

# Real bank data, no manual entry.

How I replaced a budget spreadsheet I updated by hand with a pipeline that pulls my transactions through the Plaid API, sorts them, and emails me a report every Sunday.

**Anwana Ntofon**

No real account data, amounts or merchants appear in this deck.

## THE LEDGER

**4** bank connections through Plaid

**12** budget categories

**2** runs a week, Wednesday and Sunday

**5 mo** on live data, March to August 2026

**0** servers of mine holding the data

# A budget spreadsheet I was updating by hand.

Every update meant pulling statements from each bank, copying rows across, and sorting every charge into a category myself.

The goal: real transactions land in the sheet on their own, already sorted, with a report every week. No servers to run, no statement downloads.

**Built** March 2026. **Live** twice a week through August.  
**Published** October 2026, with the issues I found along the way.

## WHAT IT DOES



### Connects 4 Plaid Items

a credit card, checking and savings accounts



### Fetches twice a week

on time-driven triggers, deduplicated on transaction\_id



### Sorts into 12 categories

ordered rules, editable by hand in the sheet



### Builds the views

a weekly view, a dashboard and a monthly budget tab



### Emails a Sunday report

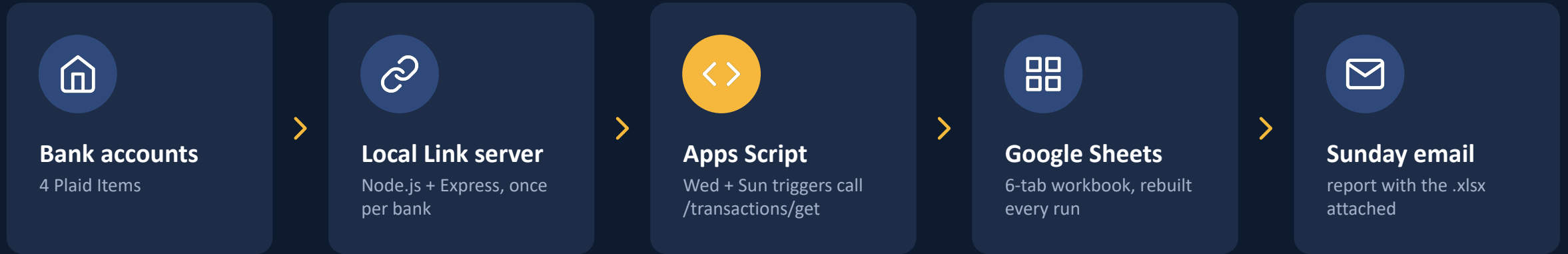
with the workbook attached as .xlsx



### Has a pause switch

stops every Plaid call without touching code

# Link once, then it runs on its own.



## WHY A LOCAL LINK SERVER

- Plaid Link runs in a browser, but the token exchange needs the secret, so it has to happen server-side
- Browser calls to Plaid are blocked by CORS anyway
- It runs once per bank. After that, Apps Script does everything



## WHERE THE SECRETS LIVE

- Apps Script Script Properties, plus a git-ignored local file for Link step
- Separate secrets for Sandbox and Production
- Nothing in the repo. A leaked token gets invalidated and rotated

# Calling the API was easy. The data wasn't.



01

## Sign convention

Plaid reports money out as positive, so income looked negative. Paychecks are flipped to read as income.



02

## Transfers aren't spending

Moves between my own accounts inflated the totals. They map to Transfer and stay out of spending.



03

## Which description to show

merchant\_name was sometimes too generic. The raw bank string is uglier, but far more consistent.



04

## Duplicates and pending

Every row keeps Plaid's transaction\_id, so repeats are skipped. Pending charges wait until they post.



05

## Timezones

Dates shifted across UTC midnight and mis-sorted weeks. Every date is now a Pacific yyyy-MM-dd string.



06

## A call cap

Failed calls count against the cap too. A PLAID\_PAUSED switch stops every call without touching code.

# From Sandbox to live bank data.



**The takeaway:** the questionnaire was easier to answer honestly because the architecture was simple. Everything lives in Google's managed environment, no servers of mine hold data, and deleting it all takes three steps.

# Known issues, documented instead of hidden.

## KNOWN ISSUES

- The fetch windows leave gaps: late-arriving charges and the last days of a month were never requested
- The month-end reminder only fires when a month ends on a run day. It fired once, so totals ran from March on
- The budget tab is pinned to Mar\_2026 by name
- The Sunday email counts Investments as spending; the dashboard doesn't

## WHAT I'D CHANGE

- Move to /transactions/sync with a stored cursor instead of date windows
- Give the month-end check its own daily trigger
- Sort on personal\_finance\_category instead of the legacy category field
- Merge manual cash entries into the main transaction view

*The code in the repo is what ran. These are documented as open issues, not silently fixed.*

THE WHOLE IDEA, IN ONE LINE

# The API call is the easy part. Understanding the data is the job.

Built by Anwana Ntofon, a customer-informed engineer who has taken an API integration from Sandbox to live data, and wants to help teams do the same.

[github.com/antofon/myledger](https://github.com/antofon/myledger) · [anwanantofon.com](https://anwanantofon.com)

**Link**

once per bank

**Fetch**

twice a week

**Sort**

into 12 categories

**Report**

every Sunday