# Logistics Cost & Delivery Analytics Platform

## Capstone Project Group 1

### 1. Project Overview

The Logistics Cost & Delivery Analytics project is an end-to-end data engineering capstone designed to simulate a real-world enterprise logistics analytics platform. The project focuses on building a scalable, reliable, and governed data pipeline capable of transforming raw operational shipment data into meaningful business insights for leadership and operations teams. The solution processes daily shipment extracts generated by regional warehouses and integrates them with reference data such as warehouses, carriers, and regions. These datasets contain common real-world data challenges including missing values, duplicate records, cost outliers, and delivery anomalies. Addressing these challenges is a core objective of the project, ensuring that downstream analytics are based on accurate and trustworthy data.

The architecture follows the Medallion pattern (Bronze, Silver, and Gold layers), which enables progressive data refinement and clear separation of responsibilities across ingestion, transformation, and analytics layers. Raw data is first ingested into the Bronze layer without modification to preserve source integrity. The Silver layer applies data cleansing, standardization, deduplication, and data quality validation using Delta Lake. Finally, the Gold layer presents a business-ready star schema optimized for reporting and analytics. Automation and scalability are key design considerations in this project. Azure Data Factory orchestrates batch ingestion with incremental watermark logic to ensure efficient daily processing and rerun capability.

Azure Databricks provides a distributed processing environment for transformations, data quality enforcement, and dimensional modeling. Power BI consumes the curated Gold layer to deliver interactive dashboards and KPIs for shipment cost governance, delivery performance, carrier SLA compliance, and data quality monitoring. Overall, this project demonstrates the application of modern data engineering best practices using Azure-native services. It showcases how cloud-based analytics platforms can be designed to handle imperfect data, support incremental growth, and deliver actionable insights to business stakeholders in a logistics domain.

## 2. Abstract

The **Logistics Cost & Delivery Analytics** capstone project presents a comprehensive cloud-based data engineering solution designed to analyze shipment costs, delivery performance, and data quality metrics across regional logistics operations. The project simulates a real-world logistics analytics scenario in which daily shipment data is ingested from multiple warehouses and processed using a scalable and governed architecture. The solution is built using Azure Data Lake Storage Gen2, Azure Data Factory, Azure Databricks, Delta Lake, and Power BI. Raw CSV datasets are ingested into the Bronze layer through automated batch pipelines with incremental watermarking. The data is then cleansed, standardized, deduplicated, and validated in the Silver layer, where data quality rules identify null values, duplicates, outliers, and delivery anomalies. Invalid or non-conforming records are isolated into a quarantine zone to preserve data integrity. In the Gold layer, the processed data is modeled into a star schema consisting of fact and dimension tables optimized for analytical workloads. Surrogate keys and incremental merge strategies are implemented to support efficient updates and high query performance. The curated dataset is consumed by Power BI to create interactive dashboards that provide insights into shipment cost trends, on-time delivery performance, carrier SLA compliance, fragile shipment handling, and overall data quality health. This project demonstrates the practical application of modern data engineering principles, including Medallion Architecture, incremental data processing, data quality governance, and business intelligence integration. The final outcome is a production-ready analytics platform that delivers trusted, high-quality insights to stakeholders and serves as a strong foundation.

## 3. Business Problem Statement

Regional warehouses submit daily shipment data containing operational and financial metrics. However, leadership faced challenges due to:

- Missing or invalid shipment costs
- Duplicate shipment records
- Delivery date anomalies
- Lack of a centralized, trusted reporting dataset

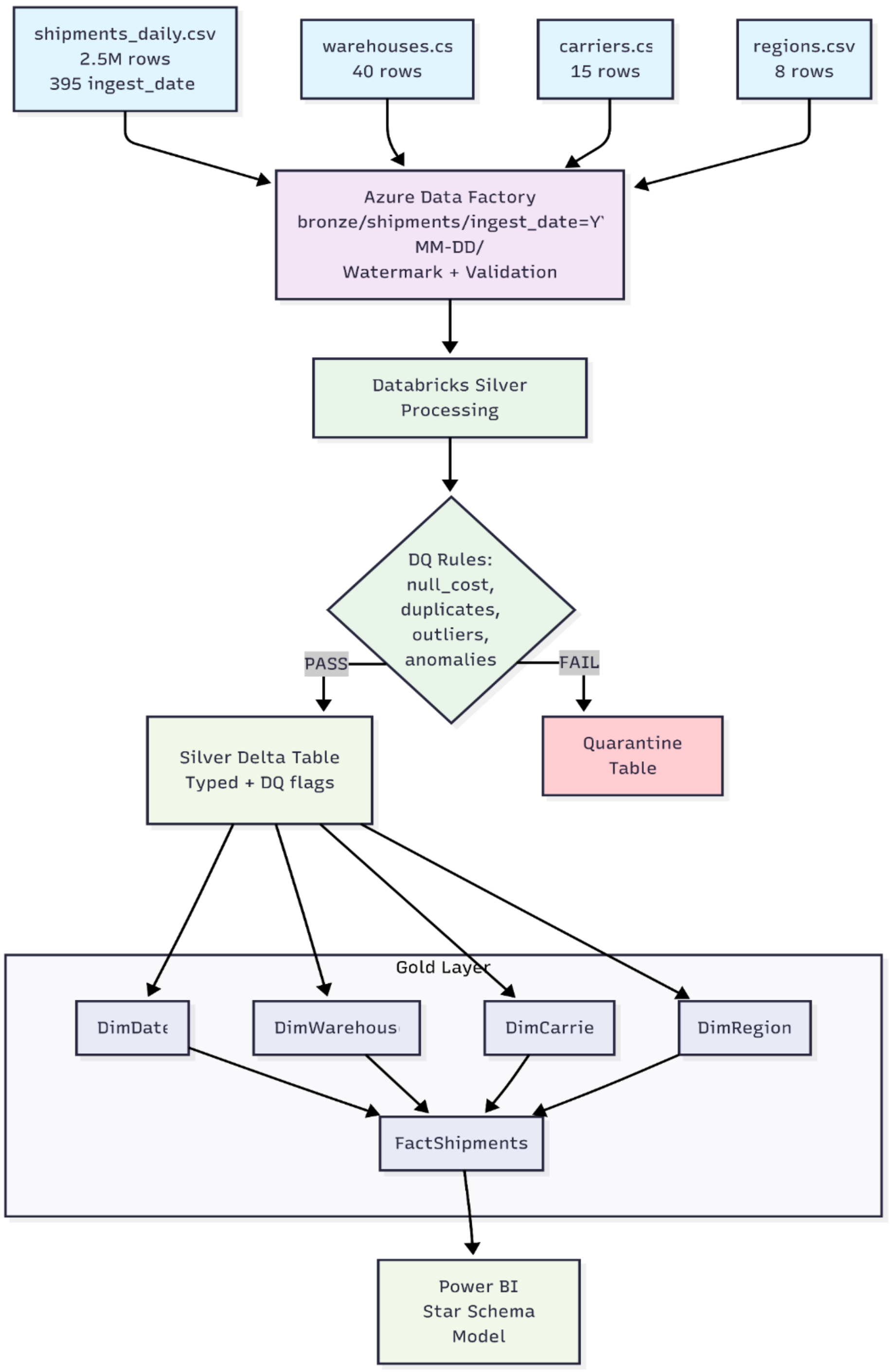
Business Objectives

- Enforce shipment cost governance
- Track delivery performance by region, warehouse, and carrier

- Monitor fragile shipment handling and SLA compliance
- Provide a trusted semantic model for Power BI reporting

## 4. Project Execution Plan

### System Architecture Overview



The **Logistics Cost & Delivery Analytics** platform is built using a modern, cloud-native architecture on Microsoft Azure. The system is designed to support scalable data ingestion, robust data quality enforcement, incremental processing, and high-performance analytics. The architecture follows industry best practices and is aligned with the **Medallion Architecture pattern (Bronze, Silver, and Gold layers)** to ensure data reliability and maintainability.

The solution integrates multiple Azure services, each responsible for a specific role in the data lifecycle, from raw data ingestion to business intelligence reporting.

---

## High-Level Architecture Description

At a high level, the system architecture consists of four main layers:

1. **Data Source Layer**
2. **Data Ingestion & Orchestration Layer**
3. **Data Processing & Storage Layer**
4. **Analytics & Visualization Layer**

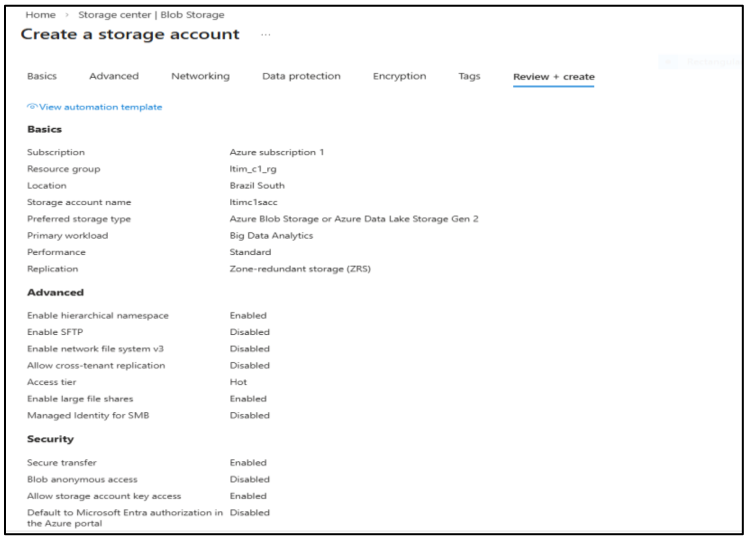
Each layer is loosely coupled, enabling independent scaling, easier troubleshooting, and clear separation of responsibilities.

---

## Sprint 0: Architecture & Setup

Sprint 0 focuses on establishing the foundational architecture required for building a scalable, enterprise-grade logistics analytics platform. This sprint covers data sourcing, cloud environment setup, storage organization, and initial service configuration using Azure-native components. The data source layer represents the operational systems responsible for generating logistics data. In this project, regional warehouses act as the primary source systems, producing daily shipment extracts in CSV format. These files contain detailed shipment transaction records along with supporting reference data such as warehouse information, carrier details, and regional mappings.

In addition to scheduled batch data, optional micro-batch delivery event files are introduced to simulate near-real-time updates to shipment delivery statuses. This approach reflects real-world logistics environments where delivery confirmations and status changes may arrive asynchronously after shipment creation. The original project specification provided approximately 3,000 records to represent 30 days of shipment activity. To better simulate enterprise-scale data processing and validate system scalability, the dataset was synthetically expanded to approximately 2.5 million records. This data generation approach allowed the platform to be tested under higher data volumes while preserving realistic data characteristics



Create a storage account

**Basics**

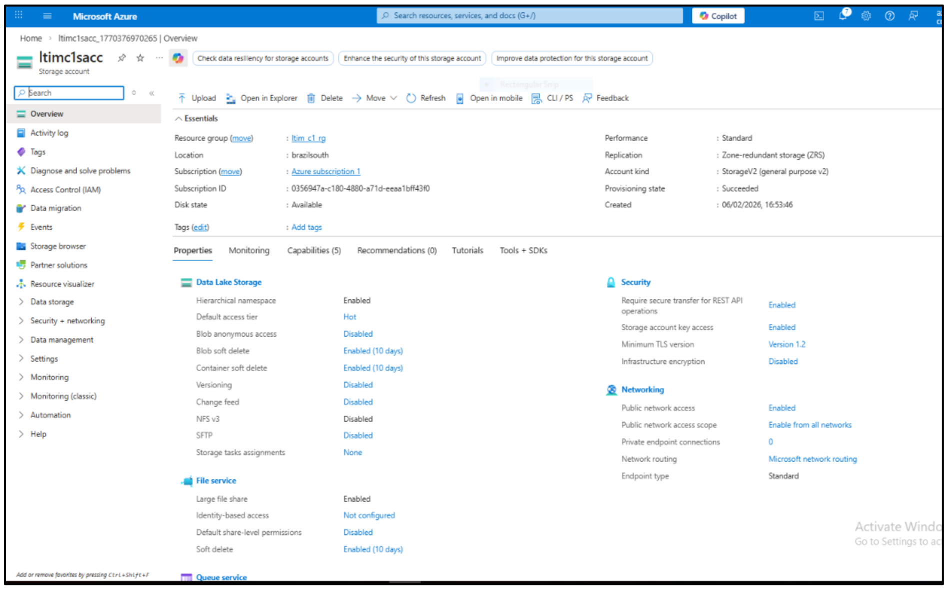
| | |
|---|---|
| Subscription | Azure subscription 1 |
| Resource group | ltim_c1_rg |
| Location | Brazil South |
| Storage account name | ltimc1sacc |
| Preferred storage type | Azure Blob Storage or Azure Data Lake Storage Gen 2 |
| Primary workload | Big Data Analytics |
| Performance | Standard |
| Replication | Zone-redundant storage (ZRS) |

**Advanced**

| | |
|---|---|
| Enable hierarchical namespace | Enabled |
| Enable SFTP | Disabled |
| Enable network file system v3 | Disabled |
| Allow cross-tenant replication | Disabled |
| Access tier | Hot |
| Enable large file shares | Enabled |
| Managed Identity for SMB | Disabled |

**Security**

| | |
|---|---|
| Secure transfer | Enabled |
| Blob anonymous access | Disabled |
| Allow storage account key access | Enabled |
| Default to Microsoft Entra authorization in the Azure portal | Disabled |



ltimc1sacc
Storage account

Essentials

| | | | |
|---|---|---|---|
| Resource group (move) | : ltim_c1_rg | Performance | : Standard |
| Location | : brazilsouth | Replication | : Zone-redundant storage (ZRS) |
| Subscription (move) | : Azure subscription 1 | Account kind | : StorageV2 (general purpose v2) |
| Subscription ID | : 0356947a-c180-4880-a71d-eeaa1bff43f0 | Provisioning state | : Succeeded |
| Disk state | : Available | Created | : 06/02/2026, 16:53:46 |
| Tags (edit) | : Add tags | | |

Data Lake Storage

| | |
|---|---|
| Hierarchical namespace | Enabled |
| Default access tier | Hot |
| Blob anonymous access | Disabled |
| Blob soft delete | Enabled (10 days) |
| Container soft delete | Enabled (10 days) |
| Versioning | Disabled |
| Change feed | Disabled |
| NFS v3 | Disabled |
| SFTP | Disabled |
| Storage tasks assignments | None |

File service

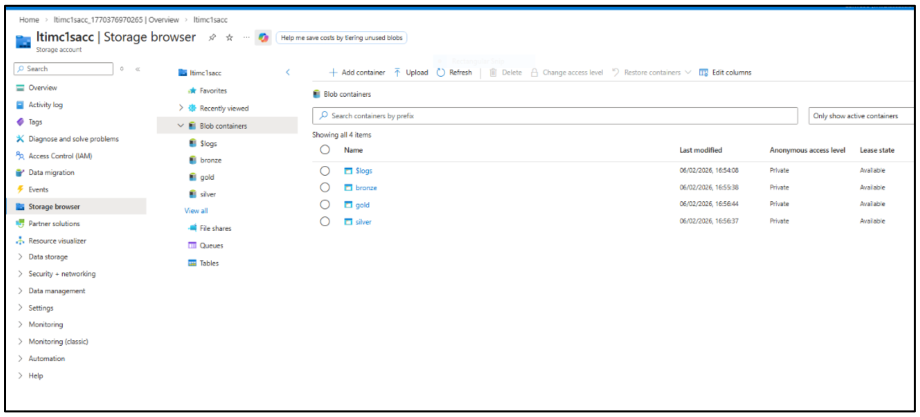
| | |
|---|---|
| Large file share | Enabled |
| Identity-based access | Not configured |
| Default share-level permissions | Disabled |
| Soft delete | Enabled (10 days) |

Security

| | |
|---|---|
| Require secure transfer for REST API operations | Enabled |
| Storage account key access | Enabled |
| Minimum TLS version | Version 1.2 |
| Infrastructure encryption | Disabled |

Networking

| | |
|---|---|
| Public network access | Enabled |
| Public network access scope | Enable from all networks |
| Private endpoint connections | 0 |
| Network routing | Microsoft network routing |
| Endpoint type | Standard |



ltimc1sacc | Storage browser
Storage account

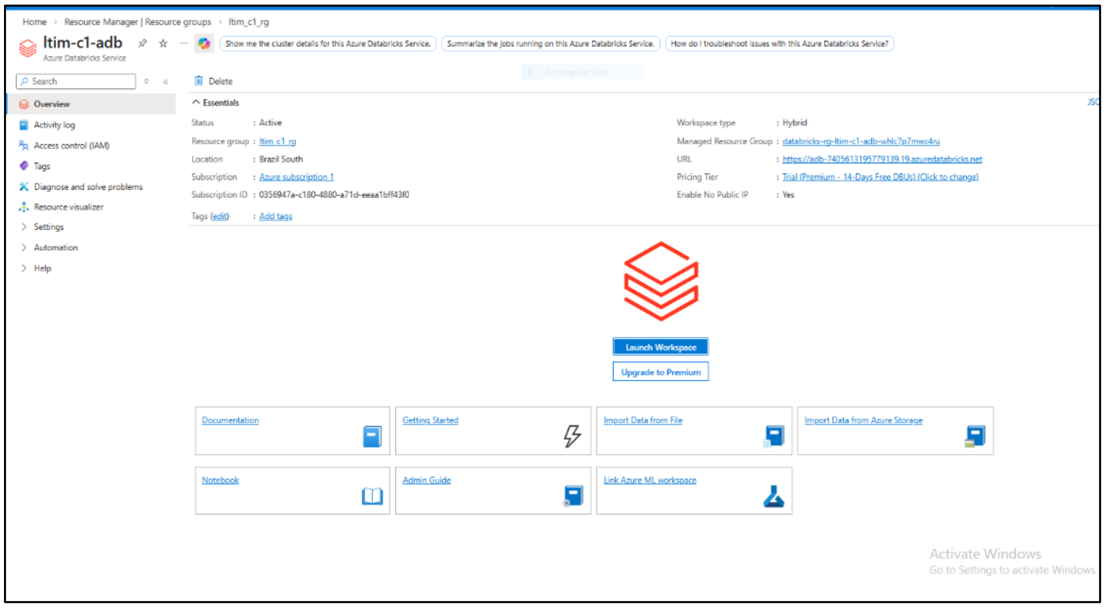
Blob containers

Showing all 4 items

| Name | Last modified | Anonymous access level | Lease state |
|---|---|---|---|
| $logs | 06/02/2026, 16:54:08 | Private | Available |
| bronze | 06/02/2026, 16:55:38 | Private | Available |
| gold | 06/02/2026, 16:56:44 | Private | Available |
| silver | 06/02/2026, 16:56:37 | Private | Available |

Setup of ADLS storage layer

## Logistics Cost & Delivery Analytics Platform

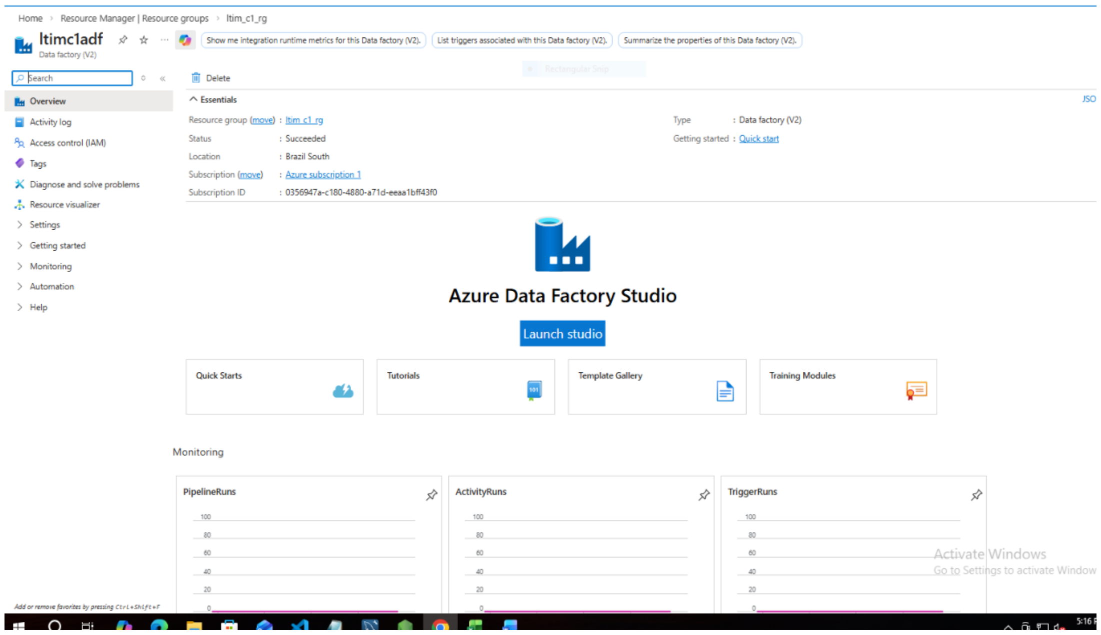

Home › Resource Manager | Resource groups › ltim_c1_rg
Show me the cluster details for this Azure Databricks Service.
Summarize the jobs running on this Azure Databricks Service.
How do I troubleshoot issues with this Azure Databricks Service?
Search
Delete
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Resource visualizer
Settings
Automation
Help
Launch Workspace
Upgrade to Premium
Documentation
Getting Started
Import Data from File
Import Data from Azure Storage
Notebook
Admin Guide
Link Azure ML workspace
Activate Windows
Go to Settings to activate Windows.

ltim-c1-adb
Azure Databricks Service

Essentials

| | |
|---|---|
| Status | : Active |
| Resource group | : ltim_c1_rg |
| Location | : Brazil South |
| Subscription | : Azure subscription 1 |
| Subscription ID | : 0356947a-c180-4880-a71d-eeaa1bff43f0 |
| Tags (edit) | : Add tags |
| Workspace type | : Hybrid |
| Managed Resource Group | : databricks-rg-ltim-c1-adb-whlc7p7mwc4ru |
| URL | : https://adb-7405613195779139.19.azuredatabricks.net |
| Pricing Tier | : Trial (Premium - 14-Days Free DBUs) (Click to change) |
| Enable No Public IP | : Yes |

Azure Databricks Setup

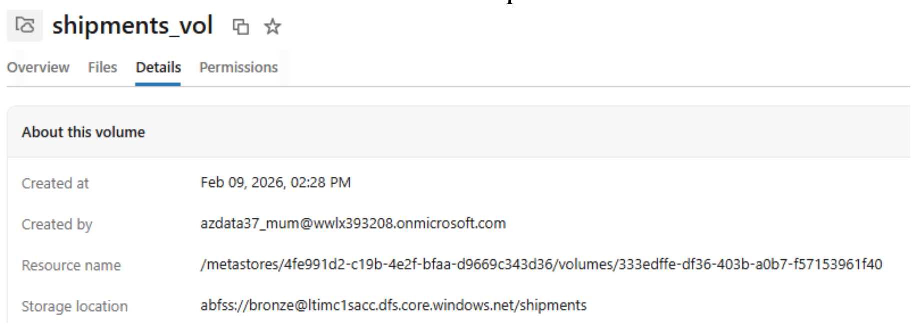

Home › Resource Manager | Resource groups › ltim_c1_rg
Show me integration runtime metrics for this Data factory (V2).
List triggers associated with this Data factory (V2).
Summarize the properties of this Data factory (V2).
Search
Delete
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Resource visualizer
Settings
Getting started
Monitoring
Automation
Help
Azure Data Factory Studio
Launch studio
Quick Starts
Tutorials
Template Gallery
Training Modules
Monitoring
PipelineRuns
ActivityRuns
TriggerRuns
Activate Windows
Go to Settings to activate Windows

ltimc1adf
Data factory (V2)

Essentials

| | |
|---|---|
| Resource group (move) | : ltim_c1_rg |
| Status | : Succeeded |
| Location | : Brazil South |
| Subscription (move) | : Azure subscription 1 |
| Subscription ID | : 0356947a-c180-4880-a71d-eeaa1bff43f0 |
| Type | : Data factory (V2) |
| Getting started | : Quick start |

ADF setup



About this volume

| | |
|---|---|
| Created at | Feb 09, 2026, 02:28 PM |
| Created by | azdata37_mum@wwlx393208.onmicrosoft.com |
| Resource name | /metastores/4fe991d2-c19b-4e2f-bfaa-d9669c343d36/volumes/333edffe-df36-403b-a0b7-f57153961f40 |
| Storage location | abfss://bronze@ltimc1sacc.dfs.core.windows.net/shipments |

## Sprint 1: Bronze ingestion

**Azure Data Factory (ADF) Setup – Ingestion & Orchestration Layer-**

Azure Data Factory (ADF) serves as the central orchestration and ingestion engine for the platform. ADF pipelines are responsible for:

- Parameterized ingestion of daily shipment files
- Incremental loading using a watermark-based strategy
- Validation checks such as file existence and row count
- Pipeline execution logging and run metadata capture

ADF ensures reliable and repeatable data ingestion while supporting reruns and recovery from failures.

The Project Implements an incremental batch ingestion pipeline using Azure Data Factory (ADF) to move shipment data from Azure Data Lake Storage (ADLS  - Raw layer) to bronze layer, while maintaining a watermark-based control mechanism to ensure:

- No duplicate ingestion
- Controlled incremental loads
- Recoverability
- Auditability

Pipeline is Fully Automated, parameterises and monitored, with alerting enabled for successful runs.

1. Source Data Structure

```
raw/
 └── shipments/
      └── shipment_daily_YYYY-MM-DD.csv
```

Each batch represent data for specific ingestion date

2. Watermark Design

**Purpose:** The watermark tracks the last successfully processed date and operational metadata required for incremental ingestion

**Watermark File (JSON)**

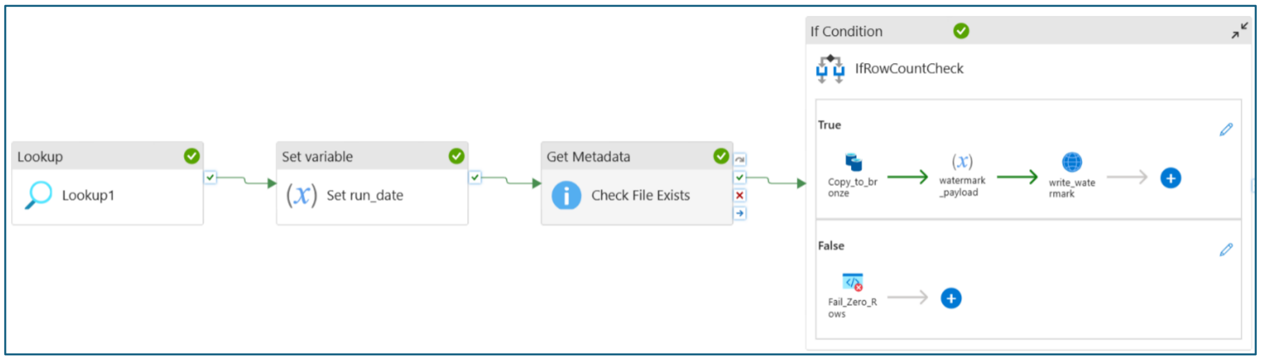
Stored in ADLS

*control/shipments_watermark.json*

**Watermark Structure**

```
{
"entity":"shipments",
"last_successfull_date":"2023-01-22",
"last_update_ts":"2026-0210T06:57:42.2073290Z",
"rows_added":821237,
"files_added":1
}
```

3. Pipeline Architecture Overview



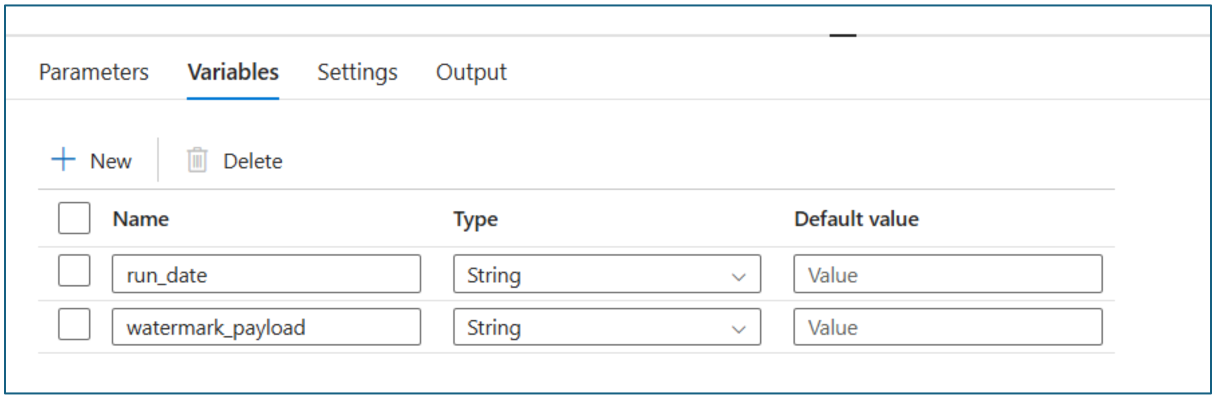
**Pipeline Flow**

1. **Lookup Activity:** Read watermark using lookup activity
2. **Set variable (*run_date*):** Calculate the next run Rate
3. **Get Metadata:** Check if the Expected source file exists
4. **Copy Data:** Copy data from Raw to Bronze layer
5. **If Condition:** Check the Row Counts condition
    1. **Truth:** If Row exists
        a) **Copy Data:** Data Movement from raw to bronze
        b) **Set Variable (*watermark_payload*):** Update the watermark variable

c) **Web Activity:** Update the watermark control file

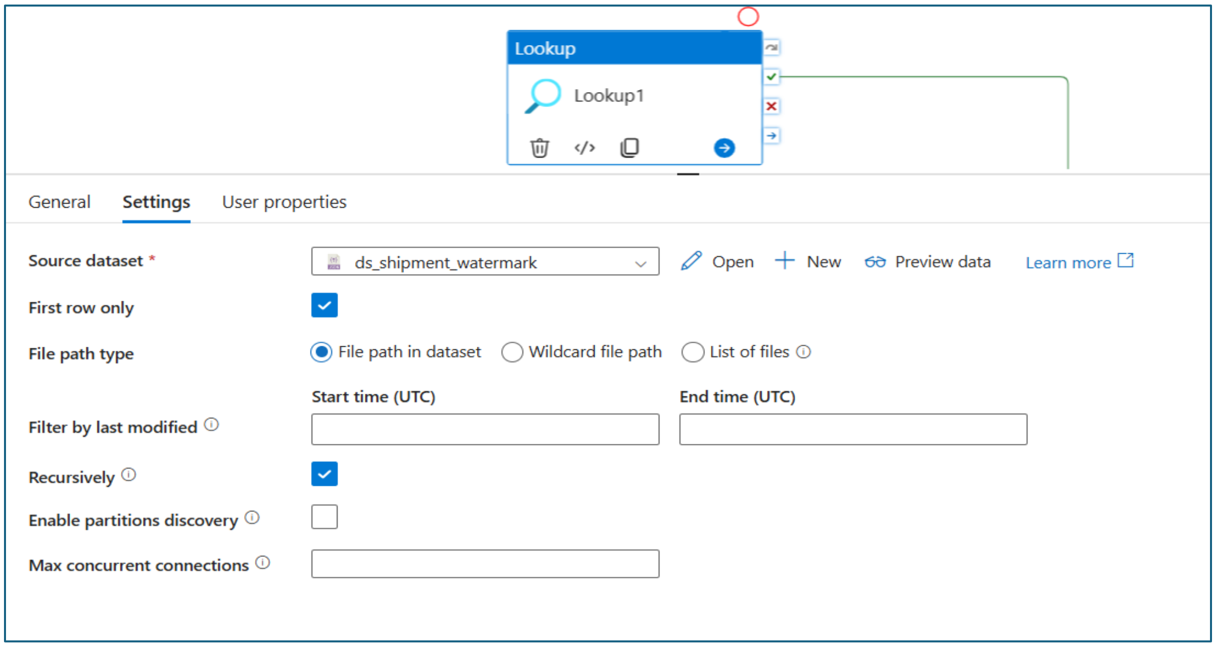
4. Activity Level Explanation

- **Pipeline Variable Initialisation:** Declaring variables in the pipeline canvas to access and modify the variables throughput the pipeline execution.



| Name | Type | Default value |
|---|---|---|
| run_date | String | Value |
| watermark_payload | String | Value |

- **Lookup Activity:** Reads the last successful ingestion date from the watermark file.



Source dataset * ds_shipment_watermark

First row only ✓

File path type (● File path in dataset) ○ Wildcard file path ○ List of files

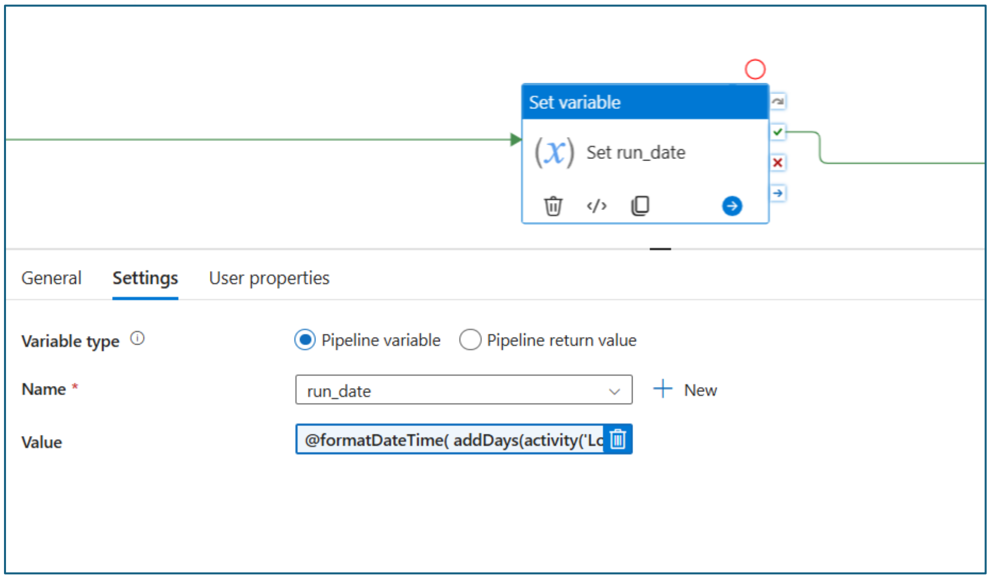
Filter by last modified — Start time (UTC) / End time (UTC)

Recursively ✓

Enable partitions discovery ☐

Max concurrent connections

- **Set Variable (run_date):** Computes the next ingestion date based on the watermark value.

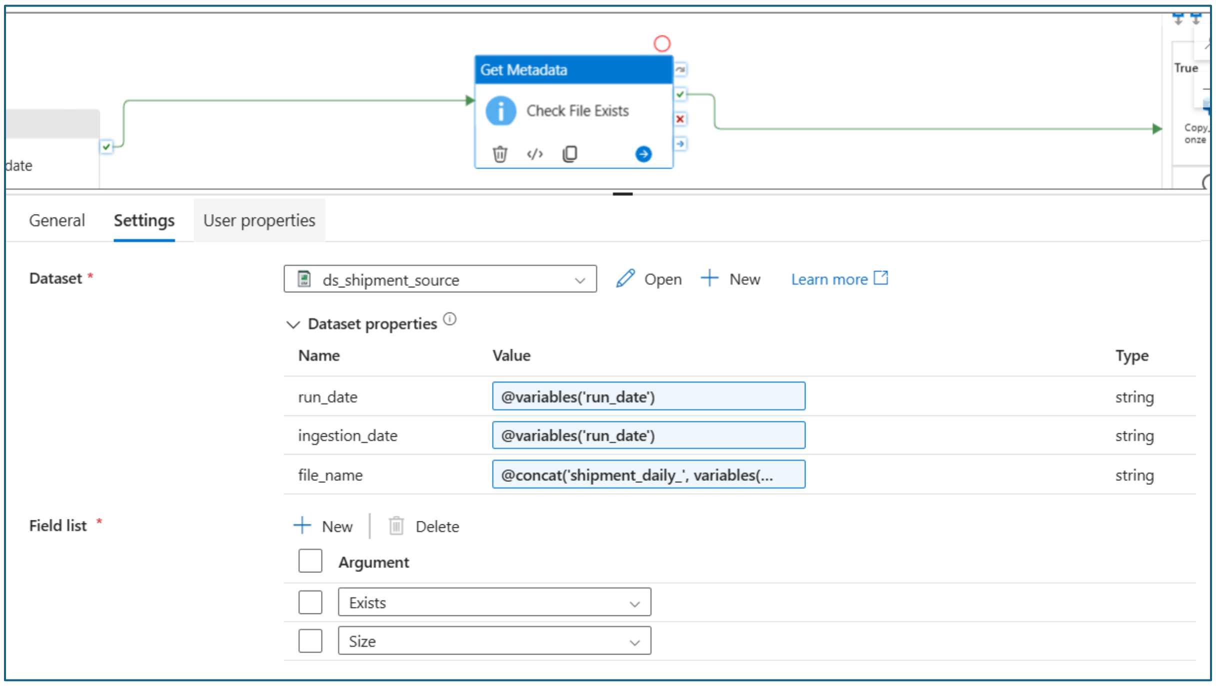


Variable type: [✓ Pipeline variable] Pipeline return value

Name *: run_date

Value: @formatDateTime( addDays(activity('Lo

- **Get Metadata**: Validates whether the expected daily file exists before ingestion.



Dataset *: ds_shipment_source

Dataset properties

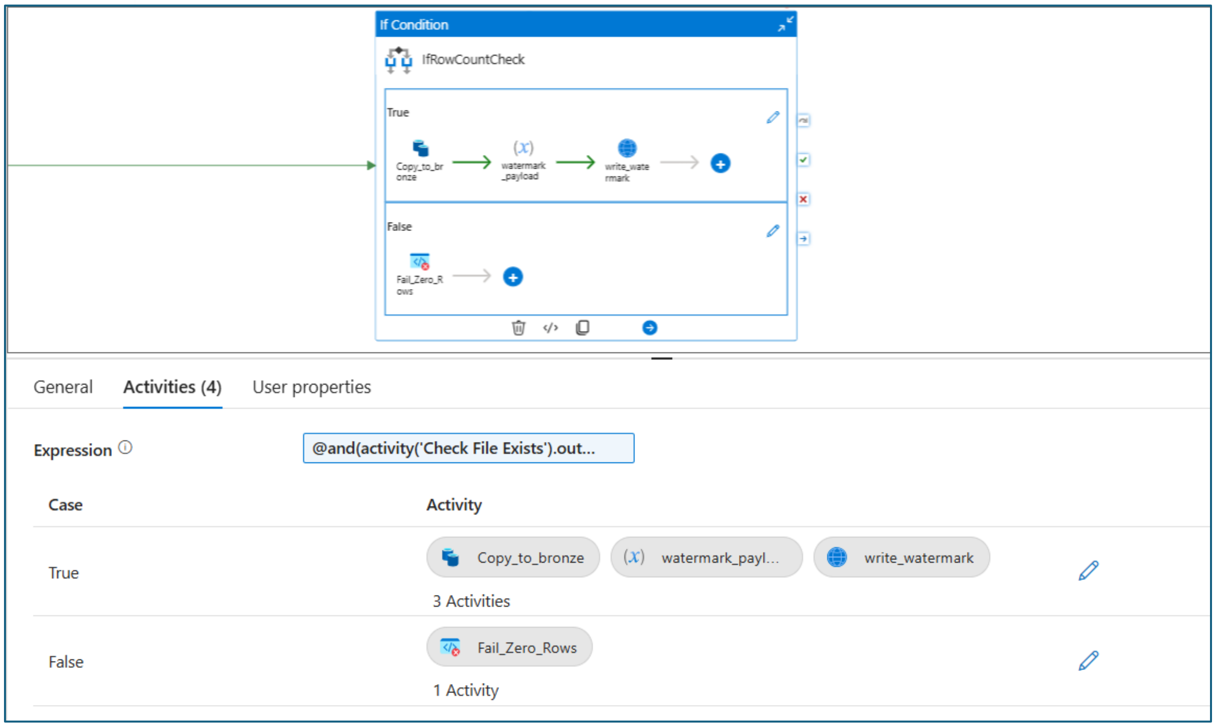
| Name | Value | Type |
|---|---|---|
| run_date | @variables('run_date') | string |
| ingestion_date | @variables('run_date') | string |
| file_name | @concat('shipment_daily_', variables(... | string |

Field list *

New | Delete

| Argument |
|---|
| Exists |
| Size |

- **If Condition**: Ensures data is copied only when the source file exists.

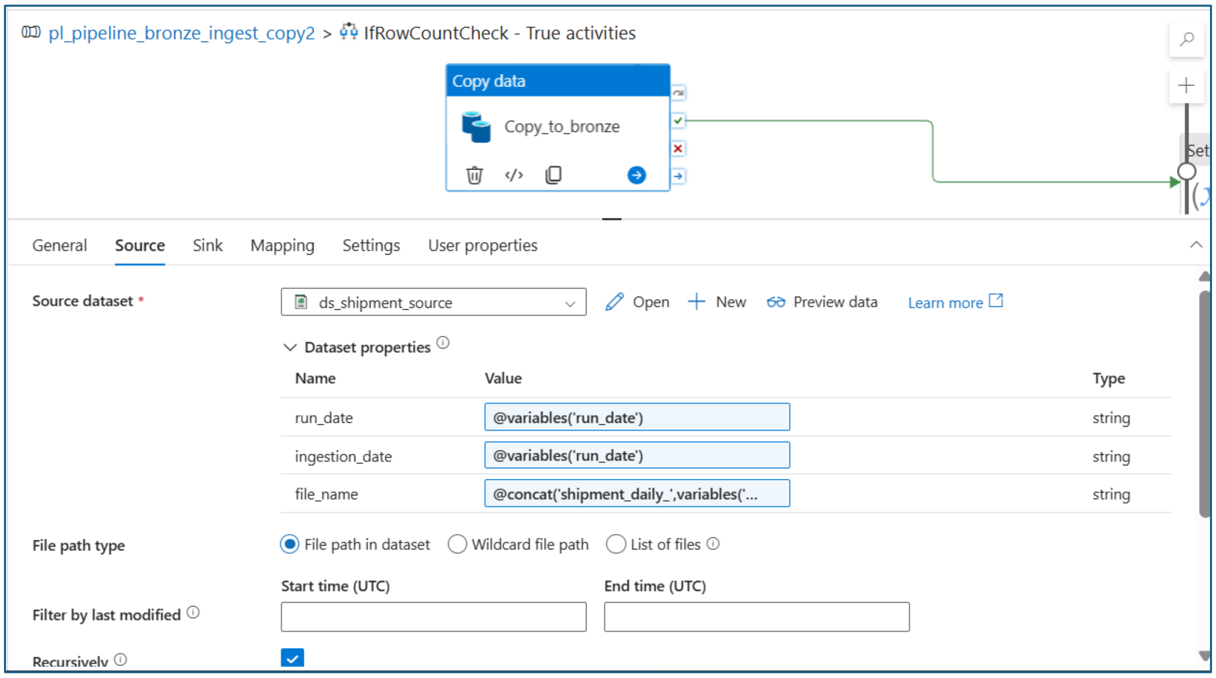


General | Activities (4) | User properties

Expression: @and(activity('Check File Exists').out...

| Case | Activity |
|---|---|
| True | Copy_to_bronze, watermark_payl..., write_watermark — 3 Activities |
| False | Fail_Zero_Rows — 1 Activity |

- **Copy Activity**: Moves shipment data from Raw storage to the Bronze layer.


pl_pipeline_bronze_ingest_copy2 > IfRowCountCheck - True activities
Copy data
Copy_to_bronze
Set

General | Source | Sink | Mapping | Settings | User properties

Source dataset *: ds_shipment_source — Open + New Preview data Learn more

Dataset properties

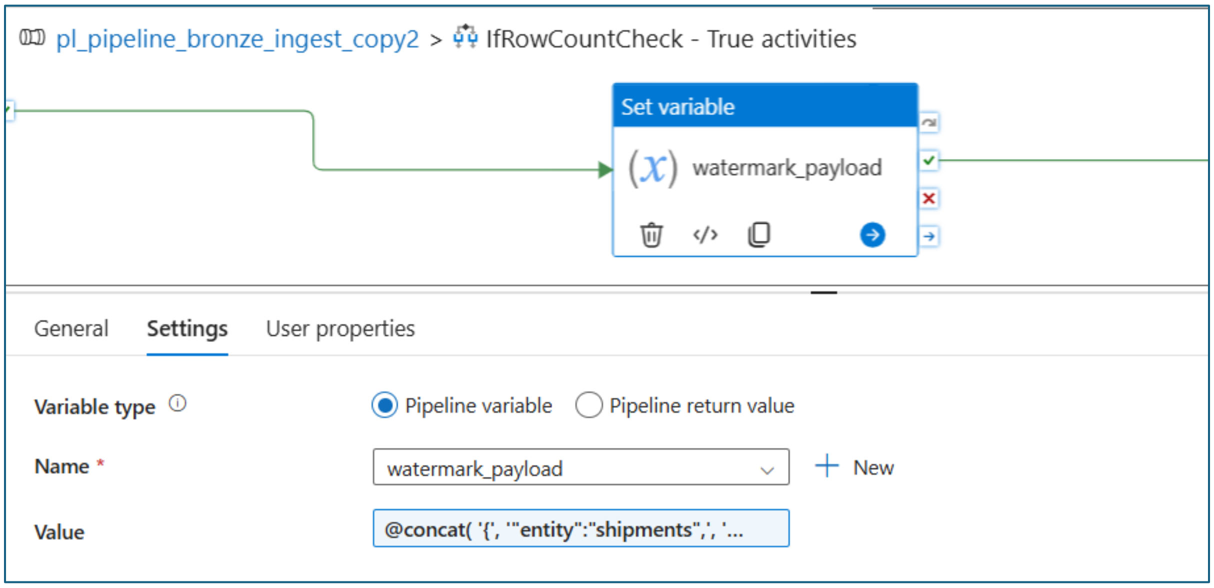
| Name | Value | Type |
|---|---|---|
| run_date | @variables('run_date') | string |
| ingestion_date | @variables('run_date') | string |
| file_name | @concat('shipment_daily_',variables('... | string |

File path type: [✓ File path in dataset] Wildcard file path List of files

Filter by last modified: Start time (UTC) End time (UTC)

Recursively ✓

- **Set Variable (watermark_payload):** Constructs a JSON payload containing updated watermark values.



pl_pipeline_bronze_ingest_copy2 > IfRowCountCheck - True activities

Set variable: watermark_payload

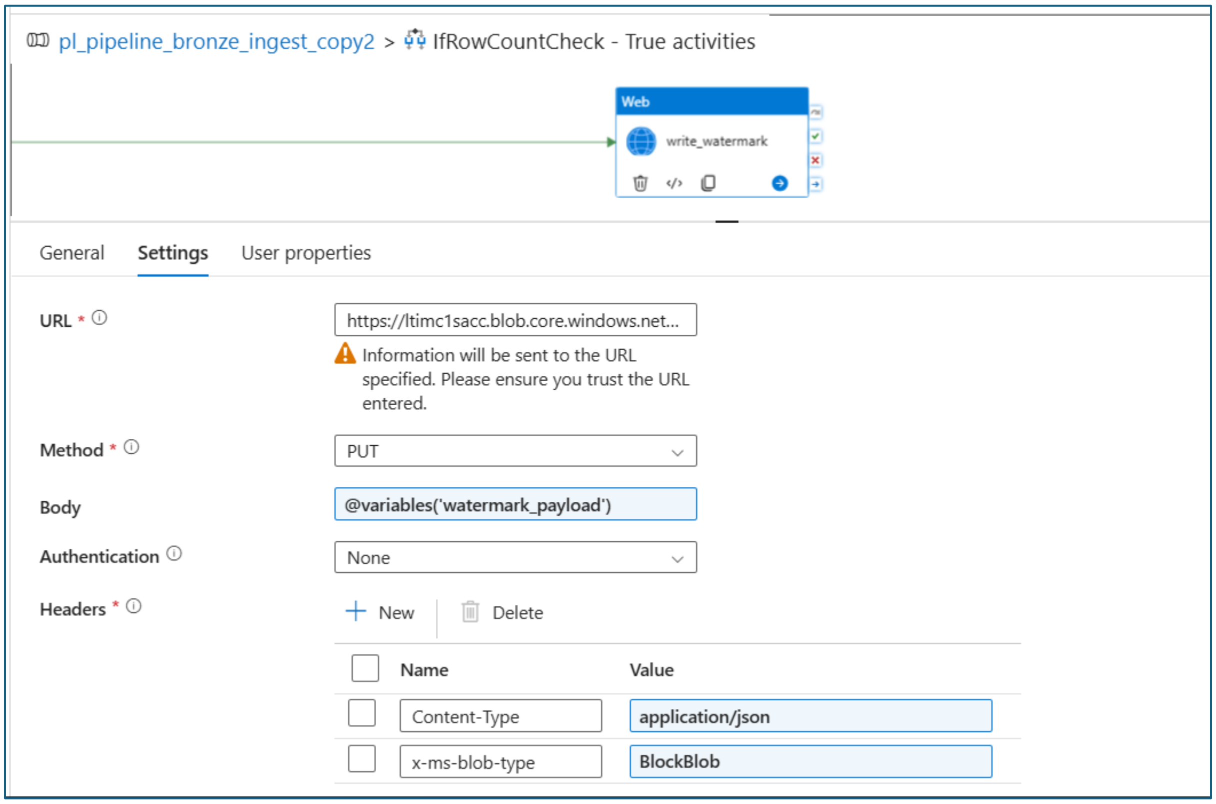
General | **Settings** | User properties

Variable type: [✓ Pipeline variable] Pipeline return value

Name *: watermark_payload

Value: @concat( '{', '"entity":"shipments",', '...

- **Web Activity**: Overwrites the watermark JSON file using an HTTP PUT operation.



pl_pipeline_bronze_ingest_copy2 > IfRowCountCheck - True activities

Web: write_watermark

General | **Settings** | User properties

URL *: https://ltimc1sacc.blob.core.windows.net...

Information will be sent to the URL specified. Please ensure you trust the URL entered.

Method *: PUT

Body: @variables('watermark_payload')

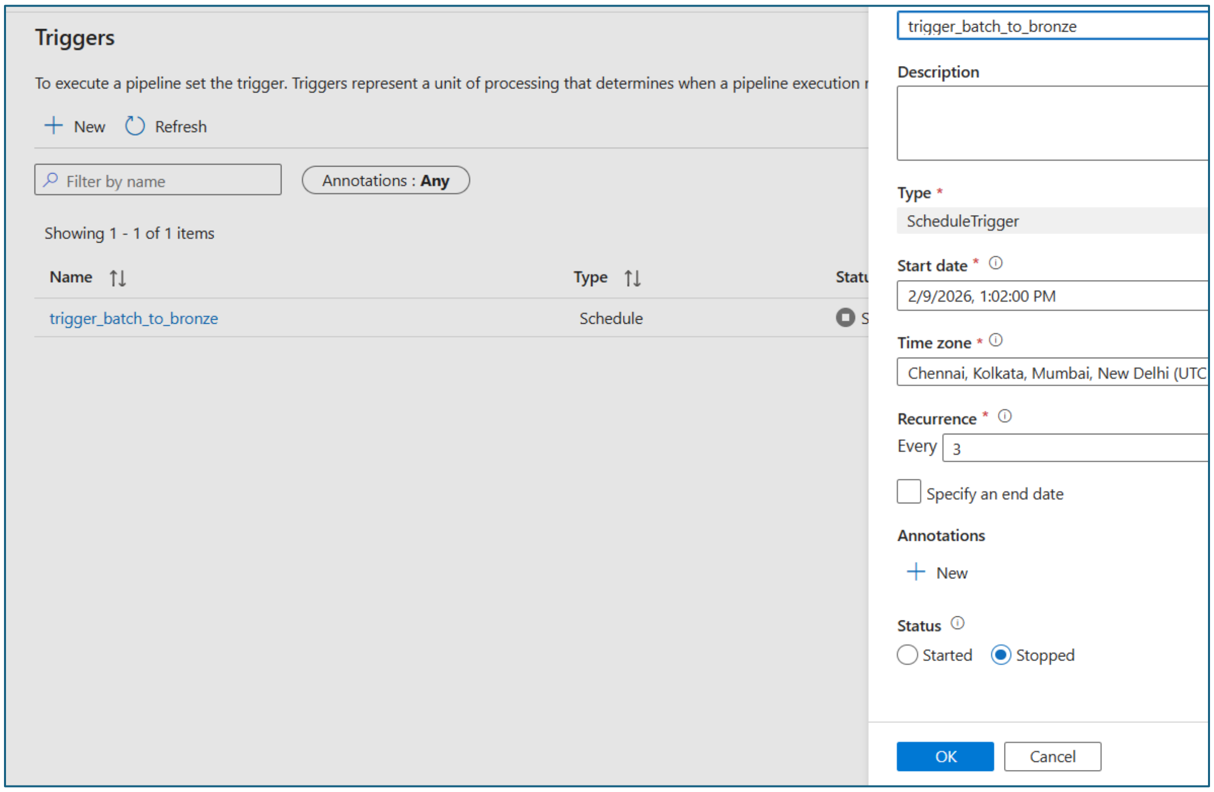
Authentication: None

Headers *: + New | Delete

| Name | Value |
|---|---|
| Content-Type | application/json |
| x-ms-blob-type | BlockBlob |

## 5. Trigger Automation

The pipeline is automated using an Azure Data Factory Schedule Trigger configured to run once per 3 minutes (Ideally once per day). No pipeline parameters are required, as the watermark controls incremental execution.


Triggers

To execute a pipeline set the trigger. Triggers represent a unit of processing that determines when a pipeline execution

Showing 1 - 1 of 1 items

| Name | Type | Statu |
|---|---|---|
| trigger_batch_to_bronze | Schedule | S |

trigger_batch_to_bronze

Description

Type *: ScheduleTrigger

Start date *: 2/9/2026, 1:02:00 PM

Time zone *: Chennai, Kolkata, Mumbai, New Delhi (UTC

Recurrence *: Every 3

☐ Specify an end date

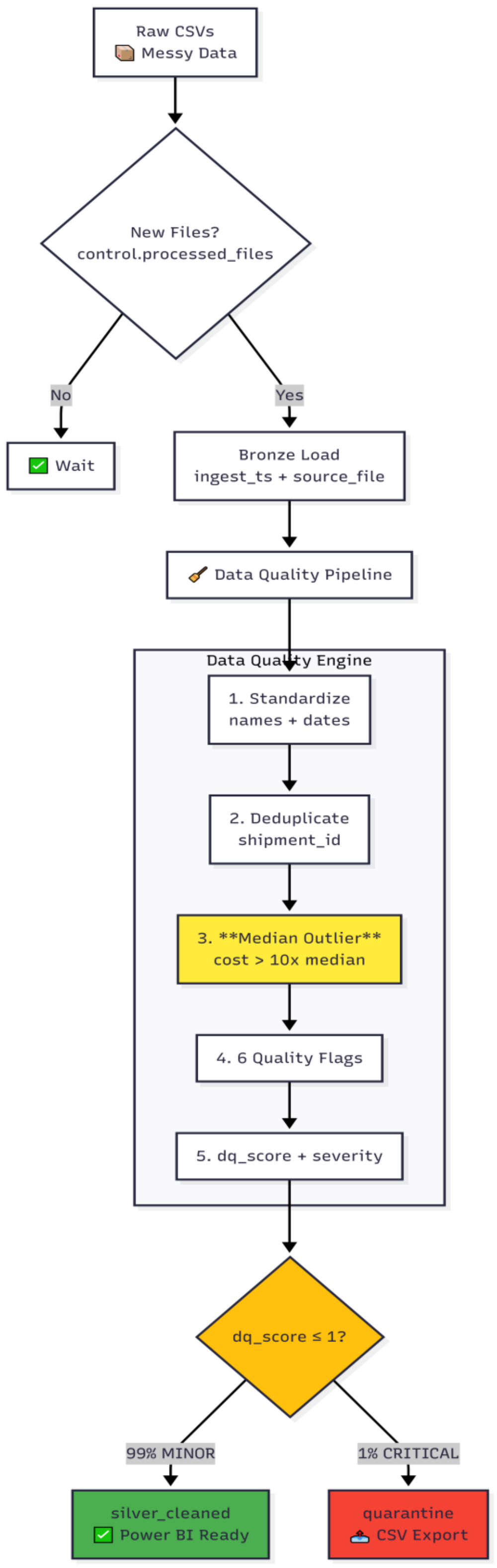
Annotations

Status: ( ) Started [✓ Stopped]

Overwrites the watermark JSON file using an HTTP PUT operation.

## Sprint 2: Silver Transformations & Data Quality Architecture

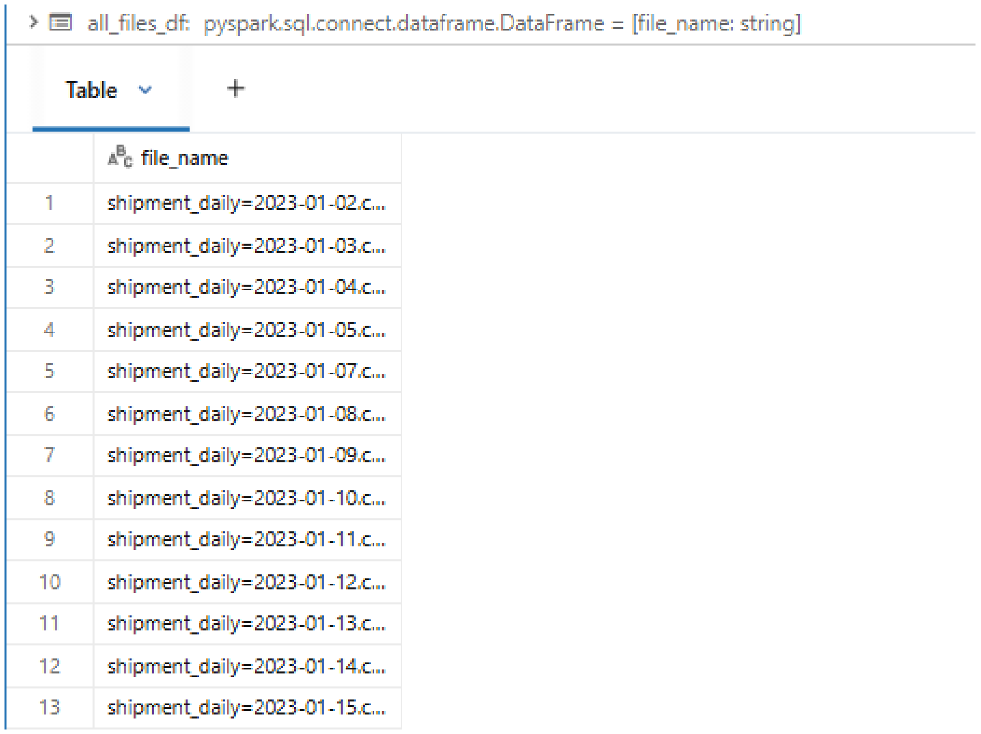
### Architecture Overview



The Silver layer is responsible for converting raw ingested data into a clean, consistent, and analysis-ready dataset. Unlike the Bronze layer, which preserves data exactly as received, the Silver layer actively enforces structural integrity, data correctness, and semantic consistency. All cleaning operations are implemented using Apache Spark in Azure Databricks, enabling distributed processing, scalability, and repeatability. The goal of this layer is to ensure that the dataset is technically reliable and logically sound before any analytical consumption.

## Incremental Control Framework for File Processing

To ensure efficient and idempotent data processing, a control framework is implemented to manage incremental file ingestion in the Silver layer. This framework prevents reprocessing of already ingested files and ensures that only new data is processed during each execution.

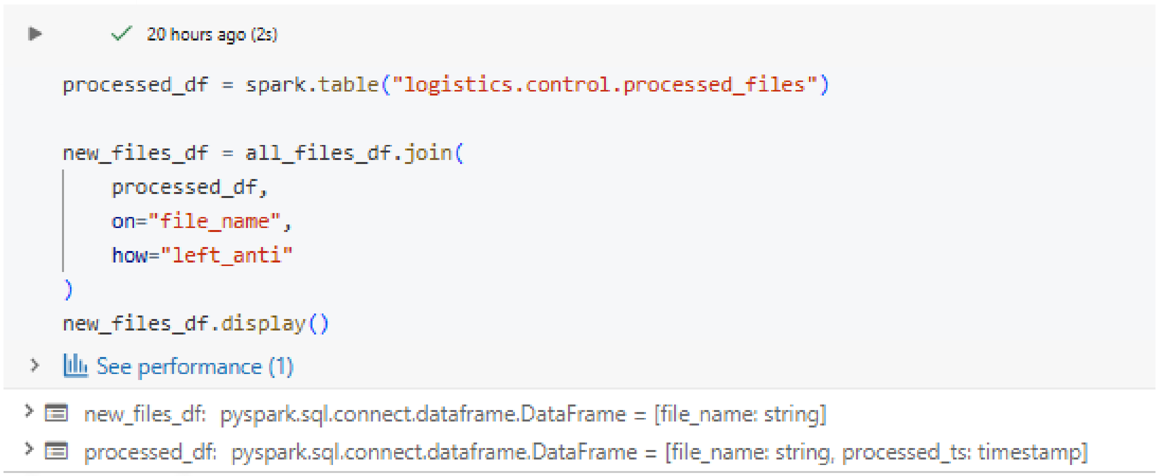


```
all_files_df:  pyspark.sql.connect.dataframe.DataFrame = [file_name: string]
```

| | file_name |
|---|---|
| 1 | shipment_daily=2023-01-02.c... |
| 2 | shipment_daily=2023-01-03.c... |
| 3 | shipment_daily=2023-01-04.c... |
| 4 | shipment_daily=2023-01-05.c... |
| 5 | shipment_daily=2023-01-07.c... |
| 6 | shipment_daily=2023-01-08.c... |
| 7 | shipment_daily=2023-01-09.c... |
| 8 | shipment_daily=2023-01-10.c... |
| 9 | shipment_daily=2023-01-11.c... |
| 10 | shipment_daily=2023-01-12.c... |
| 11 | shipment_daily=2023-01-13.c... |
| 12 | shipment_daily=2023-01-14.c... |
| 13 | shipment_daily=2023-01-15.c... |

A dataframe that shows all values present in external volume (bronze of ADLS)

A **processed files register** is maintained to store metadata of all successfully processed files, including file name and processing timestamp. This register serves as a historical reference to track ingestion progress and supports auditability. At the start of each run, the pipeline scans the Bronze layer and captures the list of available files as **new_files**. These files are then compared against the **processed_files** register using a left anti-join strategy to identify files that have not yet been processed. Only the resulting unmatched files are selected for cleaning and transformation.



```
20 hours ago (2s)

processed_df = spark.table("logistics.control.processed_files")

new_files_df = all_files_df.join(
    processed_df,
    on="file_name",
    how="left_anti"
)
new_files_df.display()
```

See performance (1)

```
new_files_df:  pyspark.sql.connect.dataframe.DataFrame = [file_name: string]
processed_df:  pyspark.sql.connect.dataframe.DataFrame = [file_name: string, processed_ts: timestamp]
```

A view of new_files_df and processed_df(left anti_join procedure)

In addition, **control files** are used to capture run-level execution metadata such as run date, number of files processed, and execution status. This enables operational monitoring and allows safe pipeline re-runs without duplicate processing. This control-driven design ensures reliable incremental processing, supports late-arriving data, and aligns with production-grade data engineering best practices.

## Cleaning and Standardization

### 1. **Column Name Standardization**

Source files received from multiple regions and systems often contain inconsistent column naming conventions, including mixed casing, spaces, and special characters. To resolve this, a standardized naming convention is applied where all column names are converted to lowercase and spaces or special characters are replaced with underscores. This ensures SQL compatibility, consistency across transformations, and improved maintainability.

### 2. **Whitespace Trimming and Text Normalization**

String-based fields frequently contain leading or trailing whitespaces due to manual data entry or system exports. All string columns are trimmed, and categorical fields such as carrier_id and delivery_status are normalized to uppercase. This prevents mismatches during joins and aggregations and ensures consistent analytical results.

### 3. **Explicit Data Type Enforcement**

Raw CSV files store all values as strings, which can cause calculation errors if left untyped. Explicit casting is applied to critical columns such as shipment_cost to double, shipment_date and delivery_date to date, delivery_days to integer, and is_fragile to boolean. This guarantees numerical accuracy and reliable time-based analysis.

### 4. **Date Consistency and Logical Validation**

Date fields are validated to ensure proper parsing and consistent formatting. This enables accurate delivery duration calculations and ensures that shipment timelines can be reliably analyzed.

5. **Handling of Duplicate Records**

Duplicate shipment records occur due to late-arriving data or upstream retries. A controlled deduplication strategy is applied using shipment_id as the business key. Records are ordered by ingestion timestamp, and only the most recent record is retained, ensuring that the latest shipment state is preserved.

6. **Standardization of Categorical Values**

Categorical fields are standardized to a controlled set of expected values. This eliminates inconsistencies caused by casing or formatting variations and simplifies downstream filtering and aggregation logic.

7. **Null Handling and Structural Completeness**

Mandatory fields are checked for completeness, empty strings are converted to null values, and optional fields are standardized. This ensures clarity between missing, empty, and invalid values.

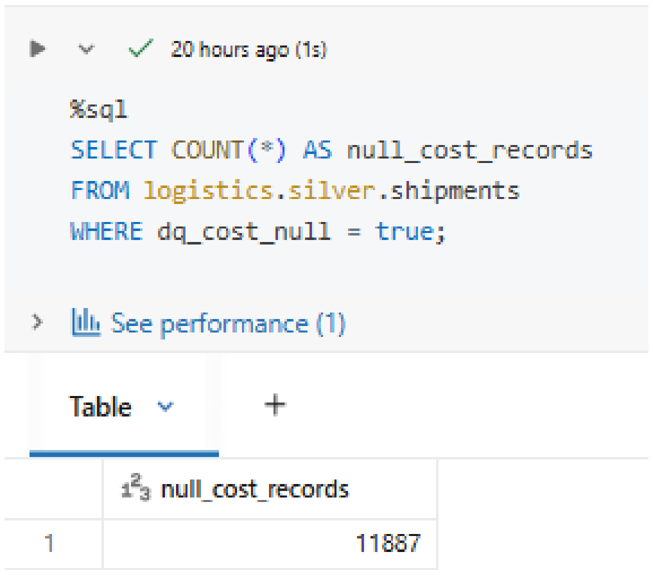
8. **Outlier Preparation and Cost Normalization**

Numeric fields such as shipment_cost are cleaned by removing formatting artifacts and ensuring positive numeric values. This prepares the data for reliable statistical evaluation in later stages. After these steps, the dataset is considered technically clean, with a standardized schema, consistent data types, duplicate-free records, and structurally reliable values.

A rule-based data quality framework is implemented in the silver layer to ensure the reliability and consistency of the dataset. Each record is validated against a defined set of data quality rules that address completeness, validity, logical consistency, and statistical anomalies. For every rule applied, a binary data quality flag is generated at the record level, where a value of one indicates a rule violation and a value of zero indicates compliance.

The data quality checks include identification of missing shipment costs, detection of zero or negative cost values, validation of delivery status against allowed domain values, logical validation of shipment and delivery dates, and identification of abnormal shipment costs using a median-based threshold. These checks ensure that both technical and business-related data issues are detected early in the pipeline.

Once all data quality flags are evaluated, a composite data quality score is calculated for each record by summing the individual flag values. This score is then used to classify records into severity levels based on the impact of the identified issues.



20 hours ago (1s)

```
%sql
SELECT COUNT(*) AS null_cost_records
FROM logistics.silver.shipments
WHERE dq_cost_null = true;
```

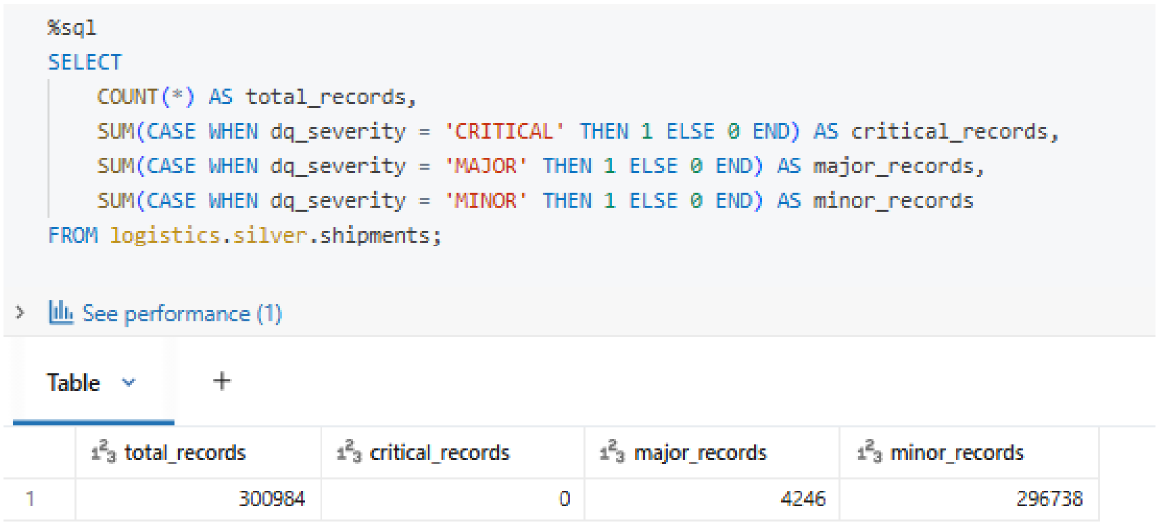
See performance (1)

Table

| | null_cost_records |
|---|---|
| 1 | 11887 |

Count of all rows having shipment_cost=NULL

Records with a data quality score of zero are classified as Minor, indicating that no data quality rules are violated and the record is considered clean and reliable for downstream processing. Records with a data quality score of one are classified as Major, indicating the presence of a single data quality issue that could impact analytical accuracy. Records with a data quality score greater than one are classified as Critical, indicating multiple rule violations and a high risk of data inconsistency or misinterpretation.



```
%sql
SELECT
    COUNT(*) AS total_records,
    SUM(CASE WHEN dq_severity = 'CRITICAL' THEN 1 ELSE 0 END) AS critical_records,
    SUM(CASE WHEN dq_severity = 'MAJOR' THEN 1 ELSE 0 END) AS major_records,
    SUM(CASE WHEN dq_severity = 'MINOR' THEN 1 ELSE 0 END) AS minor_records
FROM logistics.silver.shipments;
```

See performance (1)

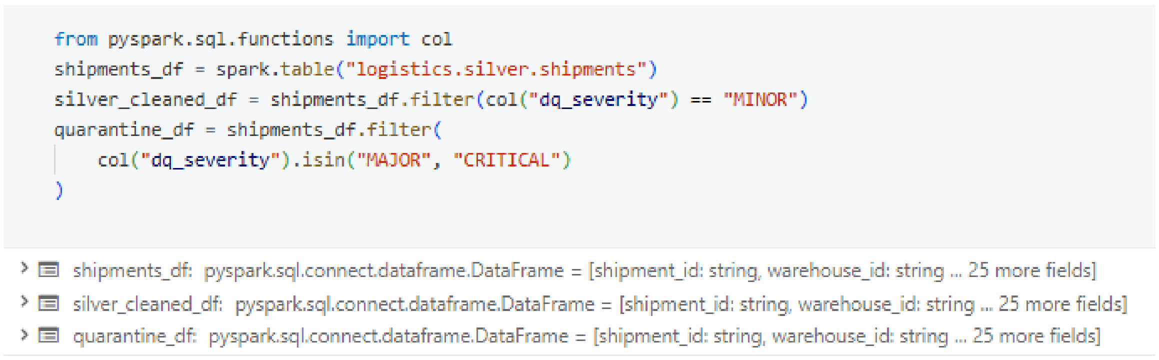
Table

| | total_records | critical_records | major_records | minor_records |
|---|---|---|---|---|
| 1 | 300984 | 0 | 4246 | 296738 |

A view of the data quality table showing the total rows in each category as records

This severity-based classification enables consistent handling of data quality issues while maintaining transparency at the record level. It allows clean data to proceed through the pipeline while ensuring that records with significant issues are clearly identified for further review and remediation. After completing data cleaning and data quality validation, the processed dataset is written back to the Silver zone in Azure Data Lake Storage using the outcomes of the DQ framework. Each record carries data quality flags, a composite DQ score, and a severity classification, which together govern how the data is persisted.

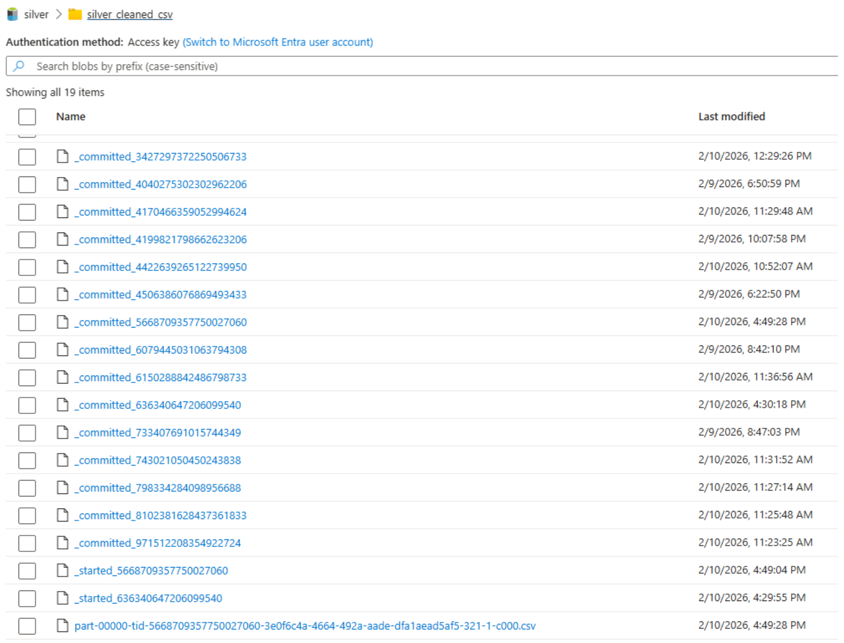
Records classified as Minor severity, indicating no data quality violations, are written to the primary Silver cleaned dataset. This dataset represents the trusted and standardized version of the data and serves as the foundation for downstream transformations and analytical use cases. Records classified as Major or Critical are excluded from the cleaned output and written to a separate location to ensure that unreliable data does not impact reporting or modeling.



```
from pyspark.sql.functions import col
shipments_df = spark.table("logistics.silver.shipments")
silver_cleaned_df = shipments_df.filter(col("dq_severity") == "MINOR")
quarantine_df = shipments_df.filter(
    col("dq_severity").isin("MAJOR", "CRITICAL")
)
```

```
> shipments_df:  pyspark.sql.connect.dataframe.DataFrame = [shipment_id: string, warehouse_id: string ... 25 more fields]
> silver_cleaned_df:  pyspark.sql.connect.dataframe.DataFrame = [shipment_id: string, warehouse_id: string ... 25 more fields]
> quarantine_df:  pyspark.sql.connect.dataframe.DataFrame = [shipment_id: string, warehouse_id: string ... 25 more fields]
```

Data quality rules set and divided into dataframes



silver > silver_cleaned_csv

**Authentication method:** Access key (Switch to Microsoft Entra user account)

Search blobs by prefix (case-sensitive)

Showing all 19 items

| Name | Last modified |
|---|---|
| _committed_3427297372250506733 | 2/10/2026, 12:29:26 PM |
| _committed_4040275302302962206 | 2/9/2026, 6:50:59 PM |
| _committed_4170466359052994624 | 2/10/2026, 11:29:48 AM |
| _committed_4199821798662623206 | 2/9/2026, 10:07:58 PM |
| _committed_4422639265122739950 | 2/10/2026, 10:52:07 AM |
| _committed_4506386076869493433 | 2/9/2026, 6:22:50 PM |
| _committed_5668709357750027060 | 2/10/2026, 4:49:28 PM |
| _committed_6079445031063794308 | 2/9/2026, 8:42:10 PM |
| _committed_6150288842486798733 | 2/10/2026, 11:36:56 AM |
| _committed_636340647206099540 | 2/10/2026, 4:30:18 PM |
| _committed_733407691015744349 | 2/9/2026, 8:47:03 PM |
| _committed_743021050450243838 | 2/10/2026, 11:31:52 AM |
| _committed_798334284098956688 | 2/10/2026, 11:27:14 AM |
| _committed_8102381628437361833 | 2/10/2026, 11:25:48 AM |
| _committed_971512208354922724 | 2/10/2026, 11:23:25 AM |
| _started_5668709357750027060 | 2/10/2026, 4:49:04 PM |
| _started_636340647206099540 | 2/10/2026, 4:29:55 PM |
| part-00000-tid-5668709357750027060-3e0f6c4a-4664-492a-aade-dfa1aead5af5-321-1-c000.csv | 2/10/2026, 4:49:28 PM |

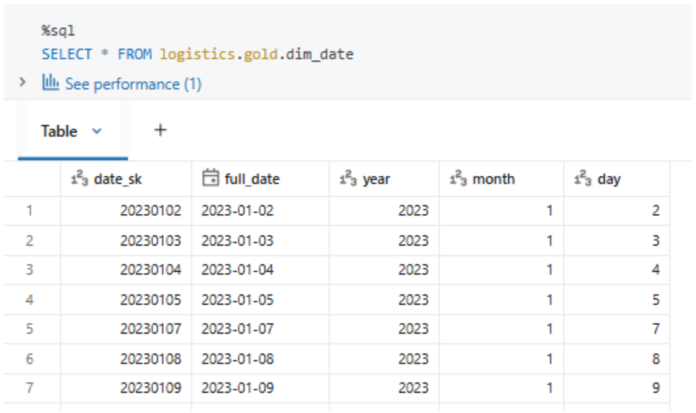
All Silver outputs are stored in Delta Lake format, providing schema enforcement, transactional consistency, and support for incremental updates. By persisting data quality attributes alongside cleaned records, the silver layer maintains transparency into data health while ensuring that only validated data progresses further in the pipeline.

## Sprint 3 – Gold Layer Modelling

### Fact, Dimensions, and Aggregations

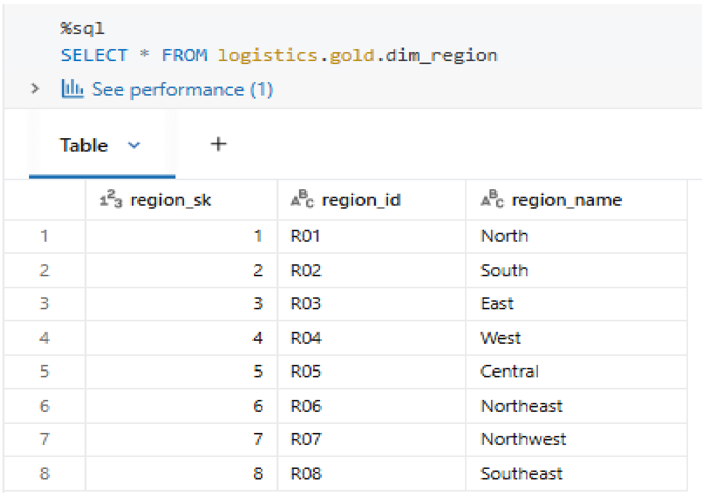
The Gold layer represents the final, analytics-ready state of the data. At this stage, all records have already passed cleaning and data quality validation in the Silver layer. The responsibility of the Gold layer is to reshape this validated data into a dimensional model that supports high-performance analytical queries, consistent aggregations, and downstream BI consumption. All transformations in this layer are executed using Apache Spark in Databricks, and the outputs are written to the Gold zone in Azure Data Lake.

Dimension tables are constructed to store descriptive attributes that provide context to shipment-level facts. These tables are designed to be relatively small, slowly changing, and highly reusable across analytical queries. For each dimension, a distinct set of records is extracted either from the Silver cleaned dataset or from reference datasets. Duplicate values are eliminated using distinct selection logic to ensure uniqueness at the dimension level. A surrogate key is generated for each dimension record using deterministic logic such as row numbering. Surrogate keys are introduced to abstract the analytical model from source-system identifiers and to guarantee stable joins even when natural keys are missing, inconsistent, or subject to change.



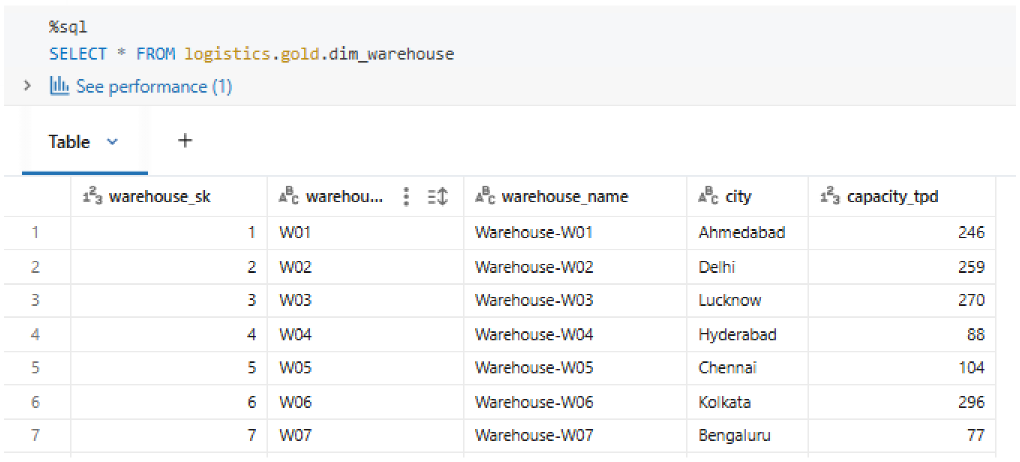
```
%sql
SELECT * FROM logistics.gold.dim_date
```

| | date_sk | full_date | year | month | day |
|---|---|---|---|---|---|
| 1 | 20230102 | 2023-01-02 | 2023 | 1 | 2 |
| 2 | 20230103 | 2023-01-03 | 2023 | 1 | 3 |
| 3 | 20230104 | 2023-01-04 | 2023 | 1 | 4 |
| 4 | 20230105 | 2023-01-05 | 2023 | 1 | 5 |
| 5 | 20230107 | 2023-01-07 | 2023 | 1 | 7 |
| 6 | 20230108 | 2023-01-08 | 2023 | 1 | 8 |
| 7 | 20230109 | 2023-01-09 | 2023 | 1 | 9 |



```
%sql
SELECT * FROM logistics.gold.dim_region
```

| | region_sk | region_id | region_name |
|---|---|---|---|
| 1 | 1 | R01 | North |
| 2 | 2 | R02 | South |
| 3 | 3 | R03 | East |
| 4 | 4 | R04 | West |
| 5 | 5 | R05 | Central |
| 6 | 6 | R06 | Northeast |
| 7 | 7 | R07 | Northwest |
| 8 | 8 | R08 | Southeast |

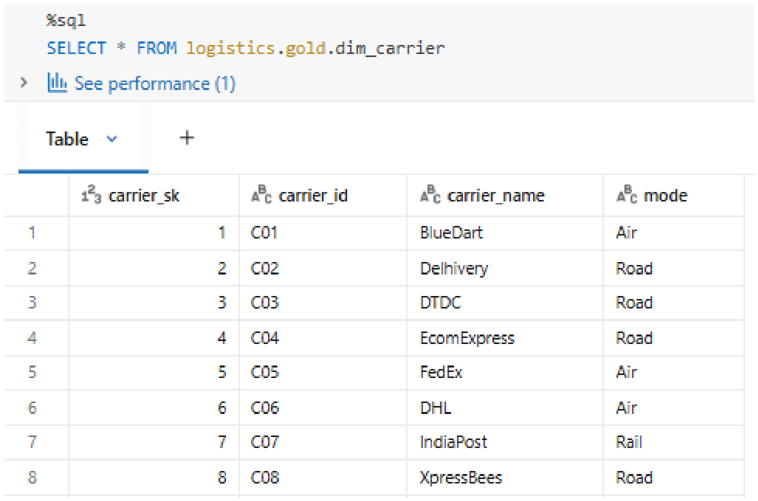


```
%sql
SELECT * FROM logistics.gold.dim_warehouse
```

See performance (1)

Table

| | warehouse_sk | warehou... | warehouse_name | city | capacity_tpd |
|---|---|---|---|---|---|
| 1 | 1 | W01 | Warehouse-W01 | Ahmedabad | 246 |
| 2 | 2 | W02 | Warehouse-W02 | Delhi | 259 |
| 3 | 3 | W03 | Warehouse-W03 | Lucknow | 270 |
| 4 | 4 | W04 | Warehouse-W04 | Hyderabad | 88 |
| 5 | 5 | W05 | Warehouse-W05 | Chennai | 104 |
| 6 | 6 | W06 | Warehouse-W06 | Kolkata | 296 |
| 7 | 7 | W07 | Warehouse-W07 | Bengaluru | 77 |



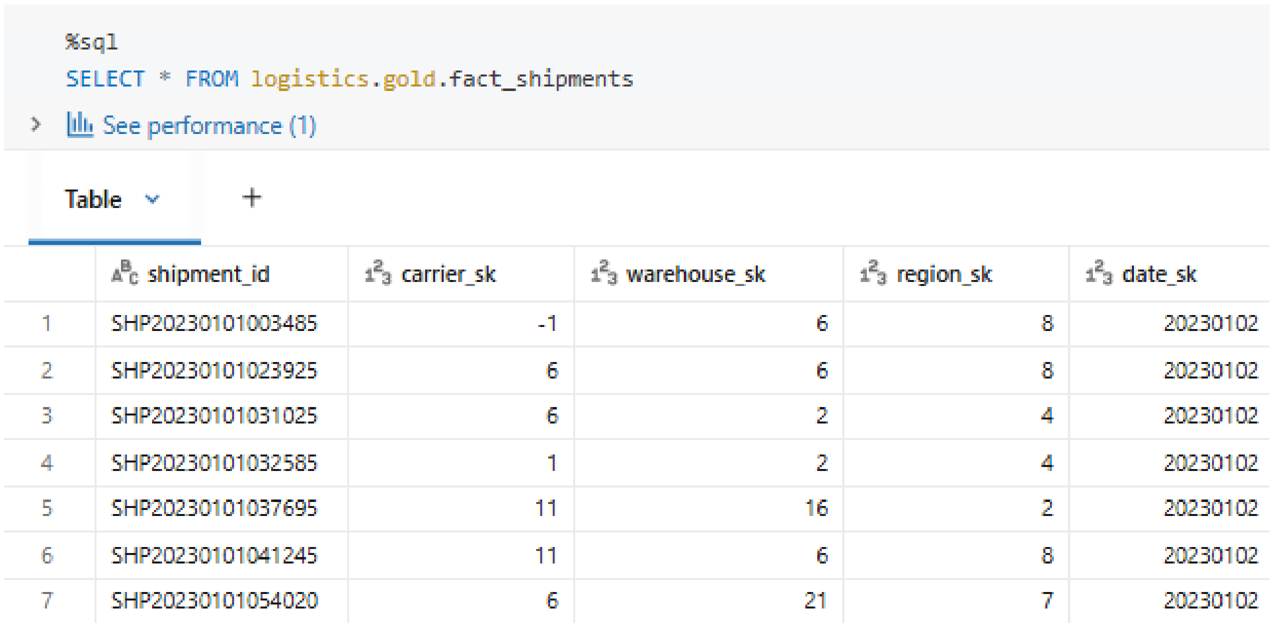
```
%sql
SELECT * FROM logistics.gold.dim_carrier
```

See performance (1)

Table

| | carrier_sk | carrier_id | carrier_name | mode |
|---|---|---|---|---|
| 1 | 1 | C01 | BlueDart | Air |
| 2 | 2 | C02 | Delhivery | Road |
| 3 | 3 | C03 | DTDC | Road |
| 4 | 4 | C04 | EcomExpress | Road |
| 5 | 5 | C05 | FedEx | Air |
| 6 | 6 | C06 | DHL | Air |
| 7 | 7 | C07 | IndiaPost | Rail |
| 8 | 8 | C08 | XpressBees | Road |

A view of all the dimension tables created



```
%sql
SELECT * FROM logistics.gold.fact_shipments
```

See performance (1)

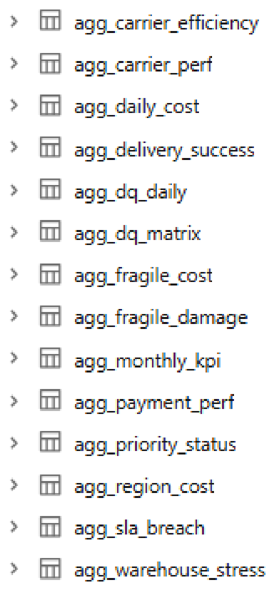
Table

| | shipment_id | carrier_sk | warehouse_sk | region_sk | date_sk |
|---|---|---|---|---|---|
| 1 | SHP20230101003485 | -1 | 6 | 8 | 20230102 |
| 2 | SHP20230101023925 | 6 | 6 | 8 | 20230102 |
| 3 | SHP20230101031025 | 6 | 2 | 4 | 20230102 |
| 4 | SHP20230101032585 | 1 | 2 | 4 | 20230102 |
| 5 | SHP20230101037695 | 11 | 16 | 2 | 20230102 |
| 6 | SHP20230101041245 | 11 | 6 | 8 | 20230102 |
| 7 | SHP20230101054020 | 6 | 21 | 7 | 20230102 |

**Fact_Shipments Table**

| Aggregation Table | What It Shows (Simple) | Why It Is Important | KPI / Usage |
|---|---|---|---|
| agg_daily_cost | Shows how shipment cost changes day by day and highlights expensive days. | Identifies cost spikes where logistics expenses increase abnormally and helps control daily spending. | Cost per Day |
| agg_carrier_perf | Compares carriers based on on-time delivery and speed. | Helps choose the most reliable and cost-effective carrier for operations. | OTIF % |
| agg_region_cost | Shows average shipment cost across different regions. | Highlights regional cost imbalances and pricing inefficiencies. | Regional Cost Variance |
| agg_warehouse_stress | Indicates warehouse load compared to its capacity. | Prevents warehouse overload that leads to delays and overtime costs. | Capacity Utilization |
| agg_delivery_success | Measures percentage of shipments delivered within SLA. | Directly impacts customer satisfaction and service quality. | On-Time Delivery Rate |
| agg_sla_breach | Identifies carriers with high shipment volume and frequent SLA breaches. | Flags high-risk carriers that can cause major operational issues. | Carrier Risk Score |
| agg_payment_perf | Compares delivery performance across payment types. | Shows how payment methods affect delivery speed and efficiency. | Payment Cycle Time |
| agg_priority_status | Evaluates cost and delays for priority shipments. | Ensures premium services remain profitable. | Priority Service Margin |
| agg_fragile_cost | Compares cost of fragile vs non-fragile shipments. | Identifies hidden cost premiums for fragile handling. | Fragile Handling Premium |

| Aggregation Table | What It Shows (Simple) | Why It Is Important | KPI / Usage |
|---|---|---|---|
| agg_carrier_efficiency | Compares carriers on cost versus speed. | Helps identify carriers that provide maximum value. | Carrier Efficiency Index |
| agg_monthly_kpi | Summarizes monthly shipment volume, cost, and delivery performance. | Provides quick insights for management and executive review. | Executive Scorecard |
| agg_dq_daily | Tracks daily data quality health. | Ensures analytical decisions are based on reliable data. | Data Quality Score |
| agg_fragile_damage | Flags delayed fragile shipments likely to be damaged. | Enables proactive damage claims and risk reduction. | Damage Risk Index |
| agg_dq_matrix | Shows multiple data quality issues across dates. | Helps quickly identify and resolve data pipeline problems. | DQ Coverage Index |


agg_carrier_efficiency
agg_carrier_perf
agg_daily_cost
agg_delivery_success
agg_dq_daily
agg_dq_matrix
agg_fragile_cost
agg_fragile_damage
agg_monthly_kpi
agg_payment_perf
agg_priority_status
agg_region_cost
agg_sla_breach
agg_warehouse_stress

A list of all the Aggregation tables in gold layer

## Sprint 4: Power BI & Validation Architecture

To expose curated Gold data to business users through Power BI dashboards while validating data accuracy, freshness, and quality across all layers.

### Architecture Overview

In Sprint 4, **Power BI** connects directly to the **Gold layer** using Databricks SQL endpoints or direct connectivity. This sprint focuses on semantic modeling, KPI creation, validation, and reconciliation to ensure business trust in the analytics.

### Key Architectural Components

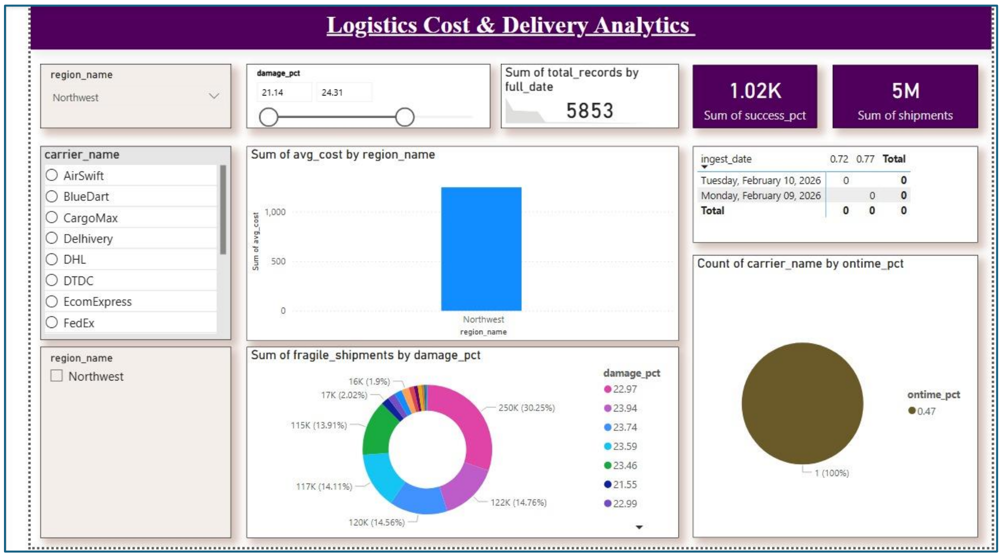
- **Source:** Gold Delta tables
- **Query Layer:** Databricks SQL / Direct Lake (as applicable)
- **Visualization Tool:** Power BI
- **Validation Layer:** Databricks notebooks and SQL checks

### Data Flow

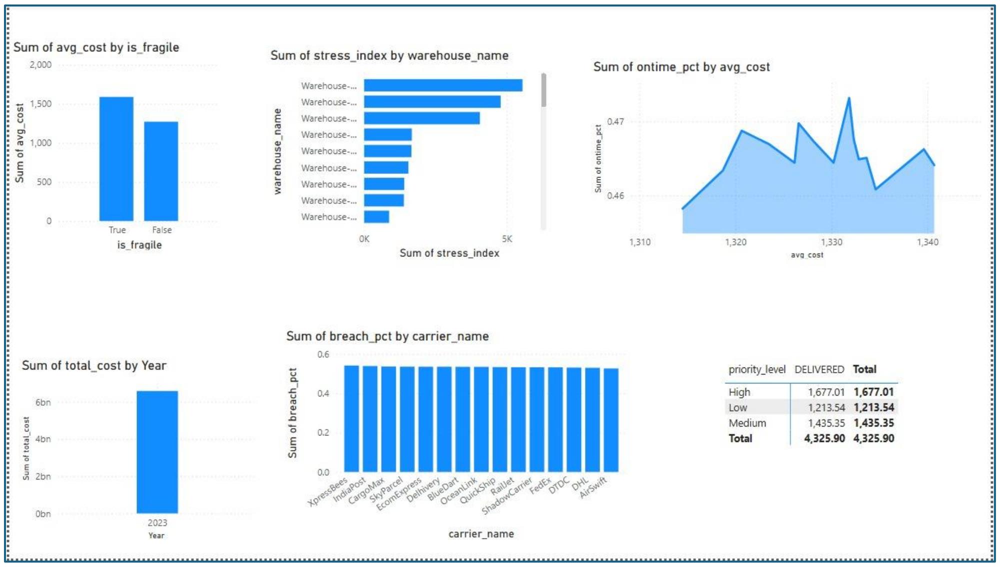
1. Power BI establishes a connection to Gold dimension and fact tables.
2. Relationships are defined using the star schema model.
3. Business measures and KPIs are created in the semantic layer.
4. Dashboards are built for executive, operational, and data quality views.
5. Validation checks reconcile record counts across Bronze, Silver, and Gold.
6. Data freshness and pipeline success metrics are verified.

### Architecture Benefits

- High-performance analytics using curated Gold data
- Clear separation between data engineering and reporting layers
- End-to-end validation ensures data accuracy and trust



| ingest_date | 0.72 | 0.77 | Total |
|---|---|---|---|
| Tuesday, February 10, 2026 | 0 | | 0 |
| Monday, February 09, 2026 | | 0 | 0 |
| Total | 0 | 0 | 0 |



| priority_level | DELIVERED | Total |
|---|---|---|
| High | 1,677.01 | 1,677.01 |
| Low | 1,213.54 | 1,213.54 |
| Medium | 1,435.35 | 1,435.35 |
| Total | 4,325.90 | 4,325.90 |

## 5. Challenges Faced During Implementation

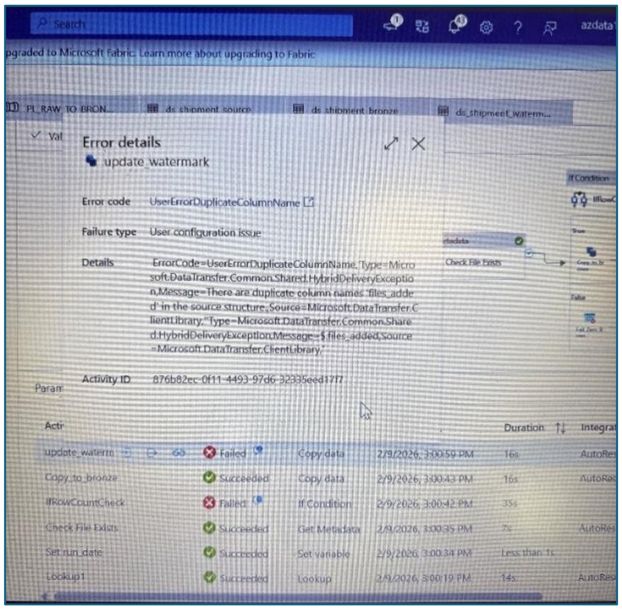
### 11.1 Data Ingestion Challenges

- Schema drift in incoming CSV files
- Handling late-arriving and duplicate data
- Watermark management for reruns
- ADF pipeline Creating Issues when updating from same watermark file is access to update

- Error Faced and Resolutions

**Duplicate Column Errors**

Caused by using Copy Activity to update JSON.



Error details
update_watermark

Error code UserErrorDuplicateColumnName

Failure type User configuration issue

Details

```
ErrorCode=UserErrorDuplicateColumnName,'Type=Microsoft.DataTransfer.Common.Shared.HybridDeliveryException,Message=There are duplicate column names 'files_added' in the source structure.,Source=Microsoft.DataTransfer.ClientLibrary,''Type=Microsoft.DataTransfer.Common.Shared.HybridDeliveryException,Message=$files_added,Source=Microsoft.DataTransfer.ClientLibrary,'
```

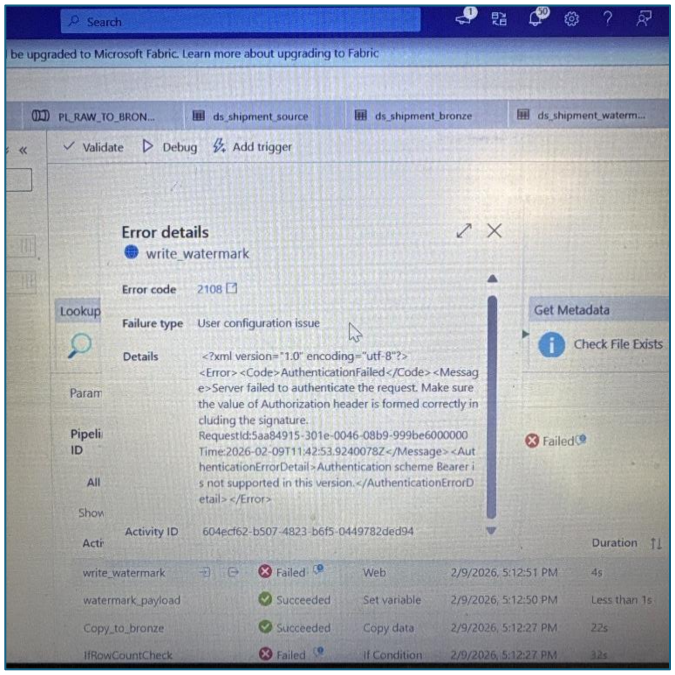
Activity ID 876b82ec-0f11-4493-97d6-32335eed17f7

| Activity | Status | Type | Run start | Duration | Integration runtime |
|---|---|---|---|---|---|
| update_waterm | Failed | Copy data | 2/9/2026, 3:00:59 PM | 16s | AutoRes |
| Copy_to_bronze | Succeeded | Copy data | 2/9/2026, 3:00:43 PM | 16s | AutoRes |
| IfRowCountCheck | Failed | If Condition | 2/9/2026, 3:00:42 PM | 35s | |
| Check File Exists | Succeeded | Get Metadata | 2/9/2026, 3:00:35 PM | 7s | AutoRes |
| Set run_date | Succeeded | Set variable | 2/9/2026, 3:00:34 PM | Less than 1s | |
| Lookup1 | Succeeded | Lookup | 2/9/2026, 3:00:19 PM | 14s | AutoRes |

Resolved by switching to Web Activity with full overwrite.

**Authentication Failures**

Managed Identity is not supported for Blob PUT operations.



Error details
write_watermark

Error code 2108

Failure type User configuration issue

Details

```
<?xml version="1.0" encoding="utf-8"?><Error><Code>AuthenticationFailed</Code><Message>Server failed to authenticate the request. Make sure the value of Authorization header is formed correctly including the signature.
RequestId:5aa84915-301e-0046-08b9-999be6000000
Time:2026-02-09T11:42:53.9240078Z</Message><AuthenticationErrorDetail>Authentication scheme Bearer is not supported in this version.</AuthenticationErrorDetail></Error>
```

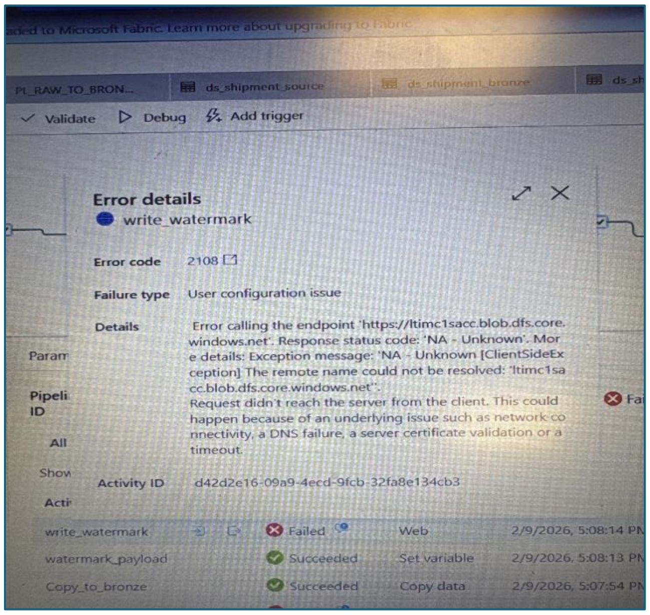
Activity ID 604ecf62-b507-4823-b6f5-0449782ded94

| Activity | Status | Type | Run start | Duration |
|---|---|---|---|---|
| write_watermark | Failed | Web | 2/9/2026, 5:12:51 PM | 4s |
| watermark_payload | Succeeded | Set variable | 2/9/2026, 5:12:50 PM | Less than 1s |
| Copy_to_bronze | Succeeded | Copy data | 2/9/2026, 5:12:27 PM | 22s |
| IfRowCountCheck | Failed | If Condition | 2/9/2026, 5:12:27 PM | 32s |

Resolved by using SAS token authentication.

### DFS Endpoint Errors

Web Activity does not support DFS endpoints.



Error details
write_watermark

Error code 2108

Failure type User configuration issue

Details Error calling the endpoint 'https://ltimc1sacc.blob.dfs.core.windows.net'. Response status code: 'NA - Unknown'. More details: Exception message: 'NA - Unknown [ClientSideException] The remote name could not be resolved: 'ltimc1sacc.blob.dfs.core.windows.net''.
Request didn't reach the server from the client. This could happen because of an underlying issue such as network connectivity, a DNS failure, a server certificate validation or a timeout.

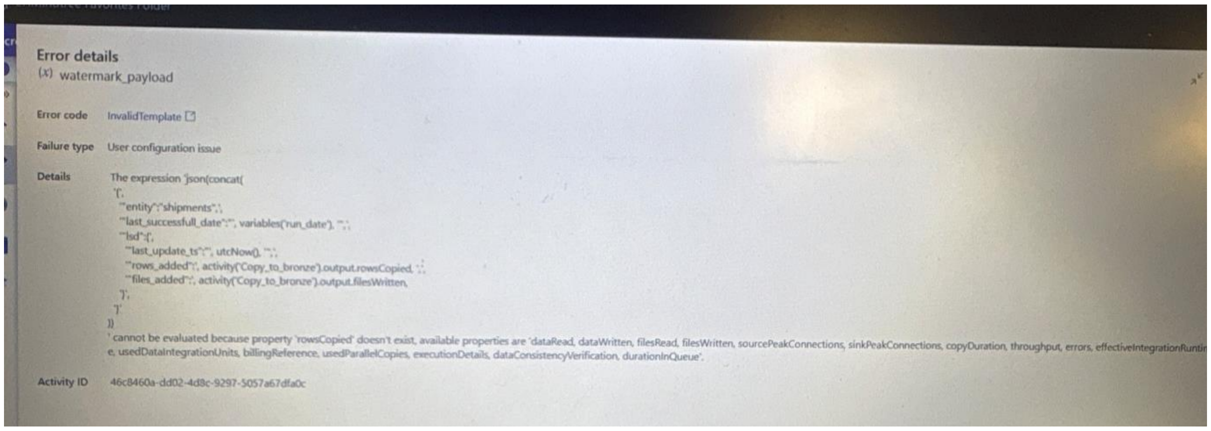
Activity ID d42d2e16-09a9-4ecd-9fcb-32fa8e134cb3

```
write_watermark     Failed     Web           2/9/2026, 5:08:14 PM
watermark_payload   Succeeded  Set variable  2/9/2026, 5:08:13 PM
Copy_to_bronze      Succeeded  Copy data     2/9/2026, 5:07:54 PM
```

Resolved by using Blob endpoint instead.

### Invalid Copy Output References

Incorrect usage of unsupported properties.



Error details
(x) watermark_payload

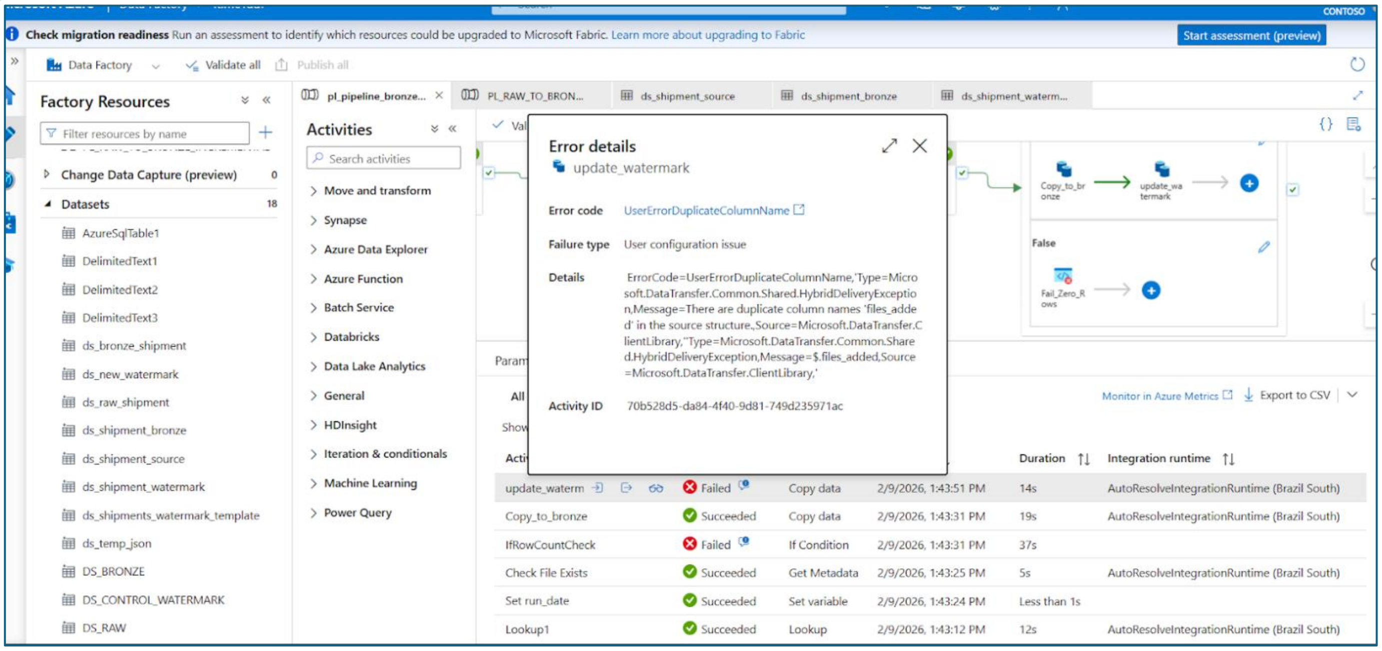
Error code InvalidTemplate

Failure type User configuration issue

Details
```
The expression 'json(concat(
 '{',
   '"entity":"shipments",',
   '"last_successfull_date":"', variables('run_date'), '",',
   '"lsd":{',
     '"last_update_ts":"', utcNow(), '",',
     '"rows_added":', activity('Copy_to_bronze').output.rowsCopied, ',',
     '"files_added":', activity('Copy_to_bronze').output.filesWritten,
   '}',
 '}'
))
' cannot be evaluated because property 'rowsCopied' doesn't exist, available properties are 'dataRead, dataWritten, filesRead, filesWritten, sourcePeakConnections, sinkPeakConnections, copyDuration, throughput, errors, effectiveIntegrationRuntim
e, usedDataIntegrationUnits, billingReference, usedParallelCopies, executionDetails, dataConsistencyVerification, durationInQueue'.
```

Activity ID 46c8460a-dd02-4d8c-9297-5057a67dfa0c

Resolved by using valid outputs such as *rowsRead* and *filesWritten*.



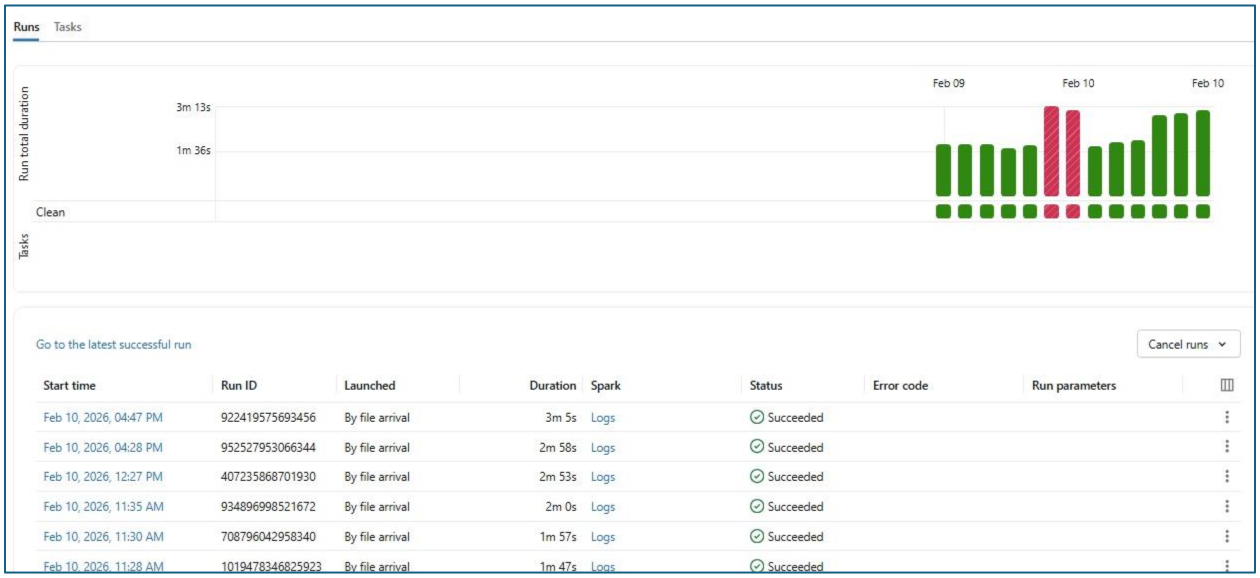
Error details
update_watermark

| | |
|---|---|
| Error code | UserErrorDuplicateColumnName |
| Failure type | User configuration issue |
| Details | ErrorCode=UserErrorDuplicateColumnName,'Type=Microsoft.DataTransfer.Common.Shared.HybridDeliveryException,Message=There are duplicate column names 'files_added' in the source structure.,Source=Microsoft.DataTransfer.ClientLibrary,''Type=Microsoft.DataTransfer.Common.Shared.HybridDeliveryException,Message=$.files_added,Source=Microsoft.DataTransfer.ClientLibrary,' |
| Activity ID | 70b528d5-da84-4f40-9d81-749d235971ac |

| Acti | | | | Duration | Integration runtime |
|---|---|---|---|---|---|
| update_waterm | Failed | Copy data | 2/9/2026, 1:43:51 PM | 14s | AutoResolveIntegrationRuntime (Brazil South) |
| Copy_to_bronze | Succeeded | Copy data | 2/9/2026, 1:43:31 PM | 19s | AutoResolveIntegrationRuntime (Brazil South) |
| IfRowCountCheck | Failed | If Condition | 2/9/2026, 1:43:31 PM | 37s | |
| Check File Exists | Succeeded | Get Metadata | 2/9/2026, 1:43:25 PM | 5s | AutoResolveIntegrationRuntime (Brazil South) |
| Set run_date | Succeeded | Set variable | 2/9/2026, 1:43:24 PM | Less than 1s | |
| Lookup1 | Succeeded | Lookup | 2/9/2026, 1:43:12 PM | 12s | AutoResolveIntegrationRuntime (Brazil South) |

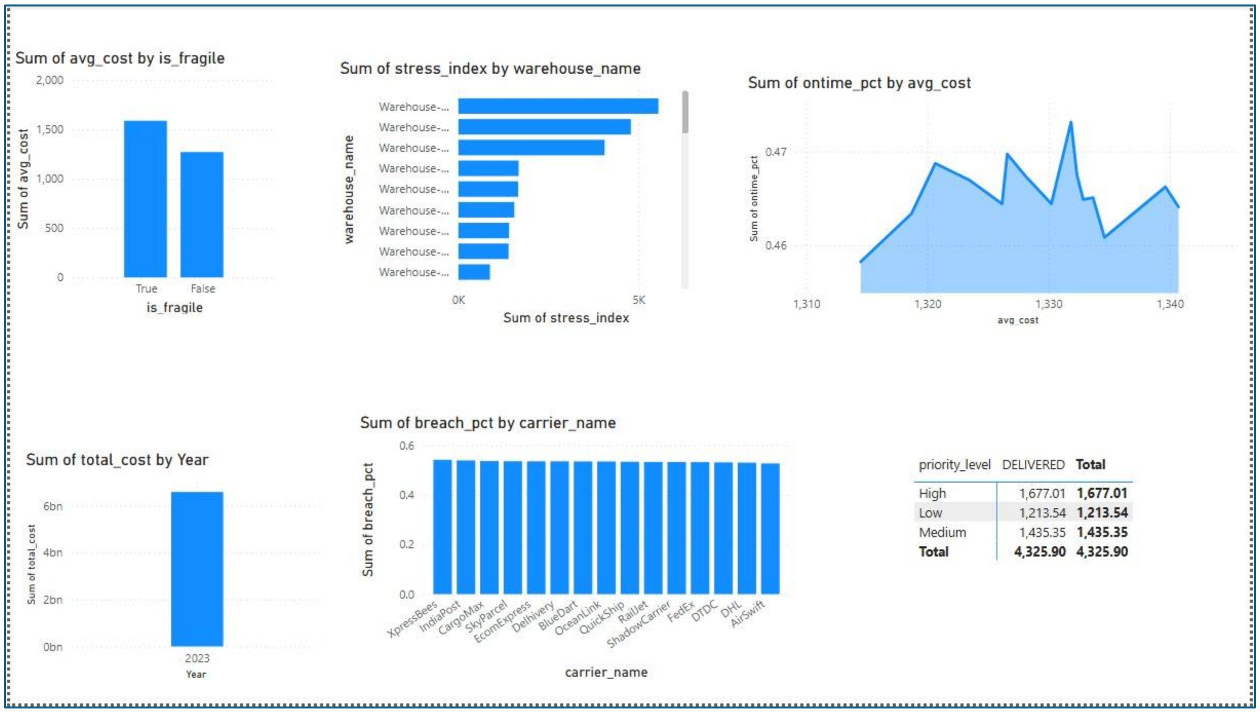
## 6. Success Highlights

- Automated daily ingestion with zero manual intervention
- Clean, reconciled data across Bronze, Silver, and Gold
- High-performing Power BI dashboards with accurate KPIs



| Start time | Run ID | Launched | Duration | Spark | Status | Error code | Run parameters |
|---|---|---|---|---|---|---|---|
| Feb 10, 2026, 04:47 PM | 922419575693456 | By file arrival | 3m 5s | Logs | Succeeded | | |
| Feb 10, 2026, 04:28 PM | 952527953066344 | By file arrival | 2m 58s | Logs | Succeeded | | |
| Feb 10, 2026, 12:27 PM | 407235868701930 | By file arrival | 2m 53s | Logs | Succeeded | | |
| Feb 10, 2026, 11:35 AM | 934896998521672 | By file arrival | 2m 0s | Logs | Succeeded | | |
| Feb 10, 2026, 11:30 AM | 708796042958340 | By file arrival | 1m 57s | Logs | Succeeded | | |
| Feb 10, 2026, 11:28 AM | 1019478346825923 | By file arrival | 1m 47s | Logs | Succeeded | | |

📷 Screenshot Placeholder – Power BI Dashboard Overview



| priority_level | DELIVERED | **Total** |
|---|---|---|
| High | 1,677.01 | **1,677.01** |
| Low | 1,213.54 | **1,213.54** |
| Medium | 1,435.35 | **1,435.35** |
| **Total** | **4,325.90** | **4,325.90** |

## 7. Conclusion

The Logistics Cost & Delivery Analytics Platform successfully demonstrates the design and implementation of a production-ready, enterprise-scale data engineering solution using Azure-native services. By leveraging Azure Data Factory for orchestration, Azure Data Lake Storage Gen2 for scalable storage, Azure Databricks with Delta Lake for distributed processing, and Power BI for visualization, the project delivers a fully automated and governed analytics ecosystem. The implementation of the Medallion Architecture (Bronze, Silver, Gold) ensures clear separation of concerns across ingestion, transformation, and analytical modeling layers. The Bronze layer preserves raw data integrity, the Silver layer enforces robust data cleaning and data quality validation, and the Gold layer provides a dimensional star schema optimized for high-performance reporting. The introduction of watermark-based incremental ingestion in ADF guarantees idempotent loads, recoverability, and operational auditability. A structured data quality framework was embedded within the Silver layer, enabling proactive detection of null values, duplicates, outliers, and logical anomalies. By classifying records into Minor, Major, and Critical severity levels, the platform ensures that only trusted data flows into downstream analytics. This governance-first approach enhances reliability and builds stakeholder confidence in reported KPIs. From a business perspective, the platform enables shipment cost governance, carrier SLA monitoring, fragile shipment tracking, regional cost optimization, and warehouse capacity visibility. The Gold-layer aggregations and Power BI dashboards transform operational data into actionable insights, supporting data-driven decision-making for logistics leadership. Overall, the project reflects modern data engineering best practices, including scalable architecture design, incremental processing, Delta Lake optimization, dimensional modeling, and end-to-end validation. It serves as a strong foundation for real-world enterprise analytics deployment within the logistics domain.