**Case Study: Enterprise Data Governance & Ingestion Framework for Insurance Domain**

**Problem Statement**

The insurance firm lacked centralized data governance and ingestion orchestration across actuarial, claims, underwriting, and policy servicing units. Key challenges included:

* Poor visibility into data lineage and governance posture
* Ad-hoc data classification and lack of glossary alignment
* Complex RBAC enforcement across cloud and on-prem data
* Inconsistent ingestion patterns and manual data loads
* No retry, SLA, or failure resumption mechanism

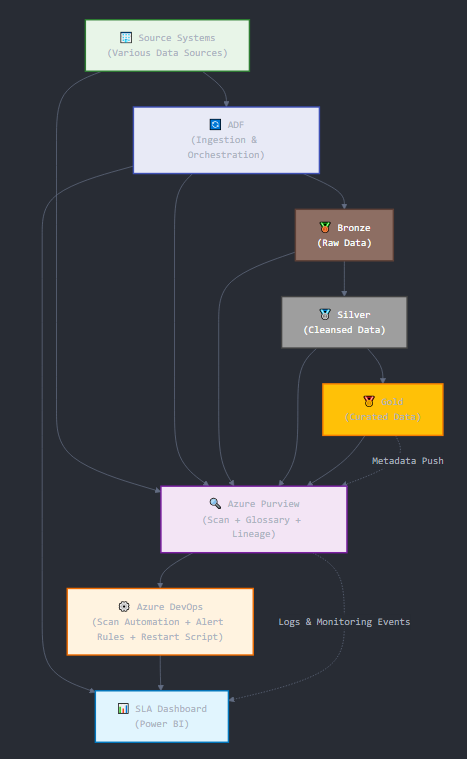
**Skill Tower Developed the Project**

* **Azure Data Engineering** (ADF, Databricks, Purview, Key Vault)
* **Cloud Security & Governance** (RBAC, Classification, DevOps)
* **Monitoring & Automation** (Alerting, Teams webhook, SLA dashboards)
* **CI/CD Automation** (Azure DevOps pipelines for scan automation and restart scripts)

**Use Case / Architecture Diagram**

**Key Layers:**

* Data Sources: Policy DB, Claim DB, Underwriting logs
* Azure Purview for governance
* Azure Data Factory for ingestion
* Azure Databricks for transformation & lineage
* Power BI for SLA dashboard



**User Stories**

1. As a **data engineer**, I need to ingest data from 10 insurance tables using configuration so that I can manage them uniformly.
2. As a **data steward**, I want to classify sensitive fields (SSN, DOB) using custom rules and glossary terms.
3. As a **governance lead**, I want to see end-to-end lineage in Purview from ingestion to Databricks output.
4. As a **DevOps engineer**, I want to automate scan trigger via pipeline to ensure consistency.
5. As a **support engineer**, I want to get Teams alerts for failures and view SLA dashboards.

**Expected Deliverables**

* Hierarchical Collection Setup with RBAC across domains (Actuarial, Claims, etc.)
* Custom scan configuration and glossary term tagging
* CI/CD pipeline to automate scan and restart ingestion jobs
* Modular ingestion pipeline with fault-tolerance, retries, SLA, and dynamic dataset
* ADF triggers (Schedule, Tumbling, Event Grid) with dependency chain
* Monitoring setup: SLA dashboard, alert rules, cost view, and webhook notifications

**Milestone & Duration**

| **Milestone** | **Duration** |
| --- | --- |
| Requirement Finalization & RBAC Setup | 2 Hour |
| Purview Collection, Scan, Glossary | 2 Hour |
| Ingestion Pipeline (10 Tables) + Dynamic Control Table | 3 Hour |
| Teams Alert, Monitor Hub, Retry Logic | 2 Hour |
| Lineage Push from Databricks to Purview | 1 hour |
| SLA Dashboard, Cost View, Test Plan | 2 Hour |
| DevOps Scan Automation + Restart Scripts | 1 Hour |
| Final Demo & Sign-off | 2 Hour |

**Implementation Notes**

* **RBAC**: Set up collection-based access in Purview per insurance unit.
* **Scan Configuration**: Use Key Vault-based credential for linked service auth; use Purview REST API to trigger scans.
* **Custom Classification**: Apply regex rules to tag SSNs, policy IDs, and dates of birth.
* **Lineage**: Use Databricks spark\_catalog integration and REST API push to update Purview lineage.
* **Retry & Resume**: Watermark-based resume, with checkpoint tracking in Delta logs.
* **SLA Alerting**: Use scheduled Power BI dashboard from Delta logs + ADF alert rules + Teams webhook.
* **Control Table**: SQL-based table to manage dynamic dataset references, status, last watermark, etc.

**Evaluation Rubrics**

| **Area** | **Criteria** |
| --- | --- |
| Governance | Glossary terms mapped, RBAC implemented, scans automated |
| Ingestion | 10+ tables ingested with watermark, fault-tolerance, resumption |
| Observability | SLA dashboard, retry logs, Teams integration, cost monitoring |
| Reusability | Flowlet use, modular dynamic pipeline, CI/CD pipelines defined |
| Lineage | Purview-to-Databricks lineage working and visible in UI |