# Creating Fabric Data Agent

With a data agent in Microsoft Fabric, you can create conversational AI experiences that answer questions about data stored in lakehouses, warehouses, Power BI semantic models, KQL databases, and ontologies in Fabric. You can ask questions in plain English and receive data-driven answers grounded in data in Onelake.

**Prerequisites**

- A paid F2 or higher Fabric capacity, or a Power BI Premium per capacity (P1 or higher) capacity with Microsoft Fabric enabled
- Fabric data agent tenant settings is enabled.
- Cross-geo processing for AI is enabled.
- Cross-geo storing for AI is enabled.
- At least one of these, with data: A warehouse, a lakehouse, one or more Power BI semantic models, a KQL database, or an ontology.
- Power BI semantic models via XMLA endpoints tenant switch is enabled for Power BI semantic model data sources.
- Build permissions to the semantic model

In this workshop we will create a data agent for transactional data in a semantic model. With this data agent, user can ask questions regarding financial activity, transactions, account balances, user/category level spend/income etc.

**Scope of the data agent:**

Before creating the data agent, it's a good practice to identify & define the scope of the data agent which will help us configure the data agent to produce accurate responses.

**Purpose**

- Track and explain customer financial behavior: transaction activity (spend/income), account balances by type, and user-level cash flow (incoming – outgoing), including breakdowns by status and category.

**What Data Agent should answer well**

- User/customer analytics: top/bottom users by net cash flow, incoming, outgoing; active users; new users (if filtering on user created date).

- Category analytics: spending by category, income by category, net by category.
- Transaction health: completed vs pending vs failed amounts (and counts if you use transaction rows).
- Account analytics: total balances, balances by account type (checking/savings/credit), credit utilization, accounts per user.

**What it cannot/will not answer**

Time-based questions like “this month / last 30 days / trends over time”.

- Merchant/store/product-level questions (no merchant, store, SKU/product dimensions).
- Profitability, fees, interest, or revenue questions (no fee/interest/revenue fields/measures).
- Operational workflow questions (no timestamps for processing stages, no SLA metrics).
- User identity/profile reporting beyond IDs (no confirmed name/segment fields).

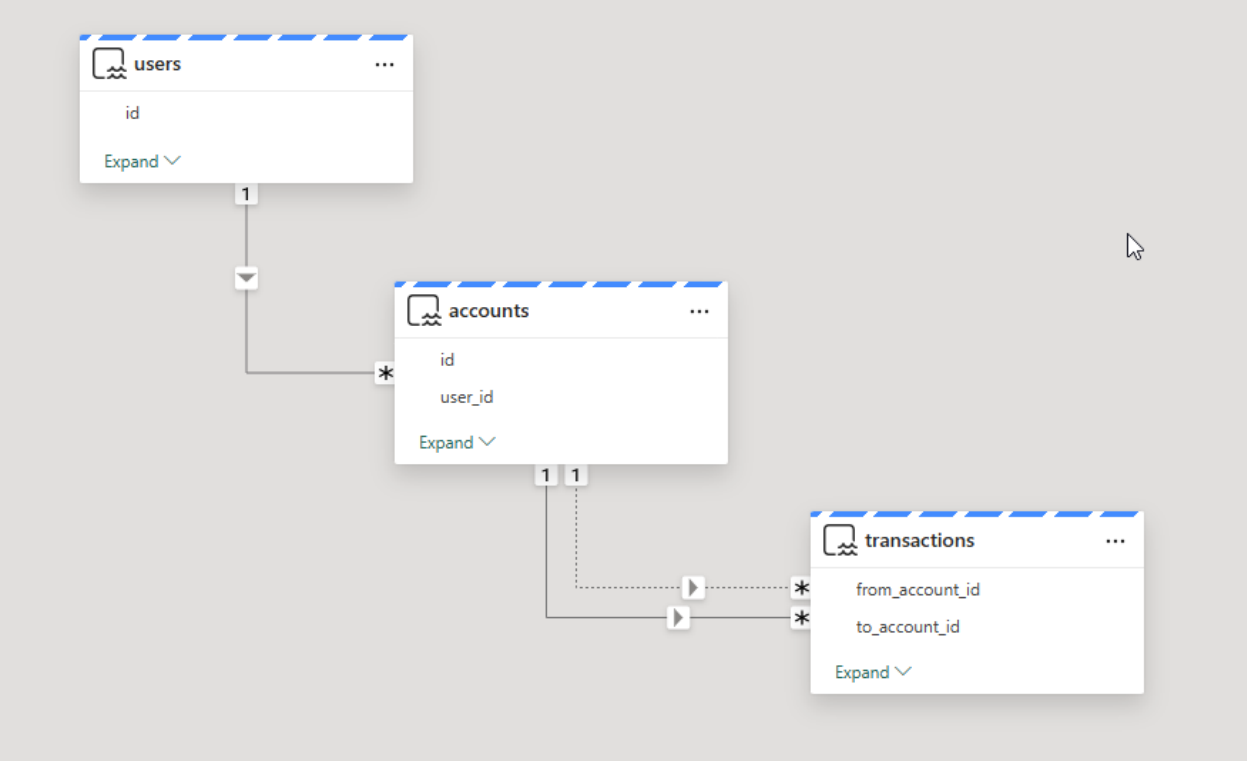
**Intended users / personas**

- *Product managers / growth*: understand customer cash flow patterns, activity, and category behavior.
- *Operations / risk*: monitor transaction status health (failed/pending), unusual net outflows, and credit utilization signals.
- *Finance / analytics*: summarize balances, deposits vs payments, and net movement across the customer base.
- *Support leads (light use):* quick checks of user/account/transaction totals (not case-level troubleshooting without transaction identifiers and timestamps).

**Semantic Model**

The semantic model has three tables that are of interest for the scope of this data agent :

- *users* : customer dimension table
- *accounts* : account dimension table of accounts held by the customers
- *transactions* : fact table of money movement




**Semantic model preparation:**

Before we add the semantic model to the data agent, first the model needs to be prepared for AI use case. This will include:

- Check schema names
- Optimizing the model & DAX for performance
- Creating measures
- Aligning the semantic model with data agent scope

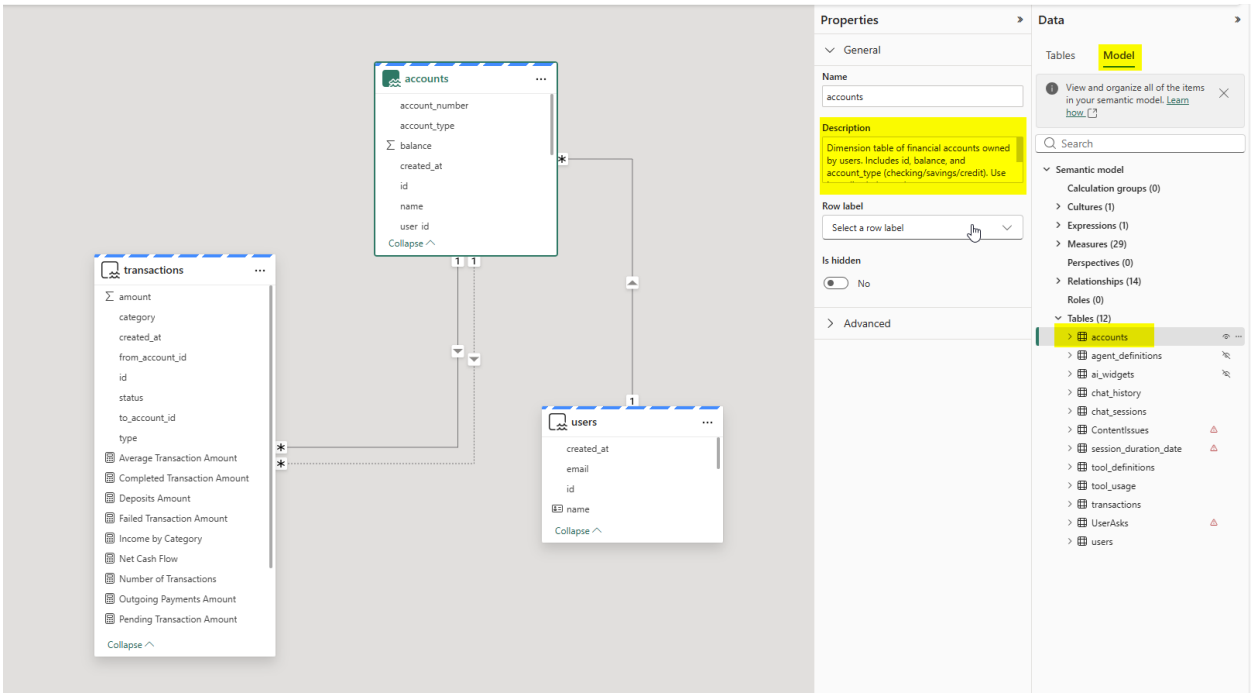
💡 Refer to the semantic_model_dataagent_checklist.md checklist for a comprehensive list of steps and dataagent_utilities.ipynb notebook for automating some of the steps.

**Add Table Descriptions**

Open agentic_semantic_model in Web and add table descriptions:

- *transactions:* Fact table of money movements. Each row is a single transaction with amount, type (ex: payment/deposit), status (completed/pending/failed), and category. Direction is determined by which account key is populated: from_account_id = outgoing, to_account_id = incoming. Use this table for spend/income by category and transaction volume.

- *accounts:* Dimension table of financial accounts owned by users. Includes id, balance, and account_type (checking/savings/credit). Use it to slice balances by account type, compute total balances, and evaluate credit utilization (credit balances may be negative).
- *Users:* Dimension table of customers (users). Use it to group and rank metrics at the customer level (net cash flow per user, incoming/outgoing per user, active/new users). If a user creation date exists (ex: created_at), use it only for "new users" style questions, not for transaction-month filtering unless explicitly intended.

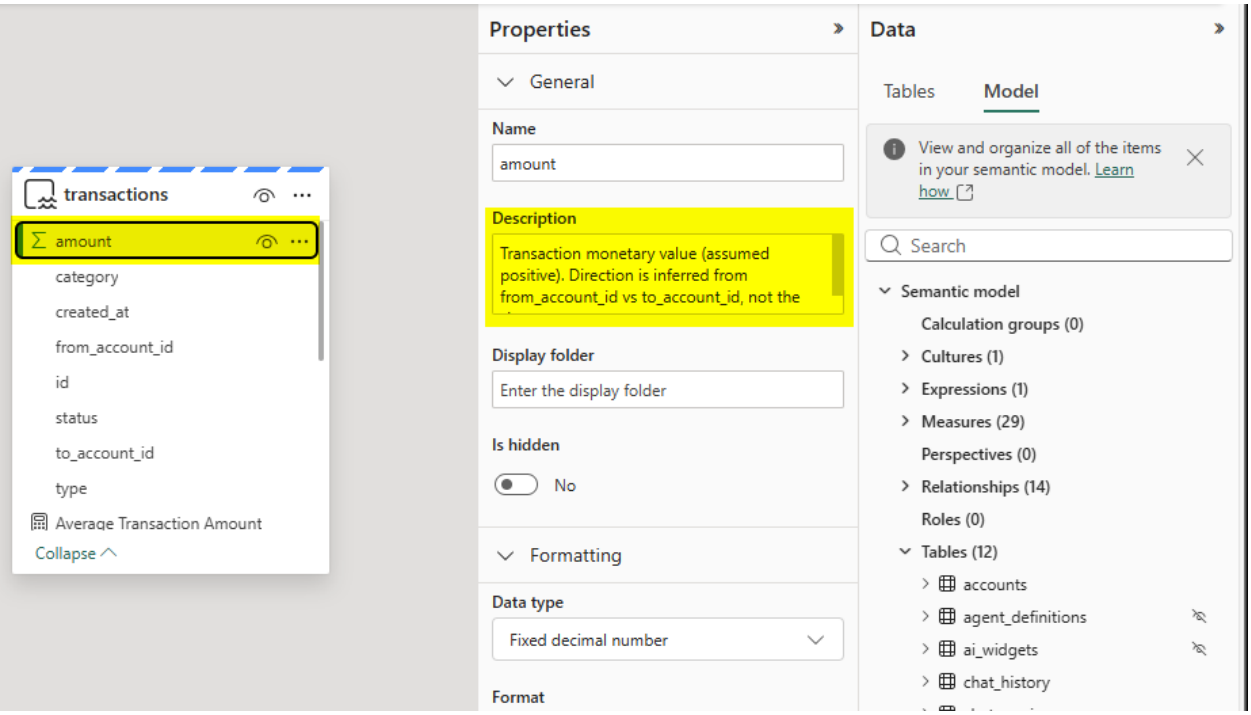



**Add column descriptions:**

Adding column descriptions helps AI understand the purpose of the columns and improves accuracy. For exercise purposes, we will add descriptions to the columns below only. In areal world project, it is recommended to add descriptions to all columns used in data agent.

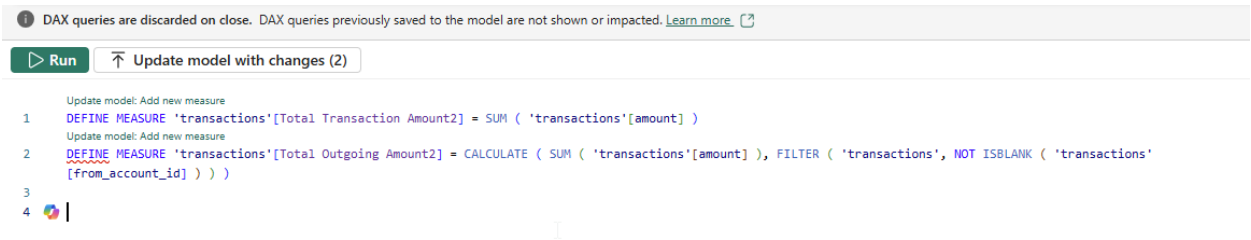
| Table | Column name | Description |
|---|---|---|
| transactions | from_account_id | Source account for outgoing transactions. If populated, treat the row as money leaving that account. |

| | | |
|---|---|---|
| transactions | to_account_id | Destination account for incoming transactions. If populated, treat the row as money entering that account. |
| transactions | amount | Transaction monetary value (assumed positive). Direction is inferred from from_account_id vs to_account_id, not the sign. |
| transactions | type | Transaction kind (ex: payment = outgoing spend, deposit = incoming funds). Used to separate spending vs income. |
| transactions | category | Spend/income classification label (ex: Groceries, Housing). Use for category breakdowns. |
| transactions | status | Processing outcome/state (ex: completed, pending, failed). Use to filter transaction amounts by status. |




**Measures:**

Explicit measures with descriptions help data agent calculate business metrics. Define measures based on the scope of the data agent.




## TRANSACTIONS MEASURES

DEFINE MEASURE 'transactions'[Total Transaction Amount] = SUM ( 'transactions'[amount] )

DEFINE MEASURE 'transactions'[Total Outgoing Amount] = CALCULATE ( SUM ( 'transactions'[amount] ), FILTER ( 'transactions', NOT ISBLANK ( 'transactions'[from_account_id] ) ) )

DEFINE MEASURE 'transactions'[Total Incoming Amount] = CALCULATE ( SUM ( 'transactions'[amount] ), USERELATIONSHIP ( 'accounts'[id], 'transactions'[to_account_id] ), FILTER ( 'transactions', NOT ISBLANK ( 'transactions'[to_account_id] ) ) )

DEFINE MEASURE 'transactions'[Net Cash Flow] = [Total Incoming Amount] - [Total Outgoing Amount]

DEFINE MEASURE 'transactions'[Number of Transactions] = COUNTROWS ( 'transactions' )

DEFINE MEASURE 'transactions'[Average Transaction Amount] = DIVIDE ( [Total Transaction Amount], [Number of Transactions] )

DEFINE MEASURE 'transactions'[Completed Transaction Amount] = CALCULATE ( [Total Transaction Amount], 'transactions'[status] = "completed" )

DEFINE MEASURE 'transactions'[Pending Transaction Amount] = CALCULATE ( [Total Transaction Amount], 'transactions'[status] = "pending" )

DEFINE MEASURE 'transactions'[Failed Transaction Amount] = CALCULATE ( [Total Transaction Amount], 'transactions'[status] = "failed" )

DEFINE MEASURE 'transactions'[Outgoing Payments Amount] = CALCULATE ( [Total Outgoing Amount], 'transactions'[type] = "payment" )

DEFINE MEASURE 'transactions'[Deposits Amount] = CALCULATE ( [Total Incoming Amount], 'transactions'[type] = "deposit" )

DEFINE MEASURE 'transactions'[Spending by Category] = CALCULATE ( [Total Outgoing Amount], 'transactions'[type] = "payment" )

DEFINE MEASURE 'transactions'[Income by Category] = CALCULATE ( [Total Incoming Amount], 'transactions'[type] = "deposit" )

**ACCOUNTS MEASURES**

DEFINE MEASURE 'accounts'[Total Account Balance] = SUM ( 'accounts'[balance] )

DEFINE MEASURE 'accounts'[Total Balance - Checking] = CALCULATE ( [Total Account Balance], 'accounts'[account_type] = "checking" )

DEFINE MEASURE 'accounts'[Total Balance - Savings] = CALCULATE ( [Total Account Balance], 'accounts'[account_type] = "savings" )

DEFINE MEASURE 'accounts'[Total Balance - Credit] = CALCULATE ( [Total Account Balance], 'accounts'[account_type] = "credit" )

DEFINE MEASURE 'accounts'[Credit Utilization] = VAR CreditBalance = [Total Balance - Credit] RETURN – CreditBalance

DEFINE MEASURE 'accounts'[Average Account Balance] = AVERAGE ( 'accounts'[balance] )

DEFINE MEASURE 'accounts'[Number of Accounts] = COUNTROWS ( 'accounts' )

DEFINE MEASURE 'accounts'[Accounts per User] = DIVIDE ( [Number of Accounts], DISTINCTCOUNT ( 'users'[id] ) )

**USERS MEASURES**

DEFINE MEASURE 'users'[Total Balance per User] = CALCULATE ( SUM ( 'accounts'[balance] ) )

DEFINE MEASURE 'users'[Total Outgoing per User] = [Total Outgoing Amount]

DEFINE MEASURE 'users'[Total Incoming per User] = [Total Incoming Amount]

DEFINE MEASURE 'users'[Net Cash Flow per User] = [Net Cash Flow]

DEFINE MEASURE 'users'[Active Users] = CALCULATE ( DISTINCTCOUNT ( 'users'[id] ), FILTER ( VALUES ( 'users'[id] ), CALCULATE ( COUNTROWS ( 'transactions' ) ) > 0 ) )

DEFINE MEASURE 'users'[New Users] = DISTINCTCOUNT ( 'users'[id] )

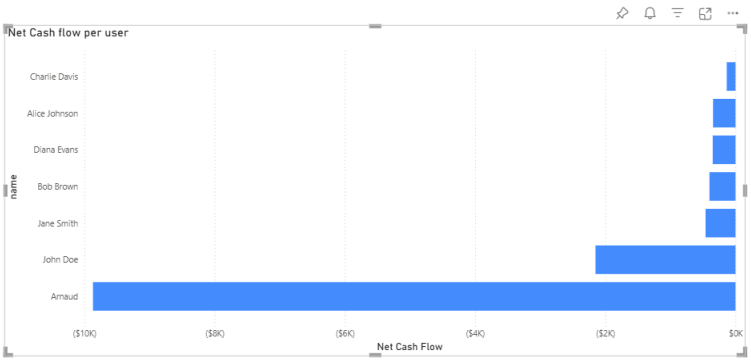
**[OPTIONAL] Measure Descriptions**

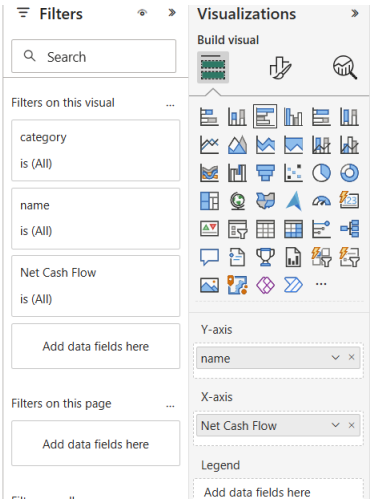
Add measure descriptions to define purpose, calculation logic in each measure

**Create a sample report / visual**

Explore the data by creating a report with a couple of visuals.

Use user column and Net Cash Flow measure to create a visual. Add category as a filter. Feel free to create more visuals. Save the report.




**Prep Data for AI:**

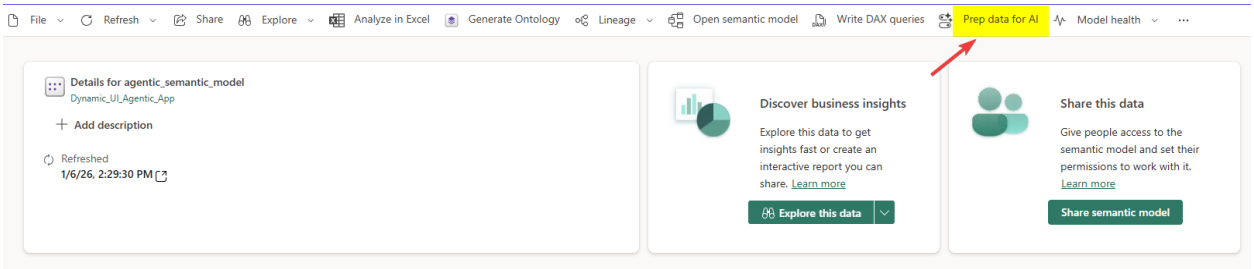
Prep for AI helps you shape how AI experiences interpret the semantic model. It lets you define which parts of the model to expose, describe how they should be used, and provide extra context to improve the accuracy of generated responses. Refer to documentation for details.

Prep for AI offers three key configuration components in Power BI:

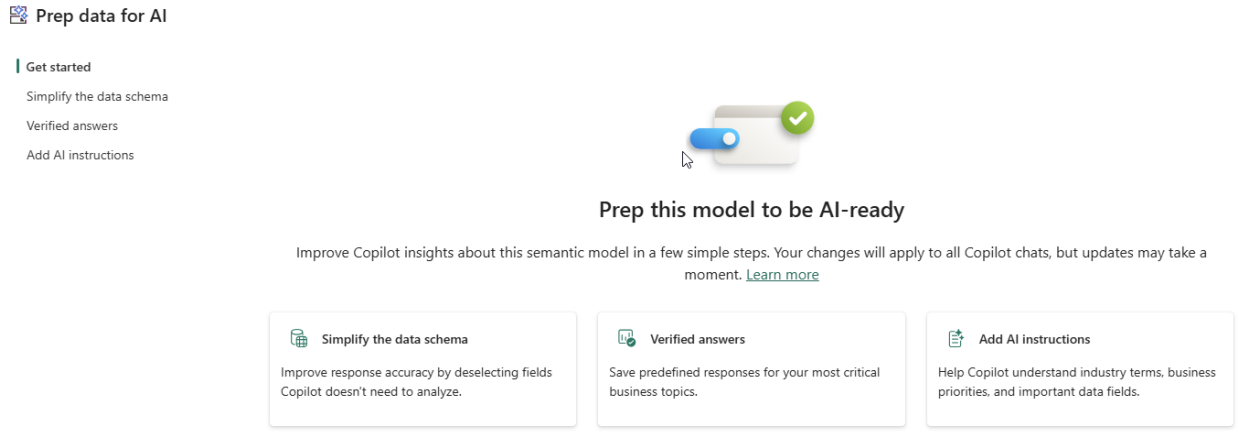
- **AI data schemas**: Define which tables and columns are prioritized for generated DAX queries.
- **Verified answers**: Predefined, validated responses to common or important questions.
- **AI instructions**: Contextual guidance to help AI generate relevant and accurate answers.

**Steps**:

1. Open Prep for AI

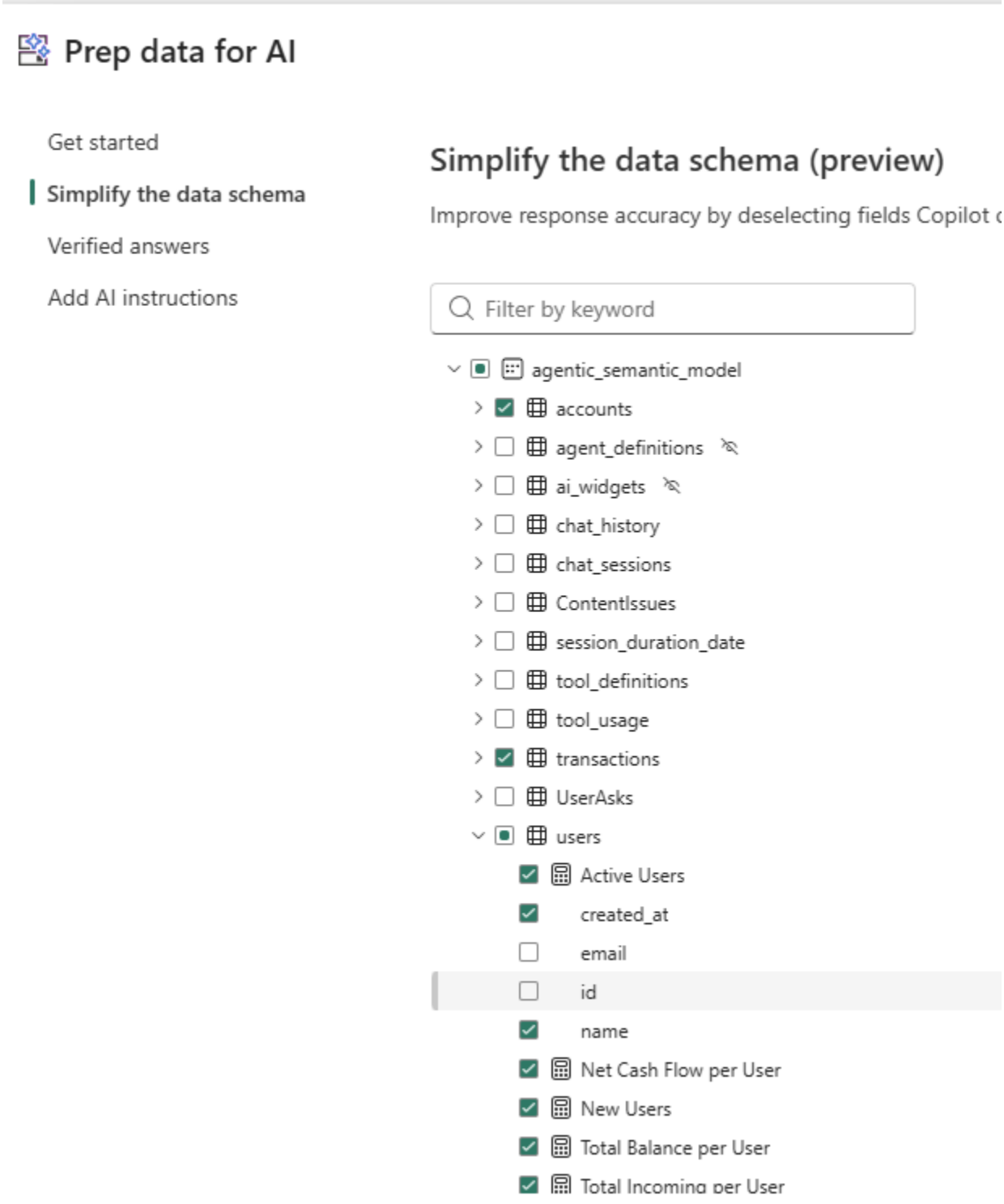



2. You can use the guided experience to configure Prep for AI



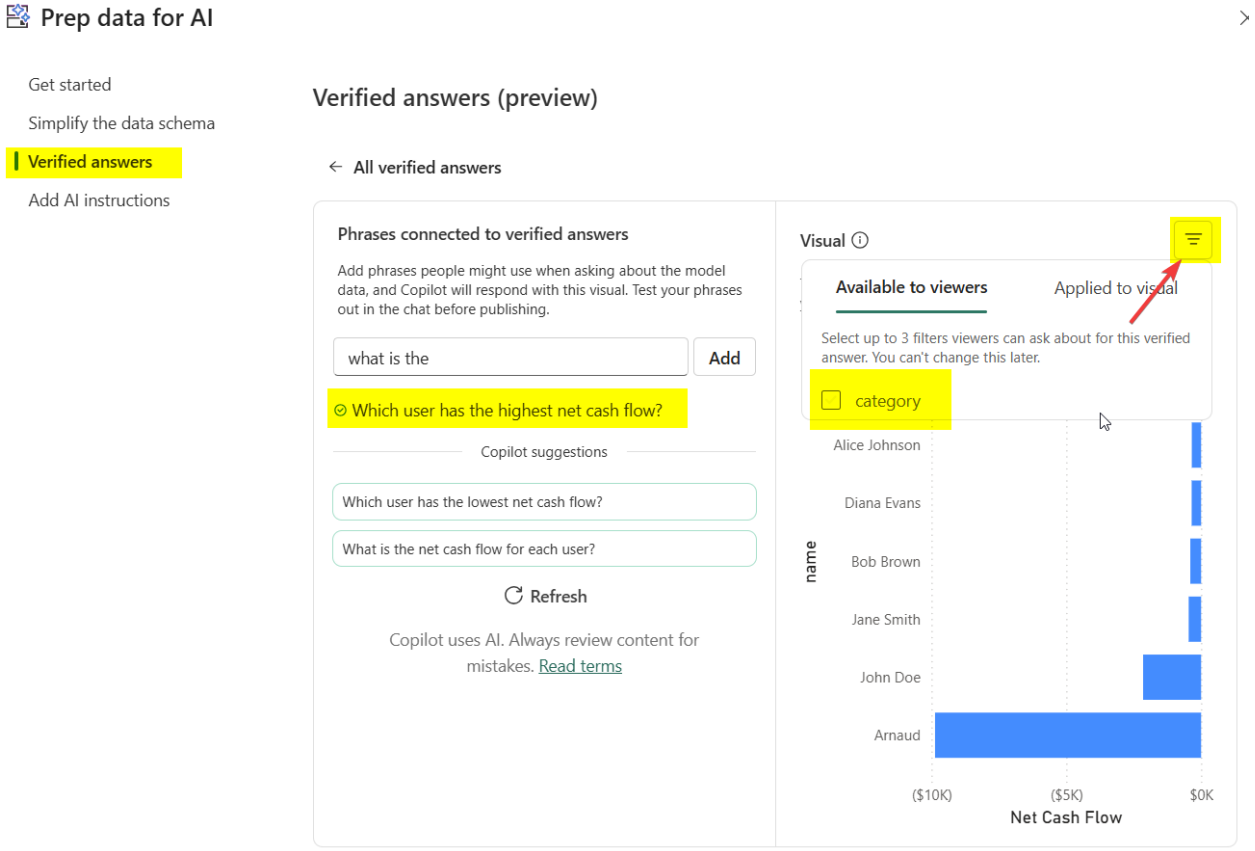

3. **Simplify data schema**

   Select only the accounts, transactions and users tables. In users table, exclude email and id columns. Always select only the colums that are in scope for data agent.




4. **Verified answers**
   Verified answers are human-approved, visual responses in Copilot that are triggered by predefined phrases. Each verified answer includes one or more trigger phrases, a visual, and optional associated filters.

Open the previously saved report in Edit mode and mark the visual as a Verified answer.




5. **Add AI instructions**
   Provide context on important data fields, business goals, and industry terms to improve Copilot insights about this semantic model.

Model purpose

- This model is for analyzing financial activity: transactions, account balances, user-level cash flow, and category-level spend/income.

Data model (what to use)

- Primary fact table: transactions (amount, type, status, category, from_account_id, to_account_id).
- Dimensions: users (customer = user), accounts.
- Treat “customer” and “user” as the same concept unless explicitly distinguished.

Core business logic (must follow)

- Outgoing money = transactions where from_account_id is populated; use measure Total Outgoing Amount.
- Incoming money = transactions where to_account_id is populated; use measure Total Incoming Amount (it activates the destination relationship internally).
- Net cash flow = incoming − outgoing; use Net Cash Flow / Net Cash Flow per User.
- Transaction "spending" means outgoing payments; use Spending by Category.
- Transaction "income" means incoming deposits; use Income by Category.
- For status-based totals, use: Completed Transaction Amount, Pending Transaction Amount, Failed Transaction Amount.

Analysis rules (how to answer common questions)

- "Who has the highest net cash flow?"
    - Rank users by Net Cash Flow per User (descending). Return top N users and include the value.
- "How much are customers spending by category?"
    - Group by transactions[category] and use Spending by Category. Include a total across categories.
- "Deposit vs payment amounts"
    - Use Deposits Amount and Outgoing Payments Amount (don't infer by sign).
- Account totals and breakdowns
    - Use Total Account Balance and the type measures (Total Balance - Checking/Savings/Credit). For credit utilization, use Credit Utilization.

Time logic / "this month"

- If a transaction date column exists in transactions, use it for month/period filtering.
- If no transaction date exists (or it's unclear), ask a clarification question: "Which column should define 'this month' for transactions?"

Output expectations

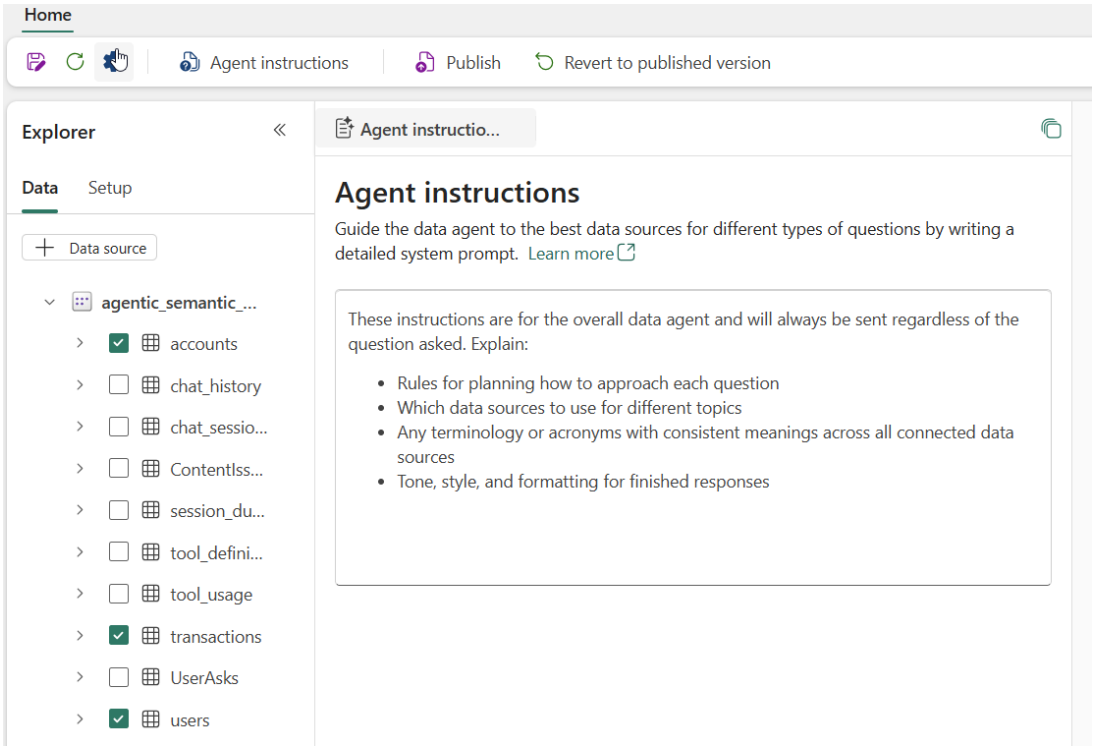
- Prefer measures over raw column aggregations when a measure exists.
- When returning breakdowns (by category, status, account_type), include:

- The breakdown table
- The total
- Top 3 contributors (if applicable)

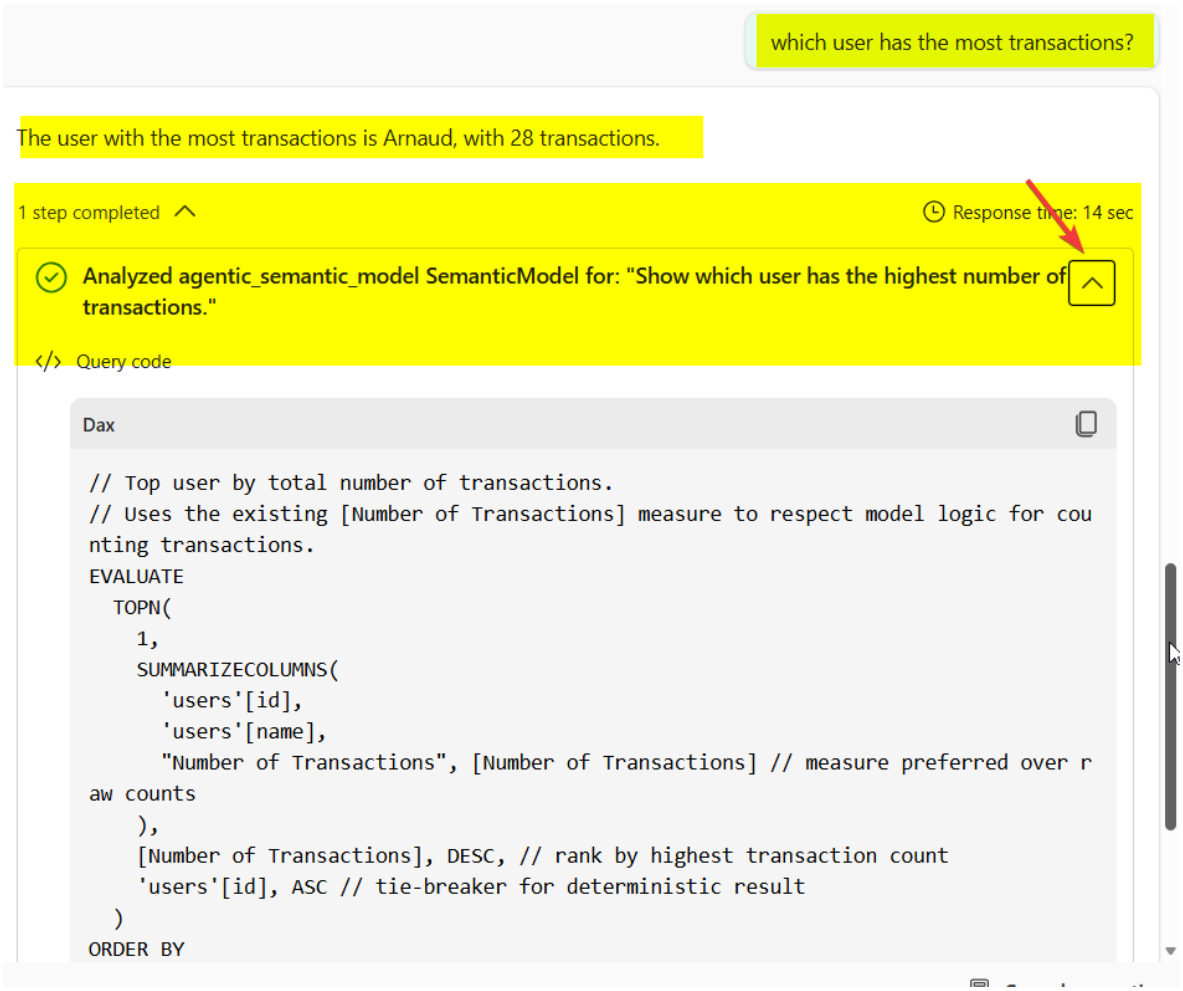
- If user-friendly names aren't available, use users[id] as the identifier.

**Semantic model is now ready for data agent !**

1. Create a new data agent and add this semantic model as a data source
2. Select the same columns you selected earlier in Prep data for AI schema selection




3. Agent instructions : Always return amounts, cost, spending, balance related responses with dollar signs, e.g. $123.1 instead of 123.1
4. Test the data agent with following questions
5. Always evaluate the responses by expanding the steps, verify the answers, DAX/SQL queries.




6. After verifying the responses, publish the data agent with a description : *This data agent can be used for analyzing financial activity: transactions, account balances, user-level cash flow, and category-level spend/income.*
7. Share the data agent with other users
8. You can programmatically evaluate the data agent, refer to the `dataagent_utilities.ipynb` for sample code
9. You can also query the data agent in VS Code using MCP server. See [Data agent as Model Context Protocol server (preview) - Microsoft Fabric | Microsoft Learn] for details.

**List of questions**

- What is the net cash flow per user, and who has the highest net cash flow?
- How much are customers spending by category (and which categories are highest)?

- What is the total account balance breakdown by account type (checking vs savings vs credit)?
- How many active users are there, and what is the average transaction amount for their activity?
- What are the totals for completed vs pending vs failed transactions (amount and count)?
- Which accounts have the highest credit utilization, and what is the total outstanding credit?
- What is income by category vs spending by category (net by category)?
- What is the average number of accounts per user, and how does it vary by user segment (if available)?
- What are deposit amounts vs payment amounts, and how do they compare overall?
- For new users (by users created date), what are their incoming, outgoing, and net cash flow?

***Challenging questions:***

- Identify users who have more outgoing than incoming transactions but maintain positive account balances. What's their average balance and primary spending categories
- For users with negative net cash flow, what's the breakdown of their spending by category, and which account types are they using most for payments