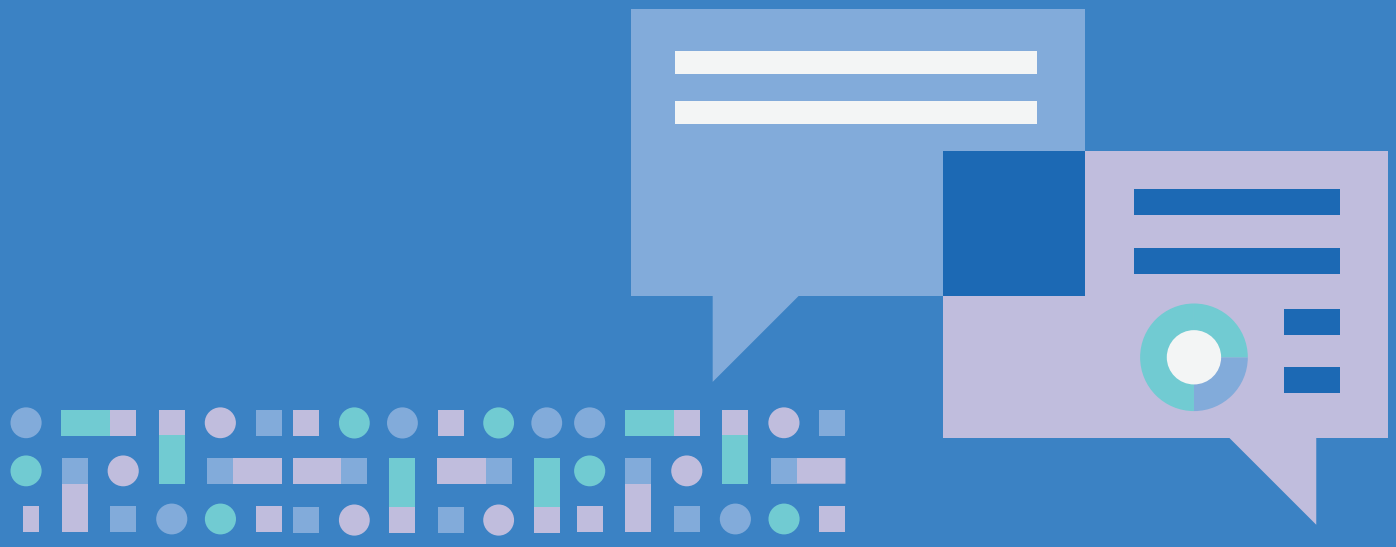
Computomic

Databricks Genie - Assisted Migration Framework

Contributors to the development of the Framework & Whitepaper:
Mahanvita Kamisetty, Shivarama Krishna, Nagendra Beerangi, Narendra Bengeri, Vamsee Mallireddy, Adarsh Vatsa, Ruma Arabatti and Rajiv Ramanjani



Abstract

This white paper presents a modern, AI-assisted approach to migrating enterprise data products from Amazon Redshift to Databricks using a declarative, YAML-driven transformation framework powered by Databricks Genie. By abstracting complex SQL and pipeline logic into reusable configuration patterns, the framework enables rapid, scalable conversion of legacy stored procedures into Databricks-native pipelines with minimal manual coding. Genie further accelerates the process by analyzing legacy logic, generating transformation artifacts, and supporting validation—reducing migration effort by up to 90%.

The result is a repeatable, ingestion-first modernization model that simplifies pipeline creation, enforces standardization, and enables teams to deliver governed, production-ready data products on Databricks with speed and confidence.

Executive Summary

In this whitepaper, we describe a migration approach that uses Databricks Genie Code Assistant and a YAML-driven transformation framework to accelerate the conversion of legacy Amazon Redshift stored procedures into Databricks-native pipeline assets. The approach centers on a generic transformation framework that reads YAML configuration, generates SQL and pipeline artifacts, and creates scheduled jobs with limited manual coding effort.

Business Context

In this initiative, we describe a migration initiative focused on moving data products from Redshift to Databricks while reducing the amount of hand-written SQL, PySpark, and platform-specific implementation logic required from developers The presenters emphasize that the framework abstracts complexity so teams can work primarily through configuration files, while Genie helps analyze code, generate new assets, and support validation activities.

Architecture Pattern

We use a declarative architecture in which users provide structured YAML input and operational configuration, after which the framework generates the SQL statements, attaches them to Databricks lakeflow declarative pipeline assets, and creates the related jobs This design shifts the development model from handcrafted code toward metadata-driven transformation generation, which improves repeatability and makes migrations easier to scale across many stored procedures and data products.

Core Framework Design

We designed the transformation framework as a generic engine capable of generating pipeline definitions and jobs by reading YAML inputs that define the desired target behavior. A user supplies pipeline metadata, SQL source file naming, cluster and scheduling information, and transformation definitions, and the framework converts those inputs into executable Databricks artifacts.

Supported Transformation Modes

Our framework supports four transformation modes: simple table transforms, joins, unions, and SQL override. The simple table transform mode handles lightweight projection and filtering, joins handle multi-table integration with explicit join conditions, union supports combining tables or static records with multiple union options, and SQL override enables direct raw SQL submission for transformations that exceed the expressive limits of the structured modes.

Mode	Purpose	Typical Use
Simple	Minimal transformation	Select columns, rename columns, apply filters
Join	Multi-source combination	Join source tables with configurable conditions and join types
Union	Consolidation	Union tables or static records using supported union strategies
SQL Override	Advanced logic	Subqueries or custom SQL patterns such as calendar generation logic

YAML-Driven Development Model

We interact with the framework primarily through YAML files rather than directly writing all runtime SQL and orchestration code. This allows a developer or analyst to focus on describing source tables, target tables, join behavior, union behavior, and pipeline metadata, while the framework takes responsibility for converting those declarations into runnable assets.

Genie-Assisted Framework Engineering

We first built a skeleton template for the transformation framework and then used Genie to enhance it iteratively Genie was able to inspect the repository path, read the files in context, summarize the framework, and assist with new conditions, **new use cases such as adding handler for CTEs or common table expressions, subqueries,** and optimization work without requiring the developer to manually trace every code path.

Workspace Context and Code Understanding

A notable capability we rely on is that Genie operates with workspace context, including awareness of repository files, schemas, Unity Catalog assets, and related workspace artifacts. Due of that context, Genie can produce a framework summary, interpret repository structure, and suggest relevant improvements much faster than a manual walkthrough of the same codebase.

Reverse Engineering Redshift Logic

One of the most valuable patterns we use is reverse engineering legacy Redshift stored procedures into YAML configurations that the new framework can understand. We provided Genie a stored procedure plus a carefully structured prompt, after which Genie analyzes logic against the existing transformation framework and generates the required YAML configuration file.

Generated Asset Quality

We find the generated YAML output to be structurally consistent with the team's manually created files, including the expected pipeline name, SQL file name, source and target metadata, and transformation logic such as filters and distinct handling The presenters estimate that Genie delivers roughly 90 to 95 percent of the needed functional output, while final review remains necessary for environment-specific elements such as catalog names, schema names, and some table references.

Human Review and Governance

Although Genie significantly reduces effort, we still include human review to ensure that environment-specific mappings are correct and that generated logic matches operational expectations The transcript makes clear that developers are expected to validate catalog alignment, schema mapping, and source-target correctness before production use, which positions Genie as an accelerator rather than a fully autonomous migration engine.

Development Lifecycle Acceleration

We consistently see Genie to compress tasks that would otherwise take hours or even a day into minutes or seconds. The acceleration appears in several lifecycle stages: understanding an existing framework, generating new YAML assets from stored procedures, enhancing framework logic, and preparing for validation and testing workflows.

Accessibility for Generalists

An important outcome of our approach is that it lowers the technical barrier to working effectively in Databricks The Genie led acceleration needs minimal expertise in Databricks, SQL, or PySpark to participate in the process, because most of the work is expressed in configuration and prompt-driven refinement rather than handwritten implementation code.

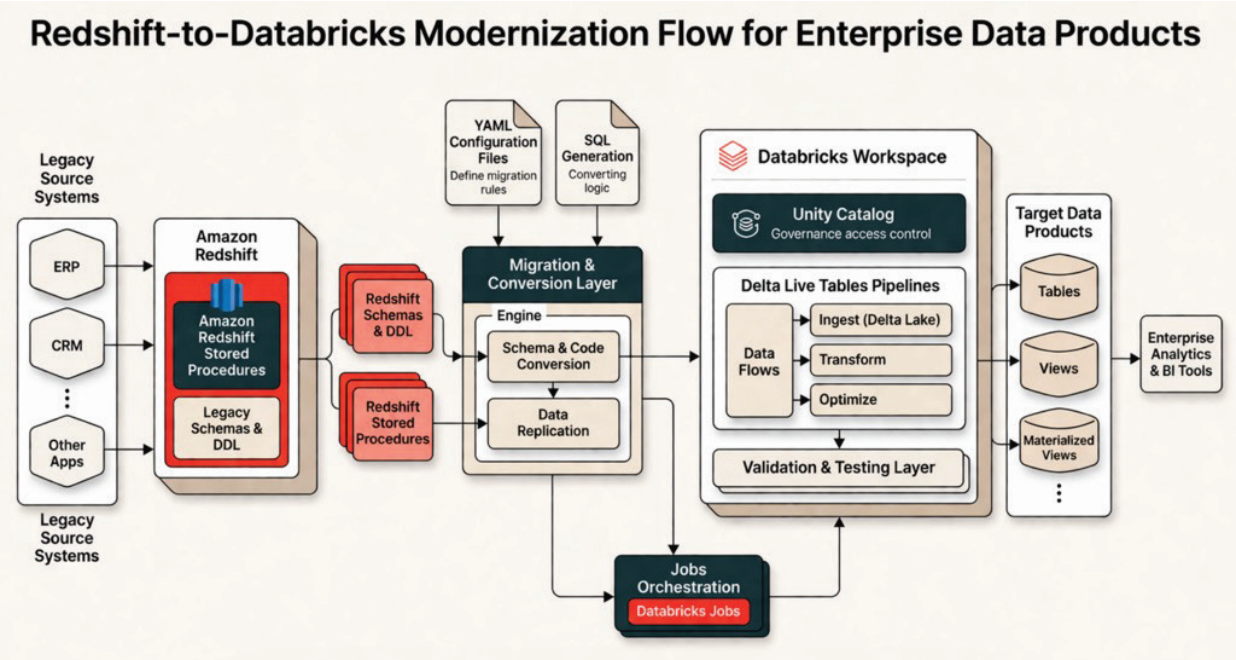
Validation and Testing Direction

We have also used Genie for validation and testing assistance, such as creating unit test cases, data comparison between source and targets to support table and view validation, as well as framework enhancements, which indicates the broader ambition of using AI support beyond generation into verification and quality control.

Design Implications

Our architecture reflects a scalable modernization pattern: convert procedural logic into declarative metadata, centralize transformation behavior in a reusable framework, and use AI assistance to bridge legacy logic into the new model This reduces repetition, encourages standardization, and can improve migration throughput when many Redshift stored procedures must be translated into Databricks-native implementations.

Architecture



Practical Takeaways

Based on our experience, teams can adopt a phased approach: build a metadata-driven transformation framework, create a small set of canonical YAML patterns, use Genie to analyze and extend the framework, and then use prompt-guided conversion of legacy stored procedures into framework-compatible configurations The accuracy improves when prompts include target catalog, schema, and reference examples, which means prompt engineering and pattern libraries should be treated as first-class migration assets.

Closing Perspective

We view Genie not as a replacement for engineering judgment, but as a productivity layer around a well-designed declarative migration framework. In that model, the combination of reusable architecture, guided AI generation, and targeted human review creates a practical path for modernizing Redshift-based data products into Databricks pipelines with less manual effort and more standardization.

Computomic

www.computomic.ai