create the prompts and describe the database that needs to be created so that the user's information is saved appropriately in order to successfully store the opening and closing balance data in their user profile, which gets updated the next day given the transactions. additionally, describe the steps necessary to get the previous month's transaction from the banking data, which show descriptions and amounts to store in the categories listed in column C of the spreadsheet.

**ChatGPT said:**

To build a personal finance app in **Lovable** that stores user-specific opening and closing balances and categorizes bank transactions using data from your spreadsheet, here's what needs to happen.

**💾 1. DATABASE DESIGN (User-specific Balance and Transaction Tracking)**

**🔐 Core Tables to Create**

**users**

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| id | string (UUID) | Unique user ID |
| name | string | User's name |
| email | string | User's email |
| bank\_linked | boolean | True if bank account is connected |
| plaid\_access\_token | string | Stored securely (hidden from UI) |
| default\_account\_id | string | Primary account for balances/transactions |

**daily\_cashflow**

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| id | string (UUID) | Record ID |
| user\_id | string | Foreign key to users.id |
| date | date | Date of the entry |
| opening\_balance | number | Pulled from bank API |
| income | number | Categorized income from transactions |
| expenses | number | Total categorized expenses |
| closing\_balance | number | Computed: opening + income - expenses |
| net\_change | number | closing\_balance - opening\_balance |
| status\_color | enum | 'green' if net\_change ≥ 0 else 'red' |

**transactions**

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| id | string | Transaction ID from bank |
| user\_id | string | Foreign key |
| account\_id | string | Plaid account ID |
| date | date | Transaction date |
| description | string | Merchant or label |
| amount | number | Positive = expense, Negative = refund or income |
| category | string | Categorized using rules based on Column C of your spreadsheet |
| source | string | 'Plaid' or 'Manual' |

**🧑‍💻 2. LOVABLE PROMPTS (Step-by-Step)**

**🔐 A. User Bank Connection Setup**

plaintext

Copy code

Create a user onboarding screen that includes name, email, and a button to securely connect a bank account using Plaid Link.

Store the returned access\_token and account\_id in the users table securely.

Mark bank\_linked as true once connection is verified.

**🕗 B. Daily Balance Pull + Update Logic**

plaintext

Copy code

Schedule a daily task at 5 AM to:

- Retrieve the previous day's closing\_balance for the user from daily\_cashflow

- Fetch the current balance from Plaid's /accounts/balance API

- Insert a new row in daily\_cashflow with today’s opening\_balance = yesterday’s closing\_balance

- Categorize any transactions from Plaid /transactions for today into income and expense

- Compute closing\_balance = opening + income - expenses

- Set status\_color = green if net\_change ≥ 0, else red

**📅 C. Monthly Historical Transactions Import**

plaintext

Copy code

On first login, fetch 30 days of historical transactions using:

GET /transactions/get

Map transactions by:

- date

- amount

- description

- category

Use fuzzy matching on description or Plaid’s category to assign categories matching column C of the spreadsheet.

Insert into the 'transactions' table and aggregate into 'daily\_cashflow' by:

- income = sum of transactions where category in ["Income 1", "Income 2"]

- expenses = sum where category is in expense groups (e.g. "Groceries", "Rent")

**🗂️ 3. CATEGORIZATION BASED ON COLUMN C (Spreadsheet)**

From your uploaded file, Column C contains these example categories:

* Opening Balance
* Income 1 / Income 2
* Total Income
* Rent, Subscriptions, Loans
* Transportation, Utilities, etc.

**Recommendation:** Create a category mapping like:

python

Copy code

category\_map = {

"Paycheck": "Income 1",

"Venmo Transfer": "Income 2",

"Netflix": "Subscription",

"Uber": "Transportation",

"Chase Credit Card": "Loan Payment",

...

}

This will let you match Plaid descriptions to your budgeting structure.

**⚙️ 4. DAILY ROLLOVER LOGIC (Example)**

plaintext

Copy code

If May 8th ends with $1,200 closing balance:

- That value becomes the May 9th opening balance

- New transactions are categorized and recorded

- App computes new daily net\_change

- Calendar view is updated accordingly

Would you like me to generate the JSON config for Lovable's data layer and UI components next?

**You said:**

yes

**ChatGPT said:**

Lovable Finance Config

99

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

{

"database": {

"tables": [

{

"name": "users",

"fields": [

{ "name": "id", "type": "string", "primaryKey": true },

{ "name": "name", "type": "string" },

{ "name": "email", "type": "string" },

{ "name": "bank\_linked", "type": "boolean", "default": false },

{ "name": "plaid\_access\_token", "type": "string", "secure": true },

{ "name": "default\_account\_id", "type": "string" }

]

},

{

"name": "daily\_cashflow",

"fields": [

{ "name": "id", "type": "string", "primaryKey": true },

{ "name": "user\_id", "type": "string", "relation": "users.id" },

{ "name": "date", "type": "date" },

{ "name": "opening\_balance", "type": "number" },

{ "name": "income", "type": "number", "default": 0 },

{ "name": "expenses", "type": "number", "default": 0 },

{ "name": "closing\_balance", "type": "number" },

{ "name": "net\_change", "type": "number" },

{ "name": "status\_color", "type": "enum", "options": ["green", "red"] }

]

},

{

"name": "transactions",

"fields": [

{ "name": "id", "type": "string", "primaryKey": true },

{ "name": "user\_id", "type": "string", "relation": "users.id" },

{ "name": "account\_id", "type": "string" },

{ "name": "date", "type": "date" },

{ "name": "description", "type": "string" },

The JSON configuration for your Lovable app is now set up in the canvas titled **"Lovable Finance Config."** It defines:

* A secure and relational database for users, daily balances, and transactions
* A daily automated workflow for fetching banking data
* UI components to handle Plaid bank connection, calendar view with red/green status, and transaction logs

Would you like me to add the category mapping logic or Plaid API endpoint samples next?

**You said:**

Explain the code that was created using the persona of a SaaS developer explaining it to a layperson

**ChatGPT said:**

Absolutely! Here's an explanation of the code we created, using the persona of a **SaaS developer explaining to a non-technical team member** — like a product manager or founder.

**🧠 "What This Code Does" – in Plain English**

We're building a **personal finance app** that automatically:

1. Connects to a user's bank account.
2. Pulls their **daily balances and recent transactions**.
3. Saves that info in a database.
4. Shows whether they're financially “in the green” or “in the red” each day.

Here’s how the different parts of the code make that happen:

**🧾 1. Database Section – Think of it like our app’s digital filing cabinet.**

We set up 3 digital "folders" or **tables**:

**📁 users**

* Stores each person’s name, email, and their secure bank connection info.
* We keep their **bank login token and default bank account ID** safely hidden from the interface but accessible to the backend.

**📁 daily\_cashflow**

* For every day, we store:
  + How much money the user started with (**opening balance**)
  + What they earned (**income**)
  + What they spent (**expenses**)
  + How they ended the day (**closing balance**)
  + Whether they were financially **green or red** (based on net gain/loss)

**📁 transactions**

* Tracks every individual transaction (like buying groceries or getting paid).
* We tag each transaction with a **date**, **amount**, **description**, and a **category** (like Rent, Food, or Income).
* It remembers which user and which bank account each transaction came from.

**🔄 2. Scheduled Task – A daily routine the app runs in the background**

Imagine a robot that wakes up every morning at **5:00 AM** and does this:

* Checks yesterday’s ending balance.
* Calls the bank (through Plaid) to ask, “What’s the account balance today?”
* Adds a new row for today in the balance sheet.
* Downloads yesterday’s new transactions.
* Sums up how much money came in and went out.
* Calculates if the person saved money (green) or overspent (red).
* Updates the calendar so the user can see where they stand.

**🖼️ 3. UI Components – What the user actually sees and interacts with**

These are the visual parts of the app:

**🔐 BankConnection**

* A form where a user enters their name and email, then clicks a button to **link their bank** securely using Plaid.

**📅 CalendarView**

* Shows each day of the month in a calendar format.
* The color of the day is **green if they saved money**, or **red if they lost money**.
* It also shows their **opening balance** for each day.

**📃 TransactionList**

* A searchable table that lets users view all their transactions.
* They can filter by date or by spending category.

**✅ Summary for Non-Technical Stakeholders**

This setup:

* **Automatically pulls** a user's financial data each day.
* **Tracks income and spending** without manual entry.
* **Visually highlights financial trends** (green/red).
* Sets us up to deliver **daily insights, alerts, and budgeting recommendations** based on real data.

Would you like me to walk through the next steps—such as how to connect this to the Plaid API or build a spending report page based on the transaction categories?