Sigma on Snowflake Best Practices

1 What is Snowflake?

Snowflake's Data Cloud natively connects to dozens of other solutions, turning data into an asset from which you can easily derive valuable insights without the cost, risk, and headache of trying to connect disparate, legacy systems. From data management to data analytics, Snowflake's partnerships and integrations with its technology partners enable customers to leverage the Data Cloud's flexibility, performance, and ease of use to deliver more meaningful data insights. One of these partnerships is with Sigma, an Elite Snowflake Technology Partner for Cloud-Based Analytics and Business Intelligence. Sigma helps enterprise customers to maximize their Snowflake investment by empowering their teams to securely explore, calculate, and analyze billions of rows of live data. This guide has been developed to provide best practices for configuring and optimizing your Sigma instance running on Snowflake.

2 What is Sigma?

Snowflake has changed the game with its powerful hybrid architecture. Analytics platforms no longer need to house and process data to create a presentation layer. They can now rely on Snowflake's Data Cloud, with its near infinite storage and scalable compute layer to perform the calculations necessary to drive business reporting. Sigma was founded on the belief that Modern Analytics solutions should be purpose built to leverage the power, scale, and speed of the most powerful compute engines in the enterprise - Snowflake.

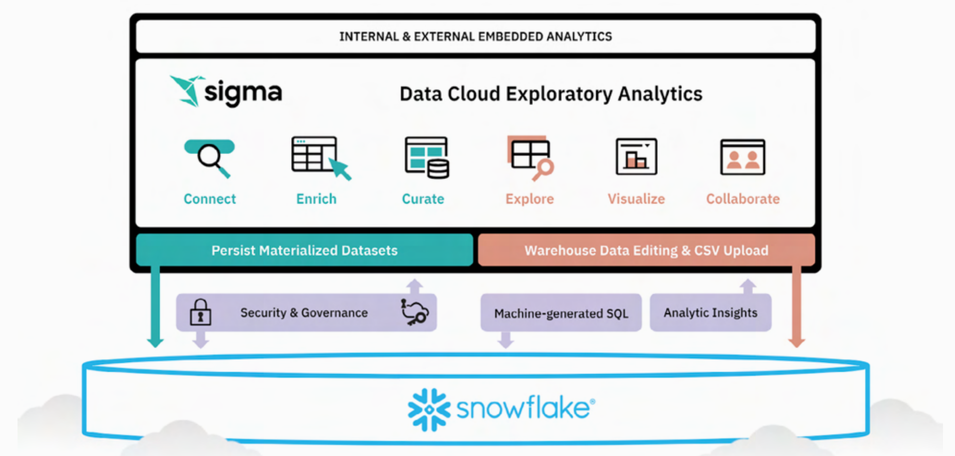
Sigma is a SaaS application that provides a front-end user interface to query, profile, visualize, and explore data stored and shared in Snowflake. Sigma utilizes a direct connection to Snowflake and a proprietary SQL generation engine to translate user interactions on a familiar spreadsheet interface into machine-optimized SQL. This SQL is able to unlock the power of Snowflake virtual warehouses and provide limitless charts, graphs, and pivot tables with a level of flexibility that is not feasible in any other tool. With Sigma on Snowflake, users can query, pivot, transform and visualize data sets of billions of rows of data, all in real-time. Sigma provides users true ad-hoc flexibility to explore their data, with no limitations.

With a familiar spreadsheet interface, Sigma is easy to use for both data teams and business users. The application has two primary objects: Datasets and Workbooks. Datasets act similarly to a database view, providing a mechanism for both data abstraction and curation. Workbooks are the front-end for dashboarding, as well as both guided and ad hoc exploration, all via a familiar Excel-like user interface.

Sigma's organizational paradigm is similar to Google Drive or Microsoft OneDrive, where users are presented with both individual and team workspaces that have folder structures established within them. Sigma prioritizes useability and familiarity to promote the best user experience possible.

As an analytics platform, Sigma aims to revolutionize the way that teams work with their data. Since Sigma does not store any data, it does not require maintaining extracts from source systems or heavy aggregation of data in intermediary tables to optimize performance, both issues legacy BI tools often suffer from. Teams are able to work directly with row-level data available in the cloud data warehouse and aggregate it using Snowflake's virtual compute warehouses. Sigma elevates data and analytics collaboration by allowing users and teams to share workbooks with other users, comment on Workbook elements, and further each other's analysis in real time. Teams no longer need to share Excel sheets with stale data to collaborate, eliminating the massive troves of dark data repositories that plague most companies.

Sigma Computing is the first company to truly deliver on the promise of self-service analytics and business intelligence (A&BI). Inspired by the tabular interface of a spreadsheet, the Sigma interface empowers anyone—from all functions of the data team to business analysts and domain experts—to analyze data, without code or extracts, and make data-backed decisions quickly. Sigma powers an iterative and community-driven approach to A&BI that aligns business goals with data analysis, freeing data teams to focus on more innovative and fulfilling initiatives.


INTERNAL & EXTERNAL EMBEDDED ANALYTICS
sigma
Data Cloud Exploratory Analytics
Connect
Enrich
Curate
Explore
Visualize
Collaborate
Persist Materialized Datasets
Warehouse Data Editing & CSV Upload
Security & Governance
Machine-generated SQL
Analytic Insights
snowflake

This guide will walk you through how to install Sigma on top of Snowflake and optimize both products to unleash their full potential in your organization!

3 Setting Up Snowflake for Sigma

In Sigma, connections are the foundational piece of infrastructure that allow for data from the data cloud to be interacted with. Sigma maintains an index of the catalogs, schemas, tables, and views in any given connection. This Process is automated and reads from the Snowflake metadata in the cloud services layer once nightly. This process is run in the background to ensure that when a user needs to build new content the available assets are presented to them without the need for a manual refresh. Under normal usage this process will result in 0 credit consumption, however in some cases when there is minimal usage in a 24 hour period, credit consumption may occur. To learn more about the cloud services layer read through this Snowflake documentation.

3.1 Warehouse Configuration

Sigma can both read from and write back to Snowflake. To provide a solid foundation for a Sigma instance, these two tasks should be delegated to separate Snowflake virtual warehouses. Depending on your use case, you may have more than one analytics or materialization warehouse.

3.1.1 Analytics Warehouse

Sigma at its core is an analytics platform that seeks to provide an accessible front-end that enables users to harness the power of the Snowflake Data Cloud. These analytics workloads require some tuning in Snowflake to provide a robust experience.

Multi-cluster warehouses should be used for analytics workloads to prevent any unnecessary queuing. The minimum and maximum values for scaling will vary primarily on the number of users expected to be leveraging the Sigma connection or specific workbook at any given time. *Determining how and when to scale out warehouses is covered in detail in section 4.1.4.*

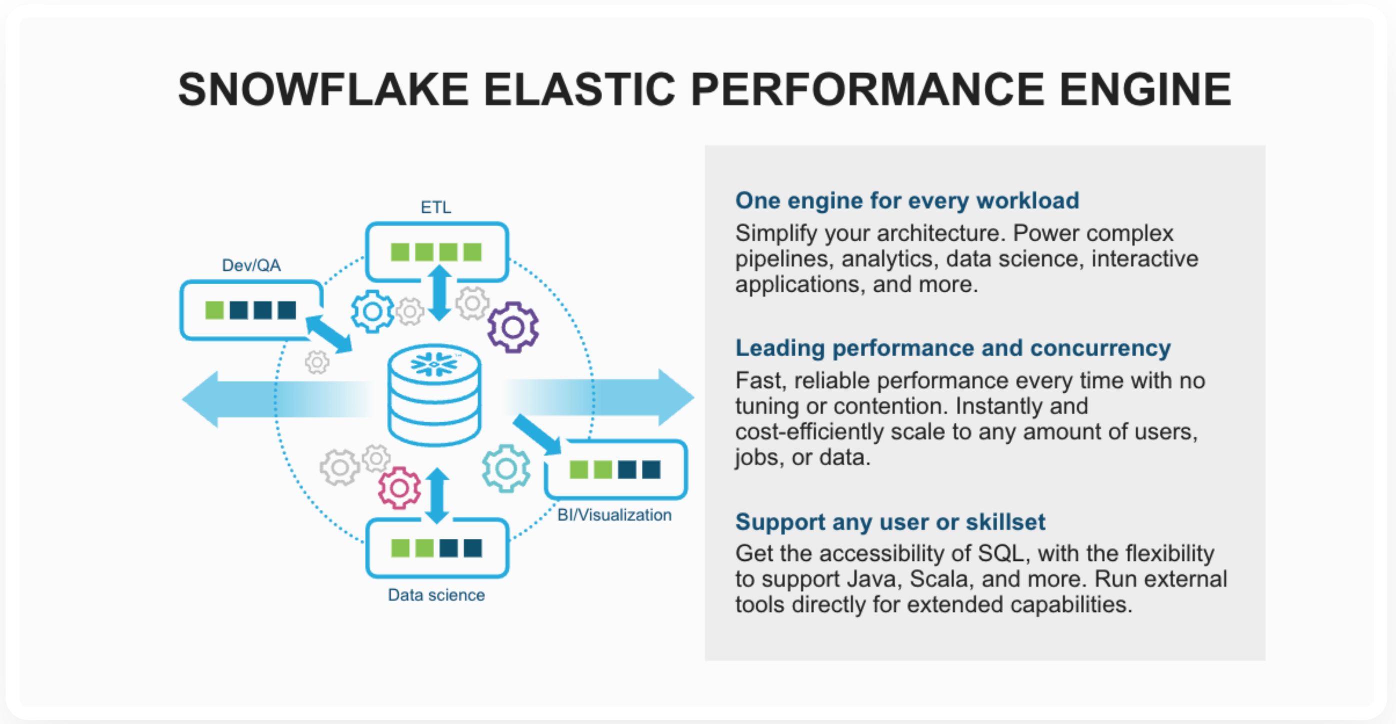
Warehouse sizes for Sigma analytics workloads should be determined by the size of the tables being queried, as well as the complexity of queries. In most cases, simple queries on small tables will only require an XS warehouse. *Determining how and when to scale up warehouses is covered in detail in section 4.1.4.*

Sample code to spin up your Sigma Analytics Warehouse:

```
CREATE WAREHOUSE IF NOT EXISTS sigma_analytics_wh
  WITH WAREHOUSE_SIZE = XSMALL
    MAX_CLUSTER_COUNT = 2
    MIN_CLUSTER_COUNT = 1
    SCALING_POLICY = ECONOMY
    AUTO_SUSPEND = 600
    AUTO_RESUME = TRUE
    INITIALLY_SUSPENDED = TRUE;
```

NOTE

In Sigma the analytics warehouse is by default set at the connection level, however each workbook can be overridden to use any warehouse available to the service account or user if leveraging OAuth. The ability to override the analytics warehouse on a per workbook basis provides administrators the flexibility they need to meet the needs of the business.



SNOWFLAKE ELASTIC PERFORMANCE ENGINE

One engine for every workload
Simplify your architecture. Power complex pipelines, analytics, data science, interactive applications, and more.

Leading performance and concurrency
Fast, reliable performance every time with no tuning or contention. Instantly and cost-efficiently scale to any amount of users, jobs, or data.

Support any user or skillset
Get the accessibility of SQL, with the flexibility to support Java, Scala, and more. Run external tools directly for extended capabilities.

3.1.2 Materialization Warehouse

Sigma Datasets can be materialized in Snowflake to mitigate the compute costs associated with consistently running complex queries during analysis. In most cases, these workloads will be very different than analytics queries: various large datasets will be periodically refreshed throughout the day, and raw compute power is more important than concurrency.

Materialization warehouses often need to be sized larger to accommodate the nature of the complex queries that they will materialize. Snowflake warehouses scale geometrically, with each size being twice as powerful as the previous. This also means that as the size of the warehouse increases, a query will take half as long to complete. To size your materialization warehouse, please follow Snowflake's guidelines on warehouse sizing, covered in Section 4.1.4.

Multi-clustering may not be required when configuring the materialization warehouse. Consider multi-clustering if your organization needs the ability to write back numerous complex materializations at the same time.

Creating separate warehouses for materialization and analytics ensures that Sigma will not run into resource contention across these different workloads, and helps provide the fastest possible user experience.

Sample code to spin up your Sigma Analytics Warehouse:

```
CREATE WAREHOUSE IF NOT EXISTS sigma_materialization_wh
   WITH WAREHOUSE_SIZE = XSMALL
      AUTO_SUSPEND = 60
      AUTO_RESUME = TRUE
      INITIALLY_SUSPENDED = TRUE;
```

3.2 Account Authentication

3.2.1 Service Accounts, OAuth, or Both

Sigma's Snowflake connection can authenticate via either OAuth or a Service Account. Each authentication type comes with unique pros and cons that should be considered when architecting the implementation.

Service Accounts are a great option for many use cases. All users that leverage this connection will be accessing data via the Service Account. In this method, grants must be provided to the Service Account User/Role in Snowflake, and then user-based access is managed and maintained inside Sigma via Datasets (row and column level security) or folder - and document-level permissions. When row-level security is required, Sigma has the flexibility to define and apply a security model via functions that can gather user information and filter result sets as necessary. Additionally, if you're considering using Dataset materialization in your Sigma Environment, Service Account authentication must be used.

A connection using OAuth will provide the same level of access that the end user would have if they logged directly into Snowflake, including any defined Row Access Policies or Dynamic Data Masking. This method allows teams to set policies once in a central location and have them cascade to the Sigma Layer. Please note - dataset materialization is not available when connecting via this OAuth.

There are situations that dictate the need for both authentication types when connecting to Snowflake. Sigma does not limit the number of connections to a given Snowflake account, meaning that two connections could be stood up, one using OAuth and the other using a Service Account. In this case Workbooks and Datasets built off of the OAuth connection would leverage Snowflake security measures that apply to the individual user, while those built off of the Service Account will present the same view to all end users.

A good example when dual connection methods might be used is deploying enterprise wide workbooks. OAuth accounts attached to an individual can be deactivated if there are no logins within a set time period, which may result in workbook connection issues. For this reason, when building out these enterprise wide objects it is recommended to leverage a service account to remove the possibility of a deactivated account causing a service disruption.

3.3 Database and Schema

3.3.1 Writeback Schema

Sigma provides a number of valuable features that require setting up access to a separate writeback schema in Snowflake.

As described earlier, Sigma Datasets can be materialized in Snowflake to mitigate repetitive complex querying. To utilize this functionality, Sigma requires a separate schema to house your materializations on Snowflake and a virtual warehouse to create them. (See discussion of Materialization Warehouse Setup in Section 3.1.2, above).

This writeback schema is also used for CSV files that are uploaded through Sigma. These CSV files are created as tables in Snowflake and can be leveraged as a data source for either Datasets or direct use in Workbooks.

Additionally, Sigma Datasets will be saved as Snowflake views when using a writeback schema, thus making them accessible by any other application or user that has access to Snowflake as well. This can be useful when developing data objects in Sigma that may be beneficial to other tools or teams.

To limit confusion, the writeback schema is not available to browse at the schema level via the Sigma UI. Sigma restricts this schema to only be available via their corresponding dataset objects.

Since Snowflake allows grants to future tables and views, ensure that any access to this schema is controlled via the Snowflake RBAC procedures you have implemented.

3.4 Generated or Custom SQL

At its core, Sigma is a world-class SQL generation engine. The SQL that is created by Sigma and sent to Snowflake is heavily optimized. When a new query is formed, it begins with a simple select statement. As tables are joined in, groupings established, and aggregate calculations put in place, the query grows more complex.

Sigma automatically leverages techniques such as subqueries, CTEs, and caching to write SQL optimized for continued exploration. At any point, the SQL backing the queries can be reviewed via the UI.

Sigma can also accept custom SQL as a starting point to build a query. This may be helpful for SQL-savvy users who desire more flexibility. Sigma's machine generated SQL is highly optimized to take advantage of Snowflake's performance and caching capabilities, and should be used as a default whenever possible. Custom SQL should be considered only in select use cases.

Use Case	Sigma Generated SQL	Custom SQL
Migrating Legacy SQL*	✓	
Data Exploration	✓	
Workbook Creation	✓	
Dynamic Parameters**		✓
Highly Complex Use Cases		✓

* *Migrating SQL one to one is a good way to validate output and move quickly, however, for the best performance when migrating legacy SQL into a Sigma data asset, it is recommended to rebuild the data asset using Sigma's UI.*

** *Custom SQL in Sigma allows for parameters to be passed through, unlocking potential to leverage dynamic filtering.*

4 Optimizing Snowflake

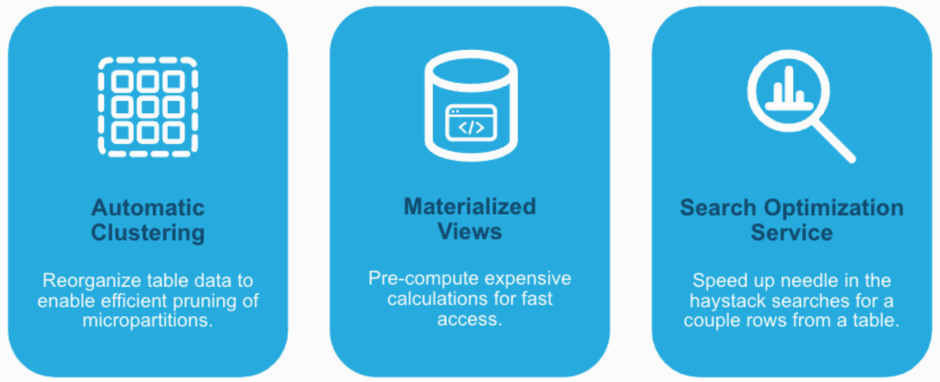
Sigma sits as a query, exploration, and analytics engine on top of Snowflake. Sigma's performance is directly tied to Snowflake's performance, so it is imperative to take advantage of the levers that Snowflake provides to ensure fast results when querying large data sets with high concurrency.

4.1 Levers to Improve Performance

Snowflake is a fully managed service that's simple to use but can power a near-unlimited number of concurrent workloads. It's fully automated so you can trade in manual management and maintenance for automations that enable you to easily operate at scale. The following areas are helpful to keep in mind in order to fully maximize Snowflake's elastic performance engine.

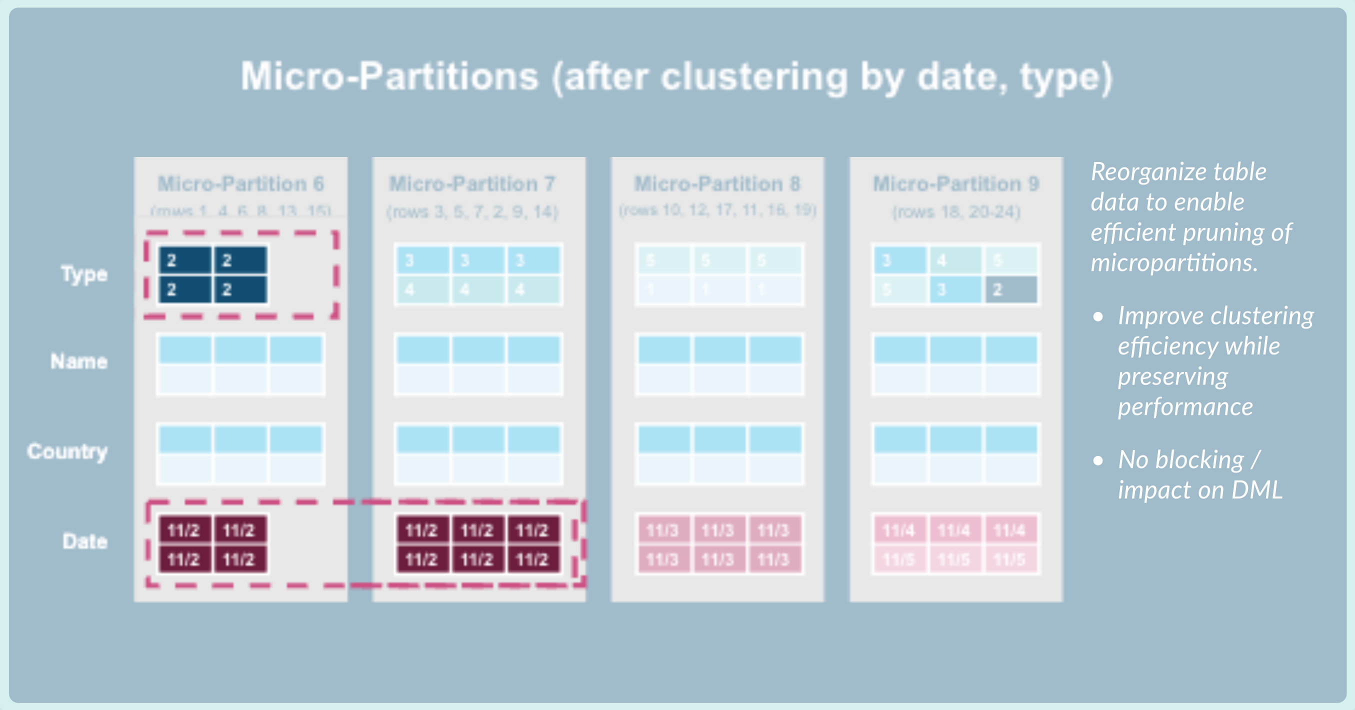
KEY PERFORMANCE SERVICES

Optimize Snowflake for your workloads using these three key services


Automatic Clustering
Reorganize table data to enable efficient pruning of micropartitions.
Materialized Views
Pre-compute expensive calculations for fast access.
Search Optimization Service
Speed up needle in the haystack searches for a couple rows from a table.

AUTO-CLUSTERING SERVICE

Enable more efficient scanning of micropartitions to speed up queries and reduce costs


Micro-Partitions (after clustering by date, type)
Type
Name
Country
Date
Micro-Partition 6
(rows 1, 4, 6, 8, 13, 15)
Micro-Partition 7
(rows 3, 5, 7, 2, 9, 14)
Micro-Partition 8
(rows 10, 12, 17, 11, 16, 19)
Micro-Partition 9
(rows 18, 20-24)
Reorganize table data to enable efficient pruning of micropartitions.
Improve clustering efficiency while preserving performance
No blocking / impact on DML

4.1.1 When to Cluster Tables

All data in Snowflake tables are automatically divided into micro-partitions. Groups of rows in tables are mapped into individual micro-partitions, organized in a columnar fashion. This size and structure allow for extremely granular pruning of very large tables which can be comprised of millions, or even hundreds of millions, of micro-partitions.

Typically, table data is sorted/ordered along natural dimensions (e.g. date and/or geographic regions). This "clustering" is a key factor in queries because table data that is not sorted (or is only partially sorted) may impact query performance, particularly on very large tables.

Queries against very large tables that join or filter on columns that are not well clustered may take longer to run, as the compute resources must scan all of the necessary micro-partitions. In these cases, Snowflake recommends defining one or more cluster keys, after which the Snowflake automatic clustering service will run in the background to co-locate the data in an optimal manner.

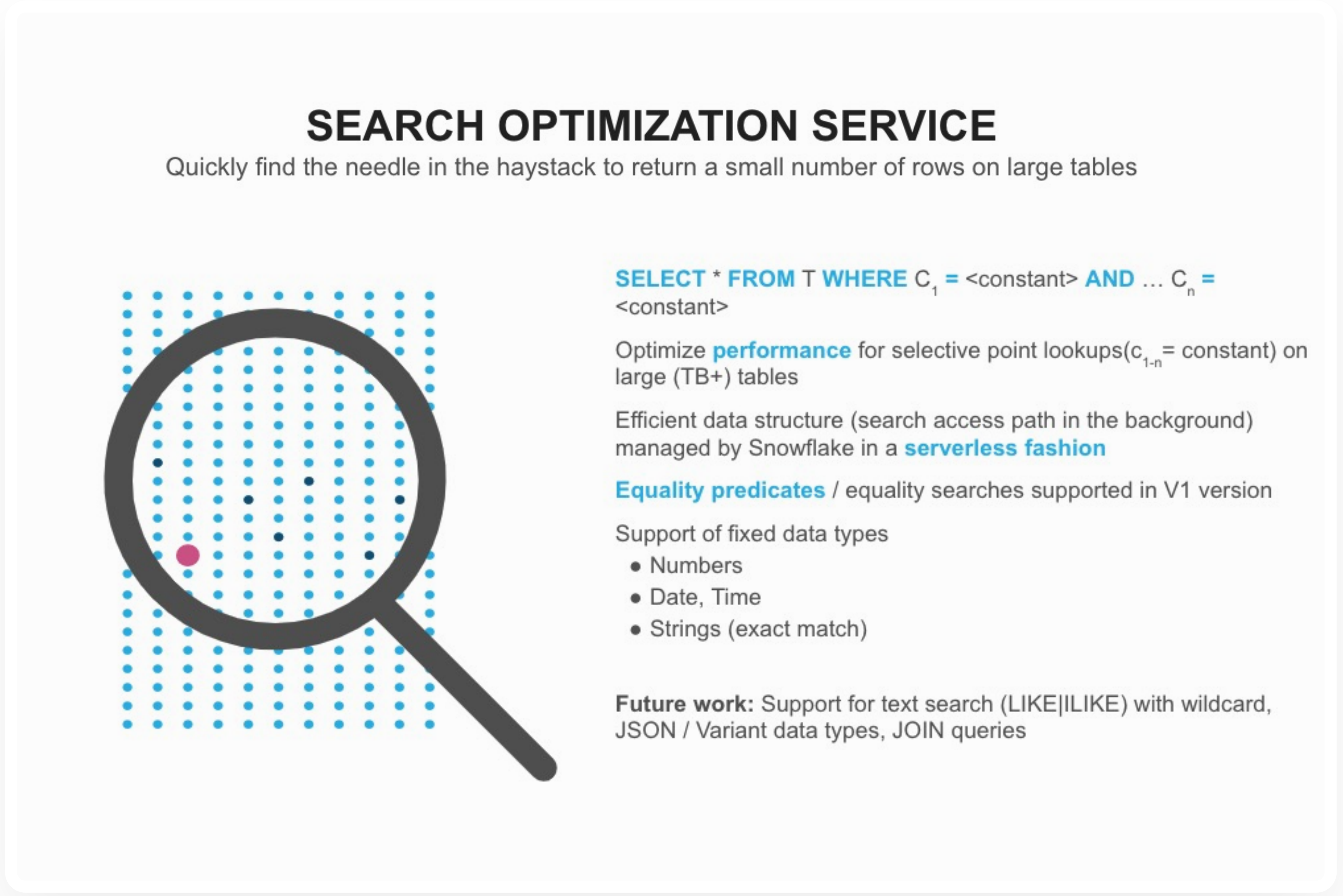
For more information please see https://medium.com/snowflake/introducing-the-snowflake-visual-table-clustering-explorer-6fbb66a15bd5.

4.1.2 When to Use Search Optimization

Snowflake's search optimization service aims to significantly improve the performance of selective "point lookup" queries on tables (queries that return only one or a small number of distinct rows). Users can register one or more tables to the search optimization service. Search optimization is a table-level property and applies to all columns with supported data types.

Search optimization works best to improve the performance of a query when the table being queried is NOT clustered, or the table is frequently queried on columns OTHER than the primary cluster key.

For more information, see https://docs.snowflake.com/en/user-guide/search-optimization-service.html.



SEARCH OPTIMIZATION SERVICE

Quickly find the needle in the haystack to return a small number of rows on large tables

SELECT * FROM T WHERE C_1 = <constant> AND … C_n = <constant>

Optimize **performance** for selective point lookups(c_{1-n}= constant) on large (TB+) tables

Efficient data structure (search access path in the background) managed by Snowflake in a **serverless fashion**

Equality predicates / equality searches supported in V1 version

Support of fixed data types

- Numbers
- Date, Time
- Strings (exact match)

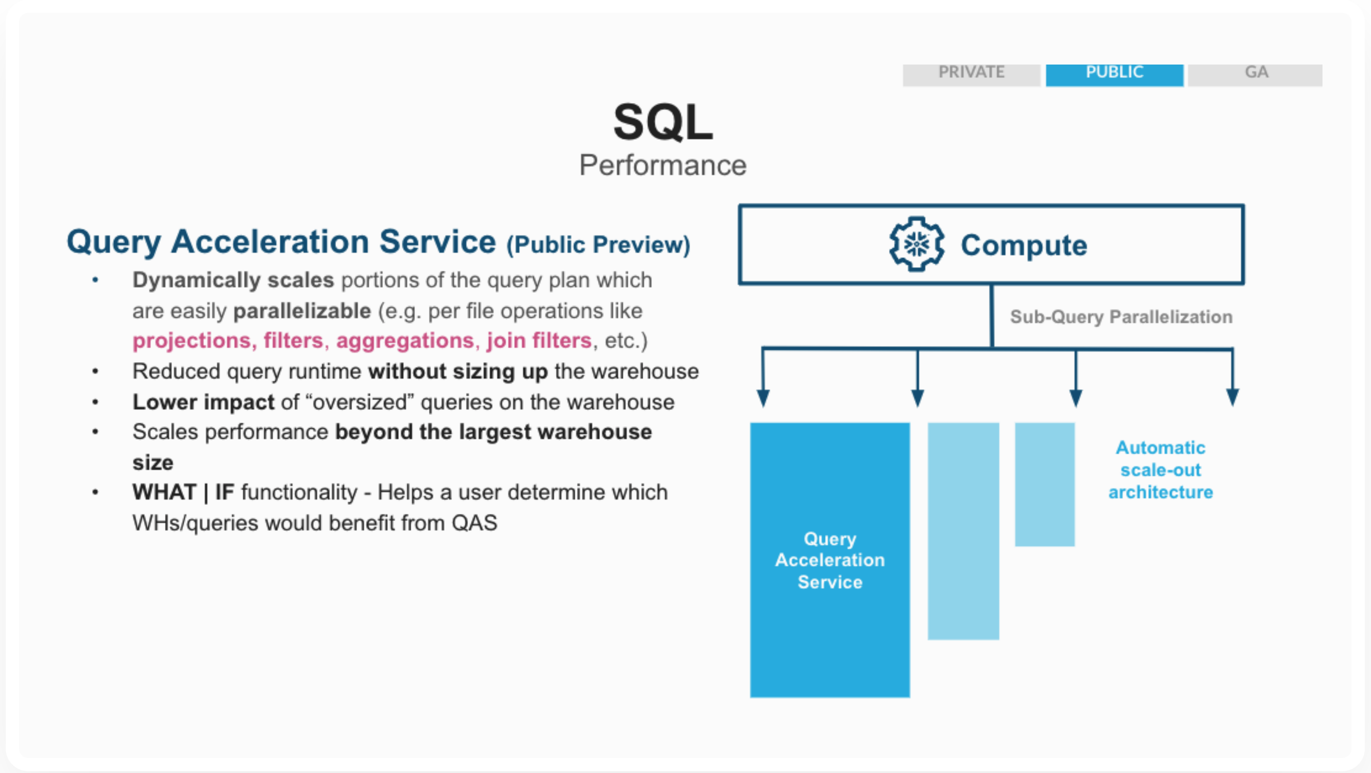
Future work: Support for text search (LIKE|ILIKE) with wildcard, JSON / Variant data types, JOIN queries

4.1.3 When to Use Query Acceleration

Snowflake's query acceleration service (QAS) is enabled per warehouse and can accelerate certain portions of its query workload. It can improve overall warehouse performance by reducing the impact of outlier queries, which are queries that use more resources than the typical query. The query acceleration service does this by *offloading portions of the query processing work to shared compute resources that are provided by the service (the availability of these shared resources may vary throughout the day)*.

Please note that the query acceleration service does not guarantee predictability around performance improvements. In order to equitably benefit queries that can be accelerated, resources might be dynamically re-allocated at any time. Performance improvements can vary for the same query executed at different times.

For more information see https://docs.snowflake.com/en/user-guide/query-acceleration-service.html.



SQL

Performance

Query Acceleration Service (Public Preview)

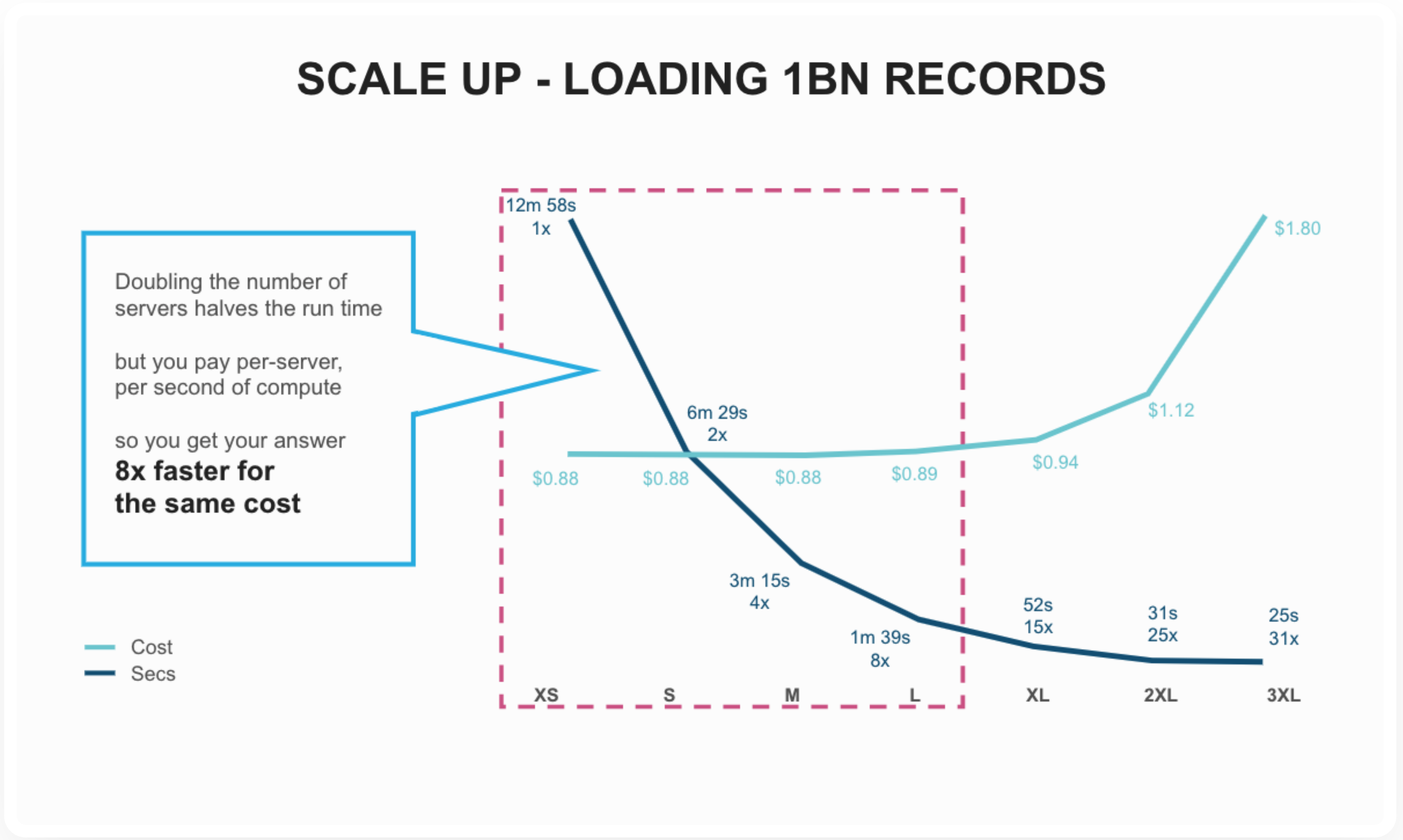
- **Dynamically scales** portions of the query plan which are easily **parallelizable** (e.g. per file operations like **projections, filters, aggregations, join filters**, etc.)
- Reduced query runtime **without sizing up** the warehouse
- **Lower impact** of "oversized" queries on the warehouse
- Scales performance **beyond the largest warehouse size**
- **WHAT | IF** functionality - Helps a user determine which WHs/queries would benefit from QAS

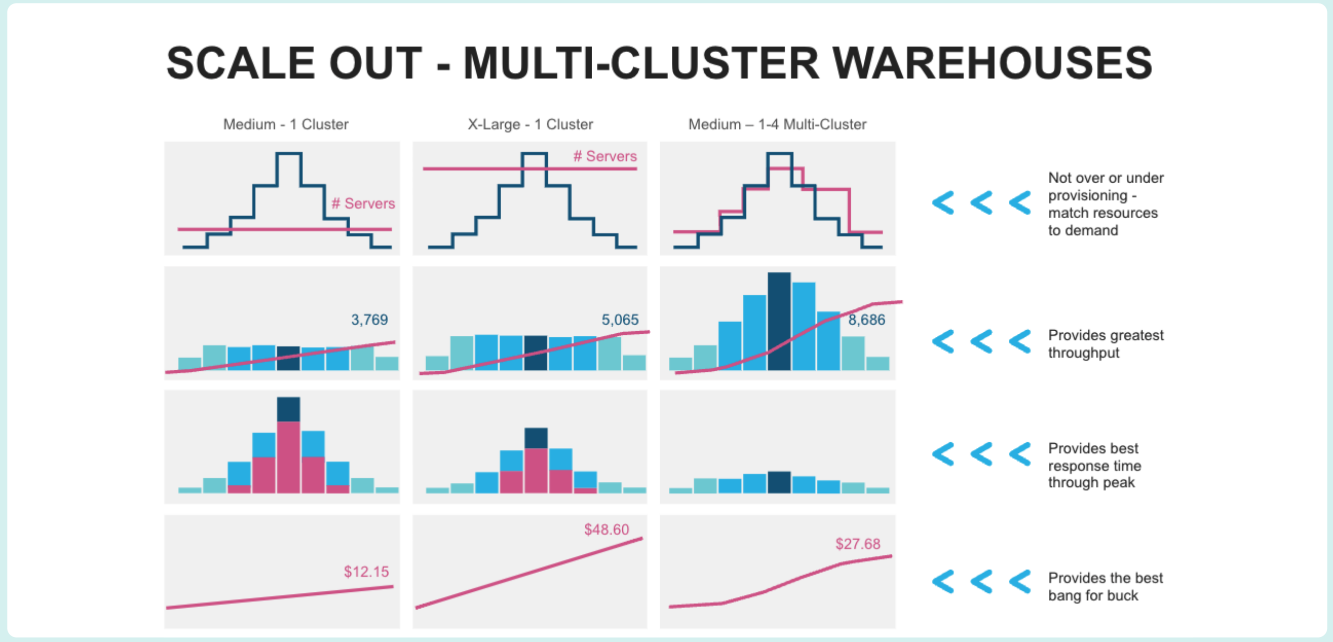
Compute
Sub-Query Parallelization
Query Acceleration Service
Automatic scale-out architecture

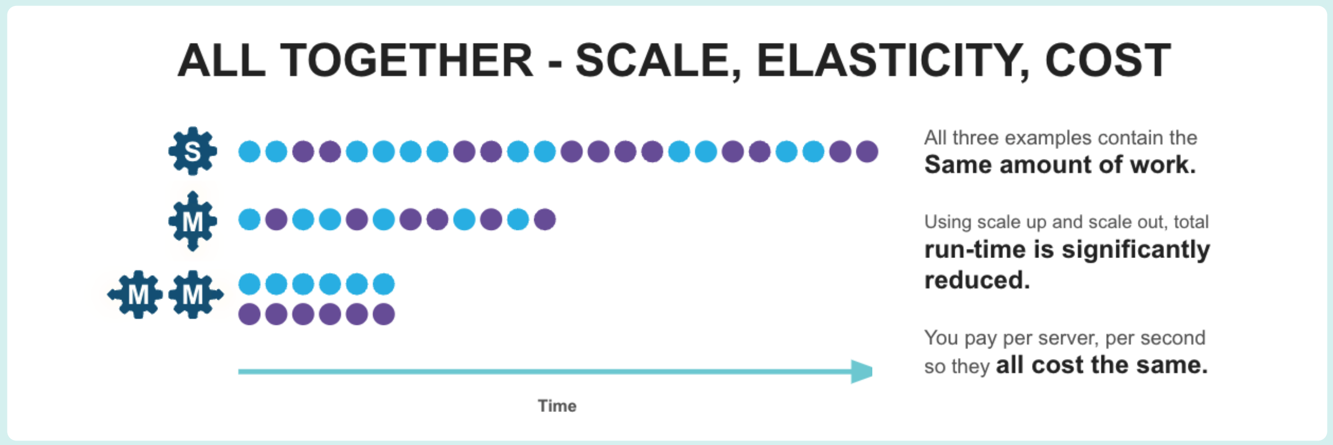
4.1.4 Warehouse Scaling Up vs Scaling Out

Snowflake supports two ways to scale warehouses:

- **Scale up by resizing a warehouse**
- **Scale out by adding warehouses to a multi-cluster warehouse**






ALL TOGETHER - SCALE, ELASTICITY, COST
S
M
M
M
All three examples contain the Same amount of work.
Using scale up and scale out, total run-time is significantly reduced.
You pay per server, per second so they all cost the same.
Time

WAREHOUSE RESIZING IMPROVES PERFORMANCE

Resizing a warehouse generally improves query performance, particularly for larger, more complex queries. It can also help reduce the queuing that occurs if a warehouse does not have enough compute resources to process all the queries that are submitted concurrently. Note that warehouse resizing is not intended for handling concurrency issues; instead, use additional warehouses to handle the workload or use a multi-cluster warehouse (if this feature is available for your account).

MULTI-CLUSTER WAREHOUSES IMPROVE CONCURRENCY

Multi-cluster warehouses are designed specifically for handling queuing and performance issues related to large numbers of concurrent users and/or queries. In addition, multi-cluster warehouses can help automate this process if your number of users/queries tend to fluctuate.

For more information see https://docs.snowflake.com/en/user-guide/warehouses-considerations.html#scaling-up-vs-scaling-out.

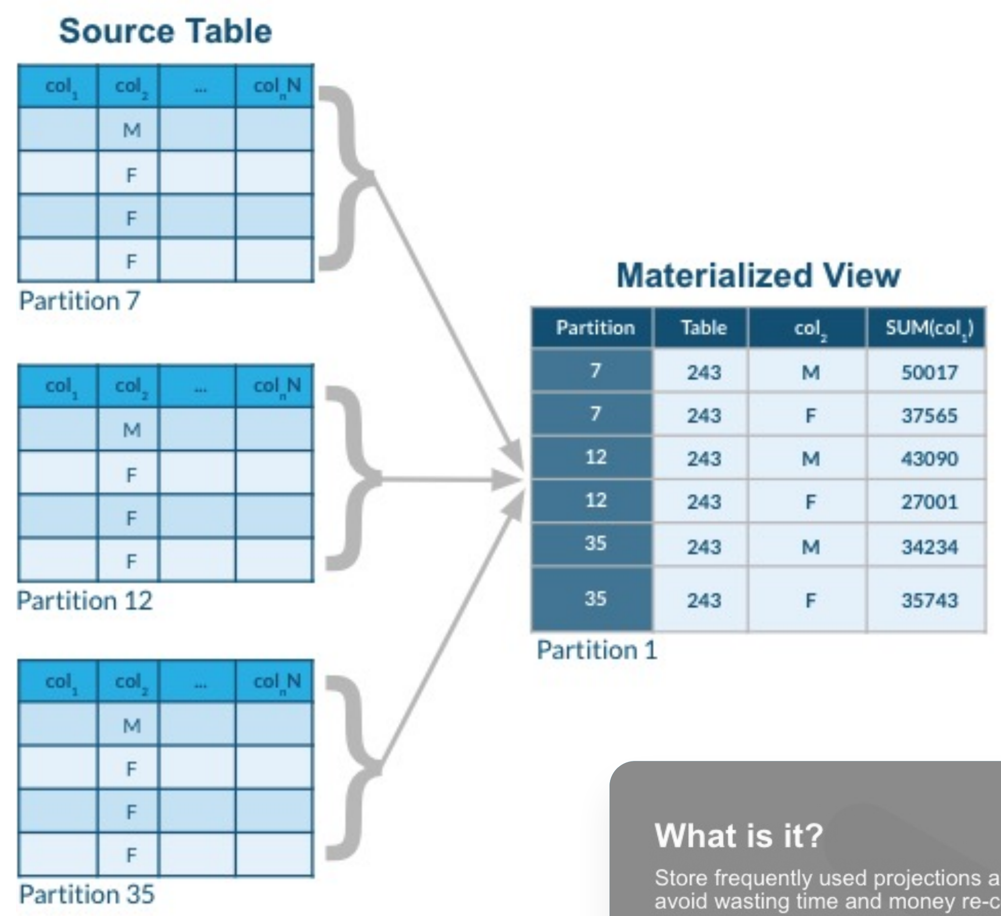
4.1.5 Materialized Views

A materialized view is a pre-computed data set derived from a query specification (the SELECT in the view definition) and stored for later use. Because the data is pre-computed, querying a materialized view is faster than executing a query against the base table of the view. This performance difference can be significant when a query is run frequently or is sufficiently complex. As a result, materialized views can speed up expensive aggregation, projection, and selection operations, especially those that run frequently and that run on large data sets.

Materialized views are designed to improve query performance for workloads composed of common, repeated query patterns. However, materializing intermediate results incurs additional costs. As such, before creating any materialized views, you should consider whether the costs are offset by the savings from re-using these results frequently enough.

For more information see https://docs.snowflake.com/en/user-guide/views-materialized.html.

MATERIALIZED VIEWS



Partition	Table	col_2	SUM(col_1)
7	243	M	50017
7	243	F	37565
12	243	M	43090
12	243	F	27001
35	243	M	34234
35	243	F	35743

What is it?

Store frequently used projections and aggregations to avoid wasting time and money re-computing

Results from MV queries guaranteed to be up-to-date

Value

Run faster queries, spend fewer compute credits

Asynchronous, incremental maintenance

No impact on DML performance

MV on External Tables

- Fast, governed access to Data Lake
- Benefit from Snowflake performance while maintaining source of truth outside

Cost

Serverless refresh, additional storage

4.2 Snowflake Monitoring

Sigma has multiple Snowflake monitoring Workbook Templates that are built into every Sigma instance. These templates provide easy visibility into key aspects of their Snowflake account:

The Contract Monitoring Workbook shows burn rates, account breakdowns and usage trends.

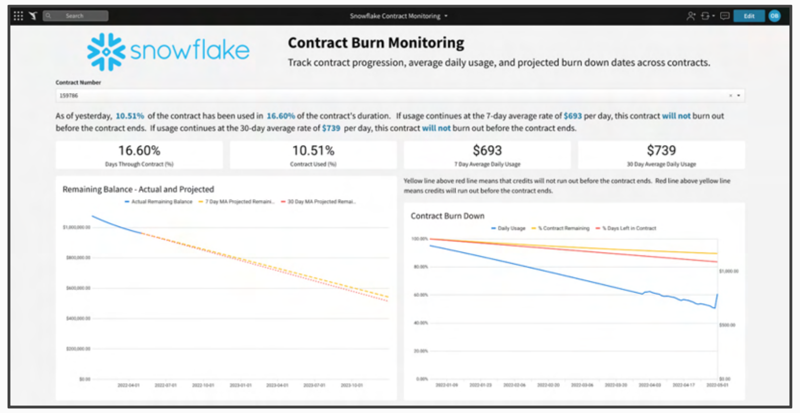
The Storage Costs Workbook shows costs by database, including a breakdown of Time Travel and Fail-Safe costs.

The Compute Costs Workbook shows warehouse-specific costs and detailed histograms, to enable deep dives to track spending.

The Reader Accounts Workbook highlights costs associated with reader accounts in warehouse breakdowns and histograms.

The Performance Monitoring Workbook provides KPIs and visuals that expose hardware-related metrics, in order to help diagnose improperly sized warehouses or other inefficiencies.

The User Adoption Workbook presents metrics around distinct users in the Snowflake tenant. This can be extremely helpful to show progress in a Snowflake implementation.



Snowflake Contract Monitoring

Contract Burn Monitoring

Track contract progression, average daily usage, and projected burn down dates across contracts.

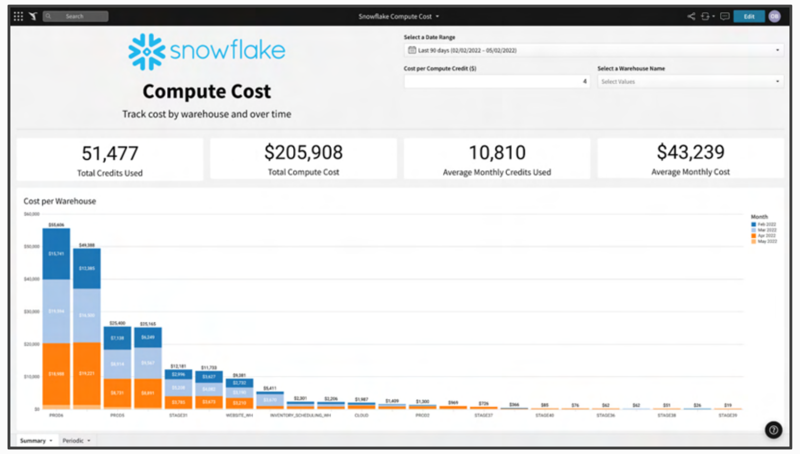
Contract Number: 159786

As of yesterday, 10.51% of the contract has been used in 16.60% of the contract's duration. If usage continues at the 7-day average rate of $693 per day, this contract will not burn out before the contract ends. If usage continues at the 30-day average rate of $739 per day, this contract will not burn out before the contract ends.

16.60%	10.51%	$693	$739
Days Through Contract (%)	Contract Used (%)	7 Day Average Daily Usage	30 Day Average Daily Usage

Yellow line above red line means that credits will not run out before the contract ends. Red line above yellow line means credits will run out before the contract ends.

Remaining Balance - Actual and Projected
Actual Remaining Balance
7 Day MA Projected Remaini...
30 Day MA Projected Remai...
Contract Burn Down
Daily Usage
% Contract Remaining
% Days Left in Contract



Snowflake Compute Cost

Compute Cost

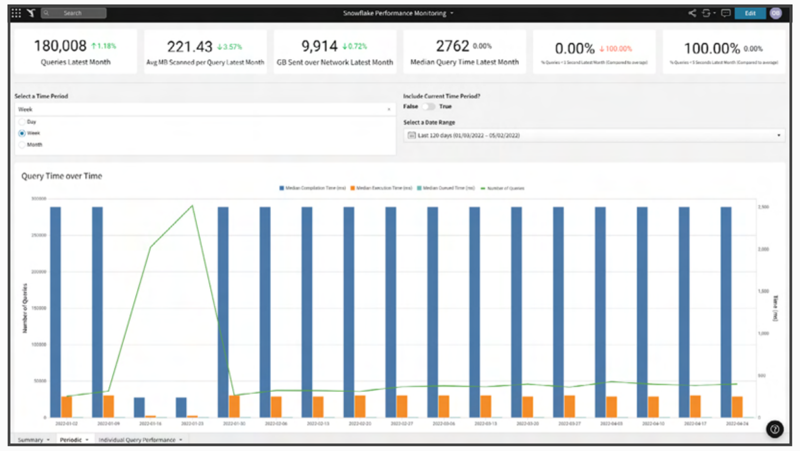
Track cost by warehouse and over time

Select a Date Range: Last 90 days (02/02/2022 - 05/02/2022)

Cost per Compute Credit ($): 4

Select a Warehouse Name: Select Values

51,477	$205,908	10,810	$43,239
Total Credits Used	Total Compute Cost	Average Monthly Credits Used	Average Monthly Cost



Snowflake Performance Monitoring

180,008 ↑1.18%	221.43 ↓3.57%	9,914 ↓0.72%	2762 0.00%	0.00% ↓100.00%	100.00% 0.00%
Queries Latest Month	Avg MB Scanned per Query Latest Month	GB Sent over Network Latest Month	Median Query Time Latest Month	% Queries < 1 Second Latest Month (Compared to average)	% Queries < 5 Seconds Latest Month (Compared to average)

Select a Time Period: Week

Day [✓ Week] Month

Include Current Time Period? False True

Select a Date Range: Last 120 days (01/03/2022 - 05/02/2022)

Query Time over Time
Median Compilation Time (ms)
Median Execution Time (ms)
Median Queued Time (ms)
Number of Queries
Summary
Periodic
Individual Query Performance

4.3 Data Modeling for Optimized Analysis

4.3.1 Modeling Upstream of Sigma

Sigma provides a modeling layer which can be equated to a traditional set of views in a presentation layer of a data warehouse. These datasets can serve many purposes, including:

- **Creating reusable and consistent content**
- **Curating content for end users**
- **Locking down defined metrics**
- **Filtering out sensitive data**
- **Filtering out sample data or test accounts**
- **Sharing selective content based on audience e.g. row-level security**
- **Creating a protective layer of abstraction between raw data and finished reports**
- **Limiting data to a consistent rolling period**
- **Last mile modeling such as flattening out a dimensional model**
- **Exploratory modeling e.g. rapid data prototyping**

Modeling in Sigma can be a powerful tool to create performant workbooks, or to understand underlying data quickly. This modeling is intended to be last-mile or exploratory in nature. When building production pipelines, best in breed transformation tools such as dbt or Matillion should be implemented. These tools are designed for enterprise production data pipeline development and have built-in version control, alerts, and orchestration.

At scale, Sigma Datasets should be used in conjunction with one of the aforementioned technologies or a comparable analog. This will set the business up with the most robust foundation with which to report and analyze from.

5 Optimizing Sigma

5.1 Workbooks Best Practices

Workbooks are a powerful tool that can provide a window into Snowflake in a simple, performant, and effective way. Given the complex nature of visualizing massive datasets, it is important to take critical foundational steps when setting up a workbook. Many of the aforementioned topics lay the groundwork for the actual process of building Workbooks in an efficient manner.

5.1.1 Using Base Tables

The first step when creating a new Workbook should be creating a new data page which will house “base tables” for all of the other elements in the workbook. These base tables can be used as a centralized location to perform data prep, apply filters to many elements at once, and create a layer of abstraction between the data source and the presentation elements.

This element hierarchy plays a fundamental role in creating maintainable and optimized workbooks that can grow and adapt to business needs. A single base table can feed any number of child elements, simplifies global filters, and allows for reuse of calculated columns and repeated business logic across multiple child objects. It is recommended that this hierarchy be used for all presentation elements—tables, pivot tables, and visualizations—in the workbook. ALWAYS start with a base table and build child elements from there.

5.1.2 Lookups and Joins

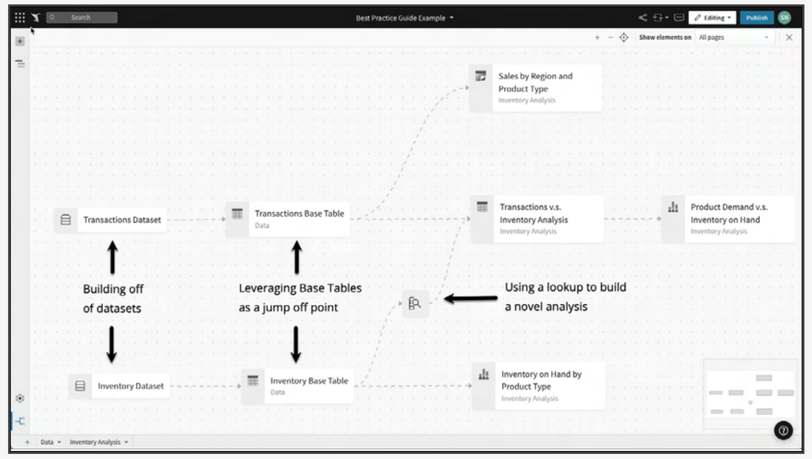
Next, when pulling in new data sources for an analysis or dashboard, there are times when it will be necessary to connect disparate pieces of data. Sigma provides a variety of options for this: Workbook Lookups and Joins, as well as Dataset Joins. These tools all provide ways of connecting and flattening models defined in the warehouse via either a streamlined UI or Excel-like functions.

In exploratory analysis or within the context of a single workbook, the use of Workbook Lookups and Joins may be a good fit. The goal of these tools is to enable end users to extend the capabilities of source data without having to rely on the data engineering teams to rebuild new datasets. This can be powerful especially in the early phases of the workbook and data product development lifecycle.

Type	Cardinality	Use Case
Lookups	1:1 1:many only when performing a rollup/aggregate calculation	Single value retrieval across disparate sources in a workbook
Joins	1:many	Flattening models locally in a workbook
Datasets	1:1 or 1:many (standard SQL joins)	Creating reusable flattened models and/or leveraging joins and hiding result sets

A NOTE OF CAUTION

As Workbooks mature, numerous levels of nested Lookups and joins can degrade performance and maintainability. Localized workbook modeling that starts to include a large number of lookups and joins is a good candidate for promotion to datasets, or into production transformation models.



If the analysis is not a one-off exercise, meaning there are multiple use cases in which these data sources may need to be connected and analyzed, then this would be a good opportunity to leverage a Dataset Join. Doing joins inside a Dataset creates a curatable asset that will both reduce redundant effort when recreating similar analyses, as well as guarantee that all analyses are using the same, correct join model. More information on Datasets can be found in section 5.2 of this document.

5.1.3 Control Elements

Once a workbook is created it is important to provide the end users effective tools to manipulate and explore the data. There are various control elements in Sigma workbooks that can either filter the data that is presented to them or provide curated drill paths to modify the way in which the data is visualized. The different filters in workbooks allow for a customizable experience that can be applied to almost every data type. Filters can be applied as either element specific controls or page level controls. In the case of page level controls there are granular settings to allow for the filter to apply to multiple elements.

It is important to remember that when using the "base table" paradigm discussed in previous sections a filter applied to the base table will affect all child elements. If more granular filtering is required then instead of filtering the "base table" the filter should be created on the child element. Filtering is not only critical for analysis in the case of large data volumes, an effective filter can make a tremendous impact on the amount of compute required by the Snowflake warehouse. A great example of this would be utilizing date filtering to only look at the pertinent records in the given analysis. It would be unnecessarily costly and compute intensive to include historical data in an analysis focusing on current reporting. Although Sigma and Snowflake are capable of pivoting a nearly limitless amount of data, the standard rules of data analysis still apply - filter early and only retrieve the data you need for your analysis.

Separate from filtering is exploration, the drill down control element provides tremendous value for the end user in this endeavor. The drill down element provides the same granular settings as the various filter control elements, allowing for n-number of distinct elements to be modified by it. The key value of the drill down element is that it provides an easy to use drill down functionality to viewer accounts that are not able to leverage the ad-hoc drill anywhere functionality granted to explorer and creator accounts.

When building workbooks in the future, strongly consider the best practices laid out in this section. Each of them have the ability to dramatically improve the overall experience.

5.2 Dataset Best Practices

5.2.1 When to Use Datasets

In any analytics and BI environment there is a need for curation of data, whether for the purpose of managing defined KPIs, ensuring security, abstracting the raw data, performing foundational transformations and dataprep, or materializing the result set. Sigma Datasets fill this need. A dataset starts off as a query to a single table in the data cloud. From there, many different actions can be applied, including joining in supplemental tables, grouping the data, creating computed columns, modifying data types, extracting JSON, and more. The benefits of building out this model in a dataset is twofold: datasets are reusable meaning they can be curated to a team, and a dataset can be materialized back in Snowflake thus substantially increasing the performance of a dataset at scale.

5.2.2 When to Build Datasets Off of Other Datasets

When architecting a dataset it is best to approach the problem by considering the end goal and working backwards, looking for places where the workload can be broken out. Since datasets can leverage other datasets as a source, creating a modular framework is a great option to reduce the complexity of any single dataset as well as improve reusability. For example a given dataset may join several tables to create a logical model. This first dataset can then be leveraged downstream inside of another dataset that may join in other data sources to create a subsequent logical model. Even though one of the datasets is a component of the other there may be business cases where an analysis needs to be performed on the second level model - consider, for instance, a LOB specific model that is curated for the specific needs of a single department. In this example there are now two distinct modular datasets that can be curated to the environment. In large environments utilizing this paradigm will cut down on the total number of datasets in an environment by removing the need for duplicated, one-off datasets.

5.2.3 When to Use Endorsement Badges

When creating production ready datasets, it is recommended that Sigma's endorsement badges are leveraged to communicate their bonafide nature to the environment. Badges available include:

- **Endorsed**
- **Warning**
- **Deprecated**

When adding a badge, always include a note in the notes section to make it clear why the badge has been applied.

5.2.4 When to Materialize Datasets

Joins and transformations may become obstacles to performance in complicated analyses. Since all data displayed in Sigma is queried directly from Snowflake, the complexity of the base query will carry through in all subsequent queries associated with sorting, filtering, and paginating the data. To solve this, Sigma datasets can be materialized back to Snowflake. This is different from the dataset view object in Snowflake. In Sigma's implementation, a "CREATE TABLE AS" statement is used to store the result set of the SQL query generated by the dataset. Materializations can be set on a schedule or triggered via the Sigma API. By materializing a dataset an extremely complex query can be flattened down into a simple "SELECT" meaning that all downstream queries are able to run under far less strain.

5.3 Embedding Best Practices

Sigma enables embedding objects through the use of iframes. Three different types of embedding are supported:

- **Public**
- **Private**
- **Application**

Public Embedding provides an open access url that requires no authentication to be accessed. This can be useful in situations where the object contains information that may be shared with the public.

Private Embedding provides a layer of authentication that guarantees the end users match up with a Sigma account. This extends Sigma's security model and enables the full workbook experience. Private embedding is often used with internal line-of-business applications such as Salesforce.

Application Embedding fully delegates authentication to the client application. An example of this would be a SaaS application leveraging embedded Sigma Workbooks to present their customers with a rich analytics platform to explore their application data. This is very useful in situations where developers want to provide the end users of their application with the power of Sigma without providing them accounts and separate licenses.

All of these methods allow end users to access embedded Sigma data objects without directly logging into Sigma.

5.4 Variant Data and when to Model Outside of Sigma

Being purpose built for the Snowflake data cloud, Sigma supports many of the cutting edge features Snowflake has to offer, such as variant data types. Sigma can natively parse JSON objects quickly and efficiently. This is a useful feature when profiling data and figuring out how to use it in the warehouse. A very powerful workflow is to push JSON objects from an OLTP platform into Snowflake, profile the data in Sigma to understand structure and business context, and then use that profile knowledge to build production pipelines.

When trying to decide if JSON parsing should be done within Sigma vs upstream of Sigma, the key question is how the parsed JSON will be used. If it is going to feed enterprise production reporting or used by other applications or widely by multiple departments, it likely makes sense to parse the data upstream of Sigma in your Raw to Clean transformation tool (see Modeling in Sigma section for more details). If it is for use within the current Sigma workbook or for ephemeral analysis, parsing in Sigma is a great option.

Additionally, Sigma cannot easily parse a JSON array (i.e. more than one JSON Object in a single column). In those cases you'll need to split the array and run a lateral flatten upstream of Sigma in Snowflake to get the desired results.

JSON Array

User ID	Name	Web Events
1	David	{click, '2022-01-01'}; {open, '2022-01-02'}

JSON Array Laterally Flattened

User ID	Name	Web Events
1	David	{click, '2022-01-01'}
1	David	{open, '2022-01-02'}

JSON Array Laterally Flattened and Parsed JSON

User ID	Name	Web Events	Action	Timestamp
1	David	{click, timestamp};	Click	'2022-01-01'
1	David	{open, timestamp}	Open	'2022-01-02'

6 Caching

6.1 Snowflake Caching

Persisted Query Results (“Result Cache”)

When a query is executed in Snowflake, the result is persisted (i.e. cached) for a period of time. Snowflake uses persisted query results to avoid re-generating results when nothing has changed (i.e. “retrieval optimization”). If a user repeats a query that has already been run, and the data in the table(s) hasn’t changed since the last time that the query was run, then the result of the query is the same. Instead of running the query again, Snowflake simply returns the same result that it returned previously. This can substantially reduce query time because Snowflake bypasses query execution and, instead, retrieves the result directly from the cache. Each time the persisted result for a query is reused, Snowflake resets the 24-hour retention period for the result, up to a maximum of 31 days from the date and time that the query was first executed. After 31 days, the result is purged and the next time the query is submitted, a new result is generated and persisted.

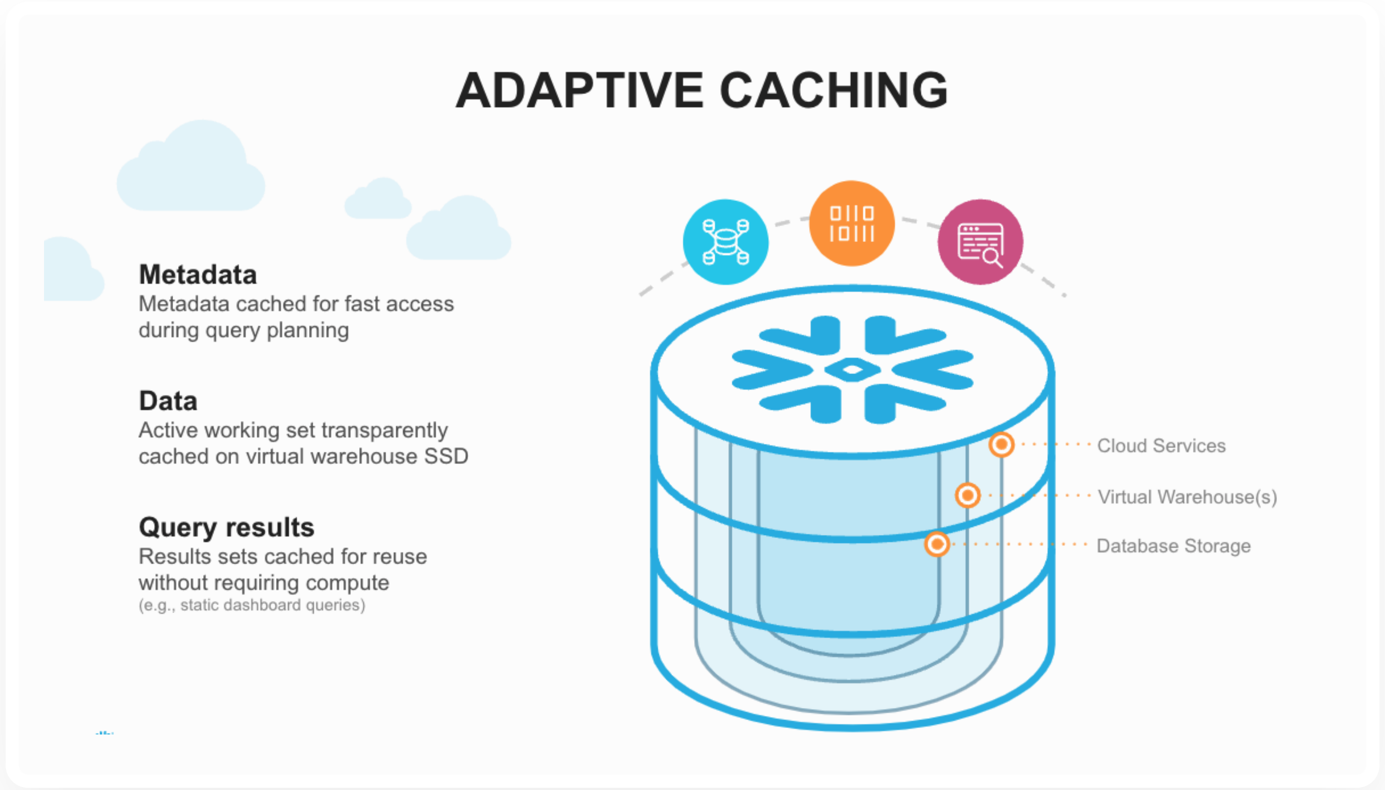
Data (SSD / Disk) Cache

Each virtual compute warehouse, when running, maintains a cache of table data accessed as queries are processed by the warehouse. This enables improved performance for subsequent queries if they are able to read from the cache instead of from remote disk storage for the table(s) in the query. The size of the cache is determined by the compute resources in the warehouse (i.e. the larger the warehouse and, therefore, more compute resources in the warehouse), the larger the cache.

This cache is dropped when the warehouse is suspended, which may result in slower initial performance for some queries after the warehouse is resumed. As the resumed warehouse runs and processes more queries, the cache is rebuilt, and queries that are able to take advantage of the cache will experience improved performance.

Keep this in mind when deciding whether to suspend a Snowflake virtual warehouse or leave it running. In other words, consider the trade-off between saving credits by suspending a warehouse versus maintaining the cache of data from previous queries to help with performance.

For additional information on this topic, see https://community.snowflake.com/s/article/Caching-in-Snowflake-Data-Warehouse.


ADAPTIVE CACHING
Metadata
Metadata cached for fast access during query planning
Data
Active working set transparently cached on virtual warehouse SSD
Query results
Results sets cached for reuse without requiring compute
(e.g., static dashboard queries)
Cloud Services
Virtual Warehouse(s)
Database Storage

6.2 Sigma Optimization

Many Sigma customers expect to use more warehouse compute due to the direct, easy access to data they can now provide to users.

To offset Snowflake compute costs, Sigma applies multiple tiers of caching and evaluation that effectively reduce warehouse load, while delivering a faster user experience. As a result, customers typically see their cost per user decrease as they increase their number of users in Sigma. In addition to Materialization and Snowflake Results Cache optimization, Sigma leverages 4 different factors to make this possible:

- **Sigma Results Cache**
- **Sigma Browser Cache**
- **Sigma Alpha Query**
- **Sigma Query Timeouts**

6.2.1 Sigma Results Cache

Sigma maintains a mapping of Snowflake result ID's. This cache actively manages a data structure containing a hash of the queries sent to Snowflake and their result ID. If a Sigma generated SQL query has been previously run, Sigma can actually request the result from Snowflake using the request ID instead of reissuing a new query.

6.2.2 Sigma Browser Cache

Sigma also maintains a cache of recent results in the web browser. As result sets are returned from Snowflake, they enter the browser cache. Every new query is checked against recent results in the browser cache before anything is sent to Snowflake. When a matching query result is found, no network request or query is issued to Snowflake.

6.2.3 Sigma Alpha Query

Aside from caching, Sigma has created a tool called Alpha Query that operates as a processing layer to calculate simple arithmetic operations instead of issuing a query to Snowflake.

For example, if a user were to calculate a percentage change ([column 2] - [column 1])/[column 1]), Sigma would use Alpha Query. This substantially decreases the total number of queries issued to Snowflake during data prototyping and exploration.

6.2.4 Sigma Query Timeout

To prevent resource intensive queries from driving up costs, Sigma enforces a query timeout. The timeout cancels any query that hits a set-time threshold. This prevents runaway queries from consuming resources and warehouse credits. The timeout threshold is customizable with a default set at 2 minutes. The customizations are made on the connection definition page.

7 Conclusion

Modern businesses need a modern data strategy built on platforms that support agility, scalability, and operational efficiency. Sigma and Snowflake are two technologies that work together to provide just such a platform. They're capable of unlocking tremendous value when used together. Following the best practices highlighted in this white paper allows you to unlock the most value possible while minimizing the amount of resources expended.

sigma

Rock-solid governance is a non-negotiable piece of every good data strategy. But it shouldn't be a stumbling block for business teams. Taking a modern approach transforms governance from a roadblock to an enabler by ensuring the right people have direct, real-time access to the data they need to make the best decisions possible.

sigmacomputing.com