Databender

FREE GUIDE

The 90-Day Data Roadmap

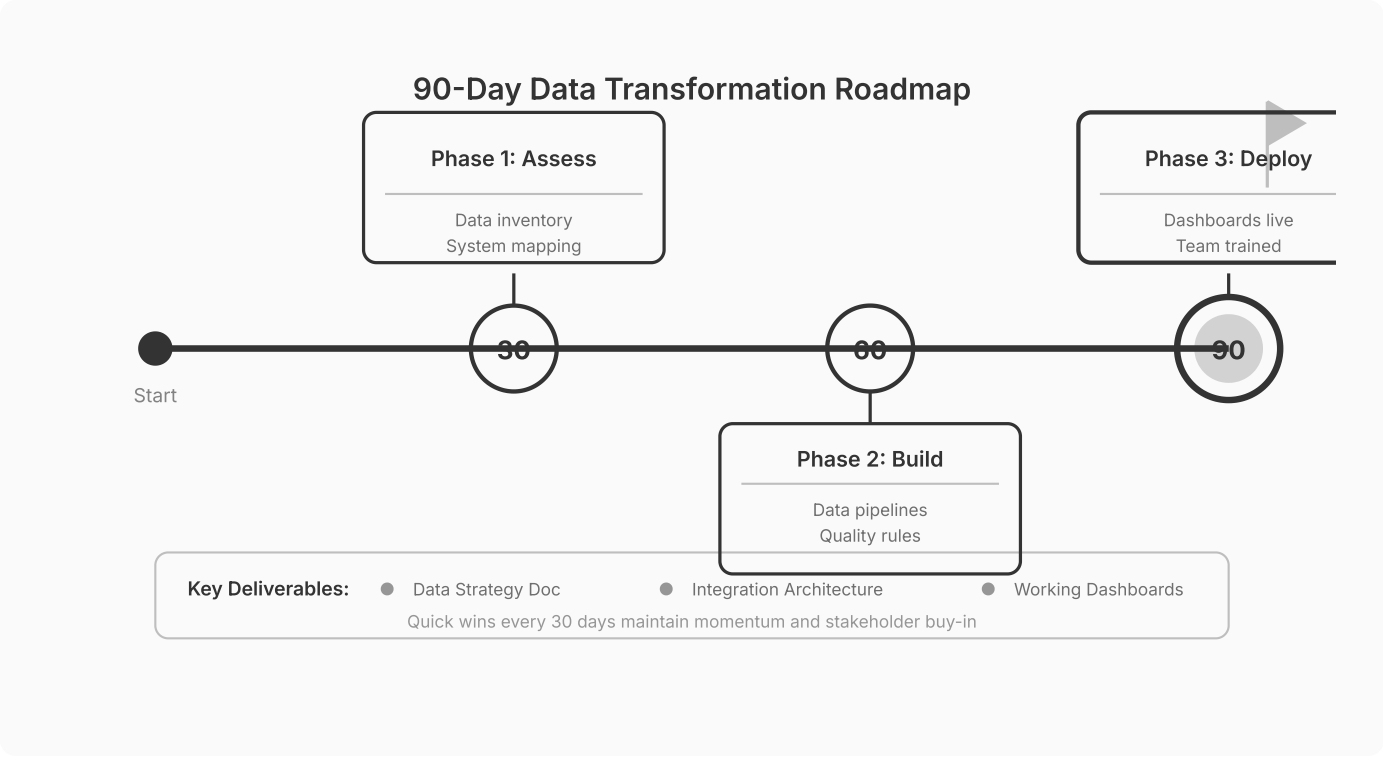
From Spreadsheets to Unified Visibility in One Quarter

databender.co | Data & AI Consulting for Growing Businesses

90 Days

To Visibility

Transform from scattered spreadsheets to unified visibility in one quarter.



You've been promised "data transformation" before. The pitch deck showed beautiful reports. The timeline said 18 months. The budget said $500,000. And somewhere around month 14, the project died quietly while everyone pretended it was still on track.

Here's the uncomfortable truth about why those projects fail: they try to do too much. They aim for perfect data infrastructure before delivering any value. They treat "data strategy" as a destination instead of a series of useful stops along the way.

The alternative isn't lowering your ambitions. It's sequencing them differently.

Ninety days is enough time to build something real. Not a complete data platform. Not a full analytics stack. But a working application that solves a specific problem and proves the approach works. Once you have that first win, the path forward becomes much clearer.

Why 90 Days Works

Ninety days is short enough to maintain momentum and long enough to build something substantial.

Shorter timelines (30 days, "quick wins") usually produce demos instead of working systems. The integration work gets skipped. The edge cases get ignored. You end up with something impressive in a meeting that breaks under real use.

Longer timelines (12+ months, "transformation programs") lose focus. Requirements change. Champions leave. By the time the system is ready, the business has moved on. And nobody remembers what the original goal was anyway.

Ninety days forces discipline. You can't boil the ocean in 90 days. You have to pick one specific problem and solve it completely. That constraint is what makes the approach work.

A building products manufacturer we worked with had been talking about "better reporting" for three years. Every proposal they'd seen involved rebuilding their data infrastructure from scratch. Eighteen months minimum. Budget in the hundreds of thousands.

Instead, we spent the first two weeks understanding what they actually needed. The CEO wanted one thing: to see daily shipments, orders in backlog, and cash position each morning before the 8 AM staff meeting. Everything else was nice-to-have.

We built exactly that. One screen, three numbers, updated automatically. Day 47. No infrastructure overhaul. No massive budget. Just the view the CEO needed, pulling from the systems they already had.

That single screen changed the company more than any 18-month project would have. Because it actually shipped.

Days 1-30: First Application Live

The first month is about proving that progress is possible.

Week 1: Problem definition. Not "we need better data." That's too vague. What specific question does someone need answered that they can't get today? Who's asking it? How often? What do they do with the answer?

The best first projects share common characteristics:

- Someone asks the same question at least weekly
- Getting the answer currently requires manual work or phone calls
- The data already exists (just in separate systems)
- The stakeholder has authority to actually use the answer

Don't pick the hardest problem. Pick the clearest one. First wins need to be unambiguous.

Week 2: System mapping. Where does the data live? What format is it in? How do we get it out? This week is technical discovery. You're figuring out what you're working with before you start building.

Most mid-sized manufacturers run five to fifteen core systems: ERP, CRM, shipping platform, production software, maybe a quality system. Each one has its own database, its own API (or lack thereof), its own quirks. Understanding the landscape prevents surprises later.

Weeks 3-4: Build and deploy. Connect to the source systems. Pull the data. Build the view. Get it in front of the person who asked for it.

Don't over-engineer. Don't build for scale. Don't add features nobody asked for. Build the minimum thing that answers the original question. Polish can come later.

At the end of Day 30, someone should be using something they weren't using on Day 0. That's the only metric that matters.

Days 31-60: Cross-System Visibility

The first application proved you could do it. The second month expands the scope.

Week 5: Integration layer. The first application probably connected to one or two systems. Now you build the infrastructure that makes connecting additional systems easier. This is where you establish the patterns you'll reuse going forward.

Integration tools like Workato, Celigo, or Boomi handle most of the heavy lifting. You're not writing custom code for every connection. You're configuring a platform that does the connecting for you.

Week 6: Data normalization. Different systems call the same thing by different names. Customer numbers don't match between CRM and ERP. Product codes have variations. Date formats conflict.

This week is about translation. Building the logic that reconciles different systems' views of the same reality. The work isn't glamorous, but it's what makes cross-system views actually work.

Weeks 7-8: Second and third applications. With the integration infrastructure in place, adding new views gets faster. The CEO screen that took four weeks in month one? A similar screen for the CFO might take one week in month two.

At the end of Day 60, multiple people should be using multiple views that pull from multiple systems. The data is starting to flow.

Days 61-90: Proactive Alerts and Automation

Applications require someone to look at them. Month three adds intelligence.

Week 9: Alert design. What conditions matter? When should someone be notified? Who should be notified? The answers depend on how your business works.

Common manufacturing alerts worth building:

- Order is late relative to promised ship date
- Inventory for a key component drops below threshold
- Customer payment is overdue from a normally prompt payer
- Production job is tracking behind schedule
- Quality issue detected on a product

The danger is alert overload. Start conservative. It's easier to add alerts than to convince people to trust them again after they've started ignoring notifications.

Weeks 10-11: Automation pilots. Some actions don't require human judgment. When a shipment goes out, update the customer record. When a payment comes in, mark the invoice closed. When a supplier confirms delivery, update the expected receipt date.

These automations aren't complex. They're just making systems do automatically what someone used to do manually. The labor savings add up.

Week 12: Documentation and handoff. What did you build? How does it work? What happens when something breaks? The last week is about making sure the system survives contact with reality after the project team moves on.

At the end of Day 90, you have working applications, active alerts, running automations, and documentation. Not a complete data platform. But a foundation that can expand.

What to Tackle First

Choosing the right first project determines whether the 90 days succeed or fail.

Good first projects:

- Order status visibility for customer service
- Daily shipment and backlog summary for leadership
- AR aging view that combines ERP data with CRM context
- Production schedule that shows promised dates against capacity
- Inventory alerts for critical components

Bad first projects:

- Complete historical analytics (too much data cleanup required)
- Predictive models (need clean data first)
- Customer self-service portals (require polish that slows initial delivery)
- Anything requiring data that doesn't exist yet
- Anything involving more than three source systems initially

The distinction is scope. Good first projects answer specific questions with existing data. Bad first projects require building too much infrastructure before showing value.

One precision machining company wanted to start with predictive maintenance. Interesting problem. But the data wasn't there. Machine sensors weren't connected. Historical maintenance records were incomplete. The "first project" would have required six months just to get the foundation in place.

Instead, they started with shift handoff visibility. The data existed in paper logs and emails. Digitizing it was straightforward. The value was immediate. And now they have the infrastructure to tackle predictive maintenance when they're ready.

What Can Wait

The hardest part of the 90-day approach is deciding what not to do.

Data warehousing can wait. A centralized data warehouse is valuable eventually. But building one takes months and delivers no value until it's complete. Start with direct integrations. Add the warehouse later when you need historical analysis across multiple domains.

AI and machine learning can wait. Everyone wants to talk about AI. But AI needs clean, connected data to work with. The 90-day foundation makes AI possible later. Starting with AI before the foundation exists just creates expensive failures.

Self-service analytics can wait. Giving everyone the ability to build their own reports sounds democratic. In practice, it creates chaos. Start with specific applications for specific people. Add self-service when you understand what reports people actually need.

Mobile apps can wait. Mobile is convenient but adds complexity. Start with web interfaces that work on any device. Build native mobile when you've proven the use case.

The pattern: build the specific before the general. Solve the concrete before the abstract. Get something working before making it perfect.

What This Costs

Ninety-day programs are smaller than multi-year transformations. But they're not free.

A typical 90-day program for a mid-sized manufacturer runs $30,000 to $60,000, depending on complexity. That covers system assessment, integration development, application building, and knowledge transfer.

Ongoing costs after the 90 days are minimal. Integration platforms charge monthly fees (usually $500 to $2,000 depending on volume). Application hosting costs are trivial. The infrastructure you build can run for years without significant additional investment.

Compare that to traditional approaches. A full data transformation program from a large consulting firm typically starts at $250,000 and runs for 18+ months. Enterprise software implementations routinely exceed $500,000. And those projects often fail to deliver usable results.

The 90-day approach costs less because it delivers less scope. But it delivers actual working systems instead of plans and promises.

What Happens After Day 90

Day 90 isn't the end. It's the foundation.

You have working integrations between core systems. You have applications that people actually use. You have a pattern for building more. The next project takes less time than the first because the infrastructure exists.

Typical expansion paths after the initial 90 days:

Add more views. The CEO screen was first. Now build the CFO screen, the operations manager screen, the sales director screen. Each one takes days instead of weeks because the data connections exist.

Extend integrations. Started with ERP and shipping? Add the CRM. Add the production system. Add the quality database. Each new connection expands what's possible.

Add intelligence. Once data flows and is clean, analytics become feasible. Which customers are trending down? Which products have margin erosion? Which suppliers are causing delays? The questions that couldn't be answered before become answerable.

Automate more. The automations from month three were pilots. Now scale them. Every manual task that can be automated frees someone to do work that actually requires human judgment.

The 90-day program is a start, not a finish. But it's a start that produces value immediately instead of promising value eventually.

Why This Works When Big Projects Don't

Big data projects fail for predictable reasons.

They take too long. Business needs change. Champions move on. By the time the system is ready, it solves yesterday's problem.

They try to do too much. Scope creep is inevitable when timelines stretch. Every stakeholder adds requirements. The project becomes an attempt to solve every data problem at once, which means solving none of them well.

They prioritize infrastructure over value. Perfect data models, elegant architectures, enterprise-grade security. All good things. But none of them help anyone until applications sit on top of them. Projects that build infrastructure first often never get to the applications at all.

The 90-day approach inverts these failure modes. It's too short for scope creep. It's too focused for stakeholder drift. And it forces value delivery before infrastructure can consume all the budget.

Perfection is the enemy of progress. A working application that answers 80% of the question beats a planned system that would answer 100% of the question if it were ever built.

Ready to build something real in 90 days? Talk to us about what your first application should be, or explore our full manufacturing solutions.