Data Architecture

AI for Decision Makers | LAM Research

Jed Rembold, Ph.D. Lucas P. Cordova, Ph.D.

2026-07-18

The Data Architecture module of AI for Decision Makers. Five segments: where data stacks break, what the modern pipeline is made of, building a mini-pipeline in Power Query, failures and governance, and a closing activity.

Table of contents

1 0. Meet Your Instructors

1.0.1 Jed Rembold, Ph.D.

Clinical Associate Professor of Computer Science Willamette University

`jjrembold@willamette.edu`

Astrophysicist, now happily in the data.

1.0.1.1 Background & Research

Trained as an astrophysicist. Ph.D. dissertation on near-Earth meteoroids, measured through lunar impact observations. Part of the Magdalena Ridge 2.4m telescope team before moving fully into computer and data science.

1.0.1.2 Teaching

Introductory programming, database management and storage, and Advanced Data Engineering. Works at the junction of the natural sciences and computing.

1.0.1.3 Education

Ph.D. Astrophysics, New Mexico Tech. B.S. Physics and Math, Linfield College.

1.0.2 Lucas P. Cordova, Ph.D.

Associate Professor Willamette University

lpcordova@willamette.edu

From building software to building software engineers.

1.0.2.1 Background & Research

Software engineer and team leader turned professor, developing the engineers I'd want on my team and studying what happens when AI becomes part of how we build, learn, and work.

1.0.2.2 Teaching

Teaching students to write great code, tame messy data, and solve hard problems.

1.0.2.3 Education

Ph.D. Software Engineering, M.S. Software Development & Management, B.S. Computer Science

1.0.2.4 Industry Experience

Hewlett-Packard, Elemental Technologies, SAIF Corporation

2 1. Pitfalls, Failures, and Governance

2.1 Learning Objectives

2.1.1 What You'll Leave With

By the end of this segment, you will be able to:

1. **Name** where a data stack breaks, using a concrete symptom instead of "the data is a mess."
2. **Explain** single source of truth, and point to one place yours is violated.
3. **Define** data lineage and say why it is the first thing to ask for when a number looks wrong.
4. **Ask** three questions of any team whose data your decisions depend on.

2.2 Part 1: The Monday Morning Disagreement

. . .

Two numbers. One company. Somebody is about to present the wrong one.

2.2.1 Yesterday You Were Michael Scott

Yesterday, you took Michael's job at the Scranton branch.

. . .

You had three tasks:

1. Decide which reps deserve a raise.
2. Forecast next quarter's sales by product.
3. Find the customer segments worth advertising to.

. . .

Every one of those starts with a number you trust. This morning is about what happens when you cannot.

2.2.2 Same Question, Two Answers

Michael needs one number for the board deck: **what did Scranton sell?**

2.2.2.1 The hand-kept tracker

The sheet Michael's team actually maintains. 155 rows, updated by hand, lives on somebody's desktop.

2.2.2.2 The system of record

`sales_order_lines`, the same file you opened in Excel yesterday. Built from orders that shipped and were invoiced.

Total: $43,020.24

Total: a different number.

. . .

Both are "the data." Both are defended by someone with a job title. **Which one goes in the board deck?**

2.2.3 Look at the Actual Sheet

Five real rows from `regional_manager_tracker.csv`:

Date	Client	Rep	Qty	Total $	Notes
01/05/22	Whitmore Nonprofit Services	83	19	249.36	check w/ Dwight
09/02/22	Ridgeline Nonprofit Services	5	23	365.24	net-30
07/07/22	Cascade Nonprofit Inc	5	~11	107.03	net-30
06/28/22	Redwood Financial Goup	104	~8	136.40	net-30
02/10/22	Conerstone Retail Associates	161	15	230.01	

2.2.4 Spot the Pitfalls

Nothing here is malicious. Every one of these is a person doing their best on a Tuesday.

. . .

Open `regional_manager_tracker.csv`.

Take 60 seconds. **What is wrong with that sheet?** Call them out.

2.2.5 The Pitfalls

Here is what is actually in there:

- `~11` and `a dozen` in a column that should be a number. **23 of 155 rows, 14.8%.**
- `Goup`, `Conerstone`, `Summi`. Client names that will never match your CRM.
- `01/05/22`. Two-digit year. January 5th or May 1st?
- `Rep` is `83`. An ID, with no name attached.
- **No order number anywhere.** You cannot match this back to anything.
- And about **2 in 5 of those totals are simply wrong.**

2.3 Part 2: Where It Breaks

. . .

Six pitfalls. You have met all of them.

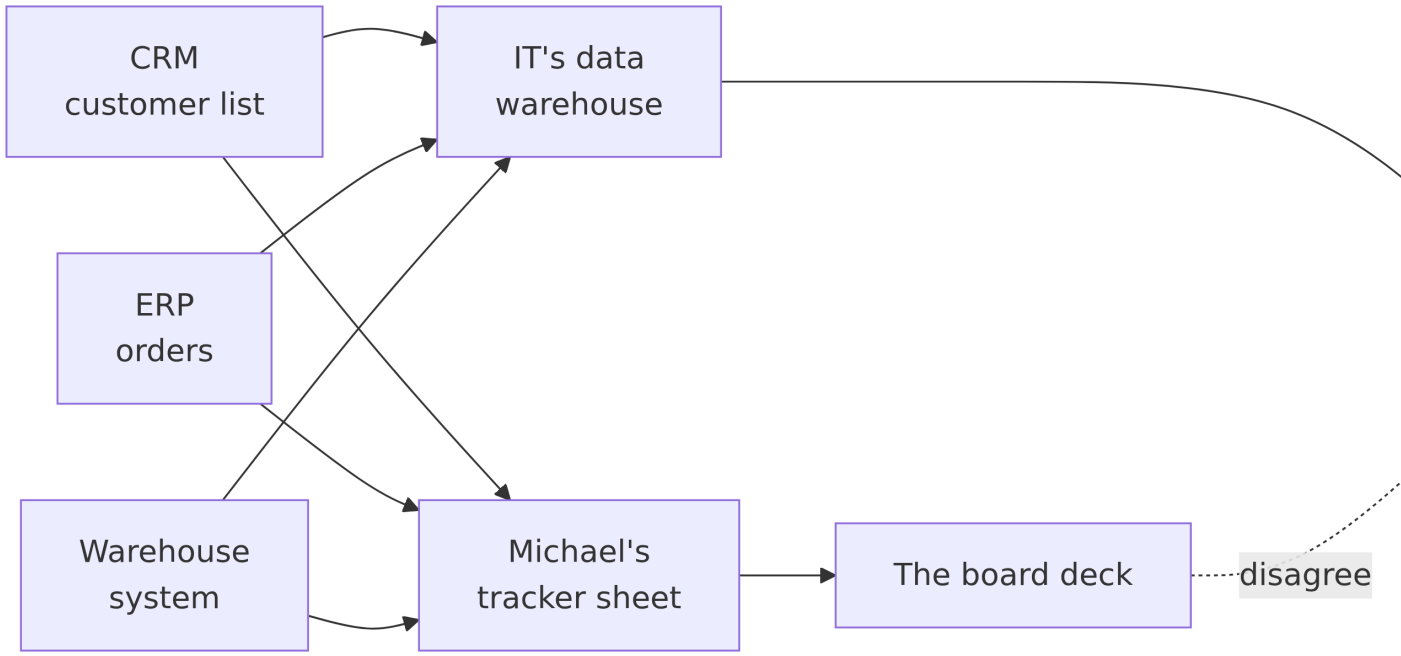
2.3.1 This Is What Messy Looks Like

Yesterday, Prof. Kitada Smalley gave you **tidy data**: one variable per column, one observation per row, one value per cell. She also showed you the messy versions.

. . .

That sheet is the messy version, in the wild, feeding a board deck.

2.3.2 The Stack Underneath



Two paths out of the same three systems. They do not agree, and **the disagreement is the deliverable** that lands on your desk.

2.3.3 Pitfall 1: No Single Source of Truth

When the same question has two owners, it has two answers.

The tracker says `$43,020.24`. The system says something else. Neither is labeled "official."

. . .

Single Source of Truth (**SSOT**) means: for any given number, exactly one system is authoritative, and everyone knows which.

. . .

You almost certainly do not have this. That is normal. **Knowing where you do not have it** is the actual skill.

2.3.4 Pitfall 2: Key Drift

To combine two systems you need a shared key. In practice, the key is missing or mangled.

. . .

In the Dunder Mifflin CRM export, of **1,923 rows**:

- **97 rows (5.0%)** have no `Acct ID` at all.
- The rest match to the customer list perfectly.

. . .

So 5% of your customers **silently vanish** from any report built on that match. No error. No warning. The total is just quietly too low.

2.3.5 Pitfall 3: Format Chaos

The same fact, written five ways, is five facts as far as a computer is concerned.

. . .

What it should be	What it actually is
A number	`$141.58` (text, with a dollar sign)
A date	`10/13/2022` here, `01/05/22` there, ISO somewhere else
A quantity	`~11`, `a dozen`
A phone number	`(682) 486-2143`, `2953356785`, `209.577.2260`
A status	`FULFILLED` vs `Fulfilled`

. . .

Every one of these is a real value in the Dunder Mifflin exports. Sorting that `AMT` column puts `$99` after `$1,000`.

2.3.6 Pitfall 4: Duplicates

Nobody sets out to double-count. It happens on export.

- CRM export: **61 exactly duplicated rows.**
- ERP export: **93 exactly duplicated rows.**

. . .

If you sum revenue off the raw ERP export, you have just billed 93 orders twice. Your number is **too high**, and it is too high in a way that looks completely reasonable.

2.3.7 Pitfall 5: No Lineage

Data lineage is the map of a number's entire journey, from the system it was born in to the cell you are looking at.

. . .

Ask of any number on any dashboard:

1. Which system did this originate in?
2. What was done to it on the way here?
3. Who changed that, and when?

. . .

If nobody can answer in under a day, you do not have lineage. You have folklore.

2.3.8 Pitfall 6: Bus Factor

The tracker lives on one laptop. One person knows the cleanup steps. Those steps live in their head.

. . .

Bus factor of one. If they take a vacation, the report is late. If they leave, the report is gone, and so is any hope of explaining last quarter's numbers.

. . .

Manual steps are not just slow. They are **undocumented by construction**.

2.3.9 Quick Quiz: Lineage

A regional revenue number on your dashboard looks 30% too high. What does asking for **data lineage** actually get you?

```
A. A backup of the old spreadsheet versions so you can recover the previous number.
B. The map of that number's journey from its origin system to the report.
C. The folder structure on the shared drive where the team's files live.
D. A list of which fields are numbers and which are text.
```

. . .

B. Lineage is the journey, not the storage. It is what lets you walk backward from the wrong number to the step that broke it, which in this case is probably those 93 duplicated ERP rows. A, C, and D are all real things. None of them tell you where a number came from.

2.4 Part 3: Three Questions to Ask

. . .

You will probably not fix most of this yourself. You will ask the team that can.

2.4.1 Most Data Failures Are Not Technical

The tracker is not broken because of bad code. It is broken because no one owns it, no one knows when it last updated, and no one checks it against anything.

. . .

That is **governance**, and governance is a management problem before it is an engineering one. You do not need to write the pipeline. You need to know what to ask the person who did.

2.4.2 The Three Questions

For any number your decision depends on:

1. **Who owns this?** One name. If the answer is "the team" or a shrug, that is your single source of truth problem and your bus factor problem in one sentence.
2. **When did it last refresh?** A figure nobody has touched since Tuesday is a different risk than a live one. Stale beats wrong, but only if you know which you are holding.
3. **What is it reconciled against?** If the answer is "nothing," you are back to the tracker versus the system. Every number you can trust is checked against something.

2.4.3 Why These Three

Each question is a pitfall turned into a habit.

Question	The pitfall it catches
Who owns this?	No single source of truth, bus factor
When did it last refresh?	Silent staleness
What is it reconciled against?	No lineage, duplicates

Three questions, asked out loud in a meeting, will save you more bad decisions than any dashboard.

2.4.4 Quick Quiz: Where to Start

Your team wants a new dashboard. What should you do first?

```
A. Inventory every data source you currently have access to and map what is in them.
B. Ask IT which tables are already modeled so you can reuse existing work.
C. Write down the decision the dashboard is supposed to change.
D. Pick the visualization tool the company already licenses.
```

. . .

C. Start with the business question, specifically the decision that hangs on it. Starting from the data you happen to have is how you end up with a beautiful dashboard full of vanity metrics that nobody acts on. A and B are the second question. D is barely a question at all.

2.4.5 What We Just Did

You now have language for a thing you already knew was broken.

Symptom you have lived	What to call it
"Our numbers do not match theirs"	No single source of truth
"Half the accounts fell out of the report"	Key drift
"It sorted wrong" / "the dates are backwards"	Format chaos
"Revenue looks too high"	Duplicates
"Where did this number come from?"	No lineage
"Only Dave knows how to run it"	Bus factor

The left column gets you sympathy. **The right column gets you a fix.**

2.4.6 The Big Ideas

1. **Two numbers for one question is not a data problem.** It is an ownership problem wearing a data costume.
2. **Broken matchs fail silently.** They do not error, they just quietly return less. Duplicates fail loudly the other way, and the two can cancel out.
3. **Lineage is the first ask, not the last.** "Where did this come from" beats "can you re-run it."
4. **Governance is three questions.** Who owns it, when did it refresh, what is it checked against.
5. **Name the pitfall, not the feeling.** "The CRM has no key on 5% of rows" gets engineering help. "The data is a mess" gets you a shrug.

2.4.7 Up Next

Next you put this to work. You will take one report you actually own and run it through the same questions, in writing, with the group.

3 2. Making it Personal

3.1 Making it Personal

The pitfalls are generic. Yours are specific.

3.1.1 Think-Pair-Share: Pitfalls

- The provided handout walks you through evaluating how some of these questions or pitfalls apply to a report or dashboard you are responsible for.
- Take 5 minutes to fill out the first two sections: identifying a report you want to analyze and thinking through what pitfalls might be applying
- Then we'll take 5 minutes to share out between partners

3.1.2 Think-Pair-Share: Lineage

- Building on the previous pitfall of lineage, it is vital to understand and break up all the discrete transformations or data movement that take place between an original data source and the final report or table
- Take another 5 minutes to break this down for your report or dashboard. What major steps does the data go through before it reaches its final destination?

- Then we'll take another 5 minutes to share between partners. Have they caught and listed all the pertinent steps?

3.1.3 Think-Pair-Share: Health Checks

- Each step of the way, from data source to final destination, it is important to work through what is happening, who owns that process, and what checks are being done on data quality
- For *each* step in your listed lineage, go through the health checklist in the worksheet, filling out each of the 5 sections
- We'll take 5-10 minutes for you to work through these independently, then give you some time to share your major findings or points of concern with a partner

3.1.4 Process Debriefing

- What are your main take-aways from this activity?
- This was the work necessary to evaluate a single report or dashboard. How does this scale to multiple reports or dashboards?

4 BREAK TIME!

5 3. Big Picture Data Pipelines

5.1 Big Picture Data Pipelines

The thing that breaks, and what it is actually made of.

5.1.1 What even is a Data Pipeline?

- Given the potential pitfalls, a good data pipeline is built to combat or insulate our data from these effects
  - A clear, clean, source of truth with well determined lineage
- To maintain this, most tasks are usually automated
  - The idea is to take data in a raw form and move it through the pipeline into a state where downstream users can easily use it to answer business questions
- As such, a pipeline:

  - Ingests information from various sources
  - Cleans it up/standardizes it, and combines it
  - Organizes for relevant business decisions

5.1.2 Mission Objective #1

- Zoomed out, creating, modifying, and managing company-wide data pipelines is likely outside your responsibilities
- Nonetheless, you likely have data or want access to data flowing through those pipelines
- May also have your own data desires or needs to communicate to the IT or data team
- So understanding a bit of the overall architecture and vocabulary used for these large-scale pipelines can be useful for communication

5.1.3 ETL and ELT

- The letters E, T, and L stand for:
  - **E**xtract: pulling the data that you want to work with
  - **T**ransform: cleaning up and standardizing that data
  - **L**oad: moving that data into a single source of truth database, commonly called a warehouse
- Historically, ETL was also the common ordering for doing these operations, with cleaning and standardizing happening before loading to the warehouse
- More recently, the pattern has largely shifted to ELT: first loading the raw data into the warehouse, and then cleaning and transforming it there
  - Largely driven by easily accessible (and scalable) compute in cloud warehouses

5.1.4 Modern Pipelines

5.1.5 Movement Timings

- Each arrow in the previous diagrams represents a movement of data
- How often should data be moved through the pipeline?
  - **Batch** loading transports entire blocks of information through the pipeline together.
    - Run on some interval, e.g. hourly or daily
    - Dashboards or reports do not get updated until the *end* of each interval
  - **Stream** loading transports individual pieces of information through the pipeline as they arrive or are generated

    * Depending on data throughput, generally more computationally expensive, and aggregates can be trickier
    * Dashboards and reports are as close to *live* as they can be

5.1.6 A Question of Schema

- Databases, unlike spreadsheets, maintain a strict *schema*, on a per-table basis
- Each column can generally hold only a single type of data
- All columns must contain the same number of rows
- Pros:
    - Downstream consumers of the data know what to expect
    - Helps eliminate egregious data cleanliness issues
- Cons:
    - Data sometimes changes form, requiring schema migrations
    - Can be inflexible if a piece of information can take on multiple forms

5.1.7 Some Quick Shopping

- Data Warehouses are terrific for serving as a single source-of-truth for an entire company or division, but can be unwieldy or potentially face governance issues of who is allowed to see what information
- A *data mart* is a small slice of the data warehouse, potentially lightly customized for use by a specific team
    - Populated after the warehouse has been updated
- Gives smaller teams a coherent and manageable source-of-truth for their specific requirements and needs

5.1.8 Interacting with a Database

- So, IT has granted you and your team access to a data mart customized for your needs. How do you use it?
- For relational databases, the primary way of interacting with the database is through the scripting language *SQL*
- SQL is a *declarative* language, so it reads more like plain-English than many scripting languages, but can still have a lot of special keywords to understand
- Keywords in SQL are followed by extra information that provides context or details to the keyword
- The keywords are not doing anything magical! Mostly just similar operations to what you might do in a spreadsheet program like Excel

5.1.9 Mission Objective #2

- While you likely don't need to often write or generate SQL yourself, you may need to view or verify written SQL from time to time
- As such, it is useful to understand how to parse different SQL queries in a context that you better understand: Excel
- So let's unpack some Excel analogs to SQL keywords, and see how they'd appear in a query

5.1.10 ExSQL?

SQL Keyword	Description	Excel Equivalent
`SELECT`	Chooses the columns to show	Hiding columns or choosing table fields
`FROM`	Tell the database what table to look in	Selecting a tab name
`WHERE`	Filters the rows based on specific conditions	Clicking the drop-down filter on a column header
`JOIN`	Merging tables together using a matching ID	Running a `VLOOKUP` or `XLOOKUP`
`GROUP BY`	Collecting and collapsing similar values to summarize	Creating a pivot table
`ORDER BY`	Ordering the rows by a particular column	Sorting a table by a column
`AS`	Rename a column or table	Renaming a column

5.1.11 Unpacking a Query

- Suppose we then have the following query. What might its equivalent actions be in Excel?

```
SELECT
  state,
  AVG(sq_ft) AS avg_ft # <4>
FROM branches # <1>
WHERE region = 'Northeast' # <2>
GROUP BY state # <3>
ORDER BY avg_ft DESC # <5>
```

① Pull data from the `branches` table
② Only keep rows that have Northeast as the region
③ Make a pivot table where each row will be a state
④ Compute the average `sq_ft` of all branch buildings across each state and rename to `avg_ft`
⑤ Order rows by decreasing size, so largest at top

5.1.12 SQL Ordering

- You may notice that the top-down ordering of an SQL query is unfortunately not really the order you would undertake those actions
- Main SQL keywords "execute" in the order of: `FROM` → `JOIN` → `WHERE` → `GROUP BY` → `SELECT` → `ORDER BY`
- The most fundamental thing to understand here is that rows are filtered before any sorts of aggregates are computed

5.1.13 Your Turn

- Yesterday, you were introduced to the Dunder-Mifflin simulated data set.
- For this exercise, you'll want to open the `employees.csv` and `branches.csv` into their own tabs in Excel
- Can you compute the same results as the following SQL queries but in Excel?

5.1.14 SQL 1

```
SELECT
    branch_name,
    salary
FROM branches
```

```
JOIN employees
    ON branches.manager_employee_id = employees.employee_id
WHERE
    branches.state = 'NY'
```

branch_name	salary
Corporate HQ	168000
Rochester	63146
Albany	50186
Buffalo	65147
Utica	63639
Yonkers	61574

5.1.15 SQL 2

```
SELECT
    department,
    AVG(salary) AS dept_avg_sal
FROM employees
WHERE bonus_eligible = TRUE
GROUP BY department
```

department	dept_avg_sal
Sales	58957.4565
Warehouse	48910.5
Accounting	61051.1282
Corporate	251250.0
Human Resources	61463.5540
Management	85333.3333

5.1.16 Big To Small

- While full, large-scale pipelines tend to use SQL and perhaps a bit of Python to shepherd data from location to location, that doesn't have to be the only example of a data pipeline
- We can also shrink things down to more local pipelines where we want the same sorts of effects:
  - A clear source of truth
  - Systematic and consistent cleaning and formatting conventions

  - A clear data lineage that showcases exactly where each piece of information comes from

- Going forward, we'll play around with some tools that you already have at your disposal that may assist with this smaller, local pipelines

6 4. Building Mini-pipelines

6.1 Learning Objectives

6.1.1 What You'll Leave With

By the end of this segment, you will be able to:

1. **Map** Excel's Power Query's three panes onto the ETL vocabulary.
2. **Build** a small repeatable pipeline that loads, cleans, and combines two messy CSV exports.
3. **Distinguish** a Merge from an Append, and say which one you need out loud.
4. **Read** the Applied Steps pane as data lineage, and refresh the whole chain with one click.
5. **Decide** when a mini-pipeline should stop being yours and become IT's.

6.1.2 Get the Data

Same Dunder Mifflin data you downloaded yesterday. We'll be in the `session2` folder. If you don't have it, you can grab it from Canvas → Modules.

. . .

File	Rows	What it is
`crm_customer_export.csv`	1,923	Customer list out of the CRM. Messy.
`erp_orders_export.csv`	4,665	Orders out of the ERP. Messy.
`regional_manager_tracker.csv`	155	Michael's hand-kept sheet. Very messy.
`customers.csv`	2,172	The clean customer list. Join target.
`sales_order_lines.csv`	15,337	The system of record for revenue.

. . .

All Scranton, 2022. Yesterday you saw how these tables connect. Today you make the connection trustworthy.

6.2 Part 1: Doing ETL with Excel

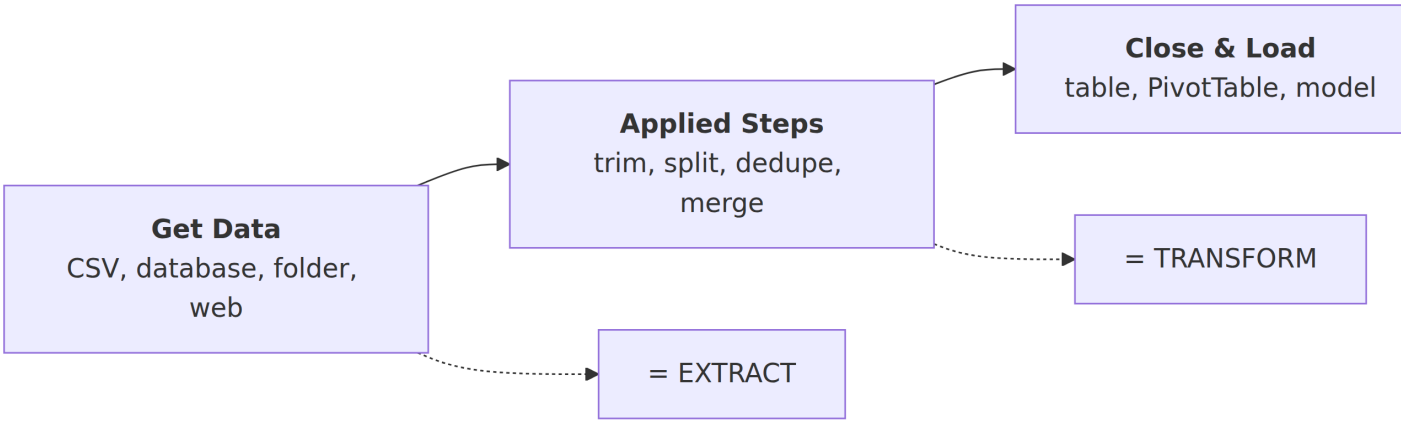
. . .

Turns out you already have an ETL tool. It has been sitting in the Data ribbon this whole time.

6.2.1 The Vocabulary You Just Learned, In A Tool You Already Have

Yesterday you heard **ETL**: Extract, Transform, Load. It probably sounded like something IT does in a cold server room.

. . .



. . .

Excel's Power Query is those three letters with a mouse attached. **Same concepts, same vocabulary, no cold server room.**

6.2.2 Why Not Just Do It By Hand?

You already clean data. You do it with copy, paste, find-and-replace, and a column of `=TRIM()`.

. . .

Yes, that works.

. . .

But then, next month's file arrives.

. . .

6.2.3 Two Ways to Clean Data

6.2.3.1 By hand

You do all of it again. From memory. Slightly differently.

No record of what you did.

6.2.3.2 Build a Mini-pipeline

Next month's file arrives. You click **Refresh**.

Every step is written down, in order.

. . .

6.2.4 Where It Lives

Excel: `Data` ribbon → `Get Data` → `From File` → `From Text/CSV`.

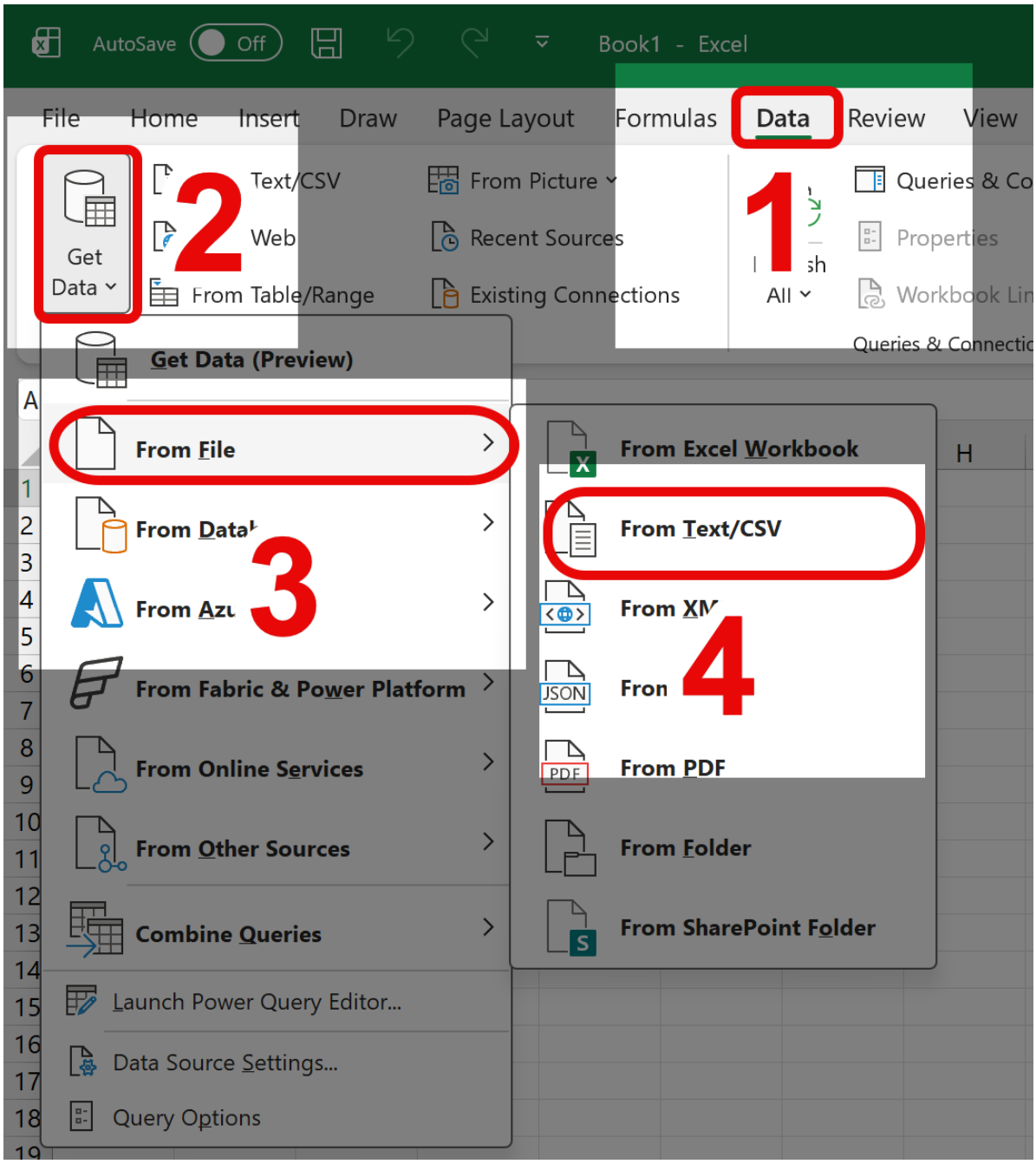


Figure 1: Excel's Power Query

6.2.5 Quick Quiz: ETL

In data engineering, what do the letters E, T, and L stand for?

```
A. Execute, Transfer, and Log
B. Extract, Transform, and Load
C. Export, Transition, and Link
D. Encryption, Troubleshooting, and Licensing
```

. . .

B. Extract, Transform, Load. When IT says "the ETL job failed," they mean one of those three steps broke, and you now have a vocabulary for asking which.

6.3 Part 2: Three Files, Three Different Cleans

Each mess is a different lesson. That is not an accident.

6.3.1 File 1: The CRM Export

`crm_customer_export.csv`, 1,923 rows. Here is what is wrong with it.

Problem	Evidence	The fix
Inconsistent case	`riverside manufacturing llc`	`Transform` → `Format` → `Capitalize Each Word`
Stray whitespace	`Keystone Retail Associates`	`Transform` → `Format` → `Trim`
Duplicate rows	**61 exact duplicates**	`Home` → `Remove Rows` → `Remove Duplicates`
Missing key	**97 rows (5.0%) have no `Acct ID`**	Filter them out, or flag them. Decide on purpose.
Phone chaos	`(682) 486-2143`, `2953356785`, `209.577.2260`	Split, or strip non-digits
Missing fields	200 blank `Segment`, 109 blank `City`	Replace with `Unknown`, do not leave null

6.3.2 The 97 Rows Are The Lesson

The duplicates and the casing are annoying. **The 97 missing keys are dangerous.**

. . .

Join CRM to `customers.csv` on `Acct ID` and you get a clean result. Excel reports no error. Your PivotTable renders beautifully.

. . .

And 97 customers are **not in it**. Your revenue is quietly 5% light and nothing anywhere says so.

. . .

This is why "did it error?" is the wrong question. The right question is "how many rows went in, and how many came out?"

6.3.3 File 2: The ERP Export

`erp_orders_export.csv`, 4,665 rows. Different mess, different fixes.

Problem	Evidence	The fix
Money as text	`$141.58`	`Replace Values` to strip `$` and `,`, then set type `Decimal`
US date strings	`10/13/2022`	Set type `Date` **using Locale** → English (United States)
Shouting status	`FULFILLED`, `RETURNED`, `CANCELLED`	`Format` → `Capitalize Each Word`
Blank ship dates	**175 blanks**	These are the cancelled ones. Keep the blank, it means something.
Duplicate rows	**93 exact duplicates**	`Remove Duplicates`

. . .

Sum that `AMT` column as-is and you get zero. It is text. Excel is not being difficult, it is being correct.

6.3.4 Using Locale Is The Whole Ballgame

`10/13/2022` is unambiguous. `01/05/22` is not.

. . .

Is that **January 5th** or **May 1st**? The file will not tell you. Power Query will guess based on your machine's regional settings, and it will guess **silently**, and it may guess differently on your colleague's laptop in a different region.

. . .

`Transform` → `Data Type` → `Using Locale...` → pick the locale **on purpose**.

6.3.5 File 3: The Tracker

`regional_manager_tracker.csv`, 155 rows. This one does not have a clean fix, and that is the point.

Problem	Evidence
Free text in a number column	`~11`, `a dozen`. **23 of 155 rows, 14.8%**
Typo'd names	`Redwood Financial Goup`, `Conerstone Retail`, `Summi Hospitality`
Two-digit ambiguous dates	`01/05/22`
No order number at all	There is no key. None.
Totals that are wrong	**About 2 in 5** disagree with the system of record

. . .

You cannot join this to anything by key, because **there is no key**. Welcome to every hand-kept spreadsheet in your company.

6.3.6 So What Do You Do With It?

Power Query has an answer, and it is a little bit magic: **fuzzy matching**.

`Home` → `Merge Queries` → check **Use fuzzy matching to perform the merge** → set a similarity threshold.

. . .

`Summi Hospitality Co` will match `Summit Hospitality Co` at about 0.9 similarity.

. . .

> ! Fuzzy matching is a triage tool, not a fix
>
> It will also confidently match the wrong thing. Use it to **find** the rows a human needs to look at, never to silently repair data you then report on. If you fuzzy match your way to a revenue number, you have invented a revenue number.

6.3.7 Quick Quiz: Which Clean?

Your `AMT` column contains `$1,234.56` and every attempt to sum it returns `0`. What happened?

```
A. The column has duplicate rows that cancel each other out.
B. The values are text, not numbers, because of the dollar sign and comma.
C. Excel's calculation mode is set to manual.
D. The column contains blank rows that break the SUM range.
```

. . .

B. A leading `$` makes the whole value text. `SUM` politely ignores text and returns zero. Strip the `$` and `,` with `Replace Values`, then set the type to `Decimal Number`. This is the single most common "the spreadsheet is broken" support ticket in existence.

6.4 Part 3: Combine

Two ways to put data together. Managers who can name the difference get better help from IT.

6.4.1 The Only Two Options

6.4.1.1 Merge = wider

Match rows **across** tables on a shared key. Adds **columns**.

"Attach each order to its customer's segment."

SQL calls this a `JOIN`.

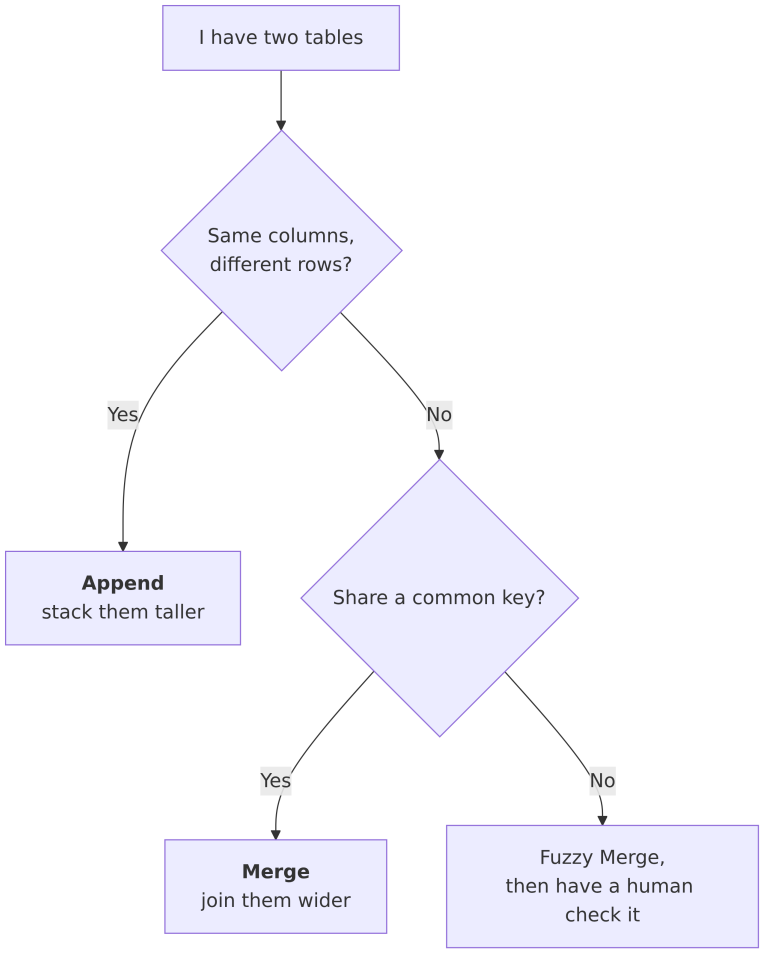
6.4.1.2 Append = taller

Stack rows **from** tables with the same shape. Adds **rows**.

"Combine Q1, Q2, Q3, and Q4 into one year."

SQL calls this a `UNION`.

6.4.2 How do you Decide?



6.4.3 Merge Has A Trap

When you Merge, Power Query asks for a **Join Kind**. The default is **Left Outer**.

. . .

That default is usually right, and it is also how you lose those 97 keyless rows without noticing.

. . .

After every Merge, do exactly one thing: **look at the row count in the status bar.** Did it change? Did it change the way you expected?

. . .

If your 1,923 rows became 1,826, you just silently dropped 97 customers. If they became 2,400, you just duplicated something.

6.4.4 Quick Quiz: Merge or Append

You have twelve monthly sales exports, all with identical columns, and you need one table for the year. Which operation?

```
A. Merge, joining each month to the next on the date column.
B. Append, stacking all twelve into one table.
C. Merge with fuzzy matching, since the file names differ.
D. Neither. You need IT to build a database first.
```

. . .

B. Same shape, different rows, so you stack them. Append. Better still, point Power Query at the **folder** with `Get Data` → `From Folder` and it appends every file in there automatically, including next month's when you drop it in.

6.5 Part 4: Hands-On

Twenty minutes. Build the thing.

6.5.1 Try It: Load and Clean the CRM

Get `crm_customer_export.csv` into Excel and make it trustworthy.

Target: a clean customer table with no duplicates, consistent names, and a decision made about the missing keys.

6.5.2 Steps

1. `Data` → `Get Data` → `From File` → `From Text/CSV` → pick the file → **`Transform Data`** (not `Load`).
2. Select `Account Name` → `Transform` → `Format` → `Trim`, then `Format` → `Capitalize Each Word`.
3. `Home` → `Remove Rows` → `Remove Duplicates`. **Watch the row count: 1,923 drops to 1,862.**
4. Set `Acct ID` type to `Whole Number`. The blanks become `null`.
5. Right-click `Acct ID` → `Remove Empty` **or** add a filter. Either way, **you decided**. Note how many you dropped.
6. `Home` → `Close & Load To...` → `Only Create Connection`.

. . .

You just built an ETL pipeline. That is Extract, Transform, and Load, in six clicks.

6.5.3 Try It: Merge to the Customer List

Attach each CRM row to its canonical record in `customers.csv`, and find out what falls out.

Hint: you need both tables loaded as queries before you can merge them.

6.5.4 Steps

1. Load `customers.csv` the same way. `Close & Load To...` → `Only Create Connection`.
2. `Data` → `Get Data` → `Combine Queries` → `Merge`.
3. Top table: your CRM query. Bottom: `customers`. Click `Acct ID` and `customer_id` to pair them.
4. Join Kind: **Left Outer**. `OK`.
5. Expand the new column (the icon) and pick `segment` and `region`.
6. Filter the expanded `segment` column for `null`.

. . .

Those nulls are your keyless rows. They are in your report, contributing nothing, and if you had summed revenue by segment you would never have seen them.

6.5.5 Try It: Reconcile the Tracker

The hard one. Michael's sheet says Scranton did **$43,020.24**. Does the system of record agree?

Hint: there is no key. You will need fuzzy matching, and you will not fully succeed. That is the finding.

6.5.6 Steps

1. Load `regional_manager_tracker.csv`. Look at `Qty`. **Set it to `Whole Number` and watch 23 rows turn into errors.**
2. Those errors are `~11` and `a dozen`. `Keep Errors` to see them. This is your data quality report.
3. Merge the tracker to `customers` on `Client  customer_name`, with **fuzzy matching** on.
4. Drop the threshold to 0.8. Watch `Summi Hospitality Co` find `Summit Hospitality Co`.
5. Now try to check the totals against `sales_order_lines`. **You cannot, cleanly.** There is no order number to join on.

. . .

That is the answer. The sheet cannot be reconciled, because it was never built to be. About 2 in 5 of its totals are wrong and there is no way to prove which ones from the file alone.

6.5.7 Debrief

Hands up. Who got a different row count than the person next to them?

. . .

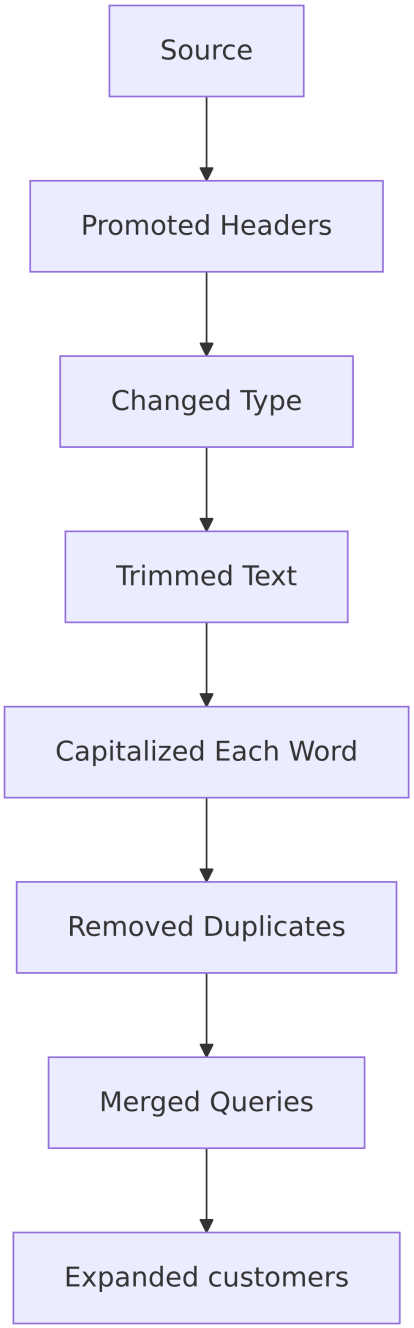
That is the entire reason Applied Steps exists.

6.6 Part 5: Refresh and Lineage

The part where it stops being a chore and starts being infrastructure.

6.6.1 Your Steps Were Being Written Down

Look at the right-hand pane. **Applied Steps.** Every click you made is there, in order, named.



. . .

Click any step. The preview jumps back to how the data looked **at that moment**. You can insert a step in the middle. You can delete one. You can rename them so a human can read them.

6.6.2 This Is Lineage

In Potential Data Pitfalls we said lineage is the map of a number's journey from origin to report, and that if nobody can trace it, you have folklore.

. . .

Applied Steps is that map. For this one pipeline, you can answer all three questions:

1. *Where did it come from?* The `Source` step names the file.
2. *What was done to it?* Six steps, in order, each named.
3. *Who changed it and when?* The workbook, and its version history.

. . .

You did not build a cleaned spreadsheet. You built a **documented, re-runnable, auditable** cleaned spreadsheet. Those are different objects.

6.6.3 The Refresh

Next month, Scranton sends a new export. Same columns, new rows.

. . .

You drop it in the same path and click **Refresh**.

. . .

Trim, capitalize, dedupe, merge, expand. All of it. In about a second. **Exactly the way you did it last month**, because it is the same steps, not your memory of them.

. . .

That is the "automate" half of the objective. And it is why your February number is comparable to your January number, which is a thing your current process almost certainly cannot promise.

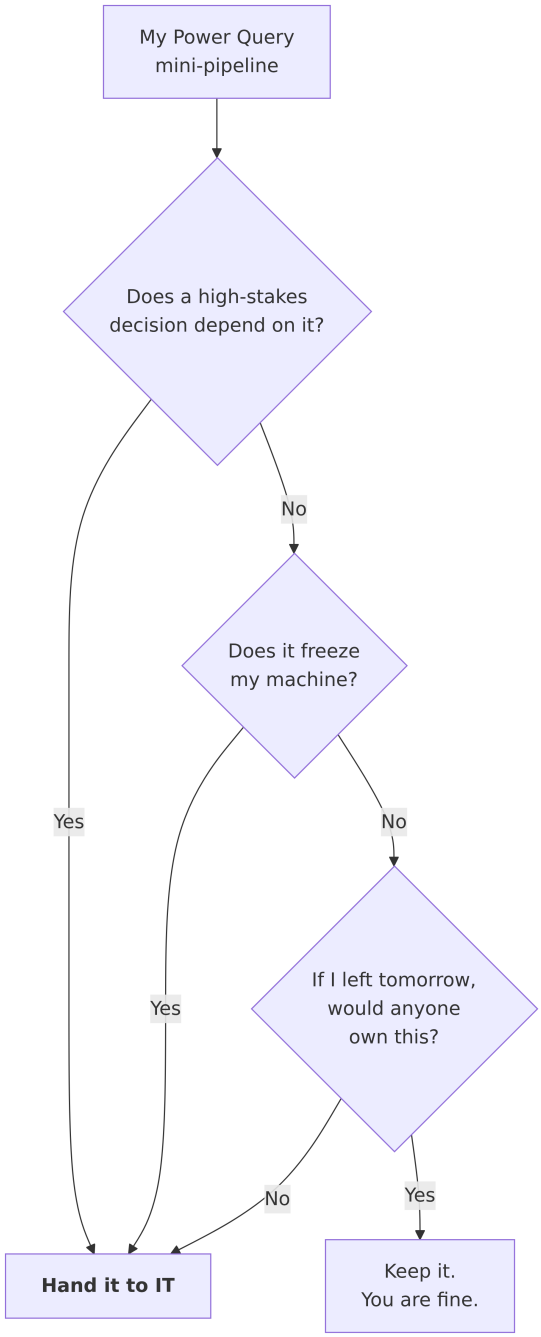
6.7 Part 6: When Do You Hand This to IT?

Know when you have outgrown your own pipeline.

6.7.1 You Built a Pipeline. Now What?

Congratulations. Now, when should it stop being yours?

6.7.2 Three Signals



. . .

Stakes. Volume. Ownership. If any one of those trips, it is time.

6.7.3 Not The Signals

Things that do **not** mean you should hand it off:

- "The file got past 50,000 rows." Row counts are not a governance policy.
- "IT told us to stop using Excel." That is a turf memo, not an engineering decision.
- "I want it fully automated so nobody ever checks the data again." **Nobody should ever want this.**

. . .

The real trigger is always about **consequence**, not size.

6.7.4 Volume: Let's Be Honest

The full Dunder Mifflin `sales_order_lines` is **215,485 rows**. Excel's limit is **1,048,576**.

. . .

So it fits. It fits fine.

. . .

Now put a `VLOOKUP` in a column next to it and try to scroll. Now refresh it. Now email it to somebody.

. . .

"It fits" and "it works" are different claims. The limit is not the row count, it is the moment your laptop fan becomes part of the meeting.

6.7.5 Quick Quiz: The Handoff

At what point does a mini-pipeline in Power Query become something you hand to the data or IT team?

```
A. As soon as the file exceeds 50,000 rows or needs more than 10 columns altered.
B. Only when IT explicitly mandates that local Excel usage stop.
C. When the report drives high-stakes decisions, handles volumes that freeze local machines,
D. When you want it fully automated so no human ever has to check data quality again.
```

. . .

C. Stakes, volume, ownership. A is arbitrary. B is politics. D is the fantasy that gets companies into trouble, because someone always has to check the data. The honest test is: *if this breaks while I am on vacation, what happens?*

6.7.6 What To Actually Say To IT

You are not asking them to "fix your spreadsheet." You are handing over a specification.

. . .

You can now say:

> "I have a mini-pipeline that **extracts** the CRM and ERP exports, **transforms** them by trimming names, deduping 61 rows, and coercing the `AMT` text into decimals, then **merges** them to the customer dimension on `Acct ID` with a left outer join. About 5% of rows have no key and currently drop out. It drives the regional revenue number the VP sees monthly, and I am the only person who knows how to run it."

. . .

That is a **requirements document**. You just wrote it by clicking around in Excel for twenty minutes.

6.7.7 Technique Summary

Technique	What It Does
`Get Data` → `From Text/CSV`	Extract. Start of every pipeline.
`Transform Data`	Opens the editor. Not `Load`.
`Trim` / `Capitalize Each Word`	Kills whitespace and casing drift
`Remove Duplicates`	Kills double-counting
`Replace Values` + `Decimal Number`	Turns `$141.58` into money
`Data Type` → `Using Locale`	Makes ambiguous dates unambiguous, on purpose
`Merge Queries`	Join on a key. Wider.
`Append Queries`	Stack same-shaped rows. Taller.
Fuzzy matching	Triage for typo'd keys. Never for silent repair.
`Get Data` → `From Folder`	Appends every file in a folder, forever
Applied Steps	Your lineage, written down
`Refresh`	Runs the whole chain again, identically

6.7.8 The Big Ideas

1. **You already own an ETL tool.** Extract, Transform, Load has been in the Data ribbon the entire time, and now you can name what it is doing.
2. **Check the row count after every join.** A broken join does not error. It just returns less, and your report renders anyway.

3. **Merge is wider, Append is taller.** Say the right word to IT and get help in an hour instead of a fortnight.
4. **Applied Steps is lineage.** The pipeline documents itself, which is the only kind of documentation that ever survives contact with a deadline.
5. **A sheet with no key cannot be reconciled.** Not by you, not by Power Query, not by IT. That is a process fix, upstream, not a data fix.
6. **Hand it off for stakes, volume, or ownership.** Never for row count, never because of a memo.

6.7.9 Share Out

Find another pair. In two minutes each:

- Which of the three files fought you hardest, and where did you get stuck?
- What did your row count do after the merge? Did anyone lose the 97?
- Name one report back at work you would rebuild as a refreshable query on Monday.

6.7.10 Where This Started

This morning opened with two numbers that disagreed and no way to tell which was right.

. . .

You can now clean the exports that feed them, merge them on a key, watch the row count, and read the Applied Steps back like a receipt. The disagreement is still possible. It is no longer a mystery.

7 5. References and Resources

7.0.1 Resources

1. Dunder Mifflin Paper Company dataset. https://dataengineering.education/files/workshop_data.zip
2. Wickham, H. (2014). "Tidy Data." *Journal of Statistical Software* 59(10). The tidy rules from Session 1, and what messy data looks like.
3. Kimball, R. and Ross, M. *The Data Warehouse Toolkit*, 3rd ed. Wiley, 2013. Chapter 1, on why a single source of truth is a design decision rather than an accident.
4. Microsoft. "Data lineage in Microsoft Purview." https://learn.microsoft.com/en-us/purview/concept-data-lineage
5. Microsoft. "Power Query documentation." https://learn.microsoft.com/en-us/power-query/

6. Microsoft. “Merge queries overview.” https://learn.microsoft.com/en-us/power-query/merge-queries-overview
7. Microsoft. “Fuzzy merge.” https://learn.microsoft.com/en-us/power-query/fuzzy-matching
8. Microsoft. “Import data from a folder with multiple files.” https://learn.microsoft.com/en-us/power-query/connectors/folder