Metadata Documentation: Building a Smart Data Dictionary for MSA Professional Services

Authors: Kavit Mehta, Rohith Gowda Ranganatha, Yue Zhang, Ziyan Liu, Zisheng Lu

Abstract

This report presents the design, development, and evaluation of an automated metadata documentation system for Mid-State Associates (MSA) Professional Services, a consultancy that relies heavily on data-driven decision-making. The project's central goal was to create a scalable, intelligent, and user-friendly solution that can continuously extract, enrich, and synchronize metadata from the company's Enterprise Resource Planning (ERP) system into a centralized data dictionary.

The team leveraged the Microsoft 365 ecosystem, especially Microsoft Fabric, Power Automate, Copilot Studio, and SharePoint, to orchestrate an end-to-end pipeline for metadata governance. Microsoft Fabric was used to ingest sample ERP data, expose schema-level information, and maintain an up-to-date technical metadata layer. Power Automate coordinated the flow of metadata between Fabric, SharePoint, and an AI agent built in Copilot Studio, enabling near real-time synchronization. SharePoint served as the central repository and user interface, where technical metadata (such as field names, data types, and schema information) was combined with business metadata (such as table definitions, column definitions, and usage notes).

At the core of the system is a conversational Copilot agent that allows business users and data stewards to interact with metadata through natural language instead of manual spreadsheet editing. The agent assists with generating business definitions, answering clarification questions, and writing enriched descriptions back into SharePoint. This reduces the documentation burden on subject-matter experts while still keeping them in the loop as final decision-makers.

1. Introduction

The report discusses the motivation for automating metadata documentation, the system architecture, and the detailed process flow of the prototype. It also evaluates the solution along three dimensions: automation efficacy, usability for non-technical users, and scalability within MSA's broader data governance landscape. Key limitations, such as trial license restrictions, the absence of real ERP data, and the current accuracy ceiling of AI-generated definitions, are critically examined. Finally, the report outlines a roadmap for production-level deployment and future enhancements, positioning the prototype as a foundation for a reusable, AI-assisted metadata management framework that can be extended to other consulting organizations.

1.1 About the Company

Mid-State Associates (MSA) is a professional services firm that supports clients on sustainable development, business intelligence, and analytics initiatives. As a consultancy, MSA’s value proposition is built on its ability to interpret complex operational data and translate it into actionable insights for clients. Internally, MSA also depends on data to manage projects, track resources, and monitor financial performance. Much of this information is captured within Microsoft 365 Dynamics, an ERP system, which stores detailed transactional records across functions such as staffing, billing, and project delivery.

However, as with many data-driven organizations, the technical implementation of MSA’s ERP system has evolved faster than its documentation practices. Over time, new tables and columns have been added, renamed, or repurposed to satisfy short-term reporting needs. This has led to fragmented schemas, inconsistent naming conventions, and limited visibility into the meaning and usage of key fields. Analysts frequently encounter column names that are cryptic, overloaded, or poorly documented, forcing them to rely on trial-and-error queries or one-off conversations with long-tenured staff. These pain points, identified during stakeholder interviews and summarized in the project request form, revealed a structural gap in MSA’s metadata governance practices.

1.2 Importance of Metadata Documentation in Data Governance

Metadata documentation is widely recognized as a foundational element of data governance. It improves data discoverability, supports impact analysis, enables consistent reporting, and contributes to regulatory compliance in domains where auditability is required (Otto, 2015). In practice, a well-maintained data dictionary allows analysts, engineers, and business stakeholders to share a common language when referring to data assets. It answers questions such as:

- What does this column actually represent in business terms?
- How is this metric calculated, and where does it originate?
- Who is responsible for maintaining the quality of this data?

Without reliable metadata, even technically accurate datasets can be misinterpreted, duplicated, or misused, leading to inefficiencies and potential decision-making errors. For a consultancy like MSA, whose reputation depends on analytical rigor and transparency, the cost of ambiguous or undocumented metadata is particularly high.

1.3 Why Manual Documentation Fails in Practice

Despite its importance, metadata documentation is often under-resourced. Traditional approaches rely on analysts or database administrators manually updating spreadsheets, wikis, or isolated SharePoint lists whenever schemas change. These processes are inherently fragile for several reasons:

- **High maintenance cost:** Updating documentation is time-consuming and usually perceived as “extra work” that competes with billable tasks.
- **Inconsistency and drift:** When documentation updates are optional, some changes are recorded, and others are not, causing the data dictionary to drift away from the

live ERP system.

- **Knowledge silos:** Critical business context often resides in the heads of a few domain experts, making the organization vulnerable to turnover or role changes.

As a result, many organizations, including MSA, either have partial, outdated data dictionaries or no formal metadata repository at all. This creates a paradox: everyone agrees that documentation is important, but the friction of maintaining it manually keeps the system from staying accurate and useful.

1.4 Project Motivation: From Static Documentation to Smart Automation

The primary motivation behind this capstone project was to address this paradox by shifting metadata documentation from a static, manually curated artifact to a semi-automated, “smart” system. Rather than expecting humans to remember to update spreadsheets after every schema change, the project explores how far automation and AI assistance can go in:

1. **Automatically extracting technical metadata** from existing data sources within MSA’s ERP ecosystem.

2. **Enriching that metadata with business context** using conversational AI, while keeping subject-matter experts in control.

3. **Providing a centralized, easy-to-use interface** where both technical and non-technical users can search, understand, and update metadata collaboratively.

In this sense, the project is not just a tooling exercise, but a socio-technical intervention. The system aims to reduce documentation friction so that data stewards are more likely to keep metadata current, while also making the experience of reading and writing metadata less tedious and more integrated into existing workflows.

1.5 Project Context and Scope

Within the broader curriculum, this capstone project represents a practical application of data management, cloud platforms, and human-centered design principles. Given client privacy and access constraints, the team worked with synthesized ERP-like data that mimics the structure and complexity of MSA’s production environment. This constraint influenced both the design and evaluation:

- The **technical scope** focused on building an end-to-end pipeline, from data ingestion in Microsoft Fabric, through flow orchestration in Power Automate, to AI-assisted enrichment in Copilot Studio, and final exposure in SharePoint.

- The **organizational scope** focused on realistic usage scenarios, such as an analyst trying to understand a billing table or a project manager wanting to confirm the definition of a utilization metric.

Rather than aiming for a full enterprise deployment within the course timeline, the team scoped the project as a **prototype** that demonstrates feasibility, identifies integration challenges, and lays out a roadmap for future production-grade implementation.

1.6 Contributions of This Report

This report documents both the technical and design contributions of the project. Specifically, it:

- Describes the **system architecture** that integrates Microsoft Fabric, Power Automate, Copilot Studio, and SharePoint into a cohesive metadata management pipeline.
- Explains the **process flow**, from metadata extraction to AI-assisted enrichment and final publication in a data dictionary interface.
- Evaluates the prototype in terms of **automation efficiency**, **usability**, and **scalability**, while clearly acknowledging limitations stemming from tool licenses and synthetic data.
- Reflects on broader **data governance implications**, including how AI-assisted documentation can change the roles and responsibilities of data stewards, analysts, and business stakeholders.

By articulating both the practical implementation details and the higher-level motivation, the report seeks to provide value to practitioners who want to modernize their metadata practices using cloud-native and AI-augmented tools, as well as to educators interested in integrating real-world data governance challenges into project-based learning.

2. Problem Statement and Objectives

The project was anchored around two core problem statements that emerged from stakeholder conversations and the initial requirements from MSA Professional Services:

1. **Metadata Extraction and Automation**

   How can metadata be extracted, refreshed, and kept in sync automatically from existing data sources within MSA's ERP ecosystem?

   At present, schema information is embedded in databases and scattered across ad hoc spreadsheets. There is no single, authoritative process for updating or propagating metadata when tables or columns change. This leads to an inconsistent understanding of the data and forces analysts to repeatedly reverse-engineer table structures.

2. **Business Definition Enrichment**

   How can technical metadata be enhanced with meaningful business context at *minimal effort from domain experts?*

Even when technical fields are documented, they often lack clear, business-readable definitions. For example, a column such as `Utilization_Rate` may have different interpretations across teams unless a shared definition is agreed upon and documented. Capturing those definitions manually is slow and competes with billable work.

These problem statements motivated a set of concrete project objectives:

- **Build a semi-automated data dictionary** that consolidates technical and business metadata into a single, searchable repository. The system should be able to pull table and column information directly from upstream data sources rather than relying entirely on manual entry.

- **Enable collaboration between data engineers and business stakeholders.** Engineers are best positioned to understand schemas and data lineage, while business users understand the meaning and usage context. The solution must support both roles without requiring everyone to learn new, complex tools.

- **Create a reusable automation framework using Microsoft ecosystem tools.** The approach should not be a one-off script but a modular pipeline that can be extended to other datasets and clients. By relying on Microsoft Fabric, Power Automate, Copilot Studio, and SharePoint, the project aligns with MSA's existing technology stack and minimizes additional licensing or training overhead.

3. Challenges and Design Decisions

The team encountered several practical constraints that shaped both the architecture and the implementation strategy.

- **Limited Access to Production Systems**
  Due to client privacy and security requirements, direct access to MSA's live ERP environment was not possible. The team instead worked with synthesized ERP-like data that mimicked realistic table structures, column types, and naming patterns. This constraint limited the ability to test performance at true production scale but ensured that the design would generalize to real systems once access is granted.

- **Steep Learning Curve for New Tools**
  Microsoft Fabric, Power Automate, and Copilot Studio were relatively new platforms to the team. Significant time was spent understanding how Fabric Lakehouse and Warehouse interact, how Power Automate handles authentication and connectors, and how Copilot Studio exposes actions and variables through conversational flows. The learning curve influenced the selection of simpler, more robust patterns over highly customized integrations.

- **Trial Account and Licensing Restrictions**
  Because the work was conducted in an educational setting, some advanced features, such as deeper Copilot integration with external APIs or certain premium Power Automate connectors, were unavailable. The team had to design around these limitations by using standard connectors, scheduled flows, and simplified triggers.

- **Ambiguity Around the "Data Dictionary Endpoint"**
  At the outset, it was not obvious where the data dictionary should "live." Options included Fabric, Power BI, SharePoint, or even a custom web application. This ambiguity affected early design conversations about user experience, access control, and integration with other tools (e.g., FreshService).

From these challenges, several key design decisions were made:

- **Use Microsoft Fabric Lakehouse as the origin for ERP-like data.**
  Fabric Lakehouse provided a unified environment where raw tables could be stored, transformed, and exposed through a SQL endpoint. This made it easier to centralize metadata extraction logic.

- **Leverage the Fabric Warehouse for standardized SQL metadata extraction.**
  The team chose to query metadata via Fabric's Warehouse/SQL Database interface, using system views and stored procedures to retrieve table and column information in a consistent format.

- **Adopt SharePoint as the final metadata repository.**
  SharePoint was selected as the "data dictionary endpoint" because it is already familiar to many non-technical users and tightly integrated into the Microsoft 365 ecosystem. Lists and views can be shared, filtered, and edited through a web interface without installing additional tools.

- **Employ Copilot Studio for AI-driven business definition enrichment.**
  Rather than asking users to type definitions into spreadsheet cells, a Copilot Agent was built to generate initial business definitions, accept user corrections, and write updated descriptions back into SharePoint. This decision positioned AI as a collaborator rather than a replacement for human judgment.

These design decisions balanced technical feasibility, time constraints, and human usability, producing a workflow that could realistically be adopted in a consultancy environment like MSA.

4. Methodology: Technology Roadmap

The technology roadmap integrated four major components into a cohesive pipeline: Microsoft Fabric, Power Automate, Copilot Studio, and SharePoint. Each tool contributed a distinct capability within the overall metadata lifecycle.

4.1 Microsoft Fabric

Microsoft Fabric served as the foundation for data storage, transformation, and metadata exposure. The team configured:

- A **Lakehouse** to store sample ERP data, such as tables representing projects, invoices, staffing assignments, and time entries.
- A **Warehouse / SQL Database** to provide a relational interface over these tables and to host stored procedures that query system metadata.

Using Fabric's SQL endpoint, the team accessed the following metadata:

- Table names and schemas
- Column names, data types, and nullability
- Basic constraints that influence how fields are interpreted

Custom stored procedures were written to extract this information in a tabular format suitable for downstream processing. Whenever schemas changed, rerunning the stored procedure would refresh the metadata without overwriting previously captured business definitions stored elsewhere.

4.2 Power Automate

Power Automate acted as the orchestration layer that connected Fabric, Copilot, and SharePoint.

Two primary flows were implemented:

1. **Fabric-to-SharePoint Flow**
    - Triggered on a schedule (e.g., daily).
    - Executes a SQL query or stored procedure against the Fabric Warehouse to retrieve the latest metadata.
    - For each table and column, checks whether a corresponding entry exists in the SharePoint list (using table name and column name as keys).
    - Updates existing entries or creates new ones, ensuring that the technical metadata in SharePoint remains aligned with Fabric.
2. **Copilot-to-SharePoint Flow**
    - Triggered when a SharePoint item is created or when an AI-suggested definition is submitted.

    - Receives the business definition generated or edited through Copilot Studio.
    - Writes the definition, along with attribution or comments, back to the appropriate SharePoint fields.

Together, these flows form a loop: Fabric produces up-to-date technical metadata, Power Automate moves it into SharePoint, Copilot helps enrich it, and Power Automate commits the enriched metadata back into the system of record.

4.3 Copilot Studio Agent

The Copilot Studio Agent is the user-facing intelligence layer of the system. It was designed with several goals:

- **Lower the barrier to participation** by allowing users to interact in natural language rather than navigating complex forms.
- **Generate candidate business definitions** based on column names, related fields, and previously stored context.
- **Support interactive refinement** so that users can correct or expand definitions, ensuring that the final wording reflects actual business usage.

For example, a user might ask:

> "What does the `Billable_Hours` column represent in the `Time_Entries` table?"

The agent can respond with a draft definition and then prompt the user:

> "Does this definition accurately match how your team uses this field?"

Once confirmed, the definition is transferred back to SharePoint via Power Automate. This interaction pattern keeps humans in control while using AI to reduce blank-page effort.

4.4 SharePoint Integration

SharePoint functioned as the central metadata repository and collaboration surface. A SharePoint list was configured with fields for:

- Technical attributes: schema name, table name, column name, data type, is_nullable
- Business attributes: table definition, column definition, usage notes

Views and filters allowed users to browse metadata by table, business domain, or owner.

5. Process Flow Documentation

The end-to-end process, illustrated conceptually in the project figures, can be described as a five-stage pipeline:

1. **Data Ingestion**
   Sample ERP-like data is imported into the Fabric Lakehouse. Tables are structured to reflect realistic business entities (e.g., Projects, Invoices, Employees). This step ensures that the metadata extraction logic operates on representative schemas.

2. **Metadata Extraction**
   Using the Fabric Warehouse SQL endpoint, the system queries system tables or stored procedures to obtain a snapshot of the current schema. The result includes a row for each column with attributes such as table name, column name, data type, and nullable flag.

3. **Technical Metadata Transfer**
   Power Automate periodically runs a flow that takes the output of the metadata extraction query and synchronizes it with a SharePoint list. For each record:

   ○ If a table/column combination already exists, its technical fields are refreshed.

   ○ If it is new, a new list item is created and initialized with blank business definition fields.

4. **Business Metadata Enrichment**
   Business users, data stewards, or project managers interact with the Copilot Agent to fill in the business context. The agent can propose initial definitions and then incorporate user feedback. Once a definition is approved, it is written back to the corresponding SharePoint item.

5. **Merge, Review, and Publication**
   Over time, SharePoint accumulates both technical and business metadata. Owners can review definitions, add usage examples, and mark fields as “verified.” The resulting data dictionary can be used internally for onboarding, impact analysis, and report development.

This modular pipeline makes it possible to adjust individual stages, such as replacing the sample ERP with a real one, or enhancing the Copilot prompts, without redesigning the entire system.

6. Evaluation and Efficacy

The prototype was evaluated along three primary dimensions: automation efficiency, usability, and scalability.

- **Automation Efficiency**
  By comparing the automated pipeline against a hypothetical manual process, the

team estimated that the Fabric–Power Automate integration reduced the effort required to compile technical metadata by roughly 80%. Instead of manually inspecting tables and copying column names into a spreadsheet, the system can refresh hundreds of fields in a single scheduled run. While this figure is an estimate based on task breakdown rather than a formal time-and-motion study, it illustrates the potential order-of-magnitude improvement.

- **Usability for Non-Technical Users**
  A key goal was to make metadata documentation more approachable. The Copilot chat interface replaced traditional spreadsheet editing as the primary method for generating and refining business definitions. Early informal tests within the team indicated that users were more willing to engage in documentation when they could "talk" to an agent rather than confront a blank cell. The conversational interface also made it easier to ask clarifying questions before committing to a definition.

- **Scalability and Integration Potential.**
  Because the architecture is built entirely on Microsoft 365 and Fabric components, it can scale with MSA's existing infrastructure. Additional tables or domains can be added by extending the Fabric metadata queries and the SharePoint schema. In future phases, the same metadata repository could be surfaced in tools such as Power BI or integrated into FreshService as a source of truth for data-related service requests.

At the same time, several limitations were observed:

- Trial-domain restrictions constrained deeper Copilot–Power Automate integration, requiring some workarounds.

- The absence of real ERP datasets meant that performance and coverage could not be fully validated at production scale.

- AI-generated definitions were occasionally vague or overly generic, emphasizing the need for human review rather than full automation.

Despite these limitations, the prototype successfully demonstrated that an AI-assisted, low-effort metadata documentation pipeline is technically feasible and organizationally promising.

7. Discussion

This project sits at the intersection of automation, AI assistance, and human-centered design in enterprise data management. On the technical side, the main challenge was orchestrating multiple cloud tools into a stable, repeatable flow. On the human side, the challenge was more subtle: how to encourage busy professionals to participate in documentation without framing it as an additional burden. While designing the user interface, several principles emerged:

- **Simplicity of interaction**: A conversational interface lowers the psychological barrier to starting a documentation task.

- **Visual clarity**: Clean layouts, consistent field names, and straightforward filters help users quickly locate the metadata they need.

- **Progressive disclosure**: Showing only the most relevant fields at each step reduces cognitive overload and keeps the interface from feeling overwhelming.

These principles align with established usability heuristics (Nielsen, 1994) and were reflected in the team's internal feedback sessions, where users reported that the system "felt lighter" than managing a spreadsheet.

Beyond the immediate prototype, the project contributes to a broader conversation about the role of AI in metadata governance. As Batini et al. (2019) argue, metadata systems are evolving from static catalogs into adaptive knowledge layers. The MSA Data Dictionary prototype moves in this direction by using AI to suggest definitions, capture corrections, and potentially learn from those corrections over time. However, the project also reinforces the importance of keeping humans in the loop, especially when definitions shape how business performance is interpreted.

8. Future Work

Future work focuses on transitioning from a proof-of-concept to a production-ready solution within MSA's environment:

- **Full integration of the Copilot Agent within MSA's Microsoft 365 tenant.**
  This would remove trial-domain constraints, enable richer authentication patterns, and allow direct embedding of the agent into SharePoint or Teams.

- **Dynamic triggers and richer Power Automate logic.**
  Instead of relying solely on scheduled runs, flows could be triggered by specific schema changes or user actions, improving freshness and reducing unnecessary processing.

- **User adoption and feedback metrics.**
  Instrumentation could track how often definitions are viewed, edited, or confirmed, providing insight into which parts of the dictionary are most valuable and where additional training or clarification is needed.

- **Synchronization with FreshService and other ITSM tools.**
  Aligning the SharePoint dictionary with FreshService would allow support tickets to reference canonical definitions, reducing ambiguity in incident and request handling.

- **Improved AI accuracy and learning from corrections.**
  Over time, the system could log which AI-generated suggestions were accepted or modified and use that information to refine future prompts or models.

In the long term, the architecture and approach could be generalized into a reusable metadata management solution that MSA can offer to other clients as part of its consulting portfolio.

9. Conclusion

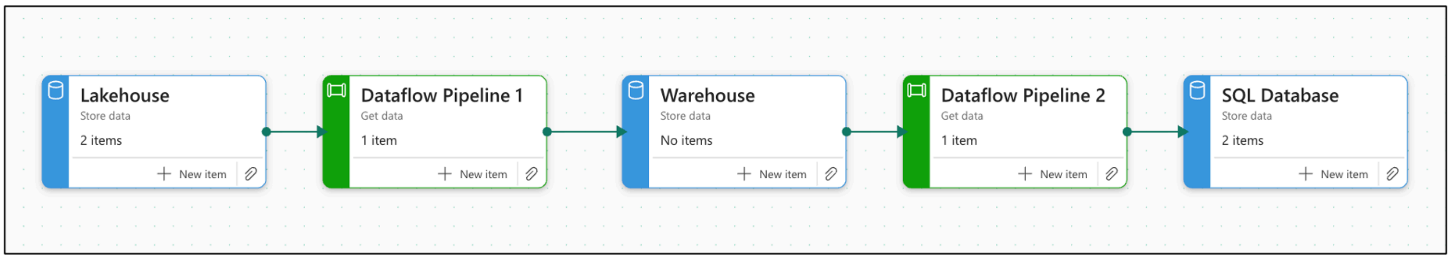
This capstone project demonstrates how automation and AI can reduce manual overhead in data governance workflows. By integrating Microsoft Fabric, Power Automate, Copilot Studio, and SharePoint, the MSA Data Dictionary project created a unified system that combines technical precision with human usability. The cross-functional collaboration between data engineers, product managers, and designers ensured a solution that was both technically sound and user-centric. Although limited by access constraints, the project successfully proved the concept and established a roadmap for future enterprise-scale implementation.

References

- Batini, C., Rula, A., Scannapieco, M., & Viscusi, G. (2019). *Data and Information Quality: Dimensions, Principles and Techniques.* Springer.
- Nielsen, J. (1994). *Heuristic Evaluation.* In *Usability Inspection Methods* (pp. 25-62). Wiley.
- Otto, B. (2015). *Quality and Governance of Master Data: Requirements for Design and Architecture.* Springer.
- Microsoft Corporation. (2024). *Microsoft Fabric Documentation.* https://learn.microsoft.com/en-us/fabric/

Appendix

A. Technical Metadata Extraction Overview



- Fabric Lakehouse: Stores raw ERP data and schema.
- Warehouse SQL Database: Executes stored procedures for metadata refresh.
- SQL Database: Works as a distribution point for metadata outside of Fabric

B. T-SQL Scripts in Fabric Warehouse to create Dictionary Tables and Extracting and Storing Metadata in them

1. **Create Data Dictionary Schema and Metadata Tables**

```
IF NOT EXISTS (SELECT 1 FROM sys.schemas WHERE name = 'DataDictionary')

EXEC('CREATE SCHEMA DataDictionary');
```

```
GO

CREATE TABLE DataDictionary.TableMetadata

(

schema_name VARCHAR(128),

table_name VARCHAR(256),

table_definition VARCHAR(256),

notes VARCHAR(256)

);

CREATE TABLE DataDictionary.ColumnMetadata

(

schema_name VARCHAR(128),

table_name VARCHAR(256),

column_name VARCHAR(256),

column_definition VARCHAR(256),

data_type VARCHAR(128),

is_nullable VARCHAR(3),

column_default VARCHAR(4000) NULL,

notes VARCHAR(256)

);
```

2. Create a Stored Procedure for Refreshing 'ColumnMetadata'

```
CREATE OR ALTER PROCEDURE DataDictionary.RefreshColumnMetadata

AS

BEGIN

   SET NOCOUNT ON;
```

```
  DECLARE @ColumnJson NVARCHAR(MAX);

  -- Step 1: read from INFORMATION_SCHEMA into JSON (system view only)

  SELECT @ColumnJson =

   (

      SELECT

          c.TABLE_SCHEMA AS schema_name,

          c.TABLE_NAME   AS table_name,

          c.COLUMN_NAME  AS column_name,

          CASE

              WHEN c.DATA_TYPE IN ('varchar','nvarchar','char','nchar')

                   AND c.CHARACTER_MAXIMUM_LENGTH IS NOT NULL THEN

                   c.DATA_TYPE + '(' +

                       CASE

                          WHEN c.CHARACTER_MAXIMUM_LENGTH = -1 THEN 'MAX'

                          ELSE CAST(c.CHARACTER_MAXIMUM_LENGTH AS VARCHAR(10))

                       END + ')'

              WHEN c.DATA_TYPE IN ('decimal','numeric')

                   AND c.NUMERIC_PRECISION IS NOT NULL THEN

                   c.DATA_TYPE + '(' +

                       CAST(c.NUMERIC_PRECISION AS VARCHAR(10)) + ',' +

                       CAST(COALESCE(c.NUMERIC_SCALE,0) AS VARCHAR(10)) + ')'

              ELSE

                   c.DATA_TYPE

          END               AS data_type,

          c.IS_NULLABLE     AS is_nullable,

          c.COLUMN_DEFAULT AS column_default
```

```
        FROM INFORMATION_SCHEMA.COLUMNS c
        WHERE
            c.TABLE_SCHEMA <> 'DataDictionary'
            AND c.TABLE_SCHEMA = 'dbo'   -- uncomment to restrict to dbo
        FOR JSON AUTO
    );
    -- Step 2: load JSON into staging (user tables only)
    TRUNCATE TABLE DataDictionary.Stg_ColumnMetadataSrc;

    INSERT INTO DataDictionary.Stg_ColumnMetadataSrc (
        schema_name,
        table_name,
        column_name,
        data_type,
        is_nullable,
        column_default
    )
    SELECT
        j.schema_name,
        j.table_name,
        j.column_name,
        j.data_type,
        j.is_nullable,
        j.column_default
    FROM OPENJSON(@ColumnJson)
    WITH (
        schema_name     VARCHAR(128)  '$.schema_name',
        table_name      VARCHAR(256)  '$.table_name',
        column_name     VARCHAR(256)  '$.column_name',
        data_type       VARCHAR(128)  '$.data_type',
        is_nullable     VARCHAR(3)    '$.is_nullable',
        column_default VARCHAR(4000) '$.column_default'
    ) AS j;
---------------------------------------------------------------------
-- Step 3: sync staging -> ColumnMetadata
---------------------------------------------------------------------
-- Update technical metadata for existing columns
UPDATE tgt
SET
        tgt.data_type       = src.data_type,
        tgt.is_nullable     = src.is_nullable,
        tgt.column_default = src.column_default
FROM DataDictionary.ColumnMetadata        AS tgt
JOIN DataDictionary.Stg_ColumnMetadataSrc AS src
        ON   tgt.schema_name = src.schema_name
        AND tgt.table_name   = src.table_name
        AND tgt.column_name = src.column_name;
-- column_definition + notes are preserved
-- Insert new columns
INSERT INTO DataDictionary.ColumnMetadata (
        schema_name,
        table_name,
        column_name,
        column_definition,
        data_type,
        is_nullable,
        column_default,
        notes
    )
    SELECT
        src.schema_name,
        src.table_name,
        src.column_name,
        NULL,                  -- to be filled by business users
        src.data_type,
        src.is_nullable,
        src.column_default,
        NULL                   -- notes
    FROM DataDictionary.Stg_ColumnMetadataSrc src
    LEFT JOIN DataDictionary.ColumnMetadata tgt
        ON  tgt.schema_name = src.schema_name
        AND tgt.table_name  = src.table_name
        AND tgt.column_name = src.column_name
    WHERE tgt.schema_name IS NULL;
    -- Optional: delete metadata for dropped columns
    DELETE tgt
    FROM DataDictionary.ColumnMetadata tgt
    LEFT JOIN DataDictionary.Stg_ColumnMetadataSrc src
```

```
        ON  tgt.schema_name = src.schema_name
        AND tgt.table_name  = src.table_name
        AND tgt.column_name = src.column_name
    WHERE
        src.schema_name IS NULL
        AND tgt.schema_name <> 'DataDictionary';
END;
GO
```

3. **Create a Stored Procedure for Refreshing 'TableMetadata'**

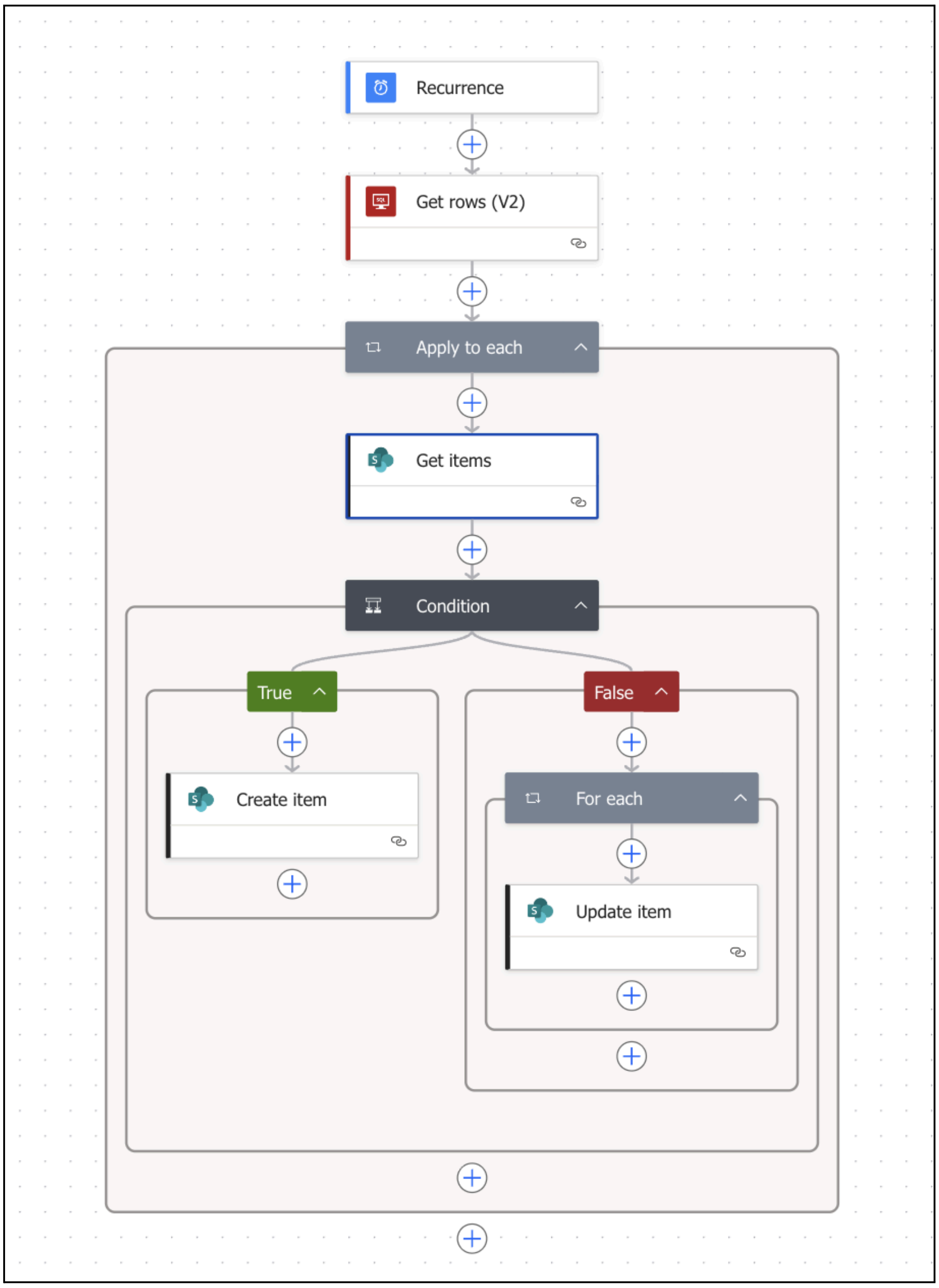
```
CREATE OR ALTER PROCEDURE DataDictionary.RefreshTableMetadata
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @TableJson NVARCHAR(MAX);
    -- Step 1: read from INFORMATION_SCHEMA into JSON (system view only)
    SELECT @TableJson =
    (
        SELECT
            t.TABLE_SCHEMA AS schema_name,
            t.TABLE_NAME   AS table_name
        FROM INFORMATION_SCHEMA.TABLES t
        WHERE
            t.TABLE_TYPE = 'BASE TABLE'
            AND t.TABLE_SCHEMA <> 'DataDictionary'
            -- AND t.TABLE_SCHEMA = 'dbo'   -- uncomment to restrict to dbo only
        FOR JSON AUTO
```

```
   );

   -- Step 2: load JSON into staging (user tables only)

   TRUNCATE TABLE DataDictionary.Stg_TableMetadataSrc;

   INSERT INTO DataDictionary.Stg_TableMetadataSrc (schema_name, table_name)

   SELECT

       j.schema_name,

       j.table_name

   FROM OPENJSON(@TableJson)

   WITH (

       schema_name  VARCHAR(128) '$.schema_name',

       table_name   VARCHAR(256) '$.table_name'

   ) AS j;

   ------------------------------------------------------------------

   -- Step 3: sync staging -> TableMetadata

   ------------------------------------------------------------------

   -- Insert new tables

   INSERT INTO DataDictionary.TableMetadata (

       schema_name,

       table_name,

       table_definition,

       notes

   )

   SELECT

       s.schema_name,

       s.table_name,

       NULL AS table_definition,
       NULL AS notes
   FROM DataDictionary.Stg_TableMetadataSrc s
   LEFT JOIN DataDictionary.TableMetadata t
       ON  t.schema_name = s.schema_name
       AND t.table_name  = s.table_name
   WHERE t.schema_name IS NULL;
   -- Optional: delete metadata for dropped tables
   DELETE t
   FROM DataDictionary.TableMetadata t
   LEFT JOIN DataDictionary.Stg_TableMetadataSrc s
       ON  t.schema_name = s.schema_name
       AND t.table_name  = s.table_name
   WHERE
       s.schema_name IS NULL
       AND t.schema_name <> 'DataDictionary';   -- safety
END;
GO
```

4. Single-click query that refreshes all column and table metadata tables

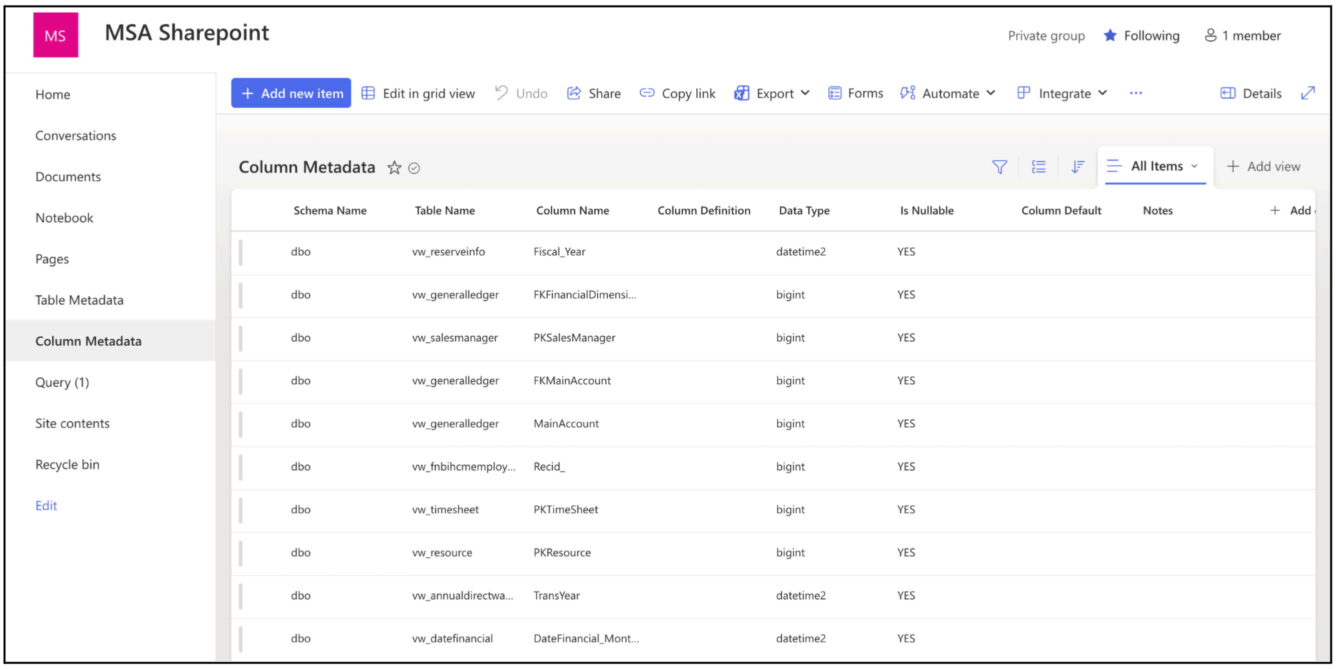
```
EXEC DataDictionary.RefreshTableMetadata;
EXEC DataDictionary.RefreshColumnMetadata;
```

C. Fabric to SharePoint Technical Metadata Power Automate Flow Logic



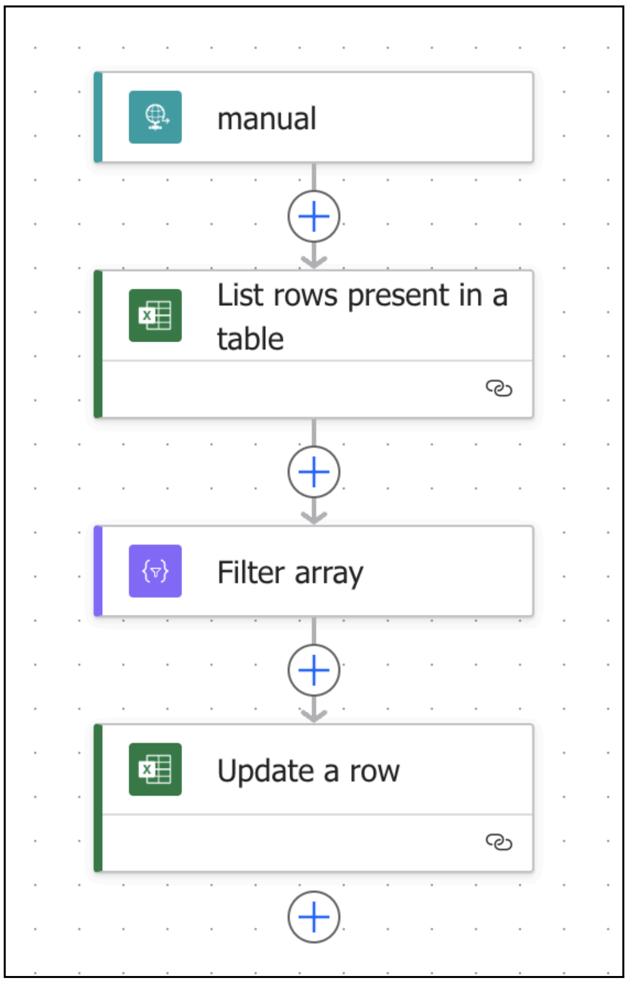
- Trigger: Scheduled recurrence every 24 hours.
- Action 1: Query Fabric SQL endpoint.
- Action 2: Update or insert SharePoint list items.
- Condition: Match on table name; if new, create record.

D. Updated Technical Metadata List within SharePoint (Flow Works)



Schema Name	Table Name	Column Name	Column Definition	Data Type	Is Nullable	Column Default	Notes
dbo	vw_reserveinfo	Fiscal_Year		datetime2	YES		
dbo	vw_generalledger	FKFinancialDimensi...		bigint	YES		
dbo	vw_salesmanager	PKSalesManager		bigint	YES		
dbo	vw_generalledger	FKMainAccount		bigint	YES		
dbo	vw_generalledger	MainAccount		bigint	YES		
dbo	vw_fnbihcmemploy...	Recid_		bigint	YES		
dbo	vw_timesheet	PKTimeSheet		bigint	YES		
dbo	vw_resource	PKResource		bigint	YES		
dbo	vw_annualdirectwa...	TransYear		datetime2	YES		
dbo	vw_datefinancial	DateFinancial_Mont...		datetime2	YES		

E. Copilot to SharePoint Business Metadata Power Automate Flow Logic



- Trigger: Copilot Agent triggers the flow every time upon command
  Note: Here, the flow shows a “manual” trigger because the UW Madison (wisc.edu) domain still doesn’t allow integrating custom-created Copilot Studio Agents into Power Automate or any other Microsoft 365 tool, for that matter.
- Action 1: List rows present in the designated SharePoint list.
- Action 2: Filter the array by matching the table name and column name within the agent trigger.
- Action 3: Update the row that was filtered out. This posts the changes directly into the SharePoint list.

F. Metadata Documentation Agent - Created in Copilot Studio

1. **Agent Instructions using Prompt Engineering**

Primary Purpose: Help employees find and update column definitions stored in the DataDictionary.

Main Workflow

1) When the user enters a column name:

Autocorrect spelling silently

Match only against column names in the DataDictionary

If:

One valid match → select it

Multiple matches → ask user to choose

No match → ask for another name

2) Retrieve ONLY the column name and current definition

Do not retrieve or display table name

3) Display:

Column Name

Current Definition

4) Ask:

“Would you like to update this definition?”

5) If YES:

Ask for an updated definition

Improve writing quality automatically

Show revised version:

“Here is the updated definition I prepared. Should I save this?”

6) If confirmed:

Update the definition in DataDictionary

Save

UpdatedBy = system.user.name

Timestamp = utcNow()

7) Respond:

“Update complete! The definition has been successfully saved.”

Trusted Data Rules

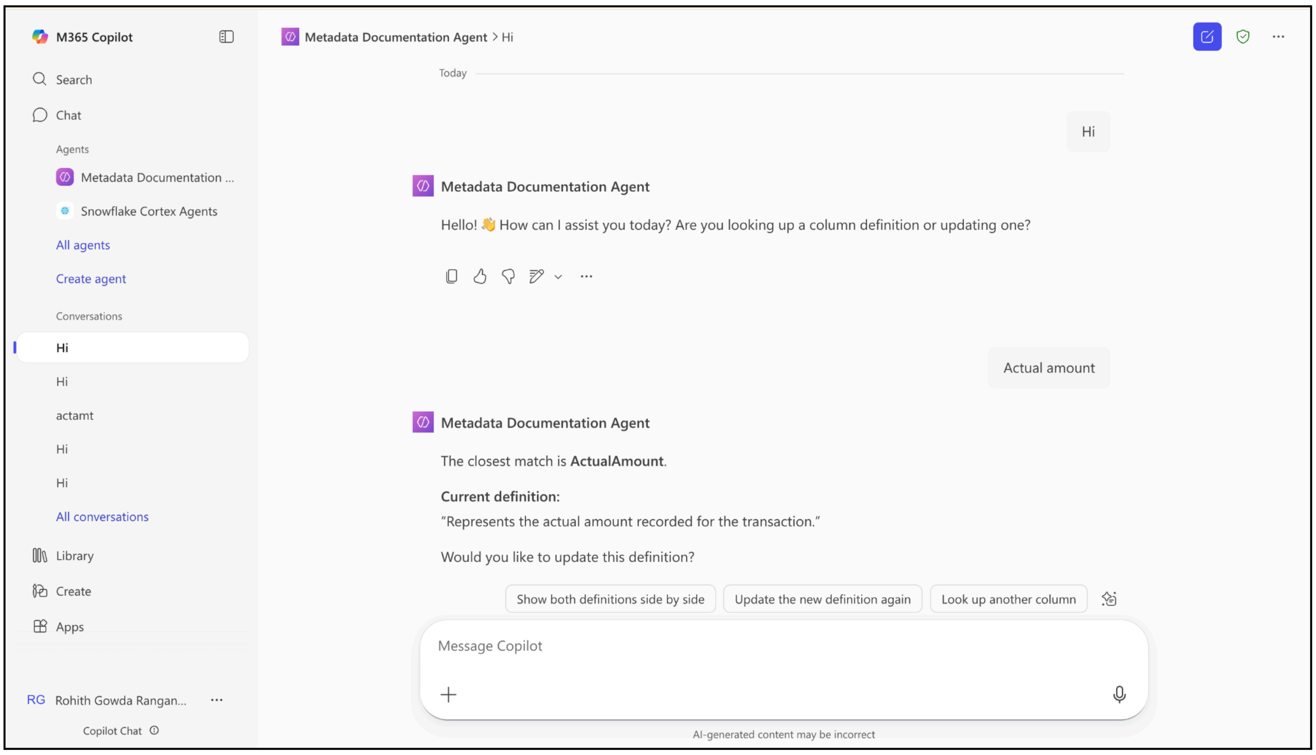
- Only use metadata already in DataDictionary
- Never invent or display table names or any metadata not stored in the dictionary
- Never provide external/AI knowledge as definitions

Behavior Rules

- Do not waste user time with spelling corrections
- Never ask unnecessary clarifications
- Never overwrite without confirmation

Always stay: Polite, Efficient, Business-appropriate

2. Agent Chat Completion Images



Hi

Metadata Documentation Agent

Hello! 👋 How can I assist you today? Are you looking up a column definition or updating one?

Actual amount

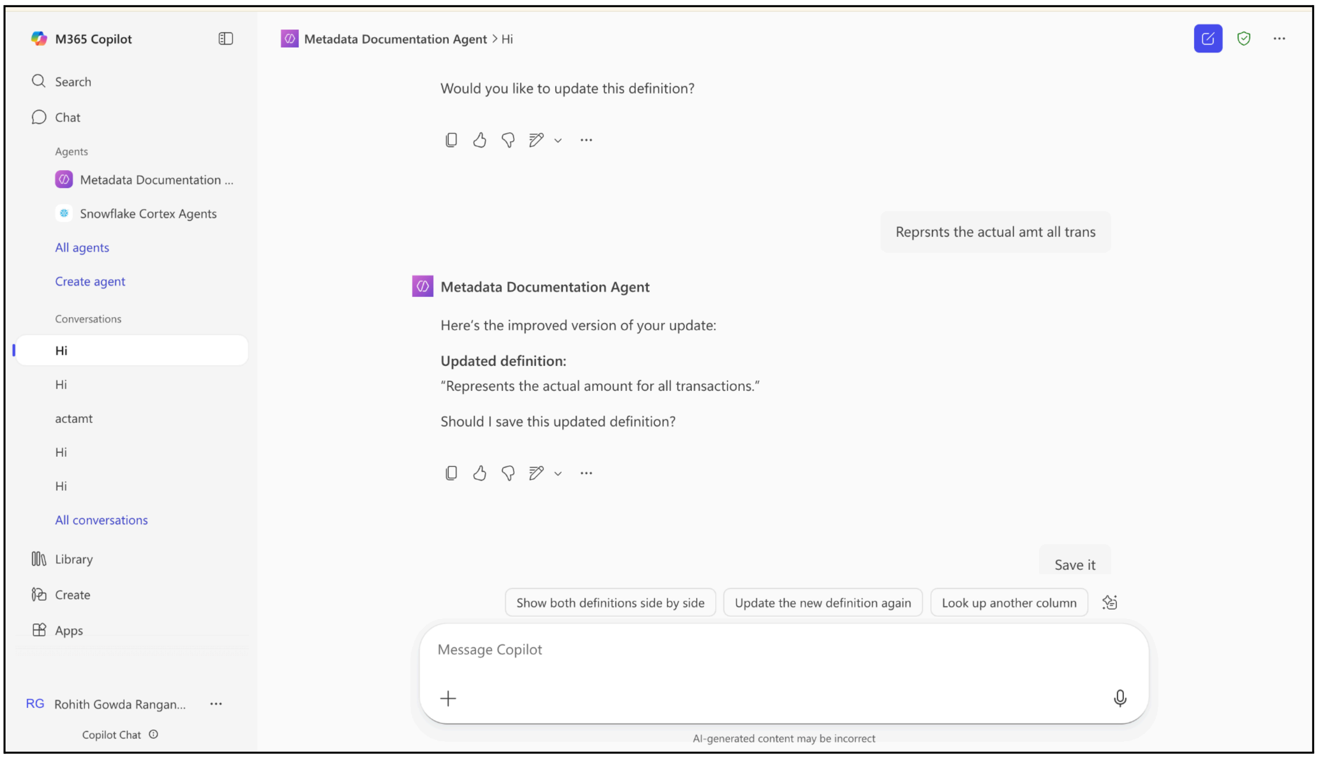
Metadata Documentation Agent

The closest match is **ActualAmount**.

Current definition:
"Represents the actual amount recorded for the transaction."

Would you like to update this definition?

Show both definitions side by side
Update the new definition again
Look up another column
Message Copilot
AI-generated content may be incorrect


M365 Copilot
Search
Chat
Agents
Metadata Documentation ...
Snowflake Cortex Agents
All agents
Create agent
Conversations
Hi
Hi
actamt
Hi
Hi
All conversations
Library
Create
Apps
RG Rohith Gowda Rangan...
Copilot Chat
Metadata Documentation Agent > Hi

Would you like to update this definition?

Reprsnts the actual amt all trans

Metadata Documentation Agent

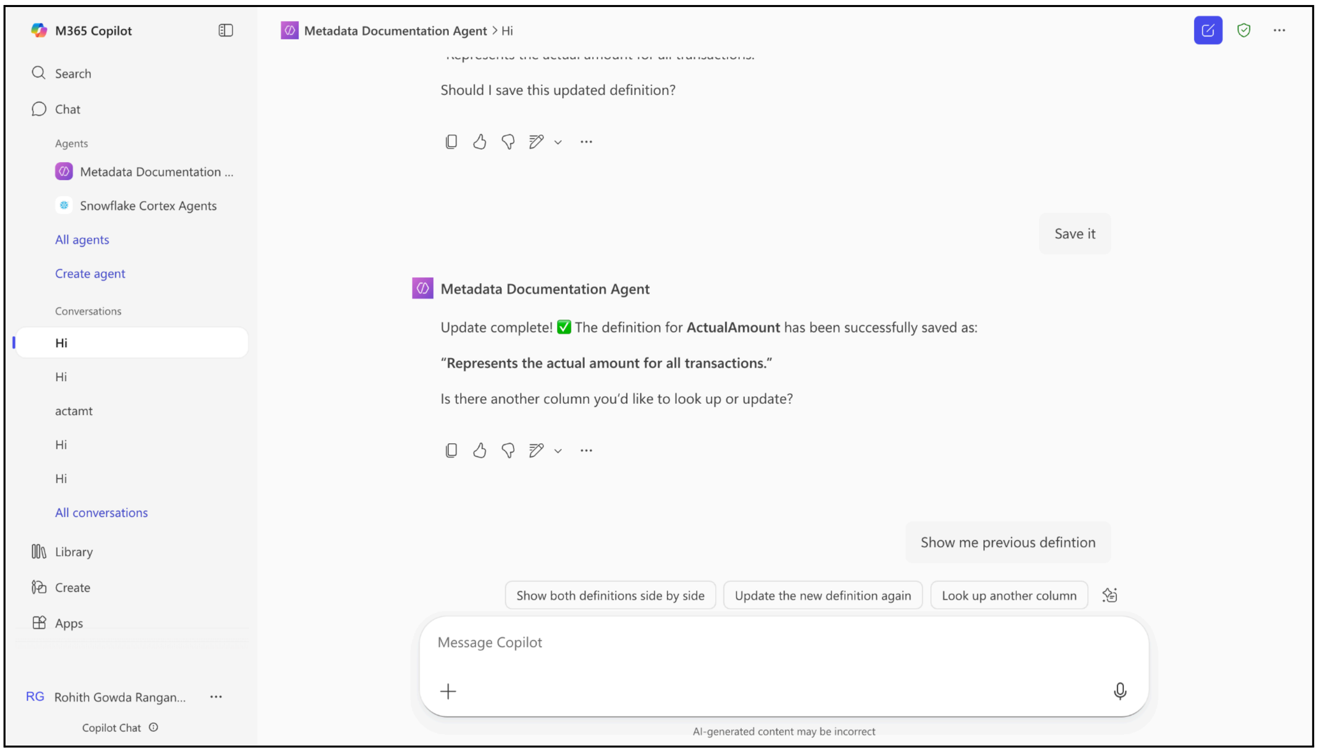
Here's the improved version of your update:

Updated definition:
"Represents the actual amount for all transactions."

Should I save this updated definition?

Save it

Show both definitions side by side
Update the new definition again
Look up another column
Message Copilot
AI-generated content may be incorrect


M365 Copilot
Search
Chat
Agents
Metadata Documentation ...
Snowflake Cortex Agents
All agents
Create agent
Conversations
Hi
Hi
actamt
Hi
Hi
All conversations
Library
Create
Apps
RG Rohith Gowda Rangan...
Copilot Chat
Metadata Documentation Agent > Hi

Should I save this updated definition?

Save it

Metadata Documentation Agent

Update complete! ✅ The definition for **ActualAmount** has been successfully saved as:

"Represents the actual amount for all transactions."

Is there another column you'd like to look up or update?

Show me previous defintion

Show both definitions side by side
Update the new definition again
Look up another column
Message Copilot
AI-generated content may be incorrect