

## Task 1: Ingestion, Redaction & Provenance Pipeline

### Objective

Build a secure, compliant data ingestion pipeline that processes multiple healthcare input sources, performs PHI redaction, produces a QLM-ready dataset, and generates full provenance and integrity logs.

### Data Sources Used

Two raw files were used (JSONL + CSV), each containing 200 patient entries:

1. **source1\_clinical\_notes.jsonl** - Clinical notes with doctor name, patient name, MRN, contact number, and free-text medical notes.
2. **source2\_lab\_reports.csv** - Lab report values such as glucose, CBC, lipid panel, etc.

Supporting metadata:

- phi\_rules.json – regex rules for PHI redaction
- schema\_definitions.json – expected schemas
- metadata.json – source descriptors

### Pipeline Overview

#### 1. Raw File Upload

- Files uploaded to: dbfs:/FileStore/task1\_input\_files/

#### 2. Ingestion (Raw → DataFrame)

- Read JSONL and CSV with strict schemas.
- Convert timestamps/dates.
- Basic row validation (missing required fields flagged).

#### 3. PHI Redaction

Applied regex rules to remove: Patient names, Doctor names, MRNs, Contact Numbers, Addresses.

#### 4. Canonical (QLM-Ready) Format

Created a unified dataset with columns: patient\_id, source\_record\_id, event\_timestamp, clinical\_text, doctor\_name, hospital, source\_system, record\_type, phi\_rules\_applied

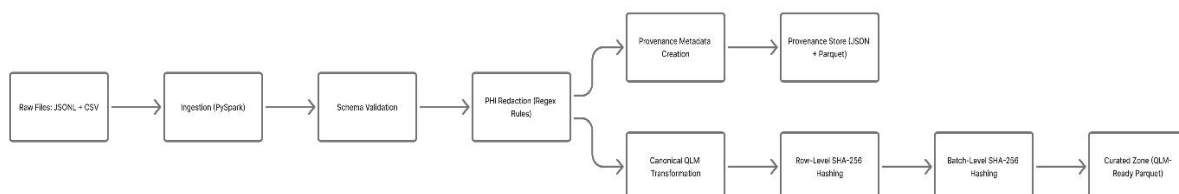
#### 5. Integrity Protection

- **Row-level SHA-256 hash** computed for each record.
- **Batch-level SHA-256 hash** computed by sorting all row hashes.

#### 6. Provenance & Audit Logs

Generated full metadata for each pipeline run: batch\_id, ingested timestamp, source files, row count, PHI rules applied, batch hash.

### Architecture Diagram:



## Task 2: Data Lake Architecture

### Objective

Design and demonstrate a compliant Data Lake architecture using Databricks and Delta Lake, supporting layered storage, governance, versioning, security, and audit capabilities.

### Layers Implemented

#### 1. RAW Zone

- Stores untouched data exactly as received. Stored in: `dbfs:/FileStore/task2_datalake/raw/`

#### 2. BRONZE Zone

Structured version of raw data. Schema applied. Written as Delta tables:

- `/bronze/notes`
- `/bronze/labs`

#### 3. SILVER Zone

- Cleaned + conformed datasets. PHI redacted (Task 1 logic reused).

#### 4. GOLD Zone (QLM-Ready)

- Combined notes + labs. Delta table stored under: `/gold/qlm_ready`

### Governance & Compliance Features

#### Delta Lake Versioning

- Every write creates a new table version.
- Time-travel with: `spark.read.format("delta").option("versionAsOf", X)`

#### RBAC (Role-Based Access Control)

Defined access matrix:

- Data Engineer → raw & bronze
- Analyst → silver
- Business / QLM → gold
- Auditor → provenance, gold

#### Encryption

- **In-transit:** TLS/HTTPS
- **At-rest:** Cloud-provider encryption (or CMK if enabled)

#### Audit Logging

Generated audit events containing: timestamp, user, action, target table, Saved under: `/audit/`

#### Provenance

A JSON record generated for each gold-layer run containing: data paths, row counts, dq failures, source files, batch ID

#### Architecture Diagram:

