Summary

I began my data management and analysis career at a multi-brand, multi-channel e-commerce retailer, where I was hired as the **Report Coordinator** for the Inventory Planning department. At the time, the role consisted entirely of manipulating ERP-exported flat files in Excel and emailing static reports to stakeholders. There was no automation, no direct database access, and no data validation—just nightly CSVs and a lot of manual effort. The process led to reports being released so late in the day that the planners were accustomed to operating on reports from the previous day.

The work was tedious and error-prone, offering little challenge beyond managing long-running Excel calculations. At one point, while juggling two desktops to keep up, I pushed both a CPU and a hard drive to failure purely from spreadsheet volume. That moment confirmed what I already suspected: if I can break a system just by using it as intended, it needs to be rebuilt.

I have never been inclined to accept inefficient processes as “just the way it is.” From the beginning, I took an iterative approach to overhauling the department's reporting strategy—improving efficiency, functionality, and impact at every stage.

This role marked the start of my data career and shaped how I approach every challenge: assess the problem, identify the root cause, build iteratively, and never settle for a solution that merely “works” when it can be made better.

Initial Improvements

The first set of changes I made focused purely on automation—with the explicit goal of ensuring no one noticed any difference in the reports they were used to. The format, layout, and delivery remained identical; the only change was how they got there.

At first, I continued working with the nightly ERP-exported flat files and doing the heavy lifting in Excel. But I introduced VBA to handle the entire workflow: splitting a single export into multiple worksheets (typically one per planner, plus a consolidated sheet and summary), applying formulas (VLOOKUPs, aggregations, and PivotTables), and applying consistent formatting. For some reports, I also built automated email delivery upon completion.

Each report I automated gave me back time to automate the next. Over time, the daily report cycle shrank dramatically—from finishing around 2–4 P.M. to being consistently done by 10–11 A.M.

Integrating Live Data

The next major improvement marked the beginning of self-service reporting within the department. I was still operating within the constraints of Excel, but I extended its capabilities by building a GUI interface powered by VBA.

The interface allowed users to select from a list of available reports using buttons, with optional radio buttons to filter by planner ID. Behind the scenes, it connected to the OLTP inventory planning system via an ODBC connection, executed parameterized SQL queries, and performed any necessary data manipulation and formatting directly in Excel. The result was a familiar, user-friendly front end backed by real-time data.

I introduced the tool during a department meeting, where I also led my first stakeholder training session—my first experience presenting technical work and supporting users in adopting new tools. The rollout was smooth, and adoption was immediate. The new system was well received across the board, with users appreciating both the increased flexibility and the time savings.

First Reporting from Data Store

At the time, our SQL Server Reporting Services (SSRS) environment was significantly underutilized, and our ERP system data was populated into a data store housed in Oracle using the same exported flat files I had been using for my daily reports. After proving myself through consistent improvements, I was granted full access to both environments.

I dove in—headfirst, without checking the water depth—and began developing my first PL/SQL procedures. I translated the existing business logic, formatting rules, and report structures into a combination of PL/SQL for data preparation and RDL files for SSRS report design.

Once the reports were validated and approved by both stakeholders and management, I deployed them to the report server and scheduled automated delivery to the planning team.

The result: planners received their daily reports before their first cup of coffee—typically between two and four hours earlier than before. More importantly, the new system provided consistency, reduced reliance on manual processes, and continued to function even if I was out sick or on vacation.

Transition from Legacy ERP to Dynamics AX

Driven primarily by PCI compliance requirements—but also by a long-overdue need for modernization—the company replaced its 35-year-old legacy ERP system with Microsoft Dynamics AX, hosted on SQL Server. The legacy system had stored credit card data in plaintext and lacked even the most basic encryption or security capabilities.

By this point, I had transitioned from the Inventory Planning department into the Data Management and Decision Support team. I was designated as the primary validator for both inventory and sales data—responsible for reconciling millions of rows and terabytes of data throughout the implementation. This required a deep understanding of both the old and new systems, as well as the business rules that governed the data.

As part of the rollout, we built a new Operational Data Store (ODS) to support reporting needs, loaded nightly through an ETL process. A Data Warehouse was also implemented using SQL Server Analysis Services (SSAS), which introduced multidimensional modeling and MDX querying to the environment. While I did gain hands-on experience with MDX and cube-based reporting, the reality of recurring overnight processing failures due to limitations in multidimensional models in earlier versions of SSAS—and the resulting 2 to 4 A.M. troubleshooting—taught me more about system fragility and sleep deprivation than any analytics value I got from SSAS.

Integrating Inventory Data in SSRS

To improve report functionality and reduce report sprawl, I began integrating data from the inventory planning system with ERP data into a single reporting layer.

The inventory system lived on an aging AS/400 backend—old enough to be collecting Social Security by then—so I worked with IT to establish a linked server connection between it and our SQL Server environment. To reduce the impact on the OLTP system, I designed the process to extract only the necessary data into temporary tables on SQL Server, where I added indexing before joining it with ERP data for reporting.

This approach enabled more complete and flexible reports, allowing planners to see inventory and transactional data in one place. It also allowed me to deprecate the Excel-based GUI and ODBC tool I had previously built, consolidating everything within SSRS.

Building My First ETL and Custom Data Store

From the moment the linked server to the inventory planning system was requested, I knew it was a stopgap. The data was siloed, the infrastructure fragile, and the architecture unsustainable. We already had an Operational Data Store (ODS) in place—it was just a matter of time before I could build a proper ETL process to bring the inventory planning data into the fold.

Once I had the necessary access and buy-in, I designed and deployed a schema and a full SSIS project to extract, transform, and load data from inventory system into the ODS. The process ran nightly, pulling fresh data, applying all necessary transformations, and loading it into a reporting-optimized schema.

Yes, it was a classic dump-and-pump: truncate the tables, bring in clean data, and rebuild indexes. No temp tables. No runtime joins on linked servers. No OLTP side-eye when someone refreshed a report at 9:01 A.M.

This shift moved all reporting traffic off the transactional systems—except in a few tightly scoped cases where real-time data was truly necessary. It dramatically improved performance, reduced failure points, and gave stakeholders faster, more reliable access to the information they needed. And for me, it was a crash course in data architecture, orchestration, and the oddly satisfying joy of seeing a full SSIS package light up green at 3 A.M.

Takeaways

My time at this organization was foundational in shaping how I approach data challenges today. It taught me the value of iterative improvement, the importance of building sustainable solutions, and the patience required to wrangle legacy systems that are older than many of the people who may read this.

I learned that automation isn't just about saving time — it's about reliability and freeing humans from tedious, error-prone work. I discovered that the best solutions are those built with both technical rigor and real-world pragmatism.

And, of course, I learned how to gracefully train my replacement—after a private equity firm swooped in, consolidated operations, and wipe out nearly 500 jobs at one of the largest employers in our community. Nothing like a crash course in corporate survival to round out your education.

Ultimately, this journey was less about systems and more about people: understanding their needs, building trust, and delivering impactful data solutions that make their work better.