Explore a dashboard file's LookML code

Previously, you learned about dashboards as code, an approach to managing dashboards by defining them in code instead of visually. Dashboard as code can be really useful for tracking changes in your dashboards, testing features before making them live, and reusing dashboards more easily. In this reading, you'll learn more about dashboards as code, including the role of parameters and elements in creating dashboard file code. You'll also explore an example of a simple dashboard file, along with a breakdown of how the code connects to what is shown to end users.

Parameters and elements

Previously, you learned how to define dimensions and measures in LookML. Part of building dimensions and measures involves setting parameters that describe how end users can interact with them. Dashboard parameters are the same; they modify the overall settings for the final dashboard that end users interact with. Here are some common dashboard parameters you'll use when creating dashboards as code:

Parameter	**Use-case**
dashboard	Creates a dashboard
title	Establishes the way the dashboard name appears for end users
description	Adds descriptions that can be viewed in the Dashboard Details panel, or in a folder set to list view
enable_viz_full_screen	Defines whether dashboard tiles appear in full-screen or expanded views for end users
extends	Bases the current LookML dashboard on another dashboard
extension	Requires the dashboard to be extended by another dashboard
layout	Defines how the dashboard will organize elements
rows	Starts a section of LookML to define the elements in each row of a grid layout dashboard
elements	Defines the elements in each row of a grid layout dashboard
height	Defines the height of a row for a grid layout dashboard

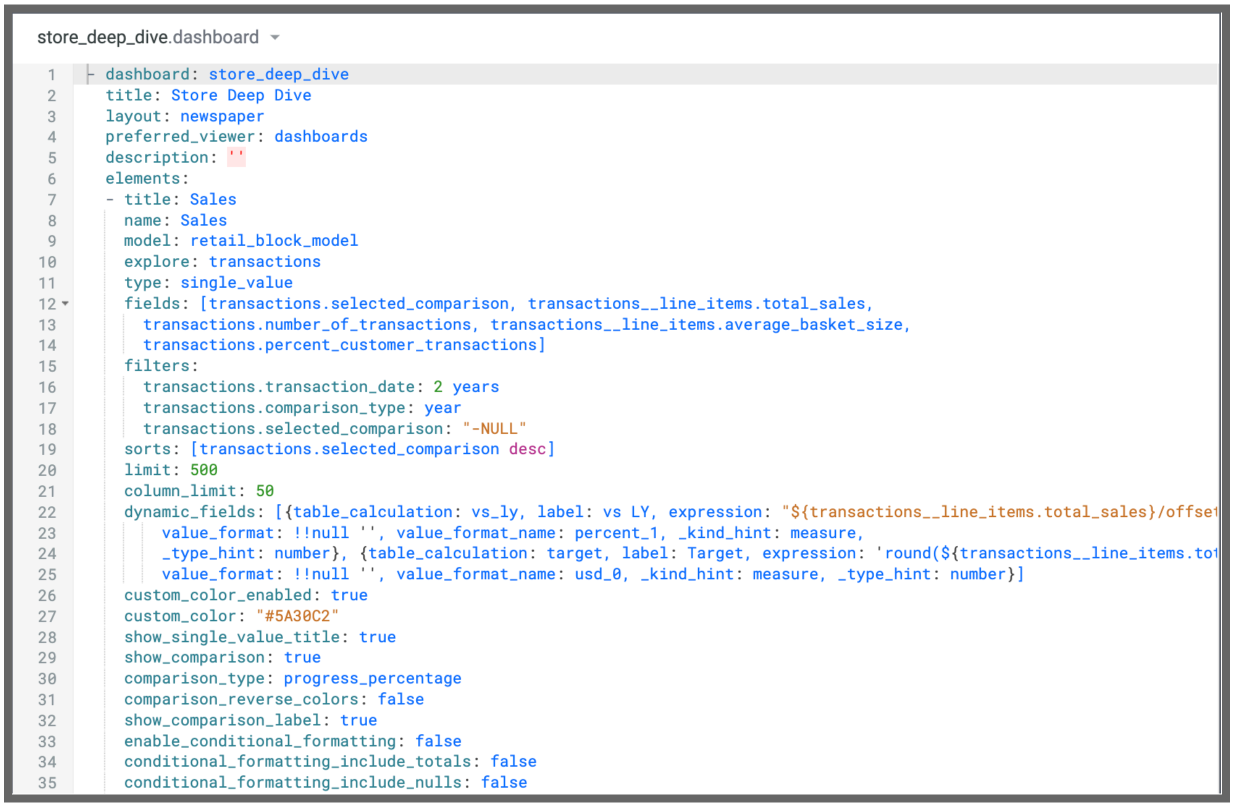
tile_size	Defines the size of a tile for a tile layout dashboard
width	Defines the width of a static dashboard layout
refresh	Sets the intervals for elements to automatically refresh
auto_run	Determines whether dashboards run automatically when opened or reloaded

For a complete list of dashboard parameters, check out the Google Cloud Looker Guide about LookML dashboard parameters.

Along with the parameters that determine and modify dashboard settings, dashboard files also include elements. **Elements** are data visualizations, text tiles, and buttons on a dashboard. These are defined in the dashboard file with element parameters. You can explore all of these parameters in the Google Cloud Looker Guide about LookML element parameters.

Breakdown a dashboard file

One of the best ways to understand dashboards as code is to connect the parameters in a dashboard file to what they represent in the end user's view. Here's a snippet of a dashboard file:



store_deep_dive.dashboard

```
- dashboard: store_deep_dive
  title: Store Deep Dive
  layout: newspaper
  preferred_viewer: dashboards
  description: ''
  elements:
  - title: Sales
    name: Sales
    model: retail_block_model
    explore: transactions
    type: single_value
    fields: [transactions.selected_comparison, transactions__line_items.total_sales,
      transactions.number_of_transactions, transactions__line_items.average_basket_size,
      transactions.percent_customer_transactions]
    filters:
      transactions.transaction_date: 2 years
      transactions.comparison_type: year
      transactions.selected_comparison: "-NULL"
    sorts: [transactions.selected_comparison desc]
    limit: 500
    column_limit: 50
    dynamic_fields: [{table_calculation: vs_ly, label: vs LY, expression: "${transactions__line_items.total_sales}/offset
        value_format: !!null '', value_format_name: percent_1, _kind_hint: measure,
        _type_hint: number}, {table_calculation: target, label: Target, expression: 'round(${transactions__line_items.tot
        value_format: !!null '', value_format_name: usd_0, _kind_hint: measure, _type_hint: number}]
    custom_color_enabled: true
    custom_color: "#5A30C2"
    show_single_value_title: true
    show_comparison: true
    comparison_type: progress_percentage
    comparison_reverse_colors: false
    show_comparison_label: true
    enable_conditional_formatting: false
    conditional_formatting_include_totals: false
    conditional_formatting_include_nulls: false
```

Here are some of the parameters in this code:

- dashboard: store_deep_dive: This parameter establishes that this file will generate a dashboard.
- title: Store Deep Dive: The title parameter sets the final title that will appear for end users viewing the dashboard.
- layout: newspaper: The layout parameter defines how the dashboard will be organized.
- elements: The element parameters define what visualizations will be displayed in the dashboard. Each element will have its own parameters. For example, the explore parameter will indicate which Explore fields are being pulled from to create this element. The type parameter indicates what type of visualization this will be.

You can begin to understand what the final dashboard will appear as for end users by exploring the parameters line by line. When you read this code, it tells you that this dashboard will focus on a retail store. It will be organized in a newspaper layout with visualization elements exploring transactions from the last two years. The code contained in this dashboard file communicates to Looker exactly what the dashboard should be like for users in an easy to edit format. That's the power of thinking about dashboards as code. Everything you need to know about this dashboard is right there.

Key takeaways

One of the benefits of using LookML is that it can help you approach dashboards as code. LookML can also help you maintain dashboard versions, share and review with team members to catch errors, and ensure the best possible experience for the end user. Like other structures in LookML, dashboards and dashboard elements are defined with parameters that modify or establish how the dashboard functions. Understanding these parameters in practice can prepare you to write your own dashboard files later.