Paper SA 16

LLM Powered Clinical Data Flow with Traceability - From Siloed Pipelines to a Semantic Fabric

Pankaj Attri, SAS Institute Inc., Cary, NC, US

Matt Becker, SAS Institute Inc., Cary, NC, US

ABSTRACT

Clinical trial data submissions for FDA review are often hampered by complex, manual processes. We present a LLM-powered solutions that can help automate the clinical data pipeline and, importantly, establish end-to-end traceability. The solution automates the conversion of raw source data into standardized SDTM, generates analysis-ready ADaM datasets, and creates final TLFs for submission, along with the associated code (used to create datasets and analytical outputs). The other important deliverable of the solution is the lineage graph view. By interpreting the Statistical Analysis Plan, the system attempts to create a direct, visual link between SAP requirements and the datasets that fulfill the requirements. This automates traceability to the study design, providing verifiable proof that protocol and SAP requirements have been met, ensuring compliance and confidence for regulatory submissions.

INTRODUCTION

Drug development has entered an era where data volume, velocity, and variety have outpaced traditional clinical programming practices. Trials are larger and more complex, often adaptive, decentralized, and biomarker-rich, integrating ePRO, wearables, imaging, and real-world data alongside conventional CRF-based sources. At the same time, regulatory expectations for standardization, traceability, and reproducibility continue to tighten. The result is a widening gap between what manual, craft-style programming can deliver and what modern development timelines require.

Clinical programmers are central to bridging this gap, but too much of their time is consumed by repetitive tasks - hand-coding SDTM and ADaM datasets, updating derivations for each data refresh, etc. These activities, while necessary, are inherently mechanical and scale poorly. They leave insufficient bandwidth for higher-order responsibilities.

AI-assisted automation is the pragmatic response. It offers a way to shift effort from repeated keystrokes to reproducible, metadata-driven pipelines that can be rapidly adapted. Automation frameworks can ingest mapping specifications, generate code, and keep data and metadata synchronized throughout the lifecycle. This does not eliminate the need for expert programmers; it elevates their role. With foundational tasks accelerated, programmers can focus on the nuanced elements that determine regulatory success: edge-case handling, efficacy datasets, and transparent traceability.

In short, automation augmented by AI enables clinical programming teams to scale with the complexity of today’s trials, reduce avoidable rework, and reallocate effort toward strategic, science-driven tasks that improve decision-making and shorten time-to-submission.

COMMON CHALLENGES IN CLINICAL PROGRAMMING WORKFLOWS

Despite mature standards and experienced teams, traditional clinical programming workflows struggle under modern pressures. The most pervasive pain point is the “double programming” model used for

quality control. While effective at catching defects, this approach exacts a significant resource toll. The opportunity cost is substantial, time spent aligning code could be redirected to scientific oversight, cross-study consistency, and early risk detection.

End-to-end traceability is another critical challenge. Regulators expect clear, defensible lineage from CRF items and external sources through SDTM and ADaM to the final analyses.

These challenges are not merely inconveniences; they are structural inefficiencies that delay timelines and increase regulatory risk. Double programming and manual documentation can ensure correctness at a point in time, but they do not scale across studies, refreshes, and submissions. Achieving both speed and assurance requires workflows where specifications, code, and metadata are synchronized by design, and where routine tasks are automated, so human expertise is reserved for the complex, judgment-intensive aspects of clinical data science.

AI AGENTS AND LLMS: IMPACT ON CLINICAL PROGRAMMING

Generative AI technologies that use Large Language Models (LLMs) can help address some of these challenges. LLMs are foundation models trained on vast corpora of text and code that can understand natural language, generate structured outputs, and produce executable code from textual specifications. Agentic AI systems build on LLMs by adding the ability to plan, reason, and act across a sequence of steps, invoking tools (APIs, databases, version control, validation harnesses) and using memory to maintain context and state. Together, they enable “workflow intelligence” rather than isolated prompt-and-response interactions.

For clinical programmers, this matters now because data volumes and study complexity have outpaced manual methods, while the maturity of LLMs, enterprise-grade orchestration, and guardrails has reached a practical threshold. Agentic systems can ingest source artifacts - protocols, CRFs, SAPs, CDISC controlled terminology, and synthesize mapping specifications, derivation logic, and code drafts that are consistent across domains and aligned to standards.

Concrete capabilities include:

- Automating generation of SDTM from Raw data sources.

- Drafting ADaM derivations (e.g., ADSL, efficacy datasets) from the SAP and TLF shells, including traceable links to statistical methods and study endpoints.

- Producing initial TLF shells and code that reflect SAP specifications.

Agentic AI can coordinate multi-step workflows: read the SAP to identify endpoints, query the metadata repository for SDTM variables, propose ADaM derivations, generate code etc. With retrieval-augmented generation, it grounds outputs in organization-specific code repositories/pre-defined macro libraries, reducing hallucination and increasing reproducibility.

Critically, these systems are designed for human-in-the-loop oversight. The AI accelerates the high-volume, repetitive foundations while surfacing assumptions and uncertainties for expert review. Programmers retain decision authority, using AI outputs as robust starting points, focusing their time on complex analyses, edge cases, and scientific interpretation.

MINIMAL VIABLE PROTOTYPE (MVP)

To automate clinical data transformation steps discussed in the previous sections a minimal viable prototype was created. The prototype allows programmers to use LLMs to auto generate data transformation mappings, code, and lineage.

Figure 1 shows a snapshot of the MVP during new project creation. In this step, the user enters basic project details and specifies folder locations where Raw, SDTM, and ADaM datasets, along with any statistical outputs, will be generated. The Clinical Data Flow is organized into optional modules: Data Standardization (Raw to SDTM), Analysis Dataset Generation (SDTM to ADaM), and Reporting & Visualization (ADaM to TLF). In the LLM specifications, the user selects the LLMs to be used.

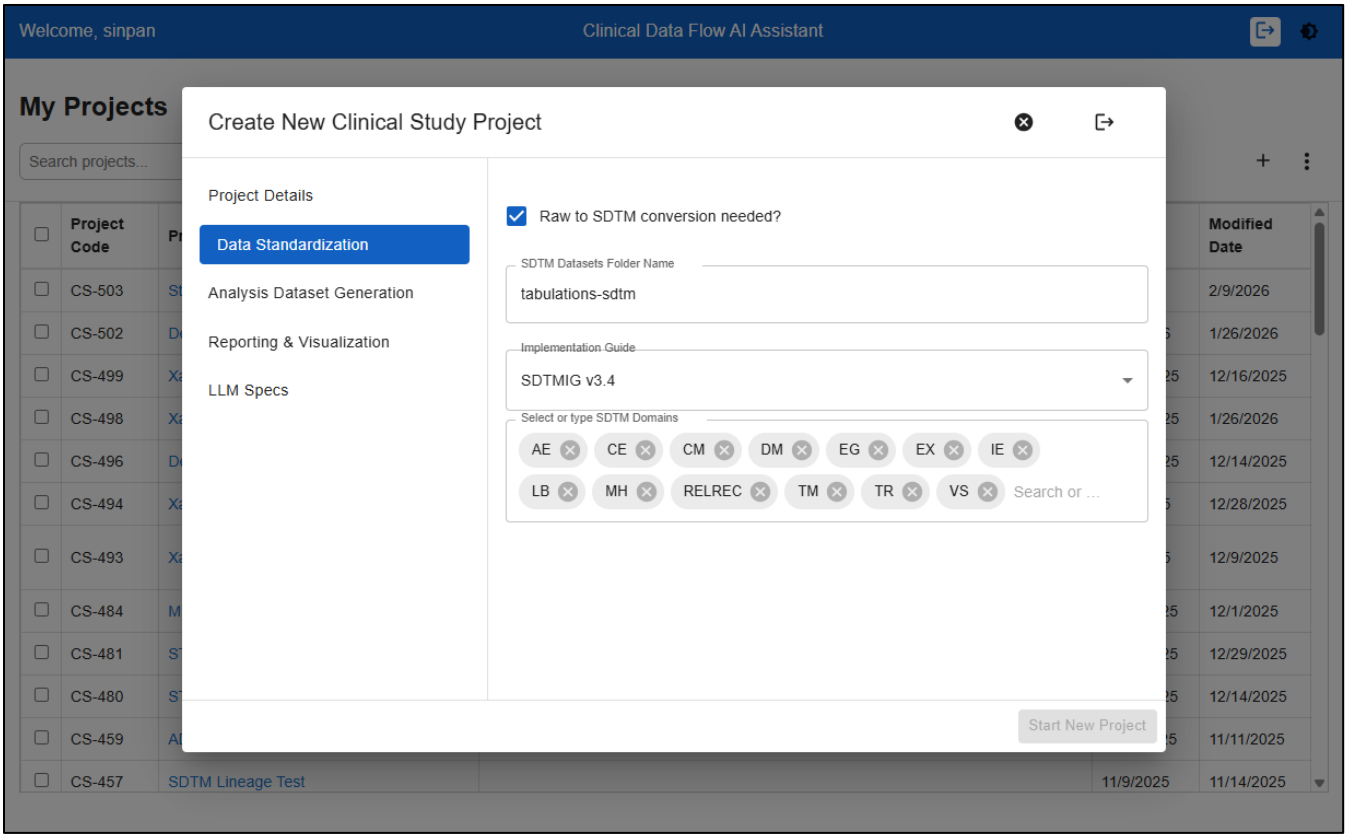


Figure 1: Create Study Project

After the project is created, the user can generate mapping rules. Figure 2 shows the project details screen, where the left pane lists the steps selected during setup and the right pane offers multiple tabs populated with information generated by the LLMs as each step is completed. The SDTM Datasets and SDTM Code tabs present the Raw-to-SDTM mapping transformations and associated code. The ADaM Datasets and ADaM Code tabs support the SDTM-to-ADaM transformation. The TLF Logic and TLF Code tabs cover the ADaM-to-TLF step; the logic tab displays English descriptions extracted from TLF shell documents, and the code tab provides generated code for each TLF item.

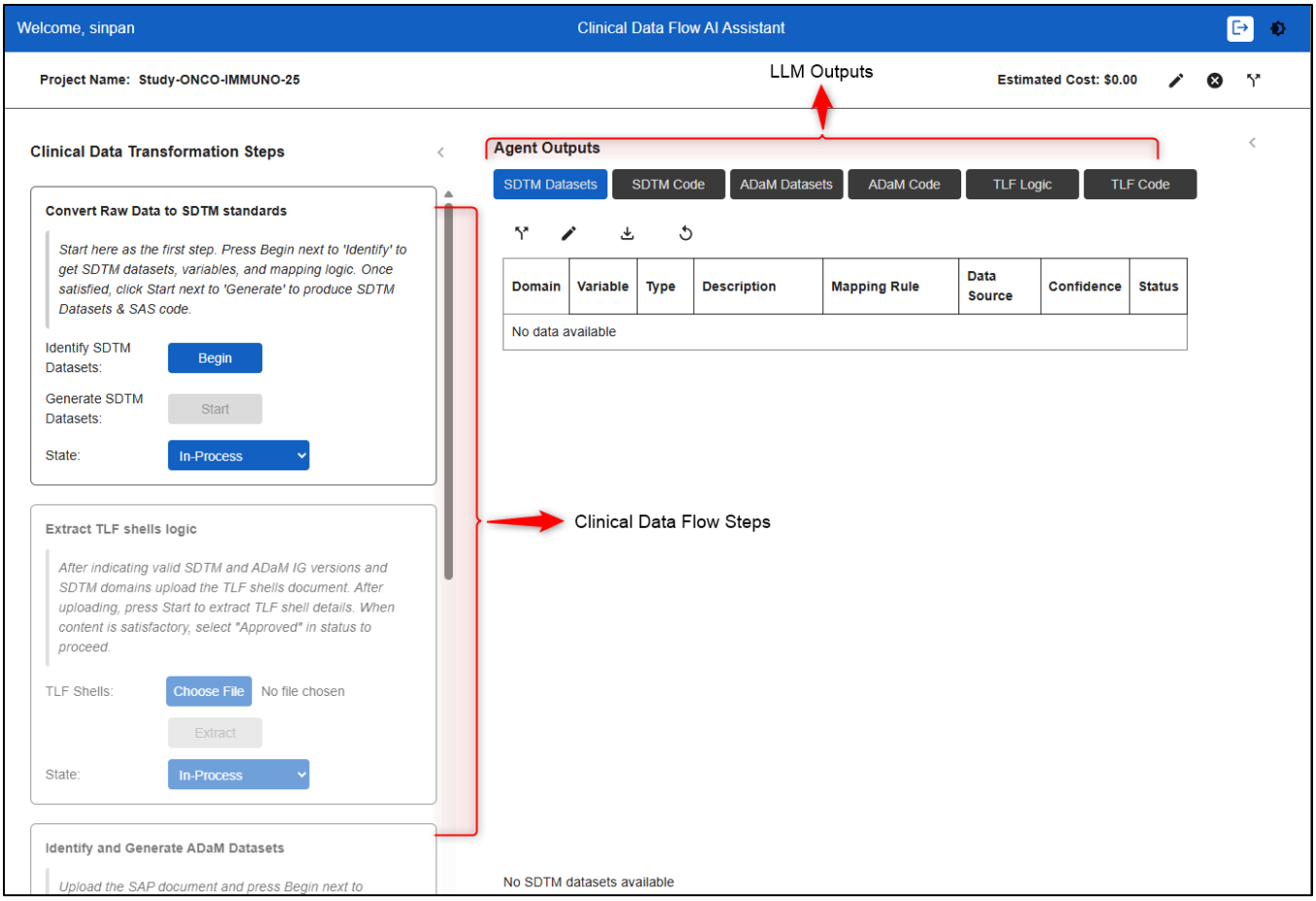
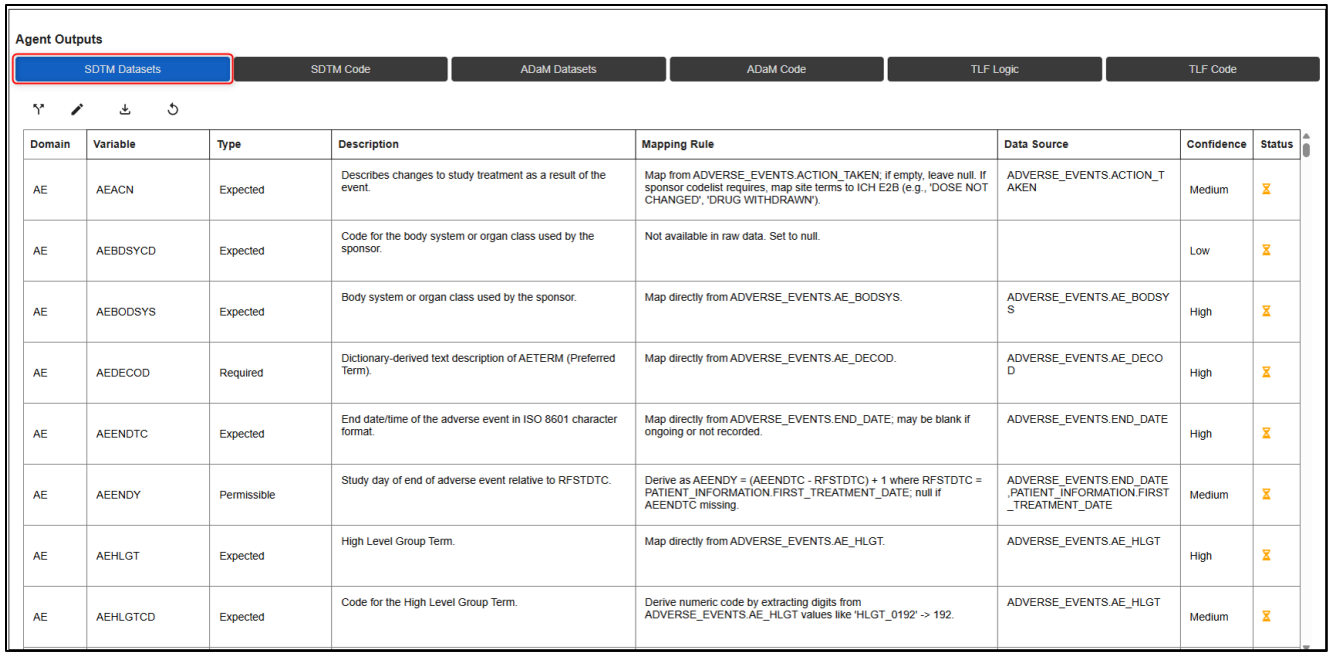


Figure 2: Mapping Rules Generation

Figure 3 illustrates the mappings returned by the LLM after analyzing the raw data schema, the selected data mapping model (such as CDISC SDTM IGs or a custom model), and any protocol requirements. These mappings can be edited, downloaded for further review, and re-uploaded.



Domain	Variable	Type	Description	Mapping Rule	Data Source	Confidence	Status
AE	AEACN	Expected	Describes changes to study treatment as a result of the event.	Map from ADVERSE_EVENTS.ACTION_TAKEN; if empty, leave null. If sponsor codelist requires, map site terms to ICH E2B (e.g., 'DOSE NOT CHANGED', 'DRUG WITHDRAWN').	ADVERSE_EVENTS.ACTION_TAKEN	Medium	
AE	AEBDSYCD	Expected	Code for the body system or organ class used by the sponsor.	Not available in raw data. Set to null.		Low	
AE	AEBODSYS	Expected	Body system or organ class used by the sponsor.	Map directly from ADVERSE_EVENTS.AE_BODSYS.	ADVERSE_EVENTS.AE_BODSYS	High	
AE	AEDECOD	Required	Dictionary-derived text description of AETERM (Preferred Term).	Map directly from ADVERSE_EVENTS.AE_DECOD.	ADVERSE_EVENTS.AE_DECOD	High	
AE	AEENDTC	Expected	End date/time of the adverse event in ISO 8601 character format.	Map directly from ADVERSE_EVENTS.END_DATE; may be blank if ongoing or not recorded.	ADVERSE_EVENTS.END_DATE	High	
AE	AEENDY	Permissible	Study day of end of adverse event relative to RFSTDTC.	Derive as AEENDY = (AEENDTC - RFSTDTC) + 1 where RFSTDTC = PATIENT_INFORMATION.FIRST_TREATMENT_DATE; null if AEENDTC missing.	ADVERSE_EVENTS.END_DATE ,PATIENT_INFORMATION.FIRST_TREATMENT_DATE	Medium	
AE	AEHLGT	Expected	High Level Group Term.	Map directly from ADVERSE_EVENTS.AE_HLGT.	ADVERSE_EVENTS.AE_HLGT	High	
AE	AEHLGTCD	Expected	Code for the High Level Group Term.	Derive numeric code by extracting digits from ADVERSE_EVENTS.AE_HLGT values like 'HLGT_0192' -> 192.	ADVERSE_EVENTS.AE_HLGT	Medium	

Figure 3: Editable Raw to SDTM Mappings

Figure 4 presents the data coverage report and lineage views created for each step, accessible from the mappings table screen. The right pane shows a lineage view that traces which source (such as raw or SDTM) fields are mapped to which target (such as SDTM or Adam) domain-variable. The left pane displays data coverage graphs and tables indicating how many raw variables were missing in the mapping and which SDTM variables remain unmapped. The left pane is shown only to Raw - SDTM step.

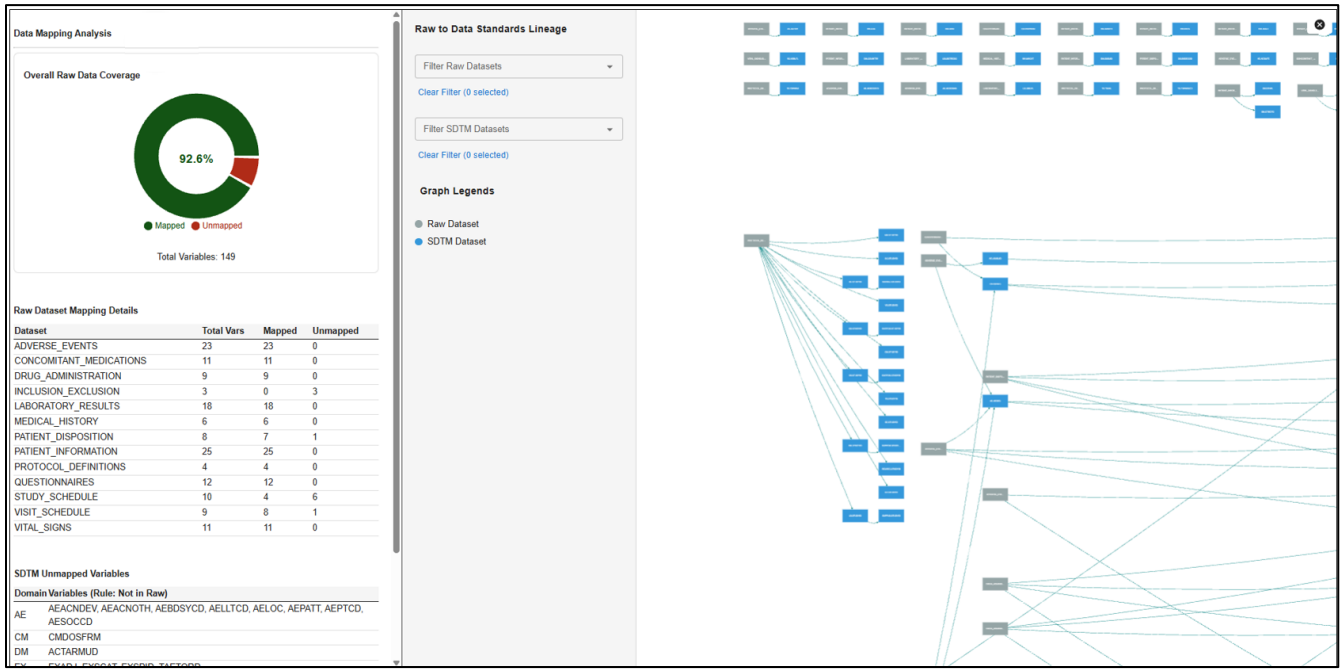


Figure 4: Lineage View and Data Coverage Reports

Once the mappings are finalized and reviewed by a human, the tool generates SAS code with an error-feedback loop. Figure 5 shows a screen with domain-specific tabs corresponding to each domain in the mappings. For each domain, tabs are available for the generated code, the last run log, and the last run output. On this screen the users can edit the code, re-run the code, or re-generate the code if needed. Any datasets created as part of running the code are stored in the folder locations specified when the project was created. The generated code also gets saved to the relevant folder locations.

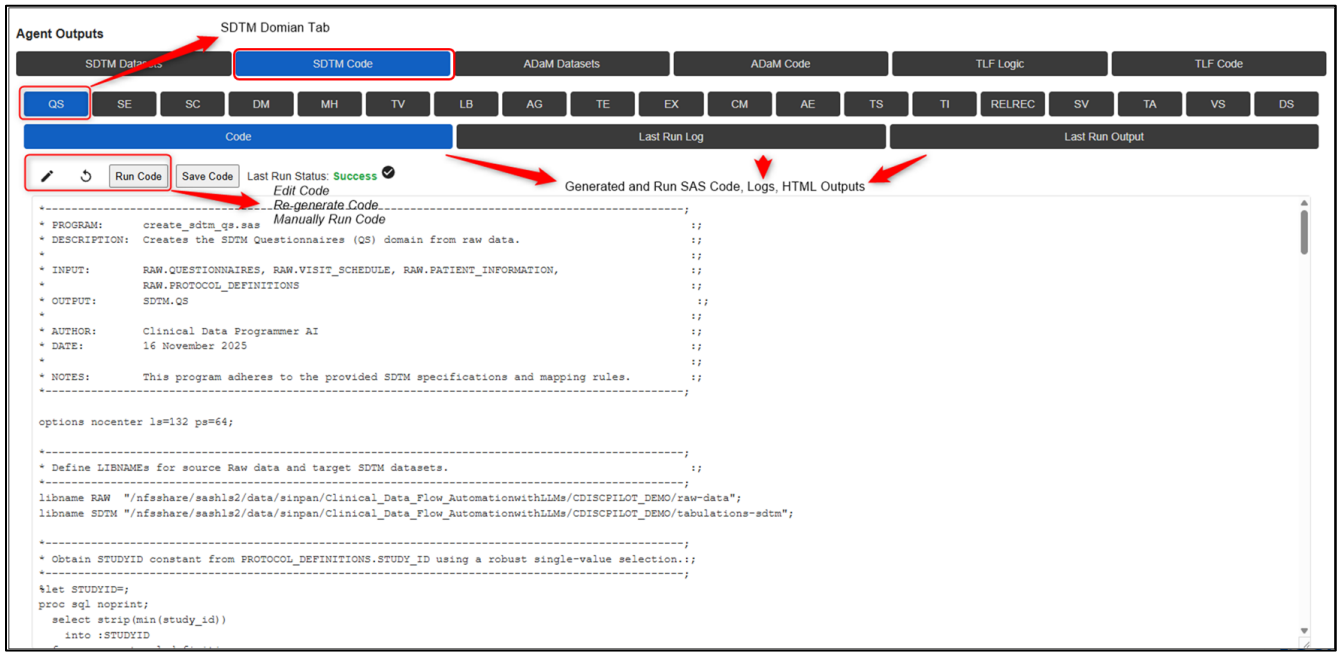


Figure 5: Generated Code

BUSINESS IMPACT - AUTOMATED MAPPING & CODE GENERATION - FASTER BUILDS, LOWER COST

Let's discuss the business impact of using these technologies. LLMs and agentic AI systems bring a set of capabilities that directly map to the bottlenecks in clinical programming. At their core, they combine document understanding, pattern recognition, and code synthesis to automate the high-volume, repeatable work that consumes most timelines while keeping expert programmers firmly in control.

Business impact across time, cost, and submission readiness:

- Compressed timelines: Automating the first 70–80% of SDTM/ADaM build and TLF coding can cut development cycles from months to days or hours for standard domains and shells. Teams reach "analysis-ready" states earlier, enabling more iterations with clinical and biostatistics stakeholders before database lock.
- Lowered operational costs: Person-hours spent on repetitive coding, reconciliation, and documentation shrink markedly.
- Strategic talent leverage: By handling high-volume foundations (often more standardized in Safety), AI frees expert programmers to focus on complex Efficacy datasets, high-level validation, time-to-

event and longitudinal derivations, handling intercurrent events, and edge cases that drive scientific credibility.

Critically, these tools are not a turnkey path to a submission package in isolation. Their value lies in jump-starting development, maintaining synchronized traceability, and amplifying expert judgment. The result is a repeatable, auditable acceleration that improves both speed and quality while elevating the programmer's role to designer, validator, and scientific partner.

CONCLUSION: AI AS A CO-PILOT, NOT A REPLACEMENT

AI-assisted automation is most effective when treated as a capable junior colleague embedded in the programming workflow - fast, tireless, and consistent, but reliant on human direction and review. In clinical programming, this "copilot" model reallocates effort from repetitive construction to expert oversight, speeding delivery without compromising rigor.

Practically, the copilot accelerates the scaffolding: it drafts SDTM and ADaM mapping shells from CRFs and specifications, proposes derivations with rule references, and generates code to create the datasets or statistical outputs. The programmer then operates at a higher altitude - curating inputs, challenging assumptions, refining edge cases, and adjudicating scientific appropriateness.

This automation changes roles, not headcount. Senior programmers become architects and reviewers who spend more time on complex efficacy derivations, cross-domain consistency, and exploratory analysis.

While this new technology introduces transformative capabilities, adoption must be deliberate and responsible. As an industry, we should establish and follow robust operating models that ensure safety, reliability, and regulatory compliance, while maintaining trust with stakeholders and safeguarding data and patients. Governance and compliance, quality with human oversight, and security and lifecycle management are just some of the many key practices that we must introduce into new business processes so organizations can responsibly harness these new capabilities, accelerate delivery, and improve quality - without compromising safety, compliance, or trust.

ACKNOWLEDGMENTS

I would like to acknowledge Matt Becker at SAS for his thought leadership and industry expertise and extend my thanks to the many other industry experts whose guidance was instrumental in making this initiative a success.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Author Name: Pankaj Attri
Company: SAS Institute Inc.
Address: 100 SAS Campus Dr, Cary, NC 27513
Work Phone: +1 919-279-5255
Email: Pankaj.attri@sas.com
Website: www.sas.com

Author Name: Matt Becker
Company: SAS Institute Inc.
Address: 100 SAS Campus Dr, Cary, NC 27513
Work Phone: +1 919-345-2507
Email: Matt.Becker@sas.com
Website: www.sas.com