I've spent a good part of my career deep in the details — working to understand systems, data, and how things actually function beneath the surface.

Sometimes that's exactly where you need to be.

Sometimes… it's where you get stuck.

I wrote this piece to explore a pattern I've seen again and again — how even strong analysis can lose its way, and what it takes to move forward again.

The Dungeon Paradigm

How Even the Best Analysis Can Get Stuck

I think I'm going to start at the end.

Partly because the most recent example is still fresh, and partly because it's one of the clearest examples I have seen.

I was recently brought into a company to help shore up a number of issues within their IT environment. They had recently gone through a major reduction in staff. A lot of knowledge had departed along with the people who built and maintained these systems, and what remained was fragmented and poorly informed. The problems weren't small, and they were *everywhere*.

One of the first things I was shown was a collection of reports — hundreds of them — all running periodically from a single PC. The machine belonged to someone who was no longer there. They could never shut it off. No one knew exactly what ran, when it ran, or how it worked.

So, I started where I usually do — at the bottom.

With the help of a small utility I created, I was able to inventory everything: what reports existed, what they looked like, what data sources they used, what parameters they took, and what results they produced. Within a week we had a complete picture (and database) of something that had previously been thoroughly opaque. It was one of those moments where things began to come clearly into focus.

And then we moved on.

The focus shifted to another problem — a legacy PC database produced a report that had evolved and been carried forward for a decade and a half. The request was simple: recreate the report and upscale it to an enterprise database. I first asked to see the report. That seemed pretty reasonable, but it never came.

I did what I had to do. I went back down.

The PC database was built from layers of queries. At first it looked manageable, but as I started to unwind it, I realized what I was actually looking at. Each query depended on another, and that one depended on another, and another — at times 30 layers deep!

By the time I reached the bottom, I had spent weeks reconstructing the logic: understanding how the numbers were produced, tracing relationships, following joins, and trying to piece together how the system worked. And then something else started to emerge.

The numbers just didn’t make sense.

In the very first step there were duplicates — joins that didn’t use full keys, rows multiplying quietly as they moved through the system. I flagged it immediately. If the foundation was wrong, everything built on top of it would be wrong too.

But the work continued, and I kept going — fixing what I could, rebuilding logic, translating it into something more structured. And with each step, I would find something else that didn’t quite line up.

A pattern was starting to emerge.

At the earliest stages, the queries were trying to boil everything down to single product identifiers. That product had a structure — effectively a bill of materials — and the logic would reduce everything down to that one item. From there, it would be linked to inventory levels and forecasted demand. That part made sense.

But later in the process, the logic shifted. That same individual product would be exploded back out into variations of its underlying components to generate a production plan. When that happened, the original numbers didn't stay at the product level — they were replicated across each of those variations/components. What started as a single number became many, and not just many, but exponentially more as it moved through the system.

At that point, the report stopped being internally consistent. It was trying to do two different things at once — demand management at the product level and production planning at the component level — and the two didn't align.

I raised this as soon as I saw it, using simple examples to show the net effect. If the base numbers were being multiplied, the final output couldn't be trusted. That led to conversations with the person who had originally helped build and maintain the system. We walked through the examples together, and each problem raised was acknowledged and confirmed.

That was the moment something shifted for me.

The issue wasn't just complexity. It was that the system had evolved in a way where the outputs were no longer tied to a single, consistent interpretation. And yet the base expectation remained the same: *recreate the report*.

I continued, tracing each layer, rebuilding the logic, correcting what I could along the way. And as I did, the pattern kept repeating — something would look reasonable at one step, then break at the next.

Eventually, I condensed everything down into a much simpler structure. What had originally been dozens of interdependent queries became a small number of clearly defined steps. Each one flowed into the next, and the final output could be reproduced in a clean and understandable way.

It worked. But it didn't solve the problem.

The numbers were still not meaningful, in the least, because the underlying issue hadn't changed. There was still no consistent way to clearly explain what the system was trying to do. The logic was clean, but the definition wasn't.

And that's when it became clear that I wasn't reconstructing a system. I was navigating the darkness of the dungeon.

Finding a Way Out

This wasn't the first time I had found myself down there.

Earlier in my career, I had moved from Chicago to San Francisco. In Chicago, I had built a vast network over many years, and work primarily came through relationships and reputation. I had not even needed a resume for a very long time. In San Francisco, that network evaporated overnight.

I took on work where I could find it, and often that meant being positioned simply as "someone who knows SQL." That wasn't the full picture, of course, but it was enough to get in the door.

One of those early engagements was with a large capital management firm responsible for an enormous volume of assets and a complex reporting process tied to regulatory filings. The goal was to redesign processes and automate those filings — to move from manual preparation to something repeatable, reconcilable, and reliable.

I started the way I usually do, from the data.

At first, things moved quickly. I was able to reproduce a portion of their reports using the underlying system and align them with what had been filed. But then the numbers stopped matching — not slightly off, but materially different.

My goal was never to question the filings; it was to understand them and to identify any undocumented outliers that needed to be accounted for. If I couldn't reproduce what had already been submitted in prior periods, I couldn't confidently automate anything going forward. So, I went deeper.

I traced everything — account by account, step by step — building a process that showed exactly how the numbers were derived and where they diverged. It took much time, and it wasn't easy. While I was trying to understand the system, the "system" was pushing back. The discrepancies weren't immediately welcomed; they were uncomfortable, if not downright objectionable.

But the work itself seemed clear, so I stayed with it.

Eventually, I brought the reconstruction forward — not as a conclusion, but as a question: this is how I'm arriving at these numbers; where precisely does it differ from what has been reported?

That question made its way to the CFO, and in that conversation something changed. For the first time, the focus wasn't on whether the output matched expectations; it was on understanding how the output was being produced.

From there, things moved quickly. The discrepancies were real, the process needed to be corrected and, once that was acknowledged, everything opened up. Within a few months, the reporting process had been fully automated — consistent, reliable, documented, and understandable.

But what I remember most about that experience is the transition — the moment where the work shifted from “why doesn’t this match?” to “what is actually happening here?”

That was the way out.

Learning to Move

There have also been times where I didn’t get stuck.

Shortly after that engagement, I took on another project — this time just a few floors up in the same building. The contrast was immediate. Instead of being brought in to reproduce something, I was brought in to help build something.

In my first meetings, I spoke with different leaders in the business. Finance needed better visibility, Marketing was dealing with growing volumes of data, and Operations was trying to manage a complex supply chain across hundreds of locations and multiple production stages. Each problem was different, yet all were connected.

I chose to focus on Operations.

What stood out right away was how much effort was going into maintaining the current process. Highly capable people were spending most of their time working through spreadsheets — reconciling, adjusting, forecasting — trying to produce something that could guide the business forward.

So we started there, not by building everything at once, but by taking small, meaningful steps.

Each week, there was something to show. Sometimes it was a small, simple automation that saved hours of effort. Sometimes it was a clearer view into the data. Sometimes it was a piece of the larger system taking shape. Individually, these weren’t the final solution, but together they were moving in a direction, and that direction was always visible and consistent with the strategy.

Over time, those pieces all began to connect. Forecasts, inventory, and production steps became part of a larger structure. It wasn’t simple, but it worked because it was grounded. Every step was tied back to what the business needed, every result could be explained, and every output could be trusted.

And everyone was part of it.

There was no resistance, no disconnect, because the work was visible and participatory as it was happening. By the time the system came together, it didn’t feel like a handoff. It felt like a shared accomplishment.

That experience stayed with me because it showed something different. You don’t have to avoid the dungeon, but you don’t have to stay there either.

What the Dungeon Teaches

I've spent a good part of my career in the dungeon.

Sometimes I've gone there intentionally, because that's where the answers are. Sometimes I've stayed too long, sometimes I fought my way out, and sometimes someone else has helped me back out.

Over time, I've learned something simple: the goal of analysis isn't to try to understand everything at every level, it's merely to understand enough to keep moving. This goes naturally with keeping aware of what you are moving towards.

The most effective analysis work doesn't happen entirely at the surface, and it doesn't happen entirely in the depths. It happens in the movement between them — knowing when to go down, knowing when to come back up, and knowing what to bring with you when you do.

The way out of the dungeon isn't hidden.

It just isn't where the analysis is focused.