End-to-End Data Engineering Pipeline with Databricks (Bronze–Silver–Gold) and Power BI Visualization

End-to-End Data Engineering Pipeline using Databricks & Power BI

1. Project Overview

Objective

The goal of this project is to build a complete end-to-end Data Engineering pipeline, starting from a raw CSV file and ending with a business-ready dashboard in Power BI.

The project demonstrates:

- **Data Ingestion**
- **Data Cleaning & Transformation**
- **Data Modeling (Star Schema)**
- **Data Lakehouse architecture (Bronze, Silver, Gold)**
- **ETL Pipeline using Databricks (PySpark)**
- **Publishing tables to Unity Catalog**
- **Connecting Databricks to Power BI**
- **Creating interactive business dashboards**

2. Technology Stack Used

Component	Tool
Cloud Platform	Databricks (Unity Catalog)
Storage	Databricks Volumes (Data Lake)
Processing	PySpark
Data Format	Delta Lake
Data Modeling	Star Schema (Fact + Dimensions)
Analytics	Databricks SQL
Visualization	Power BI

3. Dataset Used

Dataset Name

Financial Sample (2).csv

Source

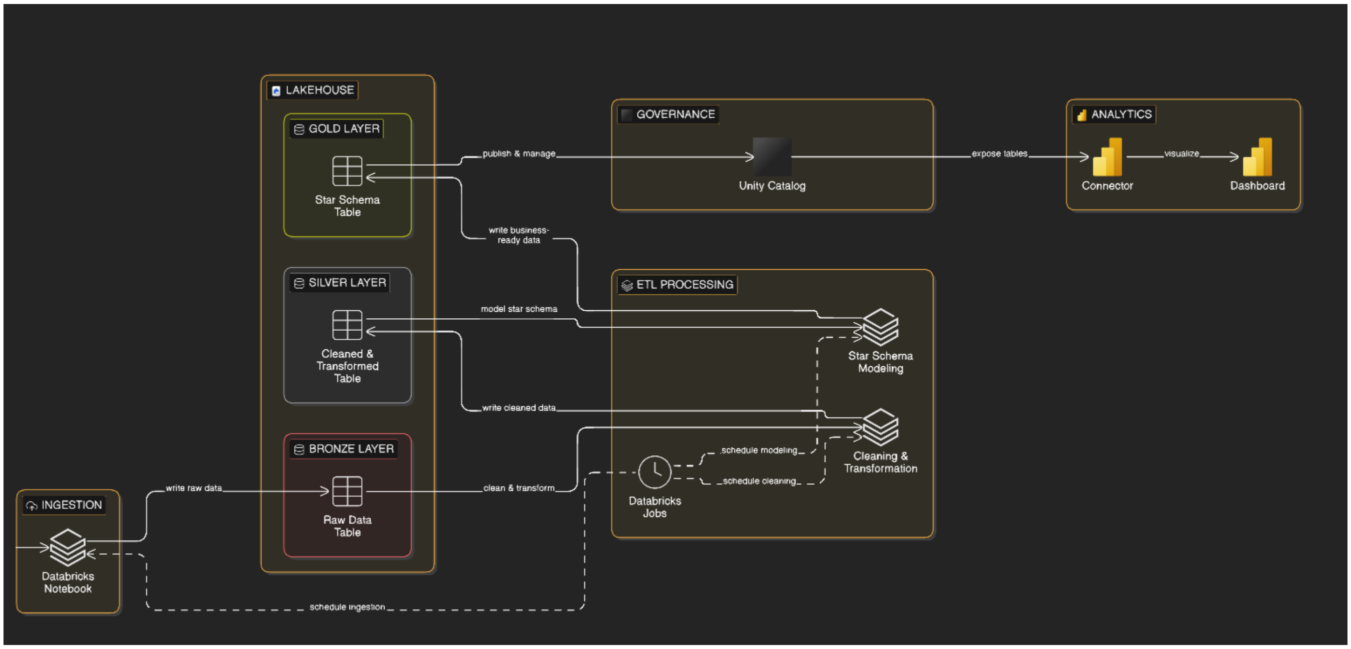
A sample financial dataset containing sales, profit, and cost-related business metrics.

Key Columns (Before Cleaning)

- Product
- Segment
- Country
- Discount Band
- Units Sold
- Manufacturing Price
- Gross Sales
- Sales
- COGS
- Profit
- Date

4. Overall Architecture

High-Level Architecture Diagram (Logical Flow)



5. Step-by-Step Implementation

STEP 1 — Create Project Folder Structure in Databricks

5.1 Create a Volume in Databricks

Go to:

- Databricks → Catalog → workspace → default

Create a new volume:

- Name: financial_de

Inside this volume, create the following folders:

- **/Volumes/workspace/default/financial_de/**
- │
- ├── **Financial Sample (2).csv (Upload your dataset here)**
- ├── **bronze/**
- ├── **silver/**
- └── **gold/**

End-to-End Data Engineering Pipeline using Databricks & Power BI

You can create folders using (Open a new Notebook in Databricks)

dbutils.fs.mkdirs("/Volumes/workspace/default/financial_de/bronze")

dbutils.fs.mkdirs("/Volumes/workspace/default/financial_de/silver")

dbutils.fs.mkdirs("/Volumes/workspace/default/financial_de/gold")

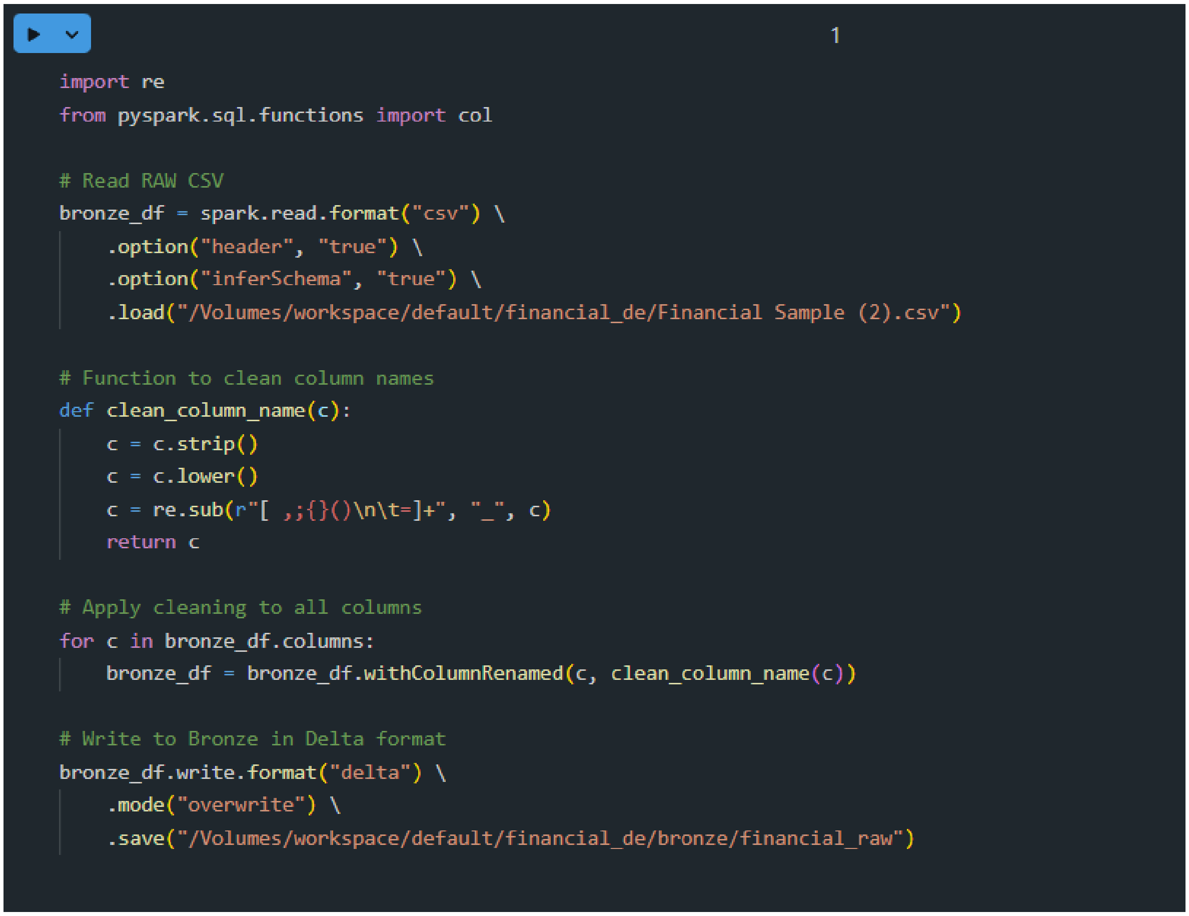
STEP 2 — Notebook 1: 01_ingest_bronze (Bronze Layer)

Purpose

This notebook:

- Reads the raw CSV file
- Cleans column names (removes spaces & special characters)
- Writes data to Delta format in bronze layer

Code



```python
import re
from pyspark.sql.functions import col

# Read RAW CSV
bronze_df = spark.read.format("csv") \
    .option("header", "true") \
    .option("inferSchema", "true") \
    .load("/Volumes/workspace/default/financial_de/Financial Sample (2).csv")

# Function to clean column names
def clean_column_name(c):
    c = c.strip()
    c = c.lower()
    c = re.sub(r"[ ,;{}()\n\t=]+", "_", c)
    return c

# Apply cleaning to all columns
for c in bronze_df.columns:
    bronze_df = bronze_df.withColumnRenamed(c, clean_column_name(c))

# Write to Bronze in Delta format
bronze_df.write.format("delta") \
    .mode("overwrite") \
    .save("/Volumes/workspace/default/financial_de/bronze/financial_raw")
```

Output

A Delta table stored in:

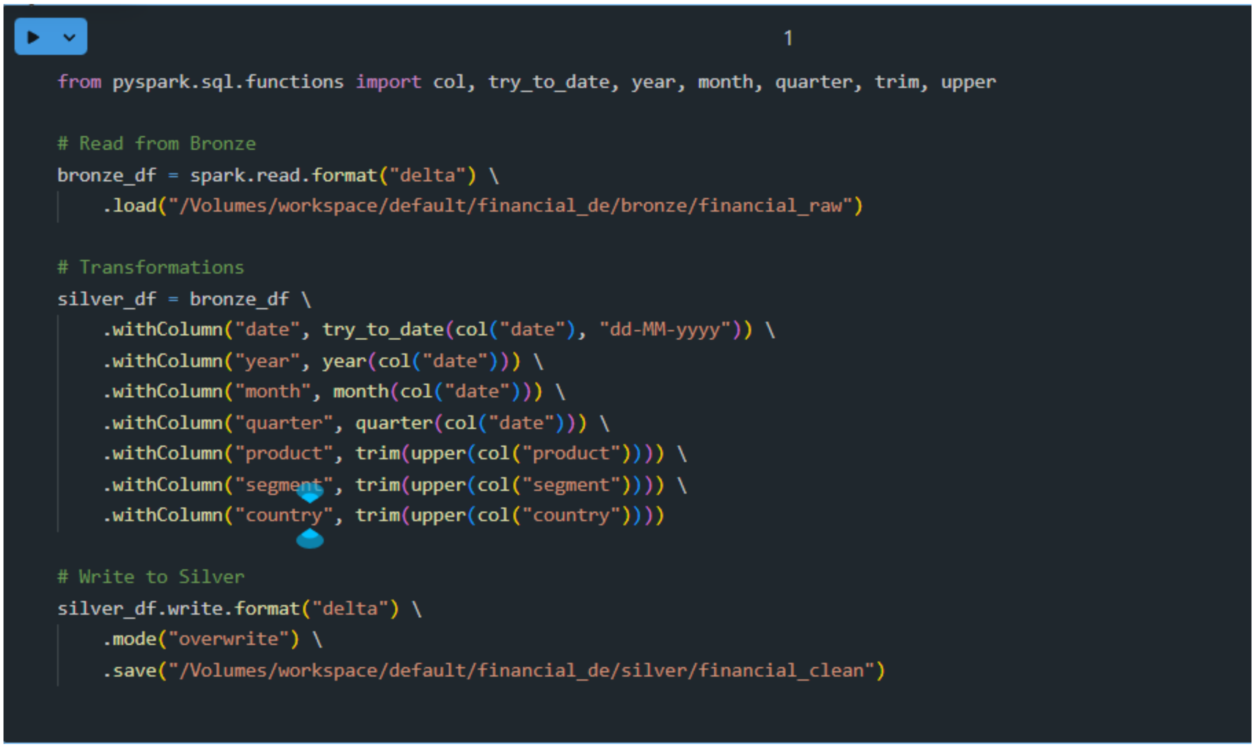
/Volumes/workspace/default/financial_de/bronze/financial_raw

STEP 3 — Notebook 2: 02_transform_silver (Silver Layer)

Purpose

This notebook:

- Reads Bronze data
- Converts date format
- Creates year, month, quarter columns
- Standardizes text columns (uppercase, trimmed)
- Writes cleaned data to silver layer



```
from pyspark.sql.functions import col, try_to_date, year, month, quarter, trim, upper

# Read from Bronze
bronze_df = spark.read.format("delta") \
    .load("/Volumes/workspace/default/financial_de/bronze/financial_raw")

# Transformations
silver_df = bronze_df \
    .withColumn("date", try_to_date(col("date"), "dd-MM-yyyy")) \
    .withColumn("year", year(col("date"))) \
    .withColumn("month", month(col("date"))) \
    .withColumn("quarter", quarter(col("date"))) \
    .withColumn("product", trim(upper(col("product")))) \
    .withColumn("segment", trim(upper(col("segment")))) \
    .withColumn("country", trim(upper(col("country"))))

# Write to Silver
silver_df.write.format("delta") \
    .mode("overwrite") \
    .save("/Volumes/workspace/default/financial_de/silver/financial_clean")
```

Output

Cleaned data stored in:

/Volumes/workspace/default/financial_de/silver/financial_clean

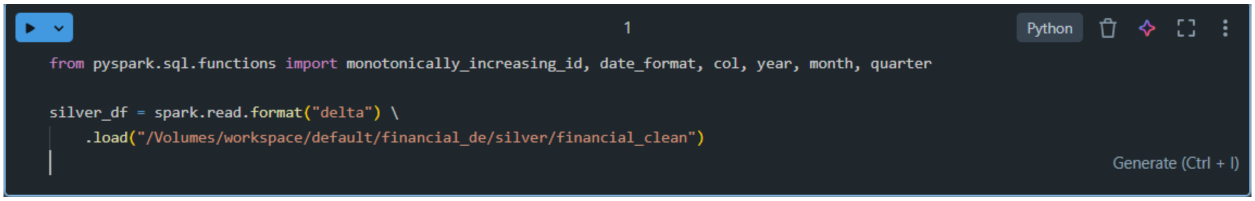
STEP 4 — Notebook 3: 03_create_gold (Gold Layer – Star Schema)

Purpose

Create:

- **Dimension Tables**
- **Fact Table (Star Schema)**

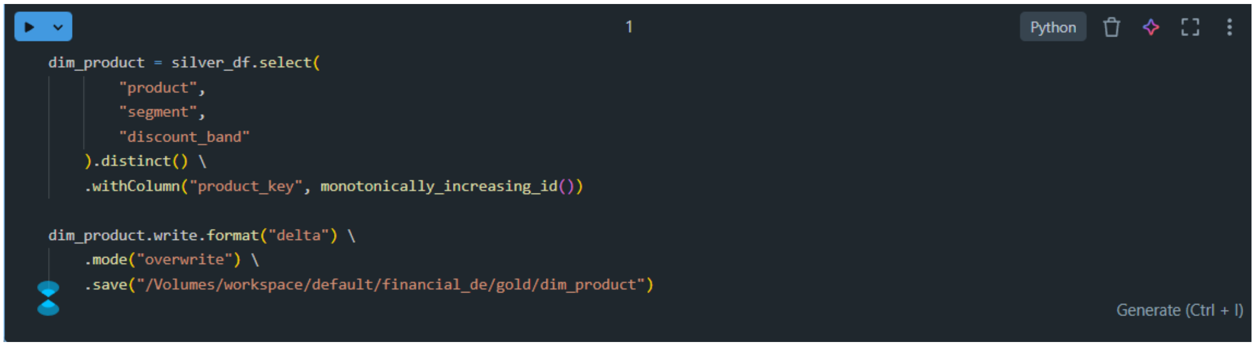
(A) Read Silver Data



```
from pyspark.sql.functions import monotonically_increasing_id, date_format, col, year, month, quarter

silver_df = spark.read.format("delta") \
    .load("/Volumes/workspace/default/financial_de/silver/financial_clean")
```

(B) Create Dim_Product

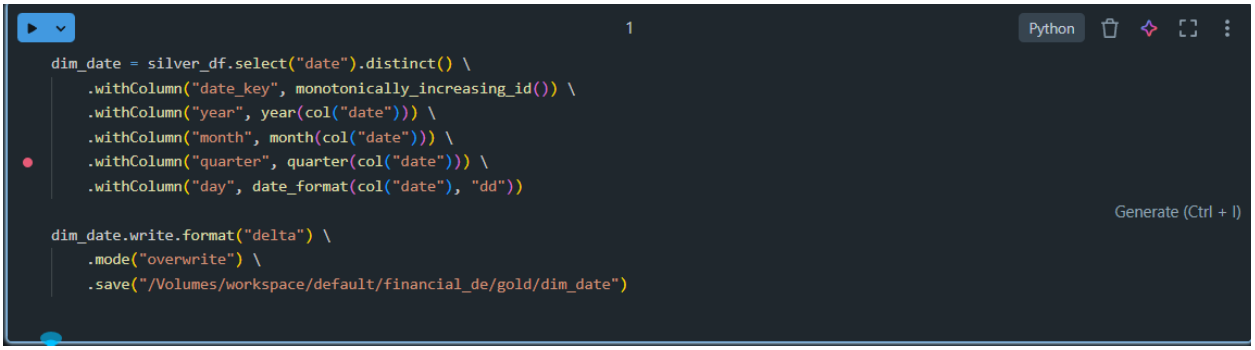


```
dim_product = silver_df.select(
        "product",
        "segment",
        "discount_band"
    ).distinct() \
    .withColumn("product_key", monotonically_increasing_id())

dim_product.write.format("delta") \
    .mode("overwrite") \
    .save("/Volumes/workspace/default/financial_de/gold/dim_product")
```

End-to-End Data Engineering Pipeline using Databricks & Power BI

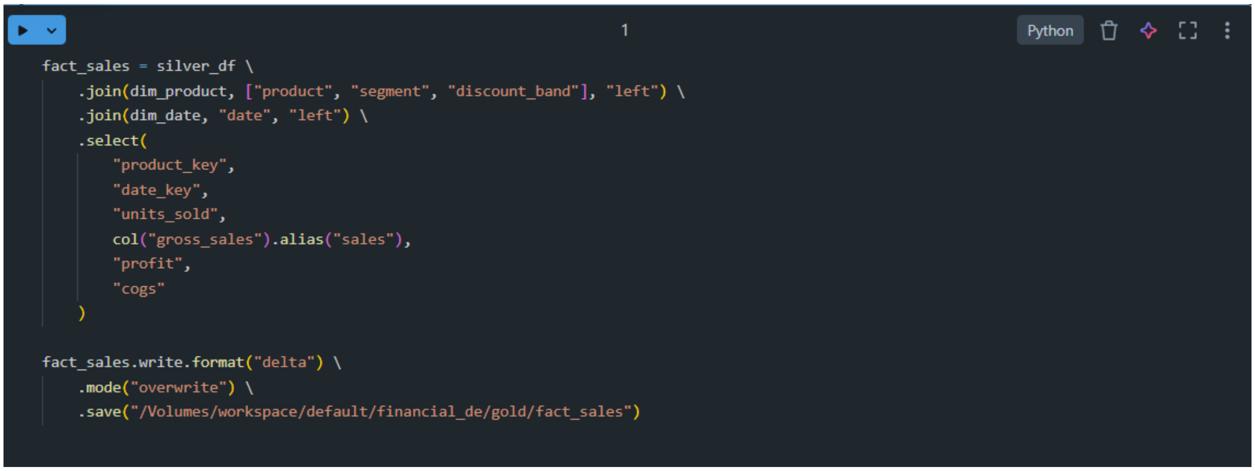
(C) Create Dim_Date



```
dim_date = silver_df.select("date").distinct() \
    .withColumn("date_key", monotonically_increasing_id()) \
    .withColumn("year", year(col("date"))) \
    .withColumn("month", month(col("date"))) \
    .withColumn("quarter", quarter(col("date"))) \
    .withColumn("day", date_format(col("date"), "dd"))

dim_date.write.format("delta") \
    .mode("overwrite") \
    .save("/Volumes/workspace/default/financial_de/gold/dim_date")
```

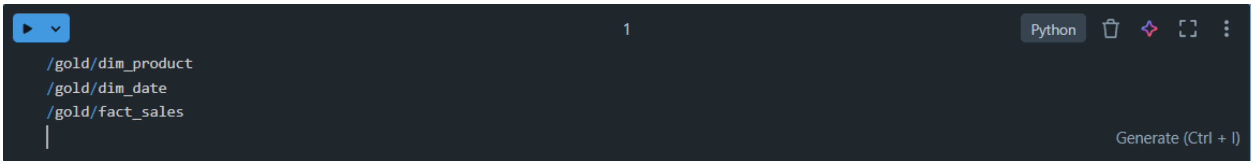
(D) Create Fact_Sales



```
fact_sales = silver_df \
    .join(dim_product, ["product", "segment", "discount_band"], "left") \
    .join(dim_date, "date", "left") \
    .select(
        "product_key",
        "date_key",
        "units_sold",
        col("gross_sales").alias("sales"),
        "profit",
        "cogs"
    )

fact_sales.write.format("delta") \
    .mode("overwrite") \
    .save("/Volumes/workspace/default/financial_de/gold/fact_sales")
```

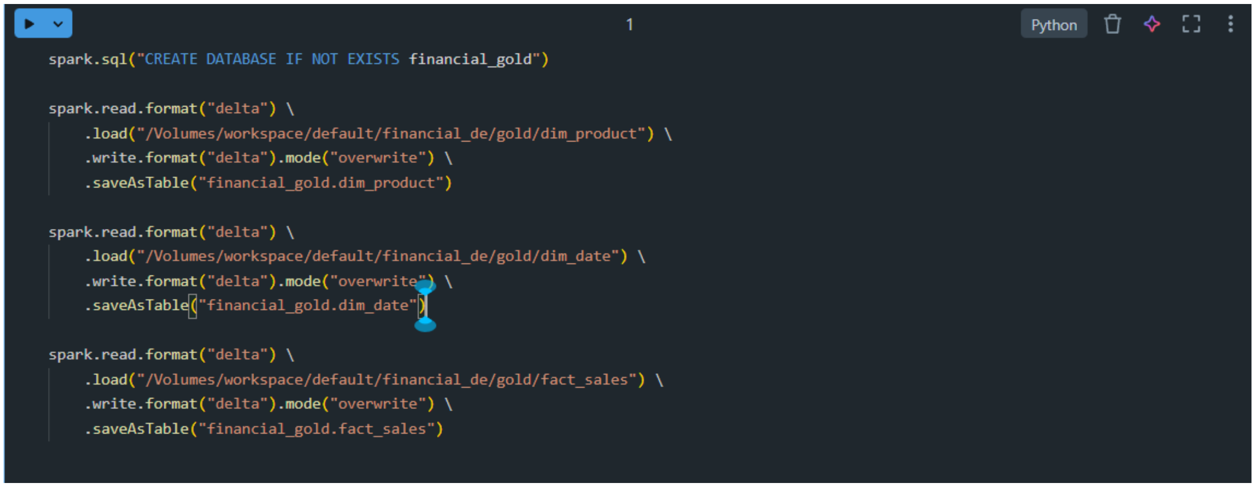
Gold Layer Output



```
/gold/dim_product
/gold/dim_date
/gold/fact_sales
```

End-to-End Data Engineering Pipeline using Databricks & Power BI

STEP 5 — Create SQL Tables in Unity Catalog



```
spark.sql("CREATE DATABASE IF NOT EXISTS financial_gold")

spark.read.format("delta") \
    .load("/Volumes/workspace/default/financial_de/gold/dim_product") \
    .write.format("delta").mode("overwrite") \
    .saveAsTable("financial_gold.dim_product")

spark.read.format("delta") \
    .load("/Volumes/workspace/default/financial_de/gold/dim_date") \
    .write.format("delta").mode("overwrite") \
    .saveAsTable("financial_gold.dim_date")

spark.read.format("delta") \
    .load("/Volumes/workspace/default/financial_de/gold/fact_sales") \
    .write.format("delta").mode("overwrite") \
    .saveAsTable("financial_gold.fact_sales")
```

STEP 6 — Connect Databricks to Power BI

In Databricks

1. Go to SQL → SQL Warehouses
2. Create or start a warehouse
3. Copy:
    - Server Hostname
    - HTTP Path

In Power BI

1. Open Power BI Desktop
2. Click Get Data → Databricks
3. Enter:
    - Server: (Databricks Hostname)
    - HTTP Path: (SQL Warehouse Path)
    - Catalog: workspace
    - Database: financial_gold

4. **Select:**
    - dim_product
    - dim_date
    - fact_sales
5. **Click Load**

STEP 7 — Create Relationships in Power BI (Star Schema)

Go to Model View and create relationships:

Fact Table **Dimension Table**

fact_sales.product_key → dim_product.product_key

fact_sales.date_key → dim_date.date_key

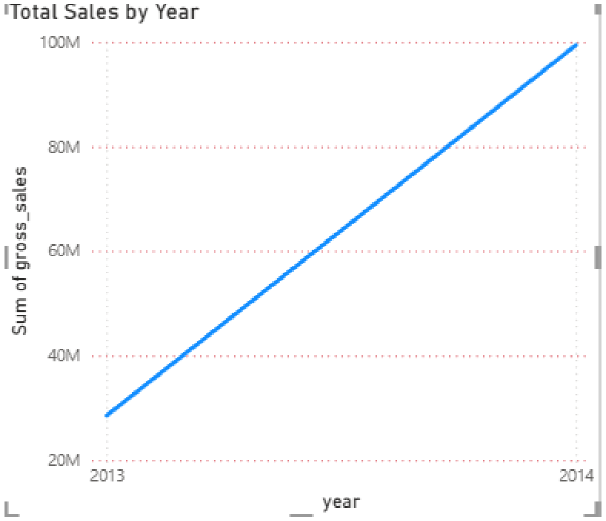
Set relationship type:

- **Many-to-One (* : 1)**

STEP 8 — Build Power BI Dashboard

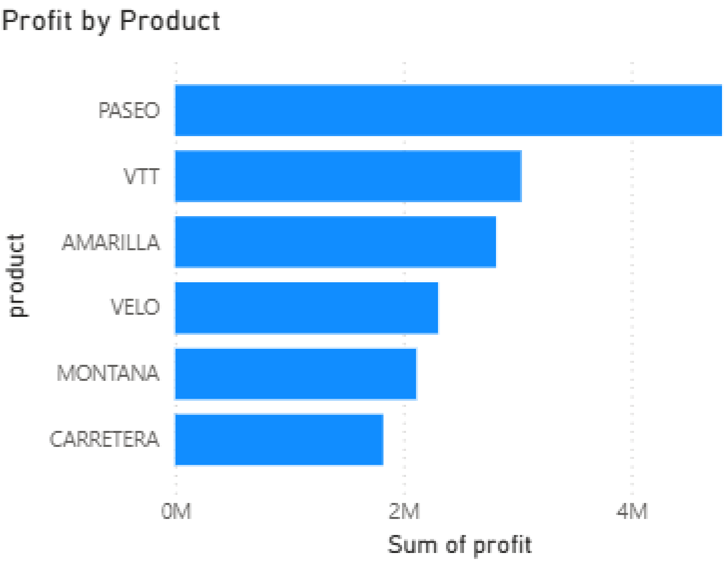
Visual 1 — Total Sales by Year (Line Chart)

- **X-axis: dim_date.year**
- **Y-axis: SUM(fact_sales.sales)**



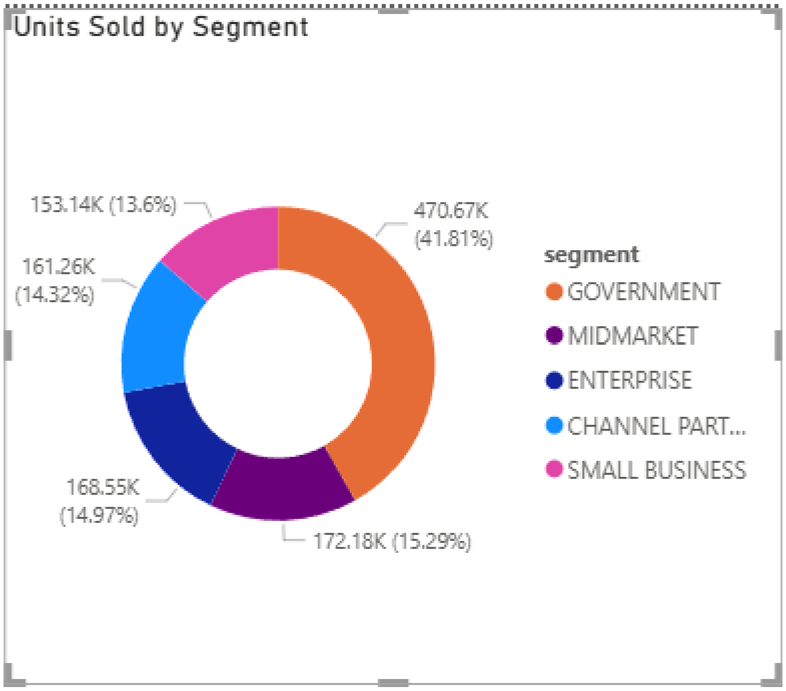
Visual 2 — Profit by Product (Bar Chart)

- **X-axis: dim_product.product**
- **Y-axis: SUM(fact_sales.profit)**



Visual 3 — Units Sold by Segment (Donut Chart)

- **Legend: dim_product.segment**
- **Values: SUM(fact_sales.units_sold)**



9. Business Insights Derived

Using this dashboard, business users can:

- **Track yearly sales trends**
- **Identify top profitable products**
- **Compare performance across segments**
- **Analyze geographic sales distribution**