Proposal: Comprehensive Data Quality Framework Using Great Expectations

# 1. Objective

To design and implement a robust, scalable, and automated Data Quality Framework leveraging Great Expectations (GX). This framework will ensure data accuracy, completeness, consistency, and integrity across all critical pipelines and datasets, enhancing trust in data-driven decision-making.

# 2. Goals

* Establish a reusable and extensible data validation framework.
* Automate profiling, rule-based validation, and exception handling.
* Integrate with existing data pipelines (e.g., Airflow, DBT, Spark, SQL).
* Generate human-readable, version-controlled validation reports.
* Enable CI/CD and GitOps practices for data quality.

# 3. Key Features

* Automated Data Profiling: Initial column stats and schema discovery.
* Rule-driven Validation: Define expectations in JSON/YAML or Python.
* Suite Management: Dynamic expectation suite creation and reuse.
* Multi-source Compatibility: Files (CSV, Parquet), Databases (PostgreSQL, SQL Server), Big Data (Spark).
* Execution Modes: Batch, real-time, and pipeline-integrated validations.
* Reporting: HTML validation reports, Slack/email alerts, and JSON summaries.
* Logging & Auditability: Centralized logging and failure traceability.
* Test Data Simulation: Use synthetic data for test case generation.

# 4. Technical Stack

|  |  |
| --- | --- |
| Component | Technology |
| Data Validation | Great Expectations |
| Workflow Orchestration | Apache Airflow, Prefect |
| Data Transformation | DBT, Spark SQL |
| Data Sources | PostgreSQL, SQL Server, S3, APIs |
| Storage & Logs | S3, PostgreSQL, Cloud Logging |
| CI/CD Integration | GitHub Actions, Jenkins, GitLab |

# 5. Architecture Overview

An illustrative architecture for the proposed data quality framework:

+-----------------+  
 | Data Sources |  
 +-----------------+  
 |  
 +-----------------------------+  
 | Data Ingestion Pipelines |  
 +-----------------------------+  
 |  
 +-------------------+  
 | GX Validation Layer|  
 | (Dynamic Suites) |  
 +-------------------+  
 / | \  
 Profiling Engine Validation Reporting  
 Engine Layer  
 | |  
 Alerting | Audit Logs  
 (Slack/Email) & Dashboards

# 6. Implementation Plan

|  |  |  |
| --- | --- | --- |
| Phase | Description | Deliverables |
| Phase 1 | Assessment & Tooling | Requirement doc, GX setup |
| Phase 2 | Data Profiling | Profiling scripts, initial reports |
| Phase 3 | Rule Engine Setup | Expectation suites, config loaders |
| Phase 4 | Integration & Automation | Pipeline hooks, CI/CD |
| Phase 5 | Reporting & Alerts | Dashboards, alert flows |
| Phase 6 | Testing & Handoff | Unit tests, knowledge transfer |

# 7. Success Metrics

* 95%+ validation coverage for critical datasets.
* 100% automated data quality checks in pipelines.
* <10% manual intervention in issue identification.
* Real-time alerts with <5 minutes detection latency.

# 8. Team & Roles

|  |  |
| --- | --- |
| Role | Responsibility |
| Data Engineer | Framework development & integration |
| QA Analyst | Validation rule creation & testing |
| Data Architect | Data source alignment & strategy |
| DevOps Engineer | CI/CD, logging, and alerting setup |

# 9. Risks & Mitigation

|  |  |
| --- | --- |
| Risk | Mitigation |
| High data volumes | Use Spark backend, batch validation |
| Changing schemas | Use dynamic profiling + alerts |
| Tool complexity | Training & documentation sessions |

# 10. Appendices

• Sample Expectation Suite in JSON/YAML

• GX Directory Structure (Data Docs, Expectations, etc.)

• CI/CD Pipeline Design for Data Quality Deployment