

August 2023



Tableau and Google Cloud BigQuery

A guide for experienced analytics professionals looking to fully leverage these powerful tools to get faster insights, covering basic product concepts, connectivity options, and optimization of the interaction of the two technologies.

Authors

Brad Brechbuhl, Lead Solution Engineer

Adiascar Cisneros, Senior Product Manager

Vikram Bandrupalli, Principal Solution Engineer



We would like to acknowledge the authors of the previous version of this whitepaper:

Pierce Young

Vaidy Krishnan

Riley Maris

Babu Prasad Elumala

Seth Hollyman

Tino Tereshko

Mike Graboski

Contents

03	Tableau and Google BigQuery
03	Technology Overview
05	Best practices for performance: Tableau
09	Best Practices for Performance: Google BigQuery
10	Connecting to BigQuery from Tableau
17	Best Practices for Performance: BigQuery with Tableau
20	Google BigQuery BI Engine
24	Case study: Top tips from Zulily's for self-service analytics with Tableau and Google BigQuery
25	Appendix A: Governance
26	Appendix B: Using Tableau to visualize Google BigQuery ML model results
26	Appendix C: BigQuery Resource Hierarchy
27	Appendix D: Additional resources





Tableau and Google BigQuery

Tableau and Google BigQuery allow people to analyze massive amounts of data and get answers fast using an easy-to-use, visual interface. Using these tools together, you can:

- Put the power of Google BigQuery into the hands of everyday users for fast, interactive analysis.
- Analyze billions of rows in seconds using visual analysis tools without writing a single line of code and with zero server-side management.
- Create interactive dashboards in minutes that connect to your Google BigQuery data and keep your organization up to speed.
- Share reports and insights on the web using Tableau Server or Tableau Cloud to allow anyone in your organization to connect from any device.
- Combine the cloud agility of Google BigQuery with the blazing speed of Tableau to uncover hidden insights and recognize return on investment faster.

Technology Overview

Google BigQuery

BigQuery can process petabytes of data in seconds in plain SQL with no fine-tuning or special skill set required. Powered by Dremel, Google's revolutionary technology for analyzing huge data sets, BigQuery provides a level of performance that large businesses previously had to pay millions to obtain—at a cost of pennies per gigabyte.

BigQuery is a data warehouse best suited for running SQL queries against structured, semi-structured, and unstructured data sets of virtually any size. Example use cases and data sets include:

- Ad hoc analytics
- Web logs
- Machine/server logs
- Internet of Things data sets
- E-commerce customer behavior
- Mobile app data
- Retail analytics
- Gaming telemetry
- Google Analytics Premium data
- Any data set for which a traditional RDBMS is taking minutes (or hours) to run a batch query

BigQuery is completely NoOps (no operation required), maintenance-free, and is part of the Google Cloud platform. Processing clusters are sized and provisioned by BigQuery at runtime, which means BigQuery does not require you to provision a cluster of servers in advance.

BigQuery pricing has two main components: compute and storage. Compute is the cost to process queries and other operations. Storage is the cost to store data loaded into Google BigQuery.

For more information about Google BigQuery pricing, see [BigQuery Pricing](#).



Tableau

Tableau helps people see and understand data. Our modern analytics platform, based on technology developed at Stanford University, puts the power of data into the hands of everyday people through visual exploration. This allows a broad population of users to engage with their data, ask questions, solve problems, share actionable insights, and create transformative value. Whether or not users are comfortable with traditional BI tools, people quickly learn to leverage Tableau to create and explore rich, interactive visualizations and powerful dashboards with an intuitive, drag-and-drop user interface powered by our proprietary language called VizQL.

The Tableau platform includes data preparation capabilities that are visual, direct, and intelligent. It also includes the ability to query published data sources with natural language phrases (Ask Data).

Tableau native optimizations

Data source connector – Tableau has a native, optimized connector to Google BigQuery that supports both live data connectivity and in-memory hyper extracts. Tableau's breadth of connectivity allows users to combine data from BigQuery with data from any of Tableau's many supported data sources. For visualizations published to the cloud using Tableau Server or Tableau Cloud, direct connectivity to Google BigQuery can be maintained.

Parallel queries – Tableau takes advantage of the capability of Google BigQuery and other data sources to execute multiple queries at the same time. Batches of independent and de-duplicated queries are grouped together and sent to BigQuery if the result is not already cached. Users should expect to see large performance gains due to parallel queries because of BigQuery's scale-out architecture.

Query fusion – Tableau takes multiple queries from workbooks and dashboards and fuse them together when possible, reducing the number of queries sent to BigQuery. First, Tableau identifies similar queries, excluding differences in the columns that are returned. Then, it combines queries where the differences are only the level of aggregation or a user calculation.

External query cache – If the underlying data source hasn't changed since the last time you ran the same query, Tableau will automatically read from the previously saved query cache, providing nearly instantaneous load times.

On-demand connections in Tableau Desktop – When you open a published workbook, Tableau Desktop visualizes information faster by only connecting to the data sources required to display the current sheet's data.

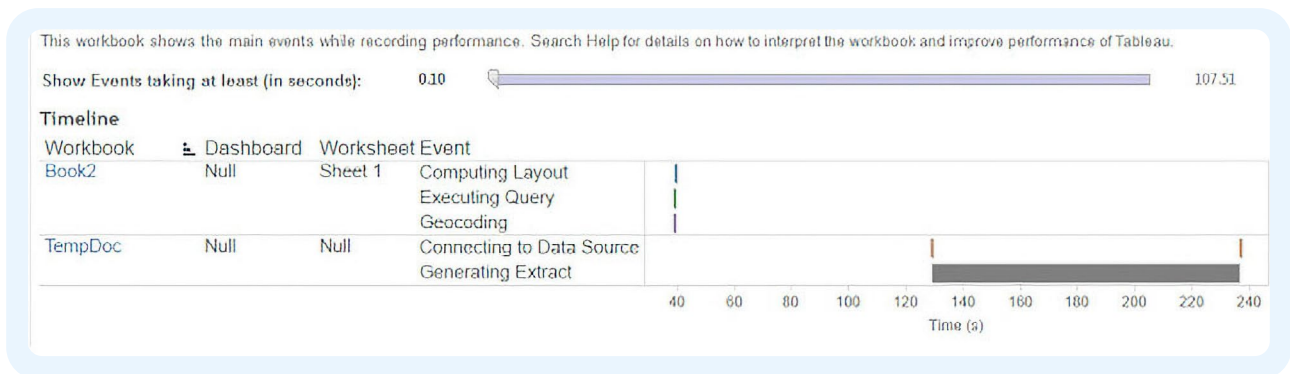


Best practices for performance: Tableau

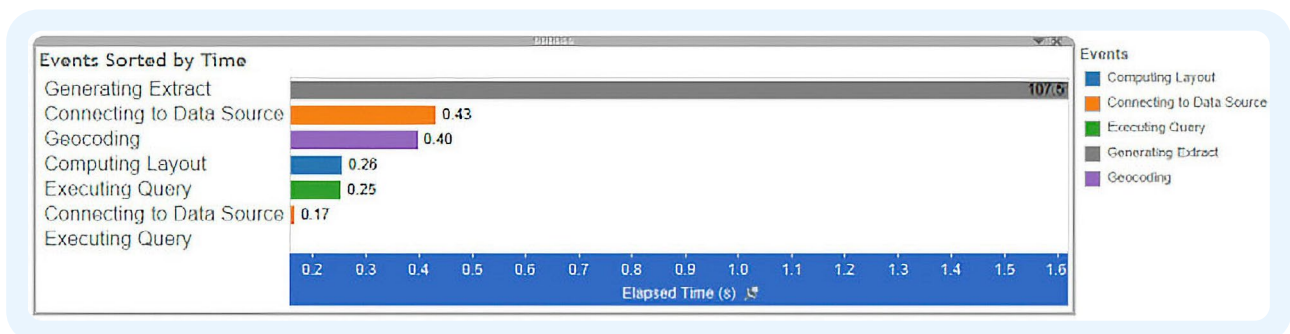
Before we get into additional tools and custom settings, our first recommendation is to always use the latest Tableau version. For Tableau Cloud users, this comes automatically. With the latest version you take advantage of the performance improvements we continually introduce to the product.

Performance Recorder

Performance Recorder is a built-in tool that records performance information about key events as you interact with a workbook, including query execution metrics. You can use that information to pinpoint slow queries when optimizing your workbooks. Performance Recorder tracks the elapsed time it takes an individual workbook to execute a query and compute the layout. In the Timeline view, the Workbook, Dashboard, and Worksheet columns identify the context for events.



Events with longer durations can help you identify where to look first if you want to speed up your workbook. In the view below, hovering over one of the green bars below will show the query generated against BigQuery.

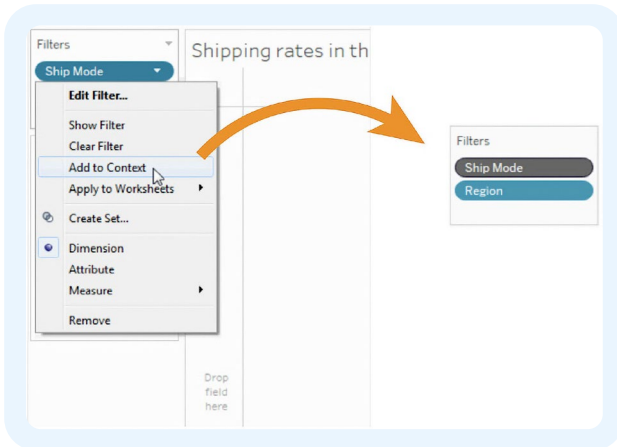


After identifying a slow query, if it's a live connection, you can often resolve the performance issue by revisiting your data model in Tableau and in your data source, or consider using an extract. If it's an extract, review your use of filters.

For more information on how to create or interpret a performance recording, see [Record and Analyze Workbook Performance](#) and [Interpret a Performance Recording](#). For Performance Recorder video examples, see [related community videos](#).

Context filters

A context filter is defined in a worksheet, and is applied before other filters. A context filter effectively reduces the size of the data source the other filters are applied to. This sequence avoids applying each filter to each record in the data source. If you are applying filters to a large data source, you can improve performance by setting up context filters.

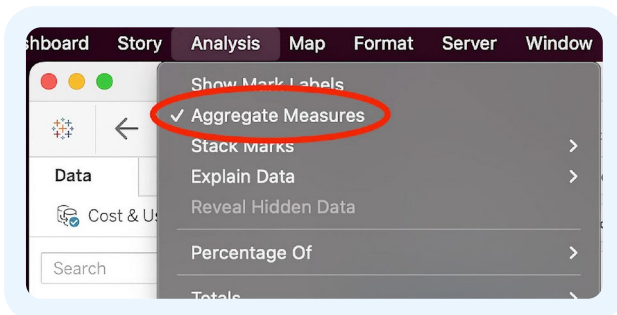


You can set one or more context filters to improve performance.

For more information (including examples) about using context filters, see [Use Context Filters](#).

Aggregate Measures

Aggregate Measures reduce the number of rows in the view. If your views are slow, you may be displaying too many rows of data. To speed up your views, consider working with aggregated measures.



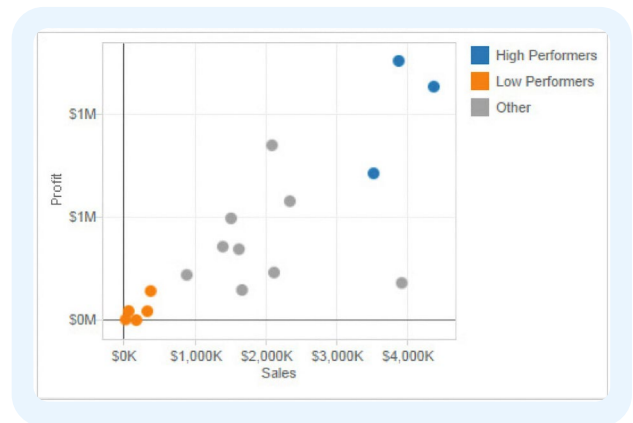
See if measures are aggregated in the Analysis menu. You can also set default aggregations for measures.

For more information (including examples) about using context filters, see [Use Context Filters](#).

Sets and groups

If you want to filter a dimension to remove members based on a range of measure values, you should create a set rather than using a quantitative filter. For instance, you can create a set that only returns the top 50 items in a dimension, rather than all of the items in a dimension.

When creating a group from a selection, make sure you've included only the columns of interest. Each additional column included in the set will affect performance negatively.



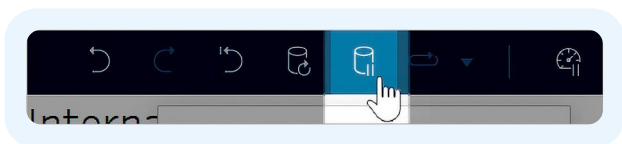
When you create groups in Tableau, you have the option to group all remaining members in an Other group.

For more information about creating sets and groups, see [Create Sets](#) and [Group Your Data](#).



Use the Worksheet Pause button

As you interact with a view, the server sometimes has to query the data source to update the view. Those queries can take time and affect responsiveness. Using the pause icon on the toolbar makes Tableau temporarily stop updates to the visualization.



For more information about controlling how views interact with data sources, see [Refresh Data or Pause Automatic Updates](#).

Turn off automatic updates

When you place a field on a shelf, Tableau generates the view by automatically querying the data source. If you are creating a data view with a lot of marks, the queries might be time consuming and significantly degrade system performance. In this case, you can instruct Tableau to turn off queries while you build the view. You can then turn queries back on when you are ready to see the result.

For more information about turning automatic updates off, see [Turn off Automatic Updates to Boost Performance](#).

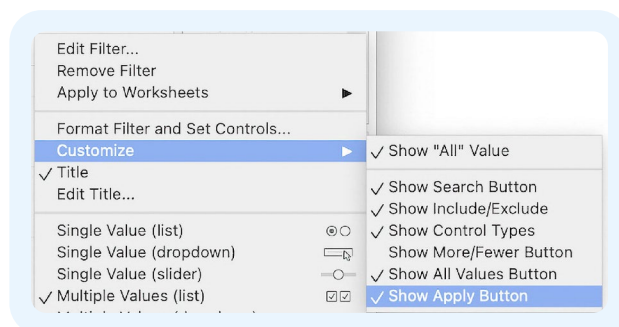
Add filters first

It's always a good idea to add filters first when you're building a view. Adding filters first reduces the volume of data loaded by the worksheet, making the view more responsive while you build it.

For more information (including examples) about the different types of filters and how to apply them, see [Filter Data from Your Views](#).

Leverage the "Apply" button with filters

If there is a filter with multiple options, adding the Apply button to the filter tells Tableau to send one query back to the datasource when the Apply button is clicked (as opposed to sending a query after each filter selection is clicked).



Consider worksheet warnings

Tableau displays a warning when it detects user actions that may affect performance. Consider the tradeoff between performance and what you need to analyze.

For example, Tableau displays a performance warning dialog box when you attempt to place a dimension with many members on a shelf. The dialog box provides four choices as shown in the figure below. If you choose to add all members, you might experience a degradation in performance.

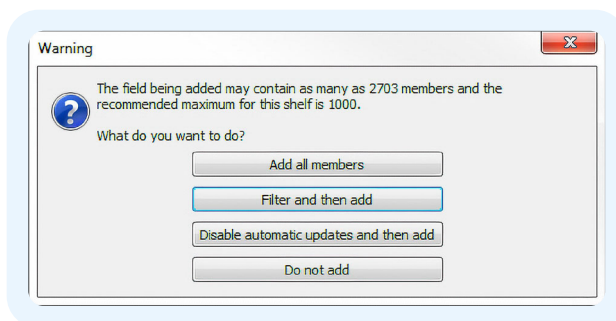


Tableau will warn when performance is at risk from placing a large dimension on a shelf.



You might also see a warning when you attempt to create too many panes in a table. In this case, Tableau warns you that the requested table “contains more than the recommended maximum number of panes.” It is best not to display more than the recommended number of panes, in part because you won’t get a useful view.

Optimize parallel queries

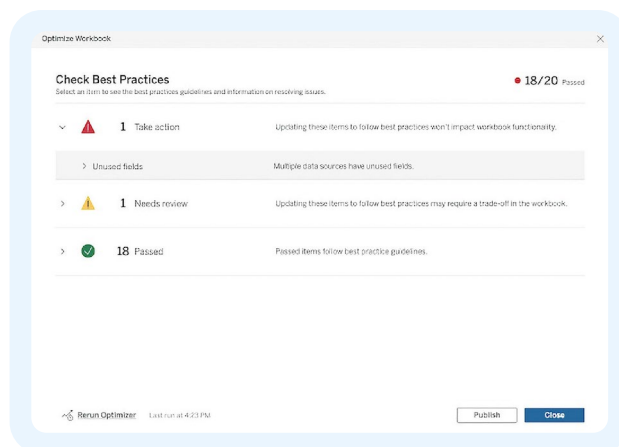
Configuring parallel queries using customization attributes improves the performance of large result sets returned from BigQuery to Tableau Cloud, Tableau Server, and Tableau Desktop. These customization attributes can be included in your published workbook or data source, as long as you specify the attributes before you publish the workbook or data source to Tableau Cloud or Tableau Server.

Before diving into parallel query optimization, consider enabling the use of the Storage API by the Google BigQuery (JDBC) connector, as explained in the [Connecting to BigQuery from Tableau](#) section later in this document.

For more information about optimizing parallel queries, see [Use customization attributes to improve query performance](#).

Workbook Optimiser

Workbook Optimiser is a tool that identifies whether a workbook follows Tableau-recommended performance best practices. It is available from the Server menu or the publishing dialog in Desktop and Web Authoring (Tableau Cloud and Tableau Server). The results page displays a prioritized list of concrete guidance, educating authors on how their workbook design impacts performance.



For more information about using Workbook Optimiser, see [Workbook Optimiser](#).

Live Connections vs Extracts

Tableau has two connection options when connecting to data sources: live connections and extracts. A live connection fetches data from the data source every time a query is generated by an operation (e.g.: loading a dashboard, selecting a filter option, or any other user interaction). An extract is a data snapshot optimized for analysis in Tableau that is queried via Tableau’s high-performant in-memory database called Hyper.

Live connections are generally chosen when the use case requires near real-time data. For all other use cases, extracts are preferred because they are more performant.





Best Practices for Performance: Google BigQuery

Cost vs. Performance

Lots of performance optimization comes from choices within Google BigQuery, and many times, these involve tradeoffs between cost and performance. Generally speaking, queries that do less work and return less rows and columns perform better and consume fewer resources which usually results in lower costs.

For more information about evaluating query performance, see [Introduction to Optimizing Query Performance](#).

Denormalize and Pre-Join

Denormalizing data is when you take individual tables and combine them or create copies based on similar or duplicate fields to create a single table. Denormalizing improves data querying performance significantly at the cost of a relatively small increase in storage consumption. Joining data is the process of connecting two or more tables based on one or more common fields.

It is often prudent to denormalize and pre-join data sets into homogeneous tables because BigQuery storage is very inexpensive and scalable. This means you will use fewer compute resources and more storage resources (the latter being more performant and cost-effective).

For more information about joins, nested, and repeated data, see [BigQuery explained: Working with joins, nested & repeated data](#).

Query Optimization and SQL Best Practices

Evaluating query performance in BigQuery involves several factors:

- Input data and data sources (I/O): How many bytes does your query read?
- Communication between nodes (shuffling): How many bytes does your query pass to the next stage? How many bytes does your query pass to each slot?
- Computation: How much CPU work does your query require?
- Outputs (materialization): How many bytes does your query write?
- Query anti-patterns: Are your queries following SQL best practices?

Many of these questions are answered in a [query plan](#) that BigQuery generates each time you run a query in BigQuery. The query plan shows execution statistics such as bytes read and slot time consumed. It also shows the different stages of execution, which help diagnose to improve query performance.

Slots and Reservations

BigQuery breaks down the computational capacity required to execute SQL queries into units called [slots](#). BigQuery then automatically calculates how many slots each query requires, depending on the query's size and complexity. BigQuery automatically manages the [slot quota](#) that your running queries share, based on customer history, usage, and spending. Most users find the default slot capacity for each project more than sufficient. Access to more slots doesn't guarantee faster performance for a query. However, a larger pool of slots might improve performance of large or complex queries, and performance of highly concurrent workloads.

Even an optimized query could run slowly due to slot contention. If your queries cannot be optimized further, consider using [reservations](#) in addition to optimizing your data model.

Google provides [guidance on query best practices](#), with examples and practice exercises. Many of the examples hold true when executing SQL queries in any modern database that supports SQL. For the rest of this document, we assume you are familiar with Google's best practices.

Connecting to BigQuery from Tableau

Install the JDBC driver

Before using the Google BigQuery (JDBC) connector from Tableau Desktop, Tableau Server or Tableau Prep make sure the [Google BigQuery JDBC driver](#) is installed. To learn more about downloading, installing, and managing Tableau drivers, please see [Database Drivers](#). To learn more about the Simba JDBC driver functionality and features, see the [driver documentation](#). If you are a Tableau Cloud user, the driver is managed for you.

Configure Tableau Server for OAuth

Before you can use Google OAuth for your Tableau Server data sources, you need to [configure Google OAuth in Tableau Server](#).

Establishing the connection

To connect to BigQuery from Tableau, select the "Google BigQuery (JDBC)" connector from the "Connect" left pane, under the "To a Server" section. The "Google BigQuery (JDBC)" connection window will appear. All values you enter in this window will be remembered for future connections. For example, once you enter your preferred Billing Project ID, next time you create a new connection the Billing Project ID will have the same value.

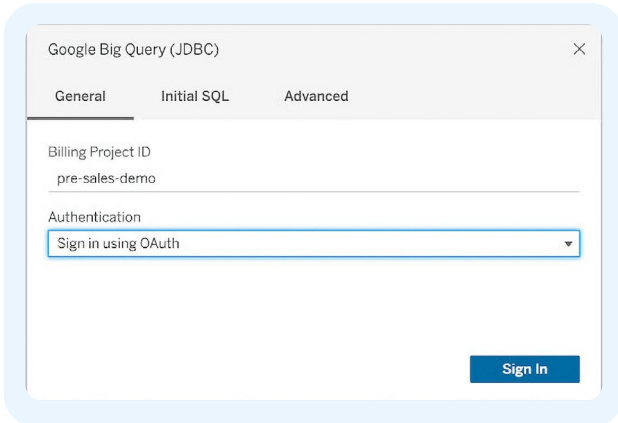
General

- Specify your billing project (refer to your Google Cloud console if you don't know your billing project)
- Select your method of authentication (OAuth or Service Account)

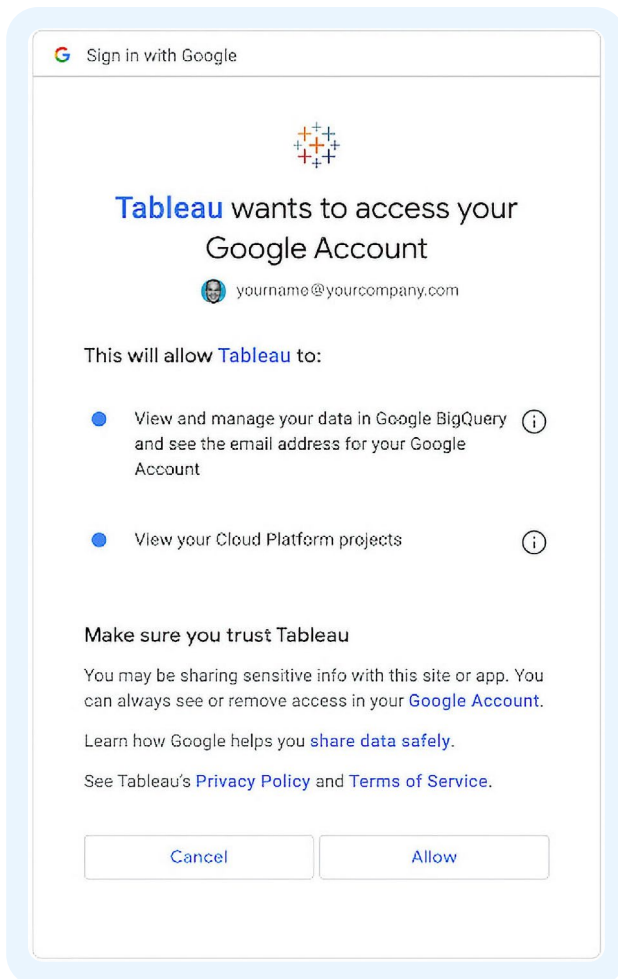


OAuth

Sign in using OAuth: Select “Sign in using OAuth” and click “Sign In.”



Select the Google Account associated with your billing project. Make sure all the boxes are checked so Tableau Cloud can communicate with your GCP instance.

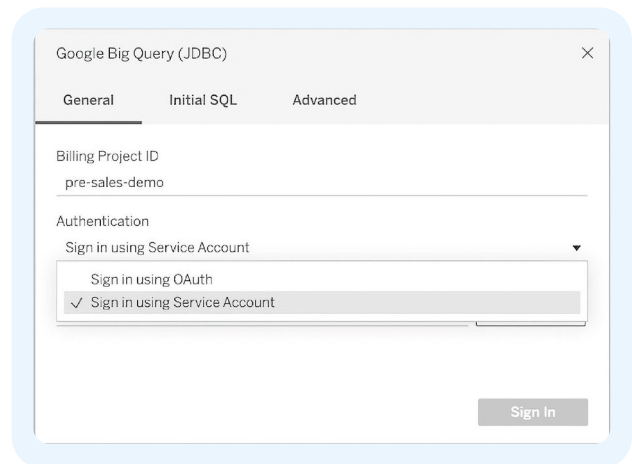


Service Account

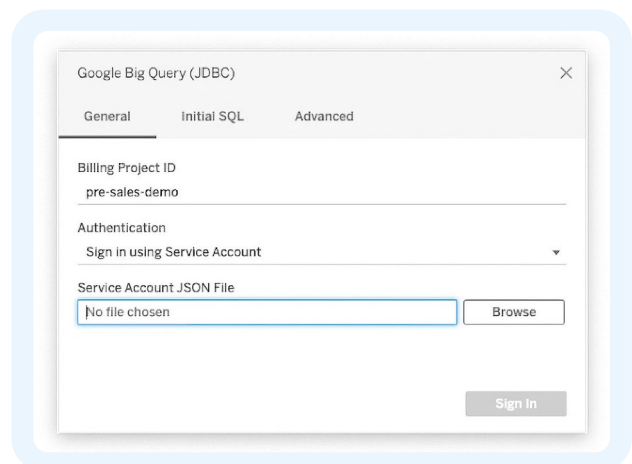
There are two ways to configure a data source connection with a service account, depending on where you're going to use it: in Tableau Desktop or in Web Authoring (Tableau Cloud and Tableau Server).

Service account connection in Tableau Desktop

From the Authentication drop-down menu, select Sign in using Service Account.



You'll need your service account authentication JSON file.



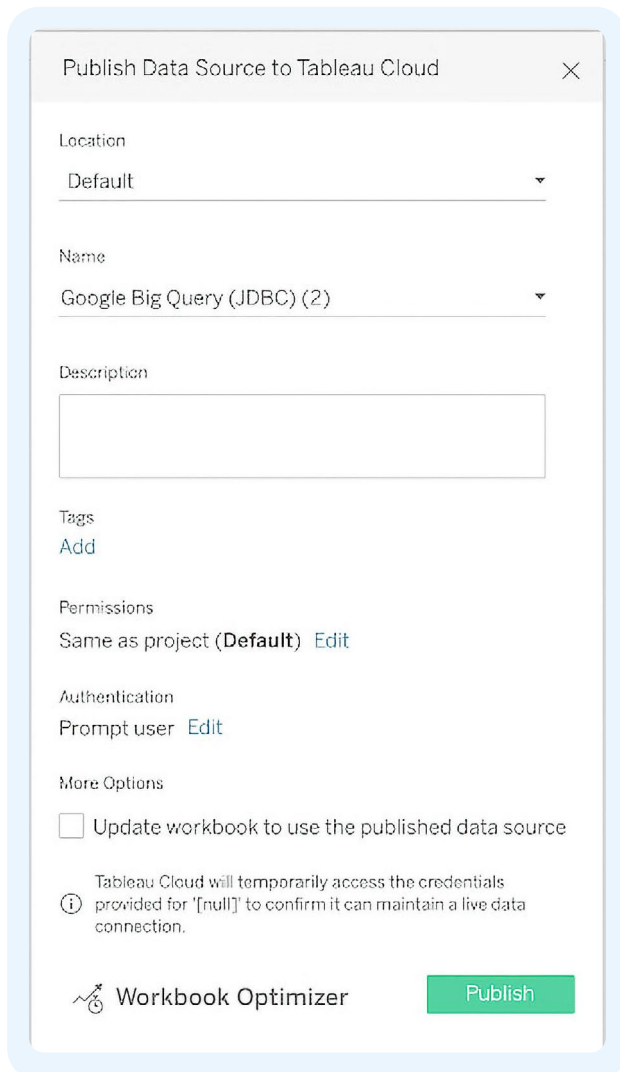
Service account connection in Web Authoring (Tableau Cloud and Tableau Server)

Service account connections cannot be created directly using Web Authoring. Instead, an existing published connection needs to be modified to sign in using a service account. That is, you first create an OAuth published connection in Web Authoring, or create a published connection of any type (OAuth or service account) in Tableau Desktop. Second, regardless of where you created it, edit the Authentication section on the newly published data source object in Web Authoring to change it to “Sign in using Service Account”.

1. Create the data source in Desktop(1a) or Web Authoring(1b)

1a. Create and publish the data source in Desktop

After creating a connection to BigQuery following the usual process, publish the data source to Tableau Cloud or Tableau Server. Accept all defaults and hit “Publish”.

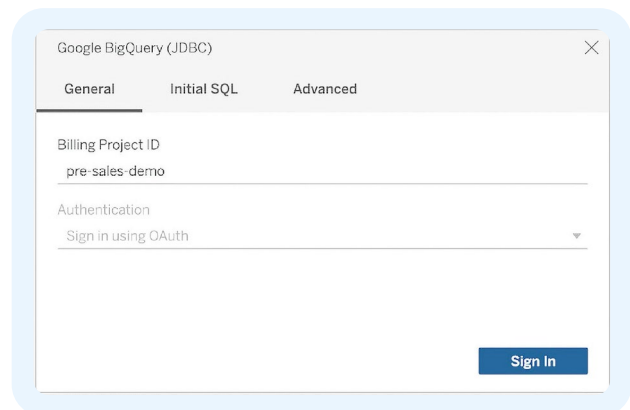


The screenshot shows the 'Publish Data Source to Tableau Cloud' dialog box. It has a title bar with a close button (X). The main content area includes:

- Location:** A dropdown menu set to 'Default'.
- Name:** A dropdown menu set to 'Google Big Query (JDBC) (2)'.
- Description:** An empty text input field.
- Tags:** A section with an 'Add' button.
- Permissions:** A section with 'Same as project (Default)' and an 'Edit' link.
- Authentication:** A section with 'Prompt user' and an 'Edit' link.
- More Options:** A section with a checkbox for 'Update workbook to use the published data source' (which is unchecked).
- Information:** A small icon followed by the text: 'Tableau Cloud will temporarily access the credentials provided for '[null]' to confirm it can maintain a live data connection.'
- Buttons:** A 'Workbook Optimizer' icon and a green 'Publish' button.

1b. Create the data source in Web Authoring to Sign In Using OAuth

When creating the data source, Web Authoring only allows signing in with OAuth. To end up with a service account connection, we need to create it first with Sign In Using OAuth, and then in the next step, edit the connection to Sign In Using Service Account.



The screenshot shows the 'Google BigQuery (JDBC)' configuration dialog box. It has a title bar with a close button (X). The main content area includes:

- Tabs:** 'General', 'Initial SQL', and 'Advanced'. The 'General' tab is selected.
- Billing Project ID:** A text input field containing 'pre-sales-demo'.
- Authentication:** A dropdown menu set to 'Sign in using OAuth'.
- Buttons:** A blue 'Sign In' button.



2. Edit the data source in Web Authoring to Sign In Using a Service Account

This sheet uses data that's on a Google Big Query (JDBC) database.
You need to sign into that server.

Sign In

After you click Sign In, Google Big Query (JDBC) will present a confirmation page. On that page, double-check that you are signed in to the correct Google Big Query (JDBC) account before you approve access.

When the data source is published, edit the connection details.

New ▾ Edit Data Source

Ask Data **Connections 1** Connected Workbooks 0

Select All

Data Source Svc Acct Test

-
-
-

Google BigQuery (JDBC)
Not embedded in connection
acisneros@salesforce.com

Name unavailable

⋮

Edit Connection...

Change the Authentication to “Embedded credentials...”

Edit Connection

Edit the selected data connection.

Authentication

Embedded credentials in the connection
Use this option if you want to schedule refreshes for data extracts

No authentication
Use this option if you do not need to schedule data extract refreshes

Network type

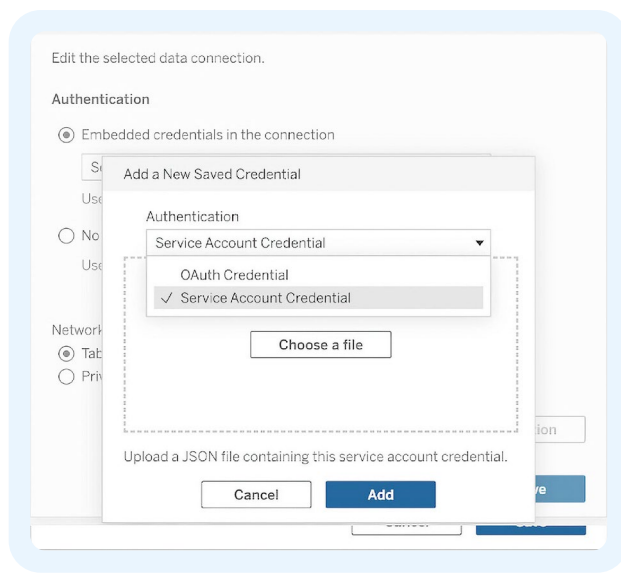
Tableau Cloud
 Private network

Test Connection

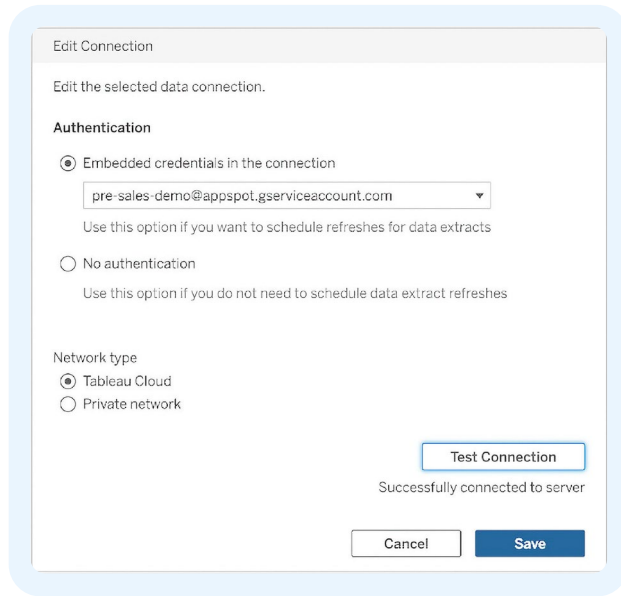
Cancel Save



Select "Service Account Credential" and provide the service account JSON authentication file location.

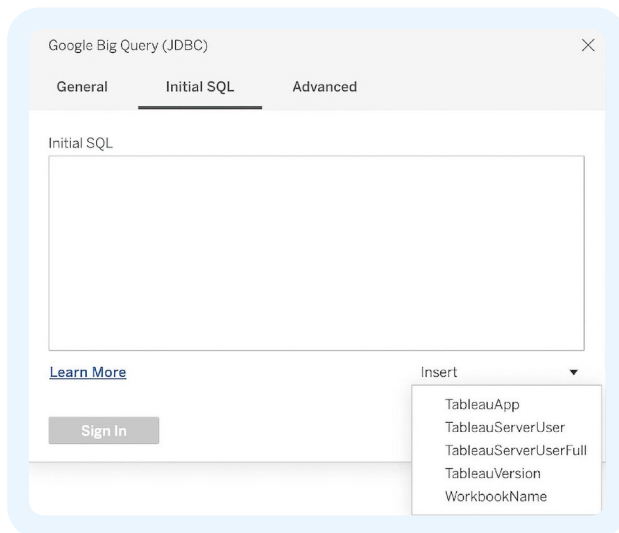


Test the connection and save.



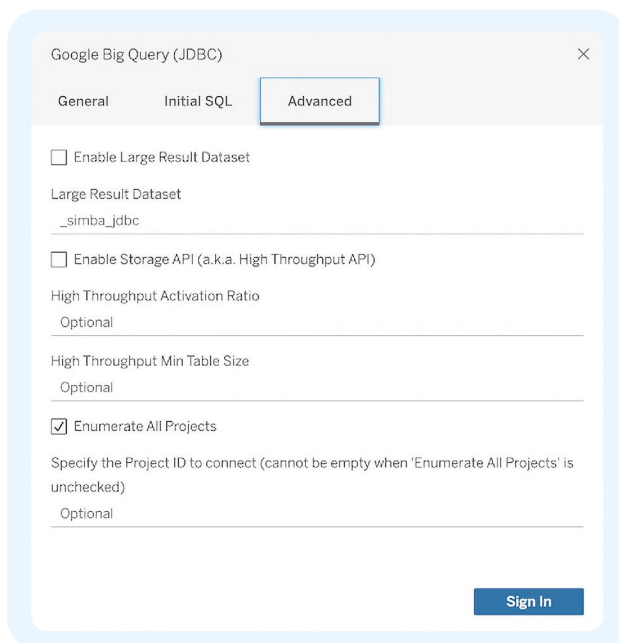
Initial SQL

Use this tab to enter [Initial SQL](#) code. The “Insert” drop down menu allows you to include Tableau parameters as part of your code.



For more information about Initial SQL, see [Run Initial SQL](#).

Advanced



In the Advanced tab, the connector allows the user to specify the following attributes:

Enable Large Result Dataset

Check this box to increase the maximum allowed request response (i.e. result) size. By default, BigQuery imposes a maximum response size on all requests (refer to [Quotas and limits](#) for specific limit values). If you do not enable large result set support, when executing queries, you might experience an error message such as “Response too large to return”.

When you check the “Enable Large Result Dataset” button, and specify a name for the “Large Result Dataset” table, all query results are written to and read from that table regardless of the query and its result size. Because of this, the result cache is not available for subsequent queries, and you are billed for every query that you make.

Enable Storage API (a.k.a. High Throughput API)

If you expect your query to return large amounts of data, check this option to enable the connector to handle the result sets more efficiently. This will allow the connector to leverage the BigQuery Storage API, which performs with higher throughput than the BigQuery API. Be aware the BigQuery Storage API must be enabled for the BigQuery project you are querying and pricing for the BigQuery Storage API is different from pricing for the BigQuery API.

If this feature is enabled, the connector checks the number of rows in an incoming result set table and the number of pages needed to retrieve all the results. If the number of rows and pages exceeds the threshold defined by the “High Throughput Activation Ratio” and “High Throughput Min Table Size” parameters (see below), the connector switches to using the BigQuery Storage API. If the connector encounters any issues initializing the BigQuery Storage API for retrieval, it falls back to using the BigQuery API, unless there is a permissions issue. To ensure best performance, do not use this feature with a named destination dataset or table.



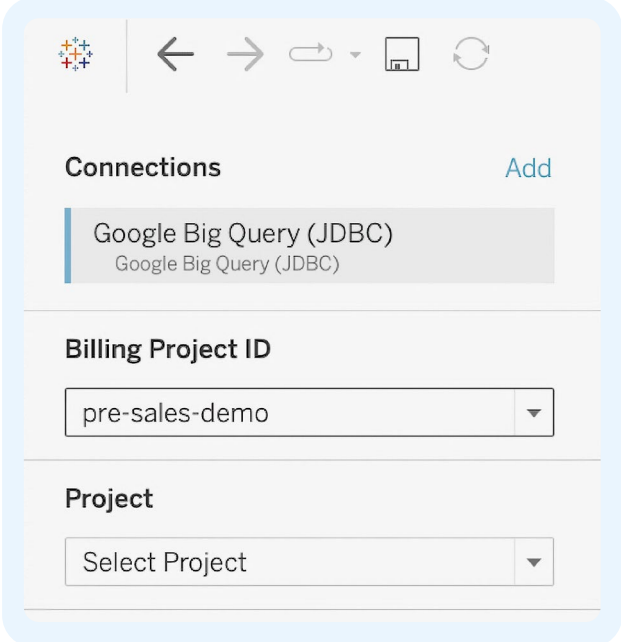
The connector will switch to the High Throughput API to more efficiently transfer the large result set based on the value of the following parameters:

- **High Throughput Activation Ratio** – When the number of pages in the result of your query exceeds this value, and the number of rows in the results exceeds the High Throughput Min Table value, the connector switches to using the BigQuery Storage API instead of the BigQuery API. The default value is 3. The default size of a page is 10,000 rows as determined by the MaxResults driver parameter (the connector does not provide the ability to modify that parameter).
- **High Throughput Min Table Size** – When the number of table rows in the result of your query exceeds this number, and the number of pages in the results exceeds the High Throughput Activation Ratio value, the connector switches to using the BigQuery Storage API instead of the BigQuery API. The default value is 100.

For more information about driver default values, see [Magnitude Simba Google BigQuery JDBC Data Connector Installation and Configuration Guide](#) and [Paging Through Table Data](#).

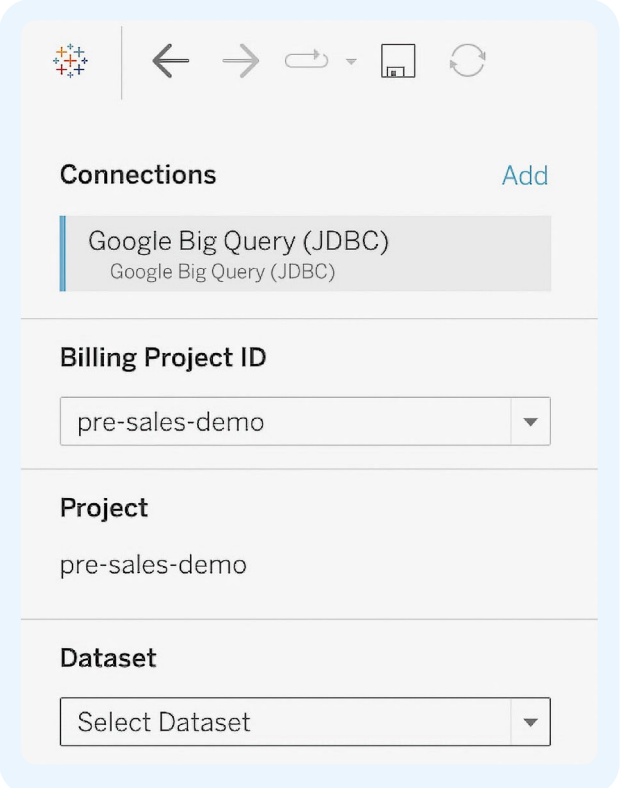
Per the default values, when the High Throughput API is enabled and the result of the query is higher than 30,000 rows (HighThroughput Activation Ratio = 3 pages, 10,000 rows in a page, 3 x 10,000 = 30,000). Note that per defaults, the determining parameter is the High Throughput Activation Ratio. In contrast, if you only set High Throughput Min Table to 40,000 rows, then the High Throughput API will be activated if the result of the query has more than 40,000 rows (which is 4 pages).

Enumerate All Projects – With this box checked, the connector will display a dropdown menu with all projects available. When you select the project, a new dropdown menu will appear to select the dataset.



The screenshot shows a configuration window for a Google BigQuery (JDBC) connection. At the top, there are navigation icons: a grid, left and right arrows, a refresh icon, a save icon, and a circular refresh icon. Below these is a 'Connections' section with an 'Add' button. A single connection is listed: 'Google Big Query (JDBC)' with a sub-label 'Google Big Query (JDBC)'. Underneath, there are three sections: 'Billing Project ID' with a dropdown menu showing 'pre-sales-demo', 'Project' with a dropdown menu showing 'Select Project', and 'Dataset' with a dropdown menu showing 'Select Dataset'.

With the Enumerate All Projects box unchecked, you must type a valid project ID (project name). After the connection is made, a dropdown menu allows you to select the dataset.



This screenshot shows the same configuration window as above, but with the 'Project' dropdown menu expanded to show 'pre-sales-demo' and the 'Dataset' dropdown menu expanded to show 'Select Dataset'.





Cross project joins

Google BigQuery (JDBC) allows you to join tables from different BigQuery projects as long as the datasets are in the same Google Cloud region. To join datasets, connect to the first table and use the Tableau data pane UI to navigate to another project and connect to additional datasets.

Best Practices for Performance: BigQuery with Tableau

Here are some best practices for analyzing BigQuery data with Tableau. Some of these recommendations apply to other data sources.

Have Tableau in close proximity to BigQuery.

This minimizes network latency. When running Tableau in Google Cloud, this means having Tableau in the same Google Cloud region as BigQuery. If running Tableau outside of Google Cloud (AWS, Azure, on-prem), this means having Tableau in close physical proximity to BigQuery.

Leverage BigQuery's compute and storage scale.

Whenever possible, perform resource intensive operations in BigQuery, like joins, calculations, transformations, etc., rather than performing those operations in Tableau. BigQuery scales compute resources with local access to data.

Return data at the highest aggregation your analysis requires.

This reduces the size of your dataset, improving speed. For example, you don't need to return data at the minute level if you need to analyze it at the month level.

Filter your data before Tableau analysis.

In Tableau, you can leverage extract and data source filters. This reduces the size of your dataset, improving speed. For example, you don't need to bring in 5 years worth of data if you're analyzing only 3 years.

For more information about BigQuery filtering, see [Best Practices For Filtering and Ordering](#).

Avoid custom SQL inside Tableau.

Custom SQL in live data connections can impact performance because Tableau issues the query to the database inside of a subquery.

For more information about how custom SQL affects performance, see [Connecting Through Custom SQL Causes Slow Performance](#).

Materialized Views

In BigQuery, materialized views are precomputed views that periodically cache the results of a query for increased performance and efficiency. BigQuery leverages precomputed results from materialized views and whenever possible reads only delta changes from the base tables to compute up-to-date results. Materialized views can be queried directly or can be used by the BigQuery optimizer to process queries to the base tables.

- A live connection to a single Google BigQuery table or materialized view is generally a better alternative to a live connection to a view because you can leverage Tableau and the table/materialized view's caching.
- Queries that use materialized views are generally faster and consume fewer resources than queries that retrieve the same data only from the base tables. Materialized views can significantly improve the performance of workloads that have the characteristic of common and repeated queries.
- Consider using [Tableau Extracts](#) wherever possible as they can improve performance. Extracts are optimized for Tableau and accessing data from extracts saves queries to the data sources.
- Live connection queries are sent from Tableau to the BigQuery API on-the-fly so any transformations which need to be done will also happen on-the-fly and will affect the time it takes to display results. Tableau's built-in query caching helps speed these operations.
- Avoid using federated tables where the data is in an external data source like Google Cloud Storage. In such cases, if you are looking to perform iterative querying on the data set, you should materialize the data in BigQuery (independent of Tableau) to enable high performance querying on the dataset within Tableau.

The following are key characteristics of BigQuery Materialized Views:

- **Zero maintenance.**
Materialized views are recomputed in the background when the base tables change. Any incremental data changes from the base tables are automatically added to the materialized views, with no user action required.
- **Fresh data.**
Materialized views return fresh data. If changes to base tables might invalidate the materialized view, then data is read directly from the base tables. If the changes to the base tables do not invalidate the materialized view, the rest of the data is read from the materialized view and only the changes are read from the base tables.
- **Smart tuning.**
If any part of a query against the source table can be resolved by querying the materialized view, then BigQuery reroutes the query to use the materialized view for better performance and efficiency.

For more information about BigQuery materialized views, see [Introduction to materialized views](#).



Initial SQL

The Google BigQuery (JDBC) connector supports Initial SQL. Tableau's Initial SQL allows Creators to include SQL commands to be executed when a connection is made to a data source. This feature is useful to customize the data source environment or force a user-specific session by executing commands during the initial connection.

The Initial SQL code executes only at specific times. For example, when you open a workbook, refresh an extract, sign in or publish to Tableau Server or Tableau Cloud. Note that Initial SQL is not run when you refresh your view.

It is important to understand that Initial SQL is different from Custom SQL. Custom SQL is code that is executed against a data source and returns a data set from SQL code you provide. Tableau always uses the provided Custom SQL query as the basis for all subsequent queries, as if it was a View in the database. Be aware that often this makes performance inferior to the queries produced automatically by VizQL against the basic Multiple Tables dialog (this will vary with the complexity of the query and the performance of the underlying database). If you use Initial SQL, sometimes you have to use Custom SQL, because objects created in Initial SQL can only be referenced from a Custom SQL object (for example, creating variables or temporary tables).

For more information about using Initial SQL, see [Run Initial SQL](#).

Customized Attributes

You can use customization attributes to improve the performance of large result sets returned from BigQuery to Tableau Cloud, Tableau Server, and on Tableau Desktop. You can have the customization attributes included in your published workbook or data source, as long as you specify the attributes before you publish the workbook or data source to Tableau Online or Tableau Server.

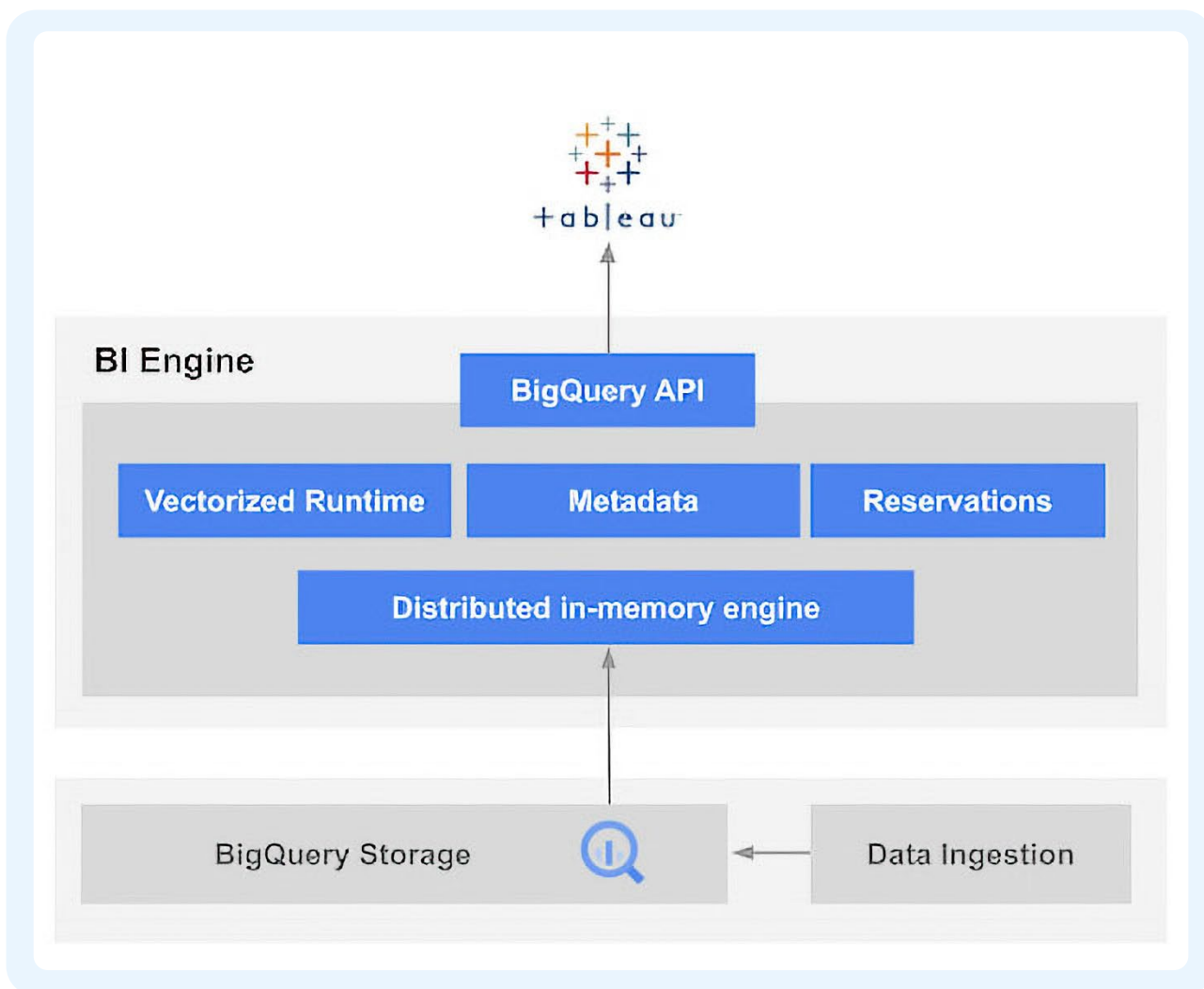
For more information about customizing attributes, see [Use customization attributes to improve query performance](#).





Google BigQuery BI Engine

What is BigQuery BI Engine?



From [GCP documentation](#), “BigQuery BI Engine is a fast, in-memory analysis service that accelerates many SQL queries in BigQuery by intelligently caching the data you use most frequently. BI Engine can accelerate SQL queries from any source, including those written by data visualization tools, and can manage cached tables for on-going optimization. This lets you improve query performance without manual tuning or data tiering. You can use [clustering](#) and [partitioning](#) to further optimize the performance of large tables with BI Engine”. In other words, BI Engine is a feature to automatically improve performance by smartly caching data from user-selected tables in pre-allocated memory. It’s like reserving memory for queries.

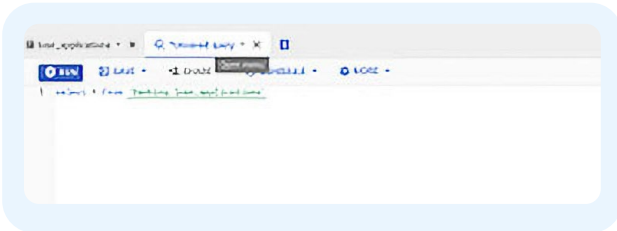
For more information about BI Engine best practices, see [Considerations for BI Engine](#).

Using BI Engine

Creating a BI Engine Reservation

Let's look at an example of adding a table to BI Engine to improve query performance.

The screenshots below show when executing a "select *" on the 750MB "loan_applications" table, you're billed based on bytes processed, which is 122 MB.



Job ID	pre-sales-demo:US.bquxjob_5036fe83_183e7326021
User	vbandarupalli@salesforce.com
Location	US
Creation time	Oct 17, 2022, 11:26:22 AM UTC-7
Start time	Oct 17, 2022, 11:26:22 AM UTC-7
End time	Oct 17, 2022, 11:26:27 AM UTC-7
Duration	4 sec
Bytes processed	121.76 MB
Bytes billed	122 MB
Job priority	INTERACTIVE
Use legacy SQL	false
Destination table	Temporary table

If you need to create a more performant dashboard on this "loan_applications" table, you now have an option to add a reservation to BI Engine. This will accelerate your query and reduce its cost.

Let's create a BI Engine reservation. To start, search for BI Engine from the BigQuery search bar.

Select the billing project where the BI engine reservation will be created, location, and memory capacity. The memory capacity determines the amount of data from queries that will be cached by BI Engine. For this example, 9 GB are selected.

1 Configure

BI Engine reservation will be assigned to your current project.

Project: pre-sales-demo

Location*: us-west2 (Los Angeles)

GB of Capacity: 9

1 / 250 Total: 9 GB

NEXT

2 Preferred Tables (Optional)

Please choose tables BI Engine should accelerate. This setting is optional - if no tables are selected, all queries within the project will use BI Engine.

Preferred Tables

Table Id (project_name.dataset_name.table_name): pre-sales-demo.Banking.loan_applications

NEXT

For on-demand pricing plans, as you configure the reservation, the console shows how much it will cost. The example below shows the estimated cost for our example.

Cost

\$368.97 monthly estimate

Total capacity	9 GB
Rate	\$0.0562 per GB/hour (\$41.00 per GB/month)
Reservation cost	\$368.97/month

[^ HIDE DETAILS](#)

Review all selections, and hit “Create” to finish.

1 **Configure**

2 **Preferred Tables (Optional)**

3 **Confirm and submit**

You agree to pay for BI Engine capacity based on hourly rate described on the right. You will be charged based on reservation size and number of hours it was created, regardless of usage. For example: Creating reservation for half a month will result in half monthly rate described on the right. Creating reservation for a day will result in hourly rate multiplied by number of hours. Creating reservation for 10 minutes will result in charge for 1 hour of usage. [Learn more](#)

Please review the [Google Cloud Platform's Terms of Service](#) before proceeding.

CREATE **CANCEL**

Once created, we can see the tables that are now called “preferred tables”.

BI Engine [+ CREATE RESERVATION](#)

Reservations

Filter Enter property name or value

Project	Size	Preferred Tables	Location	Monthly Cost	Actions
912061130034	3 GB	pre-sales-demo.Banking_loan_applications	Los Angeles (us-west2)	\$122.00	

From this point, BI Engine includes the mode it is operating on for each table queried. If the table fits completely within the reserved memory, the mode is reported as “FULL”. If only a portion of the table fits in memory, it is reported as “PARTIAL”.

Job ID	pre-sales-demo:us-west2.bqjob_3aa551c4_18400cd3f3e
User	vbandarupalli@salesforce.com
Location	us-west2
Creation time	Oct 22, 2022, 10:46:02 AM UTC-7
Start time	Oct 22, 2022, 10:46:02 AM UTC-7
End time	Oct 22, 2022, 10:46:28 AM UTC-7
Duration	26 sec
Bytes processed	790.6 MB
Bytes billed	0 B
Job priority	INTERACTIVE
Use legacy SQL	false
Destination table	Temporary table
BIEngine Mode	FULL

Connecting with Tableau

To connect to a preferred table from Tableau, you use the same process as connecting to any other BigQuery table.

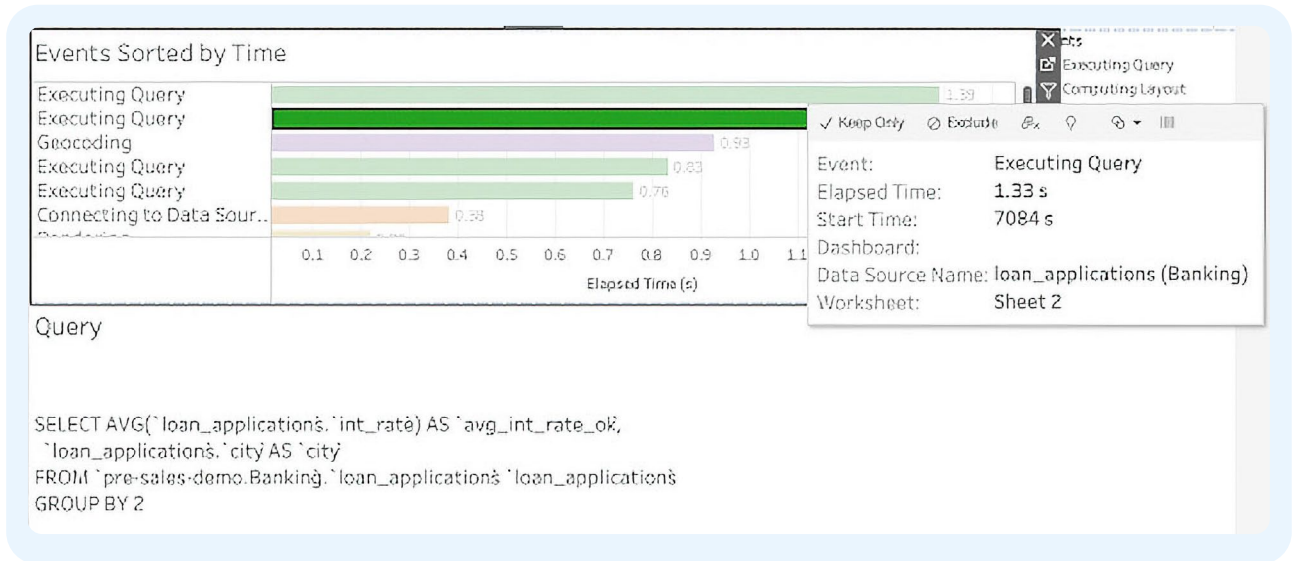
Tableau benefits from the BI Engine query acceleration. The example below shows the difference between queries without and with BI Engine:





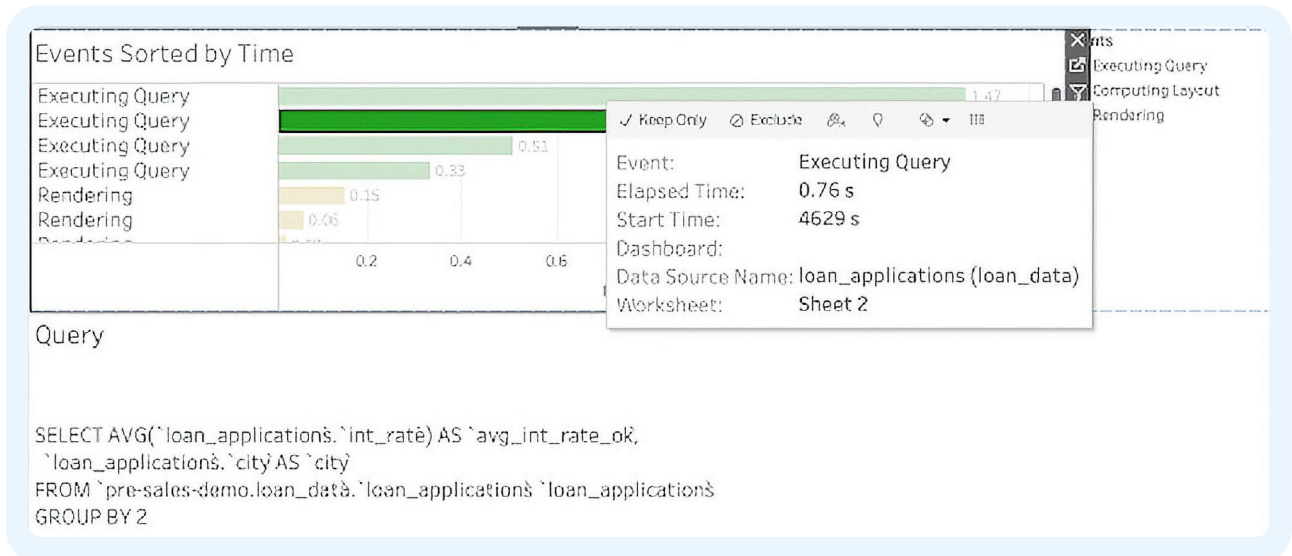
Without BI Engine:

Query time 1.33 seconds



With BI Engine:

Query time .76 seconds (almost ~50% improvement in query time)



For more information about BI Engine operational costs, see [BI Engine Pricing](#).

BI Engine Summary

- There is a default limit of memory allocated for data.
- You must specify what tables are accelerated with BI Engine.
- Tables that are queried frequently benefit the most from BI Engine.
- BI Engine and tables both need to be in the same cloud region.
- You must turn it on to use it.
- There's an associated cost to using BI Engine.
- You must manually cancel reservations.



Case study: Top tips from Zulily's for self-service analytics with Tableau and Google BigQuery

Zulily is a fast-growing e-commerce company that built a big data platform using Google BigQuery as the business data warehouse and Tableau for data access and visual analytics. Integrating BigQuery and Tableau allows the analytics to move fast on acquiring, ingesting, and using data to build reports and models without needing to involve IT in everyday activities. Additionally, business users have real-time access to key data used to make decisions quickly, without needing to involve analysts to generate basic insights.

Here are a few of Zulily's best practices:

Reduce latency by using Tableau Server on Google Compute Engine – Rather than a traditional model where regions are separate VPCs, you can leverage Google's private backbone without going to the Internet—and without additional setup. This also lets you right-size your deployments without having to over-provision.

Use Federated Sources and point Tableau at BigQuery – For data within Google Cloud, you should take advantage of BigQuery's ability to query external data sources and treat BigQuery as your data lake. In certain scenarios, you reduce the amount of data that needs to be pushed over the network and into Tableau for analysis.

Process large datasets with a live connection in BigQuery – Take advantage of BigQuery's ability to process large datasets and only bring results across the network. Set your default Tableau connection against BigQuery as "Live" unless you have a specific reason to extract the data.

For more information and the full list of all 10 tips, read the two-part blog series:

Part One: [Why Zulily created a self-service marketing analytics platform with Tableau and Google BigQuery](#)

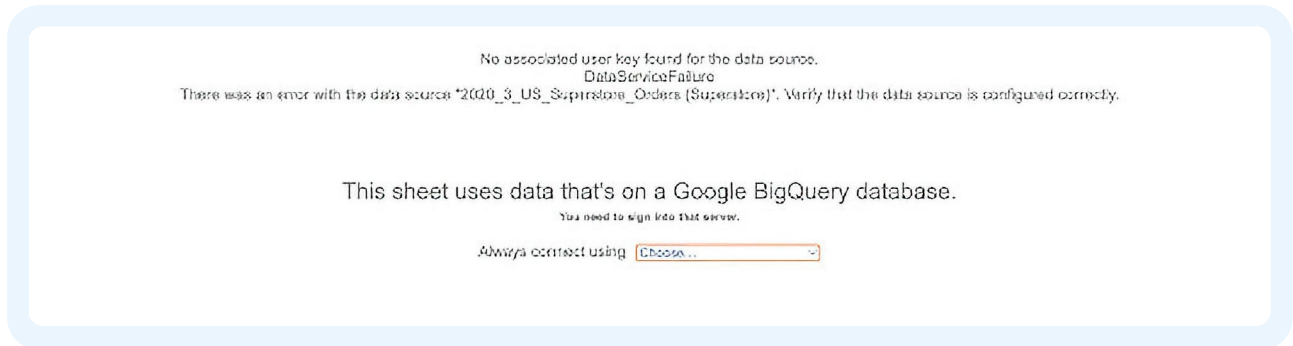
Part Two: [Zulily's top 10 tips for self-service analytics with Google BigQuery and Tableau](#)

Service vs User accounts

Tableau can connect using the two types of BigQuery authentication accounts: User and Service. A user account identifies a person to the system. User credentials ensure your application has access to BigQuery tables available to the end user. A user credential can run queries against only the end user's Cloud project rather than the application's project, meaning the user is billed for queries (and not the application).

A service account is a Google Account that is associated with your Google Cloud project, instead of a user. Use a service account to access the BigQuery API to perform operations that are not related to a particular user, such as a batch processing pipeline.

Data source connections using Service accounts can only be created with Tableau Desktop. This means, to create a data source connection with a service account in Tableau Cloud, it must be configured first in Desktop, and then published. In Tableau Cloud, when accessing BigQuery with a service account, you may be prompted for the Tableau Cloud user to associate with the connection:



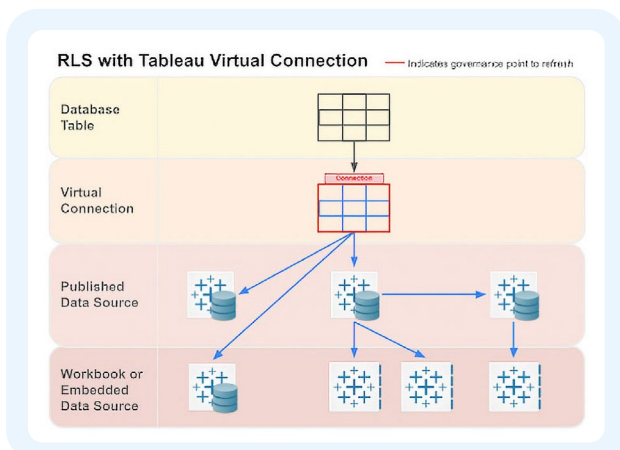
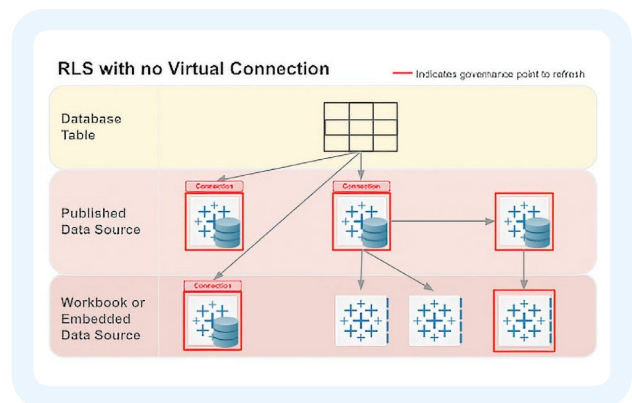
For more information about connecting with service accounts, see [Google BigQuery JDBC](#) and [Introduction to authentication](#).

Appendix A: Governance

Tableau and Google each have a definition of data governance. Tableau’s governance model emphasizes establishing trust and confidence in the data. Google’s is focused on policies, procedures, roles, and responsibilities throughout the lifecycle of data.

It is also possible to use Tableau’s data source filters, initial sql, and/or entitlement tables. However, these methods require more effort to establish and maintain RLS.

It is best practice to leverage the database platform for data level security when possible. Google BigQuery provides access control at the table, column, and row levels. You can still leverage Tableau for row level security (RLS). Using virtual connections is the recommended approach, as it allows centralized RLS.



For more information about governance, see [Overview of data security and governance](#) and [Tableau Governance](#).



Appendix B: Using Tableau to visualize Google BigQuery ML model results

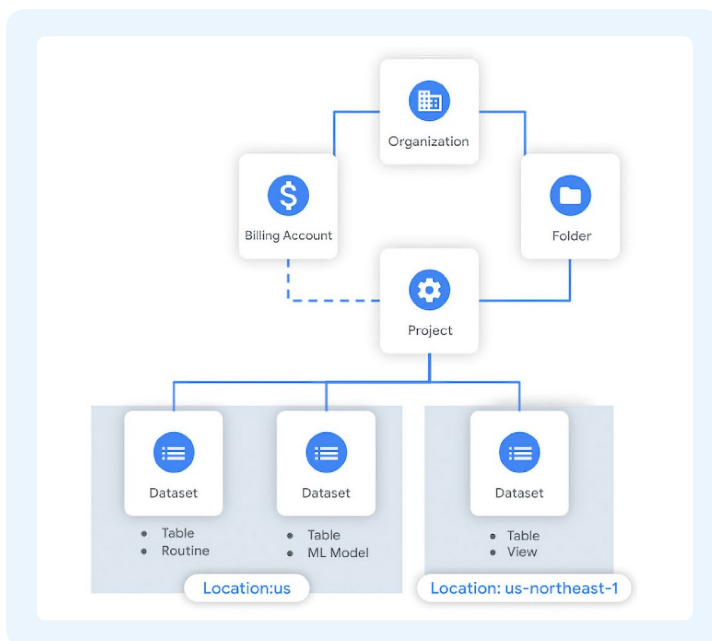
BigQuery ML allows users to use embedded machine learning technology to train models based on data stored in BigQuery. However, just like working with any other data, directly querying a database isn't always the ideal method to explore the output of your model. Tableau allows you to easily manipulate the results of your predictive models in a way that facilitates an intuitive understanding of the data.

Conveniently, Tableau Cloud and Tableau Server users can share their models and results across the organization, allowing more people to gain insights.

To see an example, see [Leveraging Google BigQuery's machine learning capabilities for analysis in Tableau](#).

For more information on creating and executing machine learning models using standard SQL queries, see [What is BigQuery ML?](#)

Appendix C: BigQuery Resource Hierarchy



BigQuery inherits the [Google Cloud resource hierarchy](#) with an additional grouping mechanism called datasets. In BigQuery, datasets are defined as logical containers used to organize and control access to your BigQuery resources. BigQuery Datasets are similar to schemas in other database systems. Tables are contained within datasets.

To get to your Google BigQuery data in Tableau, connect to the BigQuery project, select a dataset, and then select a table.

For more information about BigQuery resources hierarchy, see [Organizing BigQuery Resources](#).



Appendix D: Additional resources

[Tableau free trial](#)

[Designing efficient workbooks \(whitepaper\)](#)

[Tableau Help: Google BigQuery \(JDBC\)](#)

[Tableau Help: Google BigQuery](#)

[Tableau's Google Solutions page](#)

[Google Cloud Monitoring Dashboards](#)

[Tableau Server on Google Cloud \(whitepaper\)](#)

[Cloud Based Analytics - Secure, Agile, and Accessible BI \(whitepaper\)](#)

[Google Pricing calculator](#)

About Salesforce Industries

Salesforce Industries, delivers industry-specific cloud and mobile software that embed digital, omnichannel processes for customer-centric industries. Salesforce Industries enables companies to achieve faster business agility and time to value from the cloud across digital and traditional channels.

[Learn more at www.salesforce.com/energy](http://www.salesforce.com/energy)

Contact Us

Contact us to learn more about Salesforce Industries and Energy & Utilities Cloud.

+1 800-387-3285 | Salesforce.com



Thank You

