



TRAINING

Hands-on AI-Assisted Development

Mechanical and Electrical track Day 3

A report on the one machine to stop, and a harness you can reuse next month

Do not redistribute these materials

Redistribution of these materials is prohibited.

These materials are copyrighted by Givery, Inc. Use them only for your own learning and work reference. Do not share them outside your company, on social media or with third parties.

During the course, do not take screenshots or recordings that show these materials.



01

Today's goals

Pick 1 of the 24 machines at Plant 2 to stop. Save the prompts you use so you can call them next month.

Today: 1 machine. Next month, you reuse [how you narrowed it](#)

Out of the 24 machines at Plant 2, decide the one to stop

Fictional training data. Clean up the 83 Operations rows and list each Equipment ID with mean Operating hours, total Defects count and total Stops. You choose the conditions. Put the chosen machine and your reasons in a one-page report for next week's maintenance meeting.

One more goal

Build it for **yourself next month**, and today's steps change

What gets lost in the chat

The prompts you typed on Day 1 and Day 2 are in the chat history. Next month you would rebuild the same chat from zero. The history does not show which version worked.

What stays in the folder

Today, besides the one-page report, you save the prompts that made it as files in tpro-work. By evening you have 1 report and 6 files that produce it. Next month starts by opening the folder.

The 6 files you keep by evening

The 6 file names become **how you find them next month**

AGENTS.md

3 lines of constraints in tpro-work. Added to every later chat.

clean-operations.prompt.md

A command holding monthly Operations data preparation as one prompt.

equipment-stop-report

A skill that picks the machine to stop and writes the report. Its judgment rules and output template go in the same folder.

You build all 6, one at a time, in the exercises. No need to copy them now.

The 6 files you keep by evening (continued)

records.instructions.md

Instructions only for reading the records folder. They fix how to quote.

records-to-table

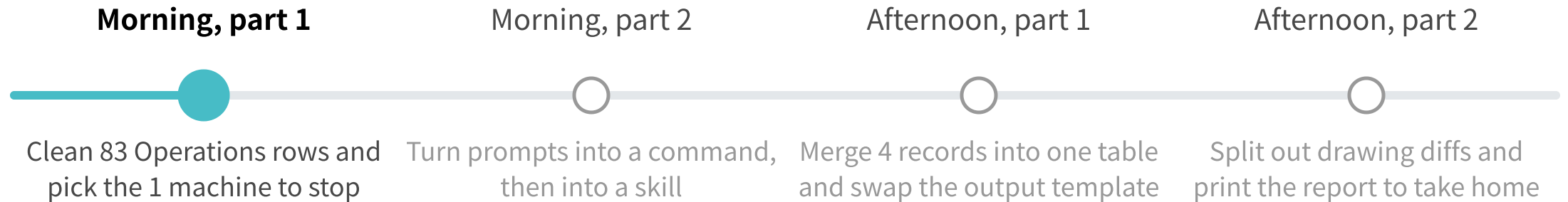
A skill that turns messy records into an 8-column table. A map of shop-floor terms goes in the same folder.

drawing-diff.agent.md etc.

Custom agents that share the work: drawing revision diffs, and checking the bill of materials against procurement requests.

Today's sequence

If the morning fails, the afternoon starts from **new material**



Opening the **wrong folder** here causes trouble all afternoon

1

Extract the ZIP and move the folders

Extract the handout ZIP and move the data and tools folders directly into tpro-work on the desktop.

2

Open the tpro-work folder

In VS Code, choose File > Open Folder and select tpro-work. If the title bar shows tpro-work, it is your workspace.

3

Look at the data folder

In the Explorer on the left, data holds equipment-data_2026Q3.xlsx, plus drawings and records.

4

Set the mode to Agent

Set the mode picker at the bottom of the chat to Agent. In Ask, requests to create files do not work.

5

Write 2 lines in decision-log.md

Create decision-log.md in tpro-work. In 2 lines, write what you want on hand to redo this next month.

If the title bar shows only a file name, start again from Open Folder.

Where Day 2 got stuck

Rewrite the blocker **as a constraint** and it keeps working

What we have done so far

Attaching the xlsx as is made Copilot ask to run a script instead of reading it, and PC restrictions blocked it.



What we do next

"Paste file contents as text first." "Do not ask for work that needs a script." Typing these 2 at the top of every chat avoids it.

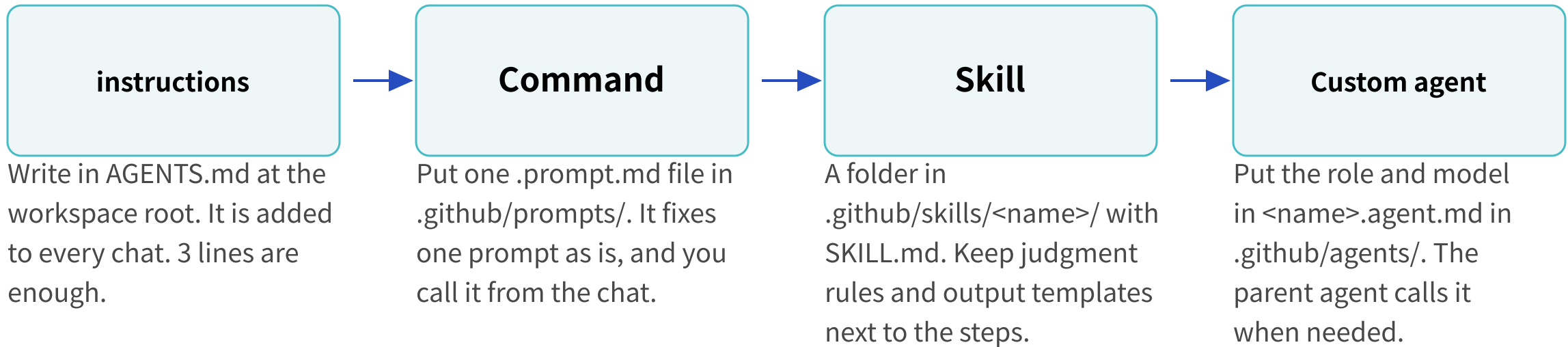


Why it matters later

Instead of typing them each time, write them once in a file at the workspace root. 3 lines that apply to every chat from today.

The 4 layers of the harness

Further left takes less effort; further right **holds more**



Comparing the 4 layers

In the wrong place, even perfect content **is not read**

Layer	Location	What goes in	How it applies
instructions	tpro-work/AGENTS.md	Premises and bans, 1 line each	Always added to every chat
Command	.github/prompts/ <name>.prompt.md	One prompt. Parts that change can be arguments	Call it by name in chat
Skill	.github/skills/<name>/	Steps and assets. Rules and templates go in the same folder	Loaded by its description. You can call it too
Custom agent	.github/agents/	Role and model. Limits on the tools it can use	Parent calls it as needed

No extra settings for any layer. SKILL.md is in capitals; AGENTS.md goes at the workspace root.

When rules or templates **must go with it**, move to a skill

Command (.prompt.md)

A single file. It holds one prompt; use it to send the same instructions in the same words every time. Adding output templates or rules makes the body that much longer.

Skill (SKILL.md)

A folder. Write the steps in SKILL.md, put rules in references and output templates in templates, and refer to them by relative path. The official docs announce that commands are deprecated and offer a way to convert them to skills.

Order of the 6 files you build today

What you build first is **used as is** in the next exercise

No.	What to build	When you build it
1	3 lines of AGENTS.md	Start of the morning. Rewrite the Day 2 blocker as a constraint
2	clean-operations.prompt.md	After narrowing 24 machines to 1, fix the prompt you used as is
3	equipment-stop-report skill	Move the report template and rules out of the command body

Once all 6 are ready, write a README last and take everything home in one folder.

Order of the 6 files you build today (continued)

No.	What to build	When you build it
4	records.instructions.md	Before reading the 4 shop-floor records, fix how to quote
5	records-to-table skill	Extract the chat that worked and add missing premises yourself
6	drawing-diff.agent.md etc.	Split drawing revision diffs and document matching into 2 roles

Write down your preamble

What you actually type now is better material than tidy text

- 1 Recall the preamble you type every time**
Recall whether you type the same explanation at the start of every chat. Anything works, like "Answer in English" or "Do not fill in numbers".
- 2 Write 1 line in decision-log.md**
Write the preamble you recalled in 1 line, word for word. No need to polish it.
- 3 Write which of the 4 layers it is**
Under that line, write which it is: instructions, command, skill or custom agent.

In the next exercise, you move that line into AGENTS.md as is.

Where AGENTS.md goes

Put it at the workspace root; it is **added from the next chat**

Directly under tpro-work	Contents
.github/	Commands and skills go here late this morning. AGENTS.md does not go here.
data/	Equipment data, records and drawings moved from the handout ZIP.
tools/	Tools such as 01_data-loader.html.
AGENTS.md	Create it here, at the same level as data. Do not write frontmatter.
decision-log.md	The 2-line note you wrote this morning.
Files up to Day 2	Leave them as they are. Do not delete them.

Do not change settings. chat.useAgentsMdFile is on by default. Turn it off only to skip AGENTS.md.

Only the form is given. Fill it in **in your own words.**

Fill in the parentheses yourself. Write each line as a "do" or "do not" rule.

- Turn files into (_____) before passing them on
- Do not ask for tasks that require running (_____)
- Do not fill in (_____) in the numbers

Keep each line to about 10 words. The longer a line is, the less it is followed.

Exercise: 3 lines of AGENTS.md

Turn what stopped you on Day 2 into **constraints from now on**.

Scenario

On Day 2, Copilot asked to run a script instead of reading the attached xlsx. Your PC could not allow it, so you got stuck. The same thing will happen today.

Steps

- 1 Create a new file directly under tpro-work and save it as AGENTS.md [1min]
- 2 Fill in the template parentheses in your own words and save the 3 lines [1min]

Result

tpro-work/AGENTS.md (3 lines, no frontmatter)

Done when

Done when

- AGENTS.md is directly under tpro-work
- All 3 lines are "do" or "do not" rules
- AGENTS.md appears in References of the chat reply

If you're stuck

- Not in References → Open a new chat. A chat already in progress may not pick it up
- Created it under .github → Move it directly under tpro-work. It is not read there
- Started looking for a setting → Leave settings alone. It is read by default

If time allows

- Add one rule under .github/instructions/ that applies only to CSV files in the data folder. Write applyTo and see whether References changes.

Pick **1 machine with evidence** for the maintenance meeting.

24 machines 1 to stop

You have 83 rows of Operations data for 24 machines at Plant 2, July to September. From them, choose the 1 machine to stop and inspect now. Each of you sets the criteria. Do not let the AI set them. This is fictional training data, unrelated to any real company or equipment.

What the Operations sheet looks like

Before pasting, **get a rough idea** of what is broken.

Where to look	Actual contents
Columns	Equipment ID / Equipment name / Month / Operating hours / Stops / Defects count
Rows	83 rows below the header, including duplicate rows and fully empty rows.
Rows 1 and 2	Row 1 is a title and row 2 is blank. The header is in row 3.
Units	Units are stored as text, as in 186.4 h and 214.2 hours.
Digits	Some cells contain full-width digits.
Month	3 formats: 2026-07, 2026/8 and R8.9 (Japanese era).
Empty cells	Found in Operating hours, Stops and Defects count.

The loader tool screen

Convert the **xlsx to text** here, then paste it into the chat.

データ読み込み 機電系 Day3

配布した Excel をブラウザで開いて中身を確認し、シート単位でコピーするための画面です。読み込みはこのPCの中だけで行われ、外部へは送信されません。

Excel を読み込む

ここに data フォルダの xlsx をドラッグして落としてください

午前 は 設備データ_2026Q3.xlsx、午後 の 自力の時間は 設備データ_2026Q1.xlsx を使います。CSV ファイルも同じように落とせます。

ドラッグがうまくいかないときは、こちらから選んでください ファイルを選択 設備デー...026Q3.xlsx

設備データ_2026Q3.xlsx を読み込みました。シートは 3 つです。見たいシートを選んでください。

稼働実績検査記録不良内訳

この表をコピー

コピーしたら VSCode の Copilot Chat の入力欄に貼りつけます。左端の番号は Excel の行番号で、コピーには含まれません。

行	A	B	C	D	E	F
1	2026年 設備稼働実績（7～9月）※研修用の架空データ					
2						
3	設備ID	設備名	対象月	稼働時間	停止回数	不良数
4	EQ-001	射出成形機A	2026-07	192.5	5	6
5	EQ-001	射出成形機A	2026/8	202.5	5	14
6	EQ-001	射出成形機A	2026/9	198.1	4	3

equipment-data_2026Q3.xlsx loaded, with the Operations tab selected

5 parts of a prompt

With all 5 parts, you can **save it as a command** as is.

Scope

State the sheet and columns to use, and the scope of the pasted table.

What to fix

Text with units, full-width digits, month formats. Name each one to fix.

What not to fix

Do not touch empty cells. Without this, they get filled with the mean.

Output format

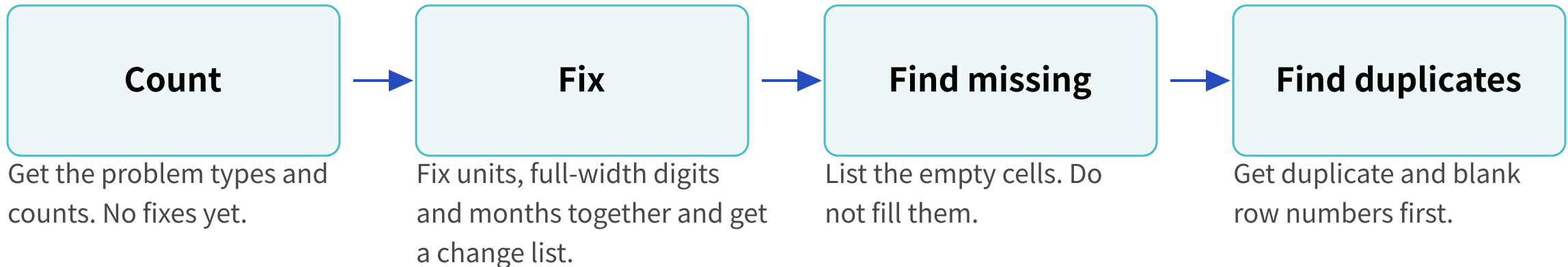
A list with 4 columns: row number, item name, value before, value after.

Save to

Save as cleaned_q3_operations.csv directly under tpro-work.

Just check that your prompt has all 5 parts. The full text is not handed out.

Ask for everything at once and you can't say **what changed**.



Exercise: Finding and fixing problems

Count before you fix, and the evidence stays.

Scenario

The Operations sheet mixes text with units, full-width digits and 3 month formats. If you have the AI fix it right away, you can't say which cell changed or why.

Steps

- 1 Copy the xlsx Operations sheet as tab-separated text in 01_data-loader.html [2min]
- 2 Paste it in the chat and get only the problem types and counts, no fixes [2min]
- 3 Have it fix units, full-width digits and months, with a 4-column change list [3min]

Result

Change list (row number, item name, value before, value after)

Done when: fixing the problems

Done when

- Problem types are listed with counts
- The change list has row numbers and item names
- Empty cells have not been changed

If you're stuck

- All clean after the first request → Ask for the change list. Unless you can say which cell changed and why, it cannot back up the report
- No list, just "Fixed" → Output format missing. Resend, naming the 4 columns
- Full-width digits remain → Name them as a target to convert to half-width

If time allows

- Apply the same prompt to the Inspections sheet and see what to rewrite to make it work

Filled-in numbers **quietly reorder** the ranking later.

Prompts that fill gaps

If you only send "Clean up the table," empty cells get the mean or 0. Later, the mean uses a different denominator and the ranking of machines to stop shifts. Reaching the report without noticing is the worst case.

Prompts that keep gaps

Write "Never fill in any values" every time. Then have it list the row numbers and item names of empty cells, and count the filled values per item. Pin down the gaps in numbers before moving on.

Exercise: Separating missing and duplicate rows

Before deleting, get a list of which rows will go.

Scenario

Aggregating with empty cells and duplicate rows left in throws off the mean's denominator. If the AI does all the deleting, no one can say how many rows were removed.

Steps

- 1 Have it list empty cells by row number and item, with valid counts per item [2min]
- 2 Before deleting, get duplicate and blank rows as a list of row numbers [2min]
- 3 Have it save to cleaned_q3_operations.csv without the listed rows [2min]
- 4 Have it report the valid row count and how many rows have Operating hours [1min]

Result

tpro-work/cleaned_q3_operations.csv

Done when: saving the CSV

Done when

- cleaned_q3_operations.csv is directly under tpro-work
- You can state the row count without duplicate and empty rows
- Empty cells stay empty, with no filled-in values

If you're stuck

- Still 83 rows → Duplicate and empty rows remain. Have it list them again
- Too few rows → Rows with missing values were deleted too. Keep those rows
- The CSV is garbled → Save it again. Finding the cause is not today's topic

If time allows

- Drop the saved CSV into the loader tool and check that it matches the original sheet
- If you finish early, open tools/02_data-export_skeleton.html and fill in the TODOs from No. 1 with AI to build a tool that turns the CSV back into xlsx.

The denominator **changes which machines** make the top 5.

Divide by all rows

Dividing every machine by 3 months makes those with only 2 months of values look lower. They were not idle, just missing records, yet they rank high as candidates to stop. You still get a number, so unless you state how to divide, you won't notice.

Divide by valid months

Divide by the months that have values, so machines with 3 months and 2 months compare fairly. Note what you divided by in decision-log.md. It is the first question at the meeting.

Write the conditions first, or the reasons come after the fact.

1

Write 3 conditions

Write 3 conditions you set yourself in decision-log.md, 3 at most.

2

Send to the AI

Pass the 3 as written and have it list machines that meet all of them.

3

If it is not 1 machine

Write that you could not narrow it down. Do not loosen conditions to force a result.

4

List measured values

Have it show measured values per condition next to the overall mean in a table.

The 3 conditions on the next page are the instructor's sample. Copying them is not deciding for yourself.

The instructor's 3 sample conditions

Replace the sample with **numbers you check at your plant**

Mean operating hours below the overall mean

The denominator of the mean is the number of months with values.

Total defects in the top 5

The top 5 when the 24 machines are sorted by total defects, highest first.

10 or more stops in total

Based on the sum of stops over the 3 months.

3 conditions at most. You may swap one for what you would check at your plant.

Exercise: Aggregating and narrowing down

You set the conditions; the AI only gives numbers.

Scenario

Aggregate the saved CSV by Equipment ID and narrow it to 1 machine by your own conditions. Adding conditions after seeing the ranking bends the reason to the conclusion.

Steps

- 1 Get mean operating hours, total defects and total stops per Equipment ID [2min]
- 2 Sort by lowest mean and by most defects; have it note how the top 5 differ [2min]
- 3 Send the 3 conditions you wrote and have it list machines meeting all of them [2min]
- 4 Have it show measured values per condition next to the overall mean [2min]
- 5 In decision-log.md, write your machine, 2 dropped ones and 2 denominator lines [5min]

Result

8 lines added to decision-log.md (3 conditions, 1 chosen, 2 dropped, 2 on the denominator)

Done when: narrowing down

Done when

- The 3 conditions are in decision-log.md before you ask the AI
- Mean operating hours use the number of valid months as the denominator
- decision-log.md has 8 new lines

If you're stuck

- Several machines left → That's fine. Admit loose conditions; write you couldn't narrow it
- Too many machines above the overall mean → The denominator is all rows. Resend the second half of the prompt
- The AI picked 1 machine → Do not let it choose. Decide yourself against the conditions

If time allows

- Do the same with the Inspections sheet. The Result column has variants and errors. Judge by the 9.9-10.3mm spec and measured values, or you miss out-of-spec lots.

The meeting asks about **the denominator** and the dropped ones.

1

Conditions: 3 lines

Write the conditions you set, in the order you set them.

2

Chosen machine: 1 line

Write the Equipment ID, Equipment name and your reason in 1 line.

3

Dropped candidates: 2 lines

Write the 2 near misses and which condition each one failed.

4

Denominator: 2 lines

Write what you divided the mean by, and why you did not divide by all rows.

Recovery slot

Now we bring **everyone to the same point.**

Catching up

If you have not saved `cleaned_q3_operations.csv`, send the instructor's prompt as is to create it.

Not narrowed down yet

You may stop after writing the 3 conditions. Add the rest to `decision-log.md` later.

Ahead of schedule

Apply the same prompt to the Inspections sheet and see what to rewrite to make it work.

In the second half, you save the prompt you just used. Without the CSV, there is nothing to save.

Break

The second half starts with **saving your prompt to a file.**

Keep these open

3 items: tpro-work, cleaned_q3_operations.csv and decision-log.md.

What not to carry over

Even if you have not finished narrowing down, the second half starts with other work.

Stop your work here. Next is a hands-on exercise.

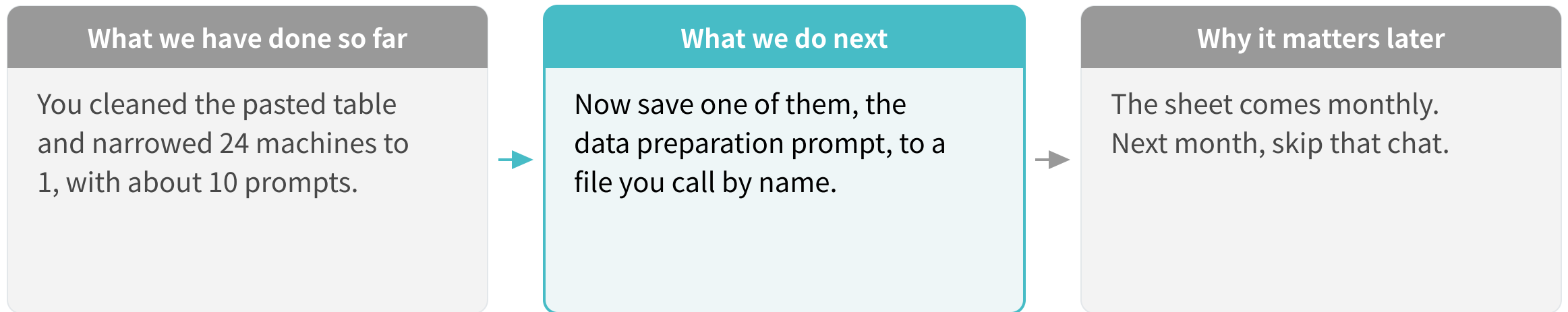
02

Turning prompts into commands

Save the prompt that just worked to a file so you can call it with a slash.

Rebuilding the same chat

One-off work doesn't need this. Save only **monthly tasks**



Exercise: Drafting the prompt

Before the AI writes it, draft one **on your own**

Scenario

Apply the same data preparation to the monthly Operations sheet. Do not send it yet. Write it directly in the editor.

Steps

- 1 Create draft-prompt.md in tpro-work [1min]
- 2 Recall the prompts you used and combine them into one [4min]
- 3 Do not send it when done. Compare it with the sample on the next page [1min]

Result

tpro-work/draft-prompt.md (your own prompt)

5 parts of a prompt

With these 5, next month you can call it **without rewriting**

Scope

Which sheet and columns. Name them, like the 6 Operations columns.

What to fix

Text with units, full-width digits, month formats. List what to fix.

What not to fix

Do not fill empty cells. Missing this changes the mean's denominator.

Output format

Column order and what to report, with valid rows and item counts.

Save to

The file name to save, with the extension, like cleaned_operations.csv.

The instructor's sample comes next. Do not copy it; compare it with yours.

Count for yourself how many parts were missing

Part	In the sample	In yours?
Scope	Paste the Operations table, then state the task	☐ / ☐
What to fix	Name the 3: unit text, full-width digits, month formats	☐ / ☐
What not to fix	The line "Do not fill empty cells"	☐ / ☐
Output format	List of fixed cells, valid row count, count per item	☐ / ☐
Save to	The file name cleaned_operations.csv	☐ / ☐

Write the number of × marks in decision-log.md. The more you have, the more the command helps.

The prompt that works here is **what you save next**

Scenario

Add the missing parts before sending. You may also send it without them. The difference in the reply becomes your material.

Steps

- 1 Add the missing parts to draft-prompt.md [1min]
- 2 Copy the Operations sheet in 01_data-loader.html and send it with your prompt [2min]
- 3 Note the returned valid row count and the count per column in decision-log.md [2min]

Result

cleaned_operations.csv, and the 2 lines of counts in decision-log.md

Have Copilot turn your last prompt **into a command as is**

- 1 Stay in the chat you just used**
Do not open a new chat. Continue the chat where you sent the draft-prompt.md text.
- 2 Write it after /create-prompt**
Type /create-prompt, then on the same line ask Copilot to turn the prompt you just sent into a command.
- 3 Add the name and description on the same line**
Add name clean-operations and description "Use for monthly Operations data preparation", then send.
- 4 Choose Workspace as the save location**
This creates .github/prompts/clean-operations.prompt.md.

If create-prompt is not in the list, type /create-skill followed by the same request. This creates .github/skills/clean-operations/SKILL.md.

The file now starts with **a block between --- lines**

Key	What it is for	What we write
name	Name after the slash. Default: file name	clean-operations
description	Short text in the suggestion list: what it is for	Monthly Operations data prep
argument-hint	Hint in the chat input box: what to pass	<sheet name>
agent	Which mode to run in: ask, agent or plan	Omitted
model	Model to use. Default: the selected model	Omitted
tools	List of tools it may use	Omitted

All 6 are optional. Filling in the top 3 is enough this time.

Tip

How it differs from the old format

If you see **mode:** on your screen, change it

Old format

It says mode: agent. Articles and blogs up to around 2025 use this form. If you copy it, it does not work as front matter and your setting is ignored.

Current format

Use agent:. The current VS Code docs use this form. If a participant's screen shows mode:, have them change it on the spot.

See for yourself where and how your text was saved

1

Open in File Explorer

If you chose Workspace, it is in `.github/prompts/`. You can open it directly.

2

Read the contents

It has front matter between `---` lines and a body below. Your text is in the body.

3

If you saved it to your user profile

Run `Configure Prompt Files...` from the Command Palette. The screen lists the source locations.

4

Do not memorize the path

The location depends on your setup. Check it on this screen each time you need it.

Press `Ctrl+Shift+P` to open the Command Palette

Exercise: Calling it from a new chat

Whether you get **the same result** shows if it is locked in

Scenario

In the same chat, Copilot answers with the earlier conversation in mind. Then you cannot tell if it is locked in. Try it in a new chat.

Steps

- 1 Open a new chat [1min]
- 2 Paste the Operations sheet, type /clean-operations and send [2min]
- 3 Compare the valid row counts from typing and from the command side by side [2min]

Result

One line in decision-log.md on what differed between typing and the command

Only the sheet name changes each time. Make it **an input slot**

The outline of clean-operations.prompt.md. It takes the target sheet as an argument.

```
---
name: clean-operations
description: Use for monthly Operations data preparation
argument-hint: <sheet name>
---
The target sheet is ${input:sheet:Operations}.
If no sheet name is given, use the Operations sheet.
```

The last part, Operations, is the input hint. Only the name is passed, so paste the table first.

Finally, **call it once more** and check these 5 points

Done when

- clean-operations.prompt.md is in .github/prompts/
- name uses only lowercase letters, digits and hyphens
- Typing / shows clean-operations in the list
- The typed prompt and the command give the same valid row count
- With the Inspections sheet pasted, you can note in decision-log.md where it fails

If you're stuck

- Not in the list when you type / → Open a new chat. If it is still missing, switch to /create-skill. That creates .github/skills/clean-operations/SKILL.md, but you call it by name the same way.
- The name has Japanese in it → Use only lowercase letters, digits and hyphens. Write Japanese in the description and body.
- Output changes on each call → The body lacks output format. Add it, with column names
- Cannot find where it is saved → Run Configure Prompt Files... from the Command Palette and check the source locations shown.

Done when (continued)

If time allows

- Turn the prompt you type most often into one more command. Decide the name and description first, then write the body; it goes faster.
- Change the argument-hint text, type / again and see how the input box hint changes

To be upfront: this format will **shift toward skills**

What you just built is in transition

The official VS Code docs say prompt files are deprecated and not loaded in Agent Host sessions. For now they still work on the legacy side, but that side is also announced to go away. The successor is Agent Skills, which you build in the next exercise.

Just a quick look: there is **a way to carry over** what you built

1

Open the Command Palette

Press Ctrl+Shift+P.

2

Find Migrate Prompt Files

It appears in the list.

3

Look at Convert to Skills...

This converts .prompt.md files to skills. Today, only check where it is.

In the next exercise, you build a skill by hand without this conversion

Recovery slot

Before the next exercise, **everyone has one command**

- 1 If you have no command yet**
Use the clean-operations.prompt.md the instructor posts in the Zoom chat [2min]
- 2 Where to copy it**
Put it in your tpro-work/.github/prompts/.
- 3 What to check**
In a new chat, type / and just check that it appears [2min]
- 4 Why we sync up here**
The next exercise starts by showing the limits of this command.

If it does not appear, open a new chat. Leave finding the cause for later

03

Moving to a skill

Move the template and judgment rules a single command cannot hold into a folder you can pass along

Adding the template every time means you lack **a place for it**

What a command holds

clean-operations.prompt.md holds one whole data preparation prompt. Call it and you get the same steps and columns. It holds only that one prompt.

What you add every time

Asking for the maintenance meeting template after the output changes the format once. A new chat undoes it. Heading count and order vary each time.

You can **store more**, so you stop adding things every time

Command (.prompt.md)

One file. It holds only front matter and a prompt, so templates and criteria must all be written into the prompt.

Skill (a folder with SKILL.md)

One folder. Besides SKILL.md with the steps, it holds rule and template files, which the body reads by relative path.

Location

Create a folder with the skill name in .github/skills/. .claude/skills/ and .agents/skills/ are also read.

Note for this setup

Skills can include scripts, but they run through the terminal and fail on PCs without Python. We use only steps and templates.

A **Japanese folder name** here stops every later exercise

equipment-stop-report

The folder name becomes the skill name. Use only lowercase letters, digits and hyphens.

SKILL.md

The steps file, named in capitals. Write name and description in the front matter, then the steps in the body.

references/

Where decision inputs go. Stop criteria go in judge-rules.md

templates/

For output templates. Replies follow the meeting.md heading order

name must be **exactly the folder name**, or it is not loaded

The top of SKILL.md. The 2 lines between the --- lines are the core.

```
---  
name: equipment-stop-report  
description: Totals Operations data, picks 1 machine to stop, and writes a report in the given  
  template. Use when monthly Operations data arrives and when making maintenance meeting docs.  
---
```

name: lowercase letters, digits and hyphens only, up to 64 characters, no hyphen at the start or end. If it differs from the folder name by even one character, it is not loaded.

Do not create folders by hand. Use the **Command Palette**

1

Press F1 to open the Command Palette

An input box appears at the top of VS Code. Type new skill there.

2

Select New Skill File...

Pick it from the list. Next, it asks where to save.

3

Choose Workspace

It is created under `.github/skills`. If you pick the personal location, you cannot use it in this course.

4

Type the skill name

Lowercase letters, digits and hyphens only. Here, type `equipment-stop-report`.

5

Check the result

It worked if `.github/skills/equipment-stop-report/SKILL.md` appears in the Explorer.

Exercise: 3 lines on the stop decision

You write which numbers mean a stop

Scenario

This morning you narrowed 24 machines down to 1. Write down the judgment you made in your head. If AI decides it, you cannot tell if next month uses the same criteria.

Steps

- 1 Write by hand 3 criteria for the stop decision: comparison with the overall mean, defects rank, minimum stops [3min]
- 2 Add line 4: if no machine meets them that month, write "not narrowed down" [1min]

Result

4 lines of judgment rules in your notes. Not a file yet

Exercise: Building the skill folder

Put the steps, judgment rules and template in one folder

Scenario

Create places for the judgment rules and template. If the folder name and name differ, it is not read as a skill, so check on screen after creating it.

Steps

- 1 Run New Skill File... in the Command Palette; name it equipment-stop-report [3min]
- 2 In the Explorer, check that the folder name matches name in SKILL.md [1min]
- 3 Create references/judge-rules.md with the 4 rule lines you wrote [2min]
- 4 Create templates/meeting.md with 5 headings for the maintenance meeting [2min]

Result

3 files in tpro-work/.github/skills/equipment-stop-report/: SKILL.md, references/judge-rules.md, templates/meeting.md

2 ways a skill gets called

A detailed description gets it used **without you calling it**

Copilot loads it on its own

Copilot loads it when needed, based on what the description says it does and when to use it. If the description is vague, Copilot misses it.

Pick it with a slash

Type a slash in the chat input box and skills appear in the list. Pick one and send to be sure it runs. You can add context after it.

How you write a reference decides if the template is read

Scenario

Write the steps in the body of SKILL.md. Do not fix the CSV name: have it use the last saved file starting with cleaned_ and put that file name at the top of the output.

Steps

- 1 Ask Copilot to write 3 steps in the body: read the judgment rules, total the CSV, output in the template [3min]
- 2 Check that references use relative paths and the CSV name is not fixed [1min]
- 3 Run /equipment-stop-report; check the file name at the top and the heading order [2min]

Result

SKILL.md with 3 steps, and one report with the CSV file name at the top

Exercise: Rewriting the description

The description decides **whether the skill is called**

Scenario

A shorter description makes Copilot less likely to load the skill on its own. See on the spot how the wording changes behavior.

Steps

- 1 Cut the description to just "Creates a report" and call it in a new chat [1min]
- 2 Restore the text that says when to use it, and call it again [1min]
- 3 Write the difference between the 2 runs in decision-log.md in one line [1min]

Result

One line in decision-log.md on the difference the wording made

Done when

Done when

- name in SKILL.md matches the folder name
- name uses only lowercase letters, digits and hyphens
- description says both what it does and when to use it
- references and templates are in the same folder, and SKILL.md refers to them by relative path
- /equipment-stop-report output follows the heading order of templates/meeting.md

If you're stuck

- A Japanese folder name does not work → name must use only lowercase letters, digits and hyphens and match the folder name. Write Japanese in the description and body.
- SKILL.md is named skill.md → The file name must be SKILL.md in capitals
- Output ignores the template → Check that the body uses relative paths with forward slashes
- Trying to put a Python script in the skill → It will not run in this environment. Build it from steps and templates only
- The skill does not load automatically → A vague description is not picked up. To be sure it is used, call it explicitly with a slash
- Path left in old setting chat.agentSkillsLocations → Delete it, or the skill is not picked up

Done when (continued)

If time allows

- Make one criterion in judge-rules.md stricter, rerun the skill, and see if the chosen machine changes. This is the first chance to confirm that the rules shape the output.

Your provisional pick

Afternoon records **change how you see** this pick's evidence

What is decided now

Your conditions either narrowed it to 1 machine or stopped at 2. If you have 2, add a condition on how to handle the denominator and see if that narrows it.

Drop one condition

The number left and the order from 2nd place change. Only removing a condition shows which one mattered.

What you have not seen yet

So far you have not seen a single record of what happened on the floor. Stops that do not show in the numbers, and differences in record keeping, come in the afternoon.

What to note

1 line for your provisional pick, 2 lines for what this decision cannot see. Reread them after lunch.

The 3 files from this morning

What helps next month is **these 3 files**, not the report

AGENTS.md

3 lines at the workspace root, added to every later chat. Constraints such as: convert contents to text first; do not fill missing values.

clean-operations.prompt.md

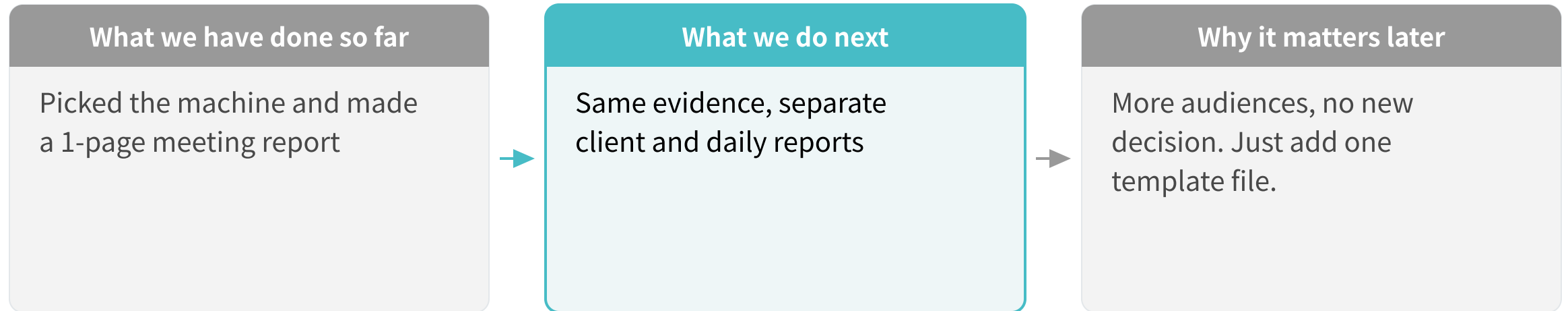
Monthly data preparation as one command. The sheet is an argument.

equipment-stop-report

A skill that picks the machine to stop and writes the report. Rules and templates are in the same folder.

Type a slash and check that both the command and the skill appear. You add to them this afternoon.

Same decision, different audiences



For the client version only, decide **what not to write** first

Maintenance: meeting.md

Target machine, supporting numbers, stop period, work during the stop, and outlook if not stopped. Equipment IDs and numbers appear as is.

Client: customer.md

No Equipment IDs. Do not assert the cause. Put actions and dates first. Dates not in decision-log.md stay blank.

Daily report: daily.md

3 lines: what, by when, and who checks. Without a line limit in the template, it grows longer.

Templates can also say what not to write. Put the client rule "do not assert the cause" here.

What switches the template is **the condition in the body**

Two additions to SKILL.md: the front matter above, and the body text below.

```
argument-hint: meeting | customer | daily
```

```
If the argument is meeting, follow the headings of templates/meeting.md;
```

```
if customer, templates/customer.md;
```

```
if daily, templates/daily.md.
```

```
If no argument is given, use templates/meeting.md.
```

argument-hint only shows how to write the argument. The condition in the body does the switching.

Exercise: Client and daily report templates

Headings alone do not remove what you want removed

Scenario

Split the same evidence into a version for outside the company and one for the morning briefing. You decide what the client version leaves out.

Steps

- 1 Create templates/customer.md. Besides headings, write 2 lines (no Equipment IDs, do not assert the cause) and 1 line (leave dates blank if unknown) [9min]
- 2 Create templates/daily.md with a 3-line format and 1 line capping it at 3 lines [5min]

Result

customer.md and daily.md in templates

Exercise: Accepting an argument

Without **a switching condition** in the body, they are not called

Scenario

Even with 3 templates, the output stays meeting until the SKILL.md body has a condition.
Add the switching condition to the body.

Steps

- 1 Ask Copilot to add a condition to the SKILL.md body that picks the template [3min]
- 2 Check that argument-hint was added to the front matter [1min]
- 3 Call /equipment-stop-report with no argument and check it uses meeting [2min]

Result

SKILL.md that accepts an argument

Exercise: Calling with 3 templates

Leave the decision alone and switch **only the output** 3 times

Scenario

Call the same skill 3 times. The decision stays the same; only the output changes.

Steps

- 1 Call it with meeting as the argument and save the output [2min]
- 2 Call it with customer and have Copilot check that no Equipment IDs remain [4min]
- 3 Call it with daily and count that it fits in 3 lines [3min]

Result

3 outputs, one per template. Keep them for the comparison.

Exercise: Comparing the outputs

Sometimes it matters more to **leave a field blank**

Scenario

Side by side, the 3 outputs show what each template dropped. Separate what may be dropped from what must stay.

Steps

- 1 Have Copilot list the numbers common to all 3 and what each template dropped [2min]
- 2 Of the dropped information, write 1 line on what must stay [1min]
- 3 Add a "what to report by when" field to customer.md and call customer again [2min]

Result

3 lines added to decision-log.md, and customer.md with the new field

Template swap checkpoints

Done when

- templates has 3 files: meeting.md, customer.md and daily.md
- Calling with no argument uses the meeting.md template
- The customer output shows no internal Equipment IDs
- The daily output fits in 3 lines
- meeting and daily show the same supporting numbers

If you're stuck

- The template does not change with an argument → The SKILL.md body has no condition. argument-hint alone does not switch it
- Equipment IDs remain in the customer output → Headings in customer.md are not enough. Write it as a rule in the body
- meeting and daily show different numbers → It is recalculating. Check that the body says to always use the cleaned CSV
- daily runs over 3 lines → The template has no line limit. Add one to it

If time allows

- Instead of adding a 4th template, add the line "do not assert the cause" to customer.md and see how the output changes. Templates can hold writing rules, not just headings.

If **only meeting** works, you can move on

1

Copy the instructor's SKILL.md

If your templates do not switch, replace that part of your file with the SKILL.md section the instructor pastes in the Zoom chat.

2

Run meeting once

Call `/equipment-stop-report` with no argument. If you get the meeting.md headings, you are done.

You can add customer and daily later. Do not spend too much time here.

04

Merging shop-floor records

Turn the daily inspections, fault history, handover memo and phone memo into one 8-column event list.

Clean tables and shop-floor documents

They are messy not because the writers were careless, but because they were never meant to be tables

This morning's equipment data

One row per machine, fixed columns, units in column names. You can see where values are missing. AI reads meaning from row and column.

The 4 documents ahead

A line may be a day or a single event. Dates are written 6 ways, results 6 ways. From July 16, even the item names change.

The actual daily inspection record

It is in records in the handout folder **exactly like this**

Excerpt from daily-inspection_2026-07.md. This is fictional training data.

```
7/1(Wed) Sasaki Power ○/Hydraulic 14.2MPa/Air 0.5/No abnormal noise/No oil leak No abnormality
July 2 Sasaki Hydraulic 14.0 Air 0.52MPa no abnormal noise OK
07/03 Sasaki Hydraulic 13.8MPa, Air 0.49, EQ-004 cutting oil seeps. Wiped, watching. Else OK
7/9 Sasaki Hydraulic 14.1MPa Air 0.5 ○
■July 16 (Thu) Muraoka Pressure 14.4 / Air pressure 0.5 / Sound no problem / Overall OK
```

July 20 was a day off, so there is no record. There are 30 lines, not 31.

Inconsistent notation in the 4 documents

Split synonyms mechanically and the counts stop matching

What varies	How it actually appears	Normalize to
Date	7/1(Wed) / July 2 / 07/03 / 2026/7/6 / 7.11 / full-width 7 / 9	2026-07-01 format
Result	No abnormality / OK / ○ / No problem / No anomaly / Watch	Normal or other
Duration	40 min / 0.5h / an hour and a half / 1.5h / 1 hr 20 min	Number in minutes
Pressure unit	14.2MPa / 14.0 / with full-width 14.1M P a	Value; unit in header
Item names	Hydraulic, Air, Noise / Pressure, Air pressure, Sound (July 16 on)	Map to one column

No abnormality and No anomaly differ only in wording; they mean the same

Written **for readers in the same workplace**, so an outside tool lacks the clues it needs

Shorthand is for insiders

"That bracket", "the Q3 issue": clear to writer and reader. For a ledger, replace them with unique names such as drawing numbers.

Old text is not replaced

The phone memo fixes errors in place, like "→ wrong, it's actually". If you pick up the cancelled part, wrong lot numbers and counts remain.

Format varies by writer

After the July 16 handover, Hydraulic became Pressure and Air became Air pressure. Only those who know of the handover can tell they match.

Gaps are invisible

July 20 was a day off with no record. With no blank line or note, skimming shifts the dates by 1 day.

Exercise: Skimming the 4 documents

Find the inconsistencies with your own eyes first

Scenario

The records folder holds 4 files: daily inspection record, fault history, handover memo and phone memo. All are fictional training data, including company, staff and equipment.

Steps

- 1 Open the 4 files in VS Code and skim them [2min]
- 2 Write 2 spots that are not in table form in decision-log.md [1min]
- 3 Find how many writers there are and where they change [1min]

Result

2 lines on inconsistencies in decision-log.md

Without applyTo, it works **only when attached by hand**

Save it as tpro-work/.github/instructions/records.instructions.md. The part between --- lines sets the scope.

```
---  
applyTo: 'data/records/**'  
---  
When quoting, give the original file name and item number.  
Convert dates to YYYY-MM-DD and keep the original form in the source column.  
Expand only abbreviations listed at the end of the handover memo; leave others as they are.
```

3 lines are enough for the body. The longer it is, the harder it is to tell which line worked.

No name in **References** means your instructions were not read

With a records file open

References above the answer lists records.instructions.md. Only then do you know the file is in the applyTo scope.

With only files outside the scope open

The same question does not show it in References. Seeing this once helps you find the cause faster when it does not apply.

Exercise: Creating the instructions file

Fix how to quote first, then work on the 4 files

Scenario

Next you turn the 4 files into a table. Put the source format and date format in the file, not in each prompt.

Steps

- 1 Create tpro-work/.github/instructions/records.instructions.md. Point applyTo at records [2min]
- 2 In the body, write 3 lines: quoting, date format and abbreviations [1min]
- 3 With a file in records open, ask one question and check that References shows the file name [1min]

Result

tpro-work/.github/instructions/records.instructions.md

Exercise: 2 records into 7 columns

Without set columns, the 2 files come back as different tables

Scenario

Put the 25 fault history entries and 30 days of daily inspections into the same 7 columns: Date, Equipment ID, Equipment name, Symptom, Action, Duration, Assignee.

Steps

- 1 Have Copilot put the fault history in 7 columns, with Duration in minutes [2min]
- 2 Have it list rows whose Equipment ID and name do not match. Do not fix them [1min]
- 3 Have it put the daily inspections in the same 7 columns. First have it write the date the item names changed and the old-to-new mapping [2min]

Result

2 tables with 7 columns (fault history 25 rows, daily inspections 30 rows)

Exercise: Cross-checking the 2 tables

Side by side, gaps you'd miss in one table show up at once

Scenario

Line up days marked No abnormality in the daily inspections with days in the fault history. Some days do not match. Count how many yourself.

Steps

- 1 Have it list days marked normal in inspections that are also in the fault history [2min]
- 2 For those days, have it show both entries word for word. No summaries [1min]

Result

In decision-log.md, the mismatched dates and both entries

Exercise: Importing the 2 memos

The key point of this exercise: do not let it guess dates for lines without a clear date

Scenario

The handover memo and phone memo have lines with no date. Keep those apart without filling in dates, then merge the 4 files into one list.

Steps

- 1 Have it put only events with a known date into the same 7 columns. List the rest separately, as originally written [2min]
- 2 Have it list the handover memo abbreviations next to their expansions. Separate those defined at the end of the memo from the rest [2min]
- 3 Have it merge the 4 files, add a source column and save as events.csv [3min]

Result

tpro-work/events.csv (8 columns: the 7 plus source)

What didn't fit the table is today's best takeaway

Scenario

Add what events.csv left out and your own counts to decision-log.md.

Steps

- 1 Write 3 lines of information that did not fit in the table. Pick things that should not be discarded just because they do not fit [3min]
- 2 Rewrite 3 handover memo abbreviations in your workplace's terms [3min]
- 3 Have it count entries for the 1 machine you chose this morning. Write that count and the number of undated items, one line each [4min]

Result

8 lines added to decision-log.md

Done when

Done when

- applyTo in records.instructions.md points to records, and its name shows in References
- events.csv has 8 columns, with file names and item numbers in the source column
- All dates are in the 2026-07-01 format

If you're stuck

- applyTo is missing → Without it, the file does not apply automatically, only when attached by hand. Point it at records
- Not in References → No records file is open. Compare with and without a file open
- Dates are guessed → Send the "do not fill in dates" instruction again. Guessed dates change the order of events
- Duration stays as text → Spell out how to convert "an hour and a half" and "1 hr 20 min". Without numbers, you cannot total them
- It fixes mismatched equipment names → Tell it to list them, not fix them
- Abbreviations are interpreted freely → Have it use the list at the end of the handover memo. Unlisted ones stay as they are

Done when (continued)

Done when

- Undated items are in a separate list, with no guessed dates
- You compared normal inspection days with the fault history and listed mismatches
- You counted how often this morning's machine appears in the 4 files

Done when (continued)

If time allows

- Use the May fault history entries to find the drawing number of the part in the handover memo. The memo gives no number; it only says to see May in the fault history.
- Get lists with and without the phone memo fixes; count how many values change

Do not enter the next block with **no table at all**

1

Put only the fault history in 7 columns

If you have no events.csv, turn only fault-history_2026H1.md into a 7-column table. It has 25 entries.

2

Move on even without all 4

Next you save the steps, not the table. The table need not be complete.

If you are stuck, the instructor posts events.csv from recovery/ in the Zoom chat. Open it and just match the column order.

Break

The next part is long. **Step away from your desk now**

Return

We resume in 10 minutes.

Keep the chat open

After the break, we reuse this chat, where you made the 4-file table. Closing it loses the flow.

Keep VS Code open. Sleep mode is fine.



05

Turning steps that worked into assets

Save the steps that turned the 4 files into an 8-column table, so you can call them as is next month

This morning's command **grew longer** with every template

Command (.prompt.md)

One file holds one prompt. You call it by typing / in the chat. There is no place for templates or term lists, so you keep adding them to the body as needed.

Skill (a folder with SKILL.md)

A whole folder: steps plus references and templates. Call it with /, or the AI loads it when the description fits.

It goes under **.github/skills**

SKILL.md

The file name is in capitals. Start with name and description. name uses lowercase and hyphens only and matches the folder name.

references/

Material the steps read. This time, the abbreviation list glossary.md.

templates/

Output templates go here, like meeting.md in this morning's report skill.

Referring to bundled files

Use paths relative to SKILL.md. Start with the folder name, like references/glossary.md, and it is read.

Scripts can be bundled too, but they cannot run here. Include only steps and materials.

Exercise: Turning the chat into a skill

Save the steps in a folder in the order that worked.

Scenario

Turning the 4 records into an 8-column table took many rounds, and the same job comes next month. Do not have the AI write new steps; extract this chat as is.

Steps

- 1 Write 2 lines: where you had to redo work and which prompt worked [2min]
- 2 Ask Copilot to turn this flow into a skill. Name it records-to-table and have it write the steps in the order that actually worked [4min]
- 3 In the Explorer, open `.github/skills/records-to-table/SKILL.md` [2min]

Result

`tpro-work/.github/skills/records-to-table/SKILL.md`

AI does not write down **where it got stuck**

The order that worked stays Why it got stuck is lost

Generated steps list only the results, reordered for efficiency. What you learned from 3 retries and the assumptions you should have checked first are left out. Whoever follows these steps next month will get stuck in the same places. So we take time here to read and add them.

Compare the order you followed with the written order

Where to look	What to compare with your own steps
Step order	You read records first, then set columns. If columns come first, have it fix the order.
Columns	All 8: Date, Equipment ID, Equipment name, Symptom, Action, Duration, Assignee, Source
Reference paths	Does the body say to read the files in references? Check the paths are relative.
Target files	Check it does not process all 4 at once and can take one file at a time.

If vague wording remains, have it add details down to column names and item numbers

3 assumptions to add by hand

The AI **never writes these on its own**

1

Always include a source column

Have each row record its file and item number. This decides if you can trace it next month.

2

Do not guess dates

Some entries, as in the handover memo, have no date. Have it list undated ones separately.

3

Look up abbreviations in the list

Have it read references/glossary.md in the same folder first. This stops free rewording.

You decided all 3 yourself when making the table. Move them from decision-log.md into the steps.

Copy the memo's abbreviation list **into a file as is**

Shop-floor term	Meaning in the memo
TL	Line leader
QC	Quality assurance
PM	Periodic maintenance
WO	Work order
SV	Customer's supervisor
Tama	Stock
Barashi	Disassembly

Exercise: Calling it from a new chat

Pass just 1 file and see if it follows the steps.

Scenario

Skills load in new chats too. Leave this chat and test with the phone memo alone.

Steps

- 1 Open a new chat, type / and check that records-to-table is in the list [1min]
- 2 Call it with phone-memo_fault-report.txt as the target [1min]
- 3 Check that the table has the same columns as the merged 4 files [1min]

Result

3 lines in tpro-work/decision-log.md on missing assumptions

Done when

Done when

- records-to-table is under .github/skills
- The steps in SKILL.md follow the order that actually worked
- It says not to guess dates
- references/glossary.md is in the same folder and SKILL.md refers to it
- /records-to-table can process a single file

If you're stuck

- Order differs from yours → It was reordered for efficiency. Have it restore yours
- Steps too vague to use → Have it name all 8 columns
- Abbreviations expanded freely → Check the body refers to glossary.md by relative path
- It processes all 4 files → Have it take the target file as an argument

If time allows

- Think of one paper procedure at your workplace and lay out just its headings in SKILL.md form. Leave the content empty; only check that it fits.

Checking your harness

Next, we **give no steps**. First, check your files

- 1 Open prompts**
Check that clean-operations.prompt.md is there.
- 2 Open skills**
Check for 2 folders: equipment-stop-report and records-to-table.
- 3 Open instructions**
Check that records.instructions.md is there.
- 4 Add anything missing**
Save the files the instructor posts in Zoom chat there. If / does not list them, open a new chat.

Without these, you will get stuck next. Take the time to get them in place.

Do this alone now, and you can do it **alone next month**

We give no steps Only which file to look at

You get the Q1 data. Get as far as this morning using only the commands and skills you made today. If you are stuck, raise your hand. We only tell you which file to look at. We do not give prompt text or fixes.

Pick the Q1 machine using only today's harness.

Scenario

Use data/equipment-data_2026Q1.xlsx. Same 3 sheets as this morning; open it to check columns and machine count. If a command fails, fixing it is part of the exercise.

Steps

- 1 Open the Q1 xlsx in 01_data-loader.html and paste Operations as text [4min]
- 2 Run data preparation with /clean-operations [6min]
- 3 Pick 1 machine with /equipment-stop-report meeting. Check the CSV name at the top and whether the judge-rules.md criteria work for Q1 [4min]

Result

The Q1 machine and the diff of your fixed harness

What needed no fix is worth more than what you fixed

Where it stopped

One line on what you called and where the result did not match.

Which file you fixed

.prompt.md, SKILL.md or judge-rules.md. Also write the line you fixed.

What needed no fix

One line on what worked as is. That is what you can reuse next month.

You will use these 3 lines to tidy up your folders at the end of the day.

Done when: practice on your own

Done when

- You picked 1 machine from the Q1 data
- You finished without asking for steps
- decision-log.md has what you fixed or why no fix was needed

If you're stuck

- A command fails partway → Fix it. Next month's data may change shape too
- Criteria do not narrow it to 1 → Write that, and you pass. Do not bend the criteria
- You are stuck → Raise your hand. We tell you only which file to look at

If time allows

- Try the same command on quality-inspection_2026H1.csv and maintenance-records_2026H1.csv. You will see where different columns make it fail.

06

Subagents and model assignment

Split tasks with different viewpoints into 2 and run each on a lighter model

Mixing prompts that check different things **hurts both**

Do it all in one chat

You ask for the drawing revision diff, then a check of the bill of materials against the procurement request. The second task pulls focus; the first misses more. Longer chats weaken the first prompt.

Split by role

Make 2 single-role files; the parent runs each one. A child sees only its role and the files given, not the parent chat. Both run at once.

With the wrong **folder or extension**, it is not read

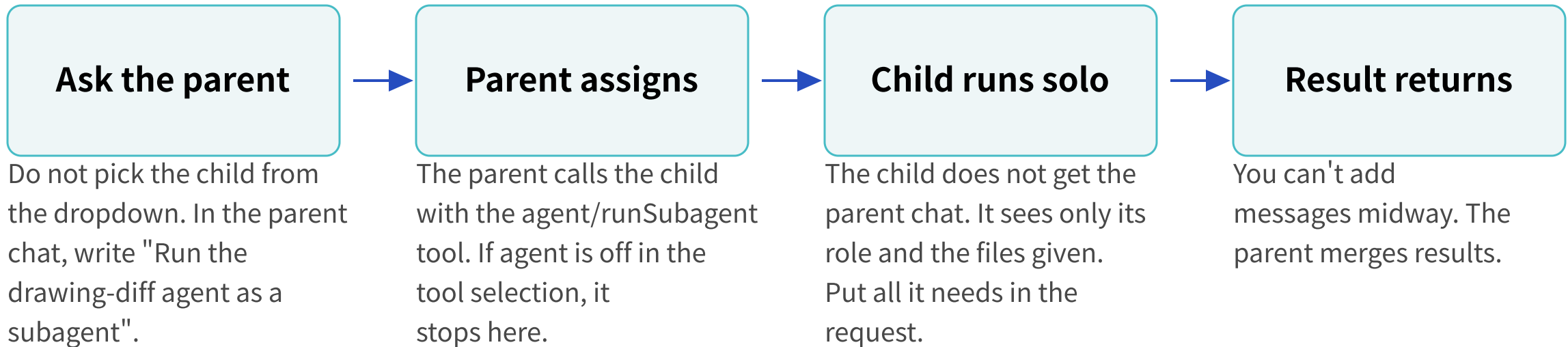
You make the 2 children. Save them as .agent.md in .github/agents and write 4 keys between the --- lines.

```
---  
name: a name with lowercase letters and hyphens only  
description: when to use it, in 1 sentence  
model: the model name exactly as shown on screen  
user-invocable: false ← only the parent can call it  
---  
In the body below, write the task in English
```

You do not make the parent. Set the mode to Agent and check that agent is on in the tool selection.

How a subagent is called

It runs **when the parent decides it is needed**



Save the assignment in a file, not in the chat

Priority	Where it is set	How we use it today
1	Named in the call	Not used today. The official example calls it by agent name.
2	The model field in .agent.md	Set it here today. Paste the name exactly as shown on screen.
3	The model selected in the parent chat	Used as is when 2 is not set

A child can't exceed the parent's cost tier. If it does, it doesn't run and returns usable models.

Keep using top models for light tasks and this drains

Billing unit

AI credits. 1 credit is \$0.01, used up by the number of tokens exchanged.

Not counted

Inline suggestions and next edit suggestions do not use these credits

How they are allotted

Each plan has a monthly credit allowance. Ask your IT department about your company's allowance

Which way it saves

Heavy parent to light child saves credits. A light parent cannot call a heavy child, so it works one way only

Write **just 2 lines** in decision-log.md

1

Work that needs a top model

Finding the cause or deciding the structure. Write just one from your own work

2

Work a light model can handle

Putting data in tables, fixing wording, tidying the format. Again, write just one

You use these 2 lines to set model in the 2 agents you create next

Exercise: Note the model names

Copy the text on screen **exactly as shown**

Scenario

A model name in an agent file has no effect if it differs from the display by one character. The list depends on org settings, so first note what this machine shows.

Steps

- 1 Open the model dropdown at the bottom of the Chat view [2min]
- 2 Copy one top and one light model into decision-log.md exactly as shown [2min]
- 3 Ask one question and check which model answered [1min]

Result

2 lines added to tpro-work/decision-log.md

Done when

Done when

- decision-log.md has 2 lines of model names
- Copied exactly as on screen, down to parentheses and version numbers

If you're stuck

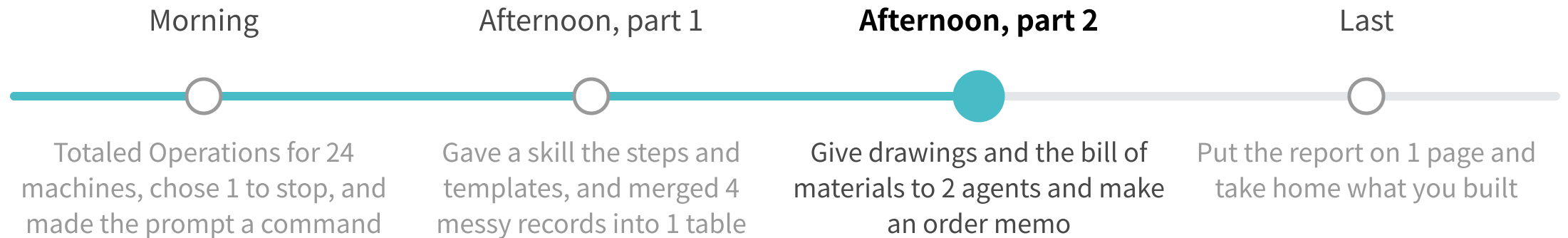
- Only one model in the list → Your organization limits it. Skip the model line and just split the roles
- Writing from memory → Copy from the screen. One wrong character breaks .agent.md

If time allows

- In the Command Palette, run Chat: Manage Language Models to see your plan's models

Break

10-minute break. Then we **start on the drawings**



AI can read them. **People make the calls**

Capability	Accuracy	Today
Reading: dimensions, symbols as text	72-84%	Agents do it; check diffs on the drawing
Reasoning: infers from read values	52-67%	Not delegated. People write the order impact
Views: matching front view and section	8-64%	Not delegated. Read section B-B yourself

Measured on MechVQA, a benchmark of mechanical drawings. General models scored 63 to 79% overall

Both **only list findings**

drawing-diff

Compares rev A and rev B and lists diffs in 4 points: dimensions, fit symbols, thread sizes and keyways. Makes no procurement decision

doc-crosscheck

Checks the bill of materials and request with the latest drawing (_svg) for mismatches in drawing no., rev, qty per unit, quantity. No verdict.

Set user-invocable: false on both so only the parent calls them

Exercise: Create the 2 agents

One role each, plus the model name you noted

Scenario

Replace 6 conveyor roller units RL-300 on conveyor line 3. Roller shaft DRW-2026-042 has rev A and rev B, and the bill of materials and procurement request have arrived. The company and parts are fictional. A wrong order wastes the downtime.

Steps

- 1 Create drawing-diff.agent.md in tpro-work/.github/agents. In model, paste the light model name from decision-log.md [4min]
- 2 Create doc-crosscheck.agent.md there too. Role: read the bill of materials, request and latest _svg, and list mismatches in drawing no., rev, qty per unit and quantity [4min]

Result

tpro-work/.github/agents/drawing-diff.agent.md and doc-crosscheck.agent.md

Always name the agent and **the document location**

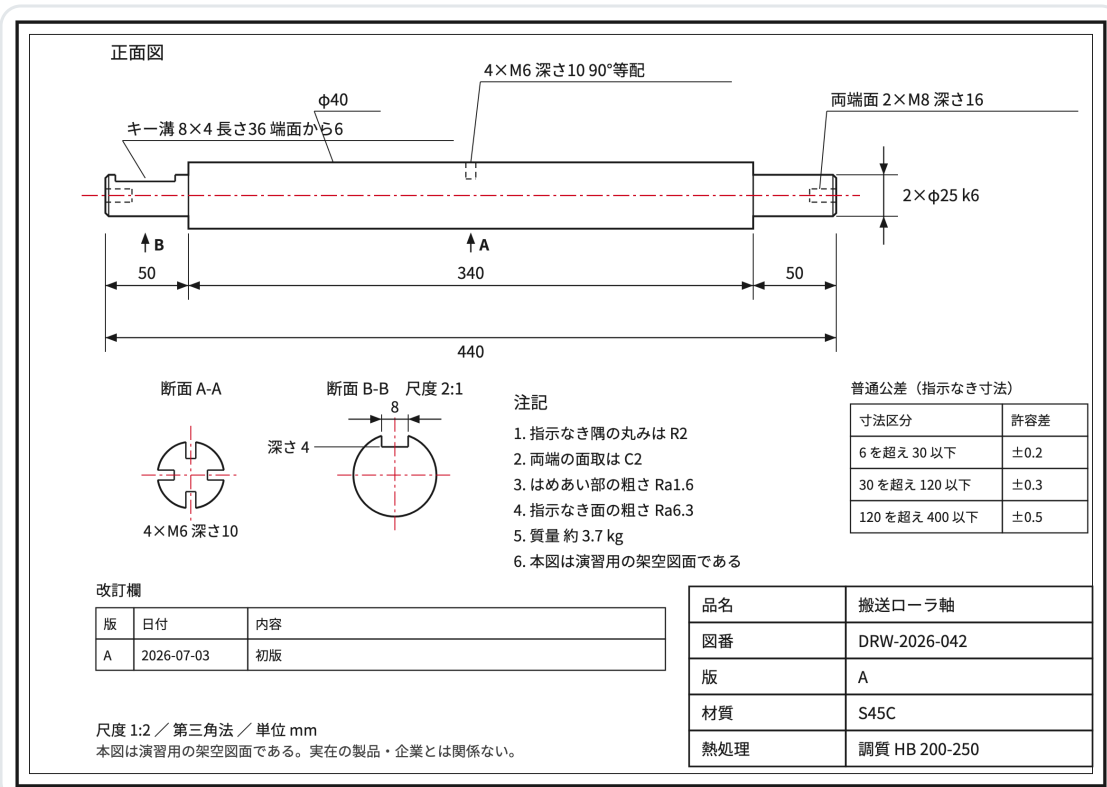
Type it as is in the parent chat. Replace < > with names and locations in your environment

```
Run the <agent name> agent as a subagent,  
and have it compare the documents in <location of the documents>.  
At the same time, run the <other agent name> agent as a subagent,  
and have it cross-check <the other set of documents>.
```

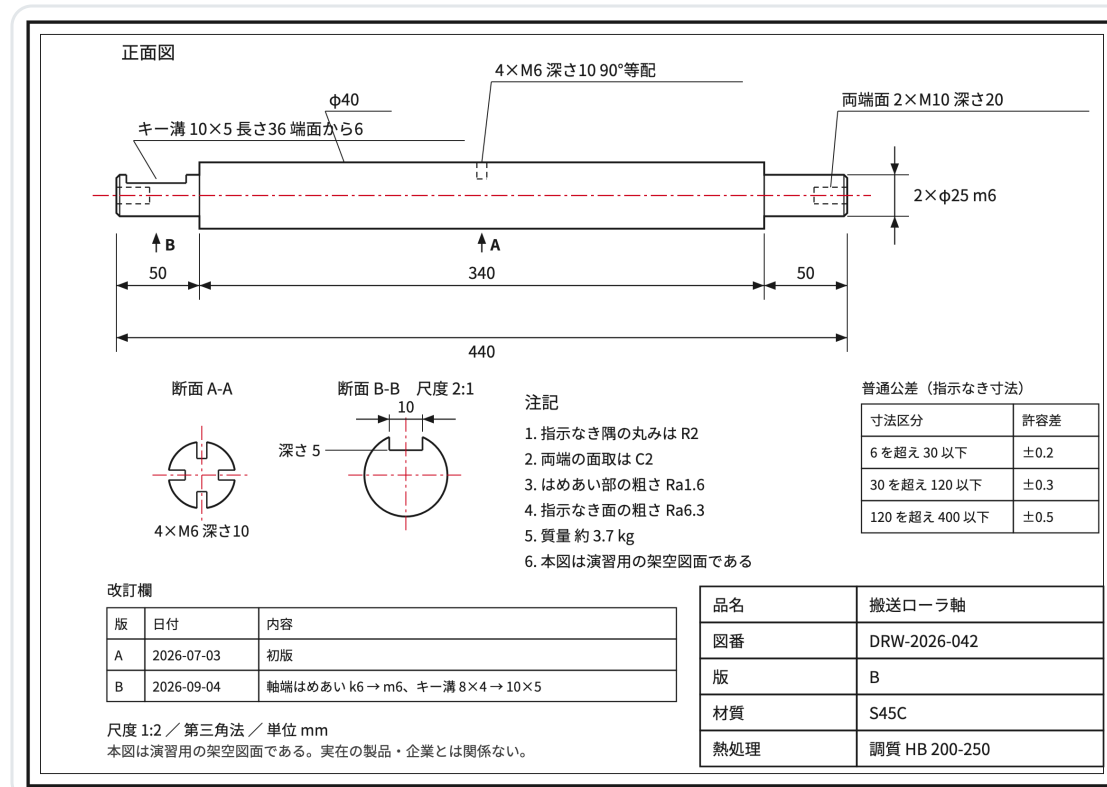
If they are not called, check that the mode is Agent and agent is on in the tools

Where to compare the drawings

You look at the PNGs; Copilot reads the 2_svg files



Old rev A



Latest rev B. Revision block at bottom right

3 things to check. **A person decides** which is right

1

Count qty per unit

List qty per unit from the bill of materials and the count on the drawing, row by row. The bill of materials rules say the drawing is right when it states the count for a machined feature

2

Check the revision

Compare the request's rev with the revision block dates; say which was latest on the request date

3

Add the basis for the quantity

The request's purpose field gives the units. Multiply by qty per unit; write the basis in 1 line

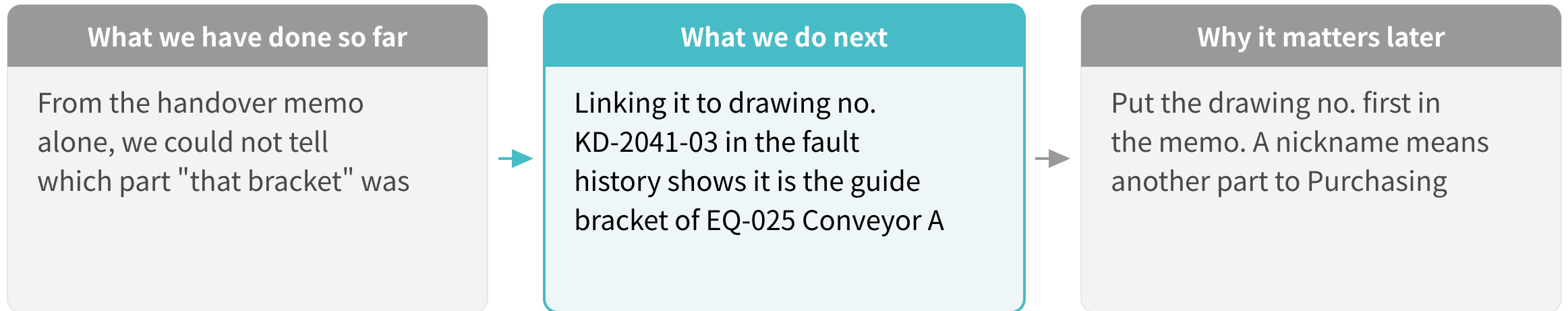
4

The parent decides which is right

Children only list mismatches. The parent chat sorts out which is right; a person checks last

If a child writes "probably", the documents do not say it. Do not let it fill in guesses

Part nicknames and drawing numbers



Exercise: Write the order memo

The parent merges both results **into 1 page**

Scenario

Both results are back. Make a 1-page order memo for the machine shop: drawing no., rev, quantity, notes. Changes not in the revision block always go in the notes.

Steps

- 1 Have it split diffs into ones in the revision block and ones not [3min]
- 2 Have the parent merge both into order-memo.md with the 4 items [2min]
- 3 Hover over the subagent section to see what that delegation used. You do not need to note the numbers [1min]

Result

tpro-work/order-memo.md

Done when

Done when

- agents has 2 .agent.md files, both with model exactly as on screen
- Revision diffs are listed in two groups: in the revision block and not in it
- Rows where the bill of materials and drawing counts differ are listed
- All 4 memo items filled; quantity has 1 line of basis from the request's unit count

If you're stuck

- Child not called → Check the mode is Agent and agent (runSubagent) is on in the tools
- Stuck loading → Have it read the _svg files, not the PNGs. PNGs are for your eyes
- Only 2 diffs → It copied the revision block. Check leader and hidden lines against rev A
- Model name in chat has no effect → Today only the model string in .agent.md counts

If time allows

- Add surface roughness symbols to drawing-diff as a 3rd check and run it again. See whether the first 2 checks lose accuracy

Handing out the order memo template

If you have none, move on with **just the template**

1

Copy only the template

Copy just the 4 headings of the instructor's memo: drawing no., rev, quantity, notes. Blanks are fine

2

Put it in the report blank

It is right to show what is not filled in. You can add what you check later

3

If it never worked

That is fine. Write it by hand and move on. The report next is the main task

The instructor pastes the order memo template in the Zoom chat



07

Report on the one machine to stop

Combine morning numbers and afternoon records on 1 page, and take today's setup home as a folder

Morning numbers and afternoon records

Side by side, you see **one side is still blank**

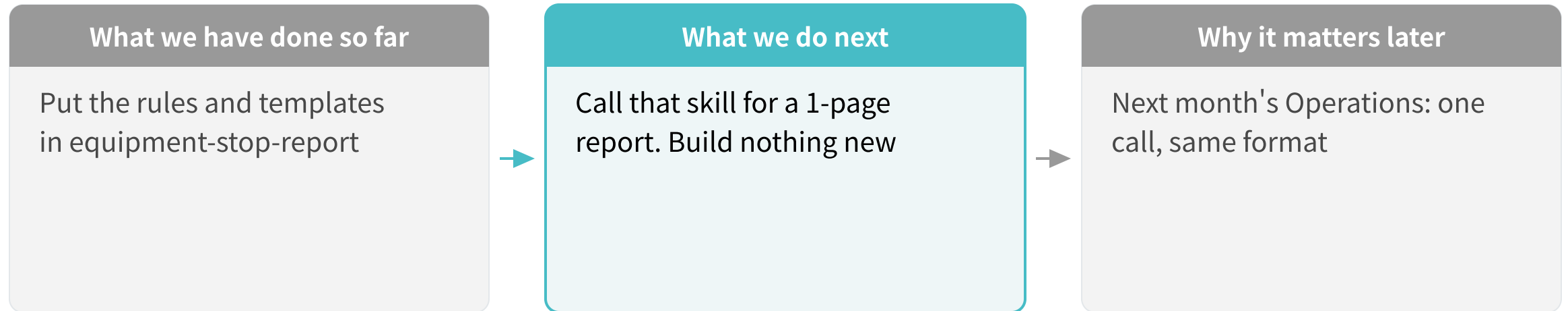
Morning: totals for 24 machines

Your morning pick and its 3 supporting numbers.
Also note whether you settled the denominator.

Afternoon: 4 records

How many entries the same machine has in the 4 records: the count you made this afternoon. The 4 records cover only some of the 24 machines

Using what you built this morning



Exercise: First page of the report

Choosing evidence is your job, so 5 minutes first

Scenario Numbers, records and the order memo: still 3 separate things. Put them on 1 page.

Steps

- 1 Pick 3 pieces of evidence: 1 each from numbers, records and drawings [5min]
- 2 Call equipment-stop-report with meeting, check that the first CSV name is cleaned_q3_operations.csv, and get page 1 [4min]
- 3 Check page 1 against the 3 lines of judge-rules.md and say aloud if it fits

Result

Page 1 of the report (on screen only for now)

Write 0 in the report **as a counted result**

The prompt template for the chat. It has 4 elements, in this order

- Target file names
- What to count (how many records this machine has)
- How to write zero (write "0" as is)
- No assertions (give 2 reasons and say you cannot tell which)

Have it count first, then tell it how to write. In reverse order, the 0 gets lost in the text

Exercise: Writing what you do not know

Where you cannot write it, **say you cannot**

Scenario

Zero records, the work during the stop, and the outlook if not stopped. Write them with known numbers only.

Steps

- 1 Tell it to give 2 possible reasons for the record count without asserting [2min]
- 2 Put drawing no., rev, qty and notes from order-memo.md into Work during stop [2min]
- 3 Tell it to write the outlook if not stopped using only known numbers [2min]

Result

A report with 3 pieces of evidence and the work during the stop

Rereading the report

1

Find unsupported claims

Have it list places in the text that assert things the data never shows

2

Turn claims into statements of fact

For judgments you keep, make clear whose judgment it is

3

List the source of each number

Have it write, one per line, which file and which item each number in the report came from

4

Delete numbers with no source

If a field becomes empty after deleting, leave it empty

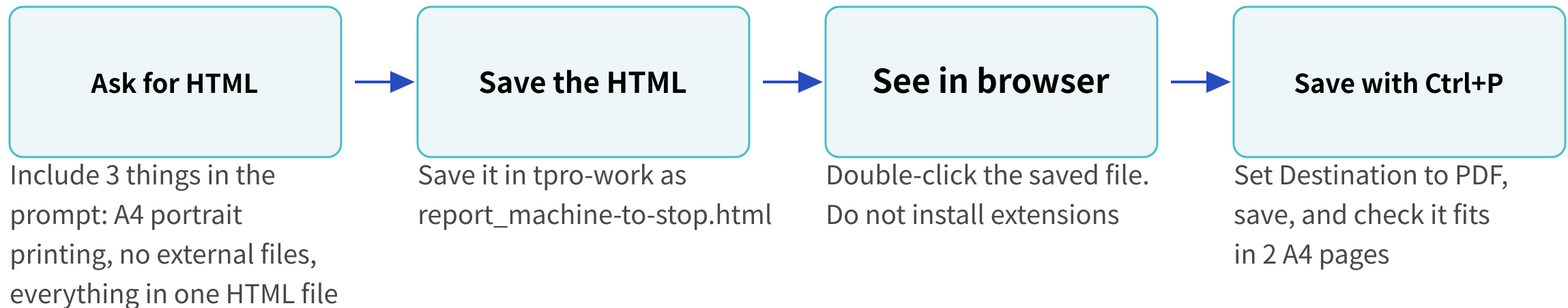
5

Save to a file

Save it as `tpro-work/report_machine-to-stop.md`

3 or fewer claims is enough. Aim for 0 and the paper says nothing

Works at your company too, **with no extra software**



What the report could not say goes in just 3 lines

Spots you almost filled with guesses, spots left with weak evidence, choices made by your own criteria.
Type them in your own words, not by AI. Next month, this is the first file you read

Done when: the report

Done when

- The report is saved to a file
- It has 3 pieces of evidence, and you can name the source file of each
- The record count is written without asserting a reason
- Work during the stop includes drawing no., rev and quantity
- Unsupported claims are 3 or fewer
- It fits on 2 A4 pages or fewer

If you're stuck

- A cost estimate appears → The handout data has no costs. Have it say it cannot give one
- Asserts the reason for 0 → No records does not prove there were no faults
- Format breaks → You added text by hand. Run it through the template again
- Numbers differ from the morning totals → It recounted in the report. Use the cleaned CSV
- Over 3 A4 pages → Run the daily template once, decide what to cut, then go back to meeting

If time allows

- Output the same content with customer and check that internal Equipment IDs and assignee names are gone. Use it as the last check before sending to a client

Tip

If you have not saved

Get it saved

Copy the chat output, paste it into a new file and save it directly under tpro-work

Thin content is fine

For now, only check that it is saved. You can improve the text after you take it home

Use the same file name

Name it report_machine-to-stop.md. The next summary checks the name

If you save it outside tpro-work, it is not in the ZIP later

What you keep

Today's order is kept **top to bottom**

Location	Contents	Use next month
AGENTS.md	3 lines of constraints: pass text, no script processing	Added to every later chat
.github/prompts/clean-operations.prompt.md	Fixed Operations data preparation	Call first on next month's sheet
.github/skills/equipment-stop-report/	Skill with 1 rule file and 3 templates. It made today's report	Call for meeting, client or daily
.github/skills/records-to-table/	Skill: messy records to an 8-column event list. With abbreviation list	For new inspection records and memos
.github/instructions/records.instructions.md	Quoting rules for records: how to write sources and dates	Auto-applies in chats on records

What you keep (continued)

Location	Contents	Use next month
.github/agents/ (2 files)	Drawing revision diffs, and bill of materials vs request	Run both when a drawing is revised
report_machine-to-stop.md and .pdf	Today's result: 3 pieces of evidence, work during stop	Bring to meeting
cleaned_q3_operations.csv / events.csv / order-memo.md	Intermediate files	Kept as sources for the numbers
decision-log.md	Decision history for the day	Read first to redo it next month

Exercise: Taking the harness home

Next month's 4 steps: **do not let AI write them**

Scenario

Today's files are spread across .github. Gather them into one set that you can follow next month.

Steps

- 1 Have Copilot list the files under .github, with each file's role in 1 line [2min]
- 2 Have it draft the README.md outline; type next month's 4 steps yourself [6min]
- 3 Open tpro-work, select everything except data, and compress it from the right-click menu. Show hidden files first so you can select .github too: on Windows, the View tab in File Explorer; on Mac, Command+Shift+. in Finder [1min]
- 4 Write 1 line on which file to edit first when you use it at work [1min]

Result

tpro-work/README.md and the take-home ZIP

Done when: taking it home

Done when

- README.md has 4 or more lines of next month's steps
- All 6 harness files are in the list
- The ZIP does not contain the data folder
- The file to edit first at work is written down

If you're stuck

- ZIP is large → The data folder is in it. Remove it
- README is still AI text → Type next month's 4 steps yourself
- Too few harness files → 6 in total passes, including ones you copied

If time allows

- Assume a shared folder: write a 1-line note on which parts of AGENTS.md and instructions the department shares.

Exercise: 3 lines for the chat

Scenario

The report is in a file. Post only the conclusion, as you would say it in a meeting.

Steps

- 1 Line 1: the Equipment ID and Equipment name of the machine to stop
- 2 Line 2: one number that supports it
- 3 Line 3: what the records showed, and the order on hold from the drawing check

Result

A 3-line post in the Zoom chat

Ended at 2 machines? Present both **with the reason why**

If you ended at 2 machines

Conditions near the sample let in 1 machine with few Operating hours values. You decide whether to add a line on the denominator to judge-rules.md.

If you narrowed it to 1

Most people reach the same machine. What differs is the denominator and the evidence number. Present both the machine and the criteria.

Completing all 3 days

**After 3 days, you have
1 report and the 6 files that produce it**

On September 26, 2026, you finished all 3 days, 24 hours. From opening VS Code on Day 1 to today's ZIP, what you keep is not what you wrote but the ability to repeat the steps next month.

Your first run next month

What to call first

When next month's Operations data arrives, first call the clean-operations command.

Where it fails

If a column name has changed, it fails there. Where it fails is where you fix the harness.

If you put it in a shared folder

Easiest setup: AGENTS.md and instructions shared by the department, skills and agents per person.

Your own data

All handout data is fictional. Before you switch to your workplace data, reread the 3 lines of AGENTS.md.

Check once more that you have the take-home ZIP.



Givery, Inc.

Teito Shibuya Bldg. 8F, 15-13 Nampeidai-cho, Shibuya-ku, Tokyo 150-0036, Japan