TASK 1 - FDA-Grade Data Provenance ETL Pipeline

1.1 Summary

- Design an end-to-end ETL pipeline to:
- Ingest clinical text from multiple sources
- Validate schema at every stage
- Scrub PHI using formal rules
- Track full provenance (timestamps, rules, transformations)
- Output QLM-ready datasets
- Expose /provenance API for lineage retrieval

1.2 Inputs

- Raw clinical text files (CSV, JSON, HL7)
- Metadata describing each source (file name, schema version, hash)
- PHI redaction rules (names, dates, phone numbers, addresses)
- Input and output schema definitions (JSON or Parquet)

1.3 Expected Outputs

- QLM-ready dataset (cleaned, PHI redacted, hashed)
- Provenance logs (per record / per transformation)
- Integrity verification (SHA-256 hash per batch or record)
- /provenance REST API
- Developer and compliance documentation

1.4 Technical Implementation

Component	Azure Service / Tool	Notes
Raw & Curated Storage	Azure Data Lake Storage Gen2	Hierarchical namespace, encrypted, immutable
ETL Processing	Azure Databricks (PySpark notebooks)	Multi-source ingestion & transformation
Multi-source Ingestion	PySpark CSV/JSON reader + HL7 parser (hl7apy)	Handles large datasets
Schema Validation	jsonschema in Databricks	Validates input/output schema
PHI Redaction	Python regex / NLP rules	Names → <NAME_REDACTED>; Phones → <PHONE_REDACTED>
Provenance Logging	JSON logs in /audit/provenance/ (ADLS)	Append-only, immutable
Integrity Verification	SHA-256 hashes in /integrity/hashes.json	Ensures data integrity
REST API	FastAPI on Azure App Service / Functions	Exposes /provenance/{record_id}
RBAC	Azure RBAC	Enforces least-privilege access
Monitoring & Audit	Azure Monitor + WORM storage	Tracks access & modifications for compliance

1.5 Example ETL Code (Databricks / PySpark)

```
from pyspark.sql import SparkSession
import hashlib, re, json
from datetime import datetime
spark = SparkSession.builder.appName("FDA_ETL").getOrCreate()


def ingest_file(path, source_type):
    if source_type == "csv":
        df = spark.read.csv(path, header=True)
    elif source_type == "json":
```

```
        df = spark.read.json(path)
    elif source_type == "hl7":
        df = parse_hl7(path)  # Custom HL7 parser
    return df

def redact_phi(text):
    text = re.sub(r"\b\d{3}[-.]?\d{3}[-.]?\d{4}\b", "<PHONE_REDACTED>", text)
    text = re.sub(r"\b([A-Z][a-z]+ [A-Z][a-z]+)\b", "<NAME_REDACTED>", text)
    return text

def compute_hash(text):
    return hashlib.sha256(text.encode("utf-8")).hexdigest()
def log_provenance(record_id, step, input_hash, output_hash):
    entry = {
        "record_id": record_id,
        "timestamp": datetime.utcnow().isoformat(),
        "transformation": step,
        "input_hash": input_hash,
        "output_hash": output_hash
    }
    with open(f"/dbfs/mnt/datalake/audit/provenance/{record_id}.json", "w") as f:
        f.write(json.dumps(entry))
```

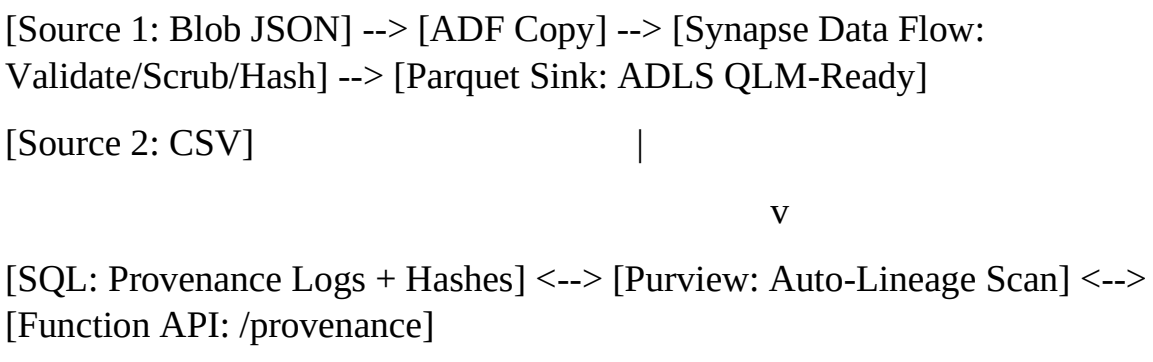
1.6 Provenance REST API

```
from fastapi import FastAPI
import json, os
app = FastAPI()
@app.get("/provenance/{record_id}")
def get_provenance(record_id: str):
    file_path = f"/mnt/datalake/audit/provenance/{record_id}.json"
    if os.path.exists(file_path):
        with open(file_path) as f:
            return json.load(f)
    return {"error": "Record not found"}
```

1.6.1 Sample /provenance API Output

```
{
  "record_id": "rec_001",
  "timestamp": "2025-02-01T12:44:21Z",
  "transformation": "PHI_redaction",
  "input_hash": "a89fd1d3b97d0ba9ac...",
  "output_hash": "19bd9f1170ffbac120..."
}
```

1.7 Lineage Architecture Diagram



1.8 Logical Architecture Diagram

Raw Input
↓
Schema Validation
↓
PHI Redaction
↓
Hashing & Integrity Checks
↓
Provenance Logging
↓
Delta Lake Writes
↓
QLM-Ready Output + /provenance API

1.9 Deployment & Configuration Guide

Databricks

1. Create service principal and mount ADLS Gen2 with OAuth2.
2. Upload ETL notebooks for ingestion, validation, PHI redaction, hashing.
3. Create Databricks Jobs for scheduled batch processing.
4. Enable cluster-level secure networking (no public IP).

App Service

1. Containerize the FastAPI app using Docker.
2. Deploy to Azure App Service or Azure Function App.
3. Use Managed Identity to access ADLS /audit/provenance files.
4. Apply Azure Front Door or APIM for secure access control.

Purview + Monitoring

1. Register ADLS, Databricks, App Service in Purview.
2. Enable diagnostic logs (Storage Analytics + Log Analytics).
3. Configure alerts for unusual access patterns.

TASK 2 - Tiered, Regulated Data Lake Architecture

2.1 Summary

Design a secure, multi-zone data lake:

Raw Zone – exact copy of data, immutable, encrypted

Curated Zone – standardized, partially PHI redacted, versioned

QLM-Ready Zone – fully PHI redacted, hashed, minimal normalized fields

2.2 Storage Zones & Azure Mapping

Zone	Description	Azure Implementation
Raw	Exact copy, immutable, encrypted	ADLS Gen2 + WORM
Curated	Standardized, validated, partial PHI removal	ADLS Gen2 + Delta Lake
QLM-Ready	Fully PHI redacted, hashed	ADLS Gen2 + Delta Lake

2.3 RBAC

Role	Raw Zone	Curated Zone	QLM-Ready Zone	Logs	API
Data Engineer	✓	✓	✓	✓	✓
Data Scientist	✗	✓	✓	✗	✗
Compliance Officer	✓	✓	✗	✓	✓
Analyst	✗	✗	✓	✗	✗

2.4 Versioning & Audit

Delta Lake tables on ADLS Gen2 (Curated & QLM zones)

Every write produces: commit logs, snapshots, time travel

Audit logs track: access, modifications, PHI redaction version, pipeline version

Logs stored in immutable WORM storage

2.4.1 Example Versioned Dataset

Describe History curated.clinical_notes;

Version	Timestamp	Operation
0	2025-02-01 10:00:00	CREATE TABLE
1	2025-02-01 11:30:12	MERGE (batch_20250201)
2	2025-02-02 09:10:44	WRITE (batch_20250202)

2.4.2 Time Travel Query Example

SELECT * FROM curated.clinical_notes VERSION AS OF 1**;**

2.4.3 Example Audit Logs

Access Log


```
{
  "user": "data_engineer",
  "action": "READ",
  "dataset": "curated.clinical_notes",
  "timestamp": "2025-02-03T16:11:20Z",
  "ip": "10.0.2.14"
}
```


Transformation Log


```
{
  "pipeline": "ETL_v3",
  "transformation": "PHI_redaction_v1",
  "timestamp": "2025-02-03T09:14:11Z",
  "records_processed": 14592
}
```

2.5 Compliance Mapping

Regulation	How Solution Meets It
HIPAA	PHI removal, RBAC, encryption
FDA	Full data lineage, integrity checks, reproducible pipelines
CFR Part 11	Timestamped transformations, immutable logs
Security	Encryption, RBAC, network segmentation

2.5.1 Expanded Compliance Mapping

HIPAA Compliance

- Full PHI removal in the QLM zone.
- RBAC ensures only authorized users access raw/curated zones.
- Encryption in transit (TLS1.2+) and at rest (Azure SSE).
- Access logs stored in immutable WORM storage.

FDA Requirements

- Complete record-level lineage via provenance logs.
- SHA-256 integrity hashing for tamper detection.
- Reproducible ETL via versioned notebooks and config files.
- Immutable WORM audit logs meet FDA documentation rules.

CFR Part 11

- Immutable audit trail via WORM + Delta commit logs.
- Timestamped entries for all transformations.
- Identity-bound changes tracked via Azure AD.
- Electronic signatures through RBAC-controlled access.
- No silent modification of data (Delta Lake ACID guarantees).

2.6 Deliverables

Architecture Diagrams

Physical (Text; PNG in repo):