

# Pipeline ETL con Arquitectura Medallión: Bike Store Relacional

## Proyecto Ingeniería de Datos en Databricks

Erickson Claudio Otaño Sánchez  
31 de marzo de 2025



# Introducción

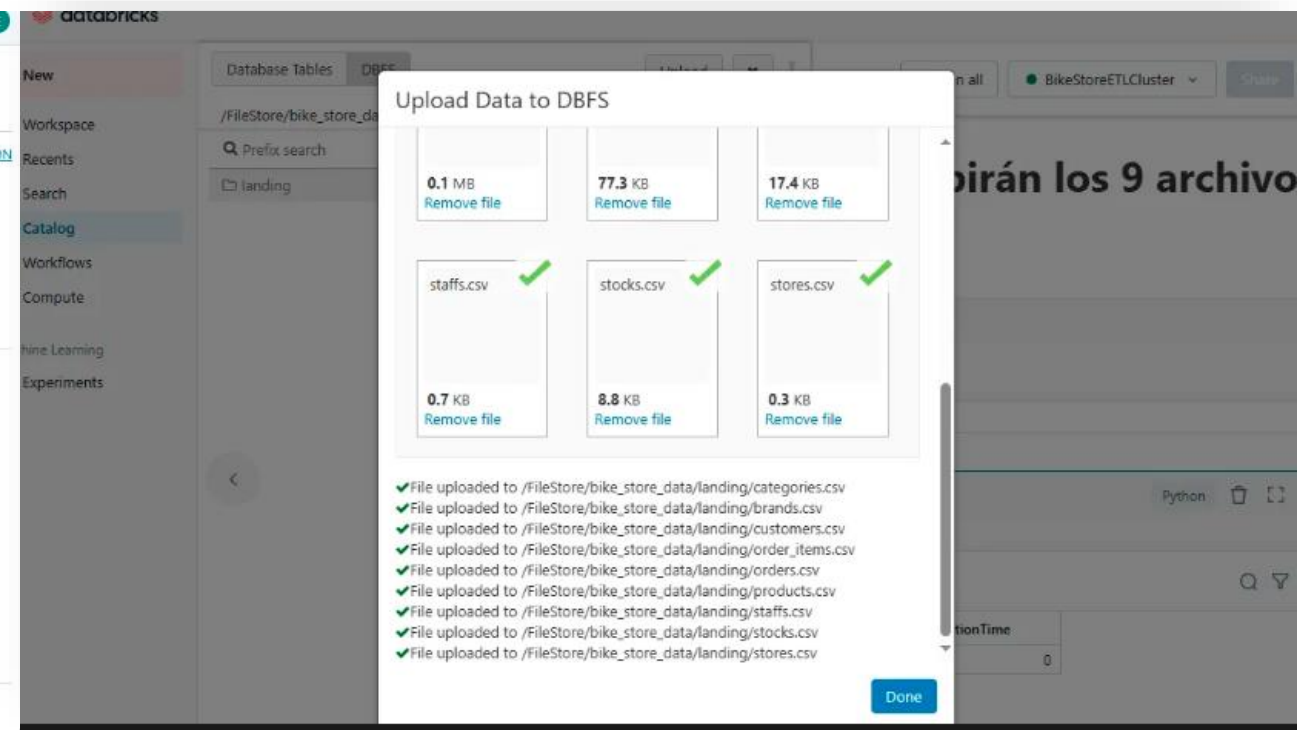
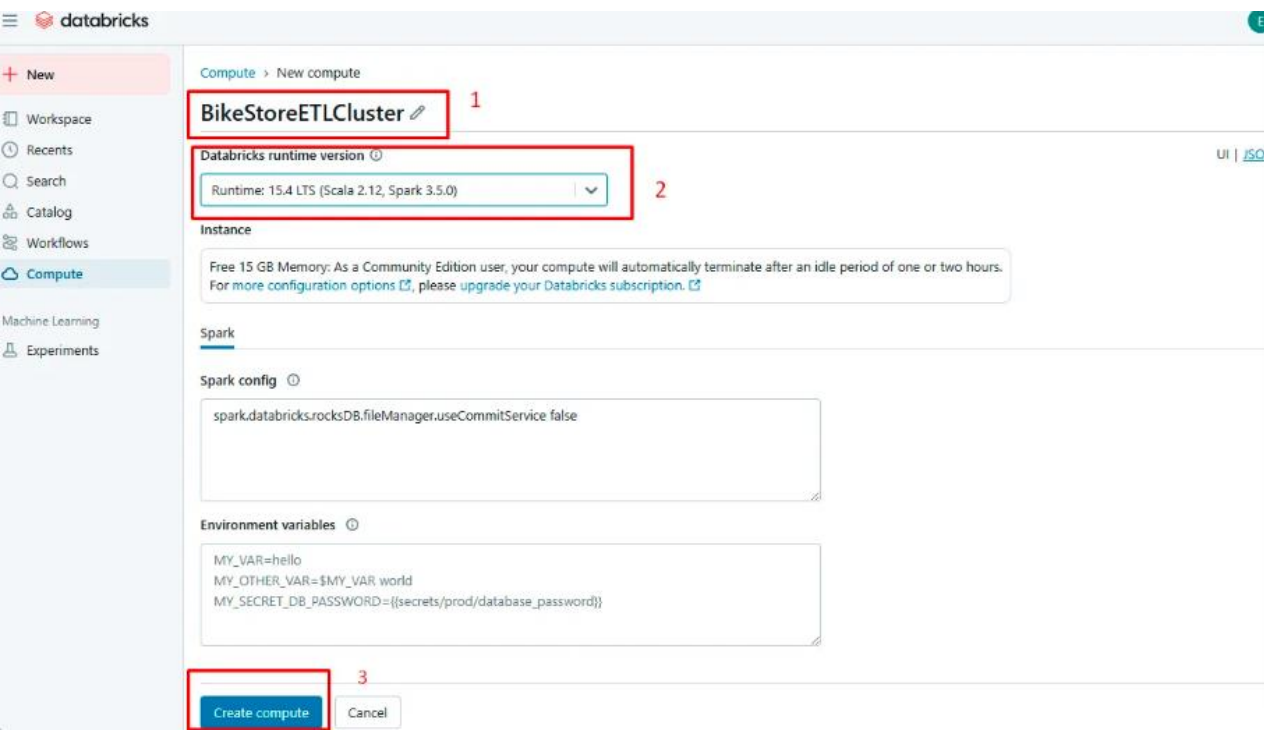
## Objetivo del proyecto:

- Construir un pipeline ETL en Databricks Community Edition usando la arquitectura Medallión (Bronze, Silver, Gold) con el dataset "Bike Store Relacional".
- Herramientas utilizadas: Databricks, PySpark, Delta Lake, Notion.Contexto: Proyecto educativo para aprender conceptos de ingeniería de datos.



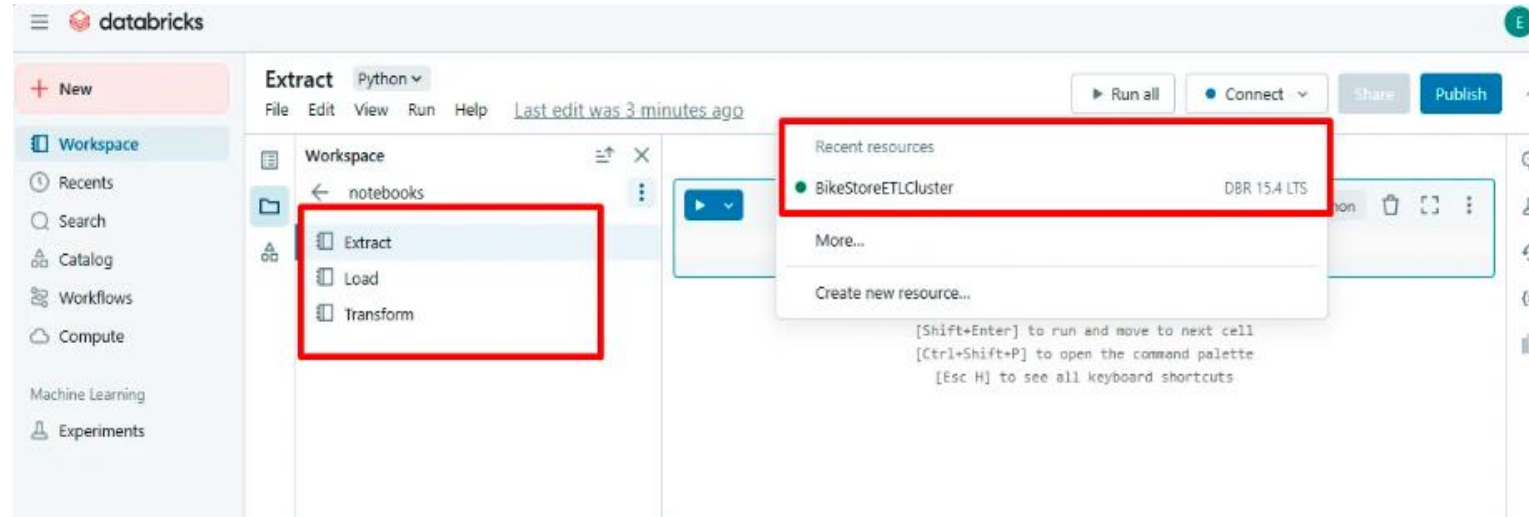
# Metodología: Configuración Inicial

- Creación del clúster BikeStoreETLCluster (Databricks Runtime 15.4 LTS (Scala 2.12, Spark 3.5.0). Memoria de 15.3 GB, 2 Núcleos). Subida de los 9 archivos CSV a dbfs:/FileStore/bike\_store\_data/landing/. Verificación con %fs ls.



# Metodología - Capa Bronze

- Creación de la carpeta bronze y organización de notebooks (Extract.ipynb, Transform.ipynb, Load.ipynb).
- Ingestión de datos desde landing a bronze usando StructType para definir esquemas.
- Adición de metadatos (ingestion\_date).
- Verificación de datos en bronze.



## Definición de esquemas con StructType

```
from pyspark.sql.types import StructType, StructField, IntegerType, StringType, DoubleType, DateType
from pyspark.sql.functions import current_timestamp

# Esquema para categories.csv
categories_schema = StructType([
    StructField("category_id", IntegerType(), False),
    StructField("category_name", StringType(), False)
])
```



# Metodología - Capa Bronze

## Exploración de datos en Bronze

< > + Code + Text

Just now (1s) 14 Python

```
df_categories_bronze = spark.read.format("delta").load("dbfs:/FileStore/bike_store_data/bronze/categories")
df_categories_bronze.limit(5).display()
```

▶ (1) Spark Jobs

▶ df\_categories\_bronze: pyspark.sql.dataframe.DataFrame = [category\_id: integer, category\_name: string ... 1 more field]

Table +

	category_id	category_name	ingestion_date
1	1	Children Bicycles	2025-03-31T02:07:32.974+00:...
2	2	Comfort Bicycles	2025-03-31T02:07:32.974+00:...
3	3	Cruisers Bicycles	2025-03-31T02:07:32.974+00:...
4	4	Cyclocross Bicycles	2025-03-31T02:07:32.974+00:...
5	5	Electric Bikes	2025-03-31T02:07:32.974+00:...

5 rows | 0.61s runtime Refreshed now

# Metodología: Capa Silver

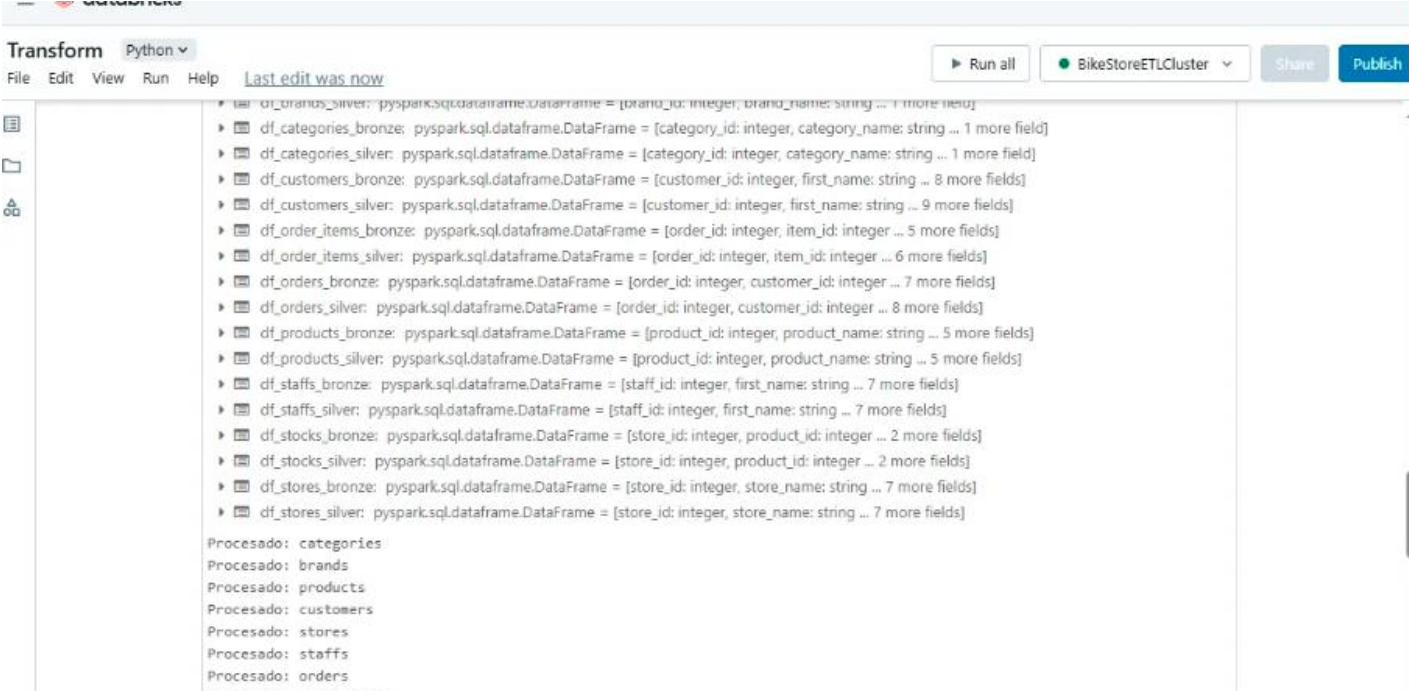
- Creación de la carpeta silver.
- Transformaciones: manejo de nulos (rellenar phone, email, shipped\_date con "N/A"), estandarización (nombres en mayúsculas), columnas calculadas (total\_price, order\_status\_desc en español).
- Guardado de tablas en silver como Delta.Verificación y exploración de datos.



The screenshot shows the Databricks Transform interface. At the top, there's a menu bar with 'File', 'Edit', 'View', 'Run', and 'Help'. Below it, a status bar indicates 'Last edit was now'. On the right, there are buttons for 'Run all', 'BikeStoreETLCluster', and 'Share'. The main area displays a code editor with a Python script. The script imports necessary functions from pyspark.sql.functions and reads data from a Delta table in the bronze layer. It then performs transformations to create a silver table, including handling nulls and standardizing names.

```
from pyspark.sql.functions import col, upper, when, lit, concat_ws

# categories
df_categories_bronze = spark.read.format("delta").load("dbfs:/FileStore/bike_store_data/bronze/categories")
df_categories_silver = df_categories_bronze \
    .withColumn("category_name", upper(col("category_name"))) \
    .dropDuplicates(["category_id"])
```



The screenshot shows the Databricks Transform interface with a list of DataFrames and their schemas. The list includes various tables in the bronze and silver layers, such as df\_brands\_silver, df\_categories\_bronze, df\_categories\_silver, df\_customers\_bronze, df\_customers\_silver, df\_order\_items\_bronze, df\_order\_items\_silver, df\_orders\_bronze, df\_orders\_silver, df\_products\_bronze, df\_products\_silver, df\_staffs\_bronze, df\_staffs\_silver, df\_stocks\_bronze, df\_stocks\_silver, df\_stores\_bronze, and df\_stores\_silver. Each entry shows the DataFrame name and its schema, including column names and data types. Below the list, there's a section titled 'Procesado:' followed by a list of processed tables: categories, brands, products, customers, stores, staffs, orders, and order\_items.

```
df_brands_silver: pyspark.sql.dataframe.DataFrame = [brand_id: integer, brand_name: string ... 1 more field]
df_categories_bronze: pyspark.sql.dataframe.DataFrame = [category_id: integer, category_name: string ... 1 more field]
df_categories_silver: pyspark.sql.dataframe.DataFrame = [category_id: integer, category_name: string ... 1 more field]
df_customers_bronze: pyspark.sql.dataframe.DataFrame = [customer_id: integer, first_name: string ... 8 more fields]
df_customers_silver: pyspark.sql.dataframe.DataFrame = [customer_id: integer, first_name: string ... 9 more fields]
df_order_items_bronze: pyspark.sql.dataframe.DataFrame = [order_id: integer, item_id: integer ... 5 more fields]
df_order_items_silver: pyspark.sql.dataframe.DataFrame = [order_id: integer, item_id: integer ... 6 more fields]
df_orders_bronze: pyspark.sql.dataframe.DataFrame = [order_id: integer, customer_id: integer ... 7 more fields]
df_orders_silver: pyspark.sql.dataframe.DataFrame = [order_id: integer, customer_id: integer ... 8 more fields]
df_products_bronze: pyspark.sql.dataframe.DataFrame = [product_id: integer, product_name: string ... 5 more fields]
df_products_silver: pyspark.sql.dataframe.DataFrame = [product_id: integer, product_name: string ... 5 more fields]
df_staffs_bronze: pyspark.sql.dataframe.DataFrame = [staff_id: integer, first_name: string ... 7 more fields]
df_staffs_silver: pyspark.sql.dataframe.DataFrame = [staff_id: integer, first_name: string ... 7 more fields]
df_stocks_bronze: pyspark.sql.dataframe.DataFrame = [store_id: integer, product_id: integer ... 2 more fields]
df_stocks_silver: pyspark.sql.dataframe.DataFrame = [store_id: integer, product_id: integer ... 2 more fields]
df_stores_bronze: pyspark.sql.dataframe.DataFrame = [store_id: integer, store_name: string ... 7 more fields]
df_stores_silver: pyspark.sql.dataframe.DataFrame = [store_id: integer, store_name: string ... 7 more fields]

Procesado: categories
Procesado: brands
Procesado: products
Procesado: customers
Procesado: stores
Procesado: staffs
Procesado: orders
Procesado: order_items
```

# Metodología: Capa Silver

The screenshot shows a Databricks workspace interface. At the top, there's a 'Transform' tab with a 'Python' dropdown. Below it, a table displays data from a Delta table. The table has 9 rows and 6 columns. The first column contains row IDs (7, 8, 9), the second column contains the full path to the Silver layer data, and the third column contains the table name. The last two columns contain counts of rows. Below the table, it says '9 rows | 1.52s runtime' and 'Refreshed now'.

7	dbfs/FileStore/bike_store_data/silver/staffs/	staffs/	0	0
8	dbfs/FileStore/bike_store_data/silver/stocks/	stocks/	0	0
9	dbfs/FileStore/bike_store_data/silver/stores/	stores/	0	0

9 rows | 1.52s runtime Refreshed now

## Exploración de datos en Silver

Leer tablas Silver y mostrar su contenido

```
df_categories_silver = spark.read.format("delta").load("dbfs:/FileStore/bike_store_data/silver/categories")
df_categories_silver.show(5)
print(f"Filas en categories: {df_categories_silver.count()}")

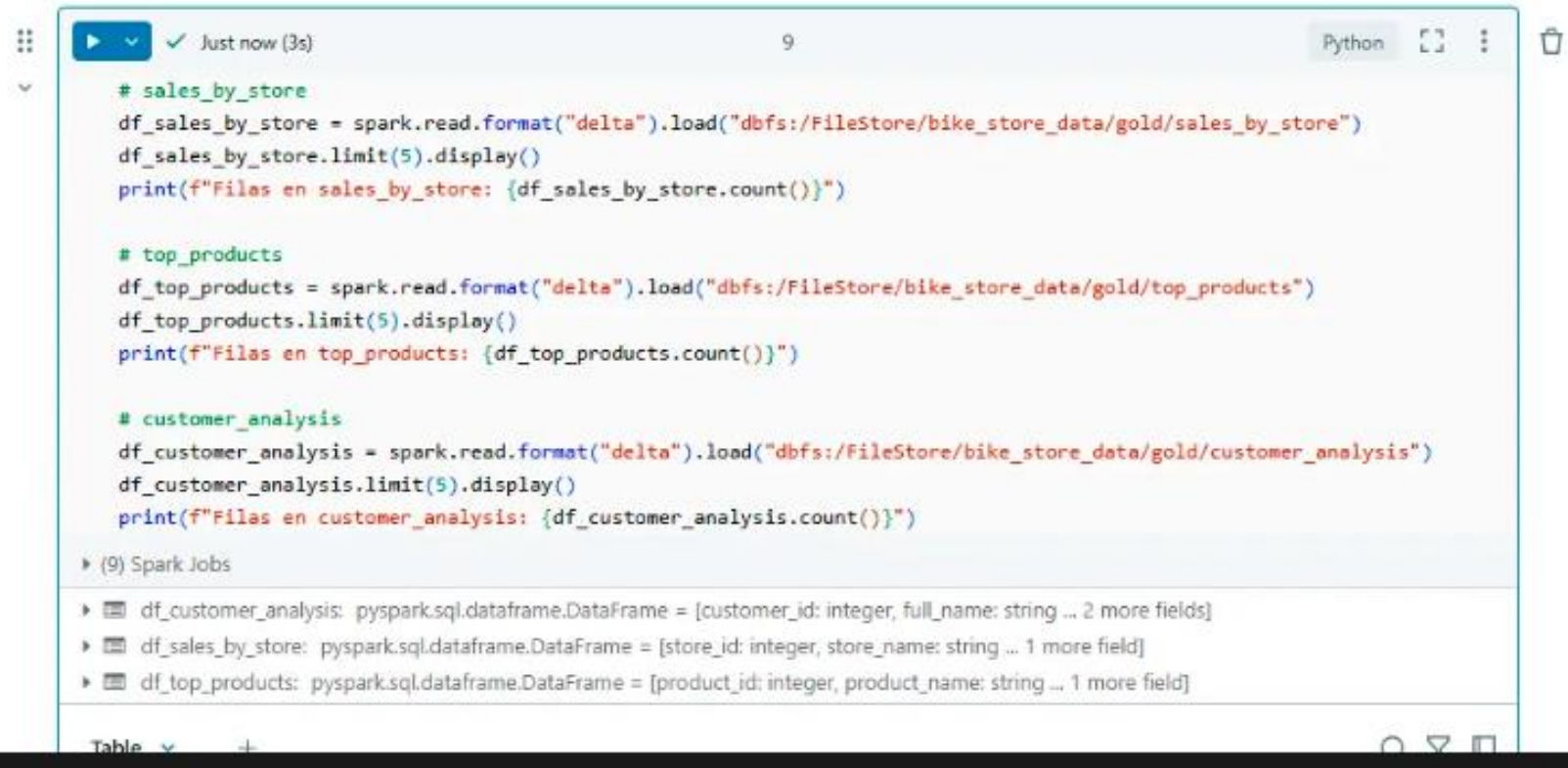
# brands
df_brands_silver = spark.read.format("delta").load("dbfs:/FileStore/bike_store_data/silver/brands")
df_brands_silver.show(5)
print(f"Filas en brands: {df_brands_silver.count()}")

# products
df_products_silver = spark.read.format("delta").load("dbfs:/FileStore/bike_store_data/silver/products")
```

# Metodología - Capa Gold

- Creación de la carpeta gold.
- Tablas analíticas:sales\_by\_store: Ventas totales por tienda.
- Top\_products: Productos más vendidos.
- Customer\_analysis: Número de pedidos y gasto total por cliente.
- Guardado de tablas en gold como Delta.
- Verificación y exploración de datos.

## Exploración de datos en Gold



```
Just now (3s) 9 Python
```

```
# sales_by_store
df_sales_by_store = spark.read.format("delta").load("dbfs:/FileStore/bike_store_data/gold/sales_by_store")
df_sales_by_store.limit(5).display()
print(f"Filas en sales_by_store: {df_sales_by_store.count()}")

# top_products
df_top_products = spark.read.format("delta").load("dbfs:/FileStore/bike_store_data/gold/top_products")
df_top_products.limit(5).display()
print(f"Filas en top_products: {df_top_products.count()}")

# customer_analysis
df_customer_analysis = spark.read.format("delta").load("dbfs:/FileStore/bike_store_data/gold/customer_analysis")
df_customer_analysis.limit(5).display()
print(f"Filas en customer_analysis: {df_customer_analysis.count()}")
```

▶ (9) Spark Jobs

▶ df\_customer\_analysis: pyspark.sql.dataframe.DataFrame = [customer\_id: integer, full\_name: string ... 2 more fields]

▶ df\_sales\_by\_store: pyspark.sql.dataframe.DataFrame = [store\_id: integer, store\_name: string ... 1 more field]

▶ df\_top\_products: pyspark.sql.dataframe.DataFrame = [product\_id: integer, product\_name: string ... 1 more field]

Table x +



# Metodología - Capa Gold

- Creación de la carpeta gold.
- Tablas analíticas:sales\_by\_store: Ventas totales por tienda.
- Top\_products: Productos más vendidos.
- Customer\_analysis: Número de pedidos y gasto total por cliente.
- Guardado de tablas en gold como Delta.
- Verificación y exploración de datos.

df\_top\_products: pyspark.sql.dataframe.DataFrame = [product\_id: integer, product\_name: string ... 1 more field]

Table ▾ +

	1.2 store_id	1.2 store_name	1.2 total_sales
1	2	BALDWIN BIKES	5215751.277499983
2	1	SANTA CRUZ BIKES	1605823.03649999...
3	3	ROWLETT BIKES	867542.2436000014

3 rows | 2.64s runtime

Refreshed 2 minutes ago

Filas en sales\_by\_store: 3

Table ▾ +

	1.2 product_id	1.2 product_name	1.2 total_quantity_sold
1	6	SURLY ICE CREAM TRUCK FRAMESET - 2016	167
2	13	ELECTRA CRUISER 1 (24-INCH) - 2016	157
3	16	ELECTRA TOWNIE ORIGINAL 7D EQ - 2016	156
4	23	ELECTRA GIRL'S HAWAII 1 (20-INCH) - 2015/2016	154
5	7	TREK SLASH 8 27.5 - 2016	154

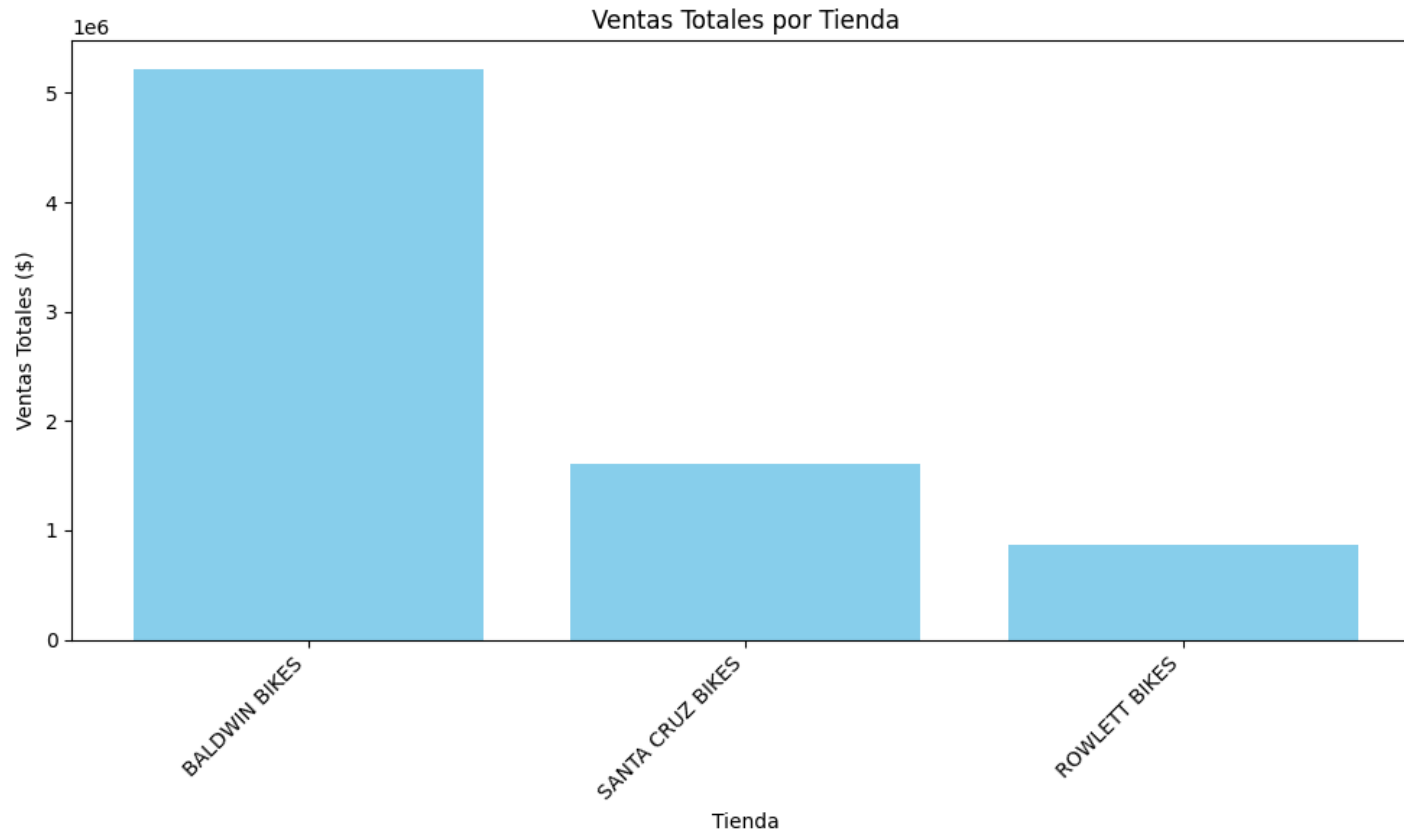
5 rows | 2.64s runtime

Refreshed 1 minute ago

# Visualización de Datos

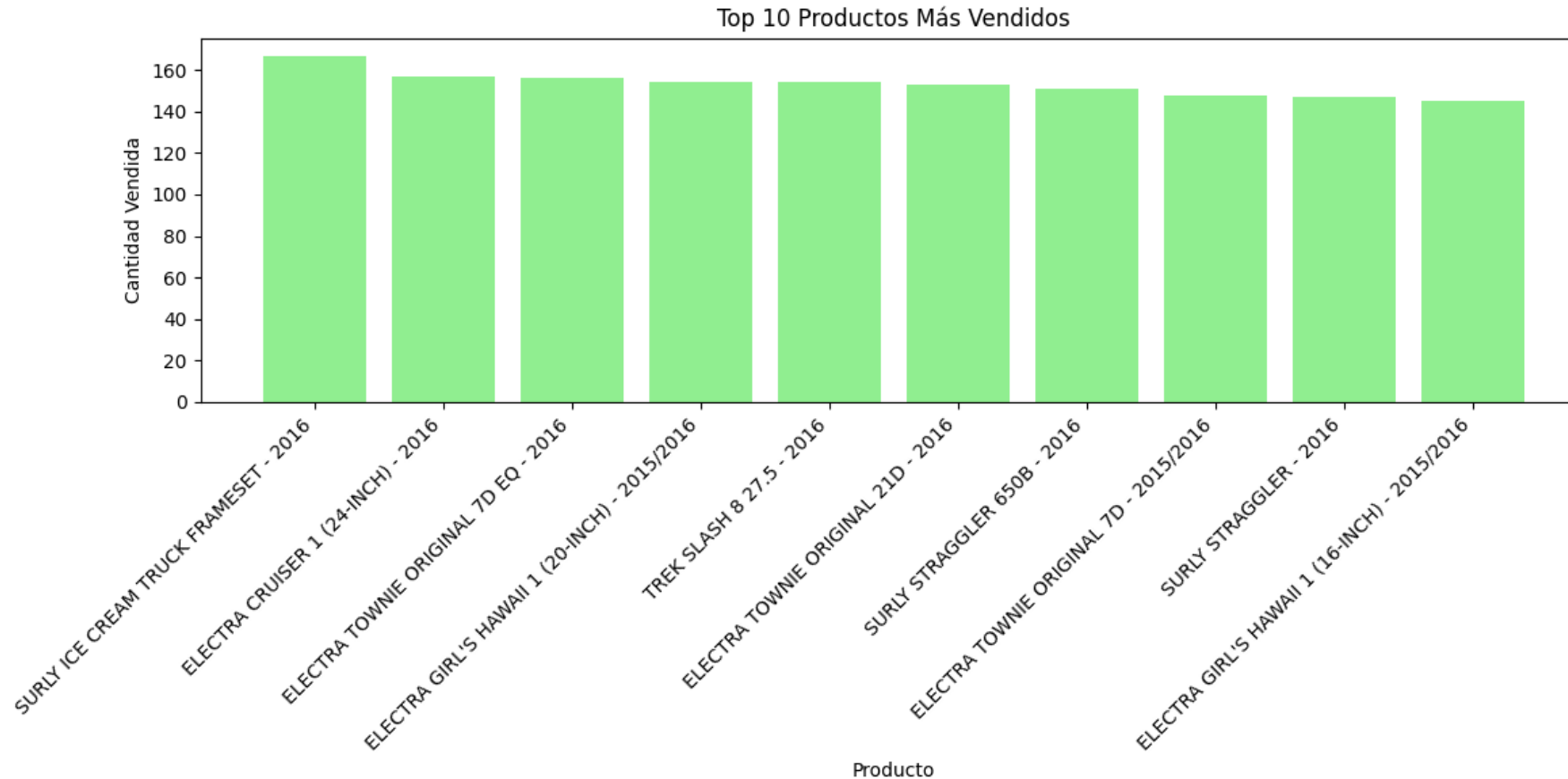
- Gráfico 1: Ventas totales por tienda (gráfico de barras).
- Gráfico 2: Top 10 productos más vendidos (gráfico de barras).
- Gráfico 3: Relación entre pedidos y gasto por cliente (gráfico de dispersión).

# Gráfico 1: Ventas totales por tienda (gráfico de barras)



Observación: Baldwin Bikes lidera con más de 5M en ventas.

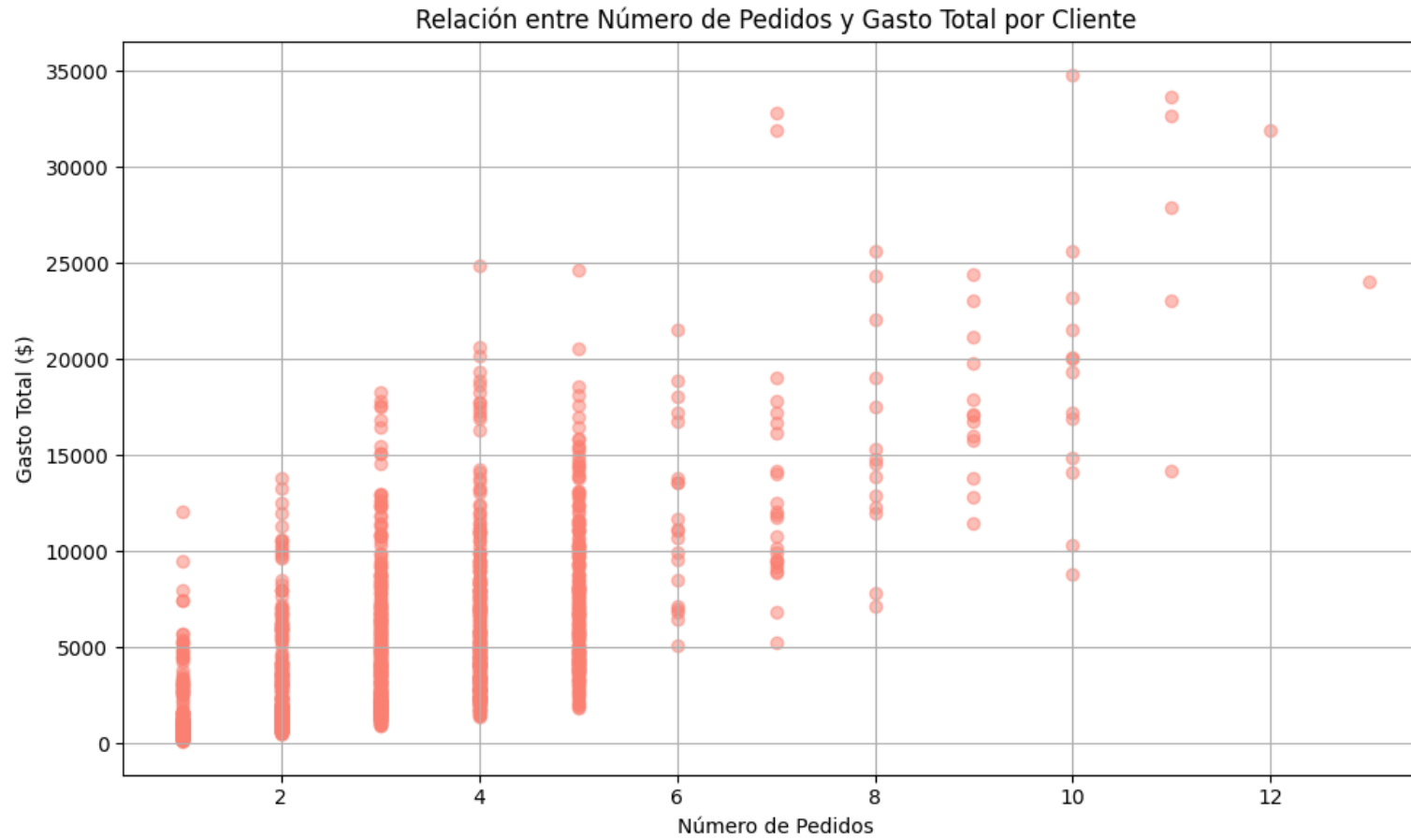
## Gráfico 2: Top 10 productos más vendidos (gráfico de barras)



Observación: Electra domina con modelos como Surly Ice Cream Truck.



# Gráfico 3: Relación entre pedidos y gasto por cliente (gráfico de dispersión)



Observación: Clientes con 8-12 pedidos gastan hasta 35,000 dólares

# Conclusiones del Análisis

- Baldwin Bikes es la tienda más exitosa, con oportunidades para replicar su estrategia.
- La marca Electra lidera las ventas de productos, sugiriendo un enfoque en esta marca.
- Hay clientes de alto valor (8-12 pedidos, hasta 35,000 dólares) que representan una oportunidad de fidelización.
- Recomendaciones: Equilibrar ventas entre tiendas, diversificar productos, y segmentar clientes.

# Automatizar el pipeline con workflows.

- En Databricks Community Edition, la funcionalidad de Workflows requiere un plan de pago, por lo que opté por una alternativa.
- Creé un nuevo notebook **PipelineOrchestrator.ipynb** en **BikeStoreProject/notebooks/**.
- Usé el comando %run para ejecutar los notebooks Extract.ipynb, Transform.ipynb, y Load.ipynb en secuencia.
- Verifiqué que el pipeline se ejecutara correctamente, confirmando que las carpetas bronze, silver, y gold contengan las tablas esperadas.
- Exploré una tabla Gold (sales\_by\_store) para confirmar los resultados.

## Confirmar Resultados del Pipeline

▶ ✓ 1 minute ago (1s) 6

```
%fs ls dbfs:/FileStore/bike_store_data/bronze
```

Table +

	<sup>A</sup> <sub>C</sub> path	<sup>A</sup> <sub>C</sub> name	<sup>1</sup> <sub>3</sub> size	<sup>1</sup> <sub>3</sub> modification
1	dbfs:/FileStore/bike_store_data/bronze/brands/	brands/	0	
2	dbfs:/FileStore/bike_store_data/bronze/categories/	categories/	0	
3	dbfs:/FileStore/bike_store_data/bronze/customers/	customers/	0	
4	dbfs:/FileStore/bike_store_data/bronze/order_items/	order_items/	0	
5	dbfs:/FileStore/bike_store_data/bronze/orders/	orders/	0	
6	dbfs:/FileStore/bike_store_data/bronze/products/	products/	0	
7	dbfs:/FileStore/bike_store_data/bronze/staffs/	staffs/	0	
8	dbfs:/FileStore/bike_store_data/bronze/stocks/	stocks/	0	
9	dbfs:/FileStore/bike_store_data/bronze/stores/	stores/	0	

↓ 9 rows | 0.68s runtime

# Automatizar el pipeline con workflows

▶

✓ 2 minutes ago (2s)

9

Python

🗑️

🔍

⋮

```
df_sales_by_store = spark.read.format("delta").load("dbfs://FileStore/bike_store_data/gold/sales_by_store")
df_sales_by_store.limit(5).display()
print(f"Filas en sales_by_store: {df_sales_by_store.count()}")
```

▶ (5) Spark Jobs

▶ df\_sales\_by\_store: pyspark.sql.dataframe.DataFrame = [store\_id: integer, store\_name: string ... 1 more field]

Table

+

🔍

🔍

🔍

	store_id	store_name	total_sales
1	2	BALDWIN BIKES	5215751.277499983
2	1	SANTA CRUZ BIKES	1605823.03649999...
3	3	ROWLETT BIKES	867542.2436000014

↓

3 rows | 1.81s runtime

Refreshed 2 minutes ago

Filas en sales\_by\_store: 3

Muestra de resultados del pipeline consultando la tabla gold “sales\_by\_stores



# Conclusiones y Aprendizajes

- **Logros:**

- Pipeline ETL completo con arquitectura Medallión.
- Tablas analíticas listas para reportes.

- **Aprendizajes:**

- Uso de Databricks y PySpark para ETL.
- Importancia de la arquitectura Medallión.
- Uso de Workflow para automatizar pipeline
- Manejo de datos con Delta Lake.



# Agradecimientos

- Smart Data
- Profesor Daniel Santos
- Compañeros