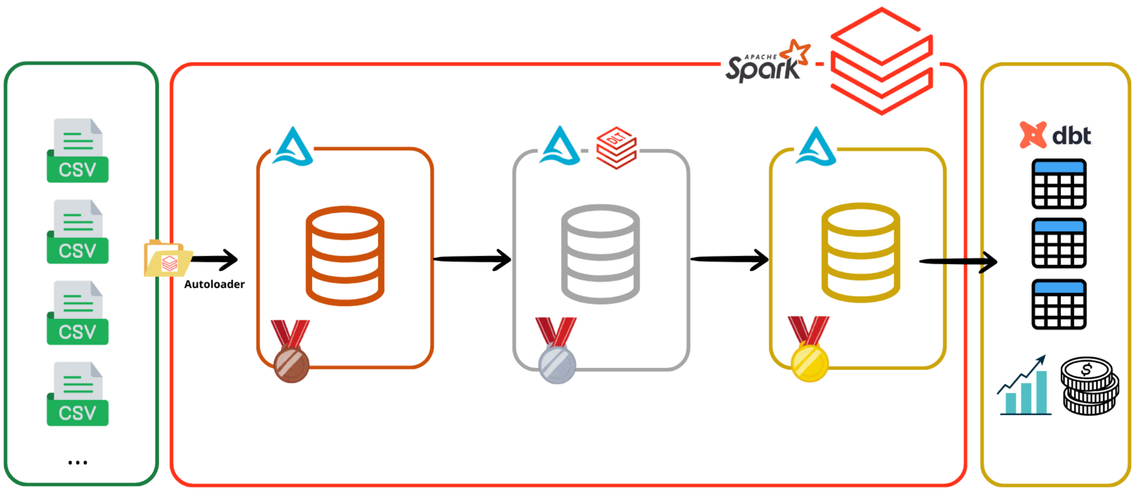
1. Introduction

1.1 Project Objective

This project implements a unified data platform under the **Data Lakehouse paradigm**, using **Databricks** as the execution engine. The architecture is based on the **Delta Lake format** and the **Medallion pattern (Bronze → Silver → Gold)** to process, enrich, and model data coming from multiple transactional and master sources (sales, customers, products, stores, campaigns, etc.), with the purpose of:

- Ensuring data quality, traceability, and consistency.
- Supporting business analysis and advanced reporting through analytical models in **DBT**.
- Enabling incremental loads and managing **Slowly Changing Dimensions (SCD)** and **Change Data Capture (CDC)** to ensure the historical evolution of data.



1.2 Scope

The project covers the full data processing lifecycle in Databricks, from the initial ingestion of CSV files to the publishing of business models in the Gold layer.

The scope includes:

- Definition of schemas and volumes in Databricks.

- Incremental and automated ingestion of data from CSV files into Delta Lake (**Raw → Bronze**) using Autoloader.
- Data cleaning, standardization, application of data quality rules, and **CDC** handling for changes (**Bronze → Silver**).
- Loading of dimensions (static and incremental) in the Gold layer, applying **SCD Type 1** strategies with **MERGE** operations in Delta Lake.
- Incremental loading of the **sales fact table**.
- Construction of analytical models in **DBT** to answer key business questions (sales by campaign, customer segmentation, salesperson performance, etc.).

1.3 General Architecture

The solution is based on Databricks' **Medallion Architecture**, structured in four layers:

- **Raw**: storage of raw source files, unmodified.
- **Bronze**: automated ingestion with Databricks Autoloader, generating Delta tables with flexible schema.
- **Silver**: data type standardization, application of business rules, and generation of clean, consistent tables.
- **Gold**: construction of dimensional and fact tables optimized for analysis, along with business models generated in DBT.

This architecture enables a scalable, incremental, and reliable data flow, ensuring that information consumers (analysts, data scientists, business areas) have access to consistent, analysis-ready data for dashboards or ad-hoc queries.

The entire architecture is implemented on **Unity Catalog**, which organizes objects under the **catalog.schema.object** hierarchy. This ensures standardized access, lineage traceability, and centralized data governance in Databricks.

---

2. Data Architecture (Medallion)

The solution is designed following the **Medallion Architecture** pattern proposed by Databricks.

This approach organizes the data flow into successive layers (**Raw → Bronze → Silver → Gold**), each with clear transformation goals and rules, ensuring scalability, cleanliness, and consistency of information.

2.1 Raw Layer

Objective: Store raw data as received from the source (CSV files).

Implementation:

- A volume is created in Databricks (rawvolume) to store the source files.

Files are organized into domain folders:

sales/
stores/
customers/
products/
salespersons/
campaigns/
dates/
times/

- No transformations or validations are applied in this layer.

Benefit: Serves as a backup of the original data, guaranteeing traceability and allowing full reprocessing if needed.

2.2 Bronze Layer

Objective: Standardize the ingestion of raw data into **Delta format**, enabling schema control and incremental loading.

Implementation:

- Databricks Autoloader is used to automatically load files from Raw.
- Data is persisted as **Delta Tables** in the **bronzevolume**, providing the foundation for **ACID transactions** and **time travel**.
- Checkpoints are applied to support incremental ingestion and **schema evolution**.

Characteristics:

- Semi-cleaned data, but may still contain nulls or inconsistencies.
- Supports incremental processing using **cloudFiles**.

Benefit: Guarantees efficient and scalable storage without losing unvalidated data.

2.3 Silver Layer

Objective: Transform and clean data from Bronze, applying business rules and ensuring proper typing.

Implementation:

- **Lakeflow Declarative Pipelines (previously known as Delta Live Tables (DLT))** is used to orchestrate the pipeline, ensuring quality management and dependencies.
- Data type conversions (e.g., dates, integers, surrogate keys).
- Integrity and consistency validations.
- Separation into **silver_*** tables (e.g., **silver_sales, silver_customers, silver_products**).

Characteristics:

- Delta tables containing consistent and standardized data, ready to be consumed by the Gold layer.
- **CDC (Change Data Capture)** control to identify new or modified records.
- Data quality rules applied to the fact table.

Example:
In **LoadStaticDimensions.ipynb**, the **dates** and **times** columns are properly typed before being loaded into Gold.

2.4 Gold Layer

Objective: Generate optimized analytical structures (facts and dimensions).

Implementation:

- **Static Dimensions**: DimDates, DimTimes.
- **Incremental Dimensions (SCD)**: DimStores, DimCustomers, DimProducts, DimSalespersons, DimCampaigns.
- **Fact Table**: FactSales, with incremental loading and linked to all dimensions.

Characteristics:

- Dimensional model in Delta Lake ready for analysis and reporting.
- Each dimension manages its **surrogate key** (*_sk) to ensure referential integrity.
- **CDC** is implemented in dimensions and facts via **MERGE operations** in Delta Lake.
- Dimensions use **SCD Type 1**, overwriting historical data to keep only the most up-to-date version of each entity (customer, product, campaign, etc.).
- The fact table implements an **incremental process (upsert with CDC)**, ensuring that new sales or modifications are efficiently reflected in FactSales.

Benefit:

- Dimensional data ready for analysis and reporting.
- Additional business models are built in **DBT Cloud**, which queries Gold tables to deliver insights (e.g., daily sales, campaign performance, customer segmentation).

---

3. Data Sources

The project is fed by a set of **CSV files** that represent key business entities (sales, customers, products, campaigns, etc.).
These files serve as the initial source of data and are stored in the **Raw layer** before being ingested and transformed within the Medallion flow.

The original data comes from **Kaggle (Retail Store Star Schema Dataset)**, with modifications to fit the project.

3.1 Received CSV Files

Main Dimensions

Customers:

- **dim_customers.csv** → initial customer dataset.
- **dim_customers_increment.csv** → incremental records (new or modified).
- **dim_customers_scd.csv** → extended version for Slowly Changing Dimensions (SCD) testing.

Products:

- **dim_products.csv** → initial product dataset.
- **dim_products_increment.csv** → incremental records.
- **dim_products_scd.csv** → dataset for SCD testing on products.

Stores:

- **dim_stores.csv** → initial store dataset.
- **dim_stores_increment.csv** → incremental records.
- **dim_stores_scd.csv** → dataset for SCD testing on stores.

Salespersons:

- **dim_salespersons.csv** → initial salespersons dataset.
- **dim_salespersons_increment.csv** → incremental records.
- **dim_salespersons_scd.csv** → dataset for SCD testing on salespersons.

Campaigns:

- **dim_campaigns.csv** → initial campaigns dataset.
- **dim_campaigns_increment.csv** → incremental records.
- **dim_campaigns_scd.csv** → dataset for SCD testing on campaigns.

Static Dimensions

- **Dates**: **dim_dates.csv** → pre-generated calendar table.

- **Times**: **dim_times.csv** → pre-generated time table (hours and minutes of the day).

Facts

- **Sales**:
    - **fact_sales.csv** → initial sales transaction dataset.
    - **fact_sales_increment.csv** → incremental sales records.

3.2 Source Tables Definition

Each file corresponds to a business entity that is later transformed into Silver and Gold tables:

- Customers → **silver_customers** → **DimCustomers**
- Products → **silver_products** → **DimProducts**
- Stores → **silver_stores** → **DimStores**
- Salespersons → **silver_salespersons** → **DimSalespersons**
- Campaigns → **silver_campaigns** → **DimCampaigns**
- Dates → **silver_dates** → **DimDates**
- Times → **silver_times** → **DimTimes**
- Sales → **silver_sales** → **FactSales**

3.3 Load Parameters (SrcParameters)

The notebook **SrcParameters.ipynb** defines the array of sources feeding the Bronze layer:

```
src_array = [
   {"src": "sales"},
   {"src": "stores"},
   {"src": "customers"},
   {"src": "products"},
   {"src": "salespersons"},
   {"src": "campaigns"},
   {"src": "dates"},
   {"src": "times"}
]
```

This array allows the Bronze ingestion process to be **dynamic**, without hardcoding each table.

- Each src value corresponds to a folder in the **Raw layer**, which contains the CSV files for that source.

---

4. ETL Processes per Layer

The data flow is implemented through a series of **Databricks notebooks**, orchestrating the Raw → Bronze → Silver → Gold load, following the **ETL (Extract, Transform, Load)** pattern.
Each layer fulfills a specific role in ensuring data quality and availability.

4.1 Initial Setup (Schemas and Volumes)

Notebook: Setup.ipynb

- Create schemas for each layer:
  - **workspace.raw**
  - **workspace.bronze**
  - **workspace.silver**
  - **workspace.gold**
- Define the **Raw volume (rawvolume)** to store the source files.

Create specific directories for each source inside the Raw volume:

/Volumes/workspace/raw/rawvolume/rawdata/sales
/Volumes/workspace/raw/rawvolume/rawdata/stores

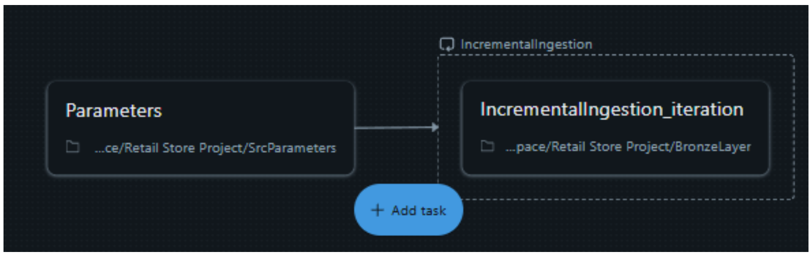
/Volumes/workspace/raw/rawvolume/rawdata/customers
...

This initializes the base structure of the **Data Lakehouse**.

4.2 Ingestion Raw → Bronze

Notebook: Bronze.ipynb

- Data loading into Bronze can be done **manually** or via a **for each Job** in Databricks.
- This job dynamically iterates over the configurations defined in **srcparameters**, allowing parameterized ingestion without hardcoding specific tables.


IncrementalIngestion
Parameters
...ce/Retail Store Project/SrcParameters
IncrementalIngestion_iteration
...pace/Retail Store Project/BronzeLayer
+ Add task

Implementation:

- Use **Databricks Autoloader (cloudFiles)** to ingest CSV files from Raw.
- Create **Delta tables** in the Bronze layer within **bronzevolume.**
- Configure checkpoints to handle:
  - **Incrementality** → only new files are loaded.
  - **Schema evolution** (**schemaEvolutionMode = rescue**).

Example process:

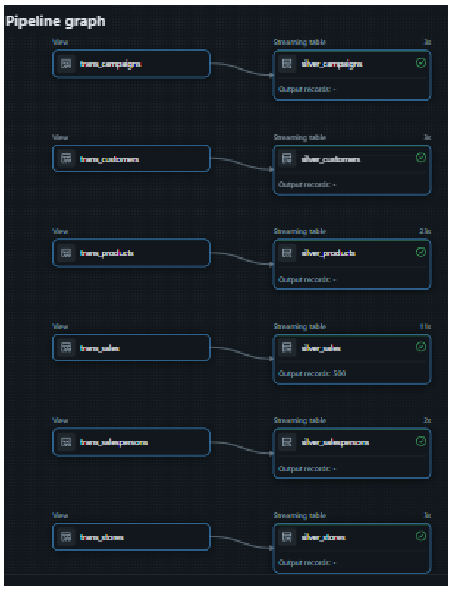
- CSV files from **customers** are moved from Raw to **bronzevolume/customers**.
- The Delta table **workspace.bronze.customers** is generated.

At this stage, the data is already in Delta Lake, although it may still contain errors or dirty values.

4.3 Transformation Bronze → Silver

Notebook: DLT_pipeline.py

The pipeline must be executed:



1. **Data Reading from Bronze**
    - Read streaming data from Bronze Delta tables (products, customers, stores, salespersons, campaigns, and sales).
2. **Cleaning and Typing**
    - Remove unnecessary columns (**_rescued_data**).
    - Cast columns to proper types (e.g., primary keys → **IntegerType**).
    - Add a **modified_date** column with the current timestamp.
    - For the **silver_sales** table, apply data quality rules with **@dlt.expect_all_or_drop()**.
3. **DLT View Creation**
    - Create **Delta Live Table views** (**@dlt.view()**) for each entity: products, customers, stores, salespersons, campaigns, and sales.
4. **Change Data Capture (CDC) with SCD Type 1**

   - Implement a CDC flow using **dlt.create_auto_cdc_flow()** to handle updates based on the **modified_date** column.
   - Apply **SCD Type 1** (overwrite modified records).

5. **Generated Silver Tables**
   - Silver tables created for each entity:
     - **silver_products**
     - **silver_customers**
     - **silver_stores**
     - **silver_salespersons**
     - **silver_campaigns**
     - **silver_sales**

4.4 Loading Static Dimensions

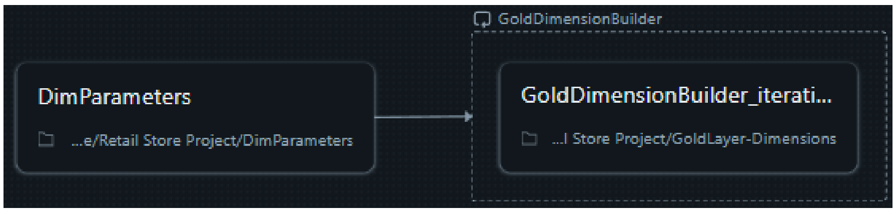
Notebook: LoadStaticDimensions.ipynb

- Process **dates** and **times**, which are static/semi-static tables.
- Flow: Raw → Bronze (Delta) → Silver (typing) → Gold (**DimDates, DimTimes**).
- Properly cast each column to its correct type (**date, int, string**).

These dimensions do not require incremental loading, since they remain unchanged over time.

4.5 Loading Incremental Dimensions (SCD)

The generation of Dim and Fact tables in Gold can also be executed **manually** or through a **for each Job** in Databricks.
This job uses the **dimparameters** file to dynamically iterate over all dimensions that must be built.


GoldDimensionBuilder
DimParameters
...e/Retail Store Project/DimParameters
GoldDimensionBuilder_iterati...
...l Store Project/GoldLayer-Dimensions

Notebooks:

- **GoldLayer-Dimensions.ipynb**
- **DimParameters.ipynb**

SCD Type 1 is applied in:

- **DimStores**
- **DimCustomers**
- **DimProducts**
- **DimSalespersons**
- **DimCampaigns**

Strategy:

- Calculate **last_load** based on the maximum value in **modified_date**.
- Filter Silver records modified after **last_load.**
- Compare with the Gold dimension.
  - **New records** → insert (**create_date** and **update_date** set to current timestamp).
  - **Modified existing records** → update (overwrite with **update_date**).

This ensures that dimensions reflect the historical state while maintaining traceability.

4.6 Fact Table Load

Notebook: GoldLayer-Fact.ipynb

The **FactSales** table is loaded incrementally.

Strategy:

- Retrieve **last_load** from Gold (FactSales).
- Filter new transactions or modifications in **silver_sales** using **modified_date**.
- Enrich data with all dimensions (DimProducts, DimStores, DimCustomers, etc.) via **joins**.
- Perform a **Delta Lake MERGE**:
    - **Insert** if it is a new sale.

This keeps the fact table always updated and consistent with the dimensions.

---

5. Business Models (DBT)

Once data reaches the **Gold layer** (dimensional model with facts and dimensions), models built in **DBT Cloud** are used to answer business questions and generate analytical views.

These models allow business areas to query aggregated and segmented information without dealing with the technical complexity of ETL processes.

5.1 List of Created Models

The main business models defined in DBT are:

- **campaignsalessummary** → campaign sales summary.
- **customersegmentsalessummary** → sales analysis by customer segment.
- **dailycampaignsales** → daily campaign sales.
- **dailymonthlysalessummary** → daily and monthly sales summary.

- **dailystoresales** → daily sales by store.
- **monthlysalesbycategory** → monthly sales by product category.
- **monthlysalesbysalesperson** → monthly sales by salesperson.
- **salespersonsalessummary** → overall salesperson performance.
- **salessummarybycategorybrand** → sales by product category and brand.
- **storesalessummary** → store sales summary.

5.2 Business Questions Addressed

Each model answers key business questions for decision-making:

Campaigns:

- What was the impact of each campaign on total sales?
- Which campaigns generate the highest return?

Customers:

- Which customer segments generate the most revenue?
- Can purchasing patterns be identified by customer profile?

Daily and Monthly Sales:

- How do sales evolve day by day and month by month?
- Are there clear seasonalities or trends?

Stores:

- Which stores perform best in sales?
- Are there significant differences across regions?

Products:

- Which categories and brands generate the most revenue?
- Which products have the best turnover?

Salespersons:

- Who are the most productive salespersons?
- What impact does each salesperson have on active campaigns?

---

6. Technical Considerations

Several technical factors must be considered to ensure **scalability, reliability, and maintainability** of the system.

6.1 Use of Delta Lake

All Bronze, Silver, and Gold tables are implemented in **Delta Lake** format.

Key Benefits:

- **ACID transactions**: guaranteed consistency for reads and writes.
- **Time Travel**: ability to query historical versions.
- **Optimized MERGE**: native support for SCD and CDC.
- **Storage optimization**: automatic compaction and small file handling.

6.2 Databricks Autoloader

For Raw → Bronze ingestion, **Autoloader with cloudFiles** is used.

Advantages:

- Incremental handling of new files.
- **Schema inference and evolution**, allowing changes in data structure.

- Use of **checkpoints**, preventing duplicate loads and maintaining traceability.

6.3 Unity Catalog

All schemas and tables are managed under **Unity Catalog**, using the **catalog.schema.object** hierarchy.

Key Benefits:

- Centralized governance and fine-grained access control.
- Unified data lineage and traceability across all layers.
- Standardized organization of objects, simplifying maintenance and scalability.

6.4 Incremental Load and SCD Strategies

- In **Bronze**, ingestion jobs use **Autoloader and checkpointing**, ensuring **exactly-once processing** and avoiding data reprocessing.
- In **Silver**, the **Lakeflow Declarative Pipeline** implements **SCD Type 1** via create_auto_cdc_flow, overwriting old values with the most recent data based on temporal sequence.
- In **Gold**, incremental loads are implemented using **MERGE in Delta Lake** for both dimensions (SCD Type 1) and facts (incremental upsert).
- The process includes a **backdate_refresh mechanism** in Gold, allowing controlled historical reprocessing when data corrections are required.

6.5 Maintainability and Scalability

- Load orchestration is fully **parameterized and dynamic** using **for each Jobs** in Databricks.

- The Bronze Job is driven by **srcparameters**, enabling ingestion of multiple data sources in parallel through a single generic notebook.
- The Silver pipeline is implemented with **DLT**, allowing transformations and quality rules to be defined declaratively. Thanks to DLT, the process ensures:
  - Full traceability.
  - Automatic dependency management.
  - Native monitoring.
  - Reliable dataset generation without extra orchestration logic.
- The Gold Job uses **dimparameters**, dynamically iterating over all dimensions to be built.

This design **decouples business logic from code**, facilitating scalability (adding new sources or dimensions without modifying notebooks) and ensuring system maintainability.

6.6 Integration with DBT

- DBT connects directly to the **Databricks SQL cluster**.
- DBT models are built on top of **Gold tables**, ensuring a clear separation between **ETL (Databricks)** and **analytical modeling (DBT)**.
- Enables versioning, deployment across environments, and centralized documentation of models.

---

7. Conclusions

The project implemented in **Databricks** demonstrates how a **Data Lakehouse architecture**, based on the **Delta Lake format** and the **Medallion pattern (Raw → Bronze → Silver → Gold)**, enables the construction of a unified data platform that is reliable, scalable, and flexible—overcoming the limitations of traditional Data Warehouse approaches.

Key achievements:

- **Robust ingestion**: using Databricks Autoloader with incremental loading and schema evolution.

- **Data quality**: cleaning, typing, and validation in Silver before publishing to Gold.

- **Optimized model**: with facts (FactSales) and dimensions (DimCustomers, DimProducts, DimStores, etc.) designed for efficient analysis.

- **Change management**: implementation of **SCD Type 1** for dimensions and **incremental loading** for facts, ensuring efficient updates of the data model by processing only identified changes.

- **Semantic layer in DBT**: business models that abstract technical complexity and enable direct analysis for business users.

- **Technical best practices**: use of Delta Lake.

Overall, the system allows organizations to answer **key questions** regarding marketing, sales, customers, and store performance in a fast and reliable manner.

---

8. Next Steps

While the current solution meets the initial objectives, there are areas of improvement and expansion that can be addressed in future phases:

8.1 Data Governance

- Implement more advanced **Data Quality Rules** in Silver (e.g., validation against master catalogs, stricter business rules).

8.2 Expansion of Use Cases

- Implement **Machine Learning models** on Silver and Gold (e.g., churn prediction, sales forecasting).

- Create **dashboards in Power BI or Tableau** connected to DBT models for executive visualization.