



SAS DECOMMISSIONING & GENAI CODE TRANSLATION PROGRAM

Modernizing Analytics at Scale with GenAI, Guardrails and Enterprise Governance

PART OF MY ANALYTICS JOURNEY



~500
SAS users across the bank



~200
Users under my program



3 Entities
Emirates Islamic,
KSA ENBD, Egypt ENBD



2026
Program Execution



Mission
Decommission SAS, modernize analytics and unlock AI-ready data science platform

THE CHALLENGE

Legacy SAS at scale is costly, siloed and hard to maintain

~500 users with decades of SAS code, operational dependencies and inconsistent practices limit agility and innovation.

THE SOLUTION

GenAI-powered code translation with guardrails, governance and a phased rollout.



Discover & Assess

Inventory code, assess dependencies and prioritize



Translate with GenAI

AI-assisted translation (Python/PySpark) with human review



Guardrails & Safety

Automated checks, security scans and coding standards



Validate & Reconcile

Functional testing, performance tuning and business sign-off



Deploy & Adopt

Production rollout, training and change management

KEY BENEFITS



Lower Cost & License Dependency
Reduce vendor costs and infrastructure maintenance



Modern, Scalable, AI Ready
Open, future-proof tech stack (Python/PySpark)



Standardized, Secure & Compliant
Built-in governance, data security and quality controls



Faster Innovation
Enable advanced analytics and GenAI use cases



Knowledge Retention
Modern, maintainable and well-documented code

ENABLING TECHNOLOGIES



Python
Core language for translation



PySpark
Scalable data processing



Databricks
Unified data & analytics platform



GitHub
Version control and collaboration



CI/CD
Automated testing and deployment

STAKEHOLDERS



Business Users



Data & Analytics Teams



IT & Platform Teams



Risk & Compliance

BUSINESS IMPACT



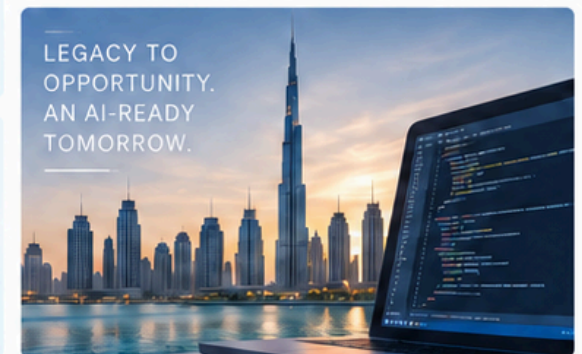
Lower Cost



Higher Agility



Foundation for Advanced Analytics & GenAI



LEGACY TO OPPORTUNITY.
AN AI-READY TOMORROW.



From legacy SAS to an AI-ready future.
Driving business value through GenAI-powered transformation.

PEOPLE

TECHNOLOGY

GOVERNANCE

IMPACT

1. Business problem and solution

The 2026 program addresses more than technology obsolescence. The legacy SAS setup had accumulated years of analytical code and business logic, but it also operated with comparatively weak centralized **data-security, access-control, code-governance and auditability**. A migration therefore had to preserve business logic while improving control, traceability and engineering discipline.

Core program dimensions

Dimension	2026 scope / approach
Enterprise estate	~500 SAS users targeted for migration to the modern data-science platform.
My program scope	~200 users across Emirates Islamic, ENBD KSA and ENBD Egypt.
Migration accelerator	Claude-based LLM utility for SAS-to-Python / PySpark translation.
Control objective	Move from fragmented desktop-style analytics toward governed access, controlled data use, versioned code and auditable deployment.
My contribution	Program management, LLM evaluation and selection, guardrail design, technology decisions and controlled adoption.

2. Controlled translation workflow

Stage	What happens	Primary control
Inventory	Identify programs, macros, dependencies, input/output datasets and business criticality.	Migration inventory, ownership and data classification.
Translate	LLM converts SAS constructs into Python / PySpark patterns with readable structure and comments.	Approved model, prompt template and coding standards.
Inspect	Generated code is checked before execution for risky imports, system calls, secrets and unintended data movement.	Static code guardrails, allow/deny rules and data-security checks.
Execute & test	Code runs only in a controlled test environment against approved datasets.	Sandbox, approved identities, resource limits and test harness.
Reconcile	Target output is compared with SAS baseline at schema, row, aggregate and business-rule level.	Parity thresholds and exception reporting.
Human review	Developer / data scientist reviews edge cases, maintainability and business logic.	Four-eyes approval / SME sign-off.
Version & migrate	Accepted code is committed and promoted through the platform lifecycle.	GitHub, release controls, audit trail and rollback.

The key principle is simple: **generated code is untrusted until it passes deterministic controls, data-security checks, test execution and human review.**

Guardrail	Examples of what is controlled	Why it matters
Library allow-list	Permit approved analytics packages; flag unknown or unnecessary dependencies.	Reduces supply-chain, licensing and runtime risk.
Dangerous system access	Flag or block subprocess, os.system, shell invocation, dynamic exec/eval and arbitrary command execution.	Prevents generated code from escaping the analytical task.
File-system controls	Restrict destructive writes, recursive deletes and access outside approved working locations.	Protects platform and data assets.
Network controls	Block arbitrary HTTP calls, sockets, FTP/SSH clients and unapproved external endpoints.	Prevents data exfiltration and uncontrolled dependencies.
Secrets & credentials	Detect hard-coded passwords, tokens, keys, connection strings and sensitive configuration.	Avoids credential leakage into code or repositories.
Database controls	Use approved connectors and identities; prohibit generated production DDL/DML unless explicitly authorized.	Separates analytics translation from production-change authority.
Resource controls	Limit memory, CPU, Spark partitions, loops and driver-side data collection.	Reduces runaway jobs and cluster instability.
Data security & access	Approved datasets, identities and purposes only; prevent unintended PII export, logging or sampling.	Strengthens confidentiality, least-privilege access and auditability versus the legacy setup.

3. Validation: proving semantic equivalence

Layer	Illustrative checks
Code	Syntax, imports, undefined variables, linting and structure checks.
Data schema	Column names, types, nullability, keys and row counts.
Transformation	Filters, joins, sorts, aggregations, dates, formats and missing-value behavior.
Numerical parity	Totals, distributions, tolerances, rounding and floating-point differences.
Business logic	Segment definitions, eligibility rules, flags, scores and downstream outputs.
Regression & performance	Golden test cases, source-vs-target comparison, runtime, memory and Spark scalability.
Approval	Exceptions documented and human reviewer signs off before migration.

4. LLM selection and governance

Dimension	What to evaluate
Translation fidelity	SAS DATA step, PROC SQL, macros, formats, dates, merges, retained variables and edge cases.
Python / PySpark quality	Correctness, readability, modularity and suitability for the target platform.
Security & privacy	Enterprise data handling, approved hosting/access pattern and no uncontrolled data exposure.
Governability	Prompt/model versioning, logging, traceability and evaluation support.
Economics & usability	Cost, latency, consistency, explanations and handling of unsupported constructs.

5. Modern engineering capability created

Capability	What changes
Python / PySpark	Open programming ecosystem and distributed processing for larger workloads.
Kedro framework	Modular, reproducible pipelines; catalogued datasets; parameterized components and clearer lineage.
GitHub / version control	Committed, reviewable and rollback-capable code instead of opaque local copies.
Testing	Unit, regression and data-quality tests formalize legacy business logic.
MLOps / LLMOps	Model, prompt, code and evaluation versions can be governed through a repeatable lifecycle.
Platform & data governance	Common environments, approved libraries, controlled identities, access policies, logging and audit trails.

6. My role and leadership narrative

I program-manage roughly **200 users across Emirates Islamic, KSA and Egypt** within a bank-wide population of roughly 500 SAS users. The role combines business migration, cross-country coordination, target-platform understanding, LLM selection and the design of guardrails that allow GenAI to accelerate delivery without transferring accountability to the model.

This project marks a further step in my technical progression: from analytics and machine learning, to Hadoop/PySpark and distributed data processing, to Kedro and GitHub-based engineering discipline, and now to **governing LLMs inside an enterprise transformation**. The migration is also a control modernization: stronger data security, access governance, traceability and reproducibility are designed into the target state.

What this case study demonstrates

- Enterprise scale - modernization affecting hundreds of analytical users.
- Cross-border leadership - direct program responsibility across three entities.
- Production-oriented GenAI - LLMs applied to a concrete software-modernization problem.
- AI governance - model selection, code guardrails, testing, human oversight and traceability.
- Technical progression - Python/PySpark, Kedro, GitHub, distributed computing and MLOps/LLMOps.
- Control modernization - improved data security, access control, auditability and governed deployment.