**Case Study: Idempotent Ingestion of 10 Tables via Configuration with Resume Capability in Azure Data Factory**

**Objective**

Design a scalable, metadata-driven ingestion framework in Azure Data Factory (ADF) to:

* Ingest **10 source tables** using configuration metadata
* Implement **watermark-based incremental ingestion**
* Ensure **idempotent (no duplicate) loads**
* Enable **resumption** from point of failure without reprocessing successful loads

**Architecture Overview**

| **Component** | **Purpose** |
| --- | --- |
| **Control Table** | Stores metadata for source-target mappings, watermark values, and status |
| **ADF Pipeline** | Orchestrates Lookup → ForEach → Copy/Data Flow activities |
| **Watermark Column** | Tracks last ingested datetime to support incremental loads |
| **Status Table or Logging** | Logs run status to avoid reprocessing |
| **Restart Logic** | Skips already successful tables dynamically |

**Control Table Schema (SQL)**

CREATE TABLE IngestionControl (

TableName VARCHAR(100),

SourcePath VARCHAR(500),

SinkPath VARCHAR(500),

WatermarkColumn VARCHAR(100),

LastWatermark DATETIME,

IsActive BIT,

Status VARCHAR(50),

LastRunTime DATETIME

);

**Sample Entries:**

| **TableName** | **SourcePath** | **SinkPath** | **WatermarkColumn** | **LastWatermark** | **IsActive** | **Status** |
| --- | --- | --- | --- | --- | --- | --- |
| Customers | raw/crm/customers | bronze/crm/customers | updated\_at | 2023-01-01 | 1 | Success |
| Orders | raw/crm/orders | bronze/crm/orders | updated\_at | 2023-01-01 | 1 | Failed |
| Products | raw/crm/products | bronze/crm/products | updated\_at | 2023-01-01 | 1 | Pending |

**Workflow Design**

**Pipeline Logic:**

1. **Lookup Control Table** – Fetch active tables
2. **Filter Out 'Success' Records** – To skip already processed
3. **ForEach Activity** – Loop through each pending table
4. **Inside ForEach**:
   * Parameterized Data Flow
   * Watermark filter
   * Update watermark and status after completion

**Data Flow (Inside ForEach)**

**Parameters:**

* sourcePath, sinkPath, watermarkColumn, lastWatermark

**Steps:**

1. **Source** → Read from blob using dynamic path
2. **Filter** → Use watermark condition:

to\_timestamp(watermarkColumn) > to\_timestamp($lastWatermark)

1. **Sink** → Write to target path (Delta format for idempotency)
2. **Post-Processing** → Send updated watermark and success/failure status to SQL table via Stored Proc or script

**Resume Logic (Idempotency)**

**On Restart:**

* Lookup only where Status != 'Success'
* Only failed or pending tables will be processed
* Success logs are persisted in the IngestionControl table

**Optional Enhancements:**

* Use Data Flow to update LastWatermark column post ingestion
* Create a PowerShell or Logic App to update status table from ADF logs in case of external errors

**Monitoring and Alerting**

* Use ADF pipeline failure email alerts or **Teams Webhook**
* Log each activity to a Delta table for monitoring:

{

"tableName": "Orders",

"status": "Success",

"rowsWritten": 230,

"runTimestamp": "2025-08-02T08:30:00"

}

**Best Practices**

* Always use timestamp-based watermark column with UTC standard
* Store previous watermark separately for rollback
* Use retry logic with exponential backoff in Copy/Data Flow activities
* Partition sink datasets by ingestion date

**Outcome**

* Achieved zero duplication using Delta idempotency and watermark filtering
* Maintained **resumability** with control table status flags
* Reduced reprocessing cost by 90% after implementing skip-success logic
* Enabled flexible table ingestion using **only one pipeline**

**Extensions**

* Add schema drift handling for evolving source structures
* Use Get Metadata + Copy for unknown schema support
* Auto-sync LastWatermark from target Delta Lake via MAX(watermark\_column)