Business Analytics

Project Presentation

Context

When I first joined Elekta, my initial responsibility was to maintain and extend an existing SaaS revenue dashboard built by previous analysts. The dashboard prioritized reporting, with several pages primarily visualizing static Excel reports. While this approach worked for surface-level reporting, it offered limited analytical depth when looking to analyze margins.

Even though the data could be manually imported, and margins could be easily calculated, it quickly became clear that the results were difficult to trust. The underlying sales and cost data suffered from many-to-many relationships, inconsistent grain, and manual overrides, which made deeper analysis unreliable even when the data appeared “clean.”

The challenge was not visualization, it was trust, consistency, and shared understanding of the data.

The Real Problem:

Everyone Had a different Number, No One could answer why.

After meeting with each team member, it was clear that, simple questions about financial health often produced different answers depending on which spreadsheet or report was referenced.

Consequently, the Finance lead asked whether the cloud cost data could be cleaned up to support better decision-making.

Several meetings were spent trying to understand what the data meant rather than acting on insights.

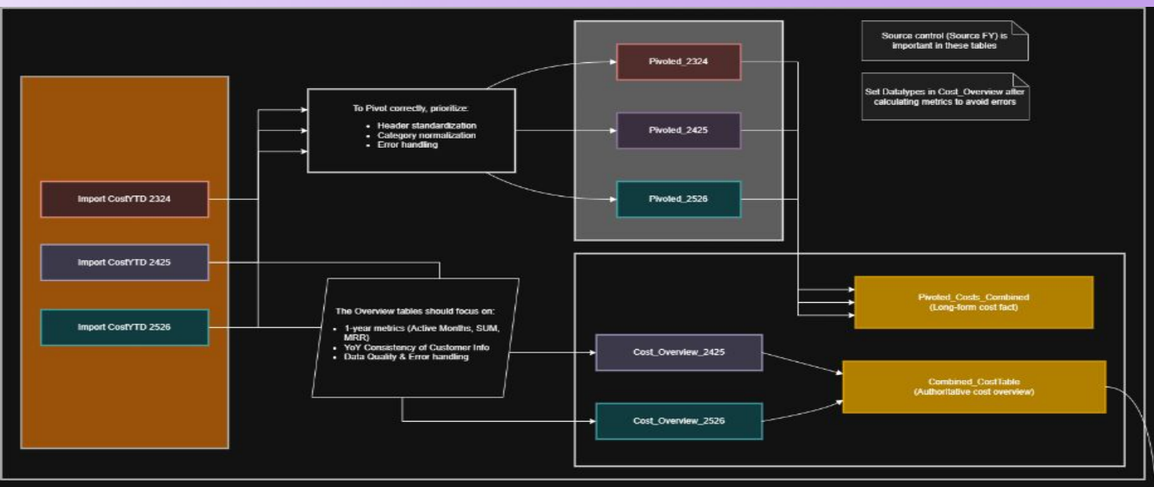
The core issue was not just data quality, but the absence of intention, ownership, and transparency in how we move data.

Product:

After testing, I built a dataflow centered on a shared, explainable data pipeline rather than a single report visual. The goal was to absorb complexity upstream to provide downstream users with consistent, contextual outputs.

To ensure each team benefitted from the pipeline, it reflected sales-defined revenue structures, finance-driven cost concerns, and leadership's need for clarity and auditability.

This allowed visual tools like the Margin Dashboard to work with prepped data, enabling exploratory analysis, not just reporting.



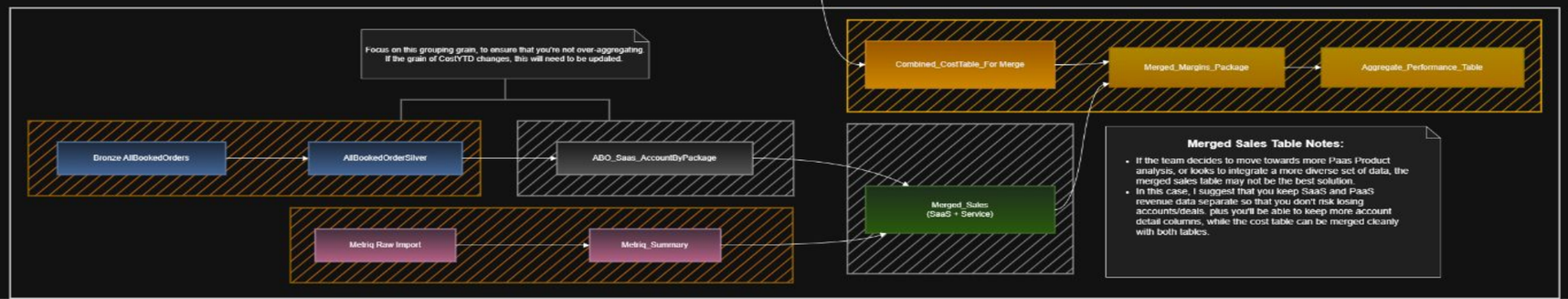


Table	
1	Value 1
2	Value 2
3	Value 3

Prepared Reports for Managers

- Product leaders and Finance Managers can now rely on multiple pre-prepared reports to analyze product health & performance.
- Annual CostTable_XXXX reports from the cloud cost dataflow help align teams on cloud costs at the account level, resulting in better alignment and stronger validation.
- Currently, Merged_Margins_Package is for finance and cloud managers to analyze Cloud margins, and can easily be filtered for product-specific views.

Dashboard: Margin Analysis

- This dashboard uses the data flow and transformations to calculate margin.
- The composite Merged_Margins_Package table is used as an overview table in the dashboard, mainly focused on customer data & cross-year comparison.
- Cost_Overview & Merged_Sales drives the calculations, with measures focused on sales analysis & dynamic filters.
- Simulation & Performance Analysis comes from disconnected tables (TargetPrice) & Supplemental parameters.

Dataflows Design Section

1 Dataflow Patterns

Successfully absorb upstream instability

2 Design Cues

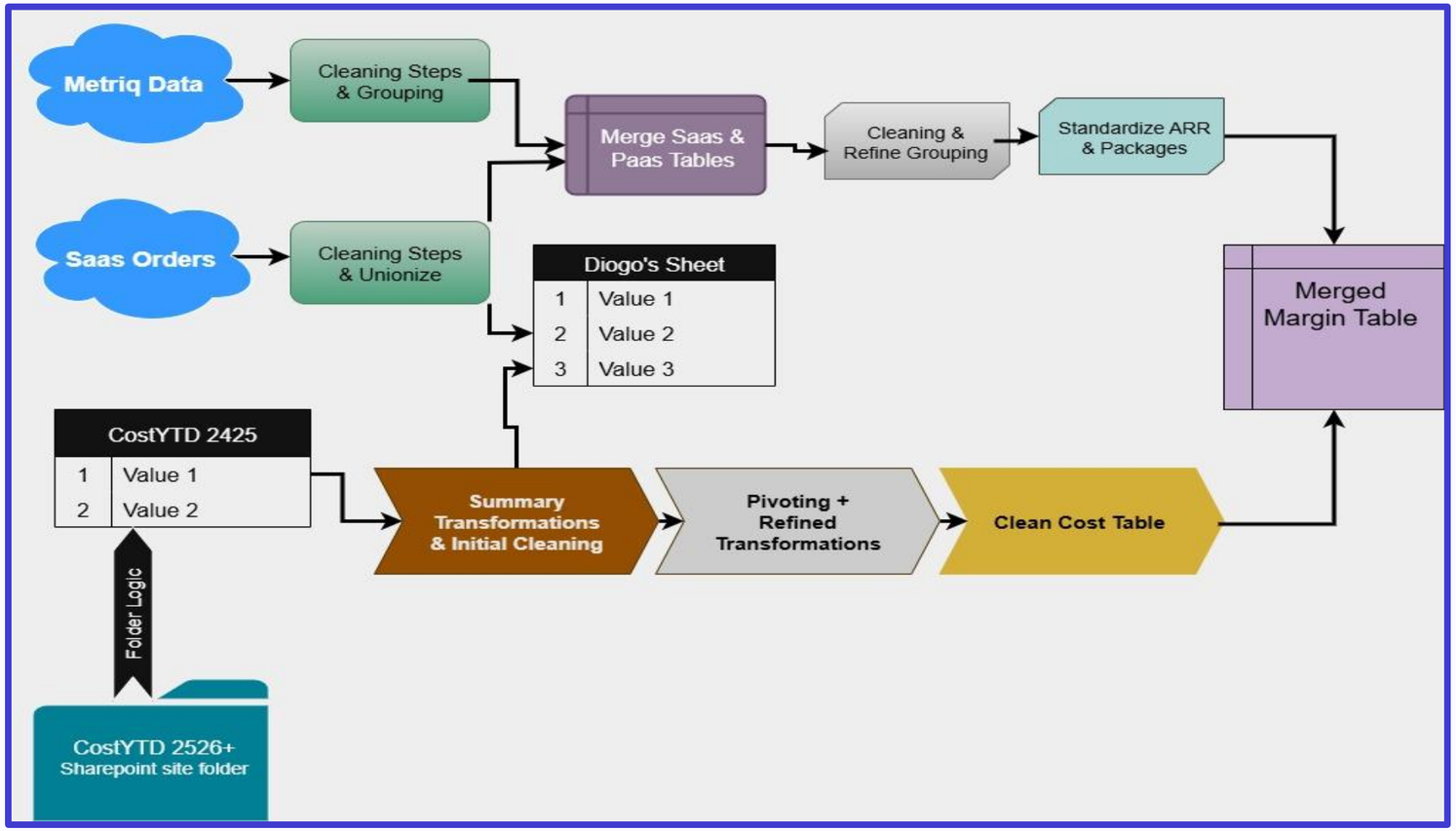
Preserve in supervised pipeline

3 Dashboard Outputs

Power margin reporting today

4 Pipeline Potential

Gold outputs + validation tables



This deck captures the design cues and considerations from my current Gen1 Power BI dataflows before the team replaces them with a supervised pipeline (e.g., Airbyte + Azure-native orchestration). The goal is not to defend Power BI dataflows as the final system, but to document the structure and decisions that made the current prototype reliable enough to support margin reporting.

What the Current Dataflows Demonstrate

TODAY'S WORKING PROTOTYPE

The current dataflows prove that even when cloud cost data is wide, inconsistent, and unstable, we can still produce clean reporting tables by enforcing strict normalization and shaping steps early. This design emphasizes header standardization, category normalization, and error handling upfront, because issues like inconsistent Excel headers, typos, missing months, and formula errors will otherwise break downstream transformations or silently skew results.

> The core value of this approach is that it absorbs upstream instability once, rather than requiring every downstream user or dashboard to build defensive logic around the same problems.

Normalize Early

Cloud cost data is naturally inconsistent across files and fiscal years, so the ingestion layer must normalize structure early.

Sequential & Auditable

Transformations should be sequential and auditable, so refresh failures show up quickly instead of propagating incorrect results.

Narrow & Keyed

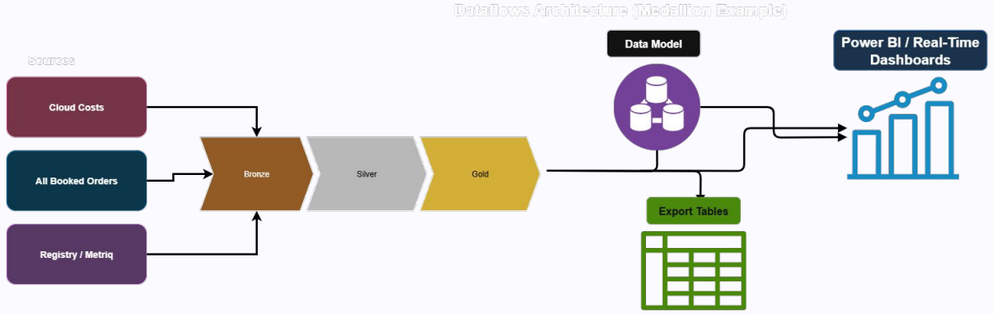
Reporting tables should be narrow, keyed, and consistent so they can be unioned across years and analyzed as time series.

Pipeline Design Cues to Carry Forward

The supervised pipeline should preserve the same design cues used in the dataflows, because these are what make the outputs reliable and explainable. The ingestion logic should remain intentionally strict and fail early rather than letting incomplete or inconsistent data propagate downstream.

Core Design Principles

- Ingestion should be strict and intentional, and ideally parameterized for scalable file selection, so that validation can be in stages.
- Header standardization and category normalization must happen before pivoting or merging
- Design the transformations based on Excel & Salesforce reports that teams are already building.
- Narrow pivoted tables make it easier to union across years and run time series analysis consistently
- Grain control must be explicit for merges between costs and sales, and should not be left to dashboard visuals
- **The data model should be defined within the data and pipeline outputs, not inside the dashboard.**



Key Principle: Fail early, normalize early, keep tables narrow + keyed.

Ingest

Transform

Distribute (Gold)

Pipeline Opportunity: Gold Outputs + Validation Tables

An ETL pipeline with a strong database/warehouse, enable complex performance and validation tables that go beyond what was practical in Power BI Gen1 dataflows. These tables are designed to make the data model scalable, auditable, and worth developing across multiple reporting surfaces without requiring manual interpretation. The goal is to produce outputs that are valid and clean, while still providing the necessary context for downstream users to understand when numbers are accurate versus when they require additional judgment.

Customer

Authoritative Customer Spine: The Customer table should act as a clean customer spine that stays stable across years, so repeated customer records don't drift in naming or identity fields. With repeating years of customer data, it may be more clean to build this table from Salesforce so that it can be monitored, coupled with account identity data pulled from Salesforce.

Validate

Continuous Validation + Overrides: Validation and constant validation is absolutely necessary for this type of cost + sales + margin data. The Validate output should surface problematic accounts with mismatching packages (especially SaaS vs PaaS), inconsistent ARR/cost values, and accounts flagged by downstream users as incorrect or questionable.

Margin Context

Downstream Explainability: The Margin Context table is for downstream users to understand where margin numbers come from and to make mismatches obvious. If product/package context from the sales side and cost side mismatch, both packages are shown with a warning that the calculation may be off.

Package Performance

Validated Performance Outputs: Package Performance represents the result of validation and context, because performance analysis is only valuable when accounts are clean enough to interpret. With month-to-month analysis of cloud spend, we can identify spikes, drops, and trend patterns that support re-quote targeting.

Operating Model: Validate → Context → Performance

The pipeline operating model should prioritize validation before interpretation, and interpretation before performance reporting. Customer identity should be standardized first so joins remain deterministic and cross-year drift can be detected early. Validate outputs should then surface questionable accounts in a reviewable and editable format so upstream owners can confirm semantics and override issues once.

Customer

Standardize identity

Validate

Surface issues

Margin Context

Expose sources

Package Performance

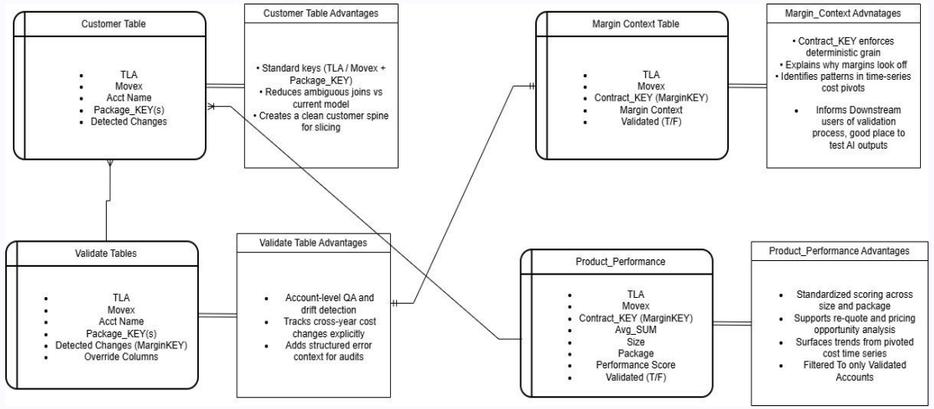
Scalable insights

Dashboards & Ad-Hoc Reports

Final reporting

Validation Example: Drift Detection

If an account is "Beverly Hills Hospital" in 2024 and "Beverly Hills Cancer Center Hospital" in 2025, the validation layer should detect the drift and push the account into a review list instead of silently passing it through. Flagging mismatching calculations will help downstream users stay ahead of incorrect decision-making.



AI Opportunity

There are strong applications for AI here, because AI can generate inline row-level context explaining why an account was flagged and what changed across transformations. AI can also speed up validation by summarizing common error patterns and helping upstream reviewers confirm issues faster. Additionally, giving AI models the business reasoning for technical logic can with new tool integrations for further pipeline development.

DASHBOARD STRATEGY

Dashboard Purpose: Visual Exploration, Not the Source of Truth

The pipeline and data model should define the data, while dashboards remain a downstream tool for visualization and interaction.

This dashboard is designed to make SaaS margin analysis understandable, interactive, and actionable, but it is intentionally not treated as the "source of truth" for the data model. Dashboards are a great visual tool for exploration and executive summaries, but they often create more questions than they answer when teams are trying to resolve simple issues quickly, such as why an account's costs spiked or why margin looks off.

The scalable solution is to keep integration and preparation upstream in the pipeline and database, and use the dashboard as a visual layer built on clean, validated, reporting-ready outputs. When those upstream outputs are governed properly, teams can rely on prepared reports downstream to answer common questions directly, which makes cloud Q&A more streamlined and specific.

Transformations kept Upstream First

Dashboards should be downstream tools, not the focal point of the data

Prepared Reports built into Visuals

Reduces repetitive cloud Q&A and interpretation debates

Trustworthy Model for multiple types of analysis

A stronger data model makes the dashboard more reusable, applicable to more use-cases

DESIGN PRINCIPLES

Dashboard Design Choices: Built from Past Issues with Reporting

This layout and logic is intentional, based on what breaks margin dashboards in practice

I built this dashboard by learning from previous issues with dashboards I have interacted with at Elekta, especially dashboards that attempt to answer too many questions with ambiguous data relationships. One major issue I wanted to reduce is the large amount of many-to-many relationships that typically exist when costs, contracts, accounts, and packages are not modeled cleanly upstream.

The Problem with Many-to-Many

Many-to-many relationships often introduce ambiguity in totals and require heavy workarounds in downstream visuals, which reduces trust and creates inconsistent outcomes between analysts. These relationships make it difficult to establish a single version of truth and force different teams to interpret the same data differently.

In the current version of this dashboard, there are still many-to-many relationships, but they are manageable because the transformations and secondary keys constrain ambiguity enough for reporting.

The Long-Term Solution

If the goal is fully scalable integration, the supervised pipeline should minimize many-to-many relationships by enforcing **deterministic grain**, **stable keys**, and **standardized dimensions**. This upstream modeling approach creates clarity and consistency that cascades throughout all downstream reporting.

Challenge

Many-to-many relationships increase ambiguity and reduce trust in margin calculations

Current State

Transformations can work around this, but require careful management and validation

Future Goal

Deterministic joins and a stronger governed model eliminate ambiguity at the source

PIPELINE LOGIC

ARR for Margin: A Pipeline-Defined Rule That Controls Margin Accuracy

This is not a dashboard calculation choice—it is an upstream alignment decision

ARR for Margin is a conditional that I wrote into the pipeline, and its purpose is to solidify which ARR value should be used in the margin calculation. This logic is driven by defining which ARR values (SaaS vs PaaS) are available in the merged sales table, since the Saas sales tables were unionized into a single framework.

Because the tables were unionized, ARR for Margin becomes a complete list of all available ARR values, which can be compared against the complete list of 1-year cloud cost values. This becomes especially important for accounts that contain both SaaS and PaaS deals, because aligning all ARR rows to all cost rows gives a structured way to select the correct contracts and interpret what the margin is actually representing.

In the new pipeline, you might consider these steps.

Unionize/Prep Sales Tables

Join SaaS and PaaS sales data into a single table with consistent structure, or keep each branch separate with its own transformations.

Align to Cloud Costs

Match complete ARR list against full list of 1-year cloud cost values for accurate margin calculation, using a customer fields to verify contracts.

Define ARR Selection Logic

Create conditional rules to determine which ARR value applies to each account context

Feed Downstream Validation

Connect the Validate + Margin Context tables for row-level reasoning and transparency, while dashboards can have more flexibility for exploration

Priority for the new ETL/Integration Pipeline

Strengthening the ARR for Margin selection function is a strong priority for the new supervised pipeline, because it can become more dynamic and context-aware, and it can directly feed validation and row-level reasoning for downstream users. Currently, there are a lot of defensive filters on the dashboard to ensure that full exploration is possible, with clear suggested values, but a more dynamic ARR selection would be the next step before integrating with other data sources.

ADVANCED ANALYTICS

What-If Analysis for Margin, Performance, and Re-Quote Potential

This is designed to support more than finance reporting, including sales-driven re-quote use cases

One intentional design choice in this dashboard is that it supports not only margin visibility, but also **performance analysis** and **re-quoting decisions**. I haven't consistently heard from other teams that calculate these margins today, so I built a framework that can validate margins and also help teams explore margin sensitivity under different assumptions.

Multi-Team Value

Through what-if analysis tools like the ARR Factor slider, dynamic cost selection across fiscal-year baselines, and simulation drivers like Base Cost and Patient Count, users can test how margins behave when contract context or cost assumptions change.
This makes the dashboard more applicable than just for finance or cloud teams, because sales teams can use it to estimate re-quote potential and understand whether a margin issue is driven by pricing tolerance, package alignment, or cost growth patterns.

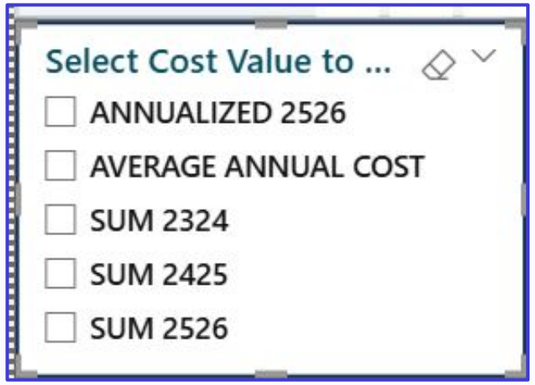
Foundation Matters

These controls are most valuable when they are backed by a strong upstream data model, because the output becomes explainable and consistent across teams. Without clean pipeline outputs, what-if scenarios can produce misleading results that undermine trust in the analysis.

What-If Controls Available in the Dashboard

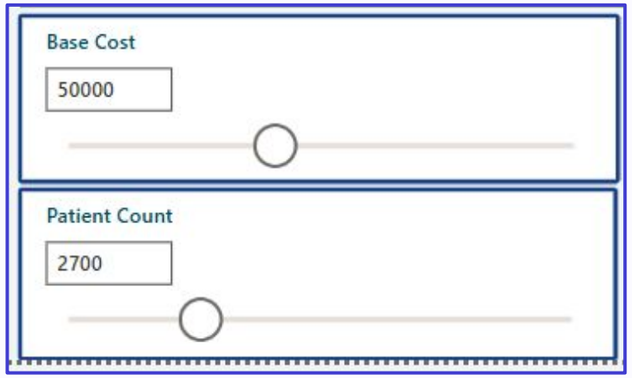
ARR Factor Slider

Simulated ARR increase for scenario testing and contract renewal modeling



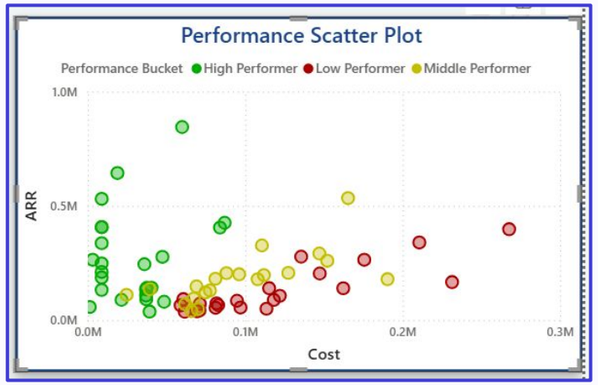
Cost Value Selection

Choose between FY totals, annualized costs, or average cost baseline



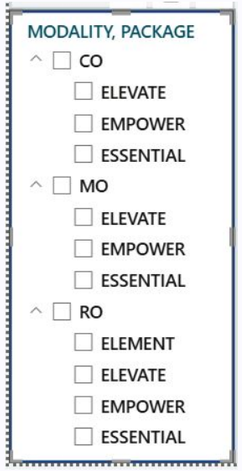
Base Cost & Patient Count

Simulation inputs for sensitivity analysis and cost driver exploration



Performance Buckets

High/middle/low performer segmentation for comparative analysis



Package & Modality Selection

Context filters for simulation scenarios and contract alignment testing

Package & Modality Selection

Context filters for simulation scenarios and contract alignment testing

GOVERNANCE MODEL

Governance Model: Prepared Reports First, Dashboards Second

Clean pipeline outputs reduce friction and keep cloud Q&A specific and repeatable

A stronger long-term approach is to leave dashboards to downstream analysts to build, while ensuring they all connect to the same governed pipeline outputs. When the pipeline produces validated, reporting-ready tables, teams can standardize definitions and avoid the situation where each analyst or manager maintains their own version of margin logic.

Why Governance Matters

Governance becomes even more important as additional products and sources are integrated, because the main drivers of the pipeline—cloud costs and SaaS integration—must remain stable over time.

If new products require ownership from other teams, it may be best to support separate pipelines or controlled branches so experimentation does not disrupt core reporting. This is one reason Power Query-style tooling felt helpful, because teams familiar with Excel can build transformations quickly.

Prepared Reporting Strategy

The goal is to create prepared reporting outputs that answer simple recurring questions directly, and then use dashboards as an exploration layer rather than the primary troubleshooting surface. This shift reduces the burden on analysts and creates consistency across the organization.

If the organization fully moves upstream into a custom pipeline, these integrations and branches will require active management and clear ownership structures.

Examples of Prepared Downstream Reports That Streamline Q&A

Package Mismatches

Accounts with mismatching packages between SaaS and PaaS configurations

ARR-to-Cost Alignment

Accounts with inconsistent ARR-to-cost alignment requiring investigation

Spend Spikes

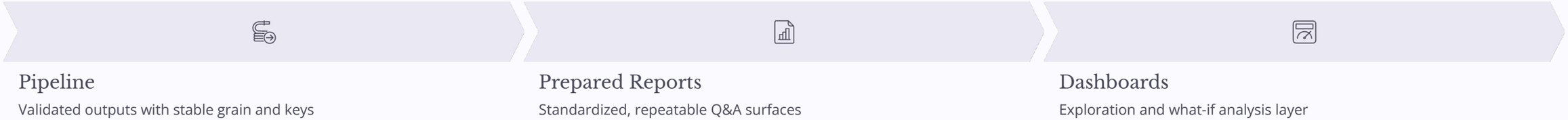
Month-to-month spend spikes and anomaly detection for cost monitoring

Re-Quote Reviews

Accounts needing re-quote review based on margin trend patterns

Flagged Accounts

Accounts flagged by downstream users with resolution status tracking



Managing New Product Integrations

Consider controlled branches for new product experimentation to protect core reporting stability while enabling innovation. This allows teams to test new data sources without disrupting established governance frameworks.

Region
RAMER

DELIVERY
AXIS

MODALITY
RO

PACKAGE
ELEVATE

PackageType
SAAS

Choice
SUM 2425

Available Accounts
Search
5D Clinics
Ackerman Cancer Center
Beaumont
Beverly Oncology & Imaging Center - Montebe...

650,000.02
Sum of TCV

122,886.66
Total Contract Cost ($)

81.1%
Total Contract Margin

110,682.57
ARR for Margin

24,577.33
Selected_Cost

77.8%
Gross Margin

0.00
Opportunity Total Cost

96,587.16
Total One Time Net

96.59K
OneTimeDiff

Powered Off
First Cost.Status

AXIS
First DELIVERY

RO
First MODALITY

AccountName	Customer Type	MarginContext
Beverly Oncology & Imaging Center - Montebello	SaaS	Packages match. Margin is aligned.

Beverly Oncology & Imaging Center - Montebello

13951	BOI	SaaS	none	29.00	ELEVATE
Movex	TLA	Customer Type	Size	Total Active Mo...	PACKAGE

AccountName

Search

Clear all slicers

- [ ] 2406 Cancer Care, LLC
- [ ] 5D Clinics

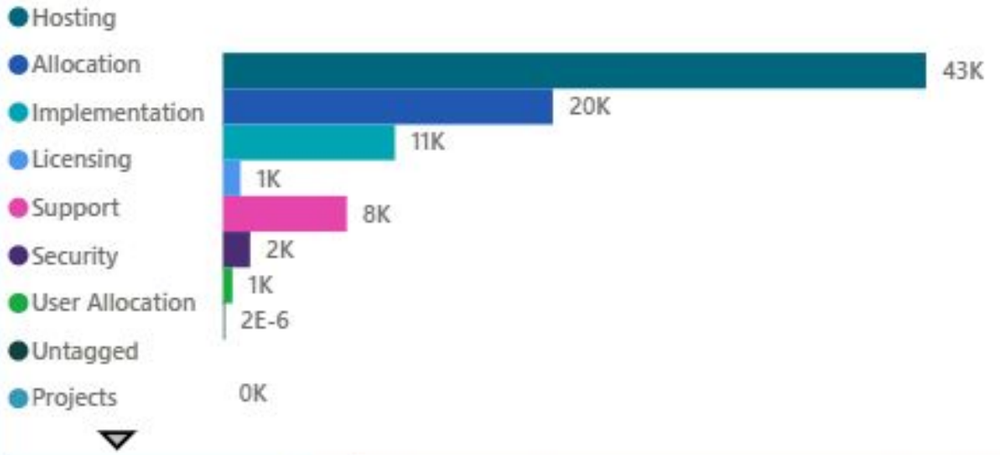
Available ARR's

SAAS

Check Package Selections

529,920	90,882.00	0.83
Target Price (Tota...	Simulated Cost (R...	Simulated Margin (%)
110,682.57	(Blank)	1.00
Dynamic_ARR	Selected_Cost	Dynamic Margin

Category Totals



Select Cost Value to Use

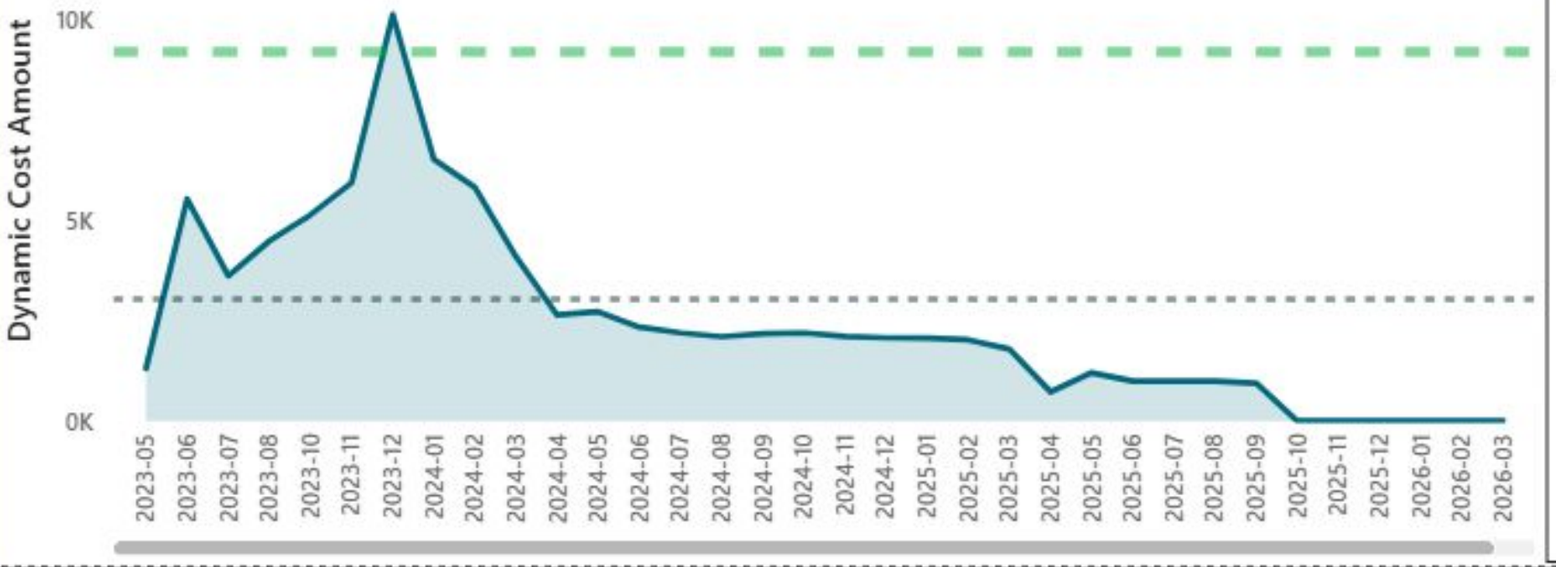
- [ ] ANNUALIZED 2526
- [ ] AVERAGE ANNUAL COST
- [ ] SUM 2324
- [ ] SUM 2425
- [ ] SUM 2526

Arr Factor

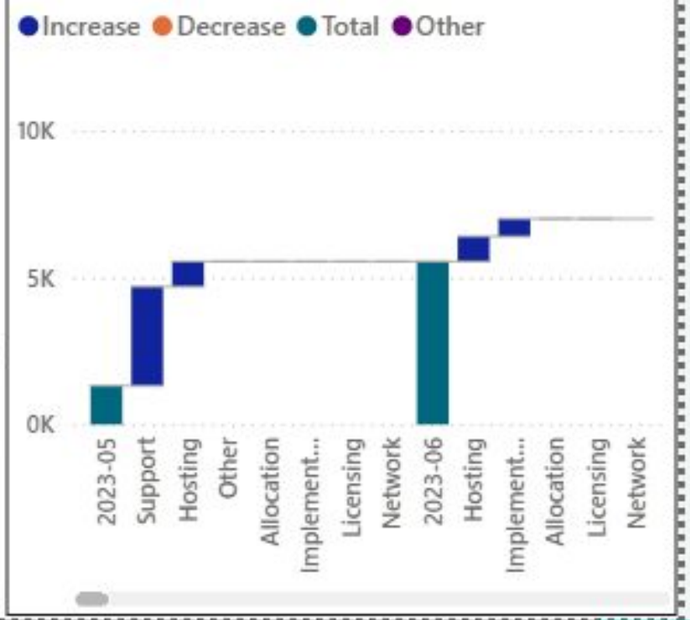
Cost Cate...

- [ ] Allocation
- [ ] Hosting
- [ ] Implementati.
- [ ] Licensing
- [ ] Network
- [ ] Other Projects
- [ ] Projects
- [ ] Security Costs
- [ ] Support
- [ ] Untagged -- ...

FY2324 | FY2425 | FY2526



Identify Categorical Cost Changes



1. Select the **Package** and **Modality** you'd like to analyze against.
2. Set the **Base Cost** & **Patient Count** to estimate the size of the customer.
3. The **Margin Simulation Table** dynamically takes the average of each Accounts ARR and Cost, so you can use the **Performance Slicer** to filter which accounts are being averaged.
4. To analyze margin of a single account, search and select the right account in the **Available Account List**.

Performance B...
- [ ] High Performer
- [ ] Low Performer
- [ ] Middle Performer

PackageMatch
True

PackageType
SAAS

MODALITY, PACK...
- [ ] CO
  - [ ] ELEVATE
  - [ ] EMPOWER
  - [ ] ESSENTIAL
- [ ] MO
  - [ ] ELEVATE
  - [ ] EMPOWER
  - [ ] ESSENTIAL
- [ ] RO
  - [ ] ELEMENT
  - [ ] ELEVATE
  - [ ] EMPOWER
  - [ ] ESSENTIAL

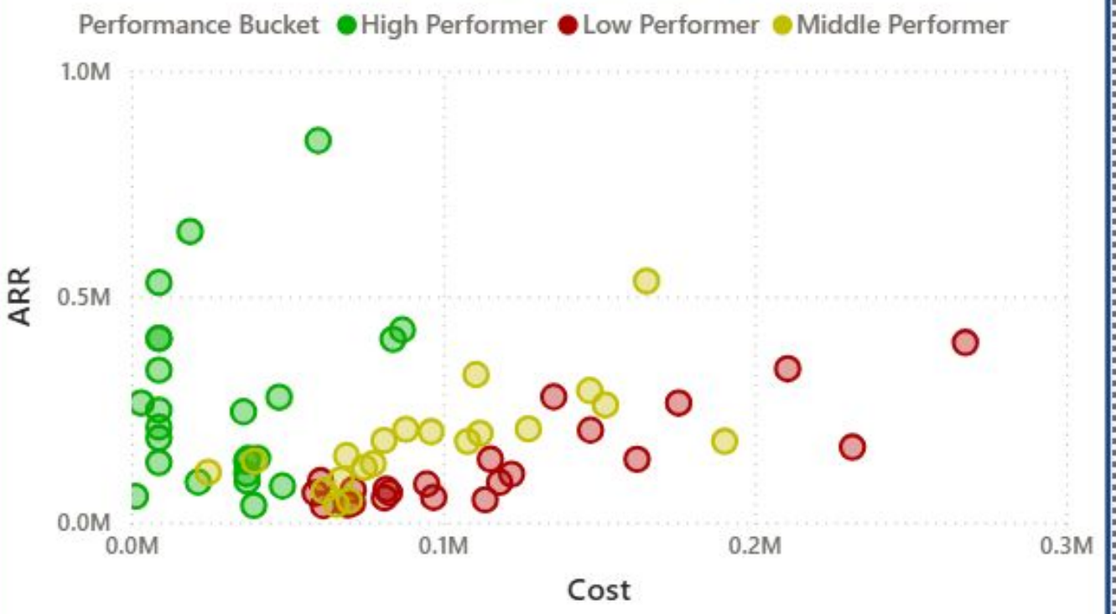
AXIS | NON-AXIS

AccountName
Search
- [ ] 2406 Cancer Care, LLC
- [ ] 5D Clinics
- [ ] Ackerman Cancer Center
- [ ] Alaska Oncology and Hematology LLC

Margin Simulation Table

6,586,160	67,500.00	86.59%
Target Price (Total...	Simulated Cost (R...	Simulated Margin (%)
196,604.77	79,725.14	34.72%
Average of ARR f...	Average of Cost.S...	Average of Margin C...

Performance Scatter Plot



Base Cost
50000

Patient Count
2700

PACKAGE	Annual Profit	Average of Margin Calc	2425_Cost	2425_Cost by Month
EMPOWER	5,039,332.84	0.48	3,070,684.90	
ELEVATE	1,964,085.32	0.55	787,240.51	
ESSENTIAL	755,169.13	0.20	1,065,633.50	
ELEMENT	-44,531.98	-0.24	432,339.61	
Total	7,714,055.32	0.35	5,355,898.51	

Personal Review:

What I Did Well, What I Learned, and What I'd Improve

- **Prototyped early to demonstrate value before scaling**
- **Built multiple dataflows and test transformations to show what was possible**
- **Created supporting materials to explain pipeline-first thinking to non-technical teams**
- **Prioritized transparency and explainability so logic was visible alongside outputs**
- **Shared multiple resources dedicated to understanding best practices and Dataflow potential**

- **Shifted documentation away from day-to-day dataflow maintenance**
- **Focused on preventing recurring issues rather than operating a single tool**
- **Emphasized why design choices mattered, not just how they were implemented**
- **Left behind principles and frameworks intended to outlast my internship term**

Above all else, I learned to take ownership of my work, to be proud of the impact I'm making, while continuously adapting to new problems.

- Sought broader and more consistent feedback earlier in the process
- Navigating the role of being the first analytics intern without technical peers
- Should have researched tooling constraints earlier (licensing, permissions, ownership)
- Learned to treat access and governance as core design requirements

I want to learn what I could've done better from those that have. This project has garnered praise from multiple leaders, but, I want to learn that executive perspective, and how to improve upon it.

Conclusion

I was told early on that Elekta makes it difficult for interns to convert to full-time employees. Early in my internship, I met with a marketing analyst in the BI team, who completed a project so impactful, that the team made an exception.

I wanted to be the Exception.

This project served as a platform for a new mindset in analytics, and one that's quickly proving to be the next frontier of data analytics. Bringing the right data to the right people, with more context & semantics, is a common use case for AI-powered analytics products such as Hex & Databricks, and I’m glad I got to see why those products are thriving.

Thank you for your time