Eckerson Group | Report

The Ultimate Guide to Embedded Analytics

Keys to Product Selection and Implementation

WAYNE W. ECKERSON AND KEVIN M. SMITH

RESEARCH SPONSORED BY

Qlik

About the Authors

Wayne W. Eckerson is an international thought leader in the data and analytics field since the early 1990s. He is a noted speaker, sought-after consultant, and widely read author. Eckerson has conducted groundbreaking research studies, chaired numerous conferences, and consulted with organizations around the world during his 25 years in the field. Eckerson has also written two books: *The Secrets of Analytical Leaders: Insights from Information Insiders* (2012) and *Performance Dashboards: Measuring, Monitoring, and Managing Your Business* (2005/2010). He has degrees from Williams College and Wesleyan University.

Kevin M. Smith is the principal consultant at NextWave Business Intelligence, where he helps companies build embedded analytics data products and bring them to market. Smith has led teams in designing SaaS products and performance management strategies; he has also led product management for companies such as Birst, ServiceSource, SAP Labs, and Qwest Communications.

Kevin is the author of multiple reports on embedded analytics product strategy, including Beyond the Technical and The Ultimate Guide to Embedded Analytics, and has developed content for GoodData, Looker, Tableau Software, Birst, Izenda, and other analytics vendors. Kevin holds a B.S. in finance as well as an MBA in quality/process management, both from the University of Maryland, College Park.

About Eckerson Group

Eckerson Group is a global research and consulting firm that helps organizations get more value from data. Our experts think critically, write clearly, and present persuasively about data analytics. They specialize in data strategy, data architecture, self-service analytics, master data management, data governance, and data science. Organizations rely on us to demystify data and analytics and develop business-driven strategies that harness the power of data. **Learn what Eckerson Group can do for you!**

Eckerson Group
RESEARCH CONSULTING ADVISORY
GET • MORE • VALUE • FROM • YOUR • DATA

About This Report

This version of the report was revised and updated for January 2023, and is sponsored by Qlik, who has exclusive permission to syndicate its content. The original report was published in February 2018.

Table of Contents

Executive Summary

As we move from an information economy to an insight economy, analytics becomes paramount. Every organization, whether it's a business enterprise or a commercial software vendor, needs to embed analytics into core applications that business people use regularly. But until recently, embedding analytics has been a hit-or-miss endeavor—mostly miss.

Embedding a chart or dashboard into an application does not guarantee that people will use it or gain value from it. Organizations need a methodology for ensuring their investments in embedded analytics pay off. And before they leap into building analytics with in-house developers, they need to understand the trade-offs and capabilities of commercial analytics products.

This report is a comprehensive guide to succeeding with embedded analytics. It steps readers through defining an embedded analytics project, selecting an embedded product, and building and sustaining the application. It describes everything from types of embedding to what to look for in commercial analytics products and to packaging and pricing embedded analytics capabilities, which is of particularly interest to commercial software vendors.

Embedded Analytics for Everyone

The world is full of paradoxes. Here's one for data analytics professionals: analytics becomes pervasive when it disappears.

For decades, business intelligence (BI) and analytics tools have failed to penetrate more than 25% of an organization. And within that 25%, most workers use the tools only once or twice a week. Embedded analytics changes the equation. By inserting charts, dashboards, and entire authoring and administrative environments inside other applications, embedded analytics dramatically increases BI adoption. The catch is that most business users don't know they're "using BI"—it's just part of the application they already use. The best BI tools are invisible.

By placing analytics at the point of need—inside operational or custom applications—embedded analytics closes the last mile of BI. Workers can see the impact of past actions and know how to respond to current issues without switching applications or context. Analytics becomes an indispensable part of the way they manage core processes and solve problems. As a result, embedded analytics has a much higher rate of adoption than traditional BI or analytics.

Embedded analytics has much higher rate of adoption than traditional BI or analytics.

Target Organizations

Embedded analytics makes existing applications more valuable for every organization. Independent software vendors (ISVs) say that embedded analytics increases the value of their applications and enables them to charge more for them. Enterprise organizations embed analytics into operational applications, such as Salesforce.com, and intranet portals, such as SharePoint. In both cases, embedded analytics puts data and insights at users' fingertips when they need it most—both to gain insights and take action.

ISV requirements. In the embedded world, ISVs have more stringent requirements than traditional organizations. (See "Twelve Evaluation Criteria" below.) ISVs must ensure an embedded product looks and feels like their host application, and thus require greater levels of customization and extensibility. Also, cloud ISVs require embedded products that work in multi-tenant environments, with seamless user administration and custom deployments. Many ISVs offer tiered pricing, which requires embedded products with flexible user provisioning. Finally, because ISVs can't always estimate how many customers will purchase the analytics, they need flexible and affordable pricing models.

Enterprise requirements. Traditional enterprises have fewer requirements than ISVs, but that is changing. More companies are pursuing digital strategies that require customer-facing Web applications. And although most don't charge for analytics, as ISVs do, many view data analytics as a key part of the online customer experience. For example, mutual funds now provide customers with interactive dashboards where they can slice and dice their portfolios and take actions such as buying and selling funds. Thus, their requirements for customization, extensibility, multi-tenancy, and security have grown significantly in recent years.

Build or Buy?

Once organizations decide to embed analytics, they need to make a few key decisions. The first is whether to build their own analytics or buy a commercial off-the-shelf tool.

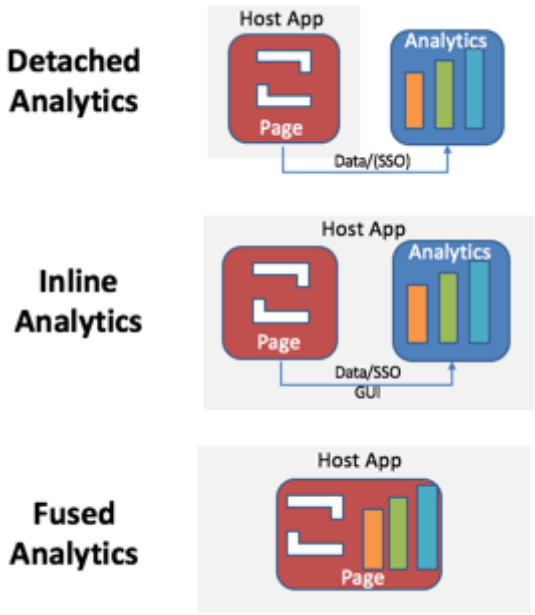
Build. Organizations with internal developers are always tempted to build their own analytics. But unless the analytics are simple and users won't request changes, it's always smart to outsource analytics to a commercial vendor. Commercial analytics products deliver best-of-breed functionality that would take in-house developers years to develop, distracting them from building the host application.

Buy. Many analytics vendors have made their tools more configurable, customizable, and integrate-able with host applications. Most see embedded analytics as a big market and aim to make their tools blend seamlessly with third-party applications. They also make it easy to customize the tool without coding, including changing the graphical user interface (GUI) or the ways users navigate through the tool or interact with its components. When extensive customization or integration is required, customers can use application programming interfaces (APIs) to fine-tune the tool's look and feel, create new data connectors, charts, event actions, and export types.

Types of Embedding

The second decision is to figure out the architecture for embedding analytics. From our research, we've discovered three primary approaches. (See figure 1.)

Figure 1. Three Types of Embedding



Detached analytics. This is a lightweight form of embedding where the two applications—host and analytics—run separately but are tightly linked via URLs. This approach works well when multiple applications use the same analytics environment, portal, or service. A common example is Google Analytics, a commercial service that multiple groups inside an organization might use to track Web traffic on various internal websites. There is no commonality between the host application and analytics tool, except for a shared URL and shared data. The two applications might also share a common authentication mechanism to facilitate single sign-on (SSO). Ths approach is rather uncommon these days.

Inline analytics. With inline analytics, output from an analytics tool is embedded into a host application—it looks, feels, and acts like the host but runs as a separate element, tab, or module within it. For example, a newspaper might embed a chart within the text of an article on a webpage. Or an ERP application might present users with a dashboard upon logging in that displays summary activity from each module in the application. Or there might be a separate tab where customers can view analytics about their activity within the application. In most cases, the embedded components sit within an iFrame, which is an HTML container that runs inside a webpage. iFrames were once the predominant method for embedding analytics content but are disappearing due to security and other concerns. (See next section.)

Fused analytics. Fused analytics delivers the tightest level of integration with a host application. Here, the analytics (e.g., a chart, table, or graphical component) sit side by side with the host application components and communicate bi-directionally with them. This created a "fused" or integrated environment where the end users aren't aware that a third-party tool is part of the experience. For example, an inventory manager can view inventory levels in various warehouses and place replenishment orders without leaving the screen. Or a retail store manager can view daily demand forecasts and then click a button to create the shift schedule for the following week. Fused analytics is facilitated by JavaScript libraries that control front-end displays and REST API calls that activate server functions. Analytics tools with extensive API libraries and programming frameworks make all their functionality available within a host application, including collaboration capabilities, alerts, reporting and augmented analytics features.

Technology Implications

Most analytics vendors can support inline analytics without much difficulty. They simply provide "embed code"—a snippet of HTML and JavaScript—that administrators can insert into the HTML code of a webpage. The embed code calls the analytics application to display specified content within an iFrame on the webpage. People who use YouTube and other social media services are familiar with embed code. iFrames are a quick and easy way to embed third-party content, and most analytics vendors support them.

But iFrames have disadvantages. Because they are frames or windows inside a webpage that are controlled by an external application or service, they pose considerable security risks. Also, they operate independently of the host webpage or application—the host can't manipulate what's inside the iFrame, and vice versa. For example, hitting the back button doesn't change what's inside the iFrame. Furthermore, iFrames behave differently depending on the browser, making them difficult to manage. Consequently, a growing number of organizations refuse to allow iFrames, and the software industry is moving away from them

Fused analytics requires tight integration between an analytics and host application. Fused analytics also requires a much greater degree of customization, extensibility, flexibility, and integration than many analytics vendors support out of the box. To simplify fused analytics, many BI vendors have wrapped their APIs in programming frameworks and command line interfaces (CLIs) that make it easy for programmers to activate all functions in the analytics tool. These Javascript frameworks and CLIs have been a boon to analytics embedding. Nonetheless, companies that want to fuse analytics into an application need to look under the covers of an analytics tool to discover its true embeddability. (See "Select a Product" below.)

Product Selection and Implementation

Another major decision is selecting a commercial analytics product to embed. Selecting a product that doesn't include a key feature you need, such as alerts or exports to PDF or printing, can undermine adoption and imperil your project. Or maybe the product doesn't work seamlessly in a multi-tenant environment, making administration cumbersome and time-consuming and contributing to errors that undermine customer satisfaction. Or your deployment drags out by weeks or months because most customizations require custom coding.

This report is designed to help you avoid these and other pitfalls of embedded analytics. Whether you are an independent software vendor (ISV) that needs to know how to embed analytics in a commercial, multi-tenant cloud application or the vice president of application development at major corporation who wants to enhance a homegrown application, this report will provide guidance to help you ensure a successful project.

The report outlines a four-part methodology:

1. **Plan the project.** Define goals, timeline, team, and user requirements.
2. **Select a product.** Establish evaluation criteria, create a short list of vendors, conduct a proof of concept, talk to references, and select a product.
3. **Deploy the product.** Define packaging and pricing (if applicable) and develop a go-to-market strategy that includes sales, marketing, training, and support.
4. **Sustain the product.** Monitor usage, measure performance, and develop an upgrade cadence that delivers new features and bug fixes.

The report's appendix drills into the evaluation criteria in depth, providing questions that you should ask prospective vendors to ensure their products will meet your needs.

1. Plan the Project

Many teams start their embedded analytics project by selecting an analytics product to deploy. Although choosing the vendor to power your analytics is a critical step, it shouldn't be the first milestone you tackle. Instead, start by considering these questions: What are you trying to build, for whom, and why? These essential questions will help you better understand your product goals and will aid in selecting the best tool to achieve your goals.

Start by asking: What are you trying to build, for whom, and why?

We recommend a six-step model to ensure that your analytics are not only technically successful, but achieve your business goals.

Establish the Focus

Setting the goals for your analytics project is an essential first step to ensure that all key stakeholders—from the executive team to the end users—are fully satisfied upon project completion.

There are three basic steps to planning for a successful analytics project: define table stakes, define delighters, and define what's out of scope.

1. **Define table stakes.** Table stakes are the essential elements the project must include in order to be considered successful. For example, an end-user organization might decide that the analytics must include dashboards for the CEO; an ISV or OEM might decide that the analytics must support a search bar for ad hoc queries; these items are considered "table stakes," and you should plan to include them as part of your development plan.

2. **Define delighters.** Delighters are product elements that are "nice to have"—not essential, but highly beneficial to customer satisfaction. These may be elements that would greatly streamline workflows, or would simply make the product more enjoyable to use. They aren't critical to using the application, but they would "delight" the customer if present.

3. **Out-of-scope items.** It's also important to decide what won't be in the product. Try to avoid putting the analytics team in the difficult situation of deciding whether to address a late-arriving customer request. Although it's impossible to identify all out-of-bounds features or services in advance, you should try to create a set of guidelines. For example, you might choose to deliver dashboards that users can tailor to their needs, but don't support raw data feeds. Or, you may decide that you'll provide a standardized set of analytics covering a variety of use cases, but you won't build customer-specific, "one-off" data models.

Define the Project Team

The composition of the project team can be a key element in the success or failure of an analytics project. For example, more than a few projects have been derailed at the last minute when the legal team—not included in the process—surfaced major issues with the plan. When structuring your analytics product team, consider including the following roles in your "core team" for the project:

> Product owner/head of product

> CTO

> Head of development/engineering

> Head of sales

> Head of marketing

> Head of operations and support

Next, identify roles that, although not involved in daily decisions, will need to be consulted along the way, including finance, billing, legal, and sales enablement/training.

Create the Plan

A best practice is to get the key project stakeholders in a single room to discuss core elements in a facilitated session. Although it might be necessary to have some participants attend remotely, in-person attendance makes for a faster and more effective session.

Too often project teams—whether analytics-focused or otherwise—fail to create a plan to guide the key steps required to bring embedded analytics to market. Without a plan, teams are liable to spend too

much time gathering requirements and too little time analyzing persona needs. Without planning, the time required to perform key tasks, such as resolving issues from beta testing, might be overlooked. The steps to creating a basic, but useful, plan are simple:

Set project goals. Setting project goals before any technical work starts is a good way to ensure that everyone involved agrees on the success criteria. Set aside time to create project goals as the first step in your analytics plan.

Set a timeline. A timeline may seem obvious, but it's important that, in addition to the overall start and end dates, you plan for intermediate milestones such as:

a. **Start/end of product design.** When will you begin and end the process of defining what functionality will be in your analytics product? Without scheduled start and end dates, this segment of the process can easily extend far longer than anticipated.

b. **Start/end of technical implementation.** When will the technical aspects of the project commence and complete? This should include items such as connecting to data sources, implementing the analytics platform, integrating with the core product, and applying user management.

c. **Start marketing efforts.** If you build it, you want them to come. But you don't want to raise expectations for a speedy arrival before development has completed. Plan on setting dates for key marketing activities, including the development of logos, advertising, creation of demos and sales collateral, and training documentation.

d. **Start/end the beta period.** Before you launch your analytics, you'll want to test your analytical application with a set of carefully chosen beta users. Pick reasonable dates for this process and be sure to include time for selecting beta users, educating them on the product, reviewing testing results, and resolving identified issues.

e. **Start/end user onboarding.** Don't forget that, once the product is complete, you still need to onboard any existing users. Plan to onboard users in tranches—define manageable groups so that your team doesn't become overwhelmed with support issues. And don't forget to leave time between each tranche so that you can resolve any issues you might find.

Create a Communication Plan

It's easy to forget that although you, as a member of the product team, might be fully aware of everything that's taking place within your analytics project, others might not know about your progress. In the absence of information, you might find that inaccurate data is communicated to customers or other interested groups. You can prevent this by establishing a communication plan, both for internal personnel and for potential customers. Although the plans will be different for those inside your walls versus external parties, all communication plans should include:

- Regular updates on progress
- Revisions of dates for key milestones
- Upcoming events such as sneak peeks or training sessions

Set Metrics and Tripwires

Once you've started your product development effort, particularly once you've started beta testing or rollout, it can be hard to identify when critical problems surface. That's why setting metrics and tripwires is a good idea during the planning phase.

It can be hard to identify when critical problems surface. That's why setting metrics and tripwires is a good idea.

Metrics are used to measure product performance and adoption. An embedded product should have a dashboard that enables ISVs and OEMs to monitor metrics and activity across all tenants using the product, alerting administrators when performance goes awry. Consider tracking:

- Product uptime
- Responsiveness of charts and dashboards (i.e., load time)
- Data refresh performance and failures
- Number of reloads
- Total users
- Users by persona type
- Number of sessions
- Time spent using the analytics
- Churn (users who don't return to the analytics application)
- Functionality used, e.g., number of reports published, alerts created, or content shared

Tripwires alert you to critical situations before they cause business-disrupting problems. They are metrics values that, if exceeded, trigger a response from the development team. As an example, you might have a tripwire that states if the product uptime is less than 99.9%, the general rollout of the

analytics product will cease until the issue is resolved. Each metric should have an established tripwire, and each tripwire should contain the following elements:

> A metric value that, if exceeded, triggers a response.

> A predetermined response such as "stop rollout" or "roll back to the previous version."

> A responsible party who reviews the metric/tripwire and determines if action is required.

Although metrics and tripwires don't ensure project success, they can greatly reduce the time —and stress for the team—to address problems.

Choose Target Users

It's a common mistake to think either that you fully understand the users' needs or that all users are the same. Many teams launch embedded analytics products without considering the detailed needs of target users or even the different user types they might encounter. Avoid this situation by creating detailed user personas and doing mission/workflow/gap analysis.

Many teams launch embedded analytics products without considering the detailed needs of their users or even the different user types they might encounter.

Here's how it works:

Step One: Choose personas. The best way to create an engaging data product is to deliver analytics that solve users' problems. This is difficult to do for a generic "user," but it can be accomplished for a specific user "persona" or user type. Start by picking two or three key user types (personas) for whom you will tailor the analytics. These might be strategic users looking for patterns and trends (like executives) or tactical users focused on executing work steps (like salespeople or order fulfillment teams). Although you may ultimately add functionality for many personas to your analytics, start with personas that can impact adoption—key decision makers—first. Get these user types engaged with your analytics application and they can help drive adoption among other users.

Step 2: Identify mission. For each chosen persona, the next task is to understand the user's mission. What is the person trying to accomplish in their role? Are they trying to improve overall sales? Are they striving to increase revenue per customer? Understanding what the persona must accomplish will help you understand where analytics are needed and appropriate cadence.

Step 3: Map workflows and gaps. Now that you understand each persona's mission, the third step is to outline the workflow they follow and any gaps that exist. These steps—and gaps—inform the project team where they can add analytics to assist the persona in accomplishing their mission. Keep it simple. If your persona is "head of sales" and the mission is "increase sales," a simple workflow might be: (a) review sales for the month (b) identify actions taken within those segments (c) recommend more effective tactics to managers.

Within this workflow, you might find opportunities where analytics can improve the effectiveness of the head of sales. Perhaps reviewing sales performance or identifying underperforming segments requires running reports rather than simply reviewing a dashboard. Maybe seeing what actions have been taken requires investigating deep within the customer relationship management (CRM) system and correlating actions back to segments.

By finding information gaps within workflows and understanding personas' missions, project teams can ensure they put high-value analytics in front of users.

By finding information gaps within workflows and understanding personas' missions, project teams can ensure that they put high-value analytics in front of users. It becomes less of a guessing game—replicating existing Microsoft Excel-based reports and hoping the new format attracts users—and more of a targeted exercise. Only analytics that truly add value for the persona are placed on the dashboard, in a thoughtful layout that aids in executing the mission. Engagement increases as target users solve problems using analytics.

2. Select an Embedded Analytics Product

Create Evaluation Criteria

Once you've defined user requirements, you need to turn them into specifications for selecting a product. The following evaluation criteria will help you create a short list of three or four vendors from the dozens in the market. The criteria will then guide your analysis of each finalist and shape your proof of concept.

We've talked with dozens of vendors, each with strengths and weaknesses. Analyst firms such as Gartner and Forrester conduct annual evaluations of Analytics and BI tools, some of which are publicly available on vendor websites. G2 provides crowdsourced research on BI tools, while the German research firm BARC publishes a hybrid report that combines analyst opinions and crowdsourced evaluations.

However, these reports generally don't evaluate features germane to embedded analytics. That's because the differentiators are subtle and often hard to evaluate, since it requires diving into the code.

The differentiators among embedded analytics are subtle and often hard to evaluate since it requires diving into the code.

Key Differentiators

For companies that want to tightly integrate analytics with a host application, there are three key things to look for:

> How quickly can you deploy a highly customized solution?

> How scalable is the solution?

> Does the vendor have a developer's mindset?

Deployment speed. It's easy to deploy an embedded solution that requires minimal customization. Simply replace the vendor logo with yours, change font styles and colors, copy the embed code into your webpage, and you're done. But if you want a solution that has a truly custom look and feel (i.e., white labeling), with custom actions (e.g., WebHooks and updates), unique data sources and export formats, and that works seamlessly in a multi-tenant environment, then you need an analytics tool designed from the ground up for embedding.

The best tools not only provide rich customization and extensibility, but they also do so with minimal coding.

The best tools not only provide rich customization and extensibility, but they also do so with minimal coding. They've written their own application so every element can be custom-tailored using a point-and-click properties editor. They also provide templates, themes, and wizards to simplify development and customization. And when customers want to go beyond what can be configured out of the box, the tools can be easily extended via easy-to-use programming frameworks that leverage rich sets of product APIs that activate every function available in the analytics tool.

Moreover, the best embeddable products give host applications unlimited ability to tailor analytic functionality to individual customers. This requires analytics tools to use a multi-tenant approach that creates an isolated and unique analytics instance for each customer. This enables a host company to offer tiered versions of analytic functionality to customers, and even allow customers to customize their analytics instance. This mass customization should work whether the host application uses multitenancy and/or containerization or creates separate server or database environments for each customer.

Scalability. It's important to understand the scalability of an analytics solution, especially in a commercial software setting where usage could skyrocket. The tool needs strong systems administration capabilities, such as the ability to run on clusters and support load balancing. It also needs a scalable database—whether its own or a third party's—that delivers consistent query performance as the number of concurrent users climbs and data volumes grow. Many vendors now offer in-memory databases or caches to keep frequently queried data in memory to accelerate performance. The software also must be designed efficiently with a modern architecture that supports microservices and a granular API. Ideally, it works in a cloud environment where processing can scale seamlessly on demand.

Developer mindset. When developers need to get involved, it's imperative that an analytics tool is geared to their needs. How well is the API documented? Can developers use their own integrated development environment, or must they learn a new development tool? Can the tool run on the host application server or does it require a proprietary application server and database? How modern is the tool's software architecture? Does it offer JavaScript frameworks, which help simplify potentially complex or repetitive tasks by abstracting API calls and removing the need for deep knowledge of the analytics tool's APIs by your developers?

Companies are adopting modern software architectures and don't want to pollute them with monolithic, proprietary software from third parties.

Increasingly, companies are adopting modern software architectures and don't want to pollute them with monolithic, proprietary software from third parties. The embedded analytics solutions of the future will insert seamlessly into host code running on host application and Web servers, not proprietary servers and software.

Twelve Evaluation Criteria

It's important to know what questions to ask vendors to identify their key differentiators and weak spots. Below is a list of 12 criteria to evaluate when selecting an embedded analytics product. (See the appendix for a more detailed description of each item.)

These criteria apply to both ISVs and enterprises, although some are more pertinent to one or the other. For instance, customization, extensibility, multi-tenancy, and pricing/packaging are very important for ISVs; less so for enterprises.

1. **Embedding.** What parts of the analytics tool can you embed, and which can you not? The best embedded analytics tools let you embed everything—including mobile usage, authoring, and administration. Are objects embedded via iFrames (i.e., inline) or modern techniques, such as Javascript programming frameworks?

2. **Customization.** What parts of the tool can you customize without coding? The best tools let you create a custom graphical interface that blends seamlessly with the host application without developer assistance. The less coding, the quicker the project deploys.

3. **Extensibility.** Does the tool provide APIs or plug-in frameworks that make it easy to add new functionality, such as new charts, data connectors, or export functionality?

4. **Data architecture.** How flexible is the data architecture? Can it query the host database and other data sources directly? Can it load data into its own in-memory or persistent database to improve scalability and performance? Can it model, clean, integrate, and transform data, if needed, using a point-and-click interface?

5. **Process integration.** Does the embedded analytics tool support bidirectional communication with the host application? Can the host navigation framework (i.e., a panel or tree) drive the analytics tool, and can the analytics tool update the host application? How much custom coding is required to support such integration?

6. **Security.** Does the tool support host application authentication and a single-sign-on framework? Does it support its own authentication framework, if needed? What level of permissions and access control does it support? Does it provide row- and column-level security?

7. **Multi-tenancy.** Can you customize a single dashboard so each tenant receives a different view? Can each tenant run its own database? Can each tenant configure permissions for its own analytics environment? Can customizations be upgraded? Most importantly, can tenants be centrally administered from a single console rather than individually?

8. **Administration.** Can the embedded product be managed from the application's management console? Does the embedded product offer an administrative tool to handle DevOps, user management, systems monitoring, systems management (e.g., load balancing, cluster management, backup/restore), cloud provisioning, and localization?

9. **Systems architecture.** What is the systems footprint of the BI tool? Does it conform to your data center or cloud platform requirements? How lightweight is the product? Does it require an application server? Database server? Semantic layer?

10. **Vendor.** How much experience does the vendor have with embedding, and to what level (e.g., detached, inline, fused)? What kind of programs does it offer to jumpstart projects? Do they offer flexible or value-based pricing to match your requirements?

11. **Analytics.** What type of analytics does the product support? Is it predominantly a reporting tool, dashboard tool, OLAP tool, self-service tool, or data science tool? Although all vendors today provide

a complete stack of functions, most excel in one or two areas. Does it offer value-added features, such as alerts, collaboration, natural language queries, and augmented analytics?

12. **Software architecture.** Does the analytics tool use a REST API and JSON to communicate between front-end and back-end components? Is the front end written with a JavaScript or Python framework? Is the software designed around microservices?

For a more detailed description of these criteria, see the **Appendix** below.

3. Build Your Analytics Application

With a plan and tool selected, the next step is to begin development. But perhaps not the development that might initially come to mind. We recommend that, alongside the technical implementation of your analytics, you develop the business aspects of your project. This phase requires you to fully consider how the analytics will be introduced to the market—how they will be packaged, priced, and supported post-launch.

Define Packaging

Start by designing the packaging for your analytics. Packaging is particularly important for software vendors who sell a commercial product. But enterprises that embed analytics into internal applications can also benefit from understanding these key principles.

Teams often consider analytics to be an "all-or-nothing" undertaking. You either have a complete set of analytics with a large set of features, or you don't have any analytics at all.

But this approach fails to consider different user needs. Expert users may desire more sophisticated analytics functionality, while novice users may need less. The "all or nothing" approach also leaves you with little opportunity to create an upsell path as you add new features. It's better to segment your analytics, giving more powerful analytics to expert users while allowing other users to purchase additional capabilities as they need them.

The Tiered Model

You never want to give users every conceivable analytical capability from the outset. Instead, use a tiered model. If you've ever signed up for an online service and been asked to choose from the "basic," "plus," or "premium" version of the product, you've seen a tiered model. Keep it simple, don't add too many levels from which buyers are expected to choose. For example, you might use the following tiers:

- **Basic.** This is the "entry level" tier and should be the default for all users. You put these introductory, but still useful analytics in the hands of every single user so that they understand the value of analytical insights. Most organizations bundle these analytics into the core application without charging extra, but they usually raise the price of the core application by a nominal amount to cover costs.
- **Plus.** These are more advanced analytics, such as benchmarking against internal teams (e.g. the western region vs. the eastern region), additional layers of data (e.g. weather data, economic indicators, or financial data), or the ability to drill deeper into charts. This tier should be priced separately, as an additional fee on top of the core application.
- **Premium.** The top tier will be purchased for use by power users, analysts, or other more advanced users. Here, you might add in features such as external benchmarking (e.g. compare performance to the industry as a whole), and the ability for users to create entirely new metrics, charts, and dashboards. This will be the most expensive tier.

Architecting your offering in this format has several key benefits for data product owners:

- **It doesn't overwhelm the novice user.** Although offering too little functionality isn't ideal, offering too much can be worse. Upon seeing a myriad of complex and perhaps overwhelming features, users may decide the application is too complicated to use. These users leave and rarely return.
- **It provides an upgrade path.** Over time, you can expect customers to become more sophisticated in their analysis needs. Bar charts that satisfied users on launch day might not be sufficient a year down the road. The tiered model allows customers to purchase additional capabilities as their needs expand—you have a natural path for user growth.
- **It makes it easier to engage users.** How can you entice customers to buy and use your data product unless they can see the value that it delivers? Including a "basic" analytics tier with minimal, but still valuable, functionality is the answer. The basic tier can be offered free to all customers as a taste of what they can experience should they upgrade to your advanced analytics tiers.

Add-on Functionality

Unfortunately, not all customers will be satisfied by your analytics product, even if it's structured into a tiered model. Some will require custom metrics, dashboard structures, and more data. Here are some "add-on" offerings that you can charge extra for:

- **Additional data feeds.** Although your core analytics product might include common data sources, some customers will require different or more data feeds for their use cases. These might include alternative CRM systems, alternative financial systems, weather, economic data, or even connections to proprietary data sources.

- **Customized models.** A "custom data model" allows buyers to alter the data model on which the product is based. If a buyer calculates "customer churn" using a formula that is unique to them, support this model change as an add-on.
- **Visualizations.** Customers often request novel ways of presenting information, such as new charts, unique mobile displays, or custom integrations.
- **More data.** The product team can augment an analytics application by providing more data: Seven years instead of five, detailed transactional records instead of aggregated data.

Services

Data applications can be complex, and they are often deeply integrated into many data sources. For this reason, you might consider offering services to augment your core analytics product:

- **Installation/setup.** Offer assistance to set up the analytics product, including mapping and connecting to the customer's data sources, training in-house support personnel, and assisting with loading users.
- **Customization.** Offer to create custom charts, metrics, or data models.
- **Managed analytics.** Occasionally, a data product customer requests assistance in interpreting the analytics results. This can take the form of periodic reviews of the analytics (e.g., quarterly check-ups to assess performance) or an "expert-on-demand" service where the customer calls when they have analysis questions.

The situations above are very different from normal product technical support. Managed analysis services can be a highly lucrative revenue source, but they can also consume more resources than anticipated and skew your business model from a product orientation to a services model.

Establish Pricing

Pricing your analytics sounds like a simple proposition—far less complex than the technical product implementation—but that's rarely the case. In fact, we've seen more than a few instances where the pricing portion of the project takes longer than the actual analytics implementation. But determining the fees to charge for your data product doesn't have to be daunting. Here are our guidelines to help you avoid pricing pitfalls.

Charge for your analytics. Analytics can help users make decisions faster, more accurately, and improve overall process cycle times. Such business improvements have value, and you should charge for providing it. However, if your analytics doesn't add value, some product teams decide to offer analytics free of charge. When there is an apparent mismatch between the value and the price of an analytics product, the answer is to revisit the persona/mission/workflow/gap step of the development process.

Start early. Setting pricing late in the process is a mistake because once the product is fully defined and the costs are set, the flexibility you have for creating pricing options is severely limited. Start early and craft price points appropriate for each product tier (basic, plus, premium) rather than trying to rework key project elements just before launch day.

Keep it simple. Complicated pricing models turn off buyers. They cause confusion and slow the buying cycle. Limit the add-ons available and include key functions in the price of the core application.

Make the basic tier inexpensive. Keep the basic tier as inexpensive as possible. You want people to try analytics and get hooked so they purchase a higher tier. Roll the extra price into the cost of the core application and ensure that every user has access to the basic tier.

Match your business model. If your core application is priced based on a fee per user per month, add a small percentage to that fee to account for the additional value of analytics. Do not add a new line item called "Analytics Functionality" that jumps out at the buyer. Make analytics a part of what your product does.

Plan for Supporting Processes

Many teams spend significant energy designing analytics, creating dashboards, and thinking through pricing, but most forget to consider support processes. Embedded analytics as part of a customer-facing data product are inherently different from enterprise or "inside your walls" analytics. Data products require marketing plans, sales training, and other processes that will allow a properly designed analytics application to flourish post-launch. Here's how to get started planning your application's supporting processes.

List the Impacted Processes

The first step to getting your supporting processes ready is enumerate exactly what will be impacted. There are two ways to go about this step:

1. **Brainstorm the processes.** The product team spends about 45 minutes brainstorming a list of potentially impacted processes. This is no different from any other brainstorming session—just be sure that you are listing processes (e.g. "process to intake/resolve/close support tickets") and not organizational names that are less actionable (e.g., "the operations group").
2. **Work from other product lists.** If you are part of an organization that has fielded products in the past, you might already have checklists for "organizational readiness" lying around. If so, the list of support processes developed by another product team might be a great place to start. You'll find that you need to customize the list a bit, but the overlap should be significant, saving you time.

Here is a list of processes commonly impacted by a new data product:

- Provisioning or installation process
- New user onboarding process
- Trouble ticket or issue management
- User experience processes
- Product road mapping process, including request intake
- Utilization tracking or monitoring
- Sales training process
- Billing process
- Decommissioning or deactivation process

Define Changes to Processes

The next step is to determine the degree to which each of the listed processes might need to change to support your new data product.

- **Create a simple flow chart** for each potentially impacted process.
- **Apply scenarios.** Pretend an issue occurs with your analytics. Can your existing process address it? Add or modify process steps to accommodate the analytics product.
- **Publish and train.** Present the new processes to any teams that might be impacted and store the process documentation wherever other process documents are kept.

Create Metrics

With new processes in place, you need to monitor process performance to ensure that everything is working as planned. For each new or modified process, create metrics to track cycle times, throughput, failure rate, cost, and customer satisfaction. Benchmark these processes against existing processes to ensure they are performing in parity.

4. Sustain Your Analytics

The process isn't over when you've deployed your analytics and brought users on board. In fact, the most successful analytics teams view embedded analytics as a cycle, not a sprint to a finish line. Post-launch, you need to consider what is working and what isn't; and you need to add or fine-tune functionality to better meet persona needs. You need to continuously improve to ensure that analytics adoption and usage doesn't decline over time.

Create a Plan to Gather Feedback from Users

People always find unique ways of using analytics functionality. Learn what your users are doing; their experiments can guide your project development. Here are three ways to gather feedback on your analytical application:

- **Use analytics on analytics.** Some platforms allow you to monitor which dashboards, charts, and features are being used most frequently. Track usage at the individual and aggregate level. What functionality is being used? How often is the functionality reloaded? Number of sessions? New, recurring, one time use counts? Three-month growth by user, tenant, type of user, etc?
- **Monitor edge cases.** The users who complain the most, submit requests, and call your help desk are a gold mine of information. They will often talk—at length—about what functionality can be implemented to make the analytics better for everyone. Don't ignore them.
- **Shoulder surfing.** Shoulder surfing is an easy way to gather insights. Get permission from users to observe them interacting with the analytics product on their own tasks at their own computers in their own environments. Shoulder surfing can uncover incredible insights that users might fail to mention in a formal focus group.

Build a Road Map to Expand Personas and Functionality

Although you started with a limited number of personas and workflows during the initial implementation, the key to sustaining your analytics is to expand both the personas served and the use cases addressed. If you started with an executive persona, consider adding tactical personas that need information about specific tasks or projects. Also, add workflows for existing personas. For example, add a budgeting dashboard for the CFO to complement the cash flow might analytics previously deployed.

Communicate the Plan

Unfortunately, in the absence of new information, users will assume that no progress is being made. Even if you can't add all the personas, workflows, and functionality required immediately, make sure to create a communication plan so users understand what's coming next and for whom.

Conclusion: Success Factors

Embedding another product in your application is not easy. There's a lot that can go wrong, and the technology is the easy part. The hard part is corralling the people and establishing the processes required to deliver value to customers.

Here are key success factors to keep at the forefront of your mind during an embedded analytics project:

1. **Know your strategy.** If you don't know where you're going, you'll end up somewhere you don't want to be. Define the goal for the project and keep that front and center during design, product selection, and implementation.
2. **Know your users.** Pushing dashboards to customers for the sake of delivering information will not add value. Dashboards, whether embedded or not, need to serve an immediate need of a specific target group. Identify and address information pain points and you'll succeed.
3. **Identify product requirements.** It's hard to tell the difference between embedded analytics tools. Use the criteria defined in this report to find the best product for your needs. It may not be one you already know!
4. **Define a go-to-market strategy.** Here's where the wheels can fall off the bus. Before you get too far, assemble a team that will define and execute a go-to-market strategy. Especially if you are an ISV, get your sales, marketing, pricing, support, training, and legal teams together at the outset. Keep them informed every step of the way. Make sure they provide input on your plan.

Following these key principles will help ensure the success of your embedded analytics project.

Appendix: Detailed Evaluation Criteria

This section provides more detail on the evaluation criteria presented in this report.

1. **Embedding.** Embedding defines what can be embedded and how.

    a. *What parts of the analytics tool can you embed, and which can you not?* The best embedded analytics tools let you embed everything—charts, dashboards, authoring, collaboration, data preparation, data modeling, database, and administration. Some analytics tools still have a desktop dependency, meaning you can't use them to embed a self-service analytics environment inside an application. The same is true for mobile applications that require users to download an app from an app store.

    b. *How are the elements embedded?* Are objects embedded via iFrames or JavaScript libraries with REST API calls? Some vendors disguise iFrames inside JavaScript embed code, and some have overcome many of the deficiencies of iFrame technology. Many now offer programming frameworks in a variety of languages that make it easy for developers to call functions without knowing the intricacies of APIs.

    c. *Where can the embedded application run?* Embedded code should display everywhere the host application runs, including Web, desktop, cloud, and mobile applications as well as edge devices. However, iFrames may corrupt the display on small screens and, as mentioned above, a native mobile application that users need to download from an external app store may nullify the benefits of embedding the analytics tool.

    d. *Does the analytics tool integrate out of the box with popular packaged applications?* Many customers want tools that embed quickly into Salesforce or SharePoint. Some tools have a wizard to accelerate this integration.

2. **Customization.** Customization changes the styling of the product, often called white labeling.

    a. What parts of the tool can you customize without coding? The best tools offer a properties editor for every object or element in the tool, including charts, tables, text labels, controls (e.g., panels, menus, buttons, filters), headers/footers, borders, backgrounds, icons, and layouts. The editor is a point-and-click interface that non-developers can use to configure and style any element, including changing colors, fonts, borders, and actions (e.g., click, hover, enter). Most tools let you package styles for various elements into a theme. Customers can apply themes globally or to individual reports, groups of users, and roles.

b. How do you customize elements that can't be configured? Most tools provide a CSS configuration file that developers can modify to customize the style of different graphical display elements. For instance, developers can add custom animations to a graphical element, special fonts not supported by the product, or tree-like navigation structures that expand/collapse upon click. Interestingly, some vendors require custom CSS scripting to replace a logo, since most customers want to customize its placement and appearance.

3. **Extensibility.** Extensibility is important when customers want to add new functionality to the analytics tool. A good tool provides one or more mechanisms for extending the functionality of a product (e.g., adding new charts, utilities, or calculations) or adding completely new functionality. Ideally, the mechanism, such as a plug-in library or software development kit, will minimize the amount of coding and make the extension available in a library with out-of-the-box options (e.g., third-party visualizations inside a chart library). Ideally, an embedded product should support the following extensions:

   a. Custom visualizations. Customize existing visualizations, create new ones from scratch, or insert third-party visualizations into the tool's charting library.

   b. Data connectors. Create new data connectors for unsupported data sources, applications, files, and external data sets.

   c. Application components. New navigation panes, menus, filters, selectors, tables, buttons, and so on with corresponding actions.

   d. Utilities and functions. Customize existing utilities, such as schedulers, backup mechanisms, export formats, printing, alerts, and dashboard layouts.

   e. Transformations. Custom functions to clean, parse, concatenate, or integrate data.

   f. Formulas. Custom formulas for calculating metrics.

   g. Other. Whatever people can dream up, they can build and integrate.

   API reference document. A litmus test of extensibility is the quality of the vendor's API documentation. It should conform to industry standards, clearly describe objects, methods, and properties, and show code samples in a variety of languages. It's also important to know whether the vendor offers libraries, tutorials, and use case scenarios to supplement API documentation.

4. **Data architecture.** The data architecture supplies data to the analytics tool and determines its scalability, performance, and flexibility to address new questions.

a. **Data sources.** What data sources and applications can the tool query out of the box? Does it support relational, Hadoop, NoSQL, OLAP cubes, cloud applications, and external data sets? Does it support generic connectors (e.g., ODBC, JDBC, OLE DB) or native connectors and application calls?

b. **Query architecture.** Does the tool query data sources directly or extract and store data locally? Direct queries give real-time data, but might overwhelm the host application. The extract-and-store approach enables customers to pre-integrate and model the data and delivers faster, more consistent performance. Tools that provide both offer the greatest flexibility.

c. **Semantic layer.** Does the tool require the creation of a semantic layer or business model that users query? Does it create it automatically, or does it let users query the source tables directly? Can a semantic layer query and join data from multiple sources (e.g., query federation)? Semantic layers simplify ad hoc queries, but add overhead during design and runtime.

d. **Data storage.** Does the tool store data locally? Does it come with its own database or let customers select a database of their choice, both for content data and metadata? Does it offer an in-memory database or a persistent one, or both? Can other applications or services query the database? A separate database improves scalability and offloads queries from the host application, but it must be managed separately. A database from a BI provider might not be as scalable as a one from a database specialist.

e. **Transformations.** Can administrators clean, integrate, and transform data to facilitate queries? Can administrators create aggregates to speed query performance?

5. **Process integration.** The hallmark of fused analytics is that it supports bidirectional communication between the host application and the analytics tool.

   a. **Write-backs.** Can the analytics tool update the host application through its API or update its underlying data store?

   b. **WebHooks.** Can the analytics tool trigger an action in the host application? For example, if a metric exceeds a threshold, can it issue an alert in the host application?

   c. **Navigation.** Can users click on a navigation pane in the host application to change what is displayed by the analytics tool?

6. **Security.** Security is a major impediment to embedding commercial analytics tools.

   a. **Authentication.** Does the tool support single sign-on? Or does it require a separate log-on and security framework? If it has its own framework, is it required or optional? If required, how easy

is it to map between security systems? Does it support authentication standards such as SAML, Open ID Connect (OIDC), oAuth, JSON Web Token (JWT), and so on?

b. **Authorization.** Roles and rights established in the parent application are passed to the analytics application to ensure end users are granted the appropriate levels of access.

c. **Application security.** How robust is the tool's permissioning structure? Do administrators have fine-grain control over what content users can see and what functions they can apply?

d. **Security.** Does the tool control access to data sources, tables, columns, and rows?

7. **Multi-tenancy.** Multi-tenant support is complex and a key issue for ISVs.

   a. **Tokenization.** Can multiple users or tenants access different views of the same report, dashboard, or data model? Does a change in the master propagate to each tenant so changes don't have to be made individually? Tokenization eliminates the need to create and maintain separate reports or data marts for each tenant.

   b. **Tenant isolation.** Can each tenant view its own data, whether it's stored in the same database, separate databases, or separate servers?

   c. **Tenant administration.** Can each tenant administer its own instance and users? Can each tenant add, edit, and delete users and groups? Can they assign permissions for content and functionality to each user and group? This is particularly important if the tenant offers tiered pricing for analytics functionality.

   d. **Global administration.** Can the host monitor and manage all tenant instances in a single view? Can the host upgrade the analytics tool without breaking customizations applied by each tenant?

8. **Administration.** An embedded application creates an administrative headache, since there are now two sets of software that must be managed and synchronized.

   a. **DevOps.** Does the product support check in/out, version control, and capture audit trails of all activity? How easy is it to migrate code from development to test to production? Can it roll back to a prior version easily? Does it sync with the host product's DevOps processes and tools?

   b. **User management.** How do administrators manage users and/or tenants? Can they manage them through the same console as the host application?

   c. **Systems monitoring.** How do administrators handle errors, timeouts, non-responsive dashboards, and so on? Can they manage them through the same console as the host application?

d. **Systems management.** How do administrators configure and manage load balancing, backup, restore, and clusters? Can they manage them through the same console as the host application?

e. **Cloud provisioning.** How do administrators configure the product for private or public clouds and elastically scale the product? Can they manage them through the same console as the host application?

f. **Localization.** What languages, currencies, symbols, date formats, and graphical icons does the product support out of the box? Can the localization be customized to support new languages?

9. **Systems architecture.** What is the systems footprint of the analytics tool? What dependencies does it have? How well does it conform to enterprise architecture and data center specifications that govern the host application?

   a. **Systems requirements.** How much RAM does the tool require? What types of hardware are needed to power it? How much disk space is needed for the tool itself as well as its data and metadata? What kind of hardware is required to power the tool for your workloads?

   b. **Application server.** Does the product require its own proprietary application server? Or can it run as JAR files or DLLs on the host application server or on a separate (non-proprietary) application server, if required for scaling reasons?

   c. **Database server.** Does the product require the installation of a proprietary database? Does it require a database to store metadata and/or application data? Can it work without its own database?

   d. **Desktop tool.** Does the product require a desktop tool to author reports and dashboards, design data models, or execute data loads?

   e. **Mobile app.** Does it require users to download a mobile app to view content via a smartphone or tablet device?

   f. **IDE.** Does the product come with its own integrated development environment, specify a third-party IDE (e.g., Microsoft Visual Studio), or can developers use their own and access the tool's functionality through a plug-in or extension?

   g. **Language frameworks.** Does the tool require the use of a language-specific framework, such as .NET, to perform custom development?

10. **Vendor.** Vendors have different levels of experience working with embedded analytics customers.

a. **Ratio.** What is the vendor's ratio of OEM customers to regular customers? How many OEM customers in total?

b. **Viability.** How financially stable is the vendor? What are its annual revenues? How many customers does it have? How many years has it been in the BI market? What are its cash reserves and investment history?

c. **Pricing.** For ISVs, usage-based pricing based on number of users, queries, or CPU cycles is hard to estimate and could generate large unplanned expenditures. The vendor needs to offer flexible pricing and work with each customer based on their requirements. Ideally, customers can choose between perpetual and subscription, and per-core, per-application, or unlimited usage. Some also negotiate a fee for every application license sold. Another approach growing in popularity is to price based on data volume.

d. **Support.** Does the vendor offer services and/or guidelines for how to sell, market, price, support, roll out the product in a commercial setting? Whether you simply need to augment your staff with a consultant or require a whole team to complete a large scope of work, evaluate the range of professional services offered and the extent of the partner network. Some things to consider:

    i. **Onboarding.** Look for a quick-start program that quickly ramps up your team on the solution, helps you identify and coordinate internal resources to support a go-to-market strategy, and sets milestones for completing each phase in the project plan.

    ii. **Account Management.** Vendors should assign account managers who are vested in your success, update you on the latest product news and trends, and coordinate vendor resources to address your issues and requests.

    iii. **Support.** A combination of live and self-service support options backed by experienced professionals should be available to help you work through any technical question. Service-level agreements should clearly set expectations for response times.

    iv. **Community.** An active user community can lend peer support and share valuable best practices so you can benefit from the experience of others.

11. **Analytics.** Finally, it's important to know the type of analytics the product offers. It's best to examine the vendor's heritage to understand its strengths and weaknesses as an analytical tool. Vendors tend to be strong in one of the following categories:

a. **Standard reporting.** Page-oriented, printable reports.

b. **Dashboards.** Interactive dashboards with strong visualizations.

c. **Ad hoc reports.** A self-service environment that enables users to create or edit reports using a business semantic layer.

d. **Visual analytics.** A self-service environment where users can connect to data sources, blend and normalize data sets, visualize and analyze the results, and share with others.

e. **Advanced analytics.** These workbenches and libraries enable data scientists to find and prepare data, create analytical models, and share them.

12. **Software architecture.** Does the analytics tool conform to a modern software architecture?

a. **API design.** Was the tool built using its own API? Does it have a separate front end that uses the API to call functions on the back end?

b. **Front end.** Does it use JavaScript frameworks rather than pure HTML with JavaScript or iFrames? Does it offer programming framework for the language of your choice? How easy is the framework to use?

c. **Back end.** Does it generate JSON and respond to REST calls?

About Eckerson Group

Wayne Eckerson, a globally-known author, speaker, and consultant, formed **Eckerson Group** to help organizations get more value from data and analytics. His goal is to provide organizations with expert guidance during every step of their data and analytics journey.

Eckerson Group helps organizations in three ways:

- **Our thought leaders** publish practical, compelling content that keeps data analytics leaders abreast of the latest trends, techniques, and tools in the field.
- **Our consultants** listen carefully, think deeply, and craft tailored solutions that translate business requirements into compelling strategies and solutions.
- **Our advisors** provide one-on-one coaching and mentoring to data leaders and help software vendors develop go-to-market strategies.

Eckerson Group is a global research and consulting firm that focuses solely on data and analytics. Our experts specialize in data governance, self-service analytics, data architecture, data science, data management, and business intelligence.

Our clients say we are hard-working, insightful, and humble. It all stems from our love of data and our desire to help organizations turn insights into action. We are a family of continuous learners, interpreting the world of data and analytics for you.

Get more value from your data. Put an expert on your side. **Learn what Eckerson Group can do for you!**

Eckerson Group

RESEARCH CONSULTING ADVISORY

GET • MORE • VALUE • FROM • YOUR • DATA

About the Sponsor

Qlik transforms complex data landscapes into actionable insights, driving strategic business outcomes. Serving over 40,000 global customers, our portfolio leverages advanced, enterprise-grade AI/ML and pervasive data quality. We excel in data integration and governance, offering comprehensive solutions that work with diverse data sources. Intuitive and real-time analytics from Qlik uncover hidden patterns, empowering teams to address complex challenges and seize new opportunities. Our AI/ML tools, both practical and scalable, lead to better decisions, faster. As strategic partners, our platform-agnostic technology and expertise make our customers more competitive.

Learn more at **qlik.com**.