System Architecture and Rule Ingestion

This document outlines the proposed system architecture and rule ingestion approaches for the Python-based reconciliation and validation tool.

System Architecture

The tool will consist of three main components:

1. **Excel Configuration and Output File:** A single Excel workbook will serve as the user interface for configuration and for presenting the output. It will contain the following sheets:
    - `Config` : For global settings, including paths to data sources (e.g., CSV, Excel, database connection strings).
    - `Rules` : To define the reconciliation and validation rules.
    - `Output` : To store the results of the checks in a structured format.
    - `Log` : To record execution details, such as timestamps, errors, and a summary of the checks performed.
2. **Python Reconciliation Engine:** The core of the tool, this Python script will orchestrate the entire process:
    - Read the configuration and rules from the Excel file.
    - Load the datasets to be reconciled and validated.
    - Parse and execute the rules.
    - Generate a timestamped copy of the Excel file with the `Output` and `Log` sheets populated.
3. **Rule Ingestion Module:** This component will be responsible for interpreting the rules defined in the `Rules` sheet. We propose a hybrid approach for maximum flexibility and user-friendliness.

Rule Ingestion Approaches

We recommend a hybrid approach that combines a simple Domain Specific Language (DSL) with optional GenAI-powered natural language rule interpretation.

1. Structured Rule Definition using a DSL

This will be the primary method for defining rules. The `Rules` sheet in the Excel file will have a predefined structure with columns that allow users to specify checks in a clear and

unambiguous way. This approach is robust and easy to parse.

Example Rule Structure:

RuleID	RuleType	Source1	Key1	Source2	Key2	CheckType	Column/Value1	Co Va
1	Reconciliation	source1.csv	id	source2.csv	id	equals	amount	t o n
2	Validation	source1.csv				greater_than	amount	0
3	Validation	source1.csv				is_in	category	["I

2. GenAI-Powered Natural Language Rules (Optional)

To enhance user experience, we can integrate the Claude API to allow users to write rules in plain English. This would be an optional feature that can be enabled in the `Config` sheet.

Workflow:

1. The user writes a rule in a dedicated column in the `Rules` sheet, e.g., `NaturalLanguageRule`.
2. The Python engine sends this natural language rule to the Claude API.
3. The API call will include a prompt that instructs the model to convert the natural language rule into our structured DSL format.
4. The Python engine will then validate and execute the structured rule returned by the API.

Example:

- **User Input (Natural Language):** "For each record in `source1.csv`, the `amount` should be the same as the `transaction_amount` in `source2.csv` where the `id`s match."
- **GenAI Output (Structured DSL):** The GenAI would generate a rule in the format described in the table above.

This hybrid approach provides the best of both worlds: the structure and reliability of a DSL for complex or critical checks, and the ease of use of natural language for simpler or ad-hoc rules.

Next, I will proceed with creating synthetic data to test this proposed system.