interworks

Tableau Dashboard Optimization Checklist

Tableau makes it easy to build beautiful dashboards that inform and engage your audience. Just as important, however is that these dashboards perform well. While there are many ways to build dashboards in Tableau, some are more efficient than others. What do me mean when we talk about optimizing a dashboard's performance?

From a user experience standpoint, we want to optimize the total time it takes a user to gain the information they need. Load times are the most obvious pain point, but we should also consider time spent finding key insights and processing complex visuals.

From a developer experience standpoint, we want to optimize the time and effort required from developers over the lifespan of the dashboard. Difficulties in maintaining a complex dashboard that breaks more often make it worth the initial investment to productionalize.

Put simply, optimizing your dashboards in little ways can add up to major improvements in user adoption and efficiency. Leaning on over a decade of deep Tableau expertise, we've pinpointed exactly how and where to achieve those performance boosts in this checklist. Happy building!

Data Structure

Live vs. Extract

- ☐ Can this data source be extracted?
- ☐ If so, what refresh schedule should be implemented?
- ☐ If using an incremental refresh, be sure to still schedule a full refresh periodically.

Hosted vs. Embedded

- ☐ Could this data source be used in other reports?

Data Source Filters

- ☐ These are typically filtered by date range or specific dimensions, like region.
- ☐ When extracting, use extract filters to reduce the size of the extract instead of data source filters. Data source filters do not restrict what is stored in the extract, just what is displayed.

Trim Columns

- ☐ Hiding unused columns helps organize the workspace with a live connection.
- ☐ Hiding unused columns when extracts are used can reduce the extract size and improve performance.

Review Data Structure

- ☐ Data granularity should drive the data model.
  - ☐ Find a reasonable balance.
  - ☐ Not all data sources should be a single table.
  - ☐ Linking 20 tables together can also cause problems.
- ☐ Use the relationship model to break out repeated values into a separate table.
- ☐ Avoid blends as much as possible and use relationships instead.
- ☐ Table order impacts query structure (most relevant with row-level security).

Calculations

Move Row-Level Calculations to Data Source

- ☐ In particular, those that are recoding strings (e.g., IF “A” THEN “B”)

Consolidate and Remove Duplicate Logic

- ☐ Centralize calculations when possible.
- ☐ Logic that is often repeated should be placed in a separate calculation.

Aggregate Calculations (These run across multiple rows of data.)

- ☐ MIN/MAX is faster than AVG/ATTR.
- ☐ COUNTD is slower than COUNT or SUM.
- ☐ Complex aggregations can be achieved with LODs, but use only as needed.

Date Logic

- ☐ Avoid hard-coding date references.
- ☐ Use an anchor date instead.
- ☐ Centralize date logic for better performance and easier updates.

Logical Operators

- ☐ CASE is more limited, but more performant, than IF.
  Sets and groups can also be useful alternatives.
- ☐ For simple renaming, consider aliasing.
- ☐ IN can be used as an alternative to OR.

General Tips

- ☐ Avoid converting numbers to strings.
- ☐ Always calculate percentages locally.
- ☐ Reduce nested calculations.
- ☐ Build scratch sheets to validate complex calculations.
- ☐ Use comments for documentation and to disable portions of syntax during testing.

Visual Complexity

High Mark Counts

- ☐ Tableau renders each mark in a visualization. Each cell
  in a large cross creates additional work.

Use Action Filters to Drill Down

- ☐ This will help avoid large initial loads, especially on detailed sheets.

Reduce Sheet Count

- ☐ Be intentional about what you are trying to answer with a dashboard.
- ☐ Limit use of multiple sheets to display a single value.
- ☐ Make use of titles and formatting as potential alternatives.

Limit and Optimize Filters

- ☐ Adding filters can increase load times.
- ☐ Filters with high cardinality require more effort to load the filter drop-down menu.
- ☐ Limit use of Relevant values setting to only when needed.
- ☐ Multiple Values (Custom List) can be used in place of Multi Select.
  Consider action filters as an alternative to basic filters.

Sizing

- ☐ Fixed-size dashboards can be more performant than automatically sized.
  Consistent sizing improves Tableau's ability to cache views between users.
- ☐ Range can be a reasonable compromise if necessary.
- ☐ Caching is prevented by user-based functions and transient
  functions like USERNAME() and NOW().

Clean Up

- ☐ Remove unused data sources.
- ☐ Hide sheets and delete unused sheets.
- ☐ Review sheet and dashboard naming.
- ☐ For workbooks with large numbers of fields and calculations, put fields into folders.

Productionalize

- ☐ Don't stop with the first draft.
- ☐ Revisit dashboards after users have some experience with them.
- ☐ Think of dashboards as dynamic products that require maintenance.
- ☐ Have a plan on how updates should be submitted and
  the frequency of edits to be supported.