Automation of Data Acquisition and Transformation Pipeline

Data Science and Automation Project

Project Overview

- Developed and automated modular data pipelines for public-sector finance datasets.
- Designed to enhance data accessibility, integrity, and transformation efficiency.
- Built using Python, R, and modern CI/CD workflows.
- Focused on improving data reliability, reducing manual intervention, and enabling reproducible analyses.

Tech Stack

- **Programming:** Python (VS Code), R
- **Version Control:** Git (GitLab)
- **Deployment:** Jenkins (CI/CD Pipeline), Artifactory (Package Management)
- **UI:** Streamlit
- **Big Data:** HDFS, Hive, Spark / MapReduce
- **ETL Workflow Management:** Apache NiFi
- **Analysis Collaboration:** CDSW (Collaborative Data Science Workspace)
- **Documentation:** Confluence

Data Pipeline Architecture

- **ETL Workflow:** Python for extraction and transformation → Git → Jenkins (automated deployment)
- **Data Loading:** Managed through NiFi into HDFS → Spark → Hive → SQL layer
- **Storage and Querying:**
  - HDFS for distributed data storage
  - Hive for structured data querying and schema management
- **Processing and Analysis:** CDSW and PySpark environments for advanced analytics
- **Deployment:** Dockerised applications and Python packages stored in Artifactory

Data Acquisition

Techniques Implemented:

- **Static Web Scraping:** BeautifulSoup for parsing HTML and XML content.
- **API Integration:** Direct data access via secure APIs.
- **Regex (re library):** Pattern matching for URLs, dates, and data identifiers.
- **Requests Library:** For HTTP GET requests and data retrieval.

Best Practices:

- Proxy usage for secure web access.
- Error handling and response validation.
- Comprehensive unit testing (unittest, pytest, responses, mock, pytest-mock).

Data Transformation

- **Python:** Data cleaning, standardisation, renaming, restructuring, and transfers (Pandas, OS, RegEx).
- **PySpark:** Large-scale transformations, aggregations, and filtering for parallel processing.
- **Hive:** SQL-based structured transformations and table creation in HDFS.
- **Outcome:** Clean, standardised, and query-ready datasets for downstream analytics.

Testing and Quality Assurance

Unit Testing:

- **unittest:** Python's built-in testing framework.
- **pytest:** Fixtures, parametrised tests, and enhanced assertions.
- **mock / pytest-mock:** Simulated responses for API calls and error scenarios.

Testing Coverage:

- Successful HTTP responses (200) and error handling (404, 500).
- Correct extraction of elements from HTML and validation of Regex patterns.
- Logging and exception testing (ValueError, TypeError, FileNotFoundError, TimeoutError).
- Edge case testing (empty inputs, nulls, boundary values).

Deployment and User Interface

- **Streamlit UI:** Enabled users to select datasets, years, and sources through an intuitive interface.
- **CI/CD:** Jenkins automated testing, integration, and deployment pipelines.
- **Best Practices:**
  - Modular programming and version control.
  - requirements.txt and setup.py for reproducibility.
  - Docker containers for isolated environments.
- **End-to-End Flow:**
  - Web scraping $\rightarrow$ Validation $\rightarrow$ Transformation $\rightarrow$ HDFS storage $\rightarrow$ Hive queries $\rightarrow$ PySpark analysis.

Impact and Key Learnings

- Improved data accessibility and reduced manual intervention across finance datasets.
- Ensured consistent and secure data processing through automation and error handling.
- Strengthened collaborative coding practices through GitLab, Jenkins, and Confluence.
- Built a Streamlit prototype to make data access more user-friendly for non-technical users.
- Gained deeper insight into software engineering practices such as CI/CD, OOP, and code modularity.

Reflection

- This project was a significant step forward in combining statistical programming with software engineering practices.
- It deepened my understanding of scalable data systems, testing frameworks, and collaborative workflows.
- The experience reinforced my passion for building reliable, accessible, and ethical data-driven products.