Capstone Project Statement

Healthcare Claims Quality & Compliance Platform

Background

MediSure is a leading healthcare insurer processing claims from hospitals, clinics, and pharmacies nationwide. Data arrives from batch CSVs, streaming EHR events, JDBC legacy databases, and nested JSON provider directories.

Their current challenges include:

- Delays in claims processing and fraud detection (often >24h lag)
- Frequent data quality issues such as duplicates, invalid codes, and mismatched
- member/provider IDs
- Lack of rollback and audit capabilities to meet compliance
- Fragmented analytics and reporting processes
- Minimal data governance and lack of role-based data access control

MediSure aims to modernize its analytics environment by moving to a governed, real-time claims processing platform using Databricks Lakehouse.

Your Challenge

Design and implement an end-to-end healthcare claims analytics platform on Databricks that:

- Ingests batch and streaming claims, member master data, provider directories,
- and diagnosis reference data (mix of CSV, JSON, streaming, JDBC)
- Implements a Bronze/Silver/Gold architecture using Delta Lake and Delta Live
- Tables (batch + streaming)
- Ensures data quality with deduplication, primary/foreign key validation, null
- checks, and business rule enforcement
- Enables incremental, ACID-compliant processing with versioning, rollback, and
- time travel
- Supports fraud scoring, compliance reporting, and advanced aggregations
- Implements end-to-end pipeline orchestration with Databricks Jobs (task
- dependencies, error handling, retries, CRON scheduling)
- Applies data governance with Unity Catalog, role-based access control,
- catalogs, and megastores
- Provides notebook-based automation including modular code, notebook

- chaining, and Git integration via Repos
- Enables CI/CD workflows and collaboration
- Delivers auditable data lineage and operational monitoring

POC Design Brief

1. **Ingestion and Processing**
   - Sources include batch CSV claims, streaming JSON claims events, JDBC member data, nested JSON provider feeds, and diagnosis reference CSV.
   - Use Auto Loader, DLT pipelines, and JDBC connectors to ingest and stage data in Bronze tables.

2. **Data Quality, Security, and Governance**
   - Deduplicate and validate data on keys; enforce data types and handle
   - null/missing values.
   - Parse and normalize nested arrays and structs.
   - Secure PII with masking and RBAC using Unity Catalog.
   - Provide full audit trails using Delta history, versioning, and rollback features.

3. **Analytics and Automation**
   - Develop fraud risk scoring and anomaly detection using SQL UDFs or Python.
   - Automate end-to-end pipelines with Jobs including scheduling, retries, and alerting.

Step-by-Step Tasks & Expectations

Section 1: Databricks Lakehouse Platform & Collaboration

- Describe platform architecture (data plane vs. control plane; what resides in your cloud account)
- Spin up and compare all-purpose vs. jobs clusters; manage runtime versions and permissions
- Organize and share notebooks via Repos (integrate Git; demonstrate version control limits)
- Demonstrate multi-language notebook features (SQL, Python, Scala)
- Use notebook chaining for modular ETL
- Collaborate: share and review notebooks with teammates

Section 2: ELT with Apache Spark

- Ingest all data sources (CSV, JSON, JDBC, streaming)
- Create views, temp views, and CTEs for raw staging
- Deduplicate on ClaimID/MemberID/ProviderID; remove nulls and invalid records
- Validate and enforce primary and foreign keys (members, providers, claims)
- Data profiling: null checks, data type enforcement, pattern extraction

- Parse nested JSON arrays in providers and claims
- Join and enrich datasets; pivot claim data for reporting
- Use UDFs and CASE/WHEN for fraud scoring and validation

Section 3: Incremental Data Processing with Delta & DLT

- Convert all core tables to Delta
- Implement DLT pipelines (batch for historical claims, streaming for new claims)
- Demonstrate Delta Lake ACID features: versioning, time travel (simulate
- accidental deletes), rollback, data/metadata comparison, partitioning & Zordering, OPTIMIZE & VACUUM
- Implement MERGE for upserts and deduplication
- Use COPY INTO for initial historical data loads

Section 4: Production Pipelines & Automation

- Build Jobs pipelines with multiple tasks moving data from bronze to silver to gold
- Set predecessor tasks, retry policies, and CRON schedules
- Debug failed tasks and demonstrate retry and alerting via email
- Monitor task execution histories and logs

Section 5: Data Governance

- Set up Unity Catalog with correct schema/catalog structures
- Assign role-based access controls (RBAC)
- Compare catalog and metastore structures
- Audit access and lineage for sensitive PII fields (email, DOB)

Deliverables

- Modular notebooks with clear, commented code for each data layer (bronze,
- silver, gold) with explanatory cells
- DLT pipeline configs and batch/streaming execution results
- Job definitions with orchestration, error handling, retry, and alerting
- Governance demos: notebooks/screenshots showing RBAC and catalog
- structure
- Presentation slide deck summarizing solution overview, architecture, and demo results

Success is measured by your ability to:

- Deliver a reliable, automated Lakehouse pipeline supporting batch and streaming data.
- Ensure data quality, governance, and audit compliance.
- Provide real-time fraud analytics with alerts and scoring.
- Clearly document and present your implementation.

XX