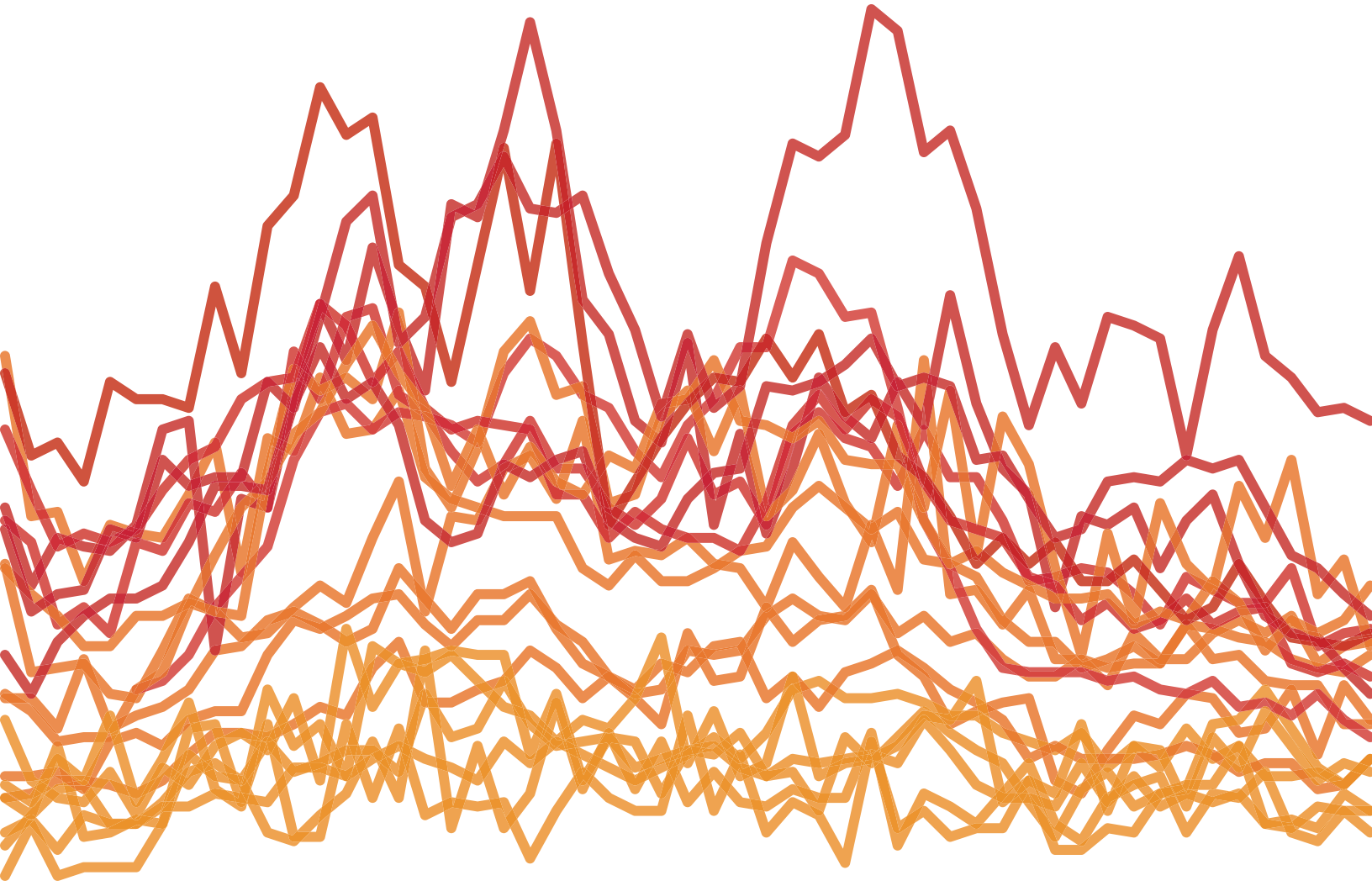


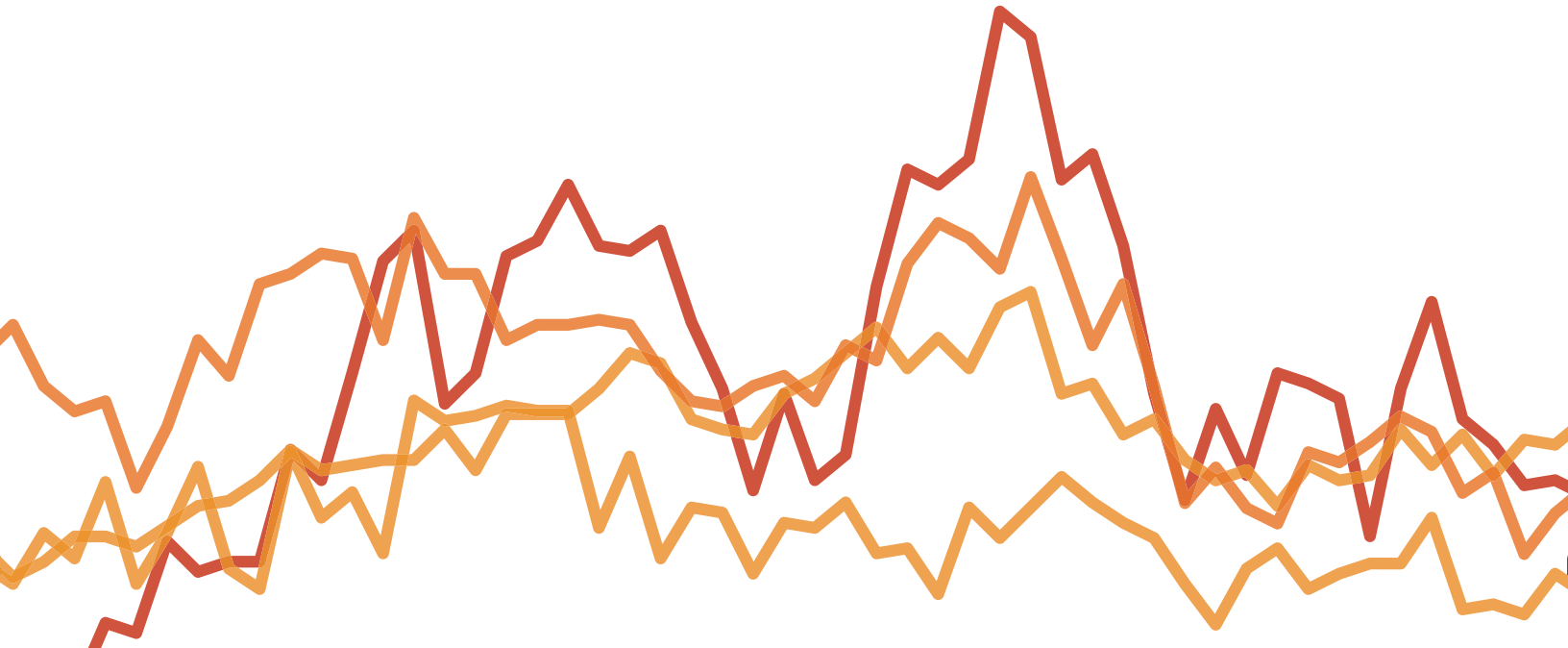
Jeff Feng, Product Manager, Tableau
Marc Lobree, Product Consultant, Tableau
Tino Tereshko, Enterprise Solutions Engineer, Google
Mike Graboski, Solutions Engineer, Google

# Google BigQuery & Tableau: Best Practices

Tableau and Google BigQuery allows people to analyze massive amounts of data and get answers fast using an easy-to-use, visual interface. Using the 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 stunning 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 and Tableau Online to allow anyone to connect from any device.
- Combine the cloud agility of Google BigQuery with the blazing speed of Tableau to recognize project value faster.

Optimizing the two technologies together will yield significant performance gains, shorten design cycles, and help users and organizations become more successful. In this paper, we will discuss techniques to optimize data modeling and query formation to maximize the responsiveness of visualizations. We will also discuss techniques to get the best cost efficiency when using Tableau and BigQuery together.



# Table of Contents

## 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 massive 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 massive, structured, and semi-structured data sets. 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 No-Ops and maintenance-free, and is integrated with the Google Cloud Platform. Unlike other cloud-based analytics solutions, BigQuery does not require you to provision a cluster of servers in advance—processing clusters are sized and provisioned by BigQuery at runtime. As your data size increases, BigQuery will automatically add processing power—yet you pay the same price per gigabyte.

**Tableau**

Tableau helps people to see and understand data. Our software products put the power of data into the hands of everyday people. This allows a broad population of users to engage with their data, ask questions, solve problems, and create value. Based on technology developed at Stanford University, our product reduces the complexity, inflexibility, and expense associated with traditional business intelligence applications. Anyone who is comfortable with Excel can leverage Tableau Desktop to create rich, interactive visualizations and powerful dashboards using a drag-and-drop user interface, as well as share them securely across organizations using Tableau Server or Tableau Online.

Tableau has a native, optimized connector to Google BigQuery that supports both live data connectivity and in-memory extracts. Tableau's data blending allows users to mash up data from BigQuery with data from any of our other 40+ supported data sources. For visualizations published to the cloud using Tableau Server or Tableau Online, direct connectivity to Google BigQuery can be maintained.

## Best Practices for Performance: Tableau

**Leverage Tableau Version 9.0**

One of the easiest ways to accelerate performance is to ensure you are using Tableau Version 9.0. Tableau 9.0 has a number of enhancements to ensure your visualizations are responsive. These enhancements include:

- Parallel Queries: Tableau will take advantage of the capability of Google BigQuery and other data sources to execute multiple queries at the same time, for a total of up to sixteen concurrent queries. 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 will take 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. For example, a workbook with a 157-million-row Tableau Data Extract file opens fifty times faster when cached in Tableau 9.0 than it does without a cache in Tableau 8.3.

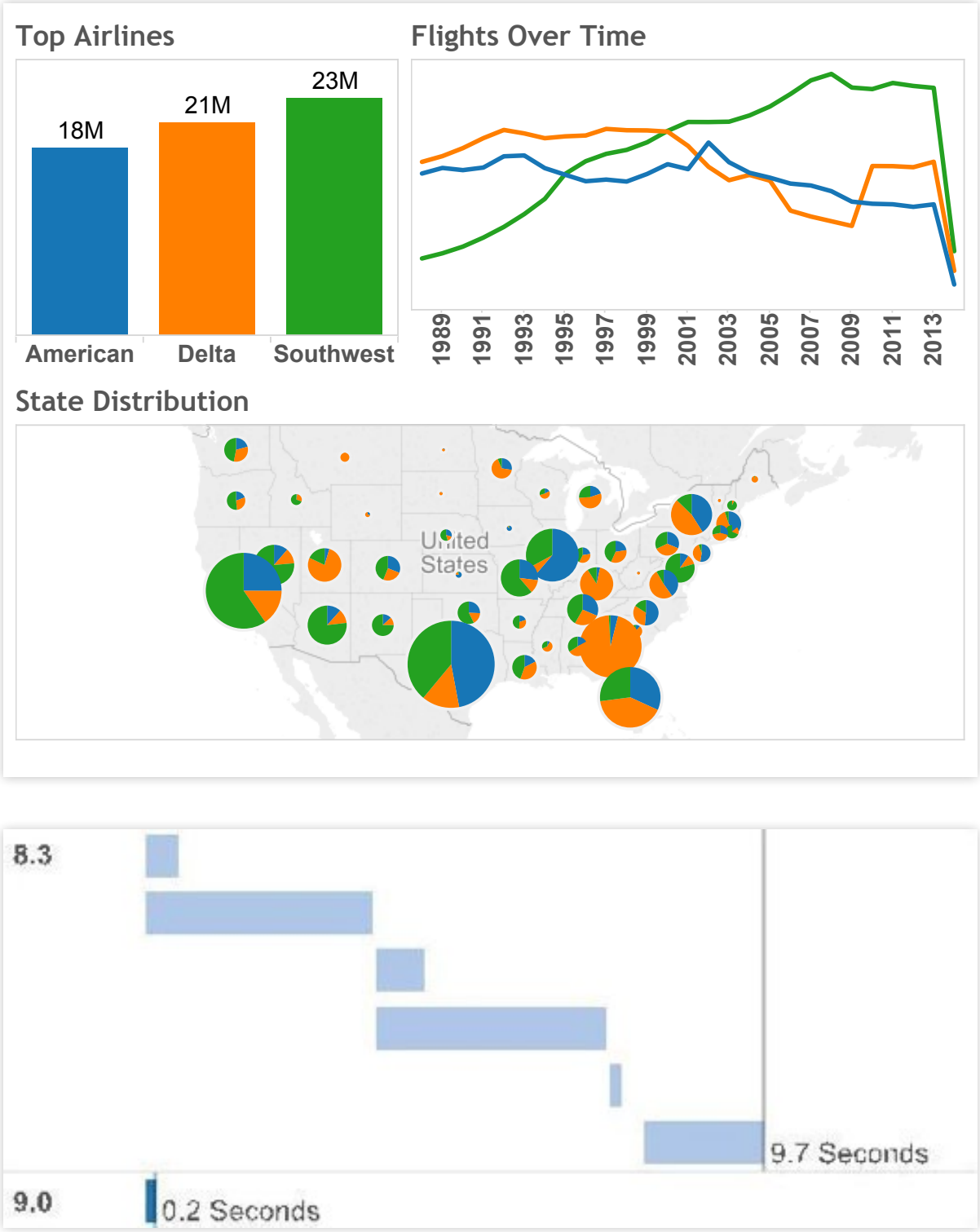



*Figure 1: Performance improvement for a 157-million-row data set opened in Tableau 9.0 versus Tableau 8.3*

**Performance Recorder**

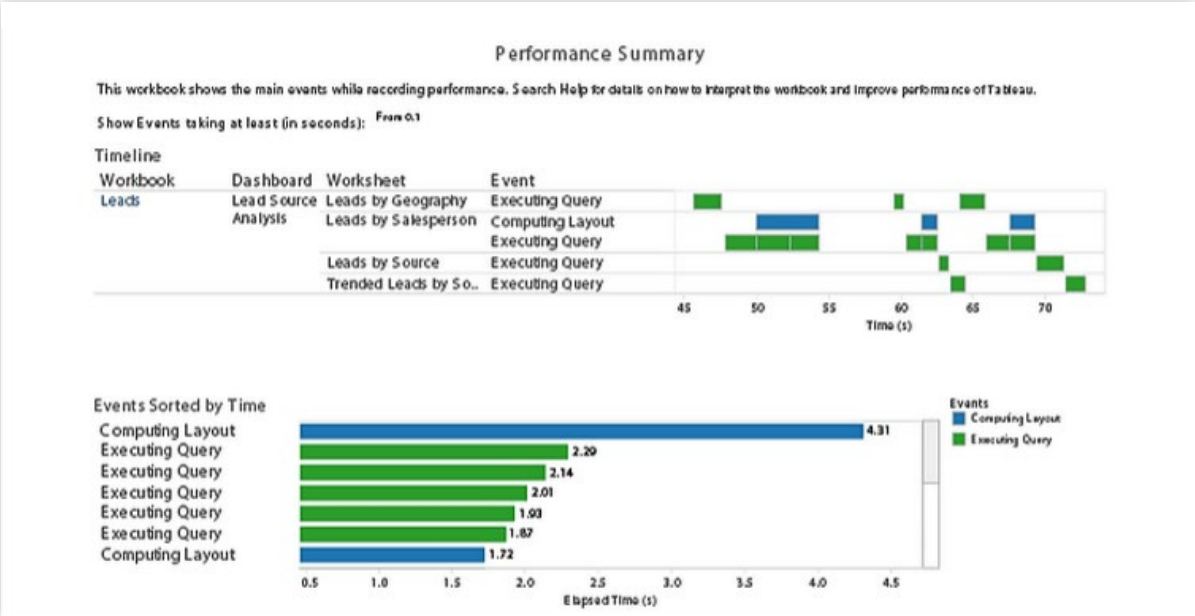
Performance Recorder is a powerful built-in tool that allows you to pinpoint slow queries and optimize your workbooks for maximum performance. It does this by tracking the elapsed time for an individual workbook to execute a query and compute the layout. Hovering over one of the green bars below will show the user the query that's being generated against BigQuery. After identifying a slow query, you can often resolve the performance issue by revisiting your data model.

For instructions on how to create or interpret a performance recording, please follow one of these links:

"We publish a dashboard that the client can then log into whenever they please. They can check in and ask: How are my sales doing? Which store is selling more? Which marketing avenue is working best for me?"

"It changes the conversation," Ghimire says. "It changes the dynamics and positions us strongly."

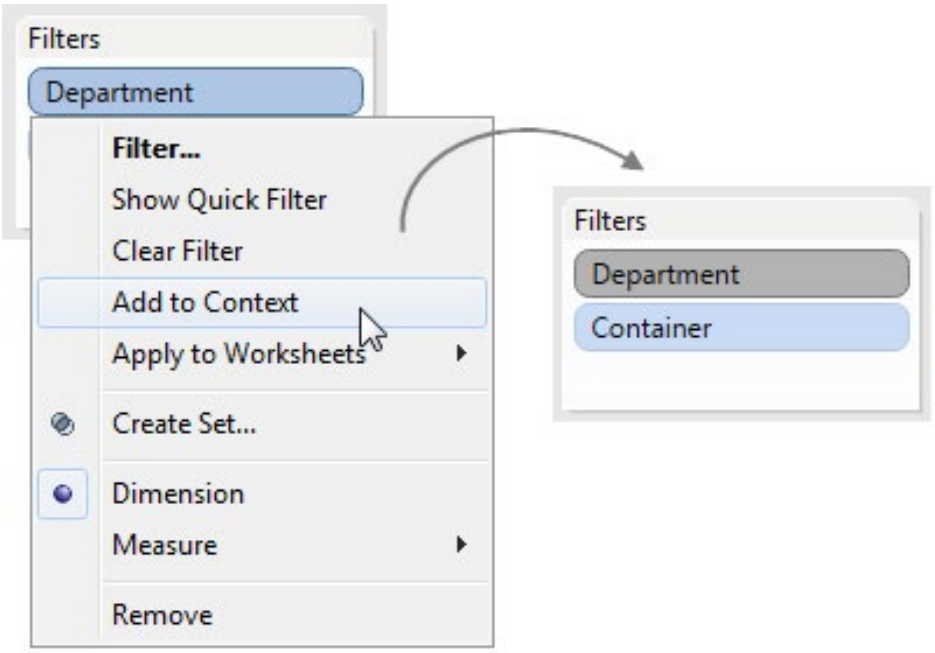
- Performance Recorder on Tableau Desktop
- Interpret a Performance Recording on Tableau Desktop
- Performance Recorder on Tableau Server
- Interpret a Performance Recording on Tableau Server



### Context Filters

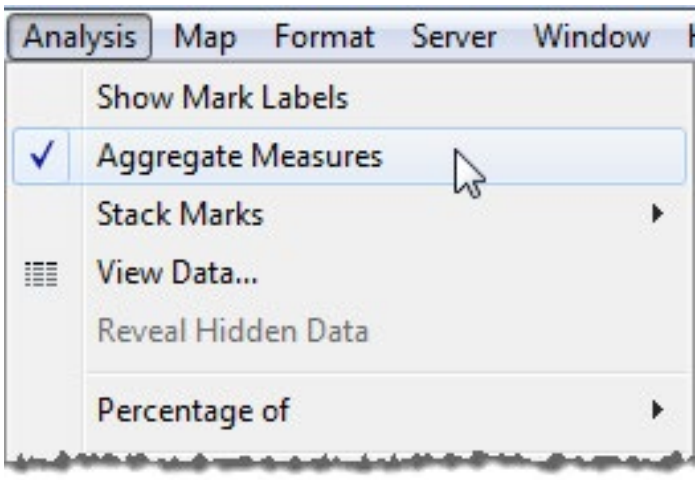
If you are applying filters to a large data source, you can improve performance by setting up context filters. A context filter is applied to the data source first, so that additional filters are applied only to the resulting records. This sequence avoids applying each filter to each record in the data source.

If you are setting filters that significantly reduce the size of the data set, and will use those filters for more than several data views, then you should set those filters as context filters. Refer to Creating Context Filters to learn how to create context filters. For more information about performance improvement with context filters, refer to Speeding up Context Filters. We also have Examples of Context Filters.



**Aggregate Measures**

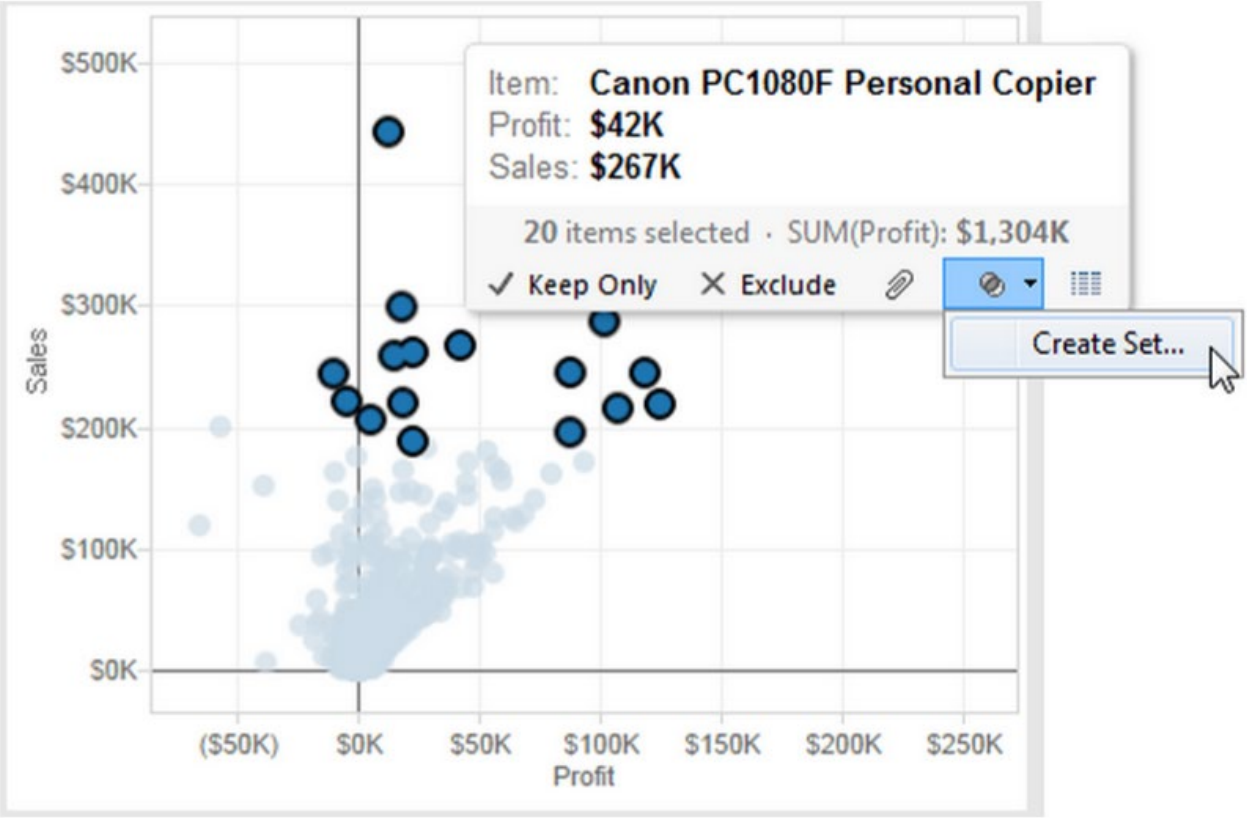
If the views you create are slow, make sure you are working with aggregated measures rather than disaggregated measures. When views are slow, it usually means you are trying to view many rows of data at once. You can reduce the number of rows by aggregating the data. In other words, make sure the Aggregate Measures option on the Analysis menu is selected.  For more information, refer to Aggregating Data.



### Sets

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 fifty items in a dimension, rather than all of the items in a dimension. For more information, refer to Creating Sets.

When creating a group from a selection as described in Creating Groups, make sure you've included only the columns of interest. Each additional column included in the set will result in decreased performance.



### Add Filters First

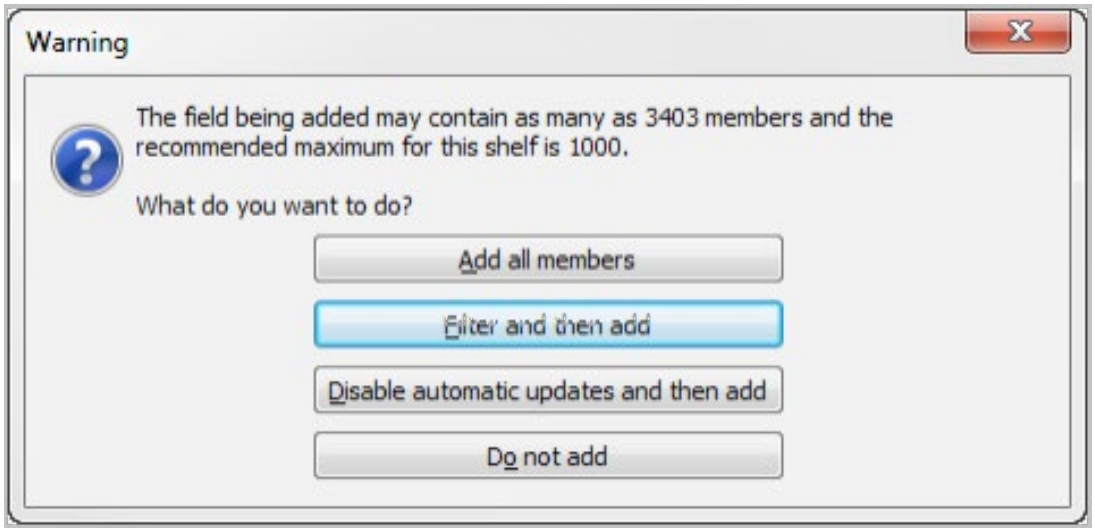
If you are working with a large data source and have automatic updates turned off, it is possible to create a really slow query when adding filters to the view. Rather than build the view and then specify filters, you should first specify the filters and then drag fields to the view. That way, when you run the update or turn on automatic updates, the filters will be evaluated first.

### 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 dense data view, 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. Refer to Managing Queries for more information.

### Look for Warnings

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



## Best Practices for Cost and Performance: Google BigQuery

**Denormalize and Pre-JOIN**

BigQuery supports very large JOINs, and JOIN performance is very good. That being said, BigQuery is a Columnar Datastore, and maximum performance is achieved on denormalized data sets.

Because BigQuery storage is very inexpensive and scalable, it is often prudent to denormalize and pre-JOIN datasets into homogeneous tables. In essence, you are exchanging compute resources for storage resources (the latter being more performance- and cost-effective).

BigQuery is an excellent ETL tool, allowing you to execute massive transforms and pipelines quickly and efficiently. Be sure to enable “Allow Large Results” when materializing data sets larger than 128MB.

More information:

cloud.google.com/bigquery/preparing-data-for-bigquery#denormalizingdata

cloud.google.com/bigquery/querying-data#largequeryresults

**Shard Tables by Date**

Some data naturally lends itself to being partitioned by date: for example, log data, or any data for which the records include a monotonically increasing timestamp. In this case, shard your BigQuery tables by date, and include the date in the table name.  For example, name your tables something like: mytable_20150501, mytable_20150502, etc.

Then, when you want to run a query that filters by date, use BigQuery’s TABLE_DATE_RANGE function:

```
SELECT
  name
FROM
  (TABLE_DATE_RANGE([mydata.people],
                    TIMESTAMP('2014-03-25'),
                    TIMESTAMP('2014-03-27')))
WHERE
  age >= 35
```

The example above will automatically include all tables between 2014-03-25 and 2014-03-27.
To use TABLE_DATE_RANGE, your tables must be named according to the pattern:

```
[arbitrary prefix]YYYYMMDD.
```

Other database systems rely on sharding to improve performance. Sharding by date actually has a negligible performance difference on BigQuery, but the main driver here is cost. Because you're processing less data, you're paying less money per query.

More information: cloud.google.com/bigquery/query-reference#tablewildcardfunctions

**Specify a Destination Table If Running Many Similar Queries**

While query caching is useful if you're running many identical queries, it won't help if you're running similar, but slightly different, queries (e.g., changing only the values in a WHERE clause between query runs). In this case, run a query on your source table and write the records you will repeatedly query to a new destination table. Then, run queries against the new destination table that you created.

For example, let's say you plan to run three queries with three different WHERE clauses:

- WHERE col1 = "a"
- WHERE col1 = "b"
- WHERE col1 = "c"

Run a query against your source table, and write the output records into a destination table:

SELECT col1

FROM source

WHERE col1 = "a" OR col1 = "b" OR col1 = "c"

By "OR"ing the WHERE clauses together, we capture all relevant records. Our new destination table is potentially much smaller than the original source table. Since BigQuery charges based on the amount of data processed in a query, running subsequent queries against the new destination table will save money instead of running the queries directly against the source table.

**Use Table Decorators to Sample the Data**

BigQuery is a No-Ops solution, requiring minimal database administration. Gone are the days of setting up primary keys, indexes, and secondary keys, and worrying about an ad-hoc query not using the indexes. BigQuery is truly an ad-hoc SQL execution engine. BigQuery accomplishes this by doing full table scans every time. In traditional technologies, full table scans are to be avoided, but BigQuery flips this notion on its head, performing them exceptionally well.

However, if you sample data from the table, every row will be traversed. Even if you run “SELECT * FROM table LIMIT 1;”, the entire table will be traversed, and you will pay for reading the entire table.

If you need to sample the data in your table, consider using Table Decorators. Table Decorators allow you to query a small partition in your table by defining time ranges. Here is an example of querying data added between an hour and half an hour ago:

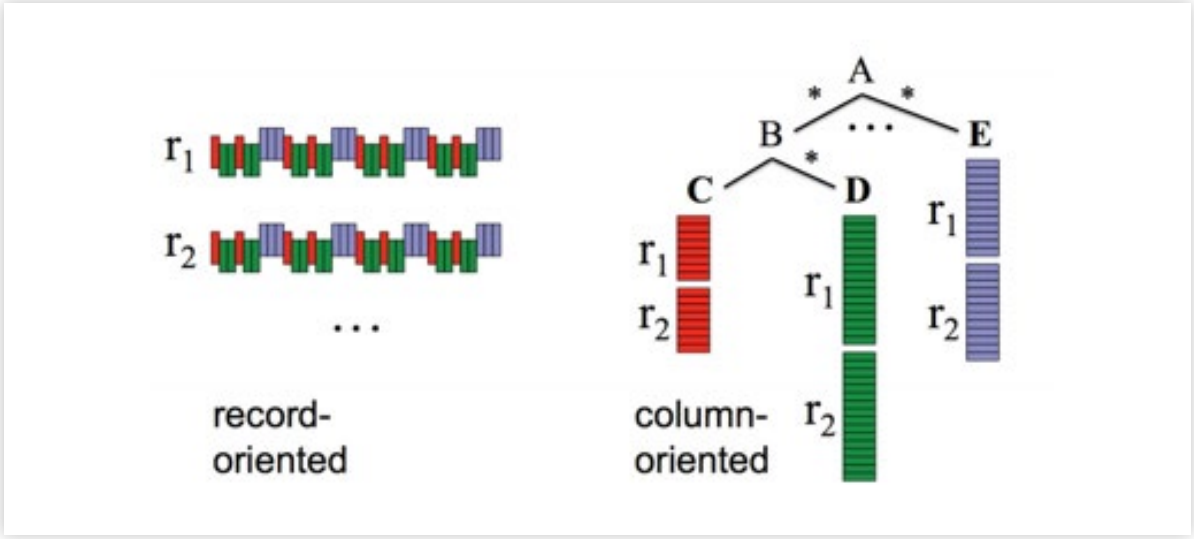
```
SELECT
  name
FROM
   [mydata.people]@-3600000--1800000
WHERE
  age >= 35
```

More information: cloud.google.com/bigquery/table-decorators

**Explicitly List Columns in SELECT**

BigQuery is a columnar datastore. Under the hood, each column in a table is stored as a separate file. Explicitly defining which columns to query will tell BigQuery to only look at those columns, entirely bypassing other columns. This results in faster processing time and cost savings.

For example, this query will only process data in the “name” column, even if the table contains thousands of columns:



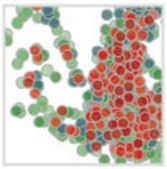

```
SELECT name FROM [mydata.people]
```

## Conclusion

By applying best practices, business users and data analysts alike will be able to maximize the performance and responsiveness of Tableau visualizations built against Google BigQuery. When these technologies are combined, users can truly visualize billions of rows of data at the speed of thought.

## About Tableau

Tableau helps people see and understand data. Tableau helps anyone quickly analyze, visualize and share information. More than 29,000 customer accounts get rapid results with Tableau in the office and on-the-go. And tens of thousands of people use Tableau Public to share data in their blogs and websites. See how Tableau can help you by downloading the free trial at tableau.com/trial.



## Additional Resources

Download Free Trial

## Related Whitepapers

Why Business Analytics in the Cloud?

5 Best Practices for Creating Effective Campaign Dashboards

See All Whitepapers

## Explore Other Resources

· Product Demo

· Training & Tutorials

· Community & Support

· Customer Stories

· Solutions