



# Proyecto final Bike Store

## Introducción al Proyecto

[Descripción del Proyecto](#)

[Problema](#)

[Solución Propuesta](#)

[Arquitectura del Proyecto](#)

## Descripción de los Datos

[Descripción de los Datos](#)

### Preparación inicial

[Descarga de la Data](#)

[Configuración de Databricks Community Edition](#)

[Creación del Clúster](#)

[Habilitar DBFS y crear notebook](#)

[Subir archivos CSV a DBFS](#)

### Implementación paso a paso

[Organización de notebooks y movimiento a la capa Bronze](#)

[Proceso de Extract de landing a Bronze](#)

[Verificar contenido en carpeta landing y definición de esquemas con structtype](#)

[Transformación de datos a la capa Silver](#)

[Carga de datos a la capa Gold](#)

[Explicación del código](#)

### Visualización de Datos y conclusiones de análisis

[Visualización de Datos en la Capa Gold con Matplotlib](#)

[Captura 1: Gráfico de Ventas por Tienda](#)

[Captura 2: Gráfico de Productos Más Vendidos](#)

[Captura 3: Gráfico de Análisis de Clientes](#)  
[Conclusiones del Análisis](#)  
[Automatizar el pipeline con workflows](#)  
[Creación del Workflow](#)  
[Confirmando resultados del workflow](#)  
[Resumen del Proyecto](#)  
[Entregables](#)

---

# Introducción al Proyecto

## Descripción del Proyecto

Este proyecto tiene como objetivo construir un pipeline de procesamiento de datos utilizando la arquitectura Medallón (Bronze, Silver, Gold) en Databricks Community Edition, con dataset "Bike Store Relational Database" de Kaggle. A través de este ejercicio, se aplicarán técnicas avanzadas de ETL (Extracción, Transformación y Carga) utilizando PySpark, comandos mágicos de SQL, RDD y joins, con el fin de transformar datos crudos en información valiosa para análisis.

## Problema

En una tienda como en este caso de ventas de bicletas, las empresas necesitan procesar grandes volúmenes de datos de manera eficiente para obtener insights accionables. En este caso, la tienda de bicicletas genera datos de ventas, clientes y productos que, aunque estén relativamente limpios, requieren un flujo estructurado para integrarlos, enriquecerlos y prepararlos para reporting. El desafío es implementar un pipeline robusto que simule un caso real, aplicando lo aprendido en el curso.

## Solución Propuesta

En este proyecto se ha estructurado un pipeline ETL basado en la arquitectura Medallón:

- **Bronze:** Ingesta de datos sin cambios en el origen llevándolo a formato Delta.
- **Silver:** Aunque desde Kaggle la data está limpia, se realizan transformaciones y simulación de limpieza de datos para poner en práctica lo aprendido en el curso.

- **Gold:** Creación de tablas agregadas para análisis y visualización. El proyecto usará PySpark y SQL en Databricks para procesar los datos, incluyendo cargas incrementales y joins, y se termina con una visualización de los resultados.

## Arquitectura del Proyecto

- Explicación de la arquitectura Medallón (Bronze, Silver, Gold)

El proyecto sigue la arquitectura Medallón, un enfoque en tres capas:

1. **Bronze:** Capa de datos crudos. Aquí se ingestan los archivos CSV de Bike Store sin transformaciones, preservando su formato original y añadiendo metadatos como la fecha de ingesta.
2. **Silver:** Capa de datos transformados. Se aplican transformaciones como normalización, limpieza simulada (duplicados o nulos), y enriquecimiento (columnas calculadas). También se simulan cargas incrementales.
3. **Gold:** Capa de datos listos para análisis. Se generan tablas agregadas y optimizadas (por ejemplo, ventas por mes o por cliente) para reporting y visualización.

Este enfoque garantiza un flujo estructurado, escalable y reusable, ideal para entornos de big data.

- Diagrama

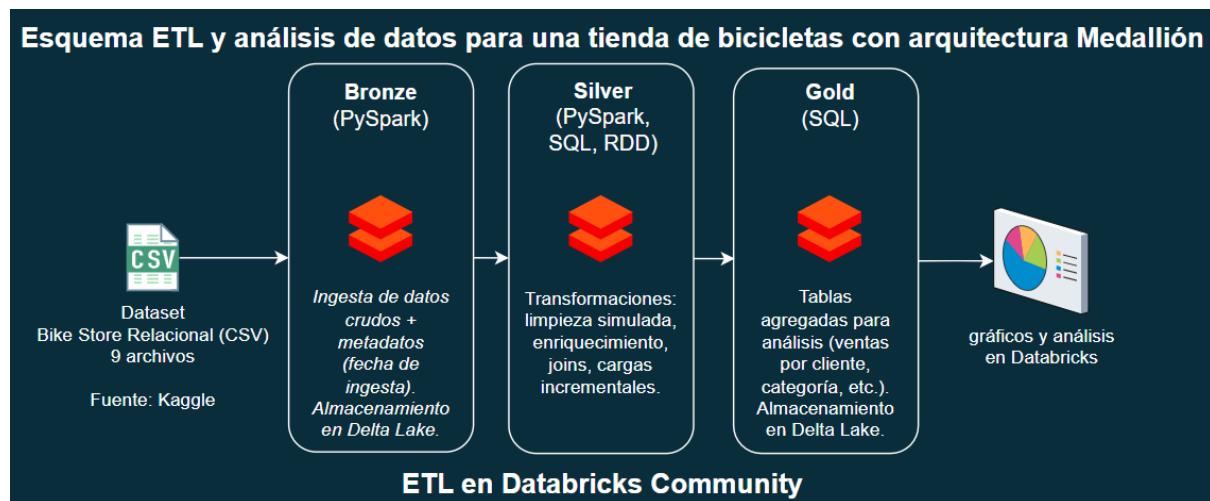


Diagrama que muestra el esquema del proyecto utilizando la arquitectura Medallón para el proceso de ETL en Databricks Community

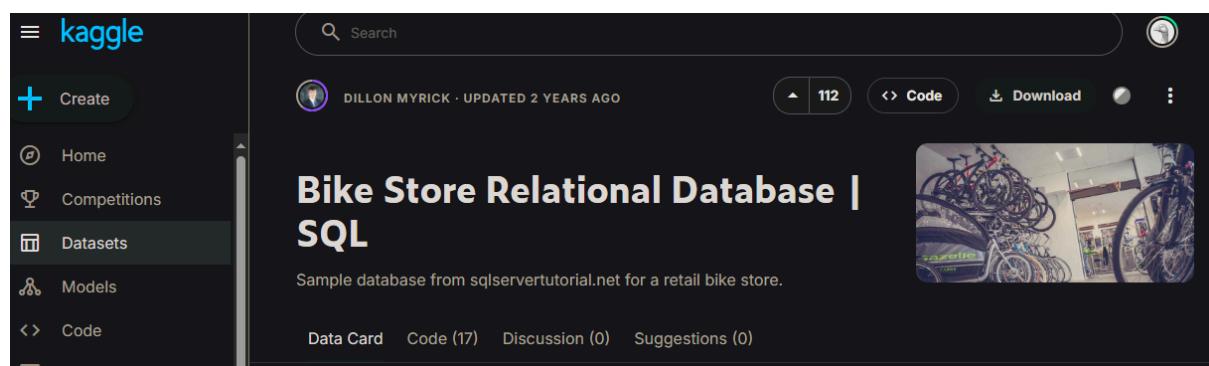
# Descripción de los Datos

## Descripción de los Datos

El dataset "Bike Store Relational Database" proviene de Kaggle (<https://www.kaggle.com/dillonmyrick/bike-store-sample-database>) y es una muestra de una base de datos relacional de una tienda de bicicletas, basada en un tutorial de SQL Server. Incluye las siguientes tablas en formato CSV:

- **categories.csv**: Categorías de productos (ID, nombre).
- **brands.csv**: Marcas de bicicletas (ID, nombre).
- **products.csv**: Productos (ID, nombre, categoría, marca, precio).
- **customers.csv**: Clientes (ID, nombre, email, dirección).
- **stores.csv**: Tiendas (ID, nombre, ubicación).
- **staffs.csv**: Empleados (ID, nombre, tienda).
- **orders.csv**: Pedidos (ID, cliente, tienda, fecha, estado).
- **order\_items.csv**: Detalles de pedidos (ID de pedido, producto, cantidad, precio).
- **stocks.csv**: Inventario (tienda, producto, cantidad).

Aunque los datos están limpios, se simularán transformaciones para aplicar conceptos avanzados como RDD y cargas incrementales.



**Bike Store Relational Database | SQL**

Data Card Code (17) Discussion (0) Suggestions (0)

**brands.csv** (120 B)

Detail Compact Column 2 of 2 columns

**About this file** Suggest Edits

The brands table stores the brand's information of bikes, for example, Electra, Haro, and Heller.

brand_id	brand_name
9	9
total values	unique values
1	Electra
2	Haro
3	Heller

**Data Explorer**  
345.4 kB

- brands.csv
- categories.csv
- customers.csv
- order\_items.csv
- orders.csv
- products.csv
- staffs.csv
- stocks.csv
- stores.csv

**Summary**

- 9 files
- 52 columns

Captura que muestra los dataset de Bike store relational Database en Kaggle

# Preparación inicial

## Descarga de la Data

1. Accedí a Kaggle y busqué el dataset Bike Store Relational Database (<https://www.kaggle.com/dillonmyrick/bike-store-sample-database>).
2. Revisé la descripción y los 9 archivos CSV disponibles.
3. Descargué los archivos y los guardé en la carpeta BikeStoreData en mi computadora.

	Nombre	Fecha de modificación	Tipo	Tamaño
so rápido	brands	21/08/2023 14:31	Microsoft Excel C...	1 KB
cargas	categories	21/08/2023 14:31	Microsoft Excel C...	1 KB
Datos (D:)	customers	21/08/2023 14:31	Microsoft Excel C...	125 KB
BA y Analític	order_items	21/08/2023 14:31	Microsoft Excel C...	111 KB
ingles_Aprend	orders	21/08/2023 14:31	Microsoft Excel C...	76 KB
NO J J J b	products	21/08/2023 14:31	Microsoft Excel C...	17 KB
NICAS DE ES	staffs	21/08/2023 14:31	Microsoft Excel C...	1 KB
aEngineering	stocks	21/08/2023 14:31	Microsoft Excel C...	9 KB
abricks Data	stores	21/08/2023 14:31	Microsoft Excel C...	1 KB

```

order_id,customer_id,order_status,or
1,259,4,2016-01-01,2016-01-03,2016-
2,1212,4,2016-01-01,2016-01-04,2016-
3,523,4,2016-01-02,2016-01-05,2016-
4,175,4,2016-01-03,2016-01-06,2016-
5,1324,4,2016-01-03,2016-01-06,2016-
6,94,4,2016-01-04,2016-01-07,2016-01-
7,324,4,2016-01-04,2016-01-07,2016-0
8,1204,4,2016-01-04,2016-01-05,2016-
9,60,4,2016-01-05,2016-01-08,2016-01-
10,442,4,2016-01-05,2016-01-08,2016-
11,1326,4,2016-01-05,2016-01-08,2016-
12,91,4,2016-01-06,2016-01-08,2016-0
13,873,4,2016-01-08,2016-01-11,2016-
14,298,4,2016-01-09,2016-01-11,2016-
15,450,4,2016-01-09,2016-01-10,2016-
16,552,4,2016-01-12,2016-01-15,2016-
17,1175,4,2016-01-12,2016-01-14,2016-
18,541,4,2016-01-14,2016-01-17,2016-

```

Captura de pantalla de los 9 archivos descargados en mi PC desde la web kaggle

## Configuración de Databricks Community Edition

1. Accedí a Databricks Community Edition (<https://community.cloud.databricks.com/>) e inicié sesión.
2. Llegué al panel principal, donde puedo crear un clúster, habilitar el sistema de archivos DBFS, crear un workspace, subir los datos de Bike Store y crear notebooks para el proceso ETL.

The screenshot shows the Databricks main interface. On the left, there's a sidebar with options like 'New', 'Workspace', 'Recents', 'Search', 'Catalog', 'Workflows', 'Compute', 'Machine Learning', and 'Experiments'. The main area is titled 'Welcome to Databricks' and shows a 'Recents' section with four notebook thumbnails: 'Load csv', 'Transform csv', '3. Dataframes y Datasets en Databricks', and '2. Fundamentos\_Spark\_Funciones\_Avanzadas', all created 22 hours ago and categorized as 'Notebook'.

Captura del panel principal de Databricks Community Edition, mostrando las opciones disponibles para crear clústeres, workspaces y notebooks.

## Creación del Clúster

1. Creé un clúster en Databricks para procesar los datos de Bike Store:
  - Accedí al panel de "Compute" desde el menú lateral izquierdo.
  - Hice clic en "Create Cluster" para configurar un nuevo clúster.
  - Configuré el clúster con el siguiente nombre: `BikeStoreETLCluster`.
  - Seleccioné la versión de Databricks Runtime disponible Runtime 15.4 LTS (Scala 2.12, Spark 3.5.0). Memoria de 15.3 GB, 2 Núcleos.

- Hice clic en "Create Cluster" y esperé a que el estado cambiara a "Running".

The screenshot shows the Databricks Compute interface. On the left sidebar, the 'Compute' option is highlighted with a red box and labeled '1'. A red arrow points from this box to the 'Create compute' button at the bottom right of the main area, which is also labeled '2'. The main area displays a 'No compute' message with a large plus sign icon and instructions to create a compute to run workloads.

Mostrando opciones en la sección compute

The screenshot shows the 'New compute' configuration page. The cluster name 'BikeStoreETLCluster' is highlighted with a red box and labeled '1'. The 'Runtime' dropdown, which is set to 'Runtime: 15.4 LTS (Scala 2.12, Spark 3.5.0)', is highlighted with a red box and labeled '2'. At the bottom, the 'Create compute' button is highlighted with a red box and labeled '3'.

Paso a paso creando el clúster en Databricks Community

**BikeStoreETLCluster**

Databricks Runtime Version: 15.4 LTS (includes Apache Spark 3.5.0, Scala 2.12)

Driver type: Community Optimized (15.3 GB Memory, 2 Cores)

Instance: Free 15 GB Memory: As a Community Edition user, your compute will automatically terminate after an idle period of one or two hours. For more configuration options, please upgrade your Databricks subscription.

**Spark**

spark.databricks.rocksDB.fileManager.useCommitService false

**Environment variables**

No environment variables

Muestra del proceso de creación del clúster

State	Name	Runtime	Active ...	Active c...	Driver	Nodes	Active ...	Source	Creator	Notebo...
●	BikeStoreETLCluster	15.4	15 GB	2 cores	Community ...	1 (0 spot)	1	UI	ericksonclau...	-

Clúster creado y disponible

## Habilitar DBFS y crear notebook

Erickson Claudio Otaño Sánchez

Settings

Delete Account

Log out

The screenshot shows the Databricks Settings page. On the left sidebar, under 'Advanced' (highlighted with a red box), there is a section for 'DBFS File Browser' which is currently enabled (switch is green). The main area displays several other settings like Usage Analytics, Third-party iFraming prevention, and Databricks Autologging.

Validando la habilitación de DBFS

## Creando Notebook temporal para visualizar navegar con comandos por el contenido de los directorios

The screenshot shows the Databricks Workspace page. Step 1 highlights the 'Workspace' button in the sidebar. Step 2 highlights the 'Workspace' folder in the tree view. Step 3 shows a context menu being used to create a new 'Notebook' (highlighted with a red box) within the workspace.

Creando Notebook temporal para validar y crear contenido en carpeta donde estarán los datos de Bike Store

The screenshot shows the Databricks interface with a temporary notebook running a Python command. The notebook has three numbered steps:

1. crear un notebook temporal
2. elegir el clúster
3. Mostrar estado actual de DBFS

The third step shows the output of the command `%fs ls`, which lists the following directory structure:

#	path	name	size	modificationTime
1	dbfs:/FileStore/	FileStore/	0	0
2	dbfs:/databricks-datasets/	databricks-dataset...	0	0
3	dbfs:/databricks-results/	databricks-results/	0	0
4	dbfs:/user/	user/	0	0

Below the table, it says "4 rows | 30.39s runtime".

Mostrando directorios del DBFS

## Subir archivos CSV a DBFS

Habilité el sistema de archivos DBFS y subí los datos de Bike Store a una carpeta `landing`:

- Verifiqué que DBFS estuviera habilitado ejecutando `%fs ls` en un notebook temporal (`TempNotebook`).
- Creé una carpeta `landing` dentro de `dbfs:/FileStore/bike_store_data/` con el comando: `%fs mkdirs dbfs:/FileStore/bike_store_data/landing`.
- Confirmé la creación de la carpeta ejecutando `%fs ls dbfs:/FileStore/bike_store_data/`.
- Subí los 9 archivos CSV del dataset **Bike Store Relacional** (`categories.csv`, `brands.csv`, etc.) desde mi carpeta local **BikeStoreData** al directorio `dbfs:/FileStore/bike_store_data/landing` usando la opción **Upload Data** en el tab **Data**.
- Verifiqué los archivos subidos ejecutando `%fs ls dbfs:/FileStore/bike_store_data/landing` en el notebook.

Se utilizó la siguiente estructura para la arquitectura Medallón

- `dbfs:/FileStore/bike_store_data/`
  - `landing/`: Contiene archivos CSV crudos desde Kaggle.
  - `bronze/`: Donde se moverán los datos desde landing después de la ingestión.
  - `silver/`: Donde se almacenarán los datos transformados.
  - `gold/`: Donde se guardará las tablas agregadas para análisis.

```

Creando directorio donde se subirán los 9 archivos CSV

4 Just now (1s)
%fs mkdirs dbfs:/FileStore/bike_store_data/landing
res20: Boolean = true

5 Just now (1s)
%fs ls dbfs:/FileStore/bike_store_data/
Table
+-----+
# | A path | B name | C size | D modificationTime |
# |-----|
# | 1 dbfs:/FileStore/bike_store_data/landing/ landing/ 0 0 |
+-----+

```

Abriendo Catalog para subir archivos:

**Temp Notebook** Python

File Edit View Run Help Last edit was 5 minutes ago

Run all Share Publish

Workspace

- Recent
- Search
- Catalog**
- Workflows
- Compute
- Machine Learning
- Experiments

07:29 PM (1s) 4

%fs mkdirs dbfs:/bike\_store\_data

res1: Boolean = true

07:30 PM (1s) 5

%fs ls dbfs:/

Table

**Temp Notebook** Python

File Edit View Run Help Last edit was now

Run all Share Publish

Workspace

- Recent
- Search
- Catalog**
- Workflows
- Compute
- Machine Learning
- Experiments

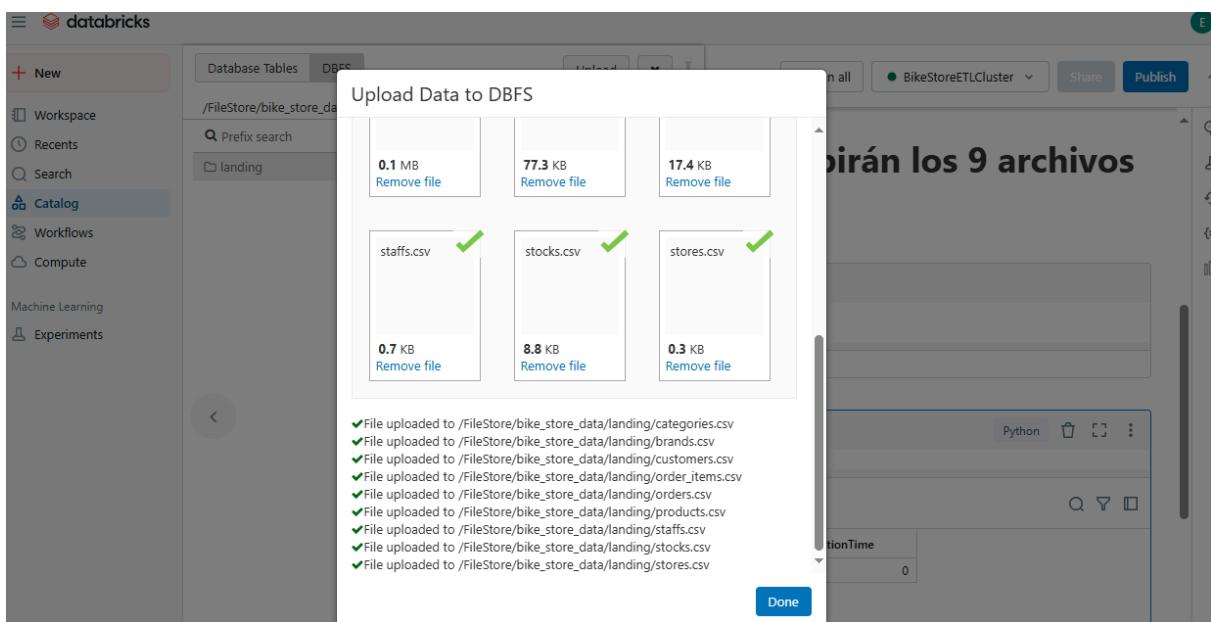
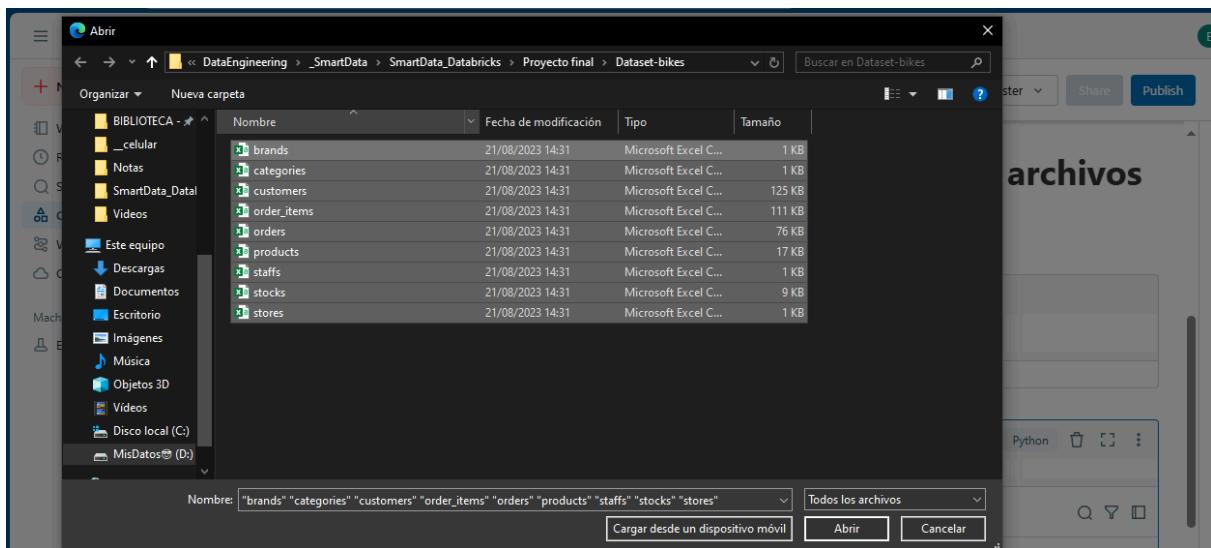
Database Tables DBFS

/FileStore/bike\_store\_data/landing

Prefix search landing

Upload

e se subirán los 9 archivos



The screenshot shows the Databricks workspace interface. On the left, there's a sidebar with options like '+ New', 'Workspace', 'Recents', 'Search', 'Catalog', 'Workflows', 'Compute', 'Machine Learning', and 'Experiments'. The main area is titled 'Temp Notebook' and 'Python'. At the top right, there are buttons for 'Run all', 'BikeStoreETLCluster', 'Share', and 'Publish'. Below the title, it says 'Last edit was now'. A progress bar indicates '1 row | 1.17s runtime' and 'Refreshed 3 minutes ago'. The central part of the screen shows a code cell with the command '%fs ls dbfs:/FileStore/bike\_store\_data/landing'. Below the code cell is a table with 9 rows, each representing a CSV file in the landing stage. The columns are 'path', 'name', 'size', and 'modificationTime'. The table shows the following data:

	path	name	size	modificationTime
1	dbfs/FileStore/bike_store_data/landing/brands.csv	brands.csv	120	1743380333000
2	dbfs/FileStore/bike_store_data/landing/categories.csv	categories.csv	162	1743380333000
3	dbfs/FileStore/bike_store_data/landing/customers.csv	customers.csv	127671	1743380334000
4	dbfs/FileStore/bike_store_data/landing/order_items.csv	order_items.csv	112937	1743380335000
5	dbfs/FileStore/bike_store_data/landing/orders.csv	orders.csv	77286	1743380335000
6	dbfs/FileStore/bike_store_data/landing/products.csv	products.csv	17390	1743380335000
7	dbfs/FileStore/bike_store_data/landing/staffs.csv	staffs.csv	726	1743380335000
8	dbfs/FileStore/bike_store_data/landing/stocks.csv	stocks.csv	8773	1743380335000
9	dbfs/FileStore/bike_store_data/landing/stores.csv	stores.csv	330	1743380336000

At the bottom, it says '9 rows | 0.66s runtime' and 'Refreshed now'.

# Implementación paso a paso

## Organización de notebooks y movimiento a la capa Bronze

1. Organicé los notebooks y moví los datos desde `landing` a la capa Bronze:

- Creé una carpeta principal `BikeStoreProject` en el Workspace y una subcarpeta `notebooks` para alojar los notebooks.

Creé notebooks separados: Extract, Transform y Load dentro del notebook.

- En el notebook Extract verifiqué los archivos en la carpeta landing.
- Creé la carpeta bronze
- Usé PySpark para leer los 9 archivos CSV
- Usé el Structtype para definir el esquema de los datos de cada archivo
- Agregué el metadato de ingestion date, y guardé archivos como Delta en bronze.

Crear la estructura para el proyecto. Carpeta General: BikeStoreProject

Databricks Workspace interface showing the 'Workspace' sidebar and a table of contents. A red box highlights the 'Workspace' button in the sidebar, and another red box highlights the 'Folder' column header in the table.

Dentro de la carpeta BikeStoreProject, una subcarpeta "notebook" . Luego crear tres notebooks para cada proceso ETL (Extract, Transform, Load).

Databricks workspace interface showing the creation of a new folder named 'notebooks'. A red box highlights the 'notebooks' folder in the sidebar, and another red box highlights the 'Recent resources' section in the notebook editor.

Tres notebooks para todo el proceso ETL

## Proceso de Extract de landing a Bronze

**Verificar contenido en carpeta landing y definición de esquemas con structype**

**Extract Python**

File Edit View Run Help Last edit was now

Run all BikeStoreETLCluster Share Publish

## Sección: Extract - Mover datos de landing a Bronze

### Verificación de datos en landing

```
Just now (6s) %fs ls dbfs:/FileStore/bike_store_data/landing
```

	path	name	size	modificationTime
1	dbfs:/FileStore/bike_store_data/landing/brands.csv	brands.csv	120	1743380333000
2	dbfs:/FileStore/bike_store_data/landing/categories.csv	categories.csv	162	1743380333000
3	dbfs:/FileStore/bike_store_data/landing/customers.csv	customers.csv	127671	1743380334000
4	dbfs:/FileStore/bike_store_data/landing/order_items.csv	order_items.csv	112937	1743380335000
5	dbfs:/FileStore/bike_store_data/landing/orders.csv	orders.csv	77286	1743380335000
6	dbfs:/FileStore/bike_store_data/landing/products.csv	products.csv	17390	1743380335000
7	dbfs:/FileStore/bike_store_data/landing/staffs.csv	staffs.csv	726	1743380335000
8	dbfs:/FileStore/bike_store_data/landing/stocks.csv	stocks.csv	8773	1743380335000
9	dbfs:/FileStore/bike_store_data/landing/stores.csv	stores.csv	330	1743380336000

9 rows | 5.50s runtime Refreshed now

### Creación de la carpeta Bronze

```
Just now (1s) %fs mkdirs dbfs:/FileStore/bike_store_data/bronze
```

[Shift+Enter] to run and move to next cell

```
res1: Boolean = true
```

```
Just now (<1s) %fs ls dbfs:/FileStore/bike_store_data/
```

	path	name	size	modificationTime
1	dbfs:/FileStore/bike_store_data/bronze/	bronze/	0	0
2	dbfs:/FileStore/bike_store_data/landin...	landing/	0	0

2 rows | 0.49s runtime Refreshed now

## 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)
])
```

Se define la estructura para los nueve archivos colocados en landing. Más detalle se puede ver en el archivo odc

Código PySpark con StructType para definir esquemas y mover los 9 archivos a bronze

**Mover datos de landing a Bronze**

```
# categories.csv
df_categories = spark.read.csv("dbfs:/FileStore/bike_store_data/landing/categories.csv", header=True,
schema=categories_schema)
df_categories_with_metadata = df_categories.withColumn("ingestion_date", current_timestamp())
df_categories_with_metadata.write.format("delta").mode("overwrite").save("dbfs:/FileStore/bike_store_data/bronze/categories")
```

**Extract Python**

File Edit View Run Help Last edit was 12 minutes ago

Run all Share Publish

BikeStoreETLCluster

(54) Spark Jobs

- df\_brands: pyspark.sql.dataframe.DataFrame = [brand\_id: integer, brand\_name: string]
- df\_brands\_with\_metadata: pyspark.sql.dataframe.DataFrame = [brand\_id: integer, brand\_name: string ... 1 more field]
- df\_categories: pyspark.sql.dataframe.DataFrame = [category\_id: integer, category\_name: string]
- df\_categories\_with\_metadata: pyspark.sql.dataframe.DataFrame = [category\_id: integer, category\_name: string ... 1 more field]
- df\_customers: pyspark.sql.dataframe.DataFrame = [customer\_id: integer, first\_name: string ... 7 more fields]
- df\_customers\_with\_metadata: pyspark.sql.dataframe.DataFrame = [customer\_id: integer, first\_name: string ... 8 more fields]
- df\_order\_items: pyspark.sql.dataframe.DataFrame = [order\_id: integer, item\_id: integer ... 4 more fields]
- df\_order\_items\_with\_metadata: pyspark.sql.dataframe.DataFrame = [order\_id: integer, item\_id: integer ... 5 more fields]
- df\_orders: pyspark.sql.dataframe.DataFrame = [order\_id: integer, customer\_id: integer ... 6 more fields]
- df\_orders\_with\_metadata: pyspark.sql.dataframe.DataFrame = [order\_id: integer, customer\_id: integer ... 7 more fields]
- df\_products: pyspark.sql.dataframe.DataFrame = [product\_id: integer, product\_name: string ... 4 more fields]
- df\_products\_with\_metadata: pyspark.sql.dataframe.DataFrame = [product\_id: integer, product\_name: string ... 5 more fields]
- df\_staffs: pyspark.sql.dataframe.DataFrame = [staff\_id: integer, first\_name: string ... 6 more fields]
- df\_staffs\_with\_metadata: pyspark.sql.dataframe.DataFrame = [staff\_id: integer, first\_name: string ... 7 more fields]
- df\_stocks: pyspark.sql.dataframe.DataFrame = [store\_id: integer, product\_id: integer ... 1 more field]
- df\_stocks\_with\_metadata: pyspark.sql.dataframe.DataFrame = [store\_id: integer, product\_id: integer ... 2 more fields]
- df\_stores: pyspark.sql.dataframe.DataFrame = [store\_id: integer, store\_name: string ... 6 more fields]
- df\_stores\_with\_metadata: 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  
Procesado: stocks

## Explorando algunos datos en bronze

### Exploración de datos en Bronze

```
< > 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 Children Bicycles 2025-03-31T02:07:32.974+00...
2 Comfort Bicycles 2025-03-31T02:07:32.974+00...
3 Cruisers Bicycles 2025-03-31T02:07:32.974+00...
4 Cyclocross Bicycles 2025-03-31T02:07:32.974+00...
5 Electric Bikes 2025-03-31T02:07:32.974+00...

5 rows | 0.61s runtime Refreshed now
```

```
< > Just now (2s) 15 Python
df_categories_bronze = spark.read.format("delta").load("dbfs:/FileStore/bike_store_data/bronze/customers")
df_categories_bronze.limit(5).display()

(2) Spark Jobs
df_categories_bronze: pyspark.sql.dataframe.DataFrame = [customer_id: integer, first_name: string ... 8 more fields]

Table + 
customer_id first_name last_name phone email street city
1          1        Kasha   Todd    NULL kasha.todd@yahoo.com 910 Vine Street Campb
2          2       Kashia   Todd    NULL kasha.todd@yahoo.com 910 Vine Street Campb
3          3      Tameka Fisher  NULL tameka.fisher@aol.com 769C Honey Creek S... Redon
4          4       Daryl Spence  NULL daryl.spence@aol.com 988 Pearl Lane Unionc
5          5     Charlette Rice (916) 381-6003 charlotte.rice@msn.co... 107 River Dr. Sacram

5 rows | 1.67s runtime Refreshed now
```

[Shift+Enter] to run and move to next cell  
[Ctrl+Shift+P] to open the command palette

## Transformación de datos a la capa Silver

Los siguientes pasos se aplican en el notebook "Transform"

- Creé la carpeta silver en DBFS con %fs mkdirs dbfs:/FileStore/bike\_store\_data/silver.
  - Apliqué transformaciones a cada tabla:
    - categories: Convertí category\_name a mayúsculas y eliminé duplicados.

- brands: Convertí brand\_name a mayúsculas y eliminé duplicados.
- products: Filtré list\_price > 0 y convertí product\_name a mayúsculas.
- customers: Rellené nulos en phone con "N/A" y creé full\_name.
- stores: Rellené nulos en email con "N/A" y convertí store\_name a mayúsculas.
- staffs: Rellené nulos en phone con "N/A" y en manager\_id con 0.
- orders: Rellené nulos en shipped\_date con "N/A" y añadí order\_status\_desc (1 = "Pendiente", 2 = "Procesando", 3 = "Rechazado", 4 = "Completado", "Desconocido").
- order\_items: Filtré quantity > 0 y calculé total\_price.
- stocks: Filtré quantity >= 0.
- Guardé las tablas transformadas en dbfs:/FileStore/bike\_store\_data/silver/ como Delta.
- Verifiqué la carpeta silver con %fs ls.
- Exploré las tablas Silver mostrando las primeras filas y el conteo de filas.

Transform Python ▾

File Edit View Run Help Last edit was 1 minute ago

Run all BikeStoreETLCluster Share Publish

**Creando la carpeta silver y confirmación**

Markdown

```
%fs mkdirs dbfs:/FileStore/bike_store_data/silver
res0: Boolean = true
```

Python

```
%fs ls dbfs:/FileStore/bike_store_data/
3
```

Table +

	path	name	size	modificationTime
1	dbfs:/FileStore/bike_store_data/bronze/	bronze/	0	0
2	dbfs:/FileStore/bike_store_data/landin...	landing/	0	0
3	dbfs:/FileStore/bike_store_data/silver/	silver/	0	0

3 rows | 0.90s runtime Refreshed 1 minute ago

The screenshot shows a Databricks notebook interface. The top navigation bar includes 'Transform' (selected), 'Python', 'File', 'Edit', 'View', 'Run', 'Help', and a status message 'Last edit was now'. On the right are buttons for 'Run all', 'BikeStoreETLCluster', 'Share', and 'Publish'. The main area has a sidebar with icons for file operations. A central box displays '3 rows | 0.90s runtime' and was 'Refreshed 36 minutes ago'. Below this, the title 'Transformaciones a la capa Silver' is shown. The code editor contains the following Python code:

```
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"])
```

Transform Python ▾

File Edit View Run Help Last edit was now Run all BikeStoreETLCluster Share Publish

df\_brands\_silver: pyspark.sql.DataFrame = [brand\_id: integer, brand\_name: string ... 1 more field]

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

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

df\_customers\_bronze: pyspark.sql.DataFrame = [customer\_id: integer, first\_name: string ... 8 more fields]

df\_customers\_silver: pyspark.sql.DataFrame = [customer\_id: integer, first\_name: string ... 9 more fields]

df\_order\_items\_bronze: pyspark.sql.DataFrame = [order\_id: integer, item\_id: integer ... 5 more fields]

df\_order\_items\_silver: pyspark.sql.DataFrame = [order\_id: integer, item\_id: integer ... 6 more fields]

df\_orders\_bronze: pyspark.sql.DataFrame = [order\_id: integer, customer\_id: integer ... 7 more fields]

df\_orders\_silver: pyspark.sql.DataFrame = [order\_id: integer, customer\_id: integer ... 8 more fields]

df\_products\_bronze: pyspark.sql.DataFrame = [product\_id: integer, product\_name: string ... 5 more fields]

df\_products\_silver: pyspark.sql.DataFrame = [product\_id: integer, product\_name: string ... 5 more fields]

df\_staffs\_bronze: pyspark.sql.DataFrame = [staff\_id: integer, first\_name: string ... 7 more fields]

df\_staffs\_silver: pyspark.sql.DataFrame = [staff\_id: integer, first\_name: string ... 7 more fields]

df\_stocks\_bronze: pyspark.sql.DataFrame = [store\_id: integer, product\_id: integer ... 2 more fields]

df\_stocks\_silver: pyspark.sql.DataFrame = [store\_id: integer, product\_id: integer ... 2 more fields]

df\_stores\_bronze: pyspark.sql.DataFrame = [store\_id: integer, store\_name: string ... 7 more fields]

df\_stores\_silver: pyspark.sql.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

Procesado: stocks

Transform Python ▾

File Edit View Run Help Last edit was now

Run all BikeStoreETLCluster Share Publish

## VERIFICACION de datos en Silver

Just now (2s) 7 Python

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

Table +

#	path	name	size	modificationTime
1	dbfs:/FileStore/bike_store_data/silver/brands/	brands/	0	0
2	dbfs:/FileStore/bike_store_data/silver/categories/	categories/	0	0
3	dbfs:/FileStore/bike_store_data/silver/customers/	customers/	0	0
4	dbfs:/FileStore/bike_store_data/silver/order_items/	order_items/	0	0
5	dbfs:/FileStore/bike_store_data/silver/orders/	orders/	0	0
6	dbfs:/FileStore/bike_store_data/silver/products/	products/	0	0
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

```

Just now (11s) 9
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")

```

	customer_id	first_name	last_name	phone	email	street	city
1	1	Debra	Burks	NULL	debra.burks@yahoo.com	9273 Thorne Ave.	Orchar
2	2	Kasha	Todd	NULL	kasha.todd@yahoo.com	910 Vine Street	Campb
3	3	Tameka	Fisher	NULL	tameka.fisher@aol.com	769C Honey Creek S...	Redond
4	4	Daryl	Spence	NULL	daryl.spence@aol.com	988 Pearl Lane	Unionc
5							

5 rows | 10.54s runtime Refreshed now

	store_id	store_name	phone	email	street	city	state	zip
1	1	SANTA CRUZ BIKES	(830) 476-4321	santacruz@bikes.sh...	3700 Portola Drive	Santa Cruz	CA	95060
2	2	BALDWIN BIKES	(516) 379-8888	baldwin@bikes.shop	4200 Chestnut Lane	Baldwin	NY	11432
3	3	ROWLETT BIKES	(972) 530-5555	rowlett@bikes.shop	8000 Fairway Aven...	Rowlett	TX	75088

3 rows | 10.54s runtime Refreshed now

	staff_id	first_name	last_name	email	phone	active	store_id
--	----------	------------	-----------	-------	-------	--------	----------

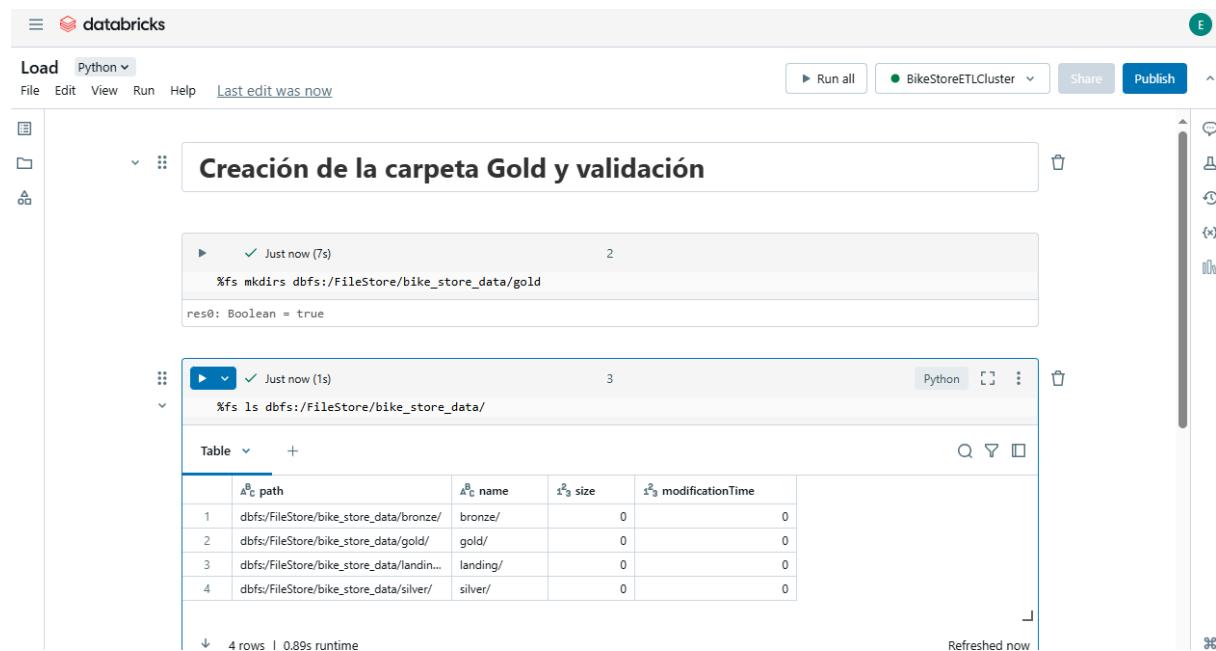
## Carga de datos a la capa Gold

Crear tablas agregadas en la capa Gold (en dbfs:/FileStore/bike\_store\_data/gold/) a partir de los datos limpios de la capa Silver. Estas tablas estarán optimizadas para análisis y reportes, como ventas totales por tienda, productos más vendidos, o análisis de clientes. Se utiliza el notebook Load.ipynb para este paso.

## Carga a la Capa Gold

Creé tablas analíticas en la capa Gold en Load.ipynb:

- Creé la carpeta gold en DBFS con %fs mkdirs dbfs:/FileStore/bike\_store\_data/gold.
- Generé tres tablas analíticas:
  - sales\_by\_store: Ventas totales por tienda, uniendo order\_items, orders, y stores.
  - top\_products: Productos más vendidos por cantidad, uniendo order\_items y products.
  - customer\_analysis: Número de pedidos y gasto total por cliente, uniendo orders, order\_items, y customers.
- Guardé las tablas en dbfs:/FileStore/bike\_store\_data/gold/ como Delta.
- Verifiqué la carpeta gold con %fs ls.
- Exploré las tablas Gold mostrando las primeras filas y el conteo de filas.



The screenshot shows a Databricks notebook titled "Creación de la carpeta Gold y validación". It contains two code cells and a data exploration section.

**Code Cells:**

- Cell 2: %fs mkdirs dbfs:/FileStore/bike\_store\_data/gold  
res0: Boolean = true
- Cell 3: %fs ls dbfs:/FileStore/bike\_store\_data/

**Data Exploration:**

A table titled "Table" displays the contents of the "gold" directory:

#	path	name	size	modificationTime
1	dbfs:/FileStore/bike_store_data/bronze/	bronze/	0	0
2	dbfs:/FileStore/bike_store_data/gold/	gold/	0	0
3	dbfs:/FileStore/bike_store_data/landing/	landing/	0	0
4	dbfs:/FileStore/bike_store_data/silver/	silver/	0	0

Below the table, it says "4 rows | 0.89s runtime".

**Carga de datos a la capa Gold**

```

Just now (22s) 5
from pyspark.sql.functions import col, sum, count

# Leer las tablas necesarias desde Silver
df_order_items = spark.read.format("delta").load("dbfs:/FileStore/bike_store_data/silver/order_items")
df_orders = spark.read.format("delta").load("dbfs:/FileStore/bike_store_data/silver/orders")
df_stores = spark.read.format("delta").load("dbfs:/FileStore/bike_store_data/silver/stores")
df_products = spark.read.format("delta").load("dbfs:/FileStore/bike_store_data/silver/products")
df_customers = spark.read.format("delta").load("dbfs:/FileStore/bike_store_data/silver/customers")

# Tabla 1: Ventas por tienda (sales_by_store)
df_sales_by_store = df_order_items \
    .join(df_orders, "order_id") \
    .join(df_stores, "store_id") \
    .groupBy("store_id", "store_name") \
    .agg(sum("total_price").alias("total_sales")) \
    .orderBy("total_sales").desc()

```

**Customer Analysis**

```

1 minute ago (22s) 5 Python
df_customer_analysis.write.format("delta").mode("overwrite").save("dbfs:/FileStore/bike_store_data/gold/customer_analysis")
print("Creada tabla: customer_analysis")

(32) Spark Jobs
df_customer_analysis: pyspark.sql.dataframe.DataFrame = [customer_id: integer, full_name: string ... 2 more fields]
df_customers: pyspark.sql.dataframe.DataFrame = [customer_id: integer, first_name: string ... 9 more fields]
df_order_items: pyspark.sql.dataframe.DataFrame = [order_id: integer, item_id: integer ... 6 more fields]
df_orders: pyspark.sql.dataframe.DataFrame = [order_id: integer, customer_id: integer ... 8 more fields]
df_products: pyspark.sql.dataframe.DataFrame = [product_id: integer, product_name: string ... 5 more fields]
df_sales_by_store: pyspark.sql.dataframe.DataFrame = [store_id: integer, store_name: string ... 1 more field]
df_stores: pyspark.sql.dataframe.DataFrame = [store_id: integer, store_name: string ... 7 more fields]
df_top_products: pyspark.sql.dataframe.DataFrame = [product_id: integer, product_name: string ... 1 more field]

Creada tabla: sales_by_store
Creada tabla: top_products
Creada tabla: customer_analysis

```

**Verificación de datos en Gold**

```

Just now (1s) 7 Python
%fs ls dbfs:/FileStore/bike_store_data/gold

Table + 
+-----+
| # path | name | size | modificationTime |
+-----+
| 1 dbfs:/FileStore/bike_store_data/gold/customer_analysis/ | customer_analysi... | 0 | 0 |
| 2 dbfs:/FileStore/bike_store_data/gold/sales_by_store/ | sales_by_store/ | 0 | 0 |
| 3 dbfs:/FileStore/bike_store_data/gold/top_products/ | top_products/ | 0 | 0 |
+-----+
↓ 3 rows | 1.31s runtime
Refreshed now

```

**Databricks**

Load Python File Edit View Run Help Last edit was now

Run all BikeStoreETLCluster Share Publish

## 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 +
```

**Databricks**

Load Python File Edit View Run Help Last edit was now

Run all BikeStoreETLCluster Share Publish

	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 | 2.64s runtime Refreshed 2 minutes ago

Filas en sales\_by\_store: 3

	product_id	product_name	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

The screenshot shows a Databricks notebook interface with two tables displayed:

- top\_products** (Refreshed 2 minutes ago):
 

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
- customer\_analysis** (Refreshed 2 minutes ago):
 

	customer_id	full_name	total_orders	total_spent
1	94	Sharyn Hopkins	10	34807.93919999999
2	10	Pamelia Newman	11	33634.2604
3	75	Abby Gamble	7	32803.0062
4	6	Lyndsey Bean	11	32675.07249999995
5	16	Emmitt Sanchez	12	31925.885700000003

 5 rows | 2.64s runtime

## Explicación del código

- **Creación de gold:** Crea la carpeta gold con %fs mkdirs.
- **Carga de datos:**
  - Lee las tablas necesarias desde silver.
  - **Ventas por tienda:** Une order\_items, orders, y stores, agrupa por store\_id y store\_name, y calcula el total de ventas (total\_price).
  - **Productos más vendidos:** Une order\_items y products, agrupa por product\_id y product\_name, y calcula la cantidad total vendida.
  - **Análisis de clientes:** Une orders, order\_items, y customers, agrupa por customer\_id y full\_name, y calcula el número de pedidos y el gasto total.
- **Guardado:** Guarda cada tabla en gold como Delta.
- **Verificación de gold:** Usa %fs ls para listar las carpetas.
- **Exploración:** Lee las tablas Gold y muestra las primeras filas y el conteo para confirmar que las agregaciones se realizaron correctamente.

## Visualización de Datos y conclusiones de análisis

Generar gráficos para las tablas Gold (sales\_by\_store, top\_products, customer\_analysis) en el notebook Load.ipynb. Esto nos permitirá visualizar las ventas por tienda, los productos más vendidos, y el análisis de clientes de manera más intuitiva.

## Visualización de Datos en la Capa Gold con Matplotlib

Creé gráficos para las tablas Gold en [Load.ipynb](#) usando Matplotlib:

- Convertí los DataFrames de Spark a Pandas para facilitar la visualización.
- Gráficos creados:
  - **Ventas por tienda:** Gráfico de barras mostrando las ventas totales por tienda.
  - **Productos más vendidos:** Gráfico de barras con los 10 productos más vendidos por cantidad.
  - **Análisis de clientes:** Gráfico de dispersión mostrando la relación entre el número de pedidos y el gasto total por cliente.
- Los gráficos se generaron directamente en el notebook y se tomaron capturas para incluirlas en la presentación.

### Visualización de datos en Gold con Matplotlib

#### Convertir los DataFrames de Spark a Pandas para facilitar la visualización

```
▶   ✓ 10:33 PM (1s)           12
    import matplotlib.pyplot as plt

    df_sales_by_store_pd = df_sales_by_store.toPandas()
    df_top_products_pd = df_top_products.toPandas()
    df_customer_analysis_pd = df_customer_analysis.toPandas()

▶ (3) Spark Jobs
```

### Gráfico 1: Ventas por tienda (sales\_by\_store)

```
10:34 PM (<1s) 14 Python
plt.figure(figsize=(10, 6))
plt.bar(df_sales_by_store_pd['store_name'], df_sales_by_store_pd['total_sales'], color='skyblue')
plt.title('Ventas Totales por Tienda')
plt.xlabel('Tienda')
plt.ylabel('Ventas Totales ($)')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

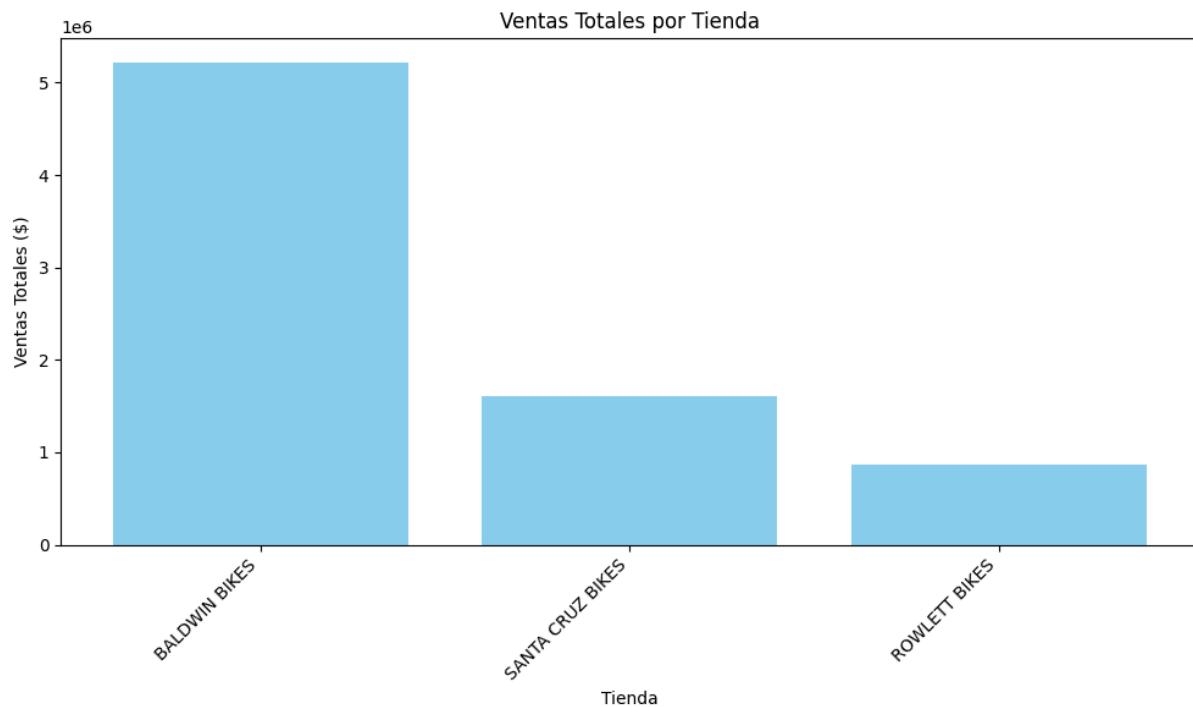
### Gráfico 2: Productos más vendidos (top\_products) - Top 10

```
10:35 PM (<1s) 16 Python
df_top_products_top10_pd = df_top_products_pd.head(10) # Limitar a los 10 productos más vendidos
plt.figure(figsize=(12, 6))
plt.bar(df_top_products_top10_pd['product_name'], df_top_products_top10_pd['total_quantity_sold'],
color='lightgreen')
plt.title('Top 10 Productos Más Vendidos')
plt.xlabel('Producto')
plt.ylabel('Cantidad Vendida')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

### Gráfico 3: Análisis de clientes (customer\_analysis)

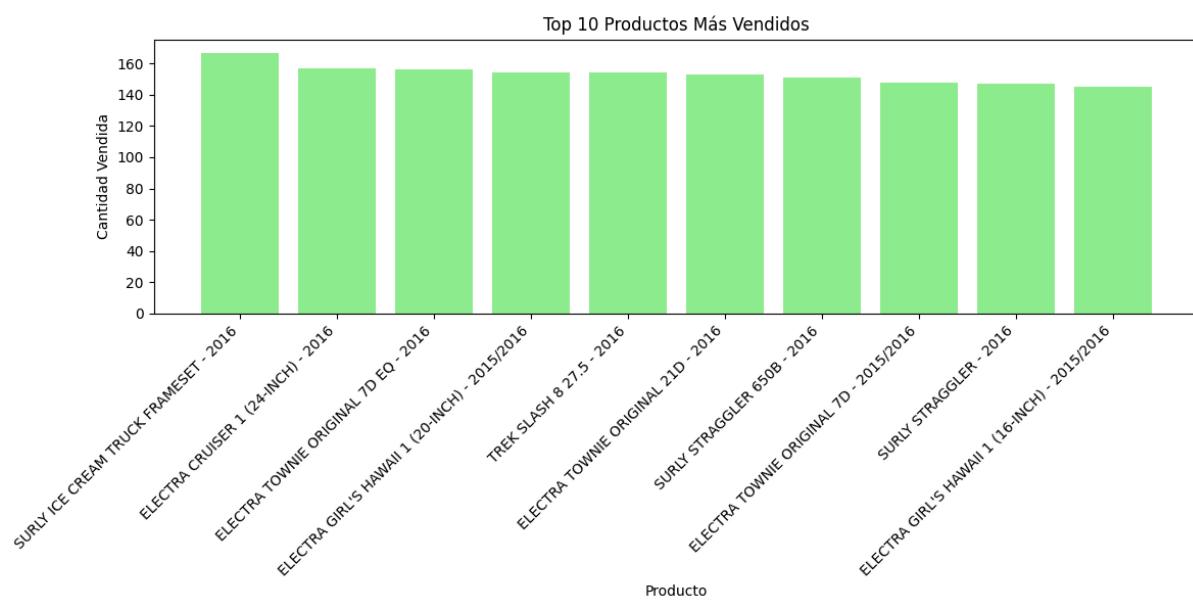
```
10:35 PM (<1s) 18 Python
plt.figure(figsize=(10, 6))
plt.scatter(df_customer_analysis_pd['total_orders'], df_customer_analysis_pd['total_spent'],
color='salmon', alpha=0.5)
plt.title('Relación entre Número de Pedidos y Gasto Total por Cliente')
plt.xlabel('Número de Pedidos')
plt.ylabel('Gasto Total ($)')
plt.grid(True)
plt.tight_layout()
plt.show()
```

## Captura 1: Gráfico de Ventas por Tienda



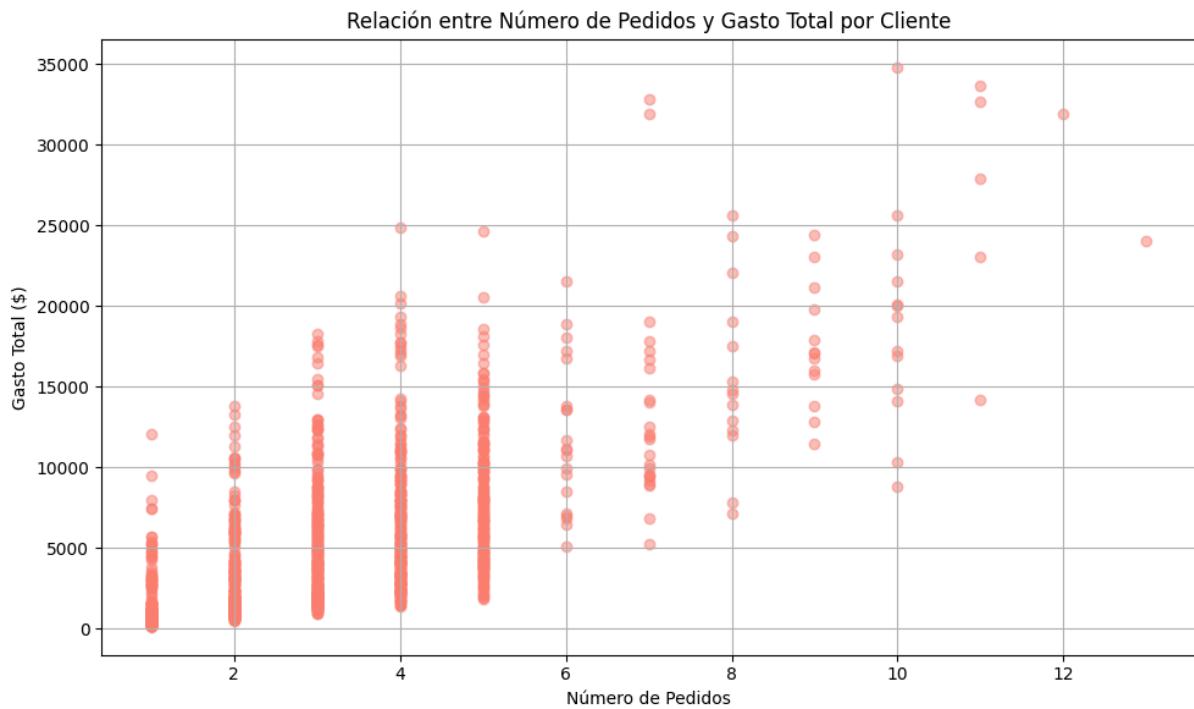
Observación: Baldwin Bikes lidera con más de 5M en ventas.

## Captura 2: Gráfico de Productos Más Vendidos



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

## Captura 3: Gráfico de Análisis de Clientes



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

Paso a paso para crear el workflow **BikeStoreETLPipeline** en Databricks que automatice la ejecución de los notebooks Extract.ipynb, Transform.ipynb, y Load.ipynb en orden, asegurando que cada etapa del pipeline ETL (Bronze → Silver → Gold) se ejecute secuencialmente.

### Creación del Workflow

Creé un notebook maestro para orquestar el pipeline ETL:

- 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.

The screenshot shows the Databricks sidebar on the left. The 'Workflows' option is highlighted with a blue background and white text. Other options like 'Workspace', 'Recents', 'Search', 'Catalog', 'Compute', 'Machine Learning', and 'Experiments' are also listed. A red box highlights the 'New' button at the top of the sidebar. A modal window titled 'Learn more about Workflows.' is displayed in the center, with the text 'For access, please upgrade your Databricks subscription' below it.

The screenshot shows the Databricks workspace interface. The left sidebar has 'Workspace' selected, indicated by a red box. The main area shows the 'BikeStoreProject' workspace structure with a 'notebooks' folder highlighted by a red box. In the bottom right, a table lists three notebooks: 'Extract', 'Load', and 'Transform'. A new row is being created, with a red box highlighting the 'Notebook' input field under the 'Type' column. The 'Create' button is visible at the top right of the table.

Creando Notebook PipelineOrchestrator.ipynb

**Orquestación del Pipeline ETL**

```
%run "/BikeStoreProject/notebooks/Extract"  
# Paso 2: Ejecutar Transform.ipynb  
%run "/BikeStoreProject/notebooks/Transform"  
# Paso 3: Ejecutar Load.ipynb  
%run "/BikeStoreProject/notebooks/Load"
```

**Orquestación del Pipeline ETL**

Interrupt Compute snapshot for version: 1

```
%run "/BikeStoreProject/notebooks/Extract"
```

**Sección: Extract - Mover datos de landing a Bronze**

**Verificación de datos en landing**

Ejecutando notebook Extract

The screenshot shows the Databricks interface with the notebook `PipelineOrchestrator` open in Python. The sidebar shows the pipeline stages: Extract, Load, PipelineOrchestrator, and Transform. The main area displays the output of the `Transform` stage, which includes a table of 5 rows and the command `%run "/BikeStoreProject/notebooks/Transform"`.

	path	name	size	modificationTime
1	dbfs/FileStore/bike_store_data/bronze/	bronze/	0	0
2	dbfs/FileStore/bike_store_data/gold/	gold/	0	0
3	dbfs/FileStore/bike_store_data/landing/	landing/	0	0
4	dbfs/FileStore/bike_store_data/silver/	silver/	0	0

Ejecutando notebook Transform

The screenshot shows the Databricks interface with the notebook `PipelineOrchestrator` open in Python. The sidebar shows the pipeline stages: Extract, Load, PipelineOrchestrator, and Transform. The main area displays the output of the `Load` stage, which includes a table of 5 rows and the command `%run "/BikeStoreProject/notebooks/Load"`. The status bar indicates the run is "Waiting".

	1	2	3	4	5	6
3	1	1	3	6	2025-04-01T01:34:10.681+00:00	
4	1	1	4	23	2025-04-01T01:34:10.681+00:00	
5	1	1	5	22	2025-04-01T01:34:10.681+00:00	

Ejecutando Load

## Confirmando resultados del workflow

## Confirmar Resultados del Pipeline

▶ v ✓ 1 minute ago (1s) 6 Python 🗑️ ⚡ ⏺

%fs ls dbfs:/FileStore/bike\_store\_data/bronze

Table + 🔍 🔍 🔍

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

↓ 9 rows | 0.68s runtime Refreshed 1 minute ago

Resultados de Bronze

▶ v ✓ 1 minute ago (<1s) 7 Python 🗑️ ⚡ ⏺

%fs ls dbfs:/FileStore/bike\_store\_data/silver

Table + 🔍 🔍 🔍

	A <sup>B</sup> C path	A <sup>B</sup> C name	1 <sup>2</sup> 3 size	1 <sup>2</sup> 3 modification...	⋮	☰
1	dbfs:/FileStore/bike_store_data/silver/brands/	brands/	0	0		
2	dbfs:/FileStore/bike_store_data/silver/categories/	categories/	0	0		
3	dbfs:/FileStore/bike_store_data/silver/customers/	customers/	0	0		
4	dbfs:/FileStore/bike_store_data/silver/order_items/	order_items/	0	0		
5	dbfs:/FileStore/bike_store_data/silver/orders/	orders/	0	0		
6	dbfs:/FileStore/bike_store_data/silver/products/	products/	0	0		
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 | 0.49s runtime Refreshed 1 minute ago

Resultados de Silver

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

Table

	<sup>B</sup> C path	<sup>B</sup> C name	<sup>I</sup> <sub>3</sub> size	<sup>I</sup> <sub>3</sub> modificationTime
1	dbfs:/FileStore/bike_store_data/gold/customer_analysis/	customer_analysis...	0	0
2	dbfs:/FileStore/bike_store_data/gold/sales_by_store/	sales_by_store/	0	0
3	dbfs:/FileStore/bike_store_data/gold/top_products/	top_products/	0	0

3 rows | 0.50s runtime      Refreshed 1 minute ago

Resultados de Gold

```
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

<sup>I</sup> <sub>3</sub> store_id	<sup>B</sup> C 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 | 1.81s runtime      Refreshed 2 minutes ago

Filas en sales\_by\_store: 3

Mostrando una tabla gold "sales\_by\_store"

## Resumen del Proyecto

En este proyecto, construí un pipeline ETL completo en Databricks Community Edition utilizando la arquitectura Medallón (Bronze, Silver, Gold) con el dataset "Bike Store Relacional". Los principales logros fueron:

- **Configuración inicial:** Configuré un clúster y subí los datos a DBFS.
- **Capa Bronze:** Ingesté los datos crudos desde `landing` a `bronze`, definiendo esquemas con `StructType` y agregando metadatos (`ingestion_date`).

- **Capa Silver:** Transformé los datos para limpiarlos y estandarizarlos, manejando valores nulos (por ejemplo, rellenando `phone`, `email`, y `shipped_date` con "N/A") y creando columnas calculadas (como `total_price` y `order_status_desc` en español).
- **Capa Gold:** Creé tablas analíticas para reportes, incluyendo ventas por tienda, productos más vendidos, y análisis de clientes.
- **Documentación:** Documenté cada paso en Notion con capturas de pantalla y explicaciones detalladas.

### Aprendizajes:

- Aprendí a trabajar con Databricks y PySpark para construir pipelines ETL.
- Entendí la importancia de la arquitectura Medallón para organizar datos en diferentes capas.
- Practiqué transformaciones de datos (manejo de nulos, estandarización, agregaciones) y el uso de Delta Lake.
- Mejoré mis habilidades de documentación y organización de proyectos.

### Resultados:

- Las tablas Gold (`sales_by_store`, `top_products`, `customer_analysis`) están listas para análisis, proporcionando información valiosa como las tiendas con más ventas, los productos más populares, y los clientes con mayor gasto.
- El pipeline es reproducible y puede adaptarse a otros datasets similares.

Este proyecto me permitió aplicar conceptos de ingeniería de datos en un entorno realista y fortalecer los conocimientos obtenidos en el curso.

## Entregables

- Documentación del proyecto en Notion
- Power Point con las capturas y explicaciones del proyecto
- Diagrama del proyecto con la herramienta draw io
- Archivo BikeStoreProjectdbc que contiene:
  - Extractdbc (Notebook con código pyspark para leer datos crudos importados y guardarlos en formato delta en carpeta bronze)

- Transformdbc (Notebook con código pyspark para transformar datos desde bronze)
- Loaddbc (Notebook con el código pyspark para la creación de tablas gold y análisis de datos)
- PipelineOrchestratordbc (Notebook con la secuencia de ejecución de los notebooks para ETL)
- Temp Notebook (contiene comandos usados en el inicio para crear carpetas para los archivos)