snowflake + looker

Configuration Guide

WHITEPAPER

Companies today are struggling under the combined weight of legacy business intelligence and data warehousing tools. These old and inefficient systems were designed for a different era, when data was a side project and access to business intelligence was limited to the executive team.

Modern companies are placing data in the center of every activity, and arming their team with the business intelligence and analytics tools they need to understand their business. The problem is that legacy data warehouses and business intelligence tools are fundamentally incapable of scaling to support the volume of data, use cases and overall complexity of modern, data driven organizations.

Snowflake and Looker represent a fundamentally different approach. Snowflake's multi-cluster shared data architecture was designed for the cloud to handle logarithmically larger data volumes at blazing speed. Looker is a business intelligence product that leverages fast direct connectivity to Snowflake and makes it easy to curate, govern, analyze, visualize and share data throughout an organization.

Both products are independently revolutionary, but in combination they can allow you to overcome many of the analytical challenges faced today. This paper will describe the methods and techniques you can use to fully utilize the power of Looker and Snowflake together, along with best practices for optimizing your processes in both products.

THE PROBLEM WITH LEGACY ANALYTICS PLATFORMS

Performance and Scalability

Analysts are the first victims of performance limitations. Analytics workloads are often pushed to off-peak times to reduce the effects of limited scalability on concurrency. Perhaps most frustratingly, there are often specific and complex rules for querying the database that can limit the ability of business users to find the data that they need. In many cases, because of the complexity of working with the database and the development intensive nature of legacy BI products, business users don't have timely access to the data and information they need.

As the number of data driven applications and use cases have skyrocketed along with data volume as a whole, scalability has become the overarching concern for database and analytics experts alike. Traditional database architectures have been unable to address those concerns completely. Shared disk data warehouses (see figure 1) are hampered with concurrent queries bottlenecking at the disk. Shared nothing data warehouses (see figure 2) struggle to partition data efficiently for multiple needs, as well as to handle joins and queries that involve multiple partitions. Even when implemented in the cloud, the limitations of both of these architectures apply equally. The larger the data volumes, the more acute each of those limitations become.

Fig. 1: Shared disk architecture is limited by the performance of the disk

Fig. 2: Shared nothing architecture is limited by the need to distribute and query data across nodes

Inflexibility

Limited scalability and performance expose another common problem: inflexibility. When facing performance and scalability problems, the knee-jerk reaction is to simply buy more database. Of course, due to the logarithmic nature of performance degradation, that rarely buys much time. It also exposes another problem: the inability to right size. People naturally purchase their data warehouse to match their needs at the point of highest demand, but rarely is that capacity used around the clock. When dealing with products that can cost millions of dollars, that unused capacity can be expensive.

Many data warehouses are also limited in the type of data they can store. The rise of the Internet of Things, and the prevalence of data formats like JSON in general, has led to a surge in the amount of semi-structured data organizations need to store and analyze. But, many traditional data warehouses are unable to house this data, and if they can, will rarely be able to query it in conjunction with other types of data.

Traditional analytics tools suffer from inflexibility of a different nature. As businesses change and adapt, their dashboards, reports and analytics evolve as well. Traditional analytics tools are often so rigid that changes to existing reports can take months, involve multiple technical resources, and hamper the ability of anyone to actually find the information they need.

Complexity

Poor scalability and flexibility inevitably lead to a third problem: complexity. Many database administrators spend the better part of their days endlessly tweaking and tuning the knobs on their database to ensure that everything is optimally performing. It's a challenging job, changing distributions, sort keys, compression, and worrying about encryption. A tweak to help a BI user in one area might lead to problems in another.

BI professionals have to deal with complexity brought about by their legacy tools. These legacy tools (and some of the new ones) have onerously complex calculation and visualization engines that force business users to ask for help with relatively straightforward analytics questions. This is time consuming for the whole team, hampers IT with distracting tasks, and prevents the line of business from being able to find the answers that they need. What's more, many BI tools force their users to make in-memory extracts of the database, which can improve performance on slower databases, but adds a layer of management and complexity that is unnecessary on a performant cloud data warehouse like Snowflake.

Hard to embed

Not all business intelligence use cases are in-application. There's a new breed of data driven company that's looking to provide embedded analytics throughout the organization, or even offer their data to their customers as a separate service or product.

Many of these organizations are relying on some kind of home grown reporting through spreadsheets or data dumps. These get the data to the end users, but they often fail to deliver insight from within that data, and go unnoticed or ignored.

Internal stakeholders and customers want something more dynamic: an interactive, ad-hoc reporting environment embedded in their line of work where every question they have is answerable in seconds. Although there are business intelligence tools that can deliver this kind of reporting, it often takes significant effort, manual maintenance, and relies on static data dumps that stifle dynamic questions. What's more, there are some embedded analytics products that charge by data volume, significantly disincentivizing companies who are trying to share a large amount of data with customers.

BUILDING A MODERN ANALYTICS PLATFORM WITH LOOKER AND SNOWFLAKE

Snowflake and Looker represent a modern approach to analytics that can help you get the most out of your data.

Precise scale for any need

Snowflake utilizes a new architecture built for the cloud: multi-cluster, shared data (see figure 3). From the end user perspective, it's a familiar SQL database, but the architecture is fundamentally different. All of the underlying data is stored in the cloud on Amazon S3. Compute is handled with independent clusters (or groups of clusters) called virtual warehouses. The operations of each virtual warehouse are completely independent of one another, and have no effect on the integrity or referencability of the underlying data. This means that you can store an infinite amount of data, and scale your compute to match an infinite workload.

Organizations that use Looker are in an excellent position to take advantage of the scalability of Snowflake. Because Looker was designed for direct connectivity to Snowflake (rather than in-memory extracts), it can take advantage of Snowflake's architecture for precise scalability. Specific workloads in Looker for separate teams or departments can utilize their own virtual warehouses, ensuring reliable performance independent of other activity on the database. Looker can also leverage multi-cluster and autoscaled warehouses in Snowflake for added performance.

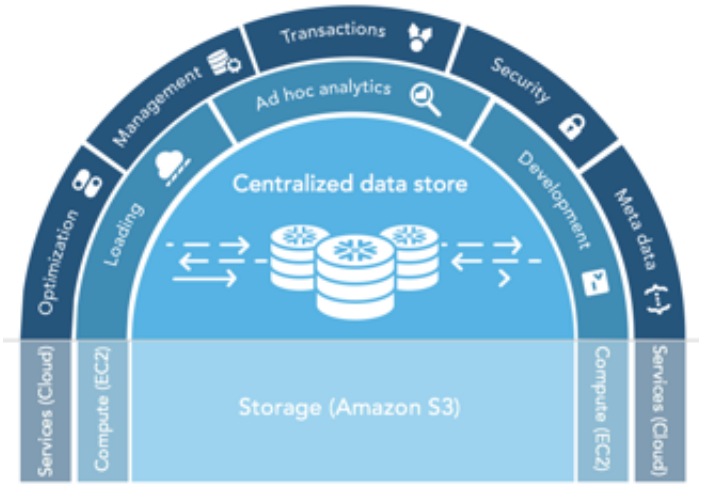

Fig. 3: Snowflake's multi-cluster, shared data architecture

Support for all types of data and analytics

Because each Snowflake virtual warehouse is independent, and can be scaled up and down on demand, organizations are able to adjust their capabilities (and cost) to demand. In other words, you are able to choose and change your data warehouse to meet your needs at any time. It's simple, but revolutionary in a world with fixed cost and massive up-front investment.

Snowflake is also able to handle structured and semi-structured data at the same time. There's no specific set-up requirements or preparation, and views can easily be created which will allow structured and semi-structured tables to be queried in Looker at the same time.

Looker matches Snowflake's flexibility with a range of options for analyzing and sharing data. To start, Looker fully supports Snowflake's native ANSI standard SQL, enabling your team to use the SQL skills they already have. Looker's modeling layer allows you to easily define relationships between structured and semi-structured data, which results in simple, self-service, and secure access for users to explore data and create their own visualizations. When it's time to share Looks or Dashboards with others, Looker Data Governance features allow you to securely and seamlessly distribute to anyone with a login and permissions.

Simplicity

Both Snowflake and Looker were designed to be straightforward to use and manage. Since Snowflake is a data warehouse as a service, you would expect the infrastructure to be fully managed, but the service extends far beyond that. Unlike many databases, Snowflake has few "knobs" to turn: it adapts to usage

patterns and dynamically responds. Optimization and tuning happen automatically, the moment that you load data into the system. Encryption is automatic.

Looker is similarly straightforward. The LookML modeling layer is a straightforward mechanism for admins to curate data and simplify the database for end users. Anyone can use these predefined models to create a visualization in moments simply by clicking and dropping. Existing visualizations can be used as a "jumping off point" for further analysis, and iterated on for increased insight. And embedded analytics is just a click away.

Seamless embedded analytics

Snowflake and Looker are the strongest solution for people struggling with unscalable, inflexible, static, and difficult to manage embedded analytics tools.

Snowflake allows you to cheaply integrate and store all of the data your organization wants to share. Then, without needing to copy or migrate data, you can create permissioned views and autoscaled warehouses to feed the dashboards you create in Looker. You can also permission through Looker instead. Instead of endless "data dumps" and extracts, Looker can directly query your embedded analytics from Snowflake, giving your end users a fresh view into the database every time.

Once you have content to share, you can use Looker's Embedded Analytics functionality, Powered by Looker, to embed Looks or dashboards using an iframe. All of the interactivity you already take advantage of in Looker will propagate to the page the Look is embedded on.

Powered by Looker doesn't charge based on the amount of data you use, so you can feel free to share as much data as your customers can handle.

OPTIMIZING SNOWFLAKE AND LOOKER

What you don't need to do

As you've already seen, Snowflake and Looker are an analytics system that requires very little optimization. We'll offer some general guidelines below on how to get the most from both tools, but it's important to note that there's a great deal that you won't need to do when using them together.

For instance, there's no need to create or manage indexes. You won't need to optimize your SQL, or tend to extracted data. There's no need to worry about data partitioning, or workload management either because those are handled automatically by Snowflake. Once these traditional points of optimization are eliminated, there are smaller and more targeted groups of best practices that should be straightforward to follow. We'll focus first on proper initial setup, and then dig in to the methods you can use to find and troubleshoot problems.

SET YOURSELF UP FOR SUCCESS

Both Snowflake and Looker provide multiple features that, if used properly, can help you to avoid performance problems altogether.

Isolate workloads in Snowflake

By leveraging different virtual warehouses when defining your Snowflake connections in Looker, you can ensure that separate query workloads do not impact each other. This can prevent data exploration from interfering with reporting. As a best practice, many organizations will have a virtual warehouse defined for each team or department.

Filter and exclude data before you start

Because most organizations give Looker access to all of their employees, it makes sense to filter and exclude data and fields you won't need in curated Views with

LookML, after you have connected to Snowflake. Following the same line of reasoning, make sure to start every new Look with the filters that make sense for the visualization you're creating. Common fields for filtering and mandatory filtering include team, region, user, and time. See Figure 4 for an example of a filter in Looker.

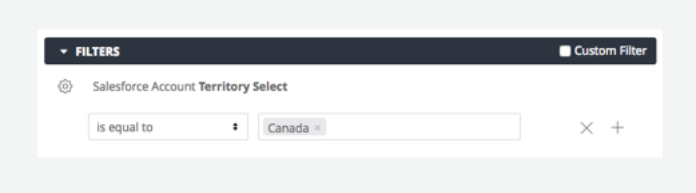

Fig. 4 : Excluding data

Bring your semi-structured data into Snowflake

Snowflake has native support for JSON, AVRO and XML data. Often, these types of datasets are loaded into separate systems that are difficult to integrate. However, since Snowflake supports these data formats and makes them accessible for analysis in Looker, it benefits you to bring them into Snowflake for analysis. This data can be ingested without predefining the schema, and a basic view can then make that data available in Looker. Additionally, tables containing semi-structured data can be joined to any other table including other tables that contain semi-structured data to provide flexible tables. You can use Looker's LookML data modeling layer to model both structured and semi-structured data, and the relationships between them, allowing users to access semi-structured data as they would any other data.

Caching

Both Snowflake and Looker provide result-set caching. This is handy if the underlying data hasn't changed and an identical query is issued. In this case, no warehouse is needed to fetch the data, making the query very fast. This has a positive implication for high-concurrency workloads—for example, many users all viewing the same dashboard throughout the day.

If the data underlying a particular dashboard doesn't change frequently (e.g., hourly, once a day, once a week, etc.) one can define an arbitrary caching period within Looker so that the queries that make up the dashboard necessarily pull from cache for the duration of time specified.

One more tip...

Templated filters allow developers to place parameters in complex SQL statements which end users can populate with values to increase pruning and filtering at query runtime. This is also a great approach when defining joins in an Explore definition.

IDENTIFYING PERFORMANCE PROBLEMS

If you've already isolated your workload and have filtered as much as possible, but are still experiencing sub-optimal performance, it might be time to dig deeper. These tools will help you to more accurately pinpoint where you are running into problems. After identifying problems with the tools in this section, read on to find suggested fixes within Snowflake, and in the construction of your Looks and Views.

1. Query timer - useful for keeping track of query execution times. Do note that both Snowflake and Looker have their own caching mechanisms, and query times may reflect the time spent pulling from cache and not the true time to execute the query.

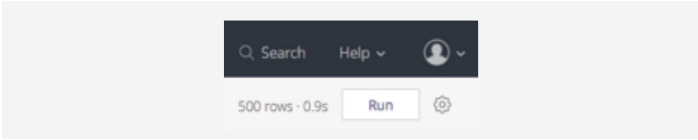

Fig. 5 : Query timer

2. SQL Tab - In any Explore, one can always see the SQL that Looker generates. From the SQL tab, one can then explore in SQL Runner to debug performance problems or view the explain plan in SQL Runner.

3. SQL Runner - a feature that is typically available to admins and developers, SQL Runner is useful for prototyping transformations in raw SQL as well as debugging performance issues by quickly changing functions used, join patterns, etc.

4. Query History - admins have the ability to view currently running, completed, and failed queries with some high-level statistics about the query. This is a useful feature to find out which users or explores are associated with long-running jobs. (See Figure 6)

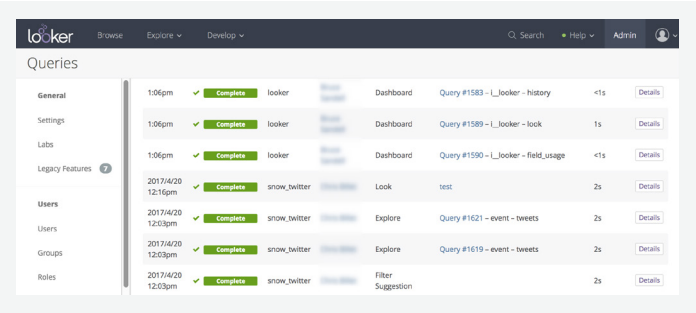

Fig. 6 : Query history

5. Persistent Derived Tables - similar to the query history panel, admins can gain insight into how their materializations or persistent derived tables (PDTs) are working. The PDT panel is useful to understand which PDTs are taking a long time to build, which PDTs are currently building (and thus creating a queue of queries that use the underlying PDT). Read on for more detail on how and when to use PDT's, as well. (See Figure 7)

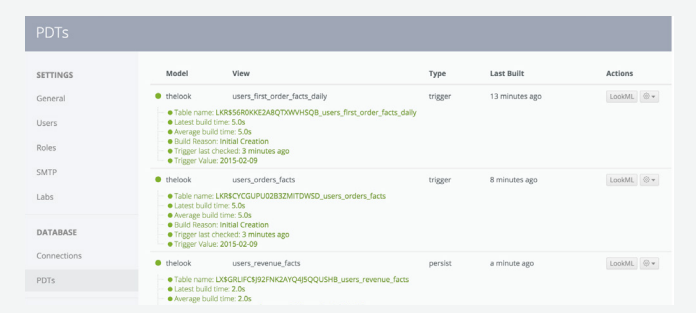

Fig. 7 : Persistent Derived Tables

6. Usage Dashboard - Without a doubt, the most useful feature for understanding performance issues and general usage patterns is the Usage Dashboard, which is available for admins. Looker exposes an internal model and dashboards based off of the underlying database that drives one's Looker instance. In addition to the pre-canned dashboards provided in this usage panel, all dashboard tiles are explorable. This allows the user to get incredibly detailed usage information about queries, views, users, scheduled tasks, etc. (See Figure 8)

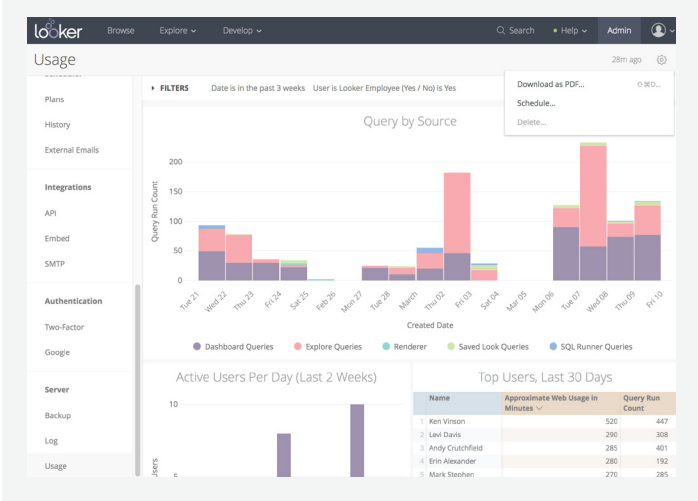

Fig. 8 : Usage Dashboard

7. Looker Log - lastly, for expert-level debugging of performance issues, Looker exposes the application log with many helpful filters, grep patterns, etc. This feature is more useful for debugging Looker-related performance issues and less for issues that stem from the underlying database. It's quite handy, nevertheless. (See Figure 9)

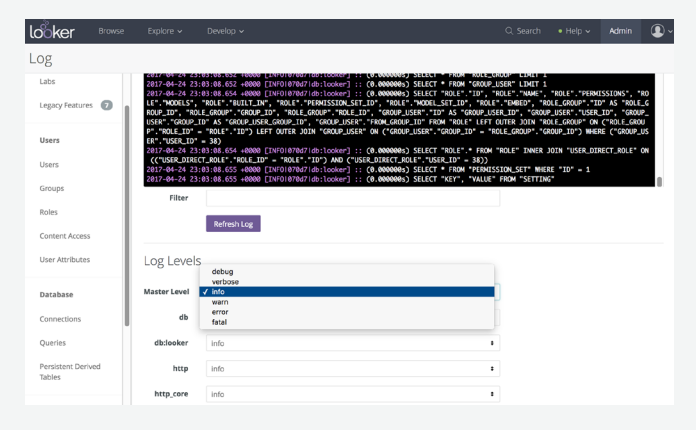

Fig. 9 : Looker Log

- Snowflake History – In a similar vein, it might make sense to view the same queries from within Snowflake. If you log in to your Snowflake instance, and click "History" from the top of the screen, you can see the queries that have been executed, and how long they took to execute. Generally, you should see a slightly higher amount of time in SQL Runner than Snowflake History. (See Figure 10)

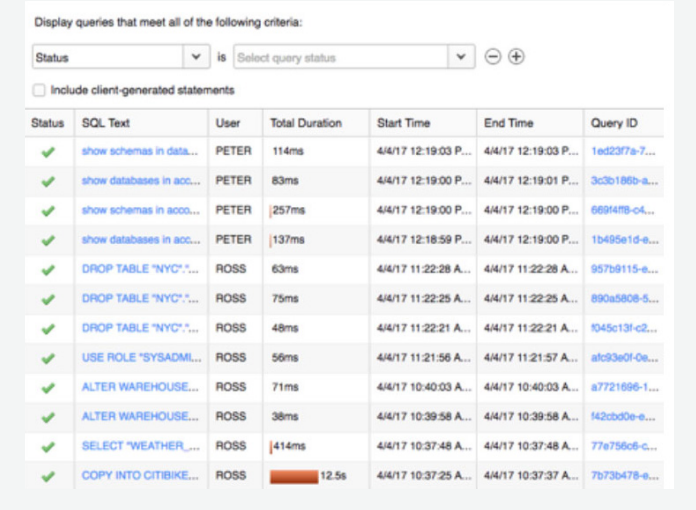

Fig. 10 : Snowflake History

- Snowflake query profiler - to truly understand why a particular query is problematic, the query profiler is the best tool available to the Snowflake user. It provides a highly detailed and visual view of the query execution plan, the database objects touched, the dependent steps, and useful statistics at every step. To access Query Profiler, click on any query ID in Snowflake history.

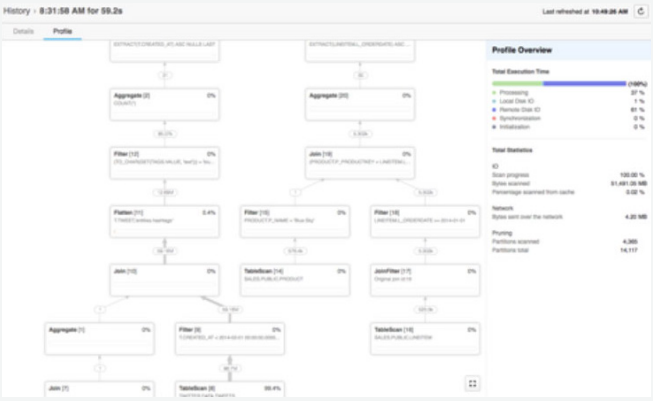

Fig. 11 : Query Profiler

ADDRESSING FREQUENT QUERIES AND CONCURRENCY

If your troubleshooting shows a relatively performant workbook, but you are still seeing degraded performance, it's possible you have a concurrency problem. In other words, there may be too many queries going to the database at the same time. There are several ways to help address this.

Use Automatic scaling in Snowflake

Snowflake's Multi-cluster Warehouse feature provides the ability to add compute resources automatically as additional Looker users increase the concurrent load on the database. This feature also automatically scales down compute resources once demand subsides. Many organizations should think about enabling automatic scaling on their reporting data warehouse.

Query Result Caching in Snowflake

Snowflake automatically caches all query results to provide extremely fast response times for queries that run multiple times throughout the day. This cache is intelligent enough to prevent users from ever seeing outdated data, but can significantly reduce the impact of queries that are frequently run. A best practice is to pre-populate the result cache after each data load for commonly run queries.

BUILDING FOR EFFICIENCY IN LOOKER

Use Persistent Derived Fields

Persistent Derived Tables effectively take whatever SQL transformation (SELECT statements only) that the user provides, and wraps it in a CREATE TABLE AS <provided select statement>. PDTs are quite useful for speeding up complex transformations; this feature set provides robust triggering and scheduling options as well.

Looker allows one to define a persistence duration using `persist_for` which means, when someone queries this thing, the CTAS is executed, and all subsequent queries hit the materialized version until its expiration.

There's also a triggering mechanism, `sql_trigger_value` that allows one to provide some arbitrary SQL which is evaluated every five minutes. When the value changes or is true, Looker drops the current version of the CTAS and rebuild it.

`Persist_for` and `sql_trigger_value` cannot be used in conjunction with one another. The latter is more commonly used.

When to use PDTs

Any transformation that involves window functions, subqueries, or common table expressions (CTEs) is a good candidate for derived tables. Looker cannot express the above SQL constructs with LookML alone. Because of this, one must expose them in a derived table, and perform any potential aggregations and group bys with dimensions and measures in LookML.

Without defining one of two possible methods for table persistence, Looker will treat the derived table as a CTE or ephemeral table when it is used in an Explore. It's advisable to default to this non-persistence path until performance becomes a concern. Snowflake is quite good at optimizing complex SQL statements that involve many CTEs and/or subqueries. However, if the transformation is simply too complex to handle at runtime and/or it touches a lot of data, adding a persistence argument (add an `order by` for clustering) is advisable. Setting a trigger to rebuild when new data are available (using `select count(*) from underlying_table`) is advisable when the data land in Snowflake in less frequent batches. For tables where the data only need to be as fresh as today or every hour, triggers like `select current_date` are most appropriate. For more information on Persistent Derived Tables in Looker, please consult the Looker Documentation.

Use templated filters

Standard (non-persistent) derived tables can benefit greatly from templated filters. This allows the developer to parameterize the SQL transformation which then allows end users (i.e., non-developers) to add filters in the UI which are passed through into the SQL. In Snowflake terms, these filters facilitate pruning, so that the smallest subset of data is scanned when the SQL is issued to Snowflake. The benefit is that very complex transformations that might be otherwise slow if scanning all of the data can be dramatically sped up. This means one can avoid persisting the transformation.

Take advantage of table calculations

Many transformations that might be done in the database using subqueries and window functions can actually be moved to Looker in the form of table calculations (see Figure 13). For result sets that are small enough to render in the browser, Excel-like post processing can be done using table calculations. Common use cases are moving averages, cumulative totals, etc, over daily, weekly, and monthly aggregate amounts. This can simplify and reduce the SQL and LookML that developers have to write, while opening up greater flexibility to end users. This also helps one avoid model bloat, where one-off transformations are exposed as derived tables but, by their very nature, aren't really utilized again.

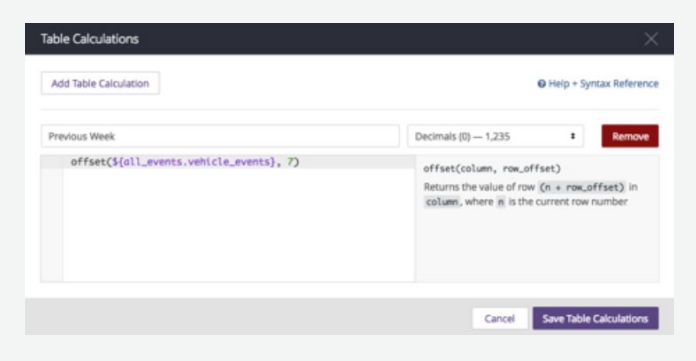

Fig. 12: Table calculations

One more tip...

If necessary, it may make sense to use clustering in Snowflake, particularly on the columns you are filtering with in Looker.

HOW TO GET STARTED

If deployed together, Looker and Snowflake can help any organization to deliver a scalable, flexible and simple analytics platform. Free trials of both products are available on-demand at any time, from the links below.

Try Snowflake On-Demand
Try Looker

Snowflake Computing, the cloud data warehousing company, has reinvented the data warehouse for the cloud and today's data. The Snowflake Elastic Data Warehouse is built from the cloud up with a patent-pending new architecture that delivers the power of data warehousing, the flexibility of big data platforms and the elasticity of the cloud – at a fraction of the cost of traditional solutions. Snowflake is headquartered in Silicon Valley and can be found online at snowflake.net.

Looker is a complete data platform that offers data analytics, exploration and insights to every function of a business and easily integrates into every departmental application to get data directly into the decision-making process. The company is powering data-driven cultures at more than 800 industry-leading and innovative companies such as Sony, Amazon, The Economist, Sears, Spotify, Kohler, Etsy, Lyft and Kickstarter. The company is headquartered in Santa Cruz, California, with offices in San Francisco, New York, London and Dublin, Ireland. Investors include Kleiner Perkins Caufield & Byers, Meritech Capital Partners, Redpoint Ventures, First Round Capital, Sapphire Ventures, and PivotNorth. For more information, Connect with us on LinkedIn, on Twitter, Facebook, G+ and YouTube or visit looker.com.