Claude Code for Accounting Research

Module 4 — WRDS MCP, Scale Discipline, and the Stata Bridge

John Barrios

Yale School of Management

Today's Arc

- **0:00–0:40** Lecture: MCP, architecture, iron laws, inspection, scale
- **0:40–1:15** Live demo: plain English to regression-ready funda extract
- **1:15–1:25** Break
- **1:25–2:45** Lab: two tracks — live WRDS or FSDS fallback
- **2:45–3:05** Debrief + exit ticket

Module 3 Ended With a Question

- CIKs identified, filings parsed — now attach fundamentals: leverage, profitability, size
- Compustat and CRSP sit behind three barriers a chat tool cannot clear:
  - **Authentication** — a licensed, MFA-gated WRDS account
  - **Licensing** — your institution's subscription, not yours to redistribute
  - **Scale** — hundreds of variables, decades of history
- Today's stack clears all three, in order, on a real account

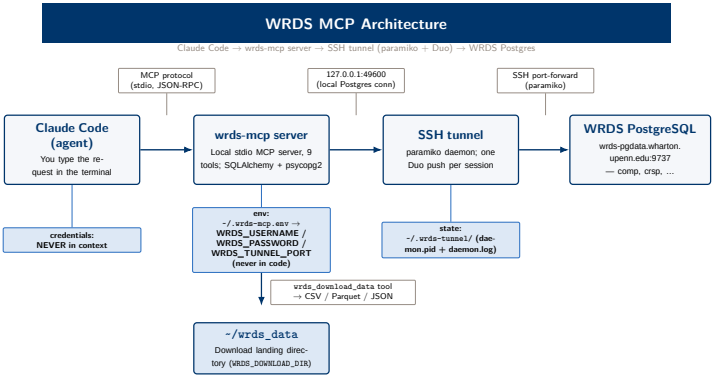
What an MCP Actually Is

- A small dedicated server does the plumbing — connection, auth, query submission — so the agent talks to a protocol, not the service directly
- `wrds-mcp`: a local Python process exposing nine named tools (`wrds_run_sql`, `wrds_download_data`, CRSP/Compustat/CCM helpers…)
- Credentials are read from the environment — never from a line of code, never from the conversation

The sentence to remember

The protocol handles auth so your credentials never touch the prompt.

WRDS MCP Architecture



Four hops — agent, `wrds-mcp` server, paramiko Duo tunnel, WRDS PostgreSQL. Credentials never enter Claude's context, at any hop.

Governance Is Not Sandboxing

The distinction that generalizes

A sandbox constrains what an agent can *do*. It does nothing to constrain what an agent can *see* once data sits in front of it.

- What actually protects the password: it never enters the chain where Claude reasons — an env file one link below, not good behavior
- Data governance is separate and legal, not architectural: your WRDS license is institutional, not yours to redistribute
- **Hard rule: zero WRDS data distribution** — not in a lab zip, not on the site, not in Slack, ever

Ritual and Hygiene

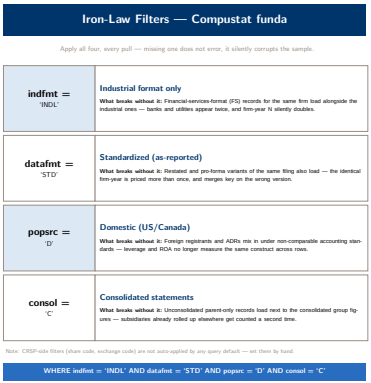
- Own virtual environment: `python3 -m venv ~/.wrds-mcp-env` — never system Python
- Credentials in `~/.wrds-mcp.env`, `chmod 600` — `.gitignore` written *before* the file exists
- Tunnel ritual, strict order: **up** → **work** → **teardown**
  - `tunnel_up.sh`, approve the one Duo push, confirm port 49600 is listening
  - Query freely — one approved session covers the whole run
  - `tunnel_down.sh` at the end — every session, not just the ones you remember

Common Failure

Pasting a password into the prompt “just this once”

Once a credential is typed into a conversation, it has been sent to the model — no cleanup afterward fully undoes that. The fix is procedural: credentials go in `~/.wrds-mcp.env`, full stop.

The Iron Laws



A query missing one of these doesn't error — it returns plausible-looking garbage.
Enforced by the wrds filter skill (SC-2).

CRSP: No Iron-Law Shortcut

- `wrds_get_crsp_returns` joins returns to names by permno and date range — and stops there
- Share-type / exchange restrictions are **not** applied automatically by any tool in this stack
- Identify which filters apply → validate before executing → execute → inspect — every time, no exceptions for “a quick test”

Inspect Before You Trust

- Four mandatory checks, under a minute total: row count vs. expectation, `.head()`/`.sample()` eyeball, null check, date-range check
- "It ran without an error" $\neq$ "it returned the right rows"

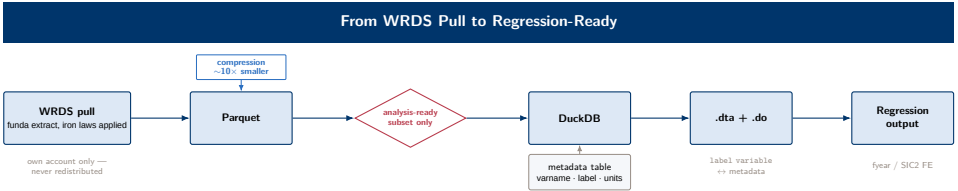
The Rationalization Table

What you tell yourself	Why it's a rationalization
"It ran without an error."	Absence of an error means valid syntax — nothing about correctness, filters, or joins.
"The row count looks about right."	"About right" isn't a number compared to an expectation written down beforehand.
"I've pulled this table before, it's probably fine."	Prior experience doesn't verify *this* query, on *this* date range, with *these* filters.
"The query worked, so it's correct."	**The most dangerous one** — collapses "executed" and "correct" into one claim that only feels the same.

Scale Discipline

- **Filter at the database, not in pandas** — `WHERE` does the work before a row crosses the wire
- **Never** `SELECT *` on a WRDS table — name the columns you need, always
- Compustat's full annual file: hundreds of variables, decades of history — never read whole into a chat window or a DataFrame

Parquet → DuckDB → Stata



Filter at the source, store compressed (Parquet), query lazily (DuckDB), export only what's analysis-ready (`.dta`). On the public fallback sample, a CSV extract compressed to roughly one-fifteenth its size as Parquet — reproducible by you, not a vendor claim.

Metadata as Context Engineering

- `dltt`, `che`, `csho`, `prcc_f` — opaque unless something on disk says otherwise
- Build once: a `compustat_metadata` table — varname, label, units, screening note
- Demo proof: close the session, open a fresh one, orient it with nothing but `SELECT * FROM metadata`

Module 1's mantra, at the data layer

Files persist; context doesn't — and it survives handing the project to a coauthor who never saw your prompts.

Feasibility-Assessment Prompting

Ask before you build — exact wording matters:

> *"I want a firm-year panel with leverage and ROA for US industrials 2000--2023 from funda. Tell me what's involved before you try anything complicated. What might be missing?"*

Surfaces the filter question and linking issues *before* a single row is pulled — not a paraphrase, the exact prompt.

What You'll See: the Query

Stylized session transcript (not a screenshot).
Plain English in, a parameterized query echoed back — no credentials, no data values, ever.

A plain-English request, and the parameterized SQL `wrds-mcp` actually issues on your behalf.

Next: Live Demo (0:40--1:15)

- Feasibility-assessment prompt on the instructor's own account (tunnel already up)
- Pull: funda extract, iron laws applied, explicit variable list, via wrds-mcp
- Inspect: all four checks, including one deliberate anomaly
- Land it: Parquet, DuckDB, metadata table — proved cold in a fresh session
- Teardown ritual, on camera

Next: The Lab --- Two Tracks

Live track

- Checkpoint-cleared students
- venv $\rightarrow$ `.env` $\rightarrow$ tunnel up
- Feasibility prompt $\rightarrow$ funda pull, own account
- Inspect, Parquet, DuckDB, metadata
- **Tunnel teardown**

Fallback track

- Duo unresolved or tunnel failure
- Identical pipeline, public FSDS-derived sample shaped like funda
- Same filters (conceptual), same inspection
- Same Parquet, DuckDB, metadata steps

The discipline is the deliverable; the data source is interchangeable. Extension (either track): the Stata bridge — export analysis-ready `.dta`, never full funda.

Verify Checklist

- `.env` exists, gitignored, `grep -r` for your password returns nothing
- All four iron-law filters present — paste the `WHERE` clause
- Four inspection checks written down with actual numbers
- Metadata table queryable; $\geq$8 variables documented
- Explicit variable list everywhere — no `SELECT *` anywhere
- No WRDS-derived file outside `~/wrds_data` / project data dir; nothing committed to git

Last, on purpose

Tunnel torn down (live track) — literally the last box on the list, so it's the last thing you check, not the thing you remember three days later you forgot.

Debrief, Exit Ticket, and Homework

- Debrief: anomalies found during inspection (collect three from the room); which iron law would have bitten whom
- Fallback-track students: did the pipeline feel different? (It shouldn't.)
- Exit ticket: standard three + write the feasibility-assessment prompt for *your own* dissertation data pull