Use case scenario illustrating the practical application of **TableLog** and **Error\_Logging** in a data pipeline context:

**Use Case: ETL Monitoring and Error Handling in a Retail Data Warehouse**

**Background**

RetailCo, a retail company, has a data warehouse that aggregates data from various source systems, including sales, inventory, and customer databases. The company runs ETL (Extract, Transform, Load) pipelines in Azure Data Factory (ADF) to load data into the warehouse. Ensuring that ETL processes run smoothly and errors are properly logged is crucial for maintaining data integrity and operational efficiency.

**Requirements**

1. **Track ETL Pipeline Performance**: Monitor the performance of each ETL pipeline, including start and end times, record counts, and overall status.
2. **Capture and Investigate Errors**: Log detailed information about errors encountered during pipeline execution to facilitate troubleshooting and resolution.

**Sample data** for **TableLog and Error\_Logging tables** based on the ETL pipeline use case:

1. **Create TableLog Table**

The TableLog table is used to store execution details for each ETL pipeline.

**TableLog Sample Data:**



1. **Error\_Logging Sample Data**



**Hints: for Implementation Steps**

**1. ETL Pipeline Execution Tracking**

**Objective**: Track each ETL pipeline execution to monitor performance and identify any issues related to data loading.

1. **Log Pipeline Start**

At the beginning of each pipeline run, log the start time and initial details.

* + **Pipeline Start Activity**: Use an **Execute Stored Procedure Activity** in ADF.

sql

INSERT INTO TableLog (SourceSystem, MaterPipline\_ID, Master\_Pipline\_Name, Pipline\_ID, Pipline\_name, Table\_name, Load\_StartTime, Status)

VALUES (@SourceSystem, @MaterPipline\_ID, @Master\_Pipline\_Name, @Pipline\_ID, @Pipline\_name, @Table\_name, GETDATE (), 'Running');

1. **Log Pipeline End**

At the end of each pipeline run, update the record with the end time, record counts, and status.

* + **Pipeline End Activity**: Use another **Execute Stored Procedure Activity**.

UPDATE TableLog

SET Load\_EndTime = GETDATE(),

TargetCount = @TargetCount,

Status = @Status

WHERE Pipline\_ID = @Pipline\_ID

AND Table\_name = @Table\_name

AND Load\_EndTime IS NULL;

* + **Parameters**: Pass values for Pipline\_ID, Table\_name, TargetCount, and Status.

**2. Error Logging**

**Objective**: Capture detailed information about errors to facilitate troubleshooting and improve ETL processes.

1. **Create Error\_Logging Table**

The Error\_Logging table is used to store error details.

CREATE TABLE Error\_Logging (

Error\_ID INT PRIMARY KEY IDENTITY,

MaterPipline\_ID INT,

Master\_Pipline\_Name NVARCHAR(128),

Pipline\_ID INT,

Pipline\_name NVARCHAR(128),

Table\_Name NVARCHAR(128),

Activity\_Name NVARCHAR(128),

Error\_Description NVARCHAR(MAX),

Error\_DateTime DATETIME

);

1. **Log Errors**

Configure your pipeline to log errors if any activity fails.

* + **Error Handling**: In ADF, use **On Failure** settings to execute an error logging procedure.
  + **Error Logging Activity**: Use **Execute Stored Procedure Activity** or **Web Activity** to insert error details.

INSERT INTO Error\_Logging (MaterPipline\_ID, Master\_Pipline\_Name, Pipline\_ID, Pipline\_name, Table\_Name, Activity\_Name, Error\_Description, Error\_DateTime)

VALUES (@MaterPipline\_ID, @Master\_Pipline\_Name, @Pipline\_ID, @Pipline\_name, @Table\_Name, @Activity\_Name, @Error\_Description, GETDATE());

* + **Parameters**: Pass values such as MaterPipline\_ID, Master\_Pipline\_Name, Pipline\_ID, Pipline\_name, Table\_Name, Activity\_Name, and Error\_Description.

**Example Scenario**

**Scenario: Monthly Sales Data Load**

1. **Pipeline Execution**:
   * RetailCo has a monthly ETL pipeline that loads sales data from their SQL Server database into the data warehouse.
2. **Tracking Execution**:
   * When the pipeline starts, an entry is made in TableLog with the Load\_StartTime.
   * After processing, the pipeline updates the entry with Load\_EndTime, TargetCount, and Status (e.g., 'Success' or 'Failure').
3. **Handling Errors**:
   * If an error occurs during the pipeline execution (e.g., data transformation fails), the pipeline logs detailed error information into Error\_Logging with the error description and timestamp.

**Benefits**

* **Performance Monitoring**: Provides insights into ETL performance, including load times and record counts, helping to identify performance bottlenecks.
* **Error Tracking**: Facilitates quick identification and resolution of errors by logging detailed information about the error context and activity.
* **Operational Efficiency**: Enhances the reliability and maintainability of ETL processes by automating performance tracking and error logging.

This use case demonstrates how TableLog and Error\_Logging can be effectively used to monitor and manage ETL pipelines, ensuring that data integration processes run smoothly and errors are promptly addressed