**Lab Document – Metadata-Driven ADF Pipeline with Logging & Restart**

**Prerequisites**

* Azure Subscription
* Azure Data Factory (ADF) instance
* Azure SQL Database or Azure Blob Storage (for control table)
* Azure Data Lake Storage Gen2
* Azure Databricks workspace (for logging to Delta)
* Sample dataset with incremental column (e.g., LastUpdatedDate)

**Step 1: Create Control Table**

**Use Case:**

Define metadata to dynamically ingest multiple tables from source to target.

**Sample Table (SQL or Blob):**

CREATE TABLE IngestionControl (

SourceName VARCHAR(100),

TableName VARCHAR(100),

SourcePath VARCHAR(500),

TargetPath VARCHAR(500),

WatermarkColumn VARCHAR(100),

IsActive BIT,

LastWatermark DATETIME

);

**Sample Data:**

sql

INSERT INTO IngestionControl VALUES

('crm', 'Customers', 'raw/crm/customers.csv', 'bronze/crm/customers', 'LastUpdatedDate', 1, '2023-01-01'),

('crm', 'Orders', 'raw/crm/orders.csv', 'bronze/crm/orders', 'LastUpdatedDate', 1, '2023-01-01')

**Step 2: Create Linked Services**

* AzureBlobStorage (source)
* AzureDataLakeStorage (sink)
* AzureSQLDatabase (for control table)
* AzureDatabricks (optional for logging to Delta)

**Step 3: Create Parameterized Dataset**

**3.1 Dataset → DelimitedText**

Add parameters:

* filePath
* fileName

Bind the dataset's file path:

@dataset().filePath/@dataset().fileName

**Step 4: Lookup and ForEach Activities**

**4.1 Add Lookup Activity**

* Query the control table:

SELECT \* FROM IngestionControl WHERE IsActive = 1

**4.2 Add ForEach Activity**

* Items: @activity('LookupTables').output.value
* Inside ForEach:
  + Copy Activity / Data Flow
  + Pass parameters using:

json

@item().SourcePath

@item().TargetPath

@item().WatermarkColumn

@item().LastWatermark

**Step 5: Watermark Filter Logic (Data Flow)**

**Inside Data Flow:**

* Add **Source**: Use parameterized path
* Add **Filter**:

toTimestamp(columnName) > toTimestamp($lastWatermark)

* Add **Sink**: Use parameterized sink path

**Step 6: Orchestration Logging to Delta (via Databricks)**

**Create a Notebook (Python):**

from delta.tables import \*

from pyspark.sql.functions import lit

import datetime

log\_df = spark.createDataFrame([{

"pipeline": dbutils.widgets.get("pipeline"),

"activity": dbutils.widgets.get("activity"),

"status": dbutils.widgets.get("status"),

"timestamp": datetime.datetime.now().isoformat()

}])

log\_df.write.format("delta").mode("append").save("/mnt/logs/pipeline\_run\_log")

**Call from ADF via Databricks Notebook activity:**

Pass parameters: pipeline, activity, status

**Step 7: Implement Restart Script Demo**

**Use Case:**

Avoid re-running successful ingestions.

**Steps:**

* After each activity, log success to a status table or blob.
* On next run, Lookup status and skip already successful ones via If Condition.

**Optional: Use Pipeline parameter StartFromTable to control resumption.**

**Success Criteria**

* Ingested multiple tables using metadata
* Filtered incrementally using watermark
* Logs generated to Delta table
* Can resume from failure point using restart logic

**Bonus Tips**

* Add alerting via Webhook (Teams) for failures
* Validate schema/data using Assert transformation in Data Flow
* Use global parameters for commonly used values (e.g., base path, watermark store)