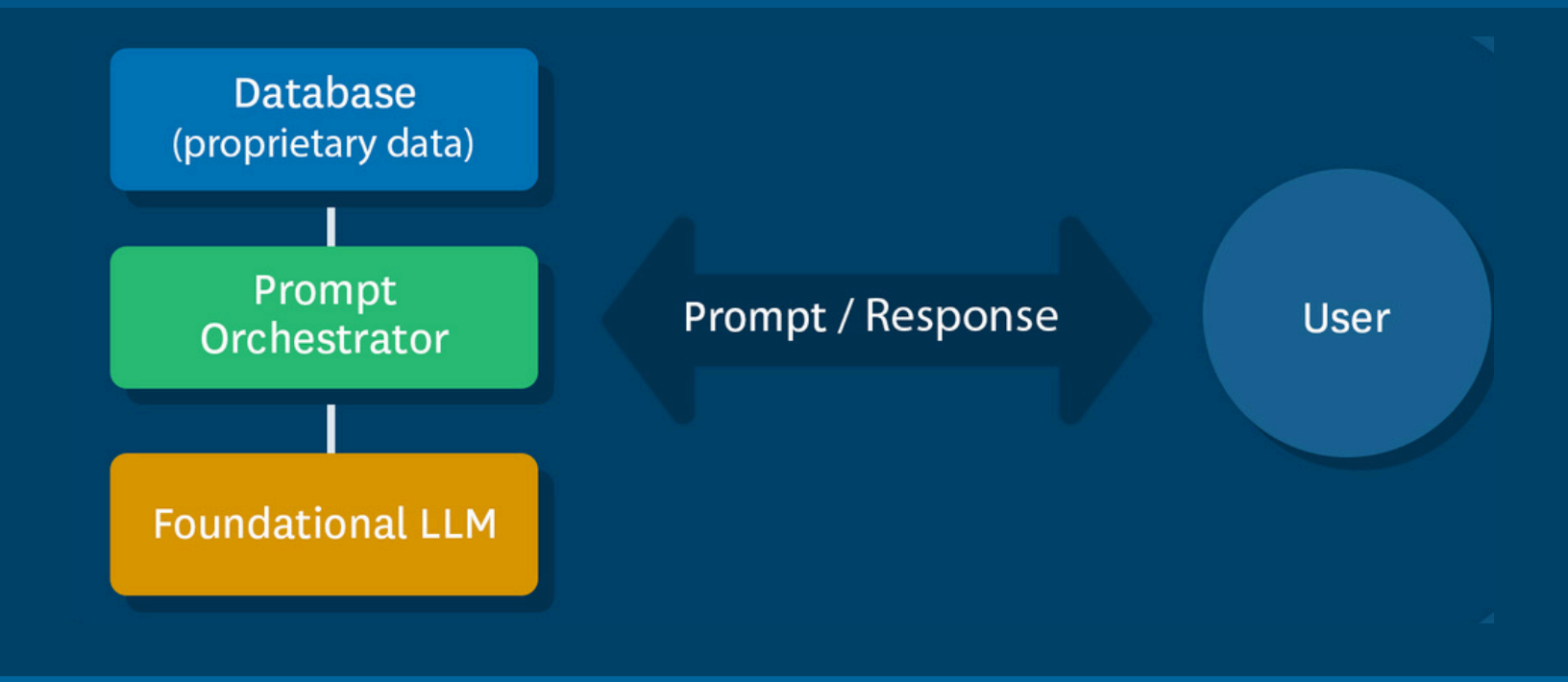
MONTE CARLO

Understanding the Basics: RAG for AI



Everything you need to know about retrieval augmented generation, why it matters, where to start, and what you need to do now to build enterprise-ready AI.

Table of Contents

Enterprise-ready generative AI is difficult to achieve.

And that's what makes it valuable.

When done right, generative AI has the potential to transform your business and your data engineering team. Organizations across industries are already applying large language models (LLMs) to save time, drive revenue, and gain an advantage over competitors.

Inspired (and threatened) by these early success stories—and GenAI's seemingly endless hype—organizations are pressuring data leaders to support new AI applications. But in the menagerie of AI everything, how do teams think critically and develop AI that drives real business value?

GenAI is a data product. And like any data product, it needs to leverage curated data for a clear business use case in order to provide real value. A chatbot powered by an API call to OpenAI isn't going to cut it.

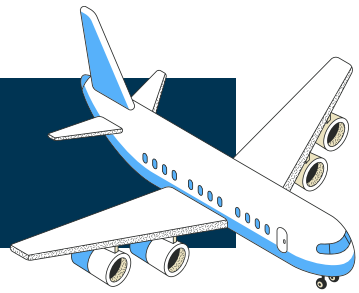
That's why teams are turning to an emerging, but already essential, framework to differentiate their AI for business value:

Retrieval augmented generation (RAG).

But what is it? And how does it help? In this guide, we'll take you through the fundamentals of RAG, including the application architecture, benefits, implementation, and a few real-world use cases to get you started.

Ready? Let's get into it.

RAG: The Basics



Let's start with the (multi)million dollar question:

How does RAG architecture work in the context of enterprise AI products?

On its own, an off-the-shelf LLM is trained on publicly available data (basically, the entirety of the internet).

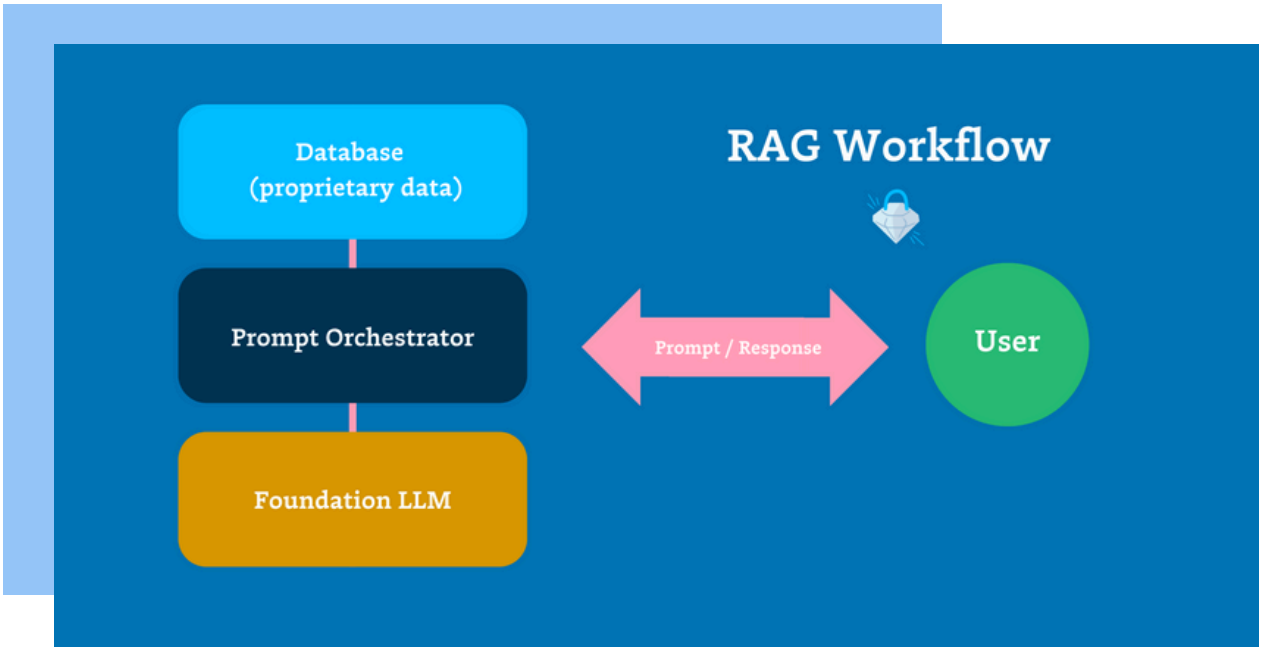
So, let's say for example an airline leveraged one of these LLMs exclusively to develop a customer service chatbot. That chatbot could probably answer a lot of general questions about air travel and destinations, but it couldn't provide some really basic functionality, like helping a customer cancel or move a ticket.

This is where retrieval-augmented generation (RAG) comes in handy. RAG is a framework that connects your LLM to a curated, dynamic database. This improves the LLM's outputs by allowing it to access and incorporate up-to-date and reliable information into its responses and reasoning.

So, if our airline's data engineering team used a RAG system, they could enhance that foundational LLM with real-time data retrieval from its own proprietary database. That means the model could access information about that specific customer, their past transactions, and the airline's cancellation policies.

With curated first-party data to augment its responses, the chatbot could then retrieve a customer's specific flight details, provide more helpful content during interactions, and even take action based on conversations.

RAG Structure



The structure of RAG architecture

In most organizations, data engineers are responsible for developing the RAG architecture. It's a complex process that involves prompt engineering, vector databases and embedding vectors (more on that in a minute), data modeling, data orchestration, and data pipelines — all tailored for RAG.

And since it's new, introduced by Meta in 2020, many companies don't yet have enough experience with it to establish best practices.

But the end result?

An incredible amount of value added to AI-powered data products.

RAG Structure

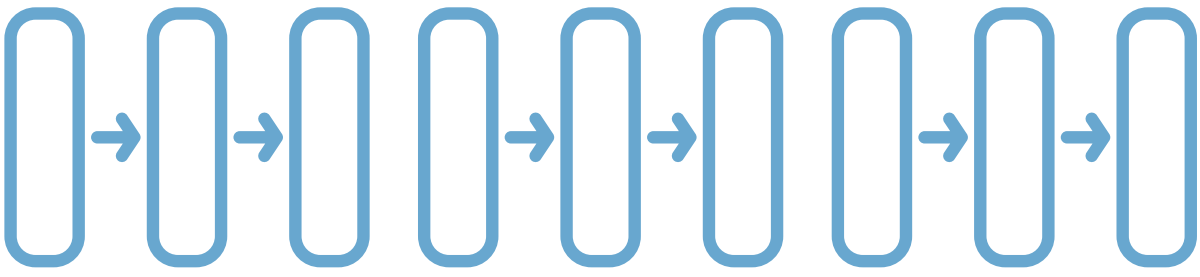
Your data is the backbone

The foundation of RAG architecture is a reliable database that acts as a repository of trusted, proprietary information. This database needs to be continually updated and maintained through a robust data pipeline — one that's typically capable of processing structured and unstructured data, because most enterprise use cases will involve both.

Understanding the RAG operational flow

Here's how a RAG flow works:

1. Query processing: The process begins when a user submits a query to the system. This query is the starting point for the RAG chain's retrieval mechanism.
2. Data retrieval: Based on the query, the RAG system searches the database to find relevant data. This step involves complex algorithms to match the query with the most appropriate and contextually relevant information from the database.
3. Integration with the LLM: Once the relevant data is retrieved, it's combined with the user's initial query and fed into the LLM.
4. Response generation: Using the power of the LLM and the context provided by the retrieved data, the system generates a response that is not only accurate but tailored to the specific context of the query.



RAG in Action: Preset

In early 2024, Maxime Beauchemin, creator of Apache Airflow and Superset and the founder & CEO of Preset, told us how his team uses RAG to bring AI-powered capabilities to Preset's BI tools.

Originally, Max and his team explored using OpenAI's API to enable AI-powered capabilities, such as customers using plain-language text-to-SQL queries. But given the limits of ChatGPT's context window, it wasn't possible to provide the AI with all the necessary information about their datasets — like their tables, naming conventions, column names, and data types. And when they attempted fine-tuning models and custom training, they found the infrastructure wasn't quite there yet — especially when they also needed to segment for each one of their customers.

"I can't wait to have infrastructure to be able to custom train or fine-tune with your own private information from your organization, your GitHub, your Wiki, your dbt project, your Airflow DAGs, your database schema," says Max. "But we've talked to a bunch of other founders and people working around this problem, and it became clear to me that custom training and fine tuning is really difficult in this era."

Given that Max and his team couldn't take a customer's entire database, or even just the metadata, and fit it in the context window, "We realized we needed to be really smart about RAG, which is retrieving the right information to generate the right SQL. For now, it's all about retrieving the right information for the right questions," he says.

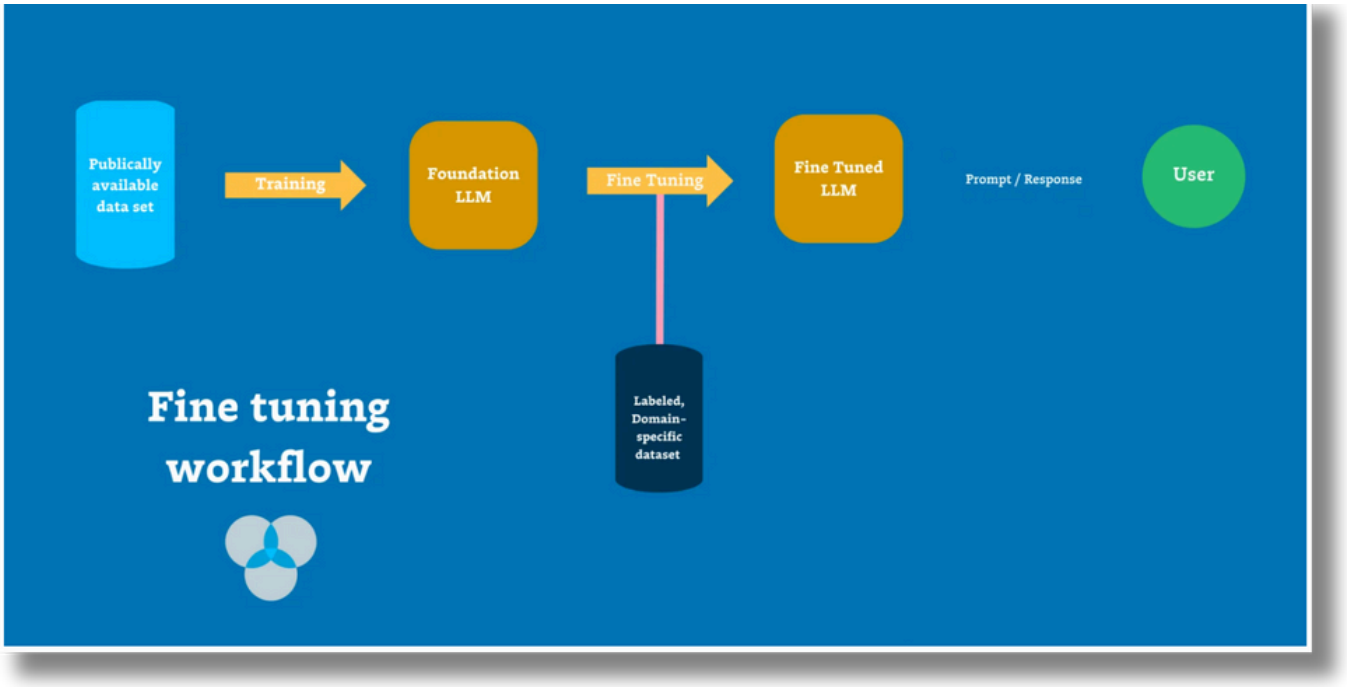
Fine-tuning

As we mentioned in our introduction, the RAG architecture has a few defining characteristics that make it a go-to option for developing valuable and differentiate enterprise AI.

But it's not the only way.

Depending on the resources available some organizations may opt for an alternative to drive value from their GenAI initiative.

So, before we dive deeper into the benefits of RAG, let's consider the most common alternative: fine-tuning your LLM.



Fine-Tuning vs RAG

In the context of enterprise-ready AI, the end goal of RAG and fine-tuning are the same: drive greater business value from AI models. But rather than augmenting an existing LLM with access to a proprietary database, fine-tuning goes deeper by tailoring the model itself for a specific domain.

Fine-tuning involves training an LLM on a smaller, specialized, labeled dataset and adjusting the model's parameters and embeddings based on new data. By aligning the model with the nuances and terminologies of a niche domain, fine-tuning helps the model perform better for specific tasks.

Unfortunately, fine-tuning isn't always the most practical solution. For one, training an LLM requires a lot of time and computational power. It also involves carefully labeling your training data, which can be quite cumbersome, even when considering that you're likely using a smaller dataset than what would be required for a RAG approach.

And unlike RAG, fine-tuning operates like a black box — since the model internalizes the new data set, it's difficult to understand the reasoning behind new responses. So, if your fine-tuned model starts hallucinating or delivering bad responses, you also won't have an easy way to find the root cause and resolve it.

Benefits of RAG for Enterprise AI

For most enterprise use cases, RAG is a better fit than fine-tuning for three reasons: it's more secure, more scalable, and more reliable.

RAG allows for enhanced security and data privacy
With RAG, your proprietary data stays within your secured database environment, allowing for strict access control; compared to fine-tuning, where the data becomes part of the model's training set and potentially exposing it to broader access without commensurate visibility.

RAG is more cost-efficient and scalable
Fine-tuning a large AI model is resource-intensive, requiring significant time and compute power. By leveraging first-party data into responses instead of model parameters, RAG is able to limit resource cost at both the compute level be eliminating the training stage as well as the human level by avoiding the weeks- or months-long process of crafting and labeling training sets.

RAG delivers more trustworthy results
The value of AI rests on its ability to deliver accurate responses. RAG excels in this area by consistently pulling from the latest curated datasets to inform its outputs. And if anything does go wrong, the data team can more easily trace the source of the response to develop a clearer understanding of how the output was formulated—and where the data went bad.

But while RAG is the preferred option for most use-cases, that doesn't mean that RAG and fine-tuning are mutually exclusive either. In fact, if you have the resource, there are certainly benefits to leveraging both—training your model to pull the most relevant data from the most targeted contextual dataset. Simply, consider your specific needs first, and make a decision that will drive the most value for your stakeholders.

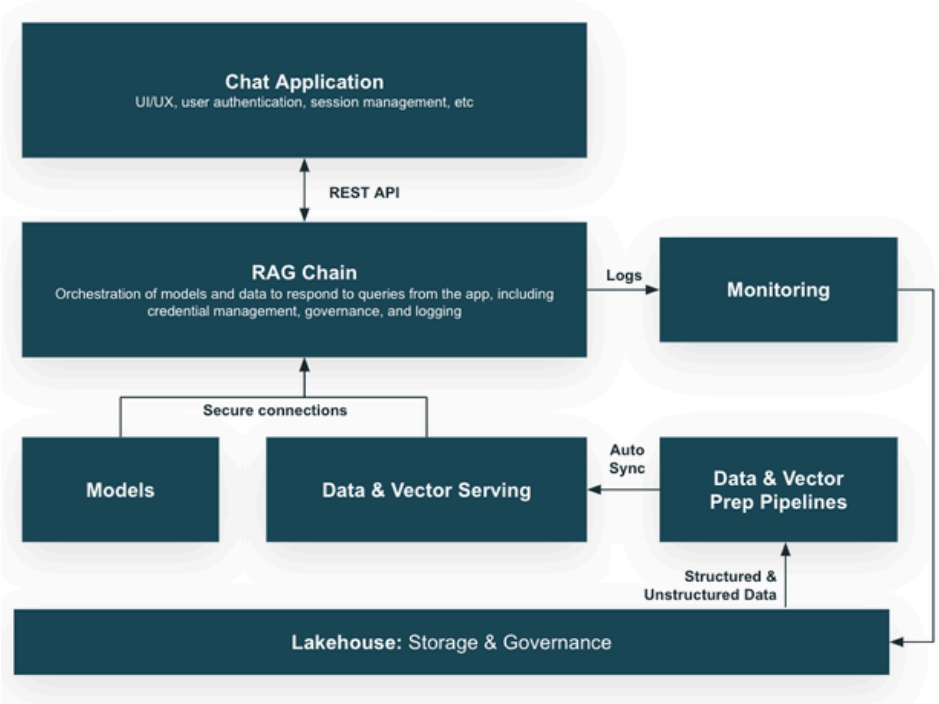
Vector Databases

While RAG relies primarily on the curation of appropriate datasets to power more valuable and differentiated responses, it's not without its tech as well. And there's no technology more notable for its contribution to the RAG process than vector databases.

A vector database (like Pinecone for example) is a database that leverages multidimensional data points, or vectors—which are often derived from complex data types like images, text, or audio—to optimize data for computational analysis.

Vector databases are engineered to perform fast and efficient searches across high-dimensional spaces by identifying data points — or vectors — that are most similar to a given query vector, allowing for more accurate and contextually relevant outputs.

Similarly, in natural language processing (NLP) tasks, RAG utilizes vector databases to find text segments whose meanings closely match the query, ensuring that the generated responses are both relevant and precise.



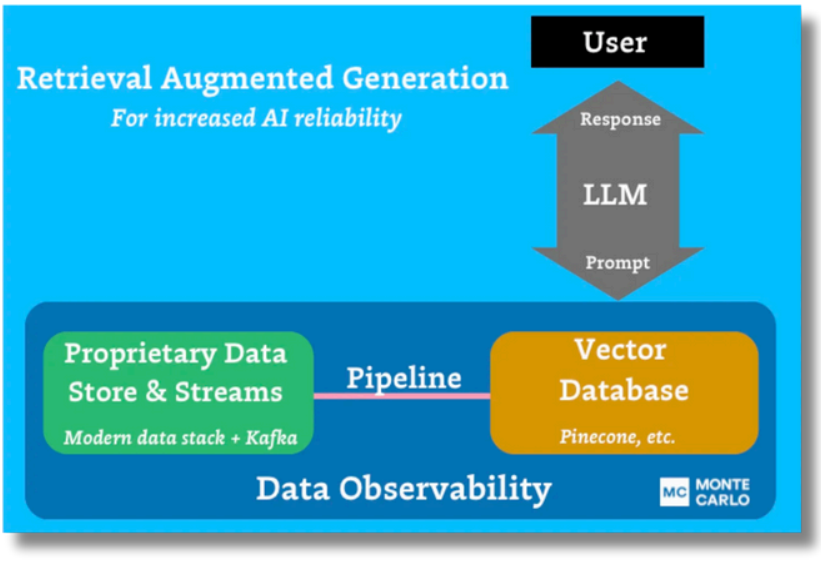
Data Observability

At its heart, AI is a data product. And like any data product, it requires current, accurate, and reliable data to be successful. RAG can improve the accuracy of AI for a given use-case—but only if the data is reliable to start. So, how do you ensure data reliability at the scale required for AI? That's where data observability comes in.

Just like any database, vector databases involve data pipelines that ingest and consume data. Their lineage can be mapped and their health can be measured using the same core pillars of freshness, volume, and schema (metadata).

Data observability gives data leaders visibility into the health of their AI pipelines end-to-end—from ingestion to output—to detect and triage data quality issues at scale, while the data lineage provided by enterprise data observability allows data teams to understand the impact resolve issues faster.

At Monte Carlo, we're committed to supporting the (still emerging) AI stack to help our customers understand at-a-glance the health of the data powering their AI models. Read the full article to find out how Monte Carlo is bringing observability to Pinecone today—and what you can expect next for reliability in the AI stack.



V. CASE STUDY

Credit Karma's Journey to Reliable GenAI

credit karma™

In addition to providing credit scores to millions of customers, fintech platform Credit Karma delivers support for finding financial products like identity monitoring, insurance, checking accounts, savings accounts, and loans.

As Monte Carlo customers, VP of Engineering Vishnu Ram were well versed in the benefits of Monte Carlo's end-to-end data observability platform. So, when it came time to get their data in order for a generative AI push, the team already had a head start.

Intuit Assist—a chatbot that helps members find answers to unique questions—leverages an LLM to generate responses. But when it comes to personalized financial information, hallucinations and inaccuracies aren't an option.

Monte Carlo gave Vishnu's team visibility into how the model was operating and how changes affected its output—such as adding more data sources to RAG (retrieval augment generation) or switching models.

"We don't have any choice — we need to be able to observe the data," Vishnu says. "We need to understand what data we're putting into the LLM, and if the LLM is coming up with its own thing, we need to know that — and then know how to deal with that situation."

"If you don't have observability of what goes into the LLM and what comes out, you're screwed."

Today, Credit Karma has 135+ Monte Carlo users across 38 domains, demonstrating not just the power of data observability—but the value any data team can create with the help of curated, accurate, and reliable data.

The Biggest Challenge for RAG: Reliable Data

If the key to valuable AI is differentiation, then the key to valuable differentiation is reliable data.

Whether you're building GenAI applications or a new marketing dashboard, the value of your outputs is always rooted in the quality of your inputs. But when it comes to enterprise AI, garbage in, garbage out, goes to a whole new level.

Even with a perfect RAG pipeline, a fine-tuned model, and a clear use case in hand, the most modest of AI ambitions will die on the vine without reliable data to power it.

Data observability gives data teams the power to identify data quality issues before they move through a RAG pipeline — and the visibility to resolve them faster.

Bad data ruins good RAG. If GenAI is anywhere on your roadmap, data quality needs to be at the top of your priority list.

Ready to explore how data observability can speed up your journey towards a cost-effective, secure, and trustworthy future in AI?

Contact our team to learn more.

VII. Additional Resources

Don’t let this be the ending point of your data quality journey toward reliable AI.

Check out more helpful resources including:

- Data Downtime Blog: Get fresh tips, how-tos, and expert advice on all things data.
- O’Reilly Data Quality Framework: The first several chapters of this practitioner's guide to building more trustworthy pipelines are free to access.
- Data Observability Product Tour: Check out this video tour showing just how a data observability platform works.
- Request a Demo: Talk to our team to get a more accurate assessment of your data downtime, its costs, and what level of value you can expect from Monte Carlo.