Module 5: Verifying AI-Generated Analysis

MGMT 675: Generative AI for Finance

Kerry Back

The Trust Spectrum

AI with Code Execution: A Fast Junior Analyst

AI with code execution is like a **fast junior analyst** — capable but needs oversight. The level of oversight depends on the stakes.

Low-Stakes

- Exploratory charts
- Directional trends
- Internal brainstorming

Quick sanity check.

Medium-Stakes

- Team tools and skills
- Recurring reports
- Internal dashboards

Test with known answers.

High-Stakes

- Client deliverables
- Regulatory filings
- Board presentations

Full verification protocol.

The more consequential the decision, the more you verify. **Not distrust — calibrated confidence.**

AI Makes Silent Analytical Errors

The Real Risk: Not Hallucination — Bugs

AI in code-execution mode doesn't invent facts the way a chatbot might. Instead, it makes **silent analytical errors**: wrong filters, dropped rows, incorrect joins.

Common errors

- Missing data silently dropped
- Date or currency parsing surprises
- Duplicates inflating counts
- Filters that are slightly off
- Aggregating before vs. after filtering

Why it's dangerous

- The output *looks* professional
- Charts render cleanly even if data is wrong
- Nobody questions a polished result
- **Confidence without verification is the real risk**

Catching a Silent Error

The prompt

"What's the average profit margin by sub-category? Exclude any sub-categories with negative total profit."

The trap

- Does AI filter *before* or *after* aggregating?
- Does it drop rows with negative profit, or sub-categories with negative *total* profit?
- The order matters — and AI often gets it wrong

How to catch it

- **"Show me the code"** — inspect the filter logic
- **"How many rows are in the result?"** — does it match your expectation?
- **"List the excluded sub-categories"** — are they the right ones?

Lesson: The more specific the filter logic, the more likely AI misinterprets it. Always verify multi-step operations.

The Verification Checklist

A 4-Step Verification Protocol

Apply this to **any** AI-generated analysis.

1. **Sanity-check with known answers**
   Ask questions where you already know the result: row count, column names, a total you can verify in Excel.
2. **Ask AI to show its methodology**
   "Show me the code" or "Show me the SQL." Read the logic, not just the output.
3. **Cross-check by rephrasing**
   Ask the same question differently. Start a fresh conversation and ask again. Do answers agree?
4. **Spot-check edge cases**
   Test the smallest group, zero values, null fields, boundary dates. Errors hide at the edges.

Trust is a process, not a setting.

Every AI output deserves the same scrutiny
you'd give a new analyst's first deliverable.

Hands-On: Validating Prior Work

Reproducibility Test

Take analysis from Modules 1–4 and test whether it reproduces.

The exercise

1. Pick a financial analysis from Module 2 (DCF or portfolio optimization)
2. Start a **fresh conversation**
3. Give the same data and ask the same question
4. Compare: do the results match?

What you're testing

- **Reproducibility**: Same data, same question — same answer?
- **Methodology**: Did AI use the same approach both times?
- **Stability**: Small rephrasing shouldn't change the result

If the numbers don't match, that's not a failure — it's **exactly why we verify**. And it's exactly why **skills** (Module 4) matter.

Auditing a DCF Model

Take the DCF built in Module 2. Start a fresh conversation. Ask Claude to **critique** it.

What to ask

- "Identify 3+ questionable assumptions"
- "Check formula consistency — does FCF match the pro formas?"
- "Is the terminal value calculation reasonable?"
- "What's the implied terminal growth vs. GDP growth?"

Why this matters

- The most valuable use of AI is often having it **check** work, not produce it
- Cross-evaluation: use a second conversation to critique the first
- Compare original vs. revised valuations

Red-Teaming a Deployed Skill

The database skill from Module 4 is the kind of tool people **stop questioning** once deployed. Test it like any tool your team depends on.

Adversarial queries

1. Query for nonexistent data ("Sales in 2030")
2. Complex multi-table join ("Revenue by artist by country")
3. Request something not in the schema ("Customer satisfaction scores")
4. Use ambiguous language ("Best employee")
5. Calculation the skill wasn't designed for ("Year-over-year growth rate")

For each failure

- **Diagnose**: Bad SQL? Missing context? Wrong assumption?
- **Fix**: Add clarification, examples, or constraints to SKILL.md
- **Retest**: Run the query again after the fix

This is how production skills get hardened.
Every failure makes the skill better.

Three Layers of Auditability

Defense in Depth

1. **RAG**: Ground answers in source documents with citations (Module 6 preview). AI answers based on **your content**, not its training data.
2. **Show the code**: AI displays the SQL or Python it used. Analysts review the logic, not just the answer.
3. **Guardrails**: Read-only database access, restricted tables, every query logged for audit. The system *prevents* mistakes, not just detects them.

The more layers you add, the harder it is for errors to reach your team. **Same principle as cybersecurity.**

Security, Compliance, and Governance

Data Policies: Training and Retention

Before sending data to any AI provider, know the answers to two questions.

Will they train on your data?

- **Consumer tiers** (free ChatGPT, free Claude) may use your input to improve models
- **API and enterprise tiers** typically do not
- Check your agreement and settings

How long do they keep it?

- Providers retain prompts for a window (often 30 days) for safety monitoring
- Enterprise contracts may offer zero retention
- Know the policy **before** you send sensitive data

The Schema-Only Pattern

AI can analyze sensitive data **without ever reading it**. It only needs the table structure.

How it works

1. Describe tables in instructions: names, columns, types
2. Ask AI to write the query
3. AI writes SQL **from the schema alone** — never sees actual records
4. Script runs locally; output stays local

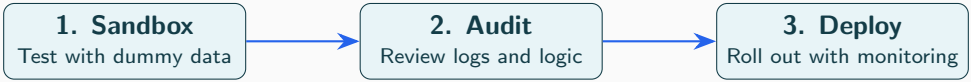
Why it's compliant

- Table metadata is **not PII** — safe to share
- Sensitive records stay on your server
- AI produces the *tool*, not the *output*
- Works for FERPA, HIPAA, SOX, GDPR

AI Adoption Roadmap

The Sandbox-Audit-Deploy Pattern

Before deploying any AI workflow to your team, follow three phases.



- **Sandbox**: Run the full workflow with synthetic data. Test edge cases. Break it on purpose.
- **Audit**: Export session logs. Verify SQL logic. Check for data leakage.
- **Deploy**: Roll out to users with monitoring. Review logs weekly.

The AI Adoption Roadmap

Each adoption stage maps to a trust level.

1. **Quick wins** (weeks) — *light verification*:
   - Chat for research, summarization, drafts
   - Upload spreadsheets for ad-hoc analysis
   - Build artifacts for one-off tools
2. **Medium-term** (months) — *test-before-deploy*:
   - Skills for recurring workflows (Module 4)
   - MCP integrations to databases (Module 3)
   - Streamlit apps for team tools (Module 7)
3. **Strategic** (quarters) — *full audit trail*:
   - Production apps with authentication
   - AI usage policy, compliance review
   - Governance framework

Exercises

Exercise 1: DCF Cross-Check

1. Take your DCF model from Module 2
2. Start a new conversation
3. Ask Claude to critique it: identify 3+ questionable assumptions
4. Revise based on the critique
5. Submit: original + revised models + summary of what changed and why

The most valuable use of AI is often having it check work, not produce it.

Exercise 2: Red-Team Your Skill

1. Take the database skill from Module 4
2. Write 10 adversarial queries designed to produce wrong answers
3. Document what breaks and fix the SKILL.md
4. Submit: original SKILL.md, adversarial queries, failures found, revised SKILL.md

Exercise 3: AI Usage Policy

Review a transcript of an AI analysis session. Then:

1. Identify potential errors, unsupported assumptions, and missing validations
2. Draft a one-page AI usage policy for a hypothetical finance team:
   - What data can/cannot go to AI providers
   - Which tools are approved
   - When human review is required
   - How session logs should be archived

Summary

Verification

- The trust spectrum
- 4-step protocol
- Red-teaming skills

Governance

- Data policies
- Schema-only pattern
- Session log auditing

Adoption

- Sandbox-audit-deploy
- Quick wins to strategic
- Trust is a process

Verify, govern, then deploy.