The Story: Why I Built My Own Data Generator

For this project to be meaningful, I needed to solve a problem that felt real, not just analyze another generic dataset. I was not interested in analyzing another generic open-source dataset, because those do not reflect the realities of my industry or the decisions I face at work. Using real company data was not possible, for obvious confidentiality reasons.

I looked for alternatives. Open datasets from the industry were either too simple or lacked the structure of real dealership sales. None matched what I needed. My goal was clear: create a dataset that behaves like real dealership invoices, without risking anyone's privacy or exposing sensitive information.

Before settling on writing my own Python script, I spent several hours trying to get both Gemini and ChatGPT to generate a realistic CSV dataset for me. Neither tool could create the kind of large, structured, and business-specific data I needed. The results were either too perfect to KPI models with no variance, or they failed completely and the file was garbage or "File not Found". After repeated attempts and a lot of wasted time, I realized that if I wanted data I could trust, I would have to build it myself. That decision put me in control of the details: every column, every rule, every distribution.

So, I built my own from scratch. I started by collecting industry KPIs for things like labor margin, parts margin, job mix, and technician efficiency. These numbers gave me realistic targets. I wanted the data to look and feel authentic.

The first hurdle was creating believable customer names and invoice numbers. I used the Faker library to generate hundreds of unique customers and realistic IDs. This gave my dataset credibility right away.

Next, I focused on the numbers. A real dealership does not use random figures. Margins and sales have expected ranges and patterns. I used NumPy to create normal distributions for every key variable, making sure most values landed close to industry targets but with enough variation to include some outliers. These outliers were intentional, since real business always produces both predictable and unexpected results.

Business logic made the data work. I encoded actual rules: counter sales do not have labor hours, trailer jobs follow different cost patterns, and some department-location combinations are more frequent than others. If a rule applied in real operations, I reflected it in the script.

To handle thousands of records efficiently, I used pandas for everything from managing data to exporting it as a CSV file. With this approach, I could produce over 45,000 invoices in seconds, fully structured and ready for analysis.

Looking back, writing this generator forced me to break down the business into its core behaviors. I had to decide how often each job type appeared, how much sales could vary, and what was plausible for technician performance. The finished product is not just clean data. It is a realistic simulation of how the business operates, but with all identifying details removed.

There are limits. My generator only covers the basics. It does not include seasonality, month-end trends, technician skill growth, or the ups and downs of the business cycle. These factors are real in day-to-day operations, but for this project, the generated data is close enough to support modeling and testing. If I ever need to, I can swap in real data without changing the process.

Now I have a flexible, safe, and realistic dataset for any analysis or model I want to build. Ultimately, building this generator was more than just a preliminary step, it was the foundation of the entire analysis. Ensuring that the insights from my model would be grounded in a reality I understood and had built myself.