

Agenda

1. Active Problem Areas
2. Vision
3. Goals
4. High-level overview framework
5. Framework architecture design
6. Proof-of-Concept outcome
7. Lessons learned

Active Problem Areas

Clinical Data Transformation



Manual, Time-Intensive Work

SAS-based transformations require heavy coding & validation effort for each clinical data point.



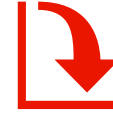
Skills Gap

Learning modern tools takes longer than traditional SAS programming



Variable Quality

Inconsistent coding standards increase error risks.



Low Reusability

Study-specific code can't be easily reused across different trials.



Complex Audit Trails

Extra manual steps needed for documentation and validation.

Vision

To revolutionize clinical data transformation by combining Python's powerful data ecosystem with Generative AI, creating a standardized, automated, and transparent workflow that accelerates study delivery while ensuring regulatory compliance and data integrity.

Goals

Accelerate Data Transformation

- Reduce manual coding effort and turnaround time from raw to standardized datasets.

Standardize and Improve Quality

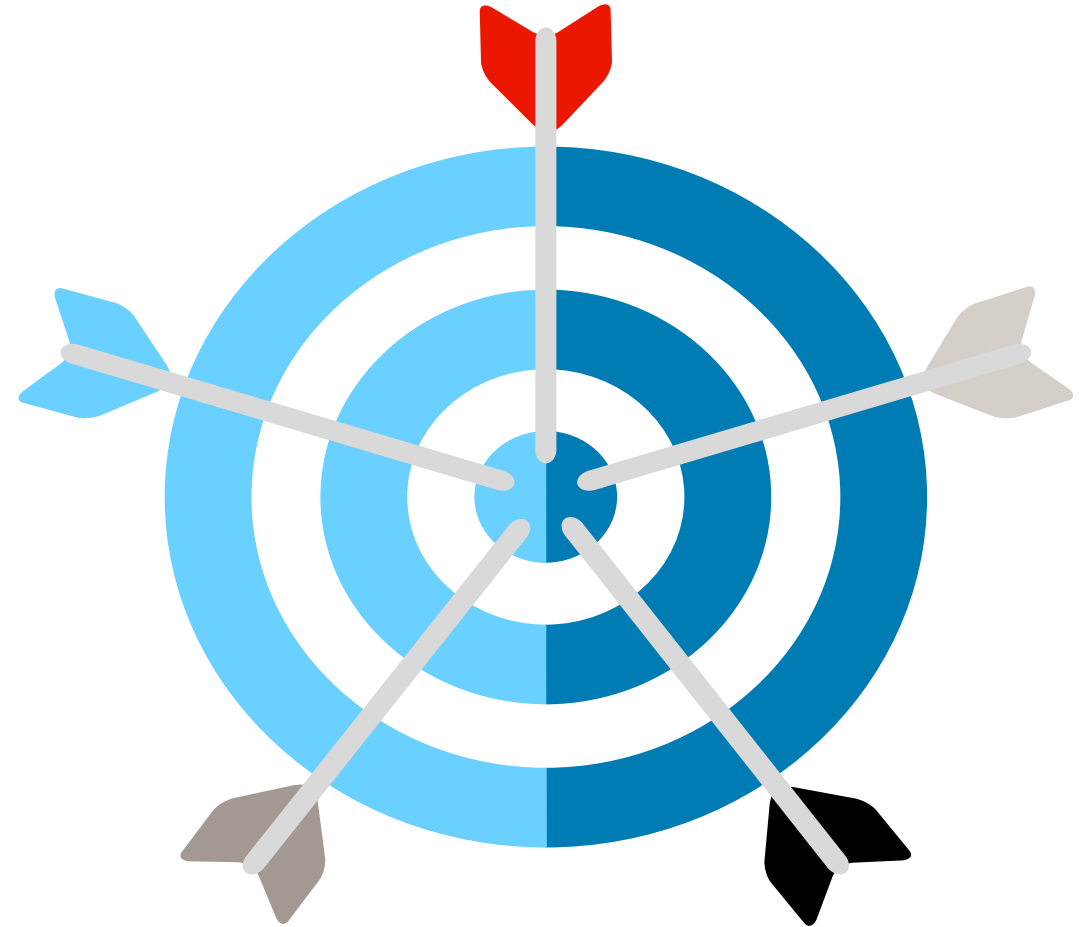
- Create reusable, metadata-driven transformation workflows that minimize human error.

Enable Intelligent Automation

- Harness Generative AI to automate code generation, variable mapping, and maintenance.

Promote Scalability and Reusability

- Modular architecture to adapt across multiple studies and therapeutic areas with minimal rework.



High-level overview framework

Python-based

Designed specifically for clinical data transformation workflows.

Leverages Python's robust data processing ecosystem.

Modular, Reusable Components

Independent functional layers for ingestion, transformation, and output.

Standardized transformation patterns reused across studies.

Integrated Metadata Management

Central repository driving variable definitions, mappings, and rules.

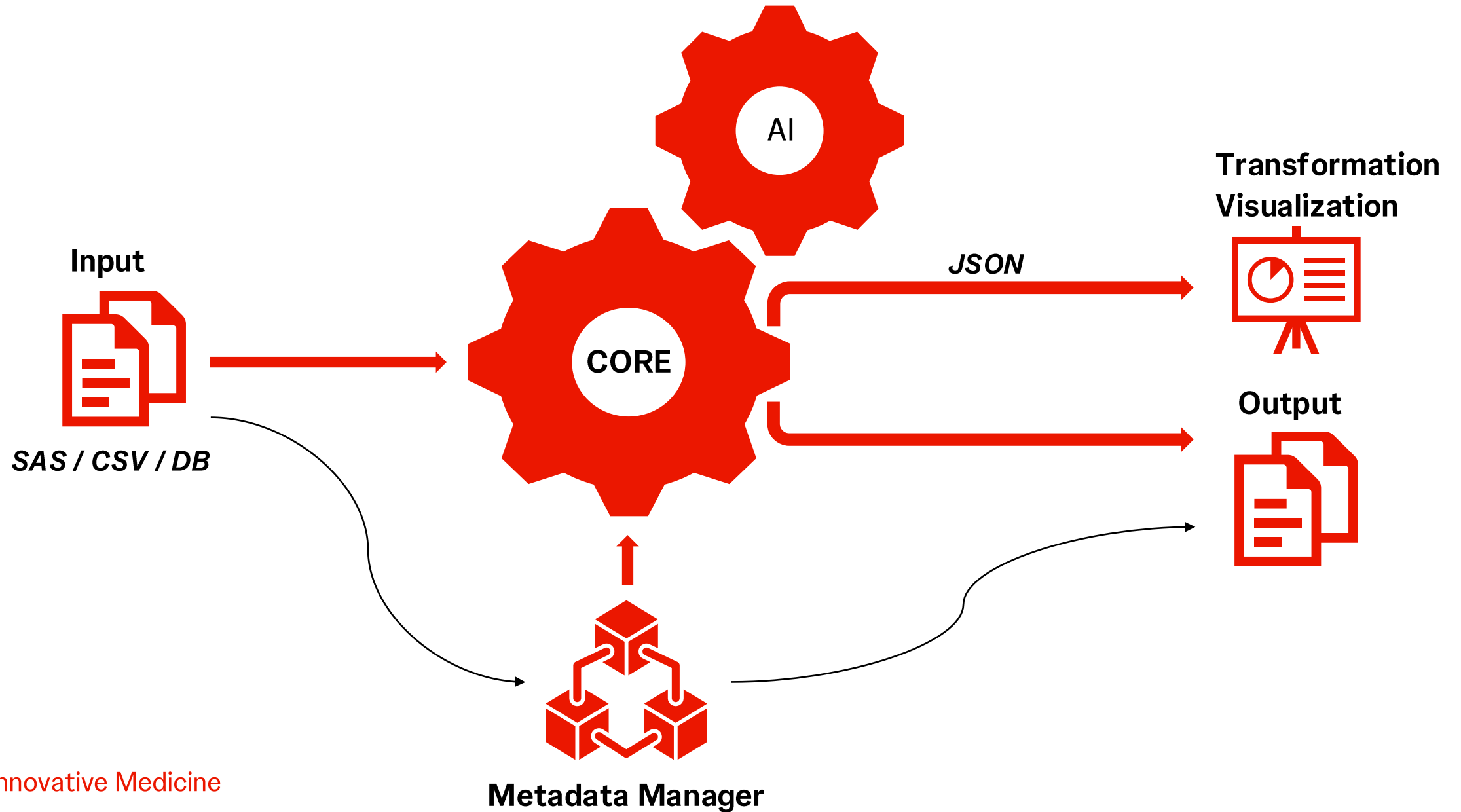
Enables consistent and easily maintainable transformations.

AI-Assisted Code Generation

Generative AI converts plain English requirements into transformation code.

Intelligent variable mapping suggestions and automated maintenance.

Framework architecture design



Framework architecture design – Core Engine



```
1 DRM_AE.AETERM.COPY(SOURCE.AETERM)
✓ 0.0s Python

COPY AETERM:
Input values: ['ABDOMEN CRUSHING', 'ABDOMINAL ADHESIONS', 'INTESTINAL ADHESIONS', 'BOWEL ADHESION']
Output values: ['ABDOMEN CRUSHING', 'ABDOMINAL ADHESIONS', 'INTESTINAL ADHESIONS', 'BOWEL ADHESION']

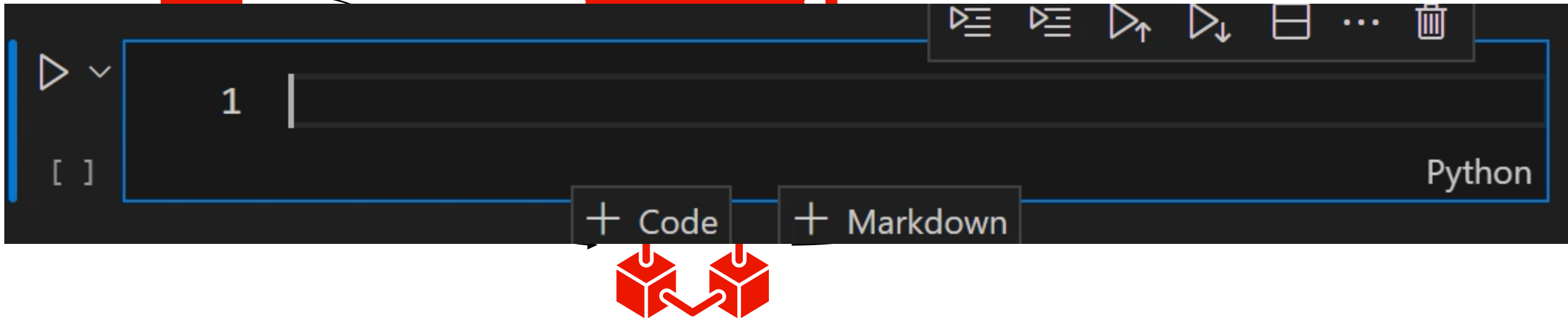
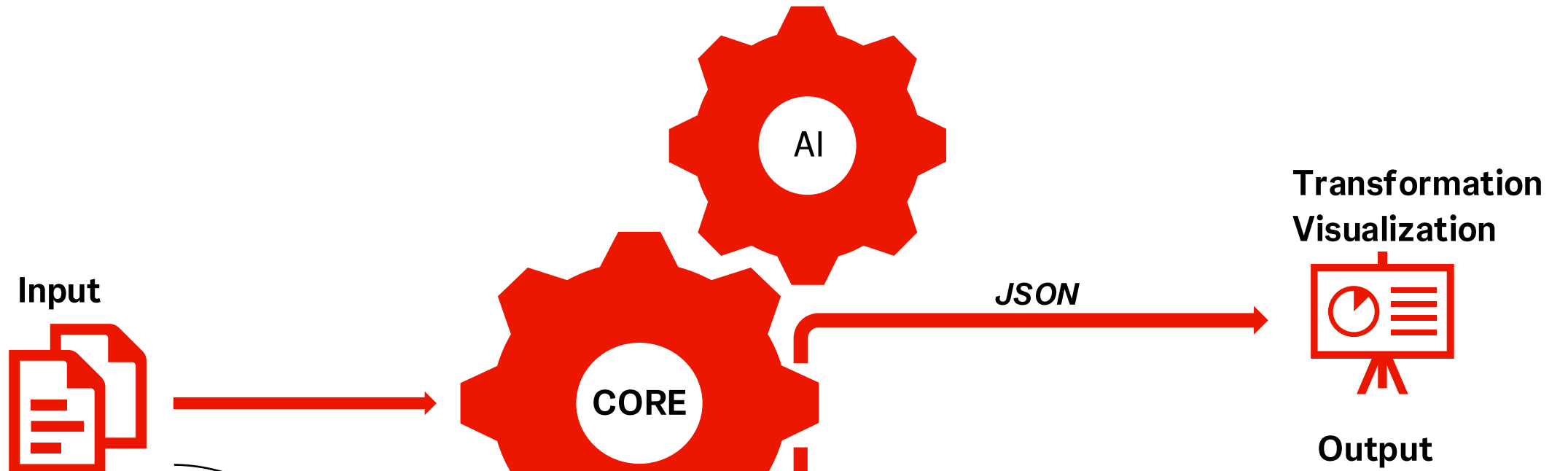
1 DRM_AE.AECAT.ASSIGN("AE OF SPECIAL INTEREST")
✓ 0.0s Python

ASSIGN AECAT:
Assigned value: AE OF SPECIAL INTEREST

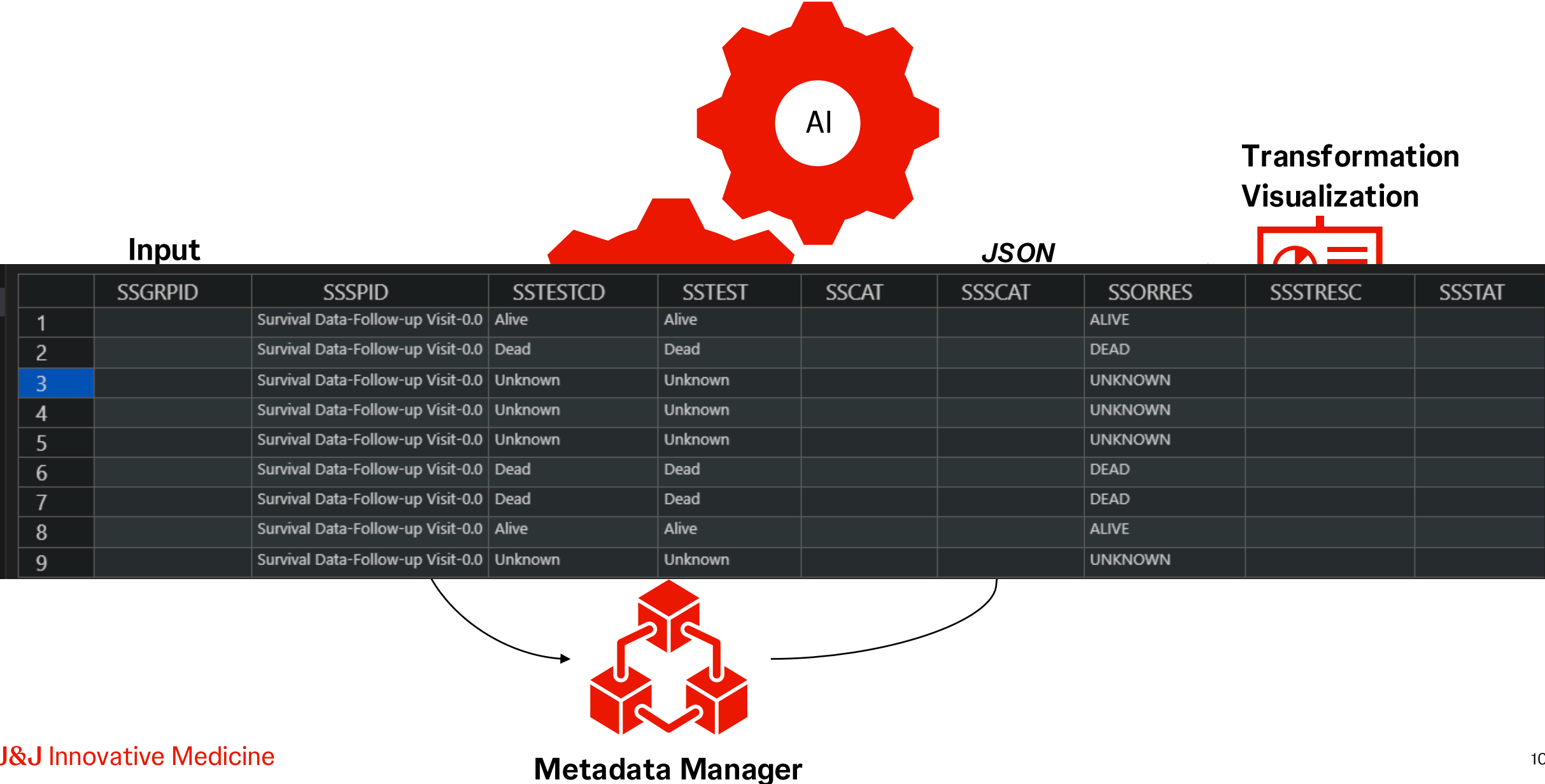
1 DRM_AE.AESEV.RECODE({'Mild': '1',
2                       'Moderate': '2',
3                       'Severe': '3'},
4                       SOURCE.AESEV)
5
✓ 0.0s Python

RECODE AESEV:
Input values: ['Severe', 'Mild', 'Moderate', '']
Output values: ['3', '1', '2']
Mapping: {'Mild': '1', 'Moderate': '2', 'Severe': '3'}
```

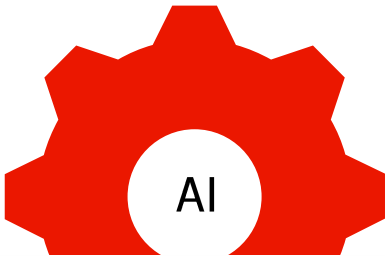
Framework architecture design - AI



Framework architecture design - Output



Framework architecture design - Output



Transformation

Input



SAS / CSV / L

TITLE: Survival Data

RAVE.SS_GL_900

SSDAT	Collection Date		1	DRM_FA	FADTC
			2	DRM_SS	SSDTC
SURVSTAT	Survival status	☐ Alive ☐ Dead ☐ Unknown	2	DRM_SS	SSORRES_DRM SSORRES
SSLKADAT	Last date known to be alive		2	DRM_SS	SSLKADTC
SSINFSRC	If survival status is dead, how was this information obtained?	☐ Public/Medical Records (if allowed per Local Regulations) ☐ Contact with relatives	2	DRM_SS	SSINFSRC SSINFSRC_DRM
OCCUR_NMAL	Was any new malignancy diagnosed since last visit?	☐ Yes ☐ No ☐ Unknown	1	DRM_FA	FAORRES_DRM FAORRES FASTAT FAREASND



Proof-of-Concept outcome

SAS + CSV datasets tested (~100K records)

Validated ingestion of heterogeneous formats.

Performance tested for medium-scale datasets.

Automatic metadata profiling

Source dataset variable detection and types.

Generates transformation-ready metadata without manual coding.

AI Assistant for transformation tasks

Code generation from plain-language requests.

Intelligent variable mapping suggestions.

Connection to Visualization tools

JSON lineage exported for source-to-target visualization tools.

Framework-ready for visual dashboards.

Lessons learned

Metadata Importance

Rich metadata is essential for effective AI-powered suggestions and automated transformations.

Comprehensive metadata management is foundational to framework success.

Generative AI Integration

Integrating large language models for code generation requires careful prompt engineering and validation of generated code.

Modular Design Benefits

Separating AI capabilities into distinct modules allows independent development and testing. This approach enables gradual feature rollout without disrupting core functionality.

Performance Optimization

Processing large clinical datasets efficiently requires careful memory management and algorithm optimization. Streaming approaches and parallel processing capabilities are essential for enterprise-scale deployment.

Thank you