PharmaSUG 2026 - Paper DS-196

Friction to Flow: LLM-Based Automation of Clinical Data Workflows

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 will be demonstrating an LLM-powered solution that automates the clinical data pipeline and, importantly, establishes 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. Each TLF is linked to its source ADaM datasets, which in turn trace back through SDTM to the original raw data. This automates traceability to the study design, supporting traceability evidence that protocol and SAP requirements have been met, ensuring compliance and confidence for regulatory submissions.

INTRODUCTION

Drug development now operates in a data landscape whose scale, speed, and heterogeneity outstrip what legacy clinical programming can comfortably support. Today's studies are bigger and more intricate, that is they are more adaptive, decentralized, and saturated with biomarkers, combining ePRO, wearable and sensor streams, imaging, and real-world evidence with conventional CRF inputs. At the same time, regulators have hard requirements for harmonization, data lineage, and reproducibility. The consequence is a growing shortfall between what manual handcrafted code can deliver and what compressed development timelines demand.

Clinical programmers are pivotal to closing this gap, yet a disproportionate share of their time is absorbed by rinse-and-repeat tasks: manually assembling SDTM and ADaM, reworking derivations at every data cut, among others. These steps are necessary but mechanical, don't scale well, and crowd out bandwidth for higher-value contributions.

A sensible path forward is AI-enabled automation. It shifts the center of effort from keystroke-level coding to standards-aware, parameterized pipelines that can be adapted quickly. Such platforms can read mapping specifications, generate code, and keep data and metadata in sync across the lifecycle. This is not about replacing experts; it magnifies their impact. With the groundwork accelerated, programmers can focus on what influences regulatory outcomes: rigorous handling of edge cases, construction of fit-for-purpose efficacy datasets, and transparent, auditable traceability.

In sum, AI-augmented automation equips clinical programming teams to keep pace with modern trial complexity, cut avoidable rework, and shift capacity toward science-driven priorities that improve decisions and accelerate submissions.

CHALLENGES IN CLINICAL PROGRAMMING

Even with well-established standards and seasoned teams, legacy clinical programming approaches are straining under today's demands. A prime example is the widespread practice of independent dual programming for quality control. While this technique is effective at uncovering discrepancies, it consumes substantial staff time and effort. The hidden cost is significant: hours spent reconciling two codebases could instead support scientific review, harmonization across studies, and earlier detection of data or analysis risks.

Source-to-result traceability is an additional pressure point. Health authorities expect an auditable, defensible provenance from CRF fields and external data streams, through SDTM and ADaM transformations, all the way to the final analyses and outputs.

These are not minor nuisances but systemic inefficiencies that elongate timelines and heighten regulatory exposure. Dual programming and hand-built documentation can validate a single data cut, yet they don't scale across multiple studies. Achieving both velocity and confidence requires operating models where specifications, implementation code, and metadata move in lockstep by default and where routine, repetitive steps are automated so that expert attention is reserved for the nuanced, judgment-heavy work of clinical data science.

LLM-DRIVEN AGENTS: REWIRING THE CLINICAL PROGRAMMING PIPELINE

Generative AI, particularly large language models (LLMs) paired with autonomous "agent" capabilities, is beginning to relieve several longstanding bottlenecks in clinical programming. Trained on extensive text and code, LLMs can interpret natural-language requirements and produce structured artifacts, including compilable code, from plain-English specifications. When embedded in agent frameworks that can plan multi-step tasks, call external tools (APIs, data stores, version control, validation suites), and retain state across iterations, these systems move beyond single prompts to orchestrate entire workflows.

The timing is right. Trial scale and intricacy have outpaced manual approaches, and the underlying AI stack (models, orchestration layers, guardrails, and security controls) is rapidly maturing to enterprise-ready levels. In this context, agent-style systems can read core study inputs like study design protocols, CRFs, SAPs, CDISC controlled terminology, and synthesize consistent mapping specifications, derivation logic, and initial code aligned to standards across domains.

Illustrative capabilities include:

- Creating SDTM mappings and datasets from raw sources and specifications, with conformance checks against CDISC rules and organizational conventions.
- Proposing ADaM designs and derivations (e.g., ADSL, efficacy and safety domains) directly from SAP requirements and TLF shells, with explicit, traceable links to endpoint definitions and statistical methods.
- Drafting TLF shells and starter programs that reflect SAP content, leveraging approved templates and parameterized macros.
- Populating controlled terminology, value-level metadata, and codelists; flagging inconsistencies and gaps for review.
- Generating unit tests, validation harnesses, and stubs for define.xml and review packages to support audit-ready traceability.
- Suggesting data quality checks and highlighting potential edge cases or ambiguous specifications.

Beyond point-in-time generation, LLM agents can coordinate end-to-end sequences: extract endpoints from the SAP, query the metadata repository for required SDTM variables, recommend ADaM transformations, and generate code. With retrieval-augmented generation, outputs are grounded in organization-specific code repositories and approved macro libraries, which curbs hallucination and improves consistency and reproducibility.

Crucially, these systems are built for expert oversight. Human reviewers approve specifications, inspect diffs, confirm assumptions, and gate promotion to production. The AI accelerates high-volume, repetitive steps and surfaces uncertainties; programmers retain authority over design choices, focusing their time on complex analyses, edge conditions, and scientific interpretation rather than boilerplate.

MINIMUM VIABLE PROTOTYPE

To put the earlier automation concepts into practice, we built a MVP that lets programmers harness LLMs to automatically draft transformation mappings, produce executable programs, and capture end-to-end lineage.

Figure 1 depicts project setup. Here, the user supplies basic project metadata and points the system to filesystem locations where Raw, SDTM, ADaM data, and any statistical outputs will be written. The end-to-end flow is modular and can be configured stepwise: Standardization (Raw → SDTM), Analysis Data Sets (SDTM → ADaM), and Reporting/Visualization (ADaM → TLF). During setup, the user also selects which LLMs to employ. Customer can bring their own LLMs. In other words, the LLMs are hosted within the customer's environment. This minimizes data privacy issues to a large extent although using synthetic data to generate specifications and code is far more secure.

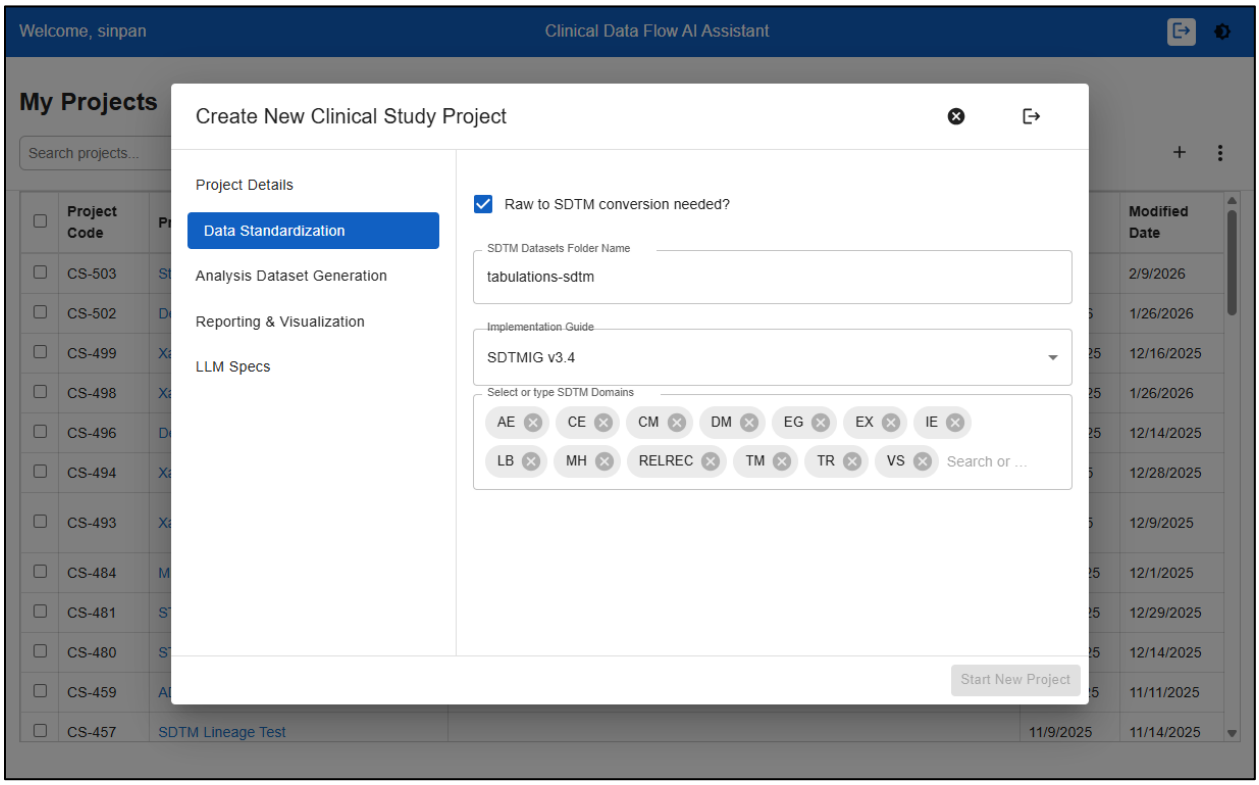


Figure 1: Project creation for a clinical study

Once the project is initialized, mapping rules can be generated. As shown in Figure 2, the project workspace presents a navigation pane on the left with the steps chosen during setup, and a set of tabs on the right that fill with LLM-generated content as each task completes. SDTM Datasets and SDTM Code tabs capture the Raw-to-SDTM specifications and programs. ADaM Datasets and ADaM Code cover the SDTM-to-ADaM stage. For the reporting phase, TLF Logic and TLF Code handle ADaM-to-TLF: the logic tab surfaces plain-English descriptions extracted from TLF shells, while the code tab contains the corresponding programs for each table, listing, or figure.

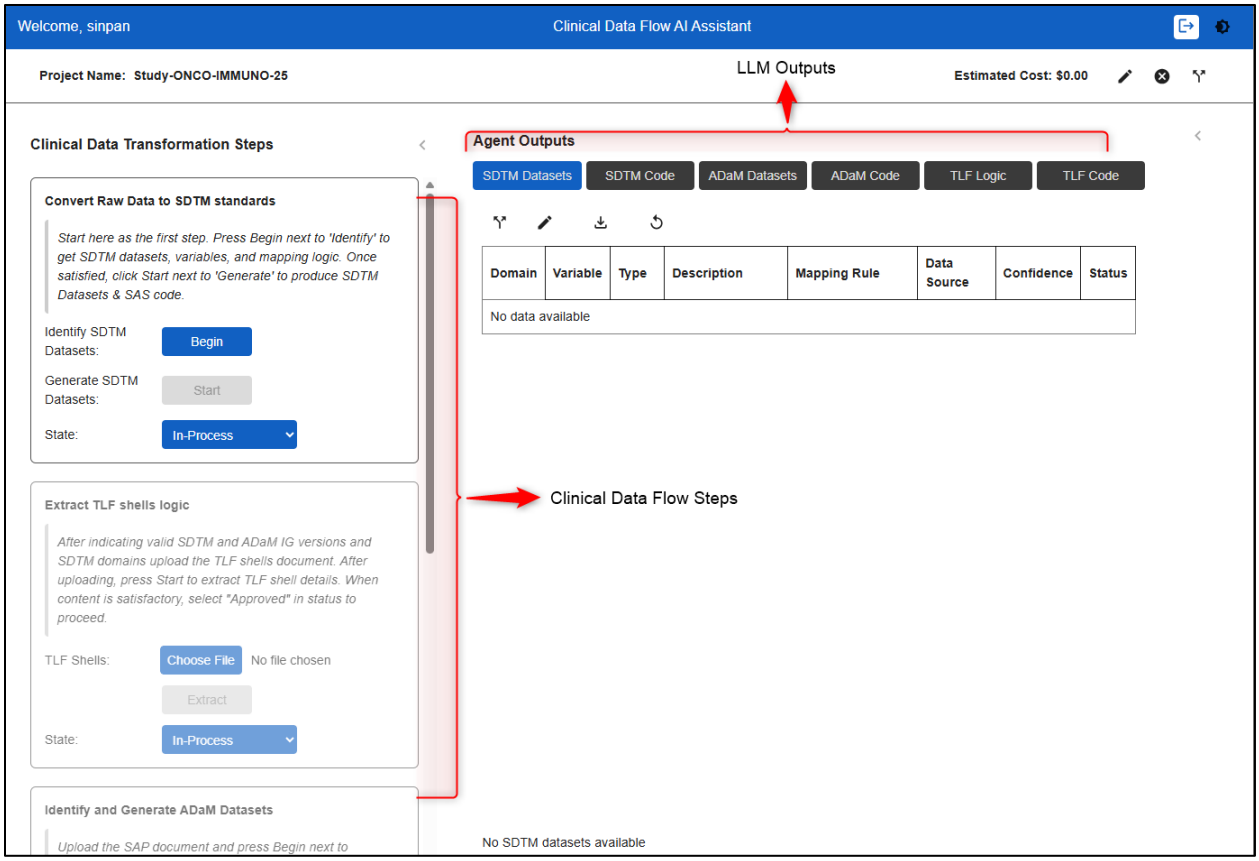
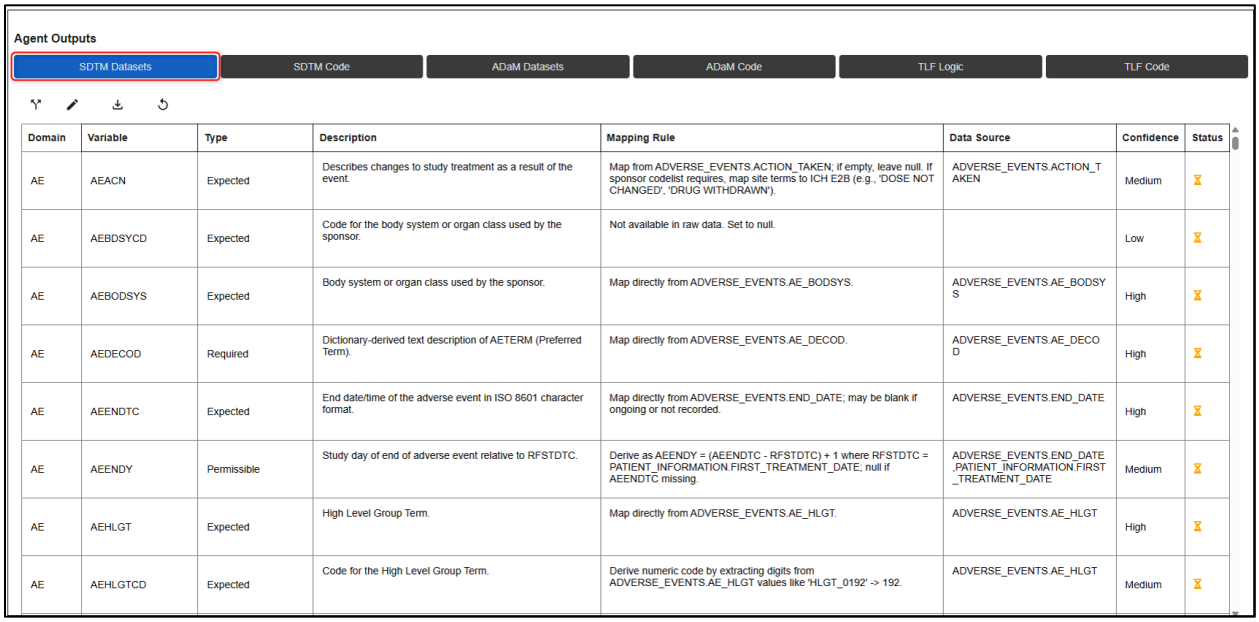


Figure 2: Project Details Screen

Figure 3 shows sample mapping proposals produced by the LLM after inspecting the raw data schema, the chosen mapping framework (such as CDISC SDTM implementation guides or an internal standard), and protocol requirements. Users can modify these mappings directly in the app, download them for offline review, and re-upload revised versions.



Agent Outputs

SDTM Datasets | SDTM Code | ADaM Datasets | ADaM Code | TLF Logic | TLF Code

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: SDTM Domains Mapping Specifications Example

Figure 4 highlights data coverage and lineage visualizations accessible from the mappings table. The right pane provides a lineage view that traces how source fields (from Raw or SDTM) flow into target domain-variable pairs (in SDTM or ADaM). The left pane presents coverage summaries—charts and tables showing which raw variables lack assignments and which SDTM variables remain unmapped. This coverage pane is displayed for the Raw-to-SDTM phase.

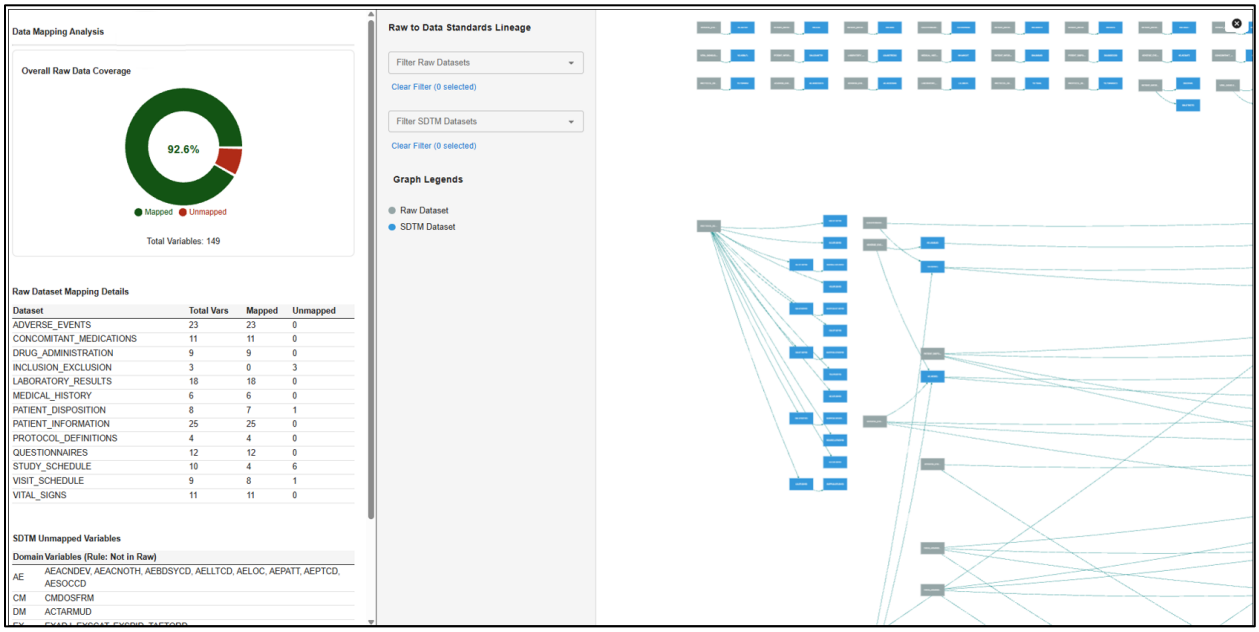


Figure 4: Data Coverage Reports and Lineage Graph View

After mappings are finalized and approved by a human reviewer, the tool generates SAS programs and executes them with an error-sensitive feedback loop. Figure 5 illustrates domain-centric tabs corresponding to each mapped domain; within each, the user can view the generated code, the most recent run log, and the latest output. From this screen, code can be edited, rerun, or regenerated as needed. All datasets created during execution are written to the folder paths specified at project creation, and the generated code is saved to the same designated locations.

Figure 5: SDTM Domains Code Screen Example

When all project stages are completed and approved, the system generates an end-to-end lineage graph. The graph's nodes represent both requirements (SAP endpoints and TLF shell items) and data assets (Raw sources, SDTM domains/variables, and ADaM datasets), with edges depicting the mappings and derivations that connect them. This provides a trace from each requirement through the corresponding outputs back to the underlying ADaM and SDTM variables and, ultimately, the raw data. Each node also displays key study metadata automatically extracted from documents such as the SAP, offering a single, audit-ready view from requirement to source. See Figure 6 for an example.

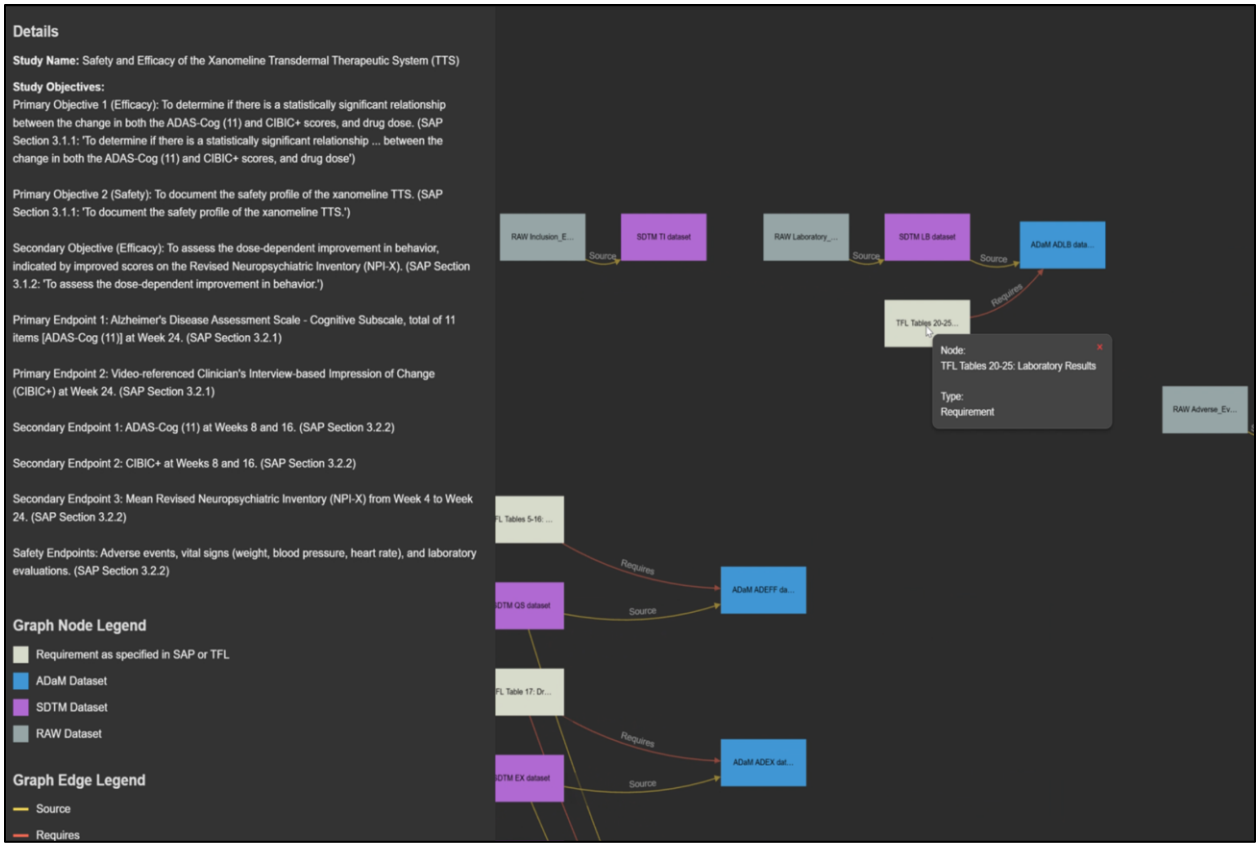


Figure 6: End-to-End Lineage View

BUSINESS IMPACT

Large language models and agent-style AI align directly with the pain points in clinical programming. By combining deep document parsing, pattern detection, and code generation, they automate the high-volume, repeatable activities that dominate timelines while keeping expert programmers in the driver's seat.

Impact on schedule, cost, and submission readiness:

- Shorter development cycles: Automating large chunks of the upfront SDTM/ADaM construction and TLF scaffolding can compress work that once took months into days or hours for standard domains and shells. Teams reach analysis-ready states sooner, allowing more iterations with clinical and biostatistics partners well before database lock.
- Reduced operating expense: Hours spent on repetitive coding, reconciliation, and documentation drop significantly, lowering overall delivery costs.
- Better use of expert talent: By taking on the high-volume baseline tasks (often most standardized in Safety), AI frees senior programmers to concentrate on the difficult work, complex Efficacy datasets, top-down validation strategies, and edge cases that underpin scientific credibility.

These tools are not a push-button route to a complete submission on their own. Their value is in jump-starting development, maintaining synchronized metadata and lineage, and amplifying expert judgment. The outcome is a governed and auditable acceleration that raises both speed and quality, while elevating programmers into roles as solution designers, validators, and scientific collaborators.

CONCLUSION: AI AS PARTNER, NOT A STAND-IN

AI-assisted automation delivers the most value when treated as a capable teammate embedded in day-to-day programming; fast, consistent, and tireless, yet guided by human judgment. In clinical programming, this “co-pilot” approach shifts effort away from repetitive build work and toward expert review and decision-making, accelerating delivery while preserving rigor.

In practical terms, the co-pilot assembles the scaffolding: it drafts SDTM and ADaM skeletons from CRFs and specifications, recommends derivations with cited rules, and produces programs to generate datasets and statistical outputs. Programmers then operate at a higher level, curating and validating inputs, probing assumptions, resolving edge conditions, and ensuring scientific defensibility. This does not completely remove the need for double programming, but the co-pilot can be thought of an either primary or secondary programmer.

This is also a role transformation, not a job reduction. Senior programmers take on architect and reviewer responsibilities, dedicating more time to intricate efficacy derivations, cross-domain harmonization, and exploratory analyses.

Adoption, however, must be thoughtful and responsible. The industry should implement robust operating models that prioritize safety, reliability, and regulatory adherence while maintaining stakeholder trust and protecting data and patients. Strong governance documented human oversight, security and access controls, audit-ready lifecycle management, and continuous quality monitoring should be built into new workflows. With these guardrails in place, organizations can tap into the new capabilities to speed delivery and elevate quality, without compromising safety, compliance, or trust.

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