Pipeline de formato medallón en Databricks

Proyecto: ETL para la empresa TechMart

Daniel Morales López – Data Engineer

2025

Introducción del Proyecto y de la Empresa

El presente proyecto titulado **“Pipeline de Formato Medallón en Databricks”** tiene como objetivo diseñar e implementar un flujo de datos escalable que permita a la empresa TechMart consolidar, transformar y analizar su información de ventas de manera eficiente. La solución se basa en la arquitectura Medallion **(Bronze → Silver → Gold)**, utilizando Apache Spark en Databricks como plataforma principal. El pipeline integra los archivos de ventas generados por cada sucursal, aplica procesos de limpieza y estandarización, y finalmente produce métricas de negocio consolidadas que sirven como base para la toma de decisiones estratégicas. Además, se desarrolló un notebook con visualizaciones gráficas a partir de la capa Gold, lo que facilita la interpretación de resultados y la generación de reportes gerenciales en tiempo real. El proyecto se ejecuta mediante un Databricks Job que automatiza las tres etapas del pipeline, garantizando trazabilidad, escalabilidad y eficiencia en el manejo de datos.

TechMart es una empresa de e-commerce que opera en Costa Rica, con 12 sucursales distribuidas en diferentes ciudades del país. Su portafolio de productos abarca cinco categorías principales: **Electrónica, Ropa, Hogar, Deportes y Alimentos**. Durante el año 2024, TechMart experimentó un crecimiento significativo en sus ventas, especialmente en fechas clave como Black Friday, Cyber Monday y la temporada navideña. Sin embargo, la compañía enfrenta un reto crítico: cada sucursal gestiona sus datos de ventas de manera independiente, lo que dificulta obtener una visión unificada del negocio. Actualmente, los reportes gerenciales se generan de forma manual a partir de múltiples archivos CSV, un proceso que consume varios días de trabajo y limita la capacidad de reacción de la gerencia. Esta situación provoca que las decisiones estratégicas se basen en información desactualizada y fragmentada, lo que afecta la competitividad de la empresa en un mercado dinámico. Con la implementación del pipeline de datos en Databricks, TechMart busca superar estas limitaciones, logrando visibilidad consolidada del negocio, comparativas entre sucursales, análisis de productos más vendidos, identificación de tendencias temporales y generación de KPIs clave como ventas totales y ticket promedio.

Objetivos del Proyecto

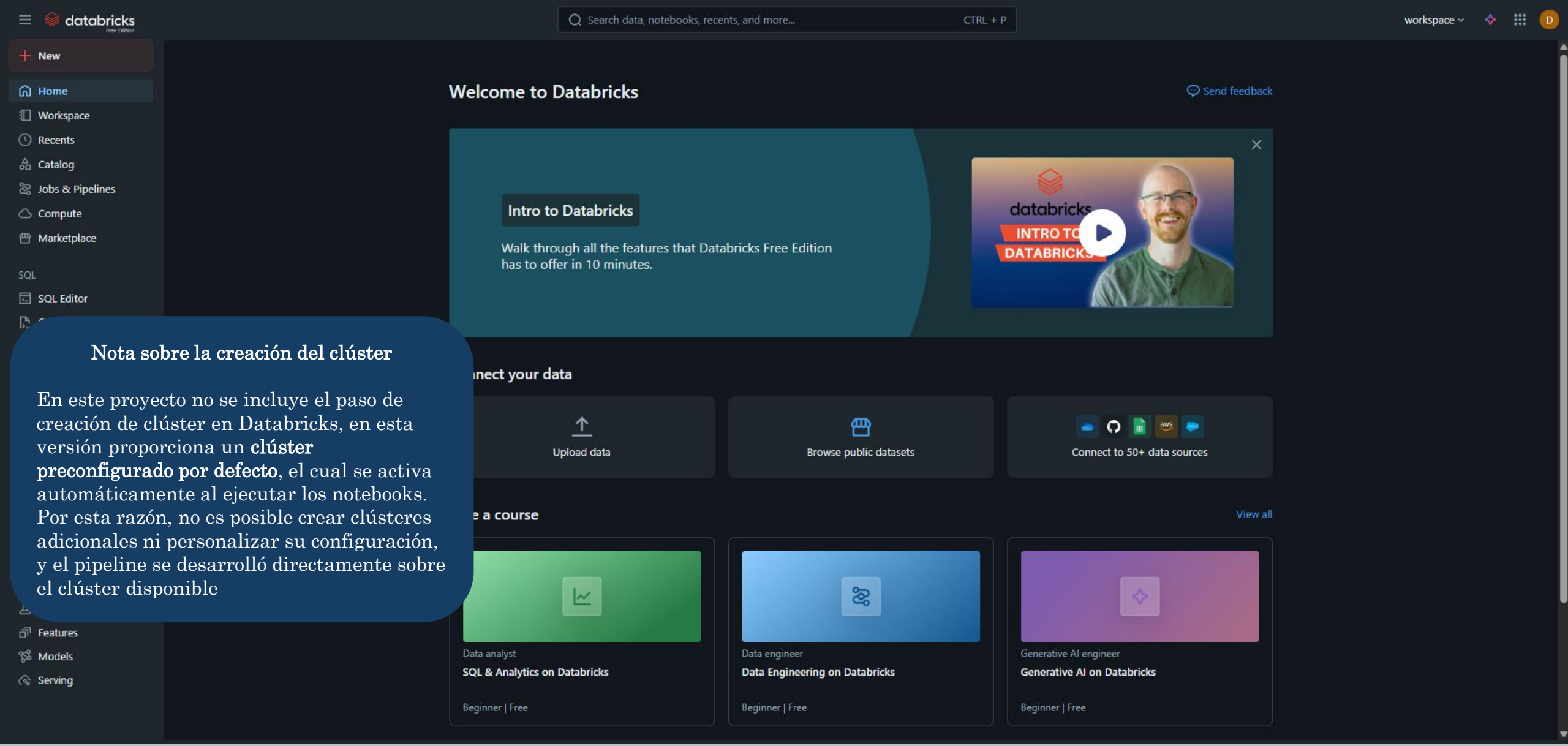
El objetivo principal del proyecto es **diseñar e implementar un pipeline de datos escalable en Databricks utilizando Apache Spark**, que permita a TechMart integrar, transformar y analizar la información de ventas proveniente de sus 12 sucursales de manera unificada. Con esta solución se busca superar las limitaciones actuales en la gestión de datos y habilitar una toma de decisiones estratégicas basada en información confiable y actualizada.

De manera específica, el proyecto persigue los siguientes objetivos:

- **Integrar los archivos de ventas diarios** generados por cada sucursal en un repositorio centralizado.
- **Estandarizar y limpiar los datos** aplicando las mejores prácticas de calidad y consistencia.
- **Transformar la información en métricas de negocio consolidadas**, incluyendo indicadores clave como ventas totales, ticket promedio y productos más vendidos.
- **Facilitar comparativas entre sucursales** para identificar patrones de rendimiento y oportunidades de mejora.
- **Habilitar el análisis de tendencias temporales**, como variaciones mensuales o estacionales en el comportamiento de compra.
- **Automatizar el proceso de integración y reportería** mediante la ejecución de un Job en Databricks que gestione las tres capas de la arquitectura Medallion (Bronze, Silver y Gold).
- **Proveer visualizaciones gráficas interactivas** basadas en la capa Gold, que permitan a la gerencia interpretar resultados y generar reportes en tiempo real.

En conjunto, estos objetivos garantizan que TechMart pueda contar con un sistema de datos moderno, confiable y escalable, capaz de transformar información dispersa en conocimiento estratégico para impulsar su crecimiento en el mercado costarricense.

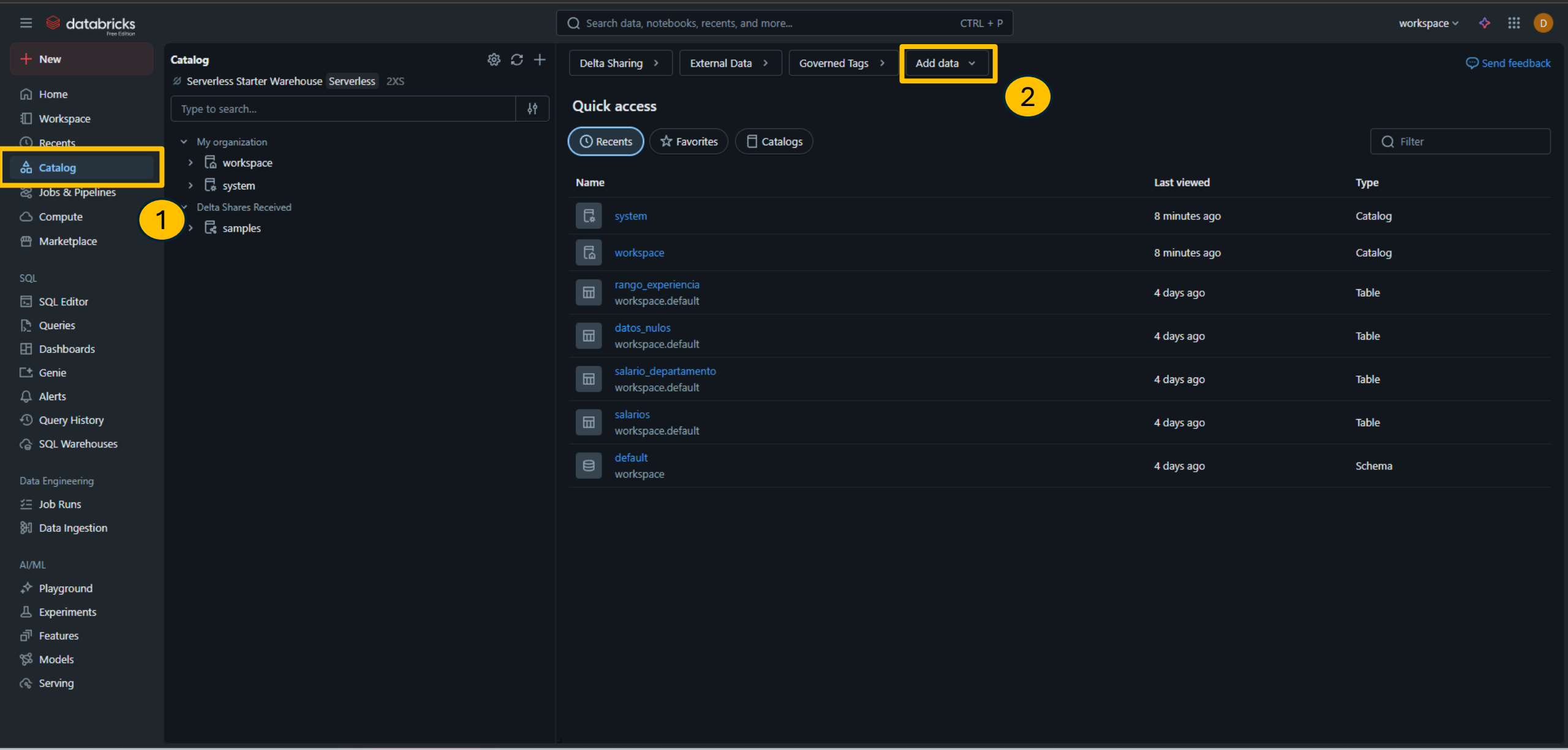
Documentación Técnica del Pipeline



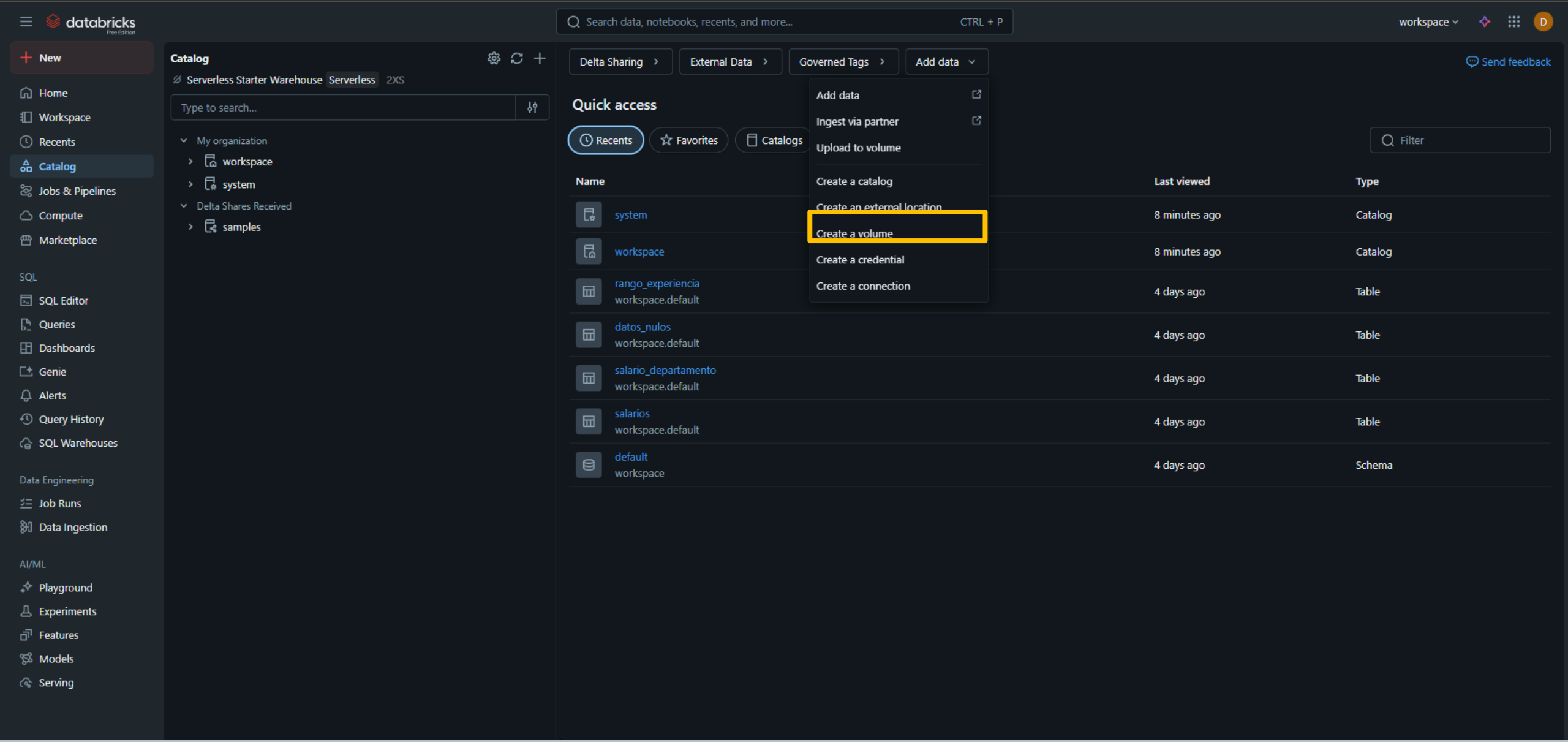
Nota sobre la creación del clúster

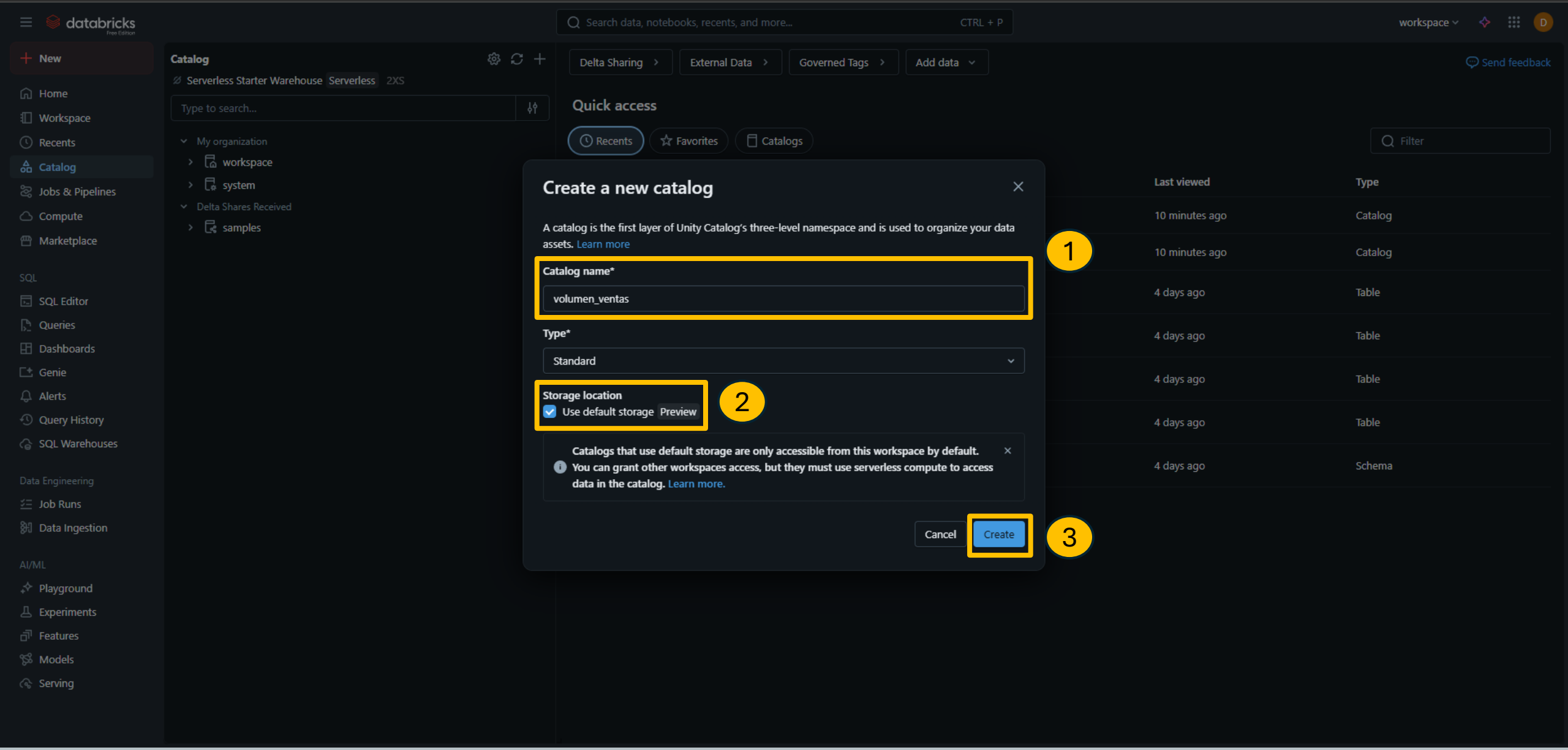
En este proyecto no se incluye el paso de creación de clúster en Databricks, en esta versión proporciona un **clúster preconfigurado por defecto**, el cual se activa automáticamente al ejecutar los notebooks. Por esta razón, no es posible crear clústeres adicionales ni personalizar su configuración, y el pipeline se desarrolló directamente sobre el clúster disponible

Para iniciar ingresamos a Databricks, por motivos de este proyecto estemos usando la versión gratuita de Databricks llamada Databricks Community Edition, Free Edition

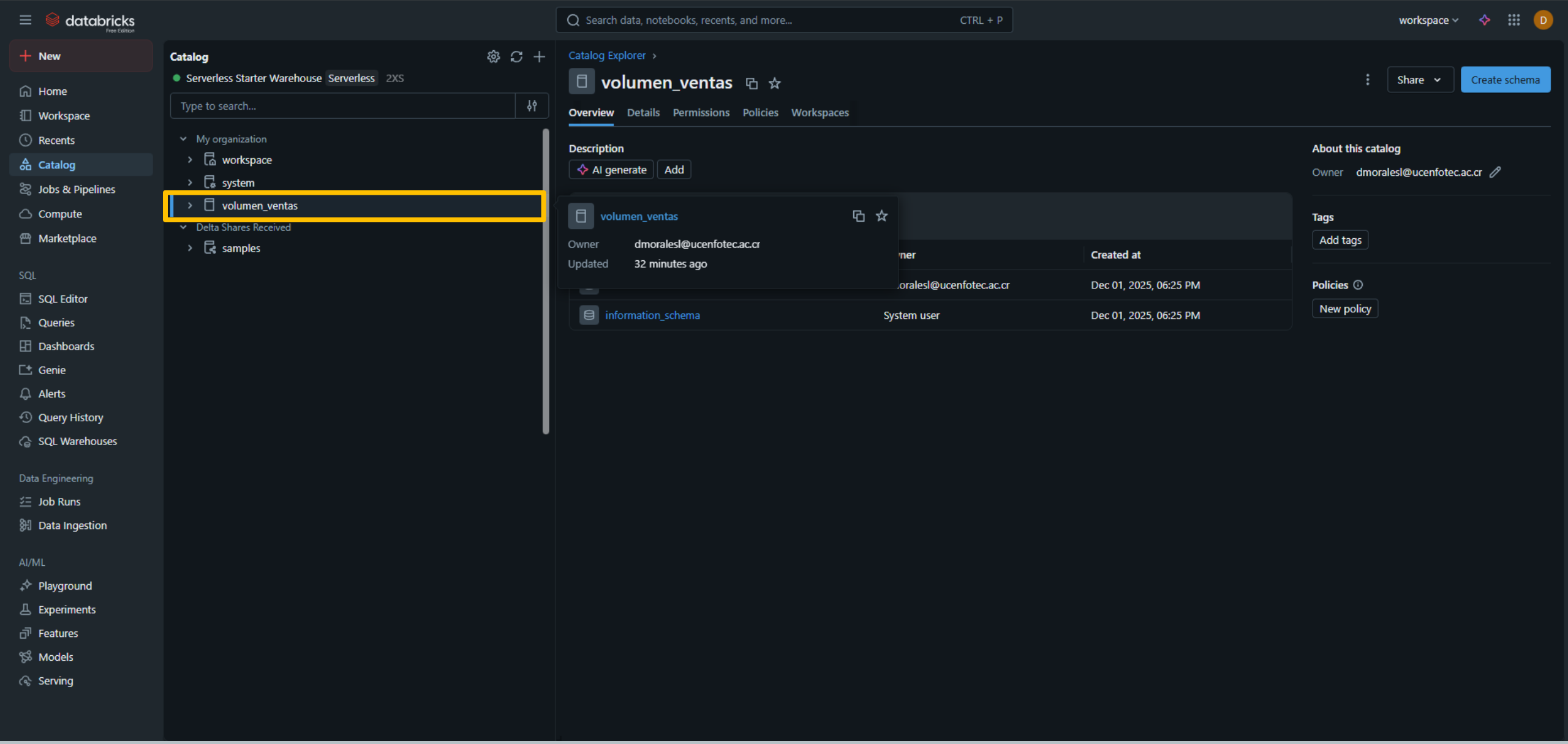


Empezaremos creando nuestro Catalog en Databricks

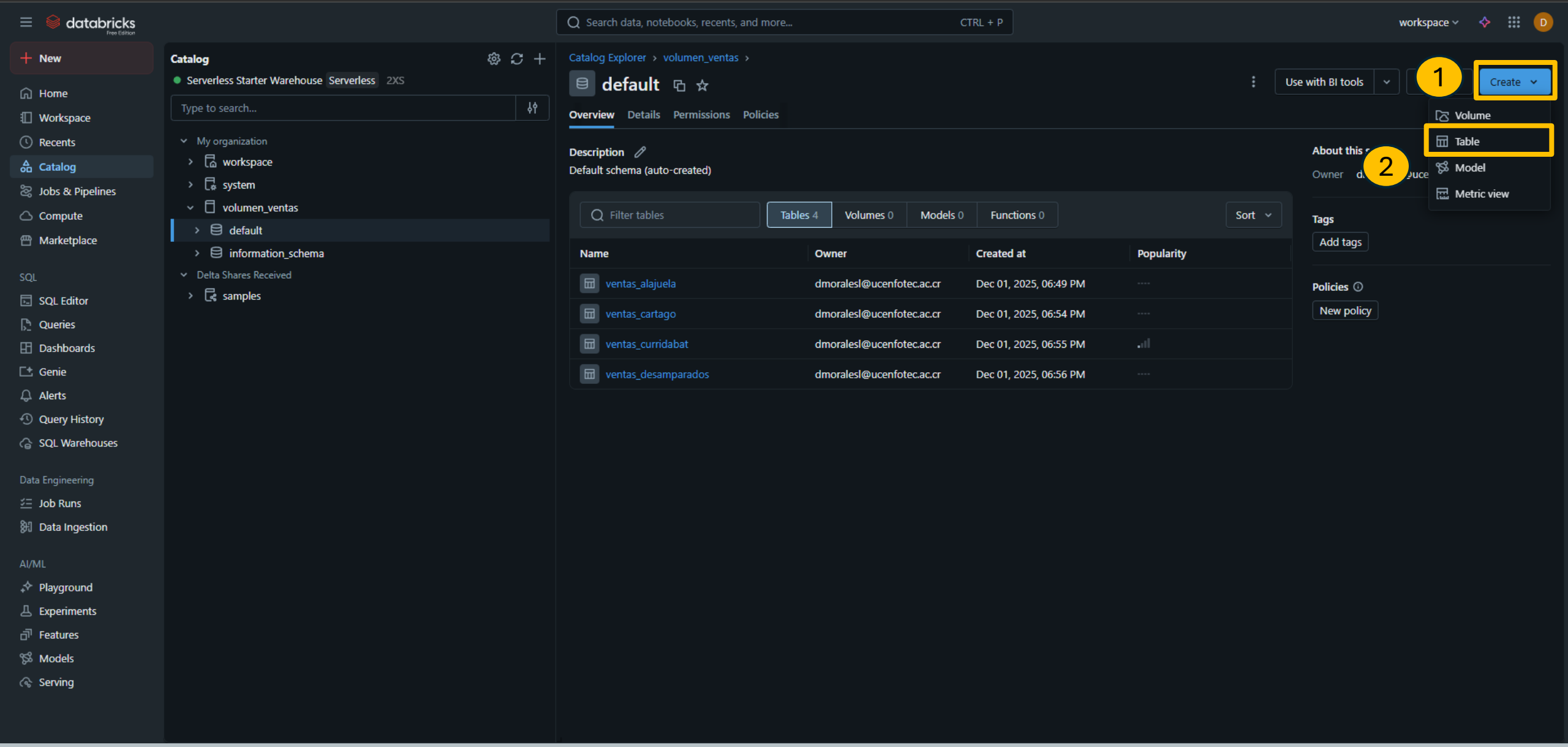

databricks
Free Edition
Search data, notebooks, recents, and more...
CTRL + P
workspace
New
Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace
SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses
Data Engineering
Job Runs
Data Ingestion
AI/ML
Playground
Experiments
Features
Models
Serving
Catalog
Serverless Starter Warehouse
Serverless
2XS
Type to search...
My organization
workspace
system
Delta Shares Received
samples
Delta Sharing
External Data
Governed Tags
Add data
Send feedback
Add data
Ingest via partner
Upload to volume
Create a catalog
Create an external location
Create a volume
Create a credential
Create a connection
Quick access
Recents
Favorites
Catalogs
Filter
Name
Last viewed
Type
system
8 minutes ago
Catalog
workspace
8 minutes ago
Catalog
rango_experiencia
workspace.default
4 days ago
Table
datos_nulos
workspace.default
4 days ago
Table
salario_departamento
workspace.default
4 days ago
Table
salarios
workspace.default
4 days ago
Table
default
workspace
4 days ago
Schema



Le ponemos primero el nombre a nuestro Catalog, después por motivos de este proyecto usaremos el almacenamiento default que nos da Databricks y ya luego por últimos lo creamos el Catalog



Ya tendríamos creado nuestro Catalog



Ahora insertaremos todos los CSV de la empresa al Catalog

New
Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace

SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses

Data Engineering
Job Runs
Data Ingestion

AI/ML
Playground
Experiments
Features
Models
Serving

Add data ›

Create or modify table from file upload

Serverless Starter Warehouse

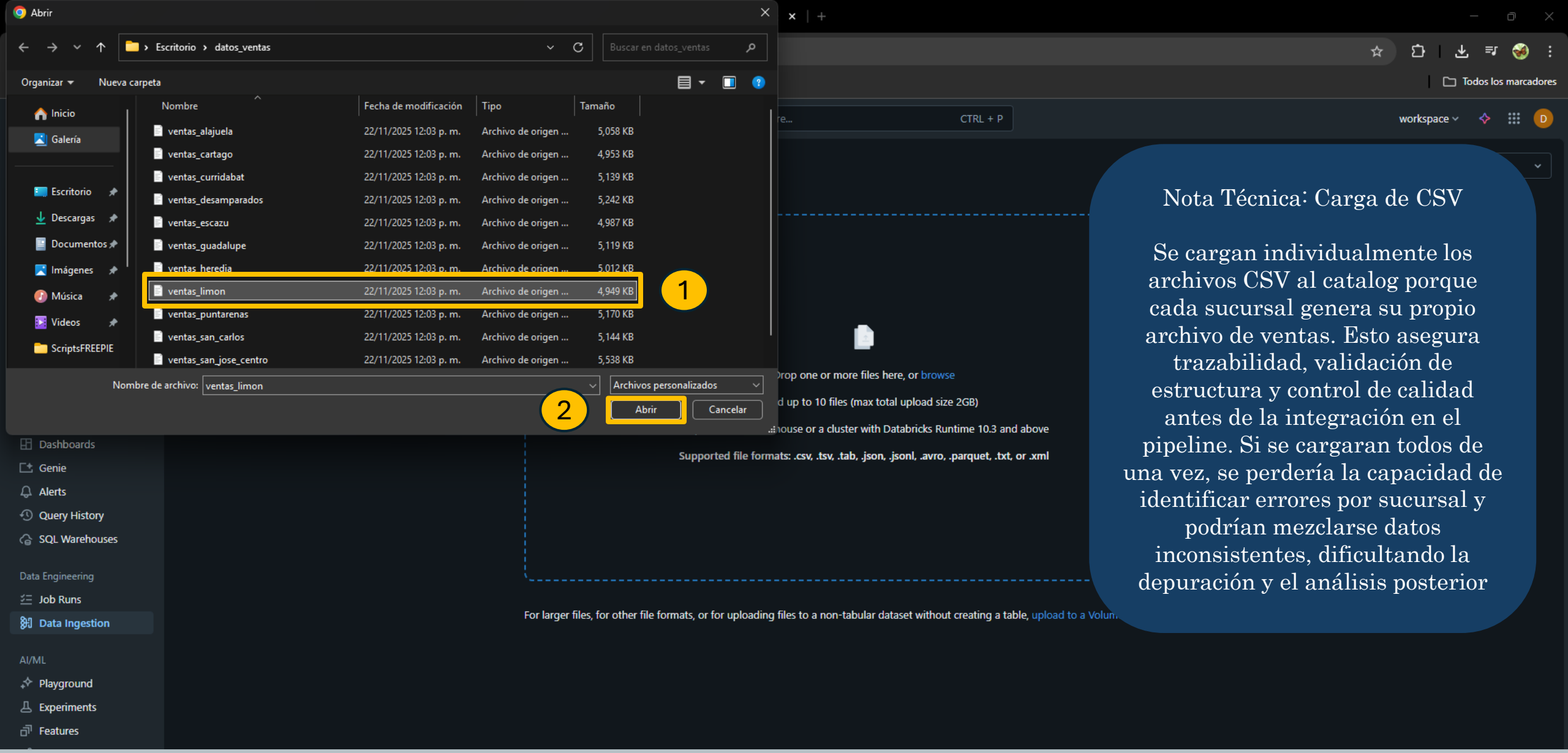
Drop one or more files here, or browse

Upload up to 10 files (max total upload size 2GB)

Requires a SQL warehouse or a cluster with Databricks Runtime 10.3 and above

Supported file formats: .csv, .tsv, .tab, .json, .jsonl, .avro, .parquet, .txt, or .xml

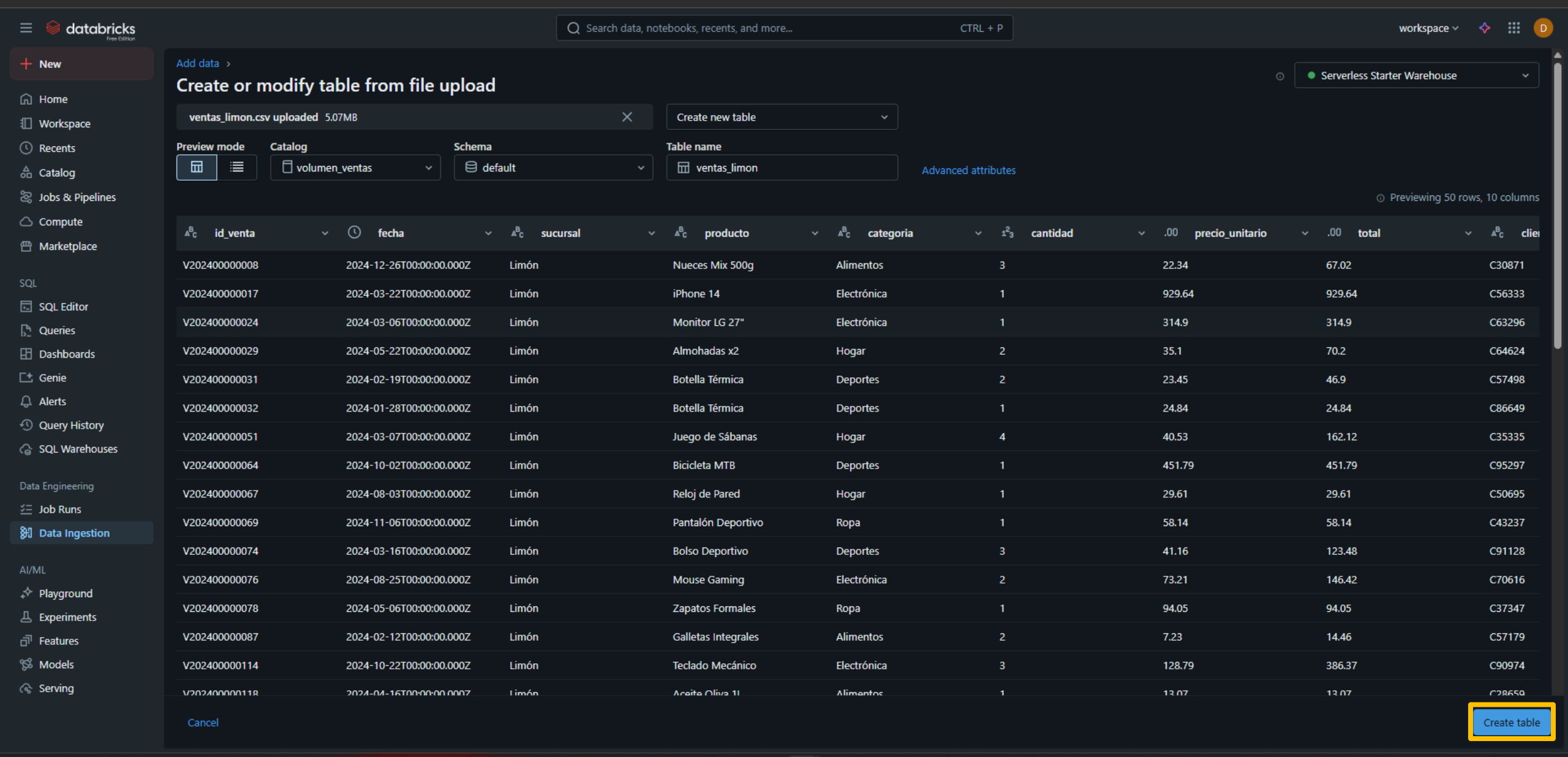
For larger files, for other file formats, or for uploading files to a non-tabular dataset without creating a table, upload to a Volume in Unity Catalog.



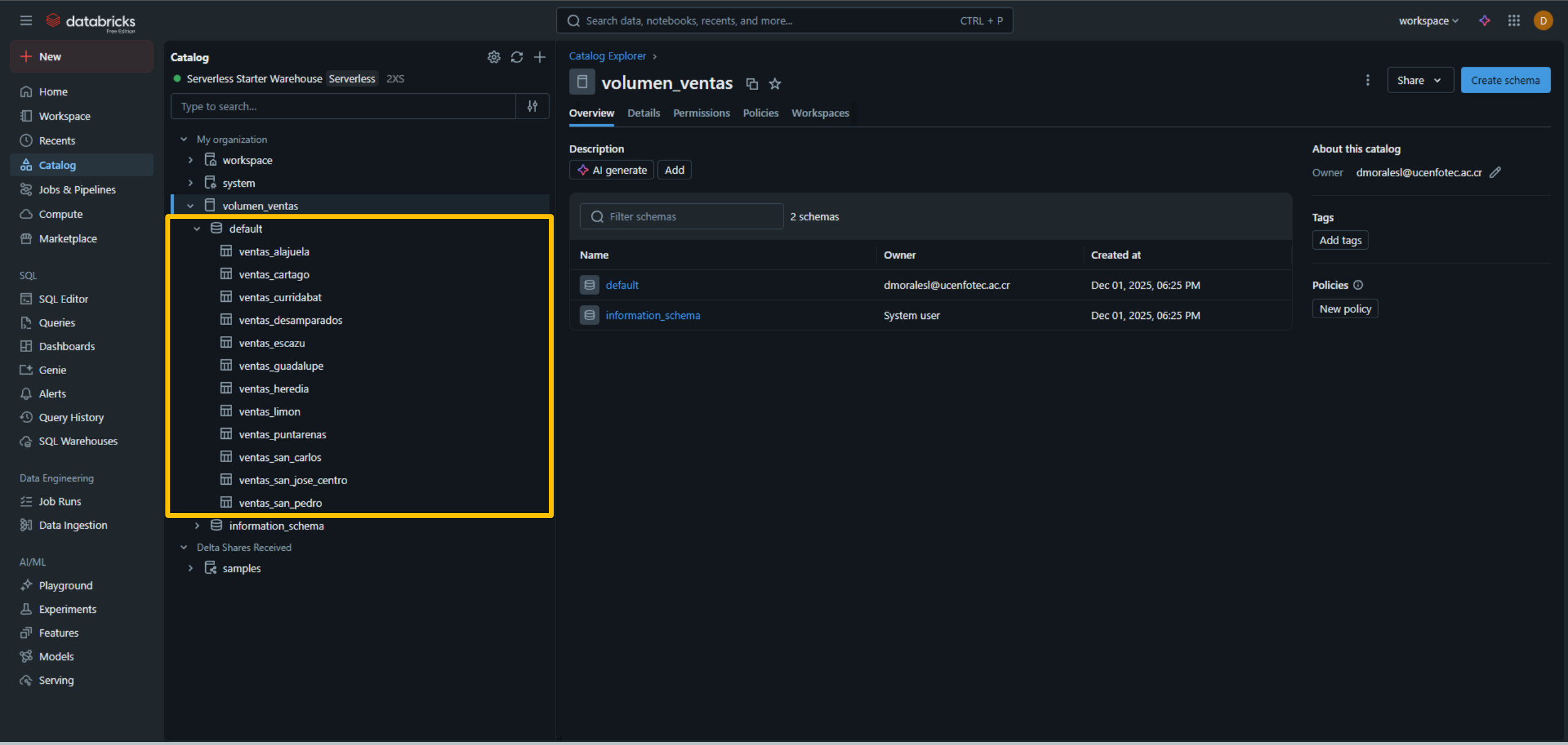
Nota Técnica: Carga de CSV

Se cargan individualmente los archivos CSV al catalog porque cada sucursal genera su propio archivo de ventas. Esto asegura trazabilidad, validación de estructura y control de calidad antes de la integración en el pipeline. Si se cargaran todos de una vez, se perdería la capacidad de identificar errores por sucursal y podrían mezclarse datos inconsistentes, dificultando la depuración y el análisis posterior

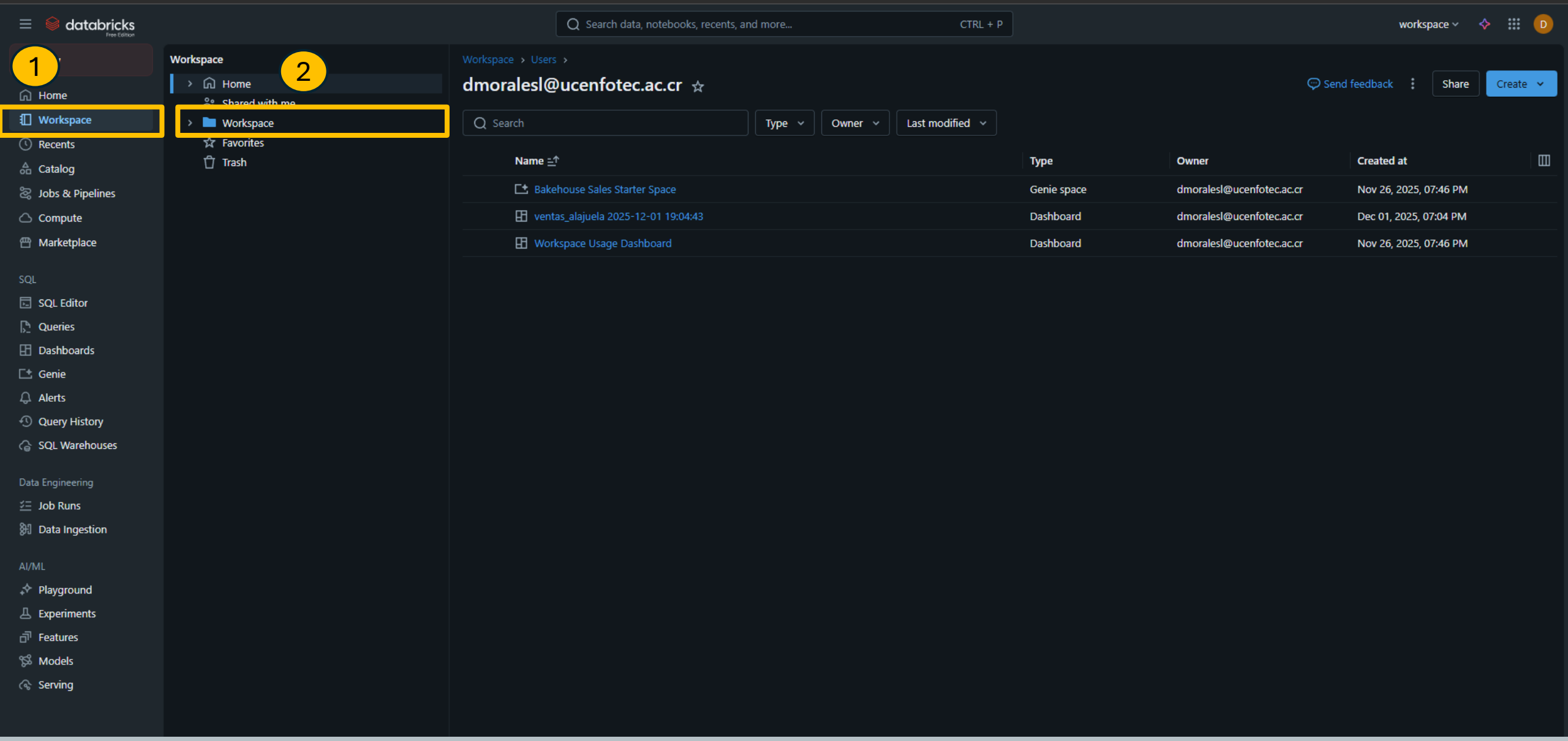
Ahora cargaremos cada CSV por separado en el Catalog



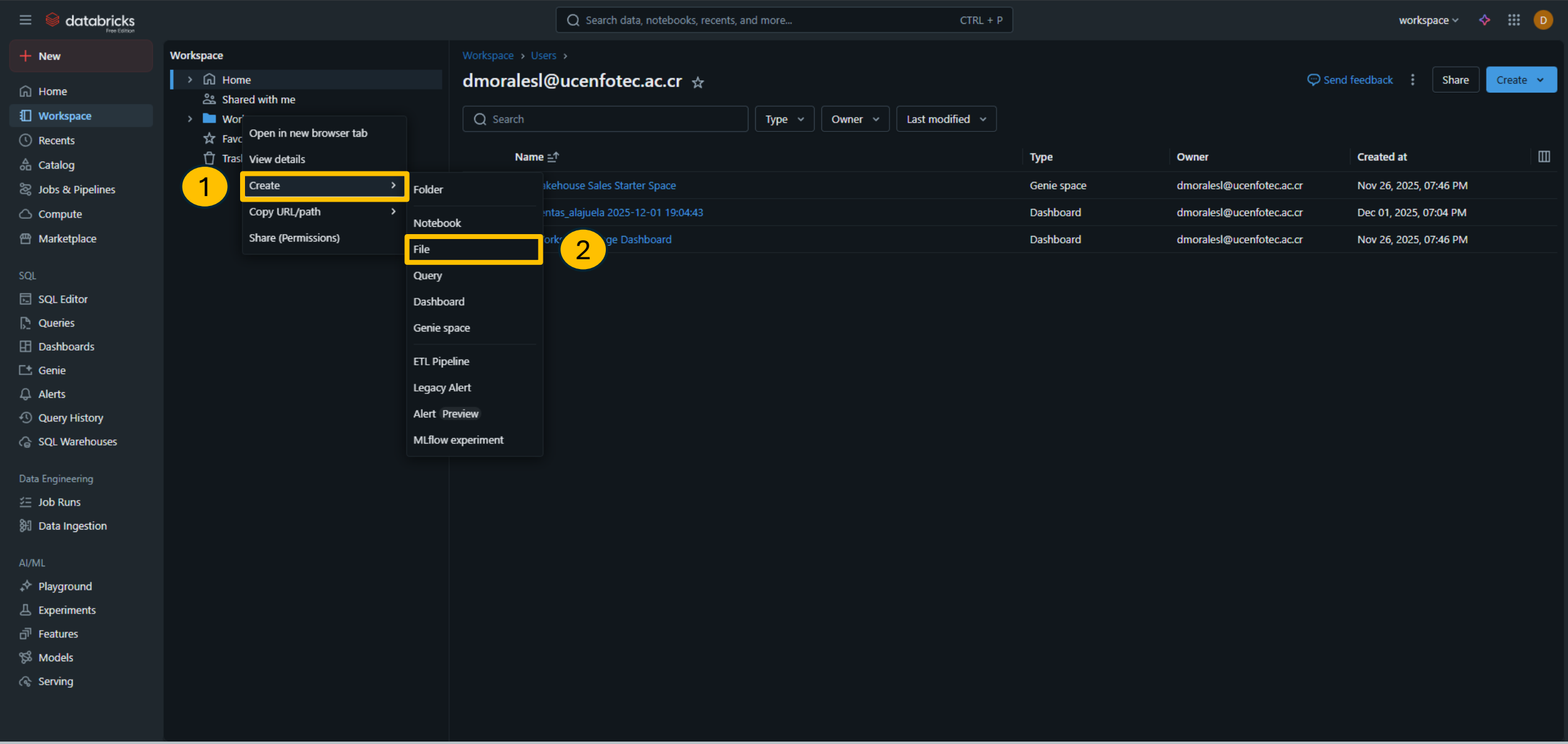
Hacemos el mismo paso con cada uno de los CSV de la Empresa

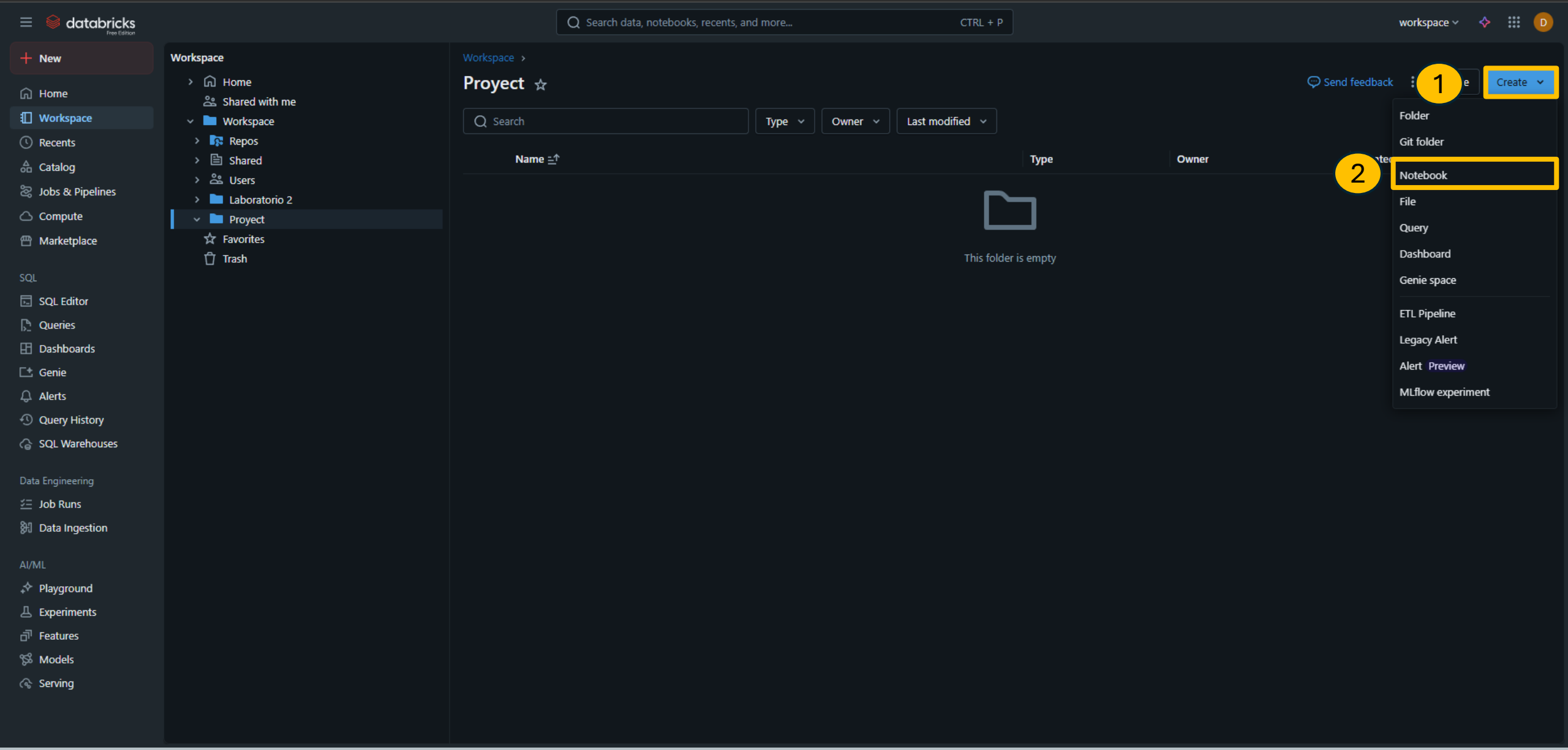


Ya tendríamos cargados todos los CSV en el Catalog



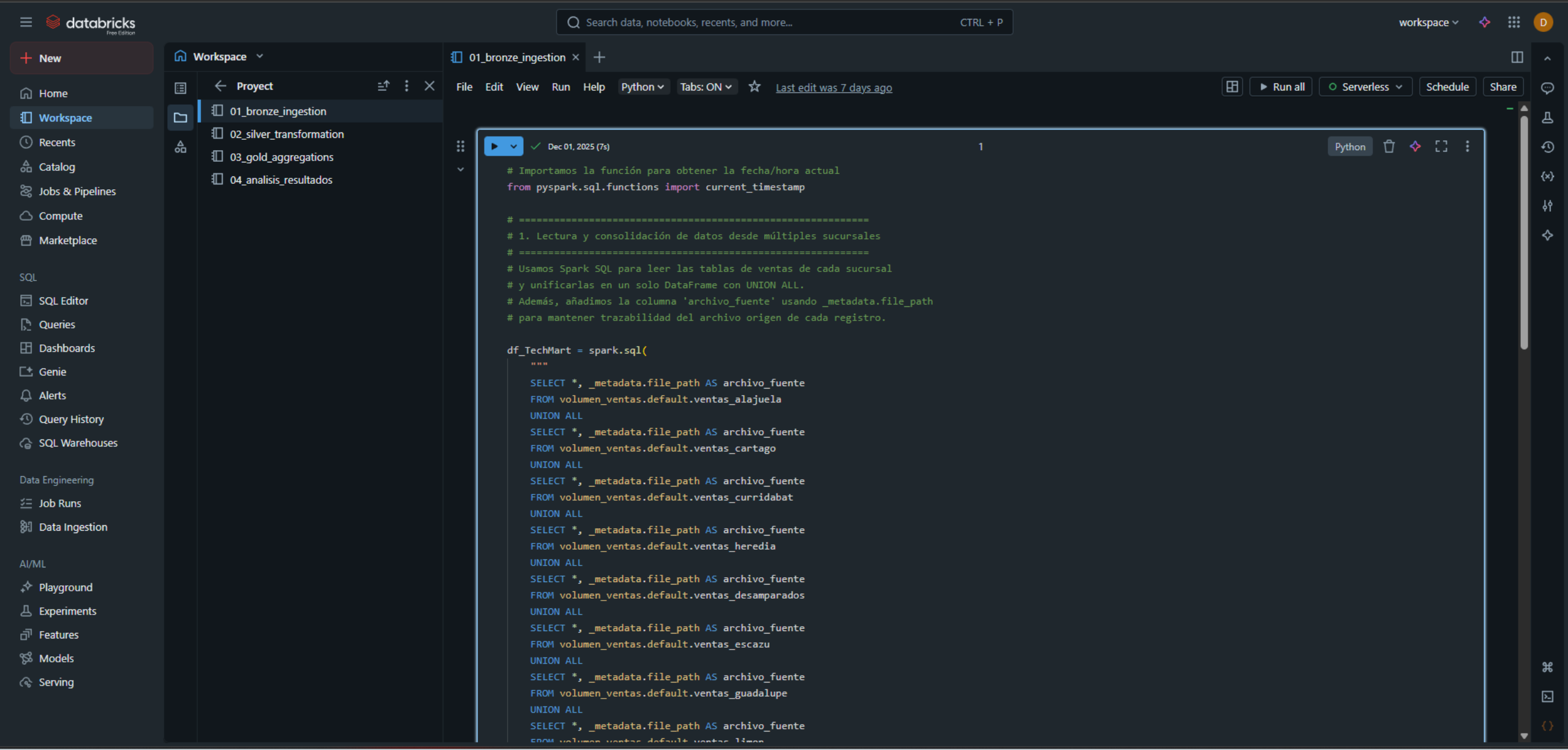
Ahora procederemos a crear un folder para guardar nuestros notebooks


databricks
Search data, notebooks, recents, and more... CTRL + P
workspace
New
Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace
SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses
Data Engineering
Job Runs
Data Ingestion
AI/ML
Playground
Experiments
Features
Models
Serving
Workspace
Home
Shared with me
Open in new browser tab
View details
Create
Copy URL/path
Share (Permissions)
1
Folder
Notebook
File
Query
Dashboard
Genie space
ETL Pipeline
Legacy Alert
Alert Preview
MLflow experiment
2
Workspace › Users ›
dmoralesl@ucenfotec.ac.cr
Send feedback
Share
Create
Search
Type
Owner
Last modified
Name
Type
Owner
Created at
kehouse Sales Starter Space
Genie space
dmoralesl@ucenfotec.ac.cr
Nov 26, 2025, 07:46 PM
ntas_alajuela 2025-12-01 19:04:43
Dashboard
dmoralesl@ucenfotec.ac.cr
Dec 01, 2025, 07:04 PM
ge Dashboard
Dashboard
dmoralesl@ucenfotec.ac.cr
Nov 26, 2025, 07:46 PM

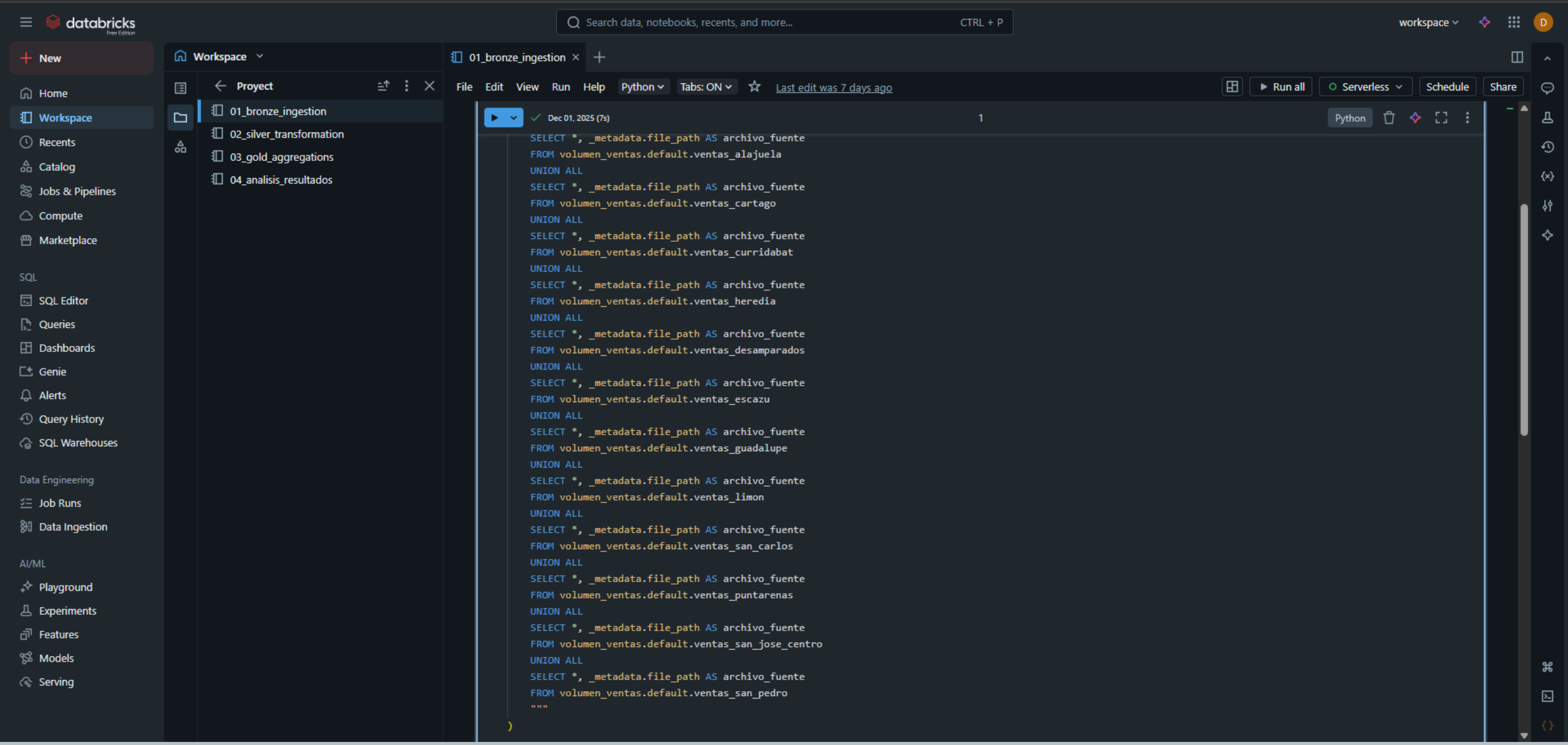


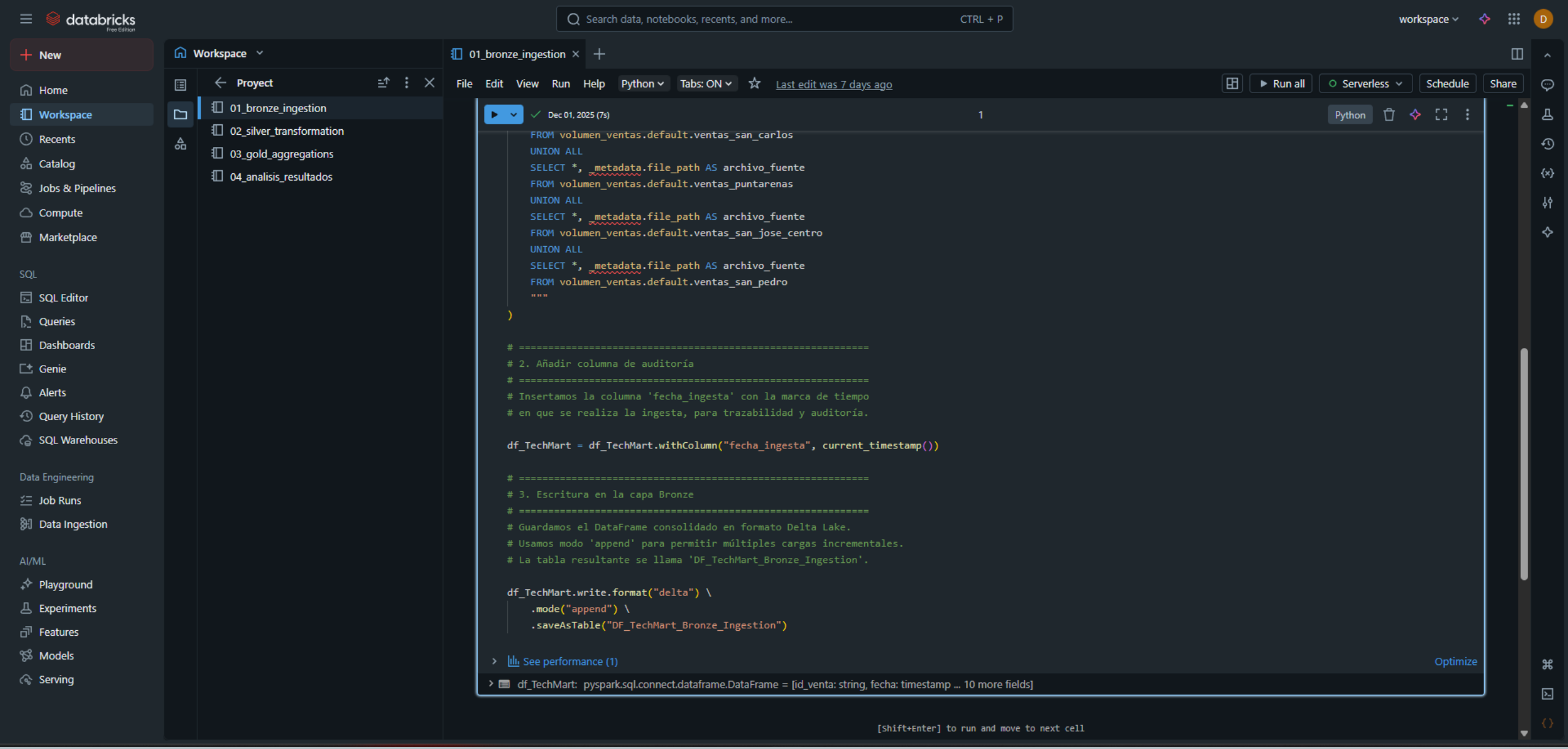
Ya cuando creamos nuestro folder empezaremos a crear nuestros Notebooks, empezamos primero por el Notebook de la etapa Bronze

Etapa Bronze

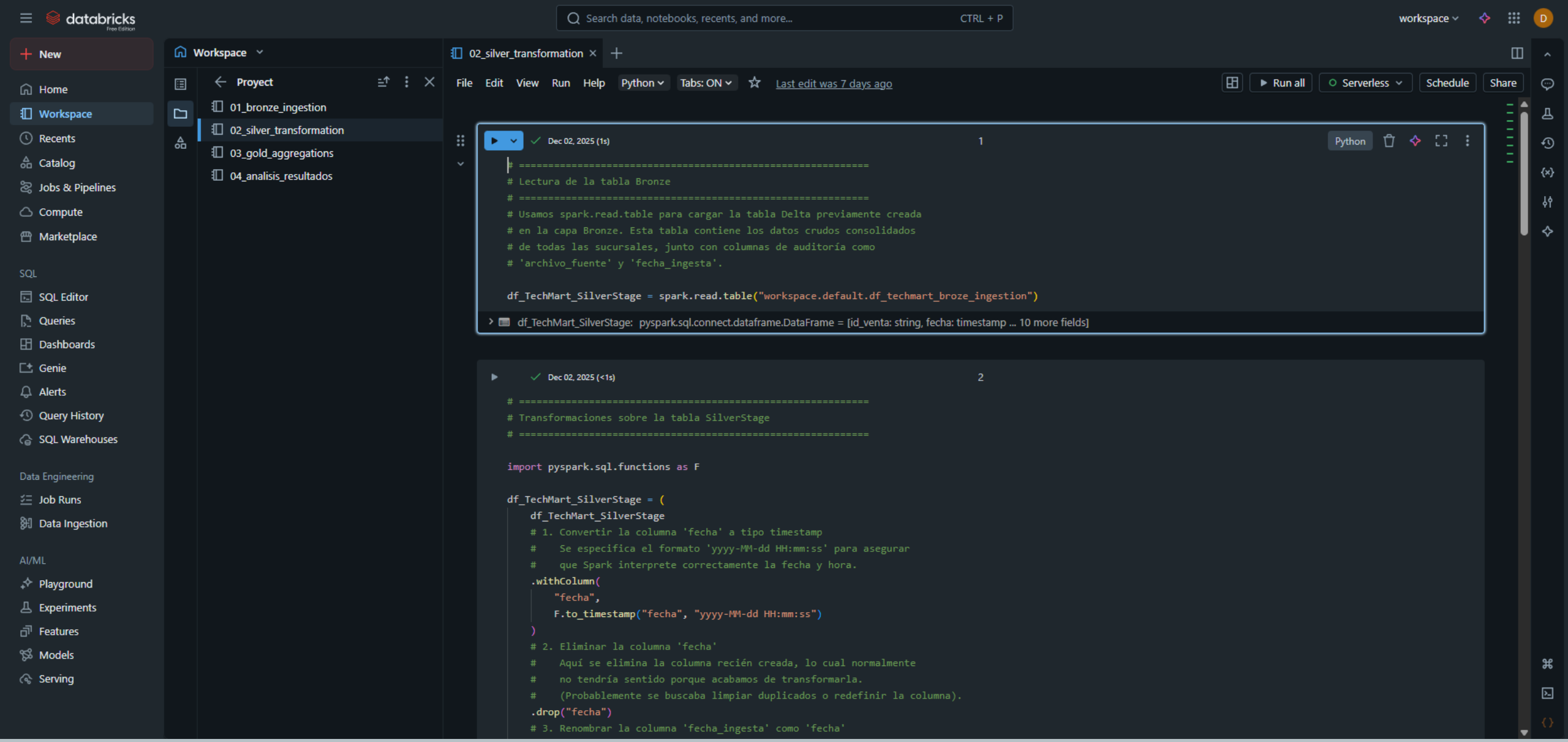


Código de la etapa Bronze (En los comentarios del código se explica el código)

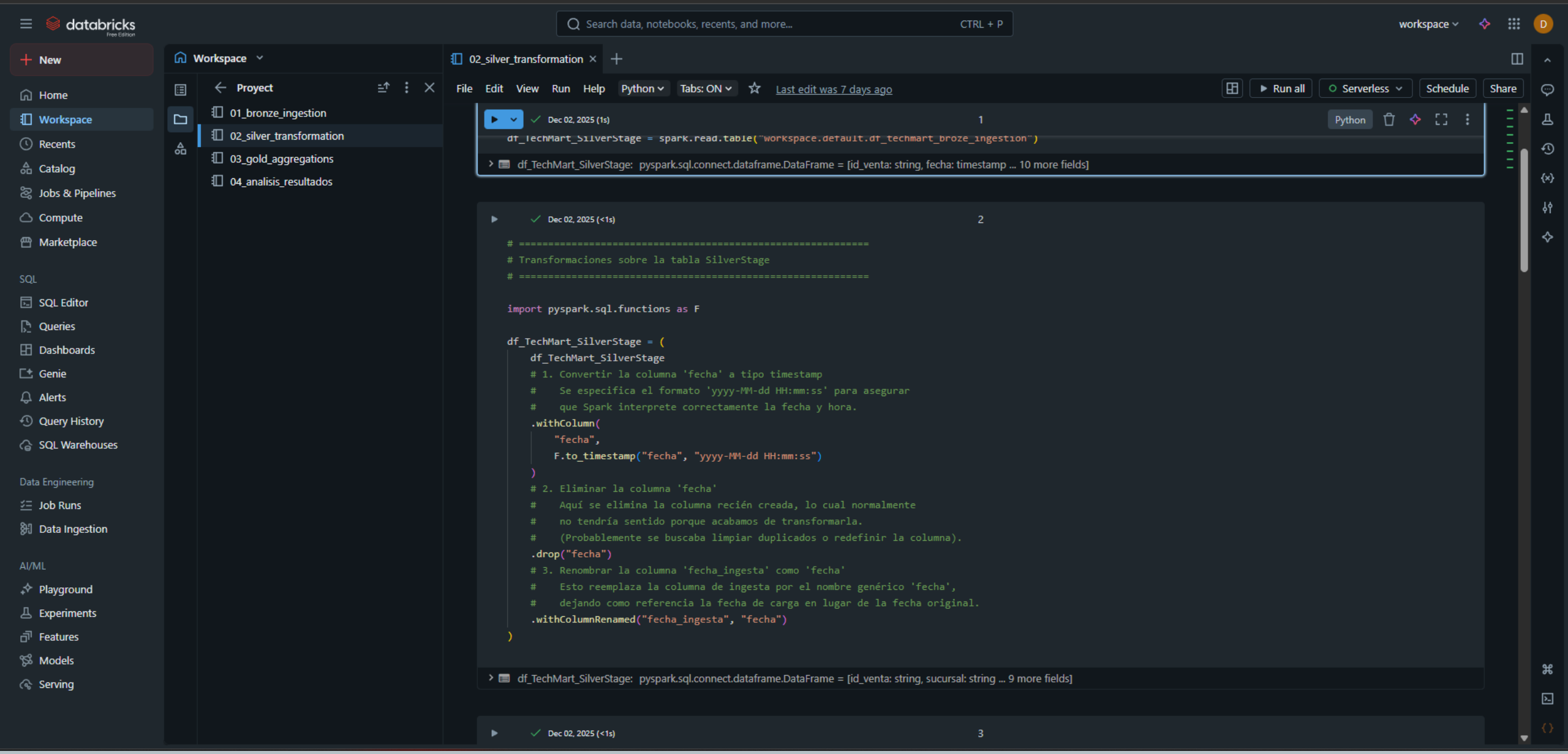

databricks
Free Edition
Search data, notebooks, recents, and more...
CTRL + P
workspace
New
Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace
SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses
Data Engineering
Job Runs
Data Ingestion
AI/ML
Playground
Experiments
Features
Models
Serving
Workspace
Proyect
01_bronze_ingestion
02_silver_transformation
03_gold_aggregations
04_analisis_resultados
01_bronze_ingestion
File Edit View Run Help Python Tabs: ON
Last edit was 7 days ago
Run all
Serverless
Schedule
Share
Dec 01, 2025 (7s)
1
Python
SELECT *, _metadata.file_path AS archivo_fuente
FROM volumen_ventas.default.ventas_alajuela
UNION ALL
SELECT *, _metadata.file_path AS archivo_fuente
FROM volumen_ventas.default.ventas_cartago
UNION ALL
SELECT *, _metadata.file_path AS archivo_fuente
FROM volumen_ventas.default.ventas_curridabat
UNION ALL
SELECT *, _metadata.file_path AS archivo_fuente
FROM volumen_ventas.default.ventas_heredia
UNION ALL
SELECT *, _metadata.file_path AS archivo_fuente
FROM volumen_ventas.default.ventas_desamparados
UNION ALL
SELECT *, _metadata.file_path AS archivo_fuente
FROM volumen_ventas.default.ventas_escazu
UNION ALL
SELECT *, _metadata.file_path AS archivo_fuente
FROM volumen_ventas.default.ventas_guadalupe
UNION ALL
SELECT *, _metadata.file_path AS archivo_fuente
FROM volumen_ventas.default.ventas_limon
UNION ALL
SELECT *, _metadata.file_path AS archivo_fuente
FROM volumen_ventas.default.ventas_san_carlos
UNION ALL
SELECT *, _metadata.file_path AS archivo_fuente
FROM volumen_ventas.default.ventas_puntarenas
UNION ALL
SELECT *, _metadata.file_path AS archivo_fuente
FROM volumen_ventas.default.ventas_san_jose_centro
UNION ALL
SELECT *, _metadata.file_path AS archivo_fuente
FROM volumen_ventas.default.ventas_san_pedro
"""
)


databricks
Free Edition
Search data, notebooks, recents, and more... CTRL + P
workspace
New
Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace
SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses
Data Engineering
Job Runs
Data Ingestion
AI/ML
Playground
Experiments
Features
Models
Serving
Workspace
Proyect
01_bronze_ingestion
02_silver_transformation
03_gold_aggregations
04_analisis_resultados
01_bronze_ingestion
File Edit View Run Help Python Tabs: ON Last edit was 7 days ago
Run all Serverless Schedule Share
Dec 01, 2025 (7s)
1
Python
FROM volumen_ventas.default.ventas_san_carlos
UNION ALL
SELECT *, _metadata.file_path AS archivo_fuente
FROM volumen_ventas.default.ventas_puntarenas
UNION ALL
SELECT *, _metadata.file_path AS archivo_fuente
FROM volumen_ventas.default.ventas_san_jose_centro
UNION ALL
SELECT *, _metadata.file_path AS archivo_fuente
FROM volumen_ventas.default.ventas_san_pedro
"""
)
# ==
# 2. Añadir columna de auditoría
# ==
# Insertamos la columna 'fecha_ingesta' con la marca de tiempo
# en que se realiza la ingesta, para trazabilidad y auditoría.
df_TechMart = df_TechMart.withColumn("fecha_ingesta", current_timestamp())
# ==
# 3. Escritura en la capa Bronze
# ==
# Guardamos el DataFrame consolidado en formato Delta Lake.
# Usamos modo 'append' para permitir múltiples cargas incrementales.
# La tabla resultante se llama 'DF_TechMart_Bronze_Ingestion'.
df_TechMart.write.format("delta") \
.mode("append") \
.saveAsTable("DF_TechMart_Bronze_Ingestion")
See performance (1)
Optimize
df_TechMart: pyspark.sql.connect.dataframe.DataFrame = [id_venta: string, fecha: timestamp ... 10 more fields]
[Shift+Enter] to run and move to next cell

Etapa Silver



Ahora creamos el notebook de la etapa Silver, este es el código de la etapa Silver (En los comentarios del código se explica el código)


databricks
Search data, notebooks, recents, and more... CTRL + P
workspace
New
Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace
SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses
Data Engineering
Job Runs
Data Ingestion
AI/ML
Playground
Experiments
Features
Models
Serving
Workspace
Proyect
01_bronze_ingestion
02_silver_transformation
03_gold_aggregations
04_analisis_resultados
02_silver_transformation
File Edit View Run Help Python Tabs: ON Last edit was 7 days ago
Run all Serverless Schedule Share
Dec 02, 2025 (1s) 1
Python
df_TechMart_SilverStage = spark.read.table("workspace.default.df_techmart_broze_ingestion")
df_TechMart_SilverStage: pyspark.sql.connect.dataframe.DataFrame = [id_venta: string, fecha: timestamp ... 10 more fields]
Dec 02, 2025 (<1s) 2
# ==
# Transformaciones sobre la tabla SilverStage
# ==
import pyspark.sql.functions as F
df_TechMart_SilverStage = (
df_TechMart_SilverStage
1. Convertir la columna 'fecha' a tipo timestamp
Se especifica el formato 'yyyy-MM-dd HH:mm:ss' para asegurar
que Spark interprete correctamente la fecha y hora.
.withColumn(
"fecha",
F.to_timestamp("fecha", "yyyy-MM-dd HH:mm:ss")
)
# 2. Eliminar la columna 'fecha'
# Aquí se elimina la columna recién creada, lo cual normalmente
# no tendría sentido porque acabamos de transformarla.
# (Probablemente se buscaba limpiar duplicados o redefinir la columna).
.drop("fecha")
# 3. Renombrar la columna 'fecha_ingesta' como 'fecha'
# Esto reemplaza la columna de ingesta por el nombre genérico 'fecha',
# dejando como referencia la fecha de carga en lugar de la fecha original.
.withColumnRenamed("fecha_ingesta", "fecha")
)
df_TechMart_SilverStage: pyspark.sql.connect.dataframe.DataFrame = [id_venta: string, sucursal: string ... 9 more fields]
Dec 02, 2025 (<1s) 3

databricks
Free Edition

Search data, notebooks, recents, and more... CTRL + P

workspace

New

Workspace

Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace

SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses

Data Engineering
Job Runs
Data Ingestion

AI/ML
Playground
Experiments
Features
Models
Serving

Workspace

Proyect
01_bronze_ingestion
02_silver_transformation
03_gold_aggregations
04_analisis_resultados

02_silver_transformation

File Edit View Run Help Python Tabs: ON Last edit was 7 days ago

Run all Serverless Schedule Share

Dec 02, 2025 (<1s) 2

```
    .drop("fecha")
    # 3. Renombrar la columna 'fecha_ingesta' como 'fecha'
    #    Esto reemplaza la columna de ingesta por el nombre genérico 'fecha',
    #    dejando como referencia la fecha de carga en lugar de la fecha original.
    .withColumnRenamed("fecha_ingesta", "fecha")
)
```

df_TechMart_SilverStage: pyspark.sql.connect.dataframe.DataFrame = [id_venta: string, sucursal: string ... 9 more fields]

Dec 02, 2025 (<1s) 3

```
# ============================================================
# Derivación de atributos de fecha en la capa SilverStage
# ============================================================

df_TechMart_SilverStage = (
    df_TechMart_SilverStage
    # 1. Extraer el año desde la columna 'fecha'
    .withColumn("año", F.year("fecha"))
    # 2. Extraer el mes (valor numérico 1-12)
    .withColumn("mes", F.month("fecha"))
    # 3. Extraer el día del mes (1-31)
    .withColumn("dia", F.dayofmonth("fecha"))
    # 4. Obtener el nombre completo del día de la semana (ej. Monday, Tuesday)
    .withColumn("dia_semana", F.date_format("fecha", "EEEE"))
)
```

df_TechMart_SilverStage: pyspark.sql.connect.dataframe.DataFrame = [id_venta: string, sucursal: string ... 13 more fields]

Dec 02, 2025 (<1s) 4

```
# ============================================================
# Normalización de texto en la capa SilverStage
```

New
Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace

SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses

Data Engineering
Job Runs
Data Ingestion

AI/ML
Playground
Experiments
Features
Models
Serving

Workspace
Proyect
01_bronze_ingestion
02_silver_transformation
03_gold_aggregations
04_analisis_resultados

02_silver_transformation

File Edit View Run Help Python Tabs: ON Last edit was 7 days ago

Run all Serverless Schedule Share

Dec 02, 2025 (<1s) 3

df_TechMart_SilverStage: pyspark.sql.connect.dataframe.DataFrame = [id_venta: string, sucursal: string ... 13 more fields]

Dec 02, 2025 (<1s) 4

```
# ============================================================
# Normalización de texto en la capa SilverStage
# ============================================================

df_TechMart_SilverStage = (
    df_TechMart_SilverStage
    # 1. Convertir la columna 'sucursal' a mayúsculas
    #    Esto asegura consistencia en los nombres de sucursal,
    #    evitando problemas de comparación por diferencias de formato.
    .withColumn("sucursal", F.upper(F.col("sucursal")))

    # 2. Convertir la columna 'producto' a minúsculas
    #    Se estandariza el nombre de los productos para facilitar
    #    búsquedas, agrupaciones y evitar duplicados por diferencias
    #    de capitalización (ej. "Laptop" vs "laptop").
    .withColumn("producto", F.lower(F.col("producto")))
)
```

df_TechMart_SilverStage: pyspark.sql.connect.dataframe.DataFrame = [id_venta: string, sucursal: string ... 13 more fields]

Dec 02, 2025 (<1s) 5

```
# ============================================================
# Eliminación de duplicados en la capa SilverStage
# ============================================================

# Usamos dropDuplicates indicando la columna 'id_venta' como clave única.
# Esto asegura que, si existen múltiples registros con el mismo id_venta,
# solo se conserve uno y se eliminen los duplicados.
# Es una práctica común en la capa Silver para garantizar
# la integridad y consistencia de los datos antes de pasar a la capa Gold.
```

databricks

Search data, notebooks, recents, and more... CTRL + P

workspace

New

Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace

SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses

Data Engineering
Job Runs
Data Ingestion

AI/ML
Playground
Experiments
Features
Models
Serving

Workspace

Proyect
01_bronze_ingestion
02_silver_transformation
03_gold_aggregations
04_analisis_resultados

02_silver_transformation

File Edit View Run Help Python Tabs: ON Last edit was 7 days ago

Run all Serverless Schedule Share

Dec 02, 2025 (<1s) 4

```
    .withColumn("producto", F.lower(F.col("producto")))
)
```

df_TechMart_SilverStage: pyspark.sql.connect.dataframe.DataFrame = [id_venta: string, sucursal: string ... 13 more fields]

Dec 02, 2025 (<1s) 5

```
# ============================================================
# Eliminación de duplicados en la capa SilverStage
# ============================================================

# Usamos dropDuplicates indicando la columna 'id_venta' como clave única.
# Esto asegura que, si existen múltiples registros con el mismo id_venta,
# solo se conserve uno y se eliminen los duplicados.
# Es una práctica común en la capa Silver para garantizar
# la integridad y consistencia de los datos antes de pasar a la capa Gold.

df_TechMart_SilverStage = df_TechMart_SilverStage.dropDuplicates(["id_venta"])
```

df_TechMart_SilverStage: pyspark.sql.connect.dataframe.DataFrame = [id_venta: string, sucursal: string ... 13 more fields]

Dec 02, 2025 (<1s) 6

```
# ============================================================
# Filtrado de registros inválidos en la capa SilverStage
# ============================================================

# Usamos filter para conservar únicamente las filas donde la columna 'total' > 0.
# Esto elimina registros con valores nulos, cero o negativos en el campo 'total',
# garantizando que solo se mantengan ventas válidas en el DataFrame SilverStage.

df_TechMart_SilverStage = df_TechMart_SilverStage.filter(F.col("total") > 0)
```

df_TechMart_SilverStage: pyspark.sql.connect.dataframe.DataFrame = [id_venta: string, sucursal: string ... 13 more fields]

databricks
Free Edition

Search data, notebooks, recents, and more... CTRL + P

workspace

New

Workspace

Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace

SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses

Data Engineering
Job Runs
Data Ingestion

AI/ML
Playground
Experiments
Features
Models
Serving

Workspace

Proyect
01_bronze_ingestion
02_silver_transformation
03_gold_aggregations
04_analisis_resultados

02_silver_transformation

File Edit View Run Help Python Tabs: ON Last edit was 7 days ago

Run all Serverless Schedule Share

Dec 02, 2025 (<1s) 6

df_TechMart_SilverStage: pyspark.sql.connect.dataframe.DataFrame = [id_venta: string, sucursal: string ... 13 more fields]

Dec 02, 2025 (<1s) 7

```
# ============================================================
# Validación de campos obligatorios en la capa SilverStage
# ============================================================

# Definimos una lista con los nombres de los campos que deben ser obligatorios.
# Estos campos no pueden contener valores nulos, ya que son críticos para el análisis:
# - id_venta   → identificador único de la transacción
# - fecha      → fecha de la venta
# - sucursal   → sucursal donde ocurrió la venta
# - producto   → producto vendido
# - total      → monto total de la transacción

campos_obligatorios = ["id_venta", "fecha", "sucursal", "producto", "total"]

# Iteramos sobre cada campo obligatorio y aplicamos un filtro
# para eliminar las filas donde ese campo sea nulo.
# De esta forma, garantizamos que el DataFrame SilverStage
# solo contenga registros válidos y completos.
for campo in campos_obligatorios:
    df_TechMart_SilverStage = df_TechMart_SilverStage.filter(F.col(campo).isNotNull())
```

df_TechMart_SilverStage: pyspark.sql.connect.dataframe.DataFrame = [id_venta: string, sucursal: string ... 13 more fields]

Dec 02, 2025 (12s) 8

```
# ============================================================
# Escritura de la tabla Silver particionada
# ============================================================

# Guardamos el DataFrame SilverStage en formato Delta Lake.
# Se utiliza 'append' para permitir cargas incrementales sin sobrescribir datos previos.
```

New
Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace

SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses

Data Engineering
Job Runs
Data Ingestion

AI/ML
Playground
Experiments
Features
Models
Serving

Workspace
Proyect
01_bronze_ingestion
02_silver_transformation
03_gold_aggregations
04_analisis_resultados

02_silver_transformation

File Edit View Run Help Python Tabs: ON Last edit was 7 days ago

Run all Serverless Schedule Share

Dec 02, 2025 (<1s) 7

```
df_TechMart_SilverStage: pyspark.sql.connect.dataframe.DataFrame = [id_venta: string, sucursal: string ... 13 more fields]
```

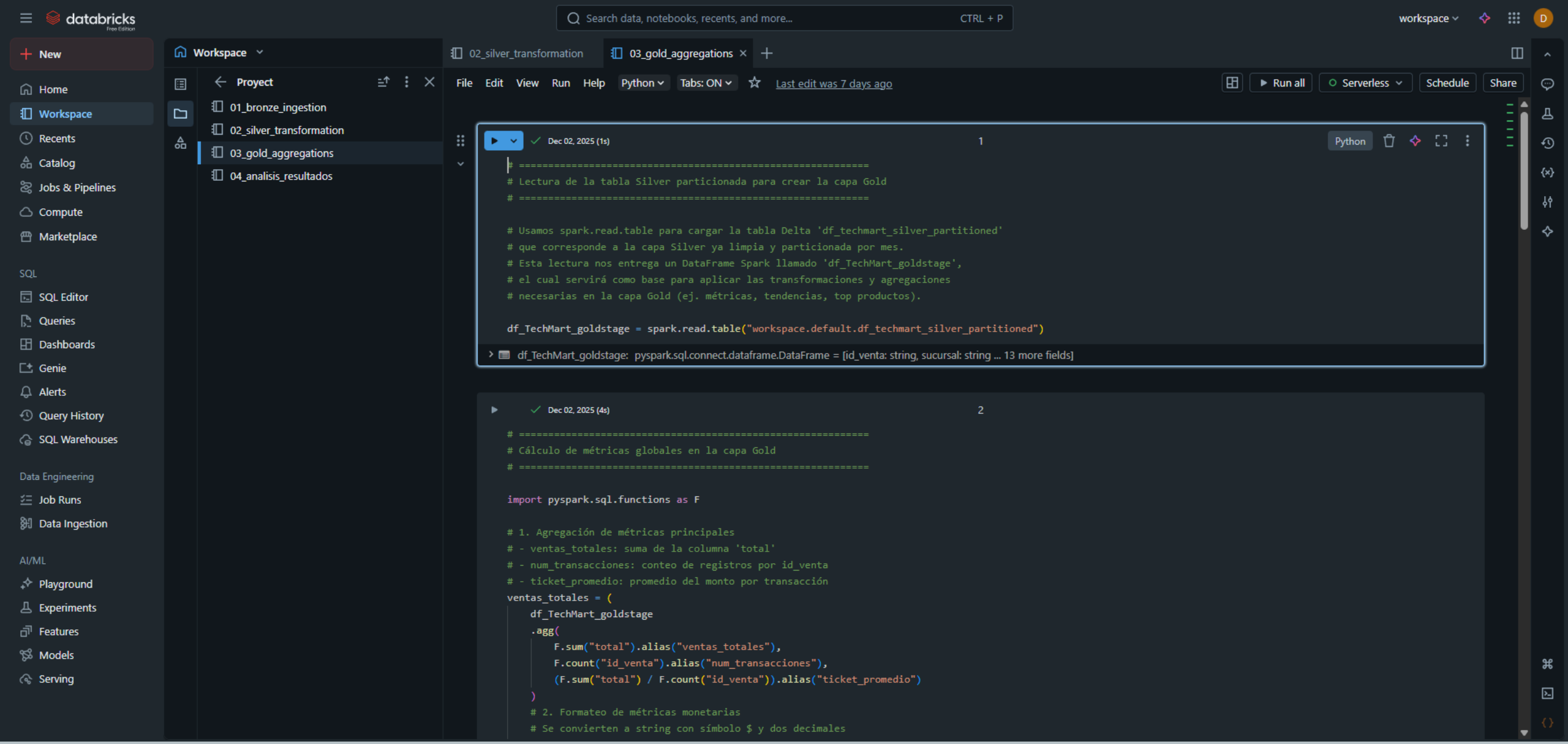
Dec 02, 2025 (12s) 8

```
# ============================================================
# Escritura de la tabla Silver particionada
# ============================================================

# Guardamos el DataFrame SilverStage en formato Delta Lake.
# Se utiliza 'append' para permitir cargas incrementales sin sobrescribir datos previos.
# La partición se realiza por la columna 'mes', lo que optimiza las consultas
# que filtran o agrupan por mes, ya que Spark puede leer solo las particiones necesarias.
# Finalmente, se registra la tabla en el catálogo con el nombre 'df_techmart_silver_partitioned'.

df_TechMart_SilverStage.write.format("delta") \
    .mode("append") \
    .partitionBy("mes") \
    .saveAsTable("df_techmart_silver_partitioned")
```

See performance (1) Optimize

[Shift+Enter] to run and move to next cell
[Ctrl+Shift+P] to open the command palette
[Esc H] to see all keyboard shortcuts

Etapa Gold



Ahora creamos el notebook de la etapa Gold, este es el código de la etapa Gold (En los comentarios del código se explica el código)

databricks
Free Edition

Search data, notebooks, recents, and more... CTRL + P

workspace

New

Workspace

Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace

SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses

Data Engineering
Job Runs
Data Ingestion

AI/ML
Playground
Experiments
Features
Models
Serving

Workspace

02_silver_transformation | 03_gold_aggregations

Proyect
01_bronze_ingestion
02_silver_transformation
03_gold_aggregations
04_analisis_resultados

File Edit View Run Help Python Tabs: ON Last edit was 7 days ago

Run all | Serverless | Schedule | Share

df_TechMart_goldstage: pyspark.sql.connect.dataframe.DataFrame = [id_venta: string, sucursal: string ... 13 more fields]

Dec 02, 2025 (4s) 2

```
# ============================================================
# Cálculo de métricas globales en la capa Gold
# ============================================================

import pyspark.sql.functions as F

# 1. Agregación de métricas principales
# - ventas_totales: suma de la columna 'total'
# - num_transacciones: conteo de registros por id_venta
# - ticket_promedio: promedio del monto por transacción
ventas_totales = (
    df_TechMart_goldstage
    .agg(
        F.sum("total").alias("ventas_totales"),
        F.count("id_venta").alias("num_transacciones"),
        (F.sum("total") / F.count("id_venta")).alias("ticket_promedio")
    )
    # 2. Formateo de métricas monetarias
    # Se convierten a string con símbolo $ y dos decimales
    .withColumn("ventas_totales", F.format_string("$%.2f", F.col("ventas_totales")))
    .withColumn("ticket_promedio", F.format_string("$%.2f", F.col("ticket_promedio")))
)

# 3. Escritura en la capa Gold
# Guardamos la tabla en formato Delta Lake con modo 'append'
# para permitir cargas incrementales sin sobrescribir datos previos.
ventas_totales.write.format("delta").mode("append").saveAsTable("df_techmart_gold_ventas_totales")
```

See performance (1) Optimize

ventas_totales: pyspark.sql.connect.dataframe.DataFrame = [ventas_totales: string, num_transacciones: long ... 1 more field]

Dec 02, 2025 (3s) 3

databricks

Search data, notebooks, recents, and more... CTRL + P

workspace

New

Workspace

02_silver_transformation

03_gold_aggregations

Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace

SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses

Data Engineering
Job Runs
Data Ingestion

AI/ML
Playground
Experiments
Features
Models
Serving

Proyect
01_bronze_ingestion
02_silver_transformation
03_gold_aggregations
04_analisis_resultados

File Edit View Run Help Python Tabs: ON Last edit was 7 days ago

Run all Serverless Schedule Share

ventas_totales: pyspark.sql.connect.dataframe.DataFrame = [ventas_totales: string, num_transacciones: long ... 1 more field]

Dec 02, 2025 (3s) 3

```
# ============================================================
# Cálculo de métricas por sucursal en la capa Gold
# ============================================================

ventas_por_sucursal = (
    df_TechMart_goldstage
    # 1. Agrupamos por la columna 'sucursal'
    .groupBy("sucursal")
    # 2. Calculamos métricas principales por cada sucursal:
    # - ventas_totales: suma de la columna 'total'
    # - num_transacciones: conteo de registros por id_venta
    # - ticket_promedio: promedio del monto por transacción
    .agg(
        F.sum("total").alias("ventas_totales"),
        F.count("id_venta").alias("num_transacciones"),
        (F.sum("total") / F.count("id_venta")).alias("ticket_promedio")
    )
    # 3. Formateamos las métricas monetarias
    # Se convierten a string con símbolo $ y dos decimales
    .withColumn("ventas_totales", F.format_string("$%.2f", F.col("ventas_totales")))
    .withColumn("ticket_promedio", F.format_string("$%.2f", F.col("ticket_promedio")))
)

# 4. Escritura en la capa Gold
# Guardamos la tabla en formato Delta Lake con modo 'append'
# para permitir cargas incrementales sin sobrescribir datos previos.
ventas_por_sucursal.write.format("delta").mode("append").saveAsTable("df_techmart_gold_ventas_por_sucursal")
```

See performance (1) Optimize

ventas_por_sucursal: pyspark.sql.connect.dataframe.DataFrame = [sucursal: string, ventas_totales: string ... 2 more fields]

Dec 02, 2025 (4s) 4

databricks

Search data, notebooks, recents, and more... CTRL + P

workspace

New

Workspace

02_silver_transformation

03_gold_aggregations

Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace

SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses

Data Engineering
Job Runs
Data Ingestion

AI/ML
Playground
Experiments
Features
Models
Serving

Proyect
01_bronze_ingestion
02_silver_transformation
03_gold_aggregations
04_analisis_resultados

File Edit View Run Help Python Tabs: ON Last edit was 7 days ago

Run all Serverless Schedule Share

Dec 02, 2025 (3s) 3

See performance (1) Optimize

ventas_por_sucursal: pyspark.sql.connect.dataframe.DataFrame = [sucursal: string, ventas_totales: string ... 2 more fields]

Dec 02, 2025 (4s) 4

```
# ============================================================
# Cálculo del Top 10 productos más vendidos en la capa Gold
# ============================================================

top_productos = (
    df_TechMart_goldstage
    # 1. Agrupamos por la columna 'producto'
    .groupBy("producto")
    # 2. Calculamos la cantidad total vendida por producto
    .agg(F.sum("cantidad").alias("cantidad_vendida"))
    # 3. Ordenamos los productos de mayor a menor según la cantidad vendida
    .orderBy(F.desc("cantidad_vendida"))
    # 4. Limitamos el resultado a los 10 primeros productos
    .limit(10)
)

# 5. Escritura en la capa Gold
# Guardamos la tabla en formato Delta Lake con modo 'append'
# para permitir cargas incrementales sin sobrescribir datos previos.
top_productos.write.format("delta").mode("append").saveAsTable("df_techmart_gold_top_productos")
```

See performance (1) Optimize

top_productos: pyspark.sql.connect.dataframe.DataFrame = [producto: string, cantidad_vendida: long]

Dec 02, 2025 (4s) 5

```
# ============================================================
# Cálculo de métricas por categoría en la capa Gold
```

databricks
Free Edition

Search data, notebooks, recents, and more... CTRL + P

workspace

New

Workspace

02_silver_transformation

03_gold_aggregations

Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace

SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses

Data Engineering
Job Runs
Data Ingestion

AI/ML
Playground
Experiments
Features
Models
Serving

Proyect
01_bronze_ingestion
02_silver_transformation
03_gold_aggregations
04_analisis_resultados

File Edit View Run Help Python Tabs: ON Last edit was 7 days ago

Run all Serverless Schedule Share

Dec 02, 2025 (4s) 4

See performance (1) Optimize

top_productos: pyspark.sql.connect.dataframe.DataFrame = [producto: string, cantidad_vendida: long]

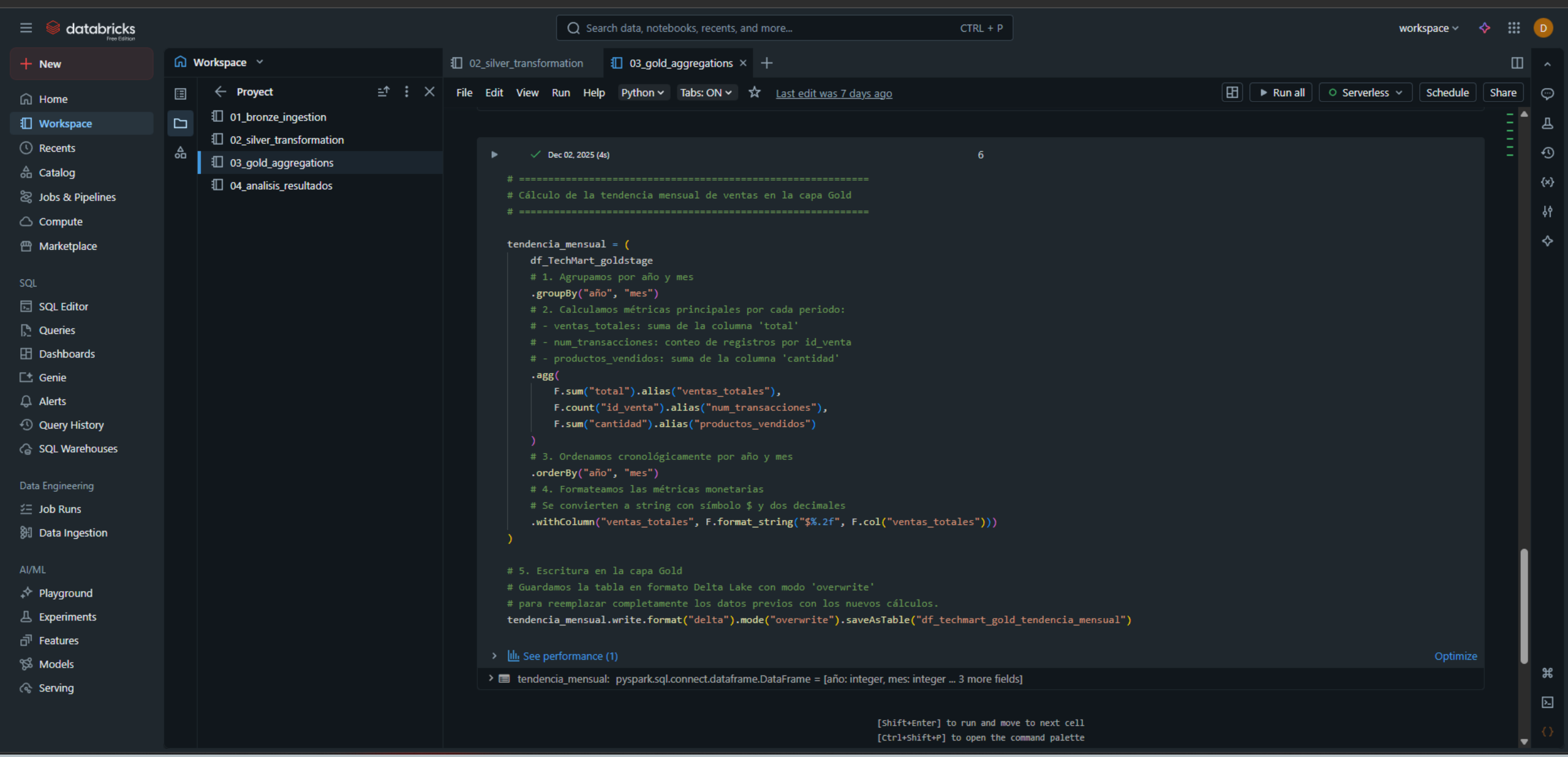
Dec 02, 2025 (4s) 5

```
# ============================================================
# Cálculo de métricas por categoría en la capa Gold
# ============================================================

ventas_por_categoria = (
    df_TechMart_goldstage
    # 1. Agrupamos por la columna 'categoria'
    .groupBy("categoria")
    # 2. Calculamos métricas principales por cada categoría:
    # - ventas_totales: suma de la columna 'total'
    # - productos_vendidos: suma de la columna 'cantidad'
    .agg(
        F.sum("total").alias("ventas_totales"),
        F.sum("cantidad").alias("productos_vendidos")
    )
    # 3. Formateamos las métricas monetarias
    # Se convierten a string con símbolo $ y dos decimales
    .withColumn("ventas_totales", F.format_string("$%.2f", F.col("ventas_totales")))
)

# 4. Escritura en la capa Gold
# Guardamos la tabla en formato Delta Lake con modo 'overwrite'
# para reemplazar completamente los datos previos con los nuevos cálculos.
ventas_por_categoria.write.format("delta").mode("overwrite").saveAsTable("df_techmart_gold_ventas_por_categoria")
```

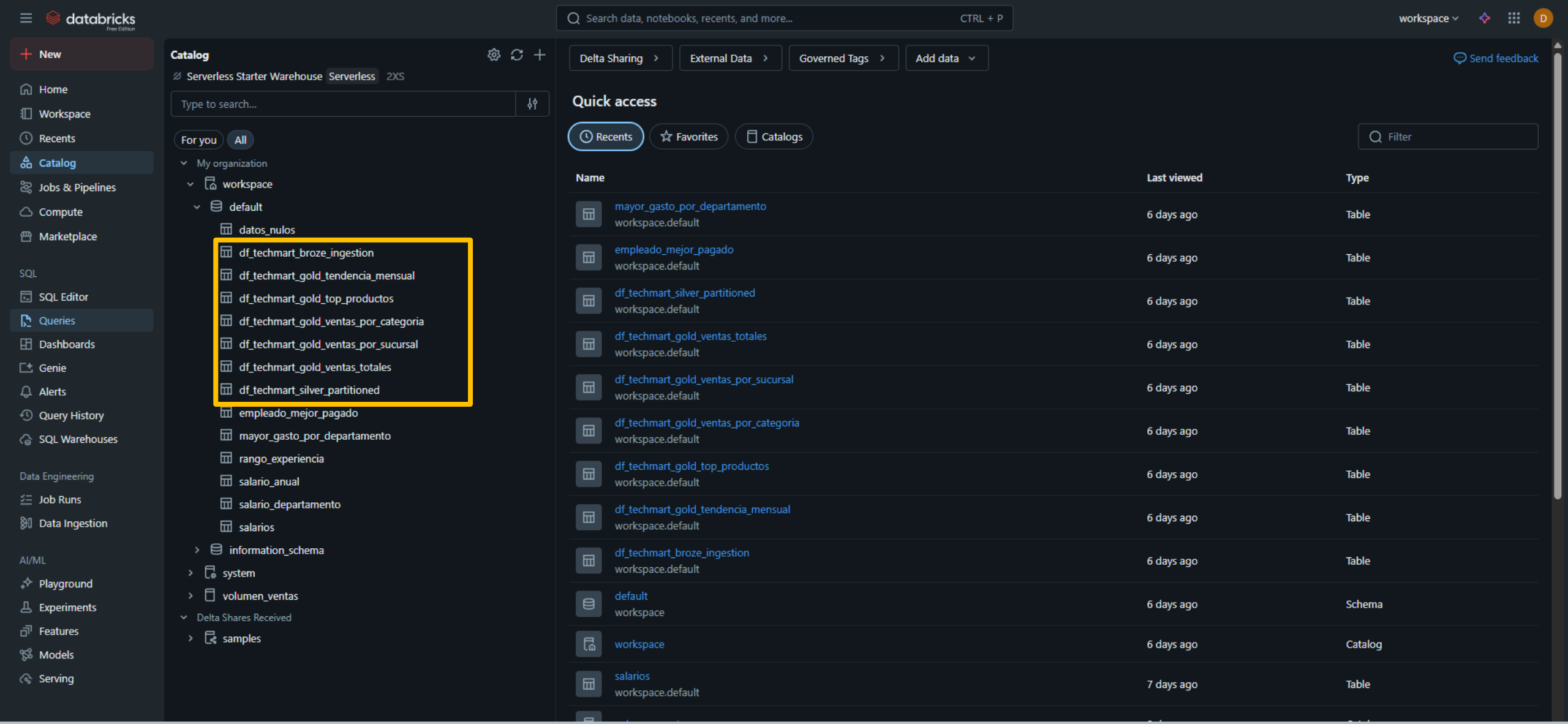
See performance (1) Optimize

ventas_por_categoria: pyspark.sql.connect.dataframe.DataFrame = [categoria: string, ventas_totales: string ... 1 more field]

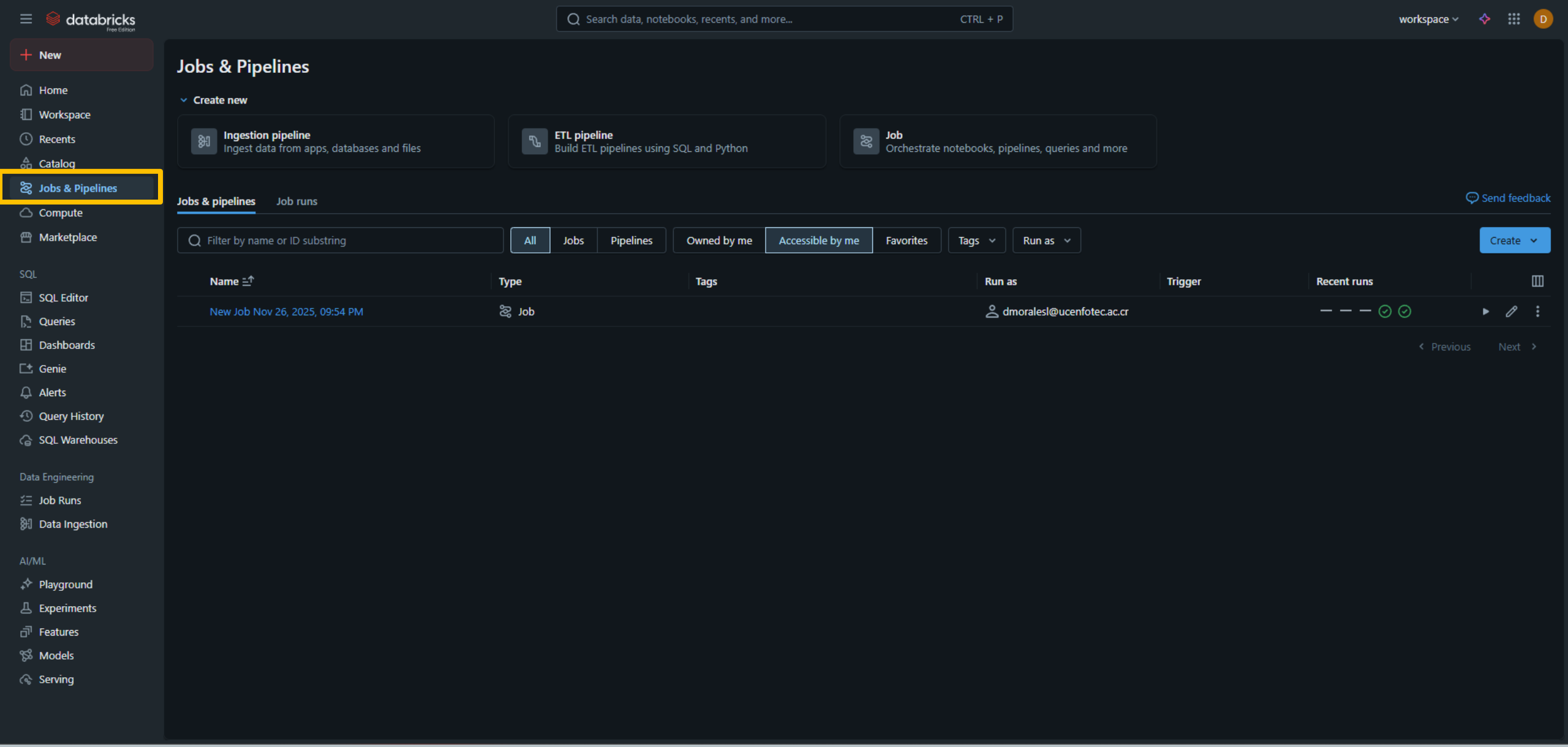
Dec 02, 2025 (4s) 6


databricks
Free Edition
Search data, notebooks, recents, and more...
CTRL + P
workspace
New
Workspace
02_silver_transformation
03_gold_aggregations
Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace
SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses
Data Engineering
Job Runs
Data Ingestion
AI/ML
Playground
Experiments
Features
Models
Serving
Proyect
01_bronze_ingestion
02_silver_transformation
03_gold_aggregations
04_analisis_resultados
File Edit View Run Help
Python
Tabs: ON
Last edit was 7 days ago
Run all
Serverless
Schedule
Share
Dec 02, 2025 (4s)
6
# ==
# Cálculo de la tendencia mensual de ventas en la capa Gold
# ==
tendencia_mensual = (
df_TechMart_goldstage
# 1. Agrupamos por año y mes
.groupBy("año", "mes")
# 2. Calculamos métricas principales por cada periodo:
# - ventas_totales: suma de la columna 'total'
# - num_transacciones: conteo de registros por id_venta
# - productos_vendidos: suma de la columna 'cantidad'
.agg(
F.sum("total").alias("ventas_totales"),
F.count("id_venta").alias("num_transacciones"),
F.sum("cantidad").alias("productos_vendidos")
)
# 3. Ordenamos cronológicamente por año y mes
.orderBy("año", "mes")
# 4. Formateamos las métricas monetarias
# Se convierten a string con símbolo $ y dos decimales
.withColumn("ventas_totales", F.format_string("$%.2f", F.col("ventas_totales")))
)
# 5. Escritura en la capa Gold
# Guardamos la tabla en formato Delta Lake con modo 'overwrite'
# para reemplazar completamente los datos previos con los nuevos cálculos.
tendencia_mensual.write.format("delta").mode("overwrite").saveAsTable("df_techmart_gold_tendencia_mensual")
See performance (1)
Optimize
tendencia_mensual: pyspark.sql.connect.dataframe.DataFrame = [año: integer, mes: integer ... 3 more fields]
[Shift+Enter] to run and move to next cell
[Ctrl+Shift+P] to open the command palette

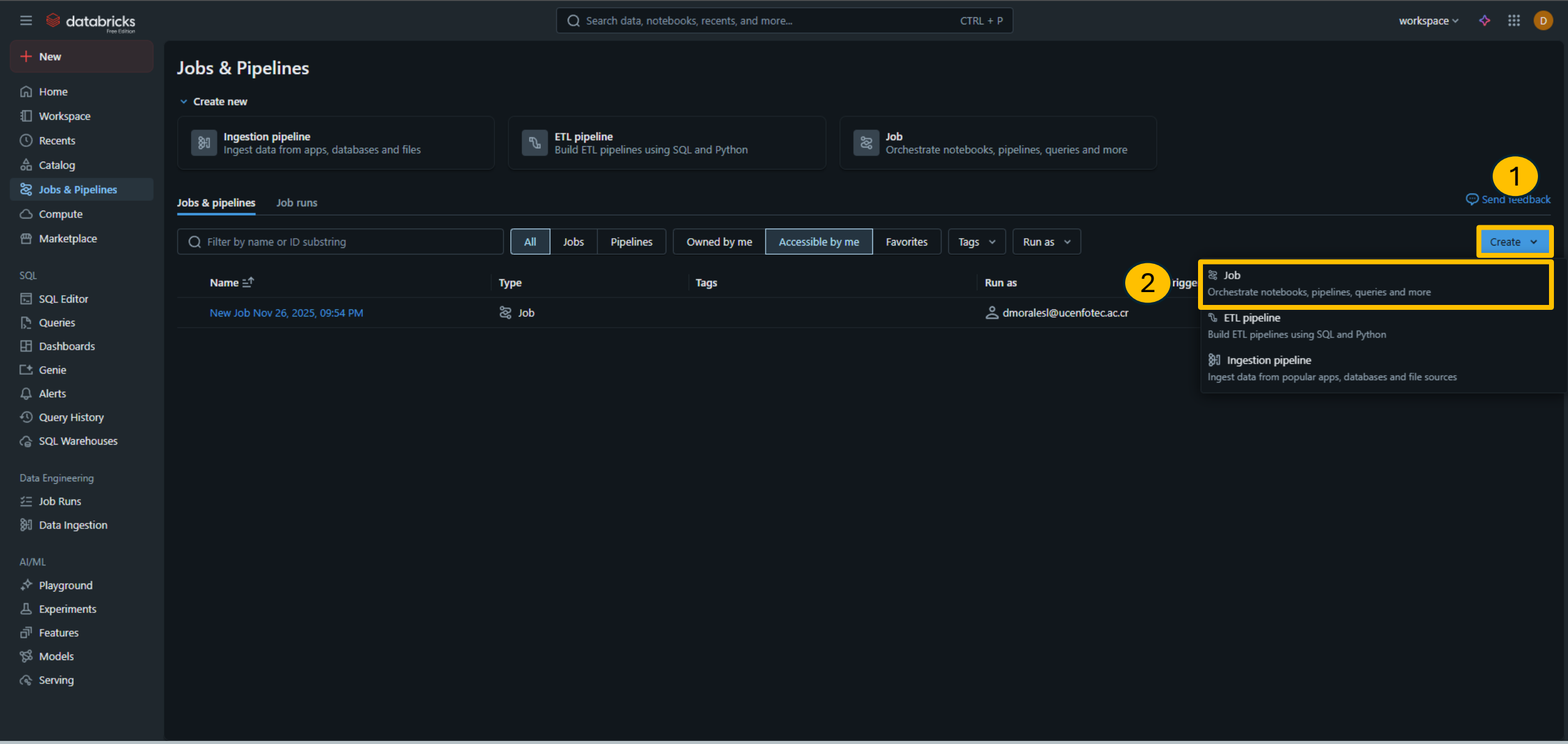
Creación y Ejecución del Job



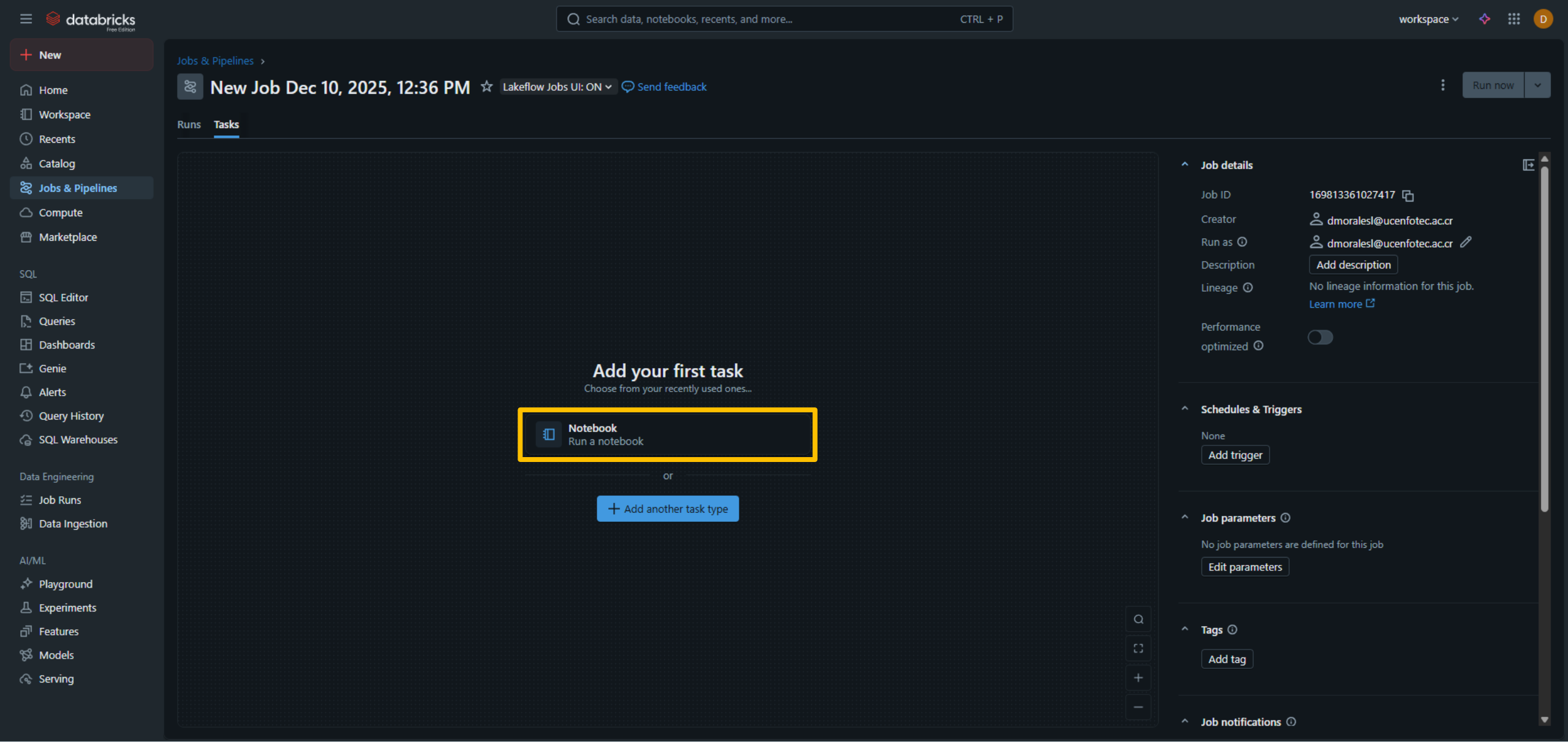
Una vez cargadas todas las etapas en formato Delta Lake dentro del catalog, el siguiente paso corresponde a la creación y ejecución del Job que orquesta el pipeline

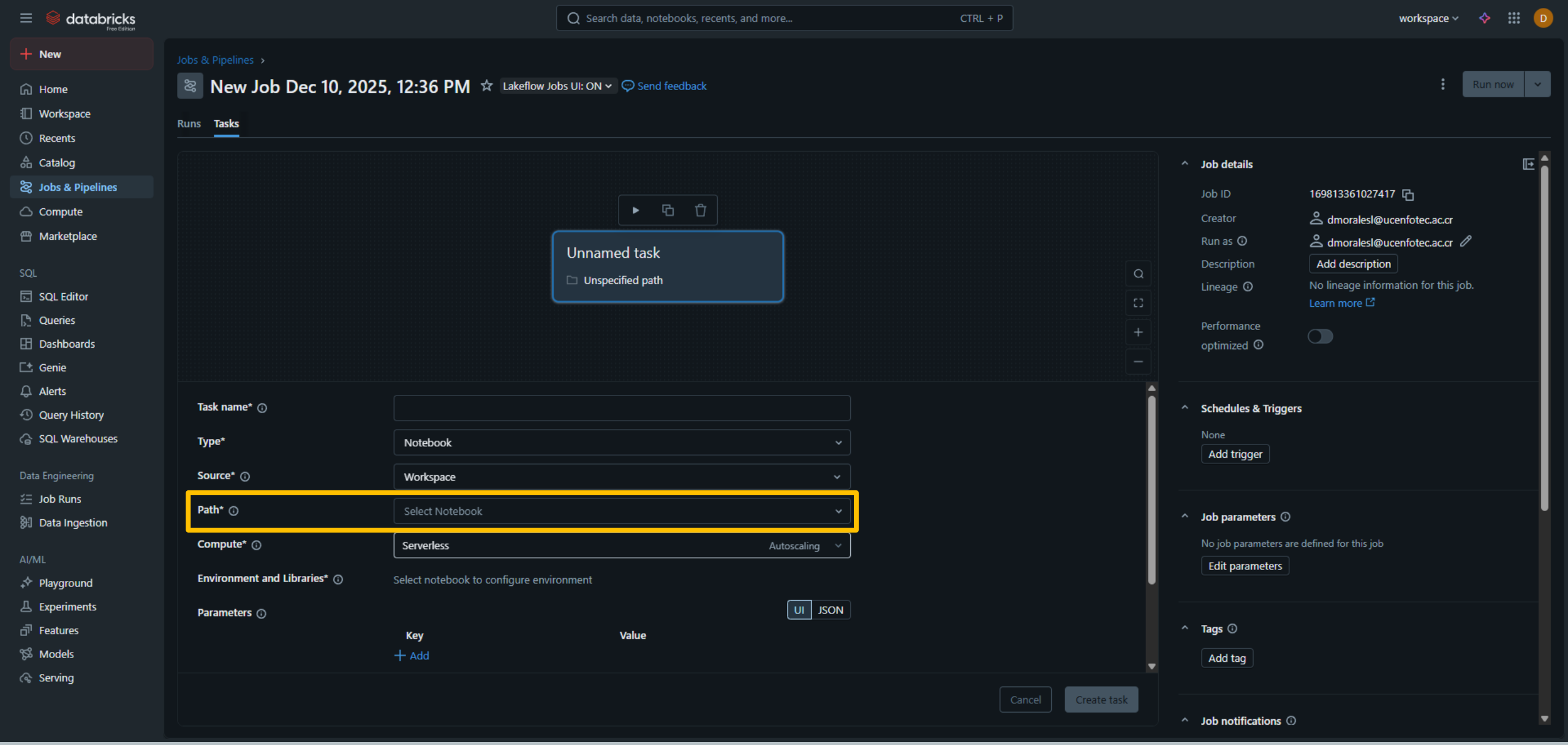


Nos dirigimos a la sección de “Jobs y Pipelines”

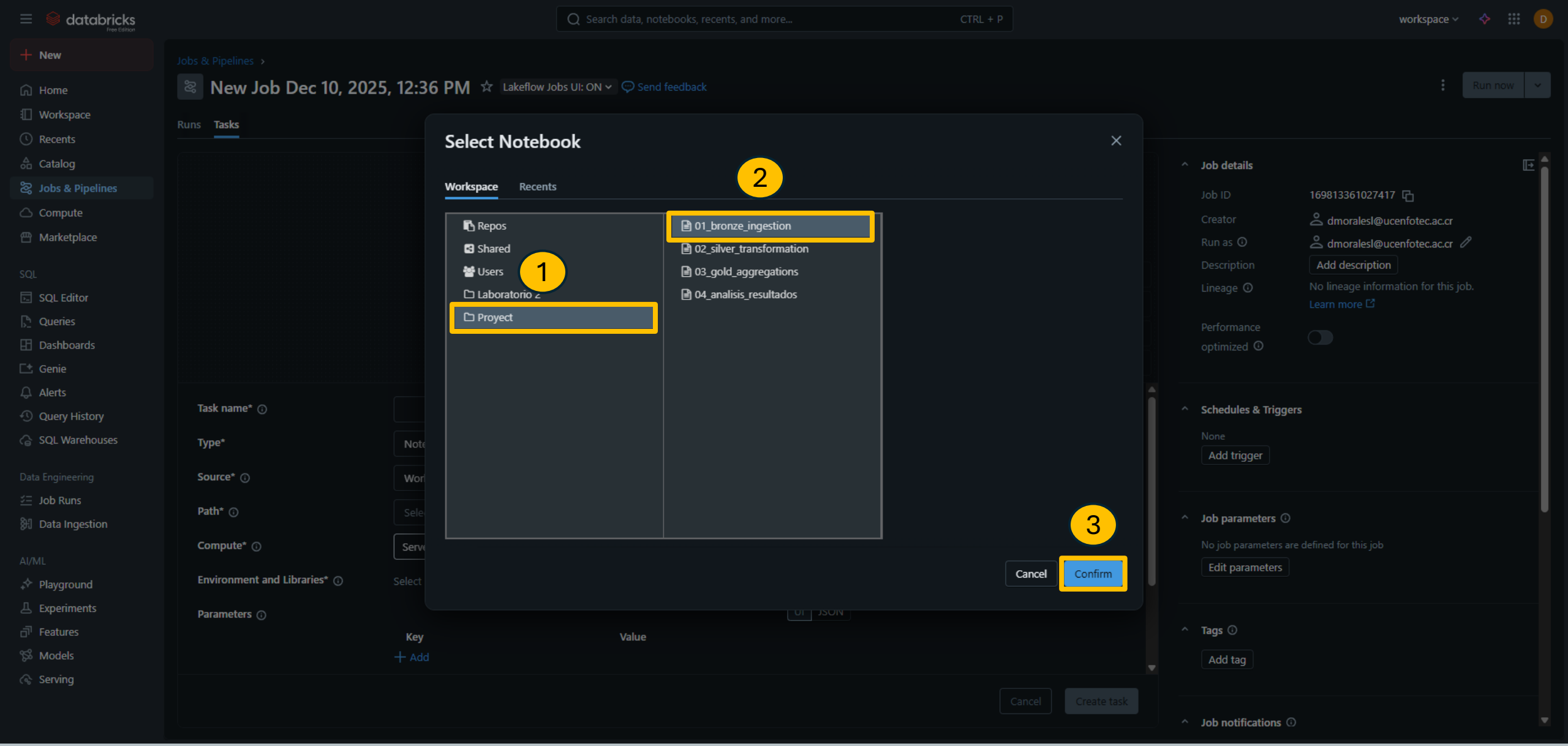


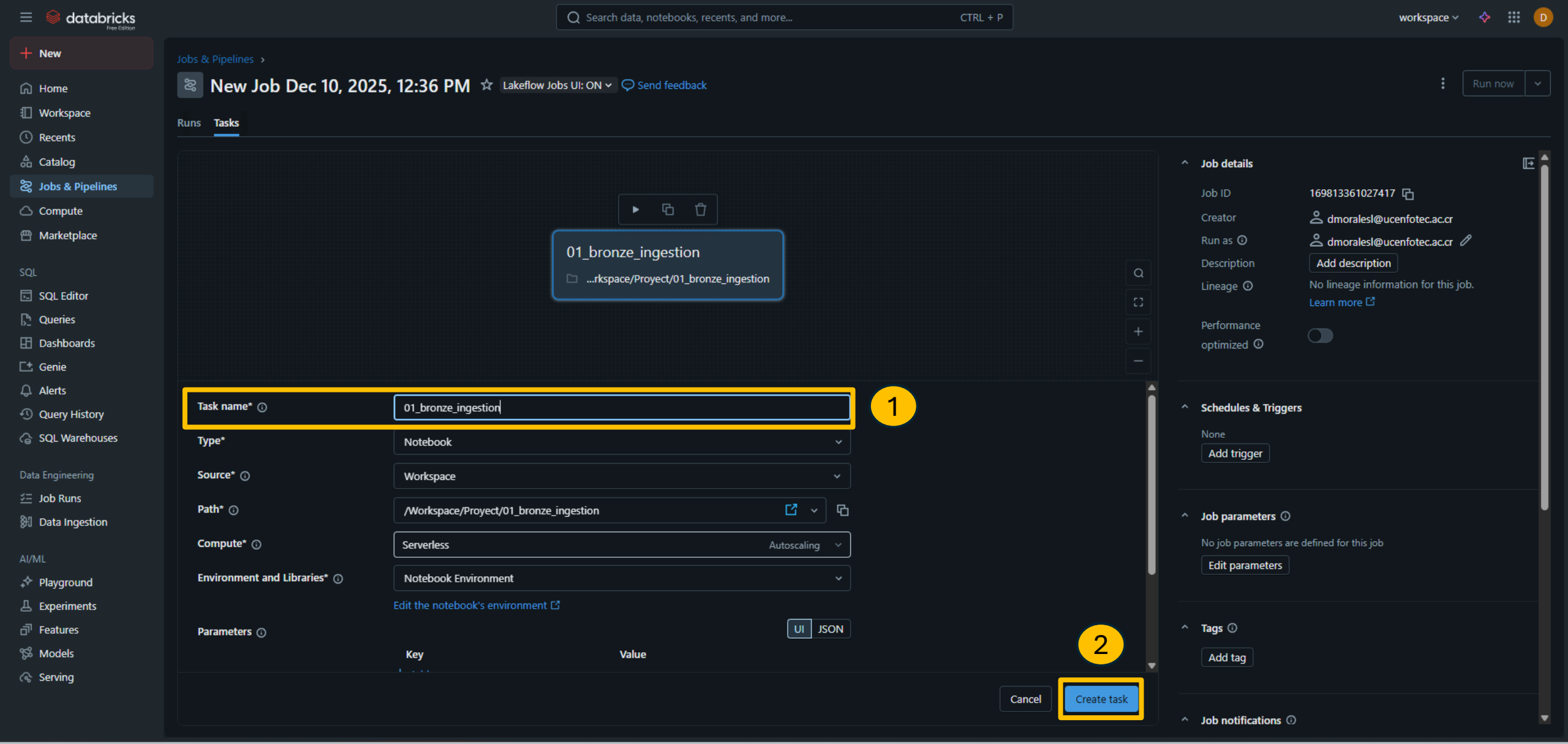
Creamos nuestro Job


databricks
Free Edition
Search data, notebooks, recents, and more...
CTRL + P
workspace
New
Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace
SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses
Data Engineering
Job Runs
Data Ingestion
AI/ML
Playground
Experiments
Features
Models
Serving
Jobs & Pipelines
New Job Dec 10, 2025, 12:36 PM
Lakeflow Jobs UI: ON
Send feedback
Run now
Runs
Tasks
Add your first task
Choose from your recently used ones...
Notebook
Run a notebook
or
Add another task type
Job details
Job ID
169813361027417
Creator
dmoralesl@ucenfotec.ac.cr
Run as
dmoralesl@ucenfotec.ac.cr
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
Tags
Add tag
Job notifications

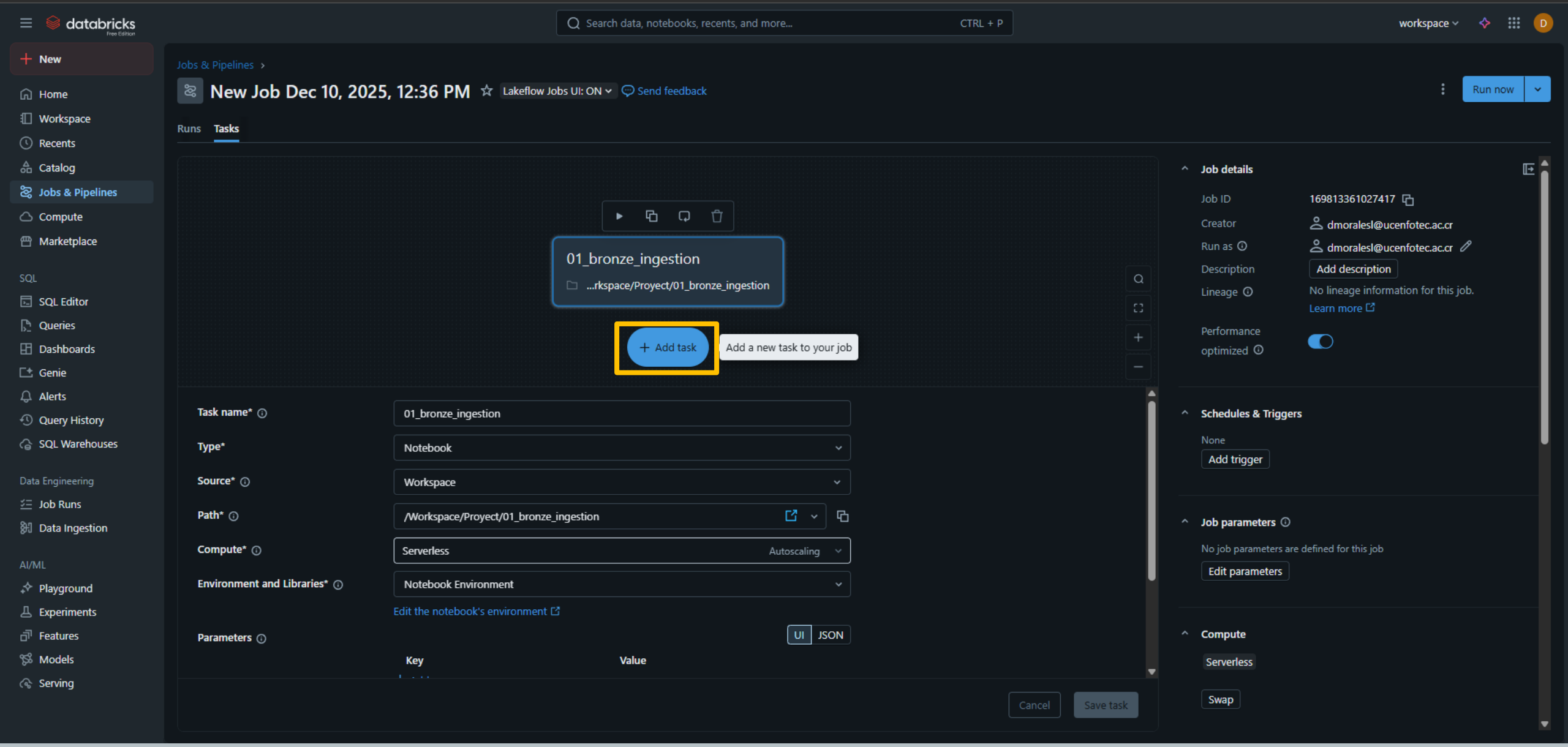


Seleccionamos el notebook de la etapa Bronze para esta primer Task

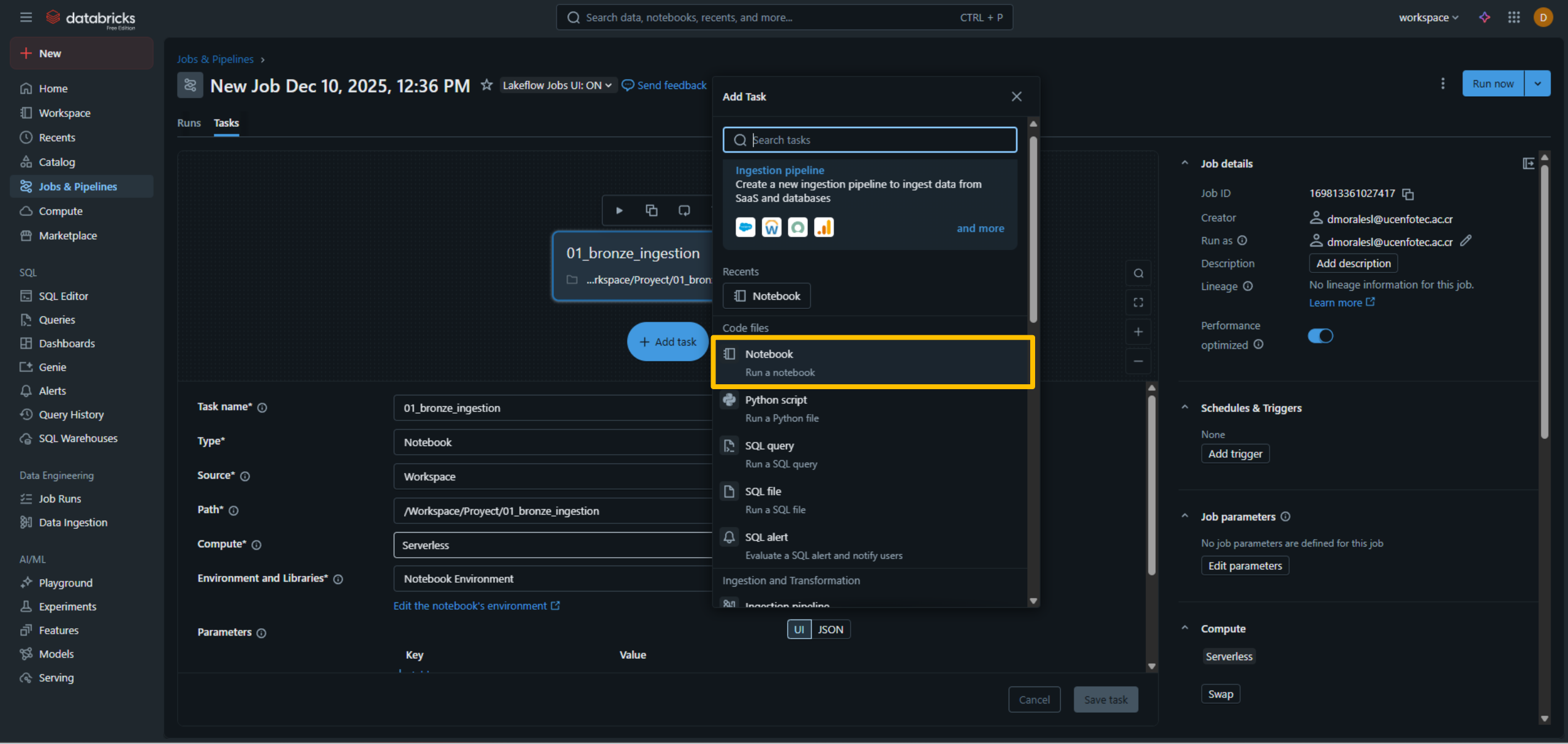

databricks
Search data, notebooks, recents, and more... CTRL + P
workspace
New
Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace
SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses
Data Engineering
Job Runs
Data Ingestion
AI/ML
Playground
Experiments
Features
Models
Serving
Jobs & Pipelines
New Job Dec 10, 2025, 12:36 PM
Lakeflow Jobs UI: ON
Send feedback
Run now
Runs
Tasks
Task name*
Type*
Source*
Path*
Compute*
Environment and Libraries*
Parameters
Key
Value
Add
Cancel
Create task
Job details
Job ID 169813361027417
Creator dmoralesl@ucenfotec.ac.cr
Run as dmoralesl@ucenfotec.ac.cr
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
Tags
Add tag
Job notifications
Select Notebook
Workspace
Recents
Repos
Shared
Users
Laboratorio 2
Proyect
01_bronze_ingestion
02_silver_transformation
03_gold_aggregations
04_analisis_resultados
1
2
3
Cancel
Confirm

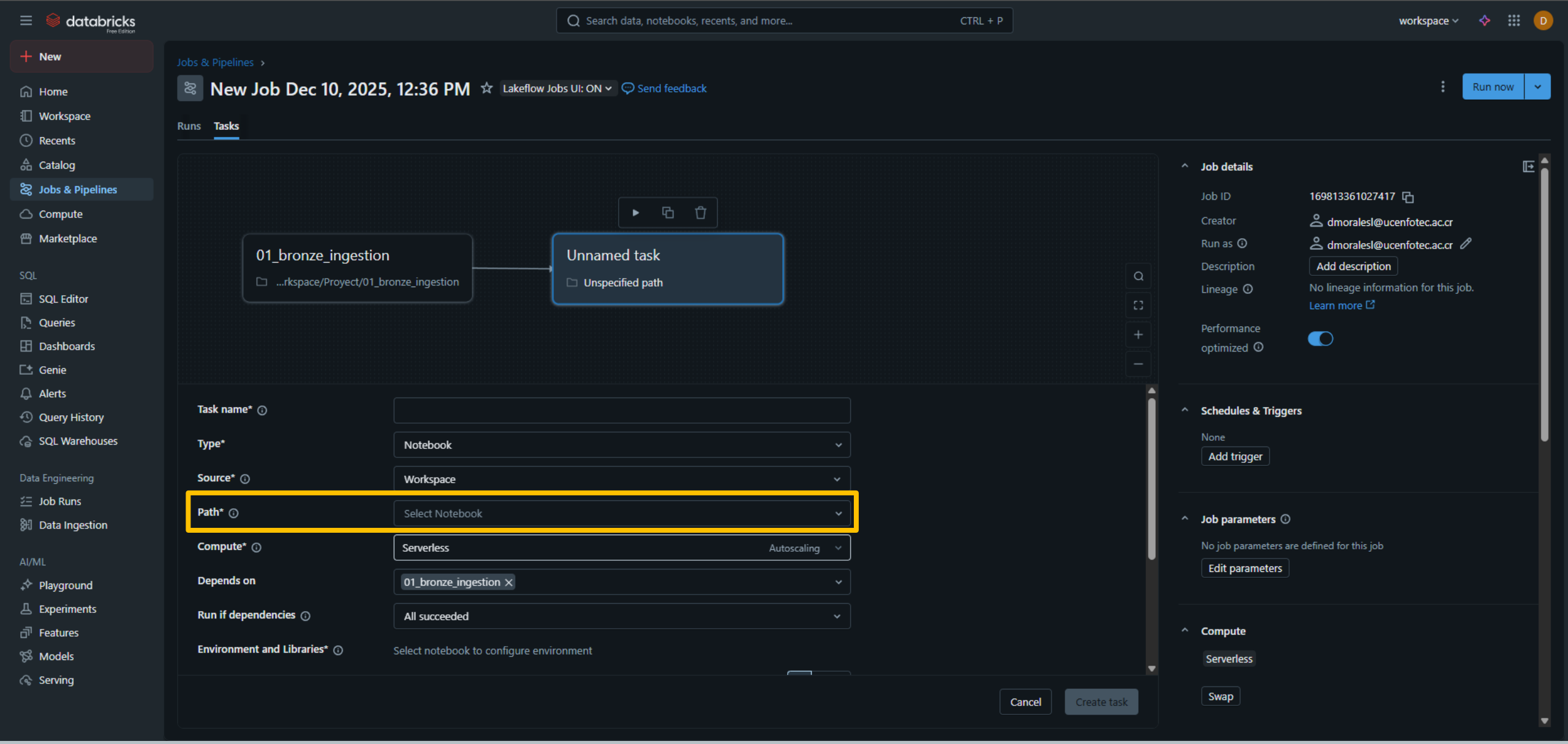


Le nombramos un nombre a nuestro Task y lo creamos

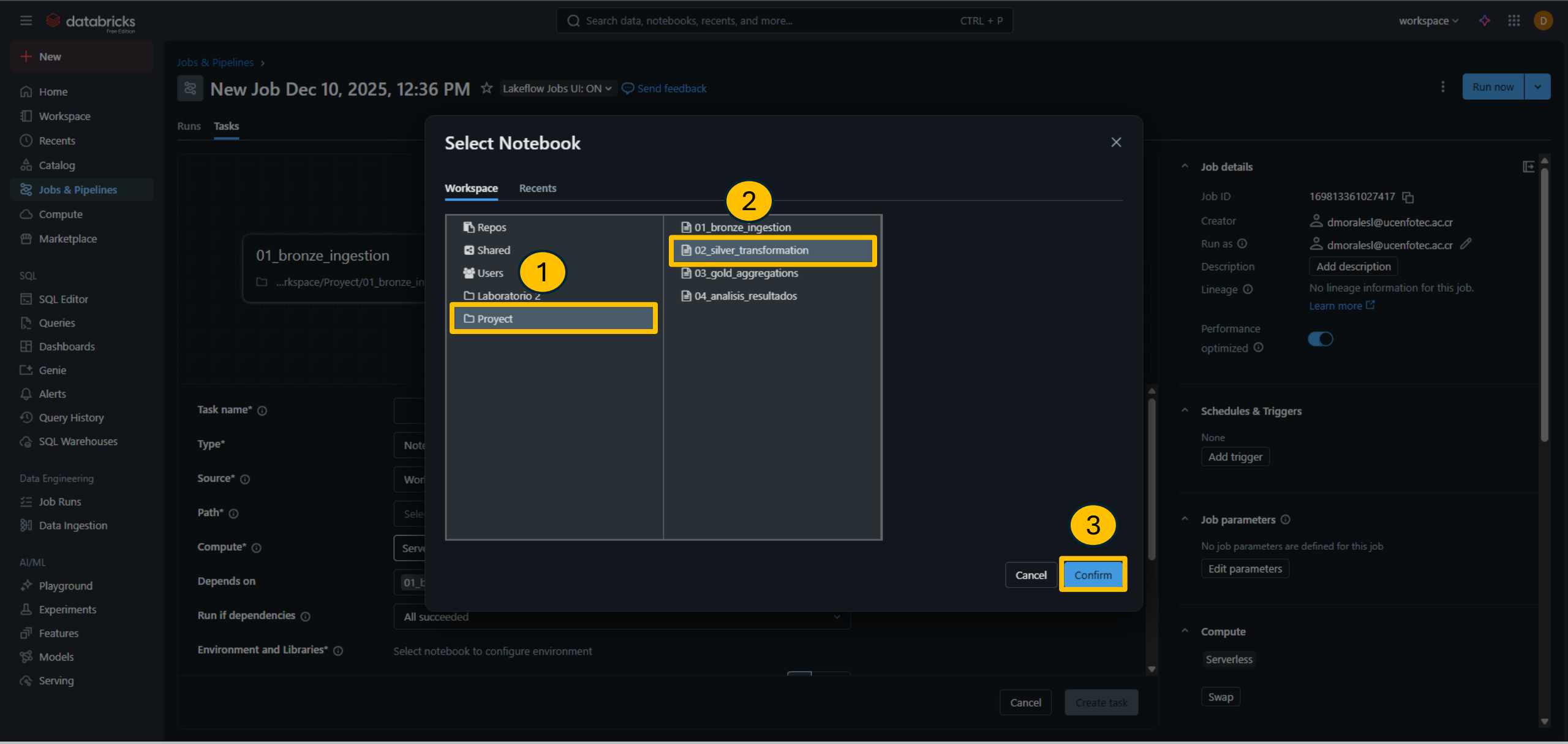


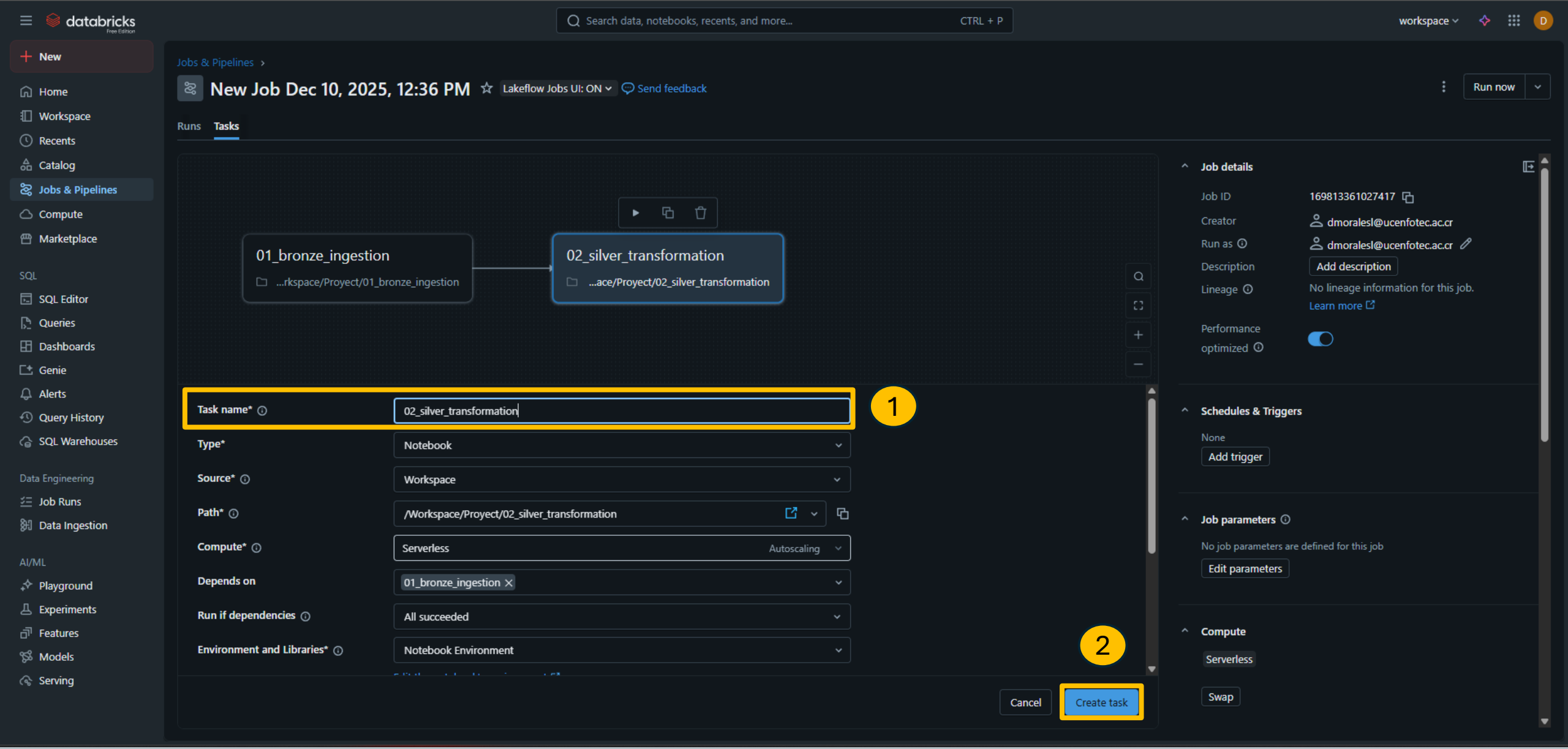
Añadimos un nuevo task a nuestro job


databricks
Search data, notebooks, recents, and more... CTRL + P
workspace
New
Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace
SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses
Data Engineering
Job Runs
Data Ingestion
AI/ML
Playground
Experiments
Features
Models
Serving
Jobs & Pipelines
New Job Dec 10, 2025, 12:36 PM
Lakeflow Jobs UI: ON
Send feedback
Run now
Runs
Tasks
01_bronze_ingestion
...rkspace/Proyect/01_bron
Add task
Add Task
Search tasks
Ingestion pipeline
Create a new ingestion pipeline to ingest data from SaaS and databases
and more
Recents
Notebook
Code files
Notebook
Run a notebook
Python script
Run a Python file
SQL query
Run a SQL query
SQL file
Run a SQL file
SQL alert
Evaluate a SQL alert and notify users
Ingestion and Transformation
Task name*
01_bronze_ingestion
Type*
Notebook
Source*
Workspace
Path*
/Workspace/Proyect/01_bronze_ingestion
Compute*
Serverless
Environment and Libraries*
Notebook Environment
Edit the notebook's environment
Parameters
UI
JSON
Key
Value
Cancel
Save task
Job details
Job ID
169813361027417
Creator
dmoralesl@ucenfotec.ac.cr
Run as
dmoralesl@ucenfotec.ac.cr
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
Swap

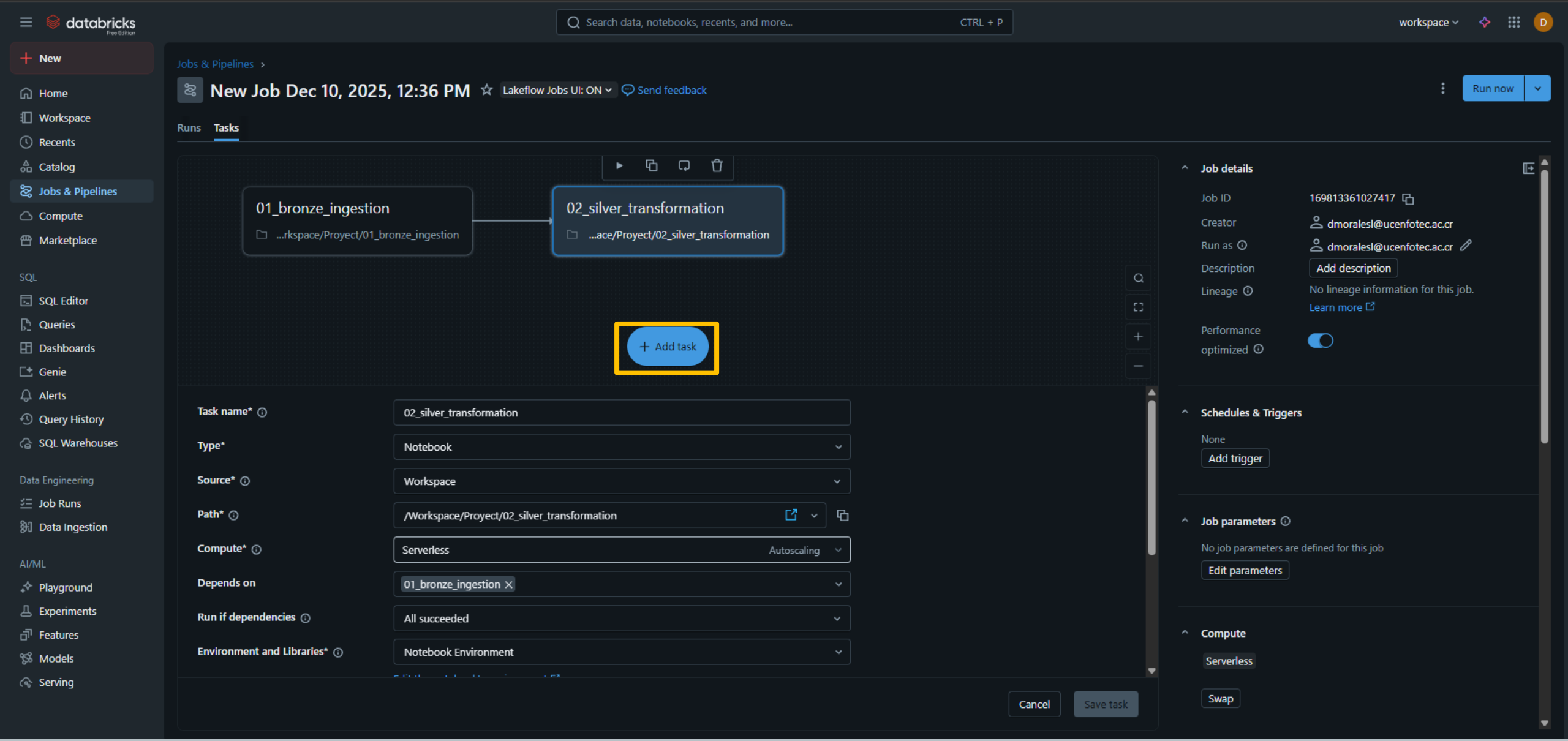


Seleccionamos el notebook de la etapa Silver para este segundo Task

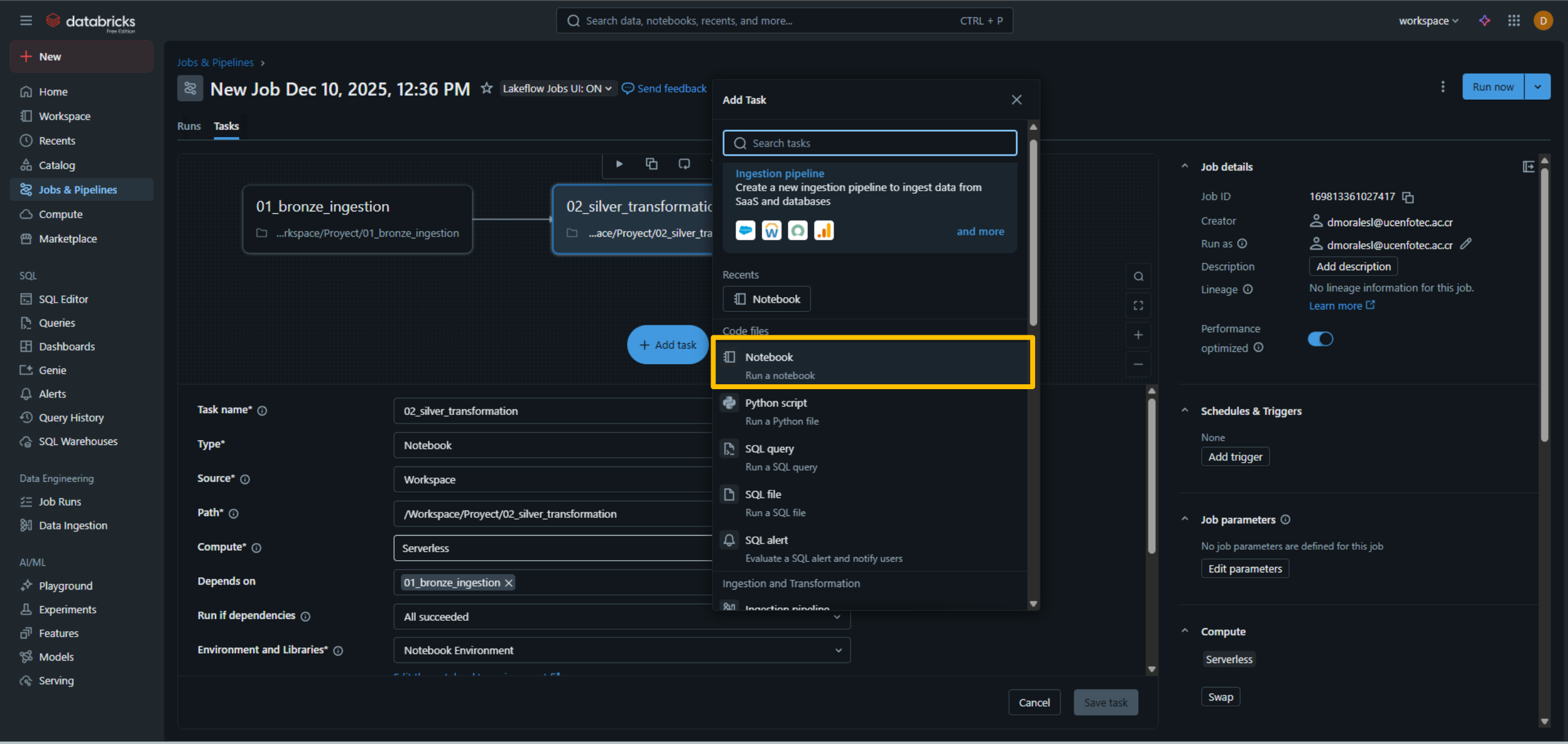

databricks
Search data, notebooks, recents, and more... CTRL + P
workspace
New
Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace
SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses
Data Engineering
Job Runs
Data Ingestion
AI/ML
Playground
Experiments
Features
Models
Serving
Jobs & Pipelines
New Job Dec 10, 2025, 12:36 PM
Lakeflow Jobs UI: ON
Send feedback
Run now
Runs
Tasks
01_bronze_ingestion
...rkspace/Proyect/01_bronze_in
Task name*
Type*
Source*
Path*
Compute*
Depends on
Run if dependencies
All succeeded
Environment and Libraries*
Select notebook to configure environment
Cancel
Create task
Select Notebook
Workspace
Recents
Repos
Shared
Users
Laboratorio 2
Proyect
01_bronze_ingestion
02_silver_transformation
03_gold_aggregations
04_analisis_resultados
1
2
3
Cancel
Confirm
Job details
Job ID
169813361027417
Creator
dmoralesl@ucenfotec.ac.cr
Run as
dmoralesl@ucenfotec.ac.cr
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
Swap

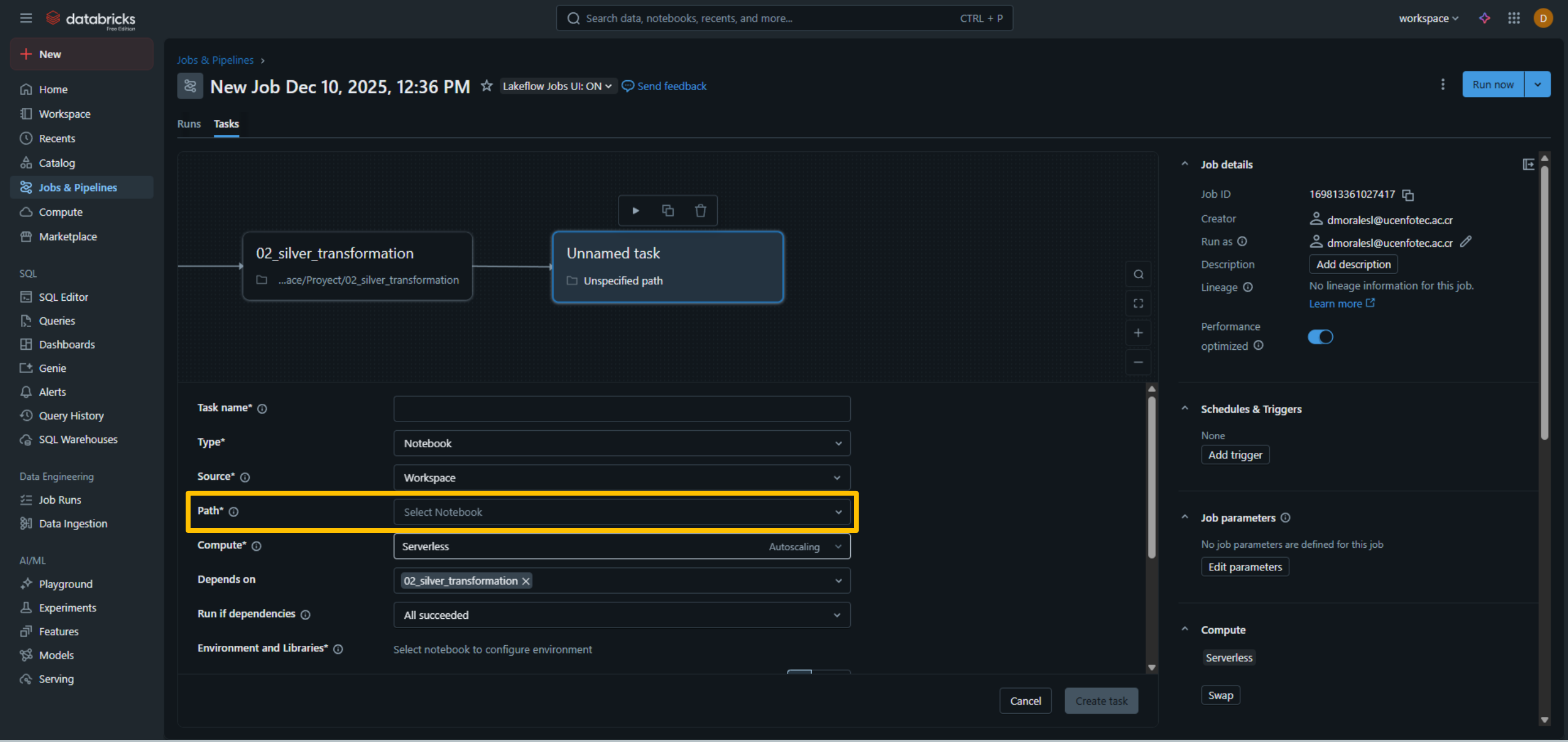


Le nombramos un nombre a nuestro Task y lo creamos

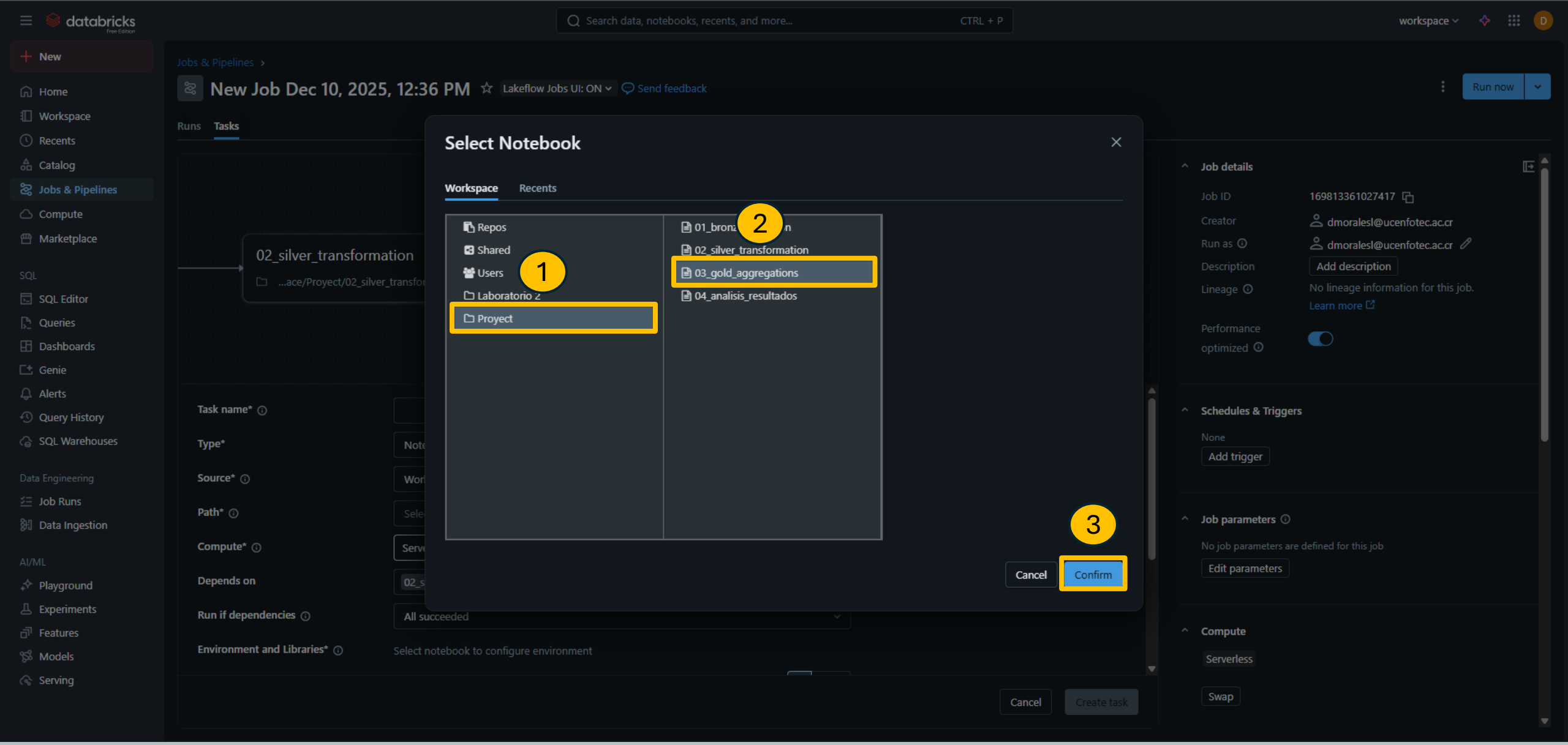


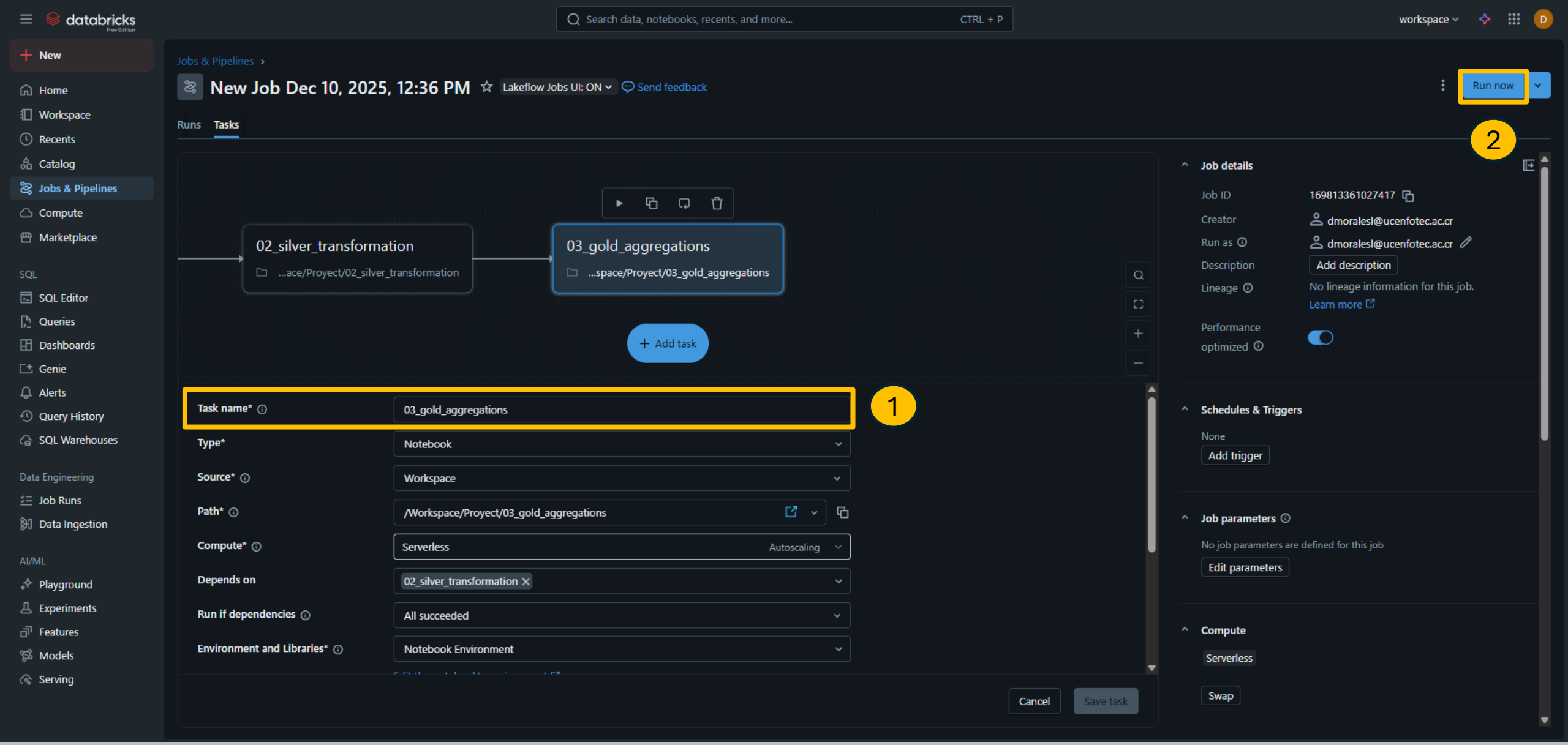
Añadimos nuestro ultimo task a nuestro job


databricks
Search data, notebooks, recents, and more... CTRL + P
workspace
New
Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace
SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses
Data Engineering
Job Runs
Data Ingestion
AI/ML
Playground
Experiments
Features
Models
Serving
Jobs & Pipelines
New Job Dec 10, 2025, 12:36 PM
Lakeflow Jobs UI: ON
Send feedback
Run now
Runs
Tasks
01_bronze_ingestion
...rkspace/Proyect/01_bronze_ingestion
02_silver_transformatic
...ace/Proyect/02_silver_tra
Add task
Add Task
Search tasks
Ingestion pipeline
Create a new ingestion pipeline to ingest data from SaaS and databases
and more
Recents
Notebook
Code files
Notebook
Run a notebook
Python script
Run a Python file
SQL query
Run a SQL query
SQL file
Run a SQL file
SQL alert
Evaluate a SQL alert and notify users
Ingestion and Transformation
Task name*
02_silver_transformation
Type*
Notebook
Source*
Workspace
Path*
/Workspace/Proyect/02_silver_transformation
Compute*
Serverless
Depends on
01_bronze_ingestion
Run if dependencies
All succeeded
Environment and Libraries*
Notebook Environment
Cancel
Save task
Job details
Job ID
169813361027417
Creator
dmoralesl@ucenfotec.ac.cr
Run as
dmoralesl@ucenfotec.ac.cr
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
Swap

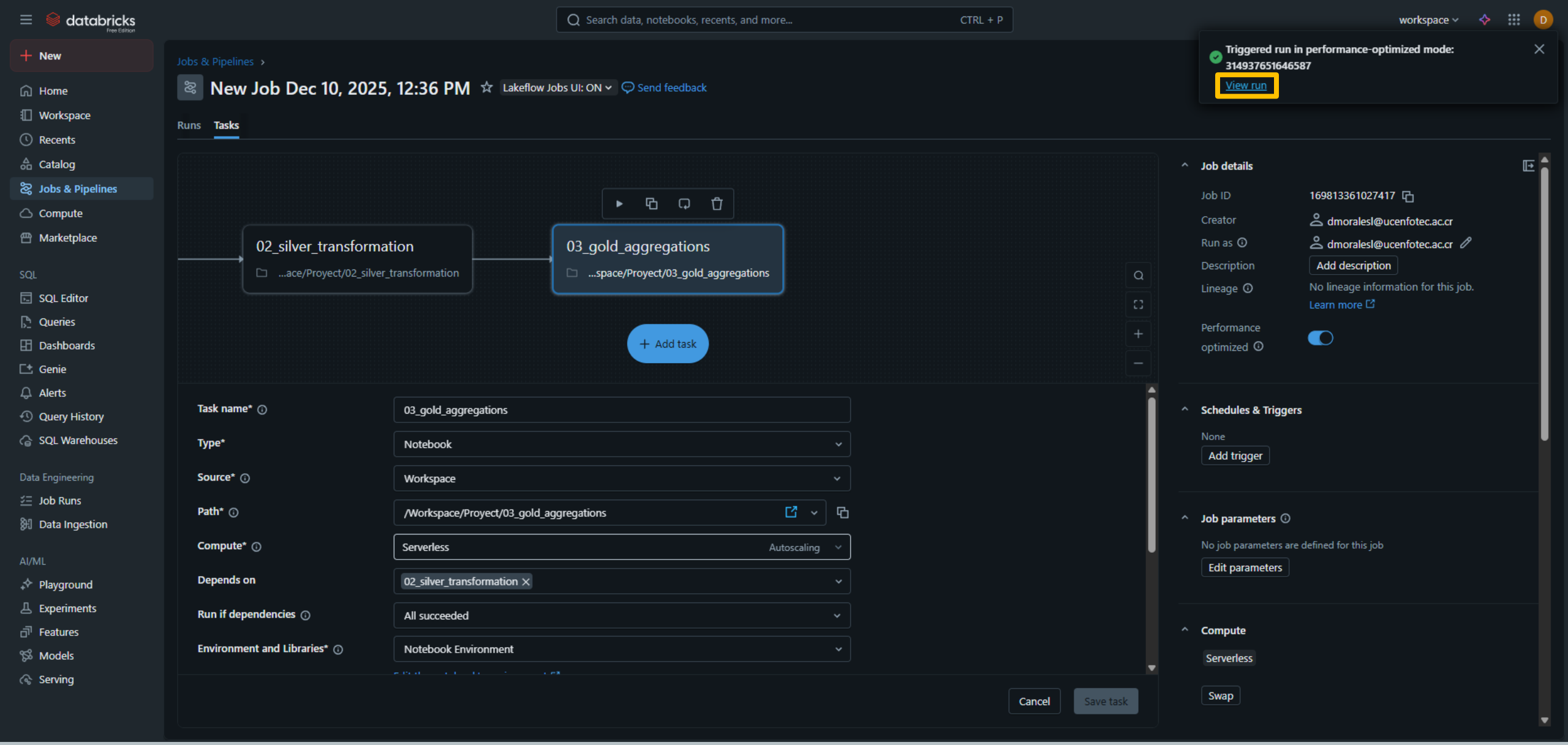


Seleccionamos el notebook de la etapa Gold para este ultimo Task

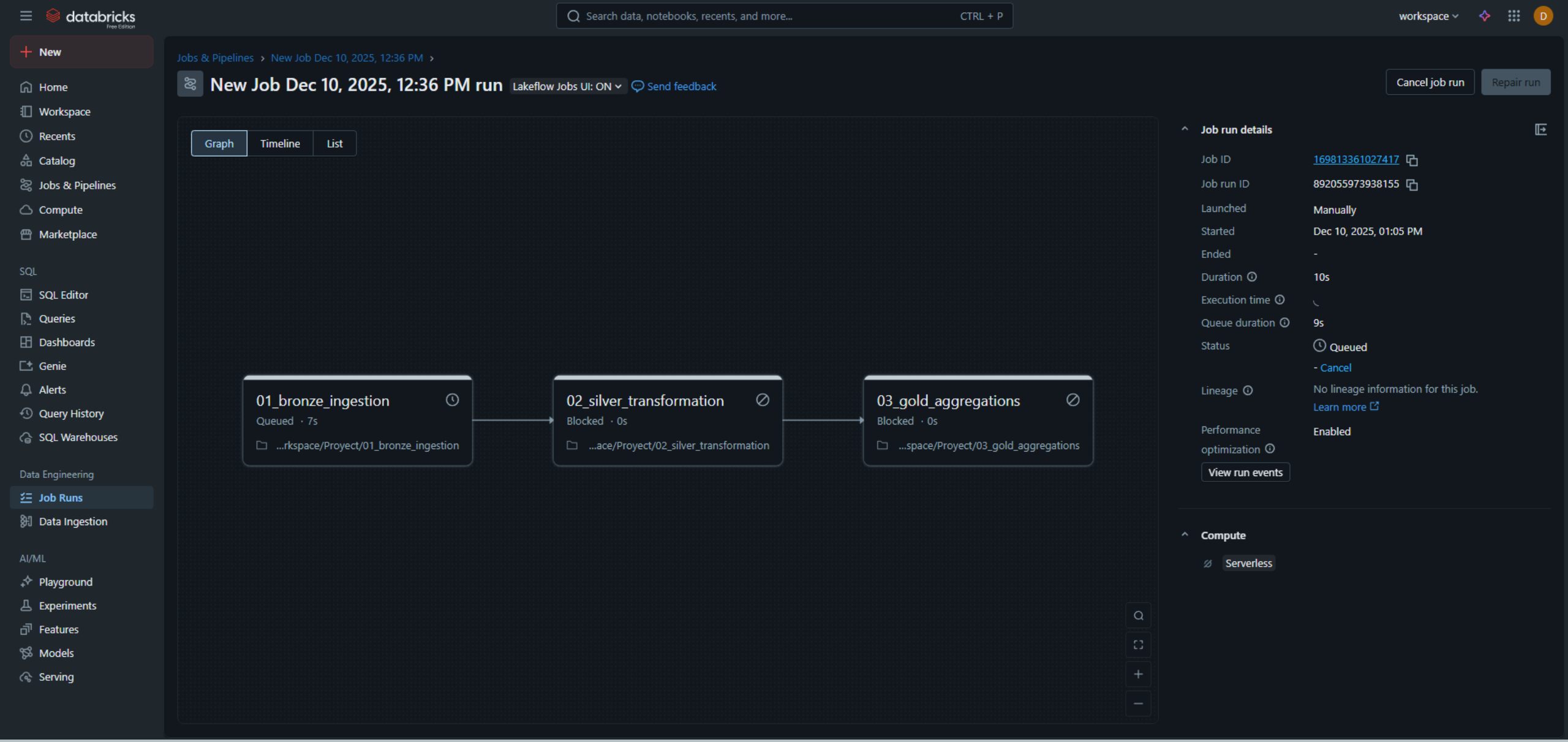

databricks
Search data, notebooks, recents, and more... CTRL + P
workspace
New
Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace
SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses
Data Engineering
Job Runs
Data Ingestion
AI/ML
Playground
Experiments
Features
Models
Serving
Jobs & Pipelines
New Job Dec 10, 2025, 12:36 PM
Lakeflow Jobs UI: ON
Send feedback
Run now
Runs
Tasks
02_silver_transformation
...ace/Proyect/02_silver_transfor
Task name*
Type*
Source*
Path*
Compute*
Depends on
Run if dependencies
All succeeded
Environment and Libraries*
Select notebook to configure environment
Cancel
Create task
Select Notebook
Workspace
Recents
Repos
Shared
Users
Laboratorio 2
Proyect
01_bronz
02_silver_transformation
03_gold_aggregations
04_analisis_resultados
1
2
3
Cancel
Confirm
Job details
Job ID
169813361027417
Creator
dmoralesl@ucenfotec.ac.cr
Run as
dmoralesl@ucenfotec.ac.cr
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
Swap



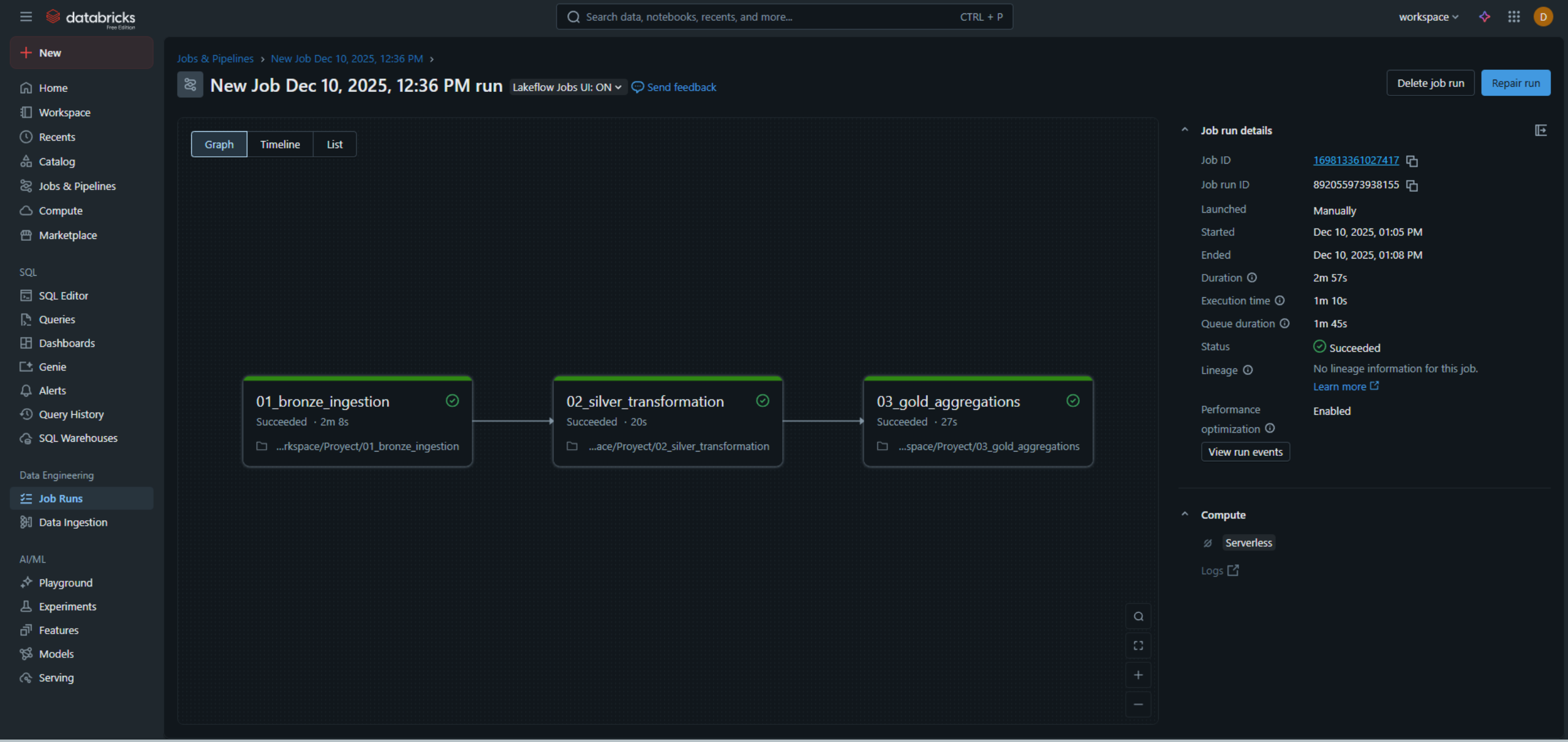
Le nombramos un nombre a nuestro Task y ejecutamos nuestro Job



Abrimos el panel del estado de la ejecución del Job



Esperémonos que se complete todo el proceso del Job



Aquí ya se completo el proceso

New
Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace

SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses

Data Engineering
Job Runs
Data Ingestion

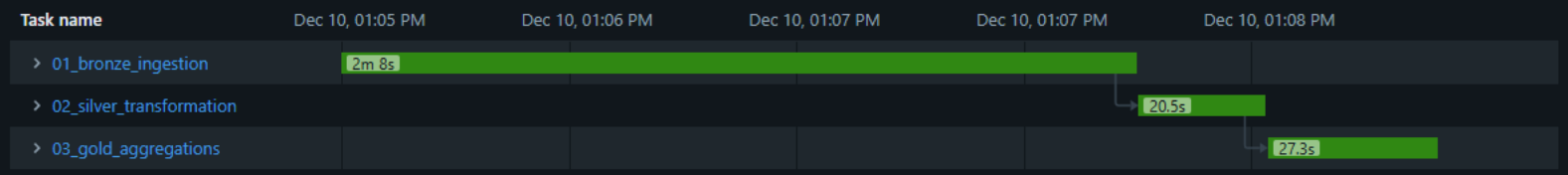
AI/ML
Playground
Experiments
Features
Models
Serving

New Job Dec 10, 2025, 12:36 PM run

Lakeflow Jobs UI: ON | Send feedback

Delete job run | Repair run

Graph | Timeline | List



Job run details

Job ID	169813361027417
Job run ID	892055973938155
Launched	Manually
Started	Dec 10, 2025, 01:05 PM
Ended	Dec 10, 2025, 01:08 PM
Duration	2m 57s
Execution time	1m 10s
Queue duration	1m 45s
Status	Succeeded
Lineage	No lineage information for this job. Learn more
Performance optimization	Enabled

View run events

Compute

Serverless

Logs

New
Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace
SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses
Data Engineering
Job Runs
Data Ingestion
AI/ML
Playground
Experiments
Features
Models
Serving

New Job Dec 10, 2025, 12:36 PM run

Lakeflow Jobs UI: ON | Send feedback

Delete job run | Repair run

Graph | Timeline | List | Filter task names starting with... | Status | Type

Status	Name	Type	Resource	Duration	Depends on
Succeeded	01_bronze_ingestion	Notebook	/Workspace/Proyect/01_bro	2m 8s	
Succeeded	02_silver_transformation	Notebook	/Workspace/Proyect/02_silv	20s	01_bronze_ingestion
Succeeded	03_gold_aggregations	Notebook	/Workspace/Proyect/03_gol	27s	02_silver_transformation

Job run details

Job ID	169813361027417
Job run ID	892055973938155
Launched	Manually
Started	Dec 10, 2025, 01:05 PM
Ended	Dec 10, 2025, 01:08 PM
Duration	2m 57s
Execution time	1m 10s
Queue duration	1m 45s
Status	Succeeded
Lineage	No lineage information for this job. Learn more
Performance optimization	Enabled

View run events

Compute

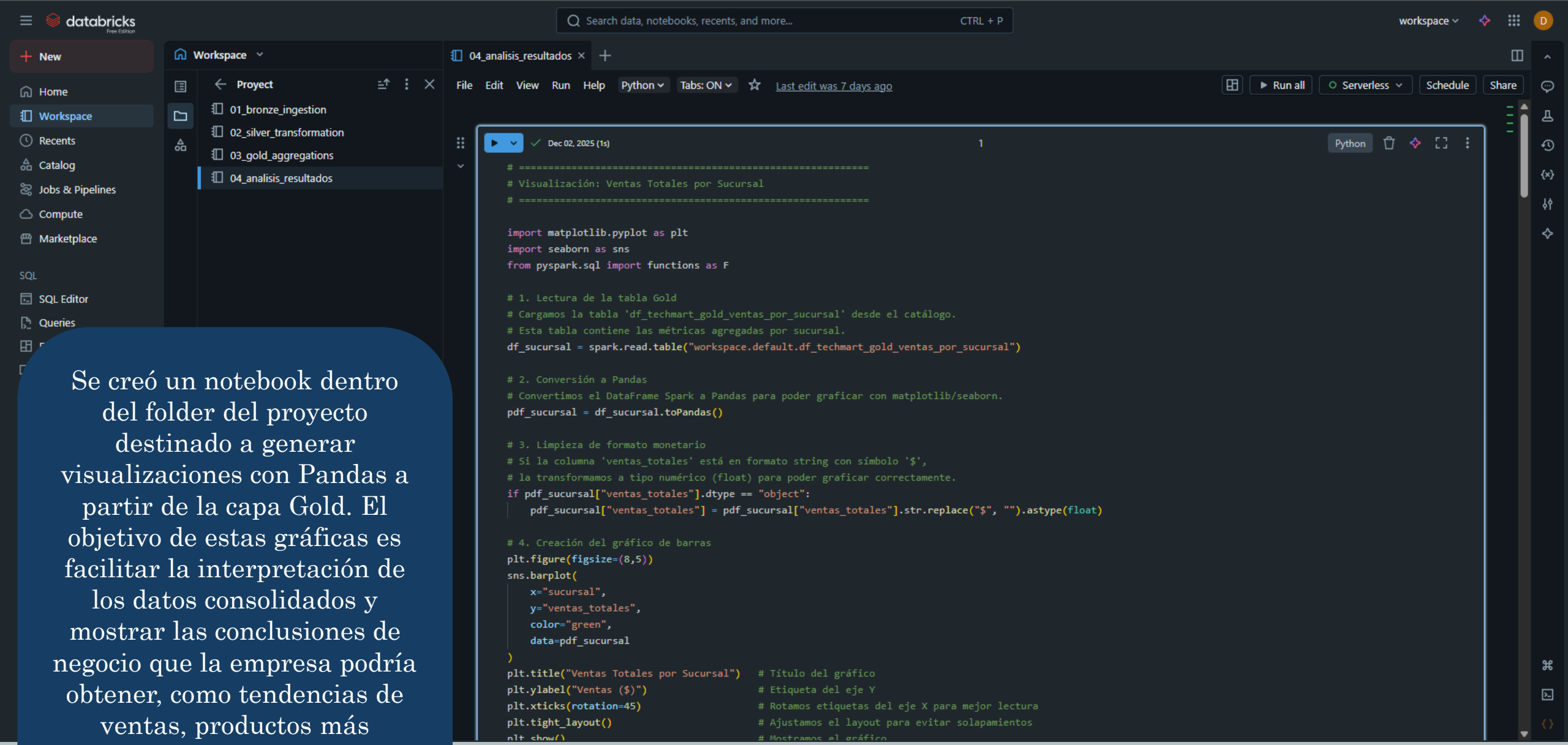
Serverless

Logs

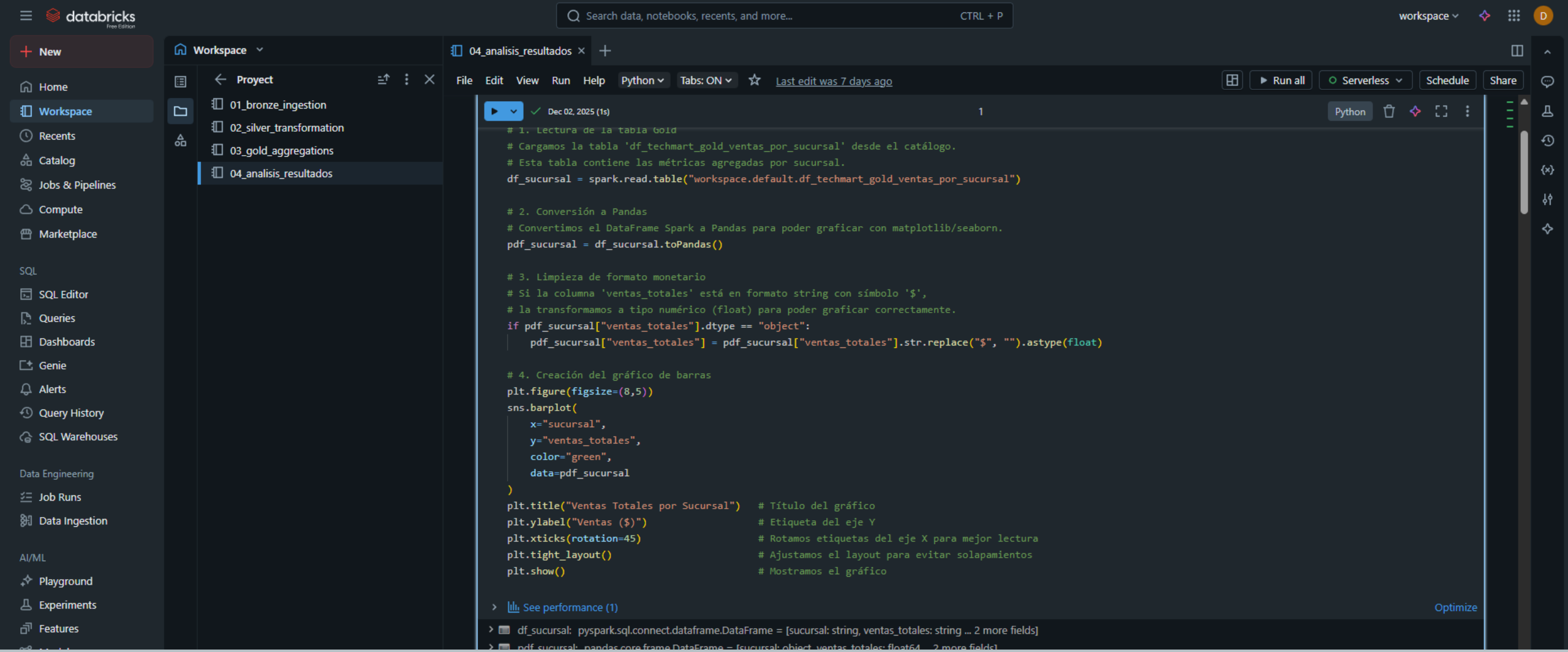
Con la ejecución exitosa del Job en Databricks, todas las etapas del pipeline (Bronze, Silver y Gold) se completaron en estado **"Succeeded" (verde)**. Esto confirma que el proceso de **Extracción, Transformación y Carga (ETL)** quedó implementado y funcionando correctamente, con los datos consolidados en formato Delta Lake dentro del catálogo y listos para análisis y reportería.

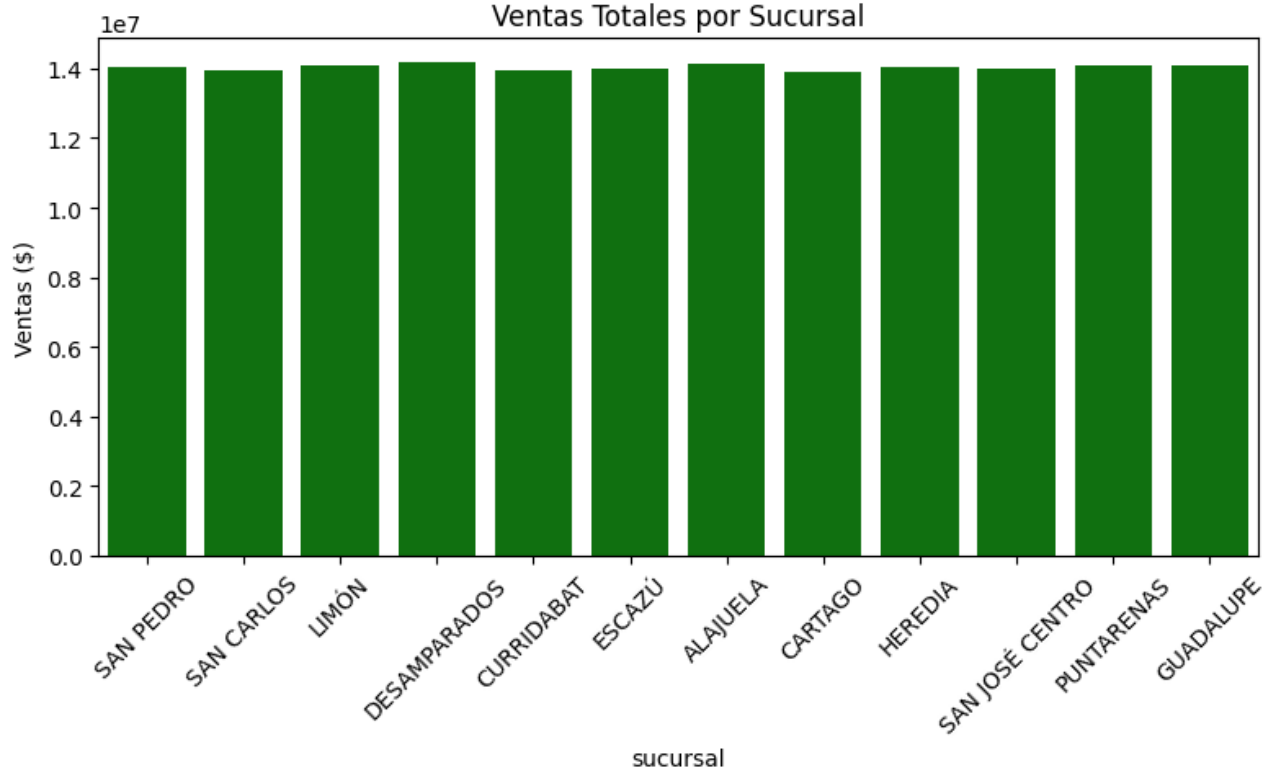
Cabe señalar que, al trabajar con la **versión gratuita de Databricks (Community Edition)**, no es posible configurar parámetros avanzados como el **tiempo de ejecución programada del Job**. Por lo tanto, la ejecución debe realizarse de manera manual cada vez que se requiera correr el pipeline.

Visualización de Resultados



Se creó un notebook dentro del folder del proyecto destinado a generar visualizaciones con Pandas a partir de la capa Gold. El objetivo de estas gráficas es facilitar la interpretación de los datos consolidados y mostrar las conclusiones de negocio que la empresa podría obtener, como tendencias de ventas, productos más demandados o comparativas entre sucursales.


databricks
Search data, notebooks, recents, and more... CTRL + P
workspace
New
Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace
SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses
Data Engineering
Job Runs
Data Ingestion
AI/ML
Playground
Experiments
Features
Workspace
Proyect
01_bronze_ingestion
02_silver_transformation
03_gold_aggregations
04_analisis_resultados
04_analisis_resultados
File Edit View Run Help Python Tabs: ON Last edit was 7 days ago
Run all Serverless Schedule Share
Dec 02, 2025 (1s)
1
Python
# 1. Lectura de la tabla Gold
# Cargamos la tabla 'df_techmart_gold_ventas_por_sucursal' desde el catálogo.
# Esta tabla contiene las métricas agregadas por sucursal.
df_sucursal = spark.read.table("workspace.default.df_techmart_gold_ventas_por_sucursal")
# 2. Conversión a Pandas
# Convertimos el DataFrame Spark a Pandas para poder graficar con matplotlib/seaborn.
pdf_sucursal = df_sucursal.toPandas()
# 3. Limpieza de formato monetario
# Si la columna 'ventas_totales' está en formato string con símbolo '$',
# la transformamos a tipo numérico (float) para poder graficar correctamente.
if pdf_sucursal["ventas_totales"].dtype == "object":
pdf_sucursal["ventas_totales"] = pdf_sucursal["ventas_totales"].str.replace("$", "").astype(float)
# 4. Creación del gráfico de barras
plt.figure(figsize=(8,5))
sns.barplot(
x="sucursal",
y="ventas_totales",
color="green",
data=pdf_sucursal
)
plt.title("Ventas Totales por Sucursal")
# Título del gráfico
plt.ylabel("Ventas ($)")
# Etiqueta del eje Y
plt.xticks(rotation=45)
# Rotamos etiquetas del eje X para mejor lectura
plt.tight_layout()
# Ajustamos el layout para evitar solapamientos
plt.show()
# Mostramos el gráfico
See performance (1)
Optimize
df_sucursal: pyspark.sql.connect.dataframe.DataFrame = [sucursal: string, ventas_totales: string ... 2 more fields]



El gráfico muestra las ventas totales en dólares por sucursal, utilizando datos consolidados de la capa Gold. Se observa que las 12 sucursales presentan niveles de ventas similares, sin diferencias marcadas entre ellas.
Esta visualización permite a la empresa concluir que, en términos de volumen de ventas, todas las sucursales mantienen un desempeño relativamente equilibrado, lo cual sugiere una distribución homogénea de la demanda a nivel nacional.
Este resultado puede ser útil para:

- Validar que no existen sucursales con bajo rendimiento crítico.
- Justificar una estrategia comercial uniforme en todas las regiones.
- Identificar oportunidades para potenciar ventas en zonas con mayor potencial, si se combinan estos datos con variables externas como población o competencia local.

databricks
Free Edition

Search data, notebooks, recents, and more... CTRL + P

workspace

New

Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace

SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses

Data Engineering
Job Runs
Data Ingestion

AI/ML
Playground
Experiments
Features
Models
Serving

Workspace

Proyect
01_bronze_ingestion
02_silver_transformation
03_gold_aggregations
04_analisis_resultados

04_analisis_resultados

File Edit View Run Help Python Tabs: ON Last edit was 7 days ago

Run all Serverless Schedule Share

Dec 02, 2025 (1s) 2

```
# ============================================================
# Visualización: Tendencia Mensual de Ventas
# ============================================================

# 1. Lectura de la tabla Gold
# Cargamos la tabla 'df_techmart_gold_tendencia_mensual' desde el catálogo.
# Esta tabla contiene las métricas agregadas por año y mes.
df_tendencia = spark.read.table("workspace.default.df_techmart_gold_tendencia_mensual")

# 2. Conversión a Pandas
# Convertimos el DataFrame Spark a Pandas para poder graficar con matplotlib/seaborn.
pdf_tendencia = df_tendencia.toPandas()

# 3. Limpieza de formato monetario
# Si la columna 'ventas_totales' está en formato string con símbolo '$',
# la transformamos a tipo numérico (float) para poder graficar correctamente.
if pdf_tendencia["ventas_totales"].dtype == "object":
    pdf_tendencia["ventas_totales"] = pdf_tendencia["ventas_totales"].str.replace("$", "").astype(float)

# 4. Preparación de columnas de fecha
# Convertimos 'mes' a entero para ordenar correctamente.
pdf_tendencia["mes"] = pdf_tendencia["mes"].astype(int)
# Ordenamos cronológicamente por año y mes.
pdf_tendencia = pdf_tendencia.sort_values(["año", "mes"])
# Creamos una columna 'periodo' con formato "año-mes" para el eje X.
pdf_tendencia["periodo"] = pdf_tendencia["año"].astype(str) + "-" + pdf_tendencia["mes"].astype(str)

# 5. Creación del gráfico de línea
plt.figure(figsize=(8,5))
sns.lineplot(
    x="periodo",
    y="ventas_totales",
    data=pdf_tendencia,
    marker="o"
)
plt.title("Tendencia Mensual de Ventas")   # Título del gráfico
plt.ylabel("Ventas ($)")                   # Etiqueta del eje Y
```

databricks

Search data, notebooks, recents, and more... CTRL + P

workspace

New

Workspace

04_analisis_resultados

Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace

SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses

Data Engineering
Job Runs
Data Ingestion

AI/ML
Playground
Experiments
Features

Proyect
01_bronze_ingestion
02_silver_transformation
03_gold_aggregations
04_analisis_resultados

File Edit View Run Help Python Tabs: ON Last edit was 7 days ago

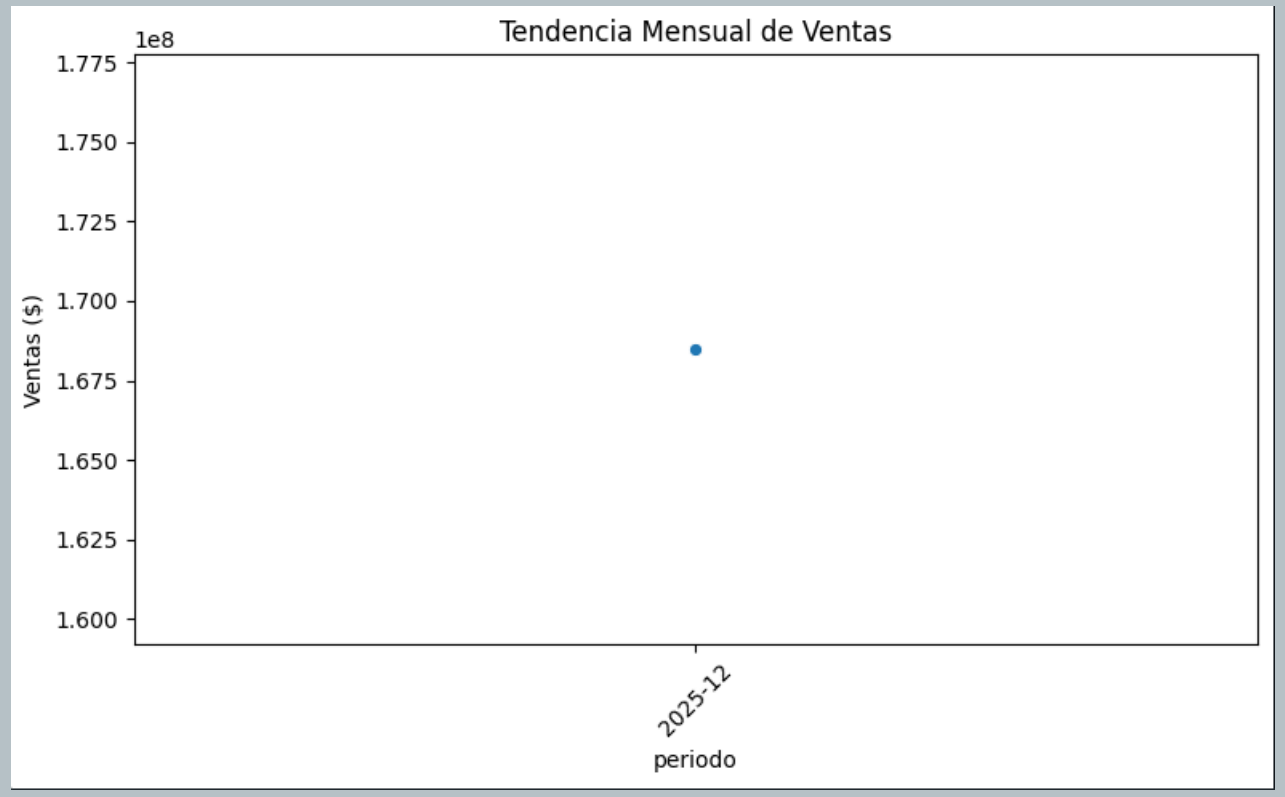
Run all Serverless Schedule Share

Dec 02, 2025 (1s) 2 Python

```python
# 2. Conversion a Pandas
# Convertimos el DataFrame Spark a Pandas para poder graficar con matplotlib/seaborn.
pdf_tendencia = df_tendencia.toPandas()

# 3. Limpieza de formato monetario
# Si la columna 'ventas_totales' está en formato string con símbolo '$',
# la transformamos a tipo numérico (float) para poder graficar correctamente.
if pdf_tendencia["ventas_totales"].dtype == "object":
    pdf_tendencia["ventas_totales"] = pdf_tendencia["ventas_totales"].str.replace("$", "").astype(float)

# 4. Preparación de columnas de fecha
# Convertimos 'mes' a entero para ordenar correctamente.
pdf_tendencia["mes"] = pdf_tendencia["mes"].astype(int)
# Ordenamos cronológicamente por año y mes.
pdf_tendencia = pdf_tendencia.sort_values(["año", "mes"])
# Creamos una columna 'periodo' con formato "año-mes" para el eje X.
pdf_tendencia["periodo"] = pdf_tendencia["año"].astype(str) + "-" + pdf_tendencia["mes"].astype(str)

# 5. Creación del gráfico de línea
plt.figure(figsize=(8,5))
sns.lineplot(
    x="periodo",
    y="ventas_totales",
    data=pdf_tendencia,
    marker="o"
)
plt.title("Tendencia Mensual de Ventas")   # Título del gráfico
plt.ylabel("Ventas ($)")                   # Etiqueta del eje Y
plt.xticks(rotation=45)                    # Rotamos etiquetas del eje X para mejor lectura
plt.tight_layout()                         # Ajustamos el layout para evitar solapamientos
plt.show()                                 # Mostramos el gráfico
```

See performance (1)

Optimize



Este gráfico muestra las ventas totales correspondientes al mes de diciembre de 2025, utilizando datos consolidados de la capa Gold. Se observa un volumen de ventas cercano a los 168.5 millones de dólares, lo que representa un nivel significativo dentro del período analizado.
Aunque el gráfico presenta un único punto, esta visualización sirve como base para implementar un análisis de tendencia temporal, permitiendo a la empresa:

- Identificar patrones estacionales o picos de demanda en meses específicos.
- Comparar el desempeño mensual frente a objetivos comerciales.
- Tomar decisiones estratégicas sobre promociones, inventario y distribución en función del comportamiento histórico.

Este resultado puede integrarse en visualizaciones futuras que incluyan múltiples períodos, facilitando el monitoreo continuo del desempeño comercial.

New
Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace

SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses

Data Engineering
Job Runs
Data Ingestion

AI/ML
Playground
Experiments
Features

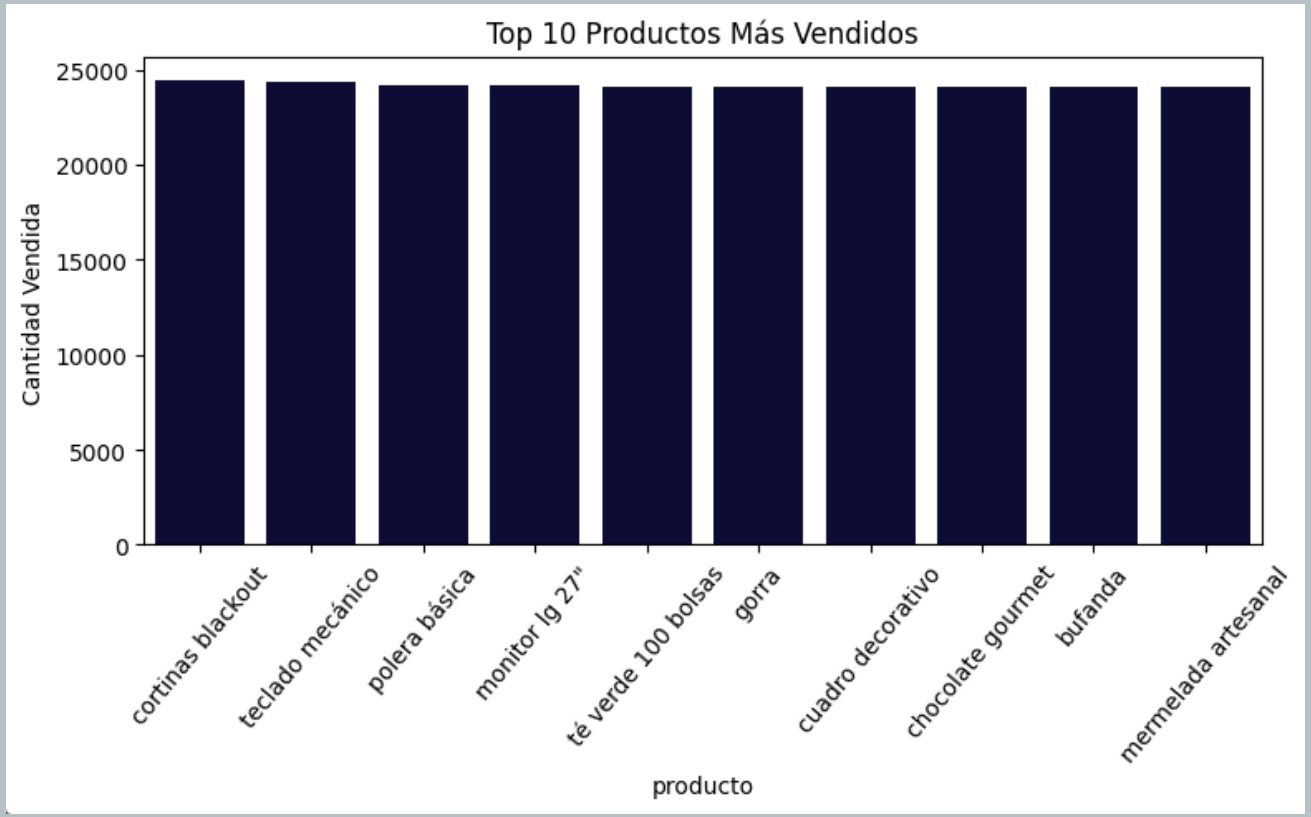
Workspace
Proyect
01_bronze_ingestion
02_silver_transformation
03_gold_aggregations
04_analisis_resultados

periodo

Dec 02, 2025 (1s) 3

```python
# ============================================================
# Visualización: Top 10 Productos Más Vendidos
# ============================================================

# 1. Lectura de la tabla Gold
# Cargamos la tabla 'df_techmart_gold_top_productos' desde el catálogo.
# Esta tabla contiene los 10 productos con mayor cantidad vendida.
df_top = spark.read.table("workspace.default.df_techmart_gold_top_productos")

# 2. Conversión a Pandas
# Convertimos el DataFrame Spark a Pandas para poder graficar con matplotlib/seaborn.
pdf_top = df_top.toPandas()

# 3. Creación del gráfico de barras
plt.figure(figsize=(8,5))
sns.barplot(
    x="producto",
    y="cantidad_vendida",
    color="#05053B",   # Color azul oscuro personalizado
    data=pdf_top
)
plt.title("Top 10 Productos Más Vendidos")   # Título del gráfico
plt.ylabel("Cantidad Vendida")               # Etiqueta del eje Y
plt.xticks(rotation=50)                      # Rotamos etiquetas del eje X para mejor lectura
plt.tight_layout()                           # Ajustamos el layout para evitar solapamientos
plt.show()                                   # Mostramos el gráfico
```

See performance (1) Optimize



Este gráfico presenta los diez productos con mayor volumen de ventas, según los datos consolidados en la capa Gold. Se observa que todos los productos incluidos desde artículos tecnológicos como el teclado mecánico y el monitor LG 27", hasta productos de consumo como té verde, chocolate gourmet y mermelada artesanal muestran niveles de venta similares, lo que indica una diversificación equilibrada en la demanda.

Este resultado permite a la empresa concluir que:

-No existe una concentración excesiva en un solo producto, lo que reduce el riesgo comercial.

-La oferta actual satisface distintos perfiles de cliente (hogar, tecnología, moda, alimentación).

-Se puede mantener una estrategia de inventario balanceada, sin necesidad de sobrepriorizar un único artículo.

Además, esta visualización puede servir como base para identificar oportunidades de promoción cruzada entre categorías o reforzar el posicionamiento de productos con alto margen.

databricks
Free Edition

Search data, notebooks, recents, and more... CTRL + P

workspace

New

Home
Workspace
Recents
Catalog
Jobs & Pipelines
Compute
Marketplace

SQL
SQL Editor
Queries
Dashboards
Genie
Alerts
Query History
SQL Warehouses

Data Engineering
Job Runs
Data Ingestion

AI/ML
Playground
Experiments
Features
Models

Workspace

Proyect
01_bronze_ingestion
02_silver_transformation
03_gold_aggregations
04_analisis_resultados

04_analisis_resultados

File Edit View Run Help Python Tabs: ON Last edit was 7 days ago

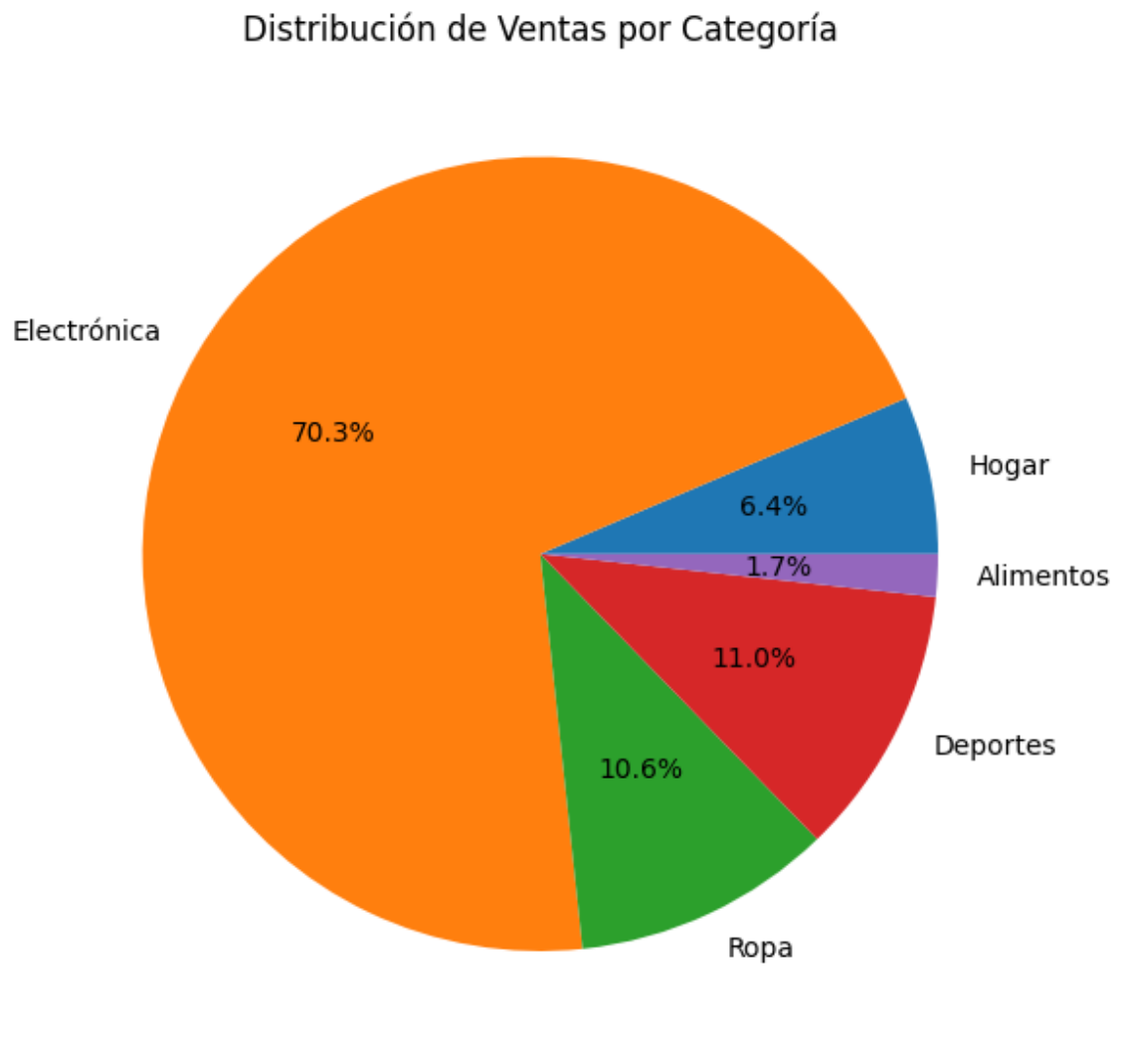
Run all Serverless Schedule Share

Dec 02, 2025 (1s) 4

```python
# ============================================================
# Visualización: Distribución de Ventas por Categoría
# ============================================================

# 1. Lectura de la tabla Gold
# Cargamos la tabla 'df_techmart_gold_ventas_por_categoria' desde el catálogo.
# Esta tabla contiene las métricas agregadas por categoría de producto.
df_categoria = spark.read.table("workspace.default.df_techmart_gold_ventas_por_categoria")

# 2. Conversión a Pandas
# Convertimos el DataFrame Spark a Pandas para poder graficar con matplotlib.
pdf_categoria = df_categoria.toPandas()

# 3. Limpieza de formato monetario
# Si la columna 'ventas_totales' está en formato string con símbolo '$',
# la transformamos a tipo numérico (float) para poder graficar correctamente.
if pdf_categoria["ventas_totales"].dtype == "object":
    pdf_categoria["ventas_totales"] = pdf_categoria["ventas_totales"].str.replace("$", "").astype(float)

# 4. Creación del gráfico de pastel
plt.figure(figsize=(6,6))
plt.pie(
    pdf_categoria["ventas_totales"],
    labels=pdf_categoria["categoria"],
    autopct="%1.1f%%"   # Muestra el porcentaje con un decimal
)
plt.title("Distribución de Ventas por Categoría")   # Título del gráfico
plt.tight_layout()                                  # Ajustamos el layout para evitar solapamientos
plt.show()                                          # Mostramos el gráfico
```

See performance (1)

Optimize

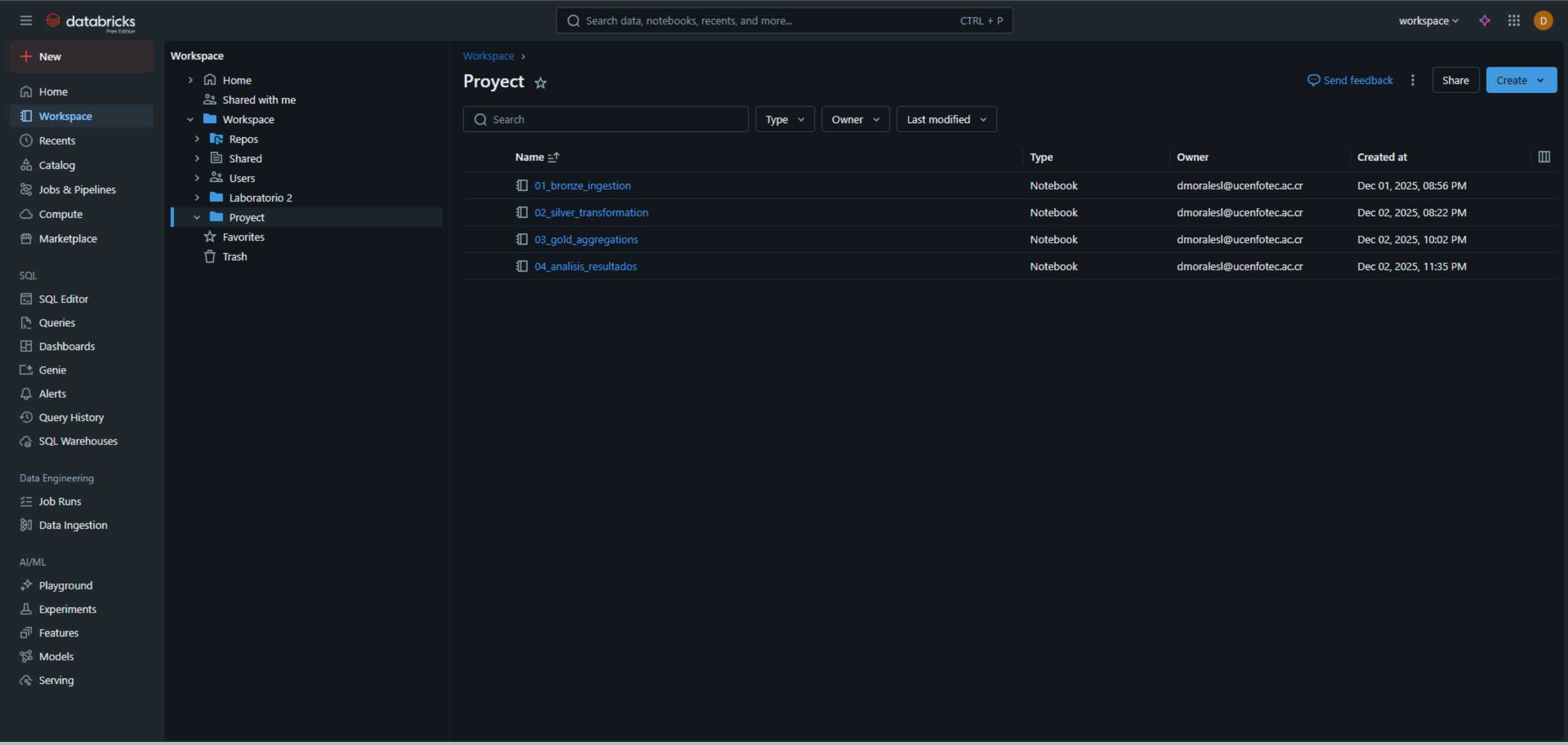


Este gráfico muestra cómo se distribuyen las ventas totales entre las distintas categorías de productos, utilizando datos consolidados de la capa Gold. Se observa una clara concentración en la categoría de Electrónica, que representa el 70.3% del total de ventas, mientras que las demás categorías Ropa, Deportes, Hogar y Alimentos tienen una participación significativamente menor.

Este resultado permite a la empresa concluir que:

- El portafolio actual está fuertemente orientado hacia productos electrónicos, lo que puede implicar una dependencia comercial de esa categoría.
- Las categorías con menor participación podrían representar oportunidades de crecimiento si se refuerzan con estrategias de marketing, surtido o posicionamiento.
- Es recomendable evaluar el margen de contribución por categoría para determinar si la concentración en Electrónica es también rentable o si conviene diversificar.

Esta visualización es clave para decisiones de inventario, campañas promocionales y análisis de riesgo comercial.



Al final del proyecto nuestro folder quedaría así

Conclusiones

La implementación del proyecto en Databricks bajo la arquitectura Medallion (Bronze, Silver y Gold) permitió estructurar el flujo de datos de manera clara y escalable. Cada capa cumplió su función:

- Bronze: almacenamiento crudo y confiable de la información.
- Silver: limpieza y estandarización para asegurar calidad.
-Gold: datos listos para análisis y visualización.

El Job en Databricks se ejecutó correctamente, con todas las tareas en estado “Succeeded” (verde), lo que confirma que el ETL quedó operativo y funcional. Cabe destacar que, al trabajar en la versión gratuita (Community Edition), la ejecución debe realizarse manualmente, ya que no es posible configurar tiempos de ejecución programada ni triggers automáticos.

Las visualizaciones generadas con Pandas a partir de la capa Gold aportan hallazgos clave para la empresa:

- Ventas por sucursal: desempeño equilibrado entre sedes, sin diferencias críticas.
- Tendencia mensual: diciembre 2025 muestra un volumen de ventas elevado, base para análisis estacionales.
- Top 10 productos más vendidos: demanda diversificada, sin concentración excesiva en un solo artículo.
-Distribución por categoría: predominio de la electrónica (70.3%), lo que evidencia fortaleza comercial pero también dependencia de un único segmento.

En conjunto, el proyecto demuestra que un pipeline bien diseñado no solo garantiza la integridad y disponibilidad de los datos, sino que también habilita la toma de decisiones estratégicas en áreas como inventario, marketing y expansión comercial.