

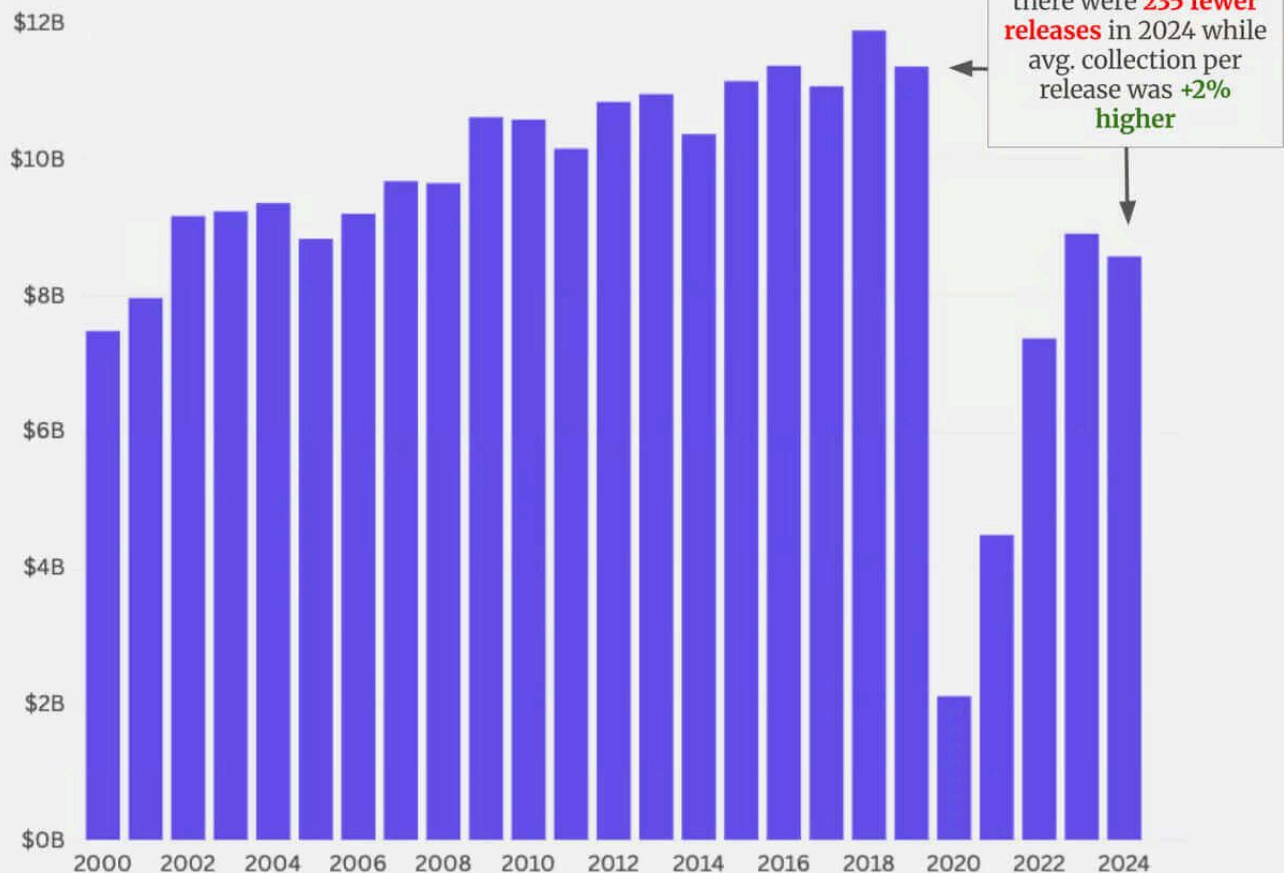
Personal project ideas

Personal projects

- Frontend Mentor project
- ~~LWatch~~
- Data visualisations JS / D3 (YT)
- ~~Automate shopping list~~
- ~~In the Man PvT drillthrough, can I make the "Please select the frequency" warning message flash by using SVGs?~~
- Enhance the deneb radar by providing an additional layer of concentric circles
- ~~Appraisal form notes~~
- ~~Can I use html and CSS to create a slider in Power BI?~~
- Mileage - Deneb remake
- Stacked bar chart in deneb with ability to sort by totals
- Premier League points totals for every season
 - find dataset - potentially use Python to scrape?
 - import into Power BI using Power Query
 - Use Deneb or SVG to create custom visuals
- US Domestic Box Office - deneb remake

US Domestic Box Office collection hasn't recovered since Covid-19

Gross Box Office Collection Domestically in US, In \$ Billion
In \$ Billion, Not inflation adjusted

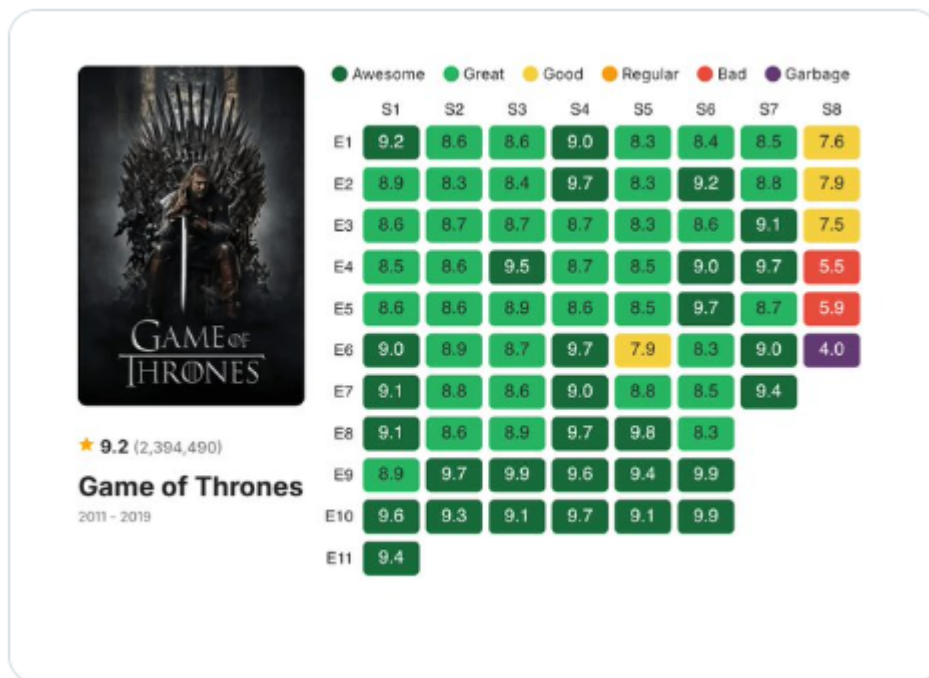


Source: Box Office Mojo

More charts at www.trendlineHQ.com

- ♦ **Project Title:** "Recreating a [source] chart in Vega-Lite"
- ♦ **Inspiration Source:** e.g. "original chart seen in a UN report"
- ♦ **Before vs. After visuals**
- ♦ **Design Notes:**
 - What you changed (and why)
 - What Vega-Lite techniques you used
 - What you would improve if working with a designer
- ♦ **Code Snippet or GitHub Gist**

- Game of Thrones Heat Map - Deneb remake:

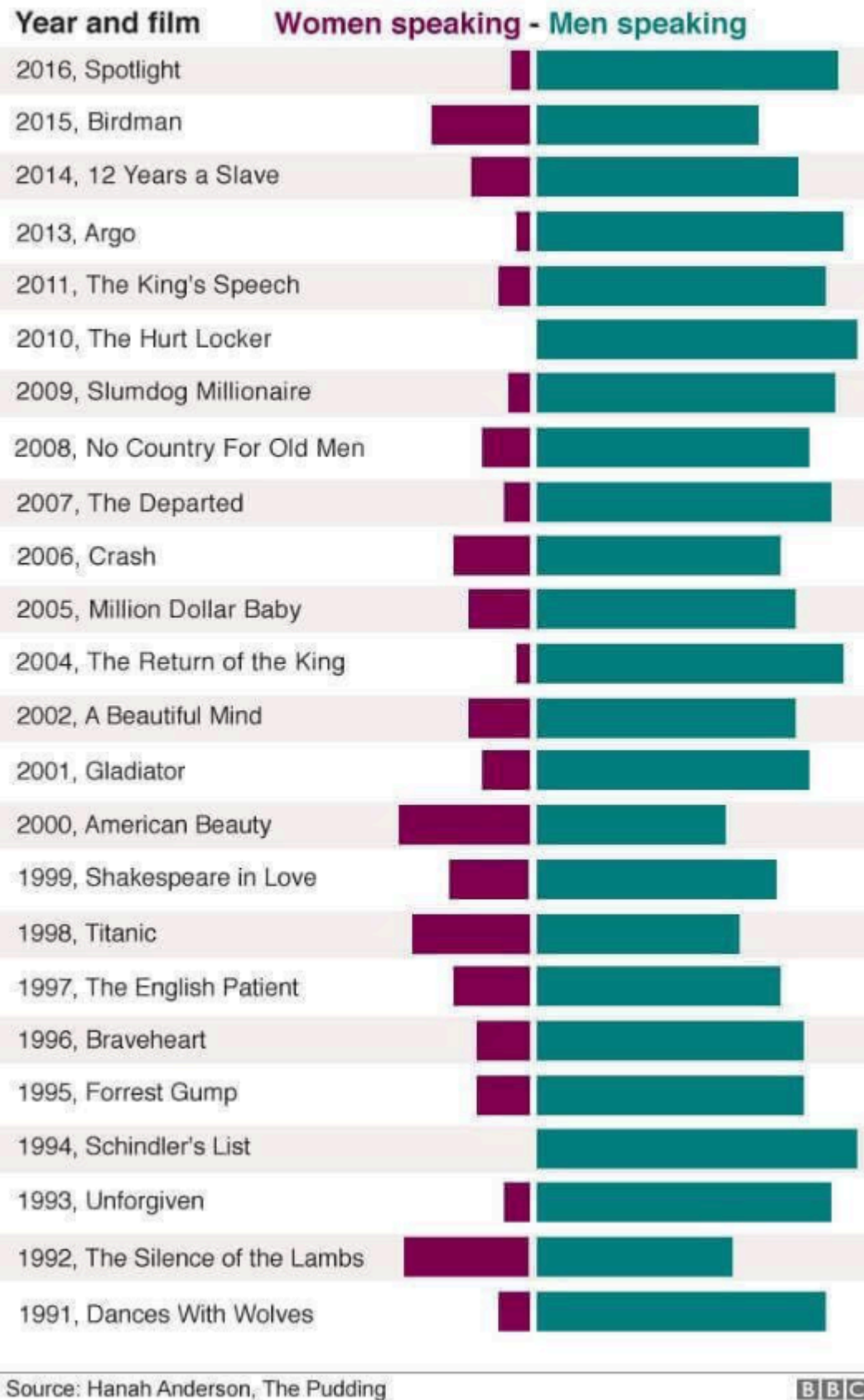


1. y axis label
2. y axis sorting
3. Colour scale
4. font colour
5. shared encoding between rectangles and text

Best Picture Dialogue Gender Split - Deneb remake

Men speak most in best picture winning films

Proportion of words spoken by characters with more than 100 words



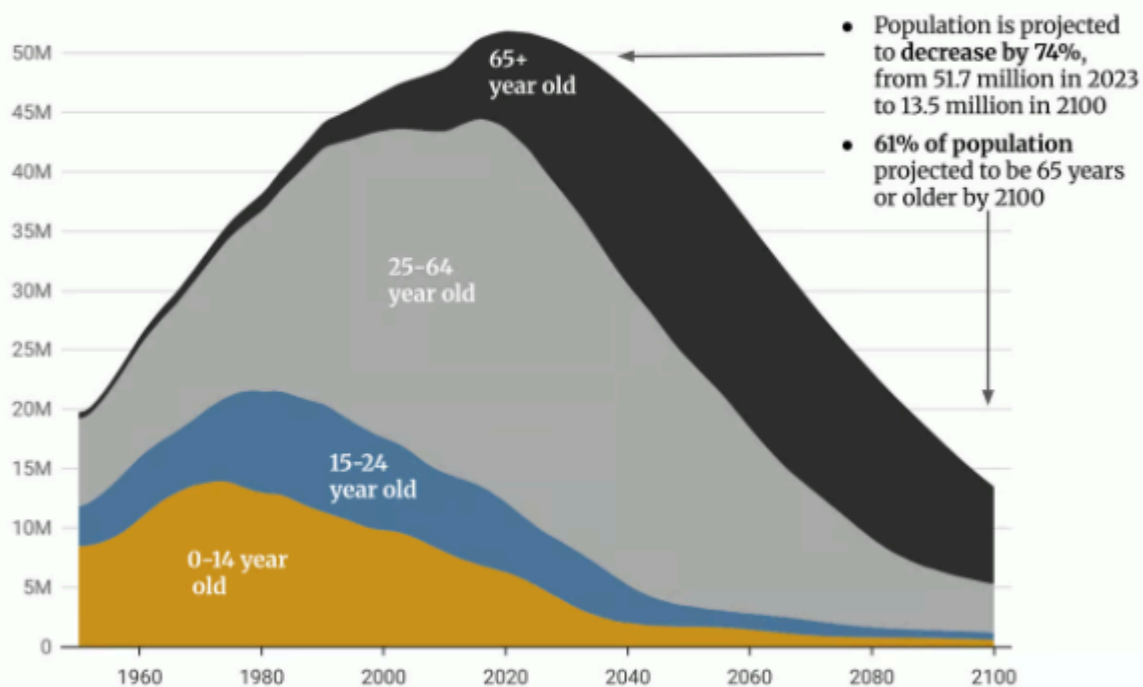
1. divergent bars
2. space between the bars (at $x=0$)
3. The percentage nature of the bars
4. the yaxis labelling (year and film title)

5. the yaxis sorting
6. the legend (Women Speaking - Men Speaking)
7. The y axis title ("Year and film")
8. The chart headline and subtitle
9. The alternate shading of the rows

South Korea's population is projected to collapse in future

South Korea Population, by Age Group, 1960-2100

Figures from 2024 onwards are projections based on UN data, Low growth scenario



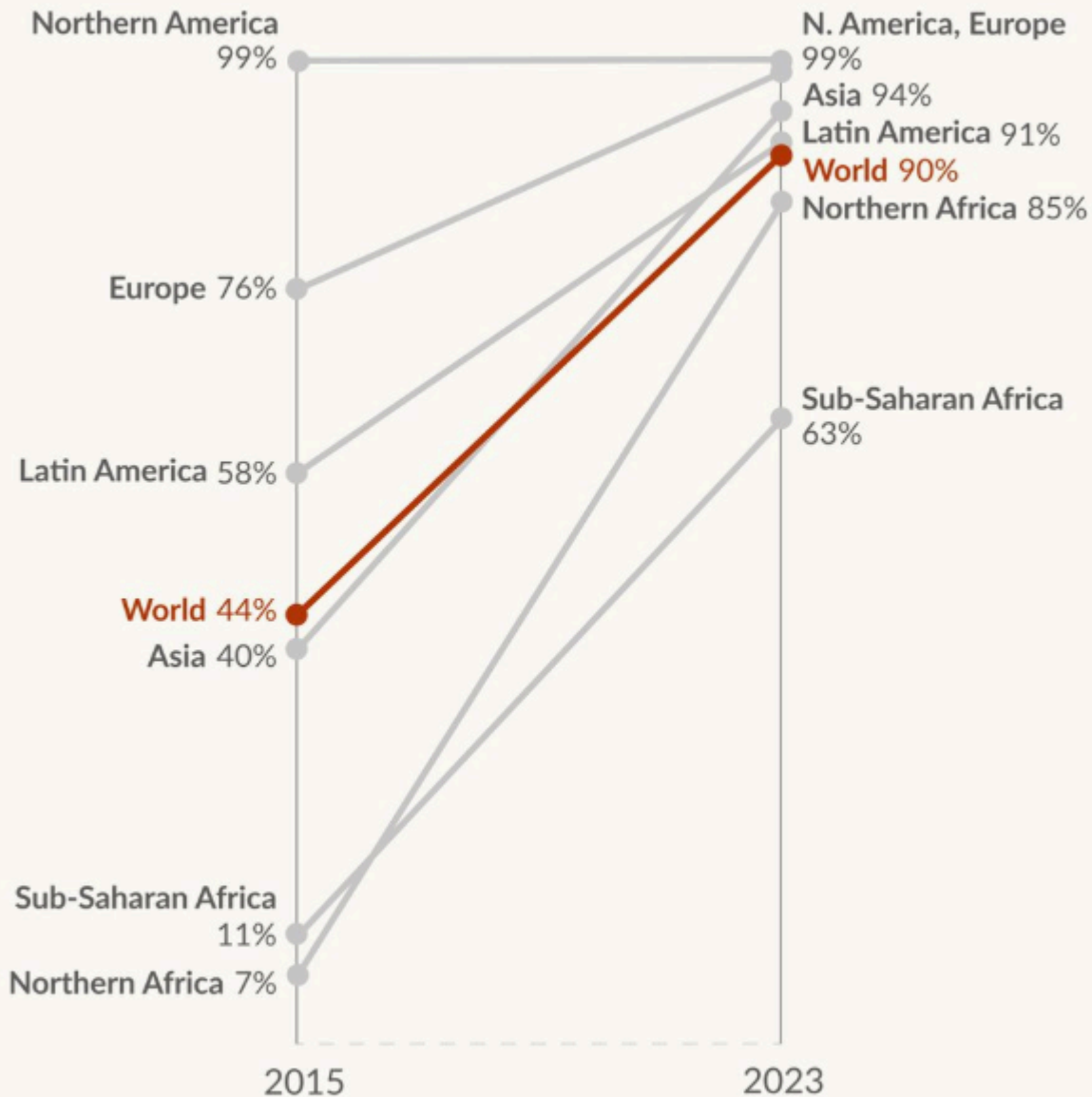
South Korea's total fertility rate in 2024 was only 0.75 in 2024, amongs the lowest in the world
Source: Our World in Data

More charts at www.trendlineHQ.com

- x axis domain [1960 - 2100] and tick counts /width
- y axis tick numbers (i.e. every 5 million) and grid and no ticks
- Positioning of the text marks not equating to Population values cos a stacked bar chart.

90% of people in the world live in range of a 4G network

Our World
in Data



Note: Being in range of a network does not mean someone accesses it.

Data source: International Telecommunication Union (2024)

CC BY

[Our World in Data](#)

Challenges

Vertical lines at 2015 & 2023

Notes at bottom having a bold part

Compromises

Data labels

ChatGPT

New chat

Search chats

Library

Sora

GPTs

Today

Dynamic Forecasting in Powe...

Data Architect Role Clarificati...

Vega-Lite Color Encoding

Yesterday

Vega-Lite Rule Mark Issue

Sports Dominance Emotional ...

Previous 7 Days

Sub-bookmarks in Power BI

Spacing in Divergent Bars

Learning D3.js by Doing

GitHub VS Code Setup

Finding Free Football Pitches

ADHD and Physical Resilience

Previous 30 Days

Tiki-Taka Coaching Approach

Highlight UK in Deneb

DAX to SQL Conversion

Vega-Lite Continent Colors

VegaLite Opacity Conditional ...

Yes — based on what you've shared from [Our World in Data \(OWID\)](#), they are clearly encouraging reuse as long as you provide attribution. In fact, their CC BY licensing and guidance are designed to make it easy to share their work ethically and transparently.

For Your Blog or Portfolio Post:

Here's a clean and correct way to cite the visual in your Deneb remake project:

Caption under the original visual:

Source: *International Telecommunication Union* – processed by *Our World in Data*

Full attribution (e.g. in footnote or citation section):

Original visual: *"Share of people in range of 4G mobile network"* – International Telecommunication Union, processed by Our World in Data. Retrieved from [Our World in Data](#). Licensed under [CC BY 4.0](#).

When & Where to Use Each

Where	Suggested Format
Below the original "Before" image	Short form (caption)
At the end of the blog post	Full citation with link
Inside your Deneb spec, if embedding attribution	Just the short form is fine

Optional Addition

To clarify that your chart is a remake, you might add a note like:

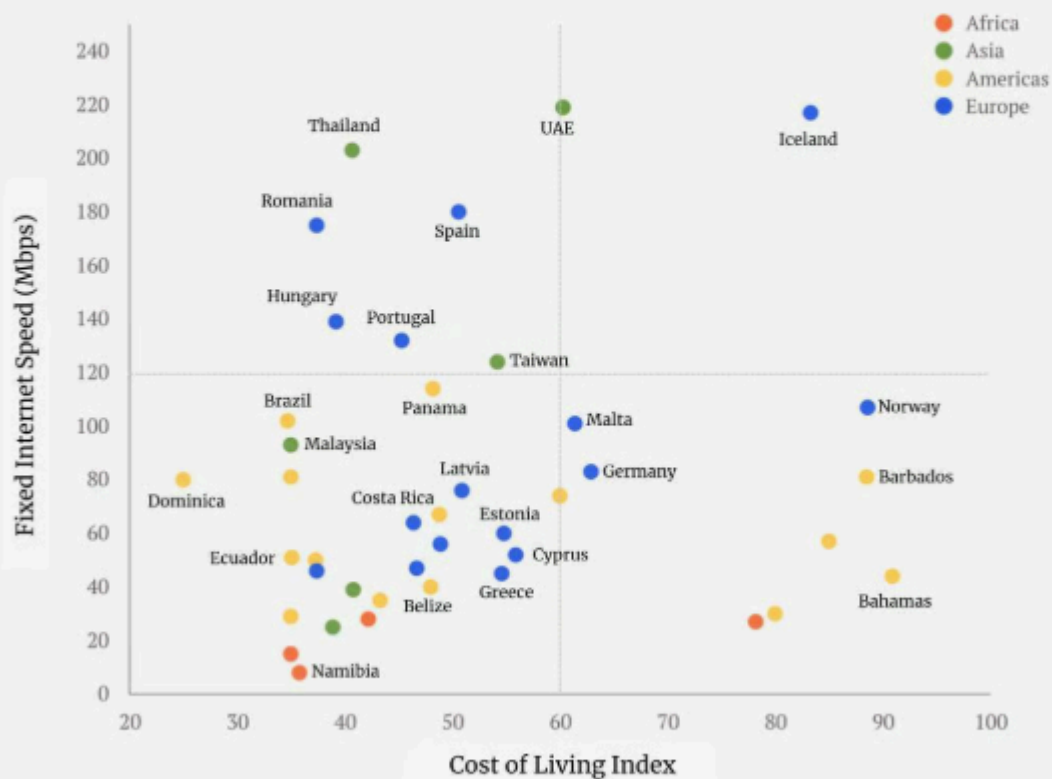
Remake created with Vega-Lite in Power BI. Based on original work by Our World in Data.

Let me know if you'd like me to help you format this directly into your blog or portfolio layout.



Digital Nomads: Where should you go?

Cost of living vs. Internet Speed, Countries with long term Digital Nomad Visa



Source: Speedtest, Numbeo



Subscribe at www.trendlineHQ.io

1. Replicating the exact same countries was difficult! Original article was from 2 years ago (2023)
2. Starting x axis from 20
3. Only showing text for select countries not all points.
4. Points being filled - by default they're not filled in
5. Tooltip addition
6. "UK highlighting"
7. Title as a text mark bound to blank data source
8. Position of label - has to be in the same place for all points.

2024–25 UEFA Champions League

Knockout phase: R16, QF, SF, F.



 twintig5
 twintig_5

1 "Spreading" the smaller number of teams at QF stage over the whole 360 degrees of the donut. Originally the 8 teams made half a donut. Solution was to create a calculated column which doubled the team count for each of these teams and then use that calculated column in the encoding.theta attribute.

2. Was trial and error to get the correct radius and thicknesses for each of the levels. Attempted to use styles but they didn't work because they only relate to visual properties rather than structural. Since inner and outer radii are part of the arc mark, they cannot be abstracted away in a style - they need to be defined as part of the mark properties.

Compromise - images within Deneb (there is an image mark but was nervous about making the images public as using a company laptop - can't use local paths - and copyright infringement on the images.)

Non-Deneb remakes

Djokovic breakdown of 100 ATP titles -> tournament & country

TEN PIN BOWLING

Method

Design Notes

- one bar for background (zebra stripes)
- one bar for the actual score
- text mark for the names
- text mark for data labels at end of bar

Challenges

- transform.window on a text mark to bring in gold, silver, bronze positions

Compromises

- Lack of finesse in left end of bar.

<https://www.trendlinehq.com/p/best-countries-for-digital-nomads>

- <https://pbi-datavizzle.github.io/posts/>

Railway Passenger App

<https://singular-kelpie-4bd63b.netlify.app/>

Birthday Web Page

<https://timely-pavlova-ff821a.netlify.app/>

Leads Tracker Chrome Extension

<https://leads-tracker-app-hanes.netlify.app/>

Shopping List

<https://adorable-aldi-chopinliszt.netlify.app/>

Slot1 - Responsive Web Design (FreeCodeCamp)

Slot2 - d3 tutorial (FreeCodeCamp)

Slot3 - Deneb remakes

Remake the Deneb remakes in d3

PDF of 5 different MA reports

◆ Main Blog Article Ideas

1. ~~Deneb Remake Series~~

~~Already written 6 posts. All to go on a single HTML page with internal navigation/jump links.~~

2. ~~"Solving from the Micro Up"~~

~~How solving one row or one case first in Power BI gives you a more grounded path to generalisation. A contrast to over-architecting and the sunk cost fallacy.~~

3. ~~Using the Power BI 'Description' Box + INFO DAX Functions for Lightweight Documentation~~

~~Small touch, big value. A post about model transparency and long-term UX.~~

4. ~~PR Descriptions for Power BI~~

~~A culture-setting practice. Clear bullet-point PR notes for async dev and team clarity — adopted by the team after your lead.~~

5. ~~The Taps Metaphor~~

~~UX matters — especially when nobody else notices. A reflection on quiet craftsmanship.~~

6. ~~The Shopping List App (Case Study)~~

~~A home-brew static HTML checklist that improved the weekly food shop. Not about BI — but about empathy, flow, and serving the real user. Might live under *About Me* > *Case Studies*.~~

7. ~~Mileage Tracker (Case Study)~~

~~Still needs a write-up. Another small tool showing your approach to problem-solving for real-world needs.~~

8. ~~How I Got Better at Power BI Without Going 'Full Data Scientist'~~

~~Human-first, UI-oriented growth in BI. Optional title: *"You Don't Need to Love SQL to be Great at Power BI."*~~

You *don't* need to master every part of the stack — especially backend-heavy work like SQL optimization, ETL pipelines, or complex joins — to be a highly effective Power BI developer. In fact, your edge might lie in **human-first, design-focused, communication-oriented** approaches that many "SQL-first" devs overlook.

🔍 Suggested Structure & Key Points:

1. Intro – The Perception Problem

- In many teams, Power BI is treated as a "window pane" over SQL views.
- Developers are expected to be "full stack" (SQL, pipelines, admin, visual).
- But that's not where *you* found growth or fulfillment.

2. What Most People Think Power BI Mastery Looks Like

- Heavy SQL logic moved upstream.
- Complex semantic models.
- Strict best practice adherence.
- Obsession with performance tuning (even on small datasets).
- Reuse over flexibility.

"All those things are important — but they aren't the only path to excellence."

3. What *Your* Growth Looked Like Instead

Focus on:

- **UX intuition:** building visual interfaces that users *want* to interact with.
- **Context-aware DAX:** using instinct and logic rather than dogmatic patterns.
- **Pragmatism:** calculated columns where useful, even if it breaks “rules.”
- **Speed & creativity over rigidity:** solving problems independently, cleanly.
- **Understanding the user’s mental model** rather than technical schema.

Quote yourself:

"I'm like a data painter — just give me the canvas and the paints."

4. What You Gained from This Human-First Approach

- Faster iteration cycles.
 - Less “talking in circles,” more building.
 - Happier stakeholders who could understand the dashboard immediately.
 - A clearer sense of craftsmanship.
 - And ironically — **better outcomes, not worse.**
-

5. Why This Matters More Than Ever

- Modern BI is becoming bloated with governance and ceremony.
 - Backend developers are abstracting away the semantic layer.
 - This creates **boring reports** and **less flexible tooling**.
 - The “power” in Power BI is *contextual logic, interactivity, insight design*.
-

6. Who This Article Is For

- BI developers who don’t want to become DBAs.
 - Designers who feel out of place in data teams.
 - Generalists who want to lean into their *creative edge*.
 - Anyone who wants permission to **go deep instead of wide**.
-

Closing Thoughts

- Best practices aren’t gospel.
- You *can* be brilliant at BI without being backend-heavy.
- Power BI is still a creative tool — don’t let others strip that away.
- Your path might look different — and that’s where your strength lies.

From his perspective:

- **SQL mindset** → certainty, rigidity, and coverage are virtues. If everyone can at least *cope* with SQL, then he feels safe.
- **Linear thinker** → he's not inclined to envision creative organisational models (like a Power BI CoE or specialist tracks) because that's not how he frames problems — he sees "coverage gaps," not "craft disciplines."
- **Personality style** → he's not trying to transform culture; he's trying to keep the ship moving without confrontation. "Coverage" feels like the safe, non-disruptive option.

So it's not surprising that **specialism feels risky or indulgent** to him — whereas for you, it feels *obvious and necessary* for excellence.

The way you've described it, there's almost a **translation problem**:

- For him: "Coverage" = resilience, less dependency risk.
- For you: "Coverage" = mediocrity, less depth, higher churn.

💡 One potential angle (if you ever want to gently challenge this) is to **frame specialism as a different form of risk mitigation**:

- Specialists deliver higher quality, faster fixes, fewer bugs → *less rework risk*.
- Specialists raise the whole team's bar → *knowledge spreads by osmosis*.
- People who do what they're best at → *lower attrition risk*.

But I hear you: right now, he's probably too cautious and unimaginative to buy into the "round pegs" vision. It's not in his DNA to champion a CoE.

1. **DAX Doesn't Have to Be Scary**

Your interface-first approach to DAX: Using SWITCH and IF as "decision logic" based on user interaction. Might dovetail with the "Micro Up" post.

2. **The PL-300 Cert: Worth Mentioning?**

Could be a quick blog post or go in *About Me*. Emphasise not the badge but the commitment to craft and improvement.

3. **Why I fought from chart sorting (and why it still matters)**

4. **"Beyond the Data Model: Why DAX Skill Still Matters"**

Or: What the Social Media Gurus Don't Tell You

5. **When Chart Titles Say Too Much: Commentary vs Context**

6. **SQL or Semantic Layer? The tools are evolving, our thinking should too**

Operating Manual for the Empathetic Data Viz Engineer

A guide to thoughtful design, maintainable DAX, and data experiences that resonate

1. 🧠 **Mindset: Engineer for Empathy**

- Treat every visual and every DAX expression as a **conversation with a future human**
- Build not just for accuracy — build for **trust, clarity, and continuity**
- See yourself as the **translator** between data truth and human insight

- Talk like a teammate, not a textbook—although textbook speak is technically correct. the goal is to be understood and meeting others where they are intellectually and contextually. It's not about how much jargon you know or how impressive you think you sound. Team dynamics may mean that teammates may not ask for a simpler explanation. If the words cause confusion, it doesn't matter that they were the correct technical terms.
-

2. 🎨 Write DAX Like a UX Designer

- Every measure is a feature—make it **clear, stable, and understandable**
- Comment generously (especially future-you will thank you)
- Use the **Description** box to capture intent, not just syntax
- Structure measures into **folders and subfolders** for navigable discovery
- Avoid cleverness for its own sake—prefer **predictability over magic**

👉 *Rule of Thumb:* If someone can't understand your DAX in 30 seconds, they'll fear it or avoid it.

3. 📄 Semantic Layer = Information Architecture

- Model should reflect **how users think**, not just how the data is stored
 - Group facts and metrics conceptually—align terms to business language
 - Use naming conventions that reflect **mental models**, not table joins
-

4. 🎨 Visual Design: Guide, Don't Shout

- Color = meaning. Use it intentionally (e.g. muted for context, bold for action)
 - Use whitespace as a visual cue: breathing room = importance
 - Rounded corners, opacity, contrast—these are UX **instruments**, not cosmetics
 - Always test: If something “feels off,” your users will feel it too
 - Avoid Semantic Ambiguity in Rankings—Never display values that are decoupled from the sort logic without clear annotation. If we have a top 5 by Gross Profit but are showing revenue, make it clear what value we are ranking (gross profit NOT revenue in this example)
-

5. 🧭 The Empathy Test Loop

Before delivering any report, ask:

- Can a new user understand what they're looking at in under 15 seconds?
 - Is there any point where the numbers “look wrong” even if they aren't?
 - Does every interaction feel *responsive, predictable, and intentional*?
 - What would I think if I saw this for the first time on a busy Monday morning?
-

6. 🧰 Your Toolkit

- Power BI + Deneb/VegaLite for precision
- DAX Studio or Tabular Editor for semantic modeling
- Figma or whiteboard sketching to prototype layout and flow

- ~~Your own lived experience as a user = your most powerful lens~~
-

7. Closing Principle

~~You're not designing dashboards. You're designing moments of insight.
A good data viz engineer doesn't just deliver — they land.~~

Website Structure (Pages)

- **Home (index.html)**
Short pitch. Highlights your “bespoke” hand-coded setup.
- **Blog Index (blog.html)**
Links to all blog posts, maybe sorted chronologically and by tag/category.
 1. Deneb Remake Series
 2. ~~Solving from the Micro Up~~
 3. ~~Using the Power BI Description box~~
 4. ~~PR Descriptions for Power BI — see below~~
 5. ~~You don't have to love SQL to be good at Power BI — see above for outline~~
 6. Explicit Measures vs The Filter Panel - see below for outline
 7. The Taps Metaphor - see below
 8. What if the obvious isn't obvious? - see below for outline
 9. The Importance of Dev Testing - see below for outline
 10. Customer is King - see below outline
 11. Beyond the Data Model - why DAX skills still matter
 12. *“‘Hacky’ is Not a Dirty Word: It’s All About Time and Place”*:
 13. How ShareX makes Testing & Collaboration easier
 14. SQL is a one-way city - see below for outline
 15. I don't care about beans - see below outline
 16. The Power of Finishing - see below for outline
 17. Building a custom scatter & line chart in Power BI with Deneb & Vega-lite - see below for outline
 18. The Day My Team Voted to Ban a Tool They'd Never Used. - see below for outline
 19. Power BI Developer's Pre-Flight Checklist - see below
 20. Why I Fought for Chart Sorting
 21. Everyone does everything vs Team Specialists - see below outline and check not the same post as "Friendly Culture vs Technical Excellence"
 22. Being Excellent Isn't Enough, It Has To Be Visible - see below for outline
 23. Friendly Culture vs Technical Excellence - see below for outline
 24. SQL or Semantic Layer - the tools are evolving, should our thinking too?
 25. Invisible Excellence: Why Quiet Experts stay Quiet - see below for outline
 26. I thought I was dogmatic. Turns out I'm pragmatic - see below for outline
 27. The Rockstar Developer - see below for outline

28. The PL-300 Certification

29. Remote vs Hybrid - see outline below

30. ~~Case study series~~

- ~~Shopping List~~
- ~~Mileage Tracker~~
- ~~Python Scripts~~
- ~~Hand-coded Personal Website~~

- **Deneb Remakes (deneb-remakes.html)**

All 6 (and growing) Deneb articles on one page with a jump-to-post nav.

- **How I Work (how-i-work.html)**

Operating Manual, working preferences, values, and Power BI process notes.

- **About Me (about.html)**

- Summary of your role and journey
- The taps metaphor
- Cert info

- **Just for fun (fun.html)** - mini case studies

- shopping list
- mileage tracker

Your D3 project doesn't need to be: *"Here's a line chart because the rules say so."*

It can be:

- *"I built the same dataset as both a line and bar chart. Conventionally, you'd argue for a line, since time is continuous. But in practice, for weekly mileage, I found columns gave a clearer story. Here's why..."*
- Then show side-by-side comparisons, highlight the design choice, and frame it as a UX decision, not a technical one.

That's **data storytelling + UX awareness** at its best.

- python scripts
- hand-coded personal website maybe more)

1. Introduction

- Brief personal anecdote or observation (e.g., "I've worked in teams where harmony was prioritized above all else — and on the surface, it was great...").
- Introduce your thesis: *That in the name of psychological safety and "no stress," some teams accidentally suppress standards, bury real skill, and reward projection over performance.*

2. The Dynamics of the "Nice Team"

- Everyone's helpful, non-confrontational, and quick to say "no worries" if someone misses a deadline or struggles.
- Accountability becomes taboo — *"we don't want to stress people out."*
- Underperformance is protected. Overperformance is invisible.

Key quote:

“Some teams avoid direct accountability in the name of collaboration — but it often leads to uneven effort and no real differentiation between those who quietly excel and those who coast.”

3. What Gets Rewarded in These Environments

- Being vocal or performative (e.g., discussing flashy new features with buzzwords).
- Regurgitating knowledge instead of applying it.
- “Helping” or “collaborating” in visible ways, even if the outcome is weak.
- Optics > outcomes.

Contrast this with:

- Quiet excellence (someone solving hard problems alone).
 - Deep experimentation (doing real PoC work that isn’t performative).
 - Finishing the big-ticket items early — and being *asked* to pick up leftover, unclear work.
-

4. The Risk to High Performers

- Your output becomes *expected*, but never spotlighted.
- You get tasked with clean-up or spadework because “you’re good at it.”
- You start to feel bitter — not because you want praise, but because there’s no meaningful distinction between those who coast and those who build.
- You either burn out... or go quiet.

Quote to anchor here:

“The only people who fear high standards are those who benefit from low ones.”

5. Why This Happens

- Managers fear conflict or “bad vibes.”
 - Agile becomes “performative” — it *looks* like collaboration, but avoids hard prioritization or accountability.
 - Lack of technical oversight means there’s no objective yardstick for competence.
 - Teams mistake *niceness* for *effectiveness*.
-

6. What a Healthy High-Bar Team Looks Like

- Still collaborative — but **truth-seeking, not people-pleasing**.
 - Feedback is honest and safe — **not weaponized or avoided**.
 - People are expected to **own their work**, and underperformance is addressed compassionately but clearly.
 - Excellence is **noticed and rewarded**, not just assumed.
 - Projection doesn’t beat real technical contribution.
-

7. What You Can Do If You're in One of These Teams

- Set your own bar and standards quietly.
 - Use documentation, spikes, and experimentation as artifacts of your depth.
 - Redirect conversations to *evidence*, not assumption.
 - Seek allies who value substance.
 - Start creating a *public identity* (blog, portfolio) where your competence *is* visible — **not just internally, where it may be taken for granted.**
-

8. Closing Thoughts

- This isn't a rant — it's a reflection on the tension between **culture and competence**.
- High-trust, high-bar environments *do* exist — but they need deliberate shaping.
- Until then, stay quietly excellent — and protect your energy.

Optional final line:

"Niceness is lovely — but excellence, shared honestly and clearly, is what actually earns trust."

=====

What If the Obvious Isn't Obvious?"

A quiet reminder that your strengths might be invisible to you — and invaluable to others.

1. Intro: The Strange Surprise of Obviousness

- Open with a small, relatable moment — like someone thanking you for a “tiny” suggestion (e.g., adding comments to a complex DAX measure).
 - Reflect on how *you* were surprised they hadn't thought of it — not out of superiority, but genuine surprise.
 - Adding comments
 - Using SHareX for screenshots and short gifs/mp4s for testing evidence
 - Pose the central question: *What if the things I assume are obvious are actually my hidden strengths?*
-

2. The Craft Gap

- Talk about the idea of “*craftsmanship*” in *software development* — commenting code, thinking through UX, simplifying logic, writing for the next developer.
 - Mention how some teams focus on shipping fast or learning on the fly — and how this can leave behind the small details that make work maintainable or delightful.
-

3. Why This Matters (and How It Shows Up)

- Give a few examples of little “obvious” things you do:
 - Naming things clearly.
 - Leaving thoughtful comments.

- Refactoring for simplicity.
 - Considering the end-user's experience even if it's "just internal."
 - Asking "why?" before building.
 - Using folders logically to organise both in DAX but also in Selection pane and bookmark pane
 - Explain why these aren't just pet peeves — they're part of building sustainable, humane systems.
-

4. It's Not About Being the Smartest

- Acknowledge this isn't about being "better" than anyone else.
 - It's about *what you value* — and noticing that not everyone values the same things.
 - Mention that diverse strengths matter: some people are explorers, others are polishers. Teams need both.
-

5. Takeaway: Quiet Strength Is Still Strength

- Wrap up with a reflection on how you're learning to see your "obvious" habits as signals — signals of care, thoughtfulness, and quality.
 - Encourage readers to notice what they take for granted in their own practice — and to own it.
-

Optional Bonus:

- End with a personal note: "This post came from a conversation after I made a small suggestion in a PR and was thanked like I'd introduced something revolutionary. It stuck with me."
-

Tone Tips:

- Use *curiosity* instead of criticism.
- Use *"I" statements* rather than generalizations.
- Frame yourself as a learner *sharing perspective*, not passing judgment.

=====

What if SQL is a one-way city?

- **Opening scene:**
 - Compare SQL development to navigating a city with only one-way streets.
 - Set up the contrast: front-end work feels like cycling or walking directly to your destination, SQL feels like driving under rigid rules.
- **The SQL "one-way system" experience:**
 - Every step has to be done in a precise order.
 - No skipping steps, no creative shortcuts.
 - Mistakes (wrong join, typo, wrong field) can derail the whole journey.
 - "Best practice police" ensure you stick to the prescribed route.
 - Often means going a long way round to achieve something simple.
- **The front-end "free route" experience:**

- Blank canvas approach — you can choose the path.
- Multiple ways to solve the same problem.
- Immediate visual feedback when things work or break.
- Creative problem-solving feels energising.
- You own the journey and the solution.

- **Reflection:**

- Both have a place and both are valuable.
- For some, the structure of SQL is reassuring.
- For others (like you), the flexibility and creativity of front-end work is what drives engagement.

- **Closing thought:**

- Personal preference isn't about right or wrong — it's about matching the work to the way you think best.
- In a "one-way city", you'd rather hop on your bike, find your own route, and enjoy the ride.

=====

Adding comments to PRs

2. The Practice: Adding Context to PRs (150–200 words)

- Describe the habit: always add a small note in the PR/commit description.
- Format suggestion:
 - **One-line summary:** what was the purpose of this change?
 - **2–3 bullets:** list the *key changes* (e.g. "Added measure: Total Orders by Region", "Replaced bar chart with line chart for trend analysis").
- Emphasize: this is not documentation in the full sense, it's just a map for the reviewer so they know where to look.
- Takes less than a minute, but provides disproportionate value.

3. Why It Matters (200–220 words)

- **For reviewers:** they immediately know what to check, don't waste time guessing.
- **For the team:** smoother reviews → faster merges → more focus on building rather than chasing down issues.
- **For the future:** commit history becomes far more useful when debugging or auditing.
- In Power BI specifically, the mix of model changes + visuals + Power Query steps means "hidden changes" are common. Context is the safety net.

Contrast this with "silent commits" → frustration, longer review cycles, even political tension in the team ("why didn't you spot that in review?").

4. Building a Documentation Culture (150–180 words)

- Reference your *previous blog*:
 - You've already written about using the **Power BI Description box** to document measures, columns, and tables.

- You also covered **commenting measures** with sprint item references and context.
 - This PR habit is the *next logical piece*: instead of only helping *future developers* (via description box) or *yourself* (measure comments), you're actively helping your **peers in real time** during reviews.
 - Together, these three small practices create a **lightweight documentation ecosystem**:
 1. **Inside the model** → Description box.
 2. **Inside the code** → Measure commentary.
 3. **Across the workflow** → PR notes.
 - None of them require big overhead. Each takes seconds. Combined, they save hours.
-

5. Personal Note / Social Proof (60–80 words)

- Mention briefly: you started doing this naturally in commits, your data architect noticed, and now it's a team-wide practice.
 - Not framed as “look at me” but as *proof of adoption*: if it worked in one team, others may benefit.
-

6. Closing Call to Action (40–60 words)

- Wrap with: next time you open a PR, add a one-liner + a couple of bullets.
 - Small effort, big impact.
 - Link to your earlier blog so readers can explore your other practices for documenting Power BI work.
-

This expanded version does three things for you:

1. Stands alone as a strong article.
 2. Reinforces your earlier post by referencing it (good for traffic and building a theme).
 3. Positions you as someone building a *holistic approach* to Power BI documentation, not just tossing out tips.
-

I don't care about beans

1. Hook / Confession (80–100 words)

Start playfully: “*My team can tell you where to find the metric for the average density of white beans. Me? I wouldn't have a clue. In fact, I'm not even sure I could tell you the difference between a bean and a legume. And yet, I'm the one building the reports.*”

👉 Sets up the central tension: you don't care about beans, but you *do* care about making the beans make sense to others.

2. The blindspot (120–150 words)

Talk about how colleagues seem to *recall the subject matter like an old friend*. They know the metrics, the business logic, the “bean world.” Meanwhile, you sometimes feel on the back foot.

👉 Acknowledge honestly that it can feel like a weakness — but is it really?

3. Where your focus lies (150–200 words)

Shift the lens: your strength is in building clarity, not memorising beans.

- You enjoy the **puzzle of DAX**.
- You think about **UX in reports** (labels, descriptions, navigation).
- You leave reports **accessible and documented** so others can trust them.
 - 👉 This is the translation role — making the SME's bean-knowledge usable for people who *don't* live in that domain.

4. Why translation matters more than recall (150–200 words)

Draw the comparison:

- Subject matter recall is valuable, but it's like knowing all the trivia about beans.
- Translation is what makes those numbers actionable for decision-makers.
👉 Without translation, knowledge stays trapped in the heads of experts. With it, everyone benefits. (This is where you can mention your other practices like descriptions, comments on PRs, etc. — they're all part of the same philosophy.)

5. Reframe the “blindspot” (100–120 words)

Wrap up: *“So no, I don’t care about beans. I care about whether a stakeholder can find the number they need in two clicks. I care about whether a report tells a story without requiring an interpreter. And maybe that’s the point: someone needs to care less about beans, and more about clarity.”*

Approx Word Count: 650–750 (right in your blog sweet spot).

Outline: Remote vs Hybrid – A Data Professional’s View

1. Hook (set the scene bluntly, your style)

- “Every Thursday I pack up my laptop, drive into the office, and sit with my team to ‘collaborate’. Truthfully? I end up doing the same work I’d be doing at home — only with more interruptions and worse coffee.”
- Quick nod to the universal debate: some love hybrid, some hate it.

2. The nature of data work

- Data analysis, BI development, DAX modeling → deep focus, quiet problem solving.
- This type of work is often *more productive remotely* (less context switching, fewer ad hoc interruptions).
- Collaboration *is* important — but async tools (Git, Teams, ShareX screenshots, PR notes) often work better than sitting shoulder-to-shoulder.

3. The hybrid argument

- Acknowledge the positives fairly: bonding, easier for spontaneous chats, onboarding juniors.
- But — highlight the cost: context switching, commuting, reduced flexibility.
- Make it clear you’re not anti-social; you just think there are better ways to collaborate.

4. Remote done well

- Give practical tips: PR commentary, ShareX for visual collaboration, structured async check-ins, clear documentation.
- Show that full remote doesn’t mean “isolation” — it just means being deliberate about communication.

5. What this means for BI teams

- The best BI teams aren’t about being in a room together; they’re about shared standards, clarity in processes, and using the right tools.
- If those are in place, location becomes secondary.

6. Close

- Acknowledge: hybrid works for some people, some teams.
 - But for deep, technical, detail-driven work like BI, **remote isn’t just a perk — it’s often the smarter choice.**
 - (Optional cheeky kicker in your tone: “Hybrid? For me, it’s the worst of both worlds.”)
-

Hook: The Debate

Open with the sales example you gave. Set it up as “two schools of thought” — explicit measures vs reusable bases + filter pane.

2. The Case for Explicit Measures

- Transparency: logic lives in the measure, easy to see, easy to debug.
- Portability: measure works consistently across visuals, no hidden filter pane dependencies.
- Bookmark reliability: avoids the “phantom filter” bug you’ve seen.
- Documentation: when stakeholders ask, you can point directly to the measure definition, not go hunting for hidden visual filters.

3. The Case for Filter Pane + Base Measures

- Reuse: fewer measures in the model (less clutter).
- Simpler for developers who think SQL-first (filtering is a WHERE clause, so keep it “outside” the calc).
- Flexibility: analysts can self-serve by layering filters.

4. The Pragmatic Angle (Your Position)

- You lean toward explicit measures, especially in enterprise reports where reliability and auditability matter and where interactivity and lots of views of the same data is high.
- You acknowledge that filter pane approaches have their place — for *ad hoc exploration* or analyst-driven dashboards.
- But for production reporting, explicit beats hidden because trust and clarity matter more than neatness. And in any case, it doesn't take long for even a simple power bi report to have several similarly named measures. Documentation is the key here not avoidance.

5. Broader Link to the SQL vs Power BI Mindset

- SQL-heads: separate base metric + filter (mirrors SELECT SUM() + WHERE).
- Power BI-heads: encapsulate business logic in the semantic model for reuse and trust.
- Both have merit, but ignoring the semantic layer weakens the “Power” of Power BI.

6. Wrap-Up / Call to Action

Something like: *“Next time you’re faced with the explicit vs filter pane choice, ask yourself: am I building a one-off analysis, or am I building a reusable, trustworthy artefact? That answer should guide the approach.”*

Pragmatism Over Purism

- - Sure, fewer measures looks “cleaner” in the object list, but if logic is hidden in filter panes, you’ve just traded visible complexity for invisible complexity.

~~~~~

~~~~~

~~~~~

~~~~~

~~~~~

~~~~~

```
- Invisible complexity is always  
harder to maintain, because the next developer can't see it without clicking  
into every visual.
```

[illegible]

Killer argument you can use

👉 “Would you rather have 50 well-named, well-documented measures... or 10 cryptic measures + 40 hidden filter pane settings scattered across 20 visuals?”

Blog 1: The Importance of Dev Testing in Data Projects

(Professional, practical, “this makes teams stronger”)

Intro

- Hook: "How many passbacks in your team could have been avoided with better Dev Testing?"
- State the problem: many developers rush to “done” without testing their own work thoroughly.
- Thesis: Dev Testing is not a nice-to-have — it’s a core part of craftsmanship.

1. Why Dev Testing Matters

- Builds trust with stakeholders (fewer embarrassing bugs spotted in demos).
- Reduces passbacks → saves time for devs, testers, and PMs.
- Encourages developers to *own* their work end-to-end, not just hand it off.

2. What Good Dev Testing Looks Like

- Test with multiple filter contexts (in Power BI: slicers, bookmarks, drill-throughs).
- Stress-test edge cases (e.g., what happens when there's no data for a selection?).
- Validate numbers against the source system or expected business rules.
- Consider performance, not just correctness.

3. How to Build a Culture of Dev Testing

- Make it an explicit step in the definition of done.
- Encourage developers to write quick test scenarios before they start coding.
- Share testing checklists across the team.
- Celebrate developers who ship clean, low-passback work.

Conclusion

- Dev Testing is invisible when it's done right, but its impact is huge.
 - The mark of a professional isn't how fast they code, but how reliably they deliver.
-

Blog 2: The Rockstar Developer Myth — Why Charm Isn't Enough

(Slightly cheekier, satirical edge, but still professional)

Intro

- Hook: "Every data team has one. The Rockstar Developer."
- Describe the archetype: charismatic, brilliant at demos, but somehow deadlines slip and bugs creep in.
- Thesis: Charm and cleverness can mask inefficiency — but teams thrive on consistency, not rockstars.

1. What Rockstar Developers Do Well

- They're often the loudest voice in the room, selling their vision.
- They can produce flashy results that impress stakeholders.
- They're confident — which can make them seem more capable than they are.

2. The Hidden Costs of Rockstar Behaviour

- Overshooting estimates while others quietly pick up the slack.
- Passbacks that frustrate testers and delay delivery.
- Lack of accountability in admin tasks (e.g., updating ADO, writing documentation).
- Team imbalance: some devs become invisible because they *do everything right* without fanfare.

3. The Unsung Hero Alternative

- The reliable developer who tests thoroughly, documents, and communicates clearly.
- The one who balances technical delivery with team process.
- The one who doesn't *need* charm because their work earns trust.

4. Why Teams Should Value Consistency Over Charisma

- Sustainable delivery beats flashy one-offs.
- Trust is built on predictability and accountability.
- Real maturity is when excellence spreads across the team, not just sits with one person.

Conclusion

- Rockstar developers make for good stories but bad long-term results.
- If you want high-performing teams, celebrate the dependable, pragmatic developers who quietly raise the bar.

2) Blog Outline: *The Power of Finishing vs. Starting*

Intro

- Hook: Starting something new always gets attention. But finishing quietly holds everything together.
- Observation: In many teams, the spotlight falls on “shiny starters” while “steady finishers” are overlooked.

Section 1: The Lure of Starting

- Why starting looks good: visible progress, high energy, easy to showcase.
- How teams reward starters: demos, updates, “classic [name]” narratives.

Section 2: The Hidden Value of Finishing

- Finishing means testing, debugging, edge cases, boring admin, making it work *everywhere*.
- This is the difference between something “working in a meeting” vs. working *in reality*.
- Finishing prevents churn, reduces passbacks, and builds trust with end users.

Section 3: The Cultural Blind Spot

- Many teams unconsciously reward performance over performance.
- Why charisma and enthusiasm can overshadow steady delivery.
- The cost: unfinished work quietly picked up by others, bottlenecks hidden, burnouts created.

Section 4: How to Rebalance

- Practical ways teams can start valuing finishing:
 - Celebrate bug fixes and polish, not just flashy new builds.
 - Track hand-offs and completions, not just starts.
 - Share credit for the invisible glue work.

Section 5: Why Finishing is a Superpower

- In the long run, finishers build sustainable reputations.
- Stakeholders remember what *works*, not what demoed well once.
- A balanced team needs both starters *and* finishers — but neglecting the latter is risky.

Conclusion

- Call to action: Next time you’re applauding a shiny start, ask — who’s making sure it finishes?
-

Outline: Building a Custom Scatter + Line Chart in Power BI with Deneb & Vega-Lite

1. Hook / Intro

- Start with the customer request: “They liked the scatter plot, but asked: can it also show change over time for each consultant?”
- Point out the limitation: native Power BI visuals can’t layer scatter + connected lines per category cleanly.

2. The Challenge

- Requirement = *scatter plot of consultants with primary vs secondary care calls + dots per month + connected lines to show trend*.
- Why native visuals don't cut it (scatter = point-in-time, line chart = requires axis continuity, combo visuals don't support this use case).

3. The Tool: Deneb + Vega-Lite

- Short intro for readers: what Deneb is, why Vega-Lite is powerful.
- Why it was the right choice: flexible, JSON-based spec, lets you "break out" of Power BI's visual limitations.

4. The Solution (Step-by-Step)

- Step 1: Prep your dataset (Consultant, Month, Primary Care, Secondary Care).
- Step 2: Define Vega-Lite spec → scatter marks for each consultant/month.
- Step 3: Add color encoding = month grouping.
- Step 4: Add line encoding = dotted line connecting each consultant's dots.
- Step 5: Format + polish for readability.

(**Tip:** you don't need to paste raw JSON — just pseudo-code and a screenshot).

5. The Result

- Show before (standard scatter) vs after (custom Deneb visual).
- Highlight how it gave the customer *exactly what they wanted* → trend over time *and* scatter relationship in one chart.

6. Lessons Learned

- Sometimes "no native visual" isn't the end of the story.
- Deneb opens doors for advanced users → fills the gap between SQL/DAX and user experience.
- Key takeaway: don't be afraid to bend the rules if it solves a genuine business problem.

7. Call-to-Action / Outro

- Suggest readers try Deneb for their "unsolvable" visual requirements.
- Invite comments: *"Have you built a custom Deneb visual? What was your use case?"*

Blog Outline: *"The Myth of Everyone Does Everything: Why Teams Need Specialists Too"*

1. Hook / Opening

- Share the common agile mantra: *"Everyone should be able to pick up any task in the backlog."*
- Acknowledge why it sounds great in theory: flexibility, shared knowledge, no single point of failure.
- Set up the tension: *But in practice, this often backfires.*

2. The Problem with "Everyone Does Everything"

- **Shallow progress:** Generalists can "keep things moving" but often don't solve root issues.

- **Task bouncing:** Difficult problems get passed around until someone with the right depth eventually handles it — wasting time.
- **Specialist ceiling:** Experts can't fully use their skills if they're constantly pulled into shallow or mismatched work.

3. The Case for Recognising Specialisms

- **Efficiency:** Specialists fix deep issues faster and more reliably.
- **Resilience:** Having both breadth *and* depth in a team means you cover edge cases and systemic problems.
- **Quality:** Specialists build scalable solutions, not just stopgaps.
- **Morale:** Specialists feel valued instead of sidelined.

4. Balance, Not Silos

- This is not an argument for rigid silos (“only Jane touches the database”).
- Instead:
 - Generalists handle breadth, day-to-day tasks, quick answers.
 - Specialists are recognised for depth and brought in when complexity spikes.
- Think of it like medicine: you want a GP for general health, but you *also* want surgeons and cardiologists.

5. Practical Ways to Get There

- **Map strengths:** explicitly acknowledge who's strong at what.
- **Pairing:** let generalists shadow specialists so knowledge spreads without bottlenecks.
- **Task triage:** match the task's complexity to the right person (not “anyone can pick it up”).
- **Celebrate both styles:** value breadth and depth equally — they serve different needs.

6. Closing Thought

- “Everyone does everything” sounds egalitarian, but often creates inefficiency and frustration.
 - The best teams are those that **embrace specialisms without silos** — where generalists keep things moving, and specialists create lasting stability.
 - *Breadth and depth together make a resilient, high-performing team.*
-

Blog Outline: Being Excellent Isn't Enough — It Has to Be Visible

1. Opening Hook

- Share a relatable observation: “In school, being conscientious and finishing fast got rewarded. In the workplace, it's not always noticed.”
 - Pose the central question: *Why does charisma and visibility often seem to outweigh quiet excellence?*
-

2. The Reality of Workplaces

- Performance vs. substance — both matter, but not equally in all environments.
 - How “quiet excellence” is often assumed as baseline, while visible contributions get praised.
 - Link to your field (data/tech): some people host workshops, demos, or write up flashy spikes; others just deliver quietly robust solutions.
-

3. Why Excellence Alone Isn't Enough

- Managers and stakeholders often can't see the hidden complexity or effort behind smooth delivery.
 - “Reliability becomes invisible” — if you always deliver, you risk blending into the background.
 - Charisma, storytelling, and bluster act as amplifiers — making work seem bigger, even if it isn't deeper.
-

4. Practical Shifts for the Quietly Excellent

- **Don't stop being excellent** — it's your foundation.
 - But consider small, authentic ways to *make it visible*:
 - Share snippets (LinkedIn posts, teasers, blogs).
 - Use artifacts: screenshots, before/after comparisons, mini writeups.
 - Offer insights during retrospectives or sprint reviews (without overperforming).
 - Emphasise *authenticity*: it's not about showboating, it's about letting others see the value that's already there.
-

5. Personal Reflection / Lesson Learned

- Share a moment (anonymised, generalised) when you realised that speed and quality weren't enough by themselves.
 - What you changed (e.g. slowed down, diverted energy into side projects/blogs).
 - What you wish someone had told you earlier in your career.
-

6. Closing Thought

- Excellence is the baseline; visibility is the amplifier.
 - If you want your work to create opportunities beyond your current sprint/task, you need both.
 - Encourage readers: “If you're quietly excellent — don't change who you are. Just find a way to make your brilliance visible.”
-

Invisible Excellence: Why Quiet Experts Stay Quiet

Intro

- Hook: Teams often have unsung heroes — the person who quietly makes things work but isn't the loudest voice in the room.
- Set up the tension: Why do these people get overlooked? What happens to team culture when they do?

1. The Quiet Expert Archetype

- Deep craft knowledge, often self-taught.
- Prefers solving problems to talking about them.
- Not flashy; doesn't seek the spotlight.
- Work is robust, often invisible because it "just works."

2. Why They Stay Quiet

- Anxiety around performance/theatre (demos, big meetings).
- Preference for craft over communication.
- Past experience: when they speak, they're often overridden by louder voices.
- They don't need external validation in the moment — the work speaks for itself (or so they think).

3. Why Teams Overlook Them

- Cultural bias toward performance and theatre.
- Leaders default to the "noisy" contributors for visibility.
- Shiny, collaborative problem-solving is seen as more fun than deferring to the SME.
- Mistaken belief: "If they're quiet, maybe they don't know."

4. The Cost of Ignoring Quiet Experts

- Slower problem-solving (reinventing wheels).
- Shallow understanding in the team (quick fixes vs deep knowledge).
- Burnout risk for the expert (invisible labour, lack of recognition).
- Mediocrity becomes the cultural norm.

5. How to Harness Invisible Excellence

- Leaders to deliberately draw out the quiet experts in structured ways (not just rely on them jumping in).
- Encourage knowledge-sharing in writing, documentation, or async — not just in the spotlight.
- Reward robust, drama-free delivery as much as flashy demos.
- Build a culture where expertise is celebrated, not just performance.



Conclusion

- Quiet experts aren't a weakness — they're often the foundation of a high-performing team.
- Great cultures don't let excellence stay invisible.

Power BI Developers Pre-Flight Checklist

Bullets to include (your starting list, expanded slightly):

- ☒ Numbers make sense (spot-check against source).
- ☒ Dataset connection correct (prod, not dev/test).
- ☒ Slicers reset to default values.
- ☒ Filter pane closed / tidy.
- ☒ Titles, labels, and tooltips are consistent.
- ☒ Unused fields removed from model.
- ☒ Bookmarks / drillthrough pages still work.

-  Performance check (no query folding breaks, no excessive visuals).
-  Accessibility basics (contrast, alt text where relevant).

Angle:

You're not saying *this is the definitive checklist for everyone*, but *this is the practical routine I use every day to make sure my reports don't come back with easy fixes or embarrassing oversights*.

Here's a **bulleted outline** for *"Hacky" is Not a Dirty Word: It's All About Time and Place*:

1. Hook / Opening (set the scene)

- The word *"hacky"* often gets a bad reputation in data teams.
- Many devs (and managers) hear *"hacky"* and immediately think: *bad practice, fragile, unprofessional*.
- But... context matters. Sometimes *hacky* = creative, playful, pragmatic. Other times *hacky* = dangerous, corrosive, high-risk.
- Thesis: *Hackiness isn't the problem. Putting it in the wrong place is.*

2. Two kinds of hackiness (the split)

- **UI Hacky (the good kind)**
 - Bookmarks, buttons, shapes, layering visuals, tricking conditional formatting.
 - Creates delight, makes users feel "wow, I didn't know Power BI could do that."
 - Low blast radius if it breaks — usually one page or one visual.
 - Encouraged: this is creative problem solving.
- **Model/Measure Hacky (the risky kind)**
 - Invented keys, unnecessary bridge tables, duct-tape relationships, quick-fix DAX.
 - Creates fragility: silent wrong numbers, unexplained discrepancies.
 - High blast radius: undermines trust in data and in the team.
 - Dangerous: often invisible until it fails under pressure.

3. Why teams confuse the two

- *"Hacky"* in everyday speech lumps them together.
- Many junior devs don't differentiate layers: they just want it to "work."
- Effort is often visible at the UI layer ("look what I made!") but invisible at the model layer ("you can't see the rot until it cracks").
- This is why cultural clarity matters.

4. How to create a healthy "hack culture"

- Don't ban hackiness — channel it.
- Encourage playful hacks in UI/experience.
- Insist on rigour in models/measures (testing, simplicity, stress-testing edge cases).
- Create space for experimentation (PoCs, sandboxes) where *hacky* approaches are allowed to breathe before being hardened.

5. Personal stance (your voice here)

- Share how you draw the line:
 - Happy to hack visuals for delight.
 - Reluctant to hack the model, because the business builds trust on those foundations.
- Why you prefer to keep the model clean and push complexity into well-written measures (portable, tweakable, auditable).
- Frustration when you hear “*hacky but works*” used casually in the wrong layer.

6. The bigger cultural point

- Hackiness is not about *what* you did, but *where* you did it.
- A mature BI team knows when to embrace creative hacks and when to slow down for rigour.
- Teams that don’t distinguish pay the price later (technical debt, eroded trust, firefighting).

7. Closing punch

- “Hacky” is not a dirty word.
- In the right time and place, hackiness is innovation.
- In the wrong time and place, hackiness is rot.
- The craft of BI is learning the difference.

Here’s a possible structure for the blog post “*Customer Is King (Especially in Data Visualisation)*”:

1. Hook (scene-setting anecdote)

- Recount the conversation in a light, narrative way (without naming names, but hinting at “a recent team discussion”).
- Show the tension: PO bringing gold-dust feedback, developers challenging the value.
- Pose the question: *if we’re not building to make life easier for the customer, what are we doing?*

PO: “The customer would like us to add an average line to the bar chart” DA / Ted: “Why? What’s the point? There is a card above the chart that displays the last 12 months average. Could we not use or repurpose that?” My thoughts: (a) customer feedback is GOLD. If it’s not difficult to implement we should absolutely do what they ask for and (b) they’re asking for a visual enhancement for a reason: makes comparison easier and cognitive load lower as can literally see how the average compares with each bar. If we’re not building to the end user’s specs (who is ultimately paying us) what are we doing other than massaging our own egos??

2. The Value of Customer Feedback

- Customer feedback = rare and valuable.
- Why it matters: end-users live with the report every day, they notice pain points you don’t.
- Ignoring them risks eroding trust (“we told you, but you didn’t listen”).
- Bonus: small requests often have big ROI (e.g., average line improves usability massively).

3. The Cognitive Angle

- Explain why a line on a chart isn’t redundant, even if the same info exists elsewhere.
- Visual comparisons are cognitively easier → less friction, faster insights.

- Quote or reference the idea: “If your user has to *think about* how to compare two numbers, you’ve already lost them.”

4. The Developer Trap

- Why devs resist:
 - Defensive mindset (“our solution is already enough”).
 - Backend-first thinking (SQL mindset over UX mindset).
 - Ego/ownership (“our way is smarter”).
- Point out: we sometimes optimise for *ourselves* rather than the end-user.

5. Customer-Centric Principles

- Translate your thinking into rules of thumb:
 1. If it’s **low effort, high clarity**, do it.
 2. Always privilege **end-user experience** over backend neatness.
 3. Ask: *Would I want this feature if I had to use this dashboard every day?*

6. A Broader Lesson

- Customer-first mindset doesn’t mean saying yes to *everything* → it means weighing requests on usability and impact.
- In data viz, especially Power BI, small UX enhancements often create disproportionate value.
- Teams that learn to embrace this win trust and adoption.

7. Close with a Call to Action

- “Next time your customer asks for something ‘small,’ don’t dismiss it. Pause and ask: what’s the experience they’re craving? Chances are, that tweak might turn a functional report into one they love.”

Core Thesis: Teams often impose arbitrary technical limits not based on merit, but on fear, familiarity, and a desire to avoid learning. This prioritizes short-term comfort over long-term capability and excellence.

1. The Hook: The Meeting Where It Happened

- Set the scene briefly: a routine refinement meeting.
- Quote (or paraphrase) the devastatingly simple question that was asked: *"If we have an unwritten rule that we don't use Power Query, shall we just remove that from the options?"*
- Describe your internal facepalm moment. The sheer absurdity of making an "unwritten rule" official without ever questioning *why*.

2. Deconstructing the "Unwritten Rule"

- What an "unwritten rule" really is: a symptom of tribal knowledge, fear, or past trauma (e.g., "someone used it badly once").
- The difference between a **principled decision** (e.g., "We avoid Power Query for these specific performance reasons in our context") and an **arbitrary limitation** ("We don't use it because we don't use it").

- The unspoken message: "Our collective ignorance is more valuable than the potential of this tool."

3. The Real Reasons Behind the Ban (The Anthropology)

- **Comfort Zone Governance:** The team is opting to stay within its collective SQL/DAX comfort zone. Learning is seen as a cost, not an investment.
- **The Illusion of Efficiency:** They believe limiting options will make decisions faster. In reality, it makes them *simpler*, not *better*. They are trading quality for speed.
- **Risk Aversion disguised as Best Practice:** It's easier to say "no" than to evaluate a tool's pros and cons. This is risk aversion masquerading as technical wisdom.
- **Groupthink in Action:** No one questioned it. The proposal was to codify the status quo, not challenge it.

4. The High Cost of Self-Imposed Limitations

- **Stagnation:** Skills atrophy. The team's toolbox becomes a *toolbox*.
- **Sub-Optimal Solutions:** Future problems will be forced into a SQL/DAX-shaped hole, even when a Power Query solution would be cleaner, faster, or more maintainable.
- **Driving Away Craftsmanship:** How does this environment look to curious, growth-minded developers? It tells them, "Your desire to learn and improve is not welcome here." This is how you lose your best people.

5. The Alternative: A Culture of Learning & Pragmatism

- What a healthy response could have been: *"We're not experienced with Power Query. Could we spike a small task with it to understand its strengths and weaknesses for our use case?"*
- **Tool Agnosticism:** The best tool for the job is the one that best solves the problem. This requires a baseline knowledge of the options.
- **Learning as a Team Value:** Framing new tools not as threats, but as opportunities to expand the team's collective capability.

6. Conclusion: A Call for Written Principles, Not Unwritten Rules

- Recap: Unwritten rules are a cancer on growth. They are unchallengeable and often irrational.
- The antidote is to replace "unwritten rules" with **"written principles."**
 - *Bad Rule:* "We don't use Power Query."
 - *Good Principle:* "We prioritize maintainability and performance. When considering a new tool or method, we evaluate it against these principles."
- Final thought: Excellence isn't born from comfort. It's born from a willingness to challenge your own assumptions, even when it's difficult. What unwritten rules is your team following, and are they serving you, or holding you hostage?

Taps Metaphor

1. The Hook: A Tale of Two Sinks

- Describe two scenarios:
 - **Sink A:** Incredible, high-pressure plumbing connected to a cheap, wobbly plastic tap that drips and is hard to turn.

- **Sink B:** Perfectly adequate plumbing connected to a beautiful, weighted, easy-to-use waterfall tap with perfect pressure.
- Ask the reader: Which sink provides a better experience? The answer is obvious. This sets up the entire argument.

2. Meet the Data Plumber

- Define the "Data Plumber": The highly skilled engineer focused on the backend—data ingestion, warehousing, pipelines, and SQL optimization.
- **Their Pride:** Their work is complex, robust, and hidden from view. They rightfully take pride in building a system that doesn't leak.
- **Their Blind Spot:** They believe their job is done when the data is in the warehouse. The front-end is a "simple" presentation layer, a mere pane of glass.

3. Meet the Tap Designer (The Analytics Engineer)

- Define the "Tap Designer": The craftsman focused on the last mile—the semantic model, the UI/UX, the clarity of reports, the performance of DAX.
- **Their Pride:** Creating an intuitive, trustworthy, and even enjoyable experience for the end-user.
- **Their Focus:** Reducing cognitive load, ensuring accuracy, and building tools that people *want* to use. They ask, "Is the water the right temperature and pressure? Is the tap a pleasure to use?"

4. The Great Mismatch: When Plumbers Choose the Taps

- - This is where the dysfunction happens. The plumber, who thinks in terms of data delivery and logic, cannot comprehend the user's experience.

- ****Insert the story here as a case study:****

> "I saw this play out just recently. A customer asked for a simple average line to be added to a bar chart. The immediate reaction from the backend-focused members of the team was bewilderment: *'Why? There's already a card above the chart that shows the average number.'

>

> "They were thinking like plumbers. The data was already delivered! The pipe had done its job. The number was present.

>

> "But the customer was thinking like someone at the sink. They didn't want to have to hold a number in their memory and visually compare it to each bar. They wanted the benchmark integrated into the visualization itself—to turn the tap and instantly see the relationship. They were asking for a better **experience**.

>

> "The plumbers saw a redundant feature. The user was asking for a more intuitive tap."

- ****Connect it back to the metaphor:****

- This story exemplifies the core disconnect. The team was focused on the *efficiency of the plumbing* (avoiding "redundancy") while completely missing the *usability of the tap* (reducing cognitive load).

- When you let plumbers design taps, you get a system that is logically sound but humanly frustrating.

5. Why This Metaphor Matters: The Bottom Line

- **The Tap is the Product:** For the end-user, the tap *is* the entire system. They don't see the plumbing. Their trust and satisfaction depend entirely on the interface.
- **Wasted Investment:** Millions can be spent on data infrastructure, but if the tap is bad, the ROI is zero. Users will abandon the tool.
- **A Call for Specialization:** Just as you wouldn't ask a master plumber to design a luxury tap, you shouldn't expect a data engineer to be a master of front-end craft. They are distinct, equally valuable disciplines.

6. The Solution: Hire a Tap Designer

- Argue for the formal recognition of the "Tap Designer" role—the **Analytics Engineer** or **Data Visualization Engineer**.
- This person acts as the crucial translator between the complex world of the plumbers and the practical needs of the users.
- Their goal is to ensure the incredible power of the plumbing is delivered through a flawless, intuitive interface.

7. Conclusion: It's Time to Value the Experience

- Recap the metaphor: Stop celebrating the hidden pipes and start obsessing over the user's experience at the tap.
- Final thought: **"The most sophisticated data pipeline in the world is a failure if the user can't get a clean, clear drink of water from it. It's time we started designing for the thirst, not just for the pipe."**

=====

Blog Post Angle: "I thought I was dogmatic. Turns out I'm pragmatic"

Working Title: *The Tyranny of 'Best Practice': Why Sometimes a 'Worse' Solution is Better*

Core Thesis: Blindly following prescribed "best practices" can be the *worst* thing for your team's velocity, knowledge sharing, and long-term health. True craftsmanship is knowing when to break the rules for a greater good.

Outline & Key Arguments:

1. **The Allure of the Purist:** Start with the common dogma. "Always use CALCULATE!" "Never use FILTER inside iterators!" "Eliminate all row context!" Show you know the "rules."
2. **Introduce the Heresy:** The 5% performance vs. 50% understanding trade-off. Frame it not as laziness, but as a **strategic calculation**. Code is read by humans far more often than it is executed by machines.
3. **The Two Audiences for Code:**
 - **The Machine:** Wants raw speed and efficiency.

- **The Human Teammate (Including Future You):** Wants clarity, transparency, and the ability to debug and modify.

4. When to "Break the Rules" (Your Pragmatic Framework):

- **For Knowledge Transfer:** The "teaching" code you described. A verbose, step-by-step measure is a better onboarding tool than a magical, optimized one.
- **For Debuggability:** Being able to `RETURN _filteredTable` is a superpower when logic gets complex.
- **For Team Cohesion:** If your team understands `SUMX` but gets lost in context transition, forcing `CALCULATE` creates a knowledge silo around you.

5. Calling Out the Influencers: This is where you get to be subtly bold.

- "The online discourse, dominated by solo experts, often optimizes for the *craftsperson in isolation*. But we work in *teams*."
- "A 'best practice' that only one person on the team can understand or maintain is, by definition, a *bad practice* for that team."
- "The goal isn't to write the most clever code; it's to build the most effective and resilient *team*."

6. The Real Best Practice: Contextual Excellence. The true skill is not memorizing rules, but developing the judgment to ask: "What does *this* code, for *this* purpose, for *this* audience, need to be?"