

# LAB 05: Streaming & Auto Loader

---

**Duration:** ~35 min | **Day:** 2 | **After module:** M05: Incremental Data Processing | **Difficulty:** Intermediate-Advanced

---

## Scenario

---

*“New order files arrive continuously in the landing zone. Set up Auto Loader for streaming JSON ingestion into the Bronze layer with exactly-once guarantees. Then explore Change Data Feed (CDF) for incremental ETL.”*

---

## Objectives

---

After completing this lab you will be able to: - Use `COPY INTO` for idempotent batch loading - Configure Auto Loader ( `cloudFiles` ) for streaming ingestion - Use `trigger(availableNow=True)` for incremental processing - Verify checkpoint-based exactly-once guarantees - Add metadata columns to streaming writes - Handle schema evolution with rescued data column - Perform a stream-static join - Use Change Data Feed (CDF) for incremental ETL

---

## Prerequisites

---

- Cluster running and attached to notebook
- Stream files available in `dataset/orders/stream/`

- Setup cell creates customers table if needed (for Task 8)

---

## Tasks Overview

---

Open `LAB_05_code.ipynb` and complete the `# TODO` cells.

Task	What to do	Key concept
Task 1	COPY INTO (Batch)	<code>COPY INTO table FROM path</code> <code>FILEFORMAT = JSON</code>
Task 2	Verify Idempotency	Re-run COPY INTO — 0 new rows
Task 3	Configure Auto Loader Stream	<code>.format("cloudFiles")</code> , <code>cloudFiles.format</code> , <code>schemaLocation</code>
Task 4	Write Stream	<code>.trigger(availableNow=True)</code> , <code>.toTable()</code>
Task 5	Incremental Processing	Re-run stream — checkpoint prevents reprocessing
Task 6	Metadata Columns	<code>current_timestamp()</code> , <code>col("_metadata.file_path")</code>
Task 7	Schema Evolution — Rescued Data	<code>schemaEvolutionMode = "rescue"</code> , <code>_rescued_data</code> column
Task 8	Stream-Static Join	Join streaming orders with static customers table
Task 9	Change Data Feed (CDF)	Enable CDF, make changes, read with <code>table_changes()</code>

---

## Detailed Hints

---

### Task 1: COPY INTO

- `FILEFORMAT = JSON`

- COPY INTO is idempotent — tracks which files have been loaded

### Task 3: Auto Loader

- Format: `"cloudFiles"`
- `cloudFiles.format : "json"`
- Schema location stores the inferred schema for evolution tracking

### Task 4: Write Stream

- `.trigger(availableNow=True)` — processes all available files and stops
- Always specify `checkpointLocation`

### Task 6: Metadata

- `current_timestamp()` for processing time
- `col("_metadata.file_path")` for source file path

### Task 7: Rescued Data

- Set `cloudFiles.schemaEvolutionMode` to `"rescue"`
- Extra columns not in the defined schema land in `_rescued_data`

### Task 8: Stream-Static Join

- Stream side:  
`spark.readStream.format("delta").table(target_table)`
- Static side: `spark.table("catalog.schema.customers")`
- Join on `"customer_id"` with `how="left"`

### Task 9: CDF

- Enable: `TBLPROPERTIES ('delta.enableChangeDataFeed' = 'true')`
  - Read changes: `table_changes('table_name', start_version)`
  - Filter: `_change_type IN ('insert', 'update_postimage')`
-

## Summary

---

In this lab you: - Used COPY INTO for idempotent batch loading - Configured Auto Loader (cloudFiles) for streaming ingestion - Used trigger(availableNow=True) for incremental processing - Verified checkpoint-based exactly-once guarantees - Added metadata columns to streaming writes - Used rescued data column for schema evolution handling - Performed a stream-static join to enrich streaming data - Used Change Data Feed (CDF) for incremental ETL

Feature	COPY INTO	Auto Loader	CDF
Format	SQL command	readStream/ writeStream	table_changes()
Scalability	Thousands of files	Millions of files	Any Delta table
Schema evolution	Manual	Automatic (rescue)	Follows source
File tracking	SQL state	Checkpoint directory	Version-based

**Exam Tip:** Auto Loader uses `cloudFiles` format. COPY INTO is simpler but Auto Loader scales better. CDF captures row-level changes ( `insert` , `update_preimage` , `update_postimage` , `delete` ).

**What's next:** In LAB 06 you will build analytical reports using window functions, CTEs, explode, and higher-order functions.