



# Democratizing SDTM Transformations: AI-Assisted Function Development with Business- Driven Governance

PRESENTED BY

Rostislav Markov

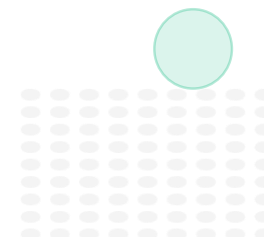
Amazon Web Services, Inc.

Jacques Lanoue

Merck & Co., Inc.

Ulf Nielsen

MSD Merck Sharp & Dohme AG

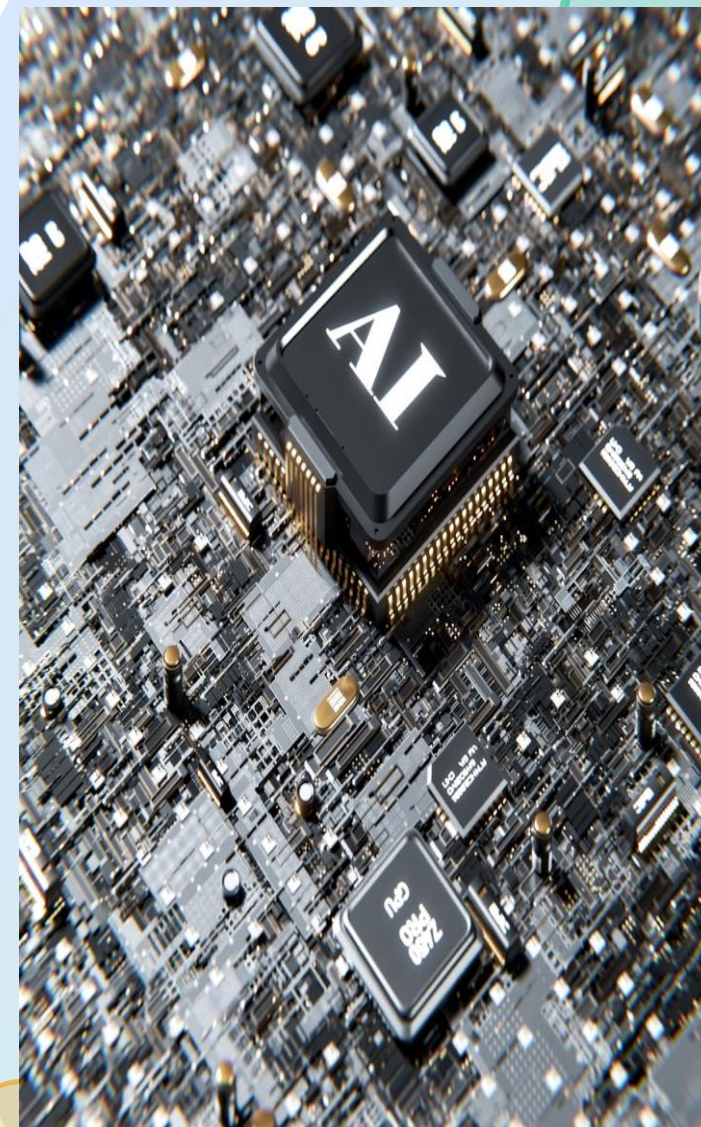


March 25, 2026



# Democratizing SDTM Transformations

In the evolving landscape of clinical data management, traditional SDTM dataset generation faces significant challenges due to rigid software development lifecycles. Our innovative '**Bring Your Own Function**' (BYOF) framework empowers statistical programmers to create study-specific functions efficiently. By integrating AI assistance and robust governance, we enhance adaptability while ensuring compliance and data quality. This presentation explores the BYOF framework's key components, guiding principles, and its transformative impact on SDTM transformations.



# Overview of BYOF Framework

1

## Rigid Development Lifecycles

Traditional SDTM systems are constrained by inflexible software development lifecycles, causing delays in adapting to evolving study requirements and limiting innovation.

2

## Democratizing Function Development

The BYOF framework empowers statistical programmers to create and deploy functions independently, enhancing collaboration and reducing bottlenecks in the development process.

3

## Context-Aware AI Assistance

Integrating AI assistance within the BYOF framework allows for context-aware code generation, improving adaptability and ensuring compliance with established standards.

LEARN

# Challenges in SDTM Dataset Generation

## Centralized Governance Bottlenecks

Centralized governance ensures data accuracy but creates significant bottlenecks, delaying necessary changes and adaptations in SDTM dataset generation processes.

## Delays in Study Adaptation

The rigid structure of traditional systems leads to delays in adapting to study-specific needs, hindering timely responses to evolving research requirements.

## Need for Flexibility

Statistical programmers require greater flexibility and speed in their workflows to meet the unique demands of each study without compromising compliance.

[LEARN](#)

# Key Components of BYOF Framework



## COMPREHENSIVE FUNCTION TEMPLATES

Templates enforce coding standards, ensuring consistency across functions. Each template includes code, metadata, automated tests, and documentation, enhancing reusability and compliance.



## CONTEXT-AWARE AI ASSISTANCE

AI generates code suggestions based on validated repository patterns, allowing statistical programmers to focus on architecture and business logic while ensuring quality and efficiency.



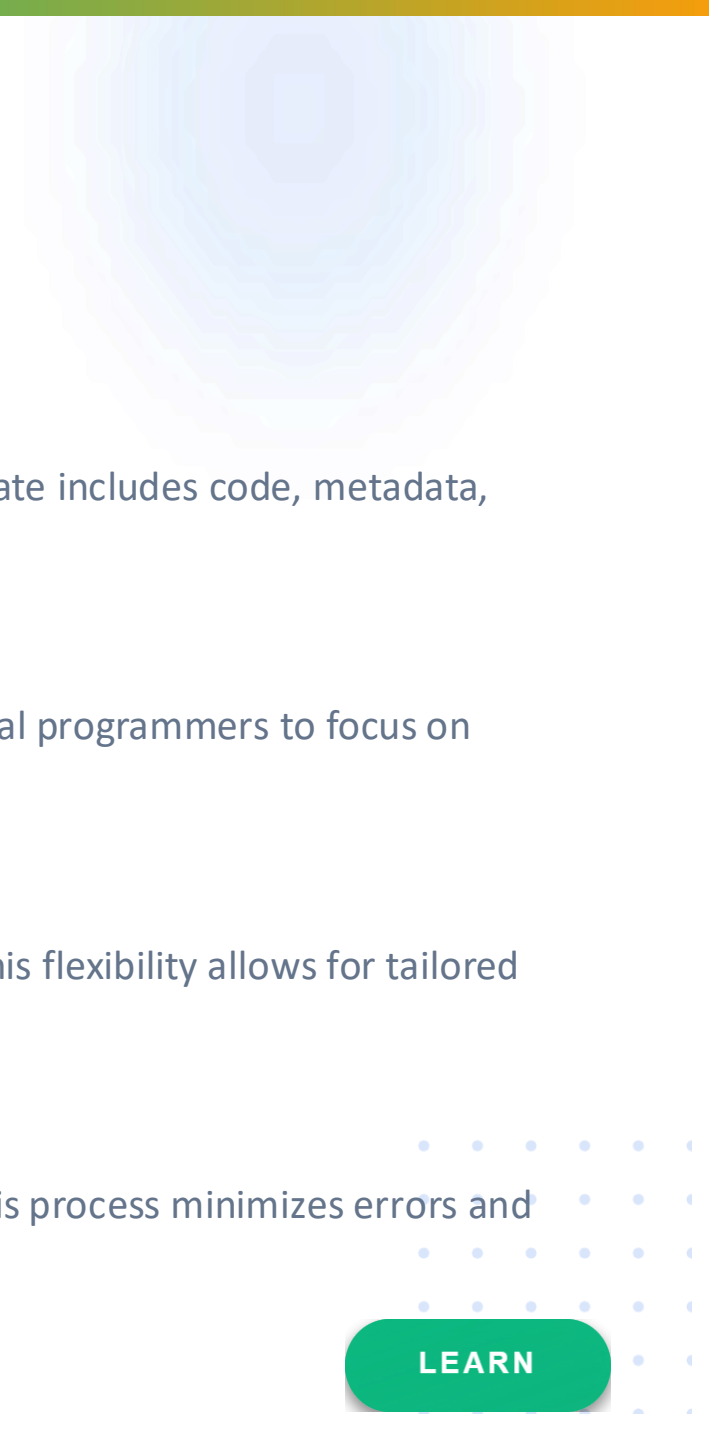
## STUDY-SCOPED DEPLOYMENT

Deployment of mapping functions is study-specific, preventing system-wide impacts. This flexibility allows for tailored adaptations without compromising existing functionalities or compliance.



## AUTOMATED VALIDATION PROCESSES

Automated validation against registry patterns ensures compliance and data quality. This process minimizes errors and maintains high standards throughout the function development lifecycle.

[LEARN](#)

# Code Repository Structure

```

byof-functions/
├── assign/
│   ├── v1/
│   │   ├── app.py
│   │   ├── metadata.json
│   │   ├── __init__.py
│   │   └── tests/
│   │       ├── scenarios.feature
│   │       ├── steps/
│   │       └── data/
│   └── v2/
│       └── ...
├── derive_ae_end_date/
│   ├── v1/
│   │   ├── app.py
│   │   ├── metadata.json
│   │   ├── __init__.py
│   │   └── tests/
│   │       ├── scenarios.feature
│   │       ├── steps/
│   │       └── data/
└── scripts
  
```

Metadata files use an extensible JSON structure:

```

{
    "id": "<auto-generated-uuid>",
    "arn": "resourceId",
    "moduleName": "byof_functions",
    "packageName": "sdtmfunctions",
    "subModuleName": "<your_module_name>",
    "version": "1",
    "name": "<your_module_name>",
    "displayName": "<your_module_name>",
    "description": "Description about
<your_module_name>",
    "studyIds": ["SID.1", "SID.2"],
    "lookUpCodeList": true,
    "validSourceDomains": ["SDM1", "SDM2"],
    "validTargetDomains": ["TDM1", "TDM2"],
    "validSourceFields": ["SF1", "SF2"],
    "validTargetFields": ["TF1", "TF2"],
    "functionType": "derivation"
}
  
```

# Behavior-driven development documentation

Feature: `call_study_day` function

The `CALL_STUDY_DAY` function is designed to calculate the reference study day of a given date, relative to `RFSTDTC` from the DM domain, move the derived value to an existing `xxDY/xxxxDY` variable in the target dataset, and record the traceability information.  
This function ensures data consistency and provides traceability of mapping.

**Scenario Outline:** Study day is derived from `RFSTDTC` in DM and date in source variable  
Given the `<source_dataframe>` exists  
And the `<source_variable>` exists  
And value within `<source_variable>` is a date  
And the `RFSTDTC` exists in `<DM>`  
And the `<target_variable>` exists in the `<target_dataframe>`  
When the `call_study_day` function executes  
Then the `<target_variable>` in `<target_dataframe>` has the derived `study_day`

Examples:

|                  |                                  |         |                              |                  |
|------------------|----------------------------------|---------|------------------------------|------------------|
| source_dataframe | source_variable                  | DM      | target_variable              | target_dataframe |
| CM, LB, EX       | CMSTDTC, CMENDTC, LBDTC, EXSTDTC | RFSTDTC | CMSTDY, CMENDY, LBDY, EXSTDY | CM, LB, EX       |

**Scenario Outline:** `RFSTDTC` existence  
Given the `<source_dataframe>` exists  
And the `<source_variable>` exists  
And value within `<source_variable>` is a date  
And the `RFSTDTC` does not in `<DM>`  
And the `<target_variable>` exists in the `<target_dataframe>`  
When the `call_study_day` function executes  
Then the `call_study_day` function throws an error

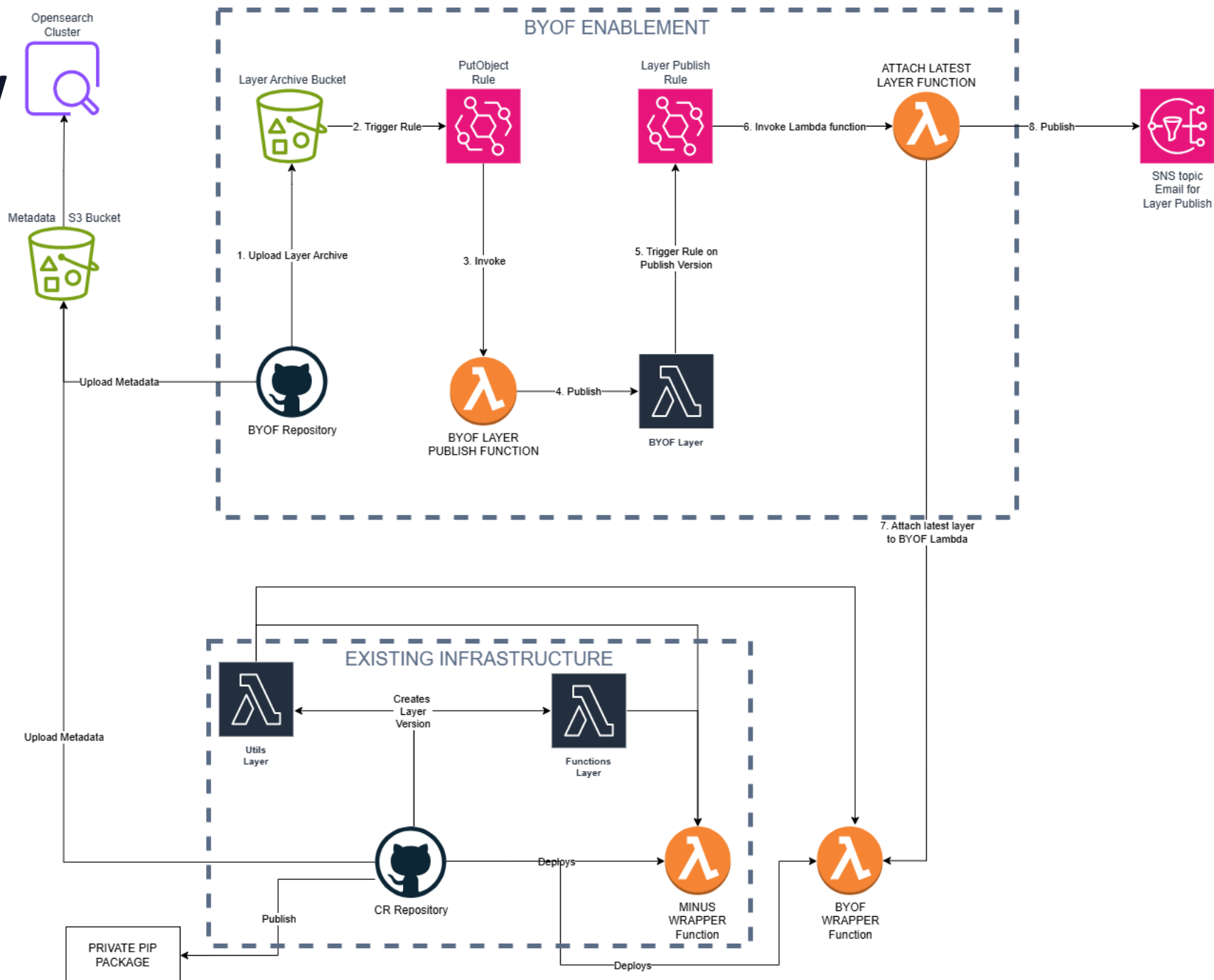
Examples: Source column contains date and `RFSTDTC` does not exists

|                  |                                  |         |                              |                  |
|------------------|----------------------------------|---------|------------------------------|------------------|
| source_dataframe | source_variable                  | DM      | target_variable              | target_dataframe |
| CM, LB, EX       | CMSTDTC, CMENDTC, LBDTC, EXSTDTC | RFSTDTC | CMSTDY, CMENDY, LBDY, EXSTDY | CM, LB, EX       |

# Publication workflow

The **automated pipeline registers functions** with the search cluster (used by the GUI to discover available functions and their metadata), **builds the compute resources**, and **performs consistency checks** including:

- **Metadata validation**
- **Version conflict prevention**
- **Visibility enforcement** (typically study-scoped for new versions pending roadmap review)



# Functions registry

 Clinical Data Layer DEV

 rostislav.markov2@msd.com ▼

Mapping & Tracking

Functions

Dataset Operations

Deliverable

Package Configuration

Configuration

Logs

Import

≡

▼ Pre-Processing Functions

Q Find function

|              | Functions ▼                                     | Version ▼ | Target Domains | Target Fields | Source Domains | Source Fields |
|--------------|-------------------------------------------------|-----------|----------------|---------------|----------------|---------------|
| <div>×</div> | <a href="#">call_clean_invisible_characters</a> | 1 ▼       | *              | **            | *              | **            |
| <div>×</div> | <a href="#">call_source_with_header</a>         | 1 ▼       | *              | **            | *              | **            |

▼ Transformation Functions

Q Find function

|              | Functions ▼                       | Version ▼ | Target Domains | Target Fields        | Source Domains | Source Fields                |
|--------------|-----------------------------------|-----------|----------------|----------------------|----------------|------------------------------|
| <div>×</div> | <a href="#">get_usubjid</a>       | 1 ▼       | *              | *.USUBJID, *.RSUBJID | *              | *.Subject.Name, *.Study.Name |
| <div>×</div> | <a href="#">get_tf_adv_za_con</a> |           |                | **                   | *              | **                           |
| <div>×</div> | <a href="#">direct_move</a>       |           |                |                      | *              | **                           |
| <div>×</div> | <a href="#">suppqual_lookup</a>   |           |                | *.SUPPQUAL*          | *              | **                           |

suppqual\_lookup Description

×

The SUPPQLOOKUP function is designed to move the value of a variable in the source data to the Target variable QVAL in the SUPPQUAL (supplementary qualifiers) set in the target dataset and record the traceability information.

# Programmatic execution via mapping specs

```
{
  "mappings": [
    {
      "SourceTable": "AE",
      "SourceVariable": "AEENDTC_RAW",
      "TargetList": [
        {
          "FunctionName": "ISO8601 (v1)",
          "SDTMTransformationFunction":
"functionId",
          "SDTMVariable": "AEENDTC",
          "Group": 1,
          "SDTMDomain": "AE"
        }
      ]
    }
  ],
}
```

```
"derivations": [
  {
    "Function": "derivationId",
    "FunctionName": "get_informed_consent",
    "Version": 1,
    "Sequence": 0
  }
],
"transformationFunctions": [
  {
    "Function": "functionId",
    "FunctionName": "std_fn_split_text",
    "Version": 1
  }
]
```

# AI-Assisted Functions Development



## AI AS CODE ASSISTANT

AI acts as a supportive tool for statistical programmers, generating code suggestions based on validated repository patterns, enhancing efficiency and accuracy in function development.



## REJECTING BLACK BOX APPROACHES

Our framework emphasizes transparency by avoiding opaque AI processes, ensuring that generated code is understandable, traceable, and reusable across different studies.



## FOCUS ON ARCHITECTURE

Statistical programmers concentrate on designing robust architectures and business logic, while AI manages implementation details, fostering a collaborative development environment.



## ENHANCING SCALABILITY

The integration of AI promotes scalability by allowing multiple versions of functions to coexist, catering to diverse study requirements without compromising existing functionalities.

LEARN

# AI Assistant roles

*“For Study SID.123, we need a modified call\_study\_day derivation. Instead of using RFSTDTC as Day 1 for all records, this protocol defines Day 1 by treatment period (different reference dates per EPOCH), and requires specific handling for partial dates (no imputation; leave day missing but still record traceability). Other studies must continue to use the original call\_study\_day behavior.”*

1

## Requirements interpreter

Turning transformation needs expressed in natural language into BDD feature files and metadata content

2

## Function recommender

Searching the repository to suggest existing functions/versions that fit the programmer’s need

3

## Code assistant

Generating new function implementations or refactoring/extending existing functions

4

## Validation and QA assistant

Proposing new test scenarios for edge cases, test data variations in tests/data/ and interpreting failing tests.

# Organizing the BYOF Code Repository

- **Multiple Versions for Flexibility**

Maintaining multiple function versions allows studies to select the most suitable version, ensuring adaptability to specific requirements without disrupting existing workflows.

- **Version-Specific Metadata Governance**

Each function version includes its own metadata.json, enhancing governance by providing detailed information on functionality, parameters, and applicable studies for accurate tracking.

- **Structured Testing for Reproducibility**

Tests are organized into scenarios, steps, and data directories, ensuring that function behavior is consistently validated across various conditions, promoting reliability in deployments.

- **Accurate Discovery and Traceability**

The repository structure facilitates easy discovery of functions and their versions, ensuring traceability and compliance with governance standards throughout the development lifecycle.

[LEARN](#)

# Transformative Impact of BYOF Framework

50%

Reduction in Development Time

Accelerating project timelines significantly

95%

Compliance Adherence

Ensuring regulatory standards are met consistently

70%

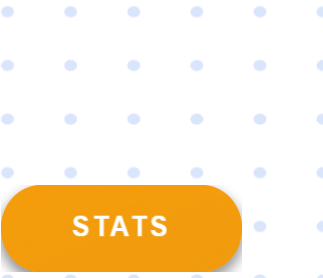
Enhanced Adaptability

Tailoring solutions to specific study needs effectively

4x

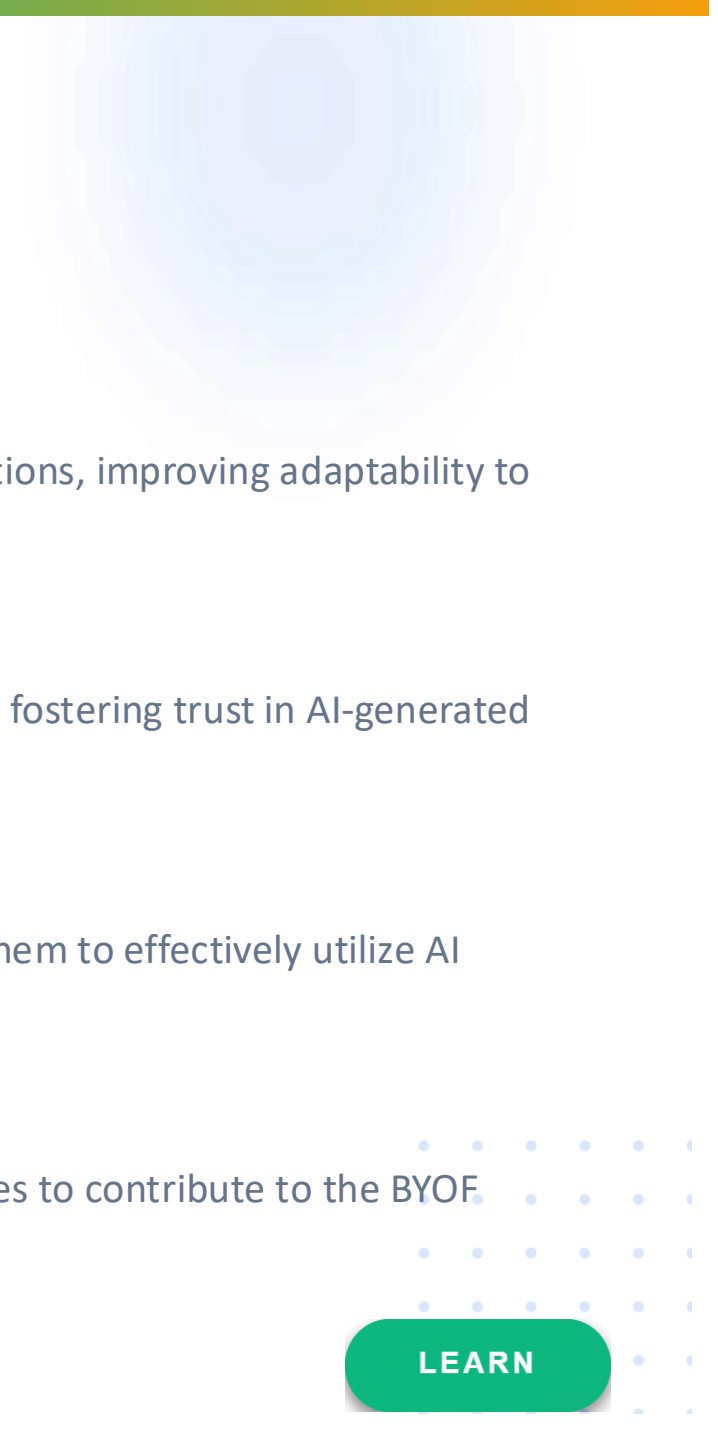
Improved Code Reusability

Maximizing efficiency across multiple studies



# Future Directions for BYOF

-  **EXPAND AI CAPABILITIES**  
Enhancing AI functionalities will enable the development of more complex mapping functions, improving adaptability to diverse study requirements and accelerating transformation processes.
-  **ROBUST GOVERNANCE FRAMEWORKS**  
Integrating advanced governance frameworks will ensure compliance and quality control, fostering trust in AI-generated functions while maintaining regulatory standards across all studies.
-  **USER TRAINING ENHANCEMENT**  
Investing in comprehensive training programs for statistical programmers will empower them to effectively utilize AI tools, enhancing their skills in function development and deployment.
-  **FOSTER TEAM COLLABORATION**  
Encouraging collaboration across teams will drive innovation, allowing diverse perspectives to contribute to the BYOF framework, ultimately leading to more effective solutions.

[LEARN](#)

# Key Considerations and Implications

## Balancing Competing Objectives

Function development must balance speed and compliance, ensuring rapid deployment while adhering to regulatory standards, which is essential for maintaining data integrity and quality.



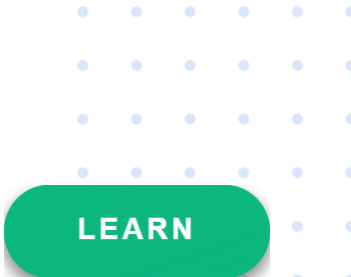
## Curated Context for AI

Providing AI with curated context enhances its effectiveness, allowing it to generate relevant code suggestions that align with study-specific requirements and organizational standards.



## Governance of Business Objects

Effective governance of business objects is crucial to ensure traceability and compliance, preventing unauthorized changes and maintaining the integrity of the SDTM transformation process.



# Acknowledgments

## 1. Collaboration with AWS and Merck

We extend our gratitude to AWS and Merck for their invaluable collaboration, which has significantly enhanced the development of the BYOF framework and its implementation.

## 2. Appreciation for Programming Community

We appreciate the statistical programming community for their ongoing support and contributions, which have been crucial in refining our approach to SDTM transformations.

## 3. Feedback from Industry Experts

We are grateful for the constructive feedback from industry experts, which has helped us improve our framework and ensure it meets the highest standards of governance.

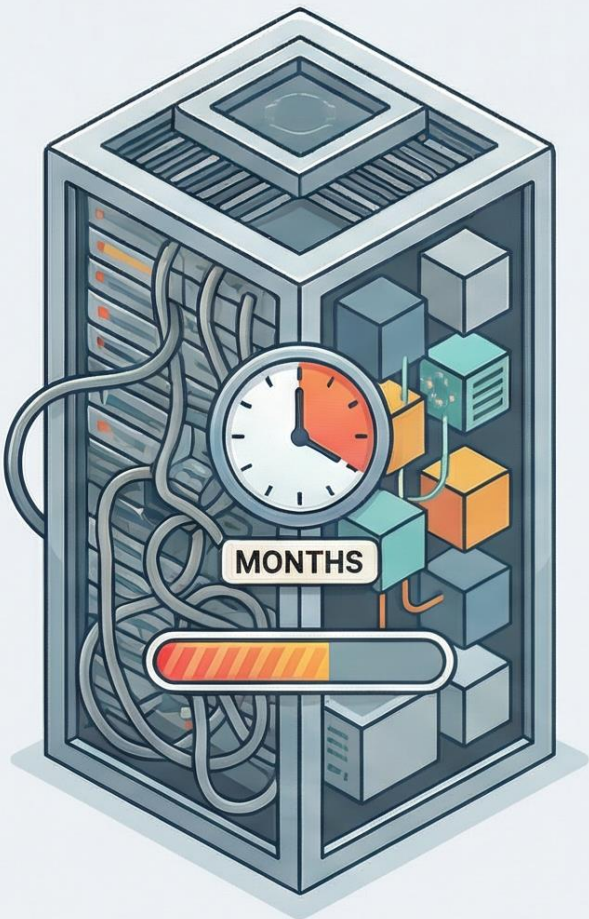
## 4. Importance of Governance

Recognizing the importance of governance, we emphasize its role in maintaining compliance and data quality within the BYOF framework, ensuring sustainable development.

[LEARN](#)

# The BYOF Framework: Revolutionizing SDTM Transformation with AI

## Solving the Centralized Bottleneck



### From Months to Minutes

Decoupling study-specific logic from the core system slashes deployment time from months to minutes.



### Functions as “Business Objects”

Logic is treated as a modular, versioned object rather than a hard-coded global system component.



Manual Debugging

System Architecture & Analysis

### Professional Role Evolution

Programmers shift focus from manual debugging to high-value system architecture and requirements analysis.

## The AI-Assisted Architecture



### The AI Agentic Workflow

AI agents (Amazon Kiro) interpret requirements, suggest Python code, and generate test scenarios.

### BYOF Stack Components

| Component | Technology Used        |
|-----------|------------------------|
| Logic     | Python (Pandas-based)  |
| Testing   | Gherkin (BDD) / Pytest |
| CI/CD     | GitHub Actions         |

### BYOF Stack Components

| Component | Technology Used        |
|-----------|------------------------|
| Logic     | Python (Pandas-based)  |
| Testing   | Gherkin (BDD) / Pytest |
| CI/CD     | GitHub Actions         |



Pandas-based



Metadata



CI/CD



OpenSearch

### Modern Python-Based Stack

Utilizes Pandas, JSON metadata, GRHub Actions, and OpenSearch for high-performance data transformation.



### Rigorous Governance Guardrails

Ensures compliance through Behavior-Driven Development (BDD) and mandatory human-in-the-loop reviews.





## Key Takeaways

The BYOF framework revolutionizes SDTM transformations by enabling rapid, study-specific function development while ensuring compliance and quality. Key takeaways include the importance of context-aware AI, the role of statistical programmers as developers, and the need for robust governance. Moving forward, organizations should embrace this democratization to enhance adaptability and efficiency in clinical data management.

