DEMO SESSION

From Filings to a *Structured Database*

Sixteen steps. You type all of them.

Jiaqi Shao · NUS Business School

Three Hours, Sixteen Steps

You leave with a database on your own laptop that you can query.

Stage	Steps	You end up with	Checkpoint
1 Scrape	Steps 1–8	`data/filings/`	Step 8, then stop
2 Parse	Steps 9–11	100 text sections	Step 11, then stop
3 Structure	Steps 12–13	`data/filings.db`	—
4 Analyse	Steps 14–16	a finding	Step 16, then stop

three checkpoints — we all stop and sync at each one. Nobody moves on alone.

The Question We Are Answering

A policy shock you have already seen in prices. Have you seen it in disclosure?

Did firms increase **tariff-related language** in their
10-K **Risk Factors** after the 2025 tariff escalation?

- **Item 1A** of the 10-K is where firms are *legally required* to state material risks.
- Your sample: **25 trade-exposed firms** $\times$ **4 fiscal years** $=$ 100 filings.
- Today is **descriptive**. Not a causal claim — a clean fact worth explaining later.

10-K — the annual report every US-listed firm files with the SEC. Item 1A is one numbered section of it.

Before We Build: Write Down Your Prediction

A pipeline is easier to evaluate when you say what you expect before seeing its output.

Individually — 3 minutes

1. Which two sectors should mention tariffs most often?
2. Should the change appear as more firms mentioning tariffs, more mentions per firm, or both?
3. What result would make you suspect a parsing error rather than a real pattern?

Pair-share — 4 minutes. Keep the prediction; we return to it at Step 15.

prediction, not hypothesis test — its job is to expose your assumptions before the output can persuade you.

The Shape of the Whole Session

Four stages. Every text-as-data project you will run has these same four.

Scrape	Parse	Structure	Analyse
EDGAR → raw HTML	HTML → Item 1A text	text → `filings.db`	SQL → a finding

Swap in earnings calls, patents or central-bank minutes and the four boxes do not move. Only Stage 1 changes.

The database is the deliverable — the HTML and the text files are scaffolding. You throw scaffolding away.

PART ONE

Scrape

- *Steps 1 to 8. Know the API before you let it write code.*

Four Facts to Put in Your First Prompt

Each one you withhold costs a debugging round. They go in Step 3.

The fact	What goes wrong without it
CIK, not ticker, is the key	it matches ticker strings and silently picks the wrong firm
API at `efts.sec.gov/LATEST/`	it guesses an endpoint, or scrapes the human web pages
10 requests/second, per IP	the run 403s halfway and you debug it live
User-Agent: real name + email	every request is rejected; it looks like the API is down

Front-load all four — Claude needs the rate limit *before* it writes the loop, not after the 403.

Step 1 · Before You Start

Four things must already be true. Check all four now, not at Step 9.

You need	Check it with	If it is missing
a terminal	Terminal on macOS, PowerShell on Windows	—
Claude Code	`claude --version`	install it as we did in Class 2
Python 3	`python3 --version`	`python.org/downloads`
the workshop kit	`ls ~/kit` lists `firm_list.csv`	download the kit, unzip into your home folder

Windows users — run PowerShell, and swap `cp` for `copy`. Everything else is identical.

Step 2 · Make the Folder and Launch

You start in your home folder. Do not run this inside another project.

```
mkdir -p tariff-disclosure/data
cd tariff-disclosure
cp ~/kit/firm_list.csv data/
ls data/
claude
```

You should see `ls data/` prints `firm_list.csv`, then Claude Code's banner. Now type `/init` — it writes a first `CLAUDE.md`. Read that file.

`CLAUDE.md` — the file Claude reads at the start of every session in this folder — open it with `!cat CLAUDE.md`.

Step 3 · The Opening Prompt

Type it. All four facts from the Four Facts slide are in here on purpose.

```
I want to build a dataset of 10-K Risk Factors from SEC EDGAR,
to study how tariff-related risk disclosure changed 2022-2025.

What I know about EDGAR:
- every filer has a CIK (Central Index Key)
- the filing API is at https://efts.sec.gov/LATEST/
- rate limit 10 requests/second, enforced per IP
- every request needs a User-Agent header with a real name and email

The 25 tickers are in data/firm_list.csv. I want each firm's 10-K
for fiscal years 2022, 2023, 2024 and 2025.
```

Shift+Enter — new line inside one message; Enter sends. If Shift+Enter sends instead, run `/terminal-setup` once — or paste the block whole.

Step 4 · Make It Plan Before It Codes

It should stop on its own. If it does not, stop it yourself.

`Esc` *then* `Shift+Tab` *until the footer reads* `plan mode`

- `Shift+Tab` *cycles* modes. One press lands on auto-accept-edits — which you do **not** want.
- Then read the plan and check four things:

1. Does it **name the files** it will create — or is it vague?
2. Did it plan to **read** the API docs before it writes?
3. Is there a step where it **checks its own output**?
4. Is there anything in there **you did not ask for**?

a plan you approve without reading — is worse than no plan — it launders the decision.

Step 5 · Answer Its Three Questions

Type your answers in the terminal. These are research decisions, not settings.

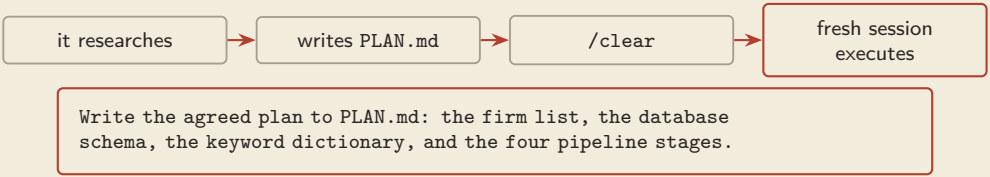
It asks	Answer	Why the choice is yours
SQLite, a flat file, or both?	**DuckDB**	you will query this for years — pick what you will use
Keyword search, full text, or LLM classification?	**keyword search**	this *is* your measurement
What entity for the User-Agent?	**your name, your email**	it is your identity making the requests

If it goes wrong It did not ask? Then tell it anyway — `use DuckDB, keyword search, and this User-Agent: your name, your email`.

you need not anticipate every decision upfront — but you must be the one who answers.

Step 6 · Send the Plan to a File, Then `/clear`

Intentional compaction: the context resets, but the decisions survive as a file.



Edit the plan, not the output — someone in the room said "add *liberation day*" — 2 April 2025 — and it entered the keyword list before any code existed.

Step 7 · New Session: Build the Downloader

Type `/clear` first. The new session knows nothing except the file.

```
Read PLAN.md. Implement Stage 1 only: download_filings.py.
Cache every file to disk, log every attempt to data/metadata.csv,
and carry on past a failure instead of stopping. Do not run it yet.
```

The clause	Why it is in there
cache every file	a re-run after a fix costs four downloads, not a hundred
log every attempt	so you can tell a missing firm from a failed request
carry on past a failure	so one bad filing does not end the batch

"do not run it yet" — read the script first — it is about to make a hundred requests in your name.

Step 8 · Run It, Politely

*The SEC counts requests **per IP**, not per person — and today we share one.*

```
Run it for the full list. Sleep 3 seconds between requests, and
skip any filing already cached on disk.
```

- **Do the room's arithmetic.** 30 of us × one request every 3s = 10/second in total — exactly the limit. At 0.15s each, we would hit 200/second and all be blocked.
- 100 filings × 3s ≈ **five minutes**. Let it run; we keep talking.

If it goes wrong Type `!cp ~/kit/filings.db data/` — the `!` runs a shell command from inside Claude Code — then tell it `data/filings.db is already built; read its schema`. Rejoin at Step 12.

403 — "forbidden" — the server refusing you, for going too fast or omitting the User-Agent.

Checkpoint 1 · After Step 8

Ask Claude to count them. Do not try to count files by eye.

```
How many .html files are under data/filings, and which tickers
have none?
```

- ✓ it answers **100**, and the list of tickers with none is **empty**
- ✓ if one is missing you know *which* — that is fine, we fix it next
- ✓ you read one permission prompt *before* you approved it

Green stickies up. Finished early? Pair with a red neighbour.

still downloading — fine — it runs while we look at what breaks.

Debrief: Is the Download Reproducible?

A folder of HTML is not yet evidence. Provenance turns it into evidence.

Field	What it identifies	Question it answers
CIK + accession	issuer + exact filing	which document did you analyse?
filing date	arrival at EDGAR	was it available after the policy event?
source URL	original location	can a reviewer open the same filing?
local path + hash	cached bytes	did the source change in your pipeline?
status + error	failed attempt	what is missing from the denominator?

Five-minute pair audit — use one `metadata.csv` row to locate the exact cached file.

TEN-MINUTE BREAK

Stop network runs. Keep your files open.

When we return: raw HTML becomes measurable Item 1A text.

When One Firm Fails: How to Diagnose It

GPS is in your list on purpose. One ticker will fail — here is the whole fix.

①	Read the log line: `no CIK found for GPS` — one line in a hundred
②	Ask Claude `why did GPS fail?` rather than assuming the code is broken
③	The answer: Gap Inc. rebranded and trades as `GAP`. CIK `0000039911`
④	Fix the ticker in `data/firm_list.csv` and re-run — the cache keeps the rest

Re-running cost **four new downloads**, not a hundred. Forty-three seconds.

Resume capability is not optional — over a hundred network calls something fails. What matters is what the fix costs.

PART TWO

Parse

- *Steps 9 to 11. Raw HTML is not research data.*

What Your Parser Is Up Against

There is no canonical 10-K. This is why Step 9 asks for several patterns.

What the filing does	Why a single pattern misses it
`ITEM 1A` in capitals, no full stop	the pattern expected `Item 1A.`
the header is wrapped in an <a> tag	the text is present but not contiguous
dot leaders in the contents page	the table-of-contents entry matches first
`Item 1A` / `Risk Factors` on *two lines*	the pattern required them on one line

You cannot write the right regex first — you write one, measure what it missed, then read what it missed.

Step 9 · The Extraction Prompt

The last two lines do all the work. Do not drop them.

```
Write extract_item1a.py.
For each cached filing, pull Item 1A (Risk Factors), ending where
Item 1B or Item 2 begins, and save it as
data/item1a/TICKER_YEAR.txt.
Try several header patterns and fall back gracefully - never stop
the batch because one filing failed.
Log every failure with the filename and the reason.
```

fall back gracefully — try pattern A, then B, then C — and record which one worked, for every filing.

Step 10 · Demand a Quality Report

Never accept "done". Ask what fraction worked, and put it in a file.

```
Write a quality report to quality_report.md: how many filings
parsed, the word-count distribution, and any extraction under 500
words. Sort ascending by word count and give each file's path.
```

You should see a number well short of 100. Mine was **112 of 120** on a wider list. The question that matters is not the number — it is **"why are the rest missing?"**

A pipeline that reports 112/120 is honest — one that reports "done" has hidden eight problems from you.

Step 11 · Read Two Failures, Then Fix

Do not ask it to "improve the parser". Give it two files to look at.

```
Open the two shortest extractions in quality_report.md and show
me the 200 characters around where Item 1A should start. Then
tell me what the pattern missed.
```

1. Honeywell and Intel put `Item 1A` and `Risk Factors` on *separate lines*
2. Allow any whitespace, newline included, between them — `[\s]*`
3. Re-run: **112 → 119 of 120**. The one holdout, 3M, needs a human

Ninety-five percent fast; the rest is judgment — 119/120 is fine for a descriptive pass. For a paper, hand-check 3M and say so.

Checkpoint 2 · After Step 11

A rate is a number. A failure is a file you can open and read.

- ✓ open `quality_report.md` — write down **your** rate, out of 100
- ✓ its first row is your **worst** extraction. Open that `.txt` and read it
- ✓ you can say *in one sentence* why that one failed

Below 90%? Red sticky — we fix it together before Part Three.

look at the data — you will not catch a systematic parsing failure from a summary statistic.

Failure Clinic: Classify Before You Fix

Different failures require different repairs. A larger regex is not a universal answer.

Symptom	Likely class	First inspection
0 words	missing boundary or no Item 1A	search raw HTML for `ITEM 1A`
40–100 words	contents-page match	inspect the next 300 characters
30,000+ words	missing end boundary	locate Item 1B / Item 2 heading
clean text, wrong firm	identity failure	check CIK and accession
plausible text, zero hits	measurement failure	inspect dictionary and case rules

Eight-minute group task — each table chooses one failed file, classifies it, and proposes the smallest diagnostic.

TEN-MINUTE BREAK

Leave your quality report on screen.

When we return: the schema becomes the research design.

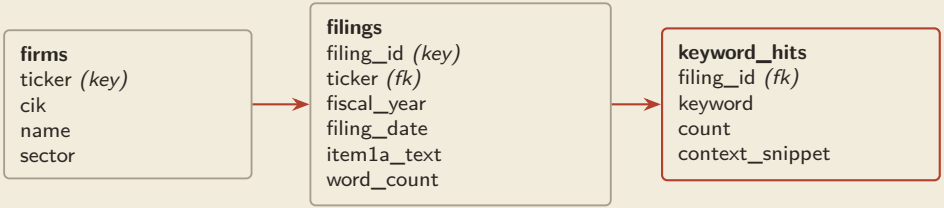
PART THREE

Structure

- *Steps 12 and 13. The schema is a research decision.*

The Schema You Are About to Approve

Three tables. Firms have many filings; filings have many keyword hits.



`context_snippet` **is the column an engineer would leave out — a count of 4 cannot tell you whether the four matches are real. "Tariff" matches inside "tariff-free".**

Schema Challenge: Which Questions Can This Database Answer?

A schema is good only relative to the questions it must support.

For each question, name the required columns and joins.

1. Did tariff language rise after April 2025?
2. Which sectors used the widest tariff vocabulary?
3. Which matches are false positives?
4. Can we reproduce the count with a revised dictionary?
5. Which missing filings could bias the yearly comparison?

Seven minutes in groups. Remove one column and explain which question becomes impossible.

deep module, small interface — three tables are enough because each table owns one kind of fact.

Step 12 · Approve the Schema First

```
Before you write any of it: show me the schema. Table names,
columns, types, keys. I want to approve it first.
```

What to check	Why
a key in every table you can join on?	or you can never merge to Compustat
the *raw text* stored, not just counts?	or you cannot add a keyword later
a snippet beside every hit?	or you cannot spot-check a single match
anything impossible to rebuild later?	that is the part to store now

Claude proposes the schema; you approve it — the reverse is how you end up with a database you cannot use.

Step 13 · Build the Database and Count

The schema is agreed. One prompt now fills all three tables.

```
Write build_db.py. Create data/filings.db with the approved
schema, load every file from data/item1a/, then run the keyword
dictionary from PLAN.md over each one, filling keyword_hits with
the keyword, its count and the surrounding sentence.
Report each table's row count.
```

You should see about 25 rows in `firms`, about 100 in `filings`, a few hundred in `keyword_hits`. **Zero** hits means the dictionary never ran — check `PLAN.md` still has it.

lost the dictionary? — the card at the back of this deck carries the full list, including "liberation day".

PART FOUR

Analyse

- *Steps 14 to 16. Now ask it something.*

Step 14 · Ask the Database a Question

```
Which firms mentioned tariffs the most?
```

What came back on my run	
manufacturing, tech hardware	most frequent, across all four years
retail	least frequent
Deere	highest throughout, rising 3 → 4 over 2022–2025
Walmart	no tariff mentions at all until 2025

your numbers will differ — different list, different years. The *pattern* should not.

Step 15 · Read Twenty Snippets by Hand

This is what `context_snippet` was for. Counts alone would hide it.

```
Show me 20 random rows from keyword_hits with their snippets,
10 from 2022 and 10 from 2025.
```

2022 — generic

"changes in trade policy could adversely affect our business"

2025 — specific

names **Section 301**, particular tariff **rates**, named **product categories**

The specificity of the language tracks the specificity of the policy — a count of "4 versus 3" would have hidden that entirely.

Step 16 · Extend the Sample to 2010

Download the 2010 10-Ks for the same firms and run the same keyword analysis. Report it beside the 2022–2025 results.

2010 vocabulary — narrow, legal	2025 vocabulary — broad, political
tariff, trade restrictions	tariff, trade tension, trade dispute
anti-dumping	trade war
countervailing duties	**liberation day**

If it goes wrong Expect fewer than 25 firms. GM re-incorporated in 2010 and Dell was private then — a data fact, not a bug, exactly like Gap's ticker after Step 8.

25 paragraphs across 18 firms in 2010 — against 26–35 per year in 2022–2025. Breadth widened, not only frequency.

Checkpoint 3 · After Step 16

The last one, and the only one that is not about software.

Write **your** finding in one sentence. It must name:

✓ what you measured ✓ over which firms and years ✓ what moved

Mine: *tariff language in Item 1A rose over 2022–2025, the vocabulary widened from legal to political terms, and some large retailers did not mention tariffs at all until 2025.*

Read it to your neighbour. If they cannot tell what you measured, it is not finished.

descriptive, not causal — a causal claim needs a design. This is the fact that would motivate one.

Capstone: Defend One Row, One Count and One Claim

Your group has five minutes to prepare and ninety seconds to report.

Show	Explain	Defend against
one filing row	CIK, accession, dates, source	wrong document
one keyword hit	term, snippet, counting rule	false positive
one yearly result	denominator and normalization	sample-size illusion
one sentence	scope, years, selected firms	causal overclaim

Listeners ask one question only: "What evidence in the database supports that?"

The deliverable is not the code — it is a claim whose path back to the source remains visible.

What You Built

Path	What it is
`PLAN.md`	the decisions, written before any code
`data/firm_list.csv`	input: 25 tickers, four sectors
`data/filings/TICKER/`	cached HTML, one folder per firm
`data/item1a/`	one `.txt` per filing — the extracted section
`data/filings.db`	**the deliverable**
`data/metadata.csv`	every download attempt, logged
`quality_report.md`	what parsed, and what did not
the four `.py` scripts	`download_filings`, `extract_item1a`, `build_db`, `analyze`
`README.md`	how a stranger re-runs all of it

Everything except `data/filings.db` is reproducible from the scripts — which is exactly why it is the one thing you back up.

Seven Things Worth Keeping

None of these are about EDGAR.

	Lesson	In one line
①	Front-load context	rate limits and API shape, before it writes code
②	Let it ask you	the design questions are yours to answer
③	Own the schema	what you store decides what you can ask later
④	Resume is not optional	caching turned a re-run into four downloads
⑤	Demand quality reports	112/120 is how you find the bug this week
⑥	DuckDB is underrated	one file, plain SQL, no server
⑦	The finding matters	a pipeline with no result is an untested one

lesson two is the new one — we let the agent interview us, instead of guessing every detail upfront.

Discussion: Three Questions

Three minutes with your neighbour. Then we take two of these out loud.

1. You extracted 99 of your 100 filings. What would you need to know about the missing one **before** you would publish the other 99?

2. Keyword counting is a measurement choice. What does it miss that an LLM classification would catch — and what **new** problem would that create?

3. Which dataset in **your own** work is currently a folder of files that should be a database?

take the third one seriously — it is the one you actually leave with.

References

[1] Baker, S. R., N. Bloom and S. J. Davis (2016). "Measuring Economic Policy Uncertainty." *Quarterly Journal of Economics* 131(4), 1593–1636. `policyuncertainty.com`

[2] Campbell, J. L., H. Chen, D. S. Dhaliwal, H.-M. Lu and L. B. Steele (2014). "The Information Content of Mandatory Risk Factor Disclosures in Corporate Filings." *Review of Accounting Studies* 19(1), 396–455.

[3] U.S. Securities and Exchange Commission. *EDGAR Application Programming Interfaces* and *Full-Text Search*. `sec.gov/search-filings/edgar-application-programming-interfaces`

[4] DuckDB Foundation. *DuckDB: an in-process analytical database*. `duckdb.org`

on the SEC docs — read the "Fair Access" page before any scraping project — it states the rate limit and the User-Agent rule.

THANK YOU

Scrape · Parse
Structure · Analyse

- *The database is the deliverable.*

APPENDIX

Reference Cards

- *Everything you will want after the room empties.*

Card: The Sixteen Steps in Order

Tear this one off. It is the whole session on a single page.

	Part One · Scrape		Parts Two to Four
1	check terminal, Claude Code, Python, kit	9	`extract_item1a.py`, several patterns
2	`mkdir`, cp the firm list, `claude`	10	quality report to `quality_report.md`
3	the opening prompt, all four EDGAR facts	11	read two failures, fix, re-run
4	force plan mode, read the plan	12	show me the schema before you write it
5	answer its three questions	13	`build_db.py`, load and count keywords
6	write `PLAN.md`, then `/clear`	14	which firms mentioned tariffs most?
7	build `download_filings.py`	15	read twenty snippets by hand
8	run it, 3s sleep, cache first	16	extend the same pipeline to 2010

checkpoints — after Steps 8, 11 and 16. Nobody moves on alone.

Card: EDGAR Endpoints and User-Agent

What you want	Where it comes from
a firm's filing history	the submissions endpoint, keyed on the CIK
the CIK format	**zero-padded to ten digits** — Gap is `0000039911`
search across filing text	the full-text service at `efts.sec.gov/LATEST/`
the document itself	the archives path, from CIK and accession number

```
User-Agent: Jiaqi Shao jiaqi.shao@nus.edu.sg
```

The header is not optional and not cosmetic — the SEC uses it to contact you before it blocks you. Verify both against the current developer page.

Card: The Keyword Dictionary

Your measurement, written down. Change it and you have changed the paper.

Group	Terms
core instrument	tariff, tariffs, duty, duties, customs
trade remedy	anti-dumping, countervailing duties, trade restrictions
policy conflict	trade war, trade tension, trade dispute, retaliation
statute-specific	Section 301, Section 232
event-specific	**liberation day**, liberation day tariff

Event terms date your sample — "liberation day" cannot appear before 2 April 2025 — which is a free validity check.

Card: Which Store, for Which Problem

DuckDB is a default, not a doctrine.

Store	Best when	Cost
DuckDB	analytical queries, one machine	**one writing process**; younger ecosystem
SQLite	many small reads and writes	row-store, so slower on wide scans
Parquet	you only scan columns, never update	no keys, no constraints
Postgres	several processes writing at once	a server to run and secure

For this project: one researcher, read-mostly, analytical queries — no server to install or administer — but close the database cleanly before you email the file, or its `.wal` goes with it.

Card: What a Causal Design Would Need

Today was descriptive on purpose. Here is the distance to a paper.

You would need	Because
a sharper timing variable	filings are annual; the shock is dated to a day
a control group	firms with no trade exposure, measured *ex ante*
exposure measured apart	import share or supplier country, not the disclosure itself
a placebo window	the same design on a year with no tariff shock

The pipeline is indifferent to which of these you add — that is the point of having built it as four separable stages.

Card: The 25 Firms

Trade-exposed by construction — which is a selection choice you should state.

Sector	Tickers
industrial & machinery	CAT, DE, MMM, HON, GE, EMR, ETN, CMI, PCAR, WHR
autos	F, GM
tech hardware	AAPL, INTC, HPQ, DELL, WDC, TXN, MU
retail & consumer	NKE, WMT, TGT, GPS, HD, LOW

25 firms × 4 fiscal years = **100 filings**. `GPS` is in the list deliberately, so the ticker failure happens to you too.

a convenience sample — fine for a descriptive pass. Say so, and say how you chose it.

Card: Common Failures and What They Mean

The six you are most likely to hit today.

Symptom	Cause and fix
`python: not found`	Python 3 is not installed — see Step 1
`ModuleNotFoundError`	let Claude run the `pip install` it proposes
`403 Forbidden`	missing User-Agent, or too fast. Add the header; sleep 3s
`no CIK found`	the ticker changed. Ask Claude to look the firm up by name
Item 1A extraction is empty	unusual header. Print the surrounding 200 characters
an extraction of 40 words	you matched the contents page. Require a minimum length

Every one of these is diagnosable by printing what you did not look at — ask the agent to show you the text, not to guess at the cause.