**1. Control-Table Schema**

**What It Is:**

A control table is a metadata-driven pattern used to control pipeline execution dynamically. It contains configuration entries for data sources, table names, paths, and parameters needed for each ingestion job.

**Typical Schema Fields:**

| **Column Name** | **Description** |
| --- | --- |
| SourceName | Name of the source system |
| TableName | Name of the table to ingest |
| SourcePath | Location in the source system |
| TargetPath | Destination in ADLS/Data Lake |
| WatermarkColumn | Column used for incremental loads |
| IsActive | Flag to enable/disable ingestion for a table |

**Best Practices:**

* Keep it normalized and easily extensible.
* Add auditing fields like LastRunTime, Status, Remarks.

**2. Dynamic Dataset Reference**

**What It Is:**

A way to create datasets that are parameterized, so they can be reused across multiple pipeline activities dynamically.

**Benefits:**

* Avoid dataset duplication.
* Fully metadata-driven pipelines.
* Easier maintenance and scalability.

**How to Implement:**

1. Define parameters in the dataset (e.g., file path, file name, table name).
2. Pass values from pipeline or control table via expressions.
3. Use in source/sink of Data Flow or Copy Activity.

**Example:**

"filePath": "@dataset().path"

**3. ForEach / Lookup**

**Lookup Activity:**

* Used to retrieve data (usually a control table or configuration) from a database or blob.
* Returns first row (single row) or array (multiple rows).

**ForEach Activity:**

* Iterates over the items returned by Lookup.
* Executes nested activities per iteration (e.g., dynamic Copy/DataFlow activities).

**Use Case:**

Execute ingestion logic for each table defined in a control table.

**Example Pattern:**

Lookup → ForEach (Item in tableList) → Execute Copy or Data Flow

**4. Watermark-Resume Pattern**

**What It Is:**

A method to perform **incremental data loads** using a watermark column (e.g., LastModifiedDate).

**Components:**

* **Watermark column:** Used to track last loaded record.
* **Watermark store:** Can be Azure SQL DB, Blob, or Delta table.
* **Pipeline logic:** Retrieves last watermark → applies filter → writes → updates watermark.

**Best Practices:**

* Always store the watermark **after** successful load.
* Handle nulls and defaults gracefully.

**Example Filter:**

WHERE LastModifiedDate > '@{pipeline().parameters.lastWatermark}'

**5. Orchestration Logging to Delta**

**Why Log to Delta?**

* Durable, queryable logs.
* Supports schema evolution.
* Scales well with Spark & Databricks.

**What to Log:**

* Pipeline Name
* Activity Name
* Status (Success/Failure)
* Start/End Time
* Parameters used
* Error messages (if any)

**How to Implement:**

* Use **Web Activity** or **Custom Logging Notebook**.
* Write structured JSON to a Delta table.

**Sample Schema:**

{

"pipeline": "IngestSalesData",

"activity": "CopyActivity",

"status": "Succeeded",

"startTime": "2025-08-02T10:00:00Z",

"endTime": "2025-08-02T10:05:00Z"

}

**6. Restart Script Demo**

**Why Needed:**

For large orchestration pipelines, failures may occur mid-way. Restart scripts help resume execution from a failure point without reprocessing already successful parts.

**How to Approach:**

* Maintain a checkpoint (log table, file, or flag).
* Evaluate which steps completed.
* Restart pipeline from next logical activity using:
  + Parameters
  + If Conditions
  + Variables

**Tools:**

* PowerShell script
* Azure CLI or REST API
* Pipeline parameters to control conditional execution

**Summary Table:**

| **Concept** | **Key Use** |
| --- | --- |
| Control Table | Metadata-driven dynamic execution |
| Dynamic Dataset | Reusable, parameter-driven dataset |
| ForEach/Lookup | Loop through dynamic configurations |
| Watermark Resume | Incremental load management |
| Logging to Delta | Durable and scalable activity tracking |
| Restart Scripts | Graceful recovery from failures |