Case Study · AI Integration

Giving Claude direct access to Google Sheets — eliminating a 5-step manual data loop

How to move from constant CSV downloads and copy-paste friction to Claude reading, writing, and creating Google Sheets autonomously

Pranoti Kshirsagar · AI Integration & Automation Specialist · thesciencetalk.com

5 steps	0 €	All sheets
eliminated from every data task	ongoing cost	accessible — not just one

01 · THE REAL PROBLEM — DEATH BY DATA FRICTION

THE GOAL

Deep integration of Claude across all business workflows. Not a one-off connection — a flexible, reusable bridge between Claude and **any Google Sheet**, for any task: grants database, social media analysis, webinar tracking, client data.

THE DAILY REALITY

Every time data needed to move between Claude and a spreadsheet, it required a **full manual loop** — downloading, converting, uploading, sharing, then copying results back out. Slow, error-prone, and capacity-draining.

02 · BEFORE VS AFTER — THE WORKFLOW

Before

1 Export CSV from source › 2 Download to local machine › 3 Upload to Google Drive › 4 Open as Google Sheet › 5 Copy-paste into Claude

After

1 Tell Claude what to do with the sheet › 2 Claude reads, writes, or creates directly › 3 Done

02B · THE HIDDEN MISCONCEPTION

COMMON ASSUMPTION THAT'S WRONG

Claude.ai has a native Google Drive connector — so surely it can access Google Sheets? **No.** The native Drive connector is limited to Google Docs only. Claude cannot read cell contents, write to cells, or create new spreadsheets through it. For any real Sheets integration, a dedicated MCP server is required — in Claude Desktop, configured separately from claude.ai entirely.

03 · BLOCKERS AND SOLUTIONS

BLOCKER	SOLUTION
× **Service account key creation blocked** — Google Workspace org policy had JSON key downloads disabled. Every standard tutorial assumes this works.	✓ Switched to **OAuth 2.0 Client ID** (Desktop app type) — bypasses org policy entirely, uses personal auth instead of service account keys.
× **Google Drive API not enabled** — mcp-google-sheets requires both Sheets API and Drive API. Enabling only one causes silent failures with no useful error message.	✓ Explicitly enabled **both APIs** in GCP Library — Sheets API and Drive API. They don't auto-enable each other.
× **uvx not resolving in Claude Desktop config** — using "uvx" as the command caused silent launch failures with no error output to debug from.	✓ Replaced with the **full absolute path** to uvx (e.g. /Users/name/.local/bin/uvx). Resolved immediately.

04 · WHAT CLAUDE CAN NOW DO WITH GOOGLE SHEETS

- ✓ Read cell contents directly — no upload, no copy-paste
- ✓ Write and update cells without manually touching the sheet
- ✓ Create brand new spreadsheets from scratch on command
- ✓ Works across all sheets — grants database, social media analysis, webinar tracking, and more

05 · FINAL SETUP — WHAT WAS BUILT

ARCHITECTURE

1. **Google Cloud project** — Sheets API + Drive API both enabled explicitly
2. **OAuth 2.0 Client ID** (Desktop app) — consent screen configured, personal email as test user
3. **mcp-google-sheets** added to claude_desktop_config.json — full absolute uvx path, OAuth credentials in env block
4. **One-time browser auth flow** — token saved automatically, no repeat login needed

06 · TECH STACK

Claude Desktop · mcp-google-sheets · Google Sheets API · Google Drive API · OAuth 2.0 · uvx · Google Workspace Business Standard · Native Drive connector — Docs only, not Sheets

07 · WHAT I'D DO DIFFERENTLY

HONEST REFLECTION

Start with OAuth from day one — even without org policy restrictions, it's more portable and avoids the security risk of stored JSON key files. Treat Claude Desktop and claude.ai as completely separate systems from the start — different configs, different MCP connections, nothing shared. Always use absolute paths in Claude Desktop configs; "uvx" alone is a silent failure waiting to happen.

Full step-by-step walkthrough published at thesciencetalk.com · More case studies at pranoti.thesciencetalk.com

Read the guide →