**Project Report: Automated Daily Data Ingestion Pipeline**

**Project: Automated Ingestion of Source Files into SQL Database**

**Author:** Sahil Srivastava

**1. Executive Summary**

This document outlines the design, implementation, and successful deployment of an automated data engineering pipeline using Microsoft Azure services. The primary objective of this project was to create a robust, scalable, and scheduled process to ingest three distinct types of daily source files from an Azure Data Lake into a structured Azure SQL Database.

The implemented solution leverages Azure Data Factory to orchestrate a daily truncate-and-load process. The pipeline dynamically identifies files, applies specific, rule-based transformations—including deriving date-based columns directly from filenames—and loads the processed data into their respective target tables. The entire process is now fully automated, ensuring that the destination database is updated daily with the latest data without any manual intervention.

**2. Project Requirements**

The core requirements for this project were as follows:

**Source Data:** Ingest three types of CSV files from an Azure Data Lake Storage container: • CUST\_MSTR\_*.csv • master\_child\_export-*.csv • H\_ECOM\_ORDER.csv

**Transformation Logic:** • For CUST\_MSTR files: Add a new Date column by extracting the date from the filename (e.g., 2019-11-12). • For master\_child\_export files: Add two new columns, date (e.g., 2019-11-12) and DateKey (e.g., 20191112), extracted from the filename. • For H\_ECOM\_ORDER files: Load the data as-is with no transformations.

**Loading Strategy:** Implement a daily truncate-and-load pattern, where the destination tables are cleared before ingesting the new day's files.

**Destination:** Load the processed data into three corresponding tables in an Azure SQL Database: CUST\_MSTR, master\_child, and H\_ECOM\_Orders.

**Automation:** The entire process must be scheduled to run automatically every day.

**3. Technical Architecture**

The solution was built using a serverless architecture composed of core Azure services, ensuring scalability, cost-effectiveness, and low maintenance overhead.

• **Azure Data Lake Storage Gen2:** Serves as the central repository (data lake) for storing the raw source CSV files. • **Azure Data Factory (ADF):** Acts as the central orchestration and ETL (Extract, Transform, Load) engine. A single ADF pipeline contains all the logic for file discovery, transformation, and loading. • **Azure SQL Database:** The destination relational database used to store the structured, processed data, ready for analytics and reporting.

**4. Pipeline Design & Workflow**

A single master pipeline, named **Daily\_File\_Load**, was created in Azure Data Factory to execute the entire workflow. The design is parallel, allowing for efficient and independent processing of each file type.

The pipeline executes the following sequence of activities:

**4.1 Get Metadata (Get\_File\_List)**

The pipeline begins by using a Get Metadata activity to scan the source container in the data lake and retrieve a complete list of all files present.

**4.2 Parallel Filtering (Branching)**

The output of the Get\_File\_List activity branches into three parallel flows, one for each file type. A Filter activity is used at the start of each branch to isolate the relevant files based on their naming convention.

• **Filter\_CUST\_MSTR:** Condition: @startswith(item().name, 'CUST\_MSTR') • **Filter\_master\_child:** Condition: @startswith(item().name, 'master\_child\_export') • **Filter\_H\_ECOM:** Condition: @startswith(item().name, 'H\_ECOM\_ORDER')

**4.3 Truncate Tables**

Following each filter, a Script activity connects to the Azure SQL Database and executes a TRUNCATE TABLE command on the corresponding destination table. This clears the table in preparation for the new data load.

**4.4 Iterate and Copy (ForEach loop)**

A ForEach loop is used to iterate over the filtered list of files for each branch. Inside each loop, a Copy Data activity performs the core data movement.

**4.5 Copy Data Activity Details**

The Copy Data activity within each loop is configured as follows:

**Source:** The source is a generic dataset pointing to the data lake container. The specific file to be copied in each loop iteration is identified dynamically using the expression @item().name.

**Transformation:** For the CUST\_MSTR and master\_child\_export branches, new columns are added at the source stage using the "Additional columns" feature. The values for these columns are derived dynamically from the filename (@item().name) using string manipulation expressions.

**CUST\_MSTR - Date column expression:**

@concat(substring(replace(item().name, 'CUST\_MSTR\_', ''), 0, 4), '-', substring(replace(item().name, 'CUST\_MSTR\_', ''), 4, 2), '-', substring(replace(item().name, 'CUST\_MSTR\_', ''), 6, 2))

**master\_child\_export - date column expression:**

@concat(substring(replace(replace(item().name, 'master\_child\_export-', ''), '.csv', ''), 0, 4), '-', substring(replace(replace(item().name, 'master\_child\_export-', ''), '.csv', ''), 4, 2), '-', substring(replace(replace(item().name, 'master\_child\_export-', ''), '.csv', ''), 6, 2))

**master\_child\_export - DateKey column expression:**

@replace(replace(item().name, 'master\_child\_export-', ''), '.csv', '')

**Sink:** The sink is configured to point to the specific destination table in the Azure SQL Database.

**Mapping:** The schema mapping is configured to ensure source columns (including the newly added ones) are correctly mapped to their destination columns.

**5. Automation and Scheduling**

To meet the automation requirement, an Azure Data Factory Trigger was created and attached to the Daily\_File\_Load pipeline.

• **Trigger Name:** Trigger\_Daily\_File\_Load • **Type:** Schedule • **Recurrence:** Every 1 day • **Status:** Activated

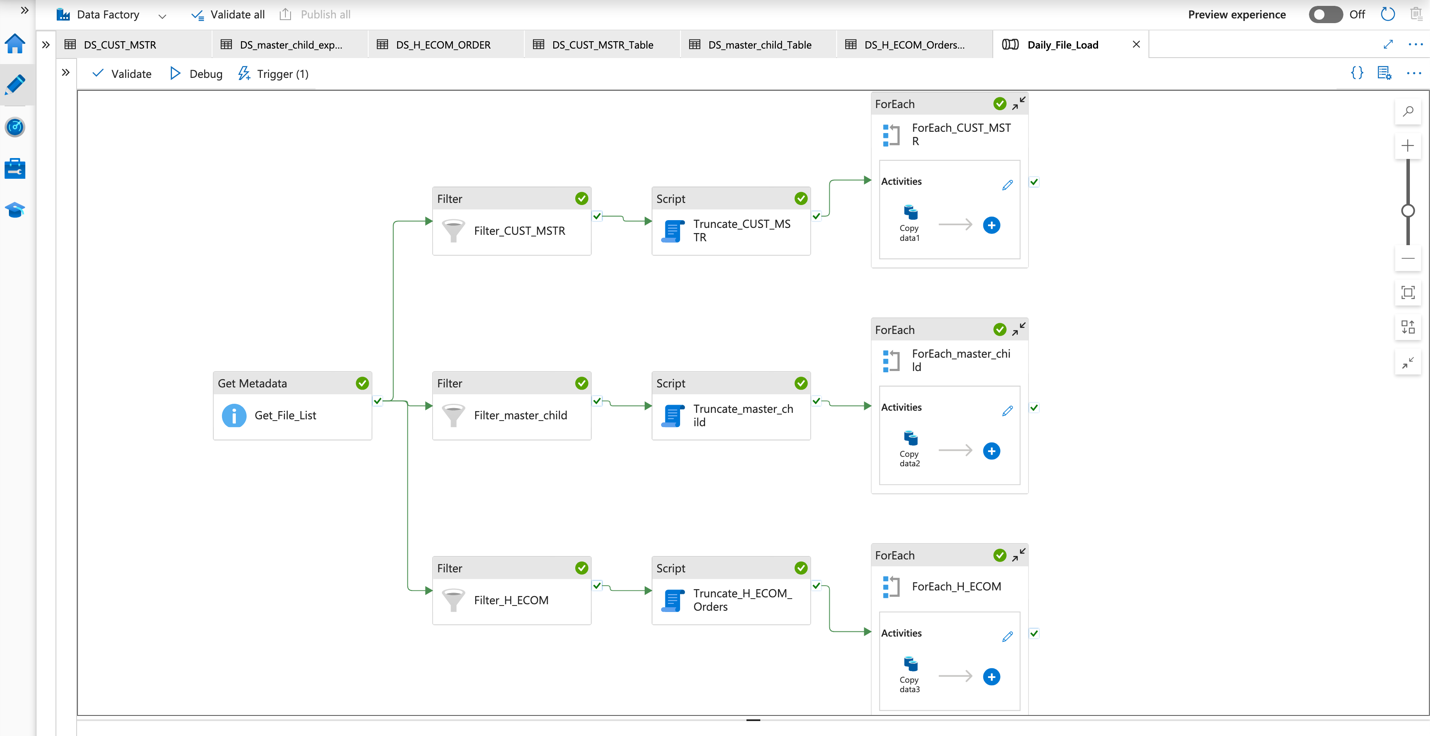
This trigger ensures the pipeline executes automatically at the specified time each day, making the entire data ingestion process hands-free.

**6. Proof of Execution**

The pipeline has been successfully developed, tested, and deployed. The following screenshots demonstrate the final pipeline design and a successful debug run, confirming that all activities completed as expected.

**Screenshot 1: Full Pipeline Canvas**

**Description:** This screenshot should show the complete Daily\_File\_Load pipeline with its three parallel branches.



**Screenshot 2: Successful Pipeline Run**

**Description:** This screenshot should show the pipeline run monitor, with a "Succeeded" status for the pipeline and all its activities.

A screenshot of a computer

Description automatically generated

**Screenshot 3: Data in SQL Table**

**Description:** A screenshot from SQL Server Management Studio or the Azure Portal's query editor showing the data successfully loaded into one of the tables (e.g., SELECT \* FROM dbo.CUST\_MSTR).

A screenshot of a computer

Description automatically generated

**7. Conclusion**

The automated data ingestion pipeline has been successfully implemented, meeting all project requirements. The solution is efficient, reliable, and fully automated, eliminating the need for manual data handling and reducing the risk of human error. This pipeline provides a solid foundation for any future analytical or reporting needs based on this data.