

How Enterprises Are Using **AI Agents** in **2026**



74% of organisations plan to use AI agents by 2027,
while only 21% can govern them.

Table Of Contents

1. Executive Summary
2. What Makes an Agent an Agent
3. Where Agents Belong, and Where They Don't
4. The Deploy–Reshape–Invent Curve
5. Fundamentals of an Agentic Enterprise
6. The Pattern of Enterprise Agents' Use Cases
7. The Governance Gap
8. Introducing Agent Debt
9. What Scalable Deployments Have in Common
10. Strategic Imperatives for Optimising Agentic AI Use Cases
11. Conclusion
12. References

Executive Summary

From The Modern Data 101 Community

This report examines the architectural, contextual, and governance foundations required to scale agents without letting agent debt compound, compiling insights from industry leaders.

Our reading of the last two years of enterprise agent deployments is that the industry has been asking the wrong central question. Vendor reports continue to focus on whether models can reason, plan, and call tools. Those capabilities matter, but they are only one part of a working agent. An agent is only as reliable as the system surrounding the model: the context it receives, the tools it can invoke, the permissions that constrain it, the checks that verify its actions, and the records that make those actions traceable. Recent Modern Data 101 research makes the same distinction from the data-platform side: the model is a component; the runtime around it determines whether that capability becomes a reliable enterprise outcome.

The evidence in this brief points to a different, under-discussed failure mode: enterprises are accumulating what we call agent debt. The compounding gap between how many autonomous decisions a system is authorised to make and how few of those decisions anyone can currently trace, own, or reverse. Agent debt behaves like technical debt, except the interest is paid in incidents rather than slower deploys, and it accrues fastest precisely where deployments look most successful.

The scale of deployment is no longer in question. IBM deployed agentic AI across 270,000 employees for an estimated \$4.5 billion productivity impact, and the market is projected to grow at a 45% CAGR over the next five years. What's missing from most that growth narrative is a serious account of who is accountable when an autonomous decision goes wrong at that scale.

That gap shows up directly in the data: a 2026 Deloitte survey found 74% of organisations expect to be using AI agents at least moderately by 2027, while only 21% currently have a mature governance model in place for them. We treat that 53-point spread as the central fact of this brief, not a footnote to it; Sections 5 through 8 are organised around quantifying that gap and proposing a fix for it.

What Makes an Agent an Agent

BCG defines an AI agent as an AI system that uses tools to accomplish goals, able to remember across tasks, decide when to access internal or external systems, and act autonomously with minimal human oversight. That definition is doing real work: a chatbot answers when asked, while an agent decides what needs to happen next and does it.

The components of AI Agents

1. Model / reasoning engine

The LLM (or other model) at the core that interprets instructions, reasons about tasks, and decides what to do next. This is the “brain” of the agent.

2. Instructions / system prompt

The persona, goals, constraints, and rules that shape how the agent behaves, such as what it should and shouldn’t do, its tone, its scope of authority.

3. Memory

- Short-term/working memory: context from the current conversation or task (often the context window itself).
- Long-term memory: persisted knowledge across sessions, like user preferences, past interactions, learned facts, usually stored in a database or vector store.

4. Tools / actions

Functions the agent can call to interact with the world beyond text generation: APIs, databases, search, code execution, file systems, or other services. This is what turns a chatbot into an “agent” the ability to do things beyond simple responding.

5. Planning / orchestration logic

The mechanism that breaks a goal into steps, decides which tool to use when, sequences multi-step tasks, and handles retries or branching logic. This can be built into the model’s own reasoning (chain-of-thought, ReAct-style loops) or handled by an external orchestrator/framework.

The components of AI Agents

6. Retrieval (often via RAG)

A way to pull in relevant external or proprietary knowledge, such as documents, embeddings from a vector database, so the agent isn't limited to what's in its training data or context window.

7. Guardrails / evaluation

Safety and quality checks: input/output validation, policy enforcement (e.g., PII redaction), hallucination checks, and evaluation metrics that monitor whether the agent is behaving as intended.

8. Environment / interface

Where the agent operates, a chat UI, a browser, an IDE, a Slack channel, an API endpoint, and how it receives inputs and delivers outputs.

9. Observability / tracing

Logging of the agent's reasoning steps, tool calls, and outcomes, so behaviour can be debugged, audited, or improved over time (this is what tools like Arize, Galileo, and LangSmith focus on).

The model is the reasoning engine, but it is not the agent by itself. A production agent depends on the system around that model: instructions, relevant context, bounded tools, permissions, feedback, verification and observability.

This can be framed as agent = model + harness. The same model can produce materially different outcomes when the surrounding system changes how context is supplied, which tools are available, what actions are permitted and how outputs are verified.

This distinction matters because enterprise reliability is increasingly determined outside the model. A capable model operating over poor context, broad permissions and weak verification can still produce an unreliable system.

BCG breaks a working agent into five parts: agent-centric interfaces (the protocols connecting it to users, databases, and sensors), a memory module (short-term context plus long-term knowledge of past tasks), a profile module (its role, goals, and behavioral patterns), a planning module (usually an LLM, assembling the plan), and an action module (the APIs and integrations it can actually use).

The observe–plan–act loop

Those five parts run in a cycle: agents observe their environment and retain memory across steps, plan by weighing goals and context through an LLM, then act through connected systems, detecting errors, fixing them, and asking for clarification when needed. The loop is self-reinforcing: each cycle updates what the agent knows about how the world responded last time, which is what separates it from a fixed automation script that behaves identically on run 1,000 as it did on run 1

A maturity spectrum, and not just a single capability

“Agent” also isn’t one fixed thing, but spans a spectrum of autonomy. [BCG illustrates this with coding assistants](#): the simplest level generates code only when a developer prompts it; a more advanced agent ingests the existing codebase and customises its output unprompted once a test is written; a further step compiles and runs the application in a test environment; and the frontier, which is not yet standard deploys tested applications to production via an automated pipeline once a human approves it.

Most enterprise deployments today sit at the first two levels. The strategic question for any function isn’t “should we use an agent,” but “which level of that spectrum does this task actually need” since jumping straight to full autonomy on a task that only needed a copilot is how governance gaps open up.

The multiplayer shift

Dust frames the organisational version of the same idea: the highest-value deployments aren’t one agent per employee, but a shared workspace where humans and agents draw on the same company knowledge, tools, and goals, so gains compound across the organisation instead of staying siloed with whoever built the agent. An agent built by one analyst and never reused is a personal productivity trick; the same agent exposed to the whole team, with shared memory of what worked, is infrastructure. Most of the ROI evidence in Section 6 comes from organisations that made that second choice.

Where Agents Belong, and Where They Don't

The instinct to deploy agents everywhere is the fastest way to waste the technology. Appian's framing is the sharpest available on this point: the question isn't where agents can be used, but where they should be, and the two lists don't overlap as much as vendors suggest.

Good fits for agents

Appian identifies the pattern directly: messy or unstructured inputs, context spread across systems and teams, a decision that requires reasoning rather than a rule lookup, and a need for an auditable record of what was done. Traditional workflow automation, RPA, and rules engines remain the better tool whenever inputs are structured, and the same decision should be made the same way every time.

Poor fits for agents

The same logic runs in reverse. Agents are a poor match where the error budget is near zero, where a wrong action is costly, or where compliance exposure means a human must own the call. The strongest use cases, per Appian, sit between “too trivial to need judgment” and “too consequential to delegate” a narrower band than most deployment roadmaps assume.

Why the fit test works: agents inherit human cognition

There's a mechanical reason Appian's test lines up with reality rather than being just a rule of thumb.

A task with all four of those properties is exactly the “messy inputs, clear reasoning step, auditable outcome” profile Appian describes, the two frameworks are describing the same underlying constraint from different directions, one architectural and one cognitive.

BCG's research notes that the LLMs powering agent planning are trained on a large body of human output, so they effectively inherit human cognition, meaning they solve problems the way people do, well when a task is broken into component parts, with small well-defined steps, relevant context, and a tight feedback loop that lets errors get corrected on the next iteration.

The Deploy–Reshape–Invent Curve

BCG’s Deploy–Reshape–Invent framework describes how far an organisation has gotten with agents overall, and it maps onto a familiar shape from the rest of this series.

Deploy

Getting employees using agentic tools inside their everyday work as fast as possible, the critical first step toward realising any value from GenAI, and the stage nearly every enterprise in this brief’s dataset has reached.

Reshape

Reimagining entire functions end-to-end rather than automating isolated tasks within them, where the cost savings and ROI numbers in Section 6 actually come from, because piecemeal deployment inside an unchanged process caps the upside.

Invent

Using agents to find growth and business-model opportunities that wouldn’t have surfaced otherwise, the stage BCG’s client work describes as reinventing the consumer experience itself, not just cutting its cost, and the stage almost no organisation in this brief’s evidence base has reached at scale yet.

The shift matters because the risks change with it. An isolated agent creates an isolated failure. An agent embedded in a business process creates downstream dependencies. An agent whose output becomes another system’s input becomes part of the organisation’s architecture.

Trailing back to ‘AI Adoption vs. AI Maturity’

Deploy–Reshape–Invent is a three-stage compression of the same climb Modern Data 101’s report brief mapped as a four-level maturity pyramid: individual use, then process redesign, then business-model change. Two research groups, working from unrelated data, keep independently re-deriving the same staircase.

As agents proliferate, complex disciplines that once required large teams, software development, customer service, business analytics, become smaller teams of humans working alongside many agents, and supervising agents becomes a core teaming skill in its own right.

The transition between these stages also changes the role of the agent itself. At the Deploy stage, an agent can remain an individual productivity tool. At the Reshape stage, it becomes part of a shared workflow. At the Invent stage, agents increasingly become infrastructure that multiple teams, systems and other agents depend on.

Fundamentals of an Agentic Enterprise

If we leave aside vendor branding and enterprise agent deployments cluster around a handful of functions, each with a slightly different reason agents fit. What follows is not an exhaustive catalog, it's the six functions where the evidence for real, measured impact is strongest.

Customer service

The most mature deployment category. A global bank cut customer service costs 10x using AI virtual agents, while an IBM-built tool resolved 70% of inquiries and cut resolution time by 26%. IBM's own research projects that 71% of executives will aim for fully touchless support by 2027.

IT operations & software development

Agents shift IT from reactive troubleshooting to predictive maintenance, and from writing code to reviewing and testing it. One IT department used agents to modernise legacy technology and lifted productivity by up to 40%, while IBM's internal AI-driven cost analysis drove \$600 million in IT savings since 2022.

IBM's internal AI-driven cost analysis drove \$600 million in IT savings since 2022.

Finance & procurement

Invoice matching, reconciliation, and anomaly detection are natural fits because the inputs are structured, but the exceptions require judgment. Dun & Bradstreet cut procurement task time by up to 20% using an AI tool for risk evaluation and vendor selection.

Human resources

IBM's internal AskHR tool now automates more than 80 HR tasks and handles over 2.1 million employee conversations annually, covering onboarding, time-off requests, and policy questions, the highest-volume, most standardised HR interactions.

Supply chain & manufacturing

IBM Consulting applied agentic AI across a supply chain spanning over 2,000 suppliers in more than 170 countries, generating over \$360 million in supplier savings over three years. A separate industrial goods company embedded agentic AI directly into daily workflows and unlocked a 2 percentage-point EBITDA boost in two years.

Legal & compliance

European insurer Wakam deployed 130 agents across departments, including dedicated contract-analysis and regulatory-monitoring agents; the legal team cut contract analysis time by 50% and reduced document search time by 40–60%.

6

The Pattern of Enterprise Agents' Use Cases

A pattern emerges that none of the individual vendor write-ups states outright: the largest gains cluster in work that is high-volume, well-bounded, and currently done by people reading and re-keying information across systems. Blog-post generation cut a consumer goods company's costs by 95% and sped output 50x; biopharma lead generation cut cycle time 25% and lifted drafting efficiency 35%. Both are pattern-matching-and-drafting tasks with a clear right answer to check against.

The weaker results, by contrast, cluster where the task requires negotiation, emotional read, or ambiguous authority. The lesson compounds across every source in this brief: agent ROI is not a function of how advanced the underlying model is. It's a function of how cleanly the task's success criteria can be defined in advance, a data-and-process question, not a model-capability question.

What Makes the Best Data Platform for AI in 2026 report reached from the platform side: agents don't create clean task boundaries and governed data access on their own, they inherit whatever architecture already exists. An enterprise with fragmented systems and undocumented processes will get agents that amplify that fragmentation faster than it can be cleaned up.

There is a second architectural constraint: the quality of the context available to the agent.

A well-defined task can still produce poor results if the agent has to infer business definitions from raw schemas, retrieve irrelevant information, or work from stale data. Modern Data 101's work on data products and agent runtimes points to the same conclusion from two directions: context needs to be relevant and governed, while the controls surrounding that context need to participate in execution rather than merely exist as documentation.

The practical equation is therefore not simply capable model + good use case. It is capable model + well-defined task + relevant context + bounded execution.



The Governance Gap

Deloitte's 2026 survey of 3,235 business and IT leaders is unambiguous about where the risk actually sits: roughly 80% of organisations lack mature governance for agentic AI, no clear boundary for which decisions an agent can make independently, no real-time monitoring for anomalous behaviour, and no complete audit trail of agent actions. This is not a future risk. It is the current operating condition of most agent deployments running in production today.

What mature governance actually covers

A policy documented in a governance system, a lineage record sitting in a catalog, or a quality alert sent after an execution may all be useful evidence.

None necessarily prevents the action that created the problem.

The more important distinction is between controls that sit beside the agent and controls that participate in the agent's execution path. Modern Data 101's recent work makes this distinction explicit: enterprise platforms already contain many of the primitives agents need, including semantic definitions, lineage, access controls, quality signals, and observability. The architectural question is whether those controls are actually wired into the runtime.

For an agent, governance increasingly has to operate at the moment of action: before a query executes, before sensitive data is returned, before an external system is modified, and before an autonomous decision crosses a defined risk boundary.

Agent sprawl is the leading indicator

As deployments grow from a handful of agents to dozens or hundreds, organisations lose visibility into who built what and which agents are actively used, and duplicate or conflicting agents proliferate across teams. Wakam's 130-agent rollout avoided this only because governance and access controls were built in before scale, not after.

Modern Data 101's report documented the same shape at the organisational level: adoption consistently outruns the maturity needed to govern it. Deloitte's 21% figure is that same gap, this time measured directly for agentic AI.

Runtime governance

For an agent, governance increasingly has to operate at the moment of action: before a query executes, before sensitive data is returned, before an external system is modified, and before an autonomous decision crosses a defined risk boundary.

Dust's enterprise framework names three areas that determine whether agents can be deployed safely at all: data protection (encryption and ensuring model providers never train on enterprise data), access control (agents inheriting only the permissions of the humans they act for), and compliance frameworks such as GDPR, HIPAA, and SOC 2.

Introducing Agent Debt

Every source in this brief describes symptoms of the same underlying condition without naming it: agents that nobody can trace, sprawl nobody planned for, decisions nobody explicitly authorised. We think it's worth naming directly, because it doesn't behave like a risk you mitigate once, it behaves like a debt you either service continuously or watch compound. We call it agent debt: the gap between the number of autonomous decisions a system is authorised to make and the number anyone can currently trace, own, or reverse.

Like technical debt, it's invisible on the balance sheet until the day it isn't, except the interest here is paid in incidents, and not slower sprints.

Accountability debt

No defined owner for a given class of autonomous decision, and no defined escalation path when the agent is uncertain. This is what Deloitte's governance question is actually measuring, not whether an agent works, but whether a human can say who was responsible when it didn't.

Context debt

Context debt is the gap between the context an agent needs to make a reliable decision and the context the architecture can reliably provide. It accumulates when definitions are fragmented, lineage is incomplete, memory is disconnected, or agents must reconstruct business meaning from raw data.

Coordination debt

This is the component the industry underestimates most, because it doesn't grow at the same rate the agent count does. Every new agent can potentially interact with every agent already deployed, so the number of possible agent-to-agent interaction paths grows roughly with the square of the agent count, not in a straight line with it. Wakam's 130-agent rollout, cited in Section 5, implies over 8,000 possible pairwise interaction paths, most never exercised, but every one a place a conflict, a duplicated action, or a race condition can hide.

The problem becomes more pronounced when agents that appear independent actually share data, tools, models, or downstream dependencies. [Modern Data 101's recent work on the independence assumption](#) highlights why agreement between multiple agents should not automatically be treated as independent confirmation.

Traceability debt

This refers to the gap between the number of derived outputs an enterprise acts upon and its ability to trace those outputs back through the dependencies that produced them.

Traditional lineage often stops at tables. AI extends the chain: data produces features, features feed models, models produce predictions, and agents turn those predictions into actions. [Modern Data 101's recent work on the Unified Dependency Graph](#) argues that these layers need to become part of one traceable dependency structure.

9

What Scalable Deployments Have in Common

Across every case in this brief, the organisations that moved agents from pilot to production, rather than stalling, which [Dust identifies as the primary challenge even once agents work technically](#), share three traits that have nothing to do with which model or vendor they chose.

Process, not tool

[Appian positions agents as participants inside long-running workflows, not standalone tools: the process defines when an agent acts, what it can do, and when it must escalate to a human](#). Wakam's rollout followed the identical logic, starting with one department (legal) and one workflow before expanding, rather than deploying broadly on day one.

Organisational change, not IT project

[IBM's own implementation guidance is blunt: organizations that treat agent deployment as a technical project struggle, while those that treat it as a business transformation initiative thrive](#). Wakam drove adoption through regular hackathons where teams built specialized agents collaboratively, a change-management tactic, not a technology rollout tactic.

The third trait is measurement discipline. [Wowinfotech's ROI framework](#) reduces the question to one line, value generated minus the total cost of running the agent, including model, integration, and monitoring costs, and every scaled deployment in this brief could answer that question with a specific number. The stalled pilots, almost without exception, could not.

Strategic Imperatives for Optimising Agentic AI Use Cases

Seven moves that separate agentic infrastructure from expensive pilots:

1. Apply the fit test before the build. Messy inputs, distributed context, and a judgment call beat a rule lookup every time. If the decision is fully specifiable in advance, use automation, and not an agent.
2. Start where the ROI case is already proven. Customer service, IT operations, and document-heavy finance and legal work have the deepest evidence base; build organisational muscle there first.
3. Build governance into the execution path while deployments are small; retrofitting controls onto hundreds of autonomous workflows is harder than making those controls part of the architecture from the beginning.
4. Agents inherit the semantics, freshness, lineage and access boundaries of the data architecture around them. Build these as machine-readable, executable controls rather than documentation an agent is expected to interpret.
5. Fund the change program, not just the model. Hackathons, department-by-department rollout, and clear handoff protocols between agents and humans matter as much as model selection.
6. Measure value minus total cost of operation from day one. If a deployment can't produce that number, it isn't ready to scale, regardless of how well the demo performs.
7. Cap the interaction surface, but also make it traceable. As agents become inputs to other agents, workflows and models, organisations need to know not only how many agents exist, but what each agent depends on and what depends on it. A unified dependency graph makes those relationships queryable before an incident rather than reconstructable after one.

11

Conclusion

AI agents have cleared the hardest technical bar. They work, and the evidence for operational value across customer service, IT, finance, HR, supply chain and legal is now extensive.

The harder problem is the system around that autonomy.

Agents need context they can trust, tools they cannot misuse, permissions they cannot exceed, decisions that can be verified, and dependencies that can be traced. As agents move from isolated assistants into shared workflows, the cost of missing any one of those foundations compounds.

That is what makes agent debt different from ordinary technical debt. Technical debt makes systems harder to change. Agent debt makes autonomous systems harder to understand, govern and reverse.

The architecture required to address it is not entirely new. Data products can provide bounded enterprise context. Semantic layers can provide shared meaning. Governance can move into the runtime. Lineage can extend into models and AI outputs. Dependency graphs can connect the entire chain.

The organisations that scale agents sustainably will therefore not simply be the ones that deploy more capable models. They will be the ones that make autonomy observable, bounded, contextual and traceable.

Agent capability may continue to commoditise. The infrastructure that makes autonomy trustworthy might not.

References

[1] AI Agents: What They Are and Their Business Impact. Boston Consulting Group, 2026.

<https://www.bcg.com/capabilities/artificial-intelligence/ai-agents>

[2] Enterprise AI Agents: Beyond Productivity. IBM Think, November 2025. <https://www.ibm.com/think/insights/enterprise-ai-agents>

[3] AI Agent Use Cases: Core Enterprise Agentic Workflow Patterns. Appian, 2026. <https://appian.com/learn/topics/enterprise-ai/ai-agent-use-cases>

[4] AI Agents for Enterprise: What They Do and How Companies Deploy Them. Dust, March 2026. <https://dust.tt/blog/ai-agents-for-enterprise>

[5] 11 Real-World AI Agent Examples & Use Cases for Enterprises. Wizr AI, August 2026. <https://wizr.ai/blog/ai-agent-examples-enterprises-use-cases/>

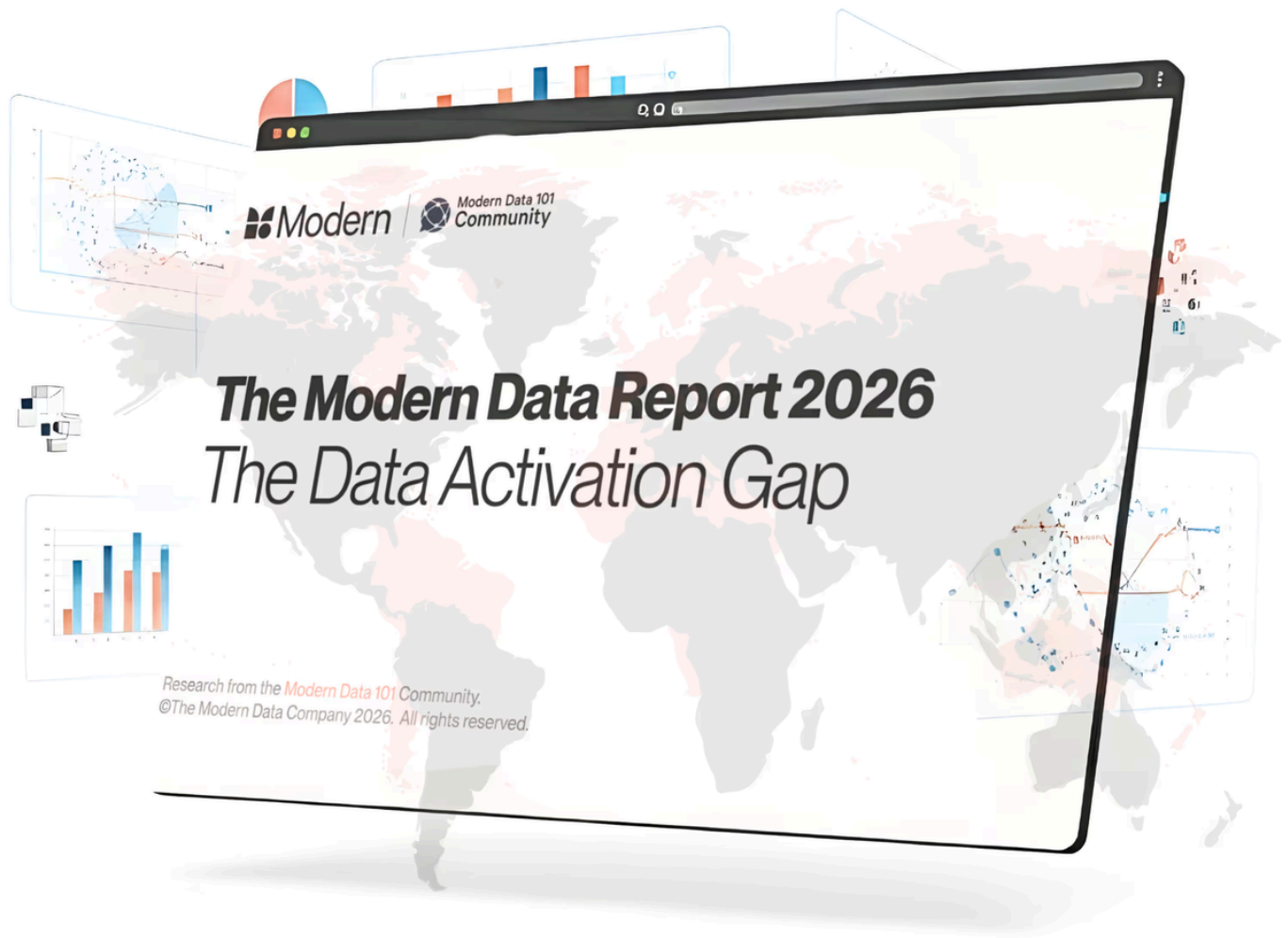
[6] Enterprise AI Agents: Practical Use Cases Driving Business Automation. Markup Designs, September 2026. <https://www.markupdesigns.com/blog/enterprise-ai-agents-use-cases/>

[7] How AI Agents Work for Enterprises to Automate Business Operations. WOWinfotech, August 2026. <https://www.wowinfotech.com/blog/how-ai-agents-work-enterprise-business-operations>

[8] Business and IT Leaders Report AI Agents Are Scaling Faster Than Their Guardrails. Deloitte Insights, survey of 3,235 leaders across 24 countries, April 2026. <https://www.deloitte.com/us/en/insights/topics/emerging-technologies/ai-agents-scaling-faster.html>

[9] AI Adoption vs. AI Maturity in 2026: Navigating the Enterprise Capability Gap. Why AI Adoption and AI Maturity Are Two Different Curves. Modern Data 101 Community Whitepaper, Vol. 02, August 2026.

[10] What Makes the Best Data Platform for AI in 2026? New Research Has an Answer. Modern Data 101 Community Whitepaper, Vol. 01, August 2026.



540+ Respondents, 64 Countries, 29 Industries

[Access the report now](#)



Modern Data 101 Community



Subscribe 
moderndata101.com