# End-to-End Databricks: DLT + Auto Loader + Unity Catalog (UC) Lineage

A practical, interview-ready walk-through with runnable SQL/Python and architecture notes.

| Layer | Tooling | Purpose |
|---|---|---|
| Ingest | Auto Loader (Structured Streaming) | Incremental file discovery with schema inference/evolution |
| Bronze/Silver/Gold | Delta Live Tables | Quality rules, dedup, watermarking, aggregations |
| Governance | Unity Catalog | Catalogs, schemas, grants, lineage |
| Serve | Materialized View | BI acceleration / cached aggregates |

## DLT Pipeline (SQL)

```
-- dlt_pipeline.sql -- Delta Live Tables pipeline: bronze -> silver -> gold --
Assumes Unity Catalog catalog.schema are set by the pipeline settings. -- Data
quality via EXPECT; watermark & dedup included. CREATE STREAMING LIVE TABLE
bronze_transactions COMMENT "Raw transaction events from Auto Loader" AS SELECT
CAST(value:transaction_id AS STRING) AS transaction_id, CAST(value:user_id AS
STRING) AS user_id, CAST(value:amount AS DOUBLE) AS amount, CAST(value:event_time AS
TIMESTAMP) AS event_time, current_timestamp() AS ingest_ts FROM
STREAM(LIVE.raw_transactions_json); -- Quality checks on bronze ALTER STREAMING LIVE
TABLE bronze_transactions SET TBLPROPERTIES ("quality" = "bronze"); -- Declare the
streaming source as a live view over the raw files table CREATE STREAMING LIVE VIEW
raw_transactions_json AS SELECT from_json(body, 'transaction_id string, user_id
string, amount double, event_time timestamp') AS value FROM
cloud_files("${source_path}", "${source_format}",
map("cloudFiles.inferColumnTypes","true")); CREATE LIVE TABLE silver_transactions
TBLPROPERTIES ("quality" = "silver") AS SELECT transaction_id, user_id, amount,
event_time, ingest_ts FROM LIVE.bronze_transactions WHERE amount > 0 AND event_time
>= (current_timestamp() - INTERVAL 30 DAYS); -- Expectations (quality rules) CREATE
EXPECTATION silver_amount_positive IF amount > 0 ON TABLE silver_transactions; --
Deduplicate by transaction_id with latest event_time CREATE LIVE TABLE
silver_transactions_dedup TBLPROPERTIES ("quality" = "silver") AS SELECT * FROM (
SELECT *, ROW_NUMBER() OVER (PARTITION BY transaction_id ORDER BY event_time DESC)
AS rn FROM LIVE.silver_transactions ) WHERE rn = 1; CREATE LIVE TABLE gold_user_spend
TBLPROPERTIES ("quality" = "gold") AS SELECT user_id, SUM(amount) AS
total_spent_30d, COUNT(*) AS txn_count_30d, MAX(event_time) AS last_txn_at FROM
LIVE.silver_transactions_dedup GROUP BY user_id;
```

## Auto Loader (Python)

```
# autoloader_ingest.py # Databricks Auto Loader to ingest raw JSON to a Delta table
in Unity Catalog. from pyspark.sql import functions as F SOURCE_PATH =
"/mnt/raw/transactions" CHECKPOINT = "/mnt/chk/autoloader/transactions" TARGET_TABLE
= "raw.transactions_raw" # UC: .. df = ( spark.readStream .format("cloudFiles")
.option("cloudFiles.format", "json") .option("cloudFiles.inferColumnTypes", "true")
.option("cloudFiles.schemaLocation", "/mnt/autoloader_schemas/transactions")
.load(SOURCE_PATH) ) # Basic cleanup + watermark for late data handling clean = (
df.withColumn("event_time", F.col("event_time").cast("timestamp"))
.withColumn("amount", F.col("amount").cast("double")) .filter(F.col("amount") > 0)
.withWatermark("event_time", "2 hours") ) ( clean.writeStream .format("delta")
.option("checkpointLocation", CHECKPOINT) .outputMode("append")
.toTable(TARGET_TABLE) )
```

## Unity Catalog Setup (SQL)

```
-- uc_setup.sql -- Create catalog/schema, sample external locations, and grants. --
Adjust names/locations per your workspace. CREATE CATALOG IF NOT EXISTS analytics
COMMENT 'Analytics catalog'; USE CATALOG analytics; CREATE SCHEMA IF NOT EXISTS core
COMMENT 'Core analytics schema'; CREATE SCHEMA IF NOT EXISTS raw COMMENT 'Raw
landing'; -- Optional: create external volume or managed locations as needed --
Create base tables if not created by pipelines CREATE TABLE IF NOT EXISTS
raw.transactions_raw ( transaction_id STRING, user_id STRING, amount DOUBLE,
event_time TIMESTAMP ) USING DELTA; -- Grants (principle of least privilege) GRANT
USAGE ON CATALOG analytics TO `data_engineers`, `analysts`; GRANT USAGE ON SCHEMA
analytics.core TO `data_engineers`, `analysts`; GRANT SELECT ON ALL TABLES IN SCHEMA
analytics.core TO `analysts`; GRANT MODIFY, SELECT ON ALL TABLES IN SCHEMA
analytics.core TO `data_engineers`; -- Future grants ALTER SCHEMA analytics.core
OWNER TO `data_engineers`; GRANT SELECT ON FUTURE TABLES IN SCHEMA analytics.core TO
`analysts`;
```

## Lineage/Grants (SQL)

```
-- uc_lineage.sql -- Explore lineage using UC system tables (availability varies by
workspace/runtime). -- Example: find lineage for a target table -- SELECT * -- FROM
system.information_schema.table_lineage -- WHERE target_table_full_name =
'analytics.core.gold_user_spend'; -- Example: find upstream tables feeding a model
-- SELECT upstream_table_full_name, downstream_table_full_name -- FROM
system.information_schema.table_lineage -- WHERE downstream_table_full_name =
'analytics.core.gold_user_spend'; -- Example: list grants to audit access SHOW
GRANTS ON SCHEMA analytics.core; SHOW GRANTS ON TABLE
analytics.core.gold_user_spend;
```

## Interview Q&A; (Senior)

Q: How does DLT ensure reliability vs notebooks?
A: Declarative DAG, managed checkpoints, expectations, automatic retries, event logs, lineage.

Q: How do Auto Loader and DLT split responsibilities?
A: Auto Loader lands raw to Delta (ingest), DLT curates (transform).

Q: How do you enforce PII governance with UC?
A: Use catalogs/schemas with grants; apply dynamic masks or row filters at table level; audit via system tables.

Q: What's your strategy for late data?
A: Watermarks and dedup on event_time; horizon windows in silver; revise gold aggregates incrementally.

Q: How do you productionize?

A: Workflows-triggered pipelines, env-specific configs, secrets, CI checks, and cost-aware autoscaling.