Module 4: Automating Financial Workflows

MGMT 675: Generative AI for Finance

Kerry Back

The Dashboard Trap

Dashboards Answer Yesterday's Questions

Organizations spend millions building dashboards. When a new question arises, the cycle restarts: requirements → design → build → deploy.

The Typical Dashboard Lifecycle

1. Business user requests a report
2. Analyst translates to requirements
3. Engineer builds SQL queries
4. Designer creates visualizations
5. IT deploys to Tableau/Power BI
6. User asks a follow-up question
7. **Back to step 1**

The Cost

- **Time**: Weeks to months per dashboard
- **Money**: BI licenses, engineering hours, maintenance
- **Rigidity**: Fixed views of fixed data

Gartner: only **20%** of analytic insights deliver business outcomes.

The Fundamental Problem

Dashboards answer **pre-defined questions**. But the most valuable analysis comes from **ad-hoc questions** that arise in the moment.

- "What happened to margins in the Southeast last quarter?"
- "Show me our top 10 customers by growth rate, excluding one-time orders"
- "Compare Q3 headcount vs. budget by department, and flag anyone over 110%"

These are simple questions. Getting answers shouldn't require a development cycle.

Natural Language as the Query Interface

The Shift: From Dashboards to Conversations

	Traditional Dashboard	**Natural Language AI**
Query method	Click filters, select dates	Ask in plain English
Time to answer	Minutes to weeks	Seconds
Follow-up questions	New dashboard request	Next sentence
Visualization	Pre-built charts	Generated on demand
Who can use it	Trained users	Anyone
Cost per question	High (amortized)	Near zero (marginal)

The dashboard was a *workaround* for the fact that databases don't speak English. Now they do.

What This Looks Like in Practice

The Conversation

- *"Show me monthly revenue by product line for 2025"*
- AI: writes SQL, produces grouped bar chart
- *"Break out Enterprise by region"*
- AI: refines query, updates chart
- *"Which region had the biggest Q2–Q3 drop?"*
- AI: calculates changes, highlights answer

What the User Needed to Know

- What questions to ask
- Whether the answers make sense
- **Nothing else**

Behind the Scenes

- 4 different SQL queries written
- 3 visualizations produced
- Derived metrics calculated

The Database Agent Pattern

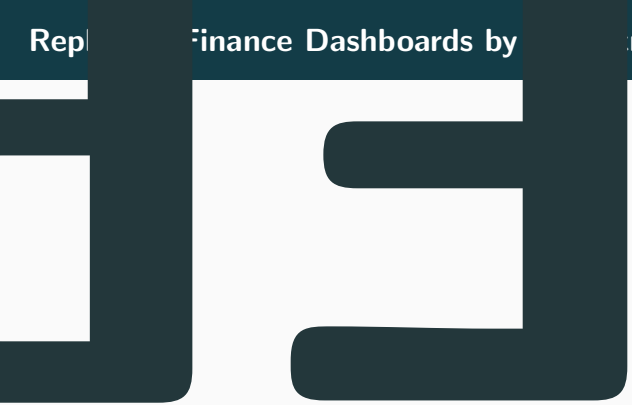
The most powerful dashboard replacement: an AI agent connected to your database.

How to Build It

1. Connect database via MCP or file upload
2. Give AI the schema: table names, columns, relationships
3. Describe the business context
4. Start asking questions

What the Agent Can Do

- Write and execute SQL queries
- Compute derived metrics (growth rates, ratios)
- Generate charts and export to Excel or PowerPoint



The Weekly Ops Review — Before and After

Before: The Dashboard Era

1. Data team pulls exports (2 hours)
2. Analyst builds slides (4 hours)
3. Manager reviews (1 hour)
4. Analyst revises (2 hours)
5. VP asks a question not on the slide
6. "We'll get back to you next week"

Total: 9+ hours per week

After: Natural Language AI

1. AI agent connected to database
2. Analyst: *"Generate the weekly ops review"*
3. AI: pulls data, charts, narrative (3 min)
4. Manager reviews, iterates in chat
5. VP asks a question not on the slide
6. **AI answers it in 10 seconds**

Total: 15 minutes + live Q&A

Skills: Encoding Expertise in Reusable Instructions

It's All Just Text

Every AI customization — skills, slash commands, custom instructions — works the same way: **additional text is added to the prompt** that the model sees.

- **Skills** = instructions for *how* (loaded when relevant)
- **Slash commands** = triggers that say *do* (typing `/name` loads that skill)
- Same pattern everywhere: Custom GPTs, Copilot instructions, `.cursorrules`, Gems

What is a Skill?

A **skill** is a set of instructions that specializes a general-purpose AI for a specific domain or task. In Claude, it's a folder with a markdown file.

A Skill Provides

- System prompt (domain knowledge)
- Workflow instructions
- Best practices and constraints

The AI Platform Provides

- The LLM (Opus or Sonnet)
- Agent control loop and tools
- Code execution sandbox

Skills let you create **specialized agents without writing agent logic**.

Anatomy of a Skill

Skill Folder Structure

```
skills/
  xlsx/
    SKILL.md      <- Main file
    scripts/
      recalc.py
    references/
      schema.md
```

SKILL.md Structure

```
---
name: xlsx
description: "Excel file..."
---

# Requirements for Outputs
- Zero formula errors
- Use formulas, not hardcodes

# Workflows
1. Choose pandas or openpyxl
2. Create/modify file
3. Recalculate formulas
```

Where Skills Live

- **Project skills**: `.claude/skills/` in your project folder
- **User skills**: `~/.claude/skills/` (shared across projects)
- Skills also work in **Claude.ai** (upload ZIP), **Cowork** (install as plugin), and **VS Code**

Format	How to Install	How to Invoke
Claude.ai (web)	Upload ZIP in Settings	Automatic or `/name`
Code tab / CLI	Place in `.claude/skills/`	Automatic or `/name`
VS Code extension	Place in `.claude/skills/`	Automatic or `/name`

The Consistency Problem

Why Skills Matter for Finance

In Module 2, you built DCF models and portfolio optimizations in ad-hoc conversations. What happens when you start a new conversation?

The Problem

- One conversation uses 10% WACC, another uses 8%
- One assumes 3% terminal growth, another 2.5%
- Portfolio optimization: 5-year history vs. 3-year
- **No audit trail, no reproducibility**

The Fix: Skills

- A DCF skill specifies: which assumptions, how to estimate them, what output format
- A portfolio skill specifies: data source, estimation window, constraints
- **Every conversation follows the same playbook**

This is how AI moves from *personal productivity tool* to *institutional capability*.

Finance Skill Examples

Variance Analysis with the Finance Plugin

What is Variance Analysis?

Variance analysis compares **budgeted** figures to **actual** results and decomposes the differences into actionable drivers. It is the core analytical task in FP&A.

Why It Matters

- Every public company does it quarterly
- Boards and investors ask "why did we miss?"
- Drives re-forecasting and capital allocation

Key Decompositions

- **Revenue**: volume vs. price vs. mix
- **COGS**: volume vs. unit cost
- **SG&A**: headcount vs. rate vs. discretionary

Revenue variance = **Volume effect** + **Price effect** + **Mix effect**

Variance Analysis: Example Data

Scenario: You are an FP&A analyst. Q1 actuals just closed. The CEO wants to know why operating income missed budget by $300K.

	Budget	Actual	$ Variance
Revenue			
Units sold	100,000	95,000	
Avg price	$50.00	$51.00	
Total Revenue	*$5,000,000*	*$4,845,000*	*($155,000)*
COGS			
Unit cost	$30.00	$32.00	
Total COGS	*$3,000,000*	*$3,040,000*	*($40,000)*
Gross Profit	*$2,000,000*	*$1,805,000*	*($195,000)*
Total SG&A	*$1,450,000*	*$1,555,000*	*($105,000)*
Operating Income	**$550,000**	**$250,000**	**($300,000)**

Using the Finance Plugin

With the finance plugin installed, one command replaces 2–4 hours of FP&A work:

Prompt

`/finance:variance-analysis`

"Here is our Q1 budget vs. actuals spreadsheet. Decompose the $300K operating income miss into volume, price, cost, and discretionary spending drivers. Produce a waterfall chart and a summary memo for the CFO."

What the Plugin Does

1. **Skill activates**: domain knowledge about variance decomposition formulas
2. **Decomposes** revenue into volume + price; COGS into volume + rate
3. **Generates** waterfall chart and writes CFO-ready memo in Word format

Plugins and the SaaSpocalypse

What is a Plugin?

A **plugin** is a packaged collection of skills, agents, hooks, and MCP servers that can be shared across teams.

A Plugin Can Include

- Skills (`SKILL.md` files)
- Agents (`AGENT.md` files)
- MCP servers (external tools)

Why Use Plugins?

- Share skills with your team
- Namespace prevents conflicts
- Install from marketplaces

Practical workflow: Start with a standalone skill for quick prototyping. When ready to share, wrap it in a plugin by adding a manifest file.

The "SaaSpocalypse": Market Reaction to Plugins

On January 30, 2026, Anthropic released 11 open-source plugins for Claude. Within days, **$285 billion** in market cap evaporated from software, legal tech, and financial services stocks.

Hardest-Hit Stocks

Company	Drop
LegalZoom	−20%
Thomson Reuters	−16%
RELX (LexisNexis)	−14%
Wolters Kluwer	−13%
Xero (accounting)	worst day since 2013
Intuit	$> -10\%$
Salesforce	−7%

Why It Happened

- Plugins showed AI could replicate *core workflows* of specialized enterprise software
- **No proprietary code** — just markdown instructions + database connectors
- Investors asked: "Why pay $50K/yr for software when a plugin does it?"

Exercises

Exercise 1: Build a Database Skill

1. Create `.claude/skills/database/SKILL.md` for a sample database (Chinook or provided dataset)
2. Include: schema descriptions, table relationships, business rules
3. Test with 5 decision-relevant queries
4. Have a classmate ask 3 untested questions

The skill should enable anyone to query the database in plain English.

Exercise 2: DCF Skill

Create a skill that encodes a consistent DCF methodology:

- Specify how to estimate each assumption (e.g., "sales growth from trailing 5-year CAGR")
- Required output tables and sensitivity ranges
- Consistent output format (Excel workbook)

Test: Run `/dcf-valuation` on two different companies in separate conversations. Verify that the **methodology is consistent** across both.

Exercise 3: Portfolio Optimization Skill

Create a skill that specifies:

- Data source (e.g., FMP API)
- Estimation window (trailing 60 months)
- Constraint set (no short sales, max 25% per position)
- Output format (weights table, frontier plot, statistics)

Test: Run on two different sets of assets in separate conversations. Verify consistent methodology.

Exercise 4: Interactive Dashboard Artifact

1. Choose a dataset (financial or otherwise)
2. Ask Claude to build an interactive artifact with filters and dropdowns
3. The artifact should replace a static dashboard
4. Compare: how long would the equivalent BI tool take to build?

Summary

Dashboards

- Natural language replaces fixed dashboards
- AI writes queries and charts
- Follow-ups are instant

Skills

- Encode expertise in text
- Solve the consistency problem
- Create specialized agents

Plugins

- Package and share skills
- SaaSpocalypse: $285B impact
- Finance is ground zero

The best dashboard is no dashboard — it's a conversation with your data.