

WHITE PAPER



# Qlik Sense Best Practices Guide

**David Freriks**

**Dalton Ruer**

Partner Engineering - Qlik

**Keith Smith**

Partner Engineering -  
Snowflake Computing

# Table of Contents

- Summary ..... 3**
- Introduction..... 3**
- Qlik Architecture ..... 5**
- Associative, in-memory apps ..... 5**
- Snowflake Architecture ..... 8**
- Snowflake Time Travel..... 10**
  - Querying Historical Data ..... 10
- Snowflake Semi-Structured Data ..... 11**
- High Level Qlik & Snowflake Integration Options ..... 12**
- Summary of Qlik & Snowflake Usage Options ..... 13**
  - On-demand Apps ..... 14
  - \*NEW\* - Qlik SaaS Direct Query..... 15
  - Live Query via Analytic Connector Apps – Qlik Sense Client Managed ..... 16
- Deep Dive Snowflake & Qlik Method Decision Tree: ..... 17**
- Technology Guidance on Integration Options - Qlik: ..... 23**
- Personas by Usage: Consumers, Analysts, Designers..... 28**
- Performance Considerations / Best Practices by Technique ..... 29**
- Data Caching for Performance & Compute Cost Consideration ..... 29**
  - Why is Caching So Important for Dashboards? ..... 30
  - Why is Caching So Important in Snowflake? ..... 30
  - Why is Qlik Sense So Great with Snowflake? ..... 31
  - Key Considerations for Caching & Cost Management with Snowflake..... 31
  - Scenarios to Avoid ..... 32
  - Best Practice Scenario Qlik Sense in-memory (or blended use case covered later)..... 32
  - Enable Bulk Reader Option..... 33
- Incremental Load Options ..... 35**
  - Incremental Loads using QVD’s..... 35
  - QVD-based Incremental Load - Insert Only ..... 37
  - QVD-based Incremental Load - Insert and Update ..... 40
  - QVD-based Incremental Load - Insert, Update and Delete ..... 42
- Incremental Loads using “Partial Load” ..... 43**
- Time Travel with Merge Best Practices ..... 45**
  - Time Travel Setup & Qlik Merge Function Details ..... 46
- On-Demand Options..... 47**
  - On Demand App Generation (ODAG)..... 47
  - Dynamic Views..... 48
- Qlik Snowflake Usage Dashboard (V3.1) ..... 50**
  - Data Model: ..... 51
  - Data Load Script: ..... 52
- Analysis Details about the Usage Dashboard ..... 53**
  - Table of Contents: ..... 53
  - Usage Cost Analysis: ..... 53
  - Enterprise Credit Usage: ..... 54

Auditing / Security: .....	55
Query Performance: .....	55
Connection Details .....	56
Database Details .....	56
<b>Appendix: Connecting Qlik to Snowflake (QSE and Qlik SaaS) .....</b>	<b>57</b>
Making Snowflake Connection .....	57
Authentication Methods .....	57
<b>Appendix: ODBC Connection Setup .....</b>	<b>58</b>
Native Connection (Qlik Sense Enterprise / Qlik SaaS): .....	58
Username and Password Option Settings: .....	60
OAUTH Option Settings: .....	62
<b>Appendix: ODBC Connection (Qlik Sense Enterprise) .....</b>	<b>67</b>
Download the ODBC driver .....	67
<b>Appendix: Installing and configuring the ODBC Driver for Windows .....</b>	<b>68</b>
<b>Appendix: Qlik Sense Configuration .....</b>	<b>71</b>
Install & Configure Qlik Sense .....	71
Creating the Qlik Sense App .....	71
<b>Appendix: Using Qlik Sense SaaS Direct Query .....</b>	<b>78</b>
<b>Conclusions .....</b>	<b>82</b>

## Summary

---

Qlik and Snowflake make a powerful combination when deploying modernized data and analytics pipelines across an enterprise. Due to emerging technologies and urgency of data, there are many architectural avenues available when deploying Qlik and Snowflake together. This document will highlight and clarify the options and best practices and discuss high level concepts, practical applications, and help distinguish when to use which approach with Qlik and Snowflake.

This document will be updated as new capabilities emerge to both Snowflake and the Qlik Sense platform. This document also assumes a working knowledge of both Qlik and Snowflake technologies by the reader.

## Introduction

---

Qlik Sense sets the benchmark for third-generation analytics platforms, empowering everyone in your organization to make data-driven decisions. Built on our unique Associative Engine, it supports a full range of users and use-cases across the life-cycle from data to insight: self-service analytics, interactive dashboards, conversational analytics, custom and embedded analytics, mobile analytics, reporting and alerting. It augments and enhances human intuition with AI-powered insight suggestions, automation, and natural language interaction. And Qlik Sense offers unmatched performance and governance, with the convenience of SaaS or on-premises deployment, or both

**Qlik Sense** is a complete data and analytics platform enabling users of all levels to explore data with agility and high performance. **Snowflake** is a cloud based,



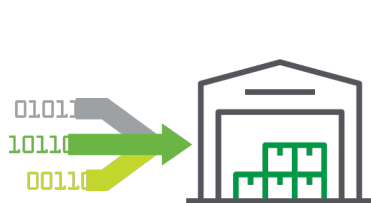
massively scalable platform that provides effective management of enterprise class data.

Qlik and Snowflake together provide a balance to optimize the “speed of thought” exploration capabilities when Qlik’s associative engine with its powerful search and AI capabilities is combined with Snowflake’s powerful and scalable database engine technology.

Blending the ideas “getting the data you need when you need it” and “getting the data how you need it” with two cutting edge technology platforms creates unique solutions that deliver enterprise analytics, reporting, dashboarding, and data science to the business.

## Qlik & Snowflake

### Accelerate Business Value with Real-time Data Integration and 3rd Generation BI



#### Get the data you need

- Continuous data ingestion
- Data Warehouse Automation
- Performance with scale



#### Uncover more insights

- Analytics for all
- Augmented Intelligence
- Smart data querying



#### Manage data intelligently

- Self-service catalog
- Flexible, rules-based governance
- Optimize Snowflake usage

## Qlik Architecture

Qlik Sense sets the benchmark for third-generation analytics platforms, empowering everyone in your organization to make data-driven decisions. Built on our unique Associative Engine, it supports a full range of users and use-cases across the life-cycle from data to insight: self-service analytics, interactive dashboards, conversational analytics, custom and embedded analytics, mobile analytics, reporting and alerting. It augments and enhances human intuition with AI-powered insight suggestions, automation, and natural language interaction. Further, Qlik Sense unmatched performance and governance, with the convenience of SaaS or on-premises deployment. Qlik Sense consists of Qlik-managed cloud-based solutions: *Qlik Sense Enterprise SaaS & Qlik Sense Business*, and a customer-managed solution: *Qlik Sense Enterprise Client-Managed*.

### Associative, in-memory apps

Qlik couples in-memory data caching technology with an Associative Engine that lets you analyze and freely navigate data intuitively. In its second generation, the proven Qlik Associative Engine

#### THE ASSOCIATIVE DIFFERENCE®

Relational databases and queries were designed in the 1980s for transactional systems, not modern analytics. Query-based tools leave data behind and limit your users to restricted linear exploration, resulting in blind spots and lost opportunities.

Qlik Sense runs on the unique Qlik Associative Engine, enabling users of all skill levels to explore their data freely without limitations. The Qlik Associative Engine brings together unlimited combinations of data — both big and small — without leaving any data behind. It offers unprecedented freedom of exploration through interactive selection and search, instantly recalculating all analytics and revealing associations to your user in green (selected), white (associated), and gray (unrelated). By keeping all visualizations in context together and retaining both associated and unrelated values in the analysis, the Qlik Associative Engine helps your users discover hidden insights that query-based tools would miss.

The Qlik Associative Engine is purpose-built for highly scalable, dynamic calculation and association on massive data volumes for large numbers of users. This unique technology is our primary advantage, with more than 25 years of innovation and investment.

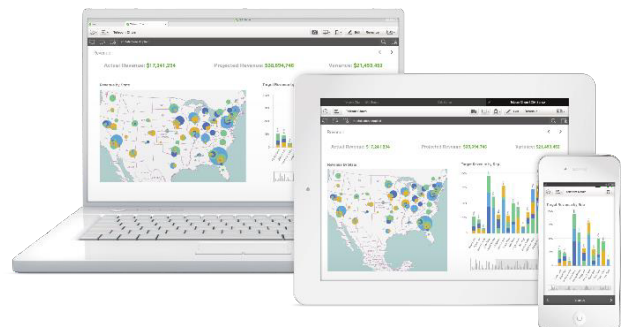
allows users to easily explore data and create visualizations based on data from multiple data sources simultaneously. These sources range from Excel<sup>®</sup> and Access<sup>®</sup> to databases such as Oracle<sup>®</sup> and SQL Server to big data sources such as Databricks<sup>®</sup>, Snowflake<sup>®</sup>, data lakes with S3, etc.

Qlik Sense uses columnar, in-memory storage. Unique entries are only stored once in-memory, and relationships among data elements are represented as pointers. This allows for significant data compression, more data in RAM, and faster response times for your users.

In some big data scenarios, data should remain at the source, which is why Qlik uses a built-in technique called On-Demand Application Generation. Data sources can be queried based on your users' selections, yet still provide an associative experience to your user. Qlik's Dynamic Views feature expands this capability further for the biggest data sources available.

## User Interfaces

Access to the Qlik Sense Enterprise SaaS environment is through a zero-footprint web browser interface (known as the Qlik Sense Hub). The Qlik Sense web browser interface makes all aspects of development, drag-and-drop content creation, and consumption possible. Qlik Sense features a responsive design methodology to automatically display and resize visualizations with the appropriate layout

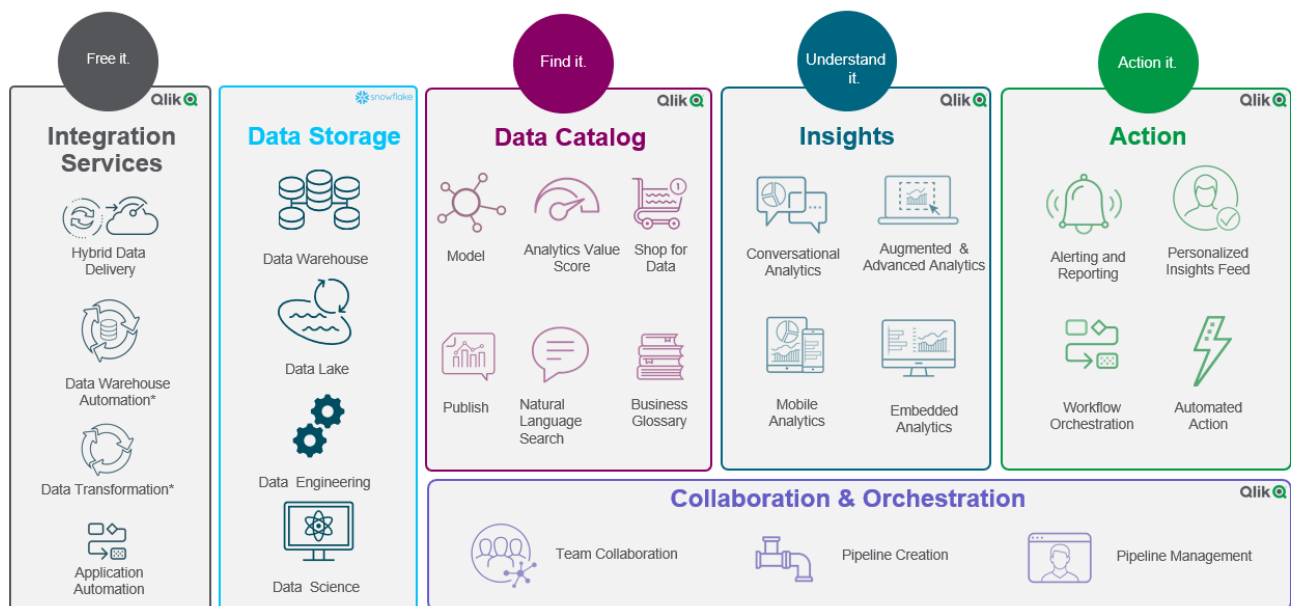


and information to fit the device — whether it is a browser on a laptop or desktop, tablet, or smartphone. Built with current standards of HTML5, CSS3, JavaScript®, and web sockets, Qlik Sense enables you to build and consume apps on any device.

In addition to the web-based interface, Qlik Sense supports conversational analytics which integrates with major chat platforms such as Slack and MS Teams and data alerting capabilities to allow users to subscribe to and be notified of key changes to their data.

A quick reference to the entire Qlik Platform including Data Integration capabilities, cataloging, and extending Qlik Sense showcases the power of our integrated suite for Snowflake.

### Qlik’s complete Data Integration and Analytics Pipeline with Snowflake Cloud Data Platform:



# Snowflake Architecture

## Snowflake Cloud Data Platform

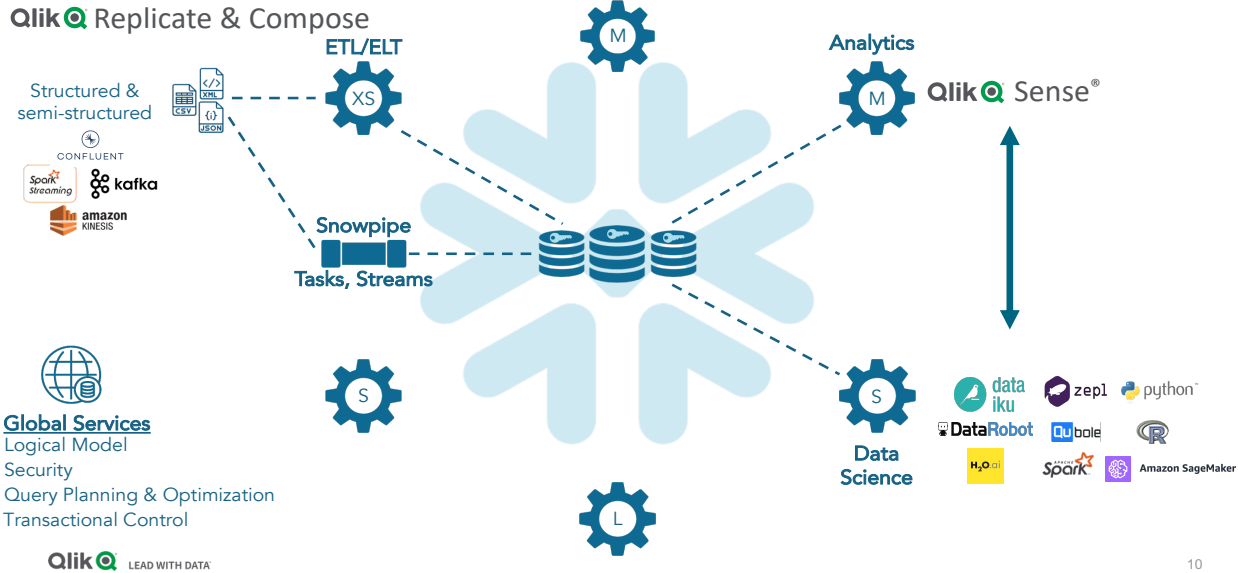


Snowflake is a fully relational ANSI SQL platform that allows you to leverage the skills and tools your organization already uses. Updates, deletes, analytical functions, transactions, complex joins, and native support for semi-structured formats allow a customer full capability to make use of all of their enterprise data.

Snowflake eliminates the administration and management demands of traditional data warehouses and big data platforms. As a true data warehouse-as-a-service built for the cloud, the separation of compute and storage allows for minimal tuning on terabytes and even petabytes of information. With built-in performance, there's no infrastructure to manage or knobs to turn. Snowflake automatically handles infrastructure, optimization, availability, data protection and more so you can focus on using your data, not managing it.

Snowflake can support all of your business data, whether from traditional sources or newer machine-generated sources, without requiring cumbersome transformations and tradeoffs. Further, Snowflake’s Data Marketplace allows you to access hundreds of partners and suppliers’ data as well as 3<sup>rd</sup> party data that you be leveraged to enrich your data.

## Varied Warehouses for Varied Workloads



Snowflake’s patented multi-cluster, shared data architecture separates storage and compute, making it possible to scale up and down on-the-fly without downtime or disruption. Automatically scale to support any amount of data, workloads and concurrent users and applications without requiring data movement, data marts or data copies.

Per Snowflake:

*“Usage-based pricing for compute and storage means you only pay for the amount of data you store and the amount of compute processing you use. Say goodbye to upfront costs, over-provisioned systems or idle clusters consuming money.”*

## Snowflake Time Travel

Snowflake Time Travel enables accessing historical data (i.e., data that has been changed or deleted) at any point within a defined period. It serves as a powerful tool for performing the following tasks:

- Restoring data-related objects (tables, schemas, and databases) that might have been accidentally or intentionally deleted.
- Duplicating and backing up data from key points in the past.
- Analyzing data usage/manipulation over specified periods of time.

### Querying Historical Data

When any DML operations are performed on a table, Snowflake retains previous versions of the table data for a defined period of time. This enables querying earlier versions of the data using the AT | BEFORE clause.

This clause supports querying data either exactly at or immediately preceding a specified point in the table's history within the retention period. The specified point can be time-based (e.g. a timestamp or time offset from the present) or it can be the ID for a completed statement (e.g. SELECT or INSERT).

For example:

- The following query selects historical data from a table as of the date and time represented by the specified timestamp:  

```
select * from my_table at(timestamp => 'Mon, 01 May 2015 16:20:00 - 0700'::timestamp_tz);
```
- The following query selects historical data from a table as of 5 minutes ago:  

```
select * from my_table at(offset => -60*5);
```

## Snowflake Semi-Structured Data

---

Data can come in multiple forms from numerous sources, including an ever-expanding amount of machine-generated data from applications, sensors, mobile devices, etc. To support these new types of data, semi-structured data formats, such as JSON, Avro, ORC, Parquet, and XML, with their support for flexible schemas, have become popular standards for transporting and storing data.

Snowflake provides native support for semi-structured data, including:

- Flexible-schema data types for loading semi-structured data without transformation.
- Automatic conversion of data to optimized internal storage format.
- Database optimization for fast and efficient SQL querying.

The VARIANT data type can be leveraged to represent arbitrary data structures which can be used to import and operate on semi-structured data. Snowflake stores these types internally in an efficient compressed columnar binary representation of the documents for better performance and efficiency. Snowflake's optimization for storage of these data types is completely transparent and produces no user-visible changes in semantics.

More details on traversing semi-structured data can be found [here](#).



## High Level Qlik & Snowflake Integration Options

Qlik and Snowflake offer a variety of integration capabilities suited for specific scenarios to maximize resources, maintain performance, and fulfill the business requirements across the organization. The number of concurrent users, data refresh frequencies, performance, user experience, and total cost will all contribute to deciding which integration options to take advantage of.

Qlik provides several ways to load and consume data from Snowflake. The table below describes each integration configuration, feel free to use this as a reference guide.

Integration Option	Method	Use Case
<u>In-Memory</u>	Full / Incremental Load (QVD)	Most common and best performing. Batch reloads leveraging a QVD for incremental updates and optimized for analytics engine.
<u>In-Memory + Snowflake</u>	Full / Incremental Load ( <u>Time Travel</u> )	Snowflake optimized reload – especially useful when data is needing to be compared current vs historical state and/or for incremental reloads
<u>In-Memory + Snowflake</u>	Partial/Merge Load ( <u>Time Travel</u> )	Situations where data is needed in near-time, or data is needing to be compared current vs historical state
<u>On-Demand</u>	ODAG	Structured Drill to Detail for access to large data volumes
<u>On-Demand</u>	Dynamic Views	Supporting details on demand inside existing app as a chart(s)
<u>In-Mem + Live1</u>	In-Mem + Live Query	Realtime data mixed with in-memory data querying against Snowflake directly (3 <sup>rd</sup> Party)
<u>Live1</u>	Live Query Only	Only live data querying against Snowflake directly
<u>Direct Query</u>	Direct Query Engine	Live SQL push down queries against Snowflake using Qlik SaaS (NEW – July 2022)

<sup>1</sup> Available at this time for client-managed only through partner offering such as the Stretch LiveQuery Connector for Snowflake.

## Summary of Qlik & Snowflake Usage Options

**In-Memory** options for how to load data into Qlik Sense:

### Full reload every time on a schedule:

- Could lead to long load times depending on data volume.
- Recommended if the data is highly volatile or has a high amount of changes.

### Incrementally load only new data

- Reload deltas on a schedule: This can be set up by a Qlik developer in the ELT script. Allows Qlik to only update the changes in the data. This method allows for near-time refresh of data into the Qlik engine and visualization layer.

## Qlik Sense - inMemory

- Search and explore across all the data, in any direction, with no pre-aggregation or predefined queries.
- Understand both related and unrelated data.
- No SQL skills required
- Augmented Intelligence to assist Users in Application creation
- Data in Memory means all available for Cognitive Engine driving Augmented Analytics and user adoption
- Multi-Sources (SF, SAP, files, ...)
- Available for Offline Mobile usage
- No additional compute cost
  
- Data Volume in Memory limited
- Data loaded and cached on disk
- Data latency due to Loading process

Response Time	██████████
User Experience	██████████
Development effort	██████████
Data Volume	██████████
Cost per User Interaction	██████

## On-demand Apps

On-demand apps help business users and IT departments derive value from big data environments in numerous ways. On-demand apps:

### On Demand App Generation (ODAG)

- Detailed Data on Demand: This is a technique typically used in big/huge data scenarios where a Qlik app is built to contain summarized data. There is a Qlik “details” app as a template and that takes parameters passed from the summary app and runs live on demand against Snowflake. The user is then presented with the appropriate slice of data based on those selections. This method is good for summary-detail analytics.

### Dynamic Views

- Live Data Visualizations: This option is used when live / near-time data is needed as part of the Qlik app. Using the ODAG framework, live data under certain query volume thresholds can be triggered to update based on user interactions with the application. As users make selections they will be prompted and if they choose to, Qlik will reload data live from Snowflake to match their selections.

## Qlik Sense – On-Demand

- Benefits from QS inMemory
- Plus, Snowflake OnDemand which brings caching in memory for more analysis
- No Data Volume limitation
- Performance of inMemory Analytics with best Data Delay
- User need to wait 5s on top of Snowflake Query Execution to load the Dynamic View and refresh the charts
- Selections on master application can be different from ones in dynamic views
- Can not leverage Cognitive Engine for data not in memory
- Can not leverage Conversational Analytics for data not in memory

Response Time	██████████
User Experience	██████████
Development effort	██████████
Data Volume	██████████
Cost per User Interaction	██████████

## \*NEW\* - Qlik SaaS Direct Query

With the July 12<sup>th</sup> Qlik SaaS update, Qlik will be adding a new feature for Snowflake called Direct Query. Direct Query in Qlik Sense allows users to build analytics applications that directly query cloud databases using SQL, as users interact through visualizations and filtering.

Direct Query uses all the same UI elements from Qlik SaaS but executes those element with runtime SQL against Snowflake. The best part is that the user still gets all the power and feedback of the Qlik Associative Engine and all the relationships that are highlighted with the green, white, grey experience Qlik users love!

Modelling is done via a modified Data Manager panel and requires understanding of primary/foreign key relationships and join strategies. All charts/KPI's create the SQL that is pushed down to Snowflake for execution. This method WILL increase Snowflake costs, and is dependent on the warehouse sizing and performance tuning applied to the data sets inside Snowflake.

### Use Cases

Expanding Qlik Sense

**Data is massive**

Analyze very large data sets where it lives in the cloud

Example Use Cases:

- Financial Fraud Detection
- IOT anomalies

**Data is near real-time**

Some scenarios require the most recent data

Example Use Cases:

- Warehouse Operations
- Fleet Management

## Direct Query

- Leverages Snowflake Scalability and Elasticity
- No Data Volume limitation
- Better Performance for Large volumes of data
- Offers Real-Time update of data
- Green/White/Grey associative experience
- SET Analysis supported
- Limited calculation capabilities
- Performance dependent on Snowflake warehouse size
- No augmented analytics capabilities
- Compute Cost is much higher as no caching
- Requires some knowledge of SQL modelling (for development)

Response Time	████████
User Experience	██████████
Development effort	██████
Data Volume	██████████
Cost per User Interaction	██████████

## Live Query via Analytic Connector Apps – Qlik Sense Client Managed

This option is available currently by using partner offerings such as the Stretch LiveQuery Connector for Snowflake. This 3<sup>rd</sup> Party connector enables live queries directly from the Qlik Sense front-end to Snowflake. No need to move data to QVDs with LiveQuery. The product is a service residing on a server and works with Qlik Sense in Windows using Server-Side Extension (SSE) integration. You can find more details on their website:

<https://stretchqconnect.com/products/livequery-for-qlik-sense/>

### Seamless integration between live queries and in-memory

The Stretch LiveQuery connector enables near-seamless integration between live queries and in-memory data. This is done in real-time as the user is using the application.

### In-memory cache

The Stretch LiveQuery connector has an in-memory result cache. This means that repeated queries can return results without accessing the warehouse. This

reduces the load on the data warehouse and increase the responsiveness of the Qlik front-end. The cache has a configurable time-to-live, when a query result in the cache surpasses this time, the result is evicted form the cache and the query is parsed on the data warehouse. This enables the administrators to balance the requirement between real-time data and load/cost on the data warehouse.

**Live Query**

- Allows Leverage Scalability and Elasticity
- No Data Volume limitation
- Better Performance for Large volume of data
- No Extract required
- Offer Real-Time update of data
- Query results cached cross users even when Virtual WH is suspended
- Performance Optimization Techniques for Guided Analytics, not a good fit for Self Service
- Limited to simple filters or linear exploration
- SQL skills required
- SQL means predetermined questions & design
- Less augmented analytics capabilities
- Compute Cost


Response Time	██████████
Time 2 Insight	██████████
User Experience	██████████
Development effort	██████████
Data Volume	██████████
Cost per User Interaction	██████████

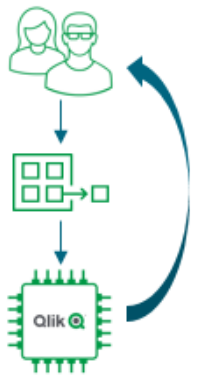
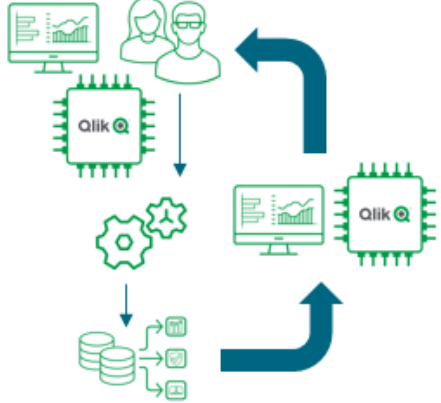
## Deep Dive Snowflake & Qlik Method Decision Tree:

Many organizations have well established data pipelines and a user base begging for access to data and insights. But between the databases and the dashboards, lay the the integrations and load architectures that make consuming data possible. This next section is to help you understand several integration options available in Qlik Sense and which to use when deploying Qlik and Snowflake together.

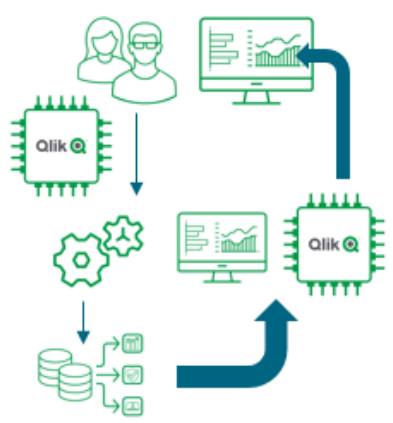
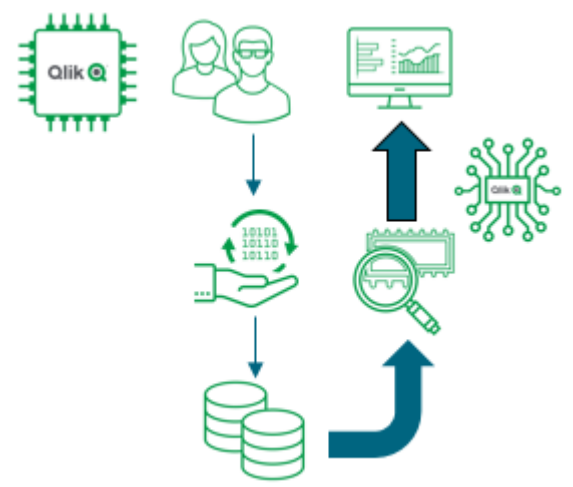
Some of the main variables when deciding will be the total number of concurrent users, system performance thresholds, costs, and the actual business requirements.

Let's begin with a high level understanding of each of the Qlik integration options before discussing which business use cases they might apply to:

<p>Incremental Load (Scheduled Data Load)</p> <p>Cached In-Memory</p>	<p>An Incremental Load is a scheduled process where Qlik loads the latest data for all of the tables needed, merges the changes to existing QVD's (the previously loaded data/cache), then rewrites those QVD's/Cache.</p> <ul style="list-style-type: none"><li>• All applications and users would have access to the new information for those tables.</li><li>• When dealing with large data sources, this is a great balance between fresh and fast as only the changed data is loaded on a pre-defined schedule after the initial bulk load.</li></ul> 
---	---

<p>Merge Reload (Live Data Load)</p> <p>Cached In-Memory + New Data Added In-Memory when user asks</p>	<p>A Merge Reload is a process activated by end users where Qlik loads the changed data for a subset of tables and the data is brought into the Associative Engine for that user.</p> <ul style="list-style-type: none"> <li>• This technology provides a balance between fresh and fast while also ensuring that the end user is confident that they are seeing the most recent values because they initiate the action rather than it being scheduled.</li> <li>• * <b>Note:</b> Merge Reloads typically only load the data for speed and do not regenerate QVD's that would be loaded in the future. Thus, a scheduled Incremental Load is still required so that all other users opening the application have all of the data.</li> </ul>  <p>The diagram illustrates the Merge Reload process. It starts with a user icon (a person with glasses) at the top. An arrow points down to a grid icon representing data tables. Another arrow points down to a Qlik logo icon representing the Associative Engine. A large curved arrow on the right side points from the Qlik logo back up to the user icon, indicating a feedback loop or refresh action.</p>
<p>On Demand Application Generation - ODAG (Live Data Load)</p> <p>Cached In-Memory For selections + Live Data based on Selections in second application when user asks</p>	<p>On Demand Application Generation is a process where an end user passes the selections they have made to a pre-defined template application. Qlik then copies that template, loads data for that specific cohort and presents it to the end user as a new application.</p> <ul style="list-style-type: none"> <li>• The calling application may be fully in-memory with aggregates, offering speed of thought action, while the spawned application then pulls Live details only if and when they are needed.</li> </ul>  <p>The diagram illustrates the On Demand Application Generation (ODAG) process. It starts with a user icon (a person with glasses) at the top. An arrow points down to a Qlik logo icon representing the Associative Engine. Another arrow points down to a gear icon representing a template application. A third arrow points down to a database icon representing data. A large curved arrow on the right side points from the database back up to the user icon, indicating the presentation of the new application. A second Qlik logo icon is shown to the right of the database icon, representing the spawned application.</p>



<p>Dynamic Views (Live Data Load)</p> <p>Cached In-Memory For selections + Charts from Live Data based on Selections when user asks</p>	<p>Dynamic Views is a process similar to ODAG but instead of surfacing a new application to the end user, selected charts from a template application are displayed in the application they are currently using. If the user changes their selections they can ask for the Dynamic Views to be refreshed.</p> <ul style="list-style-type: none"> <li>This option capitalizes on the functionality of ODAG, while also rendering the "live" cohort of data in the context of the application where the users made their selections.</li> </ul>  <p>The diagram illustrates the Dynamic Views process. It shows a user interacting with a Qlik Sense application. The user's selections are processed by the Qlik Sense engine (represented by a Qlik logo chip). The engine then refreshes the data in the application, which is displayed on a monitor. The process is shown as a continuous loop where user selections lead to data refreshes and then to updated charts.</p>
<p>Third Party Solutions (Live Data Load)</p> <p>Cached In-Memory + Charts from Live Data pulled immediately based on selections</p>	<p>There is a Server Side Extension (SSE) provided by Qlik Partner Stretch called "LiveQuery." The SSE currently supports both Snowflake and Google Big Query. So if those are data sources for your applications, this third party extension can help you meet business needs. It does as the name suggests and queries data sources in a live manner. The queries can be isolated, or they can be tied to the user's selection. They are automatically fired whenever the user's selections are changed. The results of the queries are brought back to memory directly and they bypass the Associative Engine so they are not selectable or searchable. If the queries are used as expressions for Master Items, users can ask for Insights that involve them, and can be combined with other measures.</p> <ul style="list-style-type: none"> <li>This solution is similar to Dynamic Views, with the added benefit that it is kept current with end users selections without them having to take any additional actions.</li> </ul>  <p>The diagram illustrates the LiveQuery process. It shows a user interacting with a Qlik Sense application. The user's selections are processed by the Qlik Sense engine (represented by a Qlik logo chip). The engine then queries data sources (represented by a hand holding a database icon) in a live manner. The results of the queries are brought back to memory directly (represented by a magnifying glass icon) and bypass the Associative Engine. The process is shown as a continuous loop where user selections lead to live queries and then to updated charts.</p>

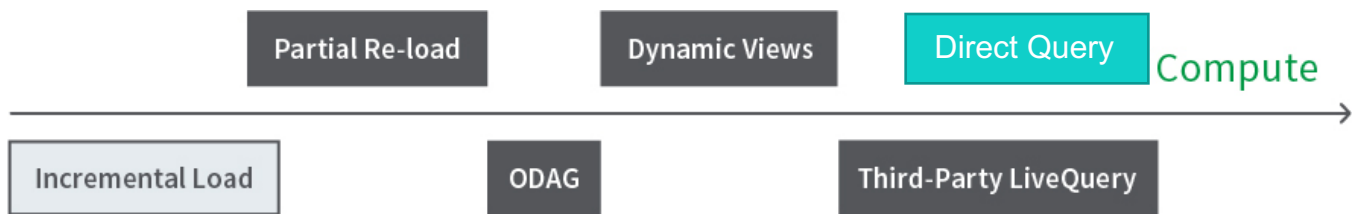


<p><b>Direct Query – Qlik SaaS</b></p> <p>– Real-time Push Down SQL Queries to Snowflake</p>	<p>Direct Query in Qlik Sense allows users to build analytics applications that directly query cloud databases using SQL, as users interact through visualizations and filtering.</p> <p>Customers need direct access to cloud databases to perform queries at the database level for specific use cases.</p> <p>But they also need to intelligently manage compute costs, performance, and user experience.</p> <div data-bbox="1198 220 1453 651" style="text-align: right;"> <p>The diagram illustrates the interaction between cloud databases and Qlik Sense. At the top, a cloud icon is connected to a server icon. A double-headed arrow labeled 'Compute &amp; Processing' connects this to a Qlik Sense icon at the bottom, which is represented as a circuit board with the Qlik logo in the center.</p> </div>
--	--

So which one is right for YOU?

As a primer to choosing the right solution for the right problem, let's begin with some very high level guidance starting with a focus on how much compute and thus cost is driven by each of the solutions. Because in many cases the difference between "We WANT Live Data" and "1 minute old will be FINE" may come down to the implementation cost. In another use case Live Data is a MUST have regardless of the compute resources and cost associated in solving the problem.

The following chart contains no scale on purpose. It is simply for portraying the concept that each technology will have increasing amounts of computing requirements associated with them.



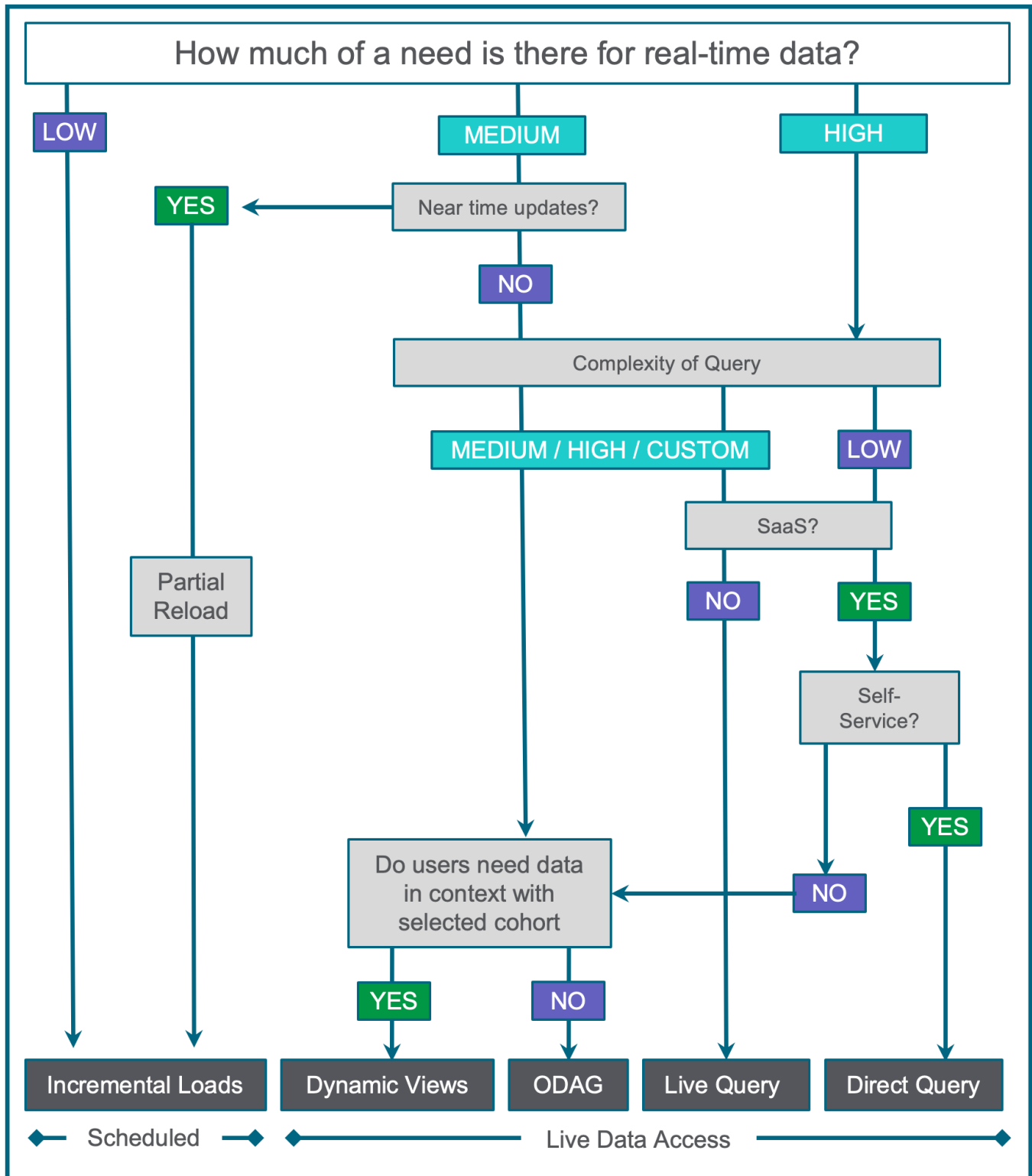
Asking the question "Why does each solution increase the amount of compute/cost?" Well, doing an **Incremental Load** means that Qlik only asks for the changes from our source 1 time per scheduled reload. As **Partial Re-Load's** of data are kicked off by the end user, the same data would need to be consumed by multiple users, thus increasing the compute needed. **On Demand Application Generation (ODAG)** applications will load all of the data for a defined cohort even if it hasn't changed in years. **Dynamic Views** require the same data loading as ODAG applications, and as they can be refreshed as often as desired by the end user to keep in synch with their selections they require more computing power. Finally, as the **third party Server Side Extension** is kept in synch with end user selections, every single end user selection requires the queries to be fired against the data source. Thus, requiring the most computing power.

Remember though, that was only a primer. It is entirely possible that one of your business use cases only requires a single Live Query for an aggregate in an application that is only used by a small set of end users. Thus changing which technology you may wish to use. Bringing us to the next point: Business driven use cases are often complex and require the answers to many questions. The following visual is intended to help provide you a thorough Business Driven Technology Guidance response on an application by application basis. It balances needs and compute power, while answering another, more complex set of questions.

Notice under "Need for Live Data Access" and "# of Users" we use the terms Low/Medium/High with no specific values. You the customer can choose to adjust the meaning of those values based on the budget to pay for the computing power. This balancing act if you will can get complicated, perhaps even more complicated than the chart. The point is that Qlik provides the architectural flexibility to help your organization find and maintain that balance.

## Technology Guidance on Integration Options - Qlik:

For applications that require access to near/real-time data.



The following provides a little more understanding for each of the decision points in the chart.

**"Need for Live Data Access"** - There are two different considerations for needing to read data live. One is based on need and simply implies that for the given application anything less than up to the second values simply won't be accepted. That may be due to political reasons, or actual life changing decisions made in a healthcare setting. The second is altogether different. It is because there is too much data to fit into memory and so subsets of it will need to be read when they are needed.

**"Need to filter values from Live Data WITH in-memory selections"** - This is a huge decision point. ODAG, Dynamic Views and LiveQuery do not allow end users to select values from the real-time data that was just read, and have those selections apply to the parent application. ODAG is a separate application entirely. Dynamic Views show charts from the template application but those charts are not tied to the parent context meaning while you can filter items in the Dynamic View charts, they will remain as separate filters for the parent application.

**"Complexity of Queries"** – This simply refers to the fact that some queries are straight forward and do not require a lot of compute power to perform "Select Sum(Field) From Table." While others might require a lot of joins and/or complicated SQL functions that require a lot of compute. Each customer, and each project might require a different scale for what Low means vs High.

**"# of Users"** – Represents the number of users that will be utilizing the application. Again, each customer/project will need to derive for themselves how

much compute power they wish to pay and thus how many users requesting live values they can sustain. 100 concurrent users may seem low to Customer A, but high to Customer B.

**"Self Service"** - If end users will have access to be able to edit the selection clause, and security to the underlying data is an issue, then the 3<sup>rd</sup> Party LiveQuery solution is not the right choice.

**"SaaS"** – At the time of writing, the Stretch LiveQuery is not supported in SaaS as it an external Server Side Extension. Thus, it isn't an option.

**"Do users need to retain the analysis path/context with the newly selected cohort of data"** - ODAG and Dynamic Views are essentially the same thing with the exception that Dynamic Views display charts from the template application inside the calling application so end users can see the cohort they selected and see the details that are loaded live. While ODAG provides a fully designed application that is popped up, it is separate from the calling application and the user may lose the context of why they had chosen a particular cohort.

The following walks through some hypothetical use case that lines up with each of the architectural solutions.

### Scenario 1 – **ODAG**: Drill to Detail Reporting/Analysis

Starting with a summary application of key metrics, a user chooses a selection of criteria which then is passed to a secondary application that is generated on demand. A customer example of when to use ODAG:

A Human Resources department has an existing application that is used by 1,290 HR administrators, managers and employees through the company who make a lot of selections. The application is currently on a one-hour incremental load schedule. They want to use real time data instead of waiting for reloads to see potential overtime issues. The details are a known subset of content in a specific format that shows potential issues, therefore ODAG provides an interim reload ability in an easily consumable detailed Qlik application.

### Scenario 2 – **Dynamic Views**: Transactional Details in Context

This scenario is best suited for when details for specific transactions need to be viewed in context with the original Qlik Sense application as a chart inside the application. A customer example of when to use Dynamic Views:

A customer has several hundred billion records from a transactional system stored in Snowflake which is too much for a single Qlik application. This data is reloaded on a schedule with the aggregates of KPI's and other relevant data but not the transactions themselves. Dynamic Views are used to get the transactions of a cohort of dimensional values that limit the records to a threshold (say under 100k rows) to be analyzed in any chart on demand. The users of the application need to be able to see the details in the context of the cohort selected.

### Scenario 3 – LiveQuery: Real-Time Status

Sales leaders are asking for an application with several KPI's. They want the new application because the existing application that contains billions of rows of sales transactional data takes too long to load, and they are required to navigate to multiple screens to get the KPI's they need. The application will only be used by Directors and above in the corporation which equates to about 50 people. Expectations are that the users for this application will not interact with filters much, they just need very current metrics.

### Scenario 4 – Partial Reload/Merge: Closing Books / Financial Reporting

Finance leaders closing the books at the end of the month need access to up to the minute general ledger details. The Qlik application contains very complicated calculations, hierarchies, and transformations not easily replicated with SQL. The application will only be used by Accountants and Sr Executive in the corporation which equates to about 25 people. Data is changing rapidly throughout the close cycle and the users need to see where the company stands at any point in time. End users have a very high level of interactivity are required to analyze the data to find issues.



## Personas by Usage: Consumers, Analysts, Designers

Many roles support the data to insight pipeline - from developers to analysts to business casual consumers. Each role though requires certain capabilities within the platform but also levels of access to live data. See the table below to learn which use case fits each persona.

Consumer	Uses prebuilt dashboards, mashups, or pushed content (Reports/Alerts)
Analyst	Deep dives into content, can create own content, understands data literacy
Designer	Builder and deployer of prebuilt content and complex design/data transforms

\*This chart assumes a superset of capabilities (i.e. Designer has Analyst and Consumer capabilities, where Consumers only have the singular capability)

Use Case	In-Mem (FL/IL)	In-Mem (TT IL)	In-Mem (TT Merge)	ODAG	Dynamic Views	In-Mem + Live <sup>1</sup>	Live Only <sup>1</sup>	Direct Query <sup>2</sup>
Business Monitoring	C	C	A			C	C	C
Business Alerting	C	C						
Business Reporting	C	C				C	C	
Embedded / Mashups	C	C	C			C	C	
Ad-hoc Analysis	A	A	A	A	A	A	A	A
Insight Suggestions	C	C	A			C	C	
Insight Exploration	A	A	A					
Storytelling	A	A	A			A	A	
Assisted Data Prep	A							
Data Ingestion (LS)	D	D	D	D	D	D	D	D
Use Advanced Analytics (SSE) <sup>1</sup>	C	C						
Develop Advanced Analytics (SSE) <sup>1</sup>	A	A	A					
Data Modelling and ETL	D	D	D	D	D	D	D	D
Application Design & Development	D	D	D	D	D	D	D	D

<sup>1</sup> Only available on client managed

<sup>2</sup> Only available on SaaS

FL: Full Load

IL: Incremental Load

TT: Snowflake Time Travel

Live: LiveQuery

## **Performance Considerations / Best Practices by Technique**

As previewed, Qlik and Snowflake together offer a range of ingestion techniques with varying levels of loading data into memory with Qlik's analytics engine and purely loading near/real-time data. There are clear strengths and advantages for each technique in addition to some having limited use cases.

It's necessary to understand the caching process between Snowflake and Qlik to distinguish how to optimize each system to support your analytics needs.

## **Data Caching for Performance & Compute Cost Consideration**

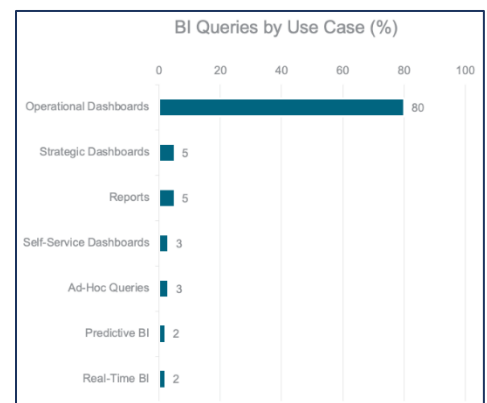
There are 3 different caches involved in processing a query in Snowflake:

- **Snowflake Metadata cache**
  - In the global services layer
  - Stores statistical information on micro-partitions; used to build the query execution plan.
  - Will eventually be "swapped out" by other queries over time; you have NO CONTROL over this cache.
- **Snowflake Data cache (table scan; partitions of a table)**
  - This is a "traditional" data cache, at the micro-partition-level.
  - This IS specific to the virtual warehouse (compute) used to process the query. As a result, if the virtual warehouse is suspended, the cache is (typically) lost. Be aware that suspending and immediately resuming might result in the warehouse never actually suspending (as an optimization) and therefore not flushing the cache.
- **Snowflake Result set cache**
  - This is the "all or nothing" result

- This is maintained by the global services layer of the Snowflake architecture, stored in blob storage (AWS S3, Azure Blob, Google GCS), and is NOT specific to any single virtual warehouse (compute) used to process the query.
- Results are retained for 24 hours, but you can tell the query processor to NOT pull from this cache.

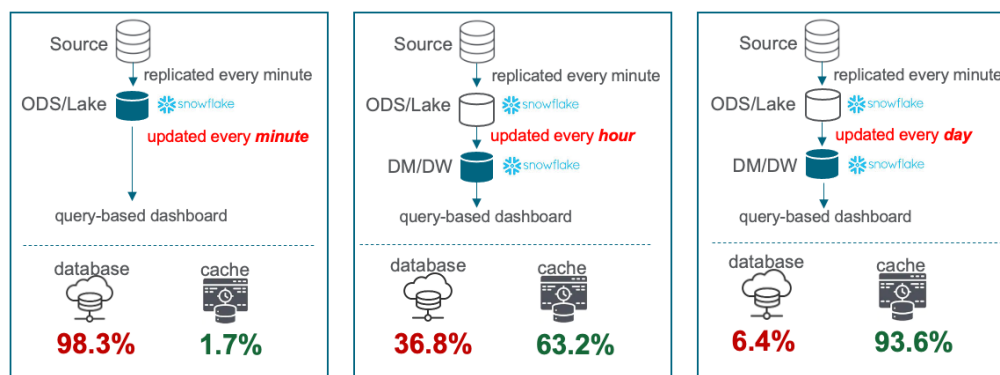
## Why is Caching So Important for Dashboards?

Caching and Query times are directly related and when breaking down the allocation of queries across analytics applications the majority of query volumes originate from Operational Dashboards used by large concurrent users.



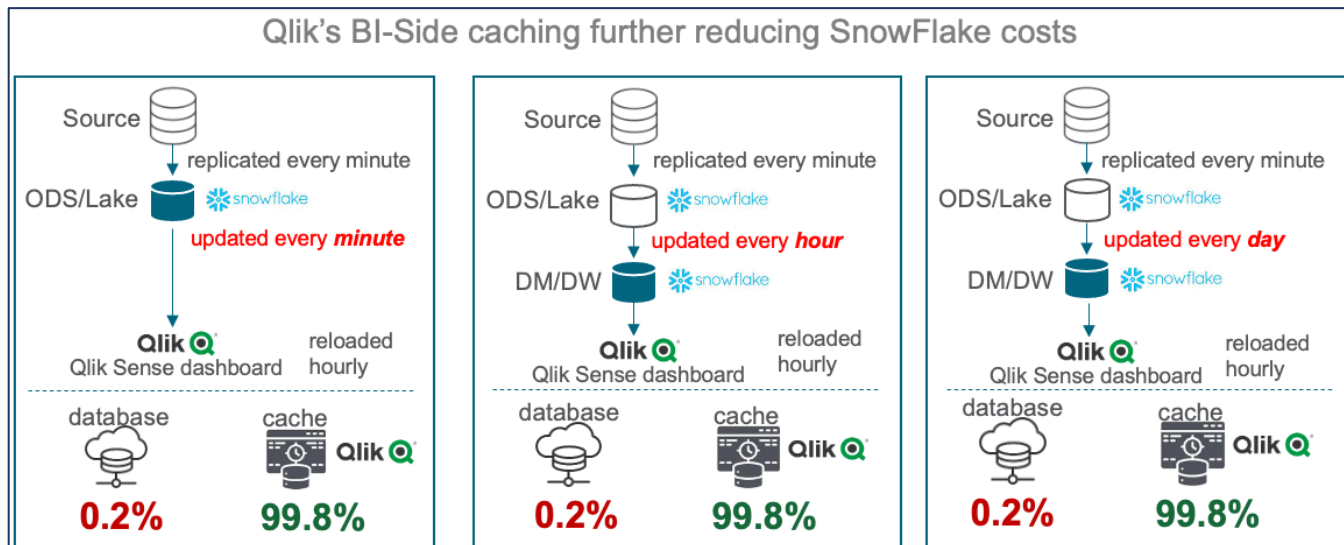
These types of queries are billed by consumption pricing and can be very compute intensive and may require a larger Snowflake warehouse or a multi-cluster warehouse, especially for larger number of users.

## Why is Caching So Important in Snowflake?



There is a huge advantage in **speed** and **cost** to isolate rapid-changing data to a separate DB, because changing data can affect the caching of results.

## Why is Qlik Sense So Great with Snowflake?



**Using Qlik's in-memory cache reduces your Snowflake costs to just reload activity.** All queries between reloads are 100% cached in Qlik's in-memory model, freeing up the Snowflake warehouses to suspend themselves and/or serve other needs, regardless of the update frequency of the underlying data, i.e., if the database tables are reloading daily/hourly then there is no reason for live access queries.

### Key Considerations for Caching & Cost Management with Snowflake

#### Realtime Data Kills Cache, and Costs Cash

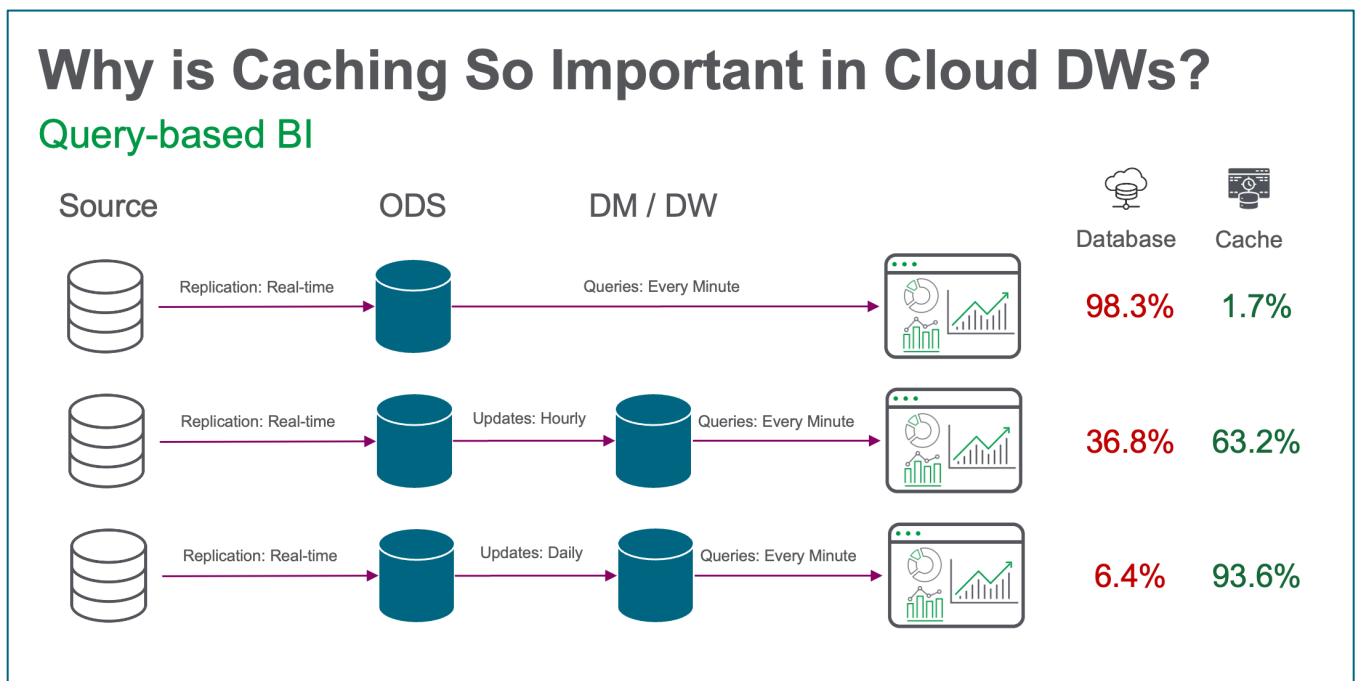
- Snowflake has very good data & result caching, which are potentially impacted if the underlying data is changing rapidly. The result cache will be more severely affected while only the modified micro-partitions of the virtual warehouse cache will be affected.
- Costs are significantly more to have the data updated every 5 min (vs hourly)!

- Do NOT update/consume on same DB – use an intermediary engine like Qlik Compose to reduce compute costs and balance loads.

## Scenarios to Avoid

Direct Query Tool Access against Near/Real Time Data

- More limited Snowflake caching
- More expensive (more credits used)
- Potentially slower performance



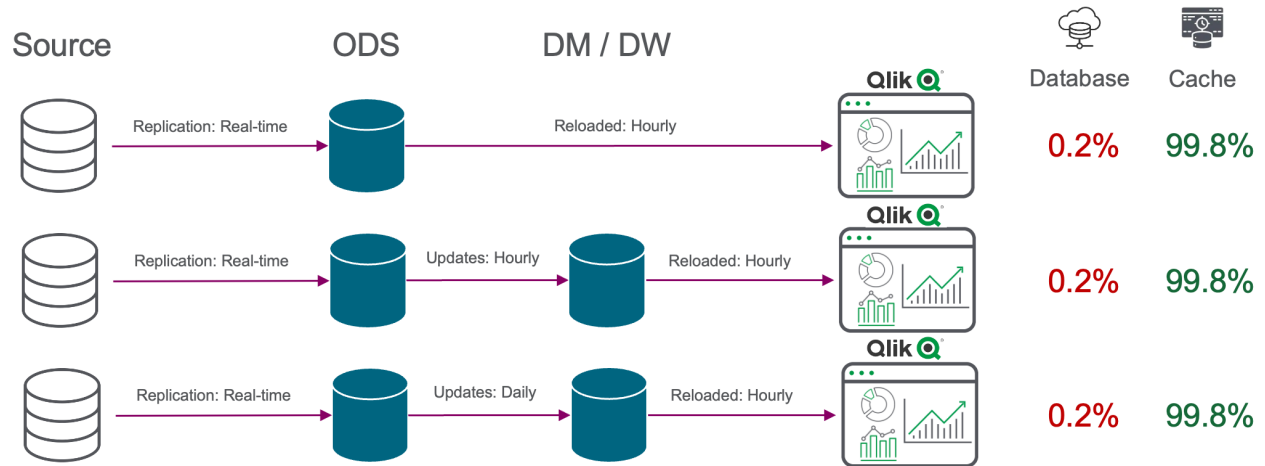
## Best Practice Scenario

Qlik Sense in-memory (or blended use case covered later)

- Best use of Snowflake caching
- Less Expensive (less credits used)
- Best performance

# Why is Qlik Sense a compliment to Cloud DWs?

## Qlik's In-Memory first approach



## Enable Bulk Reader Option

This biggest performance improvement option (Qlik SaaS only) during data load is to Enable Bulk Reader option which can result in up to a 50% improvement of Bulk Data loading for large data sets.


## Load optimization settings

*Load properties that can be configured*

Property	Description	Required
<b>Enable Bulk Reader</b>	Select this to include larger portions of data in the iterations within a load. This may result in faster load times for larger datasets. If not selected, data will be loaded row by row.	No

In the Qlik Sense Snowflake data connection setup, that option is found here:

Edit data connection ⓘ X


 Snowflake

**Miscellaneous**

Allow non-SELECT queries  
Allow non-SELECT queries

Query timeout  
150

**Load Optimization**

Enable Bulk Reader  


Max String Length  
4096

**Advanced**

Name	Value	
<input type="text"/>	<input type="text"/>	<input type="button" value="+"/> <input type="button" value="X"/>

**Name**  
Snowflake TEST

## Incremental Load Options

The following sections will discuss in detail the above strategies on how to leverage Snowflake data depending on use case.

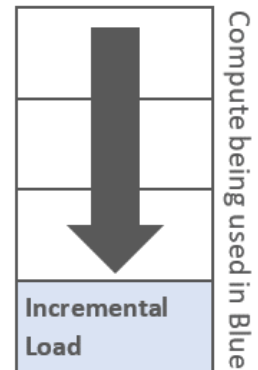
### Incremental Loads using QVD's

This is the most common technique used in Qlik environments to only load delta data.

Qlik has a data format called QVD that is binary and optimized for re-ingestion locally. It is commonly used by Qlik developers as it allows for snapshotting data, reuse of data structures after transformation, combination and pre-calculation of data, and **incremental loads**.

Here is the basic process of how it works:

- 1) Table of data is identified as large enough to not reload entirely every time.
- 2) Load all the data the first time, store data into QVD file on disk
- 3) On subsequent loads:
  - a. Identify a column in the table used for knowing that it is a new record. It can be an incrementing, numerical ID value in the table, or possibly a date/time field.
  - b. After the data is loaded from, get the *highest value* for the aforementioned column and store it in a variable (e.g. "LastDate")
  - c. CONCATENATE LOAD the new data from Snowflake but selecting from the big table and adding a WHERE clause and including the variable from above; For example, "select \* from table where datetimefield > \$(LastDate);"





- d. Save the QVD, overwriting the previous QVD. If desired, snapshot by saving an additional QVD with a timestamp or similar. This way you easily can go back to previous loads if desired.
- e. Repeat for any other tables that have a high number of values.

The above is the basic concept; However, there are nuances to account for, so the following information will dive deeper into the variants.

Here is a help document describing the process: [https://help.qlik.com/en-US/sense/May2022/Subsystems/Hub/Content/Sense\\_Hub/LoadData/use-QVD-files-incremental-load.htm](https://help.qlik.com/en-US/sense/May2022/Subsystems/Hub/Content/Sense_Hub/LoadData/use-QVD-files-incremental-load.htm)

## QVD-based Incremental Load - Insert Only

This method offers a lot more control with the ability to specify a field to do the incremental load on. There are 2 example versions for incremental load logic with dates to illustrate the flexibility and some ideas on how to do it. There are other fields to use that might suit your data better.

This version uses the last timestamp from the source data table itself.

// Example using the last actual create date of the record in the table on Snowflake

// Do the initial load from the QVD:

FactTable:

LOAD

    PrimaryKeyInt,

    X,

    Y,

    CreateTimeStamp

FROM File.QVD;

// Get the max timestamp from the data in the QVD

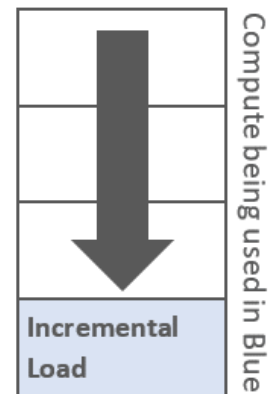
Max:

Load

    max(CreateTimeStamp) as maxtime

Resident FactTable;

// Set the variable to the maxtime derived above:



```
LET ts = peek('maxtime',0,'Max');
```

```
// Append ("Concatenate") the records since the last load from Snowflake table  
into the FactTable:
```

```
Concatenate (FactTable)
```

```
SQL SELECT
```

```
    PrimaryKeyInt,
```

```
    X,
```

```
    Y,
```

```
    CreateTimeStamp
```

```
FROM SnowflakeDB.TableName WHERE CreateTimeStamp >= $(ts);
```

```
STORE FactTable INTO File.QVD;
```

This next version sets the **last execution time** and current **reload start time** variable and uses them both for the date range in the where clause from Snowflake.

```
// Example using the reload execution start and end times.
```

```
//Set start time variable:
```

```
Let ThisExecTime = Now();
```

```
// Do the initial load from the QVD:
```

```
FactTable:
```

```
LOAD
```

```
    PrimaryKeyInt,
```

```
    X,
```

```
Y
FROM File.QVD;

// Append ("Concatenate") the records since the last load from Snowflake table
into the FactTable:
Concatenate (FactTable)
SQL SELECT
    PrimaryKeyInt,
    X,
    Y
FROM SnowflakeDB.TableName WHERE CreateTimeStamp >=
$(LastExecTime)
AND ModificationTime < $(ThisExecTime);

STORE FactTable INTO File.QVD;

// Set reload execution end time to now *if all goes well*:

Let LastExecTime = ThisExecTime;
```

## QVD-based Incremental Load - Insert and Update

This technique is a little trickier conceptually but is a simple script:

Qlik loads records inserted into the database or updated in the database after the last script execution.

A ModificationTime field (or similar) is required for Qlik to recognize which records are new.

A primary key field is required for Qlik to sort out updated records from the QVD file.

This solution will force the reading of the QVD file to standard mode (rather than optimized), which is still considerably faster than loading the entire database.

Example:

```
// Set the reload start time
```

```
Let ThisExecTime = Now( );
```

```
// Ingest just the new records into Qlik first
```

```
FactTable:
```

```
SQL SELECT
```

```
    PrimaryKey,
```

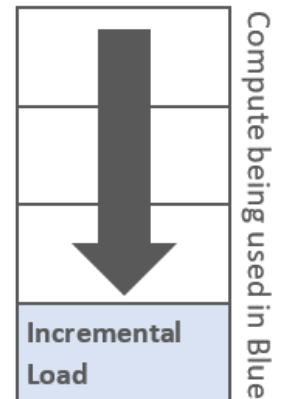
```
    X,
```

```
    Y
```

```
FROM SnowflakeDB.TableName WHERE ModifiedTimeStamp >=
```

```
$(ThisExecTime);
```

```
// Add in the records from the previously saved QVD --- *ONLY* when that record  
doesn't exist in the cohort brought in from Snowflake:
```



```
Concatenate
LOAD
  PrimaryKey,
  X,
  Y
FROM File.QVD
WHERE NOT Exists(PrimaryKey);

STORE FactTable INTO File.QVD;

If ScriptErrorCount = 0 then
STORE QV_Table INTO File.QVD;
Let LastExecTime = ThisExecTime;
End If
```

## QVD-based Incremental Load - Insert, Update and Delete

What if records were **deleted** from the source database **between script executions**? In this case we need to:

Have Qlik remove records deleted from the database after the last script execution

A field ModificationTime (or similar) is required for Qlik Sense to recognize which records are new

A primary key field is required for Qlik Sense to sort out updated records from the QVD file

This solution will force the reading of the QVD file to standard mode (rather than optimized), which is still considerably faster than loading the entire database

Example:

```
// Set the reload start time
```

```
Let ThisExecTime = Now( );
```

```
// Ingest just the new records into Qlik first
```

```
FactTable:
```

```
SQL SELECT
```

```
    PrimaryKey,
```

```
    X,
```

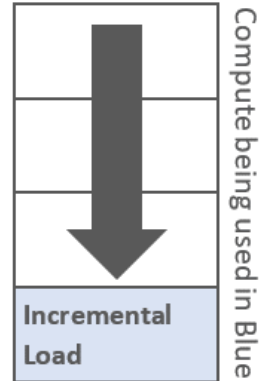
```
    Y
```

```
FROM SnowflakeDB.TableName
```

```
WHERE CreateTimeStamp >= $(LastExecTime)
```

```
AND ModificationTime < $(ThisExecTime);
```

```
// Add in the records from the previously saved QVD --- *ONLY* when that record  
doesn't exist in the cohort brought in from Snowflake:
```



Concatenate

LOAD

    PrimaryKey,

    X,

    Y

FROM File.QVD

WHERE NOT EXISTS(PrimaryKey);

// Do an inner join so that only the PrimaryKey values that still exist in the database will remain

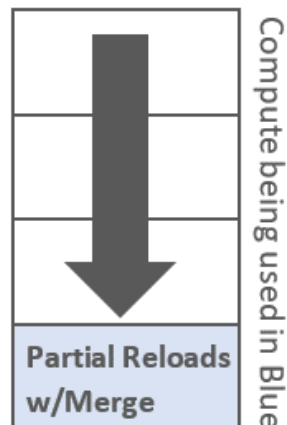
Inner Join SQL SELECT PrimaryKey FROM SnowflakeDB.TableName;

STORE FactTable INTO File.QVD;

Let LastExecTime = ThisExecTime;

### Incremental Loads using “Partial Load”

Partial Load is helpful when you have new data to append to an existing table but do \*not\* want to load the rest of the tables. This is helpful when you have faster moving data in one table and a large data set from many sources in the rest of the data model.



Examples:

**ADD LOAD** \* from Snowflake.FactTable where createdatetime > \$(lastreloaddatetime);



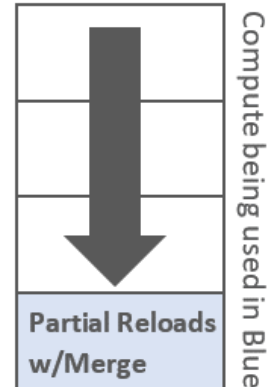
The above will run when the “Partial” option has been set on a reload AND will run when the partial flag is not set.

**ADD ONLY LOAD** \* from Snowflake.FactTable where createdatetime > \$(lastreloaddatetime);

The above will run ONLY when the Partial flag has been set and will \*not\* be run when partial is set to false in the reload.

## Time Travel with Merge Best Practices

Traditionally Qlik developers have utilized a Last Modified Timestamp field so that they could load only the changed values. However, there are many times that data tables don't have a Last Modified Timestamp so then Full Reloads were the only option.



However, Snowflake offers Change Processing as part of their Time Travel functionality. It's the ability to simply get the Change Data Capture information returned in a query. When it is enabled, it adds metadata columns behind the scenes to your tables that allow you to modify your typical SQL Selection and say, "give me only the changes that have occurred since X time."

```
78 SELECT "METADATA$ACTION", cast(CURRENT_TIMESTAMP() as varchar(100)) as "Timestamp_c",
79 "Hospital_Account_ID", "Primary_ICD_Diagnosis_Code", "Admit_ICD_Diagnosis_Code",
80 "Primary_ICD_Procedure_Code", "Primary_Payor_ID", "Total_Account_Balance_$",
81 "Total_Account_Adjustment_$", "Total_Account_Charge_$", "Total_Account_Payment_$", "HRRP_Condition"
82 FROM "GENERALHOSPITALDB"."dbo"."Accounts"
83 CHANGES(INFORMATION => DEFAULT) AT(TIMESTAMP => '2021-06-29 18:14:18.779 -0400'::timestamp_tz) END(TIMESTAMP => CURRENT_TIMESTAMP())
84 where not("METADATA$ISUPDATE" = TRUE and "METADATA$ACTION" = 'DELETE')
```

Results Data Preview

✓ Query ID SQL 853ms 3 rows

Filter result... [Download] [Copy]

Row	METADATA\$ACTIC	Timestamp_c	Hospital_Account_	Primary_ICD_Diagn	Admit_ICD_Diagno	Primary_ICD_Proce	Primary_Payor_ID	Total_Account_Bal	Total_Account_A
1	INSERT	2021-06-29 18...	11111111	777.77	V22.1	74.1	300010	2112700.0000	-65.2000
2	INSERT	2021-06-29 18...	11111122	660.11	660.13	74.1	300029	17600.0000	0.0000
3	INSERT	2021-06-29 18...	11403777	852.05	959.9	39.79	300063	11723577.7000	0.0000

You may be curious about the where function that is asking for the changes, except for Deletes when the action was an update. The key is that Snowflake stores Updates as Deletes and Inserts. So, we only want the Inserts (real insert or updated information) and the real Deletes.

Combining this CDC capture information from Snowflake with the Qlik Merge function offers a highly performant way to do handle Incremental Loads. The Merge function provides a simple way of automatically "merging" changes into an

in-memory table. You simply pass the function a few parameters and the results of the Snowflake Changes query for the table.

```
317 // Merge any changes that have been applied since last data load
318 MERGE (Timestamp_c, latestTimestampVar) on "Hospital_Account_ID" concatenate (Accounts) LOAD *;
319 SQL SELECT "METADATA$ACTION", cast(CURRENT_TIMESTAMP() as varchar(100)) as "Timestamp_c",
320 "Hospital_Account_ID", "Primary_ICD_Diagnosis_Code", "Admit_ICD_Diagnosis_Code",
321 "Primary_ICD_Procedure_Code", "Primary_Payor_ID", "Total_Account_Balance_$",
322 "Total_Account_Adjustment_$", "Total_Account_Charge_$", "Total_Account_Payment_$", "HRRP_Condition"
323 FROM "GENERALHOSPITALDB"."dbo"."Accounts" CHANGES(INFORMATION => DEFAULT)
324 AT(TIMESTAMP => '$(lastTimestampVar)::timestamp_tz' END(TIMESTAMP => CURRENT_TIMESTAMP()))
325 // don't need DELETE row event generated for Updates, only Insert treated as Upsert but deletes are needed
326 where not("METADATA$ISUPDATE" = TRUE and "METADATA$ACTION" = 'DELETE');
327
```

This functionality:

1. Speeds up traditional Incremental Load applications.
2. Can be used to handle Incremental Loads for tables without Last Modified Timestamp columns where you may be doing full reloads.
3. Partial Reloads to allow end users to quickly bring the latest and greatest changes very quickly into memory so that they can analyze them.

## Time Travel Setup & Qlik Merge Function Details

Details about Snowflake Change Tracking can be found here:

<https://docs.snowflake.com/en/sql-reference/constructs/changes.html>

Details about the Qlik Merge function can be found here:

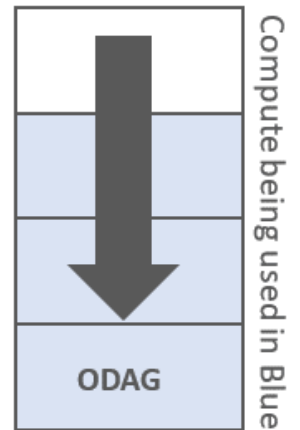
[https://help.qlik.com/en-US/sense/May2022/Subsystems/Hub/Content/Sense\\_Hub/Scripting/ScriptPrefixes/Merge.htm](https://help.qlik.com/en-US/sense/May2022/Subsystems/Hub/Content/Sense_Hub/Scripting/ScriptPrefixes/Merge.htm)

## On-Demand Options

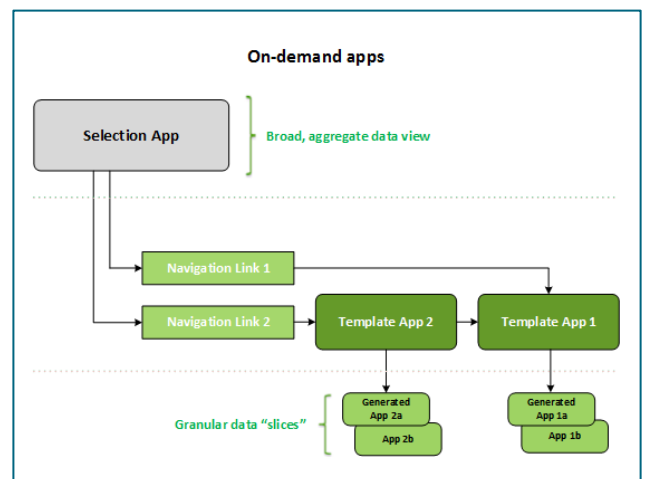
The following sections will discuss in detail the strategies on to leverage real/near time data from Snowflake.

### On Demand App Generation (ODAG)

“On Demand App Generation”, commonly called ODAG, is about bringing in just the right chunk of data you’re interested in to do your analysis. It’s a technique commonly used in **big data** scenarios where it’s just not possible or efficient to load all of the data into their Qlik app. It’s common to see this approach when the fact table exceeds 500-800 million rows (purely estimate, depends on rest of data model).



A simple example might be that user comes in to a “shopping cart app” where they see trends and explore high level aggregated data, then make selections to narrow in on the specific dimension values of interest by using the associativity in the engine. When the user has confirmed that the cohort of data is manageable enough of and has the appropriate parameters chosen, then the user can launch the “analysis app” which has all of the chart objects and layout already defined, or after loading that data it can return a blank sheet if desired.



Here is the official help document for creating and managing On Demand Apps:

[https://help.qlik.com/en-US/sense/May2022/Subsystems/Hub/Content/Sense\\_Hub/DataSource/Manage-big-data.htm](https://help.qlik.com/en-US/sense/May2022/Subsystems/Hub/Content/Sense_Hub/DataSource/Manage-big-data.htm)

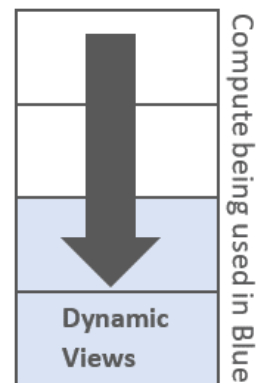
## Dynamic Views

Dynamic views enable you to connect a base app to another app. Master visualizations from that app can then be used in the base app.

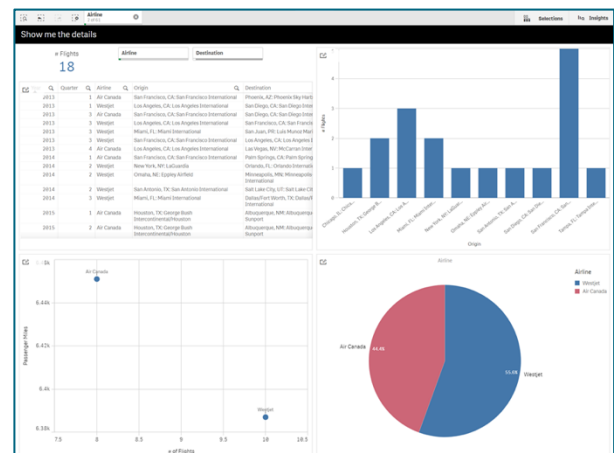
This enables app creators to use master visualizations from the template app as dynamic charts in other apps. There is no limit to the number of dynamic views you can add to your base app.

Dynamic views are made from three main components:

- Dynamic views: A mechanism added to base apps that connect to template apps and enable app creators to add master visualizations from the template app to the base app.
- Dynamic view template apps: A Qlik Sense app containing connections to data sources, such as cloud databases.
- Dynamic charts: Master visualizations in the dynamic view template app that can be added to base apps and that can be manually refreshed by users.



The template app and the base app do not need to use the same data. If you have a data set covering customer purchases, you could add a dynamic view to a template app containing weather data to look at any correlations.



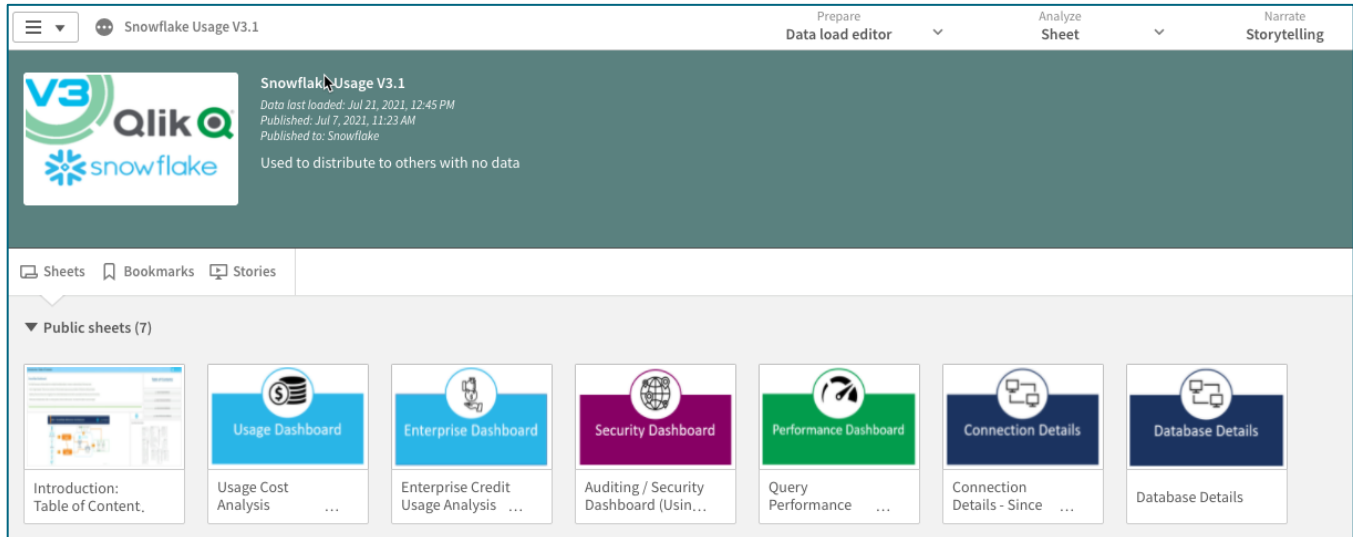
If the data queried from the template app's source can be filtered using values in your base app, you can use binding expressions in the template app's script. This enables the dynamic view to only query a subset of data specifically related to the selections in the base app from the data sources of the template app. For example, you could bind the field SalesDate in the base app to the field DailyTemperatureReadingDate in the template app.

Here is the official help document for creating and managing Dynamic Views:

[https://help.qlik.com/en-US/sense/May2022/Subsystems/Hub/Content/Sense\\_Hub/DynamicViews/dynamic-views.htm](https://help.qlik.com/en-US/sense/May2022/Subsystems/Hub/Content/Sense_Hub/DynamicViews/dynamic-views.htm)

## Qlik Snowflake Usage Dashboard (V3.1)

This Qlik Sense app combines data from multiple Snowflake usage and metadata tables to create an understanding of six key areas.



- **Usage Cost Analysis:** Analysis by various factors how Snowflake credits/spend are being consumed
- **Enterprise Credit Usage Analysis:** Allows investigation of credit/usage spend against a pre-bought credit pack from Snowflake (Enterprise Customers)
- **Auditing/Security:** Tracks logins and location from IP's that access Snowflake (We are using Qlik GeoAnalytics for IP lookup to location). Failed/Successful logins and type of connection used by version
- **Query Performance Analytics:** Tracks details of query performance, find anomalies and issues quickly, also breaks out usage by Qlik product.
- **Connection Details (NEW):** Understand which products are contributing to costs and query usage. Understand all the types of connection strings used to access Snowflake.
- **Database Details (NEW):** Understand the data structures of you Snowflake instance. Columns, Rows, Storage, etc and how all the Databases, Schemas,

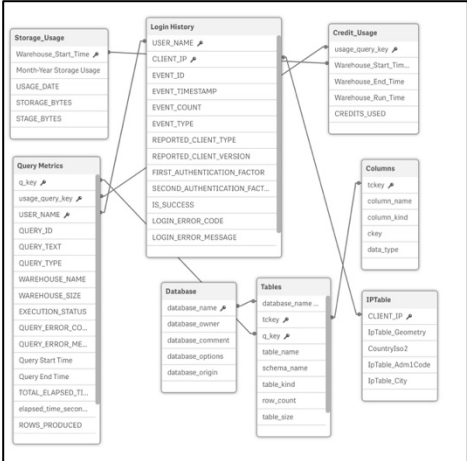
Tables, and Columns correlate. Also tracks shared(external) vs internally owned tables.

Instructions and descriptions are below. The app can be downloaded from: <https://github.com/Qlik-PE/Snowflake-Usage-Analysis-Dashboard>

Upload to your Qlik Sense server, Qlik SaaS, or Qlik Sense Desktop. Follow the instructions in the app to add your Snowflake credentials and update the GeoAnalytics connection or modify to use a public IP lookup service. A demo version using the Qlik Partner Engineering account can be accessed [Here](#).

**Data Model:**

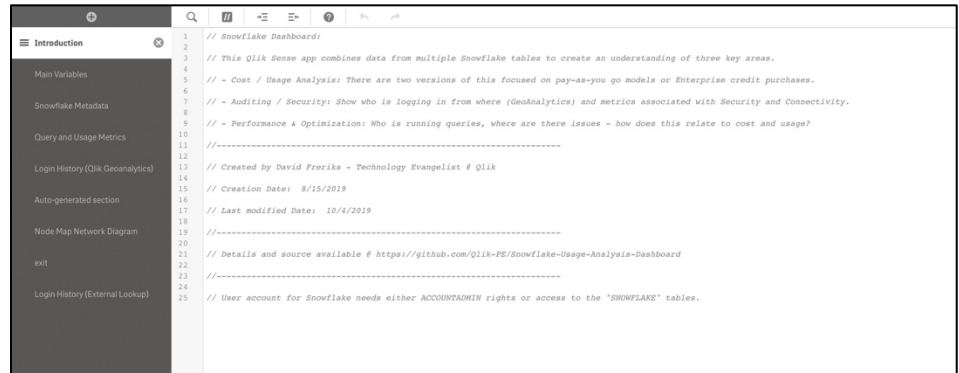
The data is collected from a series of methods and combined in Qlik’s in-memory associative engine. Qlik is unique in that unlike other BI/Visualization tools it can handle multi-grain fact scenarios with data at different levels of aggregation and granularity. For this application, we are combining metadata from databases, tables, and columns with query performance data, login information, storage costs, and usage costs. We also perform dynamic IP lookups to get geospatial information about user IP locations.





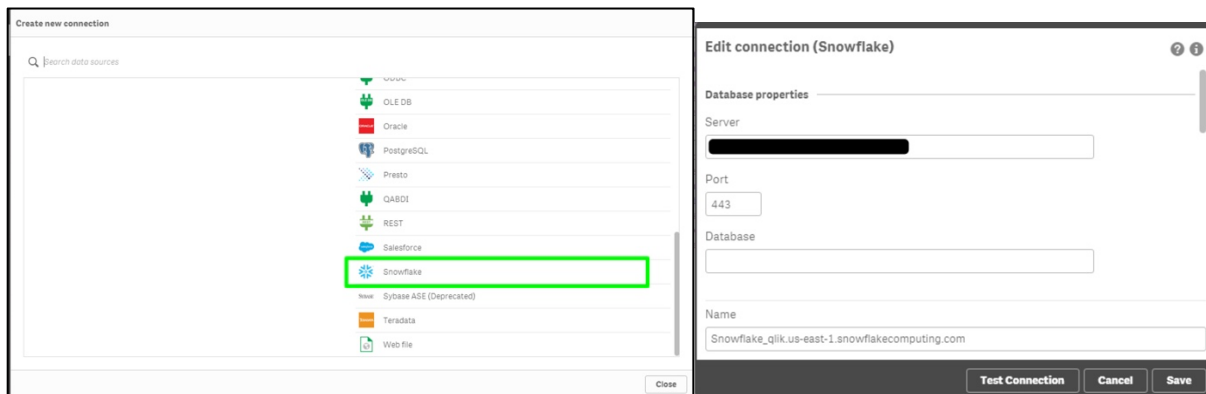
## Data Load Script:

The data is extracted using Qlik load script. The load scripts are how Qlik requests the data from the source tables, SQL functions,



```
1 // Snowflake Dashboard
2 // This Qlik Sense app combines data from multiple Snowflake tables to create an understanding of three key areas.
3
4 // - Cost / Usage Analysis: There are two versions of this focused on pay-as-you go models or Enterprise credit purchases.
5
6 // - Auditing / Security: Show who is logging in from where (GeoAnalytics) and metrics associated with Security and Connectivity.
7
8 // - Performance & Optimization: Who is running queries, where are there issues - how does this relate to cost and usage?
9
10 -----
11
12 // Created by David Prerika - Technology Evangelist @ Qlik
13 // Creation Date: 8/15/2019
14 // Last modified Date: 10/4/2019
15
16 -----
17 // Details and source available @ https://github.com/Qlik-PE/Snowflake-Usage-Analysis-Dashboard
18
19 -----
20
21 // User account for Snowflake needs either ACCOUNTADMIN rights or access to the "SNOWFLAKE" tables.
22
23
24
25
```

and geo-lookups. The model for this application has been broken into logical grouping of similar data by using tabs to help simplify understanding of the data imported. In order to map this application to your instance of Snowflake, you will need to create your own data connection to Snowflake. Starting with the September 2019 release of Qlik Sense Enterprise, there is a built-in connector inside Qlik. Older versions of Qlik Sense will require a download of the ODBC driver from the Snowflake website.



The other element in the load script is the Qlik GeoAnalytics IP lookup. This section of the script takes the unique IP's from the Login History in-memory table and passes them to the GeoAnalytics engine and returns City, State, Country, and Lat/Long values for each IP.

# Analysis Details about the Usage Dashboard

## Table of Contents:

This is the basic introduction to the layout and structure of the app.

**Introduction: Table of Contents**

**Snowflake Dashboard (V3.1):**

This Qlik Sense app combines data from multiple Snowflake tables to create an understanding of four key areas.

- **Cost / Usage Analysis:** What is the cost of your usage?
- **Auditing / Security:** Who is driving usage?
- **Performance & Optimization:** Which queries are driving usage?
- **Connection Details:** Which Qlik products are driving your usage?

**Table of Contents:**

- Usage Dashboard
- Enterprise Dashboard
- Security Dashboard
- Performance Dashboard
- Connection Details
- Database Details

**Application Data Model**

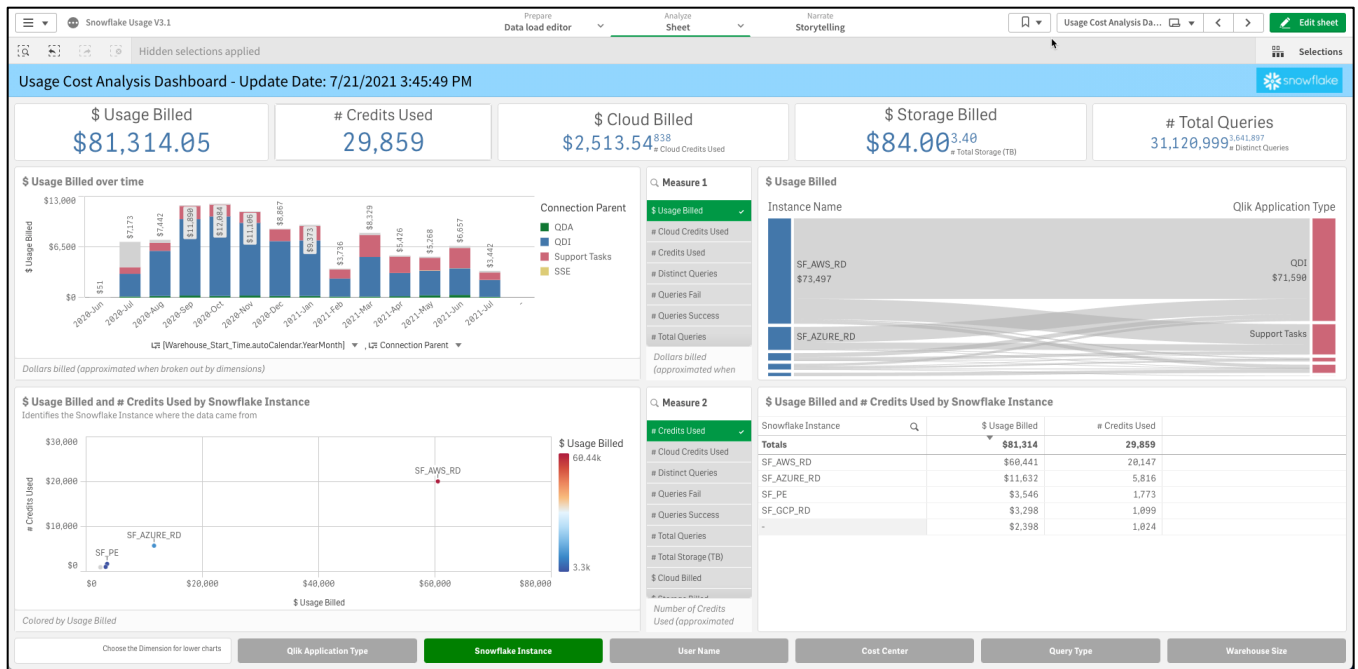
**Qlik + Snowflake Reference Architecture**

**Last reload on - 7/21/2021 3:45:49 PM**

Table Name	# Rows
<b>Totals</b>	<b>55,540,837</b>
Query_History	31,120,999
Login_History	13,135,520
Sessions	9,323,268
Columns	1,851,623
Tables	51,609
Credit_Usage	51,507
IP Table	1,989
Metering Daily History	1,551
Storage_Usage	1,544
Databases	749
IP Table-17	467
ConnectionTypes	7
SnowflakeInstances	4

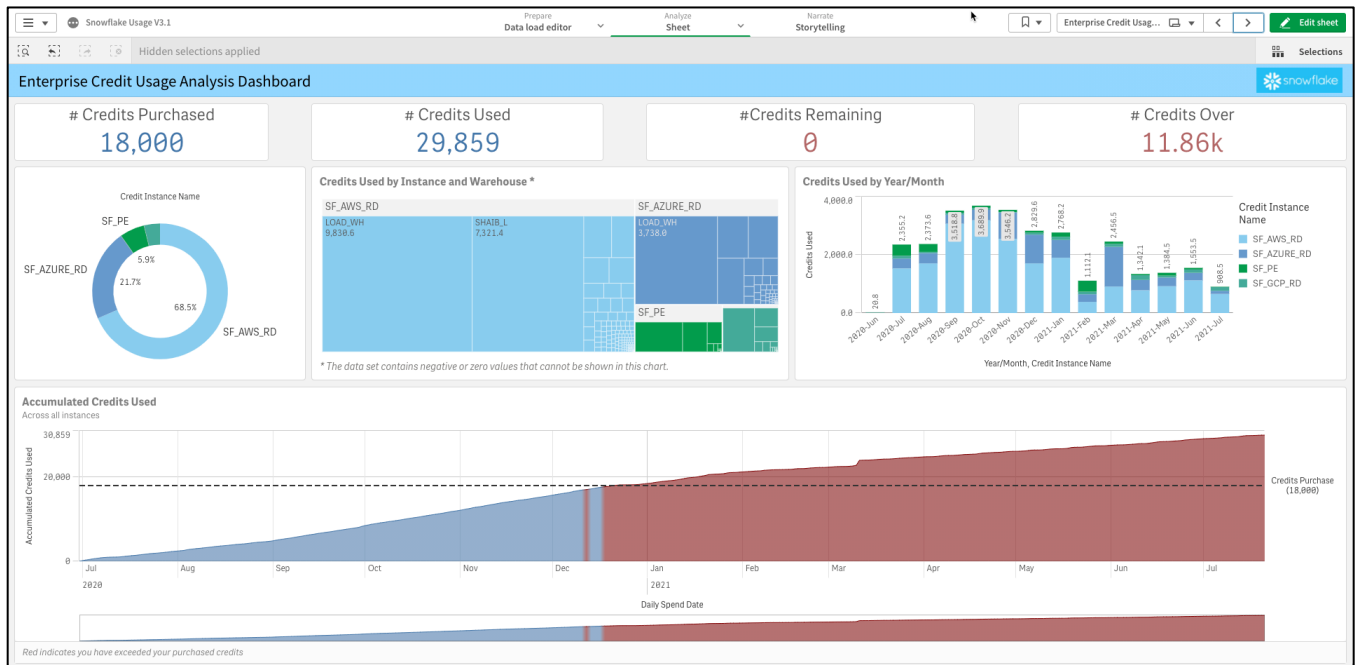
## Usage Cost Analysis:

This dashboard shows costs and usage associated with Snowflake usage. Note how Qlik's engine has mapped estimated costs down to a user level. This is a mixed grain fact situation so costs are dynamically allocated and may not be exact – but does give a general estimate on usage/cost comparison. Users can alter their cost per credit and storage costs based on their unique pricing models that may be applicable.



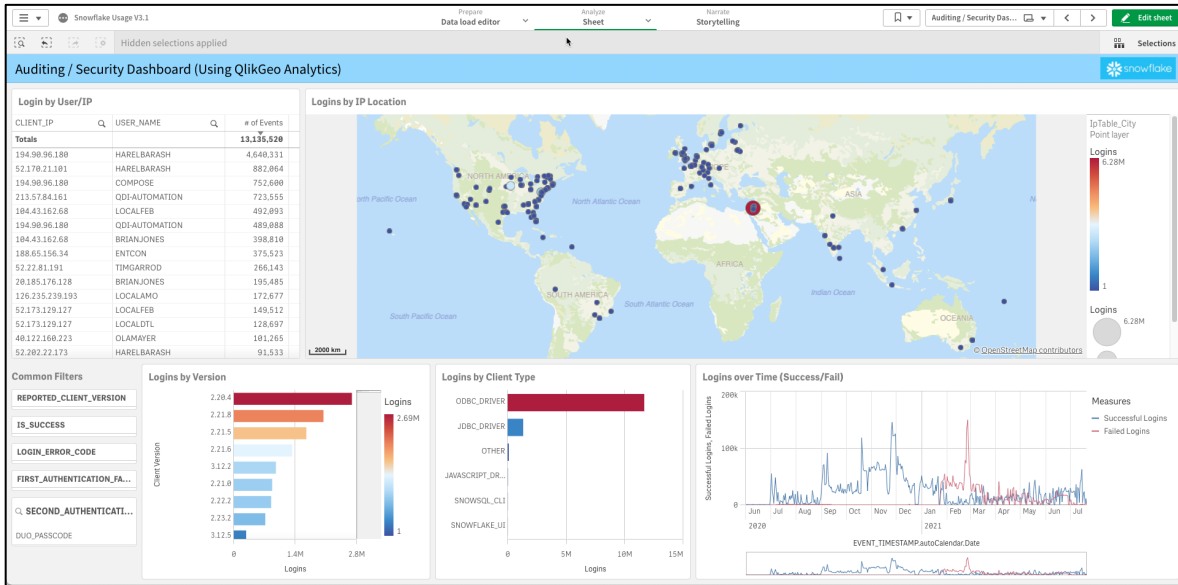
## Enterprise Credit Usage:

This dashboard is based on consumption of pre-purchased credits vs usage. The chart shows when purchased credits are running out.



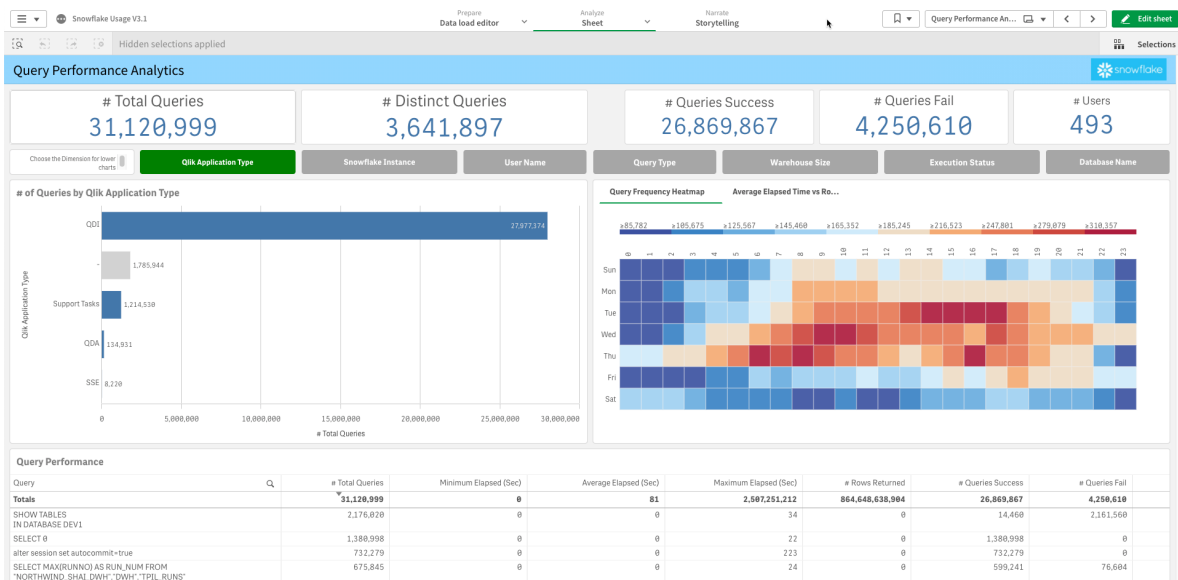
## Auditing / Security:

This dashboard uses the GeoAnalytics IPlookup feature to display where users are logging in from around the globe. Also allows for investigation how users are accessing the system, version of drivers used, and when/how often users are accessing the system.



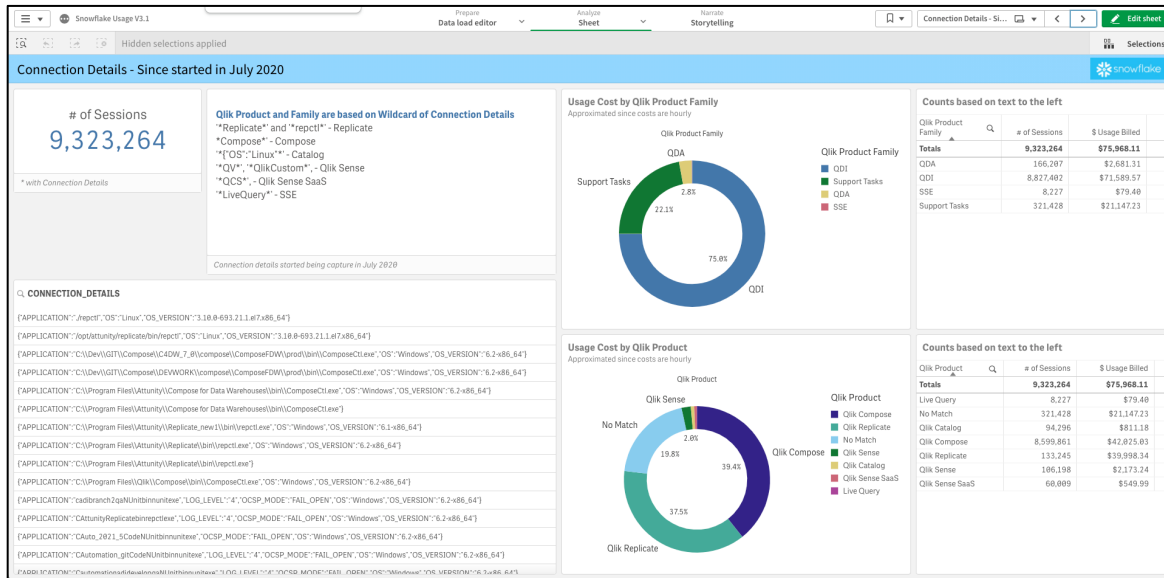
## Query Performance:

This dashboard can be used to understand query performance, usage hotspots, query volumes vs runtime, errors, and dive deep into query details.



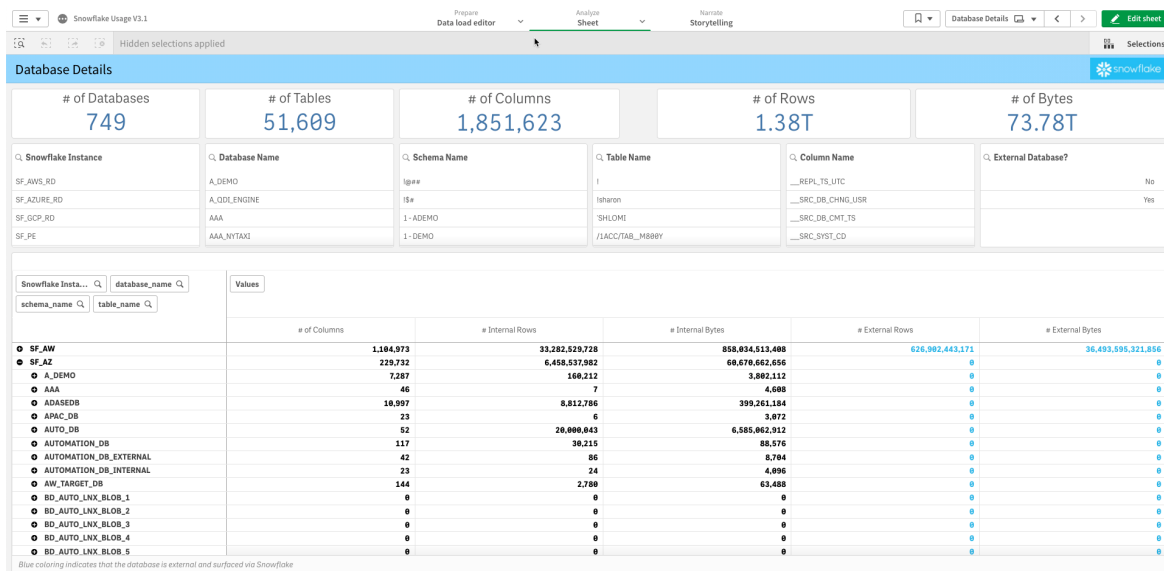
## Connection Details

Using the Wildmatch function Qlik can tag all the connections string used in Snowflake and assign them to a particular product. In this dashboard all Qlik products for both QDI and QDA have been mapped to their respective connection string information in order to understand cost by product/family.



## Database Details

Metadata browser for all Snowflake Instances, Databases, Schemas, Tables, and Columns by internal or external storage.



## Appendix: Connecting Qlik to Snowflake (QSE and Qlik SaaS)

### Making Snowflake Connection

- Native connector is available for connecting to Snowflake.
- ODBC Driver from Snowflake can also be installed and used for client managed version when using Qlik Sense Enterprise.

### Authentication Methods

- To connect to Snowflake, Qlik Sense currently supports username-password method or OAUTH to authenticate using the native connector.
- OAuth also can be used by Qlik Sense client managed to connect to Snowflake currently with ODBC downloaded driver. This can be accomplished using the Simba provided ODBC driver from Snowflake: <https://docs.snowflake.com/en/user-guide/odbc-parameters.html>

### Some Key Snowflake Feature Support via Native Driver:

- Time Travel feature
- Custom SQL feature
- Reading External tables
- Customizable connection string
- Support for Variants / Nested JSON using dot notation
- Full support for Direct/Live connections

**NOTE:** It's important to understand any and all SQL written against Snowflake will work in Qlik Sense in the SQL SELECT component of the Qlik LOAD script.

## Appendix: ODBC Connection Setup

For Qlik Sense Enterprise (Client Managed) Snowflake access there are multiple connectivity options, Native Connection or downloaded and installed ODBC Connection from Snowflake

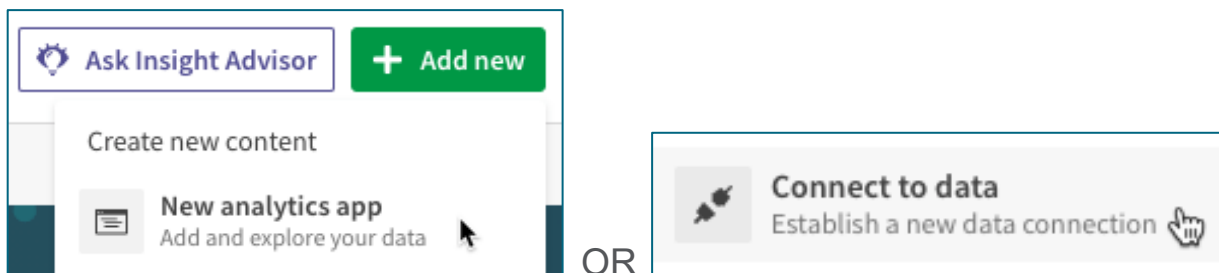
For Qlik Sense SaaS only Native Connection is available.

**For more details and explanations visit the Qlik Help Site:**

[https://help.qlik.com/en-US/connectors/Subsystems/ODBC\\_connector\\_help/Content/Connectors\\_ODBC/Snowflake/Snowflake-connector.htm](https://help.qlik.com/en-US/connectors/Subsystems/ODBC_connector_help/Content/Connectors_ODBC/Snowflake/Snowflake-connector.htm)

**Native Connection (Qlik Sense Enterprise / Qlik SaaS):**

After logging into Qlik (SaaS or Enterprise) – Create a New App or Create a new Data Connection: For this example, I will show using Qlik SaaS.



Create a name and Click Create:

Create a new app ⓘ

Name

Snowflake Test Connection

Space

Personal

Description

Tags ⓘ

Open app

Cancel Create

Start by Adding Data from Sources:

Get started adding data to your app.

Add data from files and other sources

Drag and drop files or click to browse files and connections

Choose the Named Snowflake Connector:

Snowflake

Snowflake



Complete the Connection by filling out the details of the Snowflake Connection (there are two options):

- Username and Password
- OAuth
- Key Pair

### **Username and Password Option Settings:**

- Server (required): Snowflake system name
- Port (required): 443 (SSL)
- Database (optional): If you wish to set a default schema
- Schema (optional): If you wish to set a default schema
- Warehouse (required): Warehouse name (size/compute)
- Role (optional): Will use default role of user unless specified
- Authentication:
  - User Defined Credentials:
    - New credentials: Drop-down menu item that appears if User defined credentials is selected.
    - Existing credentials: Drop-down menu item that appears if User defined credentials is selected.
    - User: User name for the connection.
    - Password: Password for the connection.
    - Credentials name: Name given to a set of user defined credentials.
- Allow Non-SELECT Queries (optional): Allows use of SHOW, USE, DESCRIBE, etc functions...
- Enable Bulk Reader (optional): Speed improvement of Bulk Data load for large datasets

## Advanced Settings: Advanced property / custom settings

Fully configured will look like this (username/password option):

### Create new connection (Snowflake)

**Database properties**

Server  
[redacted].snowflakecomputing.com

Port  
443

Database  
SAP

Schema  
[empty]

Warehouse  
LOAD\_WH

Role  
SYSADMIN

**Authentication**

Authentication Mechanism  
Username and password

**Account properties**

User defined credentials

**Credentials**

Provided credentials are shared with anyone who has access to this data connection.

User  
[redacted]

Password  
[redacted]

**Miscellaneous**

Allow non-SELECT queries

**Load Optimization**

Enable Bulk Reader

**Advanced**

Name	Value	
[empty]	[empty]	+ [trash icon]

**Name**

Snowflake [redacted]east-1.snowflakecomputing.com

Cancel Test connection Create

## OAUTH Option Settings:

- Server (required): Snowflake system name
- Port (required): 443 (SSL)
- Database (optional): If you wish to set a default schema
- Schema (optional): If you wish to set a default schema
- Warehouse (required): Warehouse name (size/compute)
- Role (optional): Will use default role of user unless specified
- Authentication:
  - OAuth: Select this drop-down option to authenticate via OAuth.
  - OAuth Server (Authorize): URL of the authorization server.
  - OAuth Server (Token) : URL of the token server.
  - Client Id: The client id when configuring the OAuth authorization server.
  - Client Secret: The client secret when configuring the OAuth authorization server. This needs to be inputted every time the connection needs to be re-authenticated.
  - Scope If scope offers offline access, re-authentication is automatic. This property is optional.
- Allow Non-SELECT Queries (optional): Allows use of SHOW, USE, DESCRIBE, etc functions...
- Enable Bulk Reader (optional): Speed improvement of Bulk Data load for large datasets
- Advanced Settings: Advanced property / custom settings

## Key Pair Options Setting:

- Server (required): Snowflake system name
- Port (required): 443 (SSL)
- Database (optional): If you wish to set a default schema
- Schema (optional): If you wish to set a default schema
- Warehouse (required): Warehouse name (size/compute)
- Role (optional): Will use default role of user unless specified
- Authentication:
  - Key Pair: Select this drop-down option to authenticate via Key Pair.
  - User : Your user id in Snowflake
  - Private Key File : A copy of the private key used to perform Key Pair authentication
  - Private Key File Password (Optional) : Password for key file if present
- Allow Non-SELECT Queries (optional): Allows use of SHOW, USE, DESCRIBE, etc functions...
- Enable Bulk Reader (optional): Speed improvement of Bulk Data load for large datasets
- Advanced Settings: Advanced property / custom settings

- Configuration Below

**Database properties**

Server  
[redacted] snowflakecomputing.com

Port  
443

Database  
QDA\_PRESALES

Schema  
HANA\_FLIGHT

Warehouse  
QDA\_PRESALES

Role  
SYSADMIN

**Authentication**

Authentication Mechanism  
Key Pair

**Account properties**

User defined credentials

**Credentials**

Provided credentials are shared with anyone who has access to this data connection.

User  
[redacted]

Private Key File Path  
private-key.pem;

Private Key File Password  
[redacted]

**Miscellaneous**

Allow non-SELECT queries

Query timeout  
600

**Load Optimization**

Enable Bulk Reader

Max String Length  
4096

**Advanced**

Name	Value	
<input type="text"/>	<input type="text"/>	<input type="button" value="+"/>

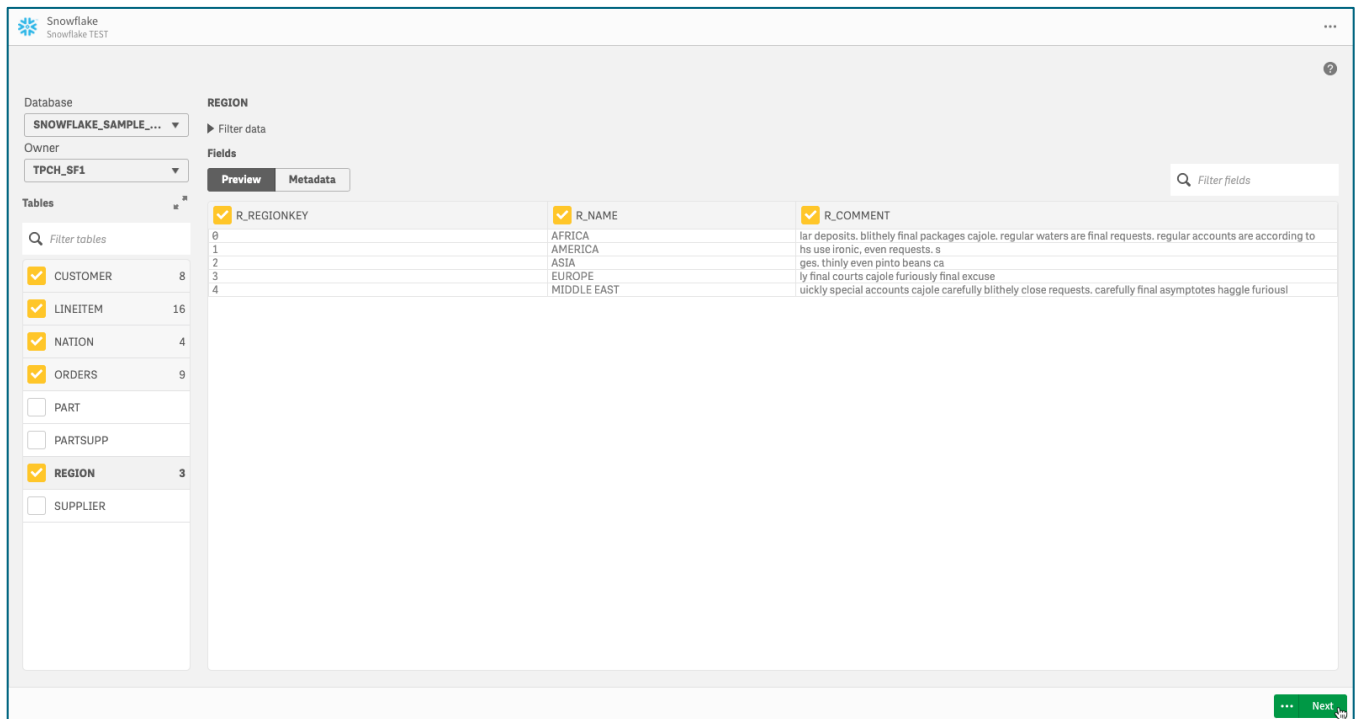
**Name**  
Snowflake - Keypair

If everything is correct, you will get:



Click CREATE.

Now we will choose the data we want, for this scenario, we will use Snowflake sample data from TPCH\_SF1.



Select NEXT:

The sample data will be loaded and profiled.

The screenshot shows the Qlik Sense interface with a data model diagram. The diagram consists of four nodes: REGION\*, NATION\*, CUSTOMER\*, and LINEITEM\*. The CUSTOMER node is highlighted with a thick border. Below the diagram, a table titled 'CUSTOMER' is displayed with the following data:

C_CUSTKEY	C_NAME	C_ADDRESS	C_NATIONKEY	C_PHONE	C_ACCTBAL	C_MKTSEGMENT	C_COMMENT
1	CustomerP000000001	VhnlApeRb ot_cE	15	25-989-741-2988	711.56	BUILDING	to the even, regular platelets. regular, ironic epitaphs nag e
2	CustomerP000000002	ISTM_NcWDrWMe4EgwfMRLChKak	13	23-768-687-3665	121.65	AUTOMOBILE	l accounts. blithely ironic theodolites integrate boldly. careful deposits eat slyly ironic, even instructions. express foxes detect slyly, blithely even accounts above
3	CustomerP000000003	MGS&TQ2WB9fm	1	11-719-748-3364	7498.12	AUTOMOBILE	requests. final, regular ideas sleep final accounts will have to unwind. foxes cajole accounts
4	CustomerP000000004	XVYSJLAG6tn	4	14-128-190-5944	2866.83	MACHINERY	n accounts will have to unwind. foxes cajole accounts
5	CustomerP000000005	KopyuHCpR84WgAGV6tpzq7T	3	13-750-942-6364	794.47	HOUSEHOLD	n accounts will have to unwind. foxes cajole accounts

One the data is connected in the Associative Model, we can LOAD DATA and test the app using Insights:

The screenshot shows the Qlik Sense interface with data insights for the CUSTOMER table. The insights are displayed in a grid layout:

- Distribution of sum(L\_ORDERKEY) for N\_NAME:** A world map showing the distribution of sum(L\_ORDERKEY) by N\_NAME. The map is color-coded by region, with a legend for N\_NAME Area Layer and L\_ORDERKEY (2,167).
- sum(L\_ORDERKEY) by R\_NAME and N\_NAME:** A treemap chart showing the distribution of sum(L\_ORDERKEY) by R\_NAME and N\_NAME. The treemap is color-coded by region, with a legend for R\_NAME and N\_NAME.
- sum(L\_ORDERKEY) by N\_NAME and L\_RETU...:** A treemap chart showing the distribution of sum(L\_ORDERKEY) by N\_NAME and L\_RETU... The treemap is color-coded by N\_NAME and L\_RETU... with a legend for N\_NAME and L\_RETU... (1,117).
- Overview of sum(L\_ORDERKEY) by N\_NAME:** A bar chart showing the distribution of sum(L\_ORDERKEY) by N\_NAME. The y-axis is labeled L\_ORDERKEY and ranges from 1T to 2T.
- Contribution of N\_NAME to overall sum(L\_O...:** A bar chart showing the contribution of N\_NAME to the overall sum(L\_ORDERKEY). The y-axis is labeled L\_ORDERKEY and ranges from 1T to 3T. The x-axis is labeled N\_NAME and ranges from 1 to 10. The chart shows a trend line and a percentage of 50.00%.
- sum(L\_ORDERKEY) by L\_SHIPDATE (YTD) an...:** A bar chart showing the distribution of sum(L\_ORDERKEY) by L\_SHIPDATE (YTD). The y-axis is labeled L\_ORDERKEY and ranges from 4T to 8T. The x-axis is labeled L\_SHIPDATE and ranges from 1 to 3. The chart shows a trend line and a percentage of 50.00%.

The data has successfully loaded!

## Appendix: ODBC Connection (Qlik Sense Enterprise)

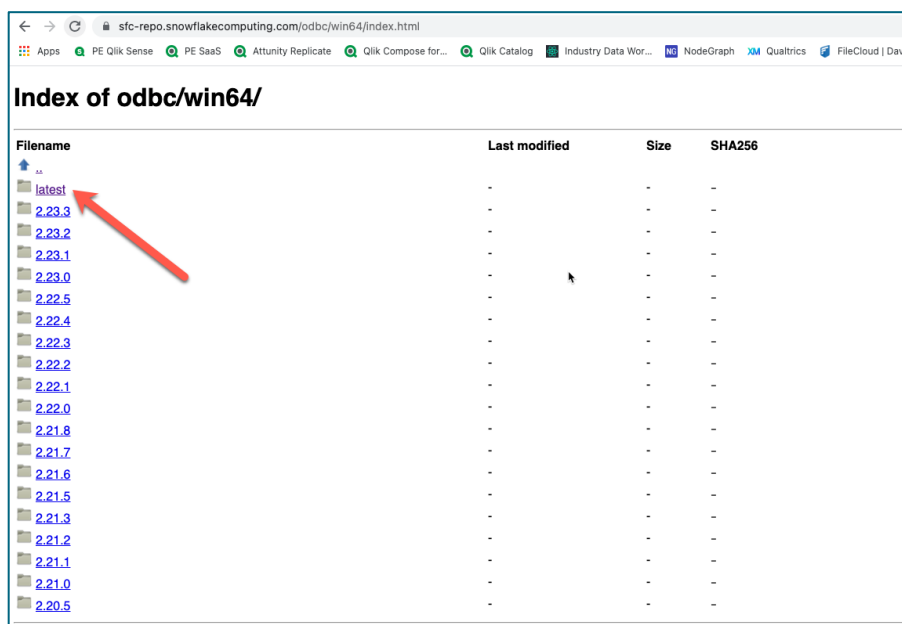
This capability is helpful when new features are released by Snowflake outside of the Qlik upgrade cycle.

### Download the ODBC driver

<https://docs.snowflake.com/en/user-guide/odbc-download.html>

The installer for the Snowflake ODBC driver is distributed through the Snowflake web interface. Before downloading the driver, you may want to first verify the version of the driver you are currently using. To verify your driver version, connect to Snowflake through a client application that uses the driver and check the version. If the application supports executing SQL queries, you can call the `CURRENT_CLIENT` function.

To download the installer for the latest version of the driver for your platform (example for Windows x64): <https://sfc-repo.snowflakecomputing.com/odbc/win64/latest/index.html>



The screenshot shows a web browser window with the address bar displaying `sfc-repo.snowflakecomputing.com/odbc/win64/index.html`. The page title is "Index of odbc/win64/". Below the title is a table with the following columns: "Filename", "Last modified", "Size", and "SHA256". The table lists several folders representing different versions of the ODBC driver, with a red arrow pointing to the "latest" folder.

Filename	Last modified	Size	SHA256
..	-	-	-
latest	-	-	-
2.23.3	-	-	-
2.23.2	-	-	-
2.23.1	-	-	-
2.23.0	-	-	-
2.22.5	-	-	-
2.22.4	-	-	-
2.22.3	-	-	-
2.22.2	-	-	-
2.22.1	-	-	-
2.22.0	-	-	-
2.21.8	-	-	-
2.21.7	-	-	-
2.21.6	-	-	-
2.21.5	-	-	-
2.21.3	-	-	-
2.21.2	-	-	-
2.21.1	-	-	-
2.21.0	-	-	-
2.20.5	-	-	-



## Appendix: Installing and configuring the ODBC Driver for Windows

---

Download can be found at:

<https://docs.snowflake.net/manuals/user-guide/odbc-windows.html>

Step 1. Double-click on the downloaded .msi file:

snowflake64\_odbc-<version>.msi

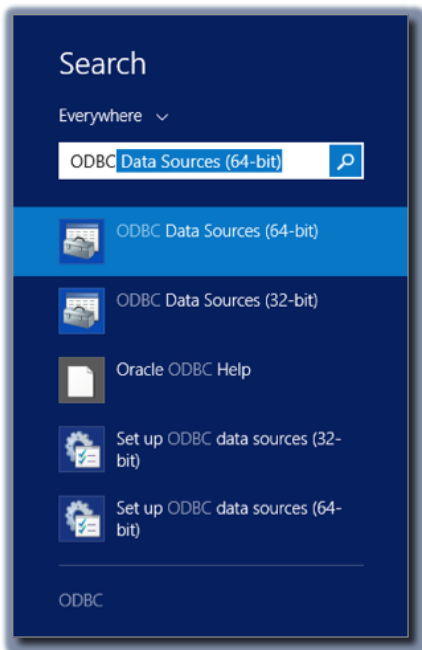
snowflake32\_odbc-<version>.msi

Step 2. Configure the ODBC Driver

To configure the ODBC driver in a Windows environment, create a DSN for the driver:

Launch the Windows Data Source Administration Tool:

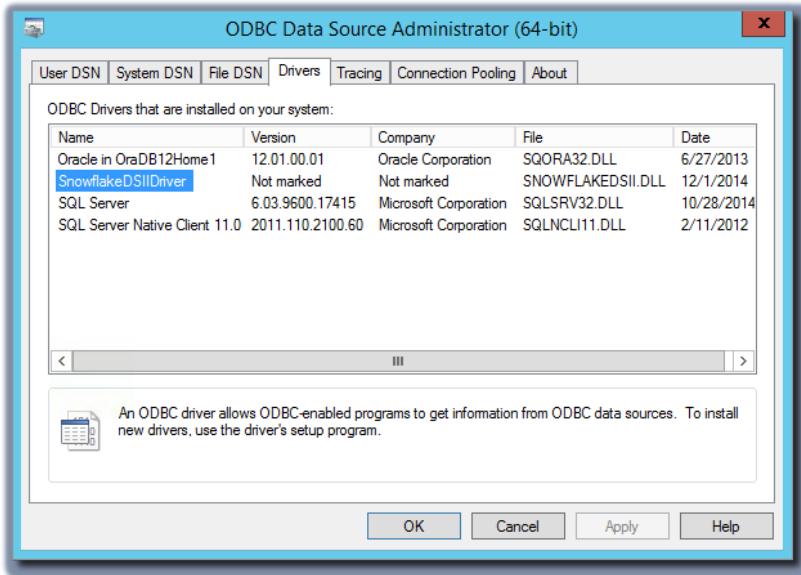
Search on your Windows machine for the launcher for the ODBC Data Source Administration Tool:



Once you find the ODBC administration tool, click on the tool to launch it and display the set-up window.

Verify that the Snowflake ODBC driver is installed:

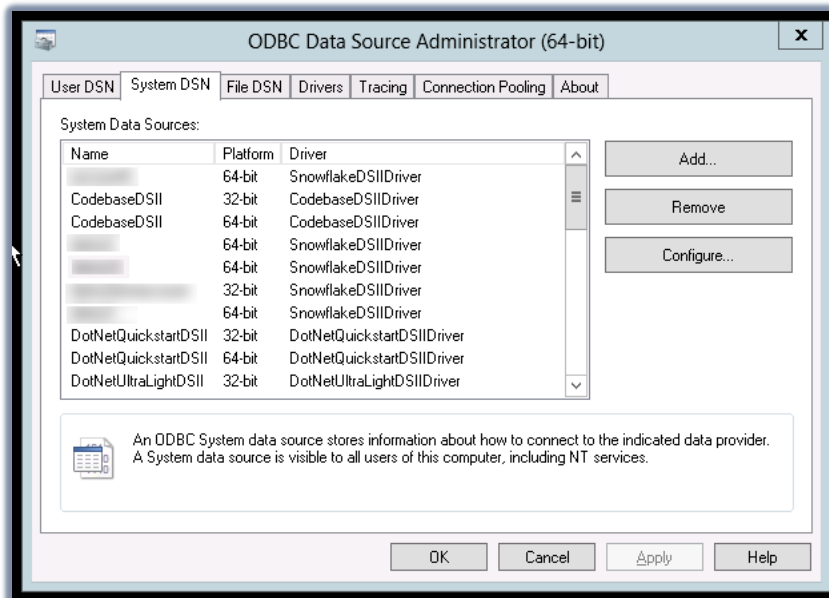
Navigate to the **Drivers** tab in the set-up window and verify that the driver (SnowflakeDSIIDriver) appears:



If you do not see **SnowflakeDSIIDriver**, then the Snowflake ODBC driver installation did not complete successfully and you need to re-install it.

Create a new DSN:

Navigate to the **User DSN** or **System DSN** tab and click the **Add** button:



Select **SnowflakeDSIIDriver** from the list of installed drivers.

Enter the connection parameters for the driver.

In the fields provided in **Snowflake Configuration dialog**, enter the parameters for the DSN:

The image shows a 'Snowflake Configuration Dialog' window. It contains the following fields and values:

- Data Source: SNOWFLAKE
- User: sglover
- Password: (empty)
- Server: qlik.us-east-1.snowflakecomputing.com
- Database: SNOWFLAKE\_SAMPLE\_DATA
- Schema: PUBLIC
- Warehouse: DEMO\_WH
- Role: (empty)
- Tracing(0-6): 4
- Authenticator: (empty)
- Proxy: (empty)
- NoProxy: (empty)

Buttons: OK, Cancel

When entering parameters, note the following:

**Data Source**, **User** and **Server** are the only parameters required to create a DSN.

The **Password** field accepts a value but does **not** store the value. This is a security precaution to ensure passwords are never stored directly in the driver.

All other parameters in the dialog are optional.

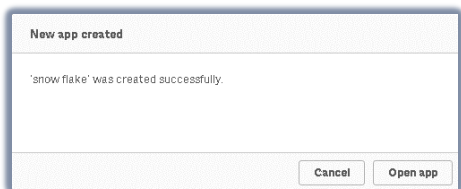
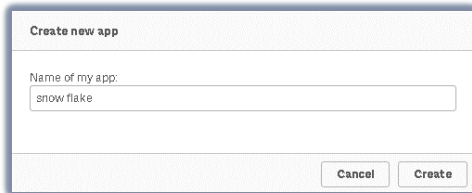
## Appendix: Qlik Sense Configuration

### Install & Configure Qlik Sense

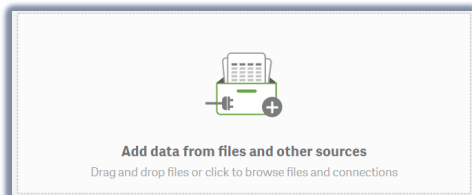
This is not covered in this guide, as we pre-assume a running Qlik Sense system. If you need to setup Qlik Sense – download Qlik Sense desktop ([Qlik Sense Desktop](#)).

### Creating the Qlik Sense App

Step 1. Open Qlik Sense and create a new App



Step 2. Select - Add data from Files and other sources



Step 3. Select ODBC



## Step 4. Create new connection

**Create new connection (ODBC)**

**User DSN** | **System DSN**

32-bit  64-bit

- Aster ODBC
- Datastax
- DataStax Cassandra ODBC DSN
- Denodo7ODBC
- gbigq
- Sample Amazon Redshift DSN
- Simba Spark
- SNOWFLAKE**

Username: sglover Password: .....

Name: SNOWFLAKE

Cancel Create

## Step 5. Add data to the app

**Add data to snow flake**

- New
- IN-APP
  - Manual entry
- FILE LOCATIONS
  - My computer
- DATA CONNECTIONS
  - ODBC SNOWFLAKE**
- DATA CONTENT
  - Qlik DataMarket

ODBC SNOWFLAKE

Database: Select database...

Owner: Select owner...

Tables: [Search] →

Data preview Metadata

Add data to snow flake

Database: SNOWFLAKE\_SAMPLE\_DB  
Owner: TPCDS\_SF10TCL

Tables: ITEM (22)

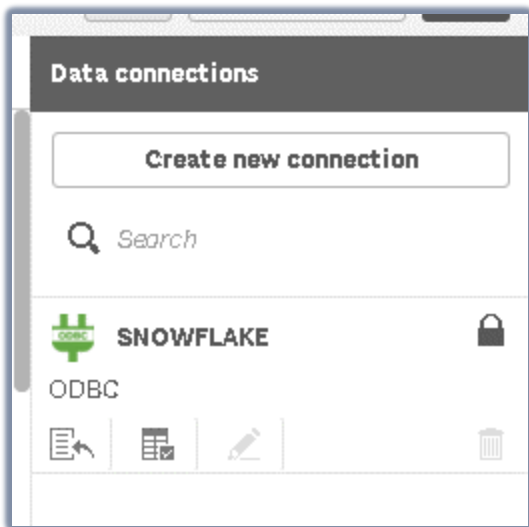
ITEM_ID	ITEM_ID	LREC_START_D...	LREC_END_D...	LITEM_DESC	LCURRENT_PR...	LWHOLESALE_C...	LBRAND...	LBRAND
109922	AAAAAAACGNKBAAA	10/27/1997	10/26/2000	Italian, upper children would give ra	0.15	0.09	10016001	corpamalgam #1
109923	AAAAAAACGNKBAAA	10/27/2000		Years should not take only only subj	97.07	64.06	10016001	scholarunivmalg #2
109924	AAAAAAAEONKBAAA	10/27/1997		Common witness try empty skills. Ed	55.07	44.05	3007001	importexport #1
109925	AAAAAAAEONKBAAA	10/26/1999	10/26/2001	Adequate, other journals choose at	1.35	44.05	3007001	scholarcorp #6
109926	AAAAAAAEONKBAAA	10/27/2001		Adequate, other journals choose at	4.51	44.05	7007007	brandbrand #7
109927	AAAAAAAHGNKBAAA	10/27/1997		Additional, new items can see force	5.22	3.49	0010008	corpmaxi #8
109928	AAAAAAAHGNKBAAA	10/27/1997		Things meet more apart from a spor	7.32	3.07	3007001	importexport #1
109929	AAAAAAAHGNKBAAA	10/27/2000		Things meet more apart from a spor	3.96	3.07	3007002	importexport #2
109930	AAAAAAAHGNKBAAA	10/27/1997	10/27/1999	Laws should speak less from a cuts	6.37	4.06	3004001	edu packexport #1
109931	AAAAAAAHGNKBAAA	10/28/1999	10/26/2001	Males meet	66.44	4.96	3004001	amalgamalgamalg #10
109932	AAAAAAAHGNKBAAA	10/27/2001		Males meet	4.38	4.96	3004001	edu packedu pack #1
109933	AAAAAAAHGNKBAAA	10/27/1997		Home new banks speak from the cli	1.52	1.06	7007002	exportimport #7
109934	AAAAAAAHGNKBAAA	10/27/1997	10/26/2000	Indicators vary even in view of a ori	1.56	1.27	7007001	importimport #1
109935	AAAAAAAHGNKBAAA	10/27/2000		Indicators vary even in view of a ori	1.13	1.27	0006006	corpnameless #6
109936	AAAAAAAHGNKBAAA	10/27/1997	10/27/1999	Departments should administer too	72.06	64.66	0003007	univnameless #7
109937	AAAAAAAHGNKBAAA	10/28/1999	10/26/2001	Departments should administer too	0.78	0.61	0003007	univnameless #8
109938	AAAAAAAHGNKBAAA	10/27/2001		Internal, male structures	0.33	0.28	0003007	namelessbrand #9
109939	AAAAAAAHGNKBAAA	10/27/1997		Different sections keep yesterday el	5.21	3.95	7010010	univnameless #10
109940	AAAAAAAHGNKBAAA	10/27/1997		Rates used to like, military flowers	1.73	0.77	0005009	scholarmax #9
109941	AAAAAAAHGNKBAAA	10/27/2000		Rates used to like, military flowers	0.61	4.99	7009002	maxbrand #2
109942	AAAAAAAHGNKBAAA	10/27/1997	10/27/1999	Details set through a members. Poli	4.06	2.08	4001001	amalgadu pack #1
109943	AAAAAAAHGNKBAAA	10/28/1999	10/26/2001	Details set through a members. Poli	08.14	2.95	0006004	corpnameless #4
109944	AAAAAAAHGNKBAAA	10/27/2001		Quiet, various systems ask small, pr	5.82	45.97	0006004	namelessmax #1
109945	AAAAAAAHGNKBAAA	10/27/1997		Num	2.87	1.66	4004002	edu packedu pack #2
109946	AAAAAAAHGNKBAAA	10/27/1997	10/26/2000	Men would need strictly popular, po	8.21	3.77	0007003	brandmax #3
109947	AAAAAAAHGNKBAAA	10/27/2000		Men would need strictly popular, po	9.74	3.77	0007003	importunivmalg #9
109948	AAAAAAAHGNKBAAA	10/27/1997	10/27/1999	Things handle never rare boys. Figu	6.18	4.07	0008009	corpnameless #9
109949	AAAAAAAHGNKBAAA	10/26/1999	10/26/2001	Troops support here important sect	1.61	4.07	0007010	brandnameless #10
109950	AAAAAAAHGNKBAAA	10/27/2001		Manufacturers should not enjoy onc	0.66	6.73	0007010	importedu pack #1
109951	AAAAAAAHGNKBAAA	10/27/1997		Familiar, final connections would fig	0.70	6.01	10007013	importunivmalg #13
109952	AAAAAAAHGNKBAAA	10/27/1997	10/26/2000	Likely flats turn particularly	32.50	16.57	0008009	namelessmax #9
109953	AAAAAAAHGNKBAAA	10/27/2000		Clear, young pupils gain corporate p	7.57	1.67	1001002	amalgamalg #2
109954	AAAAAAAHGNKBAAA	10/27/1997	10/27/1999	Foreign minutes see against a aspec	6.05	5.38	3007001	importexport #1
109955	AAAAAAAHGNKBAAA	10/28/1999	10/26/2001	Countries might not keep capital, st	3.00	1.83	3007001	edu packedu pack #2
109956	AAAAAAAHGNKBAAA	10/27/2001		Years might not	7.67	1.63	3007001	edu packexport #1
109957	AAAAAAAHGNKBAAA	10/27/1997		Hard sales know grudgingly perhaps	24.96	69.91	0006002	corpmaxi #2
109958	AAAAAAAHGNKBAAA	10/27/1997	10/26/2000	Careful minutes find upon a movem	7.67	2.37	0009009	maxnameless #9

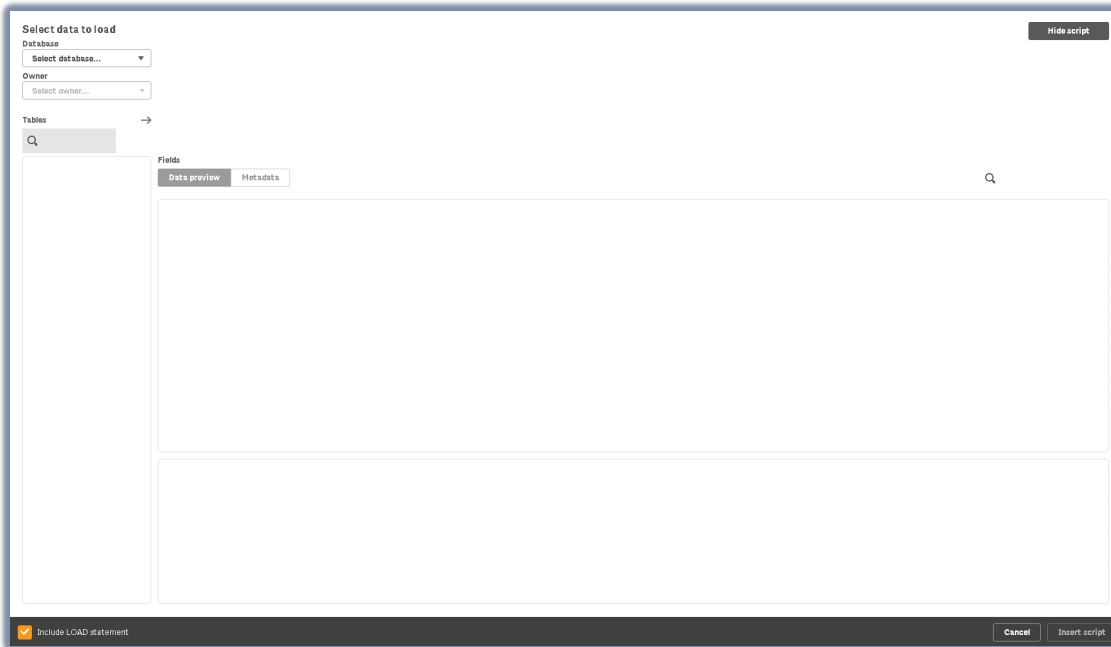
Rows: 402800

Filter fields

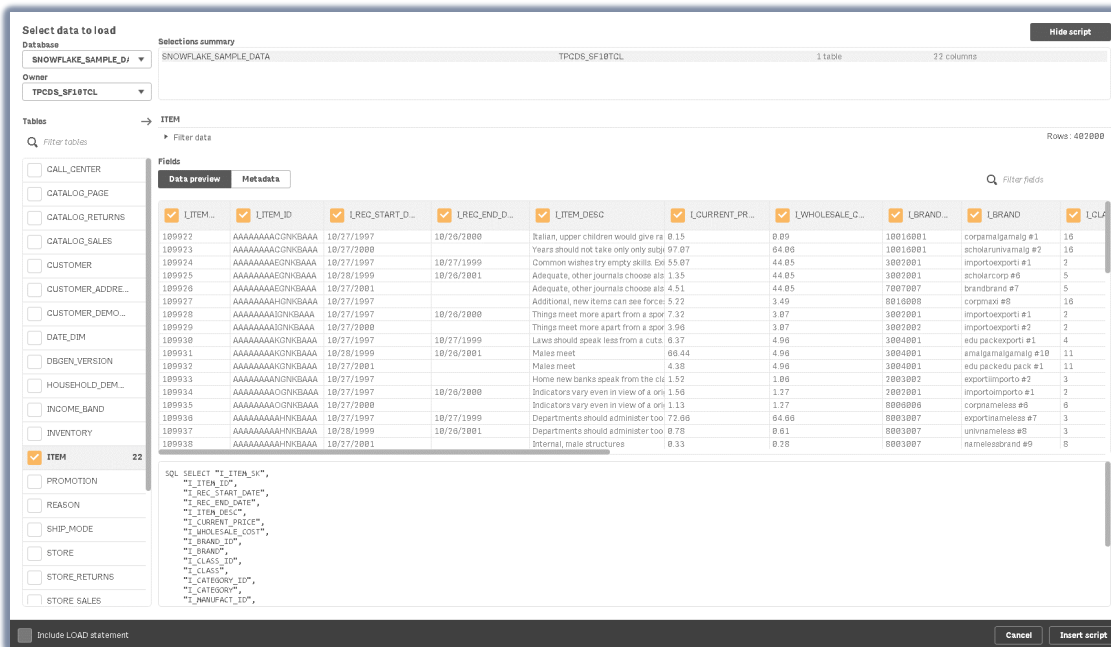
... Add data

You can also load data by “select data “





Select the “Database” and the “Owner” and click on “Insert Script”



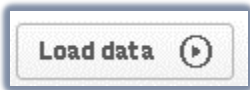
Script inserted in the “Main” section

```

384      "I_PRODUCT_NAME"
385 FROM "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF10TCL".ITEM
386 limit 10000;
387
388 //LIB CONNECT TO 'SNOWFLAKE';
389
390 SQL SELECT "P_PROMO_SK",
391           "P_PROMO_ID",
392           "P_START_DATE_SK",
393           "P_END_DATE_SK",
394           "P_ITEM_SK",
395           "P_COST",
396           "P_RESPONSE_TARGET",
397           "P_PROMO_NAME",
398           "P_CHANNEL_EMAIL",
399           "P_CHANNEL_EMAIL",
400           "P_CHANNEL_CATALOG",
401           "P_CHANNEL_TV",
402           "P_CHANNEL_RADIO",
403           "P_CHANNEL_PRESS",
404           "P_CHANNEL_EVENT",
405           "P_CHANNEL_DEMO",
406           "P_CHANNEL_OTHER",
407           "P_PURPOSE",
408           "P_DISCOUNT_ACTIVE"
409 FROM "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF10TCL".PROMOTION;
410
411
412
413
414
415 LIB CONNECT TO 'SNOWFLAKE';
416
417 SQL SELECT "I_ITEM_SK",
418           "I_ITEM_ID",
419           "I_SEC_START_DATE",
420           "I_SEC_END_DATE",
421           "I_ITEM_DESC",
422           "I_CURRENT_PRICE",
423           "I_WHOLESALE_COST",
424           "I_BRAND_ID",
425           "I_BRAND",
426           "I_CLASS_ID",
427           "I_CLASS",
428           "I_CATEGORY_ID",
429           "I_CATEGORY",
430           "I_MANUFACT_ID",
431           "I_MANUFACT",
432           "I_ITEM",
433           "I_FORMULATION",
434           "I_COLOR",
435           "I_UNITS",
436           "I_CONTAINER",
437           "I_PACKAGE_ID",
438           "I_PRODUCT_NAME"
439 FROM "SNOWFLAKE_SAMPLE_DATA"."TPCH_SF10TCL".ITEM;
440
441

```

Click on “Load data”



Data load “Progress”

**Data load progress**

**Loading data...**  
Please wait while data is loaded. This may take several minutes depending on the data volume.

Elapsed time 00:00:00

**Started loading data**

Connecting to SNOWFLAKE  
Connected

Close when successfully finished Abort



**Data load progress**

**Data load is complete.**

Elapsed time **00:01:48**

```

Lines fetched: 10,000
STORE_SALES_temp_5f7bc892-72e5-738e-2756-087218f1 <<
STORE_SALES
Lines fetched: 10,000
INVENTORY_temp_a1c8c257-7e78-70c2-2a1e-274ed0f1 << INVENTORY
Lines fetched: 10,000
INCOME_BAND_temp_77d5642b-30b2-0fa9-740c-b8e49e95 <<
INCOME_BAND
Lines fetched: 20
ITEM_temp_83f74b5c-e61d-aa13-a057-672f1adb << ITEM
Lines fetched: 10,000
PROMOTION_temp_4db6ad8b-f814-66c4-1f8e-502e81de << PROMOTION
Lines fetched: 2,000
Creating search index
Search index creation completed successfully

```

**App saved**

**Finished successfully**

0 forced error(s)

0 synthetic key(s)

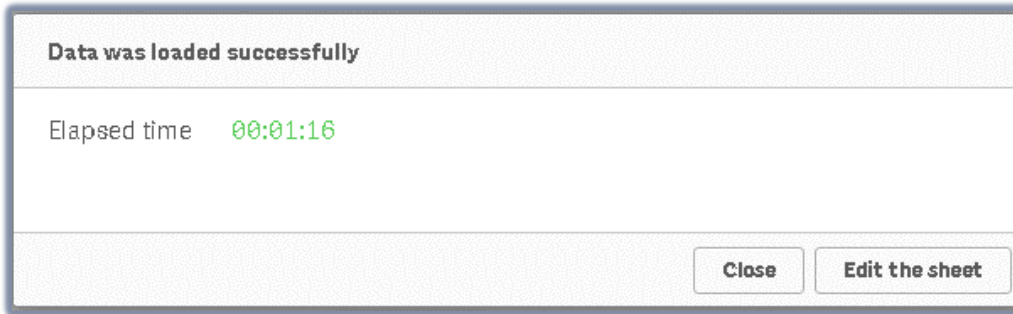
Close when successfully finished Close

Go to “Data Manager” and build the “Associations”

The screenshot shows the Qlik Data Manager interface. At the top, there are tabs for 'Associations', 'Tables', and 'Load data'. The main area displays a data model diagram with various nodes connected by lines. The nodes include ITEM, WEB\_SALES, WEB\_RETURNS, STORE\_SALES, INVENTORY, CATALOG\_SALES, CATALOG\_PAGE, INCOME\_BAND, WAREHOUSE, PROMOTION, TIME\_DIM, WEB\_PAGE, WEB\_SITE, CALL\_CENTER, CATALOG\_RETURNS, CUSTOMER, and CUSTOMER\_ADDRESS. A table view is visible at the bottom, showing columns for call center details. The table has 11 columns: CC\_CALL\_C..., CC\_CALL\_CENTR..., CC\_REC\_ST..., CC\_REC\_IN..., CC\_CLOSE..., CC\_OPND\_L..., CC\_NAME, CC\_CLASS, CC\_BMRD..., CC\_SLT, CC\_HOURS, CC\_MANAGER, CC\_MKT\_ID, CC\_MKT\_CLASS, and CC\_MKT\_DESC. The table contains 4 rows of data.

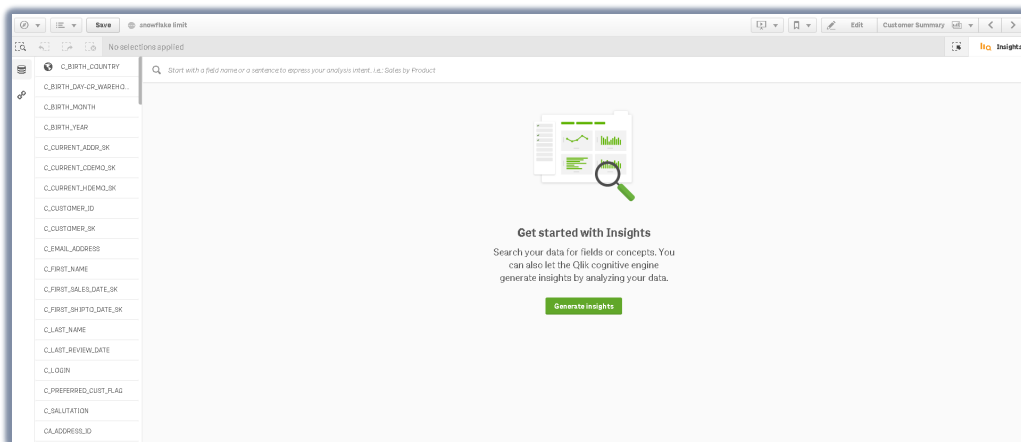
CC_CALL_C...	CC_CALL_CENTR...	CC_REC_ST...	CC_REC_IN...	CC_CLOSE...	CC_OPND_L...	CC_NAME	CC_CLASS	CC_BMRD...	CC_SLT	CC_HOURS	CC_MANAGER	CC_MKT_ID	CC_MKT_CLASS	CC_MKT_DESC
1	AAAAAAAAA	1/1/1999			2458053	MidAtlantic	large	477687035	1218659367	BAM-4PM	Bob Sawyer		More than other author	Shared stores could not members.ca
2	AAAAAAAAA	1/1/1999	11/31/2000		2458006	MidAtlantic	medium	244870074	1924996540	BAM-BAM	Felipe Perkins		A bit narrow forms matter animals. Corisat	Largely quite years out others. Question
3	AAAAAAAAA	1/1/2001			2458006	MidAtlantic	medium	244870074	1317957443	BAM-4PM	Mark Hightower		Wrong troops that work sometimes in a oct	Largely quite years out others. Question
4	AAAAAAAAA	1/1/1993	3/1/2000		2451063	North Midwest	medium	698330337	-2830274463	BAM-4PM	Larry McCray		Dealers make most historical, direct students	Rich groups catch large

Click on “Load Data”



Click on “Edit the sheet” and add the charts to the dashboard

## Step 6. Generate Insights...



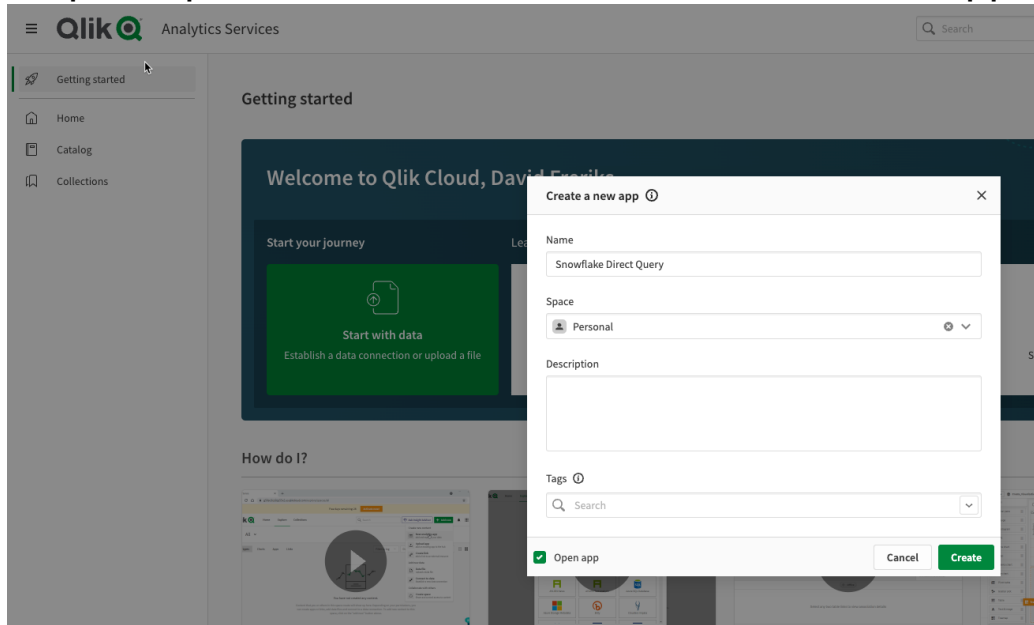
## Step 7. Explore!

By either using insights or directly building on the canvas, we can build our app exploring Customer summary.

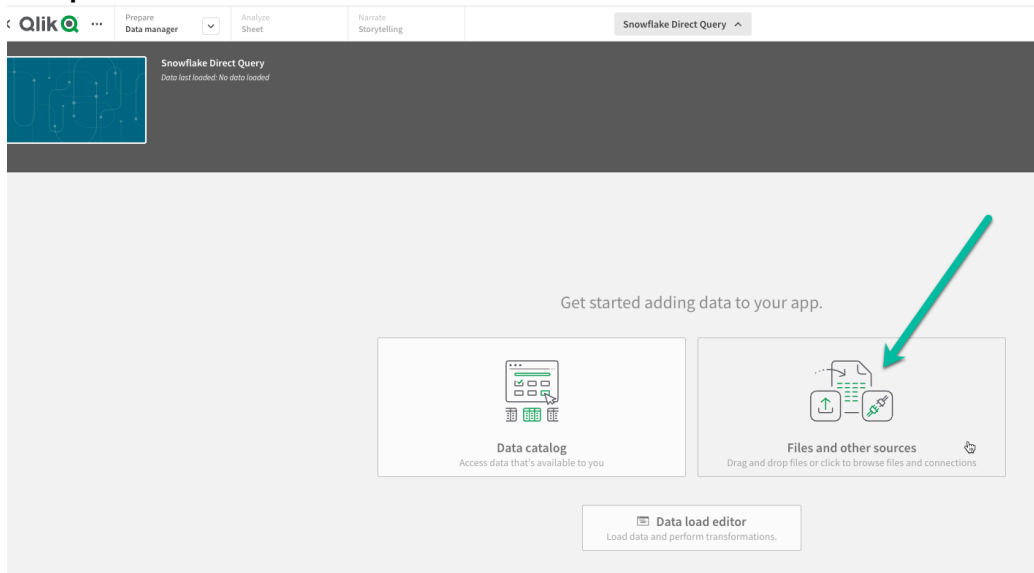
## Appendix: Using Qlik Sense SaaS Direct Query

Direct Query as discussed above is a new option to connect to Snowflake with live queries.

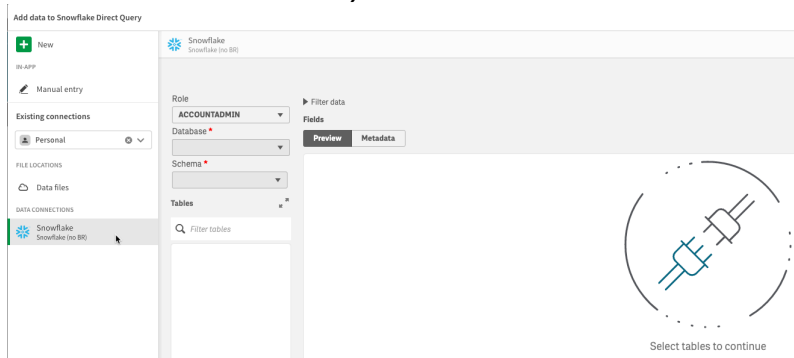
### Step 1. Open Qlik Sense SaaS and create a new app



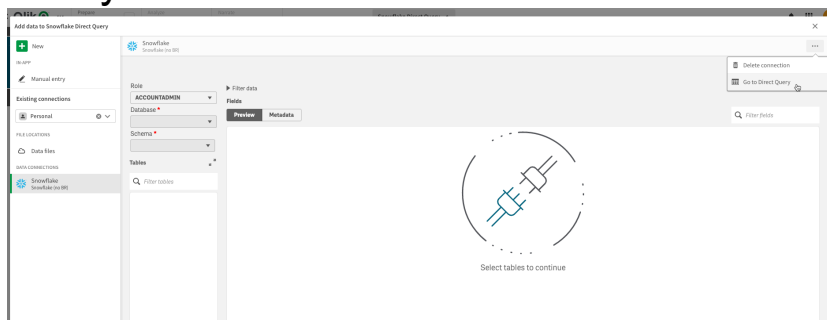
### Step 2. Choose Files and other sources



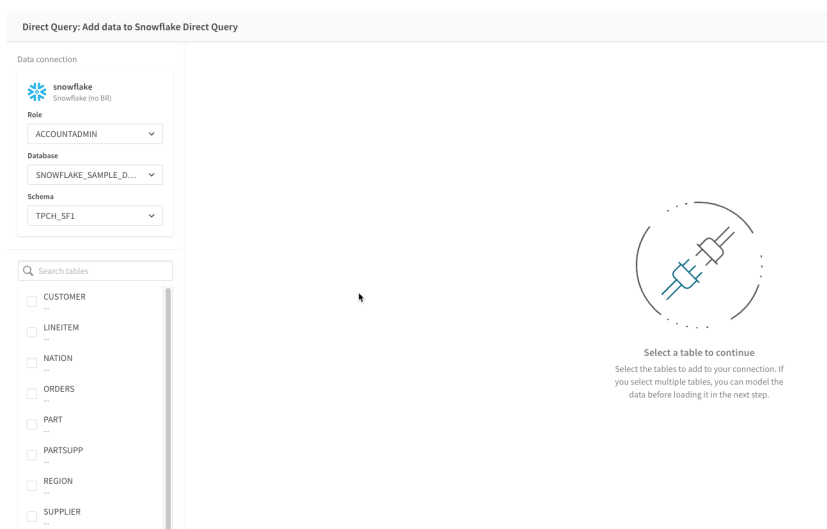
Step 3. Choose existing Snowflake Connection (or create per instructions above).



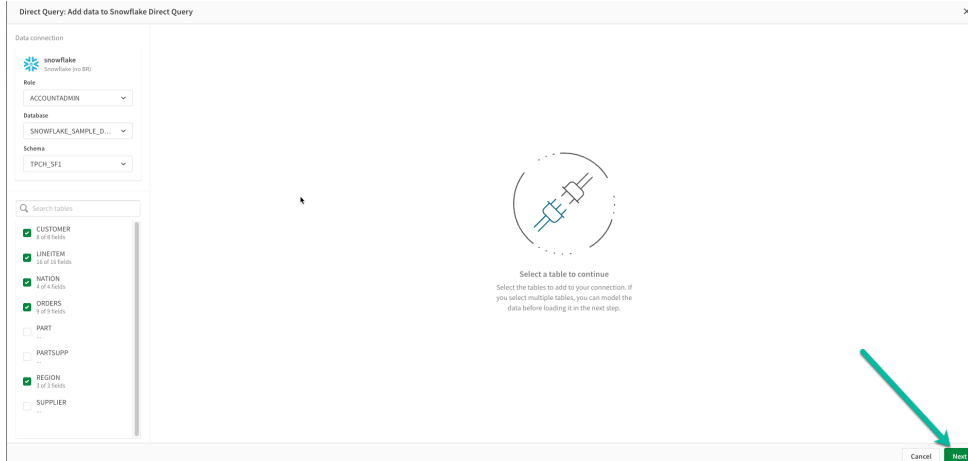
Step 4. When a Snowflake source is selected, a new option will appear in the top right corner, hit the dropdown and select “Go to Direct Query”.



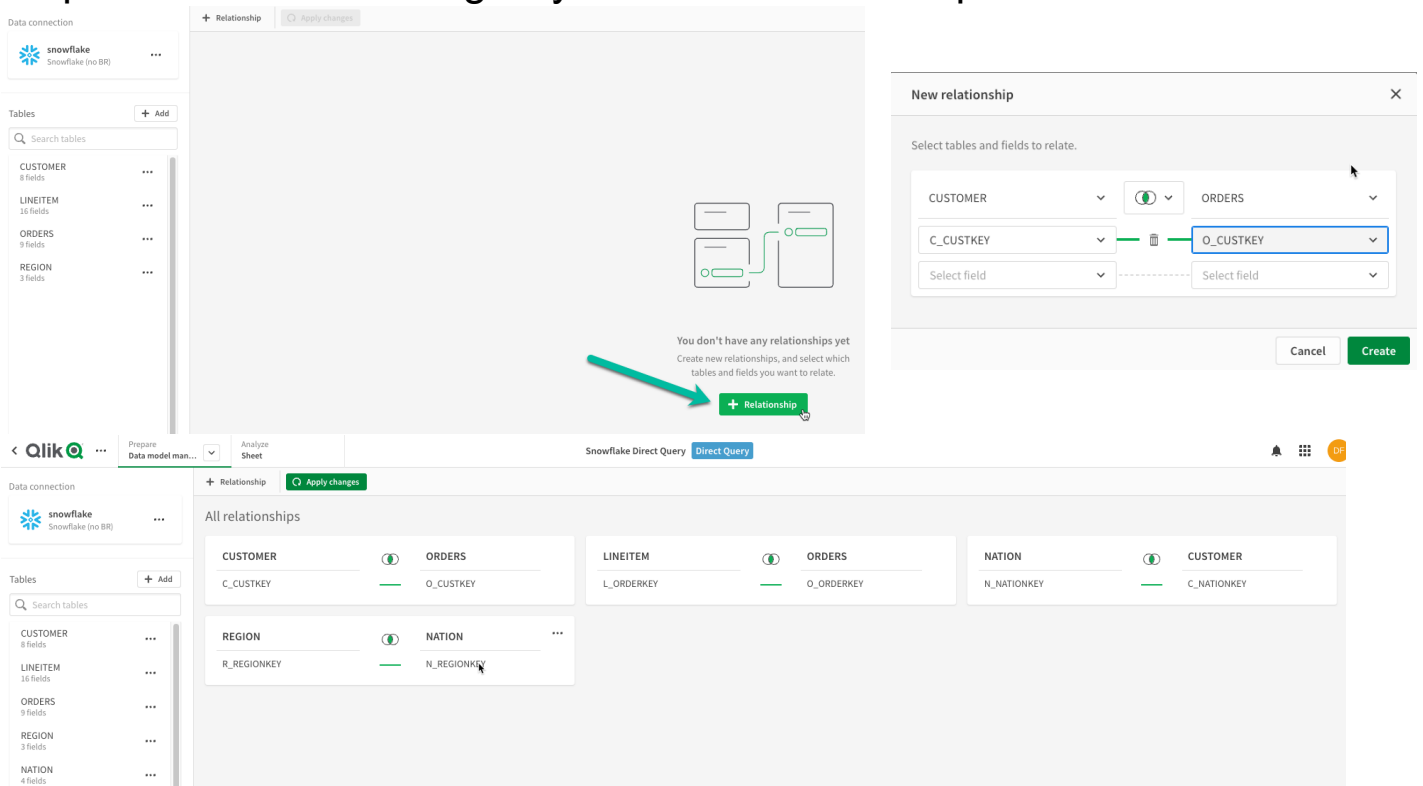
Step 5. A new interface will appear to create the data model for Direct Query (we will use Snowflake TPCH SF1 test data)



## Step 6. Choose the tables for the Query, select Next



## Step 7. Model Joins using Keys with “+ Relationship” button



## Step 8. Apply Changes

The screenshot shows the Qlik Sense interface during the 'Apply changes' step. The top navigation bar includes 'Qlik', 'Prepare Data model man...', and 'Analyze Sheet'. The main area is divided into 'Data connection' (showing 'snowflake Snowflake (no BR)'), 'Tables' (listing CUSTOMER, LINEITEM, ORDERS, REGION, and NATION), and 'Relationships'. The 'Relationships' section is active, showing a table of relationships with columns for table names and status indicators. A green 'Apply changes' button is highlighted with a mouse cursor.

All relationships	
CUSTOMER	ORDERS
C_CUSTKEY	O_CUSTKEY
REGION	NATION
R_REGIONKEY	N_REGIONKEY

## Step 9. Analyze Sheet and build Dashboard!

The screenshot shows the Qlik Sense interface during the 'Analyze Sheet' step. The top navigation bar includes 'Qlik', 'Prepare Data model man...', and 'Analyze Sheet'. The main area is divided into 'Assets' and 'Properties'. The 'Assets' section is active, showing a search bar and a list of fields. The 'Properties' section is active, showing a world map visualization. The map is titled 'My new sheet' and displays a color-coded world map. A legend on the right indicates the visualization is an 'Area layer' for 'N\_NAME' with the expression 'Sum(O\_TOTAL-PRICE)'. The legend shows a color scale from 1.67G to 1.91G. A scale bar at the bottom left indicates 5000 km.

My new sheet

Click to add title

N\_NAME  
Area layer  
Sum(O\_TOTAL-PRICE)  
1.91G  
1.67G

5000 km

© OpenStreetMap contributors

## Conclusions

---

This document showcased many integration options and best practices for using the Qlik Sense Analytics Platform with the Snowflake Data Cloud. The document discussed high level concepts, practical applications, and most importantly strategies on how to combine Qlik and Snowflake best to optimize analytics at your organization. This document will be updated as new capabilities are added to both the Snowflake engine and Qlik Sense platform.



### About Qlik

Qlik's vision is a data-literate world, where everyone can use data and analytics to improve decision-making and solve their most challenging problems. Our cloud-based Qlik Active Intelligence Platform<sup>®</sup> delivers end-to-end, real-time data integration and analytics cloud solutions to close the gaps between data, insights and action. By transforming data into Active Intelligence, businesses can drive better decisions, improve revenue and profitability, and optimize customer relationships. Qlik does business in more than 100 countries and serves over 38,000 active customers around the world.

[qlik.com](https://www.qlik.com)

© 2022 QlikTech International AB. All rights reserved. All company and/or product names may be trade names, trademarks and/or registered trademarks of the respective owners with which they are associated.