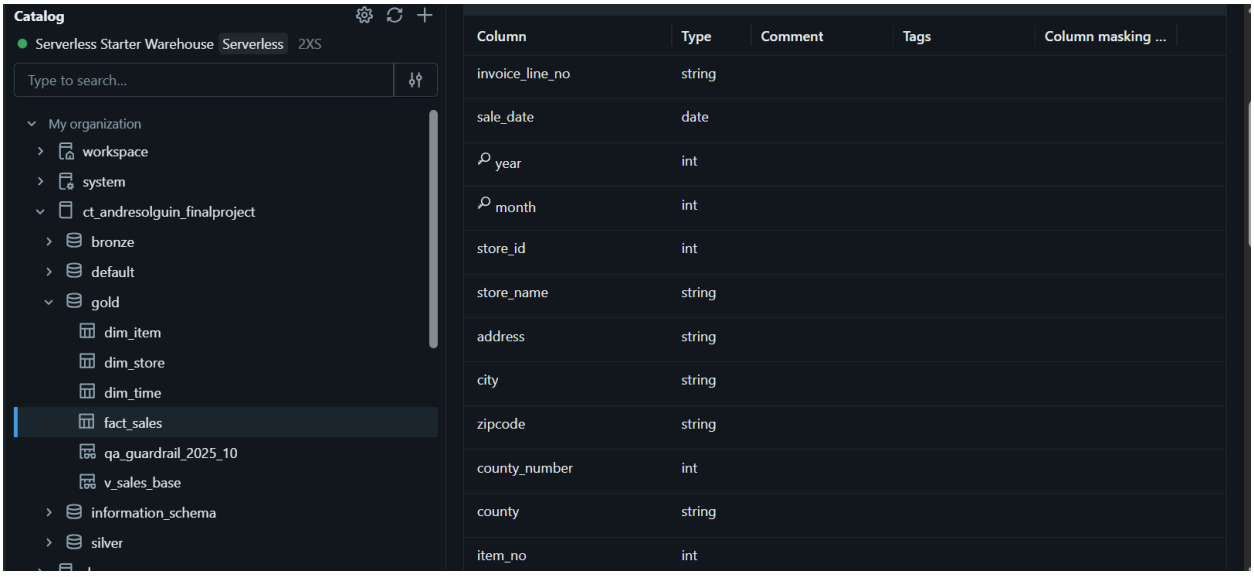
Gold:

dim_time(date_key PK, date, year, quarter, month, month_name, day,
week_of_year, day_of_week, day_name, is_weekend) [Tipo I]

dim_store(store_id PK, store_name, address, city, zipcode,
county_number, county) [Tipo I]

dim_item(item_no PK, item_desc, category, category_name,
vendor_no, vendor_name, pack_int, bottle_volume_ml_d) [Tipo I]

fact_sales(invoice_line_no PK, date_key FK, store_id FK, item_no FK,
sale_bottles, sale_dollars, sale_liters, sale_gallons,
state_bottle_cost, state_bottle_retail, year, month, updated_at)
PARTITIONED BY (year, month) — Delta.



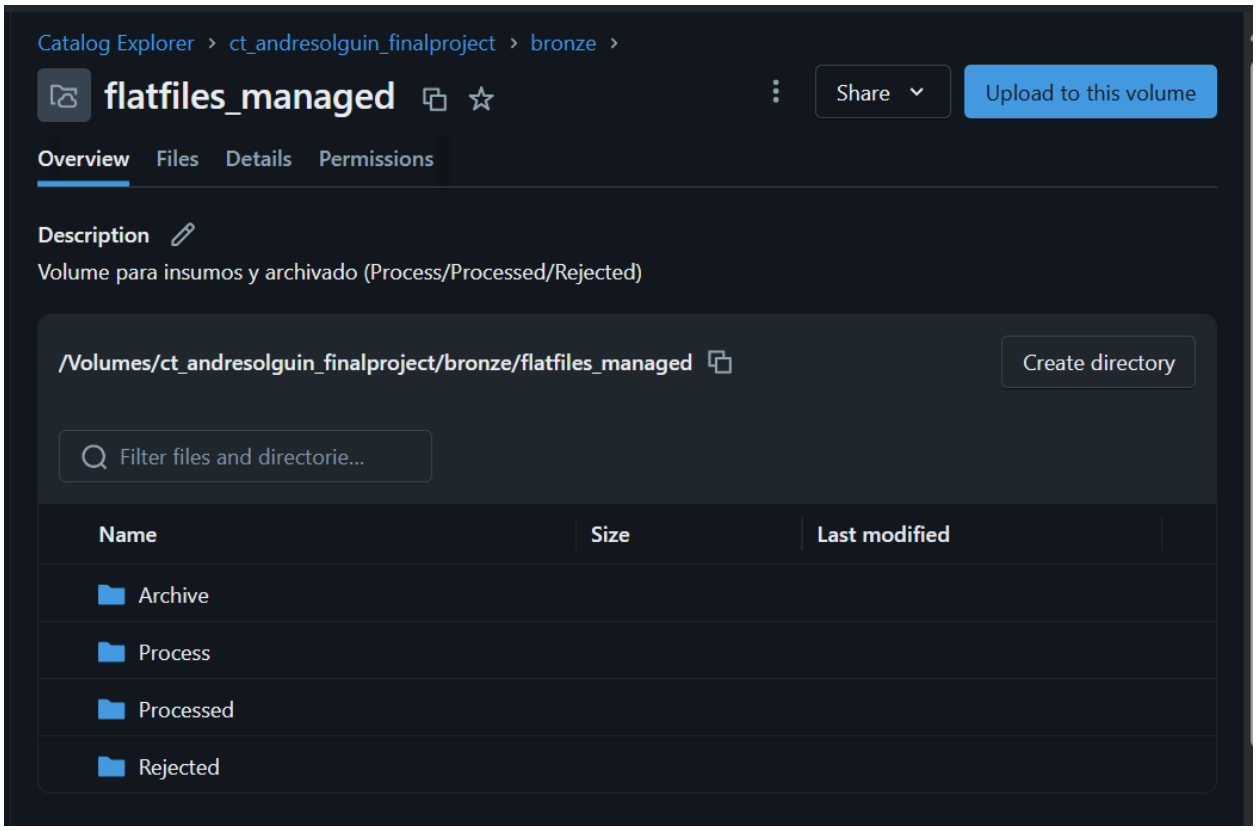
7) Naming convention

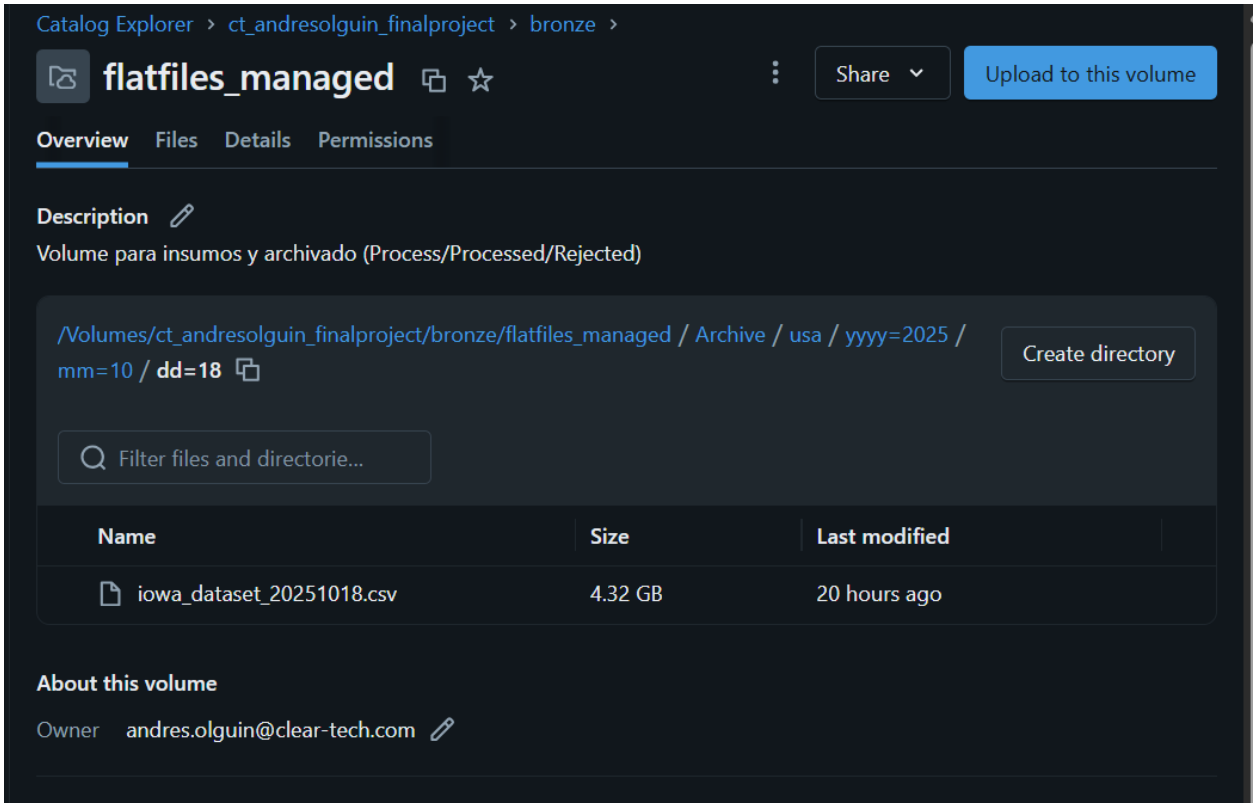
- **Catálogo:** ct_andresolguin_finalproject
- **Esquemas:** bronze, silver, gold
- **Volumen:** flatfiles_managed
- **Rutas:** /Volumes/{catalog}/{bronze}/flatfiles_managed/{Process|Archive}/{country}/yyyy=YYYY/mm=MM/dd=DD/
- **Archivos de entrada:** iowa_dataset_YYYYMMDD.csv
- **Split en 4:** iowa_dataset_YYYYMMDD_part-{1..4}.csv
- **Tablas**

  Silver: silver.iowa_sales_clean, silver.iowa_clean_v2_strict

  Gold: gold.dim_time, gold.dim_store, gold.dim_item, gold.fact_sales

- **Jobs:** CT_Bronze_Ingest, CT_Silver_Clean, CT_Gold_Publish, CT_Pipeline_Orchestrator
- **Notebooks (carpetas en Workspace):** /bronze, /silver, /gold, /queries, /workflows, /docs

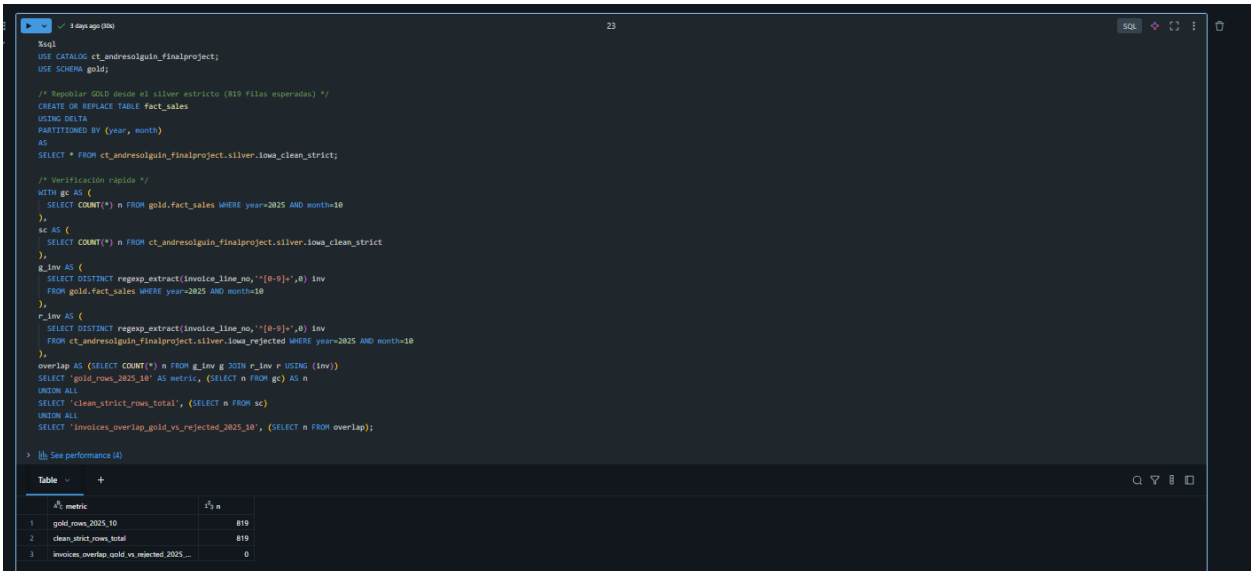




8) Diseño técnico de la solución

○ 8.1 Tecnologías utilizadas

Databricks (Notebooks, Jobs, Volumes, Unity Catalog), Delta Lake (ACID, MERGE), PySpark/Spark SQL, Lakehouse (arquitectura de medallas).

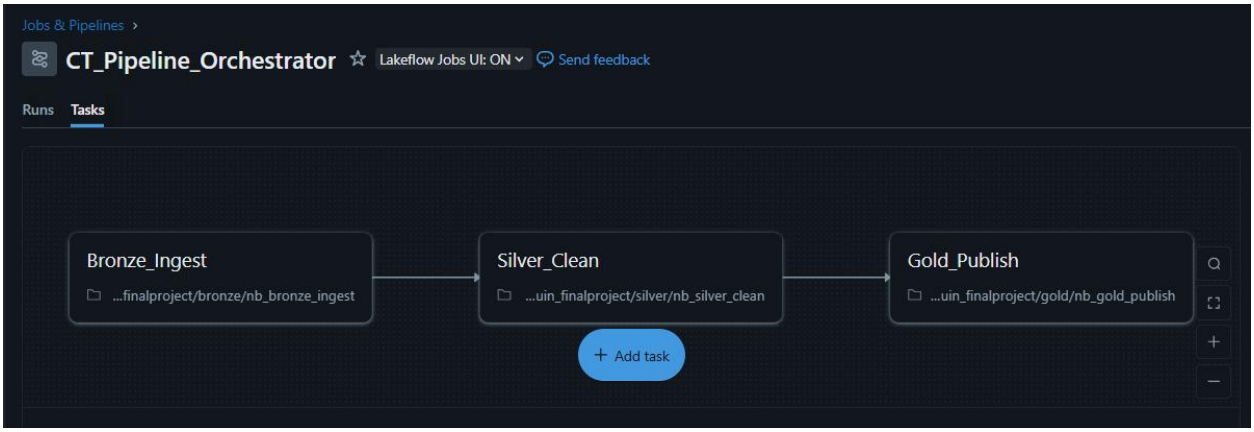


```
%sql
USE CATALOG ct_andresolguin_finalproject;
USE SCHEMA gold;

/* Repoblar GOLD desde el silver estricto (819 filas esperadas) */
CREATE OR REPLACE TABLE fact_sales
USING DELTA
PARTITIONED BY (year, month)
AS
SELECT * FROM ct_andresolguin_finalproject.silver.iowa_clean_strict;

/* Verificación rápida */
WITH gc AS (
  SELECT COUNT(*) n FROM gold.fact_sales WHERE year=2025 AND month=10
),
sc AS (
  SELECT COUNT(*) n FROM ct_andresolguin_finalproject.silver.iowa_clean_strict
),
g_inv AS (
  SELECT DISTINCT regexp_extract(invoice_line_no,'^[0-9]+',0) inv
  FROM gold.fact_sales WHERE year=2025 AND month=10
),
r_inv AS (
  SELECT DISTINCT regexp_extract(invoice_line_no,'^[0-9]+',0) inv
  FROM ct_andresolguin_finalproject.silver.iowa_rejected WHERE year=2025 AND month=10
),
overlap AS (SELECT COUNT(*) n FROM g_inv g JOIN r_inv r USING (inv))
SELECT 'gold_rows_2025_10' AS metric, (SELECT n FROM gc) AS n
UNION ALL
SELECT 'clean_strict_rows_total', (SELECT n FROM sc)
UNION ALL
SELECT 'invoices_overlap_gold_vs_rejected_2025_10', (SELECT n FROM overlap);
```

metric	n
gold_rows_2025_10	819
clean_strict_rows_total	819
invoices_overlap_gold_vs_rejected_2025_...	0

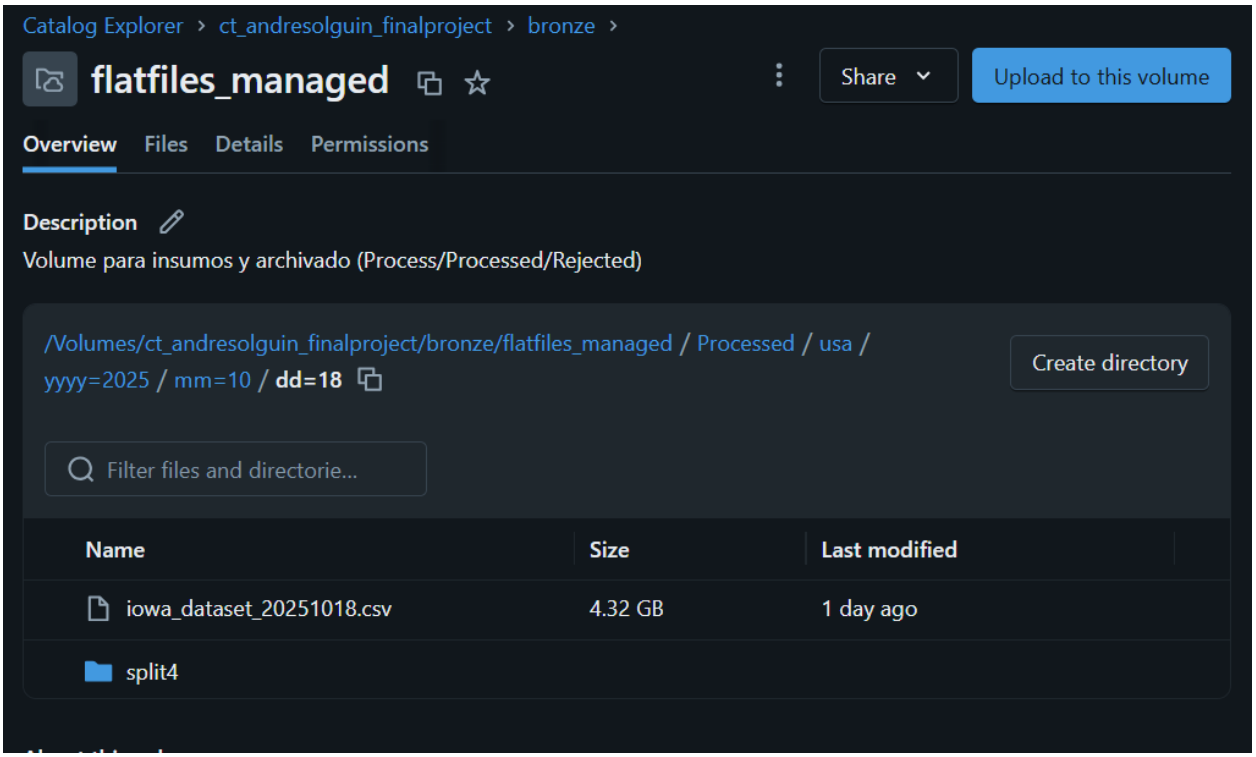
8.2 Componentes

Volumes (landing y archive), tablas Delta por capa, Notebooks parametrizados con widgets, Jobs por capa, Orquestador, Triggers, queries de negocio.



8.3 Archivos requeridos

- iowa_dataset_20251018.csv (fuente)
- iowa_dataset_20251018_part-1..4.csv (split en Bronze)



8.4 Capas y objetos

Bronze

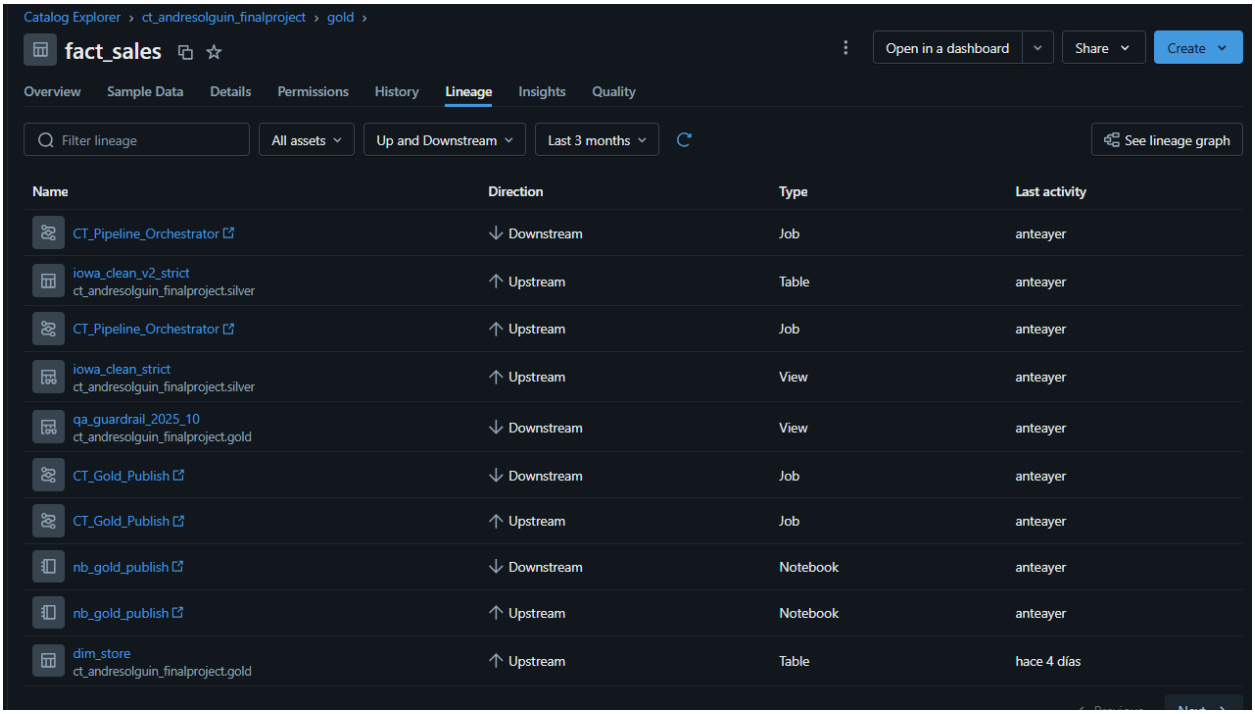
- Validación de naming (*_YYYYMMDD.csv), split en 4 archivos, archivado pos-proceso.

Silver

- iowa_sales_clean: normalización básica.
- iowa_clean_v2_strict: reglas de calidad (fecha válida, rangos de pack y bottle_volume_ml, sale_bottles entero y rango, precios > 0, retail ≥ cost, etc.).

Gold

- dim_time (rango dinámico entre MIN/MAX(sale_date) en Silver).
- dim_store, dim_item → MERGE Tipo I (sin duplicar al reprocesar).
- fact_sales → MERGE idempotente desde silver.iowa_clean_v2_strict, partition (year, month).



8.5 Tabla de Hechos (Fact)

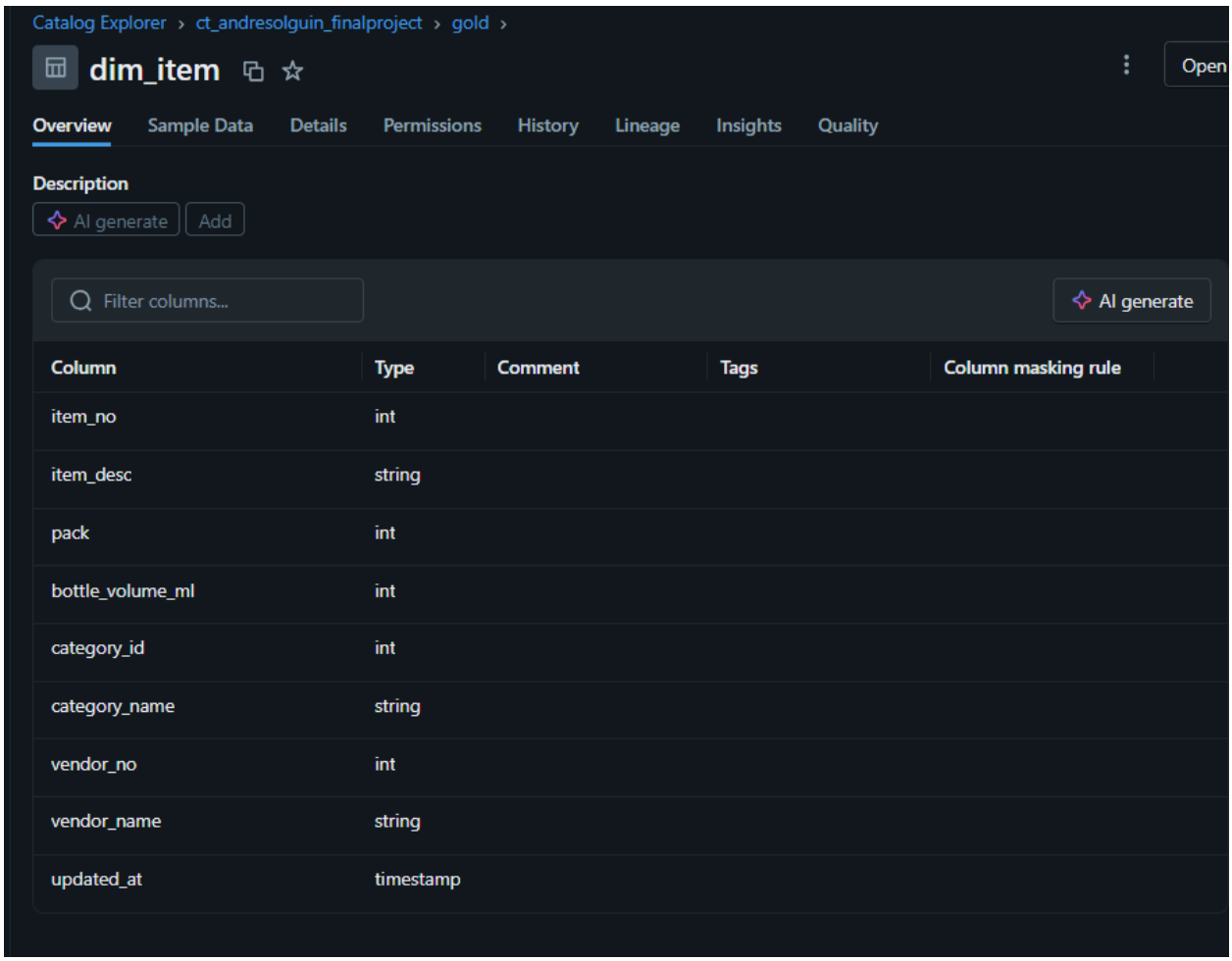
gold.fact_sales

- **PK lógico**: invoice_line_no
- **FK**: date_key → dim_time, store_id → dim_store, item_no → dim_item
- **Medidas**: sale_bottles INT, sale_dollars DECIMAL(14,2), sale_liters DECIMAL(12,3), sale_gallons DECIMAL(12,3)
- **Atributos de costo/precio**: state_bottle_cost DECIMAL(12,2), state_bottle_retail DECIMAL(12,2)
- **Particiones**: year INT, month INT

8.6 Dimensiones

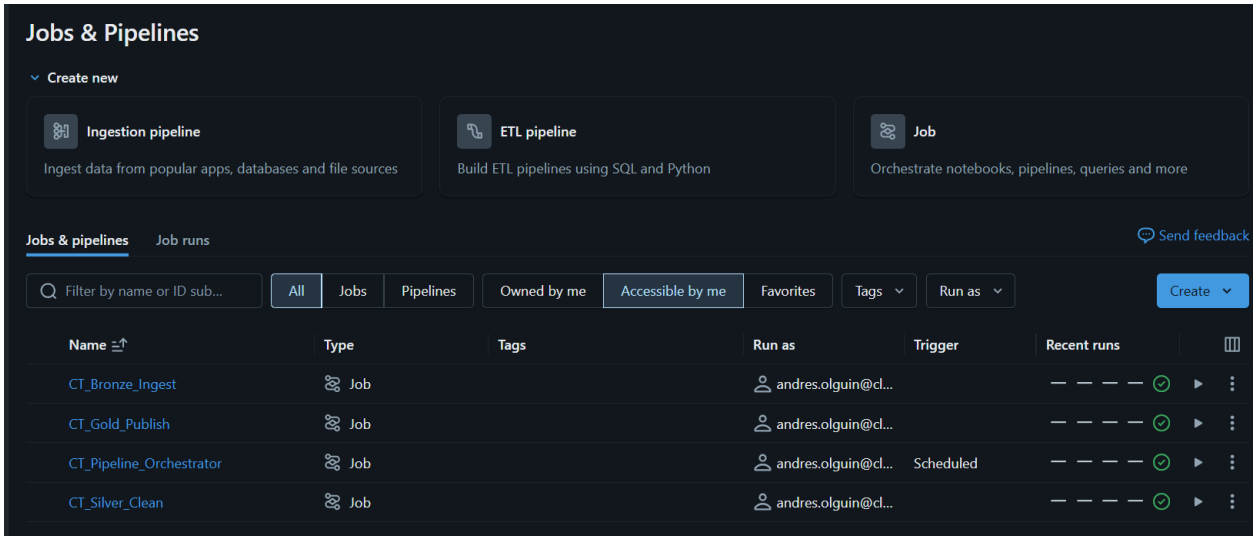
- **dim_time** (PK date_key INT yyyyMMdd) — Tipo I.
- **dim_store** (PK store_id INT) — Tipo I.
- **dim_item** (PK item_no INT) — Tipo I.

Todas cargadas por MERGE (upsert) sin duplicados al reprocesar



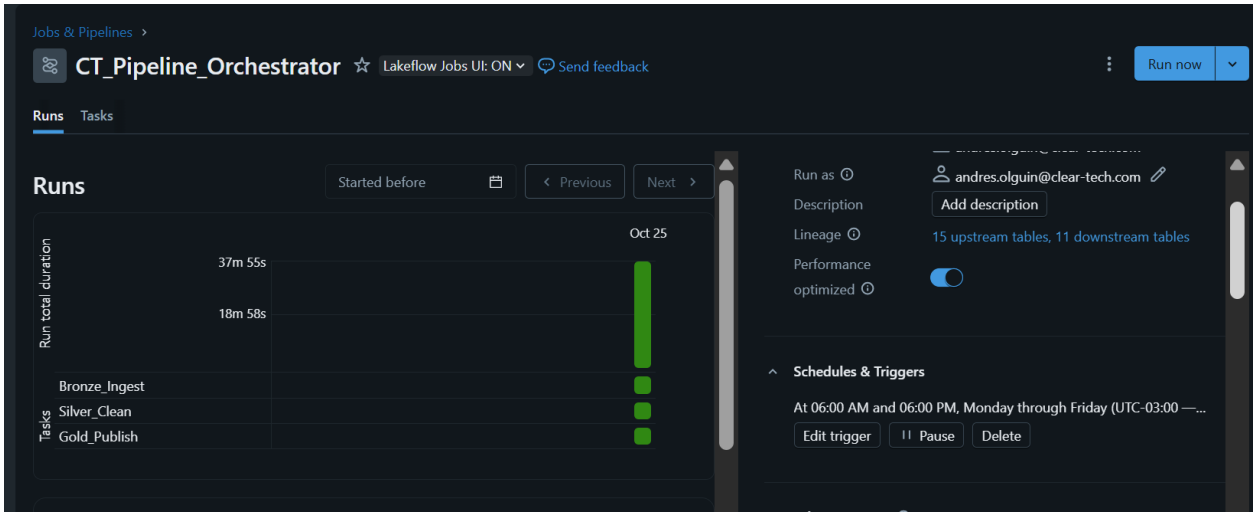
8.7 Automatización (Jobs)

- **CT_Bronze_Ingest** - ingesta/validación/archivado.
- **CT_Silver_Clean** - construcción de iowa_clean_v2_strict.
- **CT_Gold_Publish** - creación/actualización de dim_* + fact_sales.
- **CT_Pipeline_Orchestrator** - orquesta 3 tareas en secuencia.



8.8 Periodicidad (Triggers)

- Orquestador programado dos veces al día (L–V): 06:00 y 18:00.
- Todos los jobs permiten ejecución manual (“Run now”).



9) Manual de uso

Ejecución por capa

1. Abrir el job de la capa y Run now.
2. Verificar/ajustar parámetros (widgets): catalog, schema, process_date, etc.
3. Revisar Runs → Logs si hace falta.

Ejecución orquestada

1. Abrir CT_Pipeline_Orchestrator y Run now (o esperar al trigger).
2. Verificar Gantt de tasks (todo en Succeeded).

Reprocesos / Idempotencia

- Dimensiones y fact usan MERGE; es seguro reprocesar sin duplicar.
- fact_sales particionada por (year, month) optimiza pruning y cargas.

Recuperación de errores

- Revisar logs del run fallido.
- Re-lanzar la task afectada o el orquestador completo.
- Los insumos quedan archivados por fecha de proceso (trazabilidad).

10) Anexos

A. Notebooks

- /bronze/nb_bronze_ingest
- /silver/nb_silver_clean
- /gold/nb_gold_publish

B. Queries (resumen)

- Tendencias diarias/mensuales/anuales (con YoY donde aplica).
- Ranking tiendas y condados (botellas y ventas).
- Categorías Top y mayor crecimiento YoY (último año completo).
- Margen por item, categoría y vendor; panel precio–volumen (elasticidad y sugerencia de precio).