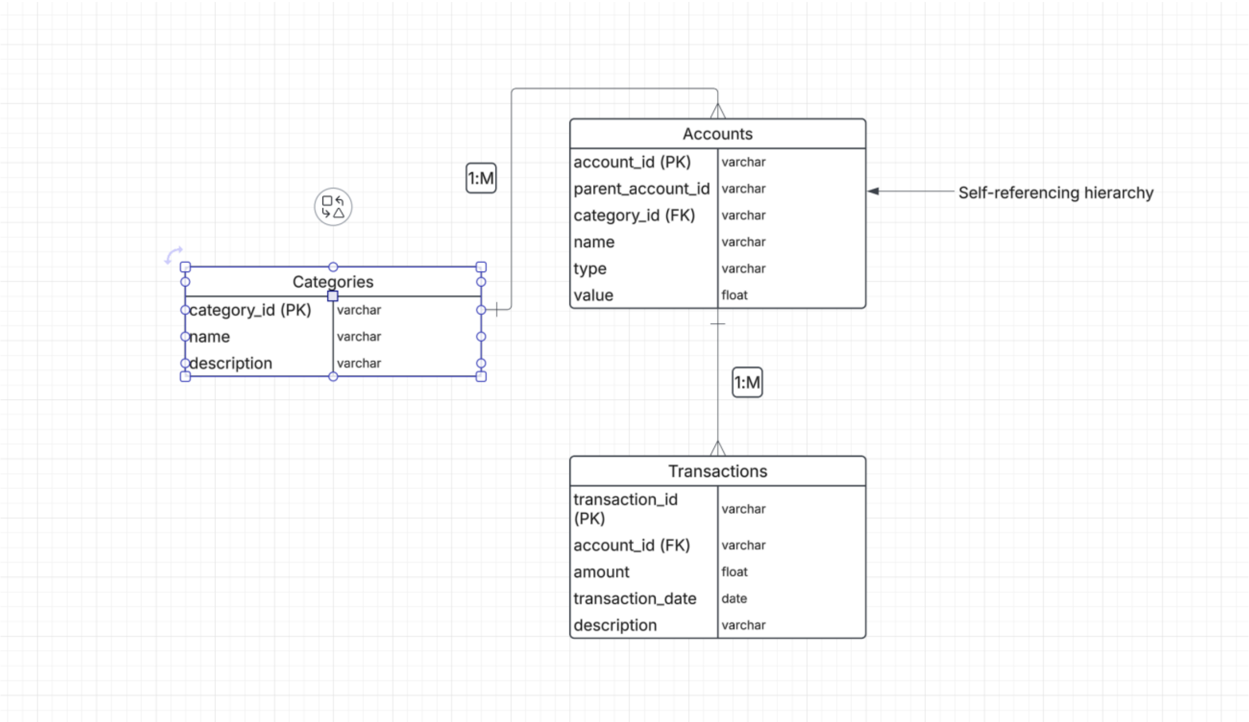
Response 1 – Data Modeling

ERD

https://lucid.app/lucidchart/6ece274c-3383-4f29-b44f-a551e029cff2/edit?view_items=ir2S8cFS8XPS&invitationId=inv_fe401157-ad14-464d-9c92-ab432c2f33d4



Accounts	
account_id (PK)	varchar
parent_account_id	varchar
category_id (FK)	varchar
name	varchar
type	varchar
value	float

Categories	
category_id (PK)	varchar
name	varchar
description	varchar

Transactions	
transaction_id (PK)	varchar
account_id (FK)	varchar
amount	float
transaction_date	date
description	varchar

1. **How Transactions Are Related to Accounts**
   We relate the transactions table and accounts table in our ERD and SQL schema using a foreign key relationship:

- One-to-Many Relationship (accounts → transactions) One account can have many transactions (Deposits, Withdrawals, Transfers, etc.). One transaction is associated with one account. This keeps all the financial activity for an account tracked.

2. **How Accounts Are Related to Categories**
   The accounts table is related to the categories table through a foreign key relationship, which organizes financial accounts into useful categories.

- One-to-Many Relationship (categories → accounts) A category ("Assets", "Liabilities", "Equity") can have many accounts. An account is part of one category. This allows for grouping and reporting of finances more effectively.

3. **How Accounts Maintain a Hierarchical Structure**
   The accounts table contains a self-referencing foreign key (parent_account_id) to enable parent-child relationships between accounts (hierarchical structure).

- Self-Referencing One-to-Many Relationship (accounts → accounts) Certain accounts may be parent accounts (e.g., "Bank Accounts"). Other accounts may be sub-accounts (e.g., "Checking Account" under "Bank Accounts"). The parent_account_id column associates a sub-account with its parent. This enables complicated financial hierarchies like having various accounts grouped under a high-level category

Response 2 – Data Validation

Key Discrepancies

1. Current Assets is short by 24,983.97

   Expected: 13,300,233.24

   Calculated: 13,275,249.27

   Missing Amount: 24,983.97

   Possible Reasons

   Missing Account: There has to be another current asset of 24,983.97 that hasn't been accounted for.

   Data Entry Error: If an adjustment was made but not accounted for in the JSON, it would create this discrepancy.

   Rounding Differences: If previous rounding errors accumulated, it could result in a small missing amount.

2. ASSETS is short by 24,983.97

   Expected: 13,318,970.87

   Calculated: 13,293,986.90

Missing Amount: 24,983.97

Possible Reasons

This variance is inherited directly from **Current Assets**.

Correcting "Current Assets" will automatically correct **ASSETS**.

3. Accounts Payable is short by 4,050.00

Expected: 83,086.72

Calculated: 79,036.72

Missing Amount: 4,050.00

Possible Reasons

Unrecorded liability: There may be an invoice or payable item of 4,050.00 that was not recorded.

Data Entry Error: Someone may have put a wrong amount into the system.

4. Current Liabilities is short by 935,489.03

Expected: 1,014,525.75

Calculated: 79,036.72

Missing Amount: 935,489.03

Possible Reasons

Missing Subcategories: "Current Liabilities" must have more than "Accounts Payable."

Other obligations are missing (e.g., short-term loans, accrued expenses, taxes payable).

5. Liabilities is short by 945,980.27

Expected: 1,025,016.99

Calculated: 79,036.72

Missing Amount: 945,980.27

Possible Reasons

This issue stems from "Current Liabilities."

If missing liabilities are found and added, this will be corrected automatically.

Key Takeaways

✅ Matches (No Issues)

13 accounts match exactly.

Account Name	Expected Value	Calculated Value	Difference	Type of Issue
Current Assets	13,300,233.24	13,275,249.27	**-24,983.97000000067**	Missing asset entry
ASSETS	13,318,970.87	13,293,986.90	**-24,983.969999998808**	Inherited from Current Assets
Accounts Payable	83,086.72	79,036.72	**-4,050.00**	Missing payable record
Current Liabilities	1,014,525.75	79,036.72	**-935,489.03**	Missing liability accounts
Liabilities	1,025,016.99	79,036.72	**-945,980.27**	Inherited from Current Liabilities

Recommended Next Steps

1. **Verify the missing amounts in the accounting system:**
    - a. Look for a missing "**Current Asset**" of 24,983.97.
    - b. Look for a missing "**Accounts Payable**" liability of 4,050.00.
    - **c.** Look for missing "**Current Liabilities**" of 935,489.03.
2. **Verify rounding differences:**
    - **a.** No major floating-point issues identified, but rounding should always be checked.
3. **Verify missing accounts in JSON:**
    - a. If you have additional assets or liabilities, make sure they are properly categorized in the JSON.

Instructions on how to run the validation script

- I used a spyder IDE to write my python script and you should be able to copy the script and run it in anaconda/spyder IDE free version.
- Using the corrected json as input directly so it would be easier to test the output.

README

Solution Strategy

Note: JSON formatting was off, and I had to check the validity of the JSON provided and fix it before starting with the interpretation

- **Interpreting the JSON Structure**

    - There are three financial categories in the JSON: assets, liabilities, and equity.
    - All three categories are parent-child hierarchical in nature.
    - There is account_id in some accounts and null in others (to denote roll-up categories).

- **Creating the Relational Schema**

    - Employed a self-referencing accounts table to preserve hierarchy.
    - Designed a categories table to classify accounts (e.g., Current Assets, Fixed Assets).
    - Added a transactions table to record financial activity (not in JSON, but helpful to audit changes).

- **Validating Roll-Ups**

    - Utilized Python to determine if parent account values match the sum of their children.
    - The mismatch vs non mismatch should ideally be thrown out as a slack notification to proactively catch these alerts - hence the format of the python script replicating the alerts from airflow.

Assumptions Made

✓ Account IDs are unique when present → If **account_id** is null, then it is a roll-up category.

✓ Expected balance (value) is authoritative → Parent account balances are equal to the sum of child accounts.

✓ No transactions in **JSON** explicitly → Added a transactions table for future use.

✓ All accounts are part of the three top-level categories: ASSETS, LIABILITIES, EQUITY.

Issues & Inconsistencies Found

- **Some Parent Accounts Have null account_id**
  - Makes it difficult to reference them uniquely in a relational database.
  - Solution: Utilized self-referencing parent_account_id to preserve structure.

- **Floating-Point Precision Errors in Summation**
  - Parent values occasionally varied slightly from children sum.
  - Solution: Applied SQL rounding methods to reduce discrepancies.

- **Liabilities Section Contains a Big Missing Amount**
  - Our calculated total for Current Liabilities was 935,489.03 off, which is missing information.

Next Steps If More Time Were Available

- **Build Data Ingestion Pipeline -**

  - I would create an airflow dag that would validate and store the validated json files in an s3 bucket
  - This enables version control
  - Scalability
  - Error monitoring
  - Automating validation of json format so the data analysts can focus on high impact tasks instead of fixing a json format
  - Alerting using airflow about incomplete data
- From the s3 bucket I would ingest the data into a warehouse preferably snowflake
- I would flatten these json files and create a dbt model with three base models for accounts, categories and transactions
- Join these models in the staging as required and create respective intermediate models and mart level models
- **Data Qualtiy and Integrity** - These dbt models enable you to write generic, custom and unit tests for these – which increases the trust of the stakeholders towards making informed decisions

- These mart level models would be exposed to a BI Layer preferably Sigma/Looker or Tableau from where the stakeholders can stay up to date and make decisions that align with the greater business goals
- I would have communicated with the respective stakeholders/customers to investigate Missing Liability Amounts → Determine where 935,489.03 is missing.
- Improve Transaction Tracking → Add real-world transaction imports to monitor changes over time